

# Classification: Support vector machines and random forests

Federica Eduati

Eindhoven University of Technology  
Department of Biomedical Engineering

2025

# Learning goals

At the end of this lecture you will:

- ▶ Explain the formulation support vector machines (SVM) for classification problems.
- ▶ Explain the formulation of tree based methods for classification (and regression) problems.
- ▶ Illustrate the application of SVM and tree based methods to some case studies and interpret the results.

Materials:

- ▶ Chapters 9, 12 and 15 from **elements**

# 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.

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = x^T \beta + \beta_0 = 0 \quad (1)$$

Where:

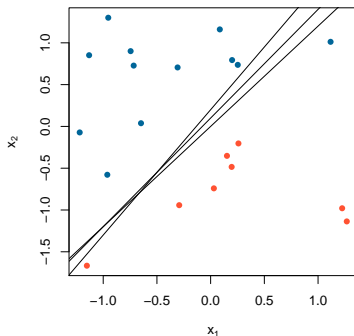
- ▶  $\beta_0 = 0$  only if the hyperplane goes through the origin
- ▶ the vector  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  is a unit vector ( $\|\beta\| = 1$ ) orthogonal to the surface of the hyperplane

# Maximal margin classifier

Imagine to have a training data of  $N$  pairs:

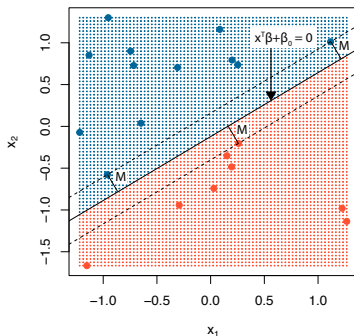
$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , with  $x_i \in \mathbb{R}^p$  and  $y_i \in \{-1, 1\}$ .

If the classes are perfectly separable, there are generally multiple hyperplanes that can separate them.



# Maximal margin classifier

The *maximal margin classifier* is the one with biggest margin between the two classes.

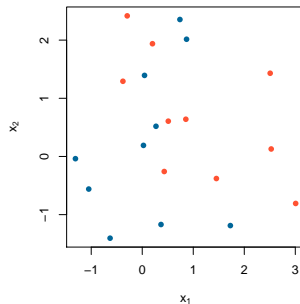
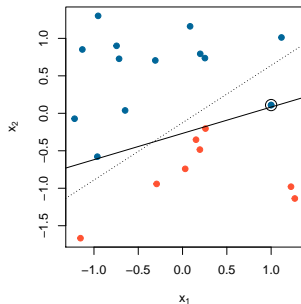


$$\max_{\beta, \beta_0, \|\beta\|=1} M, \text{ subject to } y_i(x_i^T \beta + \beta_0) \geq M, i = 1, \dots, N$$

# 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).

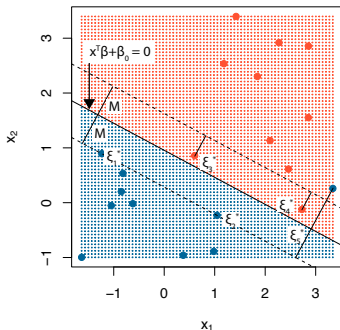
For this we can modify the optimization problem allowing some slack.

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

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

The *support vector classifier* provides a solution by maximising a *soft margin*.



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



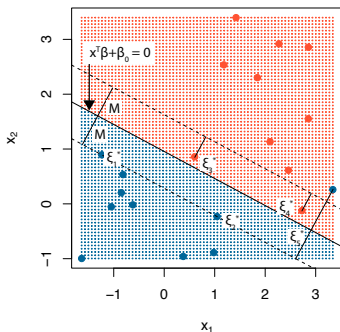
# Slack variables

The slack variables  $\xi = (\xi_1, \xi_2, \dots, \xi_N)$  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

# Mentimeter question ([www.menti.com](http://www.menti.com) code 6123 3708)

How many points are on the wrong side of the hyperplane?



# Regularization

The constant  $C$  (slack budget) is tunable and can be seen as a regularization parameter.

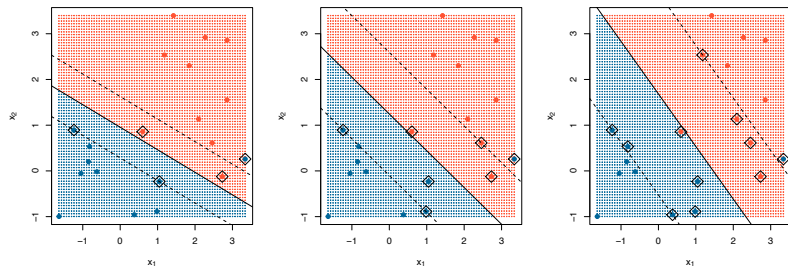
- ▶  $C = 0$  no budget for violation of the margin (maximum margin classifier)
- ▶ increasing  $C$  allows more slack allowed (wider margins)

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

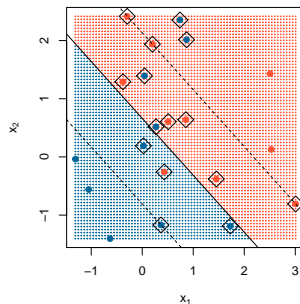
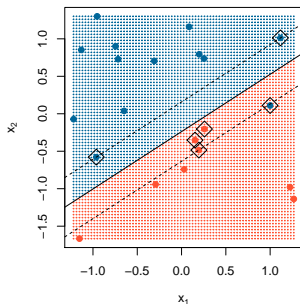
Example of support vector classifier for increasing values of  $C$ .



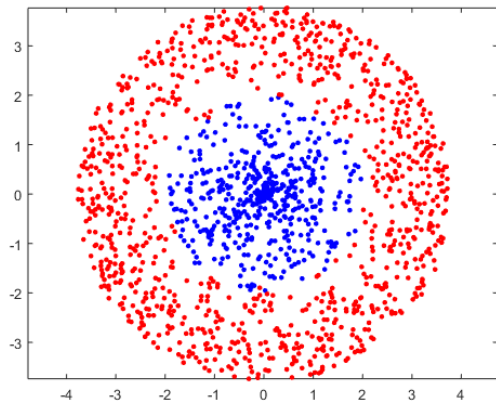
The *support points* (marked with diamonds) are the only ones that determine the orientation of the margin.

# 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



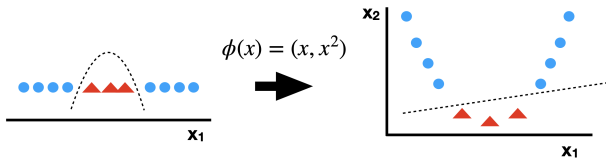
# Classification with non-linear decision boundaries

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:

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

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

Example with one dimension:



# Support vector machines

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 Machines

Inner product definition:  $\langle a, b \rangle = \sum_{i=1}^p a_i b_i$ , where  $a, b$  are vectors

The solution to the support vector classifier problem involves only the inner products of the observations. The inner product of two observations  $x_i, x'_i$  is given by

$$\langle x_i, x'_i \rangle = \sum_{j=1}^p x_{ij} x'_{ij}$$

Then, the linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^N \hat{\alpha}_i \langle x, x_i \rangle$$

where there are  $N$  parameters  $\hat{\alpha}_i$ , one per training observation.

# Support Vector Machines

$\hat{\alpha}_i$  is nonzero only for the support vectors.

So if  $S$  is the collection of indices of these support points, we can rewrite  $f(x)$  which involves far fewer terms than before:

$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle,$$

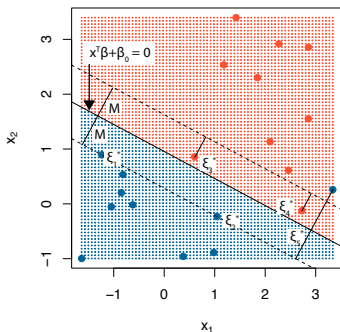
Abstraction of the inner product:

$$K(x_i, x'_i),$$

where we refer to  $K$  as *kernel*. A *kernel* is a function that quantifies the similarity of two observations.

# Mentimeter question ([www.menti.com](http://www.menti.com) code 6123 3708)

How many  $\hat{\alpha}$  do we have to estimate in this case?



# Support Vector Machines

Linear kernel:  $K(x_i, x'_i) = \sum_{j=1}^p x_{ij} x'_{ij}$

Polynomial kernel of degree  $d$ :  $K(x_i, x'_i) = (1 + \sum_{j=1}^p x_{ij} x'_{ij})^d$

Radial kernel:  $K(x_i, x'_i) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x'_{ij})^2)$

What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features?

- ▶ Computational advantage: we don't work in the enlarged feature space.
- ▶ Automatically computes inner product for high dimensional space of features.
- ▶ Avoid overfitting by automatically squashing down most dimensions.

## 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.

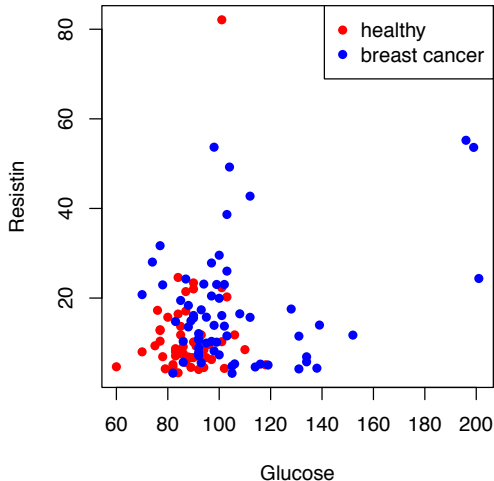
Two approaches for extending SVMs to  $K > 2$  classes classification:

- ▶ 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.

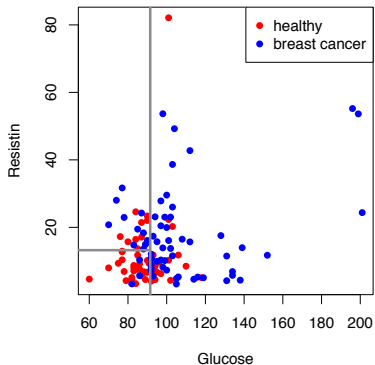
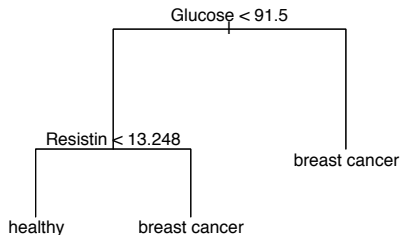


## Example

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



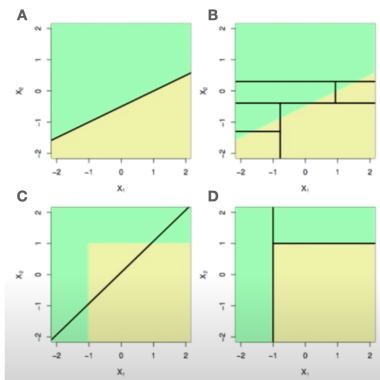
# Example





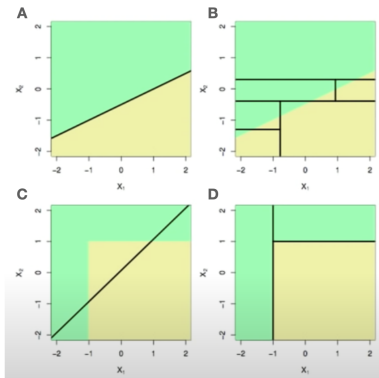
# Mentimeter question ([www.menti.com](http://www.menti.com) code 6123 3708)

For which ones the decision boundary was defined using a decision tree?



# Mentimeter question ([www.menti.com](http://www.menti.com) code 6123 3708)

Which scenarios are suitable for decision trees?



# How to build a decision-tree

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

The algorithm should build the decision tree that:

- ▶ divides the predictor space in  $M$  non-overlapping regions  $R_m$  with  $m = 1, \dots, M$  ( $M$  number of leaves). For each observation that falls in the same region  $R_m$  we make the same prediction.
- ▶ this partition should minimise the  $RSS$  (for regression) or the misclassification (for classification).

# How to build a decision-tree

It is infeasible to evaluate every possible partition of the feature space.

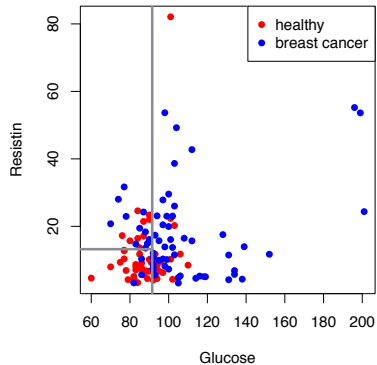
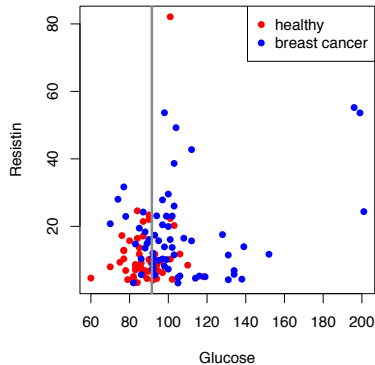
We adopt an approach that is:

- ▶ *top-down*: starts from the top of the tree and proceeds with subsequent splits.
- ▶ *greedy*: it looks at the optimal splitting at that specific step of the tree, without looking ahead.

# How to build a decision-tree

- ▶ We start from the top of the tree and we select the variable  $X_j$  and the splitting point  $s$  to define the pair of half-planes  $R_1(j, s) = \{X | X_j \leq s\}$  and  $R_2(j, s) = \{X | X_j > s\}$  that leads to the greatest reduction of the cost function.
- ▶ This will generate two nodes, for each of the node we repeat the procedure but looking only at the data in that half-plane.
- ▶ This continue until we reach a termination criterion (e.g. no region with more than 5 observations).

# Example



# 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**  $-\sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}$

# Mentimeter question ([www.menti.com](https://www.menti.com/join/61233708) code 6123 3708)

What happens for a **regression tree** if there is no termination criterion?

- ▶ The tree goes to infinity
- ▶ Each leaf has one observation only
- ▶ It is not possible to tell without seeing the data



# Mentimeter question ([www.menti.com](https://www.menti.com/join/61233708) code 6123 3708)

What happens for a **classification tree** if there is no termination criterion?

- ▶ The tree goes to infinity
- ▶ Each leaf has one observation only
- ▶ It is not possible to tell without seeing the data

# 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  $m$ .

# Mentimeter question ([www.menti.com](https://www.menti.com/join/61233708) code 6123 3708)

What is the risk of a tree with a lot of leaves?

# 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  $T$
- ▶  $\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
- ▶  $\alpha$  is a regularisation parameter (tuned with cross-validation)

# Pros and cons

- ▶ Tree-based methods are simple and easily interpretable (appealing for clinical decision making process)
- ▶ They often suffer of low prediction accuracy
- ▶ 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 losing a bit interpretability

# Bagging (or bootstrap aggregation)

General concept: the average of  $N$  observations with variance  $\sigma^2$  gives an observation with variance  $\sigma^2/N$

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.

# Random forest

How can we improve performance over bagging? Performing random subselections of the features.

This decorrelates the trees and reduces variance.

## Random forests:

- ▶ build a large number of decision-trees using bootstrapped training data (same as bagging)
- ▶ at each split select a subset of  $m$  features out of the  $p$  as possible split candidate. A typical choice of  $m$  is  $m \simeq \sqrt{p}$ .

# 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 shrunk version of the residuals of the previous tree, compensating partially the bias of the previous tree. The shrinkage factor is called  $\lambda$
- ▶ 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
- ▶  $\lambda$ : typically 0.01 or 0.001 (note: small  $\lambda$  will require large  $B$ )
- ▶  $d$ : typically 1.

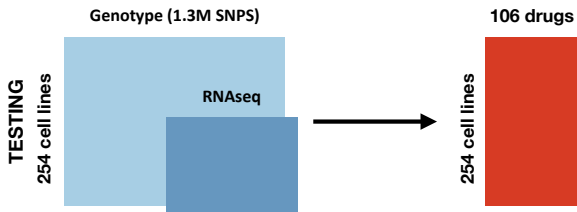
# Summary

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

# Case study: prediction of human population response

Open DREAM challenge (<http://dreamchallenges.org/>) with 213 participants.

- ▶ Subchallenge 1: predict cytotoxicity of new cell lines based on the genotype.
- ▶ Subchallenge 2: predict cytotoxicity of new compounds based on their chemical attributes.



Eduati et al., Nature Biotechnology, 2015

For both subchallenges the best performing methods were based on random forests.