# MS-C1620 Statistical inference

## 11 Kernel regression

Jukka Kohonen

Department of Mathematics and Systems Analysis
School of Science
Aalto University

Academic year 2021–2022
Period III–IV

# Contents

# Regression function

Simple linear regression (Lecture 7) is a special case of fitting a regression function to the data.

$$y_i = g(x_i) + \epsilon_i$$

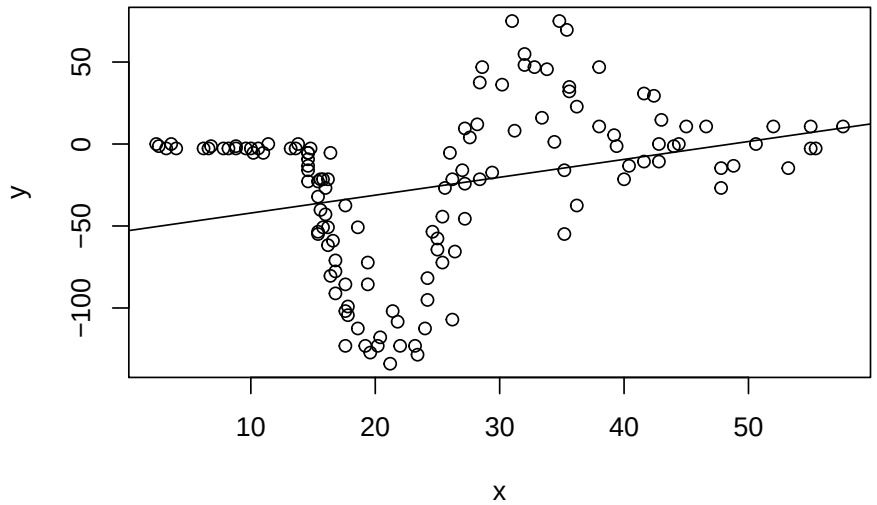
Linear model $g(x) = \beta_0 + \beta_1 x$ has two parameters.

Many other functional forms of $g$ could be used, e.g.

- multiple explanatory variables (Lecture 8)
- higher order polynomials (CV example on Lecture 9)
- piecewise regression
- **kernel regression (this lecture)**

# Example: Motorcycle data (testing crash helmets)

Head acceleration in a simulated motorcycle accident.
$x$ = time, explanatory variable (unit=ms)
$y$ = acceleration, response variable (unit=g)




Linear model fits badly, but would a polynomial be any better?

Instead of trying to fit a "global" model to all of the data, let's try to understand its behaviour "locally".

# General idea of local regression

Key idea: At any given point $x$, the value of the regression function $g(x)$ is calculated from nearby data points (not all data points).

Some variants of the idea:

- KNN regression: Average the $k$ nearest data points.
- Sliding window: Average all data points that are within $h$ units of $x$.
- Kernel regression (kernel smoothing): Average nearby data points, giving bigger weight to data points that are very near. A kernel function $K$ maps distances to weights.

# Simple kernel regression

Nadaraya-Watson regression: At any point $x$, define regression function value as a weighted average of data points

$$g(x) = \sum_{i=1}^{n} w_i y_i,$$

where the weights are calculated as

$$w_i = \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)},$$

and $K$ (kernel function) is some nice function that gives big values when $x_i$ is near $x$. The divisor just makes sure that the weights sum to one.

# Choice of kernel function

The kernel function is typically defined in two steps:

1. Choose a shape, such as a triangular function, parabola, or the density function of standard normal distribution
2. Choose a bandwidth that scales the shape to desired width = how far datapoints are used in the averaging

Example: parabolic (Epanechnikov) kernel

$$K_1(u) = \frac{3}{4}(1 - u^2)$$

for $-1 \leq u \leq 1$, and zero outside that interval.
Then scaled to bandwidth $h$ with

$$K_h(u) = K_1(u/h).$$

This is positive for $-h \leq u \leq h$.
See https://en.wikipedia.org/wiki/Kernel_(statistics) for many other kernel shapes.

# Choice of bandwidth – Effect

Large bandwidth = averaging many datapoints = very smooth regression function that only shows "large scale" features of the data. Efficiently smoothes small errors away, prone to underfitting.

Small bandwidth = averaging few datapoints = very wild regression function that follows the data very closely. But also retains its errors, prone to overfitting.

# Choice of bandwidth – Methods

Many methods exist for choosing the "best" bandwidth (see literature).

- For exploratory (visual) analysis you could just experiment with different values; see different features at different scales
- Use cross-validation: fit models to training set, choose bandwidth minimizing error on test set (R demo)
- Some rule-of-thumb formulas for optimal bandwidth value, under suitable assumptions about data (not on this course)
- Adaptive methods which use smaller bandwidth if there are many data points nearby.

# Local linear regression

Instead of taking the average of the nearby points, we can also fit a straight line to them. This is called local linear regression.

In other words, we do a linear regression, but only on the data points $x_i$ that are near $x$, and weighted by a kernel function. Then define $g(x)$ to be the value of that regression line at $x$.

Note that for each value $x$ where we are evaluating the regression function, we look at *different* "nearby" datapoints or use different weights, so the regression function $g(x)$ that we obtain need not be "linear" at all.

Nadaraya-Watson (local constant) and local linear regression usually produce similar results, except at edges of the data. (Consider what happens in time series prediction.)

Just like in "global" regression, in local regression we can also use higher degree polynomials (e.g. parabolic).

# Contents

# Kernel density estimation

The same idea, "looking at nearby datapoints", can be used to estimate the density function of a distribution, if we have a sample $x_1, x_2, \ldots, x_n$ from it.

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i)$$

You can think of the datapoints $x_i$ as representing point masses $1/n$ each, then doing kernel smoothing to distribute those masses around $x_i$ over some distance (by the kernel function).

This gives often a nicer, smoother estimate of the unknown density than a histogram.

# Contents

# More information

You can learn more about local / kernel regression from e.g. the freely available book https://web.stanford.edu/~hastie/ElemStatLearn/ (Chapter 6: Kernel smoothing methods)