Instance Based Learning

AKA: nearest neighbor methods, non-parametric, lazy, memory-based, or case-based learning

Parametric methods: summarize data sets with fixed # of parameters

(like Perceptron, Logistic regression)

Non-Parametric methods:

- Do not fit a parameterized model
- Includes Nearest Neighbor and some density estimation methods
- Often "lazy" (no gradient descent, optimization, or search) (under "\lazy" in Weka)

Nearest Neighbor Algorithm

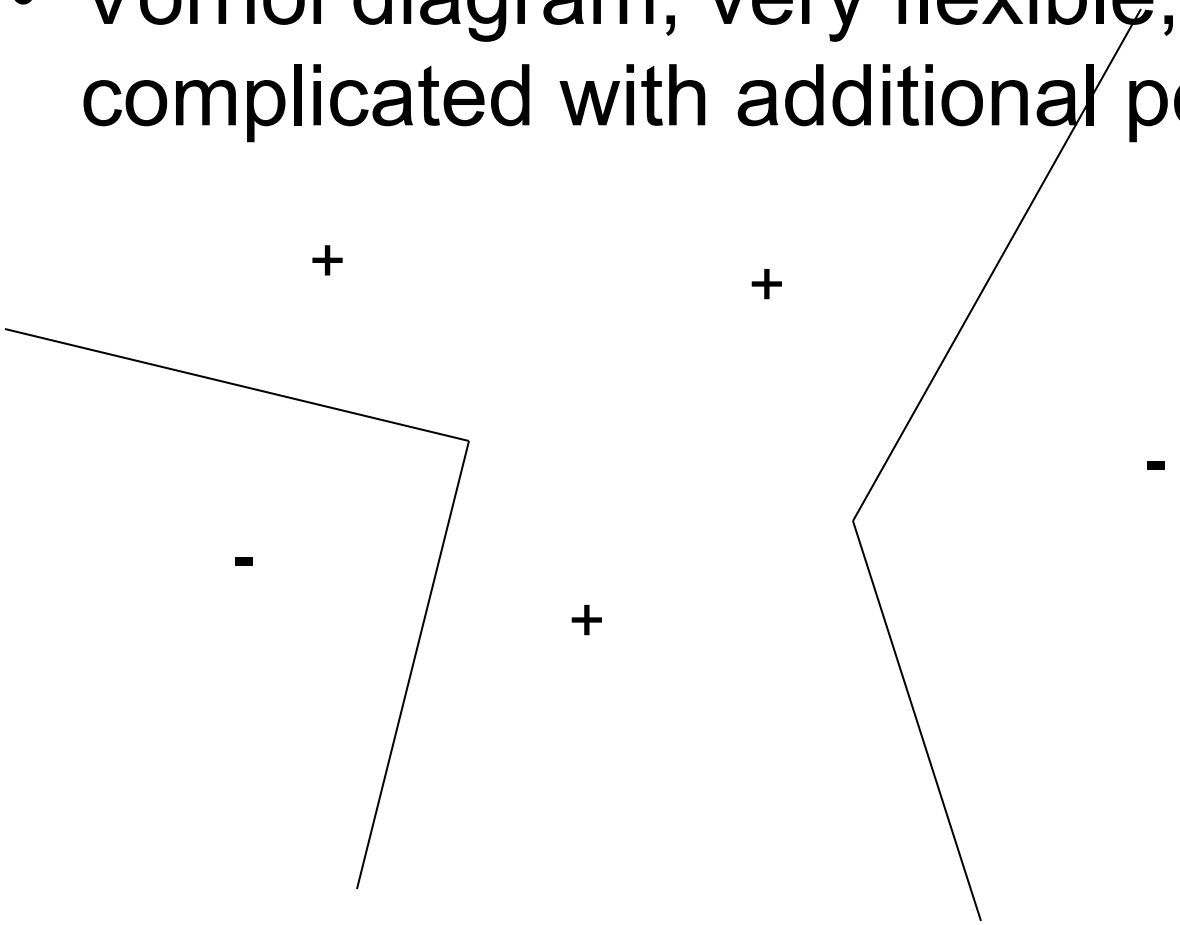
- Instances ($\boldsymbol{x}$'s) are vector of real
- Store the n training examples

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$$

- To predict on new $\boldsymbol{x}$, find stored $\boldsymbol{x}_i$ closest to $\boldsymbol{x}$ and predict with y_i
- Comments:
  - Not just simple table lookup
  - Can avoid $\sqrt{}$ by minimizing squared distance

NN Decision Boundaries

- Vornoi diagram, very flexible, gets more complicated with additional points



Nearest Neighbor Applications

- Astronomy (classifying objects)
- Medicine - diagnosis
- Object detection
- Character recognition (shape matching, using complicated distance function)
- Many others (basic theory from 1950's and '60's)

Distance metric important

- Consider expensive houses with features:
  - Number of bedrooms (1 to 5+)
  - Lot size in acres (1/6 to 1/2 plus tail)
  - House square feet (1200 to 3000+)

  Difference in square feet dominates
- Irrelevant attributes (e.g. “how far away was owner born?”) problematic
- Highly correlated attributes also bad why?

Irrelevant attribute example

- Let $x_1 \in [0,1]$ determine class:

  $y = 1$ iff $x_1 > 0.3$

- Consider predicting on (0,0) given data

  $(0.1, x_2)$ labeled 0

  $(0.5, x'_2)$ labeled 1

  where x_2, x'_2 random draws from [0,1]

  What is probability of mistake?

Irrelevant attribute example

- Let $x_1 \in [0,1]$ determine class:

  $y = 1$ iff $x_1 > 0.3$

- Consider predicting on (0,0) given data

  $(0.1, x_2)$ labeled 0

  $(0.5, x'_2)$ labeled 1

  where x_2, x'_2 random draws from [0,1]

- Chance of error ~ 27.5%!

Some tricks

- Normalize attributes for example mean 0 variance 1
- Use w_j on j^{th} component:
  - Dist($\boldsymbol{x}$, $\boldsymbol{x}'$) = $\sum_j w_j (x_j - x'_j)^2$
  - $w_j = I(x_j, t)$ ("mutual information")
- Mahalanobis Distance (covariance Σ, like Gaussians)

$$\text{Dist}(\boldsymbol{x},\boldsymbol{x}') = (\boldsymbol{x}-\boldsymbol{x}')^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{x}')$$

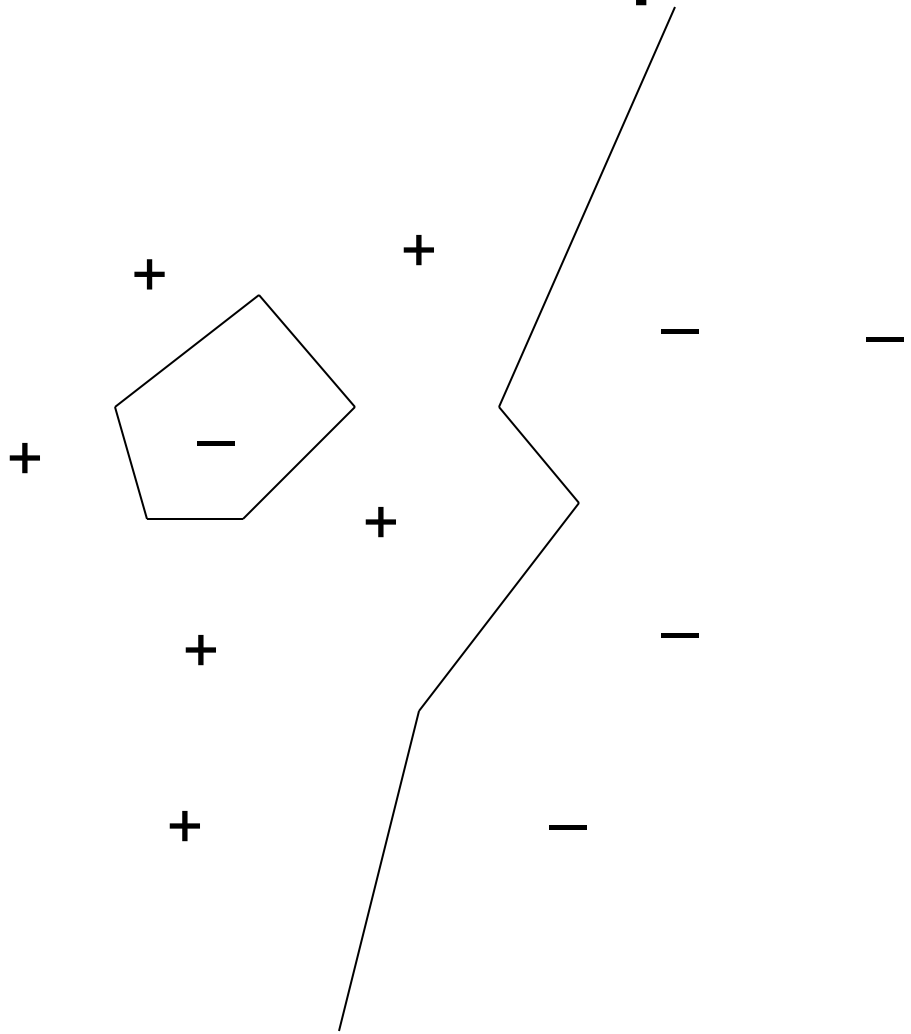
Curse of Dimensionality

- As number of attributes (d) goes up so does "volume"
- Consider 1000 training points in $[0,1]^d$ where does each point predict?
  - When d=1, interval per point ~0.001
  - When d=2, area per point ~0.001, length of side about 0.032
  - When d=10, volume per point ~0.001, length of side ~ 0.5
- Need exponentially many points (in d) to get good coverage

K-d trees

- Greatly speed up finding nearest neighbor
- Like binary search tree, but organized around dimensions
- Each node tests single dimension against threshold (median)
- Can use highest variance dimension or cycle through dimensions
- Growing a good K-d tree can be expensive

Noise can cause problems



Noise example

- Assume that “true” labels always 1, but noise randomly corrupts labels 10% of the time (making them 0)
- Bayes optimal: predict 1, test error is 10%
- Nearest Neighbor: use closest training point,
  - 90% of the time predict 1, 10% of these predictions wrong
  - 10% of the time predict 0, 90% of these predictions wrong
- Overall NN wrong 18% of the time – Can we do better?

K-nearest neighbor

- Algorithm: Find the closest k points and predict with their majority vote
- K-NN *is* Bayes optimal in limit as k and training set size go to ∞ (known since 1960's)

Edited NN

- Key Idea: Reduce memory and computation by only storing “important” points
- Heuristic:
  - Discard those points correctly predicted by others (or take incorrectly predicted points)
  - Remaining points concentrated on the decision boundary
- Finding a smallest subset of points correctly labeling others is NP-complete.

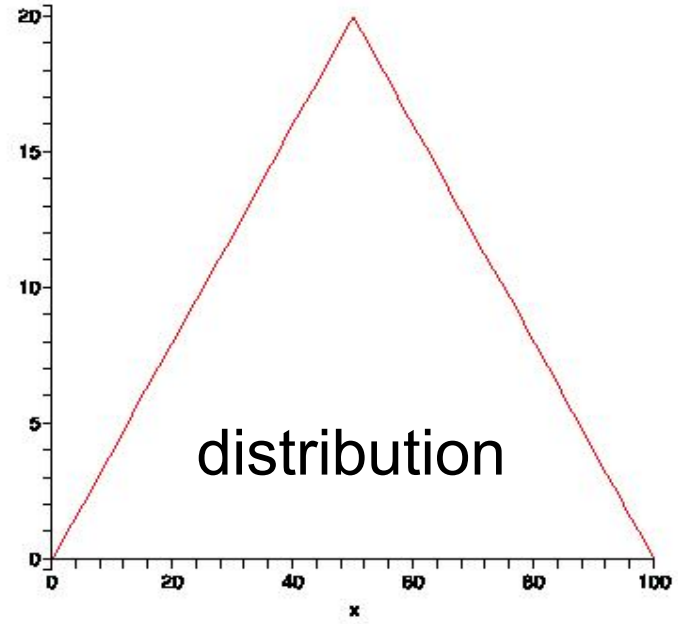
Instance Based Density Estimation

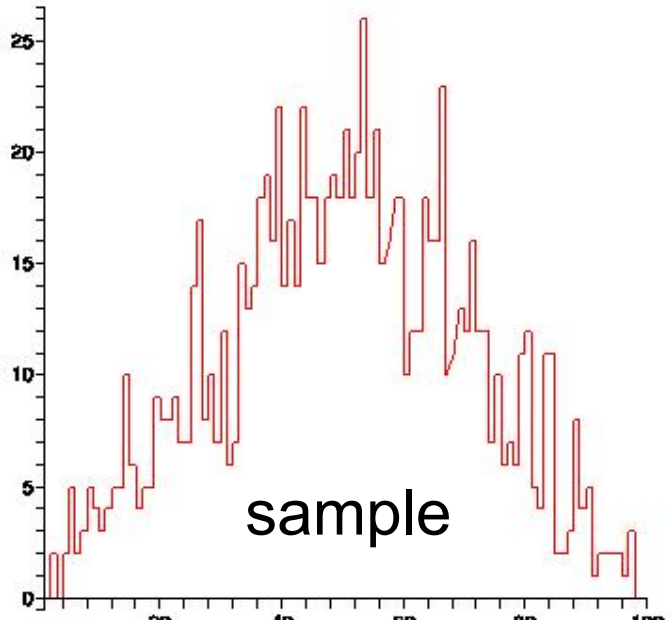
Histogram method:

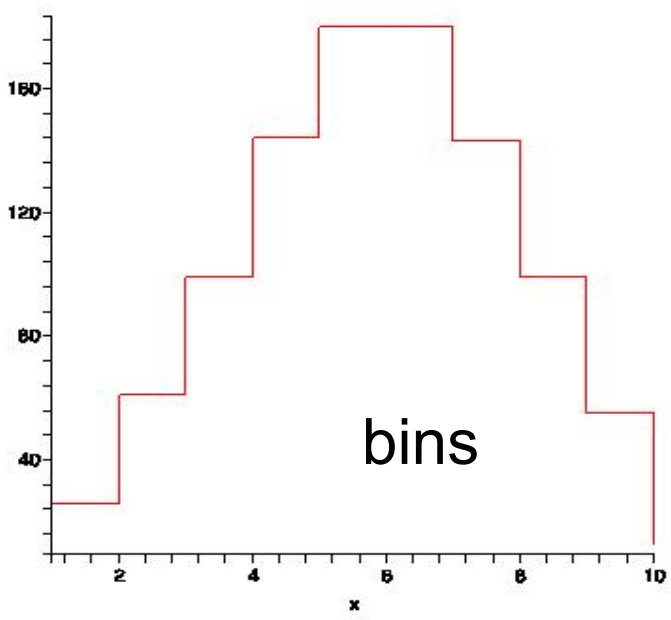
- Break instance space X into bins
- Use sample falling into bin to estimate probabilities

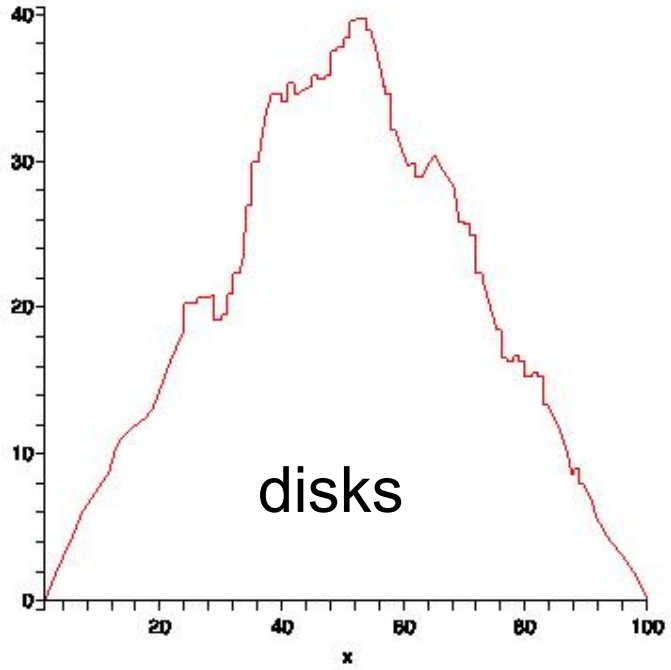
- Histogram method is parametric, not instance based
- Has edge effects

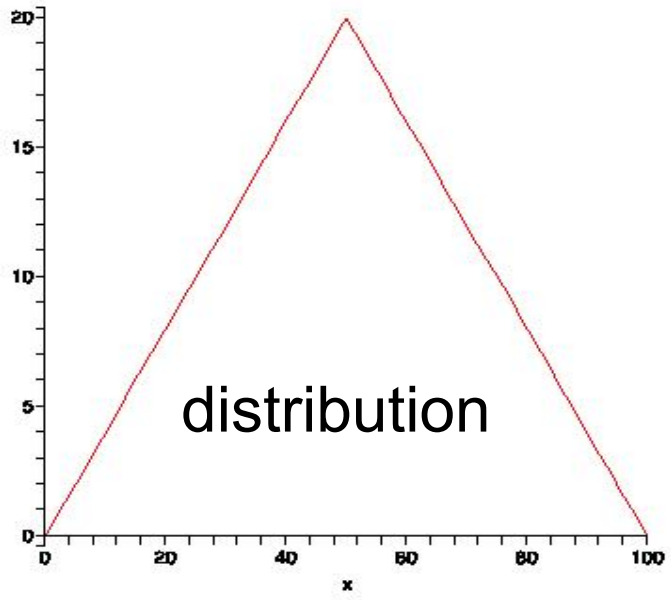
- Smoother method: Add slice of probability to area centered at example rather than to predetermined bin
- In general, have a *Kernel function* that tells how probability added (see Duda and Hart)
  - Gaussians common
  - Also called Parzon Windows
  - Often a "width" parameter controls smoothing (like σ in Gaussians)

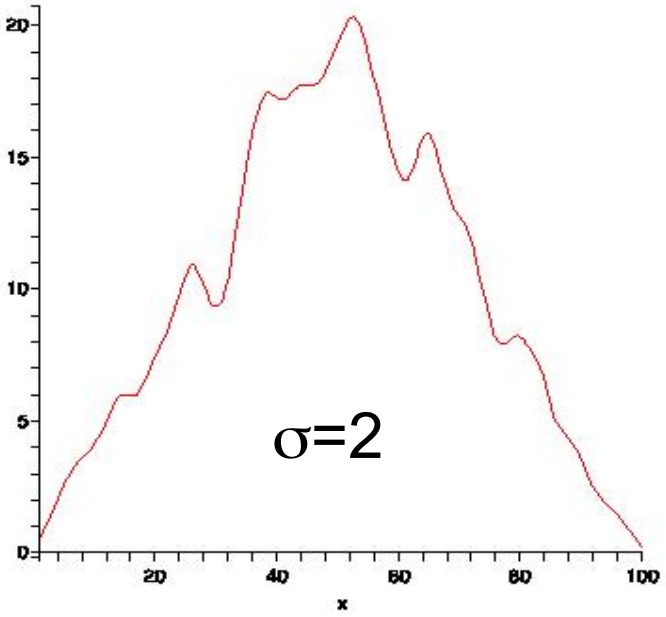

distribution
x


sample

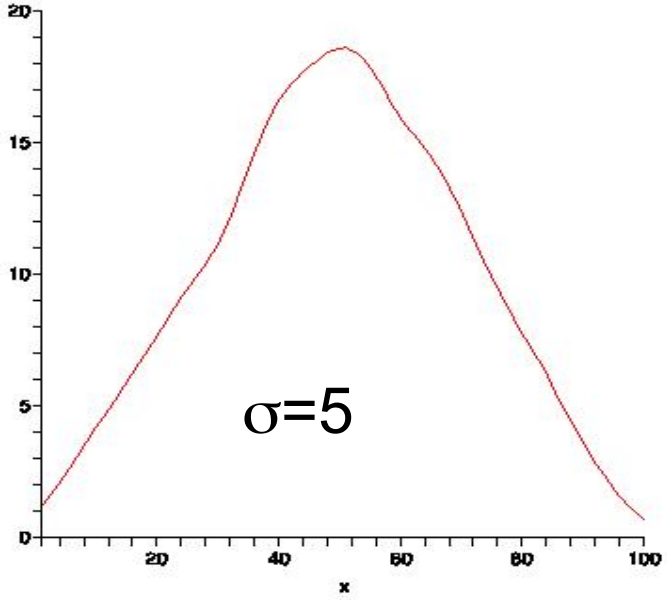

bins
x

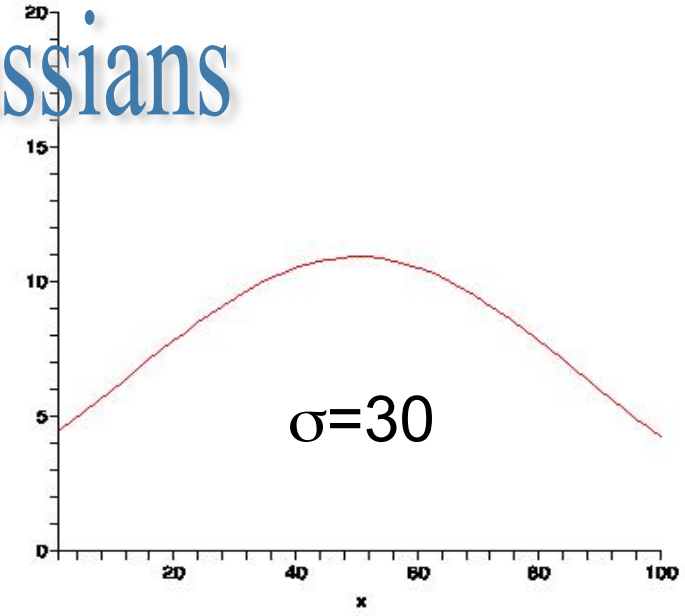

disks
x





Gaussians





Final points

Can use smoothed nearest neighbor also (whole sample votes on predictions with weights depending on distance to new point), but may be computationally expensive

Can fix k and use distance to kth nearest to estimate density

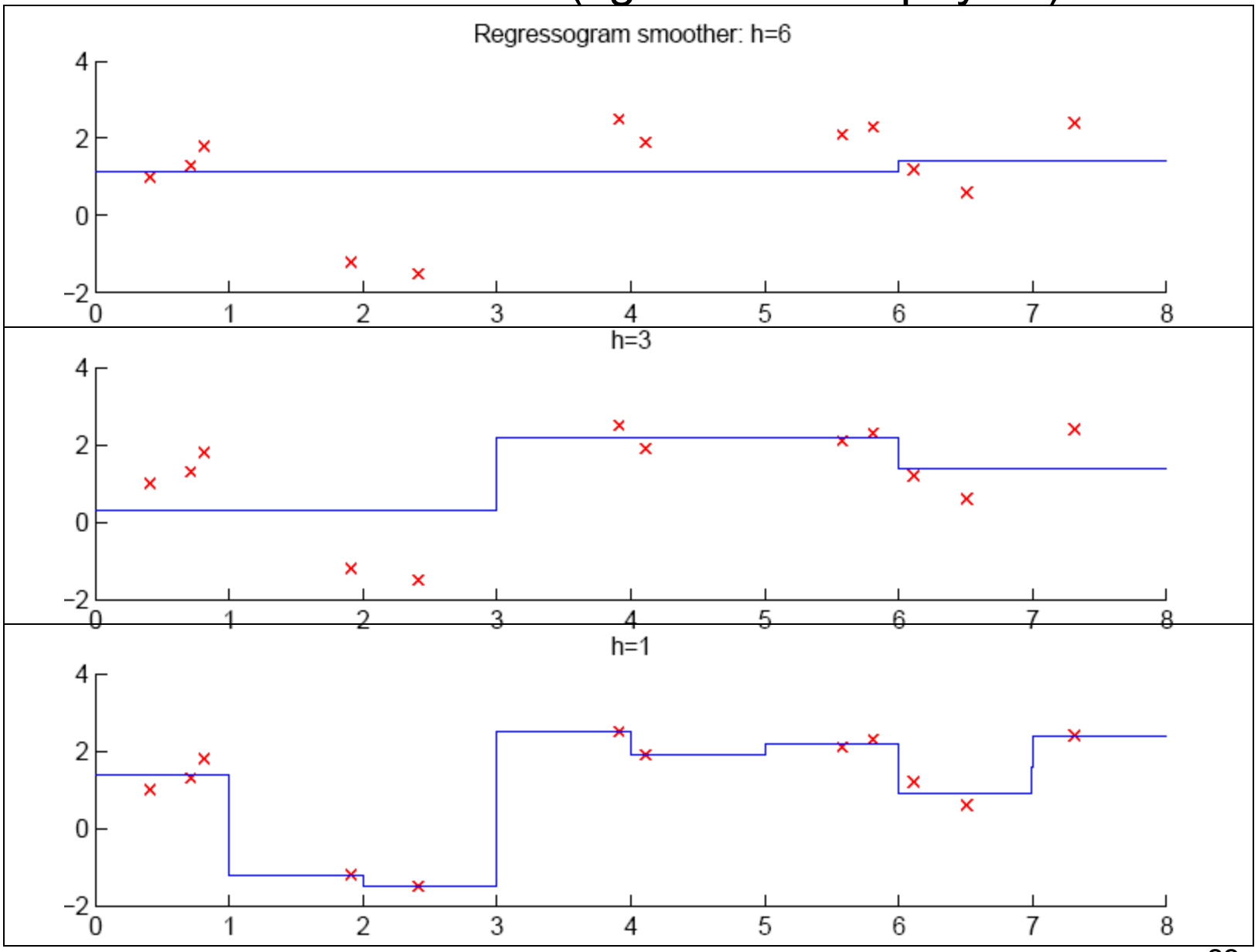
Might use cross validation to estimate smoothing parameter

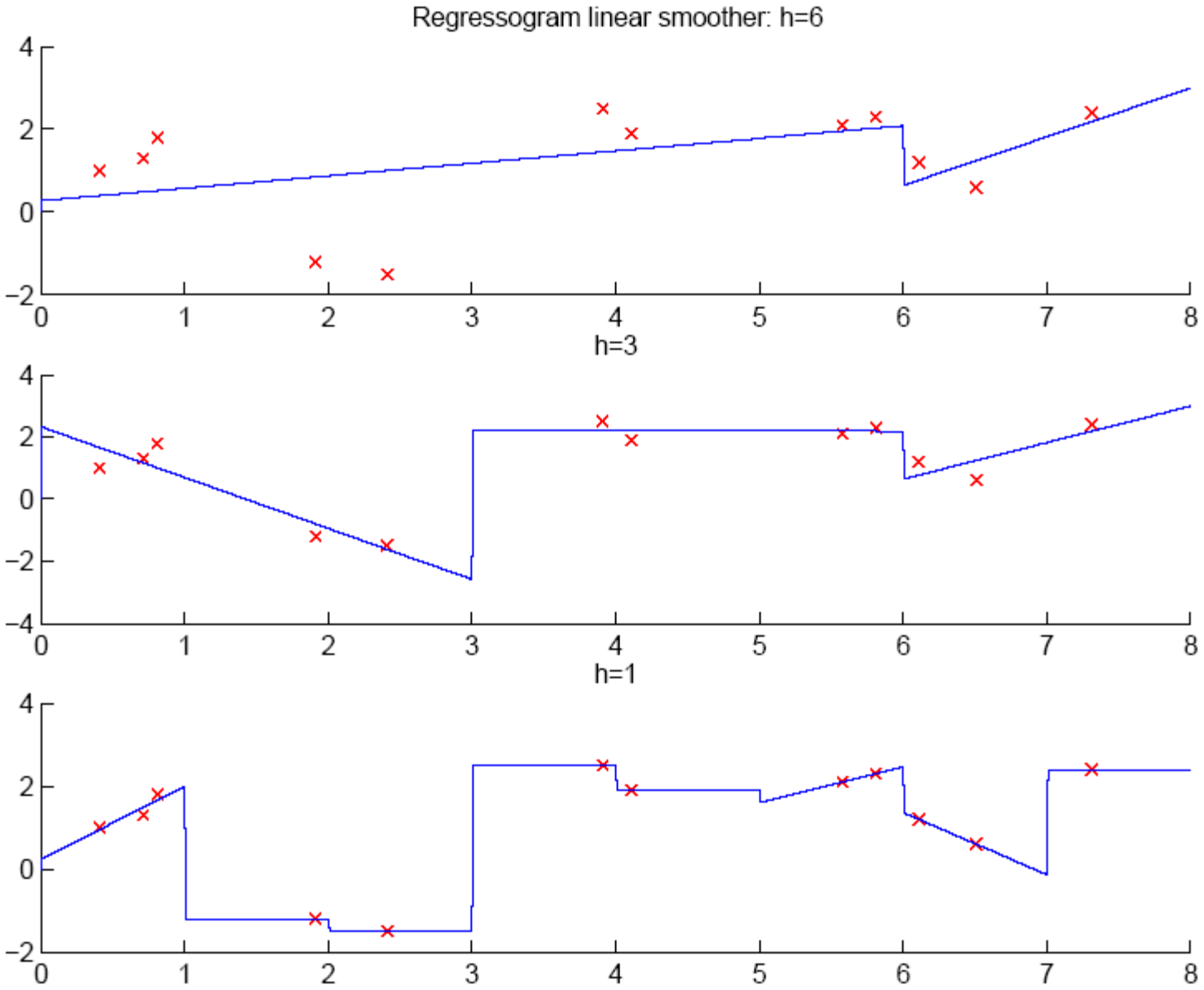
Can use density estimation for P(feature | class) and then predict class labels with Bayes' rule

Nonparametric Regression

- Sometimes called “smoothing models”
- Regressogram:
  - Partition domain into “bins”
  - Predict with bin average
  - Depends on bin size
  - Could do linear fitting within bins

h is the bin width (figures from Alpaydin)




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

Running Mean/Kernel Smoother

- Running mean smoother

Average of "close" pts:
Those within h
of point
to be predicted

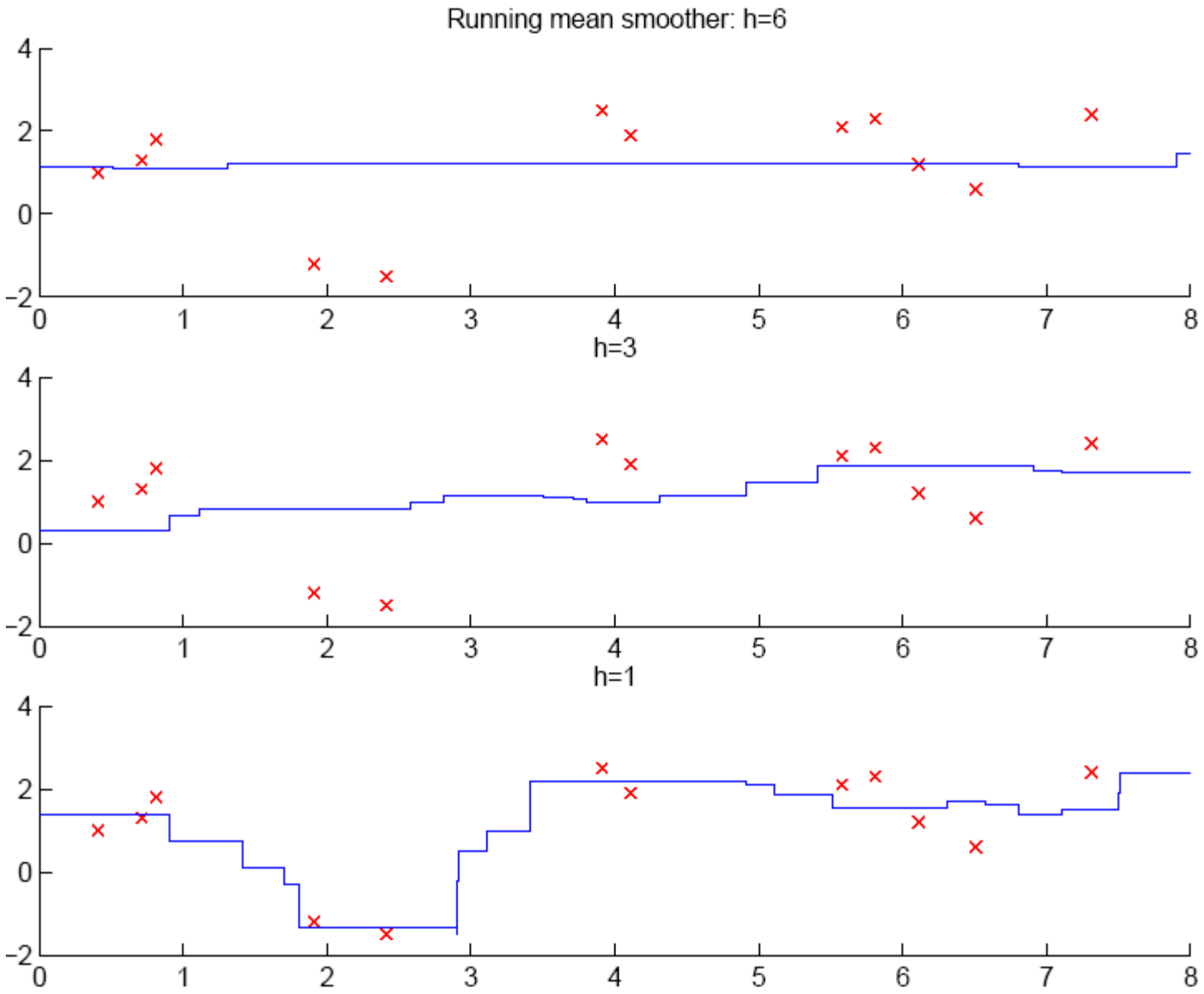
- Running line smoother (locally linear)

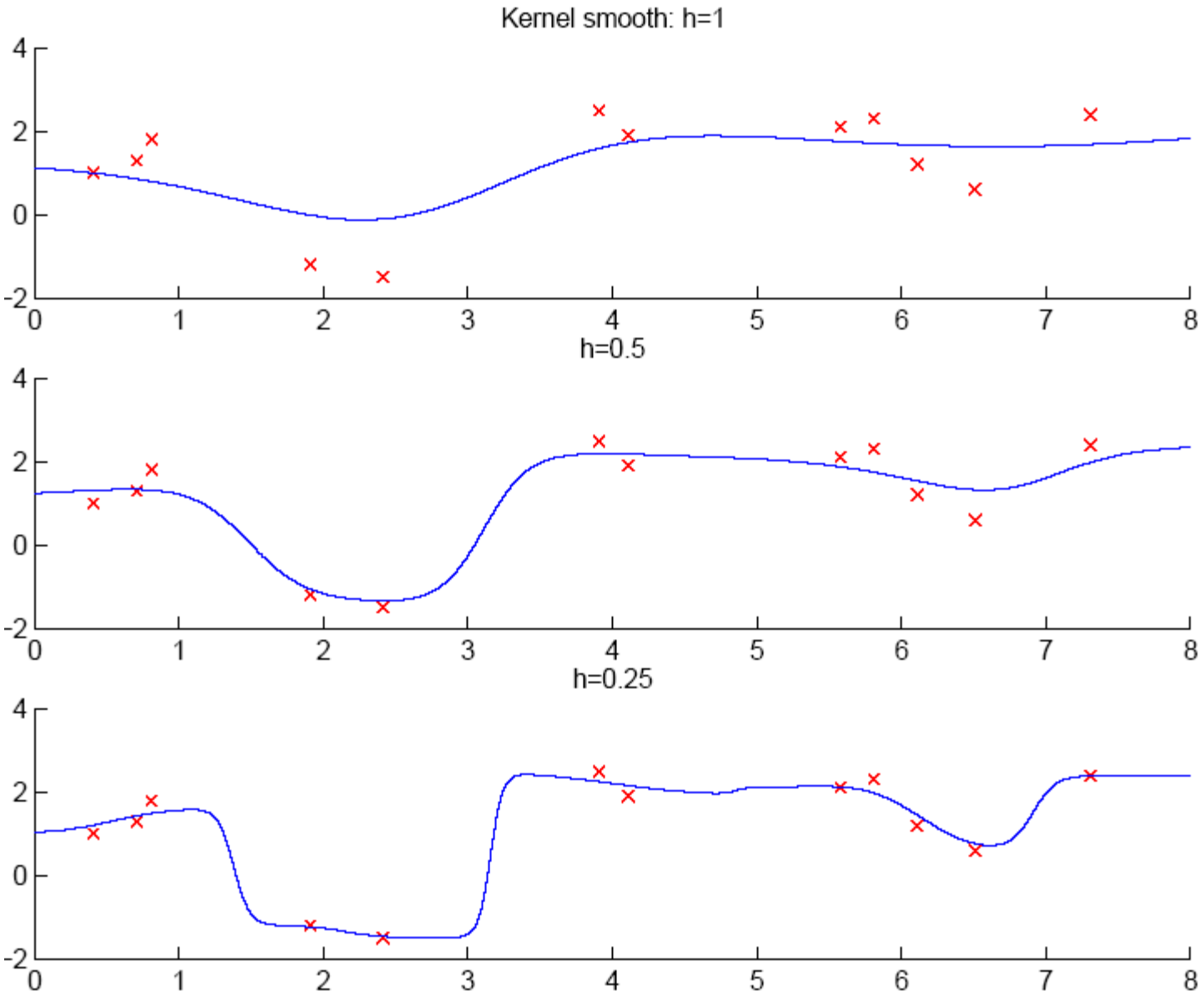
- Kernel smoother votes everyone, on x predict:

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

$K(\ )$ usually Gaussian

1-dim examples




Kernel smooth: h=1
h=0.5
h=0.25

How to Choose k (for kNN) or h?

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