Python Machine Learning Academy

Modulo 3 - Diego Gragnaniello

Classification

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Classification

- Qualitative variables take values in an unordered set C, such as:
  - eye color $\in$ {brown, blue, green}
  - email $\in$ {spam, ham}.
- Given a feature vector **X** and a qualitative response Y taking values in the set C, the classification task is to build a function C(X) that takes as input the feature vector **X** and predicts its value for Y; i.e. C(X) $\in$ C.
- Often we are more interested in estimating the probabilities that X belongs to each category in C.
- For example, it is more valuable to have an estimate of the probability that an insurance claim is fraudulent, than a classification fraudulent or not (binary).

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Just like regression,
  - Classification cannot be blindly performed in high-dimensions because you will get zero training error but awful test error;
  - Properly estimating the test error is crucial;
  - There are a few tricks to extend classical classification approaches to high-dimensions, which we have already seen in the regression context!

Classification

- There are many approaches out there for performing classification.
- We will recall the k-nearest neighbors classifier and the logistic regression.
- We will discuss:
  - support vector machines
  - decision tree.

K-Nearest Neighbours

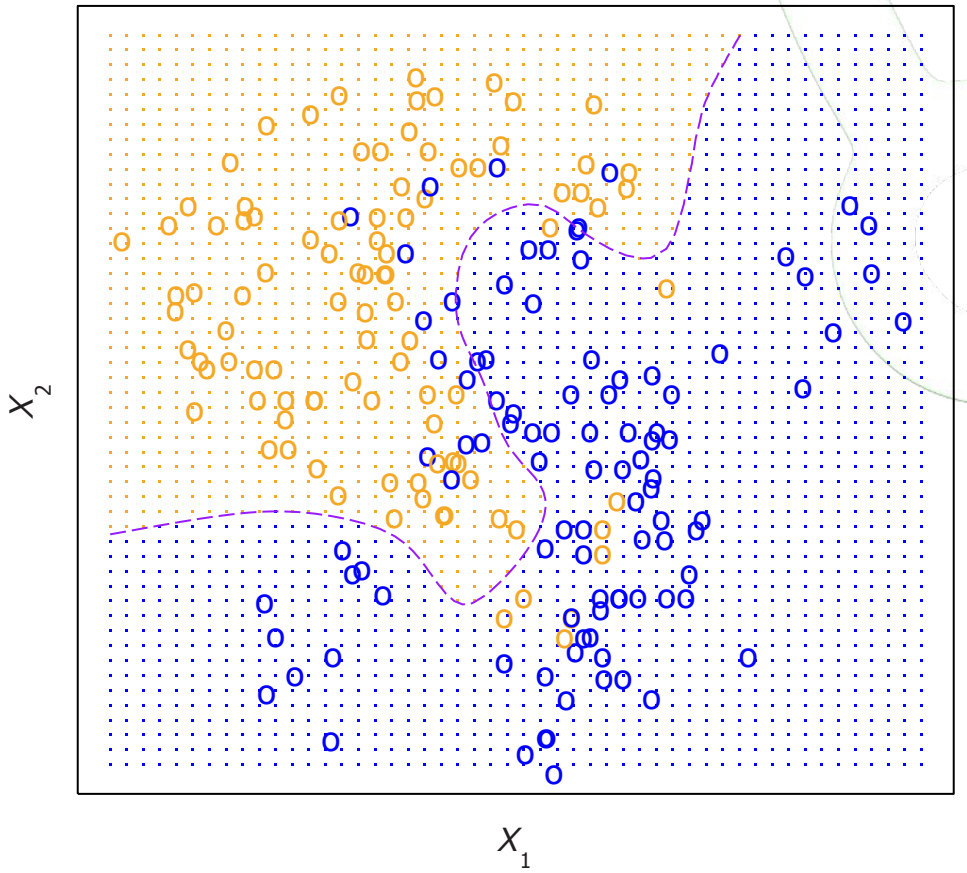
IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

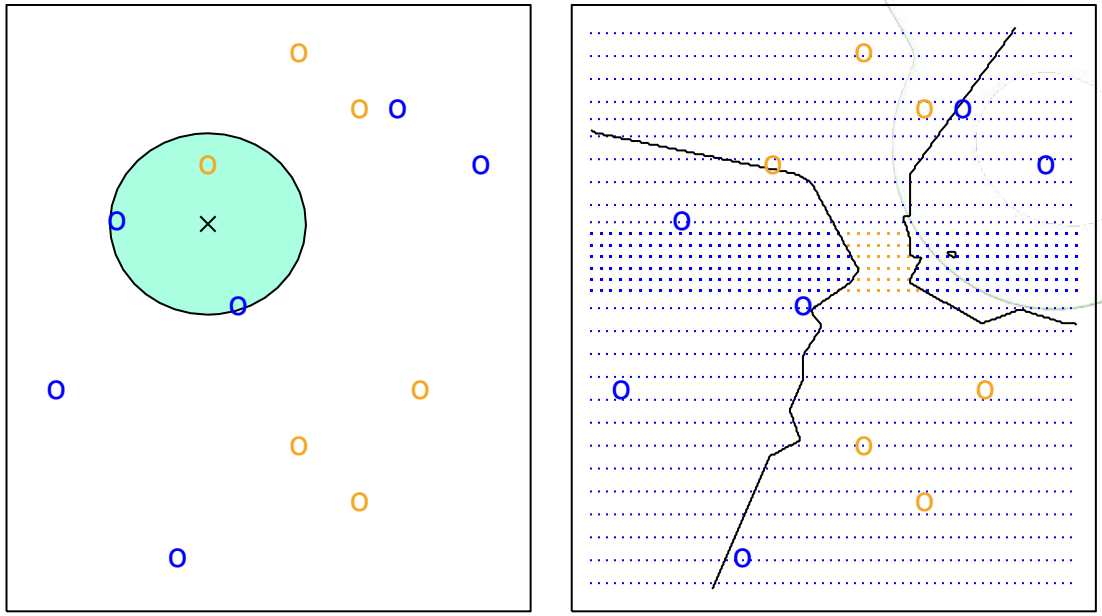
K -Nearest Neighbors

- Can I take a totally non-parametric (model-free) approach to classification?
- K-nearest neighbors:
  1. Identify the *K* observations whose *X* values are closest to the observation at which we want to make a prediction.
  2. Classify the observation of interest to the most frequent class label of those *K* nearest neighbors.

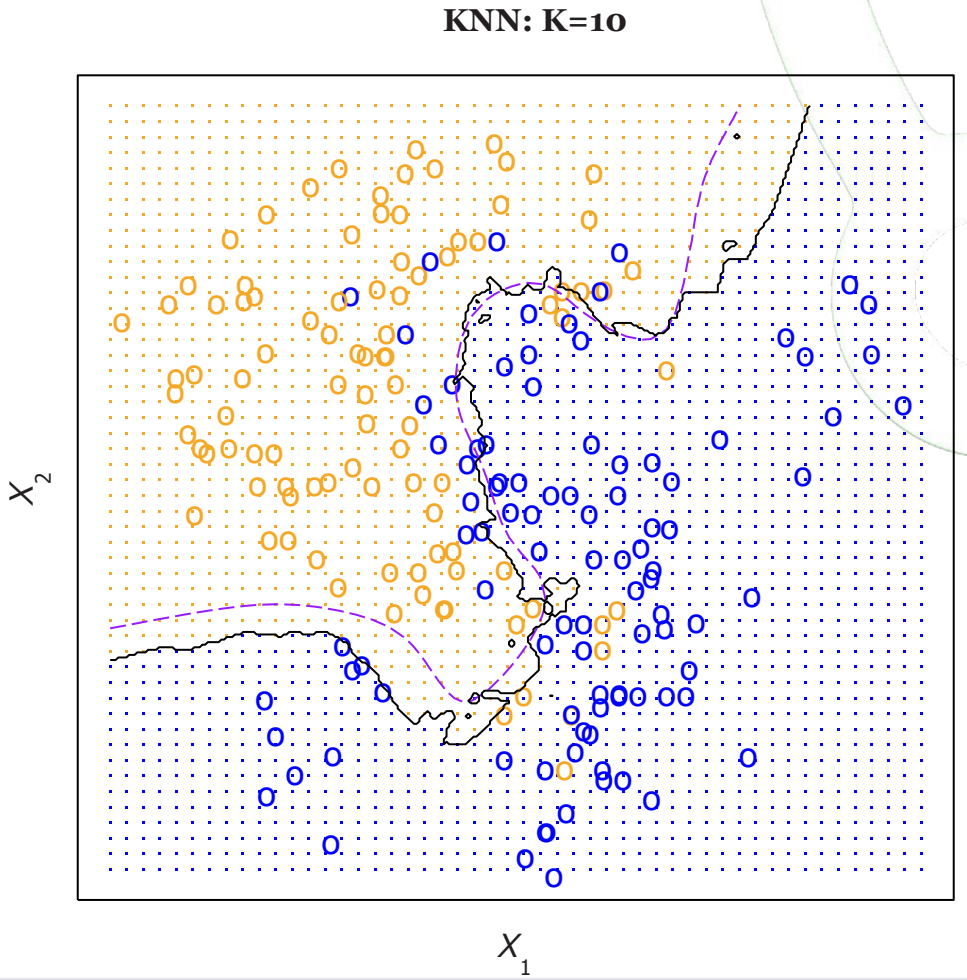
K -Nearest Neighbors



K -Nearest Neighbors

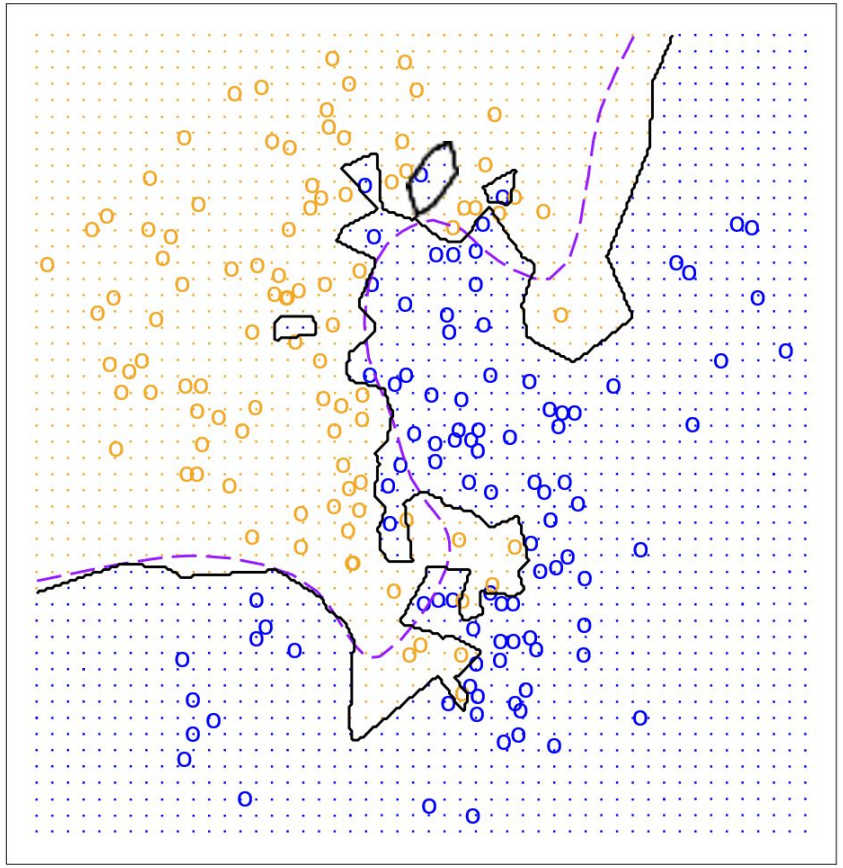


K -Nearest Neighbors

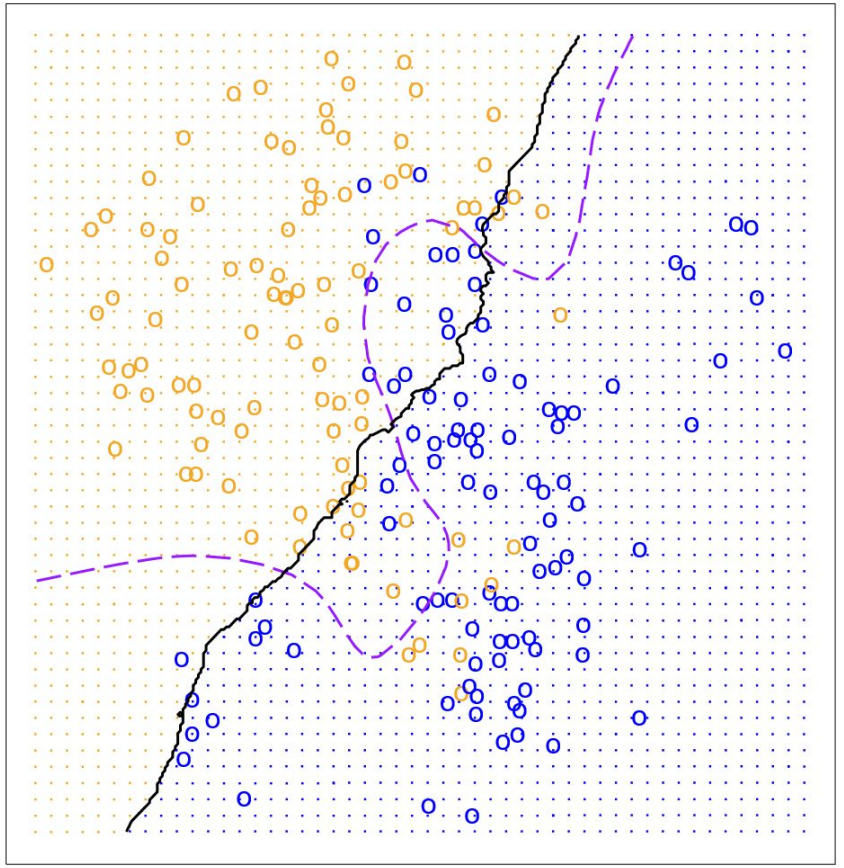


K -Nearest Neighbors

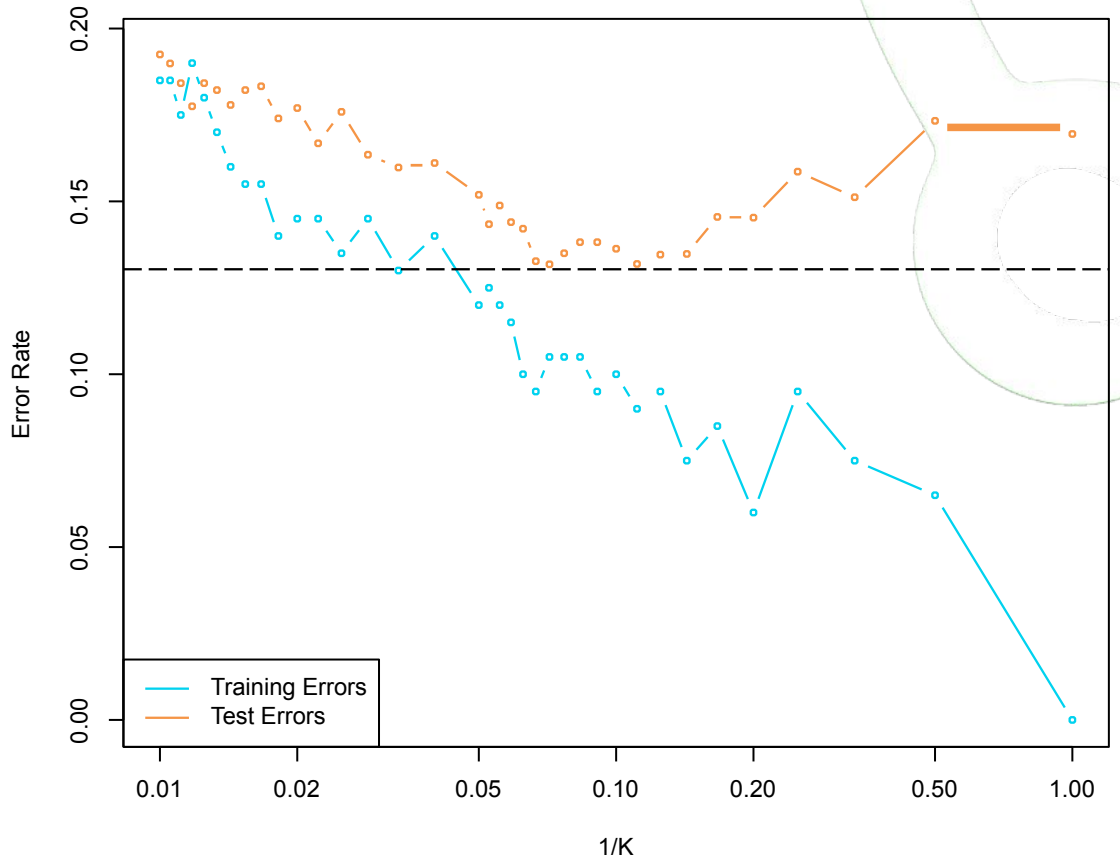
KNN: K = 1



KNN: K=100



K -Nearest Neighbors



K -Nearest Neighbors

- Simple, intuitive, model-free.
- Good option when *p* is very small.
- Curse of dimensionality: when *p* is large, no neighbors are "near". All observations are close to the boundary.
- Do not use in high dimensions!

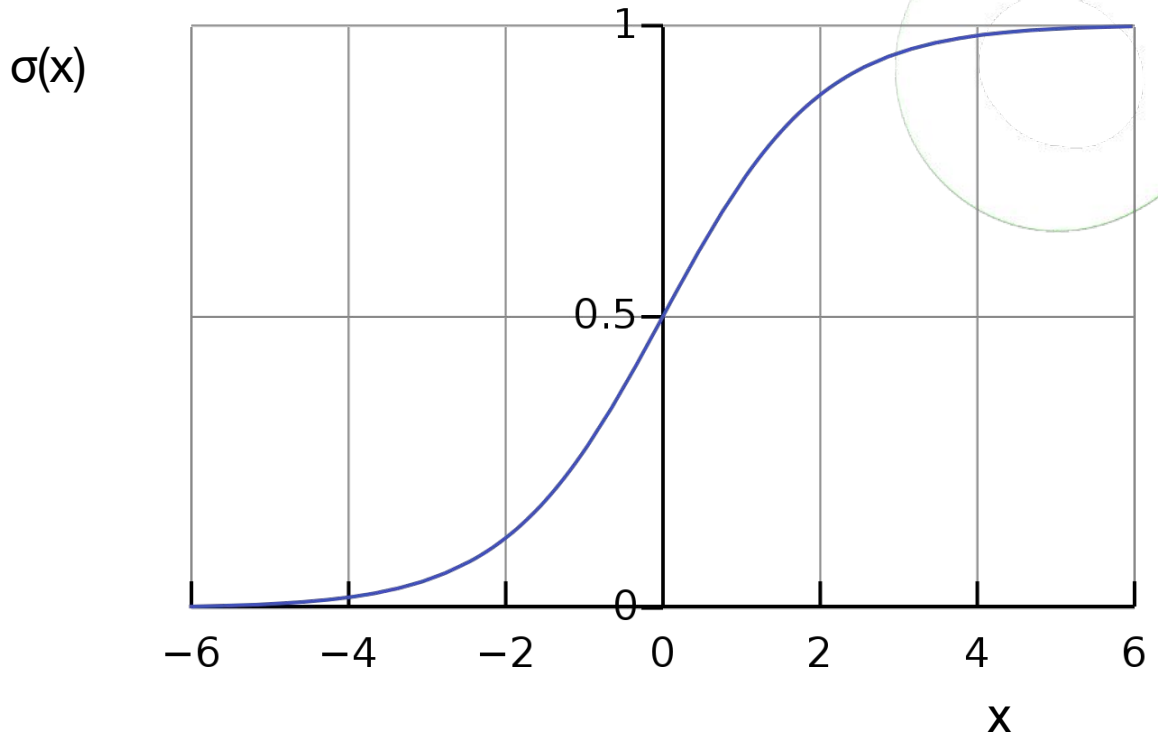
Logistic Regression

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

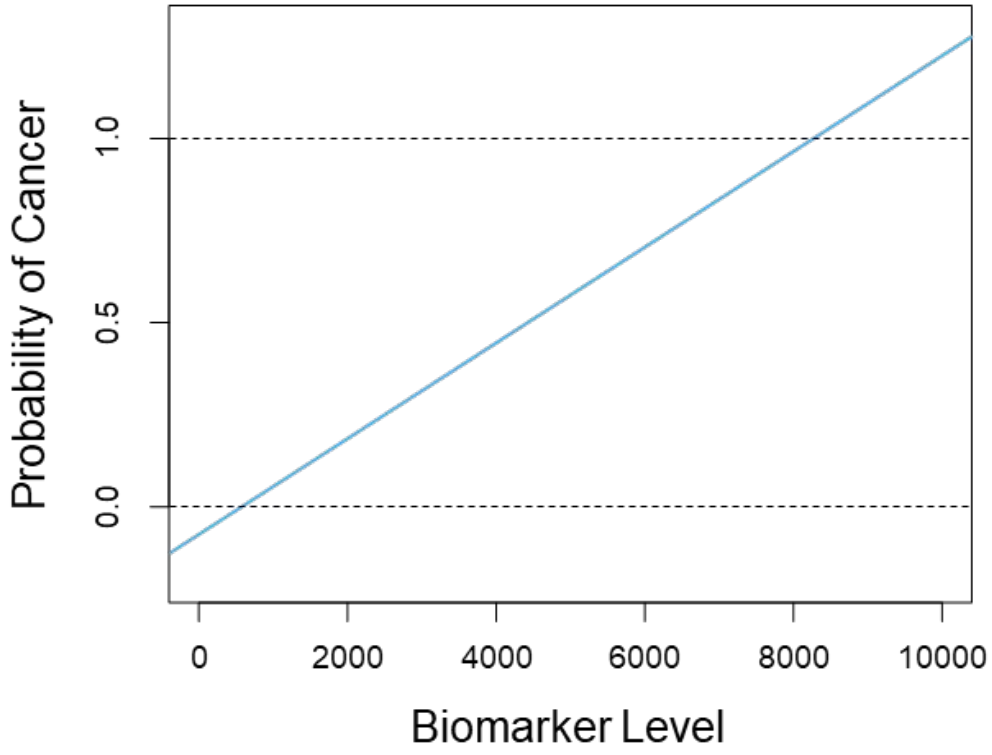
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

The logistic function

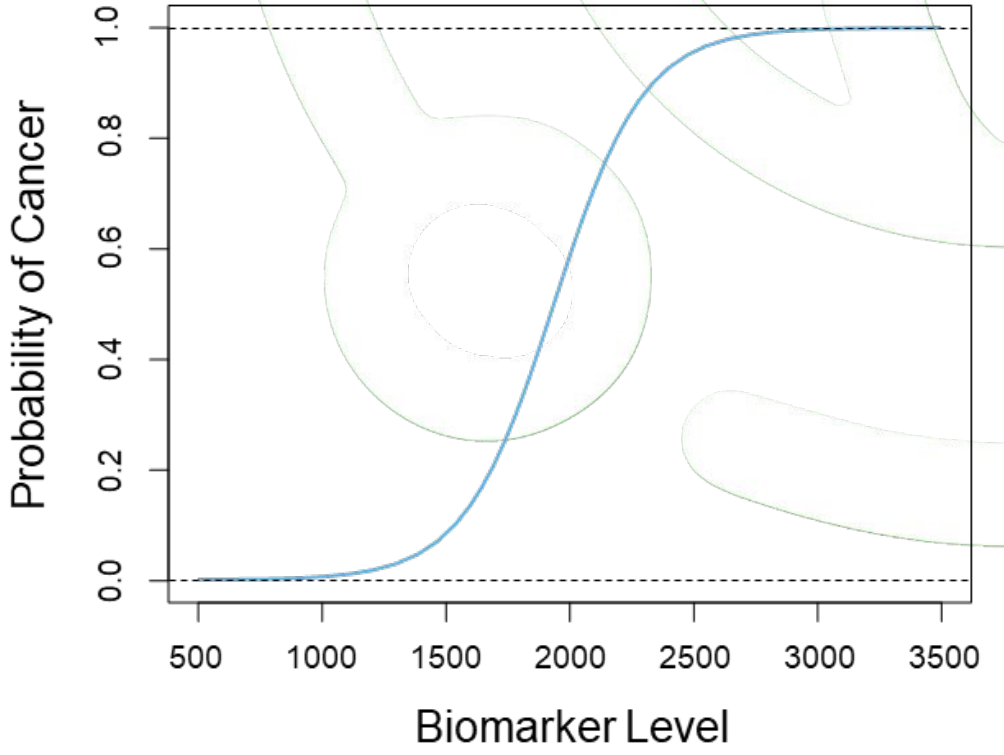
- Maps the range]-∞,+∞[to]0,1[
- Infinitely derivable
- This monotone transformation is called the **log odds** or **logit**



Why not linear regression



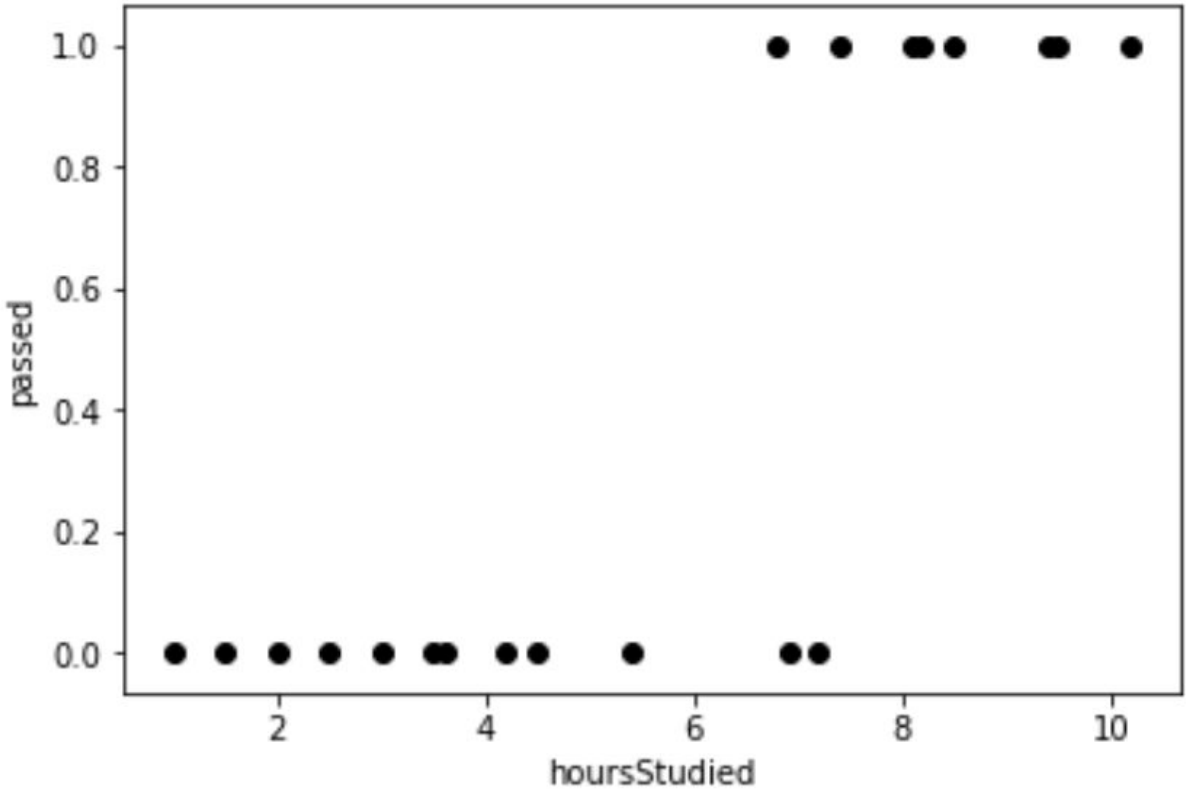
linear regression



logistic regression

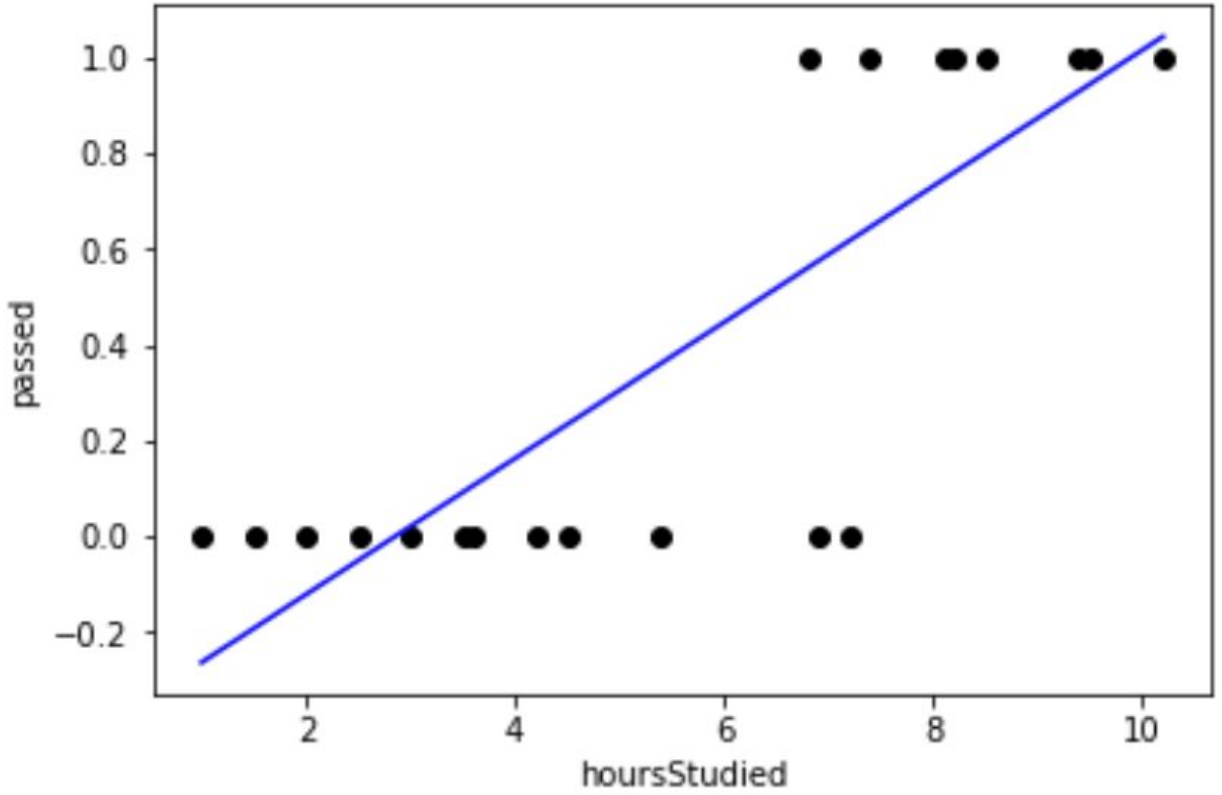
Example

- Suppose we want to predict the probability of a student passing an exam given the hours of study per week
- Training set:



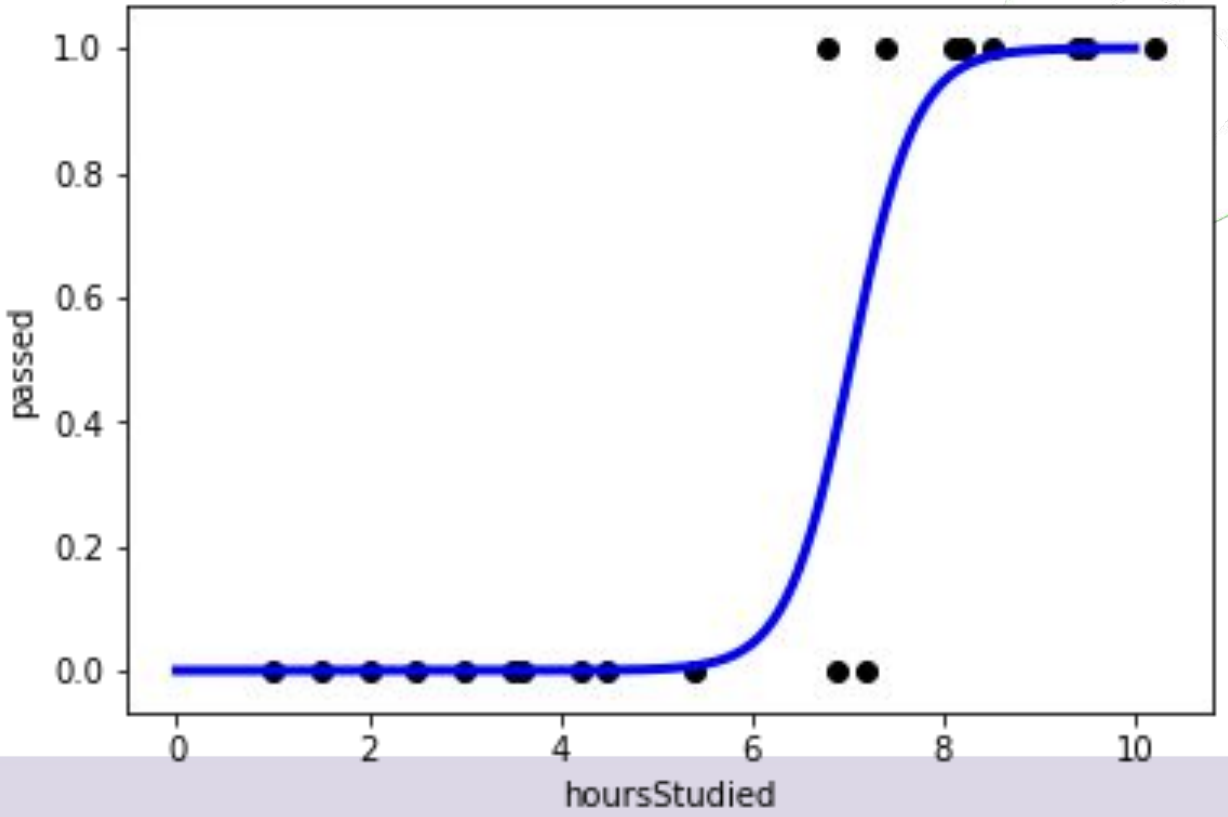
Example (2)

- Linear regression does not give a good prediction; most samples have a high squared error!



Example (3)

- With logistic regression, only border-line cases contribute significantly to the error



Probabilistic interpretation

- For a two class problem, logistic regression is equivalent to the assumption that the log of the ratio of the likelyhoods is a linear function of the features:

$$\log \frac{P(y=1 \mid x)}{P(y=0 \mid x)} = w^t \cdot x + b$$

- This assumption is satisfied for Gaussian distributions, but also for several other distributions (i.e. it is more general than a Gaussian Bayes classifier)

Logistic regression for High Dimensional data

- Ridge Logistic Regression

$$minimize_{\beta} \sum_{i=1}^{n} \ln\left(1 + e^{(-f_{\beta}(x_i)y_i)}\right) + \underbrace{\lambda\|\beta\|}_{\text{Ridge}}$$

- Lasso Logistic Regression

$$minimize_{\beta} \sum_{i=1}^{n} \ln\left(1 + e^{(-f_{\beta}(x_i)y_i)}\right) + \underbrace{\lambda|\beta|}_{\text{Lasso}}$$

Support Vector Machines

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Support Vector Machines

- Developed in around 1995.
- Touted as “overcoming the curse of dimensionality.”
- Does not automatically overcome the curse of dimensionality!!!
- Fundamentally and numerically very similar to logistic regression.
- But, it is a nice idea.

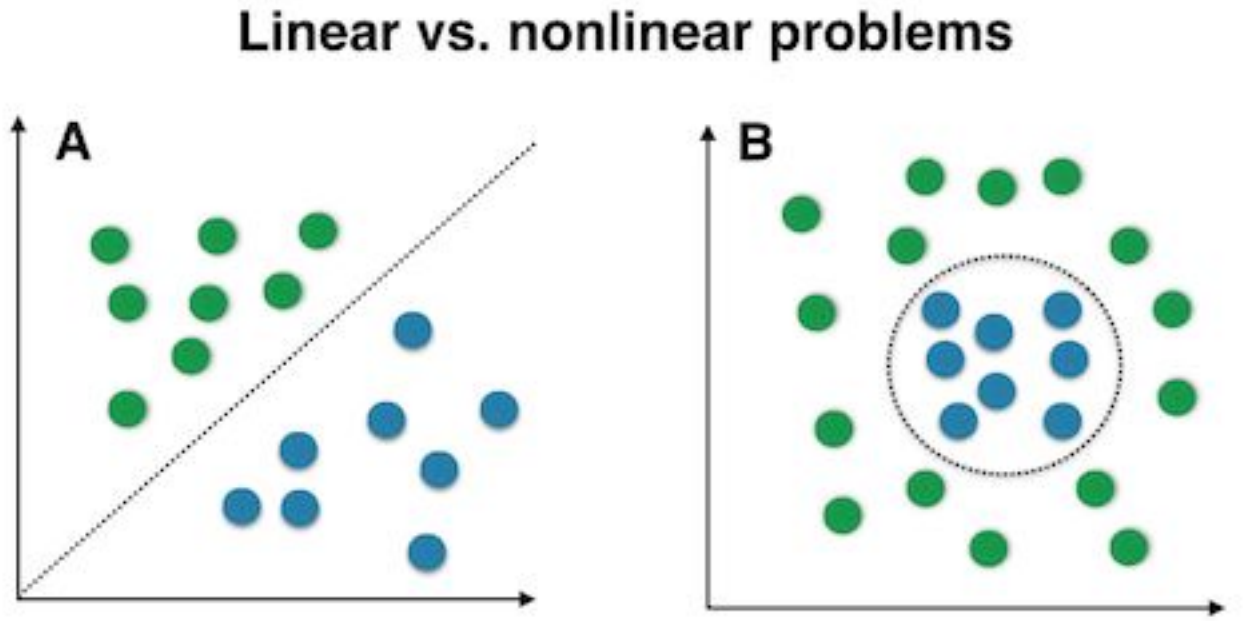
Binary classifiers

- A binary classifier is a classification function having only two classes
  - E.g. F(x) = +1 or -1 depending on the class of x

- We start considering the binary classification problem; later we will see the extension to multiclass problems

Linearly separable problems

- Problems in which the two classes can be separated by a suitably chosen hyperplane
  - In 2D a hyperplane is a straight line; in 3D it is a plane, and so on…



Maximal margin classifier

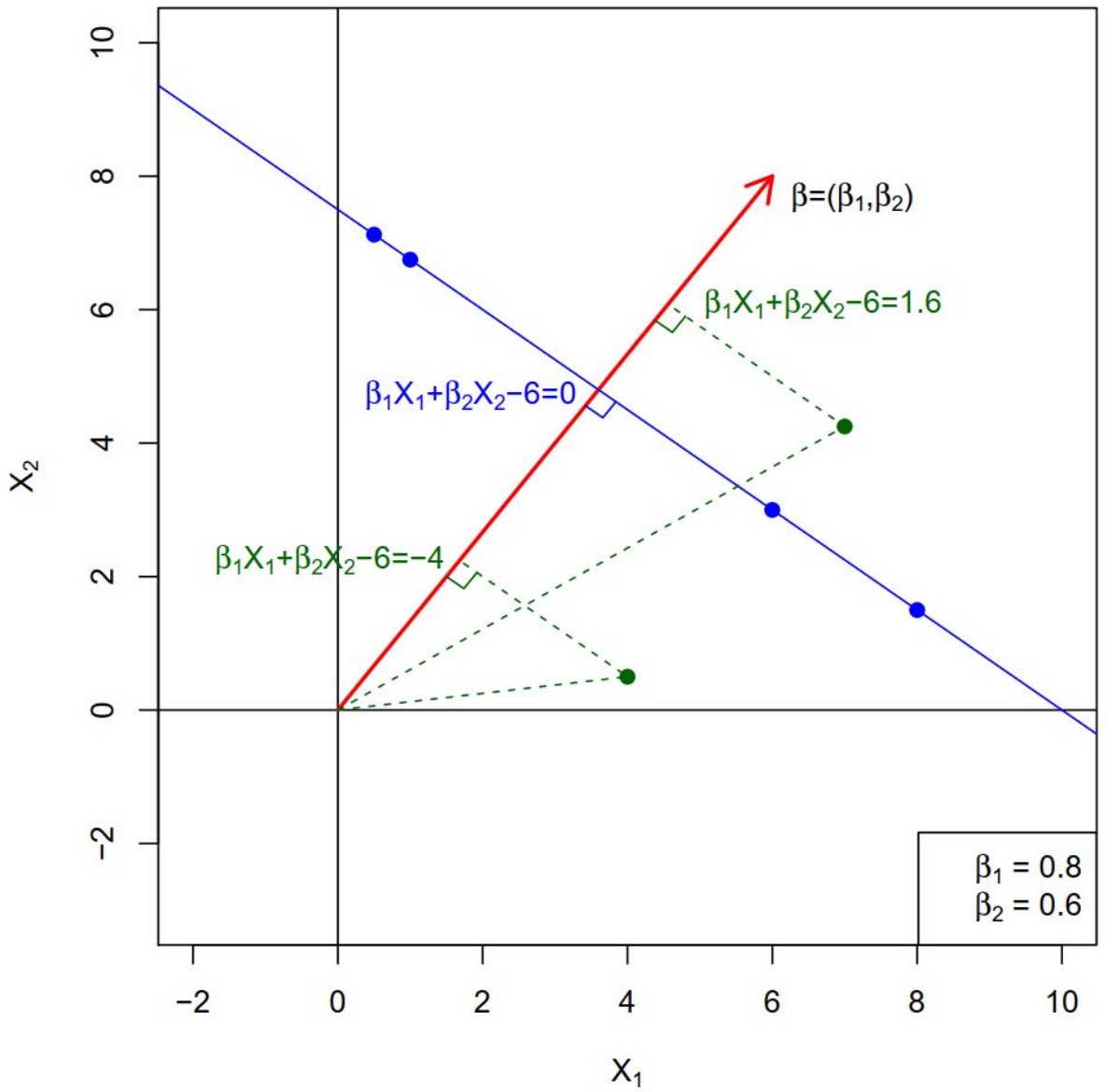
- Classification problem: find a hyperplane that separates the classes in feature space.

- In p dimensions a hyperplane is a flat affine subspace of dimension $p - 1$, with general equation (p=2 a line).

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = x^T \beta + \beta_0 = 0$$ Where:

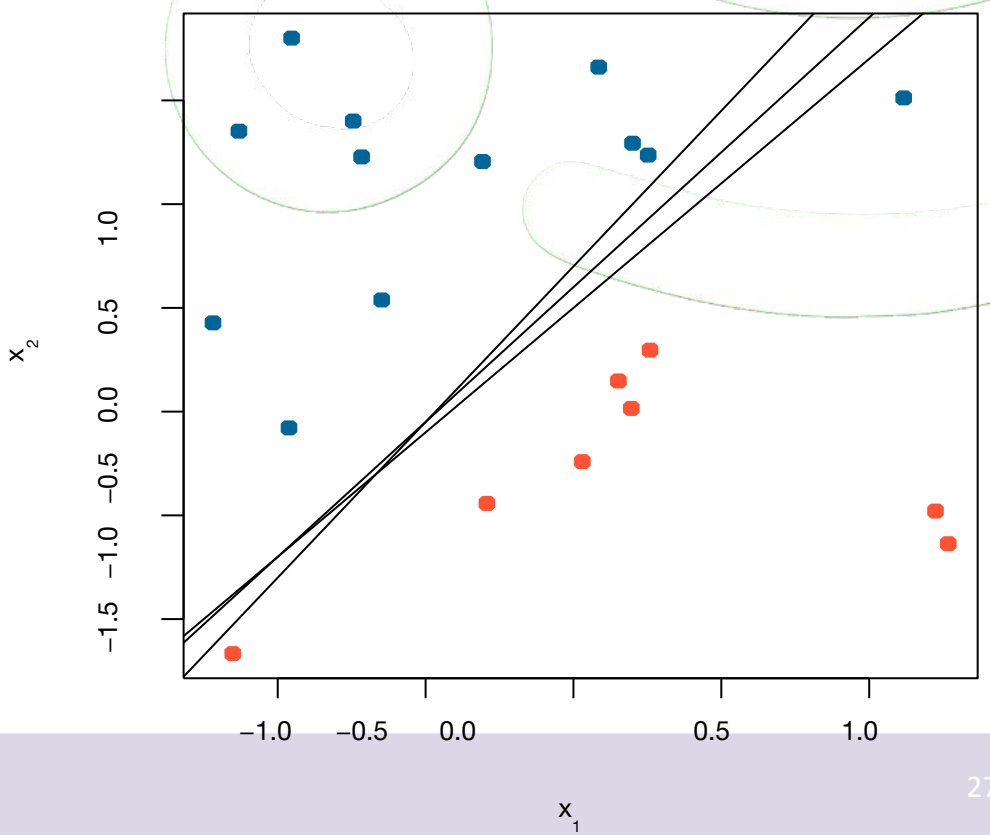
  - $\beta_0 = 0$ only if the hyperplane goes through the origin
  - the vector $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ is a unit vector ($\|\beta\| = 1 \equiv \sum_{j=1}^{p} \beta_j^2 = 1$) orthogonal to the surface of the hyperplane

Hyperplane in 2 Dimensions



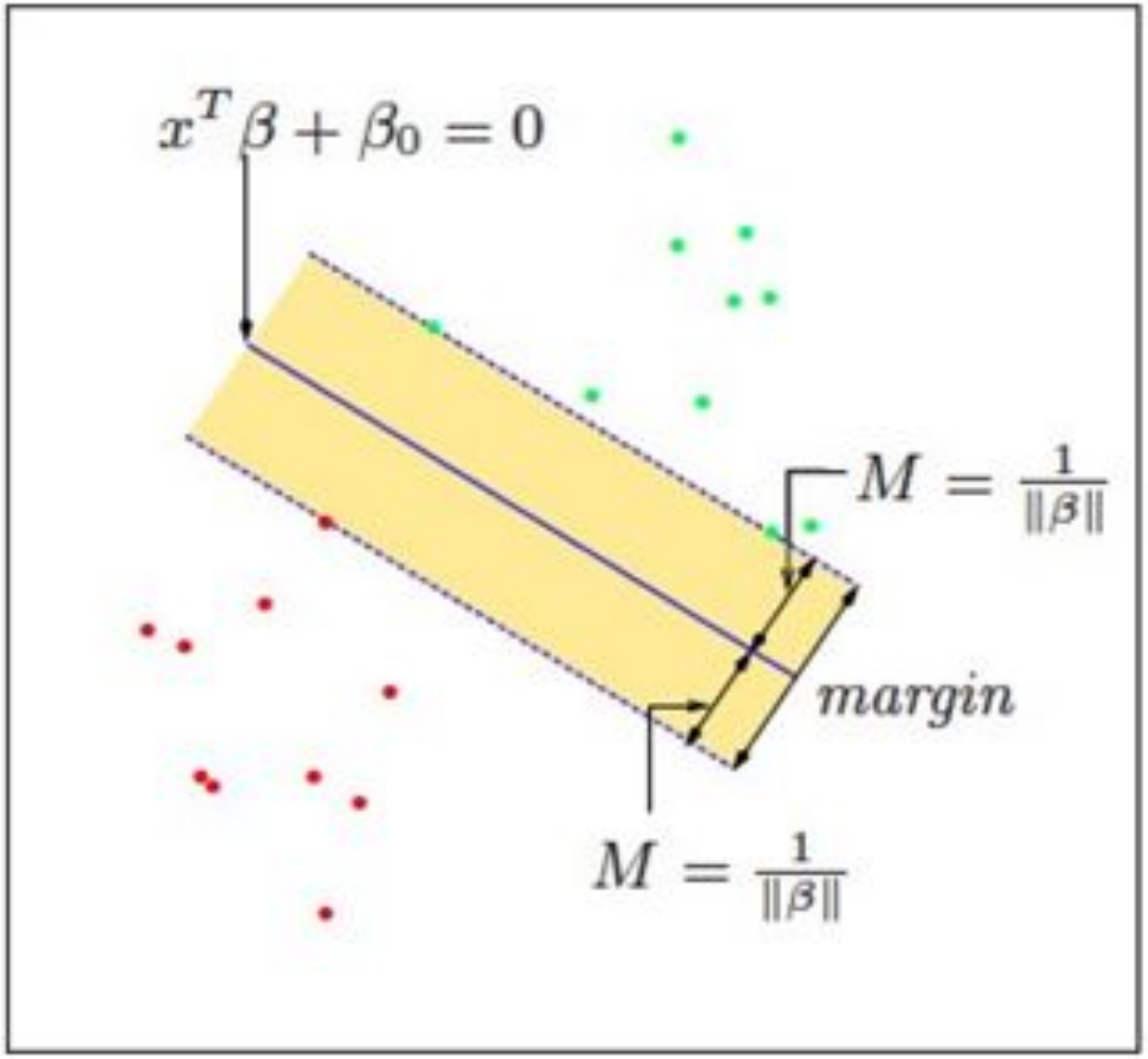
Maximal margin classifier

- Imagine to have a training data of N pairs:
  - $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, with $x_i \in \mathrm{IR}^p$ and $y_i \in \{-1, 1\}$.
- If the classes are perfectly separable, there are generally multiple hyperplanes that can separate them.
- How we choose the best?



Support Vector Machine

- In SVM the choice is to maximize

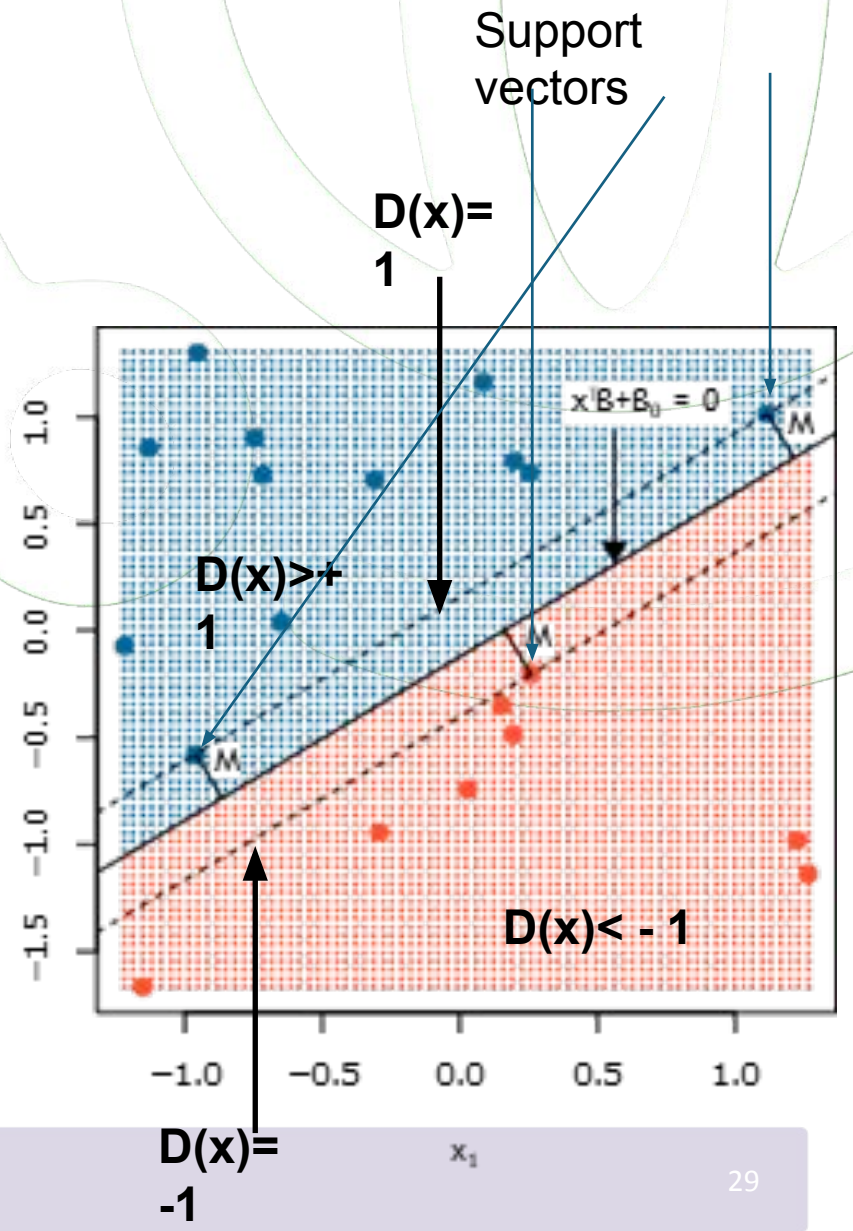


Maximal margin classifier

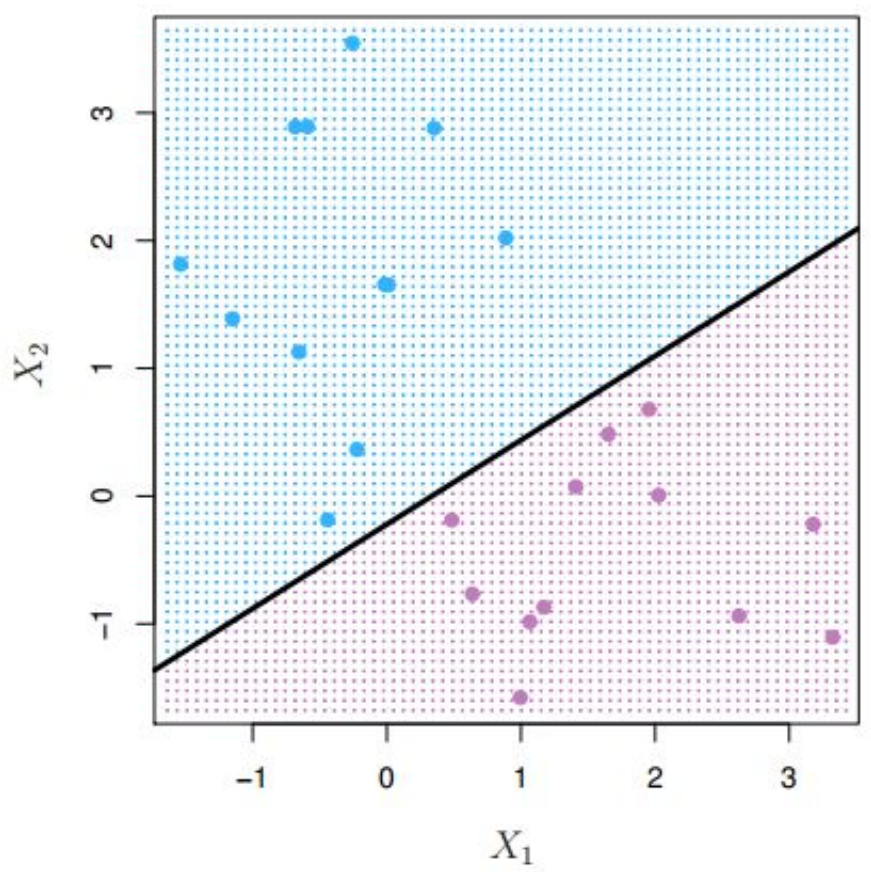
- The maximal margin classifier is the one with biggest margin M between the two classes (blue y_i=+1 and red y_i=-1).

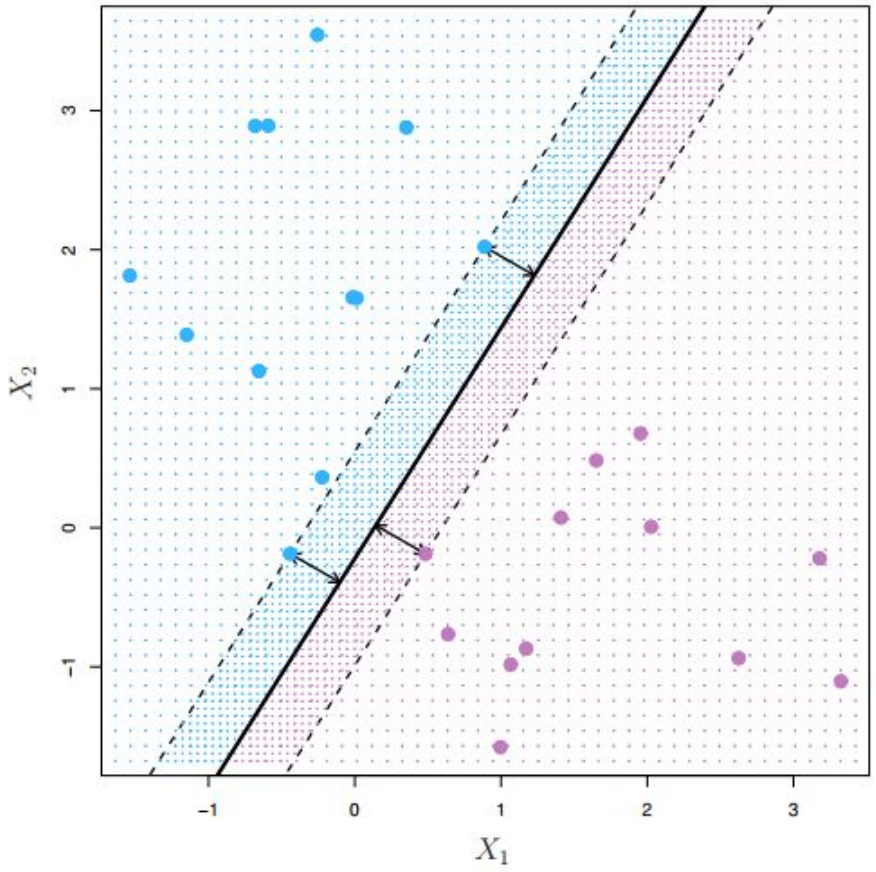
$$\max_{\beta,\beta_0,\|\beta\|=1} M, \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1, \ldots, N$$

- $D(x)=\beta_0 + \beta x^T$ is the **discriminant fuction**
- Point where $D(x_i)=y_i$ are the **Support Vectors**
- **The maximal margine hyperplane depends on support vectors only!**



Maximal marginal hyperplane





Support Vector classifier

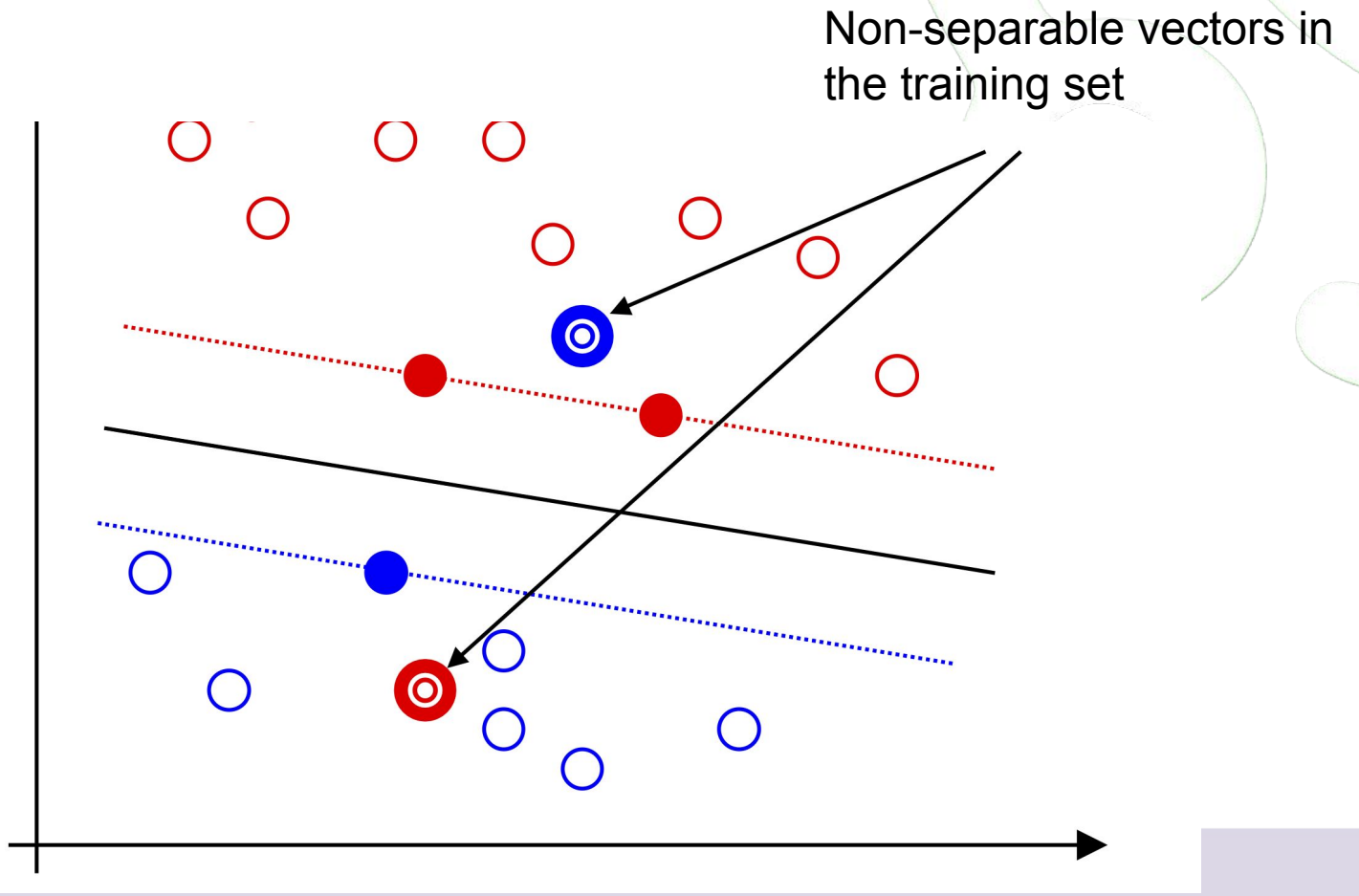
- The **β** vector that maximaze the margin can be calculated as a linear combination of the support vectors x_i: $\boldsymbol{\beta} = \sum_i \alpha_i x_i$
- The coefficients α_i can be determined solving a quadratic optimization problem
  - Computational cost: $O(p*n^2)$, where p is the dimension of the feature vectors

- Once **β** is known, the classification function is:

  - $\text{Class}(x) = \begin{cases} +\mathbf{1}\ if\ D(x) > 0 \\ -\mathbf{1}\ if\ D(x) < 0 \end{cases}$

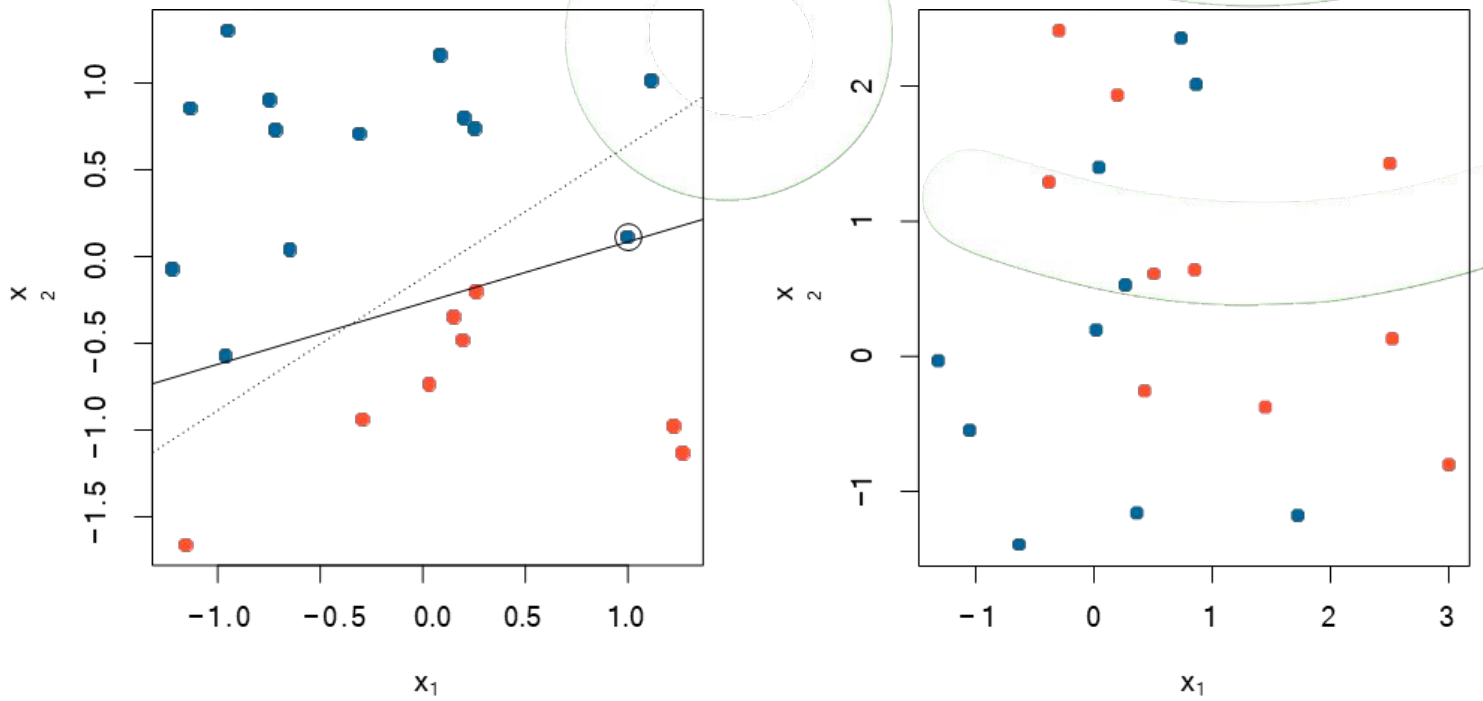
SVM with “soft margin”

- What happens if the problem is “almost” linearly separable?



Noisy or non-separable data

- The maximal margin classifier has issues in case of:
  - Noisy data with outliers leading to poor solution (left panel - just added one data point to the previous example).
  - Data non-separable by linear boundary (right panel).



Support vector classifier

- The *support vector classifier* provides a solution by maximising a *soft* margin (regularization).

- For this we can modify the optimization problem allowing some slack (for 2 predictors).

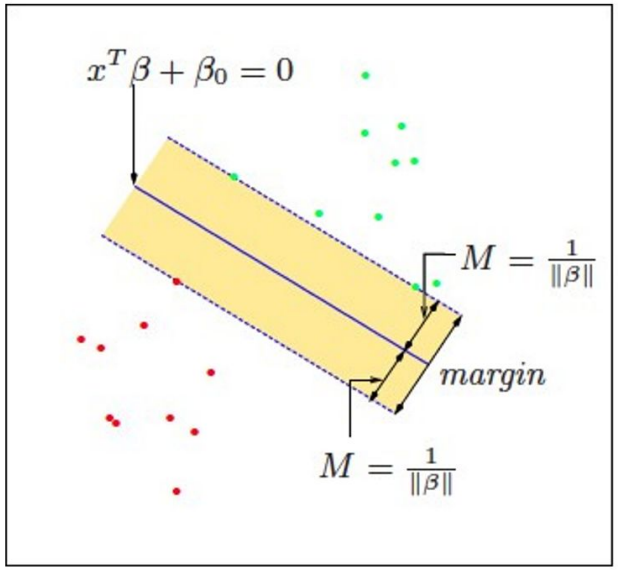
$$\max_{\beta,\beta_0,\|\beta\|=1} M, \text{ subject to } y_i(x_i^T\beta_0 + \beta_i) \geq M(1-\xi), i = 1,\ldots,N$$

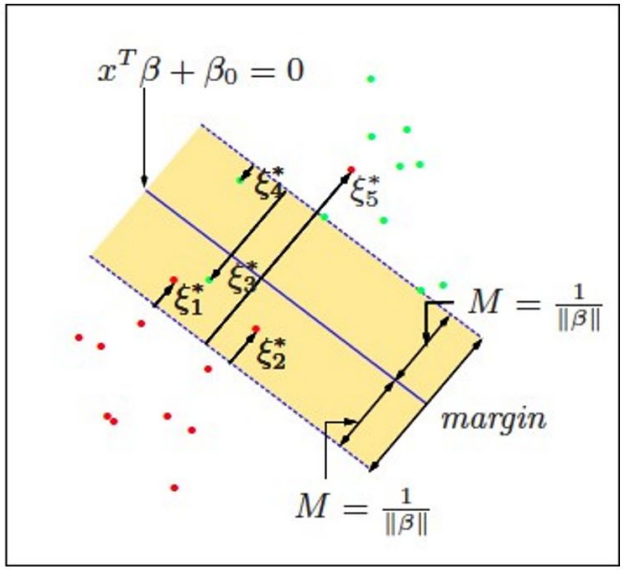
$$where\ \xi_i \geq 0\ and\ \sum_{i=1}^{N}\xi_i \leq C$$

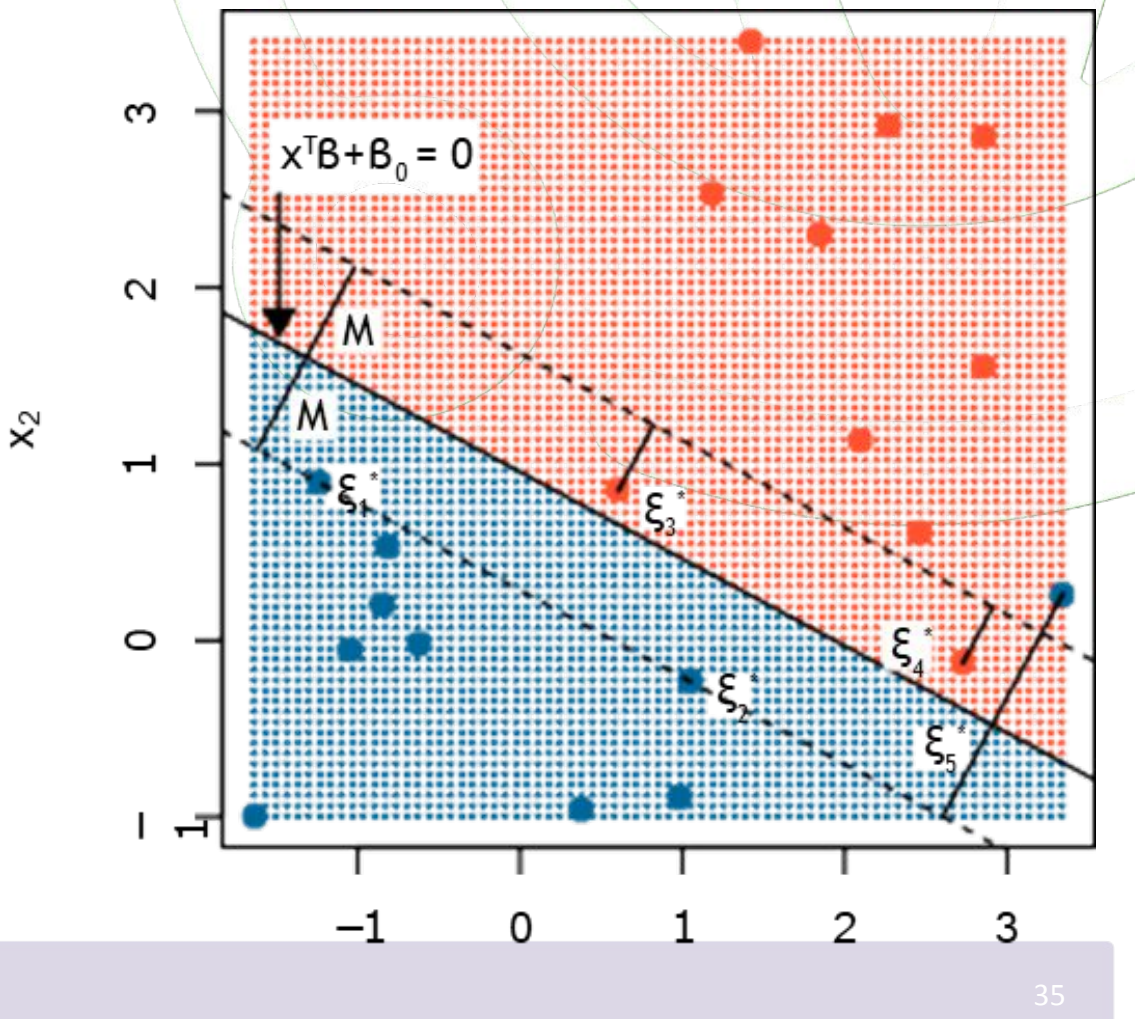
C is a constant that defines the budget we allow for the total amount of slack.

Support vector classifier

$$\max_{\beta,\beta_0,\|\beta\|=1} M, \text{ subject to } y_i(x_i^T\beta+\beta_0) \geq M(1-\xi_i), i=1,\ldots,N$$







Slack variables

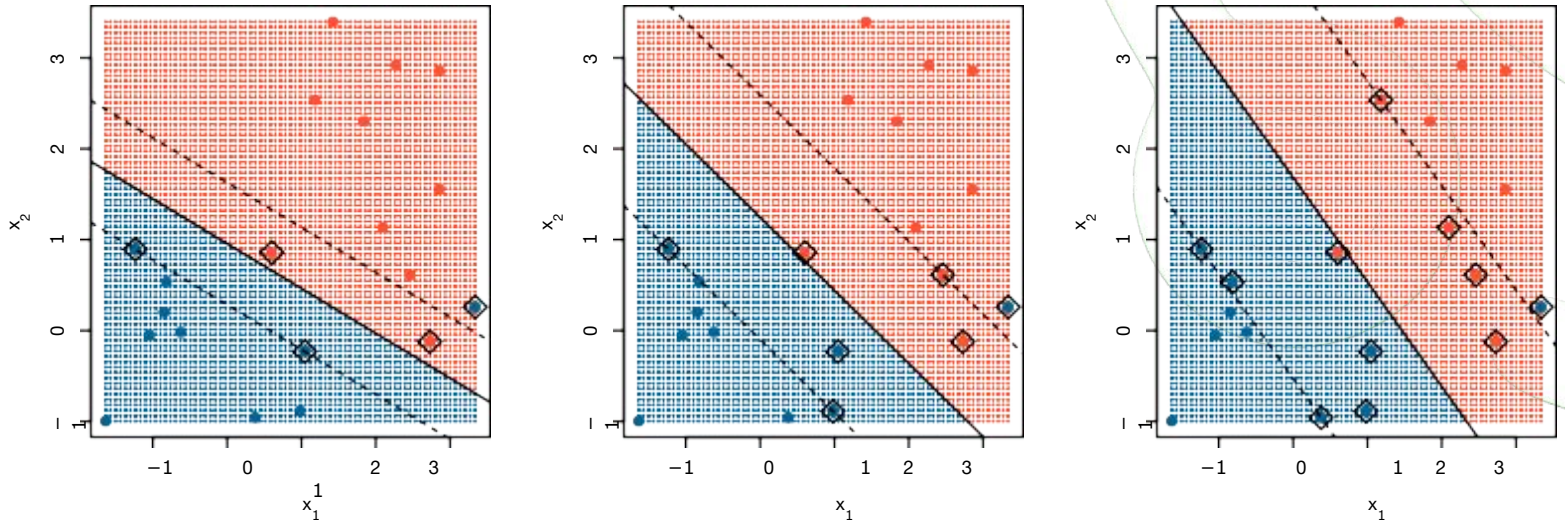
- The slack variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N)$ tell us how much each point is allowed to be on the wrong side of its margin (relative amount).
  - $\xi = 0$ when the i th observation is on the correct side of the margin
  - $\xi > 0$ when the i th observation is on the wrong side of the margin
  - $\xi > 1$ when the i th observation is on the wrong side of the hyperplane

Regularization

- The constant C (slack budget) is tunable and can be seen as a regularization parameter.
  - $C = 0$ no budget for violation of the margin (maximum margin classifier)
  - increasing C allows more slack allowed (wider margins)
- Therefore C controls the bias-variance trade-off:
  - small $C \rightarrow$ narrow margins $\rightarrow$ high fit to the data $\rightarrow$ low bias, high variance
  - large $C \rightarrow$ wide margins $\rightarrow$ more violation allowed $\rightarrow$ high bias, low variance

Example

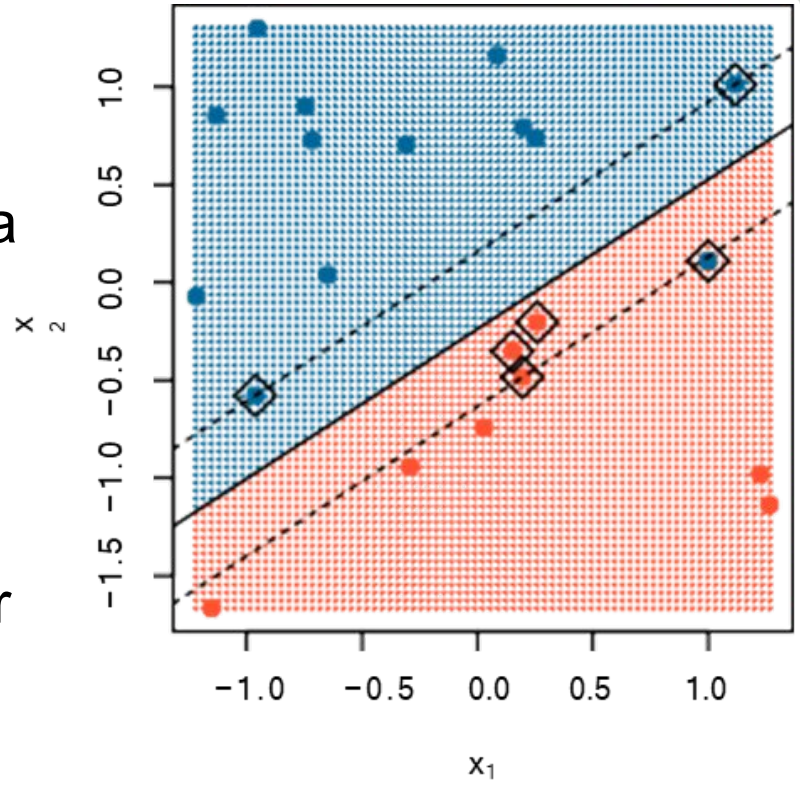
Example of support vector classifier for increasing values of C .

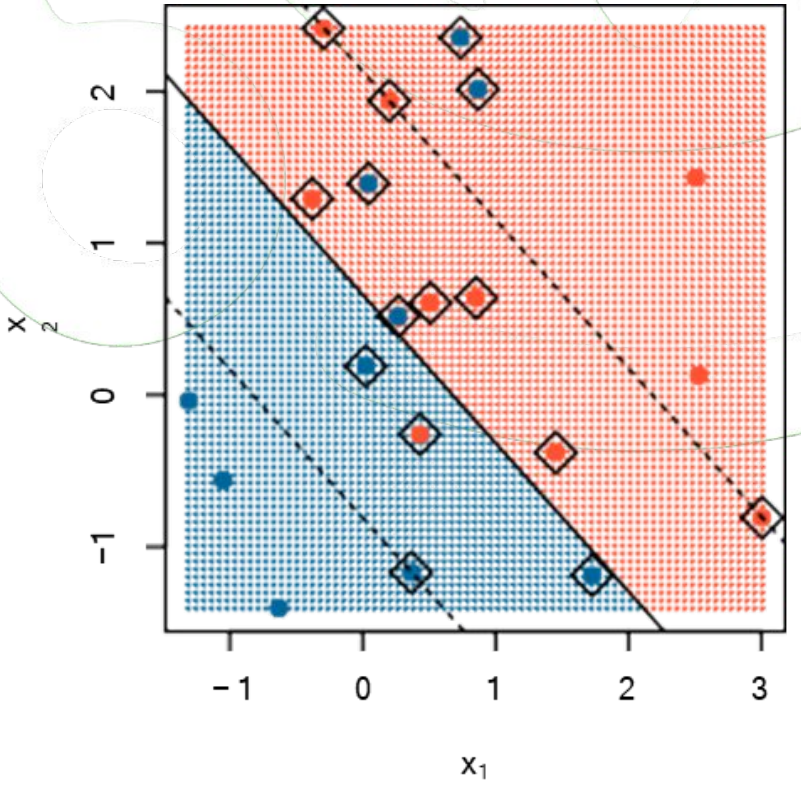


The *support points* (marked with diamonds), i.e. those with $\xi_i \neq 0$, are the only ones that determine the orientation of the margin.

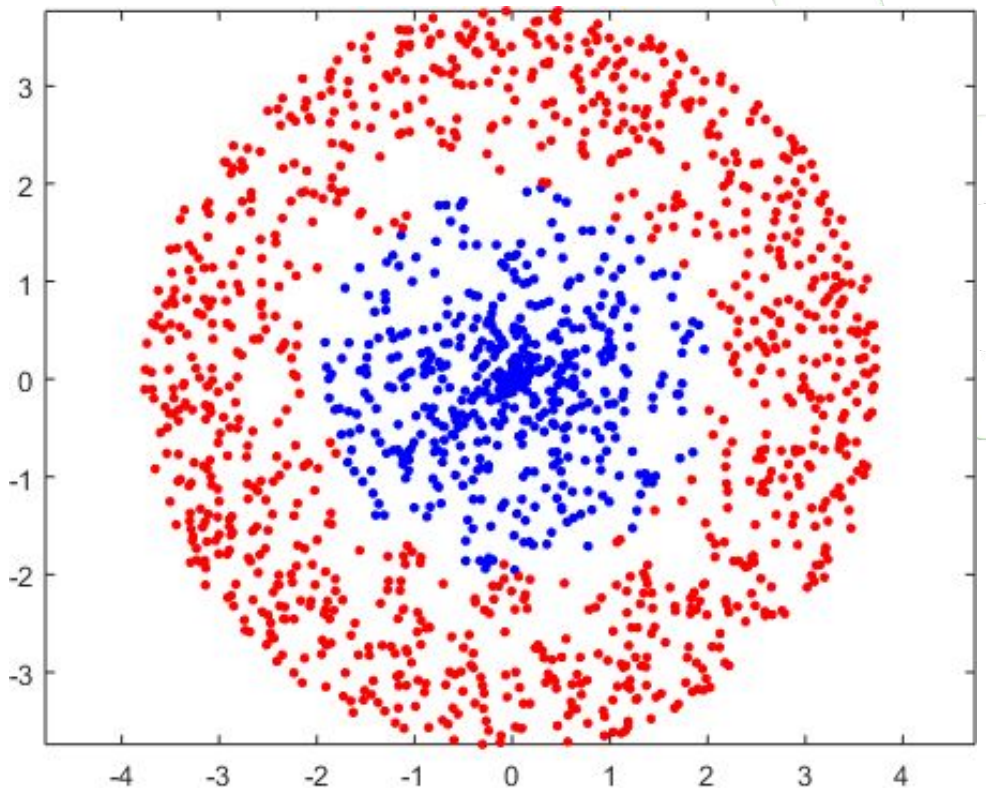
Noisy and non-separable data

- The support vector classifier allows to have a good classifier in both the examples of noisy and non-separable data that we have seen earlier, where the maximal margin classifier was not working properly.





Non-linearly separable classes



Support vector Machine

- Problems:
  - Infeasible for high p
  - Which polynomial order?
- Support vector machines (SVM): extension of the support vector classifier which enlarges the feature space by using kernels.
- The kernel approach is an efficient computational methodology to enlarge the feature space.

Support vector Machine

- The support vector machine is just like the support vector classifier, but it elegantly allows for non-linear expansions of the variables: “non-linear kernels".
- However, linear regression, logistic regression, and other classical statistical approaches can also be applied to non-linear functions of the variables.
- For historical reasons, SVMs are more frequently used with non-linear expansions as compared to other statistical approaches.

Kernel trick

- It has been proved that, under some sufficiently general condition, using a kernel in a space R^d is equivalent to map the vectors of R^d to a different space R^m, usually with m>>d, and using the dot product in R^m
- Kernel trick: if an algorithm only uses the dot product, it can replace the dot product with a (non linear) kernel
  - This is equivalent to perform a non-linear, implicit transformation of the space
  - Possibly in the new space the problem becomes linearly separable…

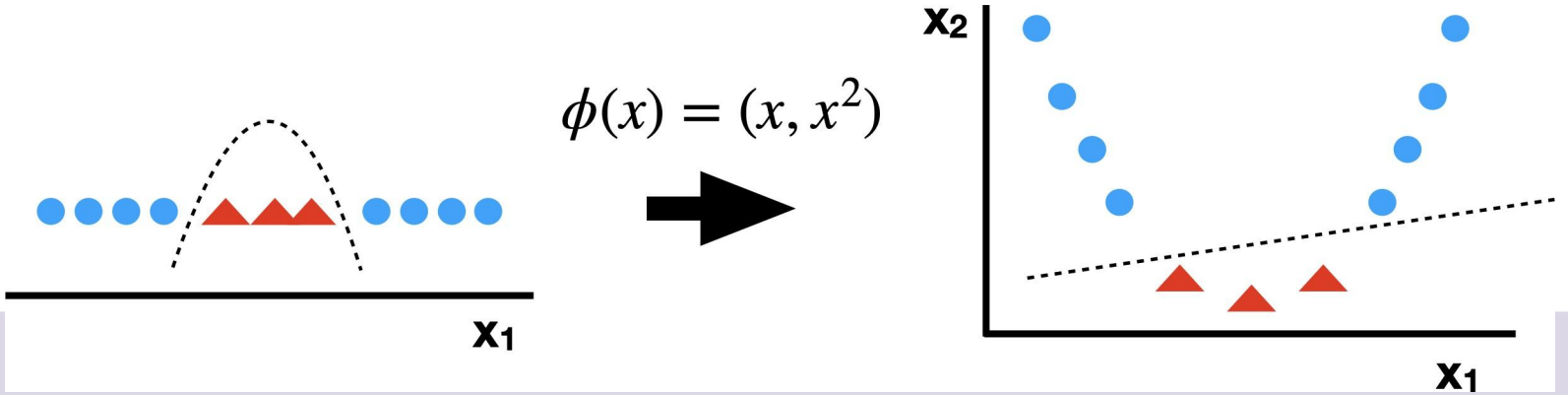
Classification with non-linear decision boundaries

- Extension of the Support vector classifier to handle **non-linear class boundaries**. Idea: use of quadratic, cubic, and even higher-order polynomial functions of the predictors. Example:

$$X_1, X_2 \quad \rightarrow \quad X_1, X_1^2, X_2, X_2^2, X_1 X_2$$

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_2 + \beta_4 X2 X_2^2 + \beta_5 X_1 X_2$$

- Example with one dimension and quadratic function:



Non linear case

- What if the problem at hand is not linearly separable?

- Solution: the “Kernel trick”

Kernel

- A Kernel is a function:

$$K : R^d \times R^d \rightarrow R$$

having some of the properties of the dot product:

$$K(x\ ,x\ ) \geq 0$$

$$K(x_1, x_2) = K(x_2, x_1)$$

$$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$$

Kernelized SVM

- The discriminant function can be written as:

$$
\begin{aligned}
D(x) &= \boldsymbol{\beta} \cdot x^T + \beta_0 \\
&= \left(\sum_i \alpha_i x_i\right) \cdot x^T + \beta_0 \\
&= \beta_0 + \sum_i \alpha_i x_i \cdot x^T
\end{aligned}
$$

Kernelized SVM

- The dot product can be replaced by a kernel:

$$D(x) = \beta_0 + \sum_i \alpha_i K(x_i, x)$$

- This change the shape of the decision regions, that are no more delimited by hyperplanes!
- Note: the choice of the kernel is a hyperparameter
- Applying a kernel may or may not give separable regions…

Kernel examples

- Polynomial kernel:

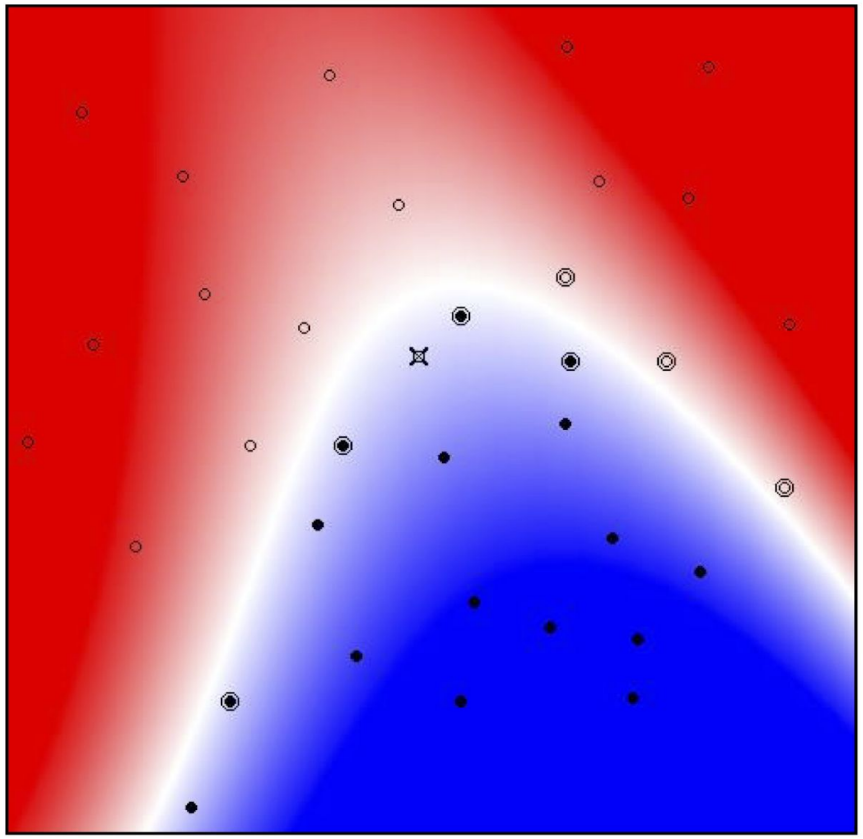
$$K(x_1, x_2) = (x_1 \cdot x_2 + 1)^q$$

- Radial Basis Function (RBF)

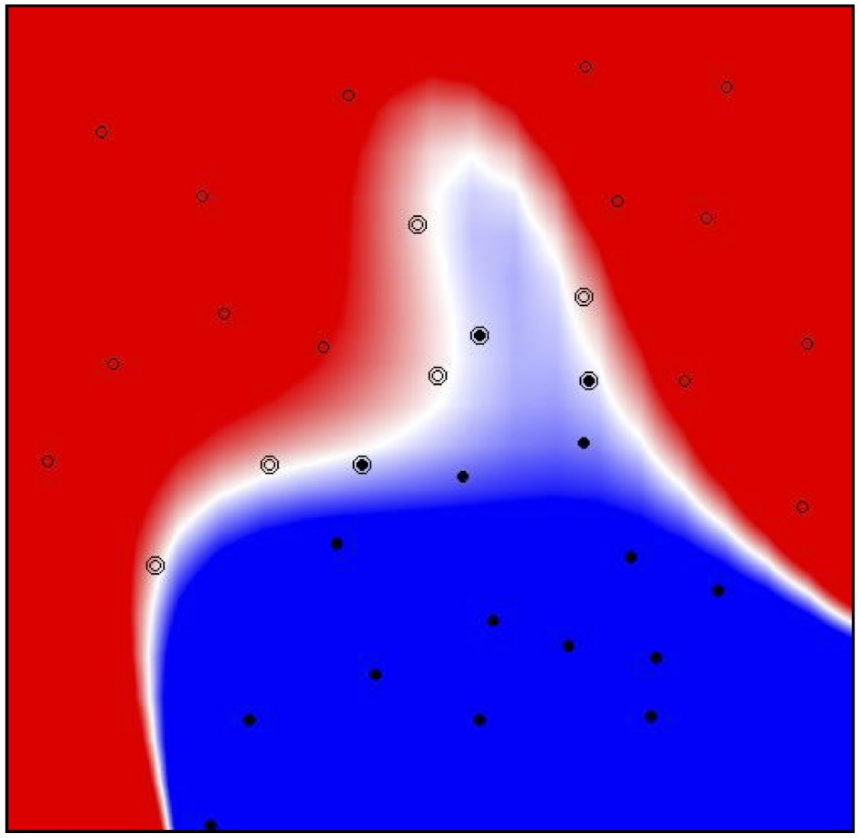
$$K(x_1, x_2) = \exp(-\|x_1 - x_2\|^2 / (2\sigma^2))$$

  - Some libraries use as a parameter $\gamma = 1/(2\sigma^2)$

Kernel examples

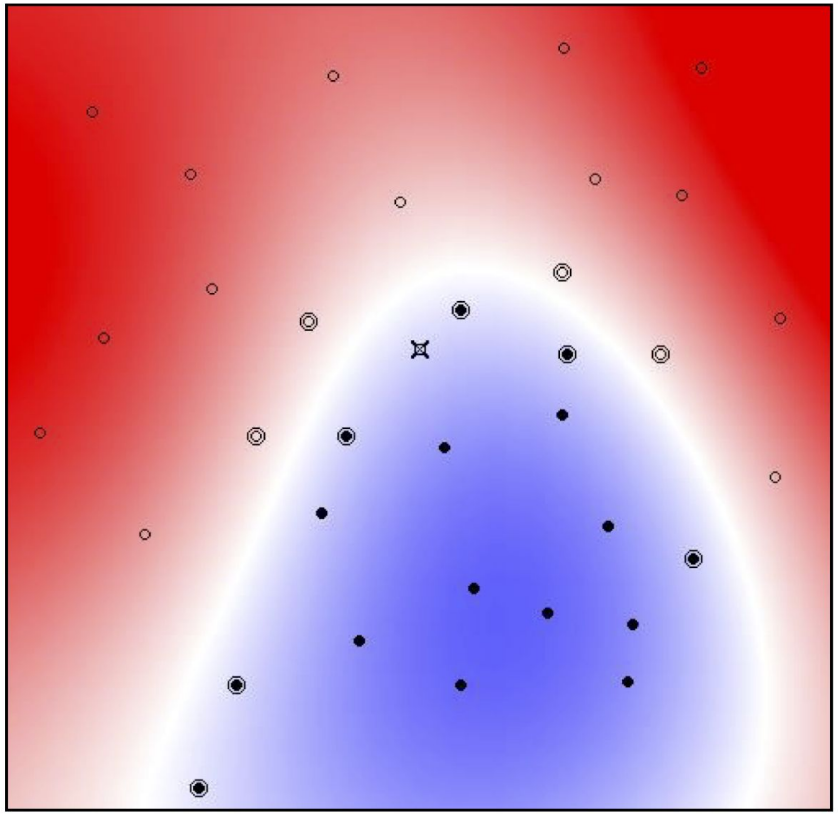


Poly, q=2

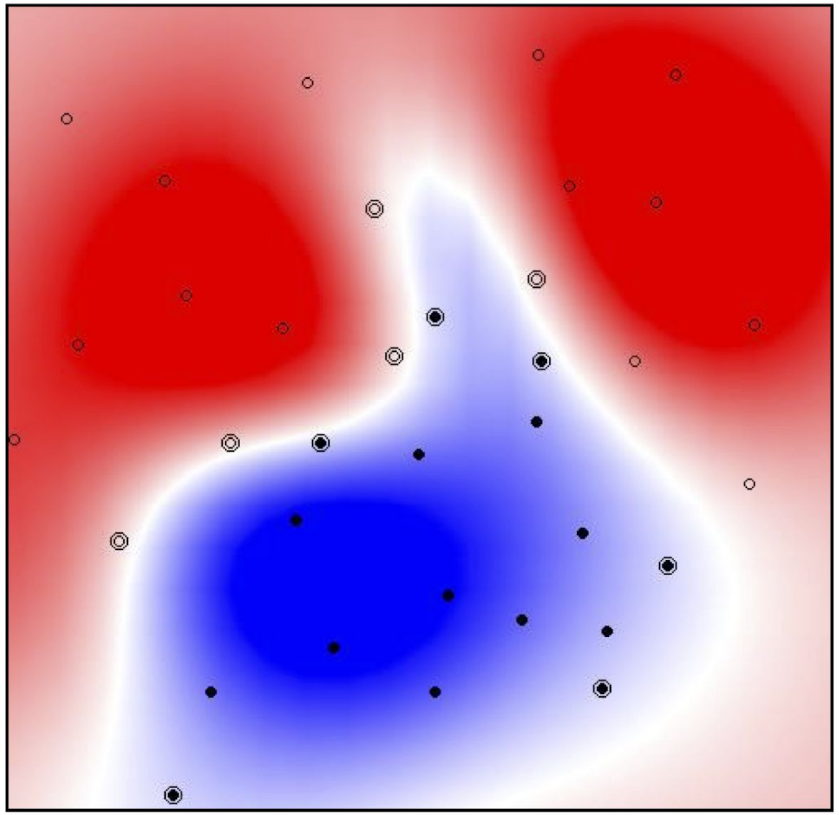


Poly, q=10

Kernel examples

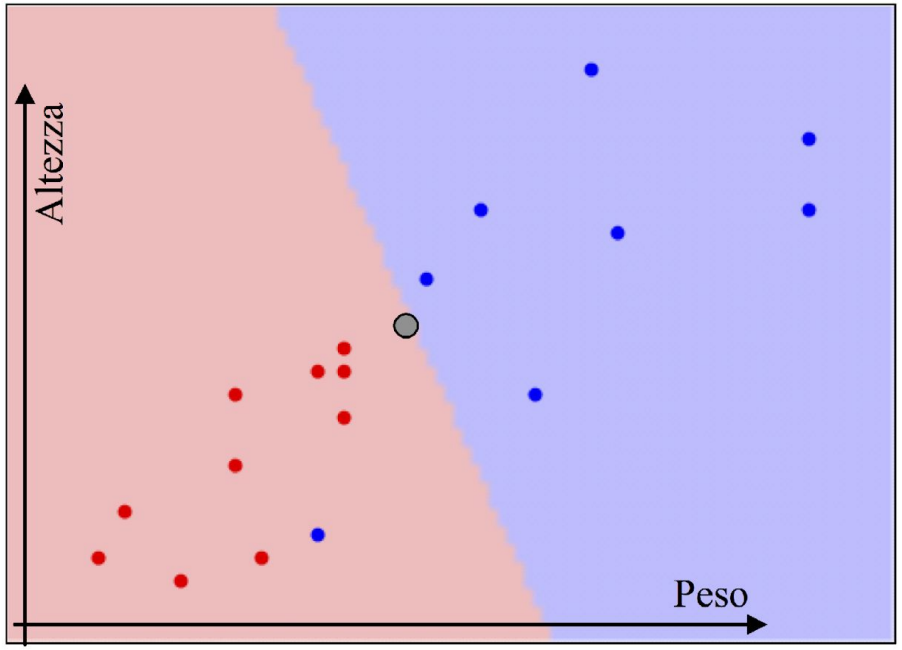


RBF, σ=1

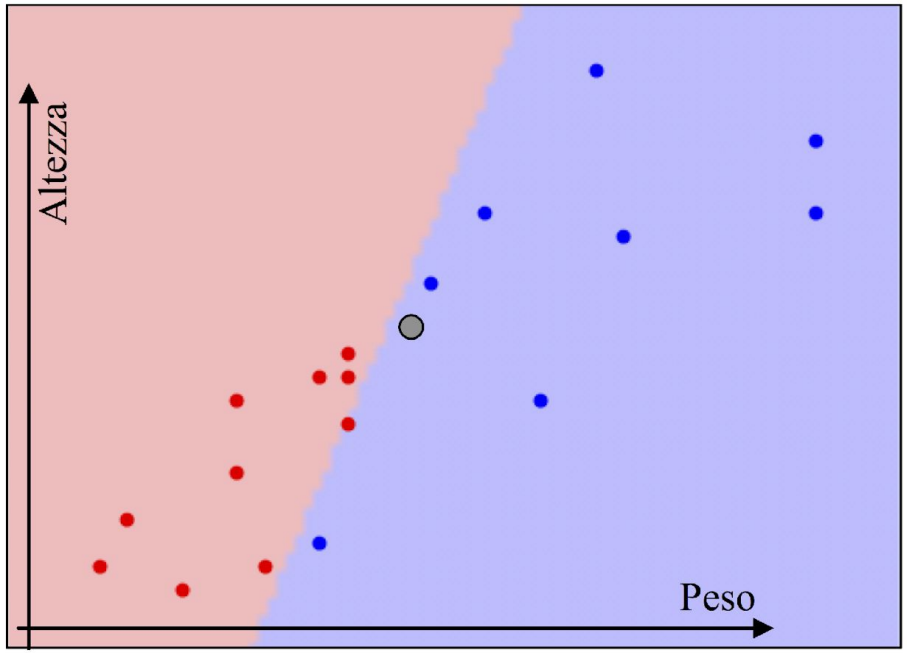


RBF, σ=0.2

Kernel examples

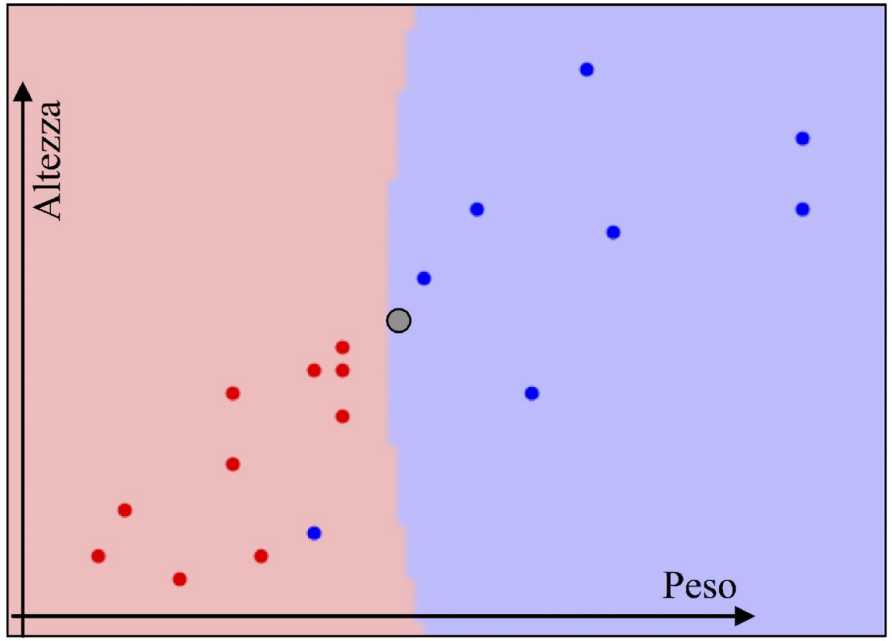


Lineare, $C = 10$

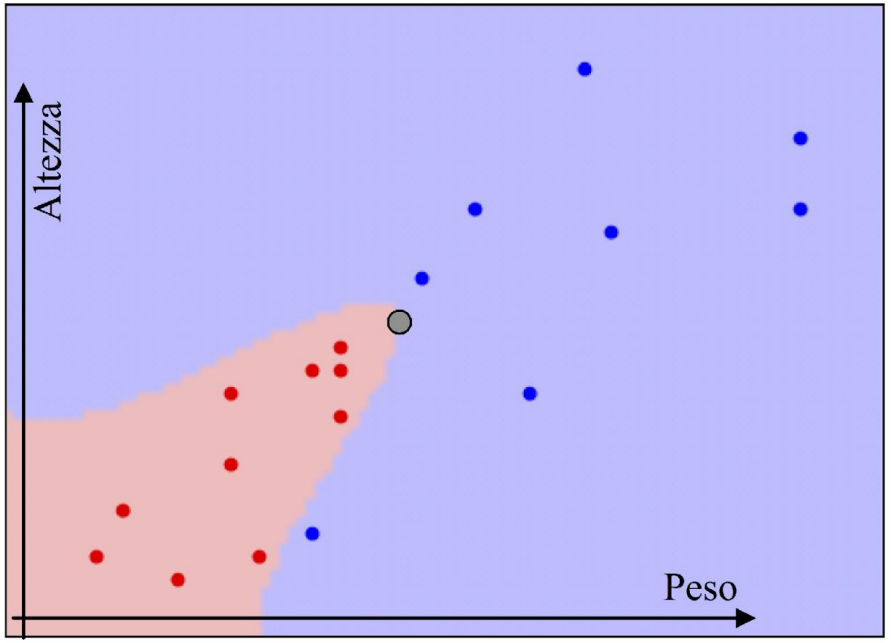


Lineare, $C = 500$

Kernel examples

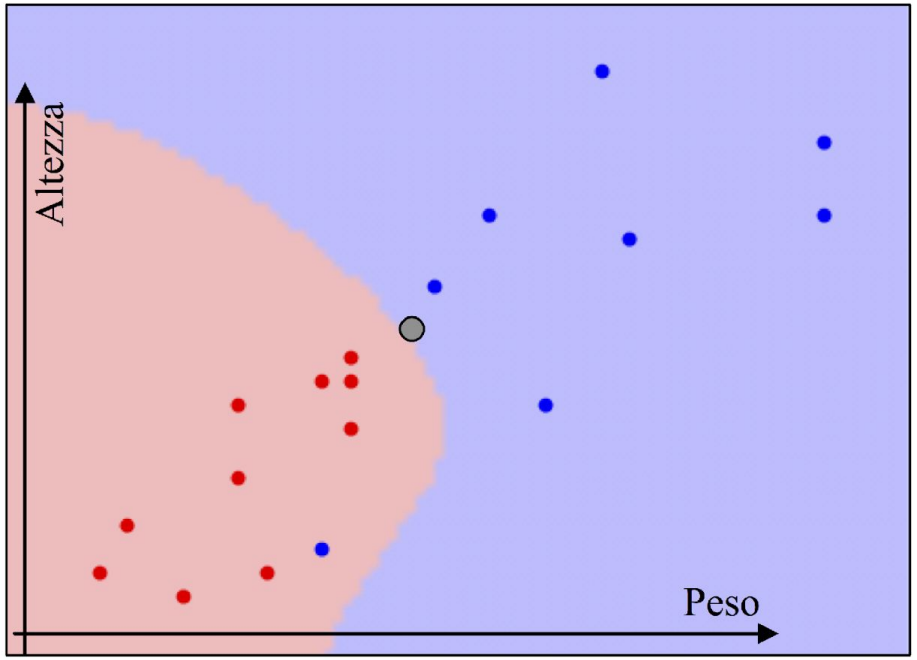


Polinomio $q = 3, C = 10$

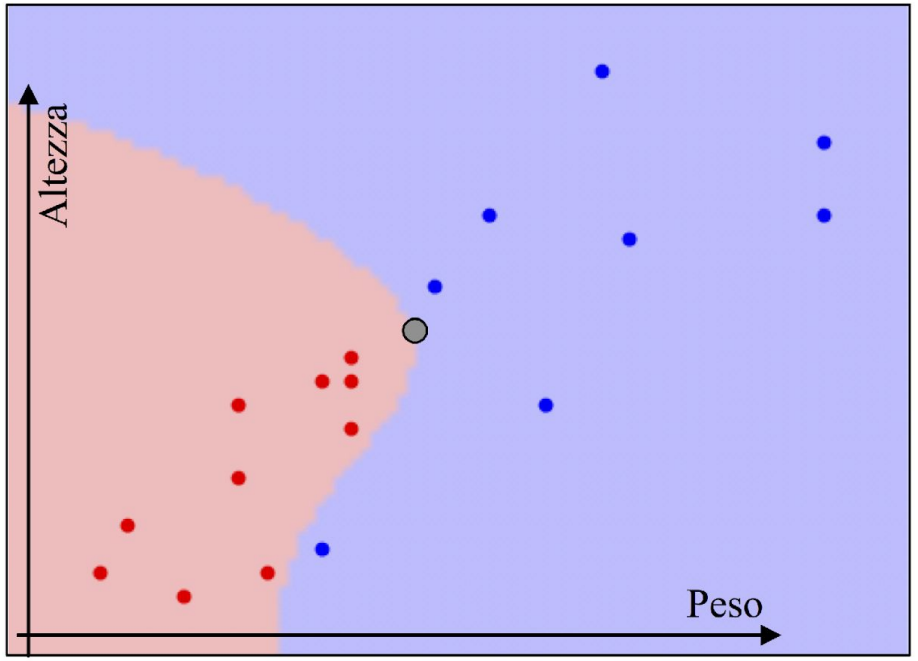


Polinomio $q = 3, C = 500$

Kernel examples

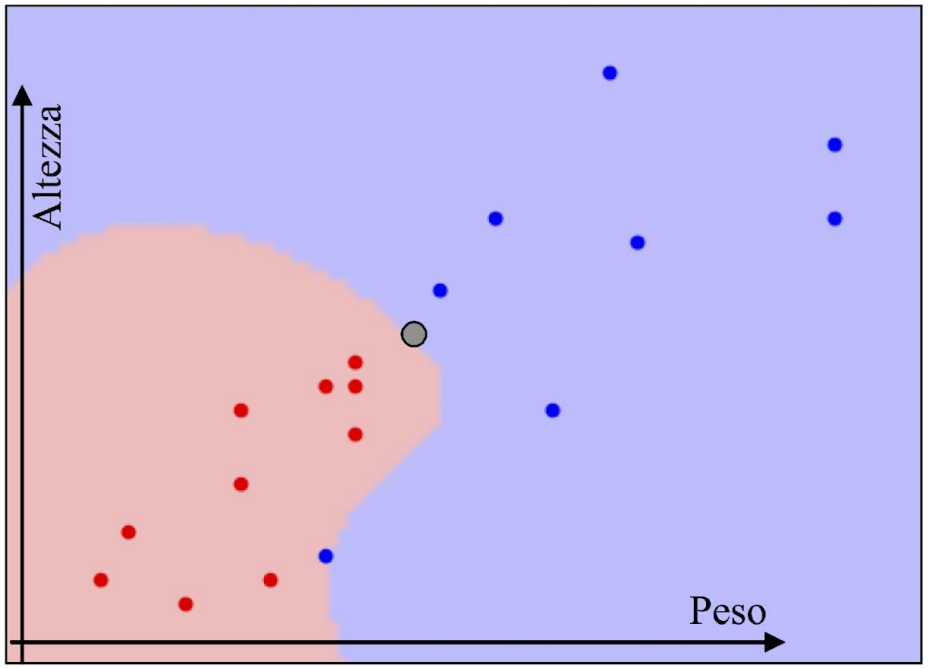


RBF, $\gamma = 5, C = 10$

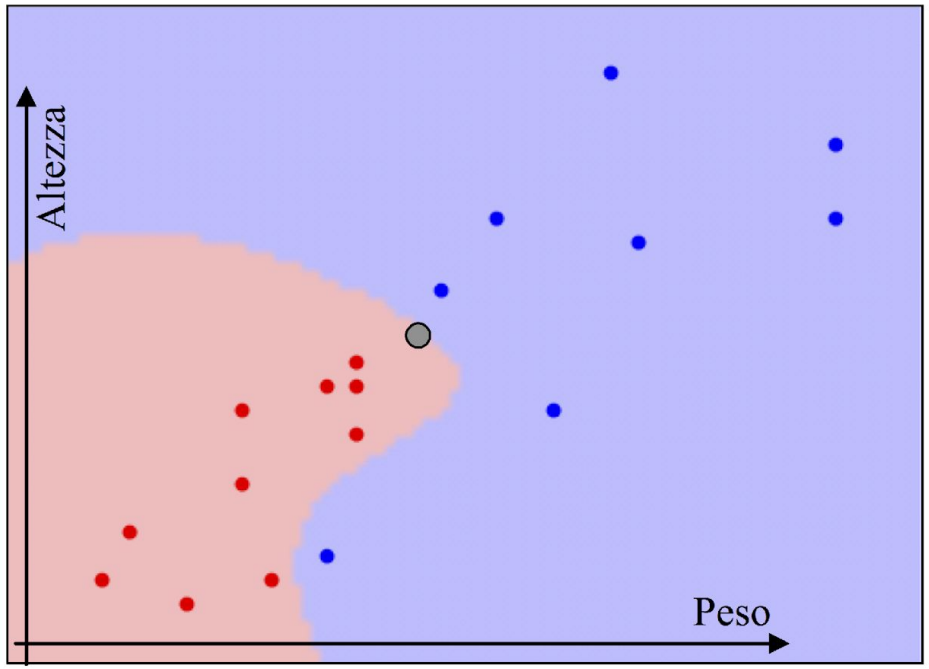


RBF, $\gamma = 5, C = 500$

Kernel examples



RBF, $\gamma = 10, C = 10$



RBF, $\gamma = 10, C = 500$

Is A Non-Linear Kernel Better?

ITINERIS

- **Yes**, if the true decision boundary between the classes is non-linear, and you have enough observations (relative to the number of features) to accurately estimate the decision boundary.
- **No**, if you are in a very high-dimensional setting such that estimating a non-linear decision boundary is hopeless.

Kernelized SVM

- NOTE: with a non linear kernel, the classifier must keep all the support vectors in memory
  - Spatial complexity
  - Temporal complexity
- For low-dimensional problems, usually kernelized versions are preferred
- For high-dimensional problems (thousands of features) is more likely that the problem is linearly separable

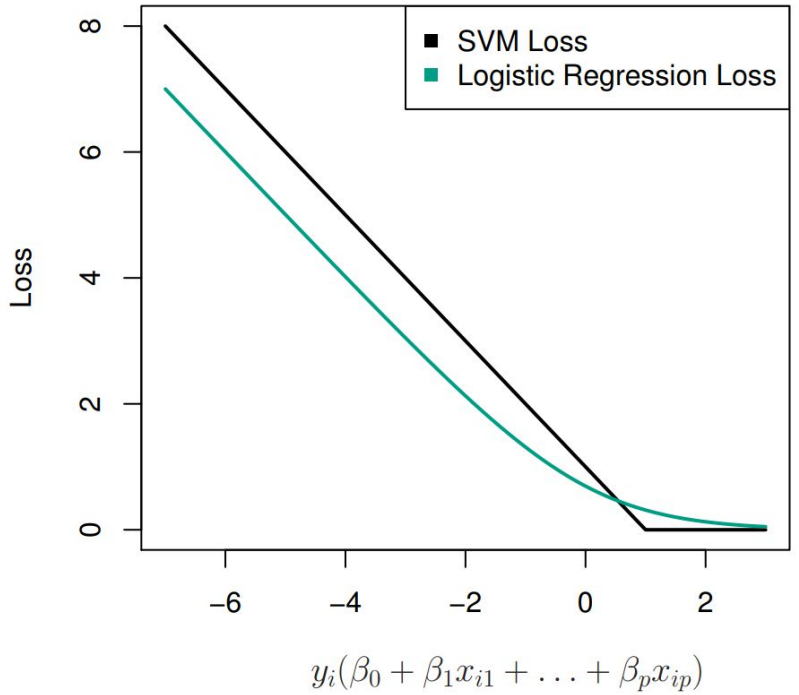
Extension to multi-class

- So far, binary classification, in other words, two-class setting.
- SVMs: concept of separating hyperplanes does not lend itself to more than two classes.
- Two approaches for extending SVMs to K>2 classes classification:
- One-Versus-One Classification: $\binom{K}{2}$ SVMs comparing pair of classes. We assign the test observation to the class most frequently selected in these pairwise classification.
- One-Versus-All Classification: K SVMs, each time comparing one of the K classes to the remaining K-1 classes. We assign the test observation to the class (SVM in this case) with the best discrimination rule

Support Vector versus Logistic Regression?

With $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ can rephrase support-vector classifier optimization as

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$



This has the form *loss plus penalty*.

The loss is known as the *hinge loss*.

Very similar to "loss" in logistic regression (negative log-likelihood).

Which to use SVM vs Logistic Regression

- When classes are (nearly) separable, SVM does better than LR.
- When not, LR (with ridge penalty) and SVM very similar.
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR as well, but computations are more expensive.

In High Dimensions…..

- In SVMs, a tuning parameter controls the amount of flexibility of the classifier.
- This tuning parameter is like a **ridge penalty**, both mathematically and conceptually. The SVM decision rule involves all of the variables.
- Can get a sparse SVM using a **lasso penalty**; this yields a decision rule involving only a subset of the features.
- Logistic regression and other classical statistical approaches could be used with non-linear expansions of features. But this makes high-dimensionality issues worse.

Tree-based classifiers

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Decision trees

- A non-parametric model used for classification and regression
- Give good results on both numerical and categorical features
- Automatically ignore irrelevant features
- Make few assumptions on the input data
- The model is easy to understand and the results are are easy to interpret (in general)

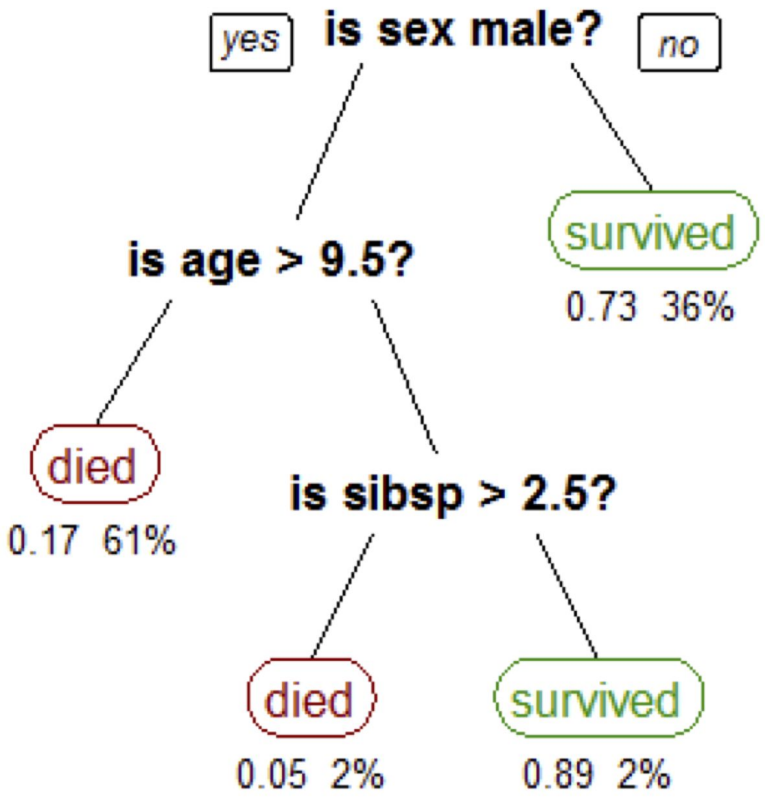
Decision trees

- To grow a decision-tree we need a set of training data with N observations consisting in p inputs and a response $(x_1, y_1) \ldots (x_N, y_N)$, where each $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$ is a vector of feature measurements for the i th case.

- In a (binary) decision tree, each internal node divides the patterns into two groups using a test on the value of a single feature
- The process is repeated recursively using the children of the node, until a leaf node is reached
- Leaf nodes represent groups of patterns that are easy to classify (e.g. using the class of the majority of them) or to regress (e.g. using a constant value corresponding to the average y of the group)

Example

- Decision tree for predicting the survival of Titanic passengers:



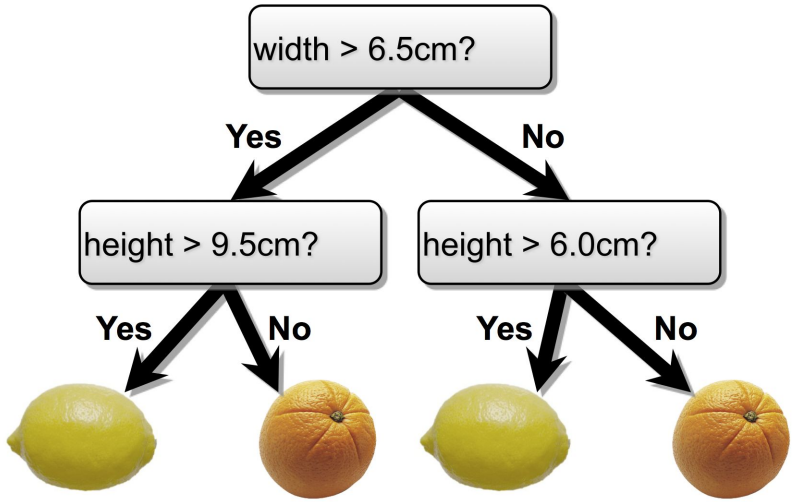
Sibsp = number of siblings on board

The number on the left is the frequency of survival

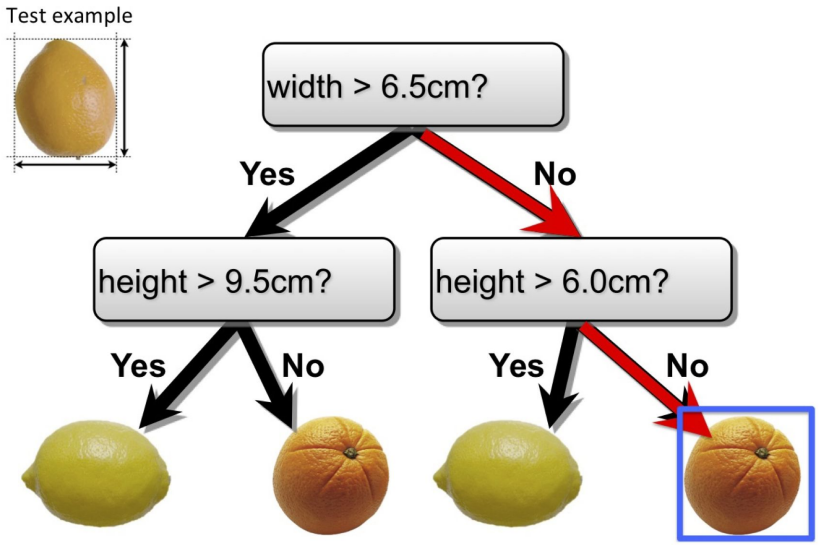
The % on the right is the size of the group wrt to the dataset

(source: Wikipedia)

Example

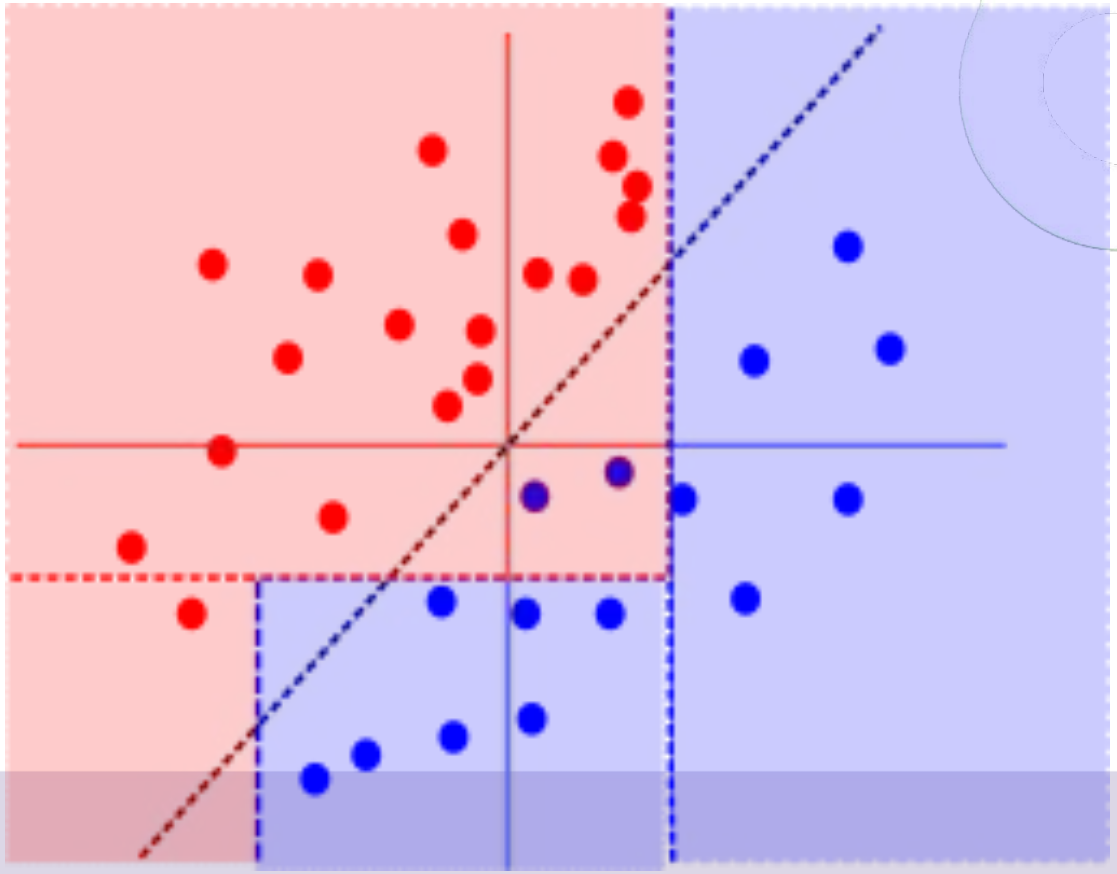


Example

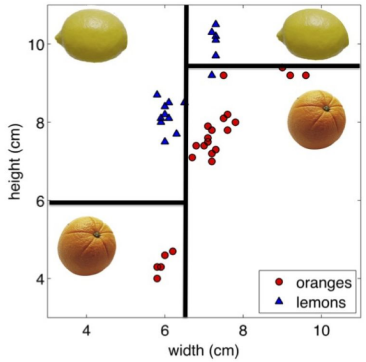

Test example
width > 6.5cm?
Yes
No
height > 9.5cm?
height > 6.0cm?
Yes
No
Yes
No

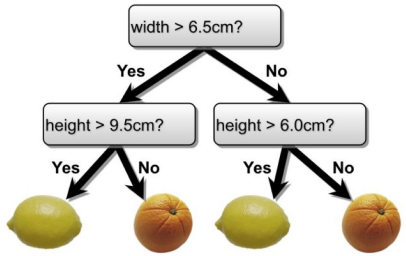
Decision regions

- Decision trees divide the feature space into regions that are piece-wise bounded by hyperplanes ortogonal to the feature space axes (in 2D, horizontal and vertical lines)



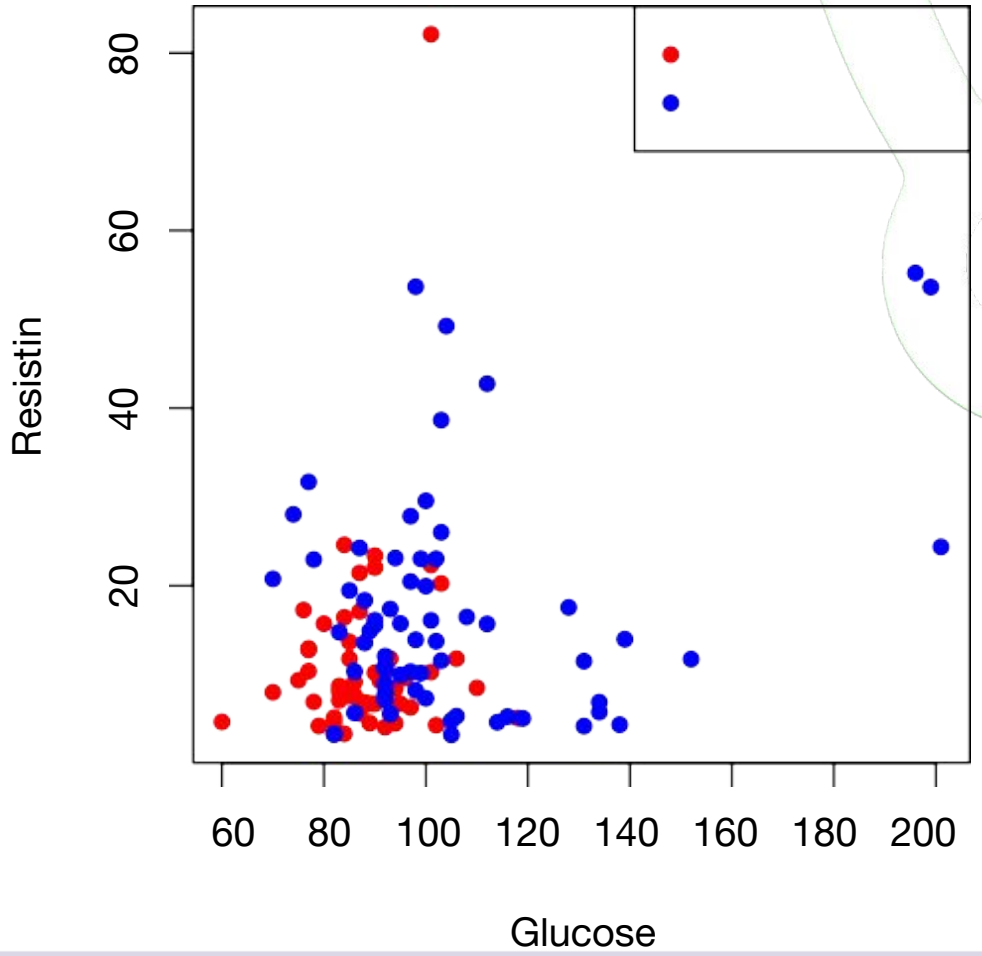
Decision regions





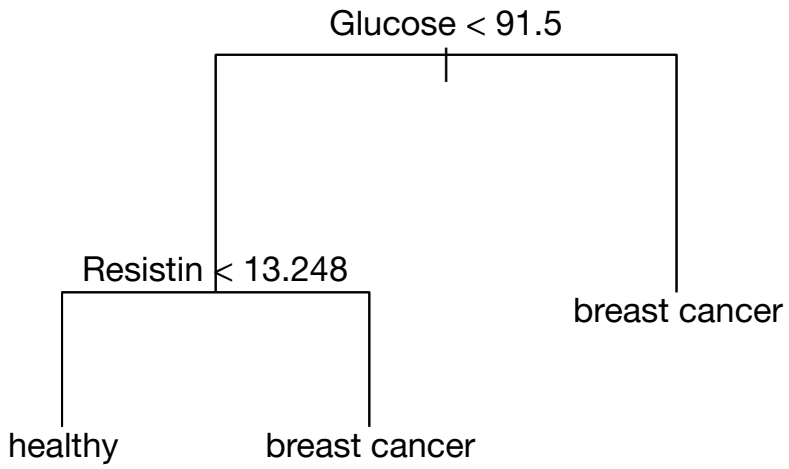
Example

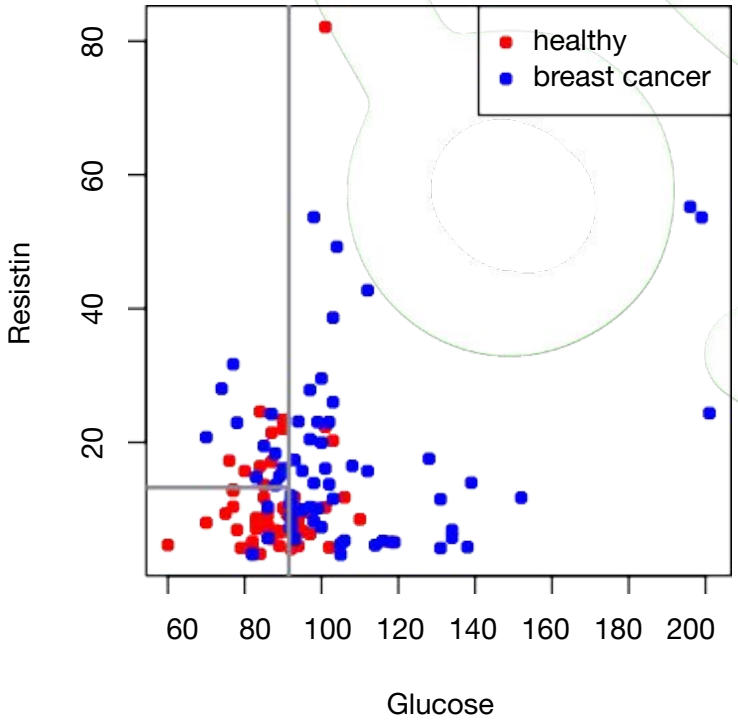
- Coimbra Breast Cancer dataset (Patricio, M., et al, BMC Cancer, 2018)



Example

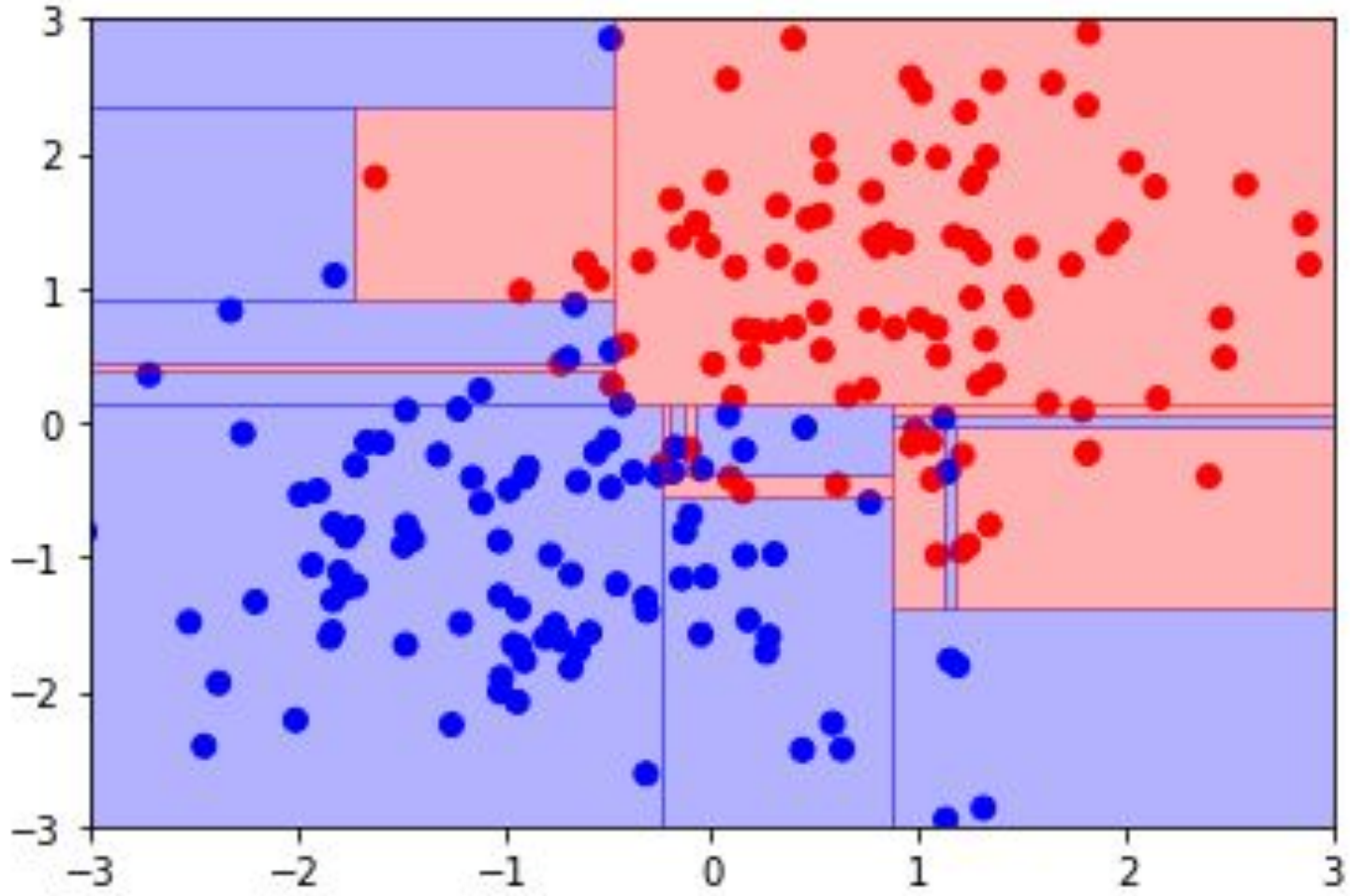
- Coimbra Breast Cancer dataset (Patricio, M., et al, BMC Cancer, 2018)





Decision regions

- These regions can be arbitrarily complex…



Learning decision trees

- We will limit our attention to decision tree classifiers (but the algorithms for regressors are similar – scikit-learn)

- Several algorithms have been proposed (ID3 - Iterative Dichotomiser 3, C4.5, CART – Classification And Regression Tree), but they have a very similar structure

Learning decision trees

- We can build a consistent decision tree that gives **100% accuracy** on the training set, if the classification is deterministic (e.g. a given feature vector x can only have one class)
  - The tree will have one training sample per leaf
  - … probably this tree will not generalize too well!

- To improve generality, we want to *reduce* the number of nodes
  - Occam's razor: the simpler solution is more likely to be correct

Learning decision trees

- The algorithms are based on a recursive subdivision of the training set until a termination condition is met

- The algorithms try to reduce the number of nodes by choosing at each step the **“best” feature** and **threshold** value for the test

- The choice is guided by an “impurity” function:

  - the best test is the one that yields two subsets which are more “pure”, i.e. closer to contain one single class

The basic algorithm

Function MakeTree(*T* : training set) : Tree

if TerminationCondition(*T*):

return Leaf(majority class of *T*)

for each feature *f*:

for each value *v* assumed by *f* in *T*:

divide *T* into T_{left} and T_{right} using the condition $f \leq v$

compute the average “impurity” of the division

$h = (n_{left} * H(T_{left}) + n_{right} * H(T_{right})) / n$

keep the subdivision with the smallest *h*

make a node *N* with condition $f_{best} \leq v_{best}$

N.left := MakeTree(T_{best_left})

N.right := MakeTree(T_{best_right})

return *N*

The impurity function

- The impurity function H is the criterion that guides the algorithm towards subdivisions that make the classification easier
- H(T) = 0 if the samples in T belong to a same class; otherwise H(T)>0
- Usually expressed in terms of the probabilities (frequencies) of $p_1,\ldots,p_k$ of the classes in T

Commonly used impurity functions

- Gini impurity:

$$H(T) = \sum_{i=1}^{k} p_i \cdot (1 - p_i)$$

- Entropy:

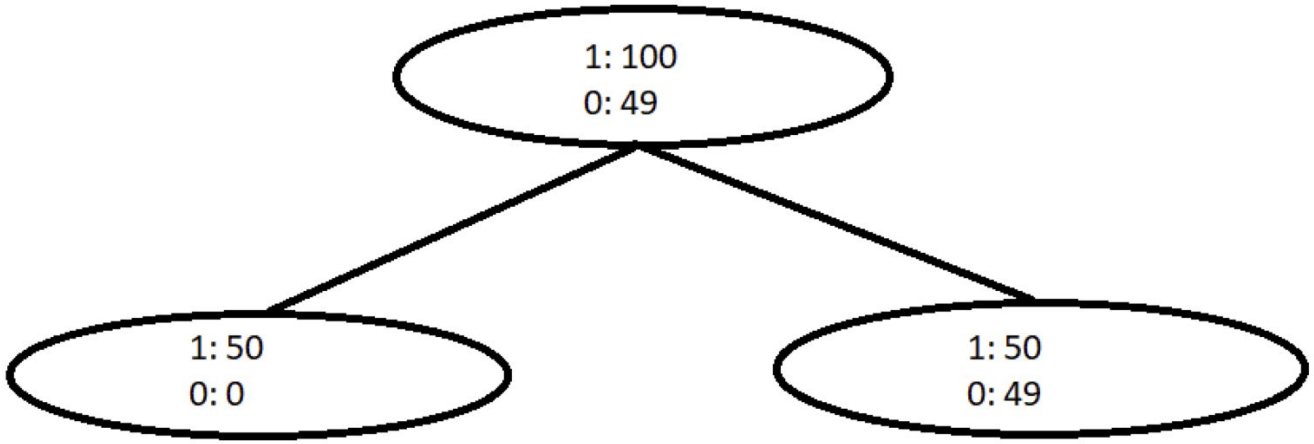
$$H(T) = -\sum_{i=1}^{k} p_i \cdot \log_2(p_i)$$

Why not accuracy?

- Why don’t we use accuracy to guide the choice of the algorithm?

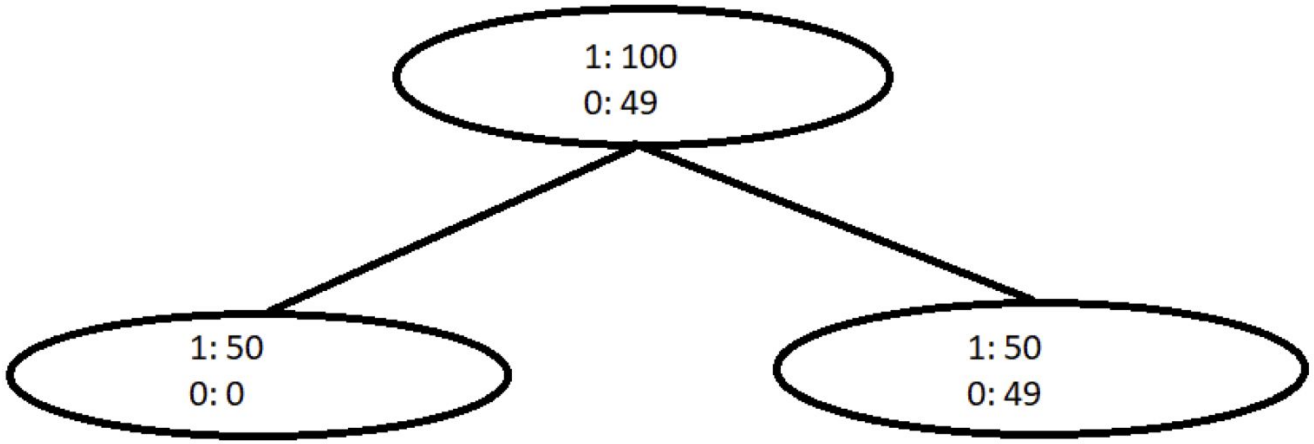
Why not accuracy?

- Accuracy would not give good information… Cosider this example with two classes:



Why not accuracy?

- Accuracy would not give good information… Cosider this example with two classes:



Before split:
Accuracy: **67%**
Gini: **0.441**

After split:
Accuracy: **67%**
Gini: **0.332**

Termination condition

- Obviously the recursive procedure stops when H(T)=0
- In order to reduce the overfitting, often an addional condition is added on the size of the training set:

$$\text{TerminationCondition}(T) = \begin{cases} H(T) = 0 \text{ OR} \\ |T| < MinSize \end{cases}$$

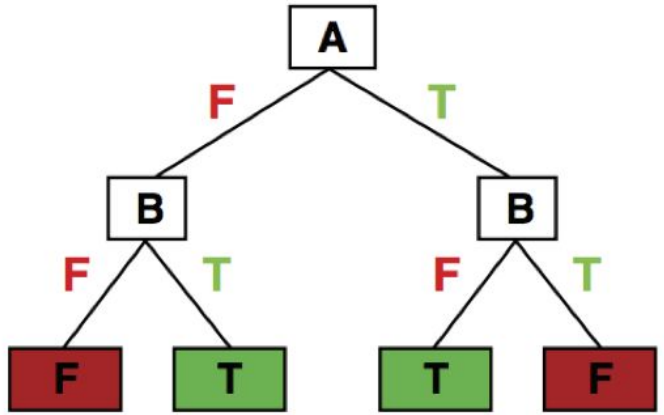
Optimality

- How many decision trees are possible with *N* two-valued features and two classes?

Optimality

- How many decision trees are possible with *N* two-valued features and two classes?
  - You can construct at least a decision tree for each possible Boolean function with *N* inputs

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Optimality

- How many decision trees are possible with *N* two-valued features and two classes?
  - You can construct at least a decision tree for each possible Boolean function with *N* inputs
  - You have at least: 2^{2^N} trees!
- The hypothesis space is huge (for N=6 you have 1.8E+19 possible trees!)
- Of course the learning algorithm cannot explore exhaustively the hypothesis space…

Optimality

- The learning algorithm is greedy
- No guarantees to find the optimal tree in any sense (e.g. the smallest tree covering the training set)
- Some algorithms (e.g. C4.5) perform an additional phase (pruning) removing nodes that give a small contribution to the overall accuracy (a sort of local optimization step, that also helps reducing overfitting)

Cost functions

Let N_m be the number of observations falling in region R_m.
For **regression**:

$$\frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2 \quad \text{, where} \quad \hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

For **classification**:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

is the proportion of training observations in region m that are from class k. Different measures of node impurity include:

- **Misclassification error**: $1 - \max_k \hat{p}_{mk}$
- **Gini index**: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- **Cross-entropy or deviance** $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Predictions using a decision-tree

What value will each leaf predict?

- For **regression**: the average of the training observations falling in the region R_m of the leaf m.
- For **classification**: the most occurring class in the region R_m of the leaf m.

Pruning a tree

Idea: build a large tree T_0 and then prune it back to a subtree T.
This is done defining a cost complexity criterion:

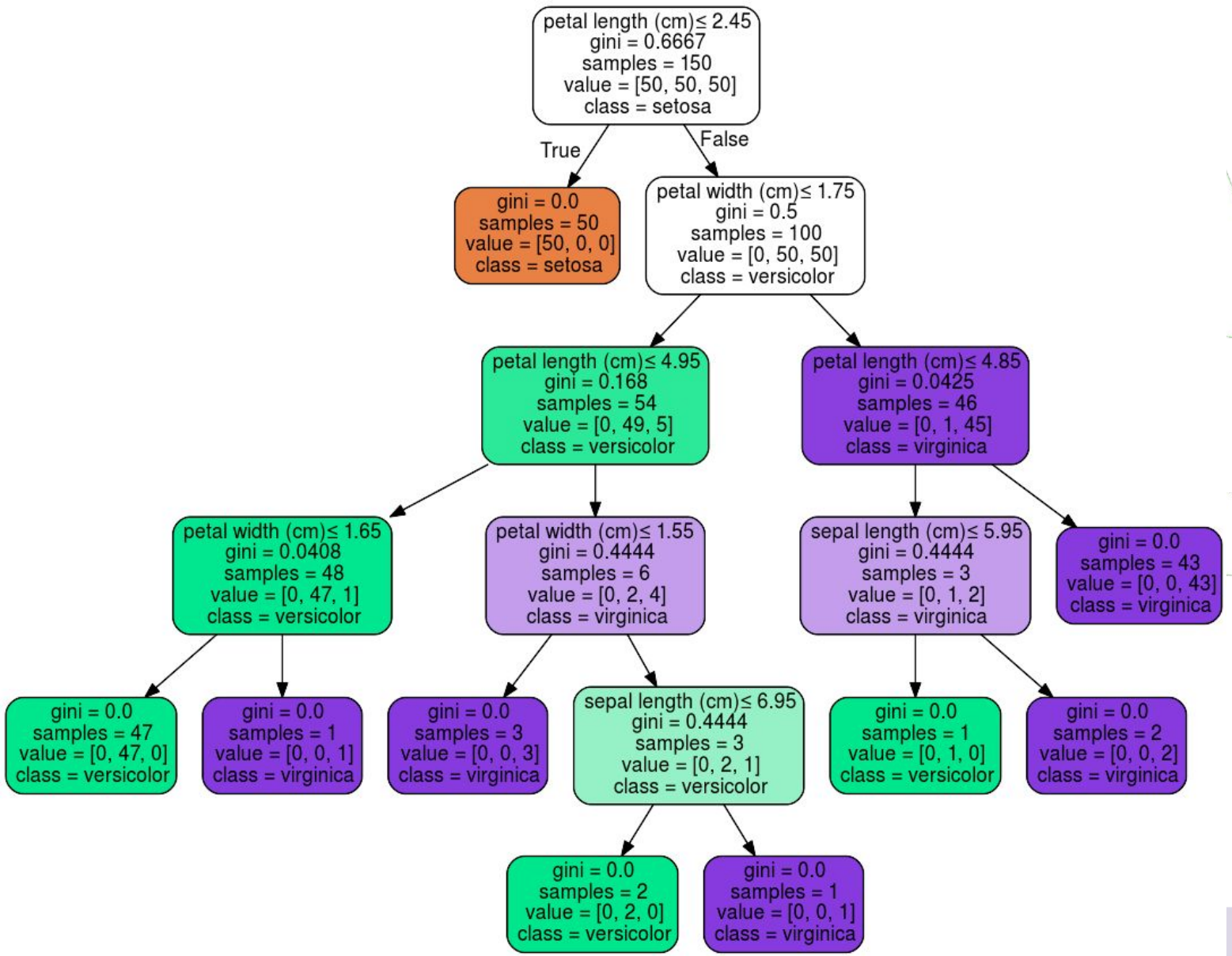
$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2 + \alpha |T|$$

where:

- $|T|$ is the number of leaves in a subtree T
- $\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
- α is a regularisation parameter (tuned with cross-validation)

Interpretability of the classifier

- Example: for the Ir



Pros and cons

- Tree-based methods are simple and easily interpretable (appealing for clinical decision making process)
- They often suffer of low prediction accuracy and overfitting
- Solutions: combine different trees to derive a consensus predictions (we will discuss *bagging*, *random forests*, *boosting*)
- Combining a large number of trees can improve predictions at the price of loosing a bit interpretability

Bagging (or bootstrap aggregation)

General concept: the average of N observations with variance σ^2

gives an observation with variance σ^2/N

Idea: Instead of pruning big trees, build multiple independent big trees and average their predictions.

How to build independent trees with only one dataset? With **bootstrap**.

How to do bagging

- Generate *B* different training datasets by bootstrapping (sampling with replacement).
- Build a decision-tree for each bootstrapped dataset (without pruning).
- Obtain the final predictions by averaging (regression) or majority vote (classification).

Bagging reduces the variance without increasing the bias.

Out of bag error estimation

How to estimate the test error of a bagged model?

Idea: on average, each bagged tree makes use of about 2/3 of the observations. We can use the remaining 1/3, called out-of-bag observations (OOB), to estimate prediction error.

For each observation *i* we consider all the trees in which the observation was OOB. This yields to about B/3 predictions for that observation that can be averaged.

Variable importance measure

Problem: we reduce variance but at the price of losing interpretability.

Idea: Obtain a measure of the importance of each predictor looking at the how much they decrease the cost function (RSS for regression, Gini index for classification) in average across the B trees.

Can be used to **select variables**!

Random Forests

- A multi-classifier approach can be used to overcome this problem: combining many (possibly independent) decision trees, a more robust classifier is obtained (a Random Forest)

- A simple combination rule is used (usually majority voting)

Random Forests

- In a Random Forests a high number of tree classifiers (e.g. 100 or more) is trained using the same algorithm (e.g. CART)

- The classifiers are made (more) independent from each other by adding *randomness* during the training in two aspects:
  - The choice of the training samples
  - The choice of the features

- This is an example of the *Bagging* (Boostrap Aggregating) meta-algorithm

Training samples

- Each tree classifier is trained using only a subset of the training set, obtained by random sampling
  - Usually 2/3 of the full training set
  - Sampling *with replacement*: a sample may be duplicated (thus even if the number of samples is the same as the full training set, the sets are not the equal)

- In this way all the classifiers see densely populated regions of the input space, but outlier samples are less likely to be seen by the majority of the classifiers

Training features

- Each tree classifier is trained using only a subset of the features, also obtained by random sampling
  - Usually:

$$\#used\_features = \sqrt{\#features}$$

- Better independence of classifiers

- Reduction of training computational cost

Boosting

With **boosting** the trees are grown sequentially.

- Instead of building a lot of large trees, with boosting we sequentially build small trees (with d splits).
- Each tree fits a shrunken version of the residuals of the previous tree, compensating partially the bias of the previous tree. The shrinkage factor is called λ
- The higher B (i.e. the number of trees), the smaller the bias and the higher the variance

Choice of hyper-parameters:

- B: cross-validation
- λ: typically 0.01 or 0.001 (note: small λ will require large B)
- d : typically 1.

Applicability of Random Forests

- Usually work “out-of-the-box”: even if they have hyperparameters, they give pretty good results with the default values

Summary

- Decision-trees are simple and interpretable but suffer from poor predictions.
- Combining multiple trees allows improving predictions at the price of loosing interpretability.
- Random forests and boosting are state-of-the-art models for supervised learning.

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani