Non-parametric methods

Introduction to Machine Learning, Ch. 8, E.Alpaydin,

P. Faion, P. Byvshev

University of Osnabrück - Advanced Machine Learning Seminar

Table of contents

1. Introduction

2. Nonparametric Density Estimation

3. Nonparametric Classification

4. Nonparametric Regression

5. Remarks

Introduction

Introduction

Nonparametric Estimation

- Parametric (single global model), semiparametric (small number of local models)
- Nonparametric (no model): Similar inputs have similar outputs
- Density function, discriminant change smoothly
- The data speak for itself
- For a given test X a 'closest' subset in training data is used for the prediction

In machine learning literature, nonparametric methods are also called instance-based or memory-based learning algorithms, since what they do is store the training instances in a lookup table and interpolate from these.

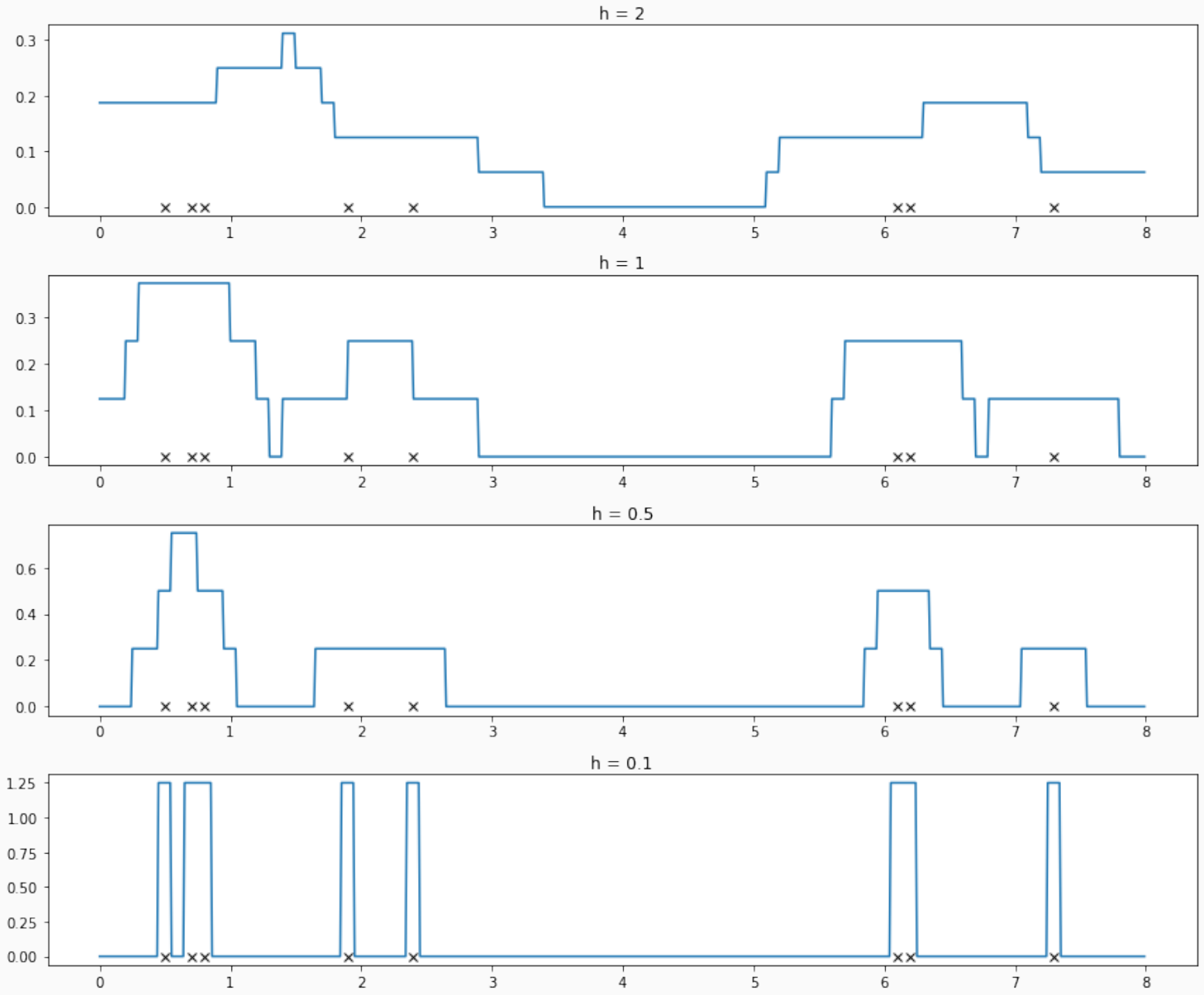
Nonparametric Density Estimation

Naive Histogram Estimator

The estimate is the sum of influences of x^t whose regions include x. Because this region of influence is hard (0 or 1), the estimate is not a continuous function and has jumps at xt h/2.

$$\hat{p}(x) = \frac{\#(x^t \text{in the same bin as x})}{Nh} \tag{1}$$

Naive Histogram Estimator



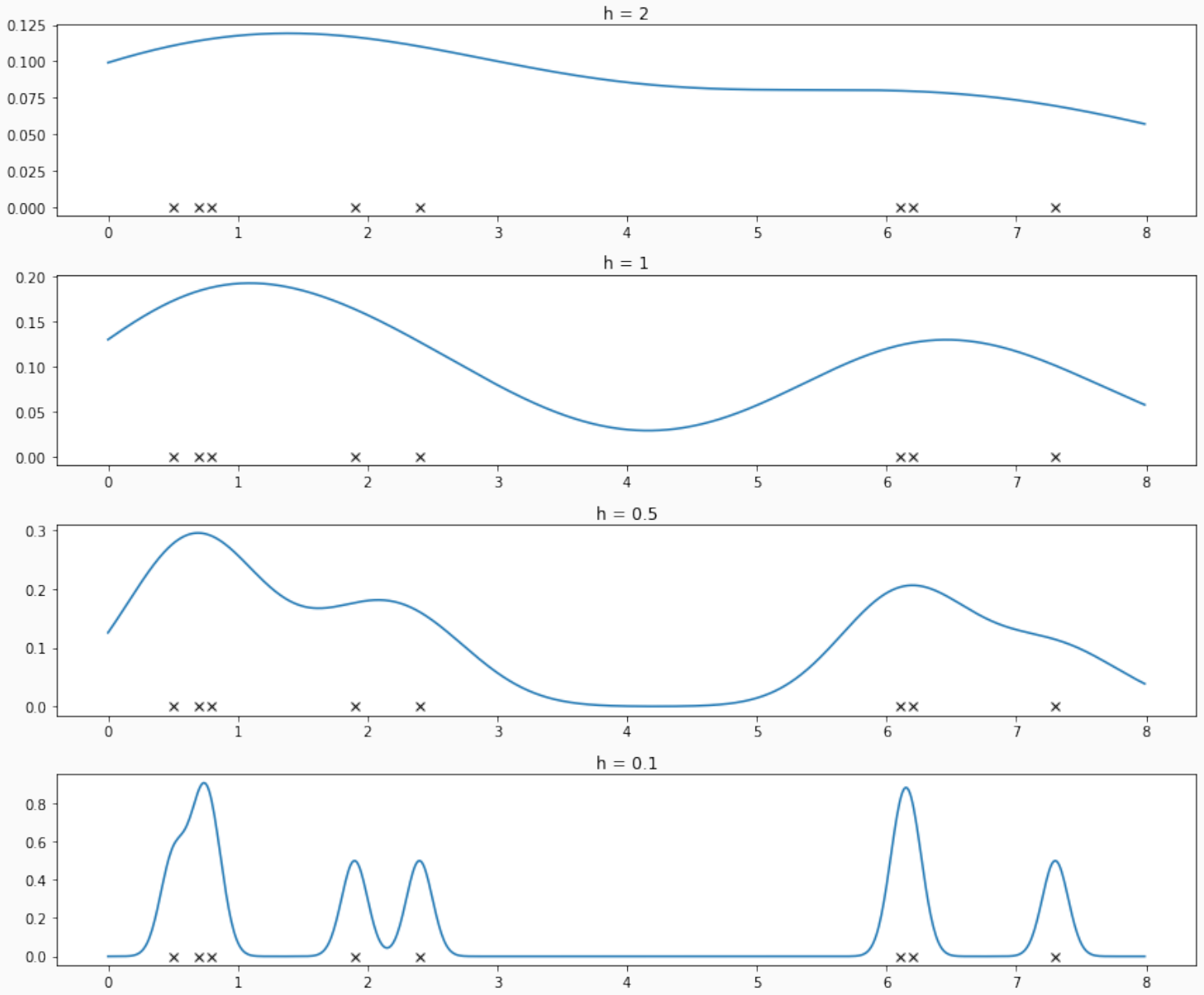
Kernel Estimator

A smooth weight function called a kernel to make estimate smooth. The most popular is the Gaussian kernel:

$$\hat{p}(x) = \frac{1}{Nh} \sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right) \tag{2}$$

$$K(u) = \frac{1}{2\pi} exp\left(\frac{-u^2}{2}\right) \tag{3}$$

Kernel Estimator



k-Nearest Neighbor Estimator

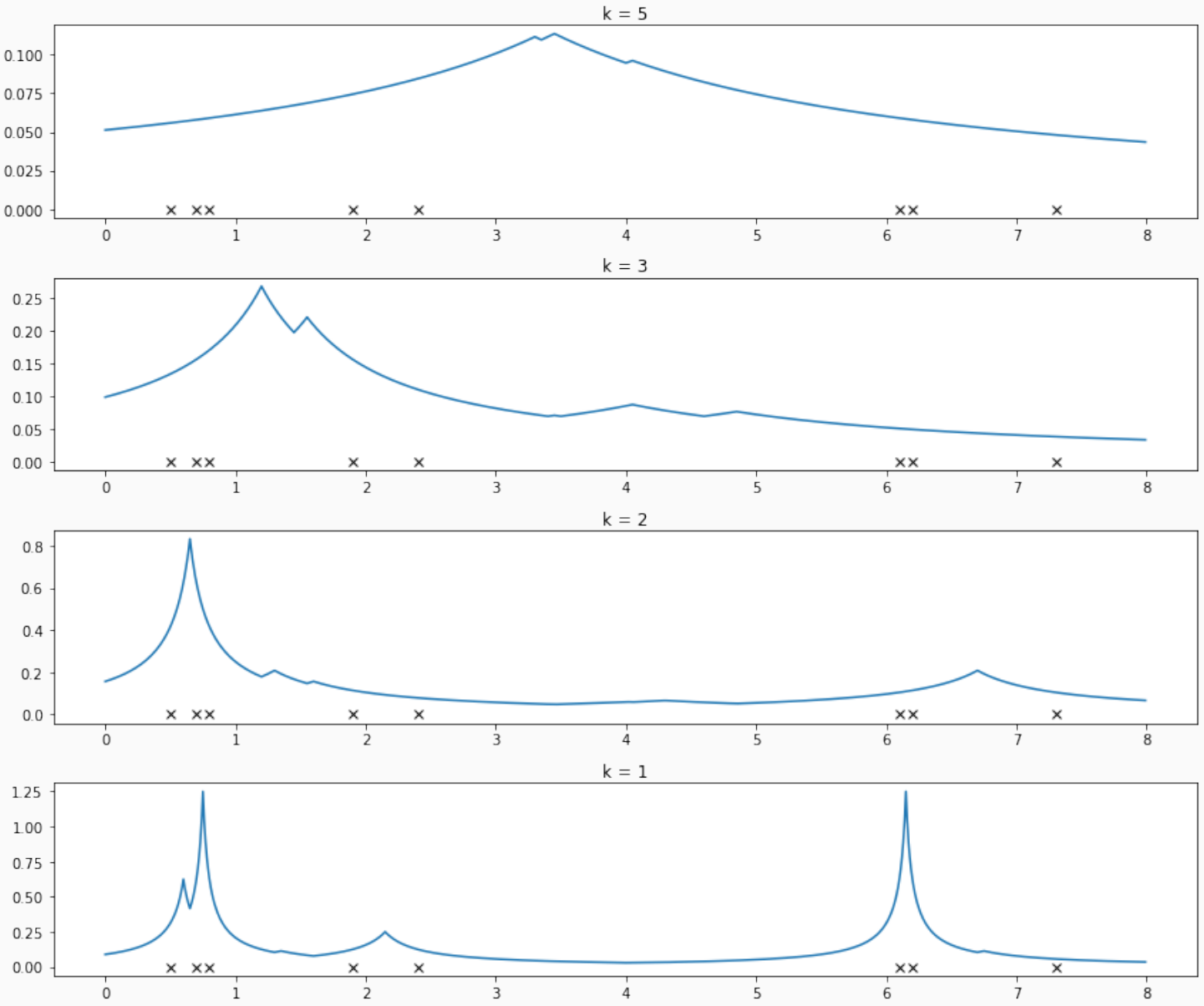
The nearest neighbor class of estimators adapts the amount of smoothing to the local density of data. The degree of smoothing is controlled by k, the number of neighbors taken into account, which is much smaller than N, the sample size.

For each x: $d_1(x) \leqslant d_2(x) \leqslant ... \leqslant d_N(x)$ is defined

The k-nearest neighbor (k-nn) density estimate is:

$$\hat{p}(x) = \frac{k}{2Nd_k(x)} \tag{4}$$

k-Nearest Neighbor Estimator



But there are still parameters!?

- output still depends on *smoothing parameters*: kernel spread h or number of neighbors k
- small parameters lead to small bias, but large variance and vice versa
- if the noise in the data is small, but we choose k or h high, we *oversmooth*
- if the noise in the data is high, but we choose k or h small, we *undersmooth*

$\Rightarrow$ use cross-validation to find the best parameter

Multivariate kernel density estimator

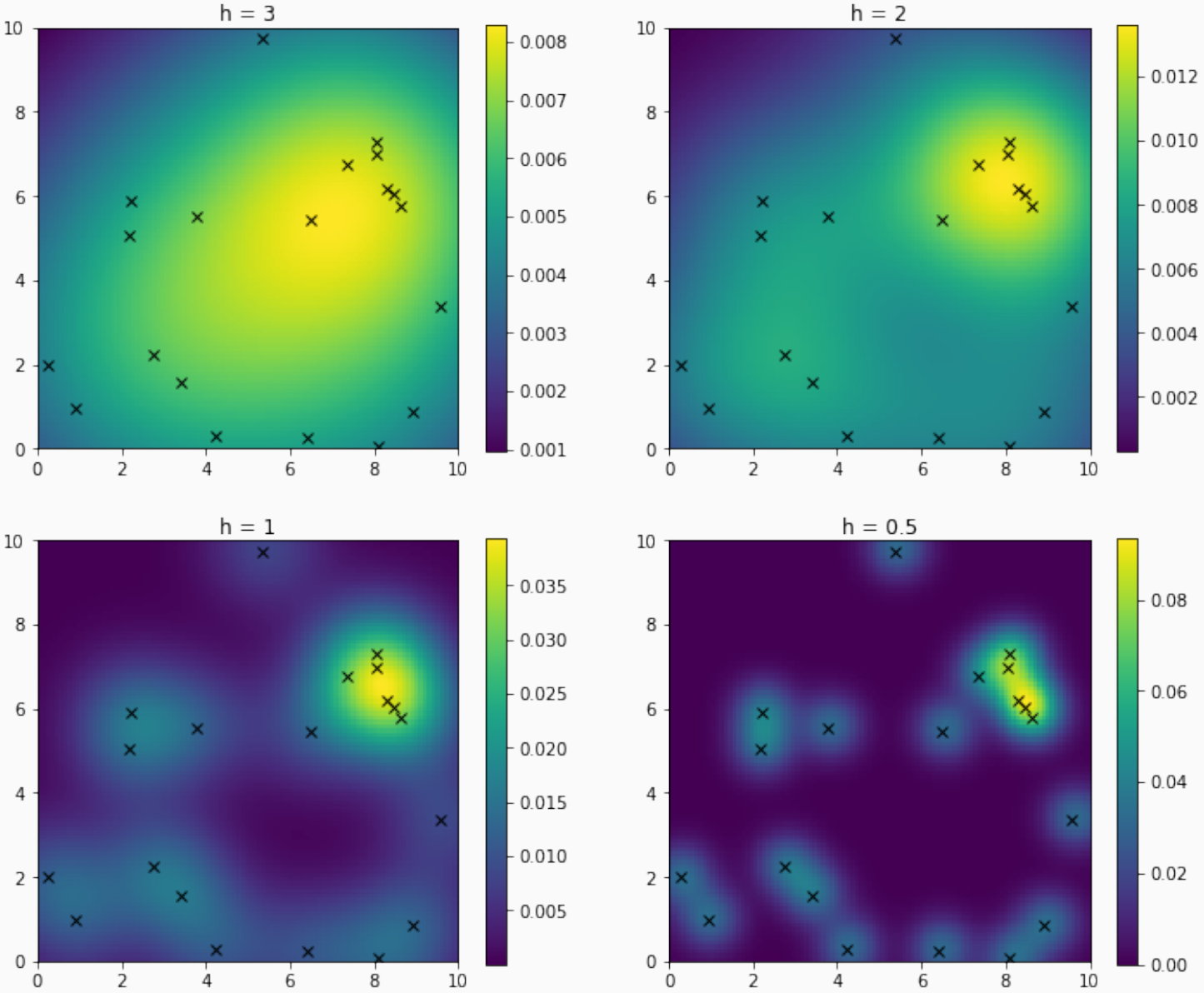
m-dimensional observations: $X = (x^t)_{t=1}^N$

Kernel density estimator:

$$\hat{p}(x) = \frac{1}{Nh^m} \sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right) \tag{5}$$

$$K(u) = \frac{1}{(2\pi)^{m/2} |S|^{1/2}} exp\left(-\frac{1}{2} u^T S^{-1} u\right) \tag{6}$$

Multivariate kernel density estimator



Nonparametric Classification

Kernel Classification

We can estimate class-conditional densities $p(x|C_i)$ also with a nonparametric kernel approach:

$$\hat{p}(x|C_i) = \frac{1}{N_i h^d} \sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right) r_i^t \tag{7}$$

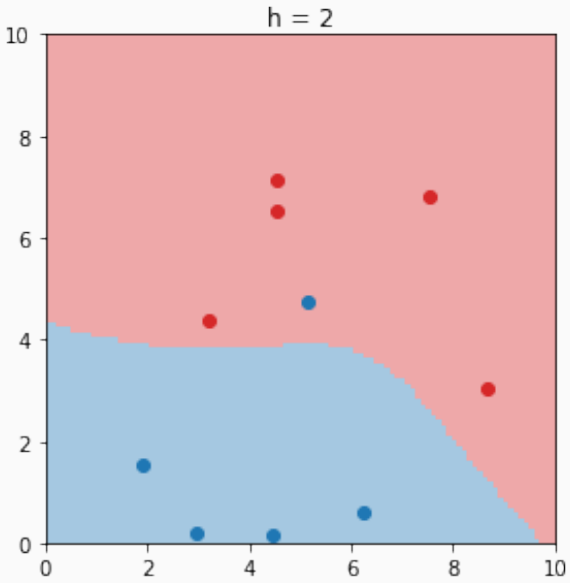
For the prior density of the classes, we have the MLE:

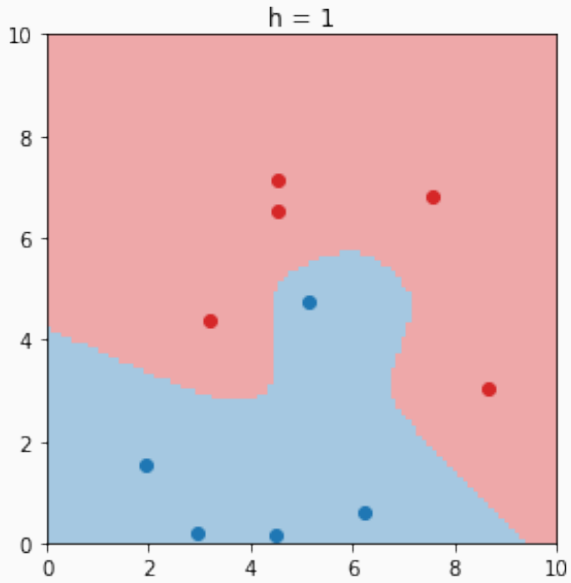
$$\hat{p}(C_i) = \frac{N_i}{N} \tag{8}$$

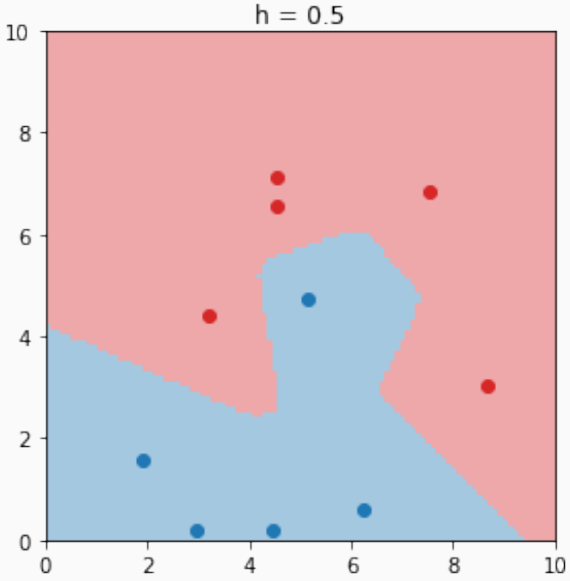
Then classify with:

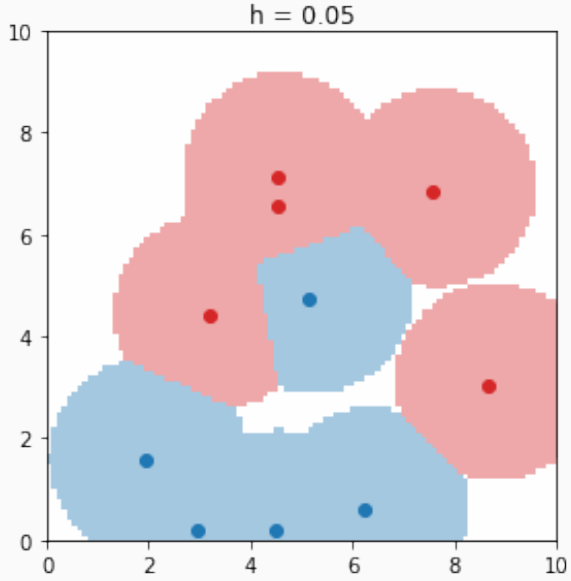
$$\text{class}(x) = \mathit{argmax}_i \{\hat{p}(x|C_i)\hat{p}(C_i)\} \tag{9}$$

Kernel Classification









k-nn Classification

For the knn approach, we get the following densities:

$$\hat{p}(x|C_i) = \frac{k_i}{N_i V^k(x)} \tag{10}$$

$V^k(x)$: the volume of the d-dimensional hypersphere centered at x, with radius $||x - x_{(k)}||$ where $x_{(k)}$ is the k-th nearest observation to x (among all neighbors from all classes of x)

k_i: the number of neighbors from the k nearest neighbors, that belong to class C_i

k-nn Classification

We can now calculate the marginal probability for the data:

$$\hat{p}(x) = \sum_i \hat{p}(x|C_i)\hat{p}(C_i) \tag{11}$$

$$= \sum_i \frac{k_i}{N_i V^k(x)} \frac{N_i}{N} \tag{12}$$

$$= \sum_i \frac{k_i}{N V^k(x)} \tag{13}$$

$$= \frac{k}{N V^k(x)} \tag{14}$$

k-nn Classification

Classification is done with:

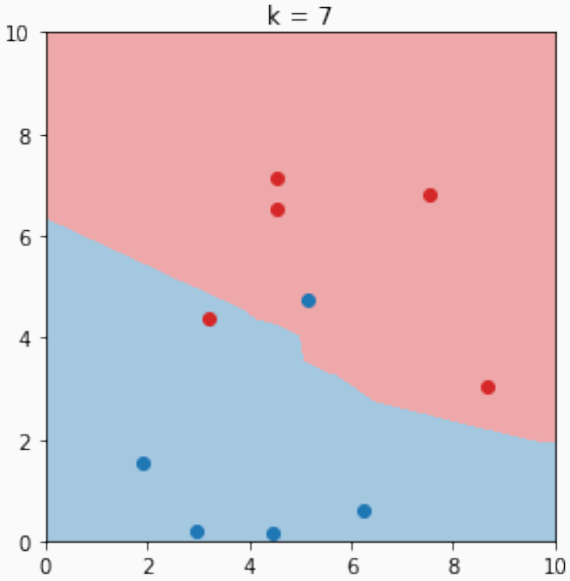
$$\hat{p}(C_i|x) = \frac{\hat{p}(x|C_i)\hat{p}(C_i)}{\hat{p}(x)} \tag{15}$$

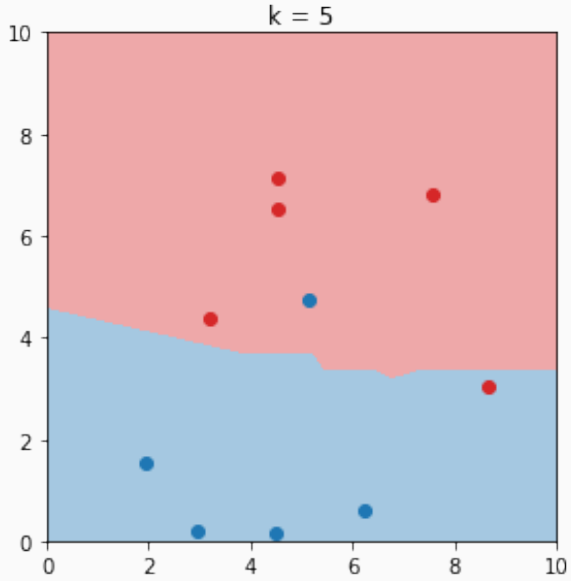
$$= \frac{\frac{k_i}{N_i V^k(x)} \frac{N_i}{N}}{\frac{k}{NV^k(x)}} \tag{16}$$

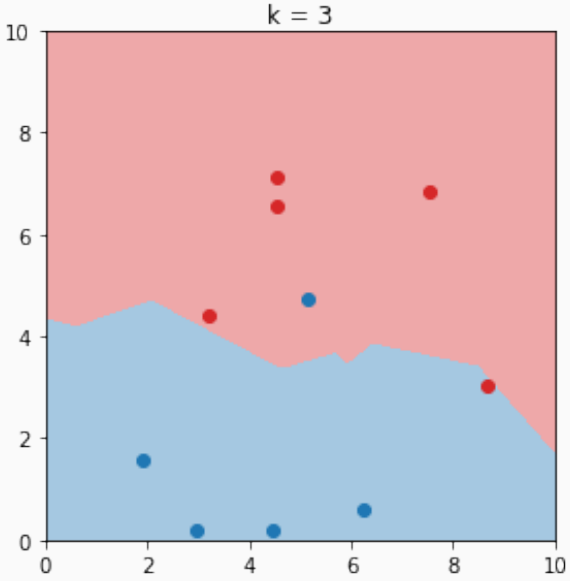
$$= \frac{k_i}{k} \tag{17}$$

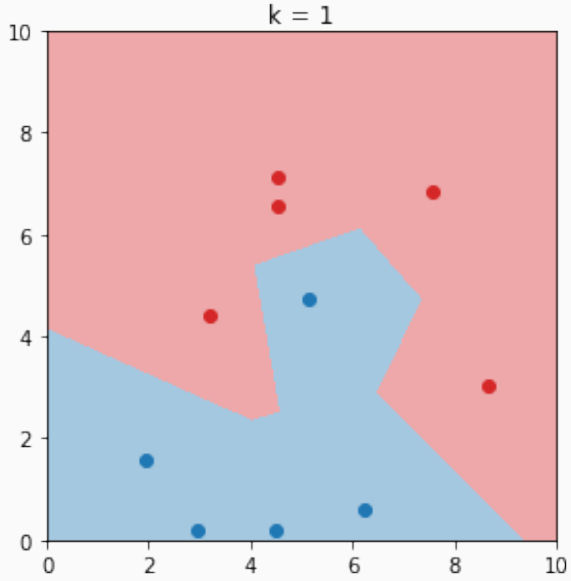
$$\text{class}(x) = argmax_i \{\hat{p}(C_i|x)\} \tag{18}$$

k-nn Classification









Condensed Nearest Neighbor

In order to descrease time and space complexity, the training set can be condensed. The error should not change.

$Z = \{\}$

While Z still changes:

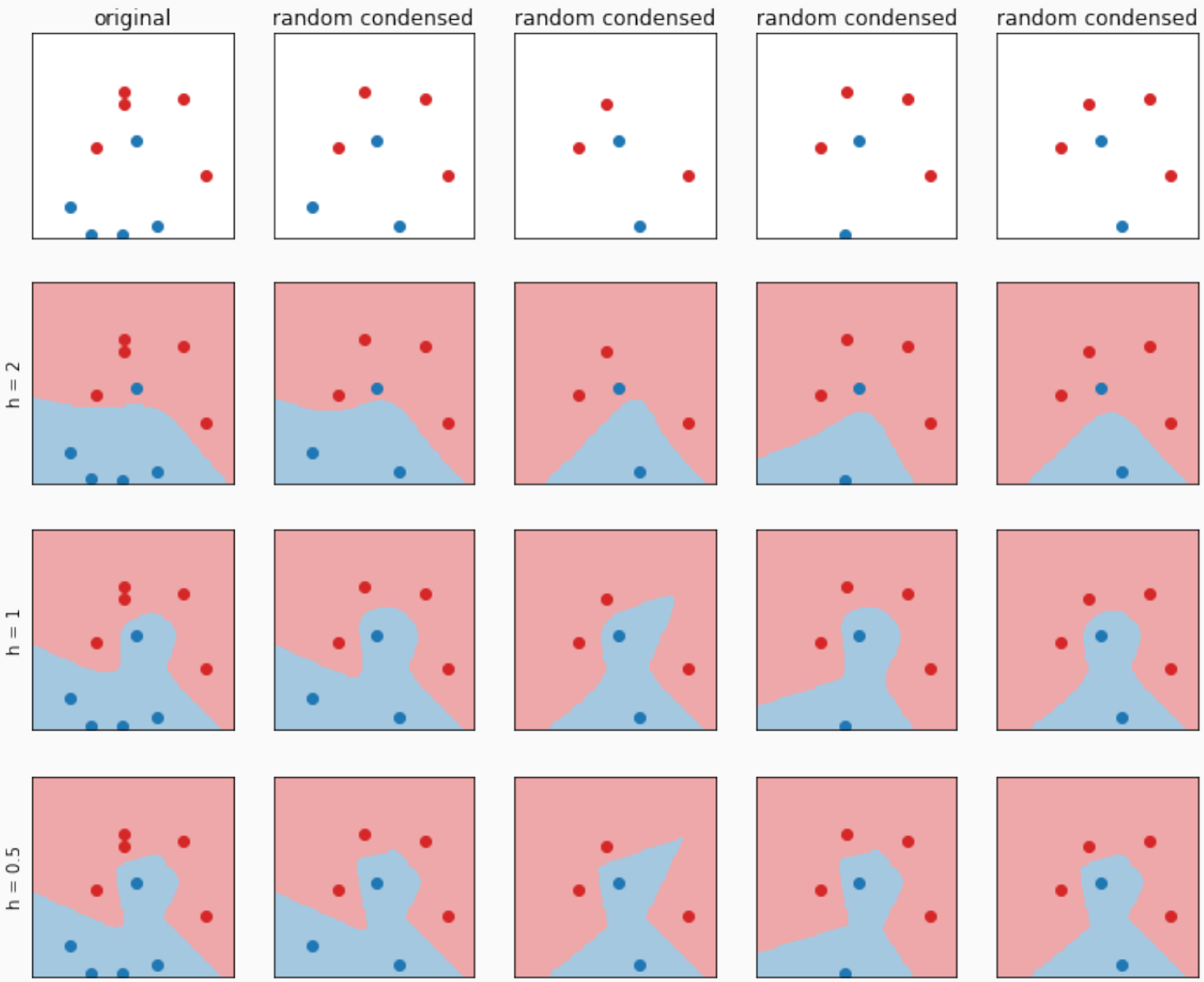
For all $x \in X$ (in random order):

Find $z \in Z$ which is closest to x

If class(x) $\neq$ class(z): add x to Z

The algorithm can yield quite different results, because of the randomness.

Condensed Nearest Neighbor



Distance-Based Classification

Nonparametric classification relies on a *distance measure* to specify, which class is the "closest".

Different distance measures make more or less sense, depending on the problem.

For parametric we had e.g. the Mahalanobis distance:

$$D(x, m_i) = (x - m_i)^T S_i^{-1} (x - m_i) \tag{19}$$

Distance-Based Classification

The main idea of nonparametric distance learning is to have a distance measure for each neighborhood, e.g. in k-nn.

Thus we need a distance measure between a point and an instance of the training set, e.g. with Mahalanobis

$$D(x, x^t|M) = (x - x^t)^T M(x - x^t) \tag{20}$$

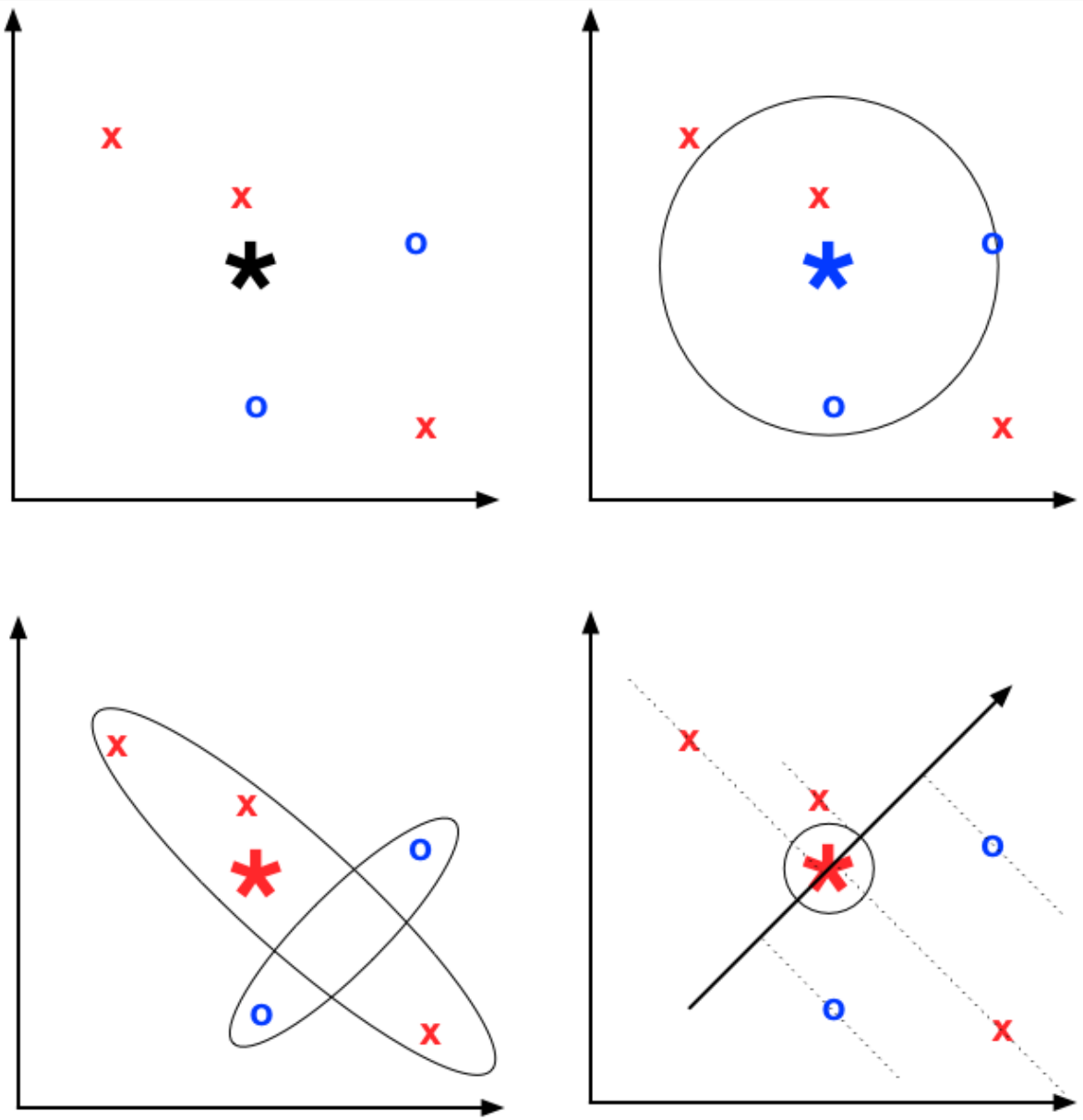
where M is a parameter and can be learned from the training set to optimize this measure (e.g. with the *large margin nearest neighbor* algorithm).

Distance-Based Classification

$$D(x, x^t|M) = (x - x^t)^T M(x - x^t) \tag{21}$$

Overfitting high dimensional data can be reduced by using a low-rank approximation of M. The Mahalanobis distance for this sparce M corresponds to the (squared) euclidean distance in a subspace.

Distance-Based Classification



Hamming distance

Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

$$HD(x, x^t) = \sum_{j=1}^{d} 1(x_j \neq x_j^t) \tag{22}$$

Outlier Detection

Local outlier factor that comfactor pares the denseness of the neighborhood of an instance with the average denseness of the neighborhoods of its neighbors

$$LOF(x) = \frac{d_k(x)}{\sum_{s \in N(X)} d_k(s)/|N(x)|} \tag{23}$$

If LOF(x) is close to 1, x is not an outlier; as it gets larger, the probability that it is an outlier increases

Nonparametric Regression

Nonparametric Regression: Smoothing Models

$X = \{x^t, r^t\}, r^t \in R$

$r^t = g(x^t) + \epsilon$

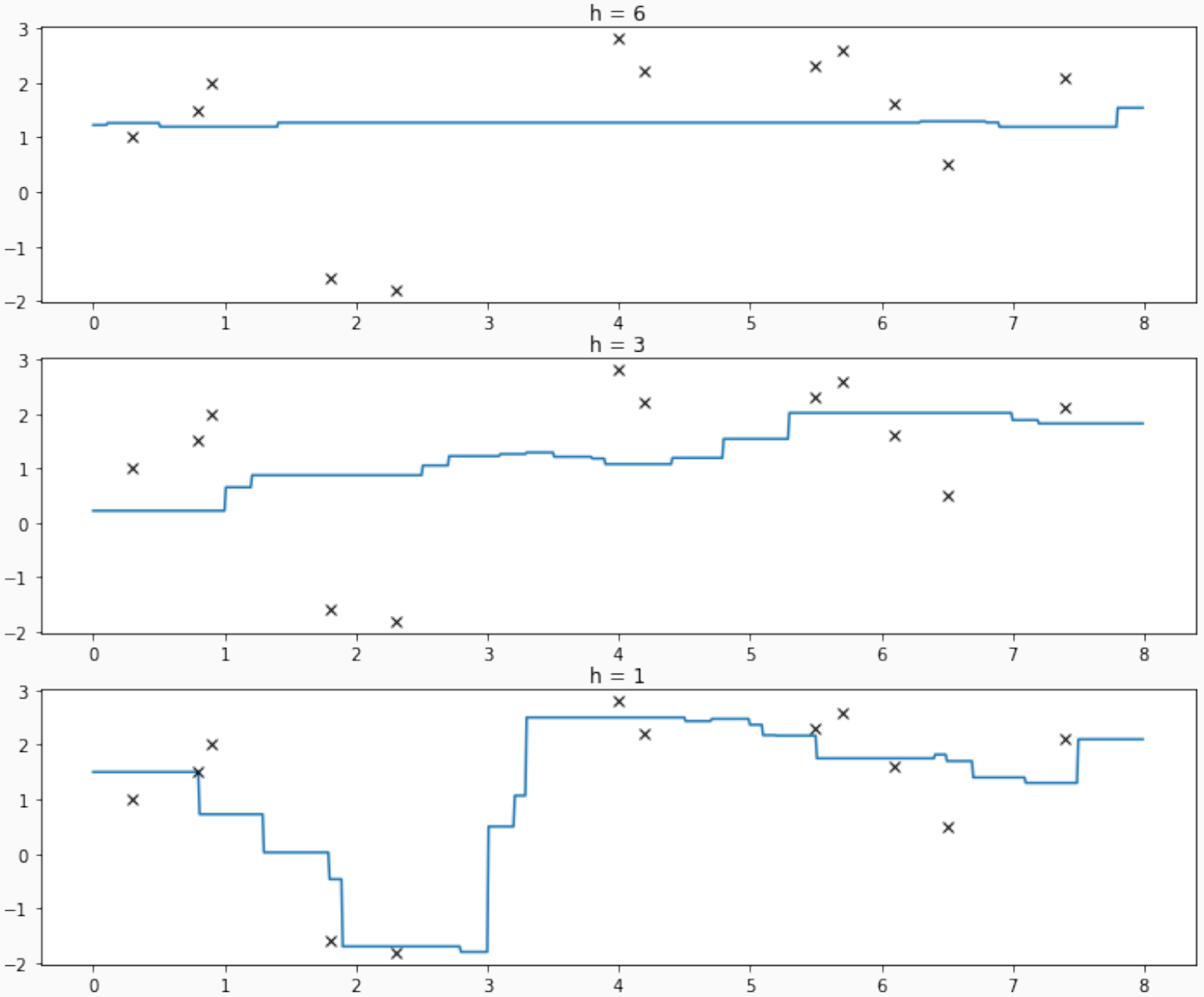
Running Mean Smoother:

$$g(x) = \frac{\sum_{t=1}^{N} w\left(\frac{x-x^t}{h}\right) r^t}{\sum_{t=1}^{N} w\left(\frac{x-x^t}{h}\right)} \qquad w(u) = \begin{cases} 1, & \text{if } |u| < h \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

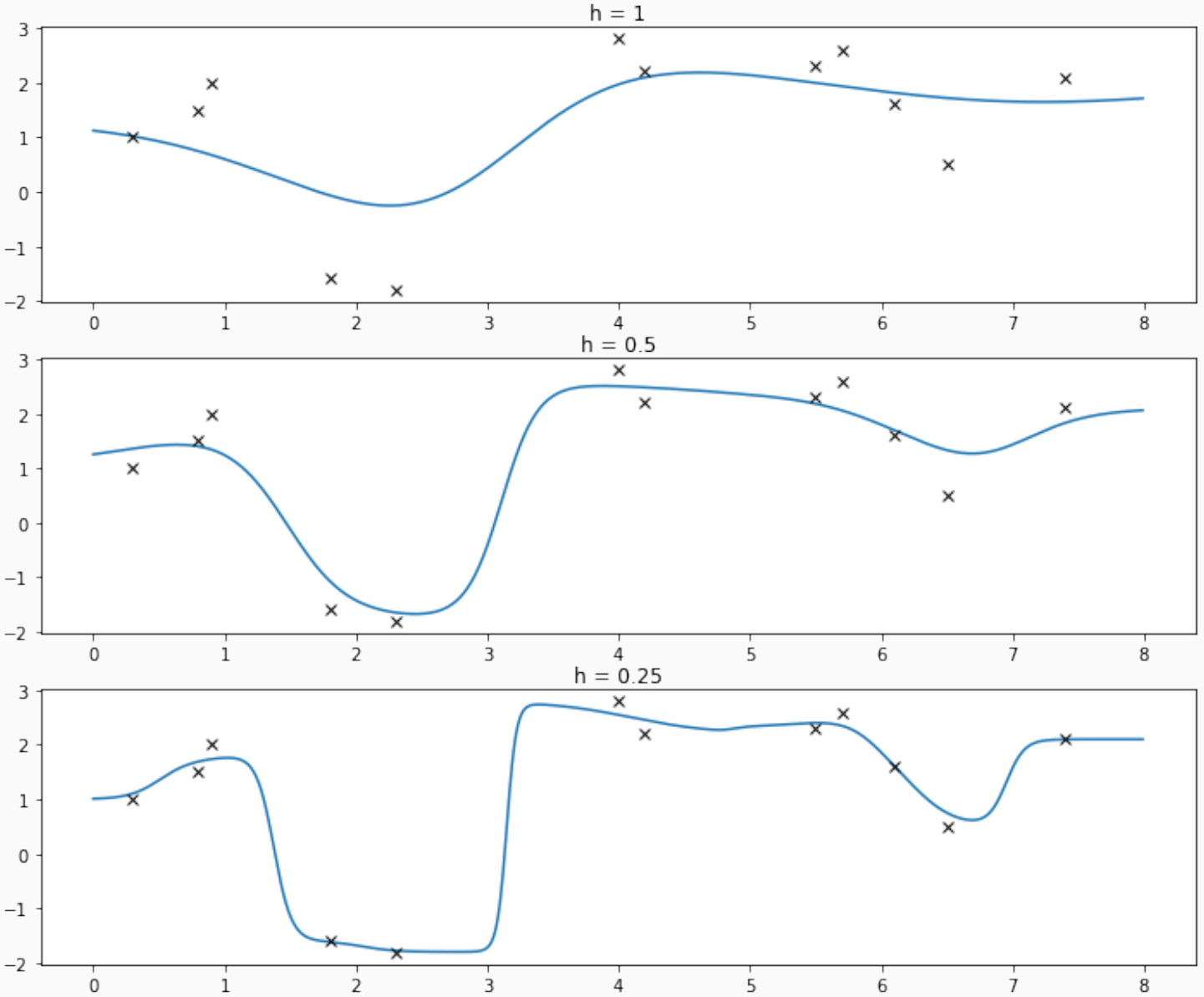
Kernel Smoother:

$$g(x) = \frac{\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right) r^t}{\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right)} \tag{25}$$

Nonparametric Regression: Smoothing Models

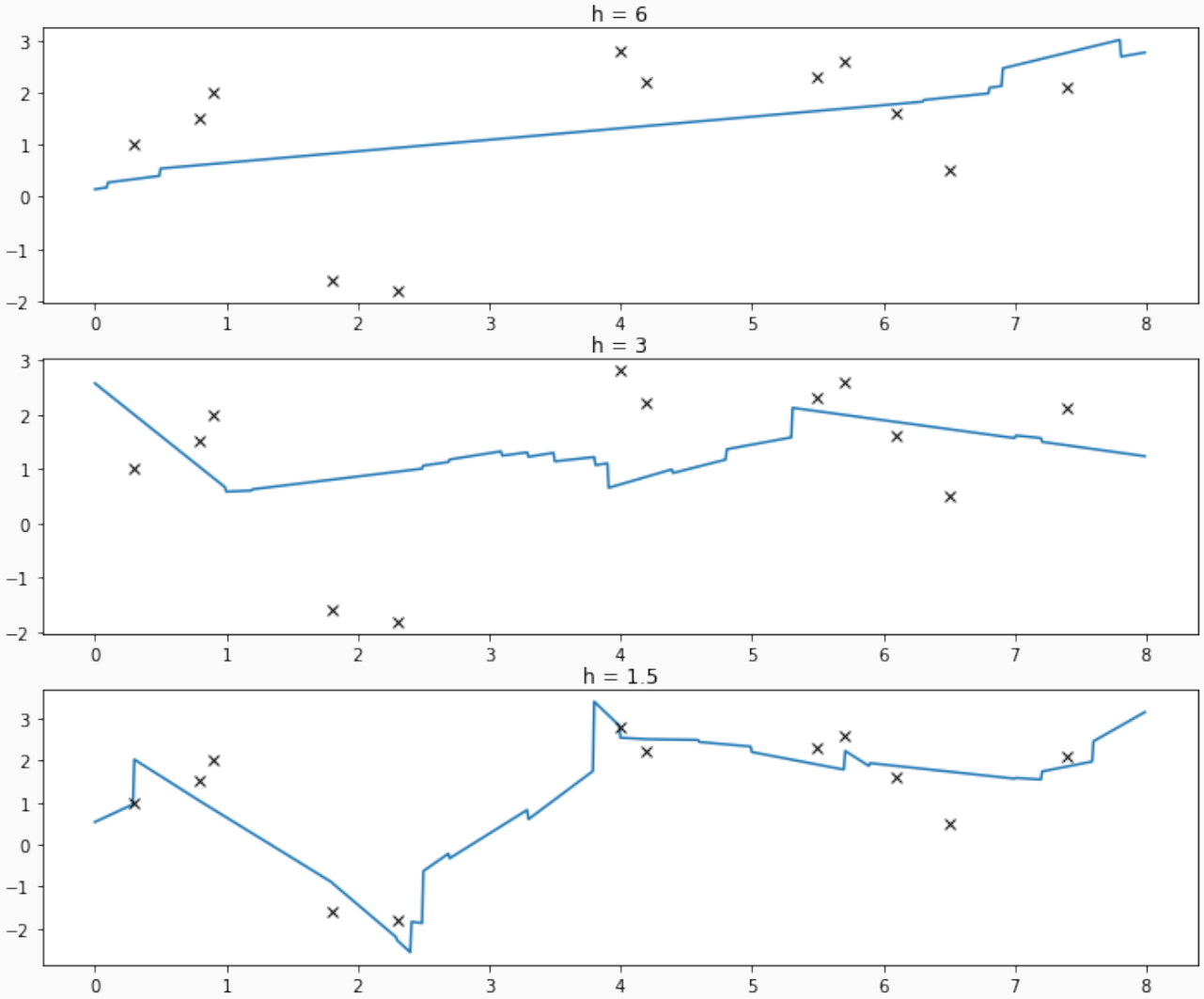


Nonparametric Regression: Smoothing Models



Nonparametric Regression: Smoothing Models

Running Line Smoother



Remarks

Remarks

- algorithms were proposed many years ago but were only recently used due to computational complexity
- for high dimensional data, the amount of needed intances increases exponentially
- another famous example are support vectore machines (see later presentation)