MATH 829: Introduction to Data Mining and Analysis
Kernel smoothing

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

March 21, 2016

Motivation:

- A (global) linear model may not be appropriate for some data.

Motivation:

- A (global) linear model may not be appropriate for some data.
- However, a linear model may be appropriate locally.

Motivation:

- A (global) linear model may not be appropriate for some data.
- However, a linear model may be appropriate locally.
- We now explore how one can fit a different but simple model separately at each query point.

Motivation:

- A (global) linear model may not be appropriate for some data.
- However, a linear model may be appropriate locally.
- We now explore how one can fit a different but simple model separately at each query point.
- As we will see, this can be naturally done, without significantly increasing the number of parameters to estimate.

Motivation:

- A (global) linear model may not be appropriate for some data.
- However, a linear model may be appropriate locally.
- We now explore how one can fit a different but simple model separately at each query point.
- As we will see, this can be naturally done, without significantly increasing the number of parameters to estimate.
- We will use local information to fit each local linear model.

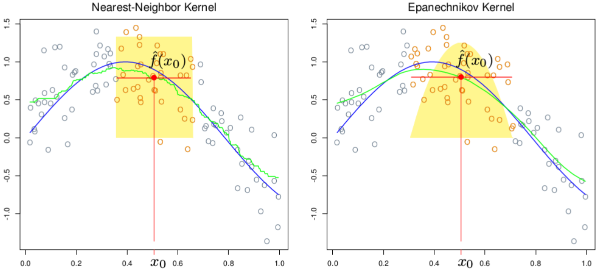
Motivation:

- A (global) linear model may not be appropriate for some data.
- However, a linear model may be appropriate locally.
- We now explore how one can fit a different but simple model separately at each query point.
- As we will see, this can be naturally done, without significantly increasing the number of parameters to estimate.
- We will use local information to fit each local linear model.
- Localization is achieved via a weighting function (*kernel*) $K(x, x_i)$, or a parametric family of kernels $K_\lambda(x, x_i)$ for $\lambda \in \Lambda$.

Recall the k-nearest-neighbor average

$$\hat{f}(x) = \text{Ave}(y_i : x_i \in N_k(x))$$

approximates the regression function $E(Y|X = x)$.

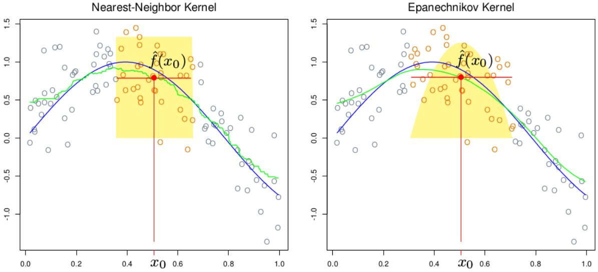


ESL, Figure 6.1.

Recall the k-nearest-neighbor average

$$\hat{f}(x) = \text{Ave}(y_i : x_i \in N_k(x))$$

approximates the regression function $E(Y|X = x)$.



ESL, Figure 6.1.

As x moves from left to right, $N_k(x)$ changes. This results in discontinuities in $\hat{f}(x)$. A *weighed average* naturally solves this problem.

Kernel smoothers

Given a function $K : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, we can construct the estimator:

$$\hat{f}(x) = \frac{\sum_{i=1}^n K(x, x_i) y_i}{\sum_{i=1}^n K(x, x_i)}.$$

Kernel smoothers

Given a function $K : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, we can construct the estimator:

$$\hat{f}(x) = \frac{\sum_{i=1}^n K(x, x_i) y_i}{\sum_{i=1}^n K(x, x_i)}.$$

We usually:

- Use a kernel that decays at some rate (to give more weight to local observations).
- Work with a parametrized family of kernels $K_\lambda(x, y)$, where λ controls the *window size*.
- Known as the Nadaraya–Watson estimator.

Kernel smoothers

Given a function $K : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, we can construct the estimator:

$$\hat{f}(x) = \frac{\sum_{i=1}^n K(x, x_i) y_i}{\sum_{i=1}^n K(x, x_i)}.$$

We usually:

- Use a kernel that decays at some rate (to give more weight to local observations).
- Work with a parametrized family of kernels $K_\lambda(x, y)$, where λ controls the *window size*.
- Known as the Nadaraya–Watson estimator.

For example, the *Epanechnikov* quadratic kernel is given by

$$K_\lambda(x, x') = D\left(\frac{|x - x'|}{\lambda}\right),$$

where

$$D(t) := \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Resulting prediction function is continuous.

A few remarks:

1. More generally, one can use an adaptive neighborhood: let $h(x_i)$ determine the width of the neighborhood at x_i. Then one can use
$$K(x, x') = D\left(\frac{|x - x'|}{h(x)}\right).$$

A few remarks:

1. More generally, one can use an adaptive neighborhood: let $h(x_i)$ determine the width of the neighborhood at x_i. Then one can use
$$K(x, x') = D\left(\frac{|x - x'|}{h(x)}\right).$$

2. Generally, there are only a few parameters to choose (e.g. only λ in the previous example).

Kernel smoothers (cont.)

A few remarks:

1. More generally, one can use an adaptive neighborhood: let $h(x_i)$ determine the width of the neighborhood at x_i. Then one can use
$$K(x, x') = D\left(\frac{|x - x'|}{h(x)}\right).$$
2. Generally, there are only a few parameters to choose (e.g. only λ in the previous example).
3. The models require little or no training; all the work gets done at evaluation time.

A few remarks:

1. More generally, one can use an adaptive neighborhood: let $h(x_i)$ determine the width of the neighborhood at x_i. Then one can use
$$K(x, x') = D\left(\frac{|x - x'|}{h(x)}\right).$$
2. Generally, there are only a few parameters to choose (e.g. only λ in the previous example).
3. The models require little or no training; all the work gets done at evaluation time.
4. The model, however, is the entire training data set.

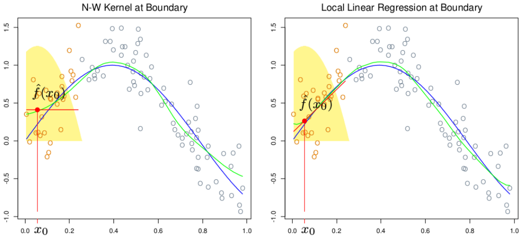
Kernel smoothers (cont.)

A few remarks:

1. More generally, one can use an adaptive neighborhood: let $h(x_i)$ determine the width of the neighborhood at x_i. Then one can use
$$K(x, x') = D\left(\frac{|x - x'|}{h(x)}\right).$$
2. Generally, there are only a few parameters to choose (e.g. only λ in the previous example).
3. The models require little or no training; all the work gets done at evaluation time.
4. The model, however, is the entire training data set.
5. Non-parametric approach.

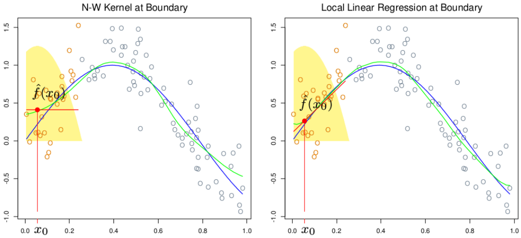
Local linear regression

Kernel smoothers can have poor performance near the boundary of the domain or in regions with very little observations.



ESL, Figure 6.3.

Kernel smoothers can have poor performance near the boundary of the domain or in regions with very little observations.

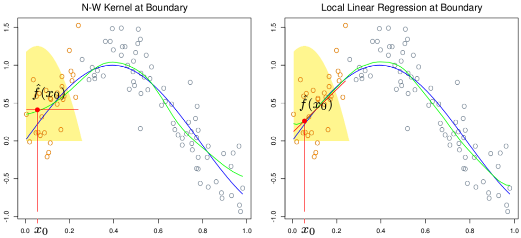


ESL, Figure 6.3.

Locally weighted regression solves a separate weighted least squares problem at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{n} K(x_0, x_i)[y - \alpha(x_0) - \beta(x_0)x_i]^2.$$

Kernel smoothers can have poor performance near the boundary of the domain or in regions with very little observations.



ESL, Figure 6.3.

Locally weighted regression solves a separate weighted least squares problem at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{n} K(x_0, x_i)[y - \alpha(x_0) - \beta(x_0)x_i]^2.$$

The estimate is then

$$\hat{f}(x_0) = \alpha(x_0) + \beta(x_0)x_0.$$

Local linear regression (cont.)

- Obtaining the solution is not harder than usual.
- More generally, note that for $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, and

$w = (w_i) \in (0, \infty)^n$,

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} w_i (y_i - x_i^T \beta)^2 \quad \Leftrightarrow \quad \min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (\tilde{y}_i - \tilde{x}_i^T \beta)^2,$$

where $\tilde{y}_i := \sqrt{w_i} y_i$ and $\tilde{x}_i = \sqrt{w_i} x_i$.

Local linear regression (cont.)

- Obtaining the solution is not harder than usual.
- More generally, note that for $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, and $w = (w_i) \in (0, \infty)^n$,

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} w_i (y_i - x_i^T \beta)^2 \quad \Leftrightarrow \quad \min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (\tilde{y}_i - \tilde{x}_i^T \beta)^2,$$

where $\tilde{y}_i := \sqrt{w_i} y_i$ and $\tilde{x}_i = \sqrt{w_i} x_i$.

- Letting $W = \mathrm{diag}(w_1, \ldots, w_n)$, we have

$$\tilde{y} = \sqrt{W} y, \qquad \tilde{X} = \sqrt{W} X.$$

Local linear regression (cont.)

- Obtaining the solution is not harder than usual.
- More generally, note that for $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, and

$w = (w_i) \in (0, \infty)^n$,

$$\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} w_i (y_i - x_i^T \beta)^2 \quad \Leftrightarrow \quad \min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} (\tilde{y}_i - \tilde{x}_i^T \beta)^2,$$

where $\tilde{y}_i := \sqrt{w_i} y_i$ and $\tilde{x}_i = \sqrt{w_i} x_i$.

- Letting $W = \mathrm{diag}(w_1, \ldots, w_n)$, we have

$$\tilde{y} = \sqrt{W} y, \qquad \tilde{X} = \sqrt{W} X.$$

So the solution is:

$$\hat{\beta} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X} \tilde{y} = (X^T W X)^{-1} X^T W y.$$

Local linear regression (cont.)

In the case of local linear regression, the weights are:

$$w_i = w_i(x_0) = K_\lambda(x_0, x_i), \qquad (i = 1, \ldots, n).$$

In the case of local linear regression, the weights are:

$$w_i = w_i(x_0) = K_\lambda(x_0, x_i), \qquad (i = 1, \ldots, n).$$

The prediction at x_0 becomes:

$$\begin{aligned}\hat{f}(x_0) &= x_0^T (X^T W(x_0) X)^{-1} X^T W(x_0) y \\ &= \sum_{i=1}^{n} l_i(x_0) y_i.\end{aligned}$$

In the case of local linear regression, the weights are:

$$w_i = w_i(x_0) = K_\lambda(x_0, x_i), \qquad (i = 1, \ldots, n).$$

The prediction at x_0 becomes:

$$\begin{aligned}\hat{f}(x_0) &= x_0^T (X^T W(x_0) X)^{-1} X^T W(x_0) y \\ &= \sum_{i=1}^{n} l_i(x_0) y_i.\end{aligned}$$

Note: We need to solve a linear regression problem at **every** x_0 where the estimator has to be evaluated.

Local linear regression (cont.)

In the case of local linear regression, the weights are:

$$w_i = w_i(x_0) = K_\lambda(x_0, x_i), \qquad (i = 1, \ldots, n).$$

The prediction at x_0 becomes:

$$\begin{aligned}\hat{f}(x_0) &= x_0^T (X^T W(x_0) X)^{-1} X^T W(x_0) y \\ &= \sum_{i=1}^{n} l_i(x_0) y_i.\end{aligned}$$

Note: We need to solve a linear regression problem at **every** x_0 where the estimator has to be evaluated.

Remark:

1. Estimate is still linear in y.
2. The weights $l_i(x_0)$ combine the weighting kernels $K_\lambda(x_0, x_i)$, and the least squares operations.
3. Same ideas can be applied to local regression with other function bases (e.g. local polynomial regression, see ESL 6.1.2).

Local linear regression - higher dimension

The same ideas apply to higher dimension. Given $K_\lambda : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, one can solve:

$$\min_{\beta(x_0) \in \mathbb{R}^p} \sum_{i=1}^n K(x_0, x_i)[y_i - x_i^T \beta]^2.$$

Local linear regression - higher dimension

The same ideas apply to higher dimension. Given $K_\lambda : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, one can solve:

$$\min_{\beta(x_0) \in \mathbb{R}^p} \sum_{i=1}^n K(x_0, x_i)[y_i - x_i^T \beta]^2.$$

For example, one can use a *radial Epanechnikov* kernel:

$$K_\lambda(x, x') = D\left(\frac{\|x - x'\|}{\lambda}\right).$$

(Note: better to scale predictors)

Local linear regression - higher dimension

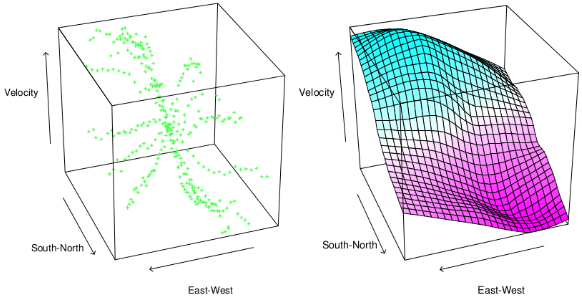
The same ideas apply to higher dimension. Given $K_\lambda : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$, one can solve:

$$\min_{\beta(x_0) \in \mathbb{R}^p} \sum_{i=1}^n K(x_0, x_i)[y_i - x_i^T \beta]^2.$$

For example, one can use a *radial Epanechnikov* kernel:

$$K_\lambda(x, x') = D\left(\frac{\|x - x'\|}{\lambda}\right).$$

(Note: better to scale predictors)



ESL, Figure 6.8. Local linear regression smoothing for velocity of a galaxy data.

Structured local linear regression models

- When the sample size is small compared to the dimension, local linear regression may not perform well.
- As we did before, we can impose more constraints on the model (i.e., add more structure).
- For example, we can weight dimensions differently.

Structured local linear regression models

- When the sample size is small compared to the dimension, local linear regression may not perform well.
- As we did before, we can impose more constraints on the model (i.e., add more structure).
- For example, we can weight dimensions differently.

Structured kernels: use a positive semidefinite matrix A to weight the coordinates:

$$K_{\lambda,A}(x,x') = D\left(\frac{(x-x')^T A(x-x')}{\lambda}\right).$$

For example, A could be a diagonal matrix that assigns different weights to different dimensions.

Structured local linear regression models

- When the sample size is small compared to the dimension, local linear regression may not perform well.
- As we did before, we can impose more constraints on the model (i.e., add more structure).
- For example, we can weight dimensions differently.

Structured kernels: use a positive semidefinite matrix A to weight the coordinates:

$$K_{\lambda,A}(x,x') = D\left(\frac{(x-x')^T A(x-x')}{\lambda}\right).$$

For example, A could be a diagonal matrix that assigns different weights to different dimensions.

Structured Regression Functions, Local Likelihood methods, etc. (see ESL).