FUNDAMENTALS OF MACHINE LEARNING

NONPARAMETRIC METHODS

CSCI3320

Prof. John C.S. Lui, CSE Department, CUHK
Introduction to Machine Learning

Overview

- In the previous chapters, we discussed the parametric and semi-parametric approaches where we assumed that the data came from one or a mixture of probability distributions of known form
- **Nonparametric method:** no such assumption can be made about the input density and the data
- Learn how to:
  - Density estimation
  - Classification
  - Regression

Nonparametric Estimation

- **Parametric Methods:** model valid over whole input space
  - **Regression:** assume linear model, output is a linear transformation of inputs
  - **Classification:** assume Gaussian and inputs are drawn from it
  - **Advantage:** reduce to a small # of parameters estimation
  - **Disadvantage:** assumption may not hold
- **Semi-parametric Methods:** assume "mixture" of models.
- **Nonparametric:** Assume *similar inputs have similar outputs*
- Functions (e.g., pdf, discriminant, regression) change smoothly
- **Approach:** from training set, find "*similar*" instances using suitable **distance measure**, interpolate them to find the right output
- Keep the training data; "**let the data speak for itself**"
- Given x, find a small number of closest training instances and interpolate
- Aka **lazy/memory-based/case-based/instance-based learning**

Big Picture

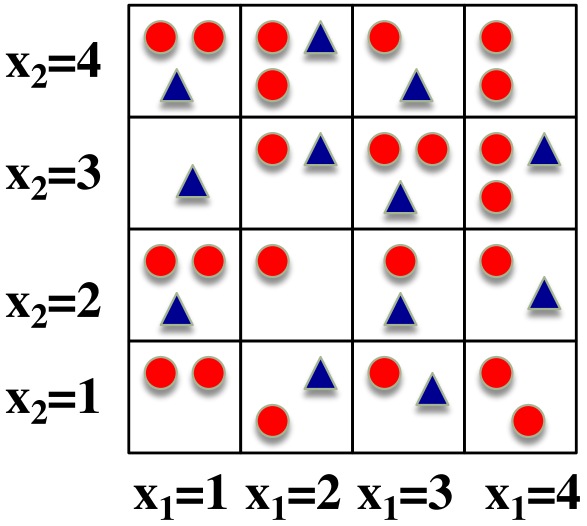
- In previous lectures, we do classification via:

$$p(C_i|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_i)P(C_i)$$

- Find the above probability for ALL CLASSES, select the class with the highest probability
- We studied various methods to estimate $p(\boldsymbol{x}|C_i)$
  - Naïve Bayesian Decision
  - Parametric
  - Semi-parametric
- **Non-parametric method**: *derive the above from the data*

Remember Bayesian Decision ?

- Consider two classes first. Class 1 Red, Class 2 Blue



Total red points=23
Total blue points=12

- $P(C_1)$= total red/total points=23/35; $P(C_2)$=total blue/total point=12/35
- $P(x_1=1,x_2=1|C_1) = 2/23$; $P(x_1=1,x_2=3|C_1) = 0/23$;
- $P(x_1=1,x_2=1|C_2) = 0/12$; $P(x_1=1,x_2=3|C_2) = 1/12$;
- $P(C=1|x_1=1,x_2=2)=?$ $P(x_1=1,x_2=2|C=1)P(C=1)=2/23*23/35=2/35$
- $P(C=0|x_1=1,x_2=2)=?$ $P(x_1=1,x_2=2|C=0)P(C=0)=1/12*12/35=1/35$
- **If there is no instance in (x_1=2,x_2=2), then we can we do?**

Nonparametric Density Estimation

- Given the training set $X = \{x^t\}_{t=1}^N$, w/c are IID and are drawn from some unknown pdf $p(\cdot)$ for the scalar x
- Denote $\hat{p}(\cdot)$ as the estimator of $p(\cdot)$
- Divide data into bins of size h (or interval)
- **Histogram**: input space is divided into equal-sized interval *h*

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

$\leftarrow$ so that $\sum_{\forall x} \hat{p}(x)h = 1$

- **Naive estimator**:

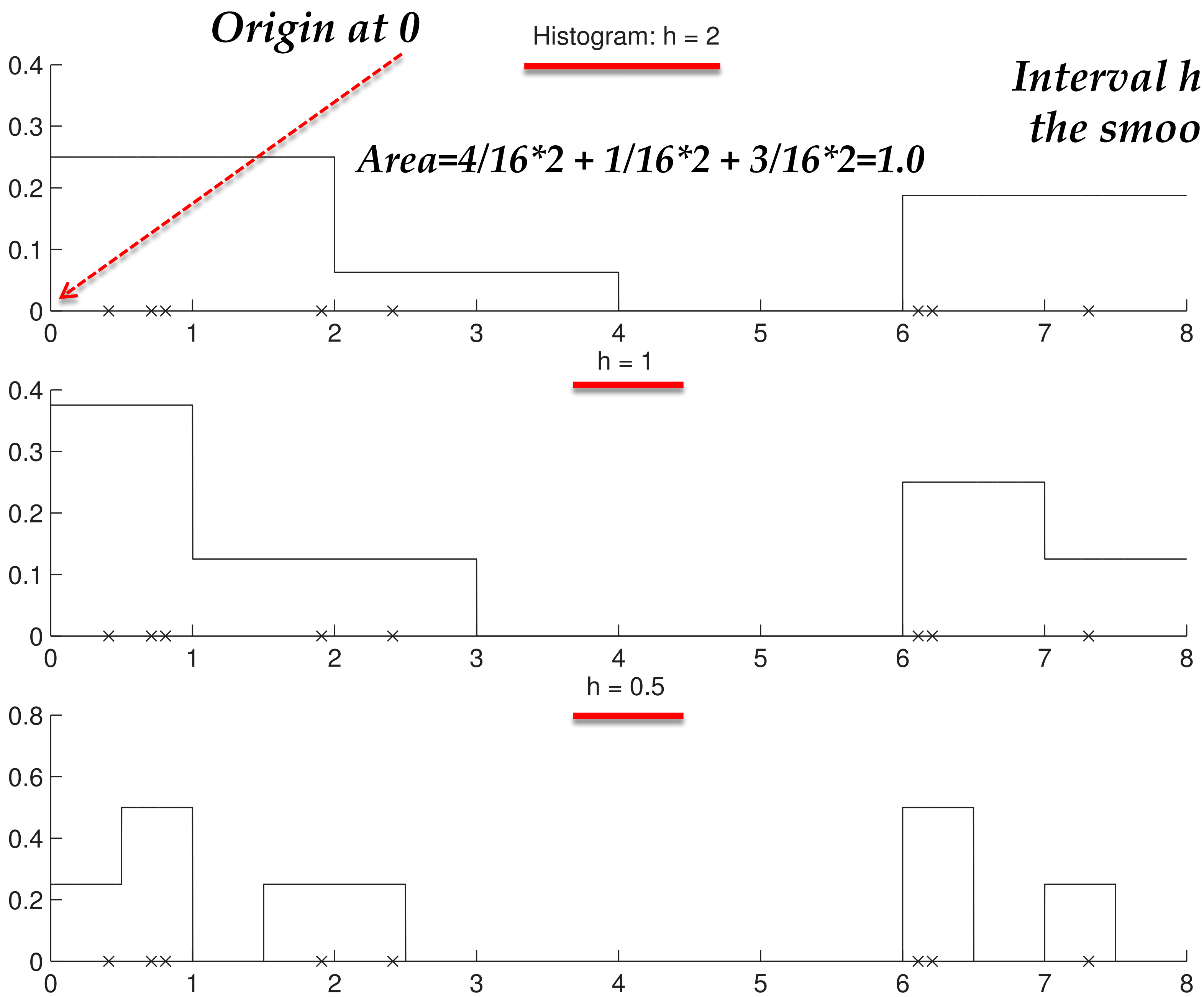
$$\hat{p}(x) = \frac{\#\{x - h/2 < x^t \leq x + h/2\}}{Nh}$$

or

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

Histogram: *N=8*

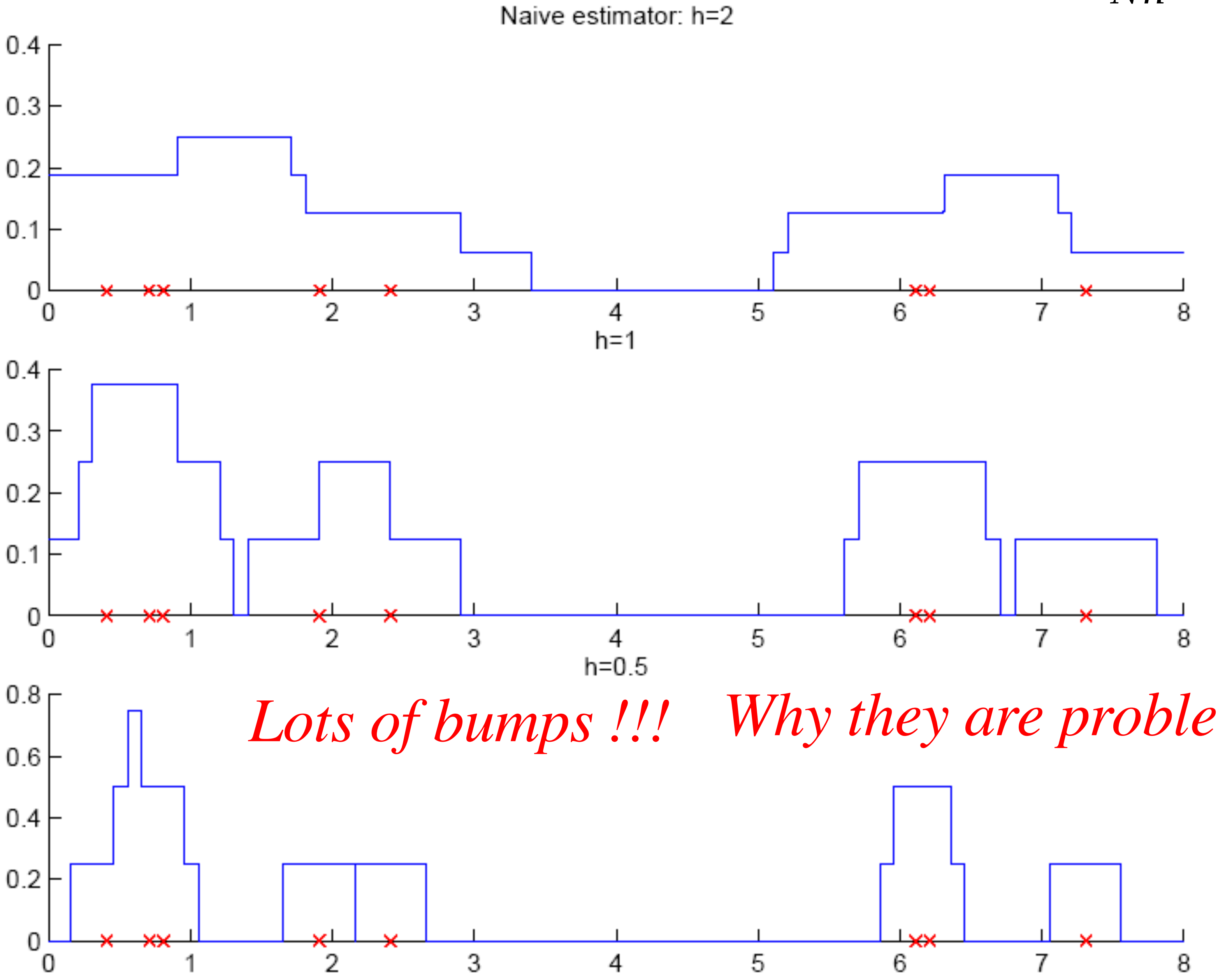
$$\hat{p}(x) = \frac{\#\{x^t \text{ in the same bin as } x\}}{Nh}$$



Interval h affects the smoothness

Naive estimator: *N=8*

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



Lots of bumps !!! *Why they are problematic?*

Kernel Estimator

- To be a "***smooth***" estimate, we use a smooth weight function called *kernel function*
- Kernel function, e.g., Gaussian kernel:

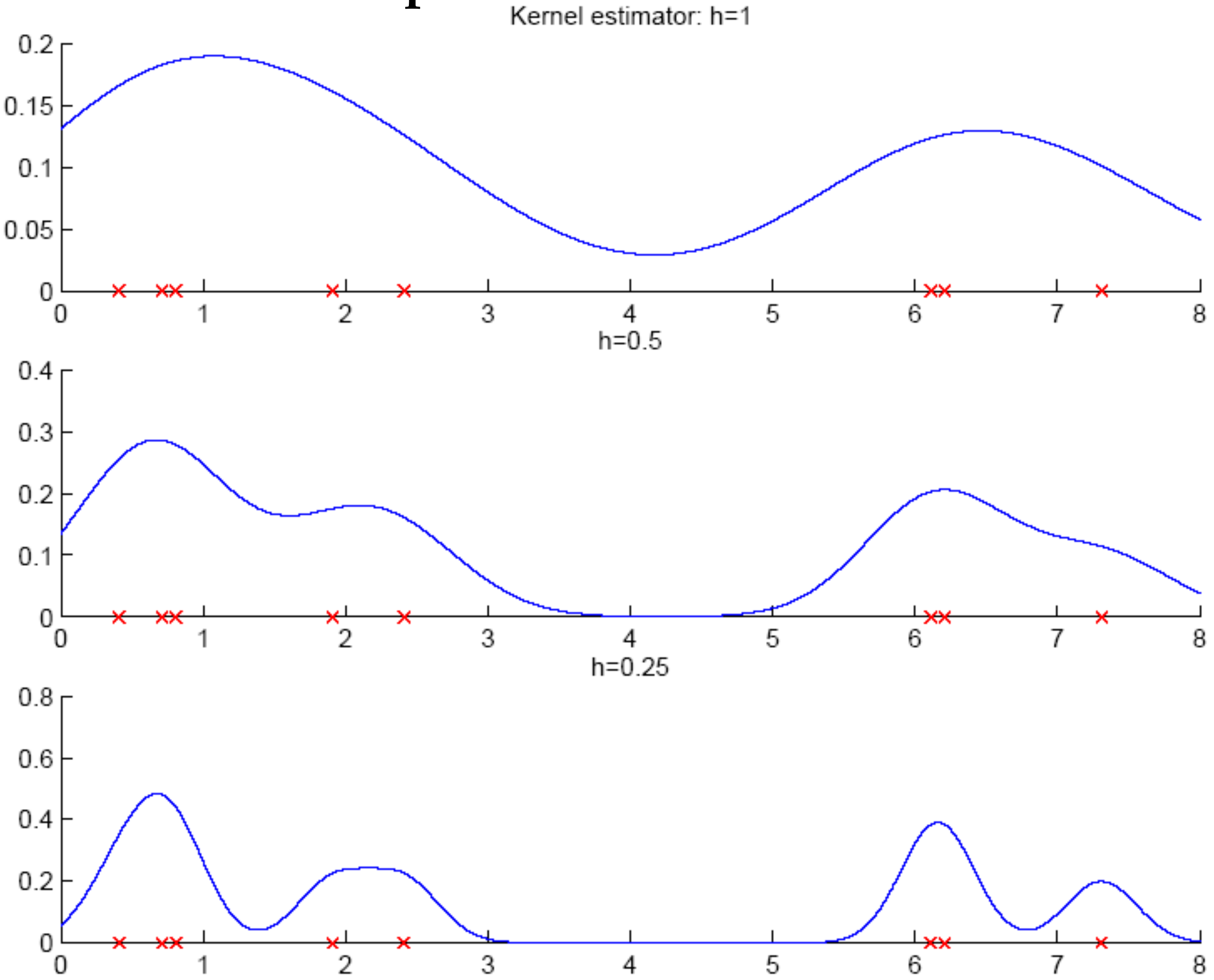
$$K(u) = \frac{1}{\sqrt{2\pi}} e^{\left[-\frac{u^2}{2}\right]}$$

- Kernel estimator (or Parzen windows)

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

- Kernel $K()$ determines the **shape** of influence, h determines the **width.** All x^t can affect estimate at x, the effect decreases smoothly as $|x - x^t|$ increases

Comment on the impact of *h*



k-Nearest Neighbor (kNN) Estimator

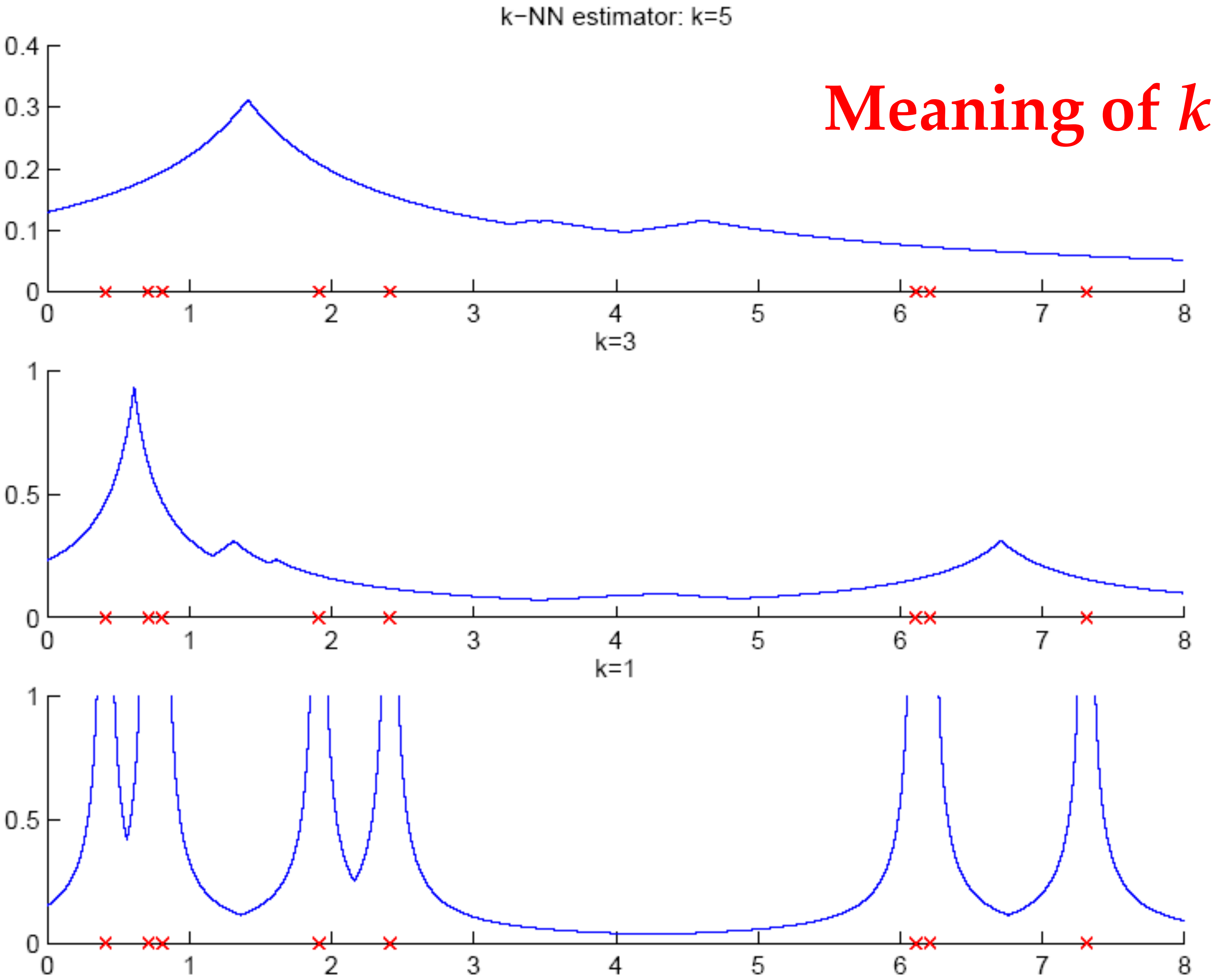
- □ Instead of fixing bin width h and counting the number of instances, fix the instances (neighbors) k and then check bin width
- □ Define $d_k(x)$, distance to k^{th} closest instance to x, *for* $k = 1,2, \dots, N$,

$$d_1(x) \leq d_2(x) \leq \cdots \leq d_N(x)$$

- □ Distance metric can be $|a - b|$
- □ The *k-nearest neighbor (k-nn) density estimate* on x is

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

Meaning of *k*


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
0
1
2
3
4
5
6
7
8
k=1
1
0.5
0
0
1
2
3
4
5
6
7
8

Generalization to Multivariate Data

- Given *d*-dimensional observations: $X = \{\boldsymbol{x}^t\}_{t=1}^N$

- The multivariate kernel density estimator is

$$\hat{p}(\boldsymbol{x}) = \frac{1}{Nh^d} \sum_{t=1}^{N} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}^t}{h}\right) \qquad \text{with } \int_{\mathcal{R}^d} K(\boldsymbol{x}) d\boldsymbol{x} = 1$$

- One possible kernel function is multivariate Gaussian

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

- Watch out for ***curse of dimensionality***. E.g., if $\boldsymbol{x}$ is eight-dimensional and we have ten bins per dimension, then we have 10^8 bins. We may have many empty bins !!!

Nonparametric Classification

- □ Estimate class-conditional densities $p(\boldsymbol{x}|C_i)$
- □ Kernel estimator (*all N_i points have influence on $\boldsymbol{x}$*)

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

where r_i^t is 1 if $\boldsymbol{x}^t \in C_i$ and 0 otherwise, $N_i = \sum_t r_i^t$; $\hat{P}(C_i) = \frac{N_i}{N}$

- □ Then the discriminant is

$\boldsymbol{x}$ is assigned to the class with the largest discriminant

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

each instance in C_i gives a weighted vote of its class

can be ignored

Nonparametric Classification

- The weight of the vote is given by the kernel function *K*
- *k-nn* estimator

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

  - k_i is # of neighbors out of k nearest that belong to C_i
  - $V^i(\boldsymbol{x})$ is the volume of the *d*-dimensional hyper-sphere centered at $\boldsymbol{x}$, with radius $r = |\boldsymbol{x} - \boldsymbol{x}_{(k)}|$, where $\boldsymbol{x}_{(k)}$ is the k^{th} nearest observation to $\boldsymbol{x}$ (among all neighbors from all classes of $\boldsymbol{x}$)
- Then we have

$$\hat{P}(C_i|\boldsymbol{x}) = \frac{\hat{p}(\boldsymbol{x}|C_i)\,\hat{P}(C_i)}{\hat{p}(\boldsymbol{x})} = \frac{k_i}{k}$$

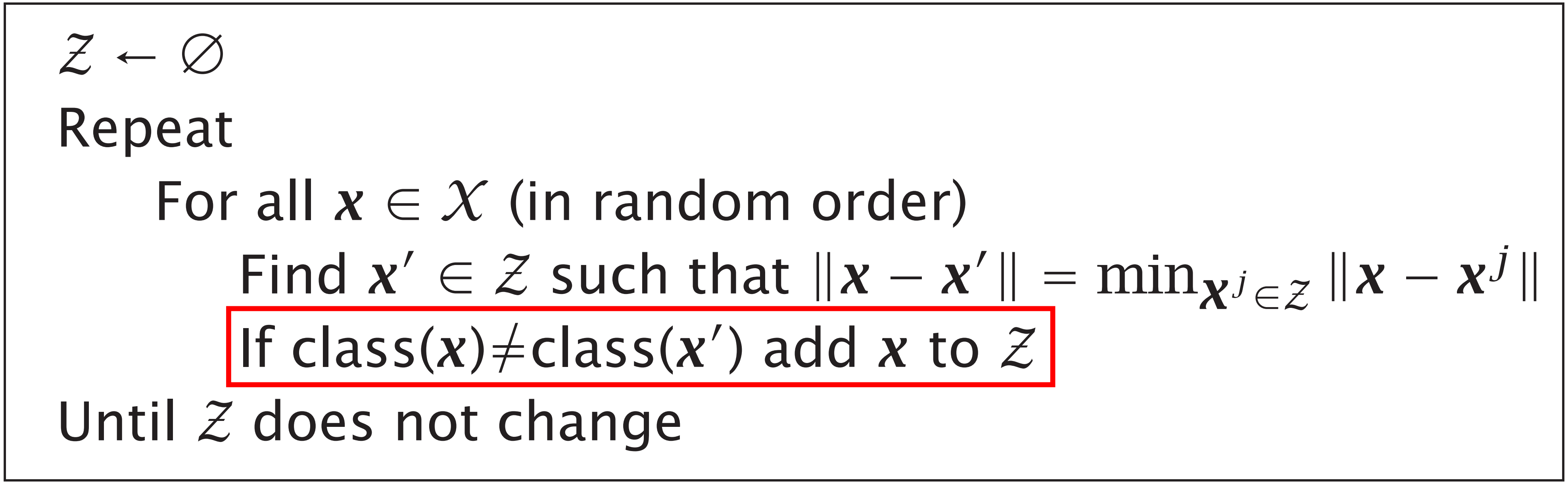
- *k-nn* classifier $\boldsymbol{x}$ assigns to the class having the most examples among the *k* neighbors of the input

Condensed Nearest Neighbor

- □ Time/space complexity of k-NN is O(N)
- □ Condensing methods are to decrease the number of stored instances without degrading performance
- □ **Idea:** Find the smallest subset $\mathcal{Z}$ of $\mathcal{X}$ that is small and is accurate in classifying $\mathcal{X}$ (Hart, 1968)
- □ **Condensed Nearest Neighbor** where **1-nn** is used as nonparametric estimator for classification
- □ **1-nn** approximates the discriminant in a piecewise linear manner, only instances that define the discriminant need to be kept (*minimal consistent subset*)

Condensed Nearest Neighbor

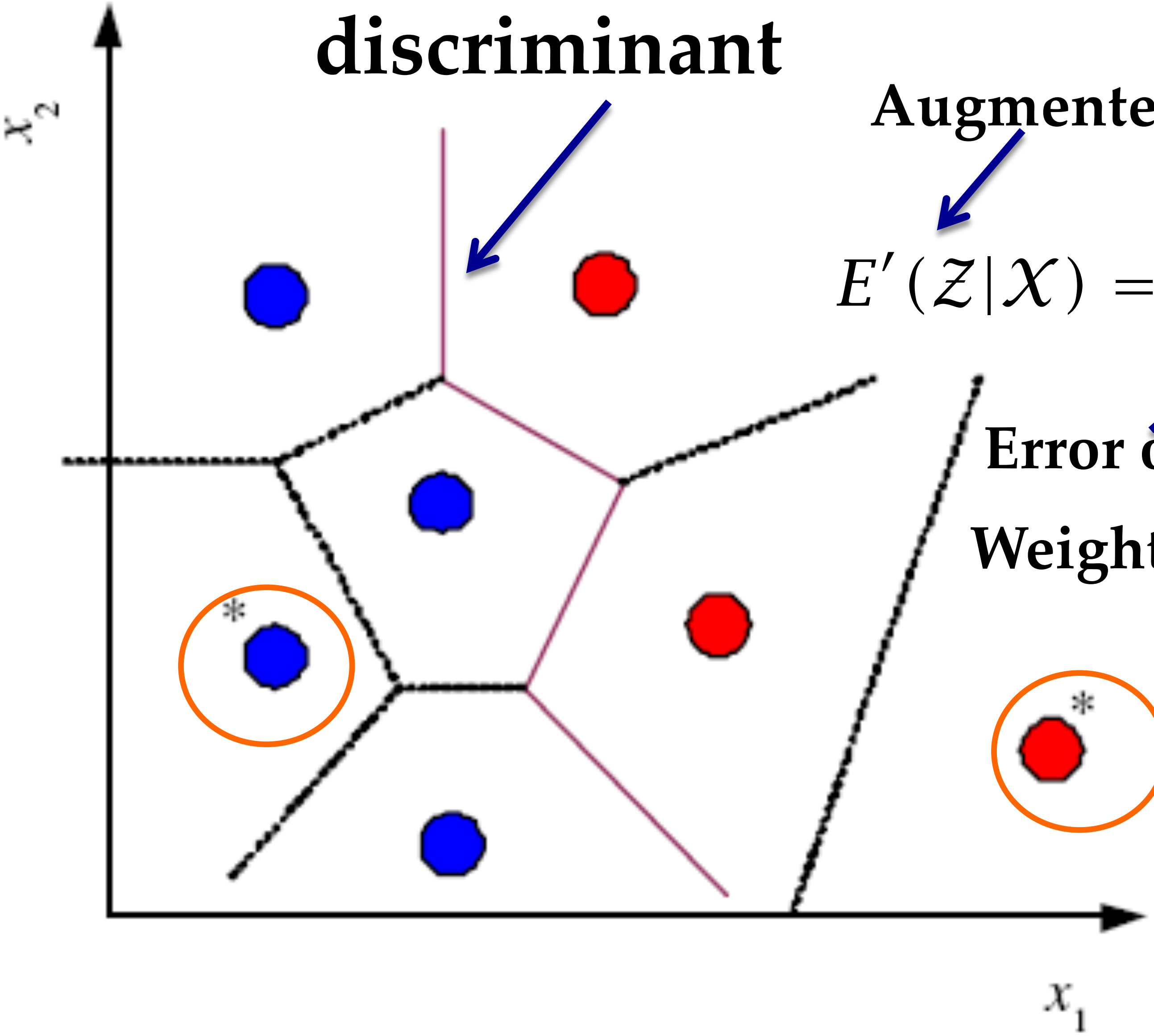
- Incremental (or greedy) algorithm: Add instance if needed



```
$\mathcal{Z} \leftarrow \emptyset$
Repeat
    For all $\boldsymbol{x} \in \mathcal{X}$ (in random order)
        Find $\boldsymbol{x}' \in \mathcal{Z}$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\| = \min_{\boldsymbol{x}^j \in \mathcal{Z}} \|\boldsymbol{x} - \boldsymbol{x}^j\|$
        If class($\boldsymbol{x}$)$\neq$class($\boldsymbol{x}'$) add $\boldsymbol{x}$ to $\mathcal{Z}$
Until $\mathcal{Z}$ does not change
```

It's a heuristic and NP-complete

Condensed Nearest Neighbor



discriminant

Augmented error function

$$E'(\mathcal{Z}|\mathcal{X}) = E(\mathcal{X}|\mathcal{Z}) + \lambda|\mathcal{Z}|$$

Error on ***X*** **storing** ***Z***

Weighted on ***Z's cardinality***

Distance-based Classification

- □ Find a distance function $D(\boldsymbol{x}^r, \boldsymbol{x}^s)$ such that
  if $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$ 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
- □ Previously, we see the parametric approach on Gaussian classes using *nearest mean classifier*

$$\mathcal{D}(\boldsymbol{x}, \boldsymbol{m}_i) = \min_{j=1}^{K} \mathcal{D}(\boldsymbol{x}, \boldsymbol{m}_j)$$

- □ For hyperspheric Gaussians where dimensions are independent and all in the same scale, it is distance-based: $\mathcal{D}(\boldsymbol{x}, \boldsymbol{m}_i) = \|\boldsymbol{x} - \boldsymbol{m}_i\|$ or $\mathcal{D}(\boldsymbol{x}, \boldsymbol{m}_i) = (\boldsymbol{x} - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1} (\boldsymbol{x} - \boldsymbol{m}_i)$

Distance-based Classification

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

  if $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$ 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., use the Mahalanobis distance **M** (a *dxd* matrix)

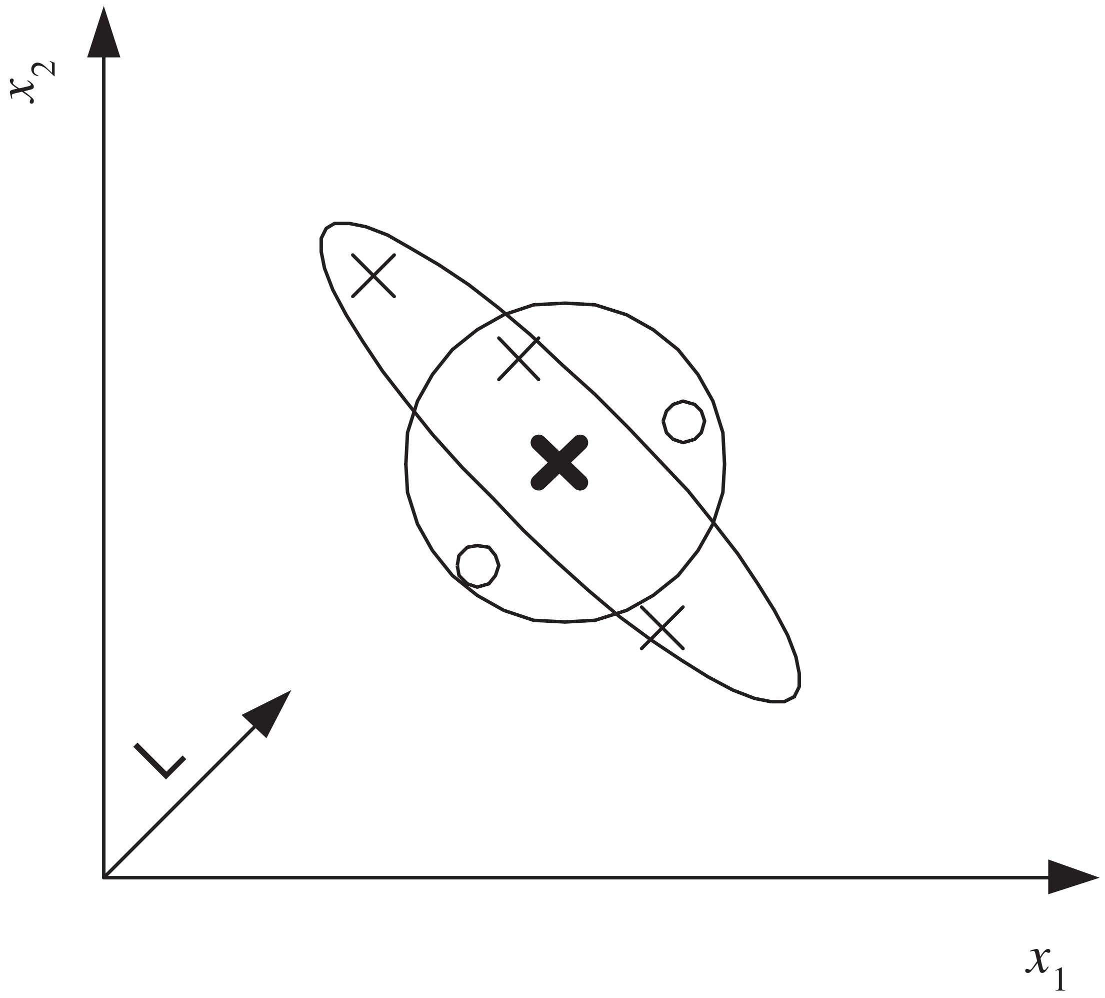
$$\mathcal{D}(\mathbf{x}, \mathbf{x}^t | \mathbf{M}) = (\mathbf{x} - \mathbf{x}^t)^T \mathbf{M} (\mathbf{x} - \mathbf{x}^t)$$

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*

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

- □ Similarity-based representation using similarity scores
- □ Large-margin nearest neighbor or SVM (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

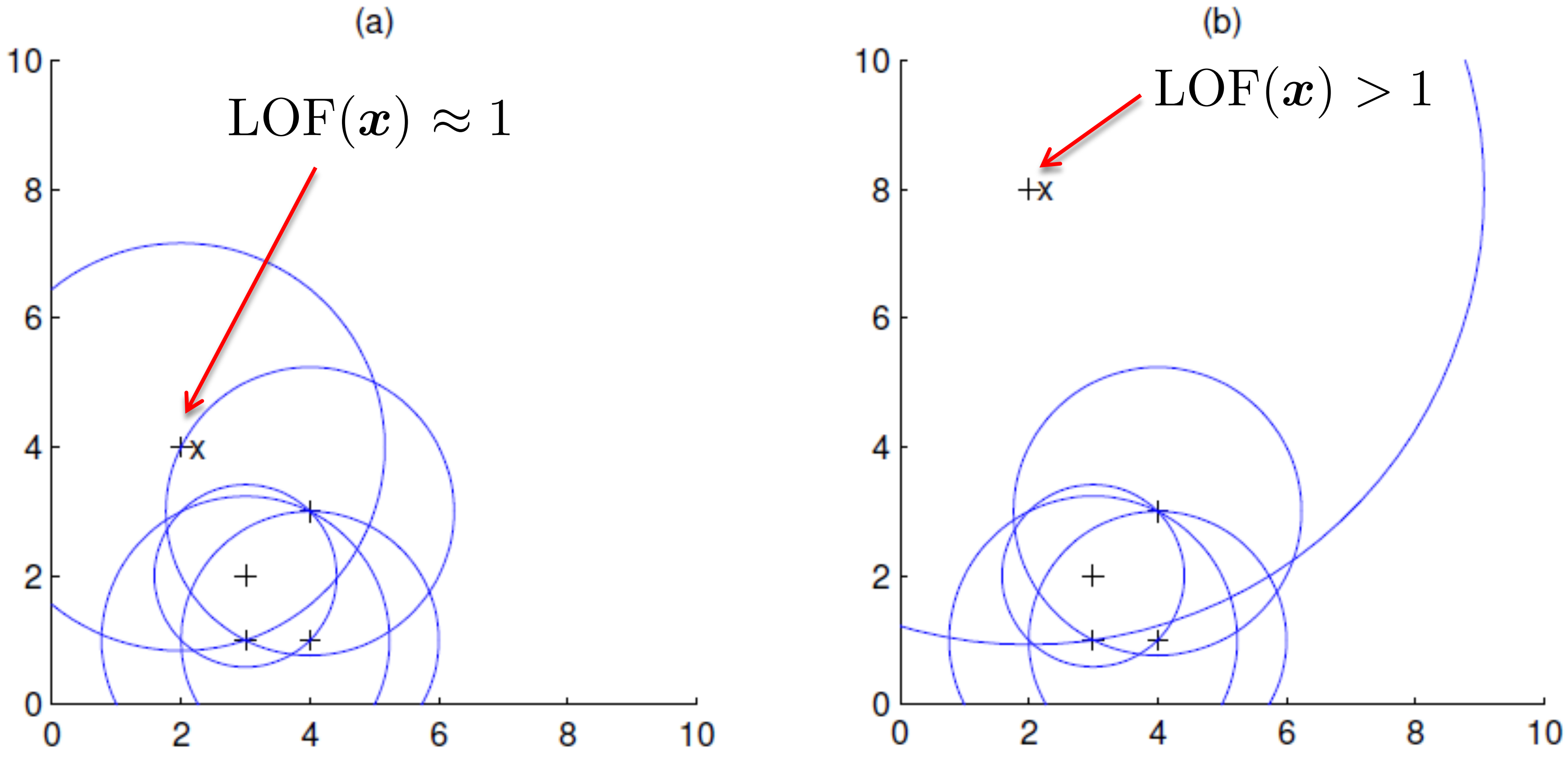
Local Outlier Factor (LOF)

$d_k(\boldsymbol{x})$: distance between $\boldsymbol{x}$ and its k^{th} nearest neighbor

$\mathcal{N}(\boldsymbol{x})$: set of instances w/c are neighbors of $\boldsymbol{x}$

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

compare $d_k(\boldsymbol{x})$ with the average of $d_k(\boldsymbol{s})$



Nonparametric Regression

- In regression, we have training set $X = \{x^t, r^t\}$ where $r^t \in \mathcal{R}$
- We assume $r^t = g(x^t) + \epsilon$
- In parametric regression, we assume a ***polynomial*** of certain order and find its coefficient that minimize the sum of squared error on the training set
- **Nonparametric regression** is used when no such polynomial can be assumed
- In nonparametric regression, given $\boldsymbol{x}$, find neighborhood of $\boldsymbol{x}$ and average the r values in the neighborhood to calculate $\hat{g}(\boldsymbol{x})$

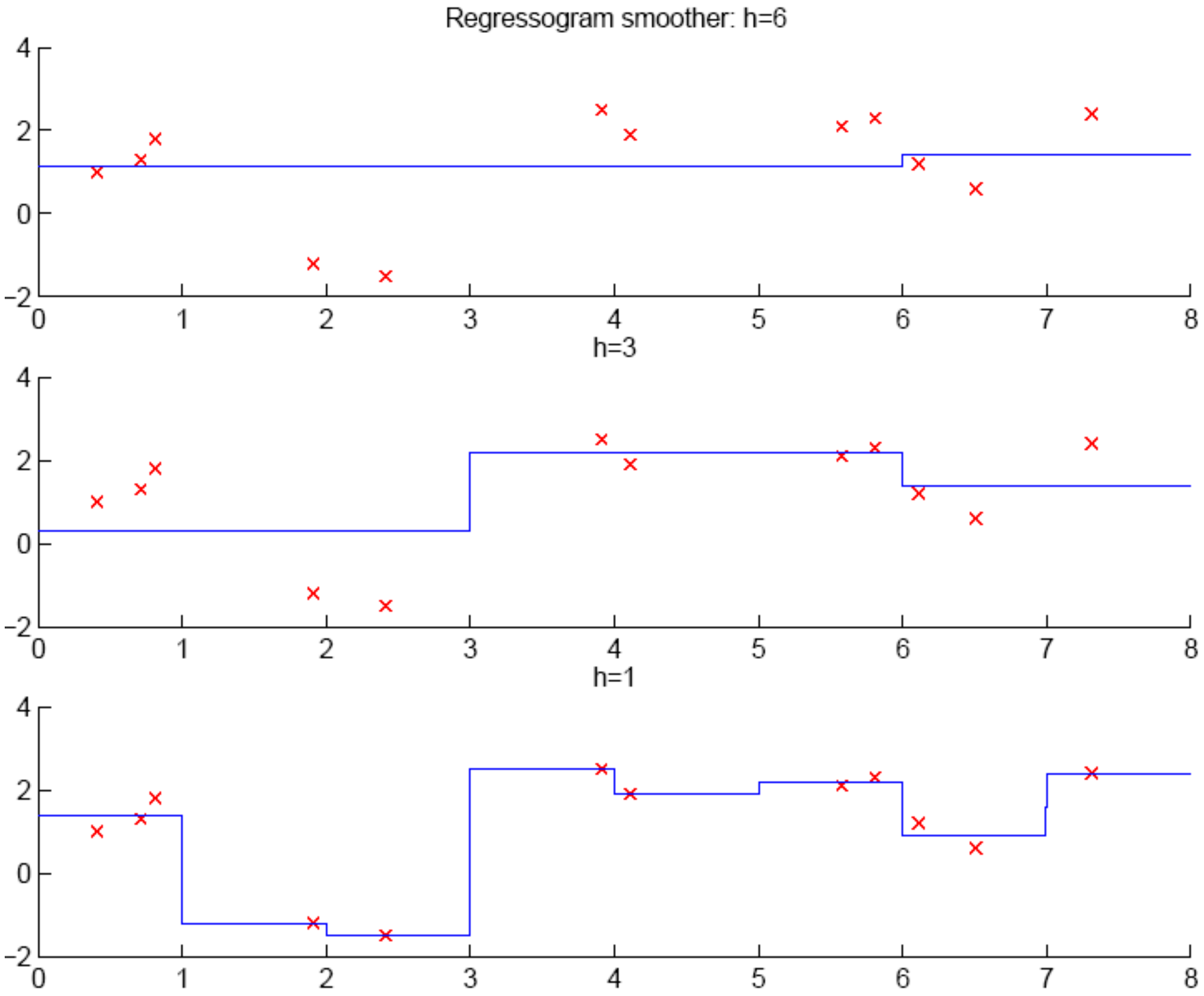
Nonparametric Regression

- Also known as the *smoothing models*
- Various method to define "neighborhood" & "average"
- If we define an origin and bin width, and average r values in the bin, we get a regressogram

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

where

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


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

Nonparametric Regression

- □ Note that there are "discontinuities" at bin boundaries
- □ To deal with that, one can use some smoothing

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

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

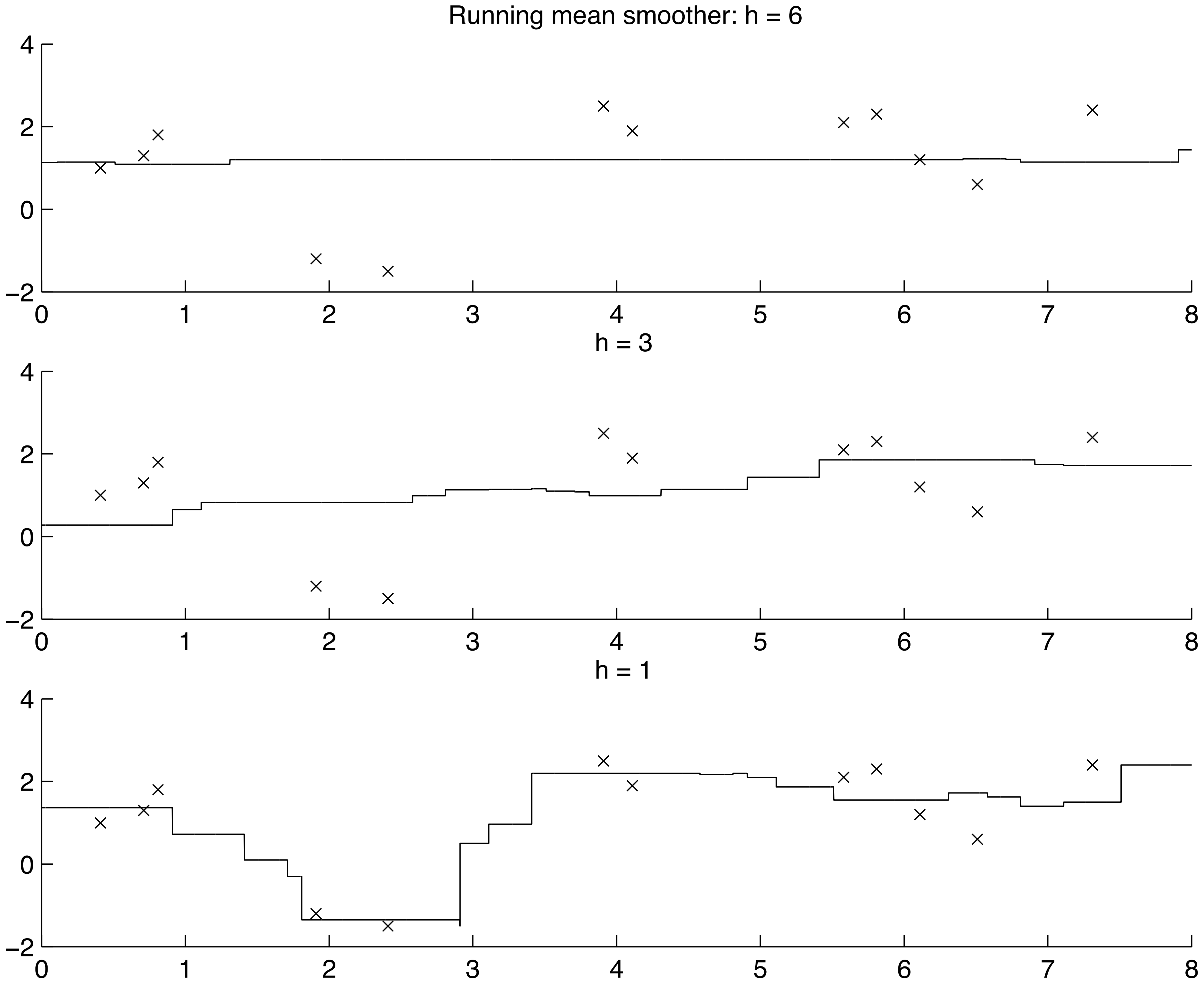
Kernel smoother

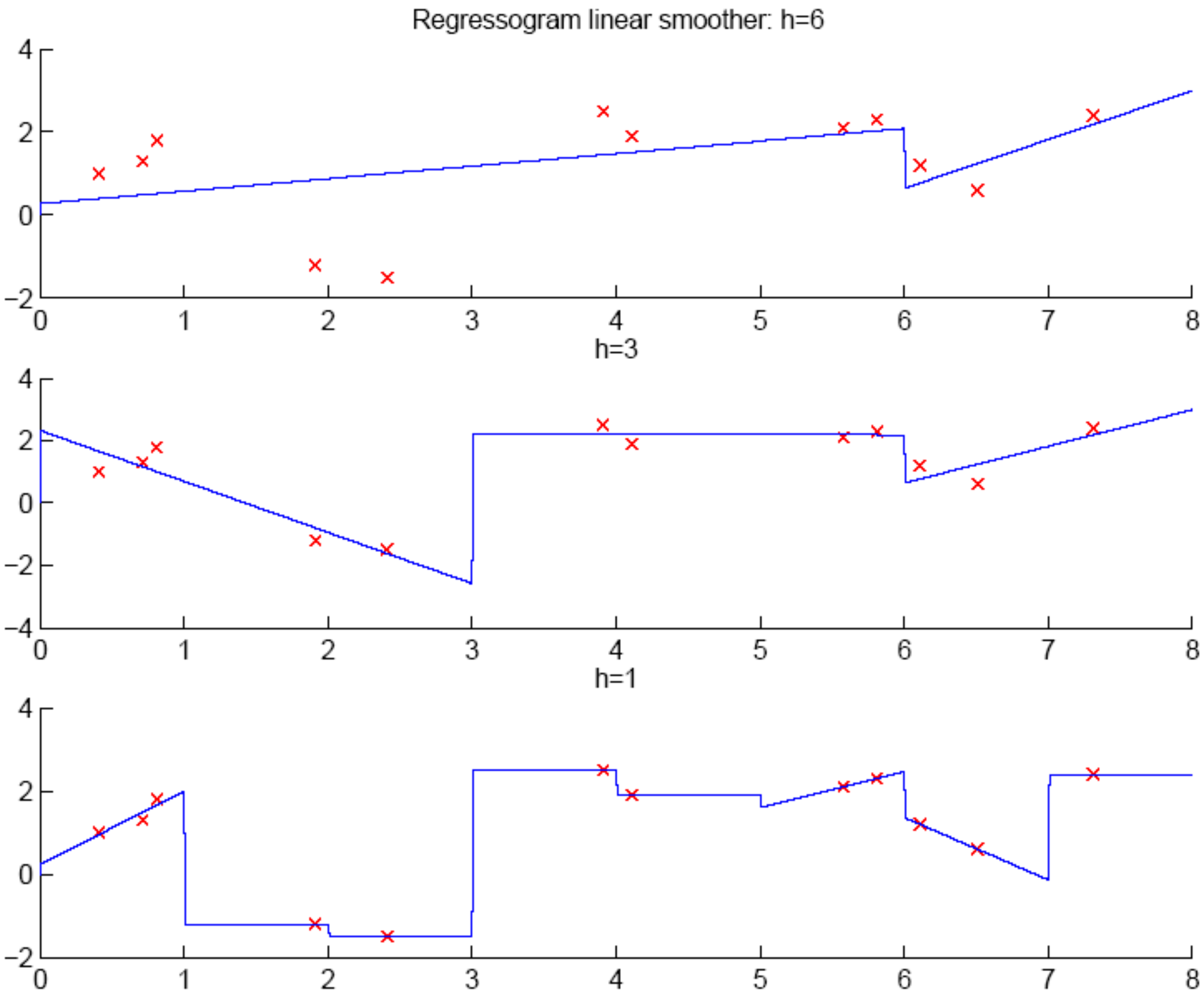
$$\hat{g}(x) = \frac{\sum_t K\left(\frac{x-x^t}{h}\right) r^t}{\sum_t K\left(\frac{x-x^t}{h}\right)}$$

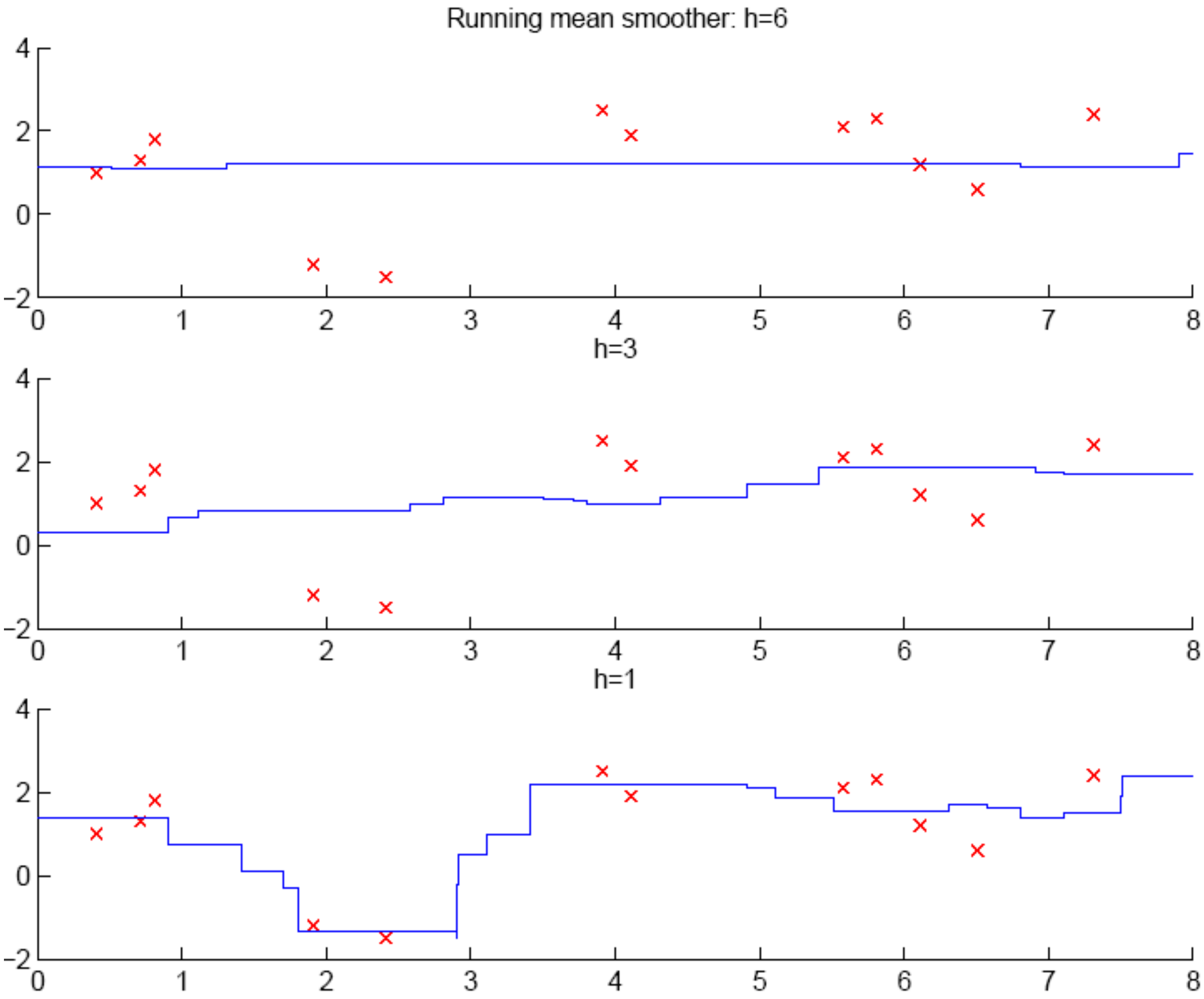
where *K*( ) is Gaussian

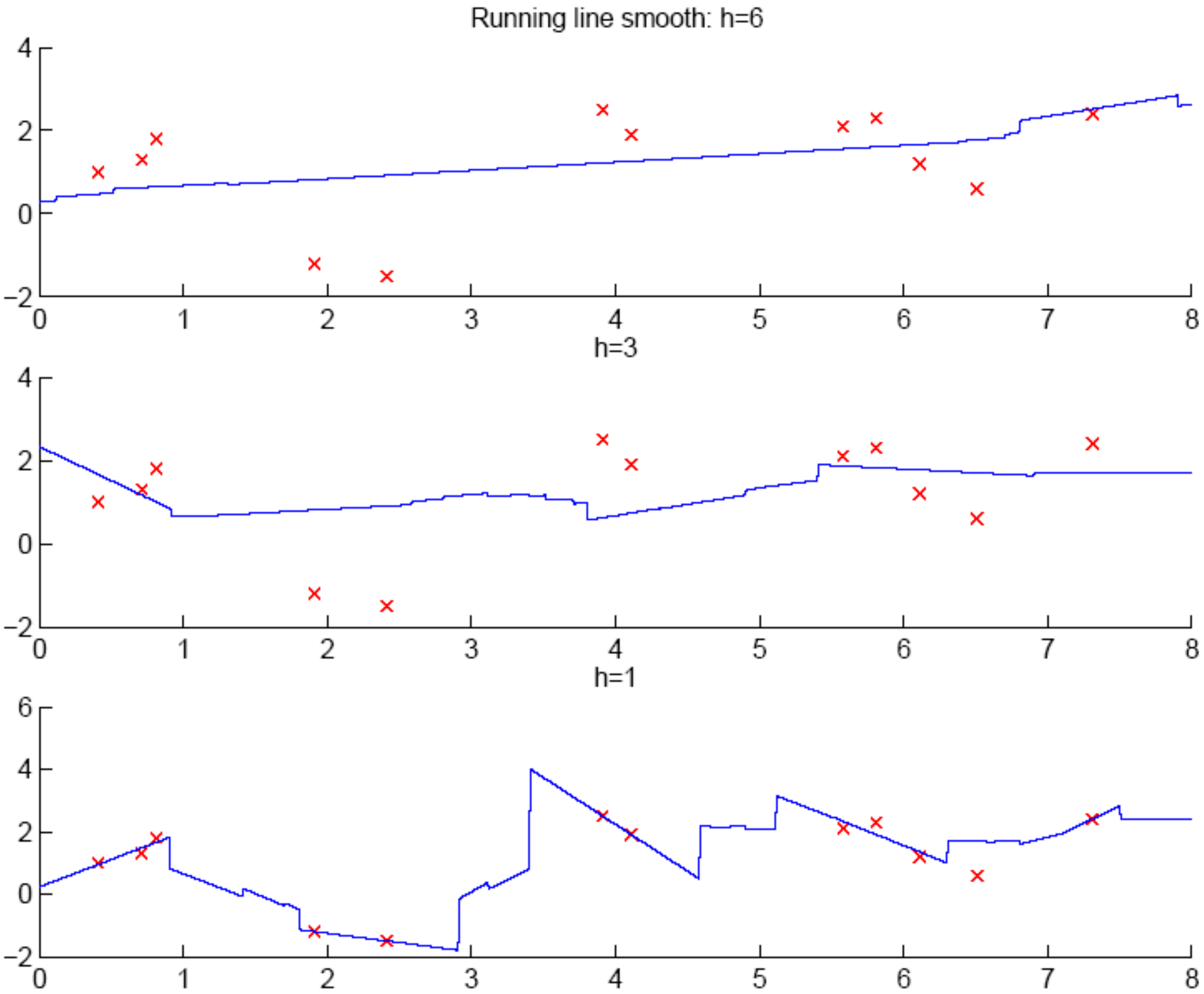
- □ Additive models (Hastie and Tibshirani, 1990)

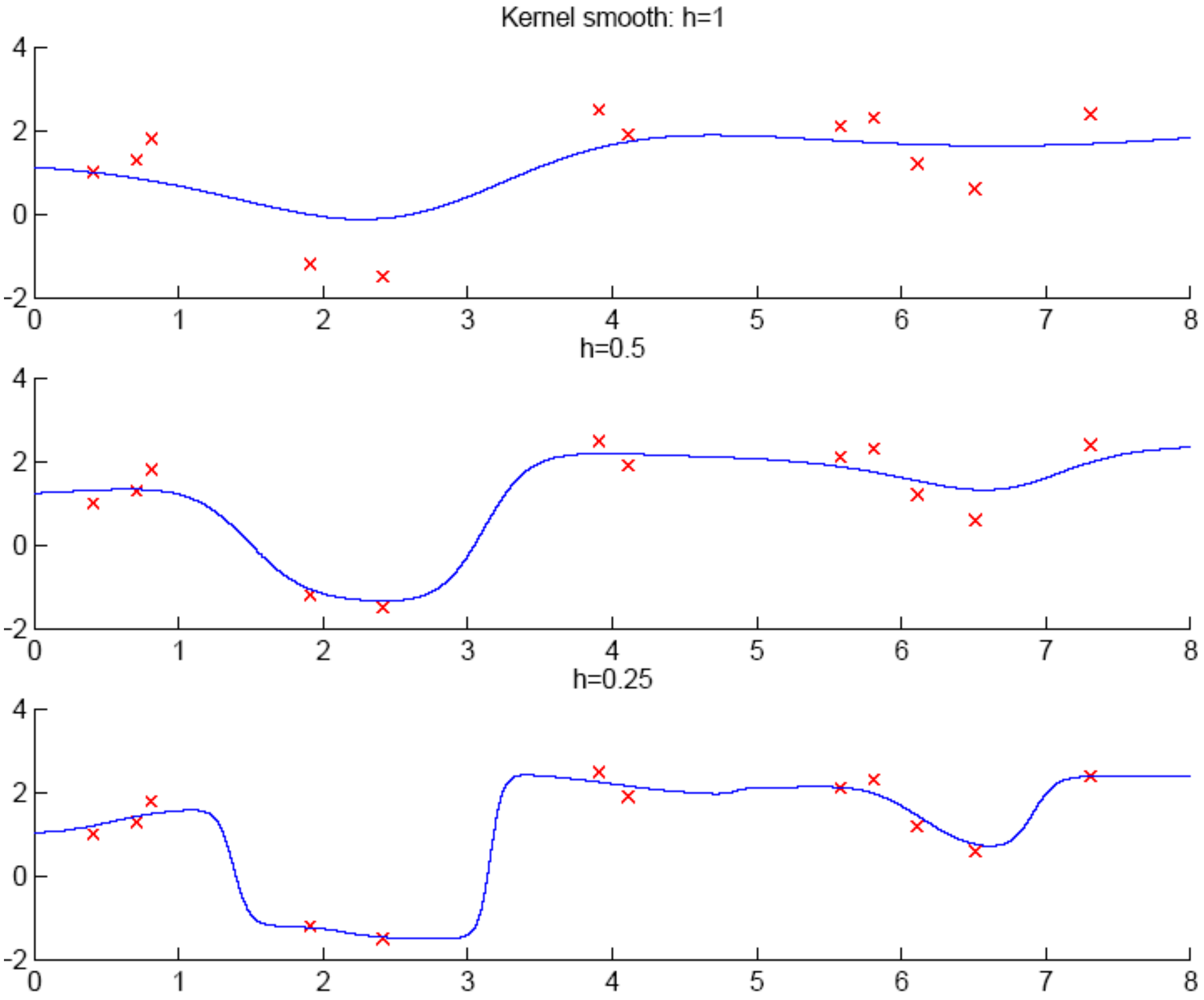
- □ Running line smoother


Running mean 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

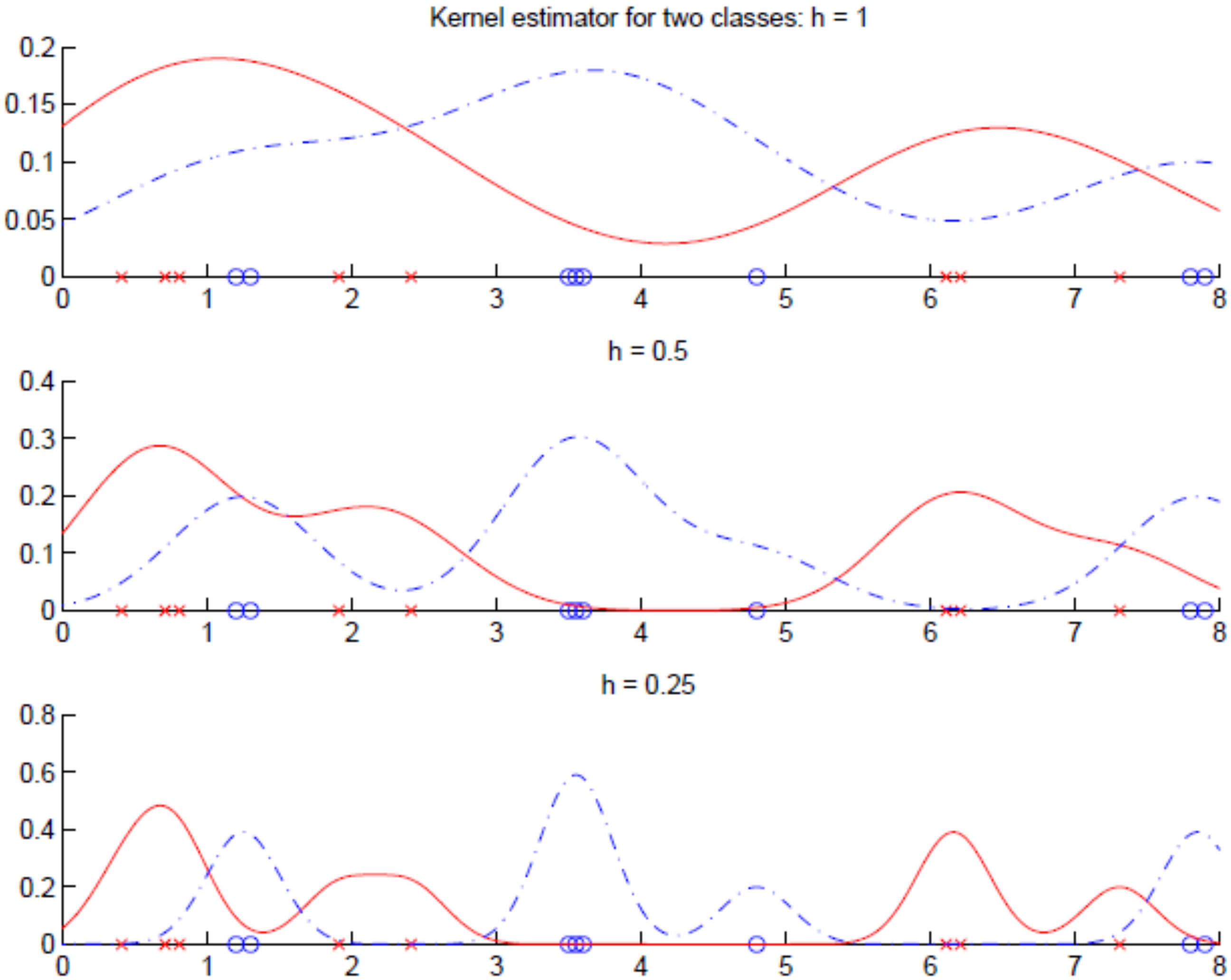

Running mean smoother: h=6
h=3
h=1
4
2
0
−2
0
1
2
3
4
5
6
7
8


Running line smooth: h=6
4
2
0
−2
0
1
2
3
4
5
6
7
8
h=3
4
2
0
−2
0
1
2
3
4
5
6
7
8
h=1
6
4
2
0
−2
0
1
2
3
4
5
6
7
8


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

How to Choose *k* or *h* ?

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


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