MACHINE LEARNING

Chao Wang (*Frank*)

wangchao@nankai.edu.cn

CHAPTER 8:

NONPARAMETRIC METHODS

Nonparametric Estimation

- Parametric (single global model), semiparametric (small number of local models)
- Nonparametric: Similar inputs have similar outputs
- Functions (pdf, discriminant, regression) change smoothly
- Keep the training data;"let the data speak for itself"
- Given x, find a small number of closest training instances and interpolate from these
- Aka lazy/memory-based/case-based/instance-based learning

Density Estimation

- Given the training set $\boldsymbol{X}=\{x^t\}_t$ drawn iid from $p(x)$
- Divide data into bins of size h
- Histogram:

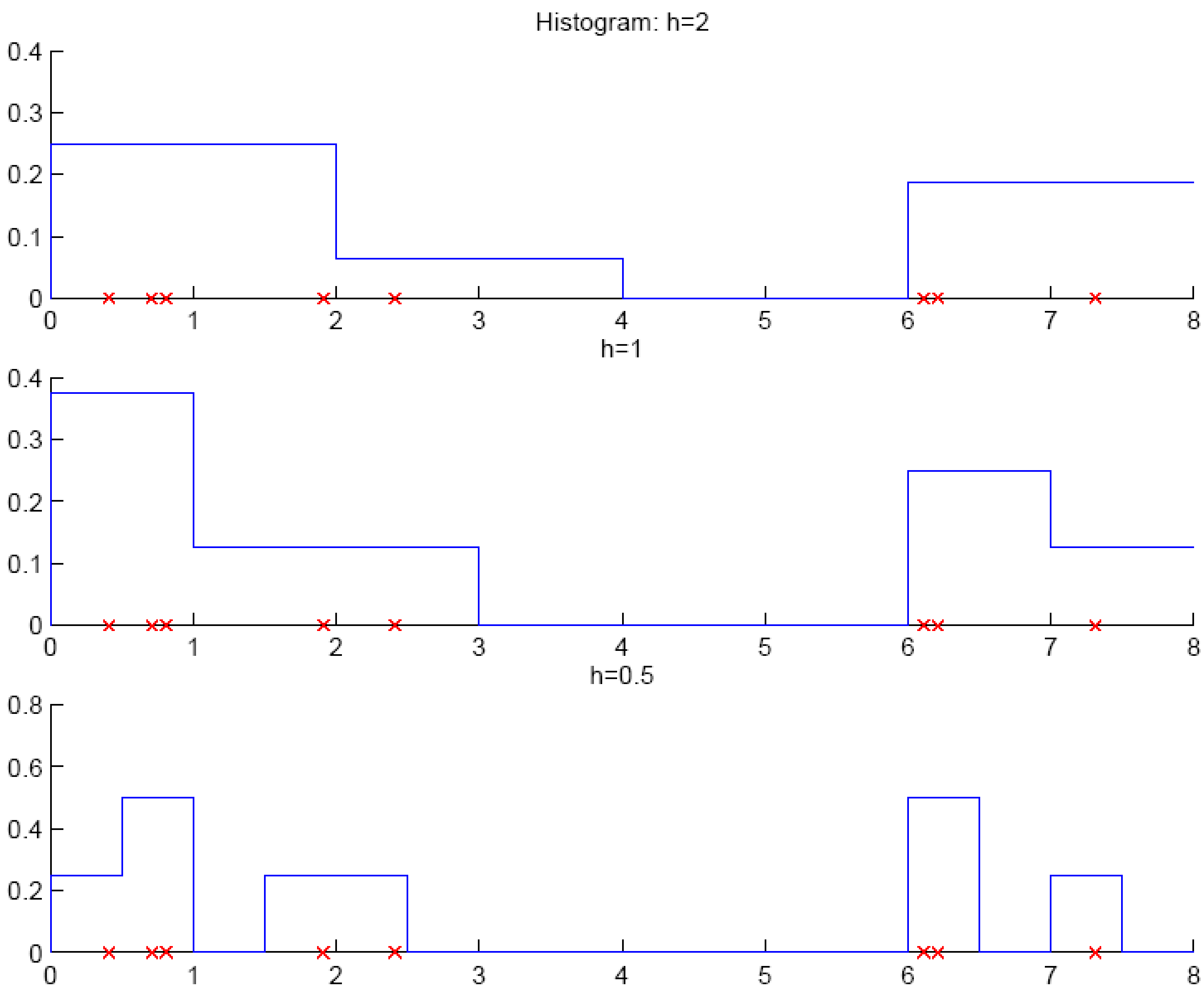
$$\hat{p}(x)=\frac{\#\{x^t \text{ in the same bin as } x\}}{Nh}$$

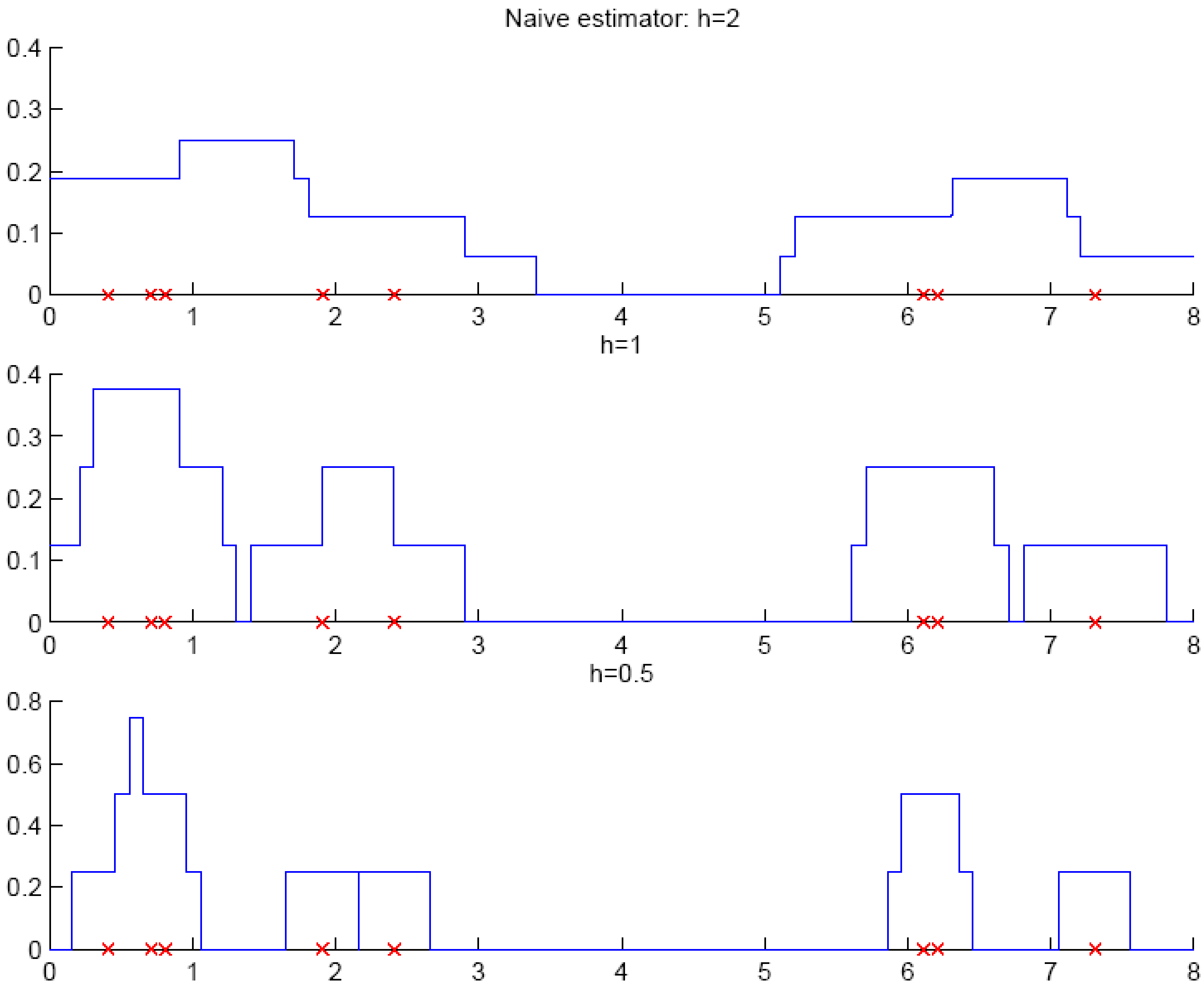
- Naive estimator:

$$\hat{p}(x)=\frac{\#\{x-h<x^t\leq x+h\}}{2Nh}$$

or

$$\hat{p}(x)=\frac{1}{Nh}\sum_{t=1}^{N}w\left(\frac{x-x^t}{h}\right) \quad w(u)=\begin{cases}1/2 & \text{if } |u|<1\\ 0 & \text{otherwise}\end{cases}$$


Histogram: h=2
h=1
h=0.5
0.4
0.3
0.2
0.1
0
0.8
0.6
0 1 2 3 4 5 6 7 8



Naive estimator: h=2
h=1
h=0.5

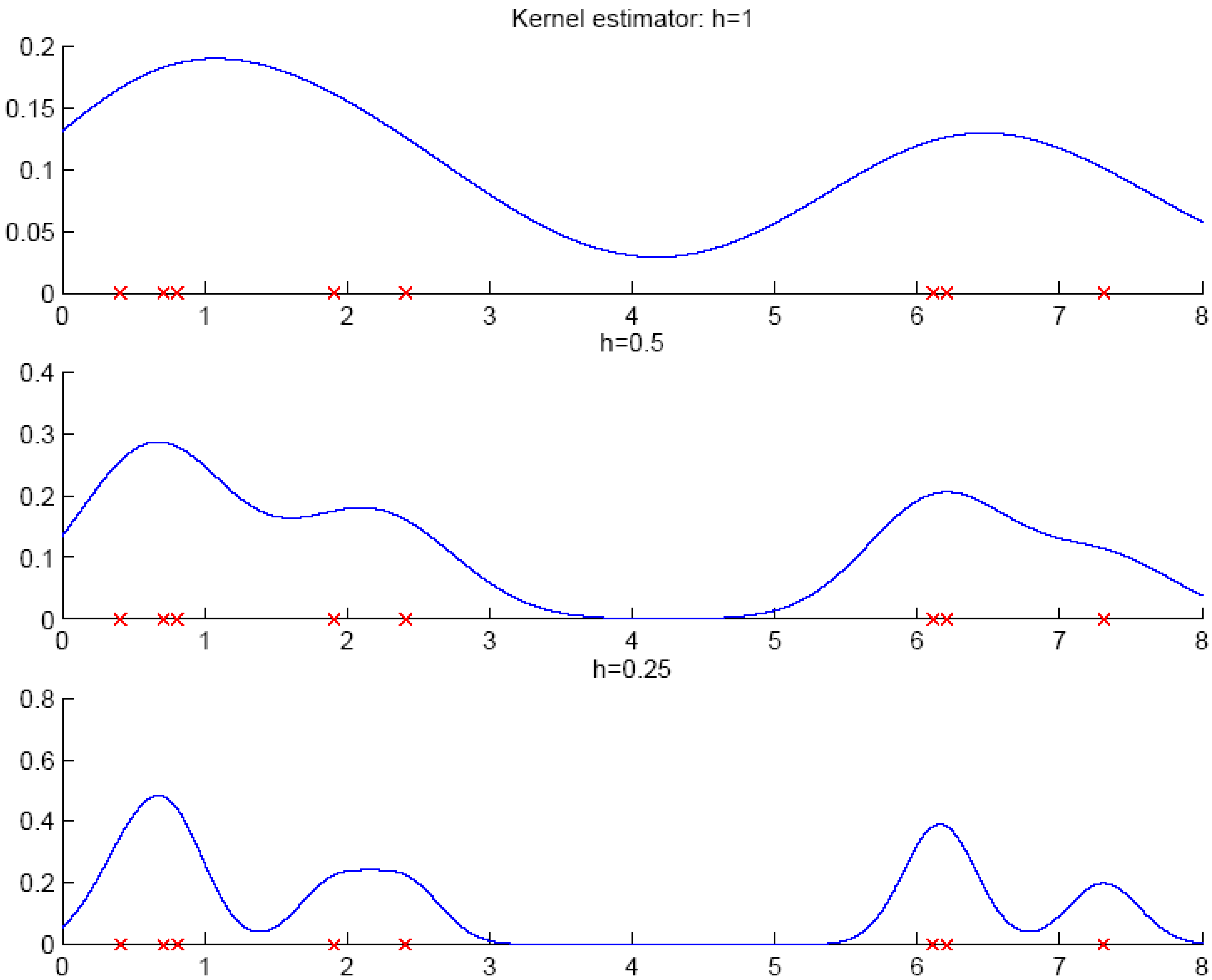
Kernel Estimator

- Kernel function, e.g., Gaussian kernel:

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right]$$

- Kernel estimator (Parzen windows)

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

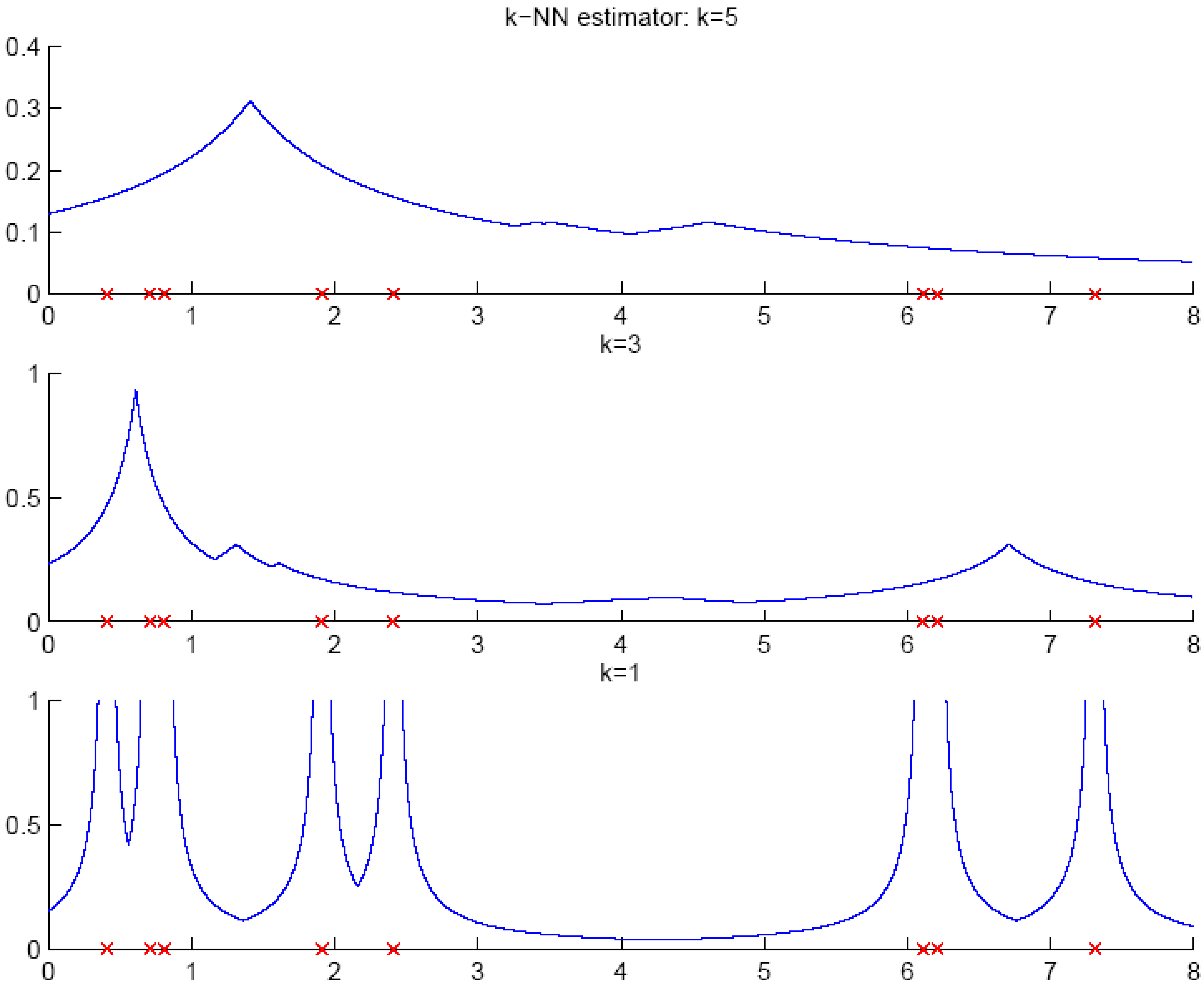

Kernel estimator: h=1
h=0.5
h=0.25
0.2
0.15
0.1
0.05
0
0.4
0.3
0.2
0.1
0
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8

k-Nearest Neighbor Estimator

- Instead of fixing bin width h and counting the number of instances, fix the instances (neighbors) k and check bin width

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

$d_k(x)$, distance to kth closest instance to x


k−NN estimator: k=5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8
k=3
1
0.5
0
k=1
1
0.5
0

Multivariate Data

- Kernel density estimator

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

Multivariate Gaussian kernel

spheric

$$K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \exp\left[-\frac{\|\mathbf{u}\|^2}{2}\right]$$

ellipsoid

$$K(\mathbf{u}) = \frac{1}{(2\pi)^{d/2} |\mathbf{S}|^{1/2}} \exp\left[-\frac{1}{2}\mathbf{u}^T \mathbf{S}^{-1} \mathbf{u}\right]$$

Nonparametric Classification

- Estimate $p(\mathbf{x}|C_i)$ and use Bayes' rule
- Kernel estimator

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

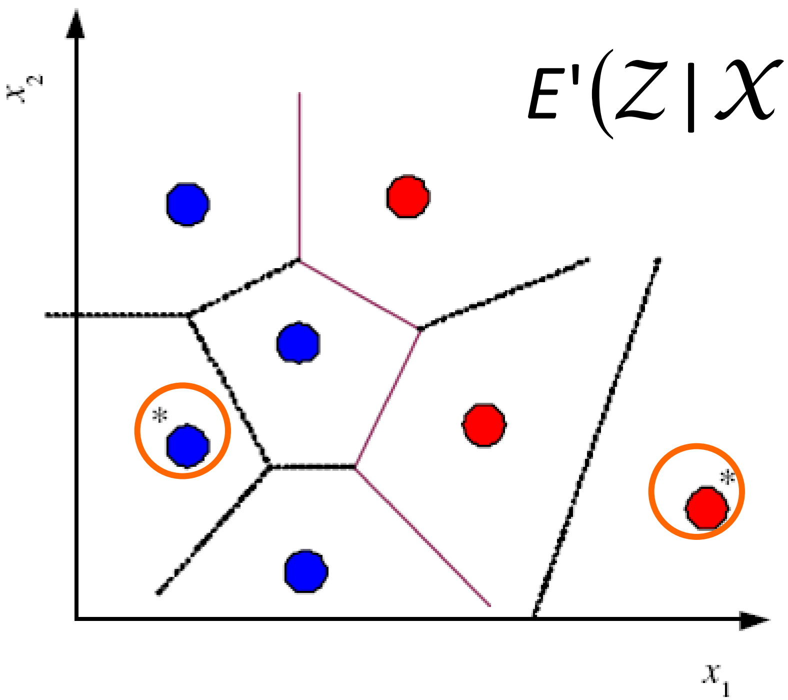
$$g_i(\mathbf{x})=\hat{p}(\mathbf{x}|C_i)\hat{P}(C_i)=\frac{1}{Nh^d}\sum_{t=1}^{N}K\left(\frac{\mathbf{x}-\mathbf{x}^t}{h}\right)r_i^t$$

- k-NN estimator

$$\hat{p}(\mathbf{x}|C_i)=\frac{k_i}{N_i V^k(\mathbf{x})} \quad \hat{P}(C_i|\mathbf{x})=\frac{\hat{p}(\mathbf{x}|C_i)\hat{P}(C_i)}{\hat{p}(\mathbf{x})}=\frac{k_i}{k}$$

Condensed Nearest Neighbor

- Time/space complexity of k-NN is O (N)
- Find a subset Z of X that is small and is accurate in classifying X (Hart, 1968)



$$E'(Z \mid X) = E(X \mid Z) + \lambda |Z|$$

Condensed Nearest Neighbor

- Incremental algorithm: Add instance if needed

$\mathcal{Z} \leftarrow \emptyset$

Repeat

  For all $\boldsymbol{x} \in \mathcal{X}$ (in random order)

    Find $\boldsymbol{x}' \in \mathcal{Z}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}'\| = \min_{\boldsymbol{x}^j \in \mathcal{Z}} \|\boldsymbol{x} - \boldsymbol{x}^j\|$

    If class($\boldsymbol{x}$)$\neq$class($\boldsymbol{x}'$) add $\boldsymbol{x}$ to $\mathcal{Z}$

Until $\mathcal{Z}$ does not change

Distance-based Classification

- Find a distance function $D(\mathbf{x}^r,\mathbf{x}^s)$ such that

  if $\mathbf{x}^r$ and $\mathbf{x}^s$ belong to the same class, distance is small and if they belong to different classes, distance is large

- Assume a parametric model and learn its parameters using data, e.g.,

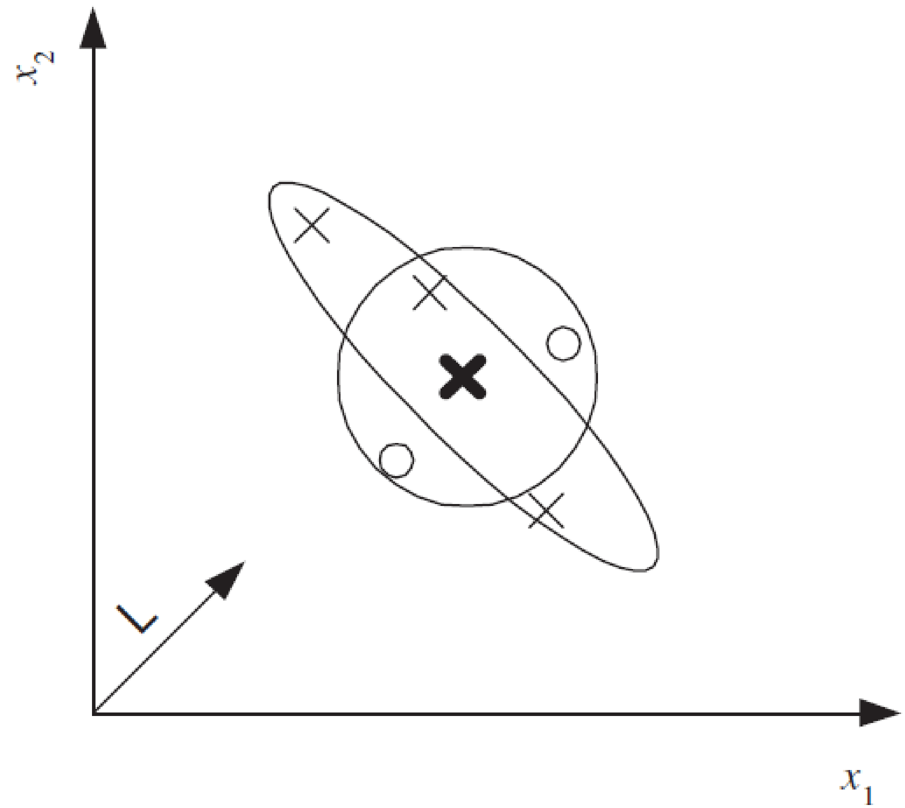
$$\mathcal{D}(\mathbf{x},\mathbf{x}^t|\mathbf{M}) = (\mathbf{x}-\mathbf{x}^t)^T\mathbf{M}(\mathbf{x}-\mathbf{x}^t)$$

Learning a Distance Function

- The three-way relationship between distances, dimensionality reduction, and feature extraction.
- **M**=**L**^{T}**L** is *dxd* and **L** is *kxd*

$$
\begin{aligned}
\mathcal{D}(\boldsymbol{x}, \boldsymbol{x}^t | \mathbf{M}) &= (\boldsymbol{x} - \boldsymbol{x}^t)^T \mathbf{M} (\boldsymbol{x} - \boldsymbol{x}^t) = (\boldsymbol{x} - \boldsymbol{x}^t)^T \mathbf{L}^T \mathbf{L} (\boldsymbol{x} - \boldsymbol{x}^t) \\
&= (\mathbf{L}(\boldsymbol{x} - \boldsymbol{x}^t))^T (\mathbf{L}(\boldsymbol{x} - \boldsymbol{x}^t)) = (\mathbf{L}\boldsymbol{x} - \mathbf{L}\boldsymbol{x}^t)^T (\mathbf{L}\boldsymbol{x} - \mathbf{L}\boldsymbol{x}^t) \\
&= (\boldsymbol{z} - \boldsymbol{z}^t)^T (\boldsymbol{z} - \boldsymbol{z}^t) = \|\boldsymbol{z} - \boldsymbol{z}^t\|^2
\end{aligned}
$$

- Similarity-based representation using similarity scores
- Large-margin nearest neighbor (chapter 13)



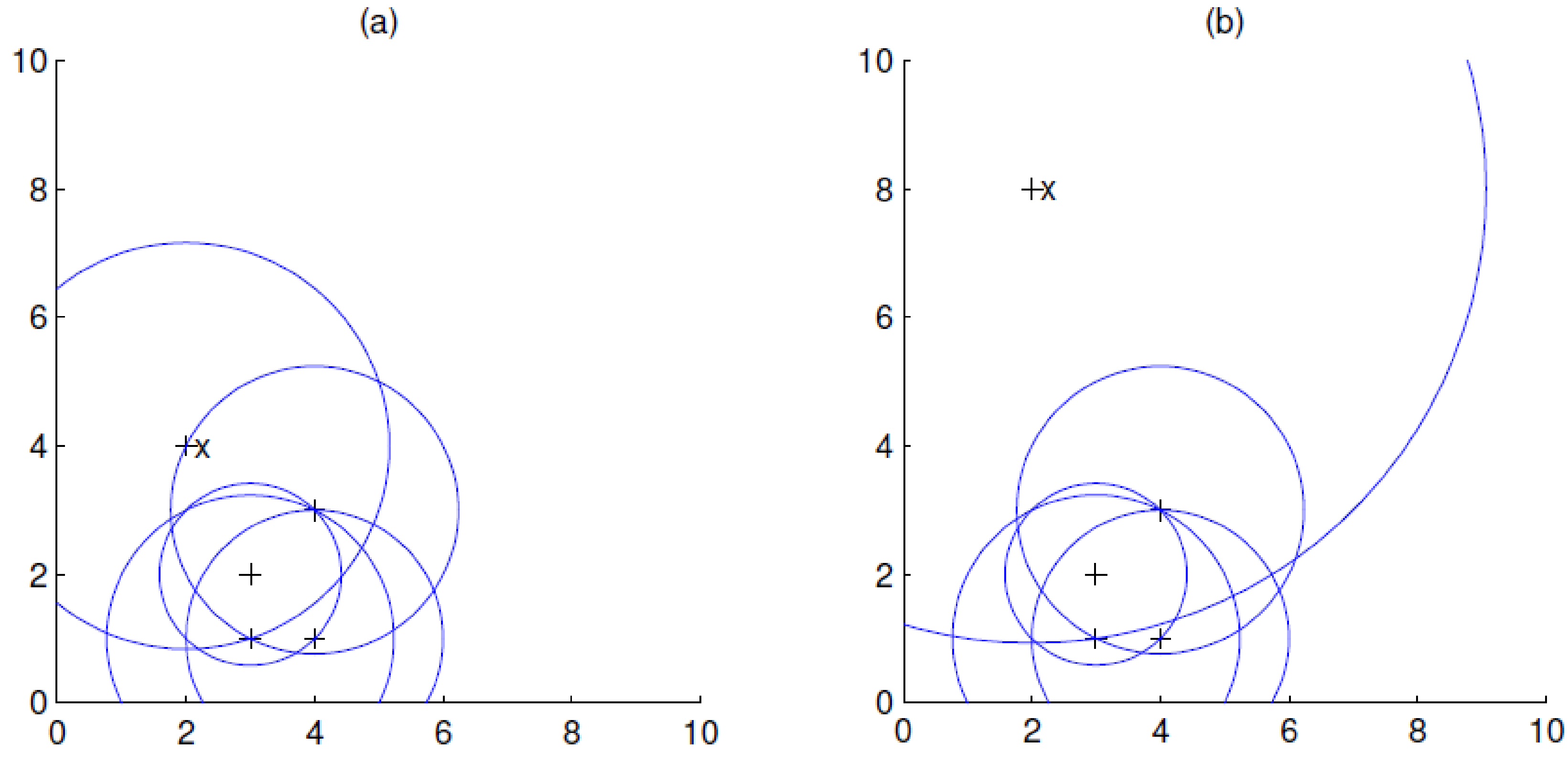
Euclidean distance (circle) is not suitable,
Mahalanobis distance using an **M** (ellipse) is suitable.
After the data is projected along **L**, Euclidean distance can be used.

Outlier Detection

- Find outlier/novelty points
- Not a two-class problem because outliers are very few, of many types, and seldom labeled
- Instead, one-class classification problem: Find instances that have low probability
- In nonparametric case: Find instances far away from other instances

Local Outlier Factor

$$\mathrm{LOF}(\boldsymbol{x}) = \frac{d_k(\boldsymbol{x})}{\sum_{\boldsymbol{s}\in\mathcal{N}(\boldsymbol{x})} d_k(\boldsymbol{s})/|\mathcal{N}(\boldsymbol{x})|}$$



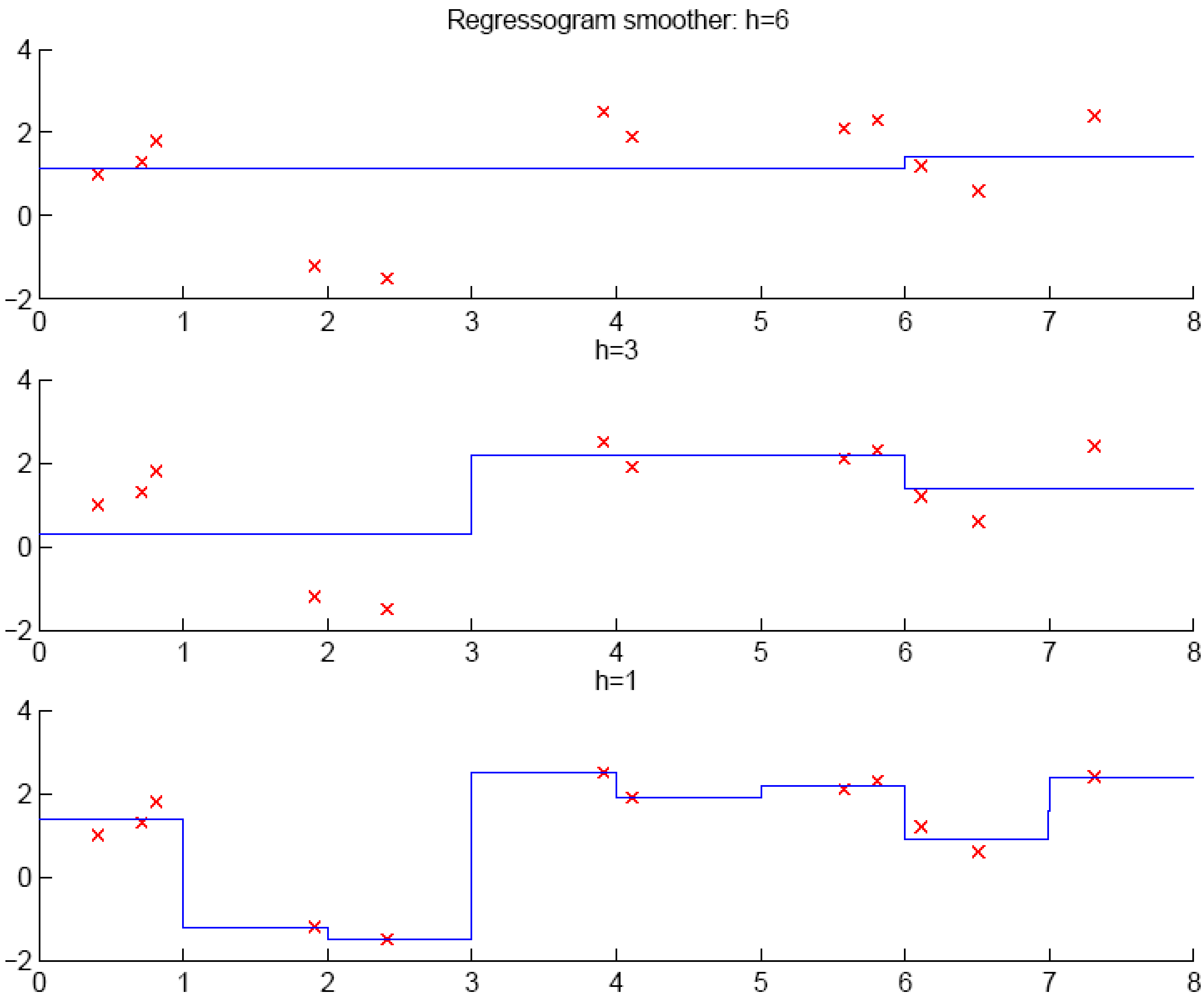
Nonparametric Regression

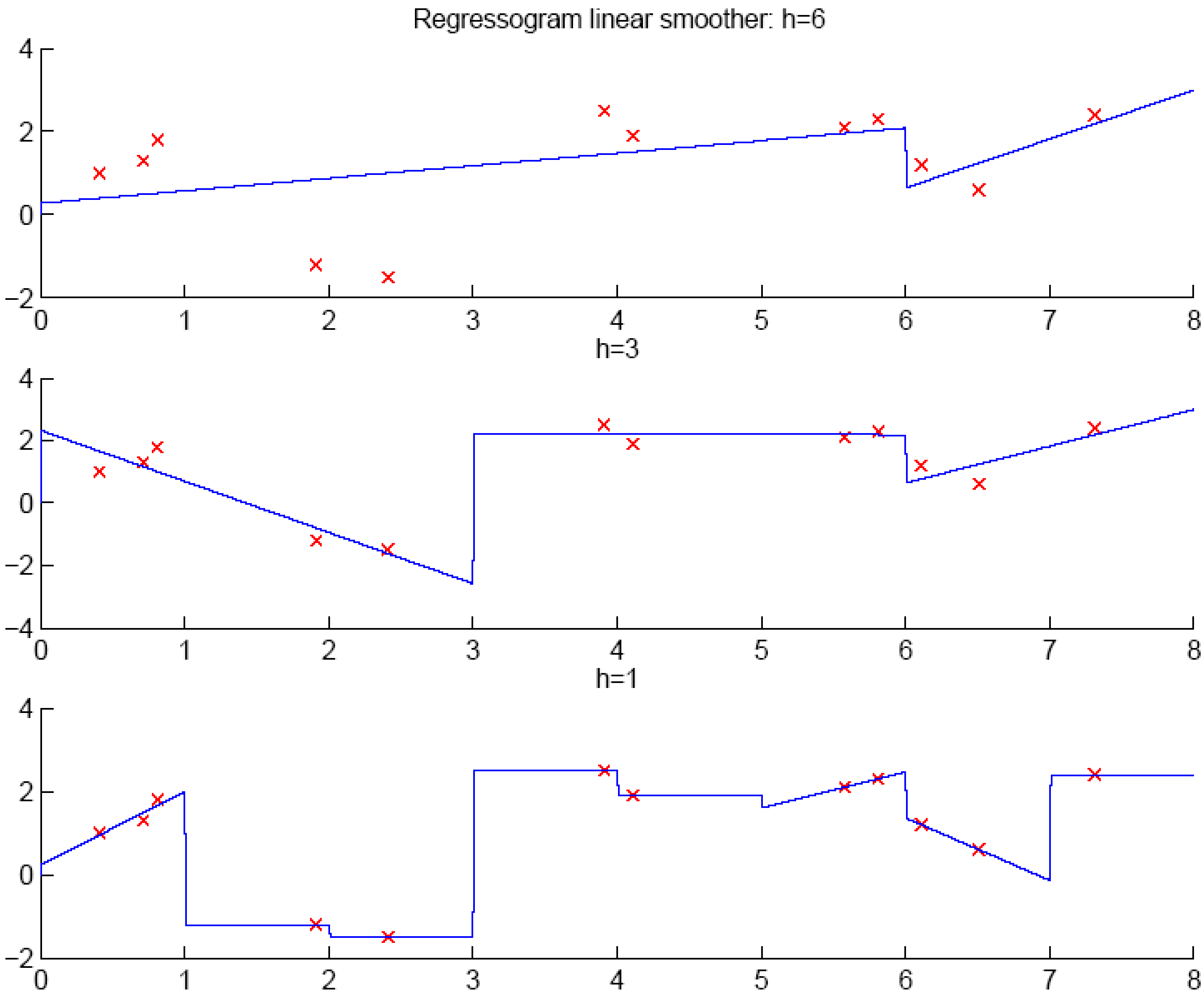
- Aka smoothing models
- Regressogram

$$\hat{g}(x) = \frac{\sum_{t=1}^{N} b(x, x^t) r^t}{\sum_{t=1}^{N} b(x, x^t)}$$

where

$$b(x, x^t) = \begin{cases} 1 & \text{if } x^t \text{ is in the same bin with } x \\ 0 & \text{otherwise} \end{cases}$$


Regressogram smoother: h=6
h=3
h=1
4
2
0
−2
0
1
2
3
4
5
6
7
8


Regressogram linear smoother: h=6
h=3
h=1
4
2
0
−2
−4
0
1
2
3
4
5
6
7
8

Running Mean/Kernel Smoother

- Running mean smoother

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

where

$$w(u) = \begin{cases} 1 & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{cases}$$

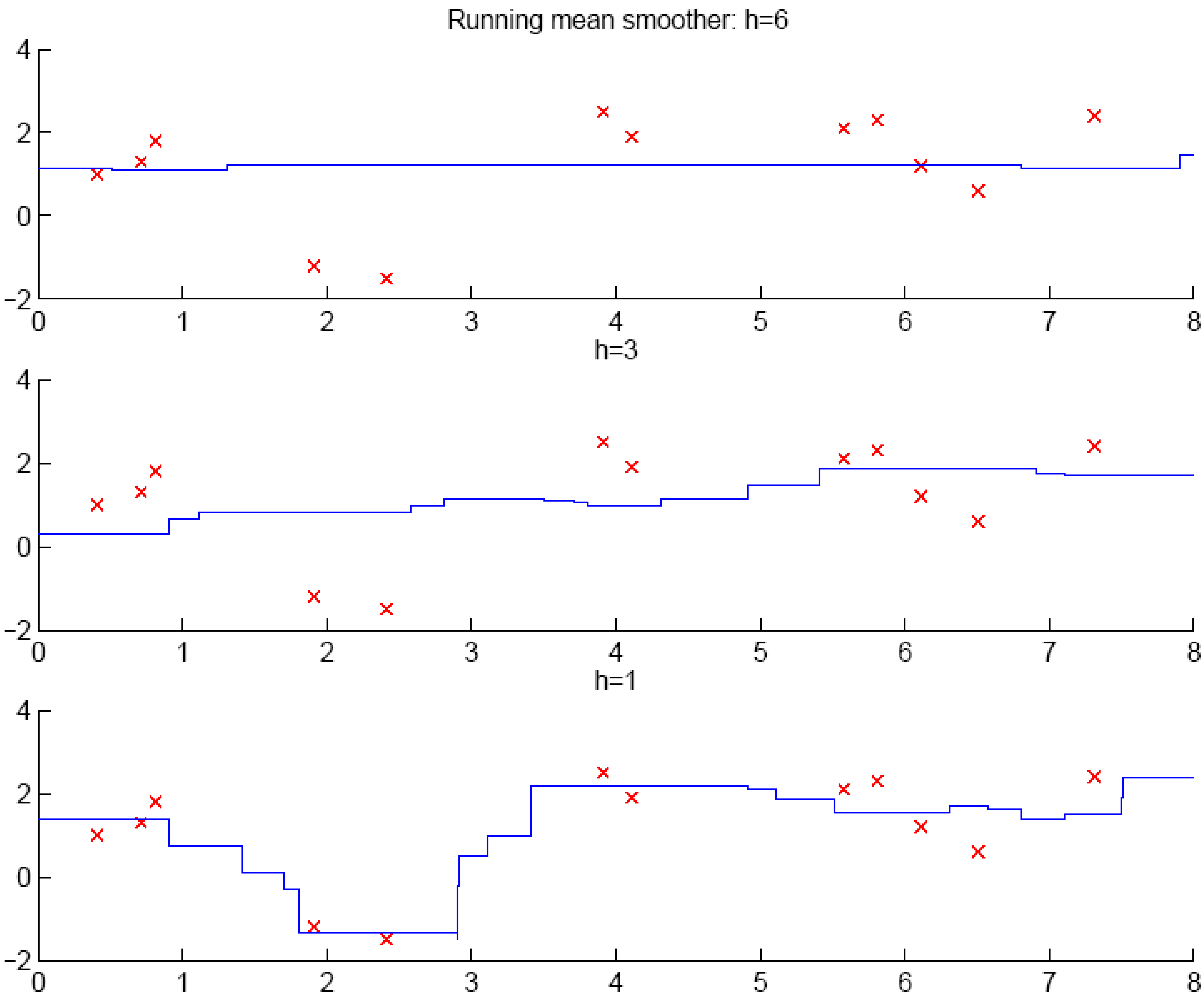
- Running line smoother

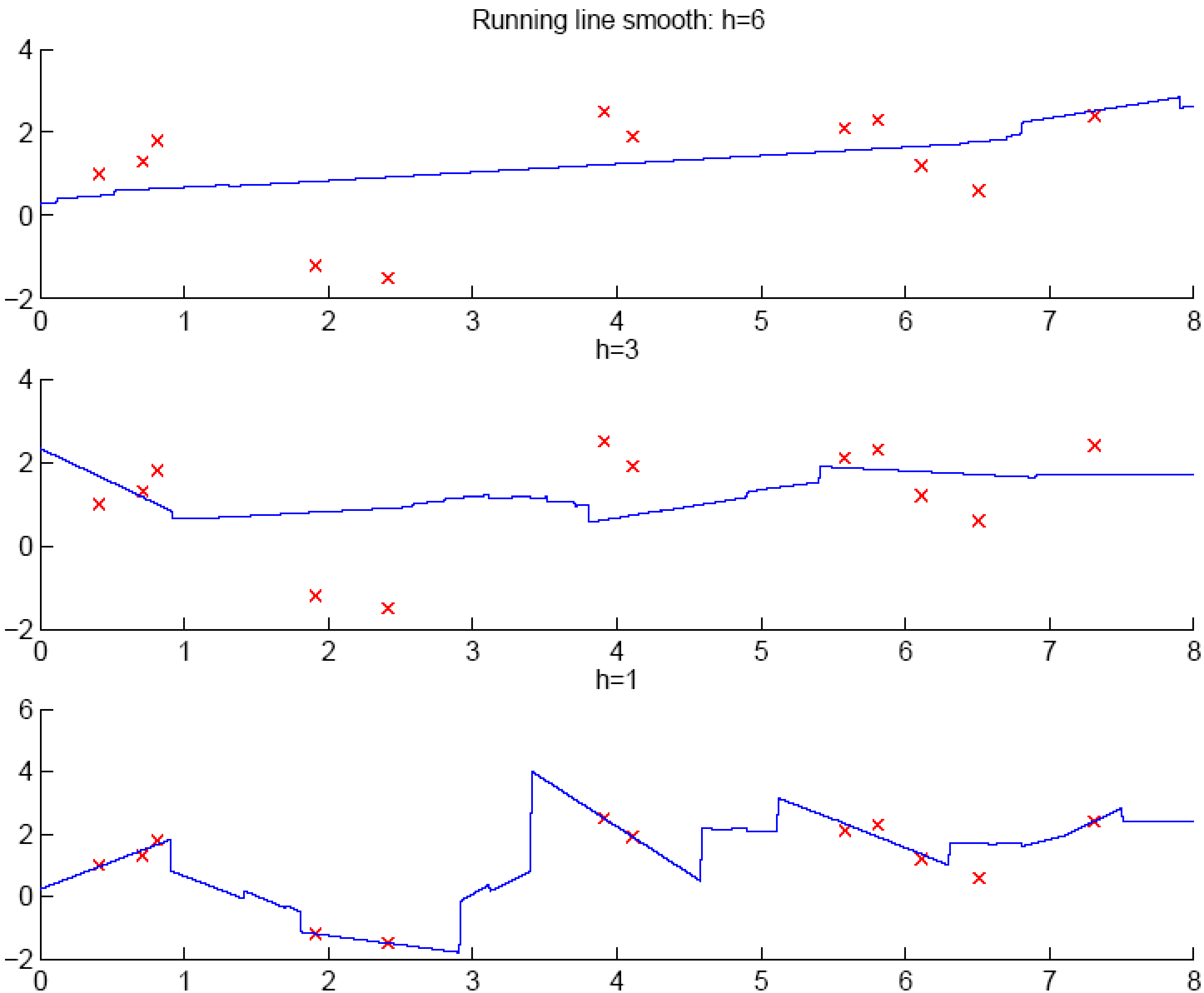
- Kernel smoother

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

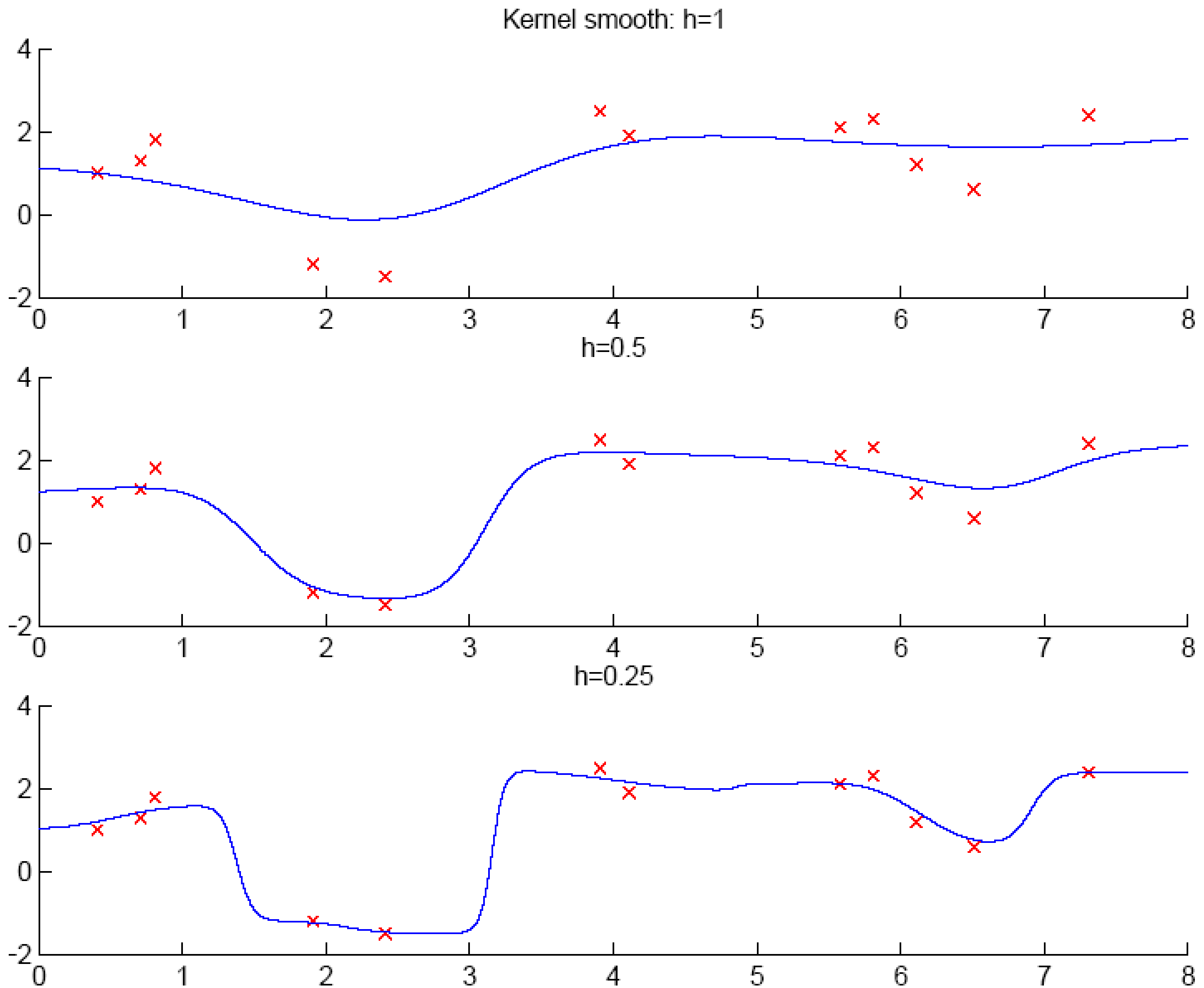
where *K*( ) is Gaussian

- Additive models (Hastie and Tibshirani, 1990)


Running mean smoother: h=6
h=3
h=1
4
2
0
−2
0
1
2
3
4
5
6
7
8

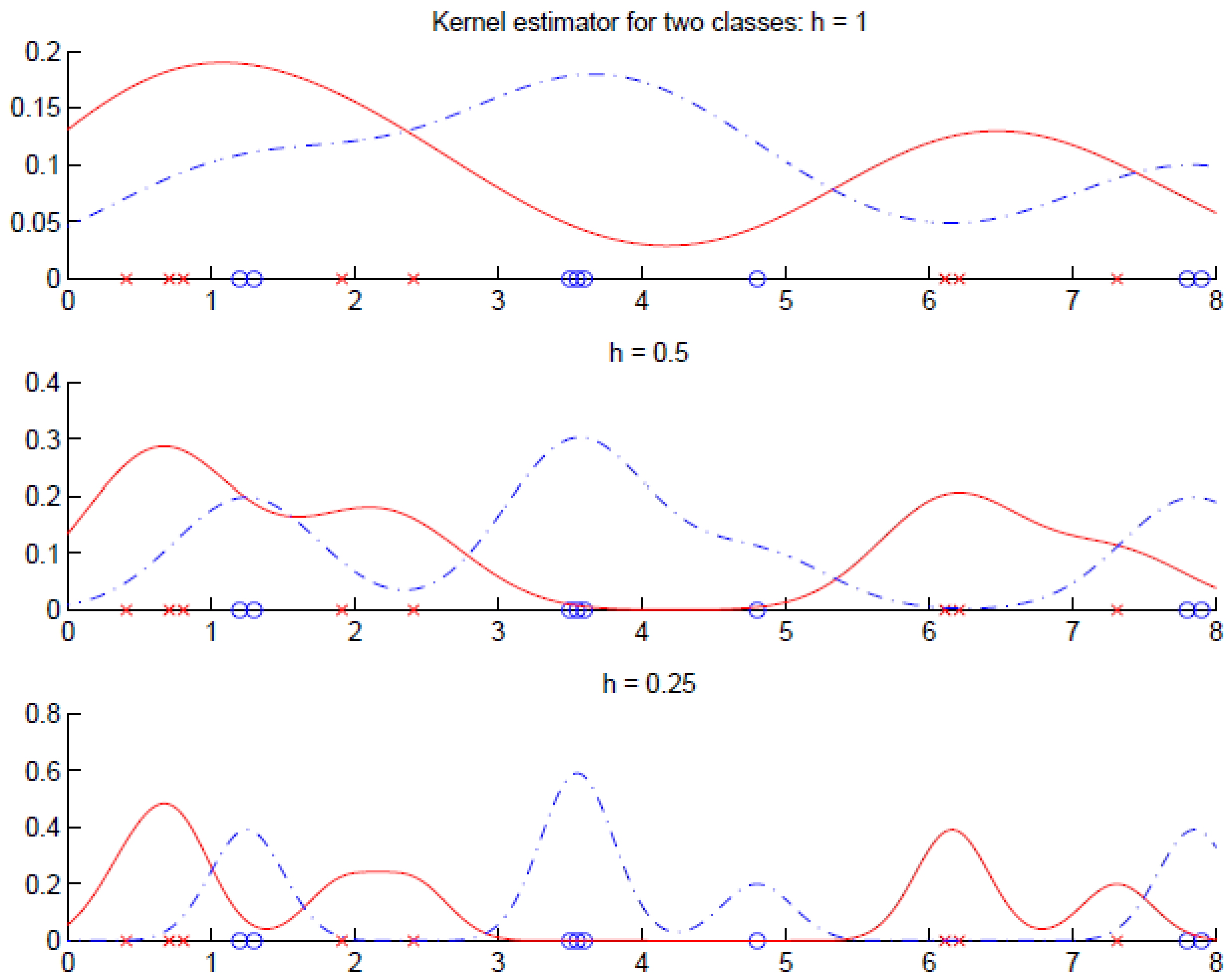


Running line smooth: h=6
h=3
h=1


Kernel smooth: h=1
4
2
0
-2
0
1
2
3
4
5
6
7
8
h=0.5
h=0.25

How to Choose *k* or *h* ?

- When *k* or *h* is small, single instances matter; bias is small, variance is large (undersmoothing): High complexity
- As *k* or *h* increases, we average over more instances and variance decreases but bias increases (oversmoothing): Low complexity
- Cross-validation is used to finetune *k* or *h*.


Kernel estimator for two classes: h = 1
h = 0.5
h = 0.25
0.2
0.15
0.1
0.05
0
0.4
0.3
0.2
0.1
0
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8