

Kernel Smoothing

Lecture 23

Parametric vs non-parametric methods

Kernel **density estimation** / smoothing

Kernel density **classification**

Kernel **regression**

Kernel (first definition):
a non-negative function that
integrates to one

A **window** or weighting
function for **local**
computation / neighborhood
methods

Kernel methods and the **Kernel Trick**
(a preamble to support vector machines)

Kernel (second definition):
A **similarity** function

Inner products in some
feature space

Parametric Methods

Models governed by a vector \mathbf{w} of parameters

During training \mathbf{w} is estimated

Training data are discarded and predictions made only with \mathbf{w}

Number of parameters is fixed

Typically slower to train, faster to predict

Example: linear regression

$$\hat{f}(\mathbf{x}) = \sum_{i=0}^p \mathbf{w}_i x_i$$

Model weights learned from training data

Non-Parametric Methods

All training data (or a subset) are used in the prediction phase

These are **memory-based** or **instance-based** methods

Typically faster to train, slower to predict

Number of parameters (may) vary with the size of the training data

Example: k-nearest neighbor regression

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} y_i$$

where

$\mathcal{N}_k(\mathbf{x}) \triangleq \{\text{k nearest neighbors}\}$

Training data

Bishop, Pattern Recognition and Machine Learning, 2006

Supervised Learning Techniques

Covered so far

- Linear Regression
- K-Nearest Neighbors
- Perceptron
- Logistic Regression
- Fisher's Linear Discriminant
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Naïve Bayes
- Decision Trees and Random Forests
- Ensemble methods (bagging, boosting, stacking)
- Neural Networks

Appropriate for:
● Classification
● Regression

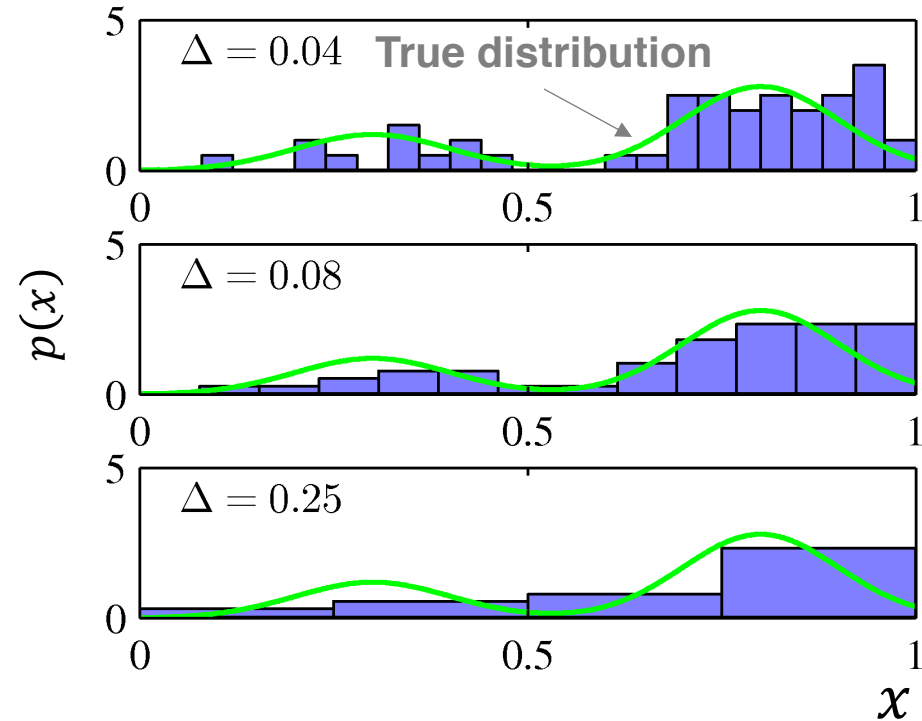
Can be used with many machine learning techniques

Non-Parametric Supervised Learning Techniques

| | | | |
|----------------|----|--|---|
| Covered so far | ● | Linear Regression | Appropriate for: |
| | ●● | K-Nearest Neighbors | ● Classification |
| | ● | Perceptron | ● Regression |
| | ● | Logistic Regression | |
| | ● | Fisher's Linear Discriminant | |
| | ● | Linear Discriminant Analysis | |
| | ● | Quadratic Discriminant Analysis | |
| | ● | Naïve Bayes | |
| | ●● | Decision Trees and Random Forests | |
| | ●● | Ensemble methods (bagging, boosting, stacking) | Can be used with many machine learning techniques |
| | ●● | Neural Networks | |

Histogram Density Estimation

Histogram



$$p(x) = \frac{n_i}{N\Delta_i}$$

n_i = # observations of x falling in bin i

N = total # observations

Δ_i = width of bin i

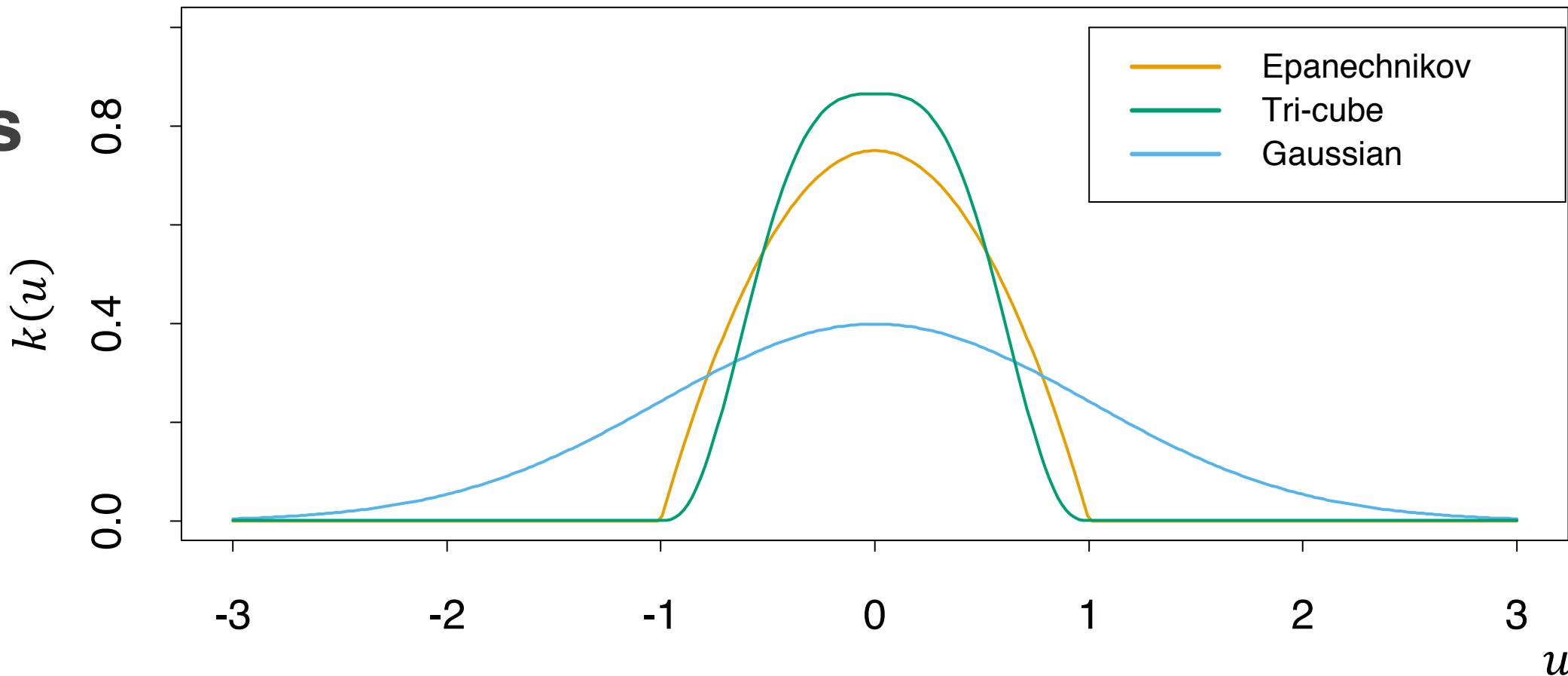
Highly dependent on the choice of bin width, Δ_i

Has discontinuities at the bin edges

Local neighborhoods do appear to be helpful

Kernel Functions

(window kernels)



Satisfy two properties:

$$k(u) \geq 0$$
$$\int k(u) du = 1$$

Epanechnikov

$$k(u) = \frac{3}{4} (1 - u^2)$$
$$|u| \leq 1$$

Tri-cube

$$k(u) = \frac{70}{81} (1 - |u^3|)^3$$
$$|u| \leq 1$$

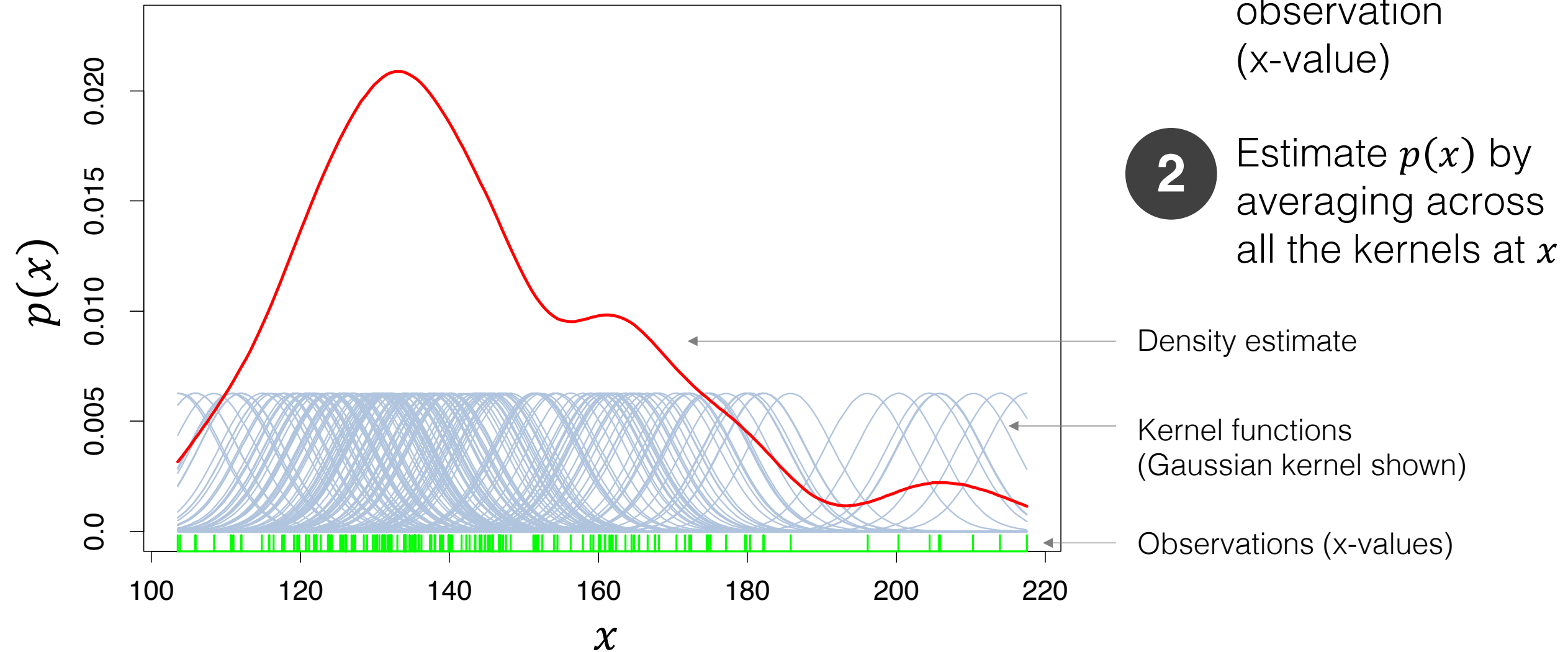
Gaussian

$$k(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$$
$$-\infty < u < \infty$$

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Density Estimation

a. k. a. Parzen Window Density Estimation



Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Density Estimation

Center the kernel function at each x-value in the dataset:

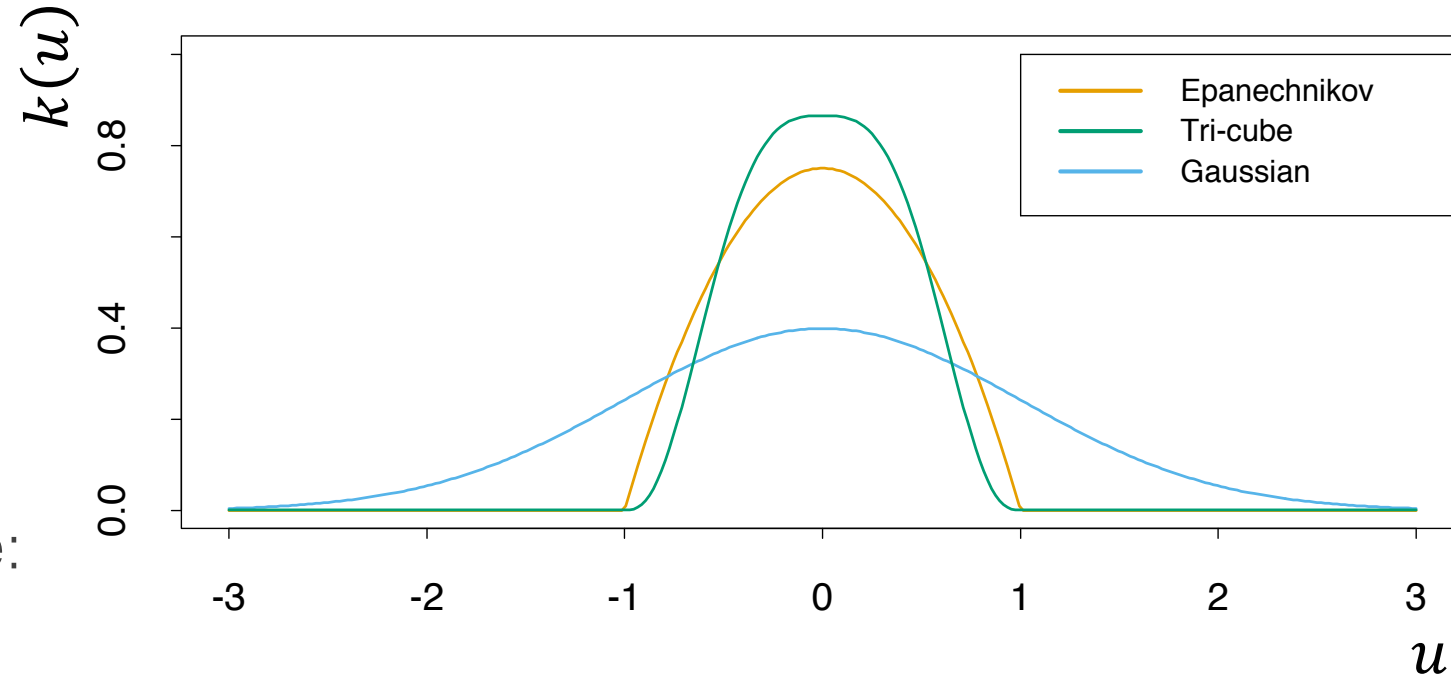
$$k(x - x_n) \quad n = 1, 2, \dots, N$$

Average over all of the kernel functions to get the density estimate:

$$p(x) = \frac{1}{N} \sum_{n=1}^N k(x - x_n)$$

Note: we can scale the width of the kernel function with a scale factor, h :

$$k\left(\frac{x - x_n}{h}\right)$$

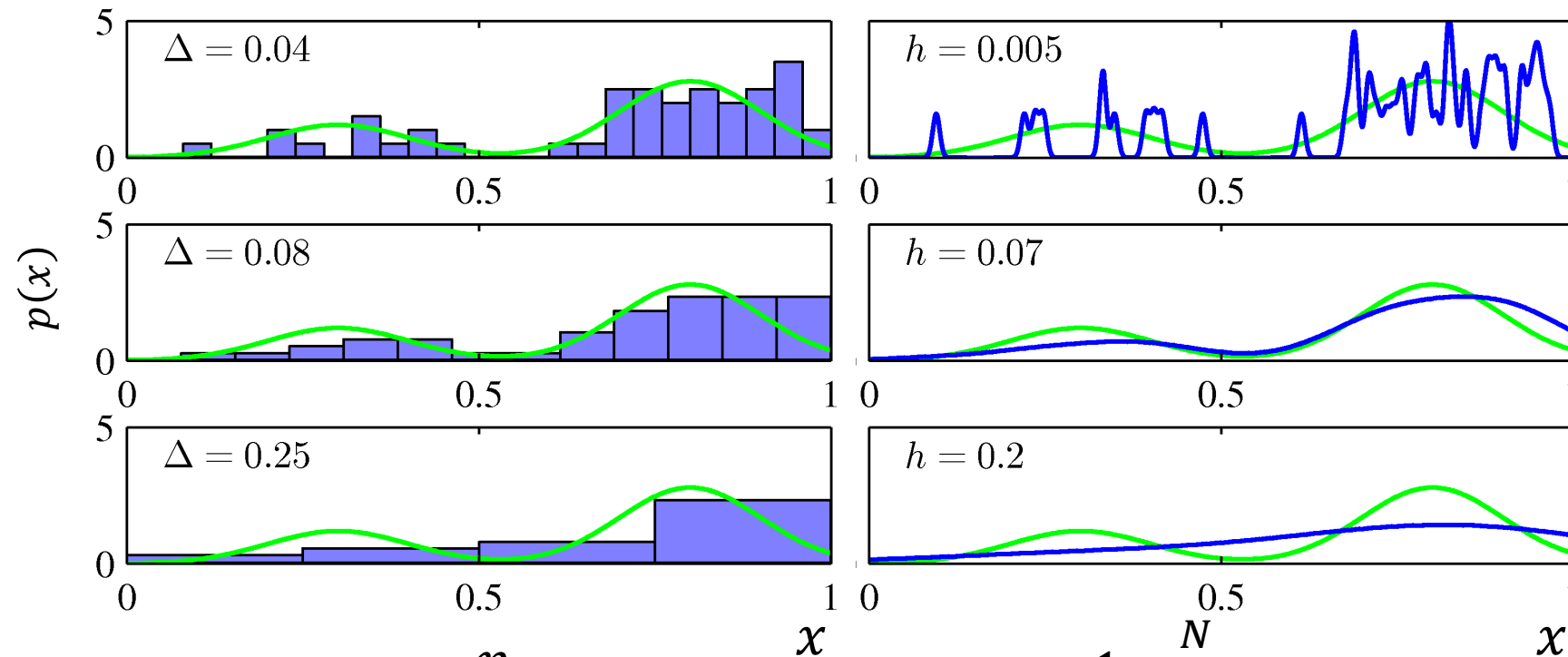


For kernel functions with **finite domains**, this means that each observation, x , will only affect the density estimate in a **neighborhood** close to the center of the kernel

Kernel Density Estimation

Histogram

Kernel Density Estimation



$$p(x) = \frac{n_i}{N\Delta_i}$$

$$p(x) = \frac{1}{Nh} \sum_{n=1}^N k\left(\frac{x - x_n}{h}\right)$$

n_i = # observations of x falling in bin i
 N = total # observations
 Δ_i = width of bin i

x_n = The n^{th} observation of x
 k = kernel function
 h = width of the kernel

Requires tuning h , the kernel width parameter

Computational cost of evaluating this density grows linearly with the size of the data

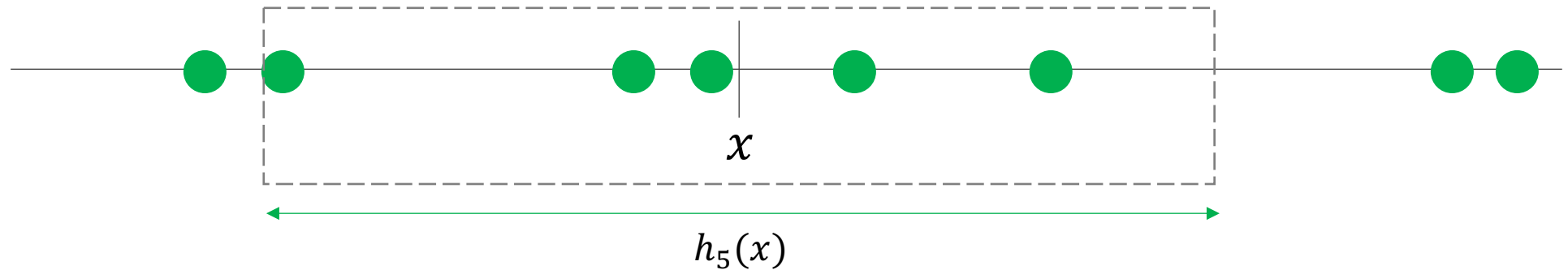
Bishop, Pattern Recognition and Machine Learning, 2006

Nearest Neighbor Density Estimation

Instead of fixing the window width and averaging the points within, we **fix the number of points** and **vary the window width**

- 1 For a point, x , determine the width of a window, $h_K(x)$, centered at x , that is just large enough to contain K points

Example for $k = 5$:



- 2 The density estimate is computed as:
$$p(x) = \frac{K}{Nh_K(x)}$$

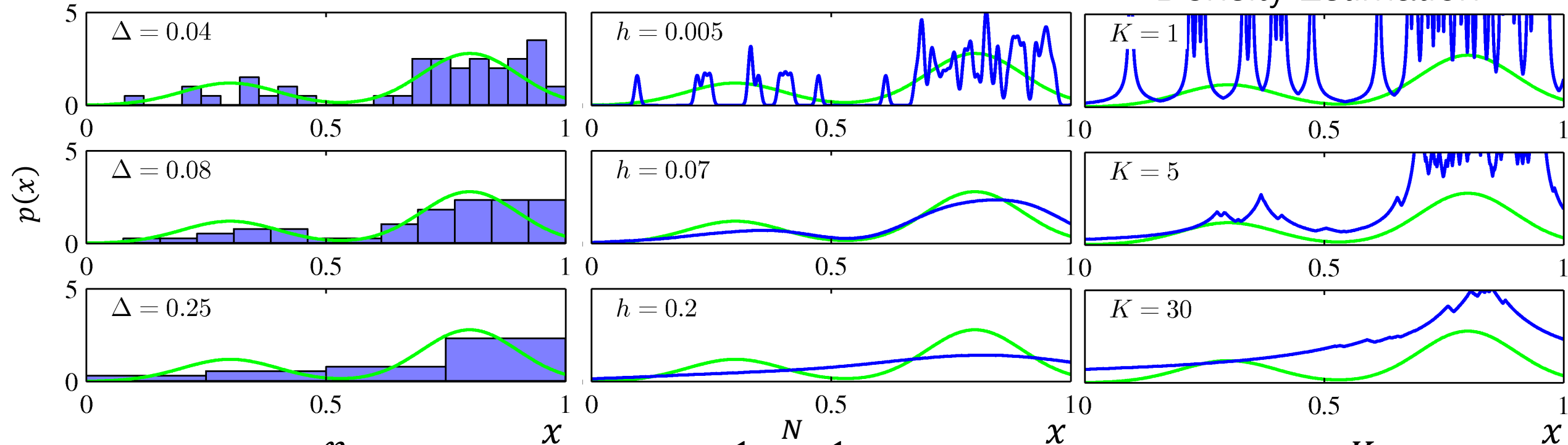
N = total number of observations

Kernel Density Estimation

Histogram

Kernel Density Estimation

Nearest Neighbor
Density Estimation



$$p(x) = \frac{n_i}{N\Delta_i}$$

$$p(x) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h} k\left(\frac{x - x_n}{h}\right)$$

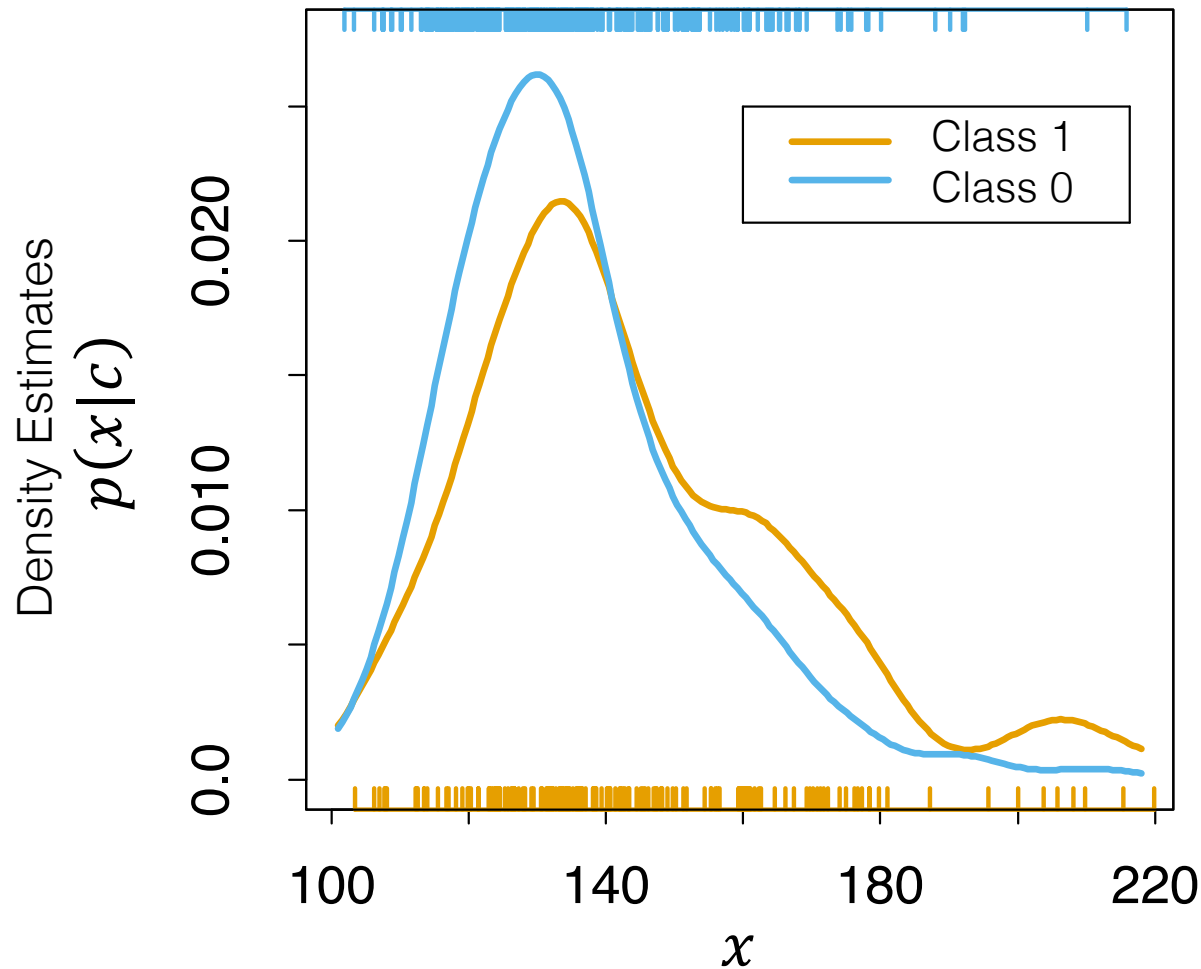
$$p(x) = \frac{K}{Nh_K(x)}$$

n_i = # observations of x falling in bin i
 N = total # observations
 Δ_i = width of bin i

x_n = The n^{th} observation of x
 k = kernel function
 h = width of the kernel

K = # points in neighborhood
 $h_K(x)$ = minimum width containing K observations

Kernel Density Classification



$p(x|c)$ = class conditional distribution
where c represents the class

Bayes' Rule
$$p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

Since there are two classes:

$$p(x) = p(x|c_1)p(c_1) + p(x|c_0)p(c_0)$$

Therefore, we have:

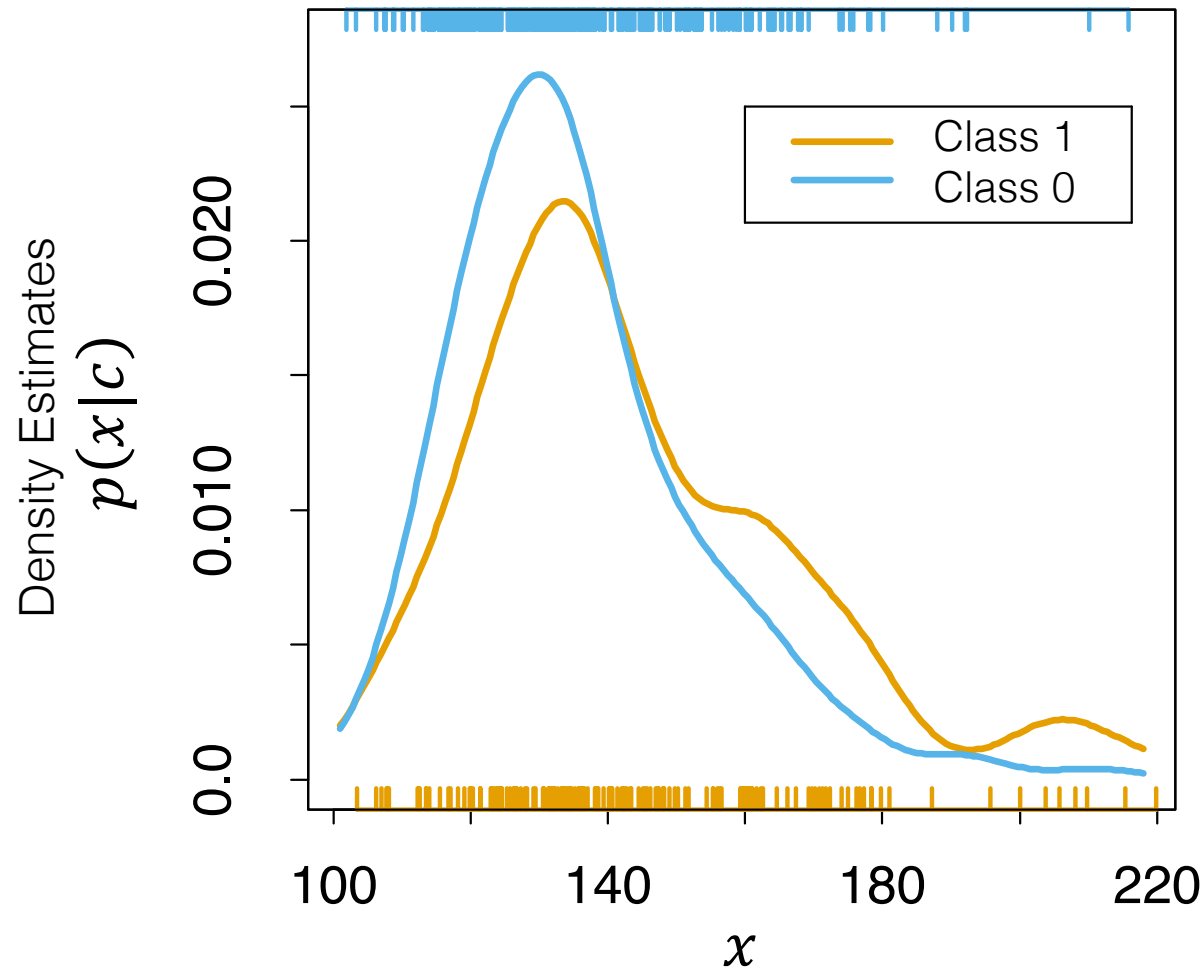
$$p(c|x) = \frac{p(x|c)p(c)}{p(x|c_1)p(c_1) + p(x|c_0)p(c_0)}$$

Class priors, $p(c)$, could simply be the fraction of observations from each class

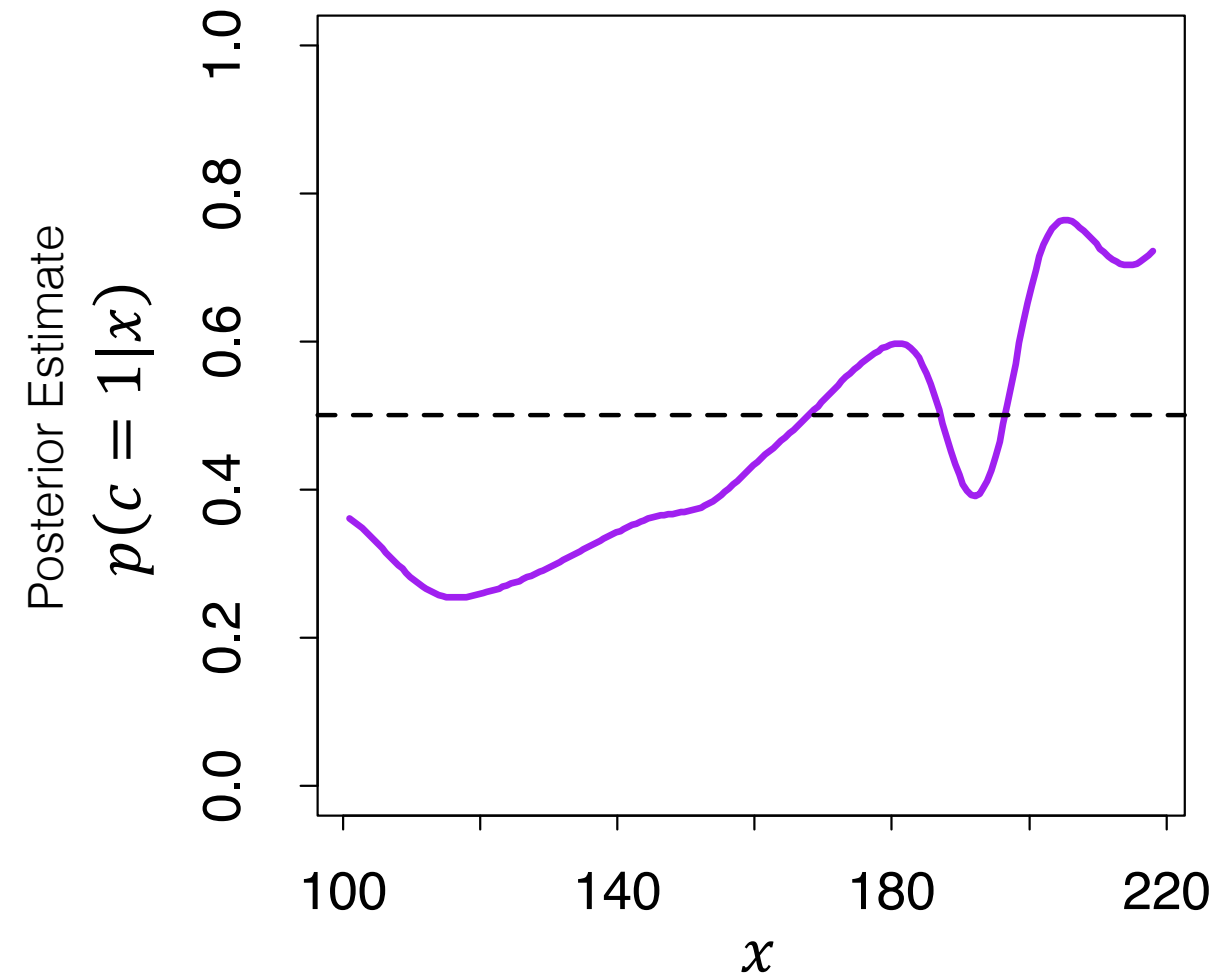
We can construct a classifier:

Predict class $c = \hat{f}(x) = \arg \max_c p(c|x)$

Kernel Density Classification



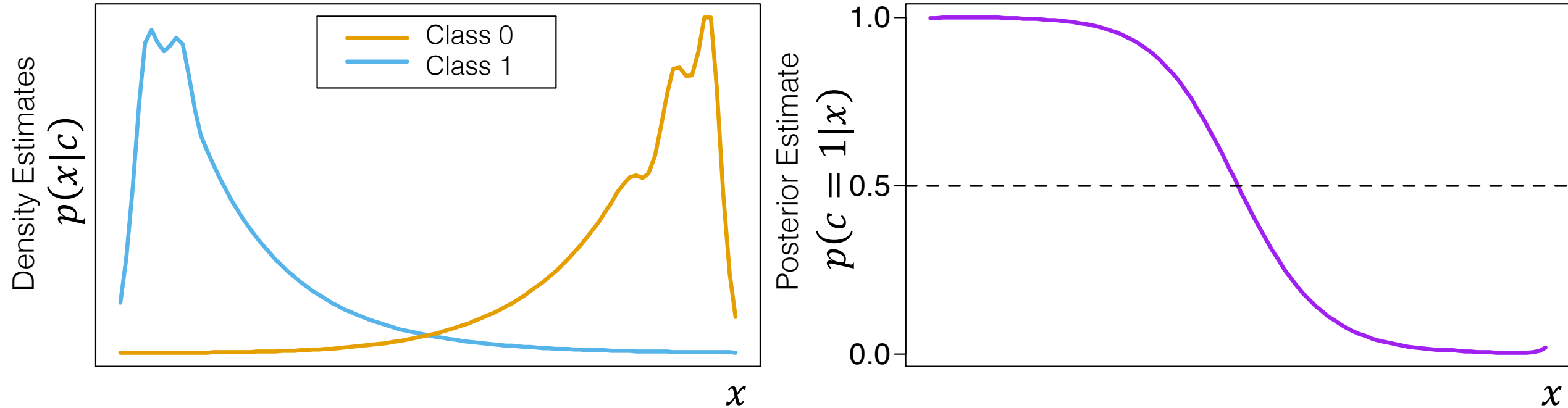
$p(x|c)$ = class conditional distribution
where c represents the class



$p(c)$ = class prior (# of samples from
that class in training set)

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Density Classification



We may not need all of the interesting structure of each class density, but will only need to estimate the posterior near the decision boundary (or boundaries)

Kernel Density Classification

...and recall Naïve Bayes

Assumption: Given the class, the features are independent

$$P(y = i | x_1, x_2, \dots, x_D) \propto P(y = i) \prod_{j=1}^D P(x_j | y = i)$$

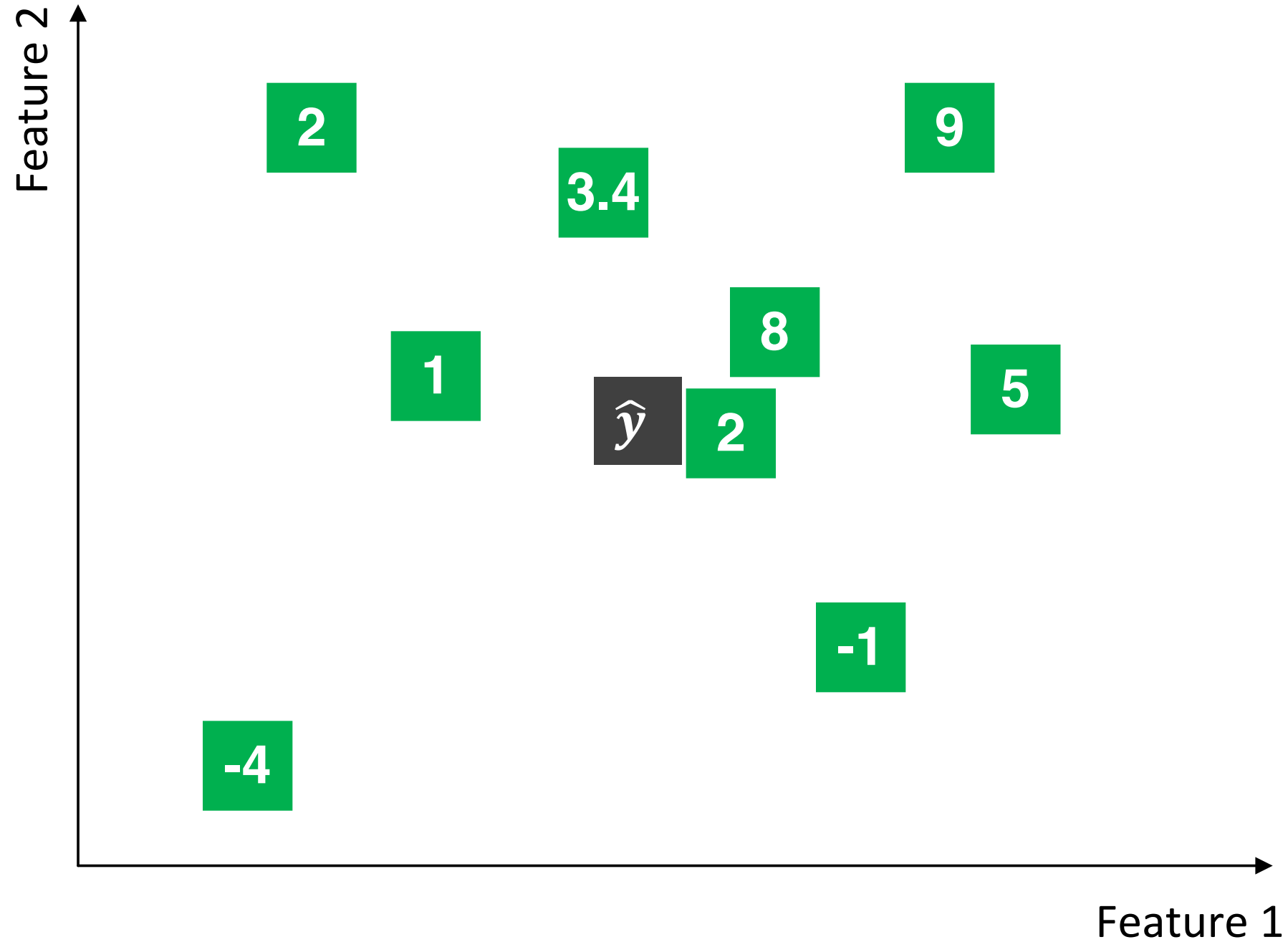
This implies we estimate the density of each feature **separately**

We can easily apply Naïve Bayes using kernel density estimates for each feature, then multiply them together

Kernel Regression

(and smoothing)

K Nearest Neighbor Regression



K Nearest Neighbor Regression

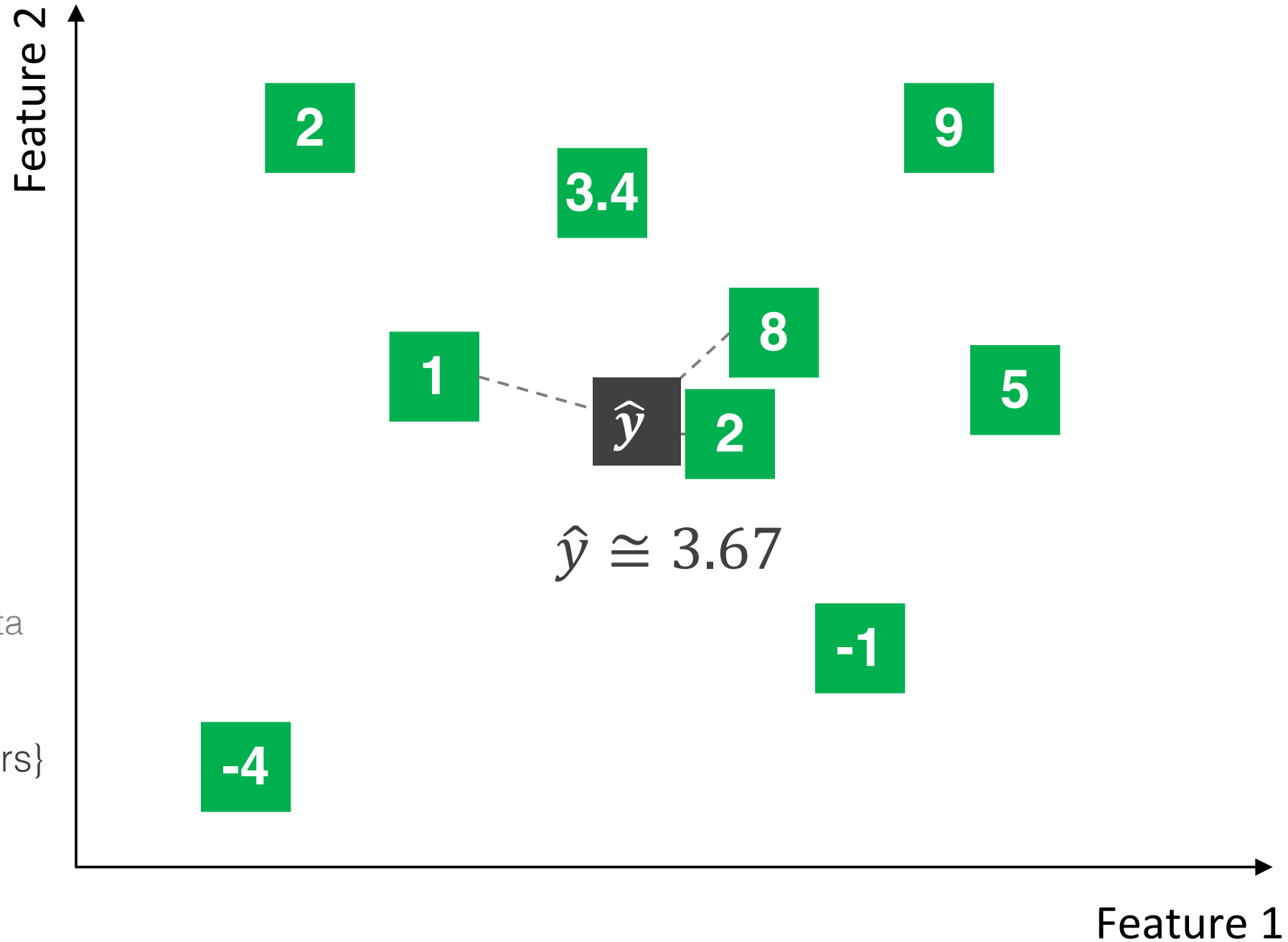
Example for $k = 3$

$$\hat{y} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i$$

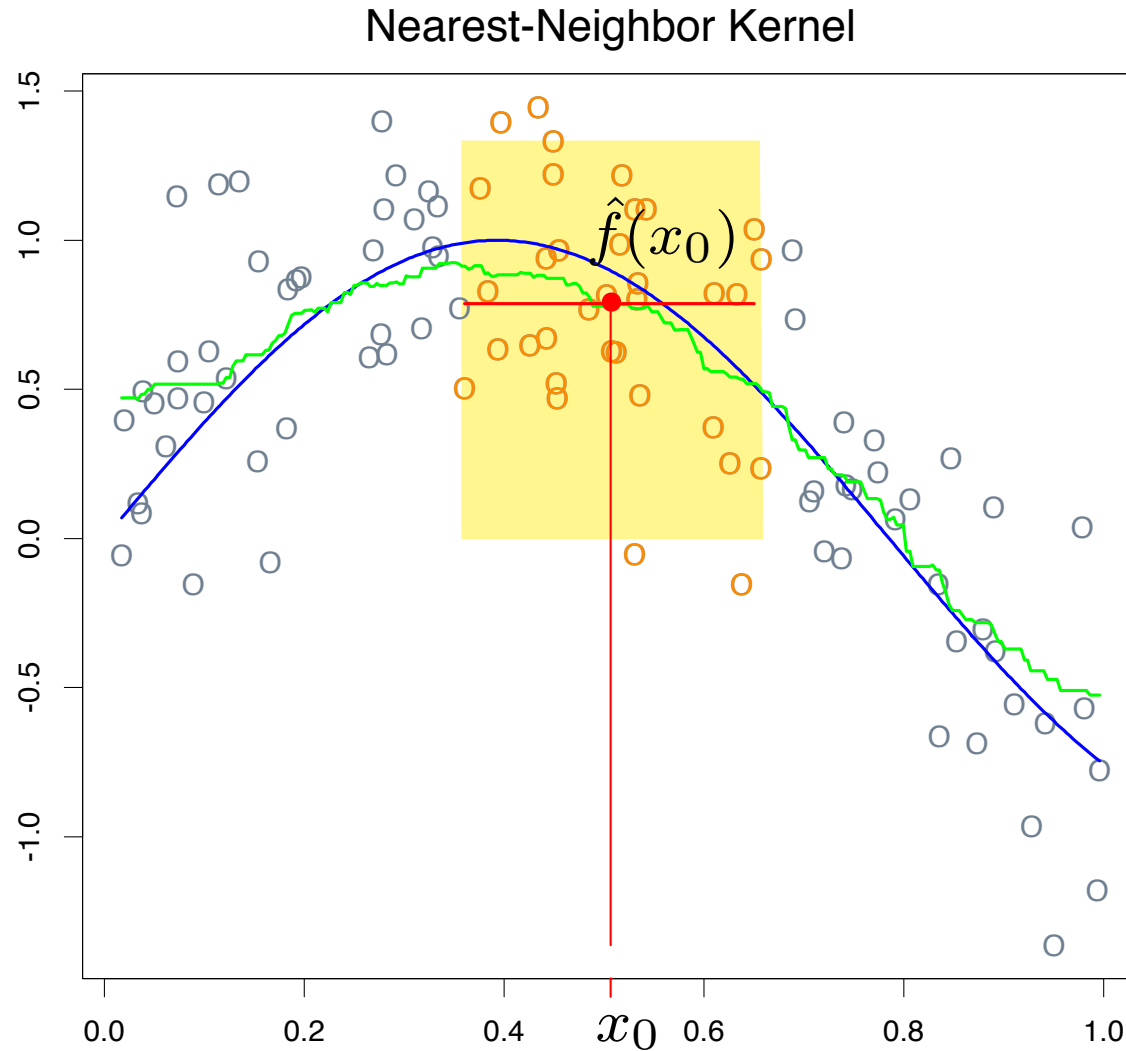
Training data

where

$\mathcal{N}_k(x) \triangleq \{k \text{ nearest neighbors}\}$



Kernel Regression



This produces a bumpy, discontinuous estimate, $\hat{f}(x)$

This is also a **local estimate** of the average of this function

This discontinuity can be avoided by using a **smoothing kernel**

Target function (blue); prediction (green); observations (circles); observations included in estimate of $\hat{f}(x_0)$ (orange circles)

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

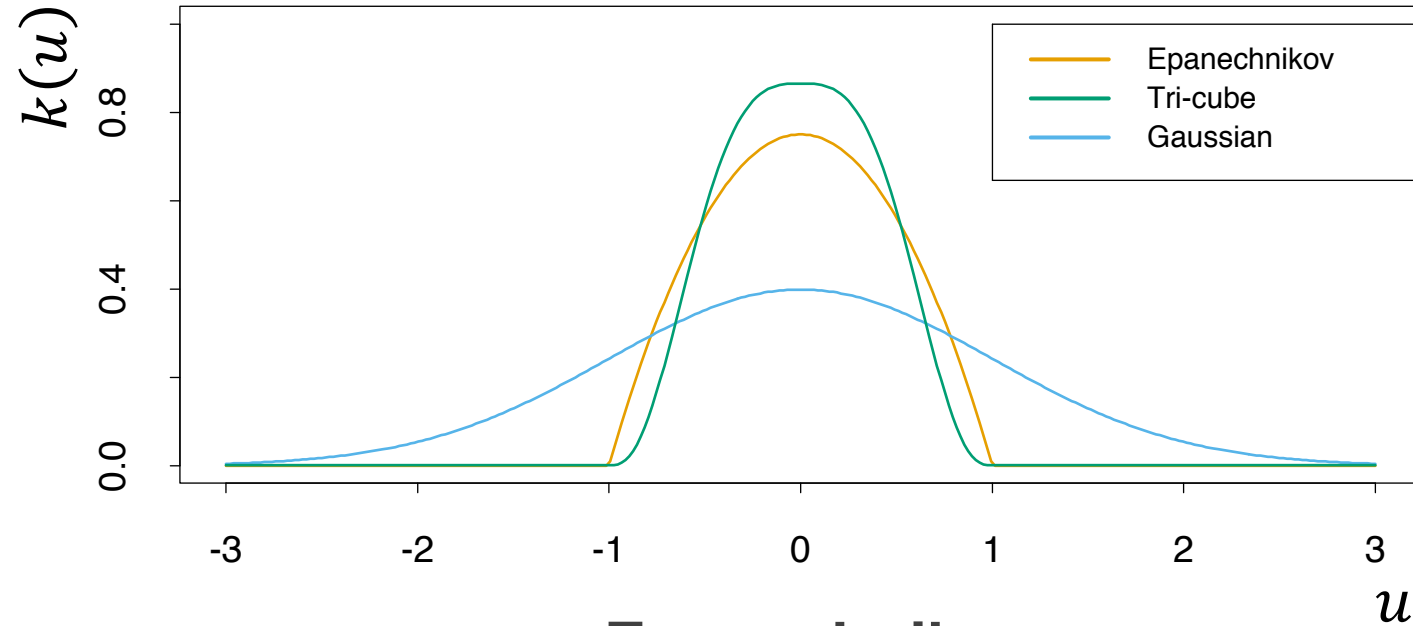
Nadaraya-Watson Kernel Regression

Local kernel weighted
average regression

Weight the average for a given x
value, by its neighbors, as defined
by your choice of kernel

$$\hat{f}(x) = \frac{\sum_{i=1}^N k(x - x_i) y_i}{\sum_{i=1}^N k(x - x_i)}$$

For kernels with a finite domain, this
only uses a subset of the training data



Epanechnikov

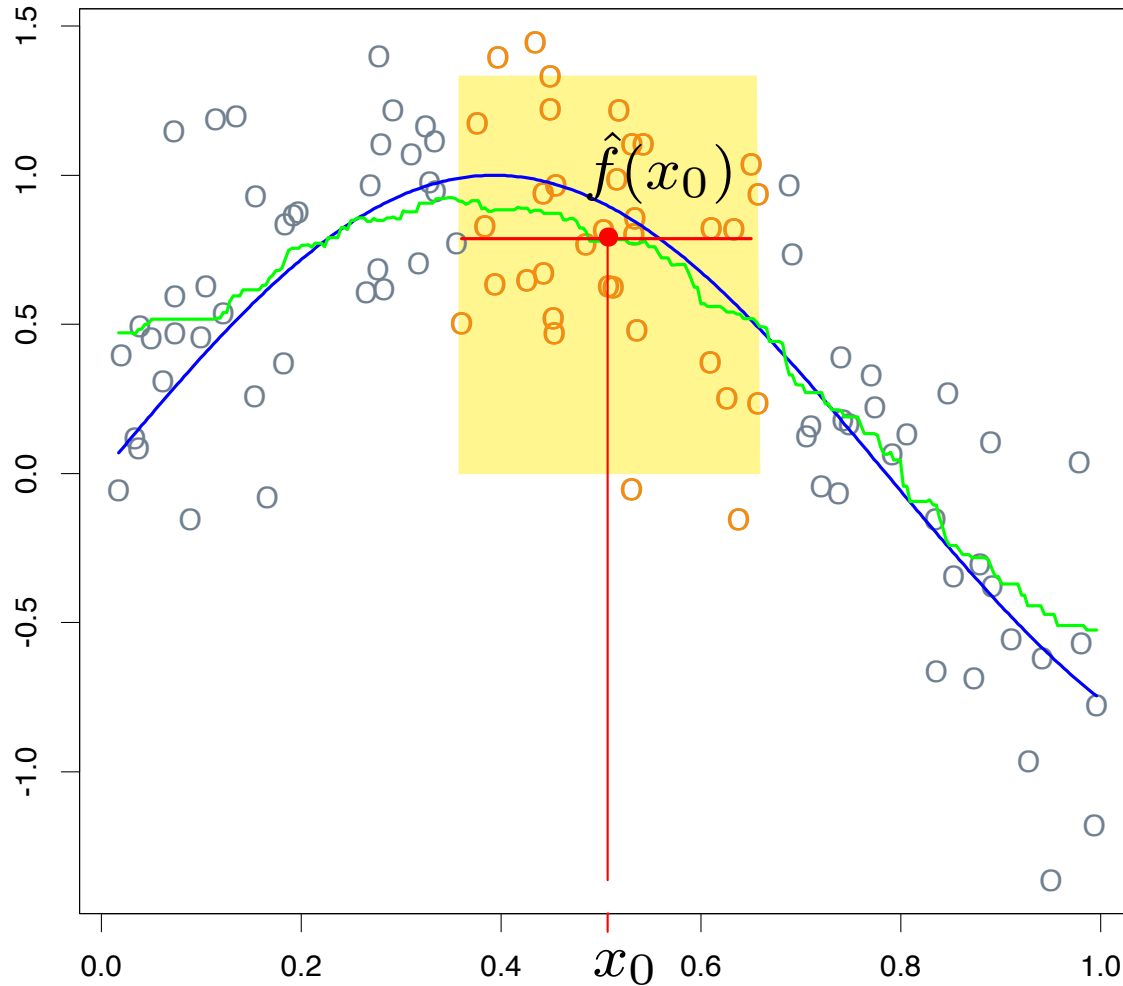
$$k(u) = \frac{3}{4} (1 - u^2)$$

$$|u| \leq 1$$

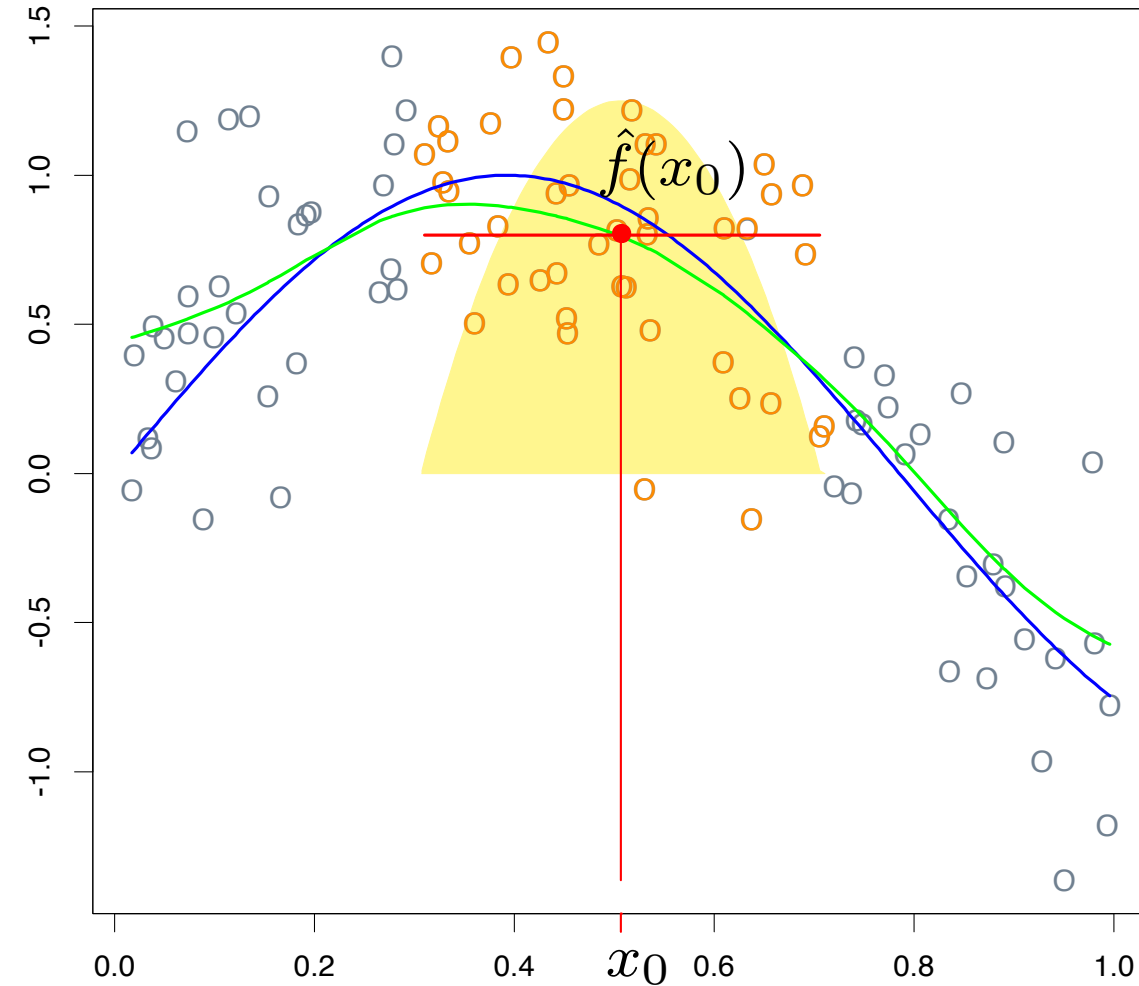
Kernel Regression

Continuous, smooth function

Nearest-Neighbor Kernel



Epanechnikov Kernel

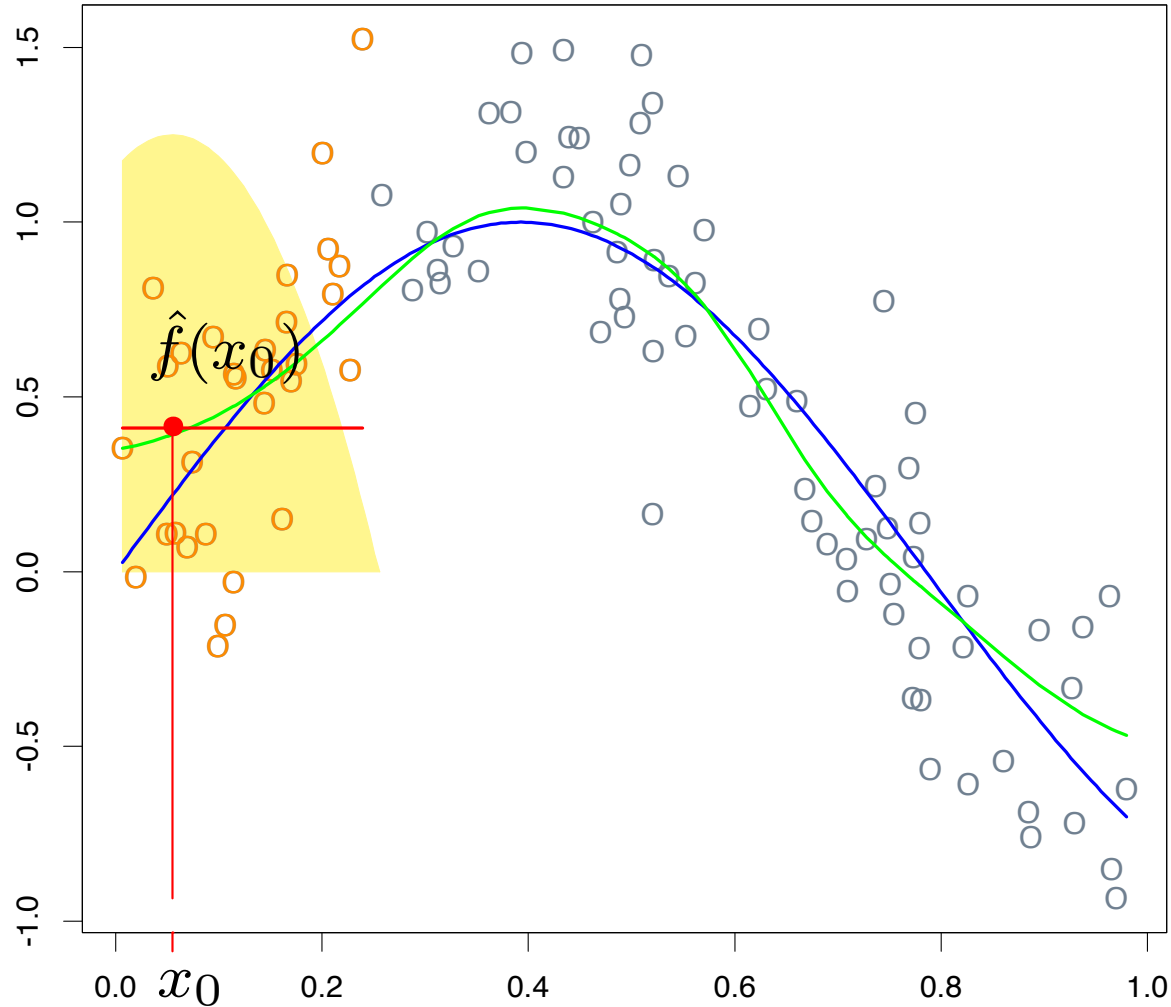


Target function (blue); prediction (green); observations (circles);
observations included in estimate of $\hat{f}(x_0)$ (orange circles)

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Regression

N-W Kernel at Boundary



At the boundaries these estimates can be badly biased due to kernel asymmetry
This can also happen in the interior if the points aren't equally spaced

One solution: fit straight lines instead of constants locally

This leads to **local linear regression**

Target function (blue); prediction (green); observations (circles);
observations included in estimate of $\hat{f}(x_0)$ (orange circles)

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Regression

Local linear regression least squares:

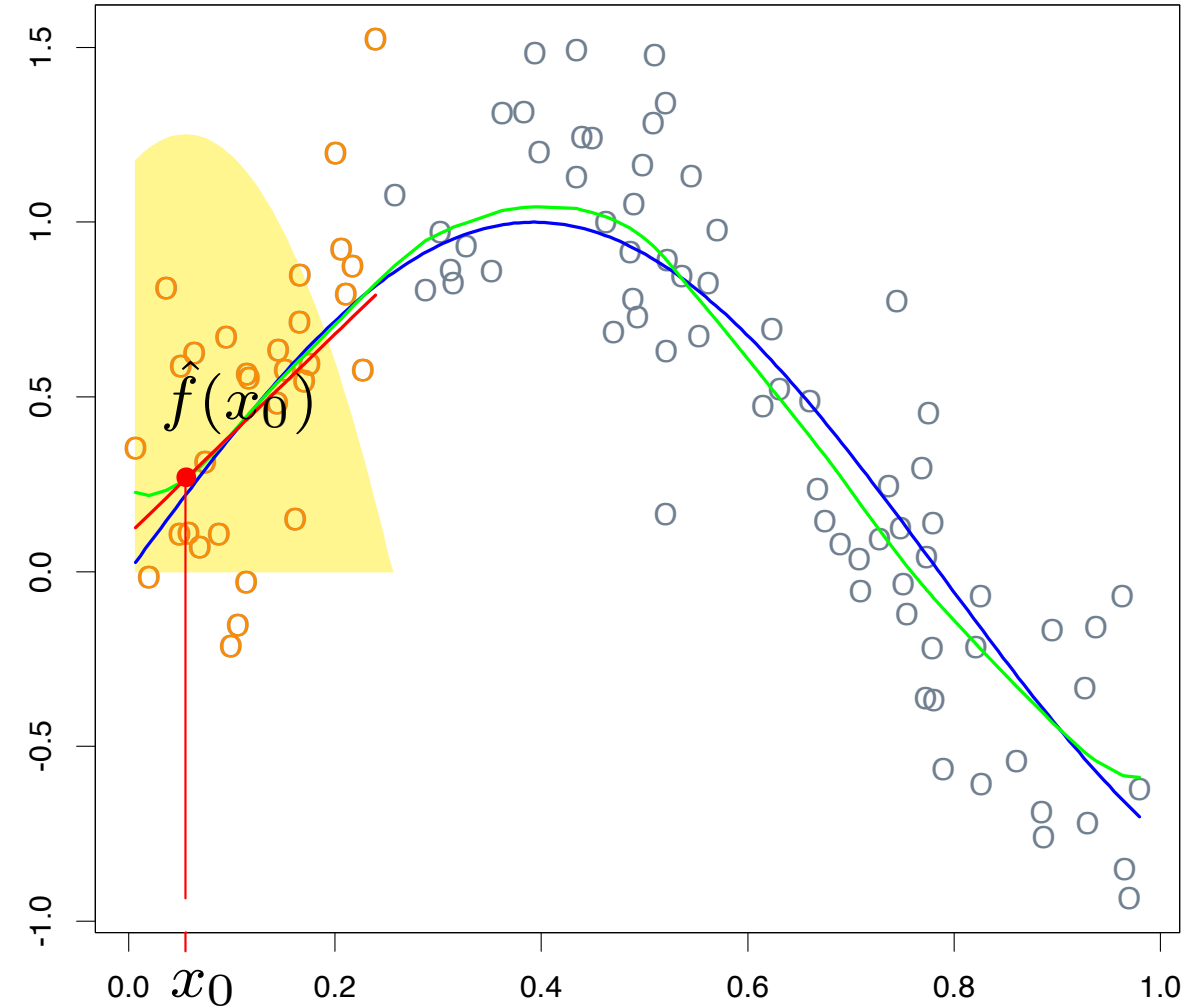
$$\min_{\mathbf{w}(\mathbf{x})} \sum_{i=1}^N \underbrace{k(\mathbf{x} - \mathbf{x}_i)}_{\text{Weighted by kernel}} \underbrace{[y_i - \mathbf{w}(\mathbf{x})^T \mathbf{x}_i]^2}_{\text{Linear regression squared error}}$$

We fit this model on the local data to evaluate a **single point**:

$$\hat{f}(\mathbf{x}_0) = \mathbf{w}(\mathbf{x}_0)^T \mathbf{x}_0$$

Target function (blue); prediction (green); observations (circles); observations included in estimate of $\hat{f}(\mathbf{x}_0)$ (orange circles)

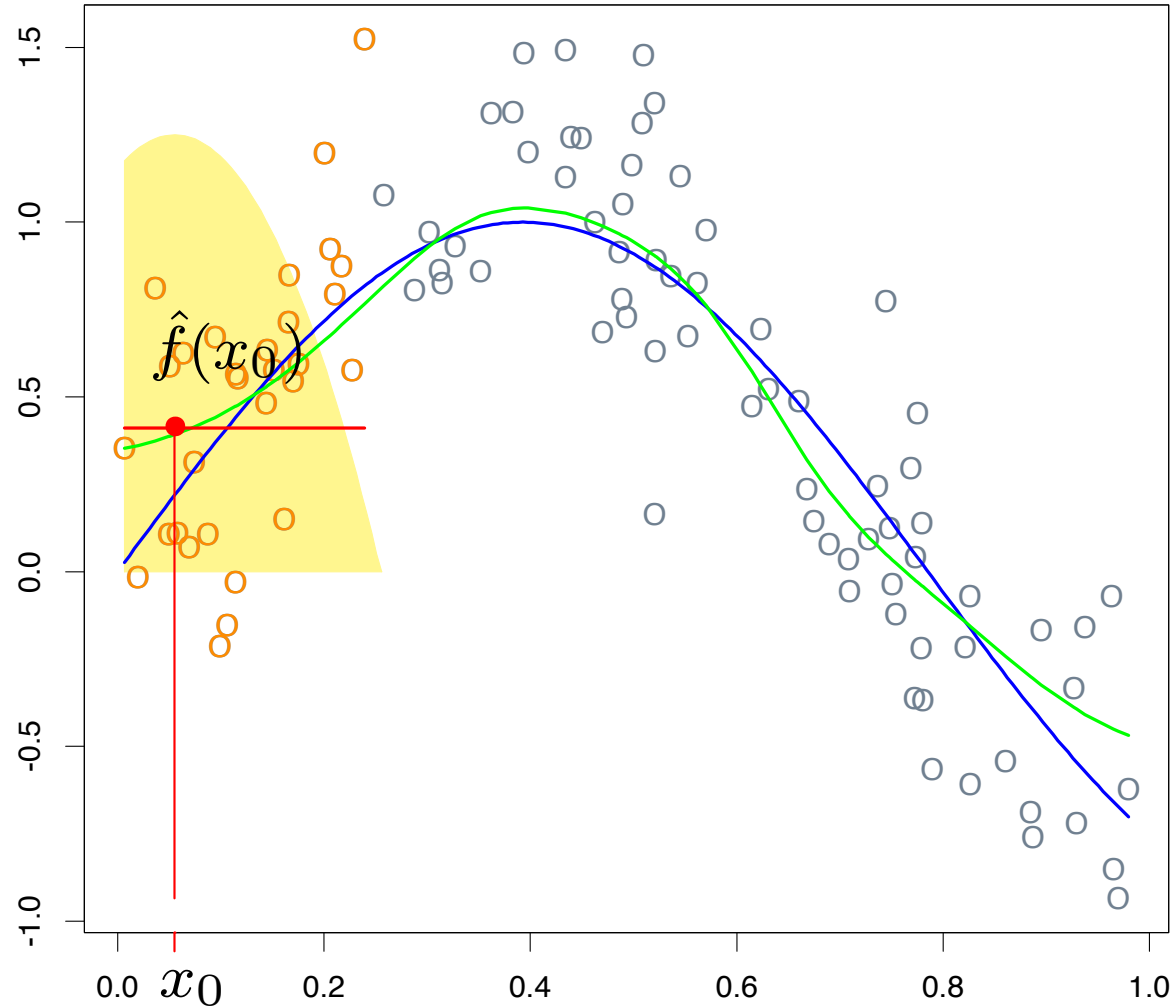
Local Linear Regression at Boundary



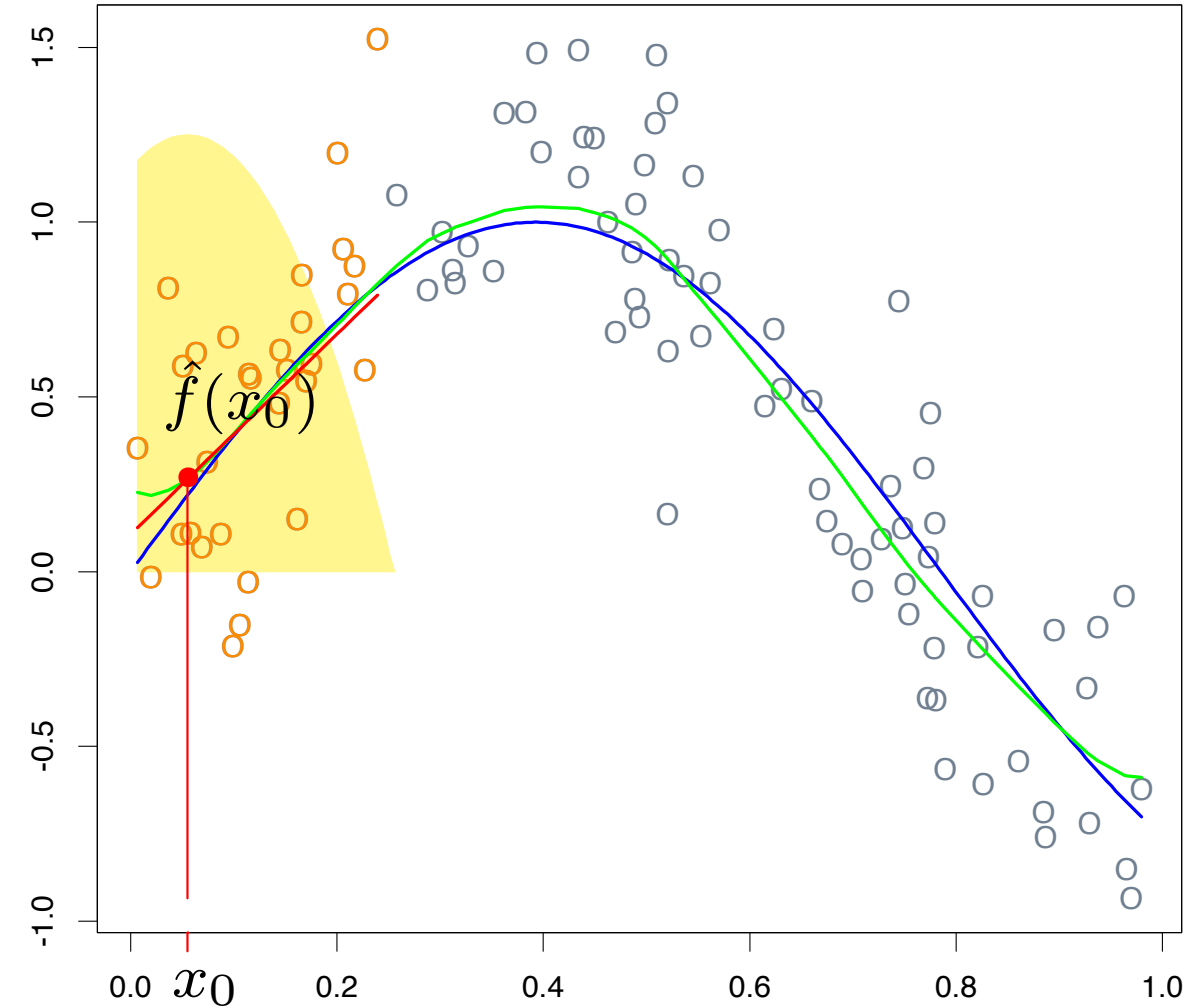
Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Kernel Regression

N-W Kernel at Boundary



Local Linear Regression at Boundary



Target function (blue); prediction (green); observations (circles);
observations included in estimate of $\hat{f}(x_0)$ (orange circles)

Hastie, Tibshirani, and Friedman, The Elements of Statistical learning, 2001

Parametric vs non-parametric methods

Kernel **density estimation** / smoothing

Kernel density **classification**

Kernel **regression**

Kernel methods and the **Kernel Trick**
(a preamble to support vector machines)

Kernel (first definition):
a non-negative function that
integrates to one

A **window** or weighting
function for **local**
computation / neighborhood
methods

Kernel (second definition):
A **similarity** function

Inner products in some
feature space