CHAPTER 8:

NONPARAMETRIC METHODS

Nonparametric Estimation

- Parametric (single global model), semiparametric ($\vee$ small number of local models)
- Nonparametric: Assume similar inputs have similar outputs.
- Keep the training data; "let the data speak for itself."
- Find the similar past instances from the training set using a suitable distance measure and interpolate from them to find the right output.
- Given x, find a small number of closest training instances and interpolate from these.
- Aka lazy/memory-based/case-based/instance-based learning

-- it stores the training examples in a lookup table and interpolate from them – O(N) space complexity and time complexity for search.

- Lazy learning – it don't compute a model with the training set, but postpone the computation of model until a test data is given.

Density Estimation

3

- Given the training set $\boldsymbol{X}=\{x^t\}_{t=1..N}$ drawn iid from $p(x)$.
- For the cumulative distribution function, F(x) and the pdf, $\hat{p}(x)$:

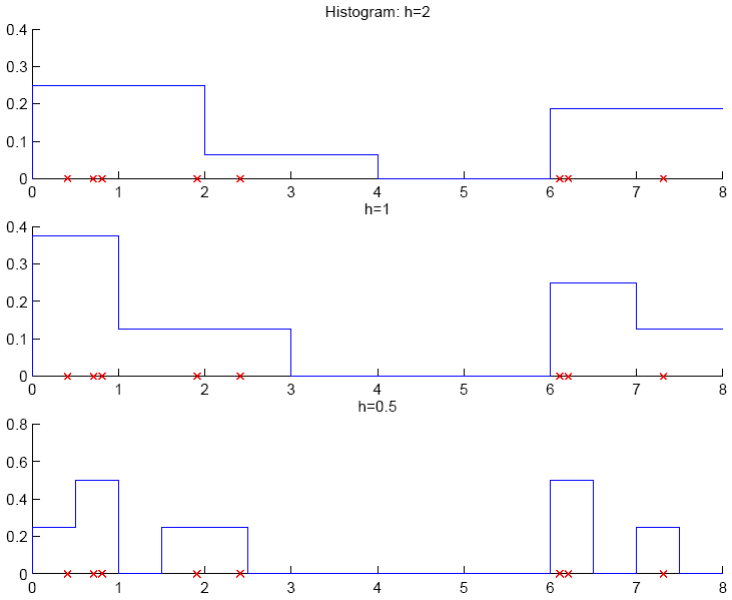
$$\hat{F}(x)=\frac{\#(x^t \le x)}{N} \quad \text{and} \quad \hat{p}(x)=\frac{1}{h}\left(\frac{\#(x< x^t \le x+h)}{N}\right),$$

h is the width of window(= bin, interval).

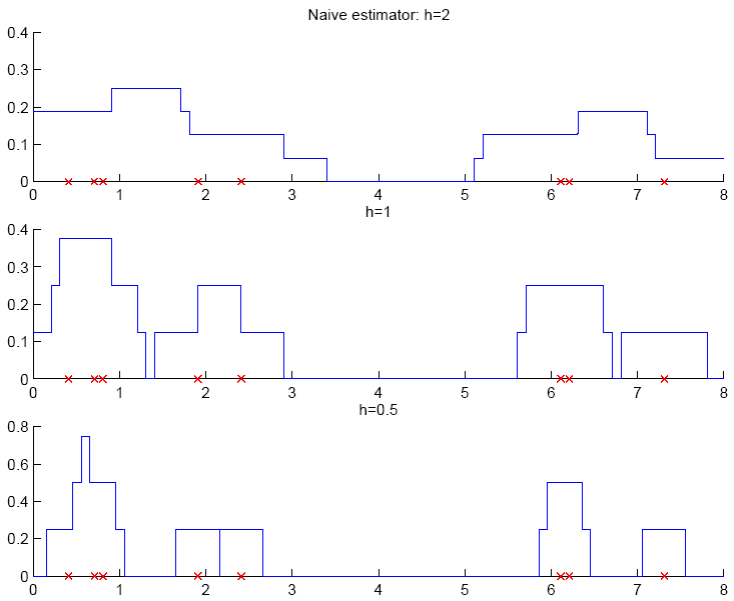
- Divide data into bins (i.e. intervals) of size *h*.
- Histogram: $\hat{p}(x)=\frac{1}{h}\left(\frac{\#(x^t \text{ in the same bin as } x)}{N}\right),$
- Naive estimator: and $\hat{p}(x)=\frac{1}{h}\left(\frac{\#(x-\frac{h}{2}< x^t \le x+\frac{h}{2})}{N}\right)$ $\left(=\frac{1}{2h}\left(\frac{\#(x-h< x^t \le x+h)}{N}\right)\right)$

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

with the weight function $w(u)=\begin{cases}1 & \text{if } |u|<\frac{1}{2}\\ 0 & \text{otherwise}\end{cases}$



4



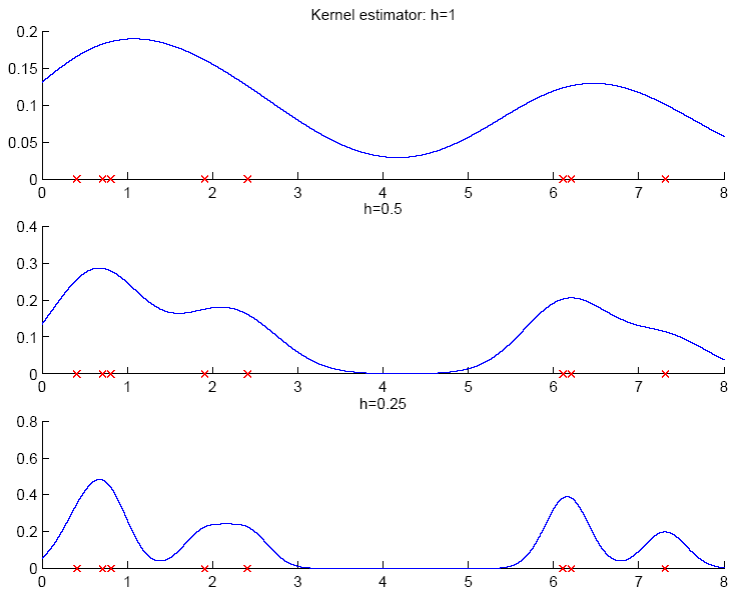
5

Kernel Estimator

6

- To get a smooth estimate, use a smooth weight function, called kernel function.
- Gaussian kernel: $K(\mathrm{u}) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{u^2}{2})$
- Kernel estimator (Parzen windows): $\hat{p}(\mathrm{x}) = \frac{1}{Nh} \sum_{t=1}^{N} K(\frac{x - x^t}{h})$

  K(·) decides the shape of the influences.
- All the x^t have an effect on the estimate at x, and this effect decreases smoothly as $|x - x^t|$ increases.
- *h* is small → each training instance has a large effect in a small region and no effect on distant points.
- *h* is larger → more overlap of the kernels and get a smoother estimate.
- *K*(u) is maximum for u=0, decreasing symmetrically as |u| increases.
- A fixed bin (i.e. window) size *h*.



7

k-Nearest Neighbor Estimator

8

- The nearest neighbor class of estimators *adapts* the *degree of smoothing* to the local density of data, which is controlled by the # of neighbors, $k << N$.
- Instead of fixing bin width *h* and counting the number of instances, fix the instances (neighbors) k and check bin width

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

  where $d_k(x)$, distance to k^{th} closest instance to *x*.
- In [*a, b*], for each x, $d_1(x) \le d_2(x) \le \ldots \le d_N(x)$:

  a distance from x to the points in the k^{th} nearest sample.
- Like a naïve estimator with $h = 2d_k(x)$.
- Where density is high, bins are small (i.e. small h, small interval).
- kNN is not a pdf since $\int^{\infty} \hat{p}(\mathrm{x}) \neq 1$.

k-Nearest Neighbor Estimator

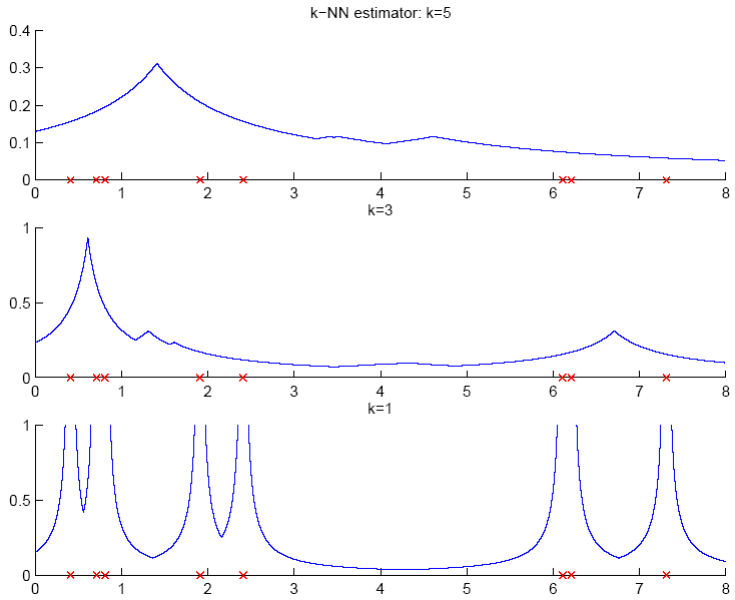
9

- To get a smoother estimate, use a kernel function whose effect decreases with increasing distance:

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

- A kernel estimator with adaptive smoothing parameter $h=d_k(\mathrm{x})$.

- The k-NN classifier assigns an instance to the class most heavily represented among its neighbors.
- More similar the instances, the more likely they belong to the same class.



10

Multivariate Data (*d*-dimensional data)

11

- The multivariate Kernel density estimator

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

with the requirement $\int_{R^d}^{\infty} K(\mathrm{x})dx = 1.$

- Multivariate Gaussian kernel
  - Spheric $K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \exp\left(-\frac{\|\mathrm{u}\|^2}{2}\right),$

    with Euclidean norm.
  - ellipsoid $K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi|\mathrm{S}|}}\right)^d \exp(-\frac{1}{2}\mathrm{u}^{\mathrm{T}}\mathrm{S}^{-1}\mathrm{u})$

    using Mahalanobis distance

    where S is the sample covariance matrix.

Nonparametric Classification

12

- Estimate the class-conditional density, $p(\boldsymbol{x}|\mathrm{C}_i)$.
- Kernel estimator of class cond. density:

$$\hat{p}(x|C_i) = \frac{1}{N_i h^d}\sum_{t=1}^{N} K\left(\frac{\mathbf{x}-\mathbf{x}^t}{h}\right) r_i^t \quad \text{where } r_i^t = \begin{cases} 1 & \text{if } x^t \in C_i \\ 0 & \text{o.w.} \end{cases}$$

N_i = # of labeled instances $\in C_i$, $N_i = \sum_t r_i^t$

- Then, discriminant is:

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

where $\hat{P}(C_i) = \frac{N_i}{N}$.

x is assigned to $C_i = \underset{i}{argmax}\, g_i(x)$.

The weight of vote is given by the kernel function *K*, giving more weight to closer instances.

Nonparametric Classification

13

- For the special case of kNN estimator,

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

$k_i = \#$ of neighbors out of the k nearest $\in C_i$,

$V^k(x)$ is the volume of *d*-dim. hypersphere centered at x

with radius $r = \|x - x_{(k)}\|$ where is the k[th] nearest observation to x:

$V^k = r^d c_d$ where as the volume of the unit sphere in d-dim.

- Then, $\hat{P}(C_i|x) = \frac{\hat{p}(x|C_i)\hat{P}(C_i)}{\hat{p}(x)} = \frac{k_i}{k}$
- *x* is assigned to $C_i = \underset{i}{argmax}\,|C_i|$

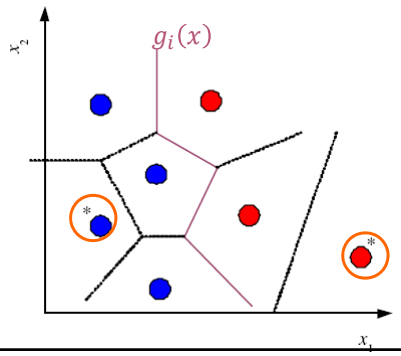
with the most examples among the *k* neighbors of the input.

If all equal vote, the class with the maximum # of voters is chosen. .

Nonparametric Classification

14

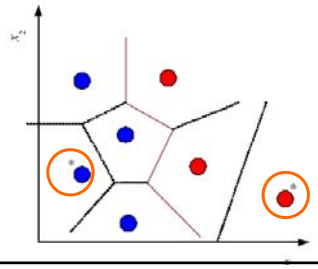
- A special case of kNN with k=1 :
  - the nearest neighbor classifier with k=1.
  - The input is assigned to the class of the nearest pattern.
  - The input space is divided in the form of Voronoi tessellation.



Condensed Nearest Neighbor

15

- 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): an error doesn't increase with Z in place of X.
- Condensed nearest neighbor with 1-NN:
- 1-NN approximates the discriminant in a piecewise linear manner and only the instances that define the discriminant need to be kept: consistent subset. → Find the minimal consistent subset, Z.



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

Condensed Nearest Neighbor

16

- A greedy algorithm to find the minimal consistent subset Z.
- 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($\boldsymbol{x}$)≠class($\boldsymbol{x}'$) add $\boldsymbol{x}$ to $\mathcal{Z}$
Until $\mathcal{Z}$ does not change

- A local search depending on the order of the training instances → different subsets → different accuracy on the validation.
- It doesn't guarantee finding the minimal consistent subset: NP-complete problem.

Condensed Nearest Neighbor

17

- Condensed nearest neighbor is a greedy algorithm to minimize training error and complexity, measured by the size of the stored subset.
- An error function:

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

where $E(X|Z)$ is the error on X storing Z,

$|Z|$ is the cardinality of *Z,*

λ is a trade-off b/t the error and complexity.

- $\Leftrightarrow$ *Cost(Z) = EmpLoss(Z) + λ Complexity(Z)*

Distance-based Classification

18

- In the nonparametric case, define locally adaptive distance function with kNN for each neighborhood – a different distance measure, i.e. locally adaptive distance function for each neighborhood.
- 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. $D(\mathrm{x}, \mathrm{x}^t|M) = (\mathrm{x} - \mathrm{x}^t)^T M(\mathrm{x} - \mathrm{x}^t)$ -- Mahalanobis distance

  where the parameter is the positive definite matrix M.

  Note that a notation $D(x_1, x_2|\theta)$ is a distance between x_1 and x_2 which is defined by a parameter θ, NOT a conditional distribution on θ.
- Then, use $D(\mathrm{x}, \mathrm{x}^t|M)$ with kNN.

Learning a Distance Function

19

- $D(\mathrm{x}, \mathrm{x}^t|M) = (\mathrm{x} - \mathrm{x}^t)^T M(\mathrm{x} - \mathrm{x}^t)$ -- Mahalanobis distance
  where the parameter is the positive definite matrix M.
- Similarity-based representation using similarity scores.
- Large-margin nearest neighbor (chapter 13):
  - M is estimated so that distance to a neighbor with the same label < the distance to a neighbor with a different label.
- To avoid the overfitting in the high dimensional input:
  - Approach 1: add sparsity constraints on M.
  - Approach 2: use a low-rank approximation where M is factored as $\mathrm{L}^T\mathrm{L}$.
- $\mathbf{M}=\mathbf{L}^T\mathbf{L}$ is $d\times d$ and $\mathbf{L}$ is $k\times d$ with $k < d$.

$$D(\mathrm{x}, \mathrm{x}^t|M) = (\mathrm{x} - \mathrm{x}^t)^T M(\mathrm{x} - \mathrm{x}^t) = (\mathrm{x} - \mathrm{x}^t)^T L^T L(\mathrm{x} - \mathrm{x}^t)$$
$$= (L(\mathrm{x} - \mathrm{x}^t))^T (L(\mathrm{x} - \mathrm{x}^t)) = (Lx - Lx^t)^T (Lx - Lx^t)$$
$$= (z - z^t)^T (z - z^t) = \|z - z^t\|^2$$

where $\mathrm{z} = \mathrm{Lx}$ is the k-dimensional projection of x.

Learning a Distance Function

20

- $\mathbf{M}=\mathbf{L}^T\mathbf{L}$ is $d\times d$ and $\mathbf{L}$ is $k\times d$ with $k < d$.

$$D(\mathrm{x}, \mathrm{x}^t|M) = (\mathrm{x} - \mathrm{x}^t)^T M(\mathrm{x} - \mathrm{x}^t) = (\mathrm{x} - \mathrm{x}^t)^T L^T L(\mathrm{x} - \mathrm{x}^t)$$
$$= (z - z^t)^T (z - z^t) = \|z - z^t\|^2$$

where $\mathrm{z} = \mathrm{Lx}$ is the k-dimensional projection of x.

- Let's learn L instead of M.
- Mahalanobis distance in d-dim. X-space → squared Euclidean distance in k-dim. space: 3-way relationship between distances, dimensionality reduction, and feature extraction.
- *Euclidean distance* in the k-dim space where k is the fewest dimension of the extracted feature.
- With discrete data, *Hamming distance* counts the # of nonmatching attributes: $HD(x, x^t) = \sum_{j=1}^{d} 1(x_j \neq x_j^t)$ where $1(a) = \begin{cases} 1 & \text{if } a \text{ is true} \\ 0 & \text{o.w.} \end{cases}$
- Application dependent similarity/distance measure: e.g.)
  - In vision, similarity scores for matching image part; In bioinformatics, sequence alignment scores; In natural language processing, document similarity measure.

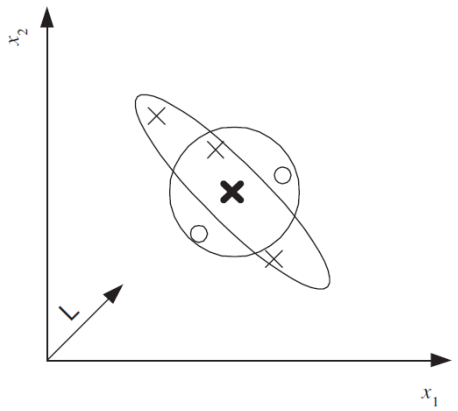
Learning a Distance Function

21

- If we have a similarity score function b/t two instances S(x, x^t), we can define a *similarity-based representation x'* of instance x as *N*-dim. vector of scores with all x^t, with s(x, x^t), t=1, .. N:

  $x' = [s(x, x^1), s(x, x^2), .. s(x, x^N),]^T$

  -- x' can be used as a vector to be handled by any learner.



K=3
Euclidean distance (circle) is not suitable,
Mahalanobis distance using an **M** (ellipse) is suitable.
After the data is projected along **L**, correct classification in that reduced 1-dim. space; Euclidean distance can be used.

22

Outlier Detection

23

- Find outlier/novelty points
- ~~A Supervised 2-class classification problem~~
- Not a two-class problem because outliers are very few, of many types, and seldom labeled.
- Instead, one-class classification problem:
  - Once we model the typical instances, then
  - Find instances that don't fit the model.
- Training data is unlabeled , containing outliers mixed with typical instances.
- Instances with the low probability under the estimated density.
  - In parametric case: with Gaussian model, an instance with high mahalanobis distance to the mean may be an outlier.
  - In semiparametric case: a mixture of Gaussians, instance that is far from its nearest cluster center or form a single cluster by itself may be an outlier.
  - In nonparametric case: Find instances far away from other instances.

Outlier Detection

24

- In nonparametric density estimation, the estimated probability is high where there are many training instances nearby and the probability decreases as the neighborhood becomes more sparse.
- Local outlier factor compares the denseness of the neighborhood of an instance with the average denseness of the neighborhood of its neighbors.
- $d_k(x)$: distance b/t instance x and its k^{th} nearest neighbor.
- $N(x)$: the set of training instances that are in the neighborhood of x., i.e. its kNN.
- For $s \in N(x)$ with its $d_k(s)$, Compare $d_k(x)$ with the average of $d_k(s)$.

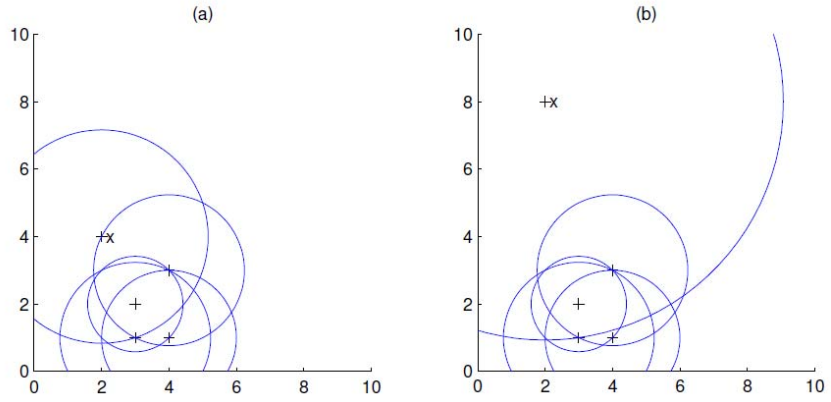
$$\mathrm{LOF}(\boldsymbol{x}) = \frac{d_k(\boldsymbol{x})}{\sum_{\boldsymbol{s}\in\mathcal{N}(\boldsymbol{x})} d_k(\boldsymbol{s})/|\mathcal{N}(\boldsymbol{x})|}$$

- LOF(x) → 1, x is not an outlier; LOF(x)↑ → P(x =outlier) ↑

Local Outlier Factor

25

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



Nonparametric Regression

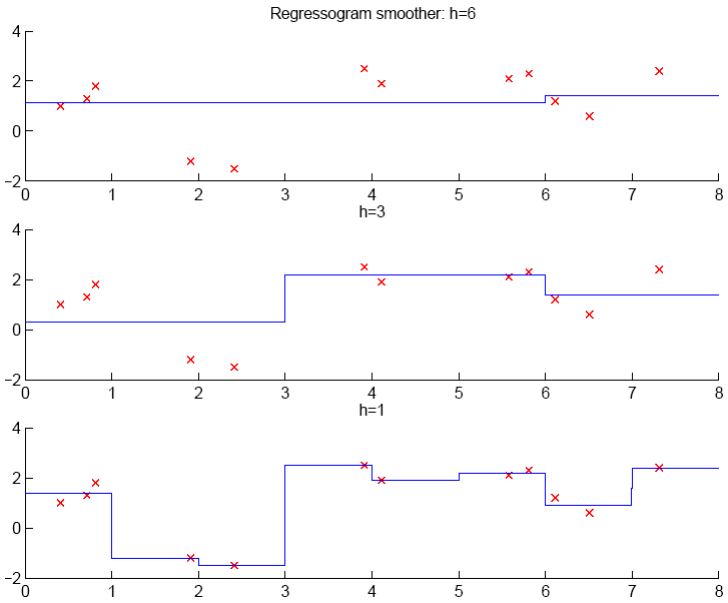
26

- Aka smoothing models
- Regressogram

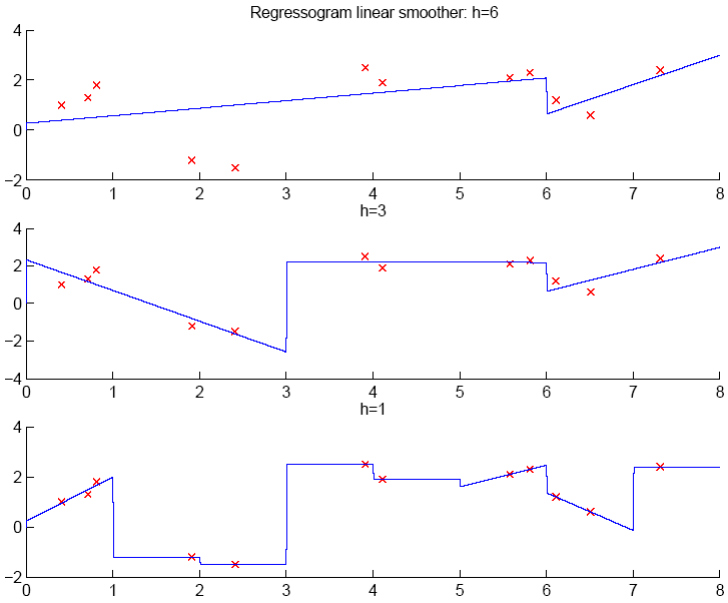
$$\hat{g}(x) = \frac{\sum_{t=1}^{N} b(x, x^t) r^t}{\sum_{t=1}^{N} b(x, x^t)}$$

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

27


Regressogram linear smoother: h=6
h=3
h=1

28

Running Mean/Kernel Smoother

29

- 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

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

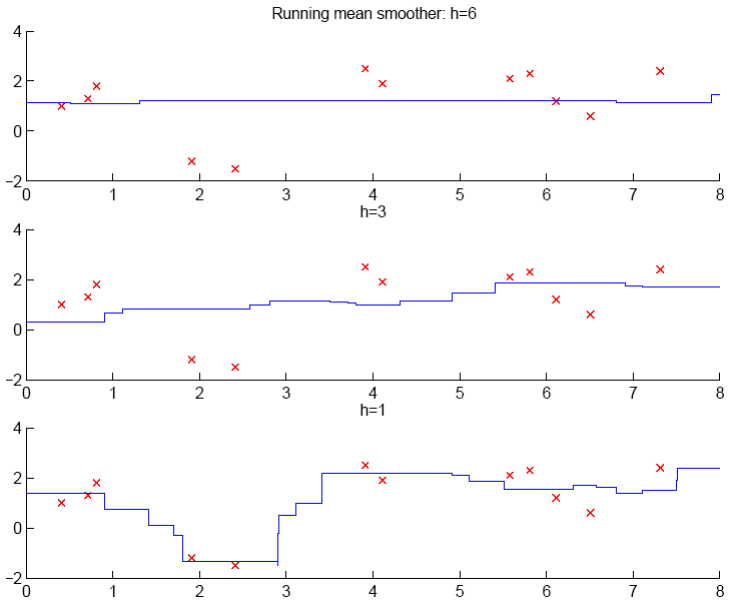
- Running line smoother

- Kernel smoother

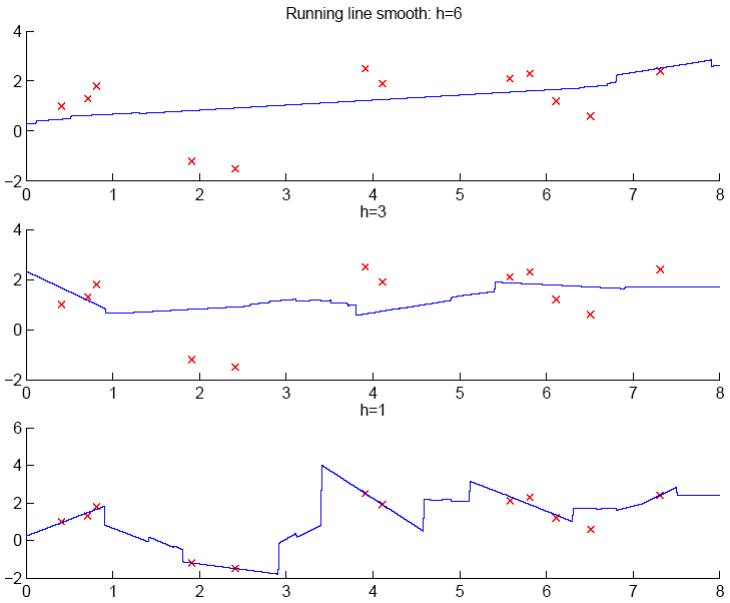
$$\hat{g}(x)=\frac{\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right) r^t}{\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right)}$$

where $K(\,)$ is Gaussian

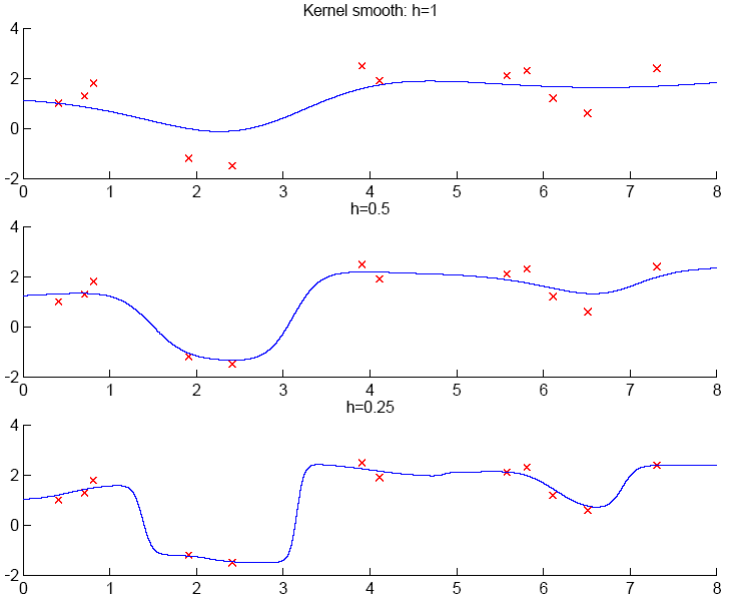
- Additive models (Hastie and Tibshirani, 1990)



30



31

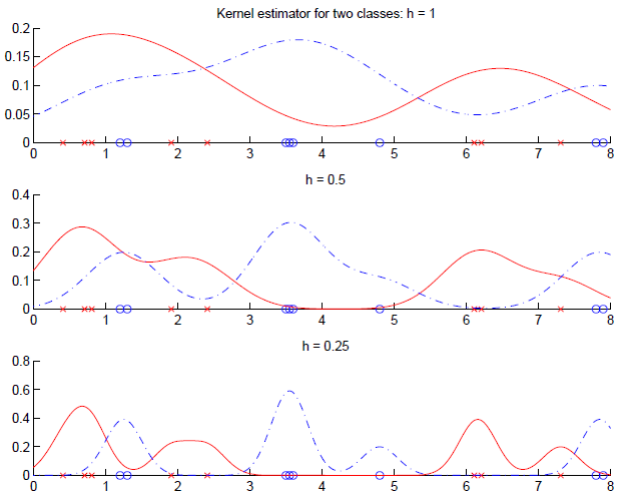


32

How to Choose *k* or *h* ?

33

- 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*.



34