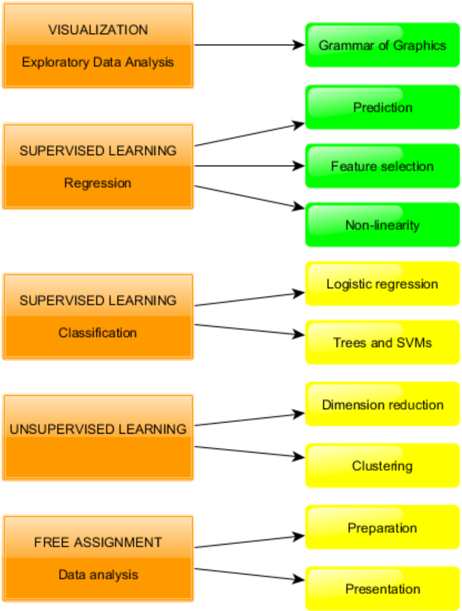
Supervised Learning: Regression

Nonlinearity

Maarten Cruyff

Program


VISUALIZATION
Exploratory Data Analysis
Grammar of Graphics
SUPERVISED LEARNING
Regression
Prediction
Feature selection
Non-linearity
SUPERVISED LEARNING
Classification
Logistic regression
Trees and SVMs
UNSUPERVISED LEARNING
Dimension reduction
Clustering
FREE ASSIGNMENT
Data analysis
Preparation
Presentation

Content

1. Polynomials
2. Splines
3. Interactions

- linear model
- regression trees

How to fit non-linearity?

Relationship x, y

- Polynomials
- Splines

Relationship $x_1 x_2, y$

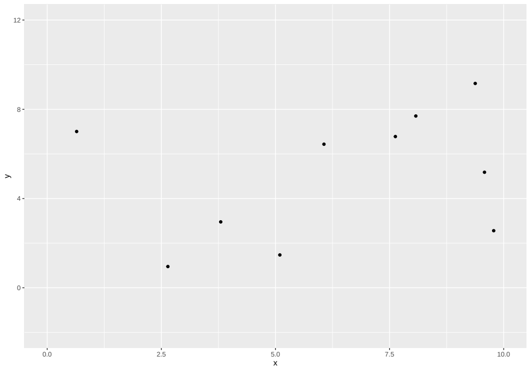
- interactions with `lm()` (linear)
- interactions with trees (non-linear)

Polynomials

Small data example

Data suggests non-linearity
- fit a curve with ‘lm()‘
- polynomials
- splines

What’s the difference and how does that work?



Fit polynomials with `poly()`

- quadratic model

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2$$

```
quad <- lm(y ~ poly(x, 2), data = d)
```

- cubic model

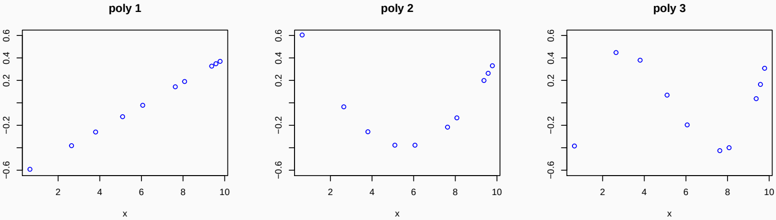
$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

```
cub <- lm(y ~ poly(x, 3), data = d)
```

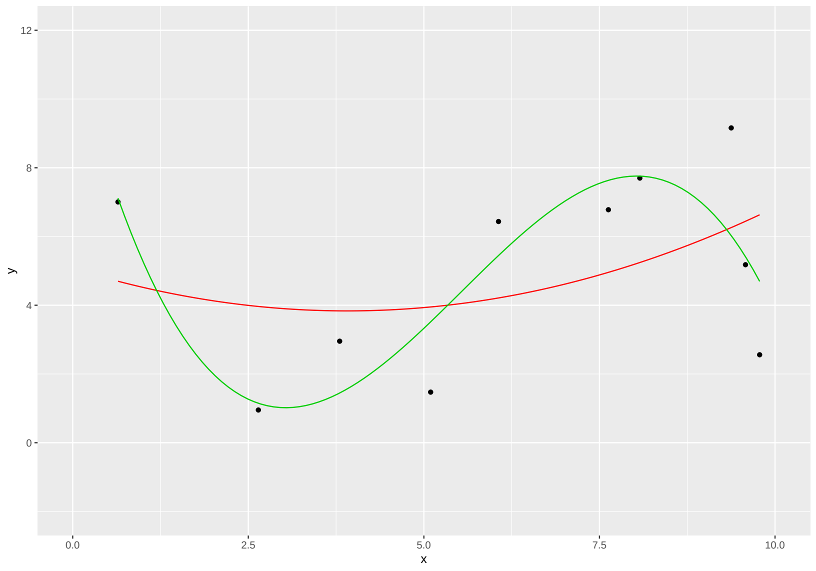
- etc.

Orthogogal basis matrix ‘poly(x, 3)

	(Intercept)	poly(x, 3)1	poly(x, 3)2	poly(x, 3)3
1	1	-0.592	0.604	-0.384
2	1	-0.381	-0.036	0.448
3	1	-0.260	-0.258	0.380
4	1	-0.123	-0.377	0.069
5	1	-0.022	-0.377	-0.196
6	1	0.143	-0.216	-0.426
7	1	0.190	-0.134	-0.400
8	1	0.327	0.199	0.037
9	1	0.348	0.263	0.164
10	1	0.370	0.330	0.308



Predictions for `quad` and `cub`



Splines

What are splines

Splines are piecewise polynomial functions

- Partition range x in n subintervals
- Fit a polynomial function $b_j(x)$ to each subinterval
- Smooth connections at the `knots` (endpoints subintervals)

$$\hat{y}_i = \begin{cases} \beta_1 b_1(x) & \text{if } x \leq a \\ \beta_2 b_2(x) & \text{if } a < x \leq b \\ \beta_3 b_3(x) & \text{if } x > b \end{cases} \tag{1}$$

B-splines

Function `bs()` of package `splines`

- creates B-spline basis matrix

```
bs(x, df = NULL, knots = NULL, degree = 3)
```

- `df` degrees of freedom (`df = knots + degree`)
- `knots` user-specified quantile for the knots
- `degree` polynomial degree

To fit a cubic B-spline

```
lm(y ~ bs(x))
```

B-spline basis matrix for `bs(x)`

```
          1     2     3
 [1,] 0.000 0.000 0.000
 [2,] 0.401 0.112 0.010
 [3,] 0.444 0.234 0.041
 [4,] 0.384 0.365 0.116
 [5,] 0.295 0.429 0.209
 [6,] 0.127 0.413 0.446
 [7,] 0.085 0.371 0.538
 [8,] 0.006 0.121 0.873
 [9,] 0.001 0.063 0.935
[10,] 0.000 0.000 1.000
attr(,"degree")
[1] 3
attr(,"knots")
numeric(0)
attr(,"Boundary.knots")
[1] 0.644768 9.780757
attr(,"intercept")
[1] FALSE
attr(,"class")
[1] "bs"     "basis"  "matrix"
```

B-spline vs polynomial

```
lm(y ~ bs(x, degree = k))
```

is identical to

```
lm(y ~ poly(x, degree = k))
```

B-splines more flexible through specification `df` / `knots`, e.g.

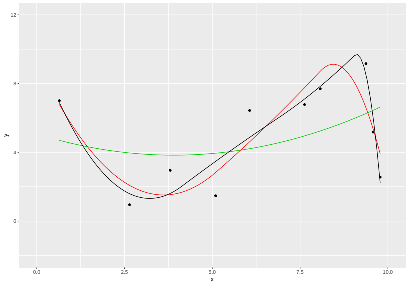
```
lm(y ~ poly(x, degree = 4))
lm(y ~ bs(x, knots = quantile(x, probs = 0.5)))
```

The basis matrix for this B-spline is shown on next slide

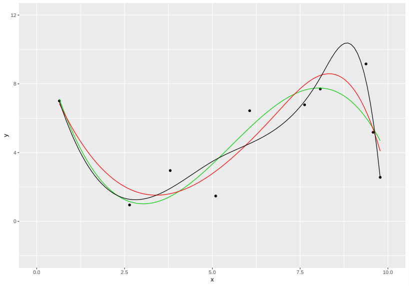
```
          1     2     3     4
 [1,] 0.000 0.000 0.000 0.000
 [2,] 0.515 0.158 0.015 0.000
 [3,] 0.504 0.317 0.061 0.000
 [4,] 0.350 0.457 0.170 0.000
 [5,] 0.204 0.487 0.307 0.000
 [6,] 0.041 0.310 0.630 0.019
 [7,] 0.020 0.222 0.684 0.074
 [8,] 0.000 0.017 0.342 0.641
 [9,] 0.000 0.004 0.188 0.808
[10,] 0.000 0.000 0.000 1.000
attr(,"degree")
[1] 3
attr(,"knots")
     50%
6.845001
attr(,"Boundary.knots")
[1] 0.644768 9.780757
attr(,"intercept")
[1] FALSE
attr(,"class")
[1] "bs"     "basis"  "matrix"
```

Varying `df`

Quadratic B-slines (df = 3, 4, 5)



Cubic B-splines (df = 3, 4, 5)

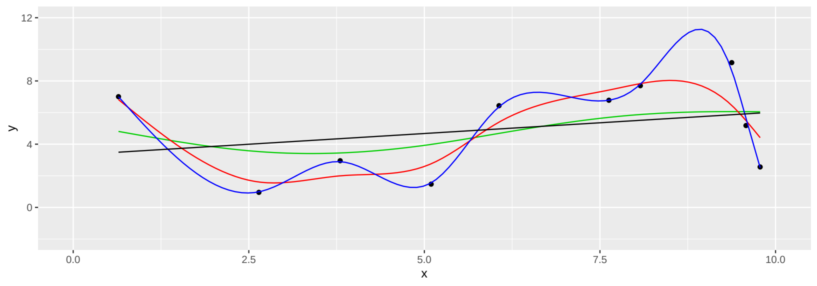


Smoothing splines

Knots at each unique value of x

- `df` penalty for ruggedness
- `cv = TRUE` is LOOCV
- `cv = FALSE` is ordinary CV (default)

```
smooth.spline(formula, df, cv = FALSE)
```



Interactions linear model

What are interactions

Main-effect models

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$$

- effect x_1 on y does not depend on $x_2, x_3, \ldots$
  - slope β_1 is fixed

Interaction-effect models

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 * x_2$$

- effect x_1 on y does depend on x_2
  - slope β_1 changes with β_{12} with unit increase in x_2
  - slope β_2 changes with β_{12} with unit increase in x_1

Example

How does math achievement depend on sex and SES?

Main-effect model: `lm(MathAch ~ SES + Minority)`

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	13.525	0.088	153.307	0
MinorityYes	-2.829	0.173	-16.354	0
SES	2.744	0.099	27.692	0

Interaction-effect model: `lm(MathAch ~ SES + Minority + SES:Minority)`

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	13.499	0.089	152.279	0.000
MinorityYes	-2.940	0.177	-16.586	0.000
SES	2.942	0.121	24.278	0.000
MinorityYes:SES	-0.597	0.210	-2.842	0.005

Regression plots

Main-effects model

Interaction-effects model

Linearity interaction effects

In the linear interaction-effects model:

- slope x_1 change linearly with changes in x_2
- slope x_2 changes linearly with changes in x_1

Non-linear approach to interactions

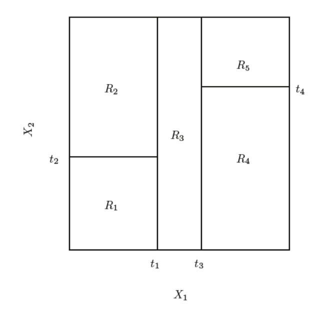
- effect x_1 changes at threshold values of x_2
- effect x_2 changes at threshold values of x_1

This is the **tree-based models** approach

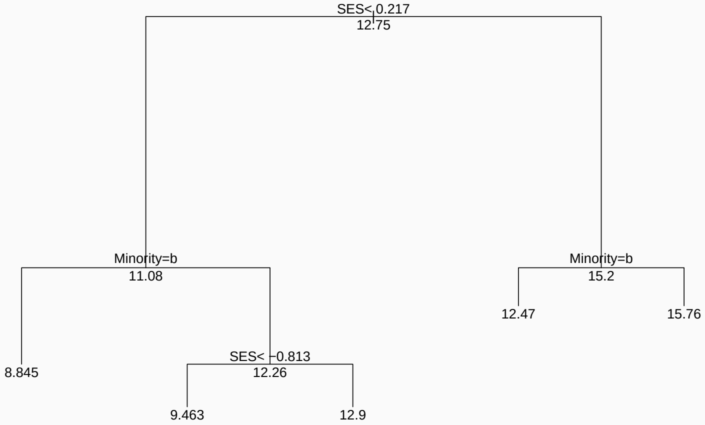
Interactions in trees

Binary recursive partioning with `rpart`

1. Partition predictor space in J square non-overlapping regions
2. For each observation in region R_j make the same prediction
3. Find partition that minimizes $\sum_j \text{MSE}_j$



Math achievement tree



Pros and cons

Pro Intuitive interpretation

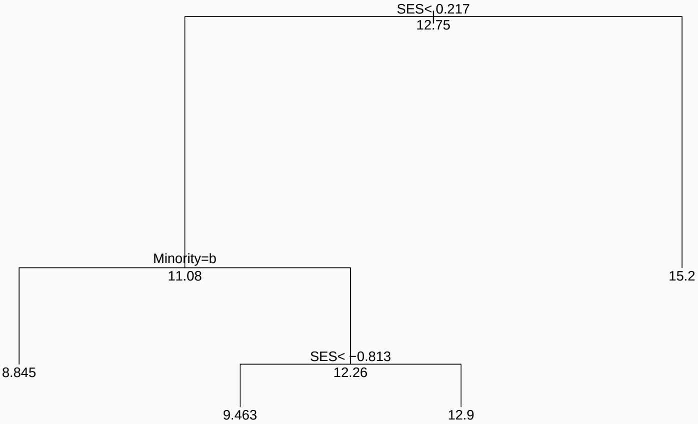
Con Recursive partioning is *top-down*, *greedy* algortithm

- Starts at the top of tree
- At each step, the best split at moment step is made
  - it does not look ahead for potentially better splits

Alternatives

- pruning (cut branches) reduces risk of overfitting
- alternatives (bagging, boosting, etc.) improve tree structure
  - to be discussed with classification trees

Pruned tree



Regression trees in R

Package `rpart`

```
train_tree <- rpart(formula,
                    data,
                    method = "anova")

plot(train_tree)
text(train_tree, cex=0.7)
```

Regression trees are different from classification trees

- classification trees later in course

Lab 2b

The lab introduces functions to fit polynomials, splines, interactions and trees.