STK2100 - Machine Learning and Statistical Methods for Prediction and Classification

Spring 2026

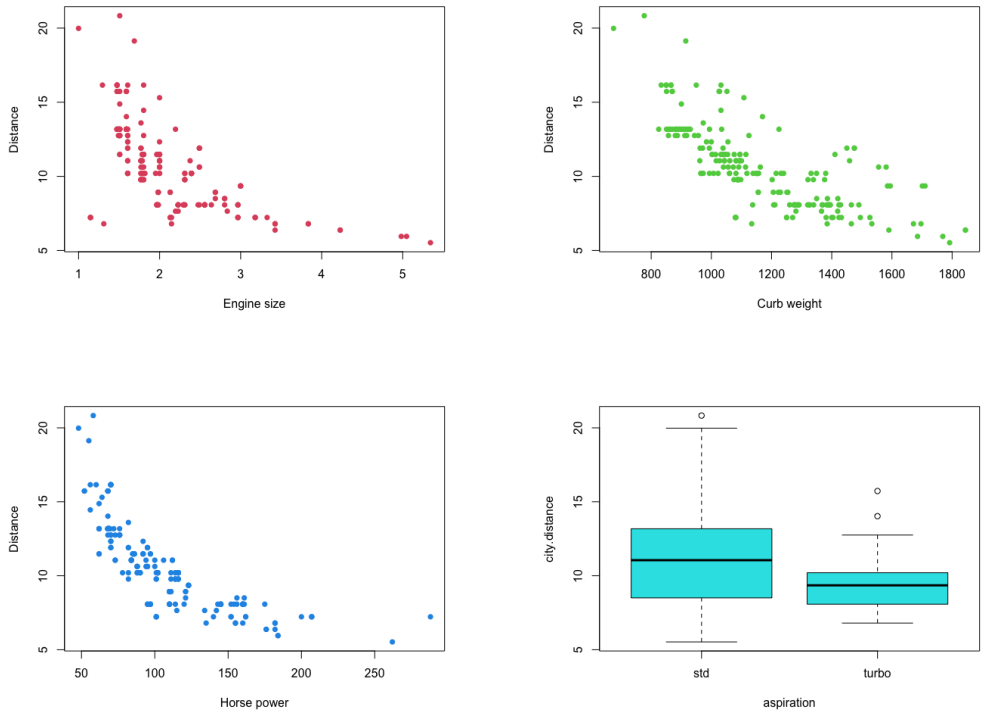
Generalised additive models and tree based methods

Covers Section 9.1 and 9.2

Ingrid Hobæk Haff
Department of Mathematics
University of Oslo

Example: Auto data

- Response: city distance covered (km/l) (`city.distance`)
- Explanatory variables:
  - engine size in l (`engine.size`)
  - curb weight in kg (`curb.weight`)
  - horse power (`HP`)
  - aspiration standard/turbo (`aspiration`)
- We want to predict the distance covered by fuel unit of a car based on properties of the car



Additive models

- Assume that we are in a regression setting with

$$Y = f(\mathbf{x}) + \varepsilon, \quad E[\varepsilon] = 0$$

- Quite often, we need more flexible relationships than linear ones to model $f(\mathbf{x})$
- So far, we used predefined basis functions to achieve nonlinearities, such as splines and kernel smoothers
- Now, we want to obtain more automatic, flexible statistical methods to model non-linear effects
- More specifically, we consider additive models

$$f(\mathbf{x}) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- The $f_j(x_j)$s are typically assumed to be smooth and modelled with splines or kernel smoothers, but could also be linear terms or (global) polynomials in x_j

Fitting criterion for additive models

- We assume the model $Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon$, and have the observations $\{(\mathbf{x}_i, y_i), i = 1, ..., N\}$, where $\mathbf{x}_i \in \mathbb{R}^p$
- The criterion used to estimate the model is then the penalized sum of squares

$$\text{PRSS}(\alpha, f_1, f_2, ..., f_p) = \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j,$$

where $\lambda_j \geq 0$ are tuning parameters

Fitting criterion for additive models

- Based on results from Chapter 5, we know that each function f_j is a cubic spline with knots at each of the unique values of $x_{ij}, i = 1, ..., N$
- However the solution is not unique, and we need further constraints, more specifically

$$\sum_{i=1}^{N} f_j(x_{ij}) = 0, \quad j = 1, \ldots, p$$

- Then, it may be shown that $\hat{\alpha} = \frac{1}{N} \sum_{i=1}^{N} y_i = \bar{y}$
- The fact that we have procedures for fitting one f_j is used to construct a fitting algorithm called the Backfitting algorithm

The Backfitting algorithm for additive models

Backfitting algorithm for additive models

Initialise $\hat{\alpha} = \bar{y}$, $\hat{f}_j = 0, j = 1, ..., p$
while Convergence not reached **do**
  for $j = 1, ..., p$ **do**
    for $i = 1, ..., N$ **do**
      $r_i \leftarrow y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})$
    end for
    Fit $\hat{f}_j = S_j(\{(r_i, x_{ij}\})$
    $\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^{N} \hat{f}_j(x_{ij})$
  end for
end while

- The criterion for convergence is typically that the functions stabilise
- The algorithm can accomodate for different smoothing operators S_j, including local polynomial regression, kernel smoothers or even polynomial fits
- This makes it possible to include simple linear terms and allows for categorical covariates

Degrees of freedom

- For one-dimensional splines, we know that the prediction may be written as $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$, so that the degrees of freedom are given by $tr(\mathbf{S})$
- In additive models, each f_j can be represented by a matrix $\mathbf{S}_j$
- Hence the degrees of freedom of f_j is given by $tr(\mathbf{S}_j) - 1$, where the -1 stems from the constraint $\sum_{i=1}^{N} f_j(x_{ij}) = 0$
- Note that this is a linear model with non-linear basis functions f_j
- Hence, "standard" testing procedures can be used to test the significance of the non-linear terms of the f_js
- The linear term of f_j has 1 degree of freedom
- This means that the non-linear part has $tr(\mathbf{S}_j) - 2$ degrees of freedom

Example: Auto data

See R code

Generalised additive models (GAM)

- Define $\mu(\mathbf{x}) = \mathrm{E}(Y|\mathbf{X} = \mathbf{x})$
- Further, assume that g is a strictly increasing, differentiable function
- Then, the generalised additive model (GAM) with link function g is given by

$$g\left(\mu(\mathbf{x})\right) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- Common link functions are
  - the identity link $g(\mu) = \mu$, which gives the additive model
  - the logit link $g(\mu) = \mathit{logit}(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$, as in the additive logistic regression model
  - the log link $g(\mu) = \log(\mu)$, used in log-additive models for Poisson count data or for Gamma regression

Additive logistic regression model

- Assume that $Y \in \{0, 1\}$
- Now $\mu(\mathbf{x}) = \mathrm{E}(Y|\mathbf{X} = \mathbf{x}) = \Pr(Y = 1|\mathbf{x})$
- The ordinary logistic regression model is given by

$$\mu(\mathbf{x}) = \Pr(Y = 1|\mathbf{x}) = \frac{\exp(\alpha + \sum_{j=1}^{p} \beta_j x_j)}{1 + \exp(\alpha + \sum_{j=1}^{p} \beta_j x_j)}$$

  or equivalently

$$\log\left(\frac{\mu(\mathbf{x})}{1 - \mu(\mathbf{x})}\right) = \alpha + \sum_{j=1}^{p} \beta_j x_j$$

- The additive logistic regression is given by

$$\log\left(\frac{\mu(\mathbf{x})}{1 - \mu(\mathbf{x})}\right) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- It can also be generalised to several classes, i.e. $Y \in \{1, \ldots, K\}$

Fitting additive logistic regression models

- Additive logistic regression models can be fitted with a backfitting algorithm within a Newton-Raphson procedure, given below
- This procedure requires that the f_js accept observation weights, which is the case for most smoothing procedures

Fitting additive logistic regression models

Local scoring algorithm for additive logistic regression models

Initialise $\hat{\alpha} = \log(\bar{y}/(1-\bar{y}))$, $\hat{f}_j = 0, j = 1, ..., p$
for $i = 1, \ldots, N$ **do**
    Define $\hat{\eta}_i = \hat{\alpha} + \sum_{j=1}^{p} \hat{f}_j(x_{ij})$ and $\hat{p}_i = \exp(\hat{\eta}_i)/(1+\exp(\hat{\eta}_i))$
end for
while Convergence not reached **do**
    for $i = 1, \ldots, N$ **do**
        Construct the weights $w_i = \hat{p}_i(1-\hat{p}_i)$
        Construct the working target variable $z_i = \hat{\eta}_i + \frac{y_i - \hat{p}_i}{w_i}$
    end for
    Fit an additive model to the z_is with weights w_i, using a weighted backfitting algorithm, to get new estimates $\hat{\alpha}$, $\hat{f}_j$, $j = 1, \ldots, p$
end while

Example: E-mail spam data

- Response: indicator of whether the e-mail is spam True/False (`y`)
- Explanatory variables: there are 57 covariates, which are 57 of the most commonly occurring words and punctuation marks in the e-mails (`x1`,...`x57`)
- We want to predict whether an e-mail is spam or not based on the frequencies of the 57 commonly used words and punctuation marks

	`george`	`you`	`your`	`hp`	`free`	`hpl`	`!`	`our`
`spam`	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51
`email`	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18

See R code

Regression trees

- Another way of modelling non-linear effects and potentially complex interactions in $f(\mathbf{x})$ is through classificaion and regression trees (CART)
- We will consider regression trees first
- The model is then constructed by dividing the covariate space into separate regions R_m, $m = 1, \ldots, M$, and modelling the repsonse with a constant c_m inside each region
- More specifically, the model is given by

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m I(\mathbf{x} \in R_m)$$

Regression tree in two dimensions

Let us start with an example with a response Y and two numerical covariates x_1 and x_2

The procedure is as follows

- Binary splits are performed in one dimension at a time
- Regions are defined by sequential splits of existing regions by thresholding w.r.t. one of the covariates
- The response in each region is modelled with the average value of the y_is corresponding to $\mathbf{x}_i$s that fall into that region

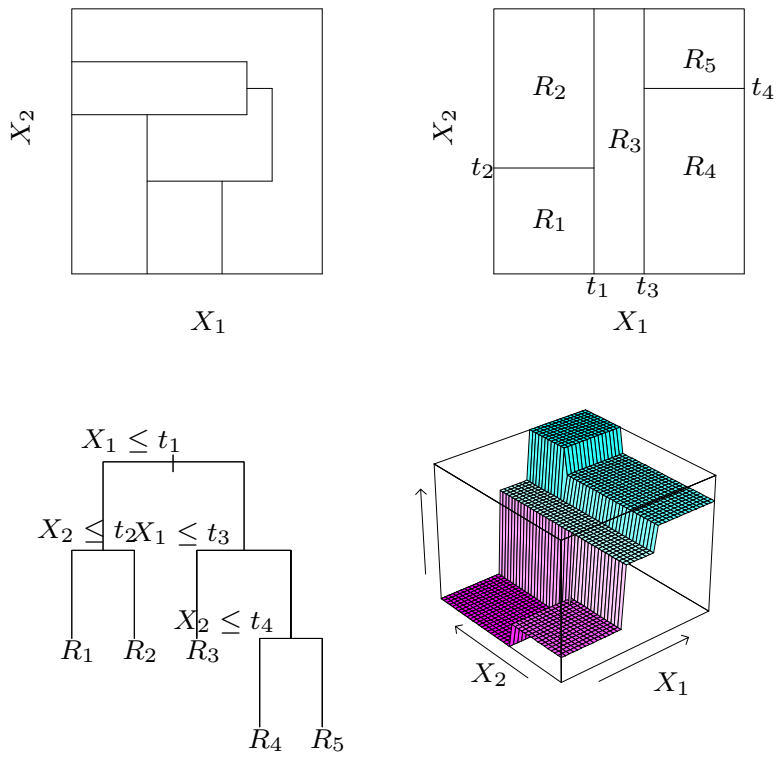
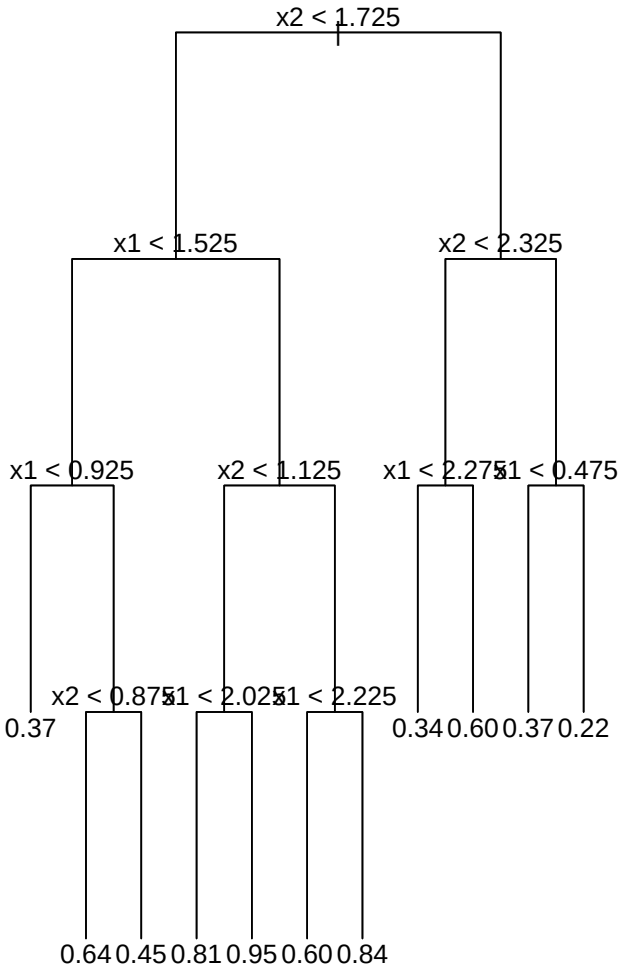
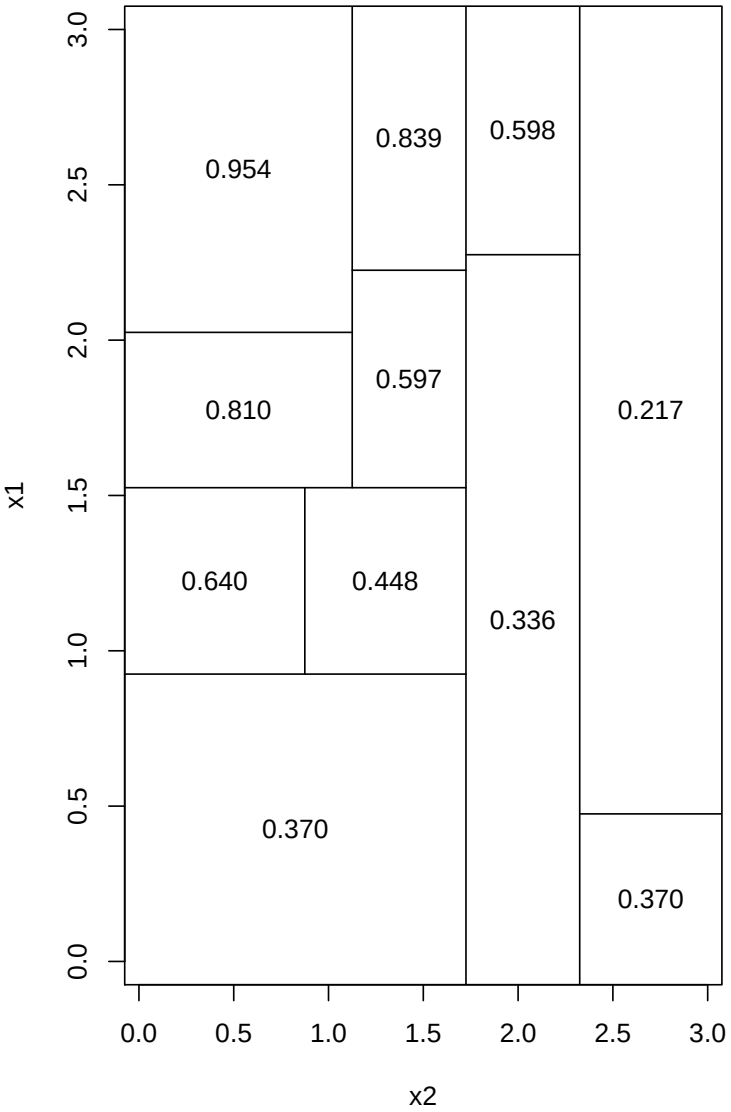


Illustration of a tree structure





Fitting regression trees

- Fitting a regression tree consists in finding the splits that define the regions R_m, and estimating the constants c_m, $m = 1, \ldots, M$
- The objective function that is used in the fitting is

$$
\begin{aligned}
\text{RSS} &= \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 \\
&= \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I(\mathbf{x}_i \in R_m) \right)^2 \\
&= \sum_{m=1}^{M} \sum_{\{i:\mathbf{x}_i \in R_m\}} (y_i - c_m)^2
\end{aligned}
$$

- Searching through all possible combinations of splits is computationally impossible
- Instead, a greedy search is performed

Fitting regression trees

- For all covariates x_j, $j = 1, \ldots, p$, and all possible split points, i.e. unique values of x_j, define the two regions

$$R_1(j, s) = I(x_j \leq s) \quad \text{and} \quad R_2(j, s) = I(x_j > s)$$

and with $N_m = \sum_{i=1}^{N} I(\mathbf{x}_i \in R_m)$, $m = 1, 2$, let

$$\hat{c}_1 = \frac{1}{N_1} \sum_{\{i:\mathbf{x}_i \in R_1\}} y_i \quad \text{and} \quad \hat{c}_2 = \frac{1}{N_2} \sum_{\{i:\mathbf{x}_i \in R_2\}} y_i$$

- Choose the pair (j, s) that minimises

$$\sum_{\{i:\mathbf{x}_i \in R_1\}} (y_i - \hat{c}_1)^2 + \sum_{\{i:\mathbf{x}_i \in R_2\}} (y_i - \hat{c}_2)^2$$

- Next, we partition the data into the two regions R_1 and R_2, and repeat the splitting process
- This is continued until there are $M = N$ regions, or a pre-specified minimum node size has been reached

Pruning of regression trees

- The fitting procedure results in a large tree, that is likely to overfit the data
- A possible strategy is to stop the growing of the tree earlier
- However, it is more advantageous to first grow a large tree, and then prune it
- Let T_0 be the large tree that has been grown, and let $T \subset T_0$ be a sub-tree, obtained by pruning, i.e. collapsing some of the internal nodes of T_0
- Introduce the penalty or cost-complexity

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|, \quad \alpha \geq 0$$

where $N_m = \sum_{i=1}^{N} I(\mathbf{x}_i \in R_m)$ and $Q_m(T)$ is the node impurity given by

$$Q_m(T) = \frac{1}{N_m} \sum_{\{i:\mathbf{x}_i \in R_m\}} (y_i - \hat{c}_m)^2$$

Pruning of regression trees

- The idea is, for each α, to find the sub-tree $T_\alpha \subseteq T_0$ that minimises $C_\alpha(T)$
- It may be shown, that for each α, there is a unique T_α, so that when we increase α, we can find optimal trees by a sequence of pruning operations on the current tree
- To optimise $C_\alpha(T)$, we sequentially remove the leaf which minimises $C_\alpha(T)$, i.e. the leaf which makes the smallest increase in $\sum_{m=1}^{M} N_m Q_m(T)$
- The tuning parameter α must be chosen, for instance using cross-validation

Example: Auto data

See R code

Classification trees

- Assume now that $Y \in \{0, 1, ..., K-1\}$
- Following ideas from regression trees, classification trees model the response with a constant probability $p_{m,k}$ of class k, for $k = 1, \ldots, K$, in each region R_m, i.e.

$$p_k(\mathbf{x}) = \sum_{m=1}^{M} p_{m,k} I(\mathbf{x} \in R_m)$$

- The classification is then obtained as

$$\widehat{Y} = \arg\max_k p_k(\mathbf{x})$$

Fitting classification trees

- Given the regions R_m, the $p_{m,k}$s are estimated by

$$\hat{p}_{m,k} = \frac{1}{N_m} \sum_{\{i:\mathbf{x}_i \in R_m\}} I(y_i = k)$$

- The regions R_m are found in the same way as for regression trees, based on a fitness criterion of the type $\sum_{m=1}^{|T|} N_m Q_m(T)$, with a suitable node impurity $Q_m(T)$

Fitting classification trees

- Impurity measures that are used for classification are
  - the deviance, also called the cross-entropy D:

$$Q_m(T) = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}), \text{ with } D = -2l(\hat{p}_{11}, \ldots, \hat{p}_{K|T|}) = \sum_{m=1}^{|T|} N_m Q_m(T)$$

  - the misclassification error, with $\hat{p}_{mk(m)} = \max_k \hat{p}_{mk}$:

$$Q_m(T) = \frac{1}{N_m} \sum_{\{i:\mathbf{x}_i \in R_m\}} I(y_i \neq \hat{y}_i) = 1 - \hat{p}_{mk(m)}$$

  - the Gini index:

$$Q_m(T) = \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- After a large tree has been grown, pruning can be performed as for regression trees, using the same impurity mesure as for growing the tree

Tree methods - properties

- Advantages of tree methods:
  - Logical simplicity and ease of "communication"
  - Simple, compact mathematical formulation
  - Speed of computation
  - Can easily deal with categorical covariates
- Disadvantages of tree methods:
  - Instable results
  - Difficult to capture additive structure
  - Difficult to do statistical inference
  - Lack of smoothness

Example: E-mail spam data

See R code