Chapter 9:
Decision Trees & Random Forests

In this chapter we will consider tree based learning. This is decision trees and based on that bagging and random forests. Decision tree learning is one of the oldest non-linear machine learning methods. Moreover, in [1] M. Fernandez-Delgado et al. evaluated 179 classifiers arising from 17 families (neural networks, support vector machines, nearest-neighbors, random forests, ...). They found that the classifiers most likely to be the best are random forests and SVM with gaussian kernel. That means if you have a standard data set with some numerical features, then it is very likely that random forests or SVM with gaussian kernels will be the best method for this data set. This result does not hold for image classification for example, where you need of course a deep neural network.

Here we only regard numerical datasets. Those are given by d-dimensional datapoints $\mathbf{x} \in \mathbb{R}^d$ each equipped with a label $y \in \mathbb{R}$ (for Regression problems) or $yin\{-1, +1\}$ (for Classification problems). As before we have some training data with that we construct our decision tree or random forest. Those tools shall predict labels for new datapoints. So let us start.

9.1 Decision Trees

Definition: Decision Trees
*A **decision tree** is a decision support tool from operations research that uses a tree-like graph or model of decisions and their possible consequences.*

A non-ML example for such a decision supporting tree can be found in Figure 1. An example for ML can be found in Figure 2. Here we have 2-dimensional datapoints $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$. For each given datapoint we follow the tree down to a terminal node and take this label as predicted label for $\mathbf{x}$. The induced decision region of this decision tree are shown in Figure 3. To each given datapoint (x_1, x_2) we assign the appropriate label from point (x_1, x_2) in the figure. We see that the first node '$x_1 > 2.52$' splits according to the x_1-coordinate at 2.52. To the left region $+1$ is assigned and if we are in the right region the decision tree performs another check, the node $x_2 > 5.83$. Note that such a visualisation can be only done for 2-dimensional data.
So we know how to apply a given tree to new datapoints. In the following we consider the question how to construct such a decision tree that predicts well on a given dataset.

In order to solve this we do the following. Assume we have d-dimensional datapoints. Then we first construct a complete binary tree, that is with depth d

[1] [M. Fernandez-Delgado, E. Cernadas and S. Barro, 2014: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?]

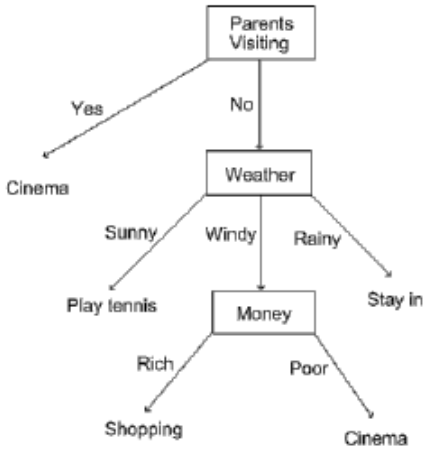

Figure 1: A decision tree for what to do at a normal day.

and 2^d terminal nodes. Afterwards we delete some of the internal nodes, that produce no substantial different. Firstly we explain how to grow the tree, afterwards how to reduce it again using pruning.

For growing the tree three question arises.

1. For each internal node, which feature to take?
2. For each internal node, which threshold to take?
3. Which labels to use at the terminal nodes?

In other words: For a d-dimensional dataset an internal node is of the structure $x_j > t$. We have to specify which feature $j \in \{1, \ldots, d\}$ (Question 1.) and which threshold t (2.) to take. A terminal node has the structure $y = l$ for l a label and we have to choose a label l (3.).

Question 1. and 2. correspond and we answer them together. But first some notation. As before we consider the node '$x_j > t$' with j the feature that we split upon, t the split threshold and $R_1(j,t), R_2(j,t) \subset \mathbb{R}^d$ denotes the two regions under the internal node $x_j > t$. That is $R_1(j,t) = \{\mathbf{y} \in \mathbb{R}^d \mid x_j > y \text{is false}\}$ and $R_2(j,t) = \{\mathbf{y} \in \mathbb{R}^d \mid x_j > y \text{is true}\}$, see also Figure 4. In other words R_1 and R_2 are the two regions induced by splitting the j-th feature using the threshold t, where R_1 corresponds to everything under the left hand subtree and R_2 to everything under the right hand subtree. We look at the questions for the regression and the classification case.

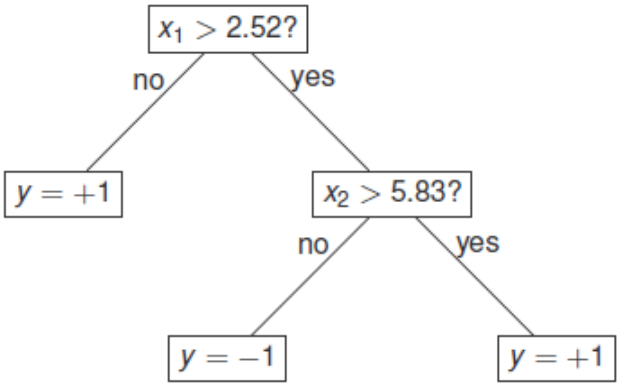

Figure 2: A decision tree for predicting labels of 2-dimensional datapoints $x = (x_1, x_2) \in \mathbb{R}^2$.

For the regression case we have labels $l \in \mathbb{R}$ and we choose the feature j and threshold t by:

$$\arg\min_{j,t} \sum_{k=1}^{2} \min_{C \in \mathbb{R}} \sum_{i:\mathbf{x}^i \in R_k(j,t)} \left(y^i - c\right)^2$$

That means, we choose j and t such that for each region $R_1(j,t)$ and $R_2(j,t)$, respectively, we can find an appropriate value c such that the mean squared error of c and the labels y^i of the points $\mathbf{x}^i$ in $R_1(j,t)$ and $R_2(j,t)$, respectively, is minimal. One can show that in the optimum c is the mean label of points in that region $R_1(j,t)$ or $R_2(j,t)$ respectively. We define $\hat{y}_{j,t,k}$ to be this value. Formalised this means, that for a region $R(j,t)$ the optimal c is given by $c = \text{mean}\left(\{y^i : \mathbf{x}^i \in R_k(j,t)\}\right) =: \hat{y}_{j,t,k}$. So in other words we choose the feature j and treshold t such that the average squared distance of labels of the points under a region to the mean label in that region is minimal.
To assign a label at a terminal node (question 3.) just take the mean label of points in the region under this terminal. So if this terminal node is the left child of the internal node given by j and t we take $\hat{y}_{j,t,1}$ for the label.

For the Classification case we have labels $l \in \{-1, +1\}$ and we want the predicted labels to be in $\{-1, +1\}$, too. Therefore taking the mean label of points does not work anymore. So for this case we use a little bit different strategy.
Define

$$p_{j,t,k} = \frac{|\{i \mid \mathbf{x}^i \in R_k(j,t) \wedge y^i = +1\}|}{|\{i \mid \mathbf{x}^i \in R_k(j,t)\}|}$$

the fraction of instances in region $R_k(j,t)$ that are labeled $+1$. Hence, $1 - p_{j,t,k}$ is the fraction of instances in that region that are labeled -1. Now we choose

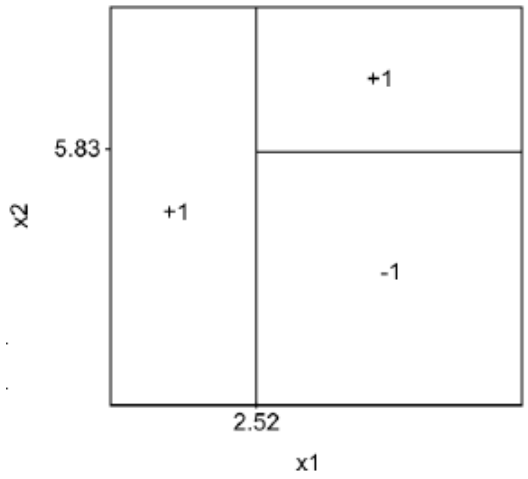

Figure 3: A visualisation of the induced decision regions of the decision tree in Figure 2.

the feature j and threshold t by

$$\arg\min_{j,t} \sum_{k=1}^{2} g\left(p_{j,t,k}\right)$$

where g is the **Gini impurity measure** given by $g(p) := 2p(1-p)$, see also Figure 5. Observe that the minimum of g is attained for very small or large values of p. Therefore j and t are chosen such that $p_{j,t,k}$ is very small or large but preferably not near to 0.5. $p = 0.5$ would mean that we have as many points with positive labels as points with negative labels. So this means our algorithm searchs as much disparity as possible and that is exactly what we want.
To assign a label at a terminal node (question 3.) just do a majority vote over the labels of the instances (that are the points $\mathbf{x} \in \mathbb{R}^d$) falling into the region under the terminal node. If for example in the region there are two points with positive label and one with a negative one, this terminal node gets the label +1.
To reduce the size of the tree we use **pruning**.

Definition: pruning
Remove internal nodes that produce no substantial decrease in prediction accuracy as measured on a hold-out data set.
That is we train the tree on a training data set until it is fully grown, then we start pruning the tree on another data set, called hold-out data set. Thereby we successively remove nodes of the tree that do not decrease the prediction accuracy very much. For further reading see [2] Chapter 9, e.g. **cost-complexity**

[2] [T. Hastie, R. Tibshirani and J. Friedman, The elements of statistical learning, 2nd edition, Springer series, 2009]

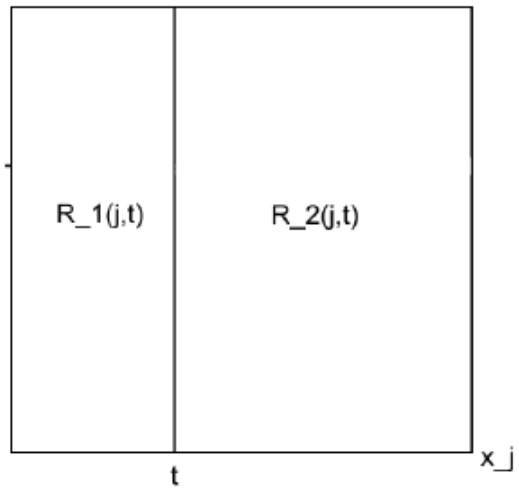

Figure 4: The two regions $R_1(j,t)$ and $R_2(j,t)$ for the internal node given by j and t, that is for the node '$x_j > t$'.

pruning.
We want to mention that pruning is especially important to remedy overfitting. A large, fully grown tree with 2^d terminal nodes is prone to overfit!

9.2 Random Forests

One major problem of the decision trees is their high variance: Often a small change in data can result in a very different series of splits. For example consider a feature that we split very late. If we split this feature at the beginning, we get a completely different tree. This is bad for reproducability but we can use this to gain advantage. The technique **bagging** averages many trees to reduce this variance. That means we repeat our method, decision tree, several times each on a slightly different data set. Then we combine the different results. In detail this means:

1. We draw each data set randomly with replacement from the training data such that it has the same size as the original training set. (i.e. one data point could occure more than once)

2. Grow a tree on each data set.

3. In the case of regression we average the resulting prediction functions and for the classification case we perform majority vote.

The resulting ensemble of trees is called a **forest**.
In principle, this trick can be applied to any arbitrary machine algorithm (e.g.,

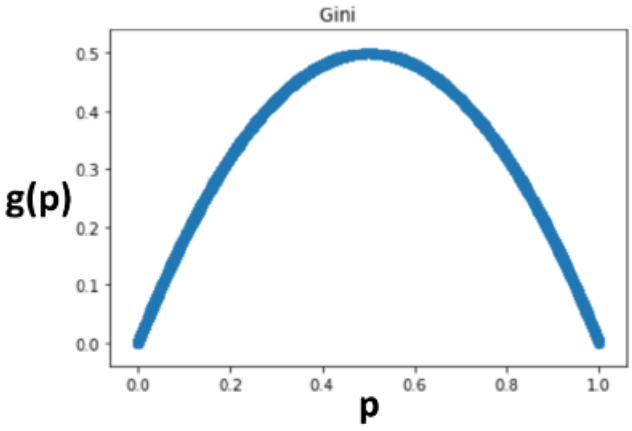

Figure 5: The Gini impurity measure g.

SVM, KRR, neural network), but in practice it works best with (small) decision trees because they have a very high variance.

With this we can define random forests.
Definition: Random Forests
Random forests are pretty much the same as bagging of decision trees except for one difference: When growing a tree of the forest, for each split of an internal node in the tree, we randomly select a subset of $m < d$ many features and choose the optimal split feature j only among those m features.
In other words, for each internal node we use a new random draw of m features, call the set of those features M, and instead of $\arg\min_{j:t}$ we consider $\arg\min_{j \in M,t}$. Typically this subset of features is very small. If for example $d = 1000$ then a common choice for m is 40. Finally, we again look at the spam data and compare the different methods in Figure 6. Gradient Boosting is strongly related to random forests.

So all in all in this chapter we have learned about Decision Trees, which are an interpretable classifier or regression method based on growing a decision tree. Moreover, we have learned about Random Forests, which takes Decision Trees to the next level by applying bagging to grow an ensemble of trees, a forest, but then introducing some randomisation in the splitting process.

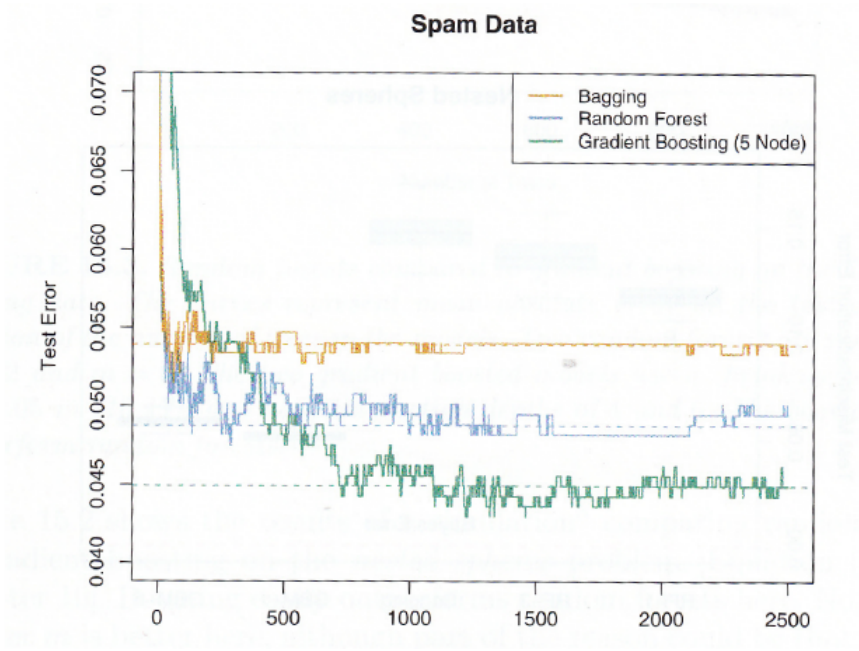

Figure 6: The performance of the algorithms. From up to down: Bagging, Random Forests and Gradient Boosting. The x axis gives the number of trees.