Statistical Learning and Prediction

Marjolein Fokkema

m.fokkema@fsw.leidenuniv.nl

Beyond Linearity

Four topics:

- Polynomial regression
- Step functions
- Polynomial regression + step functions = splines:
- Support Vector Classifier
- Support Vector Machine

Polynomial Regression

- The easiest form of (parametric) nonlinearity is to use a polynomial.
- E.g., a cubic (third order) polynomial:

$$y_i = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon_i$$

- More generally, polynomials of order d:

$$y_i = \alpha + \sum_{j=1}^{d} \beta_j x^j + \epsilon_i$$

- Higher order polynomials are very flexible functions (sometimes too flexible)

Polynomial Regression

- Same trick can be applied in logistic regression (and other generalized linear models, e.g., binomial, count responses):

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \alpha + \sum_{j=1}^{d} \beta_j x^j$$

- No interest in the individual coefficients (difficult to interpret, what is large / small effect?)

- Interest in the shape of the association between predictor and response

Step functions

- Define cut points $c_1, c_2, c_3, \ldots, c_K$ and with these functions

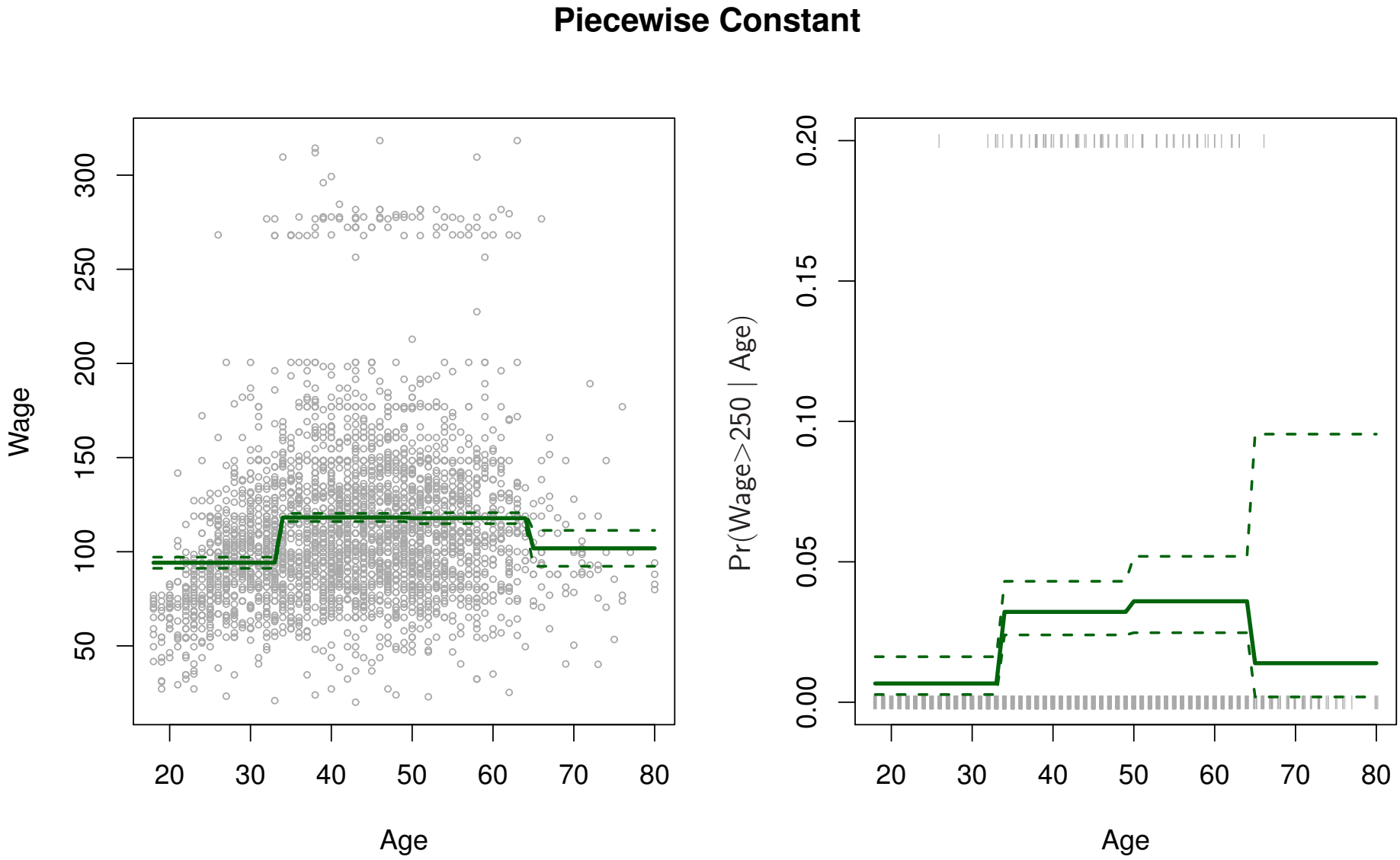
$$\begin{aligned} C_0(X) &= I(X < c_1) \\ C_1(X) &= I(c_1 \leq X < c_2) \\ &\ldots \\ C_{K-1} &= I(c_{K-1} \leq X < c_K) \\ C_K(X) &= I(c_K \leq X) \end{aligned}$$

- Use these $C_k(X)$ as predictors in a linear/logistic regression, e.g.,

$$y_i = \alpha + \sum_{k=1}^{K} \beta_k C_k(X) + \epsilon_i$$

- Step functions can be used to describe regression and classification trees (where they can be a function of multiple predictor variables; see later)

Example - Step function



Basis functions

- Both polynomials and step functions are special cases of more general *basis functions*:
- Define functions/transformations $b_1(X), b_2(X), \ldots, b_K(X)$ of the variable X.
- Use these transformations in a linear regression

$$y_i = \alpha + \sum_{k=1}^{K} \beta_k b_k(X) + \epsilon_i$$

- The functions $b_k(X)$ are fixed and known.

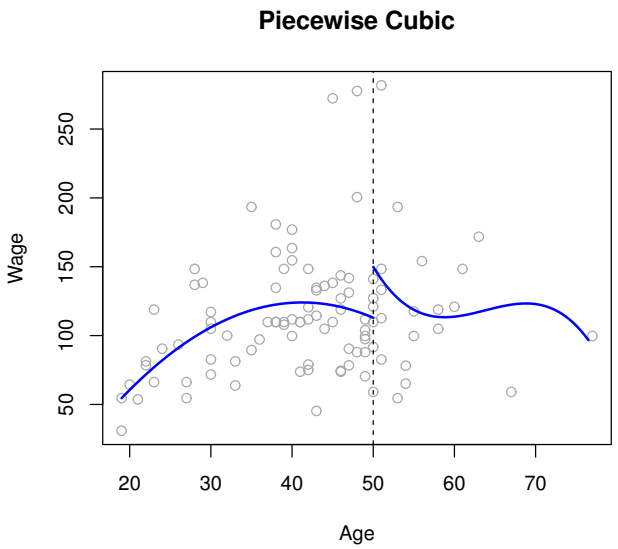
Regression Splines

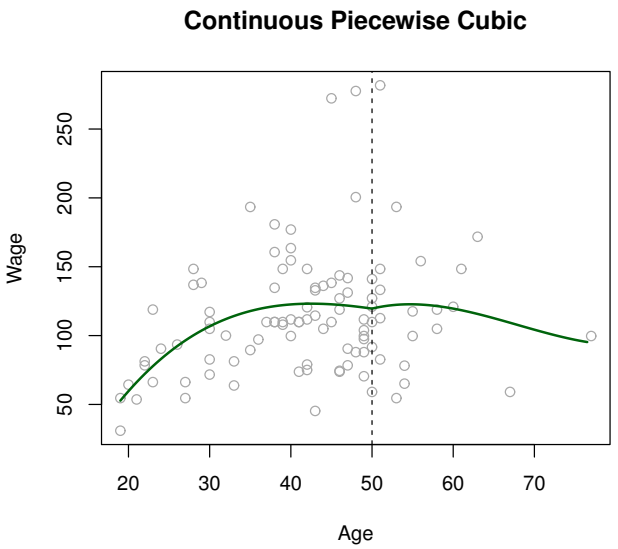
- Combine the step idea with the polynomial functions
- Create a cutpoint c:

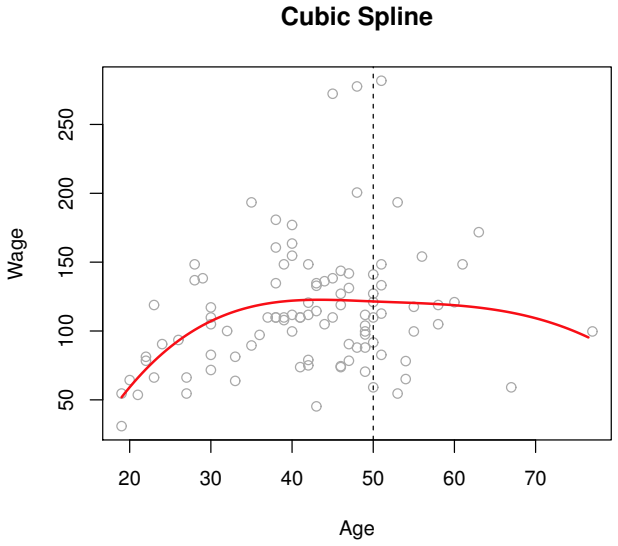
$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 \text{ if } x_i < c \\ \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 \text{ if } x_i \geq c \end{cases}$$

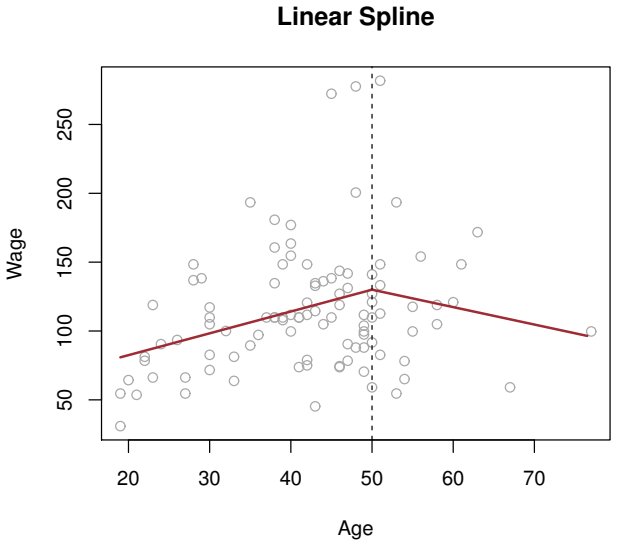
- This is a piecewise polynomial with 1 knot (cutpoint).
- See next slide: upper left. Can give erratic behavior near the boundaries.

Example - Regression Splines









Regression Splines

- At cutpoint c it makes a sudden jump (it is discontinuous).
- We rather have a continuous curve.
- Therefore, we add the constraint that the fitted curve must be continuous (upper right plot).
- Better, but V-shaped join looks unnatural.
- Add two constraints: first- and second-order derivatives of the polynomial curve have to be continuous.
- Thus, we require that the curve is very *smooth*.
- Lower left plot: cubic spline.
- A cubic spline with K knots uses a total of $K + 4$ degrees of freedom.

- Lower right plot: *linear spline* with one knot, $K = 1$ (a.k.a. hinge function).

- General definition: A degree-d spline is a piecewise degree d polynomial with continuity in the derivatives up to degree $d - 1$.

- In the figures we used a single knot, but we can increase the number of knots, giving more flexible curves.

- Note we have again a bias-variance trade-off: the more knots and the higher the order of the polynomial, the less bias, but the more variance.

Regression Splines

- A cubic spline with K knots can be modelled as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

- The basis functions are given by:

$$\begin{aligned} b_1(x) &= x \\ b_2(x) &= x^2 \\ b_3(x) &= x^3 \\ b_{3+j}(x, \xi_j) &= (x - \xi_j)_+^3 = \begin{cases} (x - \xi_j)^3 \text{ if} & x > \xi_j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

  for $j = 1, \ldots, K$

- The function has an intercept and $K + 3$ regression weights, thus using $K + 4$ degrees of freedom.

Regression Splines

- Unfortunately, a cubic spline often still has large variance at the outer ranges of the predictors.

- A *natural spline* is a regression spline with additional boundary contraints: the function is constrained to be linear at the boundaries.

- This uses lower degrees of freedom: quadratic and cubic effects are zero at both boundaries, yielding K degrees of freedom.

Choosing the number of knots

- Where should we place the knots?
  - Prior knowledge / information.
  - Place the knots in a uniform way, for example based on quantiles.

- How many knots should we use? (or equivalently: How many degrees of freedom?)
  - Prior knowledge / information.
  - Determine by cross validation.

Smoothing Spline

- Instead of working with a set of basis functions it is also possible to use the *fit + penalty* approach to fit smooth functions:

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- with λ a non-negative smoothing parameter, which penalizes wiggliness of the fitted function, thereby controlling the bias-variance tradeof.

- The function g that minimizes this function is known as a *smoothing spline*.

- $g''(t)^2$ is the second derivative of the function g; the amount by which the slope is changing, i.e. a measure of wigglyness

Smoothing Spline

- It is also possible to use the *fit + penalty* approach to fit smooth functions:

$$\sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- with λ a non-negative tuning parameter, which penalizes wiggliness of the fitted function, thereby controlling the bias-variance tradeof.

- The function g that minimizes this function is known as a *smoothing spline*; it consists of basis functions, too.

- $g''(t)^2$ is the second derivative of the function g; the amount by which the slope is changing, i.e. a measure of wigglyness

Generalized Additive Models

- Often we have multiple predictors: $x_{i1}, \ldots, x_{ip}$
- We can generalize the ideas:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

- where $f_j()$ are smooth nonlinear functions (polynomials, cubic splines, natural splines, smoothing splines).
- It is an additive model, the smooth versions of each variable are added.
- Can be generalized to other GLM response variable types (e.g., binomial, count responses).

Exercise - splines

- Get the `nesda.sav` file (Netherlands Study of Depression and Anxiety) from the github repo.
- Read it into R as follows:

```
library("foreign")
read.spss("nesda.sav", use.value.labels=F, to.data.frame=T)
```

- Separate the observations into training (N=400) and test (N=200) parts.
- Use function `gam` from package **`mgcv`**.
- Fit a smoothing spline, predicting `mdd` based on `s(aconscie)`.
- Set the `method` argument of function `gam()` to `"REML"`.
- Use the `summary` and `plot` functions to inspect and interpret the result.

Exercise - splines

- Type `?s` and check out the meaning of the `k`, `bs` and `sp` arguments.
- Repeat the analysis, but now fit an unpenalized smoothing spline: `s(aconscie, sp = 0)`. Also fit a spline using 20 basis functions `s(aconscie, k = 20)`.
- Use the `summary` and `plot` functions to inspect and compare the results.
- Fit a GAM containing smoothing splines of both `aconscie` and `neurot` as predictors of `mdd`. Evaluate the misclassification rate on the test dataset, using the `predict()` function.

Support Vector Machines

- Maximum Margin Classifier
- Support Vector Classifier
- Support Vector Machine

Maximum Margin Classifier

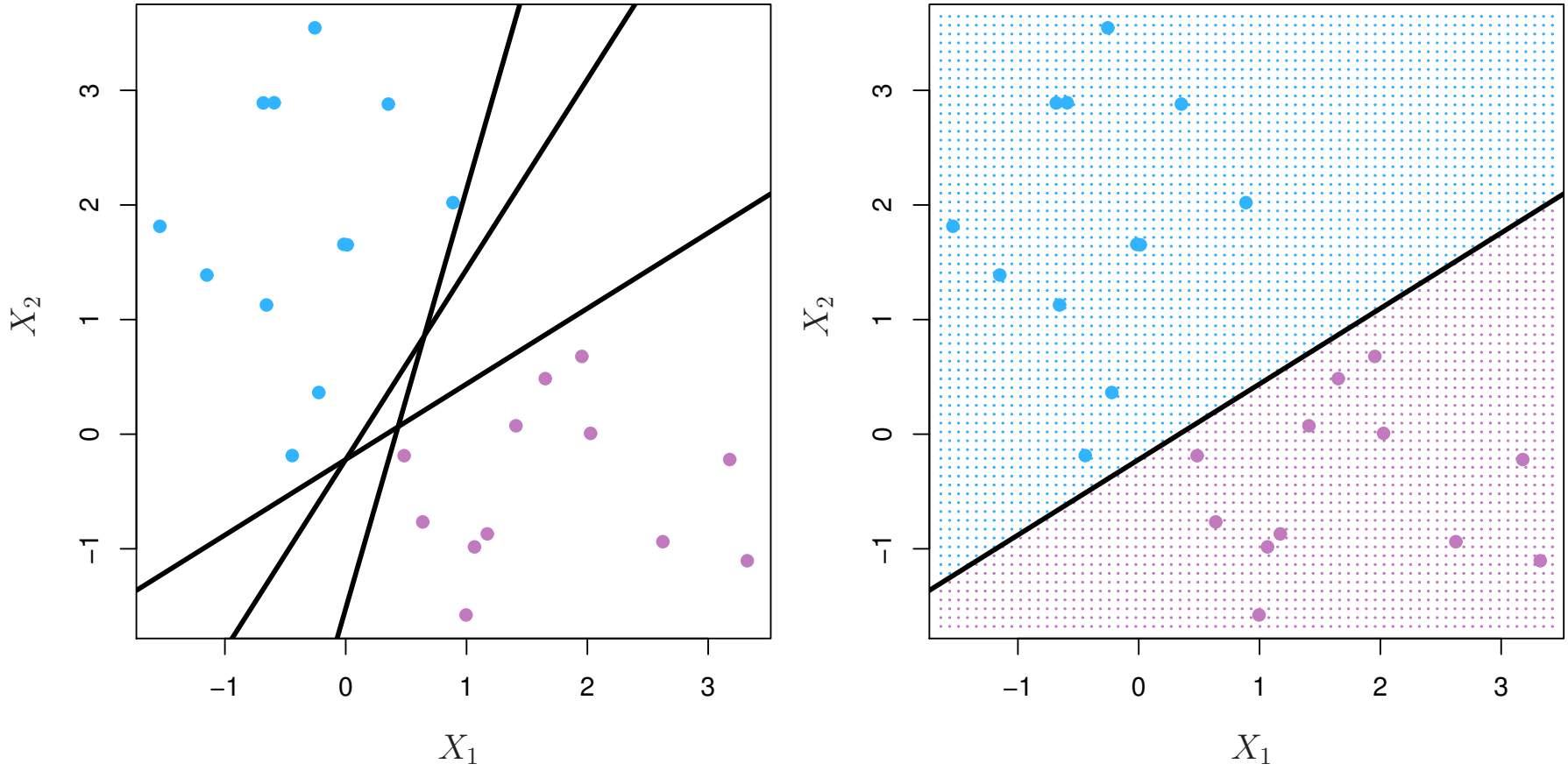
- Classification based on a *separating hyperplane*.
- That is, finding in the feature space a (hyper)plane that separates the two classes.
- A hyperplane can be seen as an equation of the feature variables:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- If we code $y \in \{-1, 1\}$ then the separating hyperplane has the following property

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) > 0$$

Example - Seperating hyperplane



Maximum Margin Classifier

- There may be several perfectly separating hyperplanes, then the one with the largest margin is chosen.

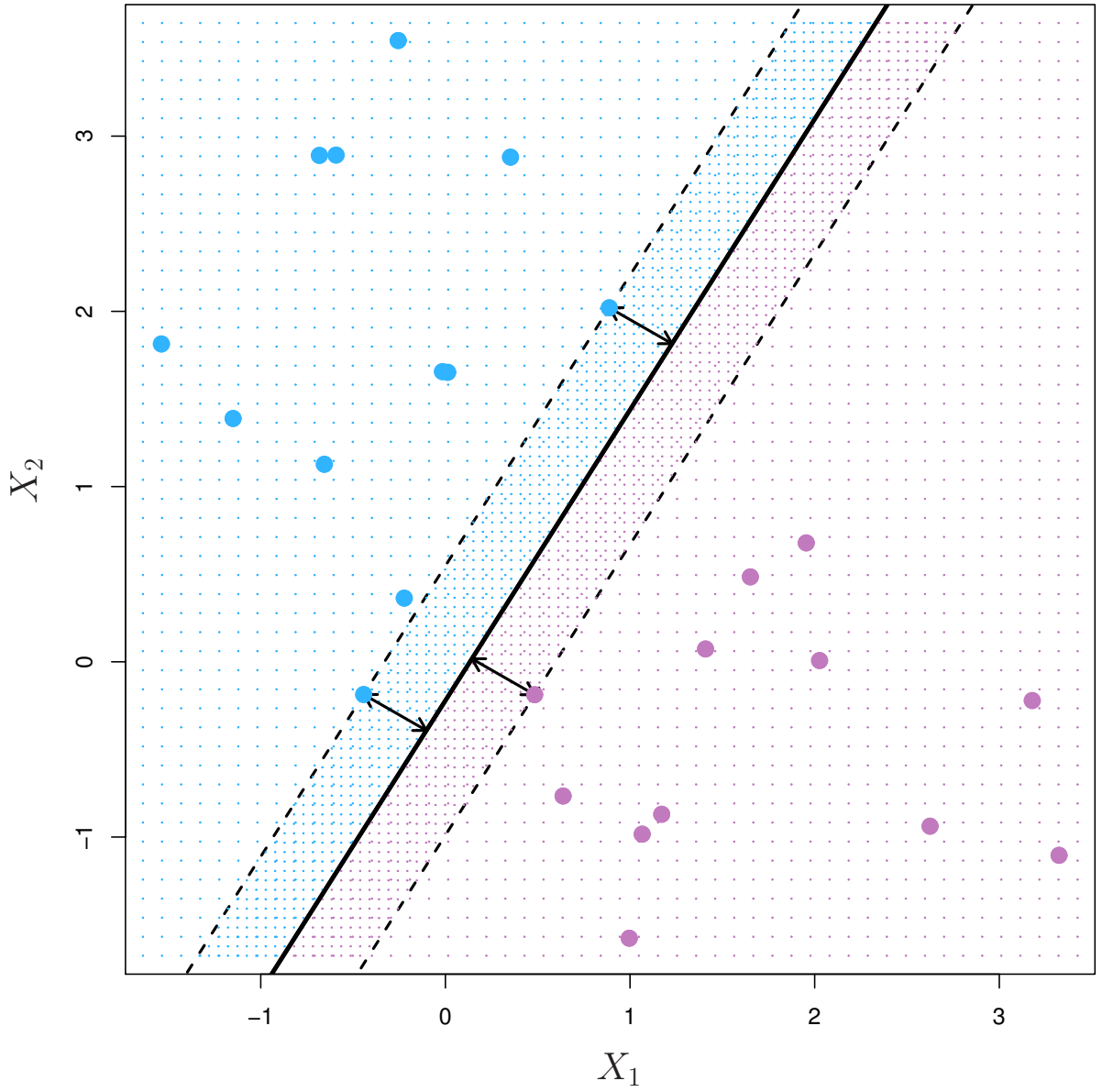
- This amounts to:

$$\text{maximize } M(\beta_0, \ldots, \beta_p)$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) \geq M \;\; \forall\, i = 1, \ldots, n$$

Example - Maximum Margin Classifier



Support Vector Classifier

- Often no perfectly separating hyperplane exists.
- Generalize the maximum margin classifier idea using a *soft margin*.
- We allow some observations to be on the wrong side of the margin or even on the wrong side of the hyperplane.
- We introduce *slack variables*: $\epsilon_1, \ldots, \epsilon_n$ and a tuning parameter C:

$$\text{maximize } M(\beta_0, \ldots, \beta_p, \epsilon_1, \ldots, \epsilon_n)$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) \geq M(1 - \epsilon_i) \;\; \forall\, i = 1, \ldots, n$$

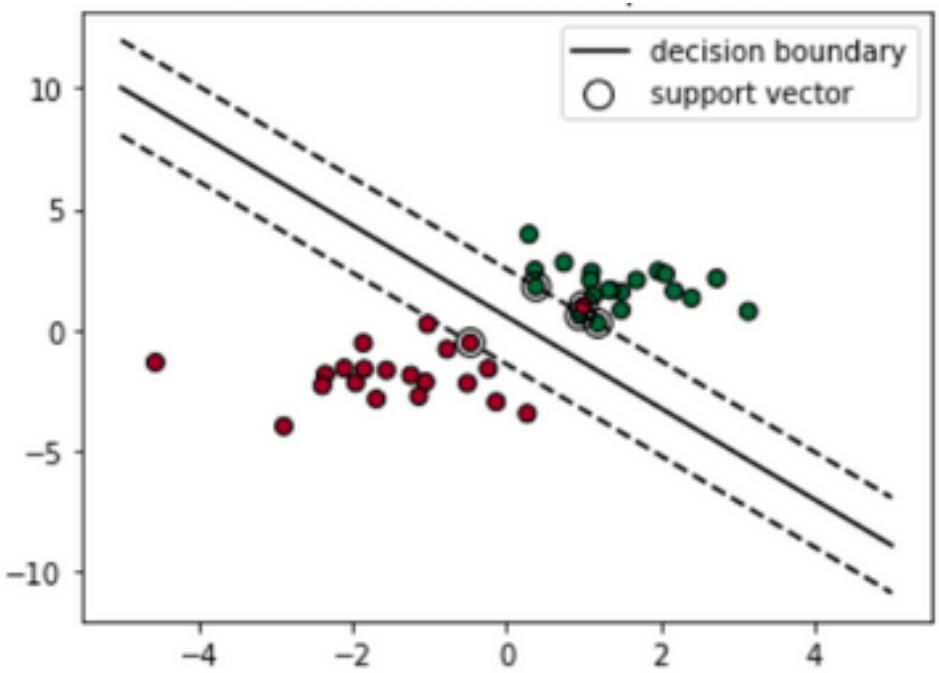
$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C$$

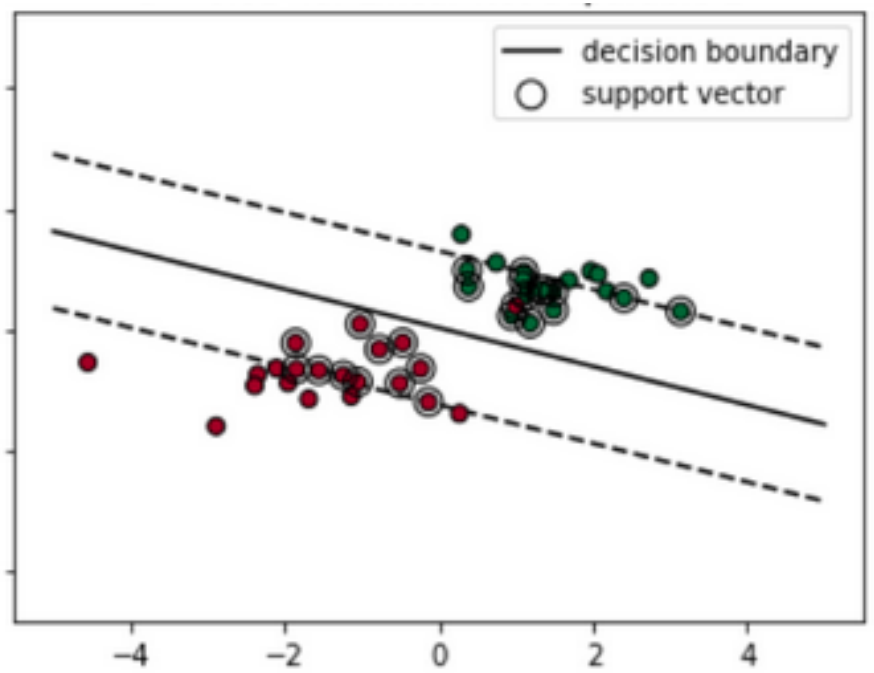
Support Vector Classifier

- The *slack variables*: $\epsilon_1, \ldots, \epsilon_n$ tell us where the i-th observation is located, relative to the margin and hyperplane:
  - If $\epsilon_i = 0$, the observation is on the correct side of the margin.
  - If $\epsilon_i > 0$, the observation is on the incorrect side of the margin.
  - If $\epsilon_i > 1$, the observation is on the incorrect side of the hyperplane.

- The tuning parameter C functions like a budget:
  - Thus, if $C = 0$, there is no budget for violations.
  - As C increases, the classifier becomes more tolerant of errors (allows for more *slack*) and the margin becomes wider.

Support Vector Classifier





Support Vector Classifier

- Observations that are on the margin or on the wrong side of the margin are called *support vectors*

- The decision rule is based only on a small fraction of training observations, which makes it quite robust to behavior of observations far from the hyperplane.

- In contrast, Linear Discriminant Analysis depends on the mean of all observations to construct the discriminant function and is therefore sensitive to points far from the decision line.

Support Vector Machine

- In practice the decision boundary is often nonlinear
- We can enlarge the feature space to allow for nonlinear boundaries
- Instead of using a support vector classifier on

$$X_1, X_2, \ldots, X_p$$

use

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

or

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2, X_1X_2, X_1X_3, \ldots$$

- In the enlarged space the decision boundary is linear, in the original space the decision boundary is nonlinear.

Support Vector Machine

- There are many ways to enlarge the feature space by using all kinds of functions of the different Xs.

- The computational burden increases rather fast with increasing number of features.

Example Inner Product

	mdd	aconscie	neurot
1	0	-1.10	-0.22
2	0	-0.09	-1.42
3	0	-1.27	-0.33
4	1	-0.93	0.55
5	0	-0.59	-0.43

Example Inner product

- $\langle x_i, x_{i'} \rangle$ is the *inner product* defined by $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$
- For person 1 and 2

$$\langle x_1, x_2 \rangle = -1.10 \times -0.09 + -0.22 \times -1.42 = 0.41$$

- It is the 'covariance' between the observations for person 1 and 2
- The inner product is the correlation or more general a measure of similarity
- Important to first standardize the variables, otherwise a variable may be dominant

Example Inner product

	1	2	3	4	5
1	1.25	0.41	1.46	0.90	0.75
2	0.41	2.03	0.58	-0.70	0.67
3	1.46	0.58	1.71	1.00	0.89
4	0.90	-0.70	1.00	1.17	0.31
5	0.75	0.67	0.89	0.31	0.54

Support Vector Machine

- The linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$$

- where $< x_i, x_i' >$ is the *inner product* defined by $< x_i, x_i' >= \sum_{j=1}^{p} x_{ij} x_{i'j}$
- The inner product is the correlation or more general a measure of similarity.
- It is possible to define the matrix K with all inner products:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- The K is called the kernel, and is an $n \times n$ matrix.

- It is possible to generalize the kernel, for example:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

  gives a polynomial kernel of degree d. It is still a $n \times n$ matrix.

- Using this kernel amounts to fitting a linear support vector classifier in a higher-dimensional space involving polynomials of degree d.

- Another choice of kernel is the *radial basis kernel* which is defined by:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

  For this kernel the *implicit* feature space is infinite dimensional.

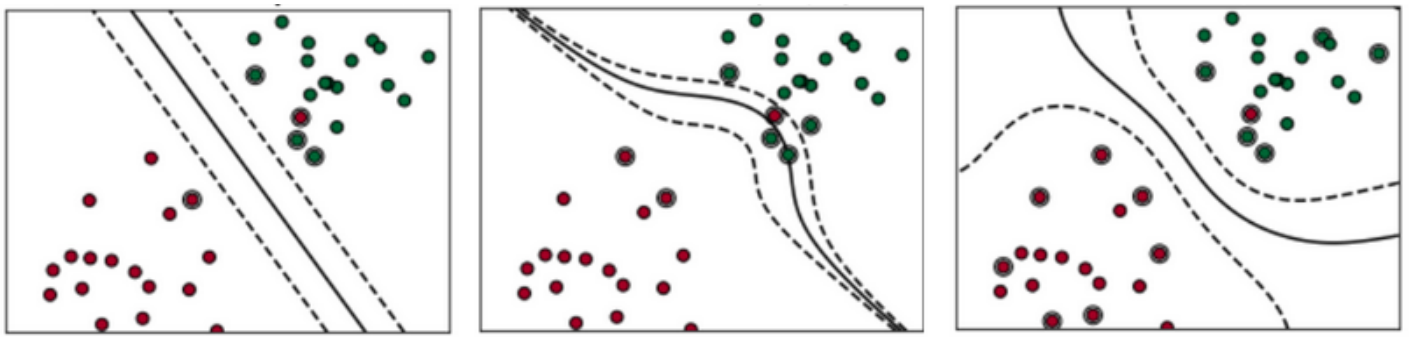
  γ is a tuning parameter, with higher values yielding a smoother decision boundary (i.e., higher bias and lower variance).

- In order to use kernels properly it is important to scale the variables first (different scalings give different kernels give different solutions).

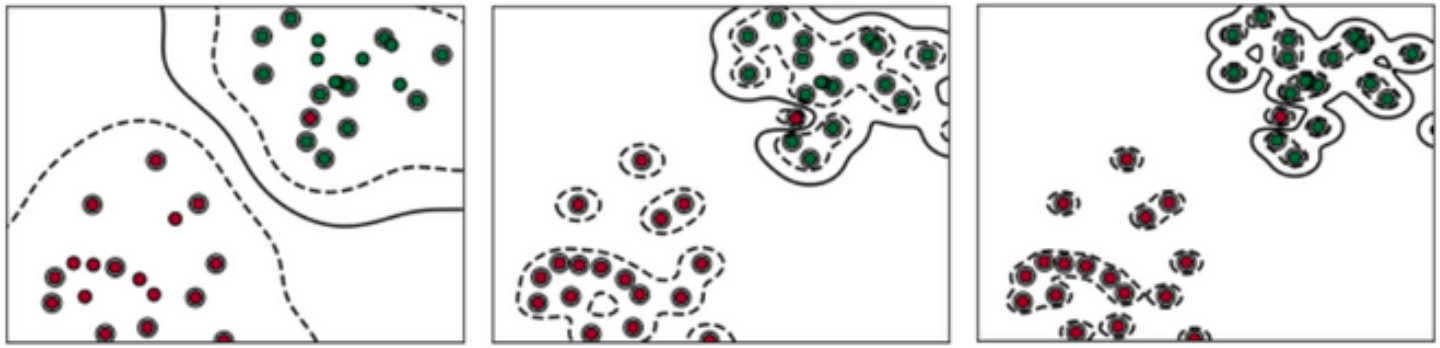
- The main advantage of kernels is that they avoid the need to actually transform to an enlarged space; the only thing needed is to compute the kernel and apply the support vector classifier on this kernel.

- This is alo known as the kernel trick.

Support Vector Machine



Linear, polynomial, radial basis kernels



Radial basis kernel with $\gamma = 1, 10$ or 20.

Exercise - Support Vectors

- Use `aconscie` en `neurot` as predictors
- Use `mdd` as response (first make it a factor)
- Use the library **`e1071`**
- To find optimal tuning parameters use the function `tune()`
  ```
  out=tune(svm, formula, data, kernel="..",
  ranges=list(cost=c(0.001,..,100)))
  ```
- With the optimal budget (cost) fit the model, make a plot and predictions for the test data set:
  ```
  svmfit=svm(formula, data, kernel="..", cost=..,
  scale=FALSE)
  ```

Exercise - Support Vectors

- Fit a support vector machine with radial basis kernel, tuning the budget and γ parameter::
  (`ranges=list(cost=c(0.1,1,10,100,1000),`
  `gamma=c(0.5,1,2,3,4))`)

- With the optimal budget (cost) and γ fit the model again, make a plot and predictions for the test data set

Exercises from book

- Chapter 7: 1, 4, 7, 8.

  For 7 and 8, it is helpful to first make a correlation plot (e.g., `plot(Wage)`), and then pick a small number of predictors you want to include in your model.

- Chapter 9: 1, 2, 3, 8.