

Memory Based Learning  
(with Bayesian underpinnings)

**Frank Dellaert**  
Machine Learning Course  
Department of Computer Science  
Carnegie Mellon University

October 28 1997



# Memory Based Learning

---

## Outline:

- Density Estimation
- Locally Weighted Averaging
- K-Nearest Neighbors
- Locally Weighted Regression
- Radial Basis Function Networks
- KD-Trees

PS. Sloppy notation and vague concepts will be used throughout the lecture.

## Mixture of:

- Duda & Hart 1979. Pattern Classification and Scene Analysis.
- Chris Bishop 1995. Neural Networks for Pattern Recognition.
- Tom Mitchell 1997. Machine Learning.

# Key Question

---

Interesting classification problem:

- Training instances: (midterm grade, G/U)
- $(x_i, C_i)$ , where  $C_i \in \{C_g, C_u\}$
- $x_i$  is a one-dimensional vector !!
- Given a midterm score  $x$ , how likely is it to be produced by a graduate student ?

Some questions for you:

- Q: what is this measure ?
- A: posterior probability  $P(C_g|x)$
- Q: how do we compute it ?
- A: Bayes law

$$P(C_g|x) = \frac{P(C_g|x)P(C_g)}{P(x)}$$

# A speck of Decision Theory (because you need it)

---

- Actions  $\alpha_1, \alpha_2, \dots, \alpha_a$ .
- Classes  $\omega_1, \omega_2, \dots, \omega_s$ .
- $a \neq s$
- cost  $\lambda(\alpha_i|\omega_j)$
  
- Q: given  $x$ , probability of class  $\omega_j$  ?
- A:

$$P(\omega_j|x)$$

- Q: expected cost of taking decision  $\alpha_i$  ?
- A:

$$R(\alpha_i|x) = \sum_{j=1}^s \lambda(\alpha_i|\omega_j)P(\omega_j|x)$$

# Making an Optimal Decision

---

- *conditional risk*:

$$R(\alpha_i|x) = \sum_{j=1}^s \lambda(\alpha_i|\omega_j)P(\omega_j|x)$$

- *Bayes risk* =

$$\alpha_{optimal} = \arg \min_i R(\alpha_i|x)$$

Example:

$\omega_1 = Graduate Student$

$\omega_2 = Undergrad$

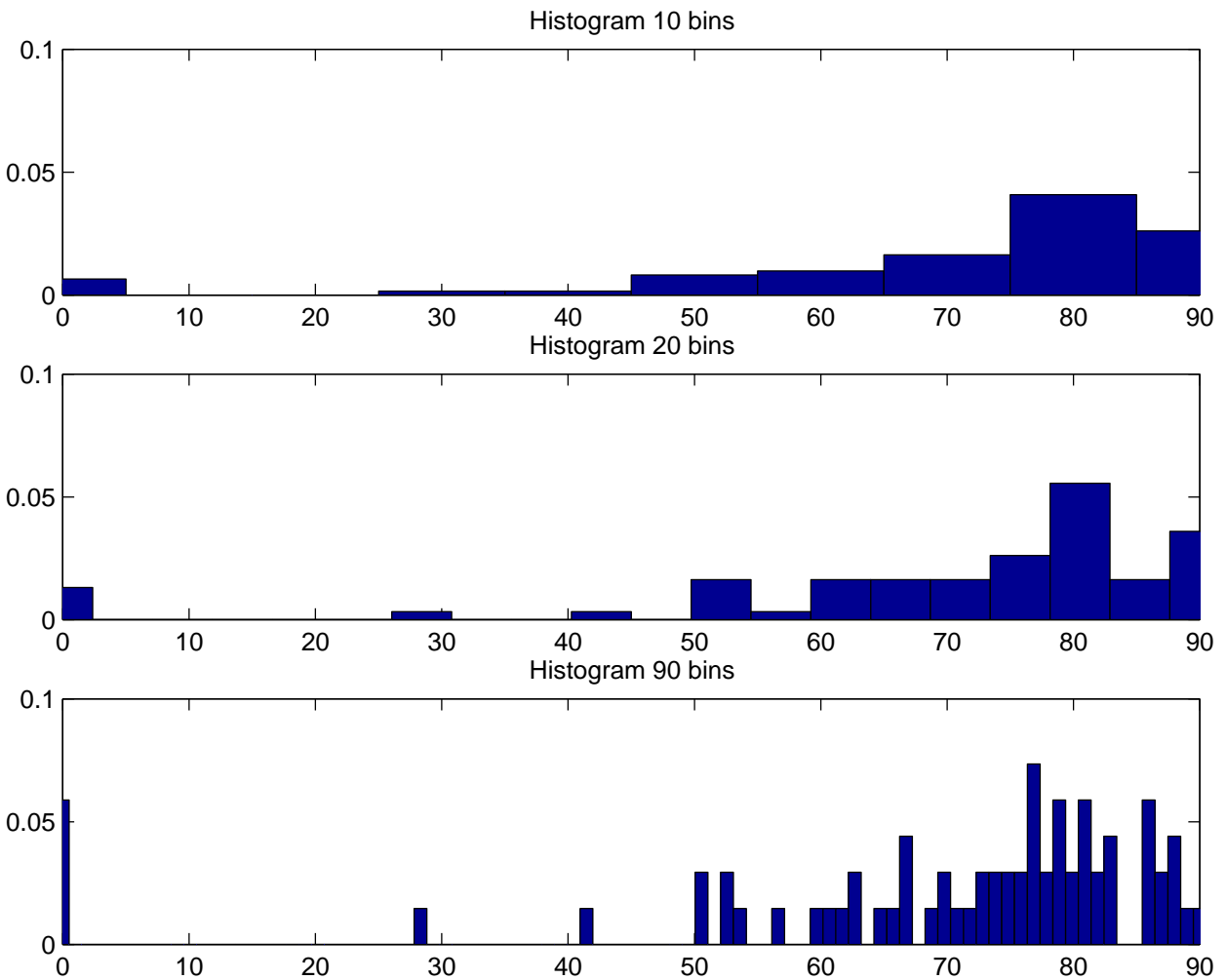
$\alpha_1 = Say Grad$

$\alpha_2 = Say Don't know$

$\alpha_3 = Say Undergrad$

# Histograms

---



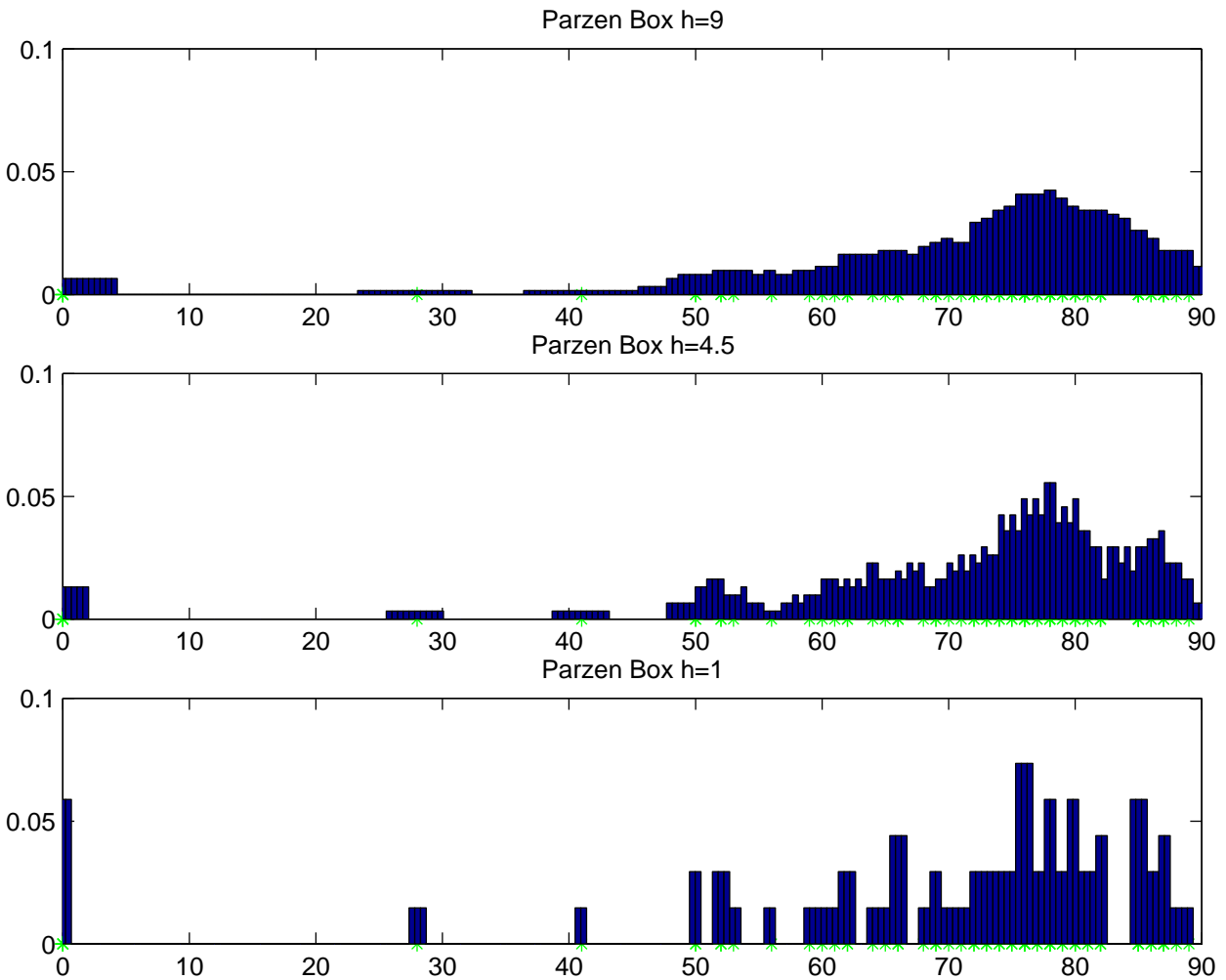
# Properties of Histograms

---

- non-parametric
- *smoothing parameter* = number of bins
- we can discard data afterwards
- discontinuous
- boundaries arbitrary
- d dimensions  $\rightarrow M^d$  bins
- **curse of dimensionality**

# Dropping boxes

---



# Properties Dropping boxes

---

- non-parametric
- *smoothing parameter* = size of the box
- need the data always
- still discontinuous
- boundaries data-driven
- a bit less of a curse

# Foundation of Density Estimation

---

- Assume  $P(x)$  slowly varying and continuous
- Q:  $P_V$  of landing in a volume  $V$  around  $x$ ?
- A:  $P_V \approx P(x)V$
- Q: how many out of  $N$  samples fall into a volume  $V$  ?
- A:  $E[K] = P_V N \approx P(x)VN$
- Q: what is an estimator for  $P(x)$  ?
- A:

$$P(x) \approx \frac{K}{NV}$$

Note: *biased* estimator

# Parzen Window Density Estimation

---

Hypercube kernel (Parzen window):

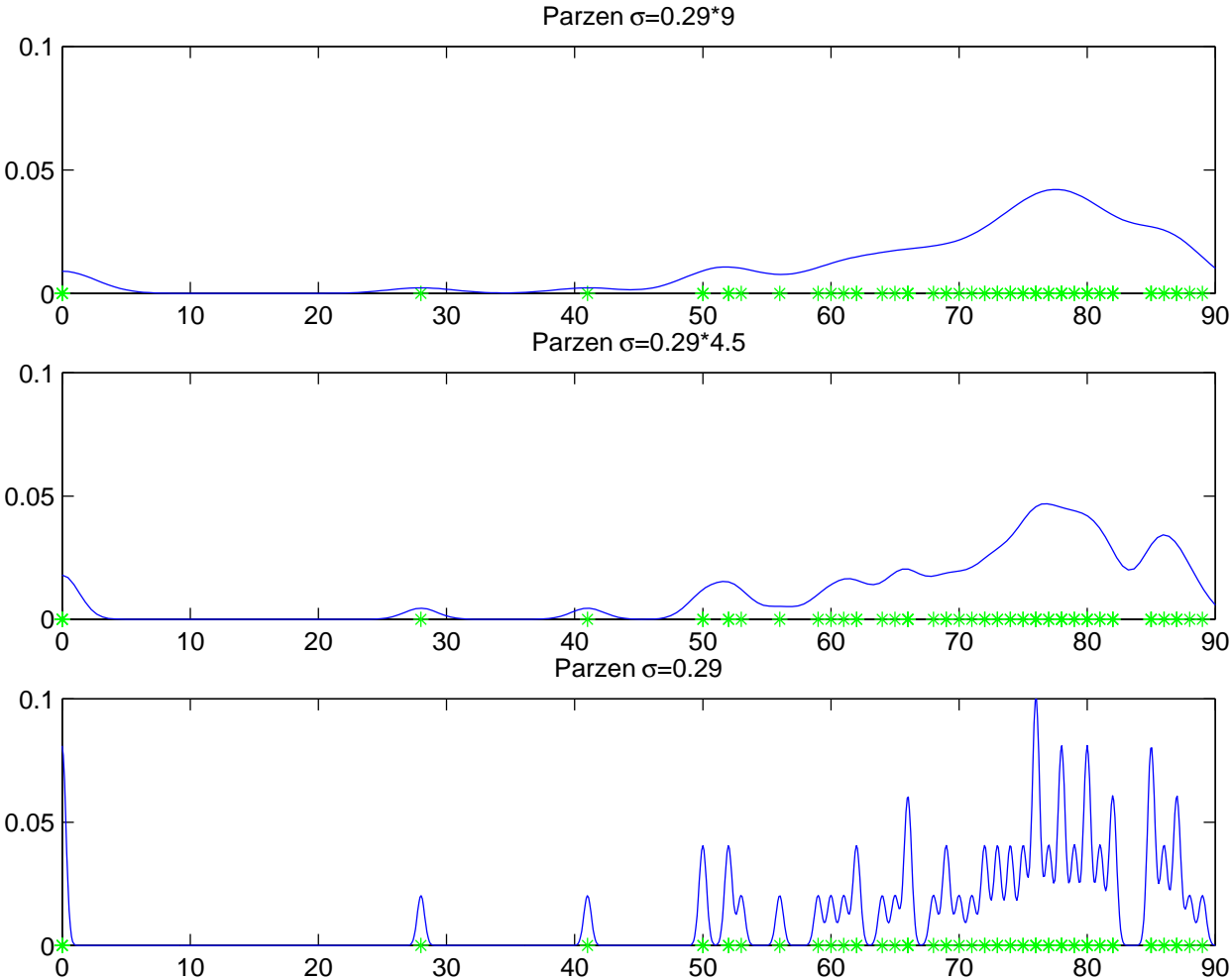
$$H(u) = \begin{cases} 1 & |u_j| < 1/2 \quad j = 1, \dots, d \\ 0 & \text{otherwise} \end{cases}$$

$$K = \sum_{i=1}^N H\left(\frac{x - x_i}{h}\right)$$

$$P(x) \approx \frac{1}{N} \sum_{i=1}^N \frac{1}{h^d} H\left(\frac{x - x_i}{h}\right)$$

**Trick:** Think about only one box, not many boxes.

# Gaussian Parzen Windows



# Gaussian Parzen Windows

---

$$P(x) \approx \frac{1}{N} \sum_{i=1}^N G(x, x_i, \sigma)$$

Properties:

- non-parametric
- *smoothing parameter* = standard deviation
- need the data always
- continuous
- no boundaries
- more expensive

# Densities and Classification

---

Summary:

1. Histograms
2. Hypercubes
3. Gaussians

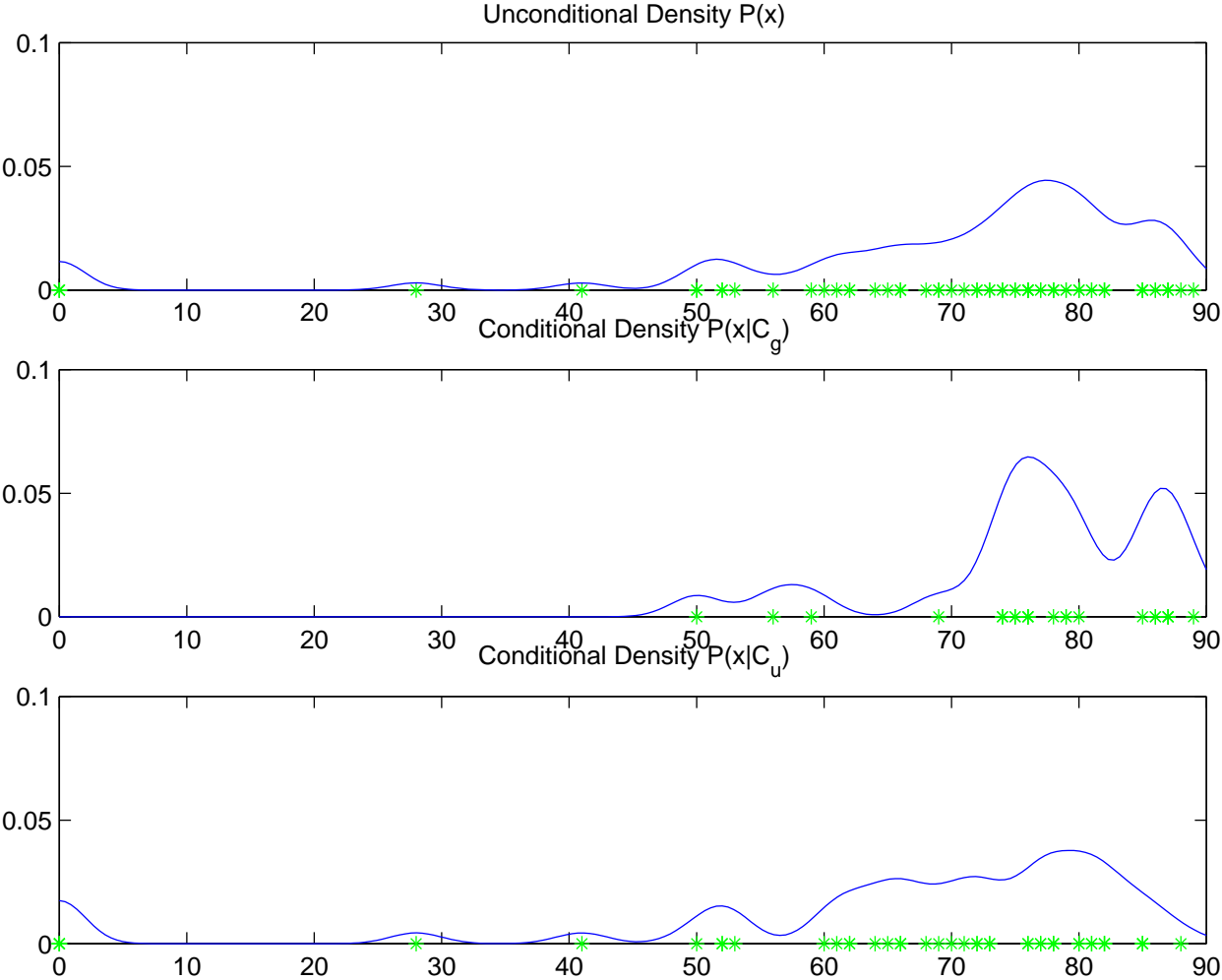
For classification, we need:

$$P(C_g|x) = \frac{P(x|C_g)P(C_g)}{P(x)}$$

$$P(x|C_g) \quad P(C_g) \quad P(x)$$

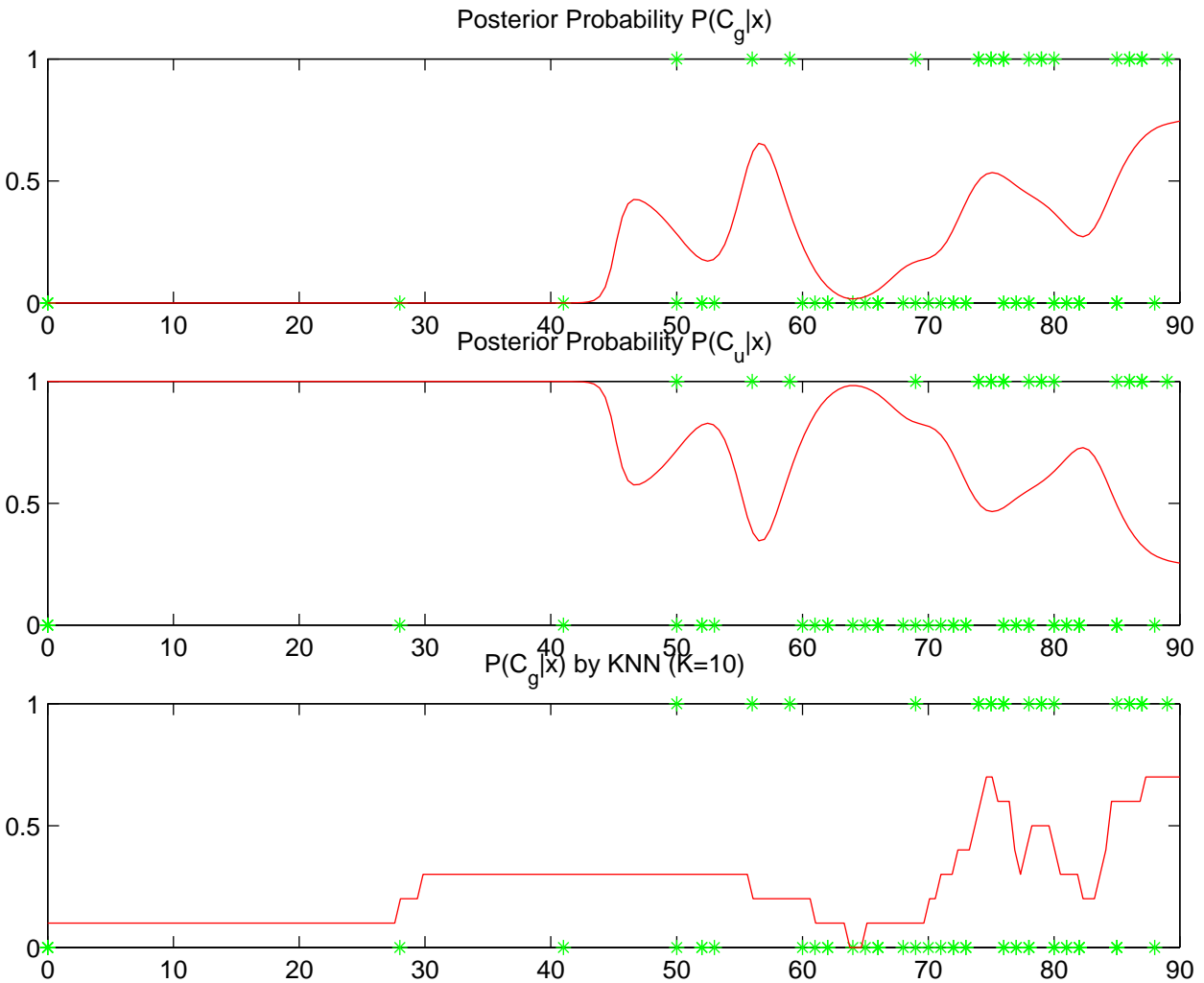
$$P(x) = P(x|C_g)P(C_g) + P(x|C_u)P(C_u)$$

# Conditional Densities



Note: Area under the curve is 1. **Not probabilities but Densities.**

# Parzen Windows Classification



Note: area under the curve is **not** 1.  
**Probabilities.**

# Locally Weighted Averaging

---

- another classification technique
- much simpler !
- memory based
- simply keep big instance database
- form locally weighted average at query point
- take that as an estimate for  $P(C_g|x)$

Locally Weighted Averaging:

- Assign a real value  $f(i) = 1$  to positives,  $f(i) = 0$  to negatives.
- Form a weighted average:

$$P(C_g|x) \approx \frac{\sum_{i=1}^N G(x, x_i, \sigma) f(i)}{\sum_{i=1}^N G(x, x_i, \sigma)}$$

# LWA and Density Estimation

---

**Very Cool Fact: Locally Weighted Averaging  
≡ Parzen Window Classification**

Remember Parzen Window Approximations:

$$P(x|C_g) \approx \frac{1}{N_g} \sum_{i \in G} G(x, x_i, \sigma)$$

$$P(x|C_u) \approx \frac{1}{N_u} \sum_{j \in U} G(x, x_j, \sigma)$$

$$P(C_g) \approx \frac{N_g}{N}$$

$$P(C_u) \approx \frac{N_u}{N}$$

## LWA and Density Estimation (2)

---

Now just apply Bayes law:

$$P(C_g|x) \approx \frac{\frac{1}{N_g} \sum_{i \in G} G(x, x_i, \sigma) \frac{N_g}{N}}{\frac{1}{N_g} \sum_{i \in G} G(x, x_i, \sigma) \frac{N_g}{N} + \frac{1}{N_u} \sum_{j \in U} G(x, x_j, \sigma) \frac{N_u}{N}}$$

$$P(C_g|x) \approx \frac{\sum_{i \in G} G(x, x_i, \sigma)}{\sum_{i \in G} G(x, x_i, \sigma) + \sum_{j \in U} G(x, x_j, \sigma)}$$

Assign a real value  $f(i) = 1$  to graduate students,  $f(i) = 0$  to undergrads. Then:

$$P(C_g|x) \approx \frac{\sum_{i=1}^N G(x, x_i, \sigma) f(i)}{\sum_{i=1}^N G(x, x_i, \sigma)}$$

# Properties of LWA

---

- very cool classification tool
- memory based
- lazy learning
- global vs. local
- can use any old kernel
- probabilistic interpretation
- can be slow
- curse of dimensionality

# K-Nearest Neighbors

---

- problem with KR: fixed kernel width
- sparse  $\rightarrow$  enlarge volume
- dense  $\rightarrow$  decrease volume
- idea: fix  $K$ , not  $V$
- $\rightarrow$  **K-nearest neighbors**

Remember:

$$P(x) \approx \frac{K}{NV}$$

$$P(x|C_g) \approx \frac{K_g}{N_g V}$$

$$P(C_g) \approx \frac{N_g}{N}$$

## K-Nearest Neighbors (2)

---

Now, for classification:

$$P(C_g|x) == \frac{P(x|C_g)P(C_g)}{P(x)}$$

$$P(C_g|x) \approx \frac{\frac{K_g}{VN_g} \frac{N_g}{N}}{\frac{K}{VN}}$$

$$P(C_g|x) \approx \frac{K_g}{K}$$

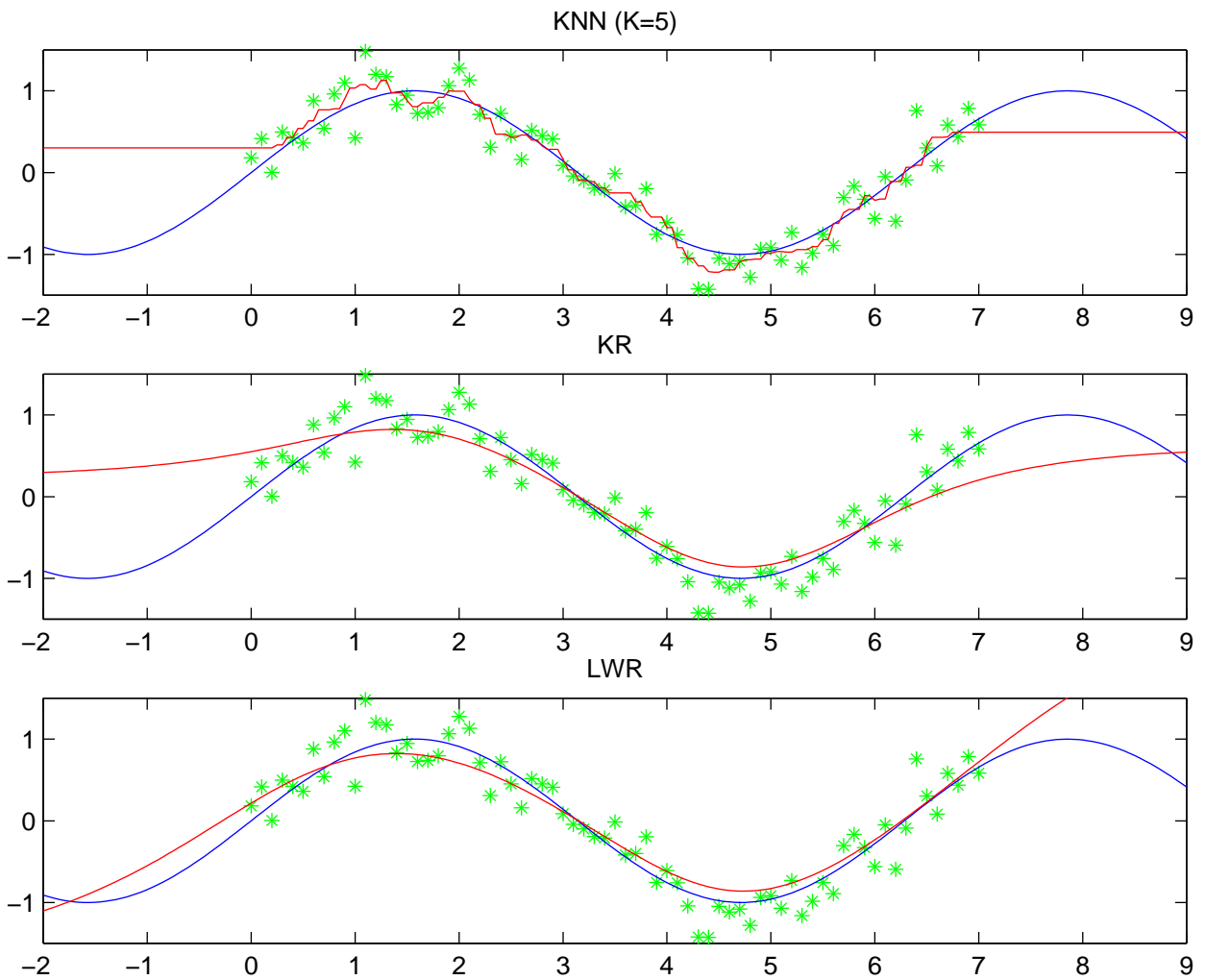
**Very Cool Fact:** Correct classification = majority vote of K nearest neighbors !

Note: smoothness assumption.

Boy/Girl counter-example.

# Locally Weighted Regression

Assume function is locally linear



# Properties of LWR

---

Advantages of LWR (L & Q):

- Training is almost free
- Incremental training trivial
- Does not forget old data
- Variable Resolution
- Accurately fit low-dim, complex functions
- Meaningful confidence intervals, gradients
- Cross-validation cheap

Neural Nets:

- Faster prediction (but: KD-trees)
- High-dimensional input data (but: PCA)
- Hard to make definite statements

# Radial Basis Function Networks

---

Extremely short intro (but very cool technique).

For Regression (Chapter 8):

- Approximate  $f$  by sum of Gaussians
- Two step training process:
  1. Choose Nr Kernels and Position
  2. Very fast training by matrix inversion

For Classification (Lowe in Arbib):

- Parametric density estimation
- Gaussian mixture model
- Idea: *pool* the kernels
- Use EM or such some thing
- Then simply apply Bayes law

# KD-trees

---

- Speed up Memory-Based Learning
- Recursively subdivide input space
- Store some computed values in nodes
- Make judicious approximations