Non-Linear Regression

Business Analytics Practice
Winter Term 2015/16
Stefan Feuerriegel

Today’s Lecture

Objectives

1. Understanding the need for non-parametric regressions
2. Familiarizing with two common variants: GAM and LOESS
3. Being able to apply the above methods in R and visualize the result

Outline

Outline

Linear Trend Line

Example

- Load dataset to visualize the relationship between age and wage

```
library(ISLR)
data(Wage)
Wage.small <- Wage[1:250, ] # fewer points look nicer on slic
```

- Load package `ggplot2` for visualization

```
library(ggplot2)
```

- Plot linear trend line, (→ OLS via `method="lm"`)

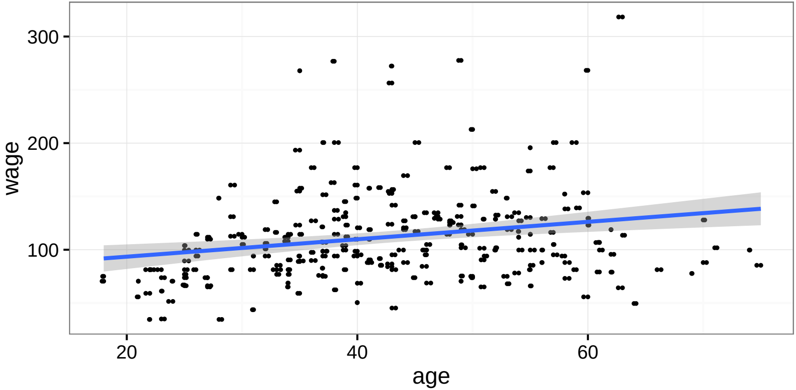
```
ggplot(Wage.small, aes(x=age, y=wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="lm") +
  theme_bw()
```

- `geom_jitter(...)` jitters points to reduce overlaps
- `geom_smooth(...)` is a default way to add smoothed lines

Linear Trend Line

Example

- Blue line is linear trend line with standard errors (gray area)



- With higher age, the wage flattens and even shrinks
- Relationship is obviously not linear

Motivation

Problem description

- Linear relationships show that variables are dependent
  $\rightarrow$ e. g. increase in income $\Rightarrow$ increase in happiness
- In reality, the true nature of effects and relationships is often non-linear
  $\rightarrow$ e. g. there is a maximum level of happiness
- Default regression techniques cannot adequately adapt to it

Solution: non-linear regressions

Advantages

1. Identify which relationships are non-linear
2. Visualize the shape of non-linear effects
3. Improve the fit (and/or the prediction performance) of models

Non-Linear Regressions

Overview

1. Polynomial regression
   $\rightarrow$ adding quadratic, cubic, $\ldots$ terms
2. Step-wise functions
   $\rightarrow$ similar to dummies for specific intervals
3. Splines
   $\rightarrow$ piecewise polynomial function
4. Generalized additive models
   $\rightarrow$ non-linear transformations for each term, but in additive fashion
5. Local regressions
   $\rightarrow$ sequence of regressions each based on a small neighborhood

Polynomial Regression

Definition

- Linear regression extended by adding quadratic, cubic, ... terms

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots$$

- In practice, compute $x_1 := x$ and $x_2 := x^2$, etc. before estimating

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots$$

- Degree of polynomial fixed beforehand or chosen by cross validation

Evaluation

- Coefficients often not relevant $\rightarrow$ difficult to interpret
- Mostly interested in t-test for coefficients or the fitted model
- Explosive growth nature of polynomials makes extrapolation difficult

Polynomial Regression in R

- Generate sample data

```
set.seed(0)
x <- runif(50, min=0, max=100)
y <- sin(x/50*pi) + runif(50, min=-0.5, max=0.5)
```

- Generate polynomial terms of up to degree *d* via `poly(x, degree=d, raw=TRUE)`, then perform least squares

```
m <- lm(y ~ poly(x, degree=3, raw=TRUE))
```

Note: `raw=TRUE` chooses default polynomials; else it uses orthogonal ones which are numerically more convenient

- Manual alternative

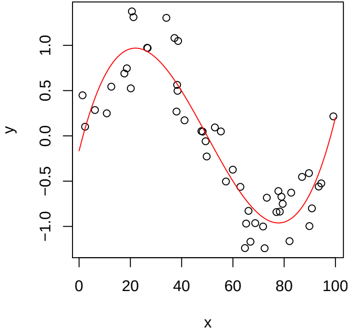
```
m <- lm(y ~ x + I(x^2) + I(x^3))
```

Note: `I(...)` is necessary to interpret arithmetic operations in a formula as such

Polynomial Regression in R

- Visualize result by manually generating the fitted line

```
predict_x <- seq(from=0, to=100, by=1)
# Named dataframe to avoid generating polynomial terms
predict_y <- predict(m, newdata=data.frame(x=predict_x))

plot(x, y)
lines(predict_x, predict_y, col="red")
```



Polynomial Regression in R

- ANOVA tests can identify the best-fit model

```
m.d2 <- lm(y ~ poly(x, degree=2, raw=TRUE))
m.d3 <- lm(y ~ poly(x, degree=3, raw=TRUE))
m.d4 <- lm(y ~ poly(x, degree=4, raw=TRUE))
anova(m.d2, m.d3, m.d4)

## Analysis of Variance Table
##
## Model 1: y ~ poly(x, degree = 2, raw = TRUE)
## Model 2: y ~ poly(x, degree = 3, raw = TRUE)
## Model 3: y ~ poly(x, degree = 4, raw = TRUE)
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1     47 12.1890
## 2     46  3.8570  1    8.3321 97.4464 7.786e-13 ***
## 3     45  3.8477  1    0.0093  0.1085    0.7434
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The P-value comparing $d = 2$ and $d = 3$ is almost zero
- Quadratic model is not sufficient $\rightarrow$ cubic is preferred

Stew-Wise Functions

Definition

- Fit a piecewise constant function

$$y(x) = \begin{cases} c_1 & \text{if } x \in I_1 \\ c_2 & \text{if } x \in I_2 \\ \dots & \dots \end{cases}$$

- No default procedure to choosing intervals
- Implemented via set of dummy variables for each interval $I_1, \ldots$
- Useful for interaction terms, i. e. $\boldsymbol{x}_1 \cdot \boldsymbol{x}_2$
  $\rightarrow$ e. g. salary depends on the interaction of education and tenure

Stew-Wise Functions in R

- Generate sample data

```
set.seed(0)
x <- c(runif(20, min=0, max=40), runif(20, min=40, max=100))
y <- c(runif(20, min=0, max=10), runif(20, min=30, max=40))
y <- y + runif(40, min=-5, max=5)
```

- Estimate linear model with dummies

```
m <- lm(y ~ I(x < 40))
coef(m)

##   (Intercept) I(x < 40)TRUE
##      35.27628     -30.15988
```

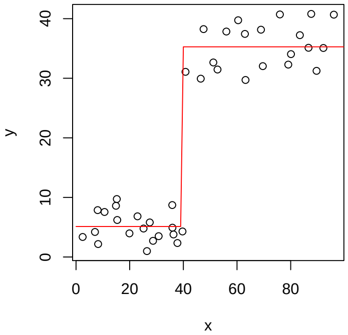
- Alternative is to split data via `cut(x, breaks=...)`

```
x2 <- cut(x, breaks=c(0, 40, 100))
coef(lm(y ~ x2))

## (Intercept)  x2(40,100]
##    5.116405   30.159878
```

Step-Wise Regression in R

- Visualize result by manually generating the fitted line

```
predict_x <- seq(from=0, to=100, by=1)
# Named dataframe to avoid generating polynomial terms
predict_y <- predict(m, newdata=data.frame(x=predict_x))

plot(x, y)
lines(predict_x, predict_y, col="red")
```



Spline Regression

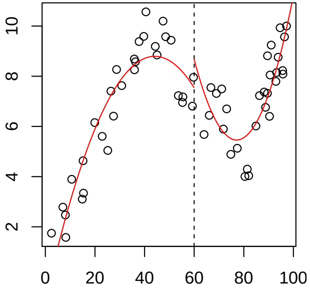
- Divide the range of variables into k distinct regions $I_1, \ldots, I_k$
- Fit a polynomial function to each region

$$y(\boldsymbol{x}) = \begin{cases} \beta_0^{(1)} + \beta_1^{(1)} \boldsymbol{x} + \beta_2^{(1)} \boldsymbol{x} + \ldots & \text{if } \boldsymbol{x} \in I_1 \\ \ldots & \ldots \\ \beta_0^{(k)} + \beta_1^{(k)} \boldsymbol{x} + \beta_2^{(k)} \boldsymbol{x} + \ldots & \text{if } \boldsymbol{x} \in I_k \end{cases}$$

- Can be rewritten with basis functions b_i as

$$y(\boldsymbol{x}) = \beta_0 + \beta_1 b_1(\boldsymbol{x}) + \beta_2 b_2(\boldsymbol{x}) + \ldots$$

- Smoothing splines work similarly but enforce a certain continuity



Outline

Splines

To be added . . .

Outline

Generalized Additive Models

Generalized Additive Models (GAM)

- Extend standard linear model by allowing non-linear functions
- Outcome depends linearly on (smooth) non-linear functions f_j

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

- Model is called additive, since we calculate separate $f_j(x_{ij})$ for each x_i
- f_j can be polynomials, though splines are more common

Pros and Cons

Advantages

- Fits automatically non-linear f_j to each x_i $\rightarrow$ no need for manual trying
- Non-linearity achieves a more accurate fit
- Model is additive, thus it is possible to examine the effect of each $\boldsymbol{x}_i$ on $\boldsymbol{y}$ individually

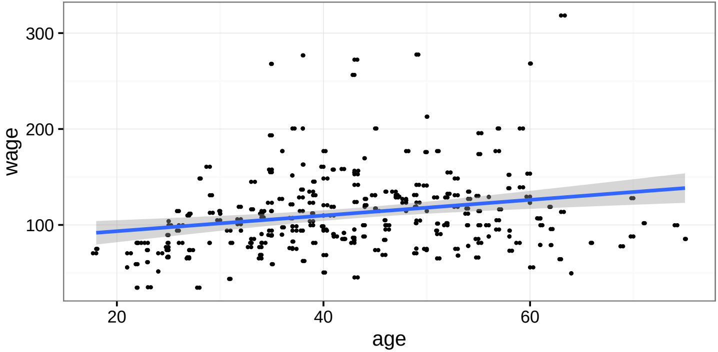
Drawbacks

- Restricted to an additive model
  $\rightarrow$ important interactions can be missed

GAM Smoothing in ggplot2

- `ggplot2` has a built-in support for GAM via `method="gam"`

```
ggplot(Wage.small, aes(x=age, y=wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="gam") +
  theme_bw()
```



GAM in R

- Load the `gam` package

```
library(gam)
```

- Estimate model, e. g. with smoothing splines

```
m.gam <- gam(wage ~ s(year, 4) + s(age, 5) + education,
             data=Wage)
m.gam

## Call:
## gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wa
##
## Degrees of Freedom: 2999 total; 2986 Residual
## Residual Deviance: 3689770
```

- `s(variable, df)` introduces smoothing splines of degree `df`
- `ns(variable, df)` are natural splines
- `education` is a factor and thus not treated
- Detailed summary on results via

```
summary(m.gam)
```

GAM in R

- Component-wise plots show the effect of each term

```
par(mfrow=c(1, 3))
plot.gam(m.gam, se=TRUE, col="blue")

## Error in 1:object$nsdf:  argument of length 0
```

- One might think that effect of `year` is linear

GAM in R

- ANOVA test identifies best-fit model
  → e. g. excluding `year` or assuming a linear or non-linear effect

```
m.gam1 <- gam(wage ~ s(age, 5) + education, data=Wage)
m.gam2 <- gam(wage ~ year + s(age, 5) + education, data=Wage)
m.gam3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data=Wage)
anova(m.gam1, m.gam2, m.gam3)

## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 0.0001419 ***
## 3      2986    3689770  3   4071.1 0.3483897
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- GAM with linear `year` is better than without (P-value < 0.001)
- Non-linear effect of `year` is not necessary (P-value > 0.05)

Outline

Local Regression

Local regression (LOESS)

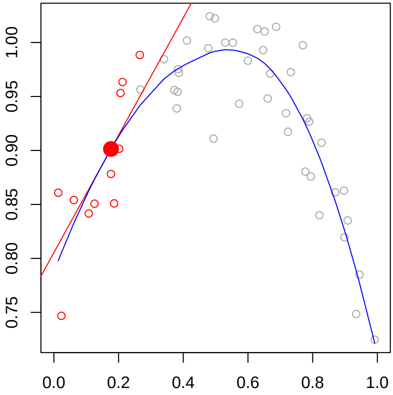
- Based on earlier locally weighted scatterplot smoothing (LOWESS)
- Locally weighted regression using nearest neighbors
  - Weight points stronger that are closer
  - Put less weights on the points further away

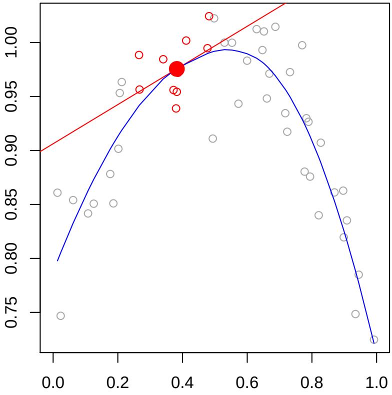
Non-parametric approach

- Smoother has no pre-defined form but is constructed from data
- As such, the underlying distributions can be unknown
- However, needs more observations to infer relationships from data

LOESS

Idea: locally weighting nearest neighbors





LOESS

High-level procedure

1. Choose a point x_0 at which LOESS is calculated
2. Choose subsets whose size is determined by a smoothing parameter
   - Proportion α of all points which influence the curvature of the smoothing at each point
   - Subset consists of the α nearest neighbors of x_0
   - Useful values of α are often in the range 0.25 to 0.5
3. Fit a low-degree polynomial to each subset
   - Fitting is done by weighted least squares
   - Gives more weight to points closer to x_0
   - Degree is often linear or quadratic
4. Repeat the above steps for all points in the dataset

Pros and Contras

Advantages

- Independent of a specific model (or distributions) that fit all the data
- Very flexible and thus can adapt to complex relationships
- No need to estimate a global function
  $\rightarrow$ only use nearest neighbors with a smoothing parameter

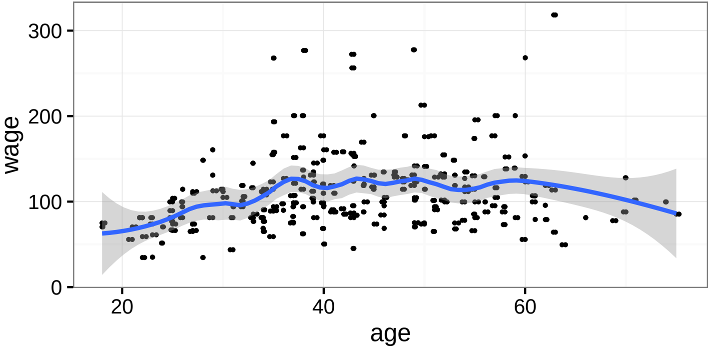
Drawbacks

- High computational costs
  $\rightarrow$ ggplot2 uses LOESS by default for up to 1000 data points
- Requires large and fairly dense dataset for good results
- Result is not a closed form solution, which can be further evaluated or interpreted

LOESS in ggplot2

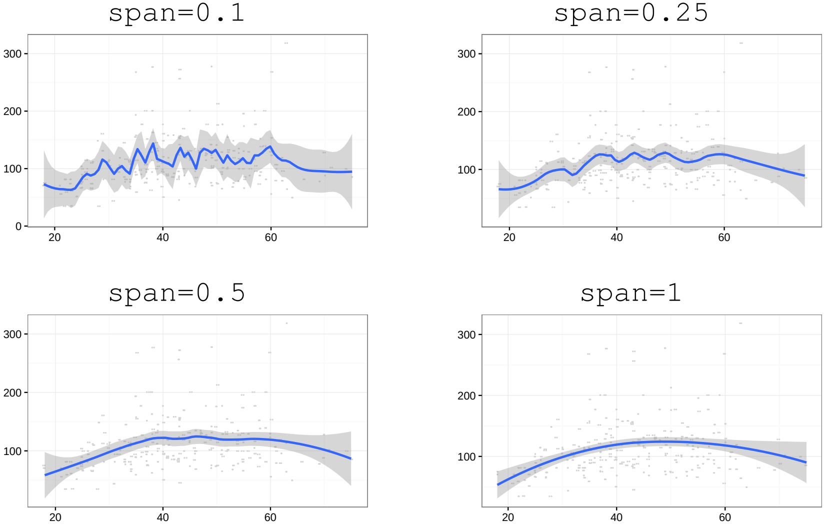
- `ggplot2` has a built-in support for LOESS via `method="loess"`

```
ggplot(Wage.small, aes(age, wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="loess", span=0.3) +
  theme_bw()
```



LOESS in ggplot2

Parameter `span` controls the intensity of smoothing



Outline

Wrap-Up

Summary

- Non-linear models can effectively supplement least squares
- Various non-linear models are available
  $\to$ manual tests necessary to find a good model
- For non-parametric methods, the fitting model can be unknown
- `ggplot(...)` is helpful to quickly gain first insights or for nice visualizations

Further readings

- Section 7 in the book "An Introduction to Statistical Learning"
- Package `mgcv` is a newer alternative to `gam`