Non-Parametric Methods

Dr. Jianlin Cheng

Computer Science Department
University of Missouri, Columbia
Fall, 2015

Slides Adapted from Book and CMU, Stanford Machine Learning Courses

Parametric methods

- Assume some functional form (Gaussian, Bernoulli, Multinomial, logistic, Linear) for
  - $P(X_i | Y)$ and $P(Y)$ as in Naïve Bayes
  - $P(Y|X)$ as in Logistic regression
- Estimate parameters (μ,σ^2,θ,w,β) using MLE/MAP and plug in
- **Pro** – need few data points to learn parameters
- **Con** – Strong distributional assumptions, not satisfied in practice

Non-Parametric methods

- Typically don’t make any distributional assumptions
- As we have more data, we should be able to learn more complex models
- Let number of parameters scale with number of training data

- Today, we will see some nonparametric methods for
  - Density estimation
  - Classification
  - Regression

Histogram density estimate

Partition the feature space into distinct bins with widths Δ_i and count the number of observations, n_i, in each bin.

$$\widehat{p}(x) = \frac{n_i}{n\Delta_i}\mathbf{1}_{x\in \mathrm{Bin}_i}$$

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- Δ acts as a smoothing parameter.

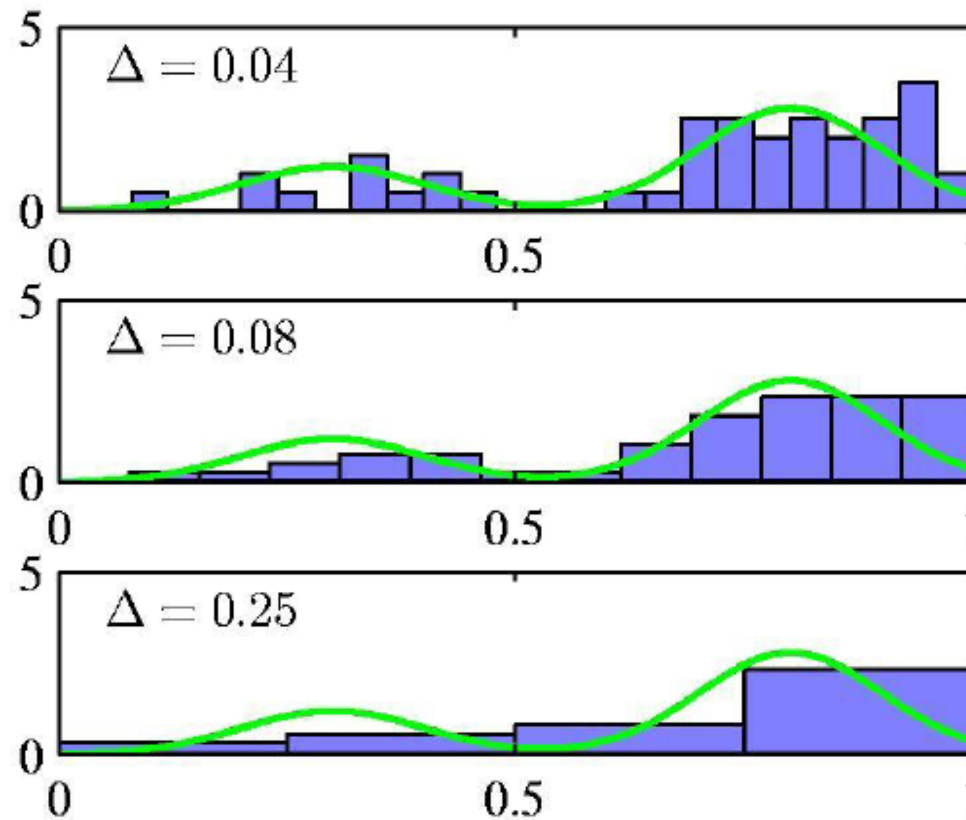


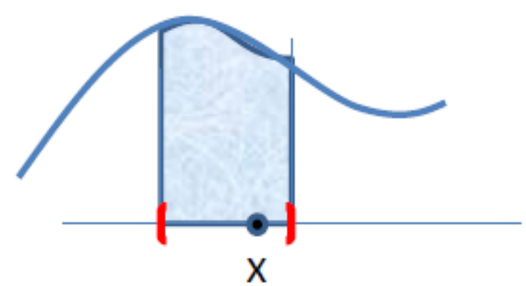
Image src: Bishop book

Effect of histogram bin width

$$\widehat{p}(x) = \frac{n_i}{n\Delta}\mathbf{1}_{x\in \text{Bin}_i}$$

bins = 1/Δ

$$\widehat{p}(x) = \frac{1}{\Delta}\frac{\sum_{j=1}^{n}\mathbf{1}_{X_j\in \text{Bin}_x}}{n}$$



Bias of histogram density estimate:

$$\mathbb{E}[\widehat{p}(x)] = \frac{1}{\Delta}P(X \in \text{Bin}_x) = \frac{1}{\Delta}\int_{z\in \text{Bin}_x} p(z)dz \approx \frac{p(x)\Delta}{\Delta} = p(x)$$

Assuming density it roughly constant in each bin (holds true if Δ is small)

Bias – Variance tradeoff

- Choice of #bins

bins = 1/Δ

$$\mathbb{E}[\widehat{p}(x)] \approx p(x) \text{ if } \Delta \text{ is small}$$

(p(x) approx constant per bin)

$$\mathbb{E}[\widehat{p}(x)] \approx \widehat{p}(x) \text{ if } \Delta \text{ is large}$$

(more data per bin, stable estimate)

- **Bias** – how close is the mean of estimate to the truth
- **Variance** – how much does the estimate vary around mean

Small Δ, large #bins ⟺ "**Small bias**, **Large variance**"

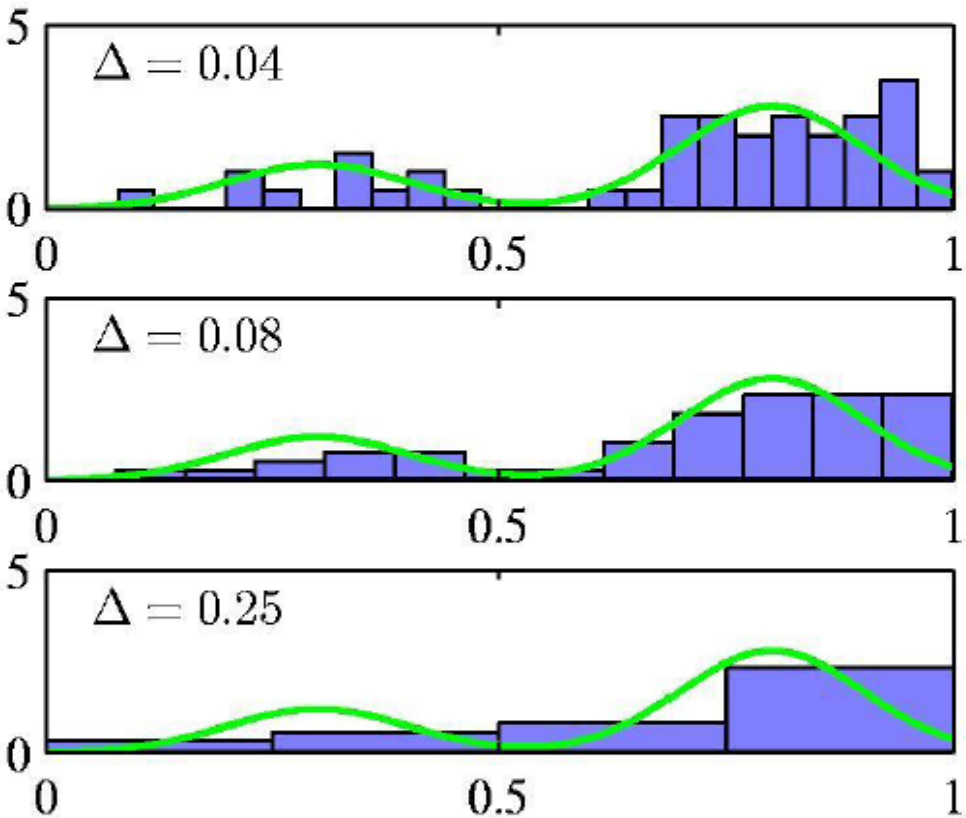
Large Δ, small #bins ⟺ "**Large bias**, **Small variance**"

Bias-Variance tradeoff

Choice of #bins

$$\widehat{p}(x) = \frac{n_i}{n\Delta}\mathbf{1}_{x\in \text{Bin}_i}$$

\# bins = $1/\Delta$



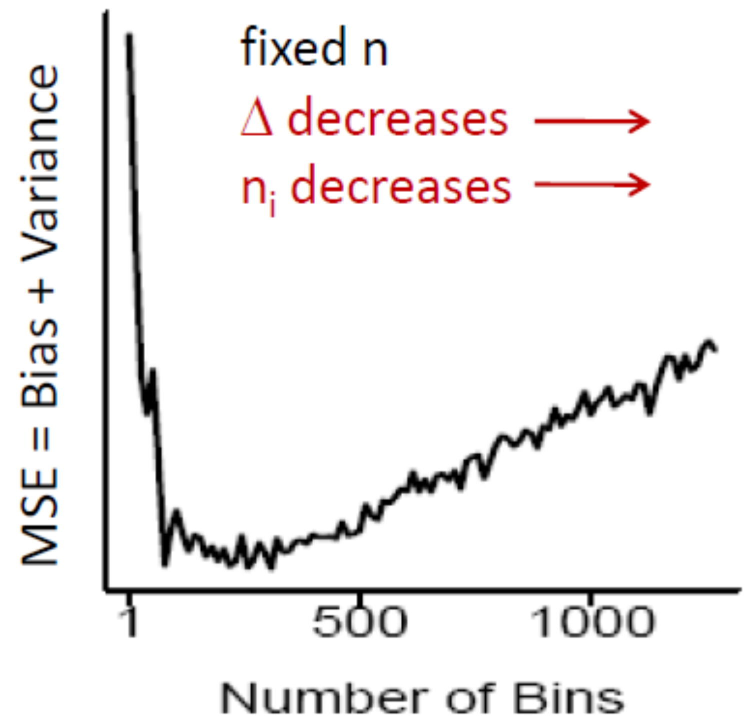
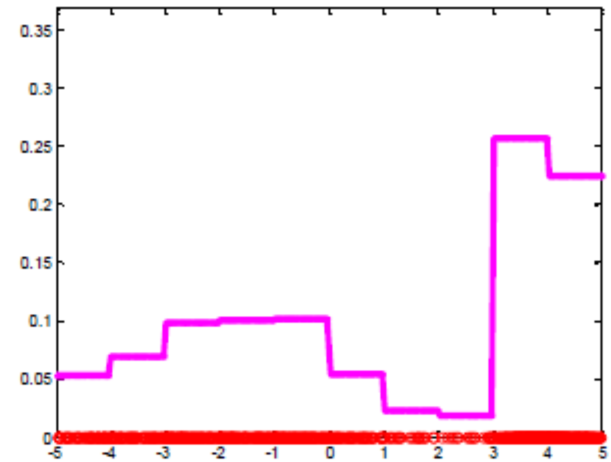


Image src: Bishop book

Kernel density estimate

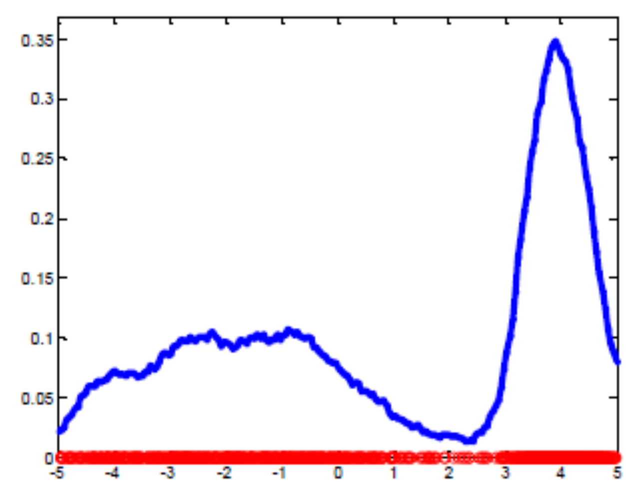
- Histogram – blocky estimate

$$\widehat{p}(x) = \frac{1}{\Delta}\frac{\sum_{j=1}^{n} \mathbf{1}_{X_j \in \text{Bin}_x}}{n}$$



- Kernel density estimate aka "Parzen/moving window method"

$$\widehat{p}(x) = \frac{1}{\Delta}\frac{\sum_{j=1}^{n} \mathbf{1}_{||X_j - x|| \le \Delta}}{n}$$



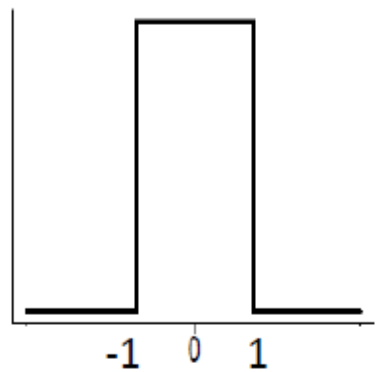
Kernel density estimate

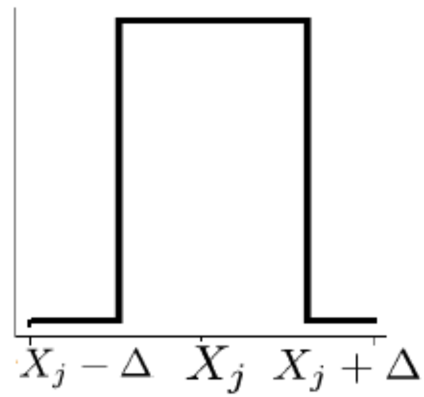
- $\widehat{p}(x) = \frac{1}{\Delta} \frac{\sum_{j=1}^{n} K\left(\frac{X_j - x}{\Delta}\right)}{n}$ more generally

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

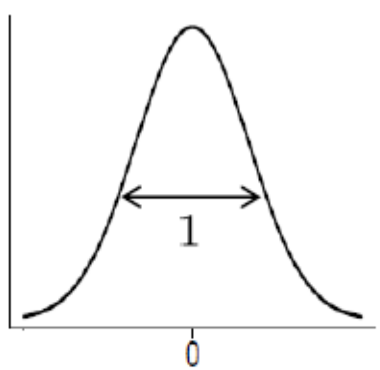
$$K(x) = \frac{1}{2} I(x),$$

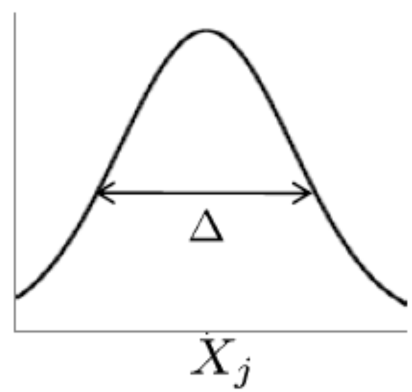




Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

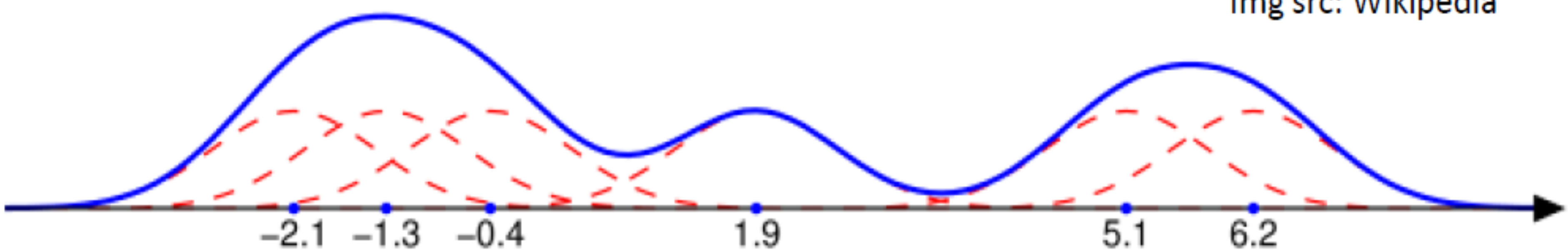




Kernel density estimation

- Place small "bumps" at each data point, determined by the kernel function.
- The estimator consists of a (normalized) "sum of bumps".

Img src: Wikipedia



Gaussian bumps (red) around six data points and their sum (blue)

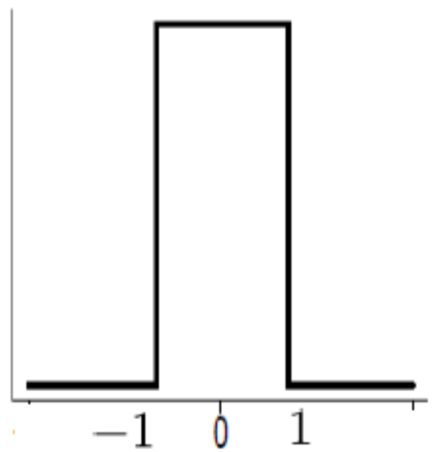
$$p(x) = \frac{1}{6\sqrt{2\pi}}\left(e^{\frac{(x+2.1)^2}{-2}} + e^{\frac{(x+1.3)^2}{-2}} + e^{\frac{(x+0.4)^2}{-2}} + e^{\frac{(x-1.9)^2}{-2}} + e^{\frac{(x-5.1)^2}{-2}} + e^{\frac{(x-6.2)^2}{-2}}\right)$$

- Note that where the points are denser the density estimate will have higher values.

Kernels

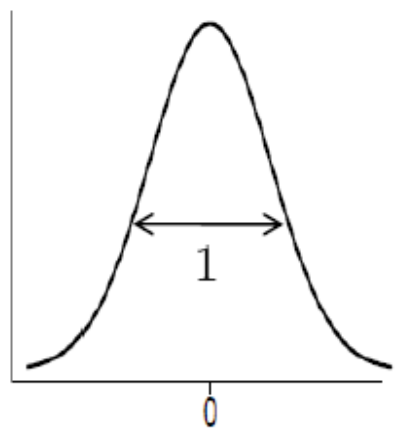
boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$



Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



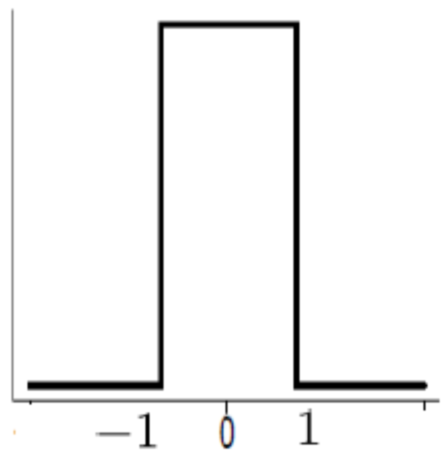
Any kernel function that satisfies

$$\begin{aligned} K(x) &\geq 0, \\ \int K(x)dx &= 1 \end{aligned}$$

Kernels

boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$

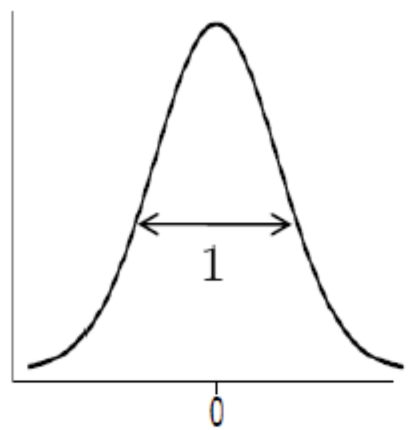


Finite support

– only need local points to compute estimate

Gaussian kernel :

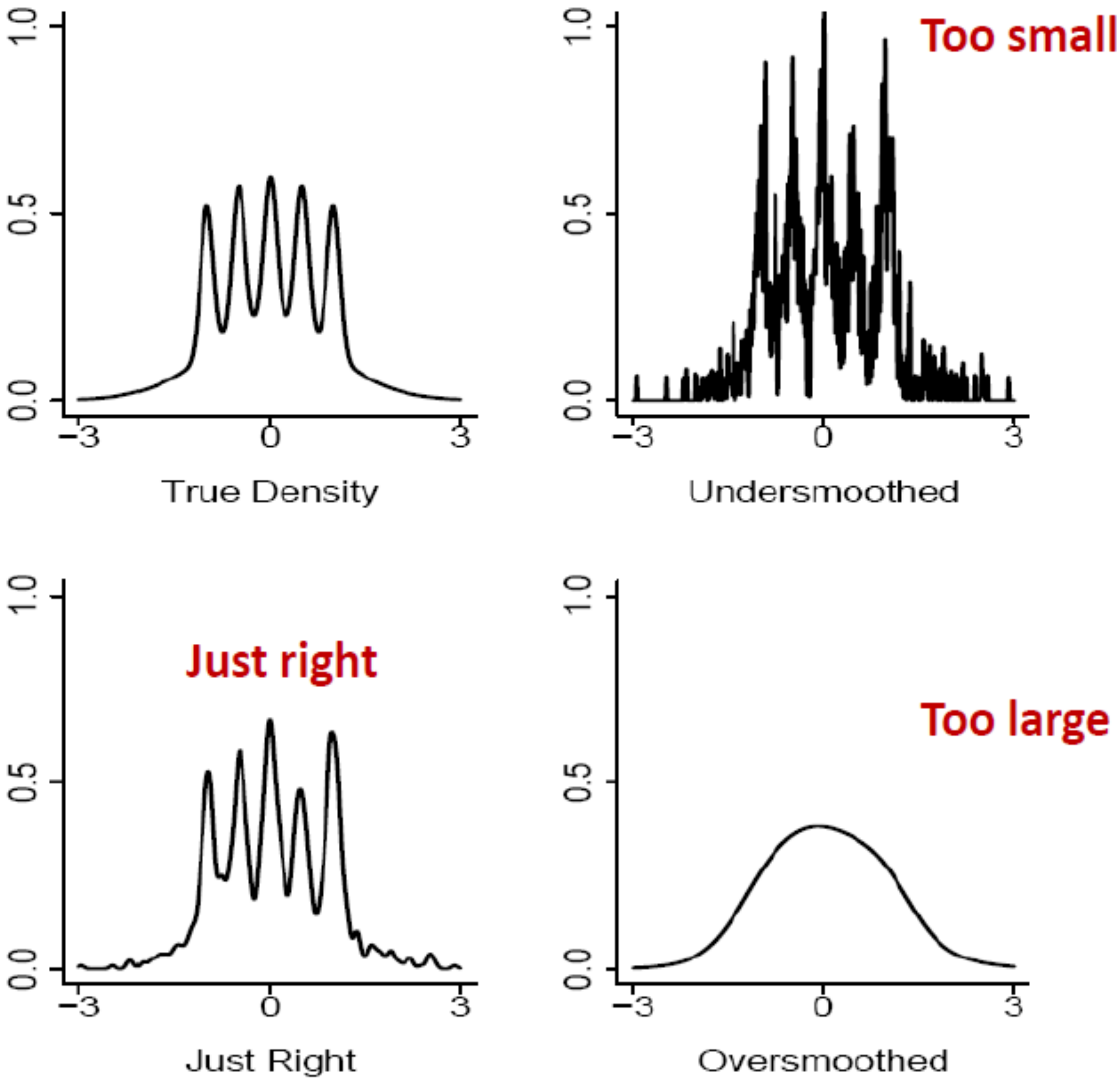
$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



Infinite support

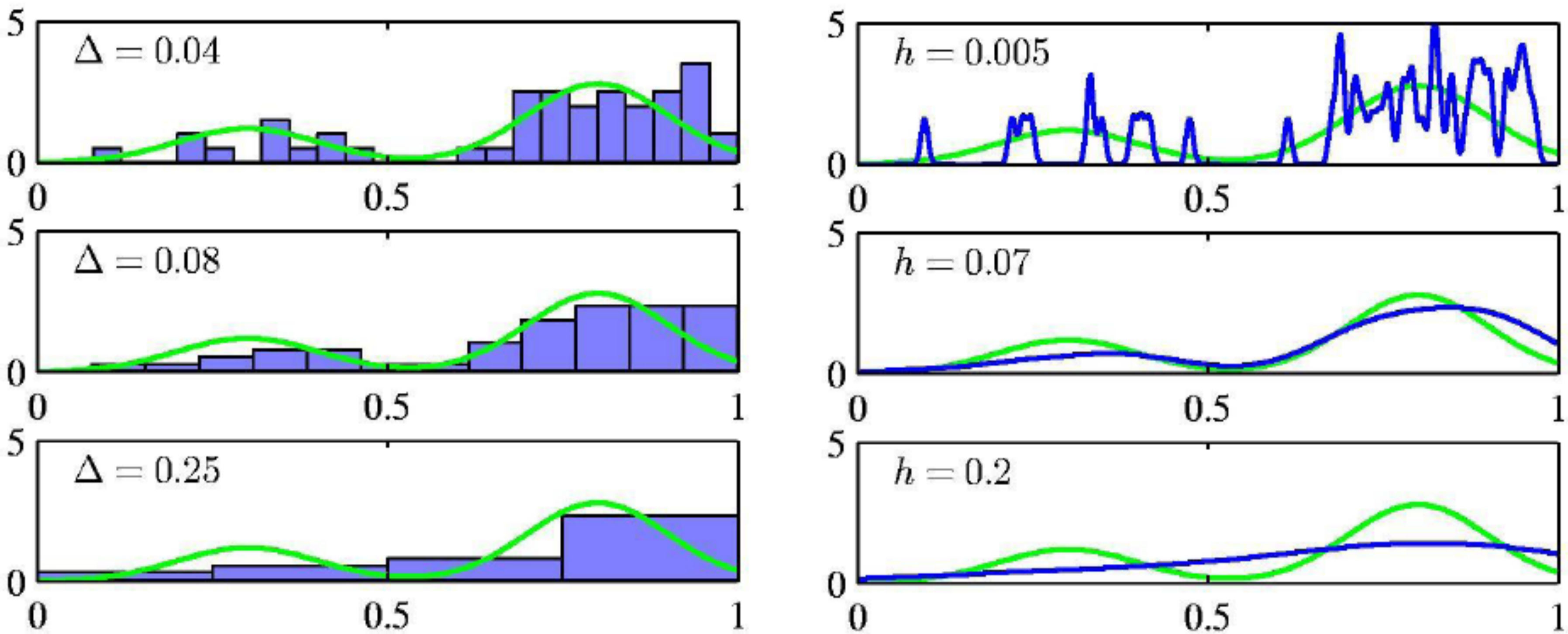
- need all points to compute estimate
-But quite popular since smoother

Choice of kernel bandwidth



Bart-Simpson Density

Histograms vs. Kernel density estimation



$\Delta = h$ acts as a smoother.

k-NN (Nearest Neighbor) density estimation

- Histogram $$\widehat{p}(x) = \frac{n_i}{n\Delta}\mathbf{1}_{x \in \mathrm{Bin}_i}$$
- Kernel density est $$\widehat{p}(x) = \frac{n_x}{n\Delta}$$

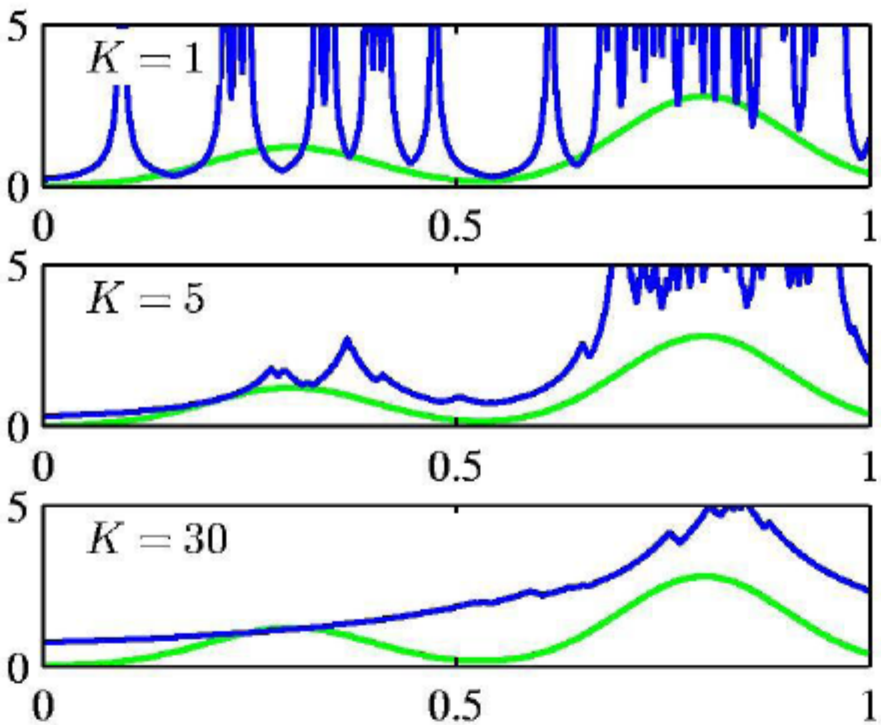
Fix Δ, estimate number of points within Δ of x (n_i or n_x) from data

Fix $n_x = k$, estimate Δ from data (volume of ball around x that contains k training pts)

- k-NN density est $$\widehat{p}(x) = \frac{k}{n\Delta_{k,x}}$$

k-NN density estimation

$$\widehat{p}(x) = \frac{k}{n\Delta_{k,x}}$$

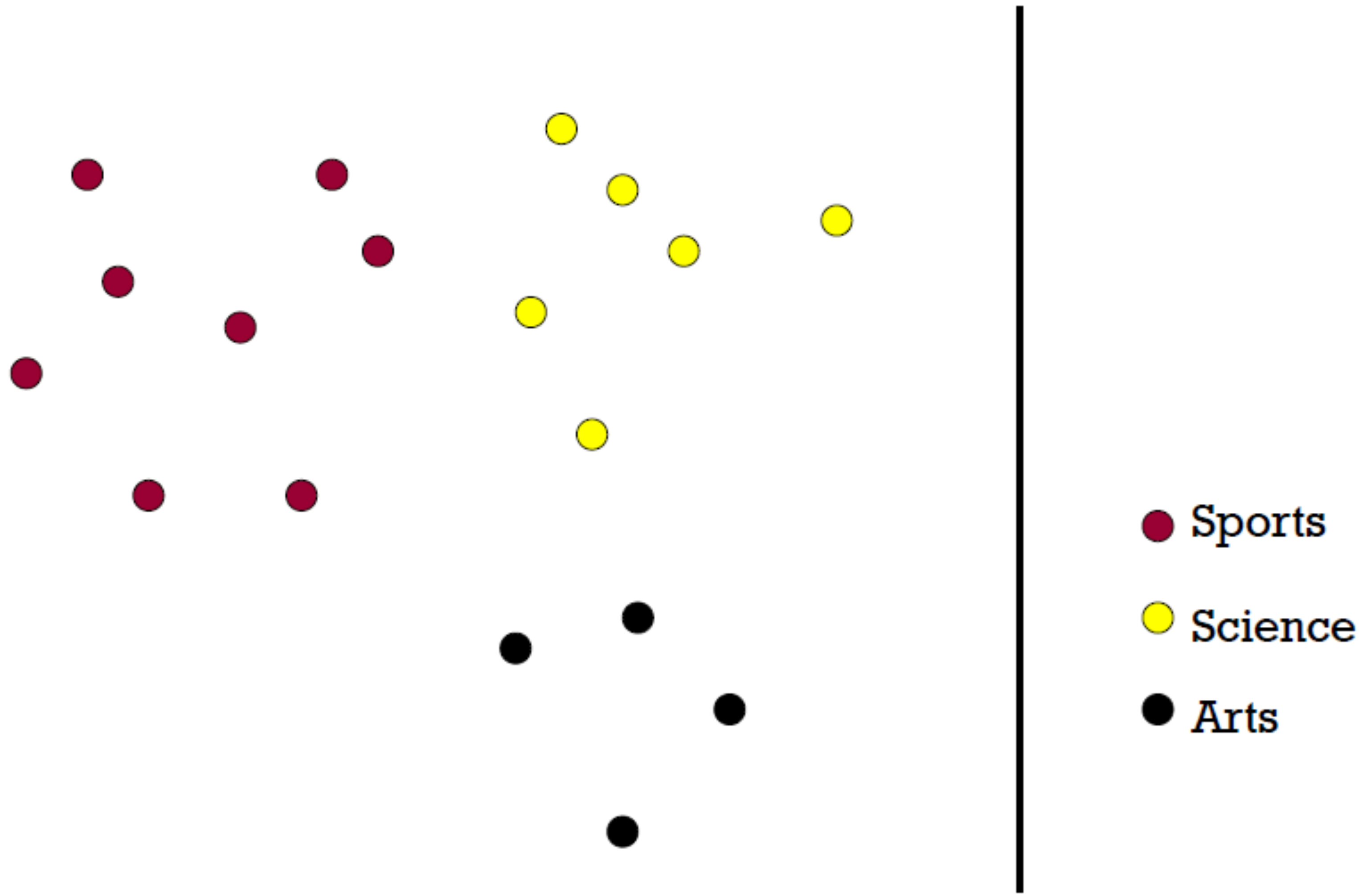


k acts as a smoother.

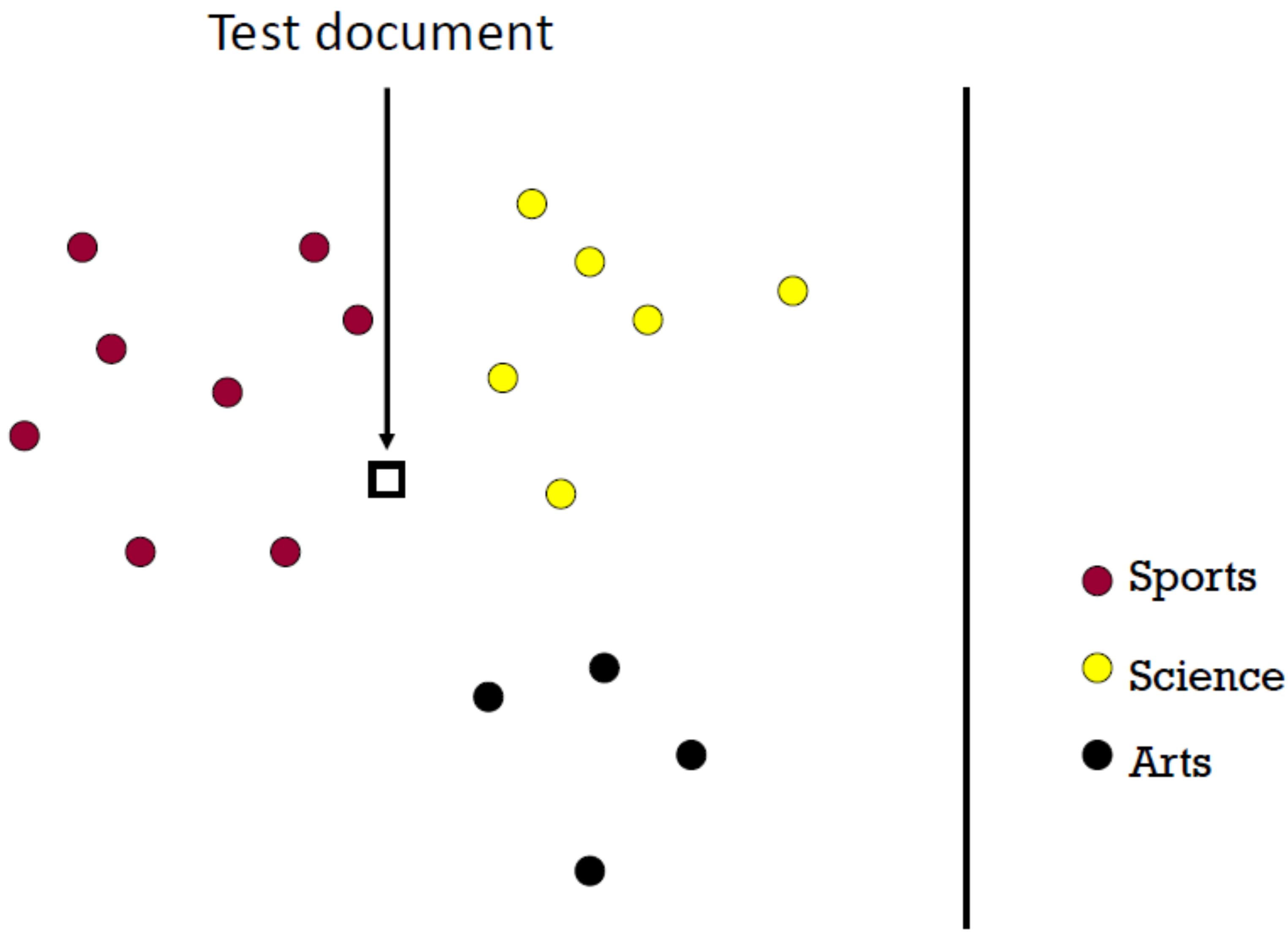
Not very popular for density estimation - expensive to compute, bad estimates

But a related version
for classification quite popular
…

k-NN classifier



k-NN classifier



k-NN classifier

- Optimal Classifier: $$f^*(x) = \arg\max_y P(y|x) = \arg\max_y p(x|y)P(y)$$

- k-NN Classifier: $$\widehat{f}_{kNN}(x) = \arg\max_y \ \widehat{p}_{kNN}(x|y)\widehat{P}(y) = \arg\max_y \ k_y$$ (Majority vote)

$$\widehat{p}_{kNN}(x|y) = \frac{k_y}{n_y \Delta_{k,x}}$$

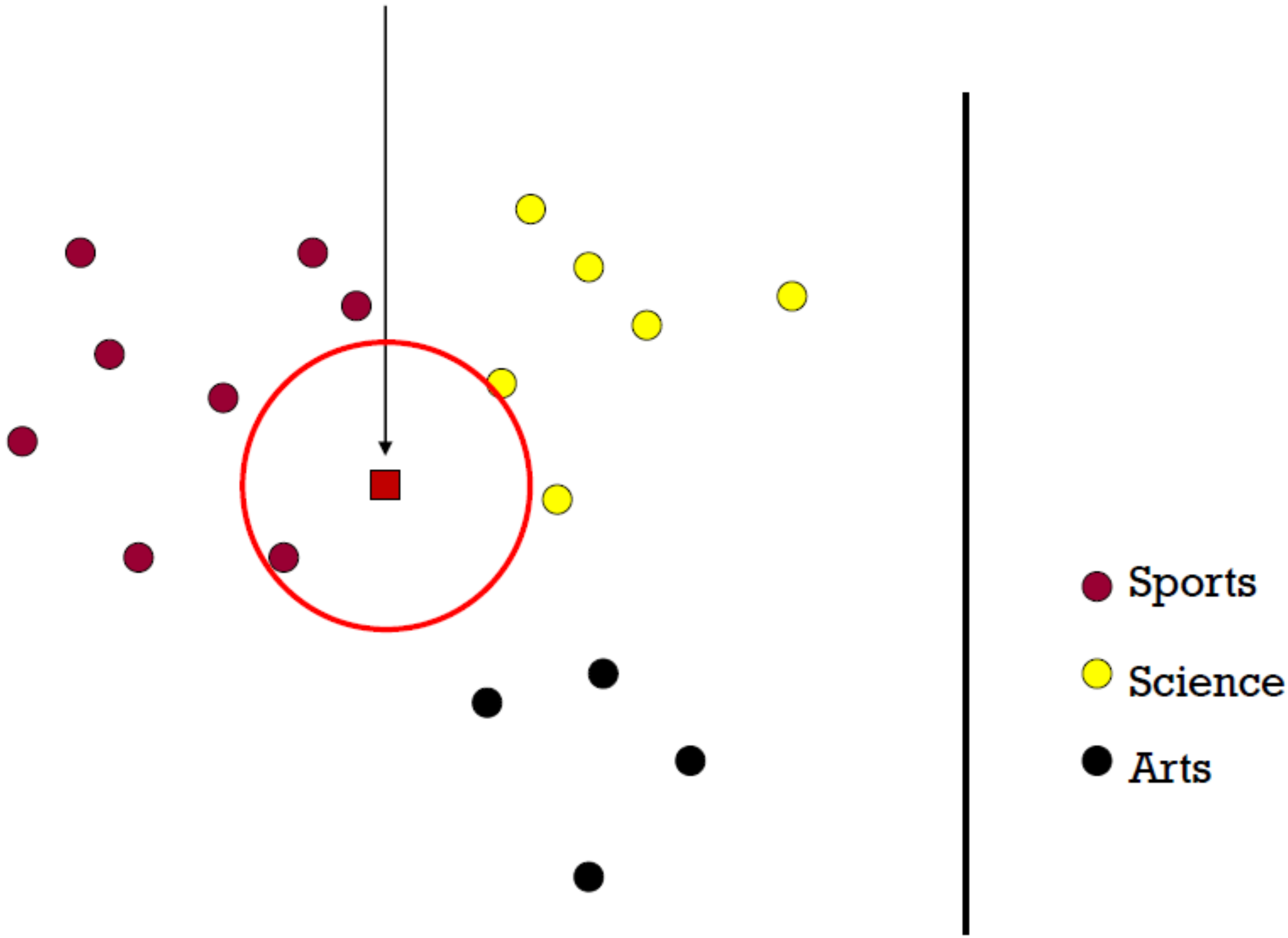
k_y → # training pts of class y that lie within Δ_k ball

n_y → # total training pts of class y

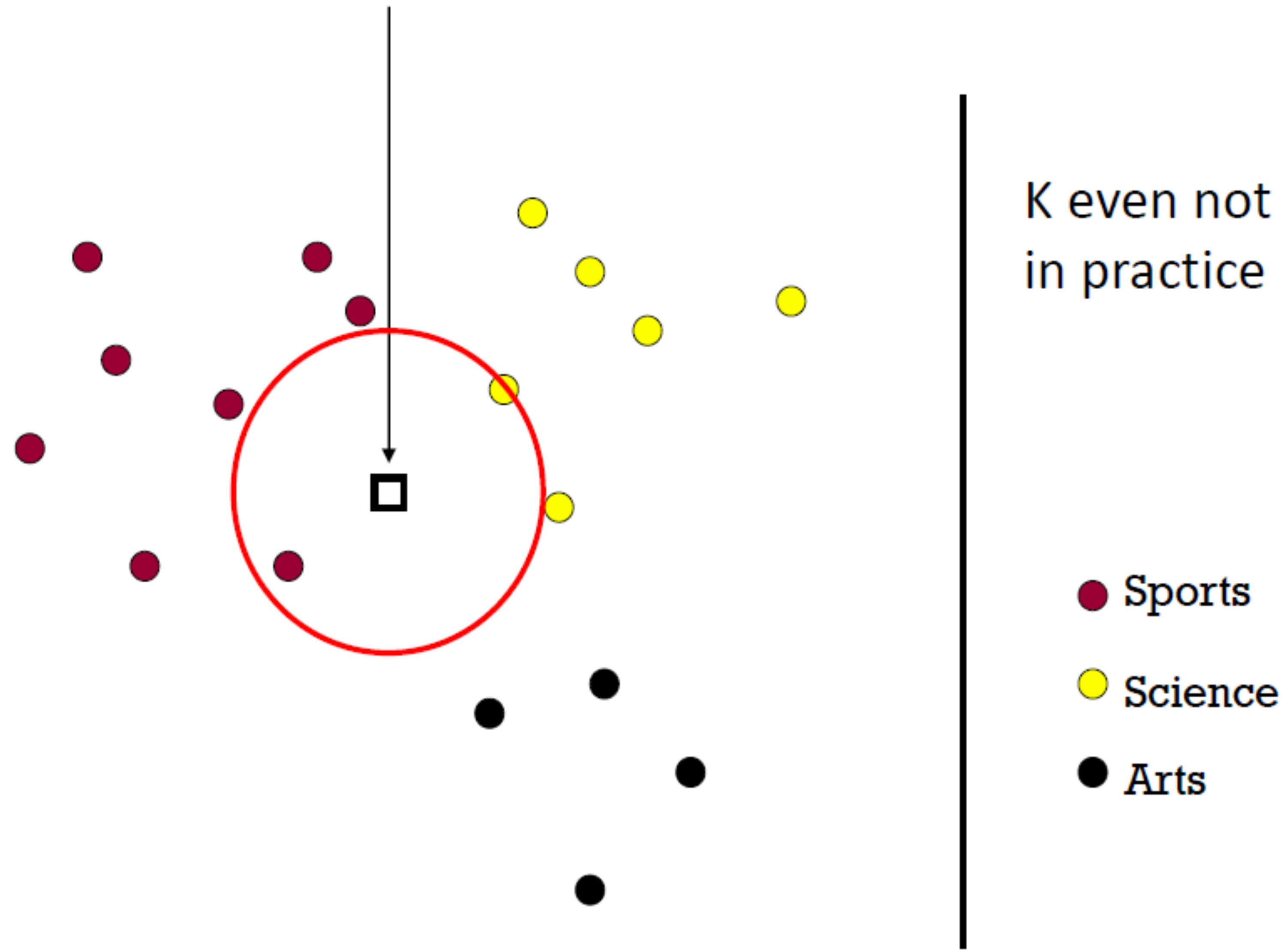
$$\sum_y k_y = k$$

$$\widehat{P}(y) = \frac{n_y}{n}$$

1-Nearest Neighbor (kNN) classifier

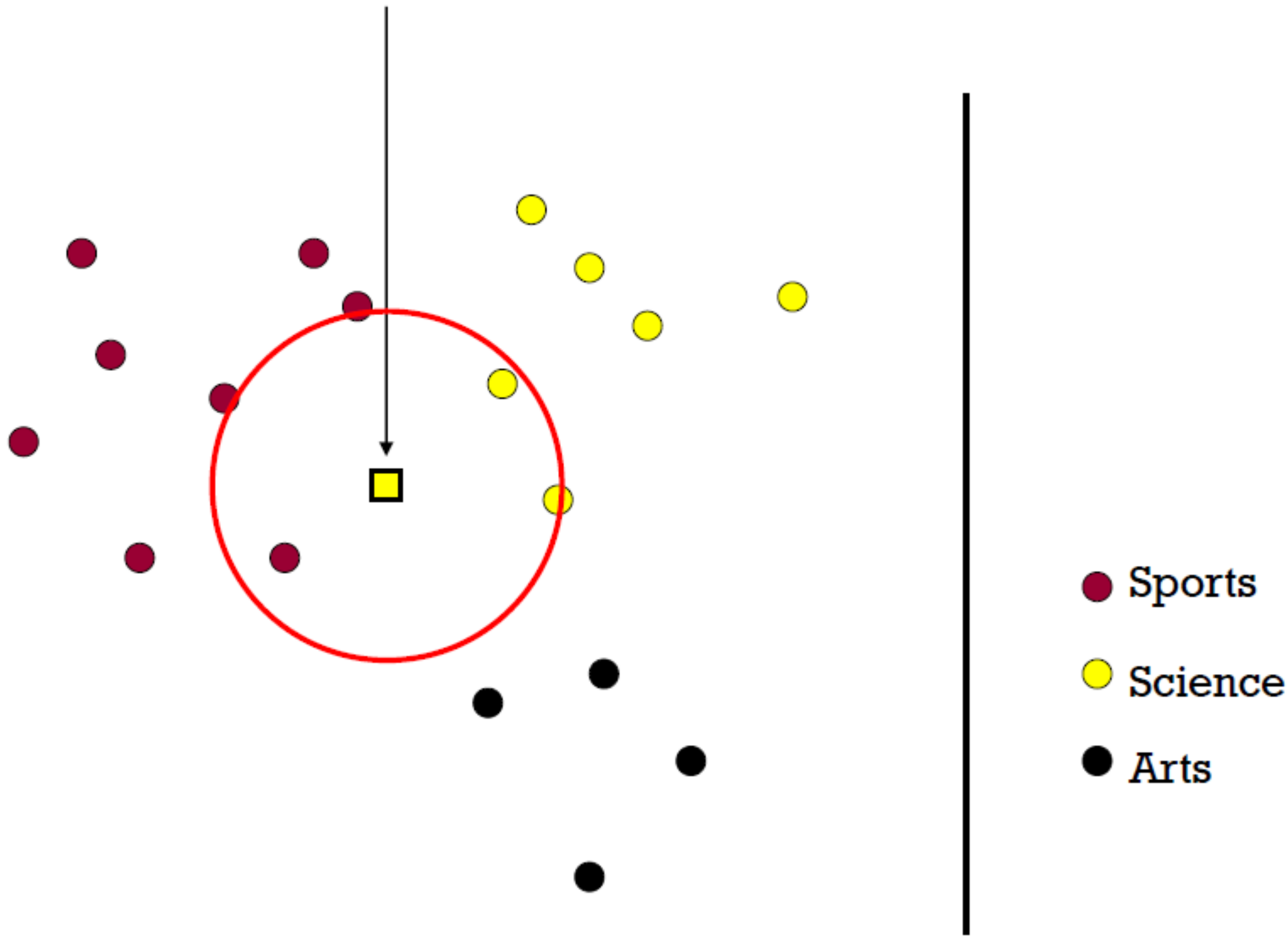


2-Nearest Neighbor (kNN) classifier



K even not used in practice

3-Nearest Neighbor (kNN) classifier



What is the best K?

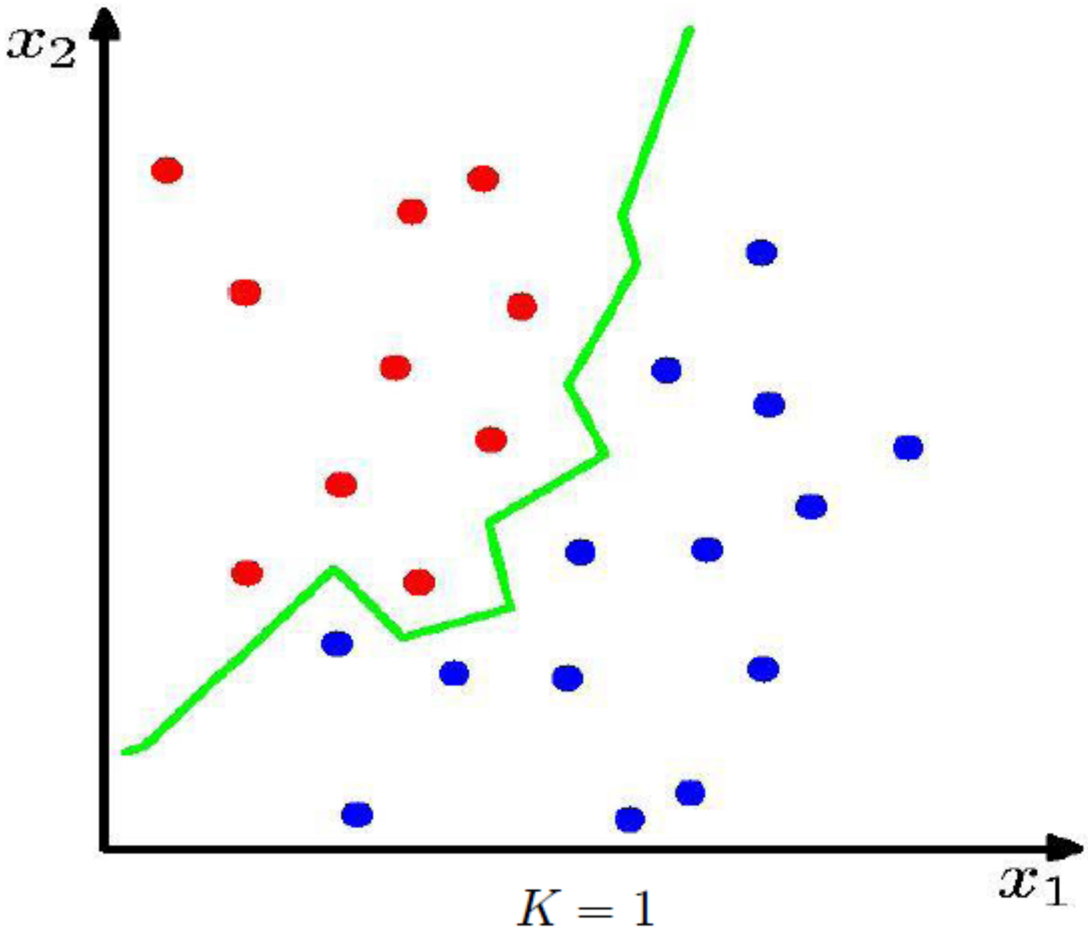
Bias-variance tradeoff

Larger K => predicted label is more stable

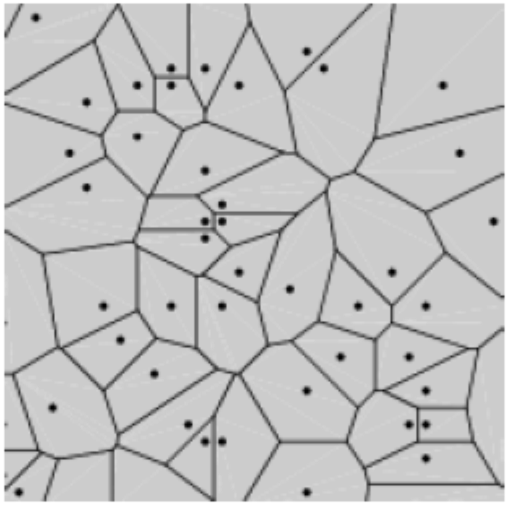
Smaller K => predicted label is more accurate

Similar to density estimation

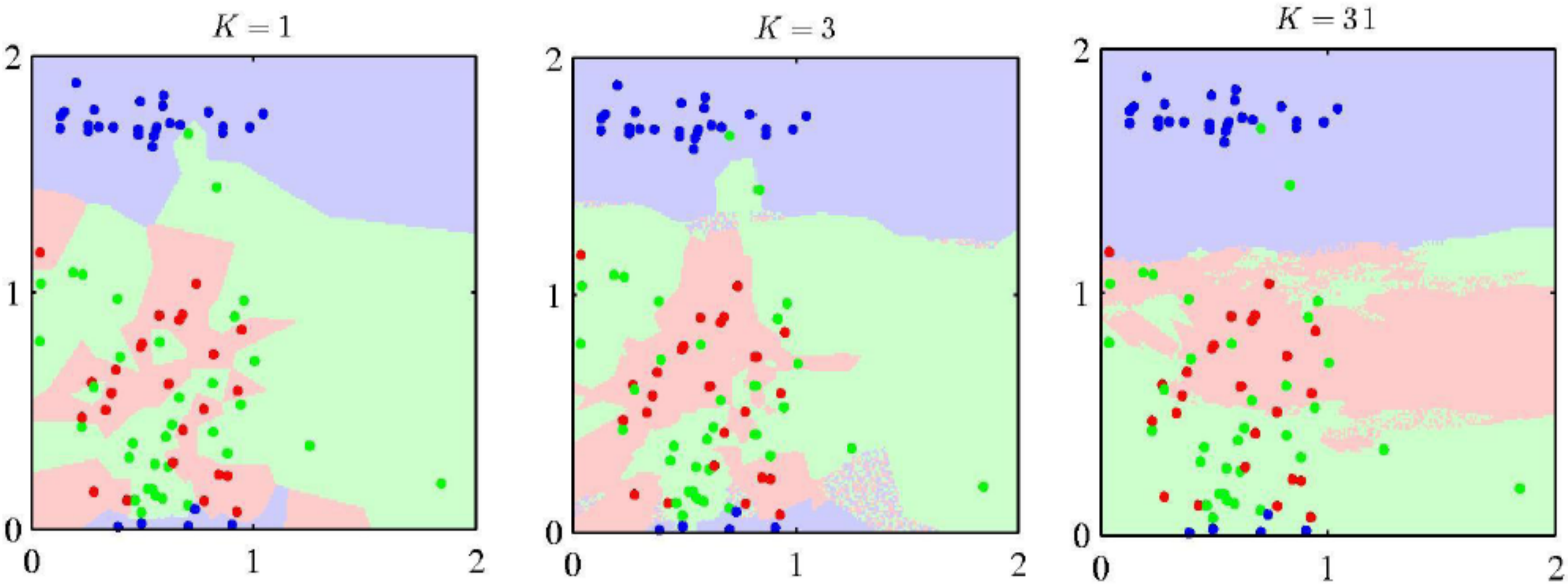
1-NN classifier – decision boundary



Voronoi
Diagram



k-NN classifier – decision boundary



- K acts as a smoother (Bias-variance tradeoff)
- Guarantee: For $n \to \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error.

Case Study:
kNN for Web Classification

- Dataset
  - 20 News Groups (20 classes)
  - Download :(http://people.csail.mit.edu/jrennie/20Newsgroups/)
  - 61,118 words, 18,774 documents
  - Class labels descriptions

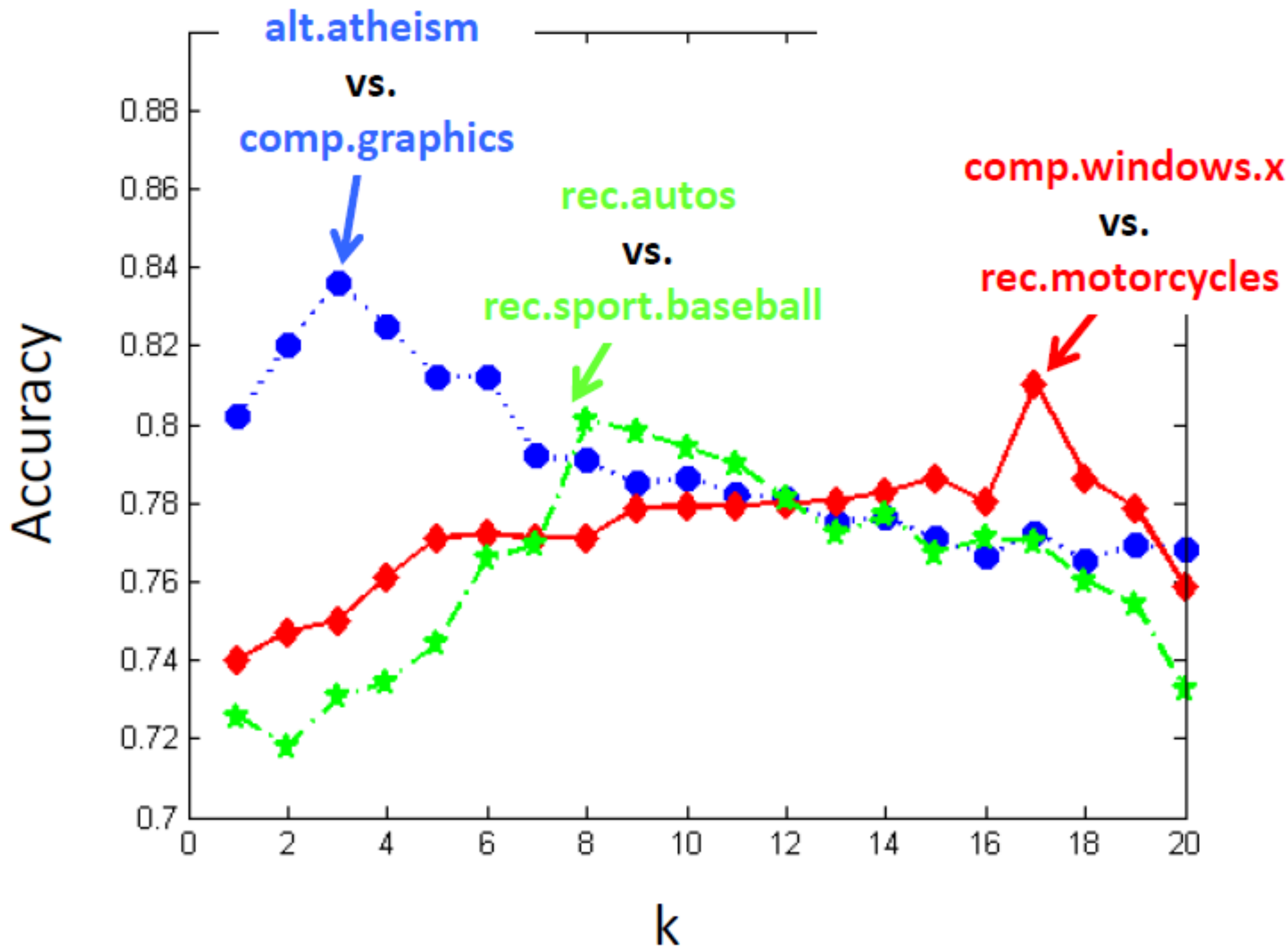
comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Experimental Setup

- Training/Test Sets:
  - 50%-50% randomly split.
  - 10 runs
  - report average results
- Evaluation Criteria:

$$Accuracy = \frac{\sum_{i \in test\ set} \mathrm{I}(predict_i == true\ label_i)}{\#\ of\ test\ samples}$$

Results: Binary Classes

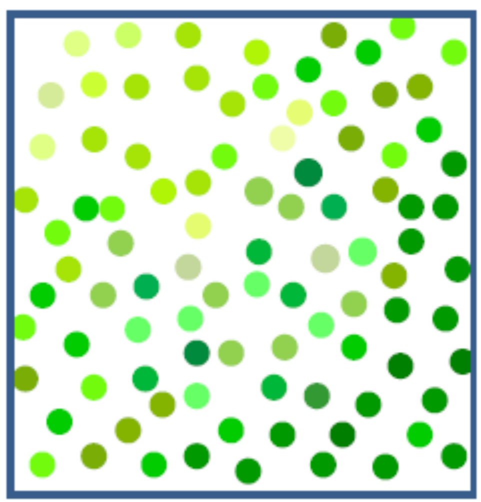


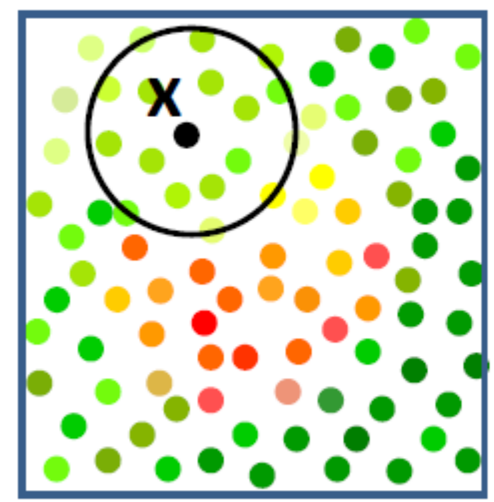
From Classification to Regression

Temperature sensing

- What is the temperature in the room?

at location x?





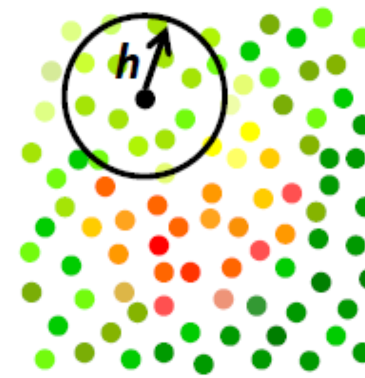
$$\widehat{T} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

$$\widehat{T}(x) = \frac{\sum_{i=1}^{n} Y_i \mathbf{1}_{||X_i - x|| \leq h}}{\sum_{i=1}^{n} \mathbf{1}_{||X_i - x|| \leq h}}$$

Average

"Local" Average

Kernel Regression



- Aka Local Regression
- Nadaraya-Watson Kernel Estimator

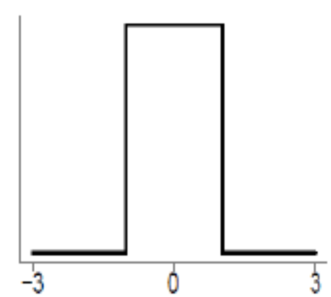
$$\widehat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \quad \text{Where} \quad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

- Weight each training point based on distance to test point
- Boxcar kernel yields

  local average

boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$

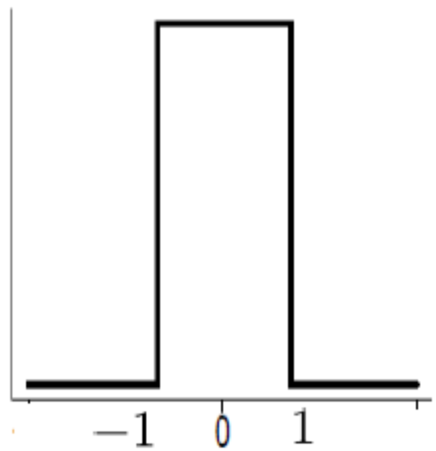


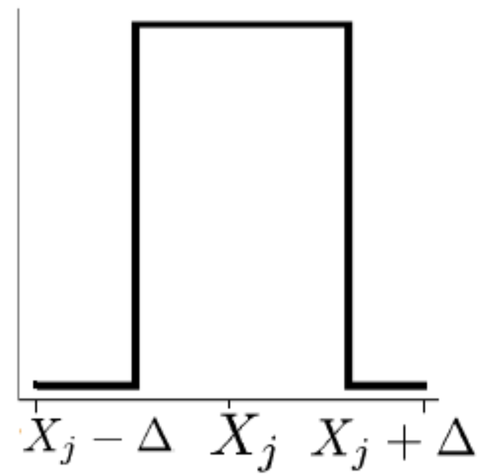
Kernels

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

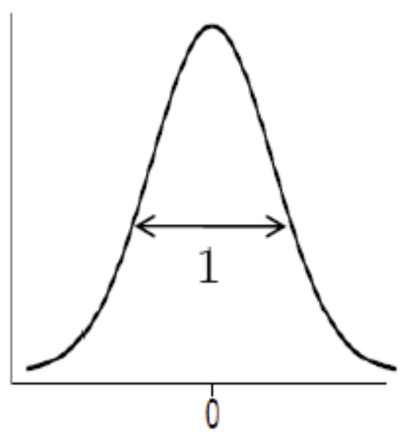
$$K(x) = \frac{1}{2} I(x),$$

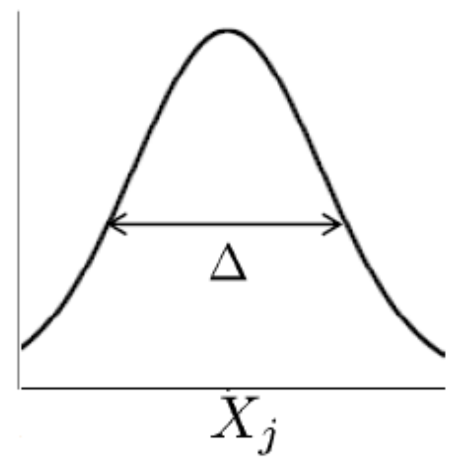




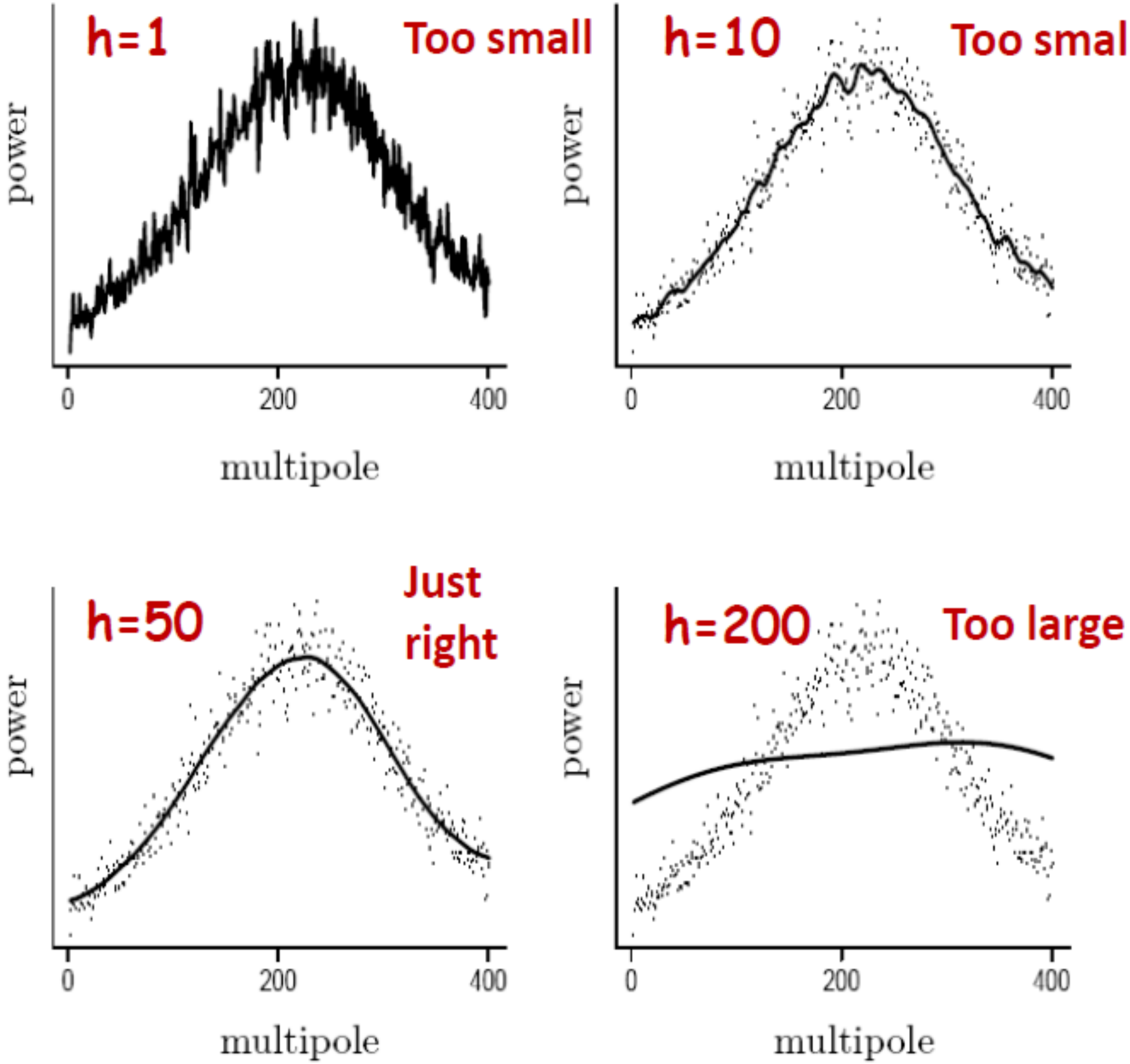
Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$





Choice of kernel bandwidth h



Choice of kernel is not that important

Kernel Regression as Weighted Least Squares

$$\min_f \sum_{i=1}^{n} w_i (f(X_i) - Y_i)^2 \qquad\qquad w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

Weighted Least Squares

Kernel regression corresponds to locally constant estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta$ (a constant)

Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta$ (a constant)

$$\min_{\beta} \sum_{i=1}^{n} w_i(\beta - Y_i)^2$$

constant

$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

$$\frac{\partial J(\beta)}{\partial \beta} = 2\sum_{i=1}^{n} w_i(\beta - Y_i) = 0$$

Notice that $\sum_{i=1}^{n} w_i = 1$

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

Local Linear/Polynomial Regression

$$\min_f \sum_{i=1}^{n} w_i(f(X_i) - Y_i)^2$$

$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

Weighted Least Squares

Step:

1. Calculate the weight w_i of each X_i with respect to X
2. Do weighted linear regression to obtain the weight of each dimension

Least Squares Estimator

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$= \boldsymbol{A^T A \beta \beta^T - 2\beta^T A^T Y + Y^T Y}$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = \boldsymbol{= 2A^T A\beta - 2A^T Y = 0}$$

$$\boldsymbol{\beta = (A^T A)^{-1} A^T Y}$$

Least Squares Estimator

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$= A^TA\beta\beta^T - 2\beta^TA^TY + Y^TY$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0 \quad = 2A^TA\beta - 2A^TY = 0$$

$$\beta = (WA^TA)^{-1}WA^TY$$

Weighted Least Squares Estimator

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sqrt{w}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = \sqrt{w}(\mathbf{A}\beta - \mathbf{Y})^T \sqrt{w}(\mathbf{A}\beta - \mathbf{Y})$$

$$= \sqrt{w}\boldsymbol{A^T A\beta\beta^T} - 2\sqrt{w}\boldsymbol{\beta^T A^T Y} + \sqrt{w}\boldsymbol{Y^T Y}$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\hat{\beta}} = 0 \quad = 2\sqrt{w}\boldsymbol{A^T A\beta} - 2\sqrt{w}\boldsymbol{A^T Y} = 0$$

$$\boldsymbol{\beta} = (\boldsymbol{W A^T A})^{-1}\boldsymbol{W A^T Y} = (\boldsymbol{A^T W A})^{-1}\boldsymbol{A^T W Y} = (\boldsymbol{A^T}\sqrt{\boldsymbol{w}}\sqrt{\boldsymbol{w}}\boldsymbol{A})^{-1}\boldsymbol{A^T}\sqrt{\boldsymbol{w}}\sqrt{\boldsymbol{w}}\boldsymbol{Y}$$

Summary

- Instance based/non-parametric approaches

Four things make a memory based learner:

1. *A distance metric, dist(x,X_i)*
   Euclidean (and many more)
2. *How many nearby neighbors/radius to look at?*
   k, Δ/h
3. *A weighting function (optional)*
   W based on kernel K
4. *How to fit with the local points?*
   Average, Majority vote, Weighted average

Summary

- Parametric vs Nonparametric approaches

  - ➢ Nonparametric models place very mild assumptions on the data distribution and provide good models for complex data

    Parametric models rely on very strong (simplistic) distributional assumptions

  - ➢ Nonparametric models (not histograms) requires storing and computing with the entire data set.

    Parametric models, once fitted, are much more efficient in terms of storage and computation.

What you should know…

- Histograms, Kernel density estimation
  - Effect of bin width/ kernel bandwidth
  - Bias-variance tradeoff
- K-NN classifier
  - Nonlinear decision boundaries
- Kernel (local) regression
  - Interpretation as weighted least squares
  - Local constant/linear/polynomial regression

Demo – KNN Classification Boundary

https://www.youtube.com/watch?v=96cb-6StcIc