Local Polynomial Kernel Regression

Presented by Ksenia Bushmeneva

February 6, 2010

Definitions and Motivation

- **Kernel smoothing** refers to a general class of techniques for non-parametric estimation of functions. It can be useful in two classes of problems:
  - density function estimation
  - non-parametric regression estimation
- **Benefits of non-parametric estimation**: it is a lot less rigid than the parametric estimation, i.e. it imposes fewer restrictions on the functional relationship between the covariates and the outcome variable.
- **Application of non-parametric regression**: can suggest a simple parametric model that fits the data well (diagnostic). Estimation and prediction (same as parametric regression).

Parametric vs. Non-parametric

Parametric regression (linear regression)

$$y_i = \mu(x_i) + \epsilon_i = x_i'\beta + \epsilon_i,$$

where $\mu(x)$ is a known (linear) function and $E(y|x) = \mu(x) = x\beta$.

Goal: to obtain an estimate $\hat{\beta}$ of β, and to find fitted values $\hat{y}$, where $\hat{y} = W\,y = x\hat{\beta} = x(x'x)^{-1}x'y$

Non-parametric regression (Kernel regression, Local polynomial kernel regression)

$$y_i = \mu(x_i) + \epsilon_i,$$

where $\mu(x)$ is some unknown function and $E(y|x) = \mu(x)$.
$\mu(x)$ is known as the mean response curve.

Goal: to obtain an estimate $\hat{\mu}(x; h)$ of $\mu(x)$.

Kernel regression

- The goal of the Kernel regression is to find the appropriate weights matrix, such that $\hat{y} = W\,y = \hat{\mu}(t)$, where $\hat{y}$ is a weighted average.
- **How do we assign weights to our observations?**
- Nadaraya (1964, 1965) and Watson (1964) proposed one of the most common methods to assign weights using:

$$W_i = \frac{K(\frac{t_i - t}{h})}{\sum_{i=1}^{n} K(\frac{t_i - t}{h})}$$

- Then $\hat{y}$ at the point t is equal to:

$$\hat{y}(t) = \hat{\mu}(t) = \frac{\sum_{i=1}^{n} K(\frac{t_i - t}{h})\, y_i}{\sum_{i=1}^{n} K(\frac{t_i - t}{h})}$$

where the $\hat{y}(t)$ is simply as weighted sum of all y's and $\sum_{i=1}^{n} W_i = 1$, K() is a Kernel function (e.g. Normal density), h - bandwidth.

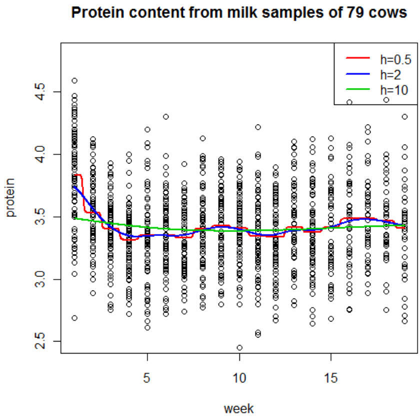
Kernel regression in R

The following built-in R function evaluates the Nadaraya-Watson estimate in R:

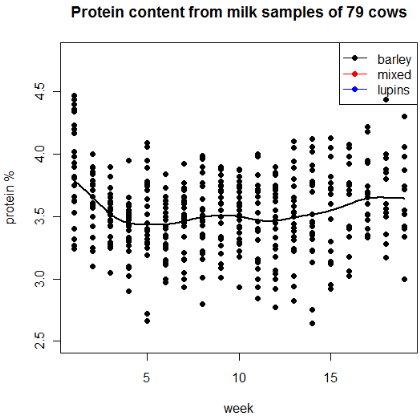
ksmooth$(x, y, kernel = c(''box'', ''normal''), bandwidth = 0.5, range.x = range(x), n.points = max(100, length(x)), x.points)$

- x - input x values
- y - input y values
- kernel - the kernel to be used.
- bandwidth - the bandwidth.

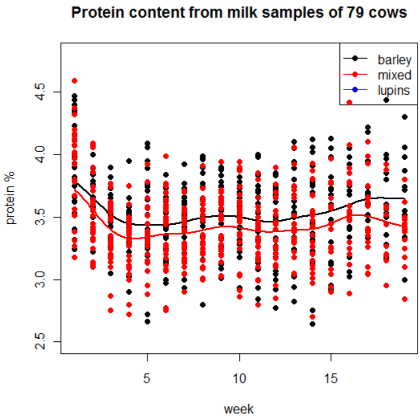
Kernel regression results: cow data



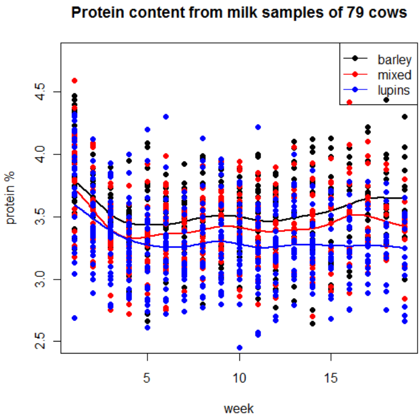
Kernel regression results: cow data - by diet



Kernel regression results: cow data - by diet



Kernel regression results: cow data - by diet



Conclusions from Kernel regression

- "Wash-out" period is needed for all 3 diets in order to estimate the diet effect;
- On average barley diet has the best performance in terms of protein content, and lupins diet performs the worst;
- For lupins diet, the protein count declines and stays low for the entire period of observation;
- For barley diet, the protein count declines initially, but then slowly recovers close to initial level.

Local Polynomial Kernel Regression

Advantage: helps to mitigate the boundary problem common to Kernel regression.

Boundary problem arises since the Kernel weights at the boundary of the data are no longer symmetric. As a result $\hat{y}$ is either overestimated or underestimated.

- **Polynomial regression**:
  $y_i = \beta_0 + \beta_1(t_i - t) + \beta_2(t_i - t)^2 + ... + \beta_p(t_i - t)^p + \epsilon_i$
- **Kernel**: use Kernel function to assign weights.
- **Local**: new set of weights is generated for every point t in the sample => which results in new vector of $\hat{\beta}$ for every y_i .

Estimation

- Suppose our sample consists of (y_i, t_i), $i = 1...n$ pairs of data. Then $\hat{\beta}$ can be obtained by solving the following weighted least squares problem:
  $\hat{\beta} = argmin \sum_{i=1}^{n} K(\frac{t_i - t}{h})(y_i - \beta_0 - \beta_1(t_i - t) - \beta_2(t_i - t)^2 - ... - \beta_p(t_i - t)^p)^2$

Estimation: continues

- Standard least squares theory leads to solution:

$$\hat{\beta} = (X_x^T W_x X_x)^{-1} X_x^T W_x Y$$

Where $Y = (Y1, ..., Y_n)^T$ is a vector of responses.

$$X_x = \begin{bmatrix} 1 & t_1 - t & (t_1 - t)^2 \\ . & . & . \\ . & . & . \\ . & . & . \\ 1 & t_n - t & (t_n - t)^2 \end{bmatrix}$$

is an [nx3] design matrix for the 2nd degree polynomial, which can be extended to p-th order.
And $W_x = diag\left\{K(\frac{t_1 - t}{h}), ..., K(\frac{t_n - t}{h})\right\}$ is an [nxn] diagonal matrix of weights.

Estimation: continues

Then the predicted value of y at point t is simply equal to β_0 and can be found from:

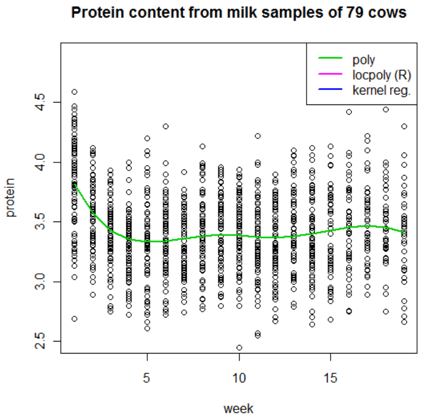
$$\hat{y}(t) = \hat{\mu}(t; p, h) = \beta_0 = e_1^T \, (X_x^T \, W_x \, X_x)^{-1} X_x^T \, W_x \, Y \quad (1)$$

Where $e_1^T = (1\,,0\,,0\,....\,0)$ is a $[(p+1)x1]$
Thus in order to estimate each $y(t)$ the following 3 steps need to be carried out:

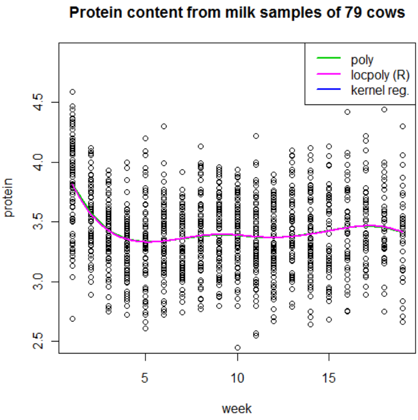
1. construct X_x matrix for each $t \in \{t_1, ..., t_n\}$
2. construct W_x matrix for each $t \in \{t_1, ..., t_n\}$
3. estimate $\hat{\mu}(t; p, h)$ using equation (1).

Nataraya-Watson estimator is a special case of local polynomial kernel estimator for p=0, i.e. $\hat{\mu}(t; 0, h)$.
Local linear kernel estimator is also a special case of local polynomial kernel estimator for p=1.

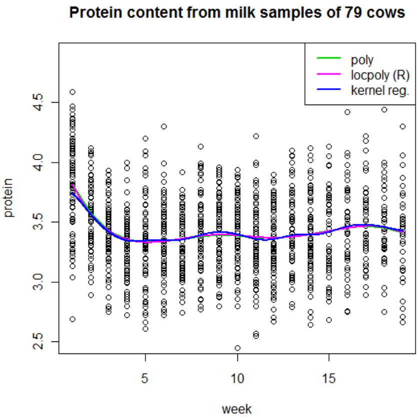
Local polynomial regression results: amalgamated cow data



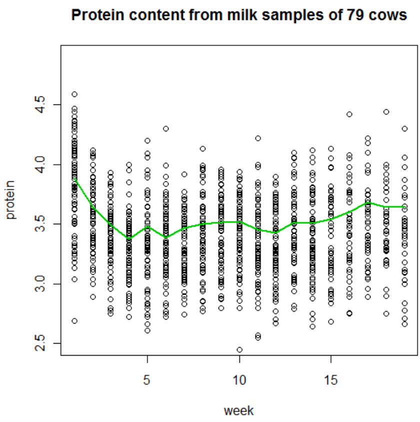
Local polynomial regression results: amalgamated cow data



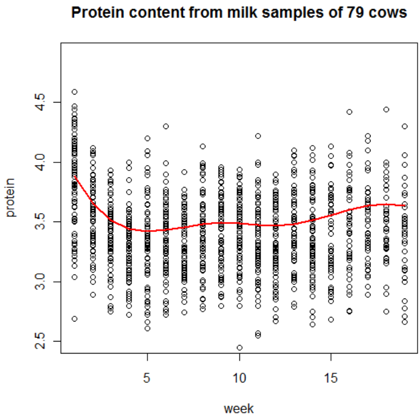
Local polynomial regression results: amalgamated cow data



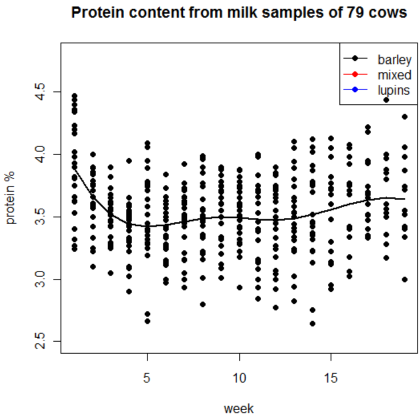
Local polynomial regression results: for h=0.5 and h=2



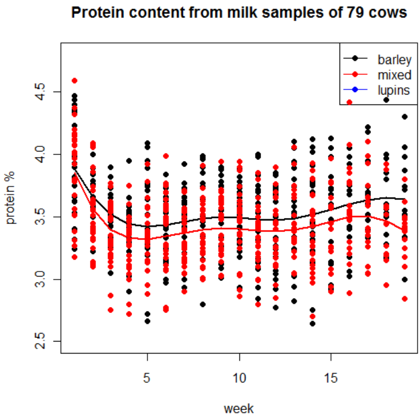
Local polynomial regression results: for h=0.5 and h=2



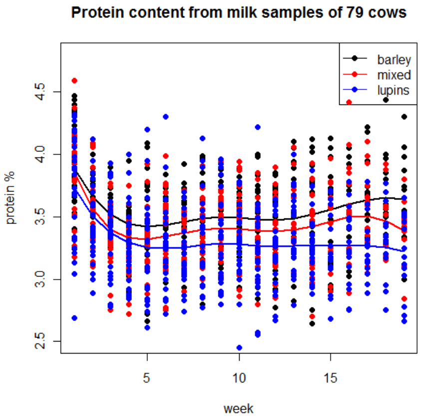
Local polynomial regression results: by diet types



Local polynomial regression results: by diet type



Local polynomial regression results: by diet type



locpoly R-function

library(KernSmooth) contains the R function **locpoly** for estimating the regression function using local polynomials.

locpoly(*x, y, drv = 0L, degree, kernel = "normal", bandwidth, gridsize = 401L, bwdisc = 25, range.x, binned = FALSE, truncate = TRUE*)

- x - vector of x data. Missing values are not accepted.
- bandwidth - the kernel bandwidth smoothing parameter.
- y - vector of y data. This must be same length as x, and missing values are not accepted.
- drv - order of derivative to be estimated.
- degree - degree of local polynomial used.

Thank you!