Support vector machines and random forests

Federica Eduati

Eindhoven University of Technology
Department of Biomedical Engineering

2019

Learning goals

At the end of this lecture you will:

- Have a general understanding of support vector machines for classification.
- Have a general understanding of machine learning methods based on decision trees (including random forests).

Materials:

- Chapters 9, 12 and 15 from Friedman et al., *The Elements of Statistical Learning*

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.

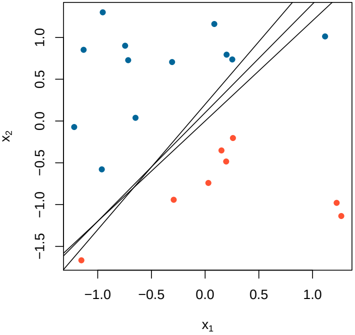
$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = x^T \beta + \beta_0 = 0 \quad (1)$$

Where:

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

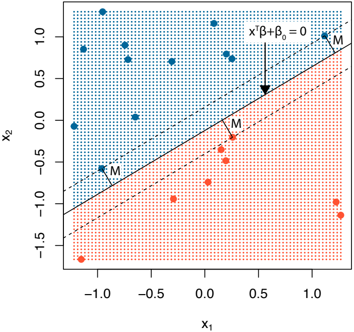
Maximal margin classifier

Imagine to have a training data of N pairs:
$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, 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.

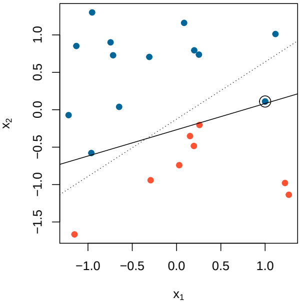


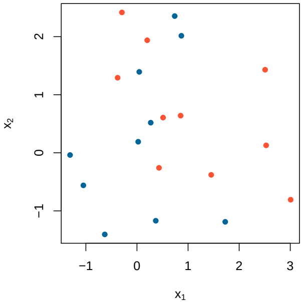
$$\max_{\beta,\beta_0,\|\beta\|=1} M, \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq M, i = 1, \ldots, N$$

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).

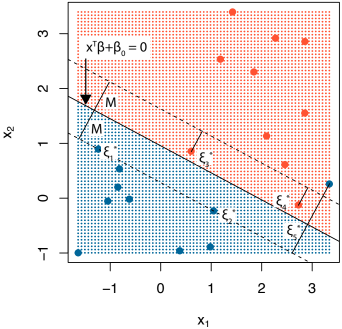
For this we can modify the optimization problem allowing some slack.

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

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

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



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

Support vector classifier: 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 ith observation is on the correct side of the margin
- $\xi > 0$ when the ith observation is on the wrong side of the margin
- $\xi > 1$ when the ith observation is on the wrong side of the hyperplane

Support vector classifier: regularization

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

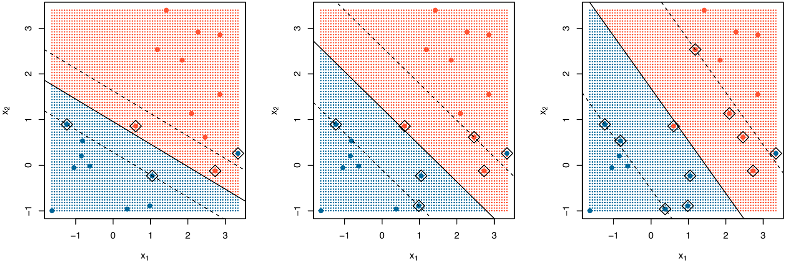
- $C = 0$ no budget for violation of the margin (maximum margin classifier)
- increasing C allows more slack allowed (wider margins)

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

Support vector classifier: example

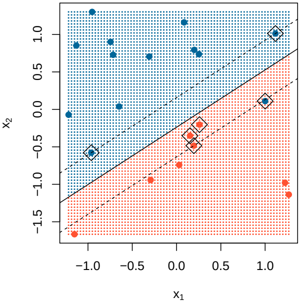
Example of support vector classifier for increasing values of C.

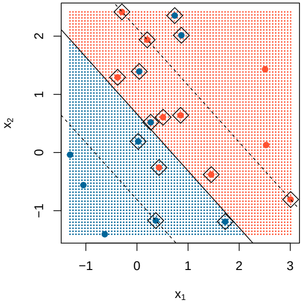


The *support points* (marked with diamonds), i.e. those with $\xi_i \neq 0$, are the only ones that determine the orientation of the margin.

Support vector classifier: 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.





Support Vector Machines: Classification with non-linear decision boundaries

Extension of the Support vector classifier to handle **non-linear class boundaries**.

- Linear regression: use of quadratic and cubic terms.
- Support vector classifier: use of quadratic, cubic, and even higher-order polynomial functions of the predictors.

For instance, from p features to 2p features:

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

Then, the optimization problem would become:

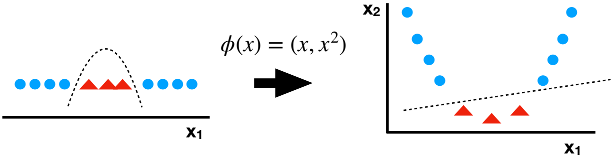
$$\max_{\beta, \beta_0, \|\beta\|=1} M, \text{ subject to } y_i({x_i}^T\beta + {x_i^2}^T\beta + \beta_0) \geq M(1 - \xi_i), i = 1, \ldots, N$$

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

Support Vector Machines: The Support Vector Machine

Why does this lead to a non-linear decision boundary?

- Enlarged feature space: linear decision boundary
- Original feature space: non-linear (quadratic polynomial)



Many possible ways to enlarge the feature space.

- Be careful with large number of features (computationally demanding).

The *support vector machine*(SVM): extension of the support vector classifier which enlarges the feature space by using **kernels**. Here, the kernel approach is a efficient computational methodology to enlarge the feature space.

Support Vector Machines: The Support Vector Machine

Inner product definition:$\langle a, b\rangle = \sum_{i=1}^{r} a_i b_i,.$

Observation: 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

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

The linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle, i = 1, \ldots, N$$

where there are n parameters α_i, one per training observation.

Support Vector Machines: The Support Vector Machine

α_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:

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

Generalization of the inner product of the form:

$$K(x_i, x_i'),$$

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

Support Vector Machines: The Support Vector Machine

Linear kernel: $K(x_i, x_i') = \sum_{j=1}^{p} x_{i,j} x_{i,j}'$
Polynomial kernel of degree d: $K(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{i,j} x_{i,j}')^d$
Radial kernel: $K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p} (x_{i,j} - x_{i,j}')^2)$

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.

Extension to multi-class

So far, binary classification, in other words, two-class setting.

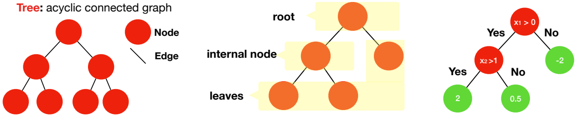
SVMs: concept of separating hyperplanes does not lend itself to more than two classes.

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.

Decision tree

What does a decision tree represent?



Decision tree: binary tree, graphical representation of a function.
Root & internal nodes: splitting condition
Leaves: predictions

How do we know what the optimal splitting point is at each node?

Objective function: maximize the **Information Gain (IG)**

$$IG(D_p, f) = I(D_p) - \left(\frac{N_{left}}{N_p} I(D_{left}) + \frac{N_{right}}{N_p} I(D_{right})\right)$$

The lower the impurity of the child nodes, the larger the information gain.

Classification Trees

Classification problem: a class vote for each tree, and then classifies using majority vote.

As impurity metric or splitting criteria:

$$\textbf{Entropy}: E(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t)$$

$$\textbf{Gini impurity}: i(t) = 1 - \sum_{i=1}^{c} p^2(i|t)$$

where $p(i|t)$ is the proportion of the samples that belong to class c for node t.

Regression Trees

Regression problem: the predictions from each tree at a target point x are simply averaged.

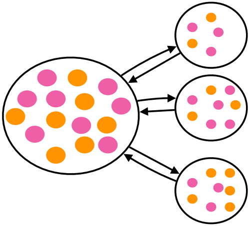
As impurity metric or splitting criteria:

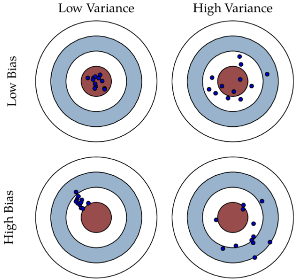
$$\textbf{Mean square error}: MSE(t) = -\frac{1}{N_t}\sum_{i \epsilon D_t}(y^{(i)} - \hat{y}^t)^2$$

where N_t is the number of training samples at node t, D_t is the training subset at node t, $y(i)$ is the true target value, and $\hat{y}^t$ is the predicted target value

Bagging or bootstrap aggregation

Decision trees can become much more powerful when used as ensembles.





- The samples are drawn with replacement.
- High-variance, low-bias procedures, such as trees.

- *From Understanding the Bias-Variance tradeoff, by Scott Fortmann Roe.*

- The entire forest will have lower variance but not at the cost of increasing the bias.

Random Forests: motivation

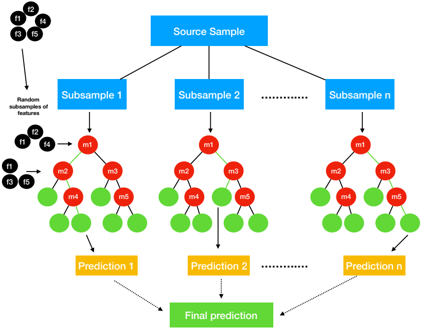
A way of bagging decision trees: combining the predictions of n different models, each of which having profound different insights into the relationships of the data.

Trees are appropiate for bagging: high-variance, low-bias procedures.

Random Forests builds a large collection of de-correlated trees, and then averages them. It brings in the insights from each of them. So this idea is called **Ensembling**.

Ensemblubg is a machine learning technique, both for regression and classification tasks. It can also be used for feature selection.

Random Forests: algorithm



Workflow:

- **Bootstrap sampling**: to grow the tree, a random subsample of the total dataset is used.
- **Model building**: a random subset of all features is chosen as a "splitter variable".
- **Bootstrap aggregating**

Details:

- **Out of Bag Samples**
- **Variable Importance**
- **Overfitting**

Random Forests are popular

Advantages:

- **Small sample size**
- **High-dimensional feature space**
- **Complex data structures**

Applications:

- Predicting drug responses for cancer cell lines [Riddick et al., Bioinformatics, 2010.]

Genomic characterizations $\rightarrow$ Large number of features
$\rightarrow$ RF utilize the top features based on bootstrap aggregation
$\rightarrow$ Good performance.