Statistical Methods in Artificial Intelligence
CSE471 - Monsoon 2016 : Lecture 02

Avinash Sharma

CVIT, IIIT Hyderabad

Course Content

- Introduction
- Density Estimation
- Linear Classification
- Neural Networks
- Probability Densities
- Bayesian Classifiers
- Dimensionality Reduction
- Support Vector Machines
- Kernel Methods
- Clustering Techniques
- Decision Tree/Graphical Models

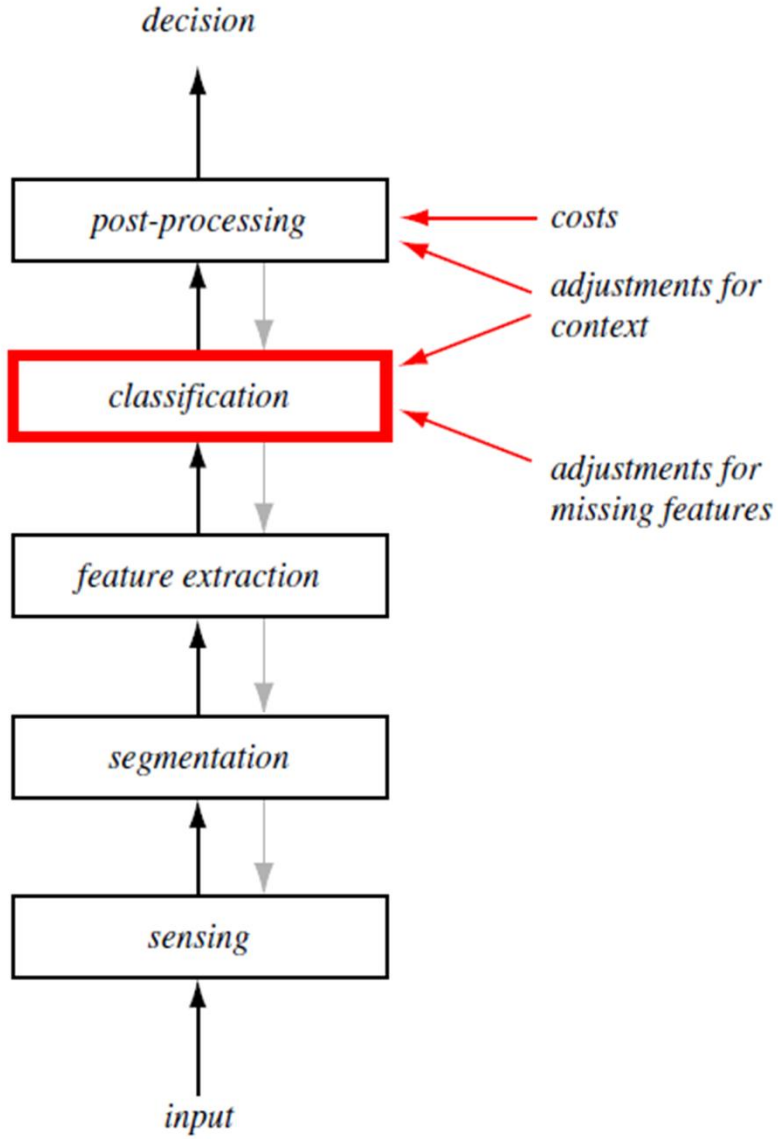
Reference Material

- Books
  - ❖ **“Pattern Classification” by Duda, Hart & Stork**
  - ❖ “The Elements of Statistical Learning” by Hastie, Tibshirani and Friedman
  - ❖ “Machine Learning : A probabilistic Perspective” by Kevin P. Murphy
- Pre-requisite
  - ❖ Basics of Linear Algebra, Calculus, Probability Theory and Statistics. Programming in Matlab and C/C++.
- Course Website http://moodle.iiit.ac.in
- Online Courses/Tutorials and Research Papers

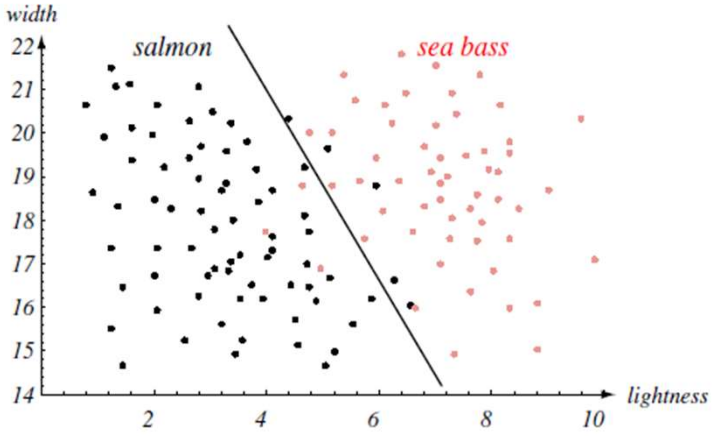
Lecture 02: Plan

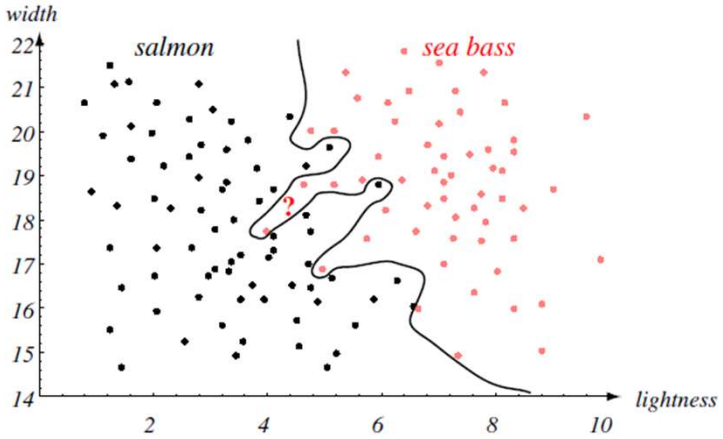
- Recap
- Probability Density Function
- Parzen Window Density Estimation
- KNN Density Estimation
- KNN Classifier
- Classifier Evaluation

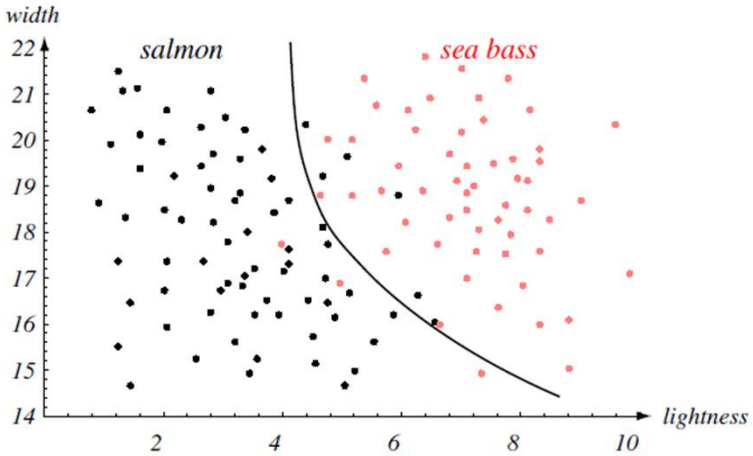
PR System Flow



Toy Examples Walkthrough





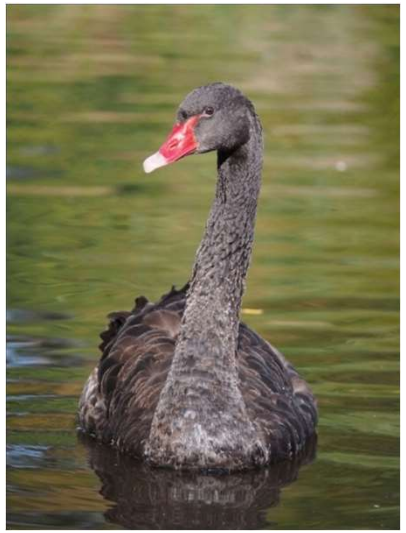


- **iid – independent and identically distributed**
- **Curse of Dimensionality**
- **Generalization v/s Overfitting**
- **Penalizing complex models**

Challenges

- Segmentation
  - Recovering the Signal from noise/clutter
- Feature Extraction
  - Invariance to Translation, Rotation and Scaling
  - Occlusion
  - Rate
  - Deformation
  - Selection of best features

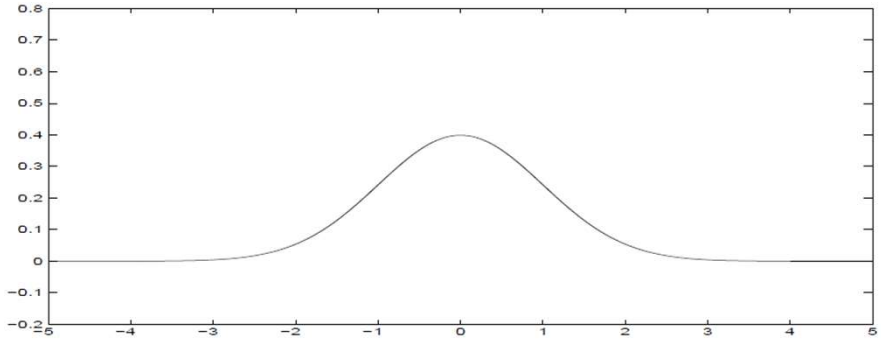
Challenges



- Classification
  - Noise “Any property of the sensed pattern which is not due to true underlying model but instead to randomness in the world or the sensor”.
- Post Processing
  - Error Rate
  - Risk
  - Context
  - Multiple Classifiers

Probability Density Function

- The mathematical definition of a continuous probability function $p(x)$ is
  - $P(a < x < b) = \int_a^b p(x)dx$
  - $p(x) \geq 0 \ \ \forall x$
  - $\int_{-\infty}^{\infty} p(x)dx = 1$
  - For example, Gaussian or Normal density function



$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- For a vector $\mathbf{x}$ and a non-negative $p(\mathbf{x})$, the probability that $\mathbf{x}$ is inside a region $\mathcal{R}$ is

$$P = \int_{\mathcal{R}} p(\mathbf{x})\, d\mathbf{x}$$

Density Estimation

- Given a set of n data samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we can estimate the density function $p(\mathbf{x})$.
- The probability P that a vector falls in a region $\mathcal{R}$ is given by $P = \int_{\mathcal{R}} p(\mathbf{x})\, d\mathbf{x}$.
- Assuming that $\mathcal{R}$ is so small $(h \to 0)$ that $p(\mathbf{x})$ does not vary much within it, we can write

$$P = \int_{\mathcal{R}} p(\mathbf{x})\, d\mathbf{x} \approx p(\mathbf{x}) \int_{\mathcal{R}} d\mathbf{x} = p(\mathbf{x})V$$

where V is the "volume" of region $\mathcal{R}$.

Density Estimation

- On the other hand, suppose that of n data samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ are independently drawn according to the probability density function $p(\mathbf{x})$, and there are k out of n samples falling within the region $\mathcal{R}$, we have: $P = k/n$

- Since, we saw that $P = p(\mathbf{x})V$, we can write

$$p(\mathbf{x}) = \frac{k/n}{V} \ .$$

Parzen Window Density Estimation

- Consider that $\mathcal{R}$ is a hypercube centered at $\mathbf{x}$.
- Let h be the length of the edge of the hypercube, then $V = h^2$ for a square, and $V = h^d$ for a hypercube.
- We can write a window function as :

$$\varphi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right) = \begin{cases} 1, & \frac{|x_{ik} - x_k|}{h} \leq 1/2,\ k = 1,2,\dots,d \\ 0, & \text{otherwise} \end{cases}$$

This indicates whether a data point $\mathbf{x}_i$ is inside the hypercube (centered at $\mathbf{x}$, and width h) or not.

Parzen Window Density Estimation

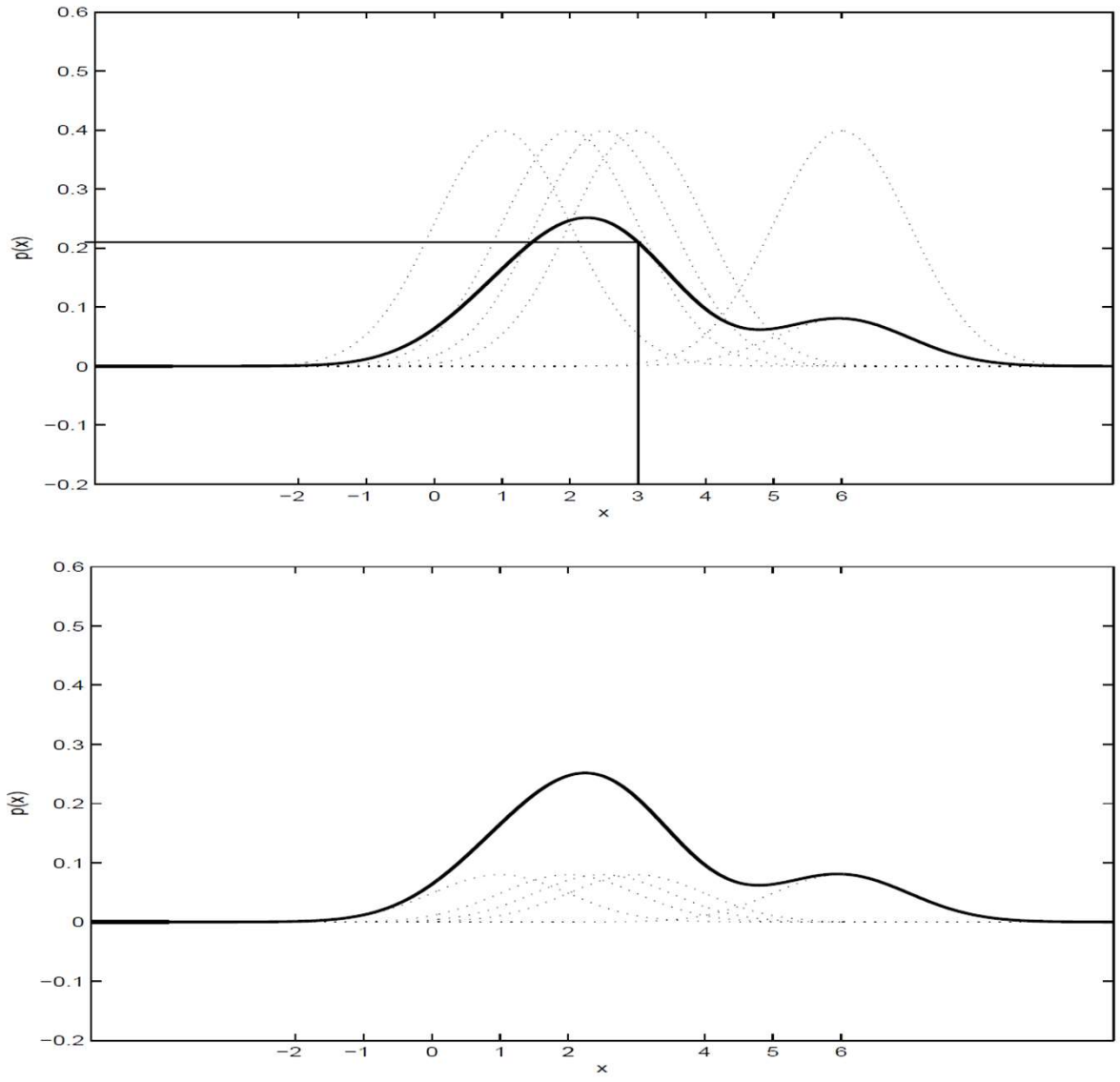
- The total number k samples falling within the region $\mathcal{R}$, out of n, is $k = \sum_{i=1}^{n} \varphi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)$

- The Parzen probability density estimation formula is:

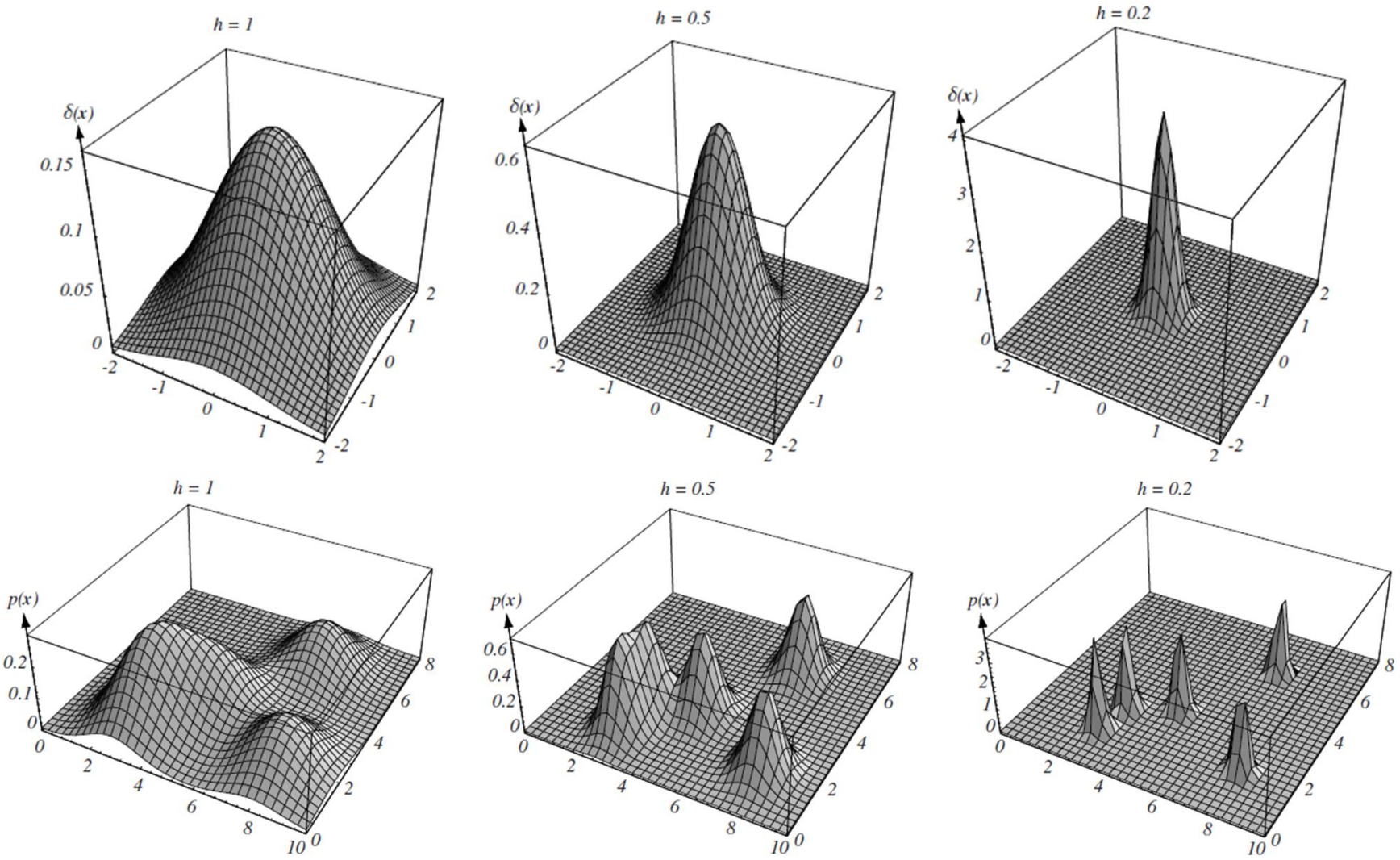
$$p(\mathbf{x}) = \frac{k/n}{V} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d}\varphi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)$$

- We can generalize the idea and allow the use of other window functions so as to yield other Parzen window density estimation methods.

Parzen Window Density Estimation

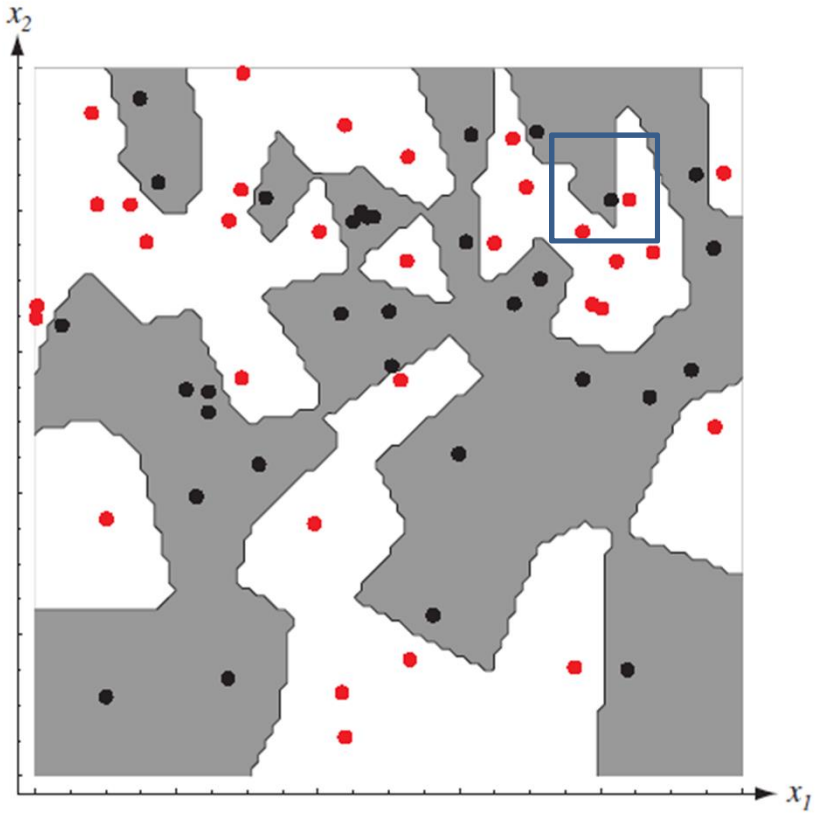


Parzen Window Density Estimation

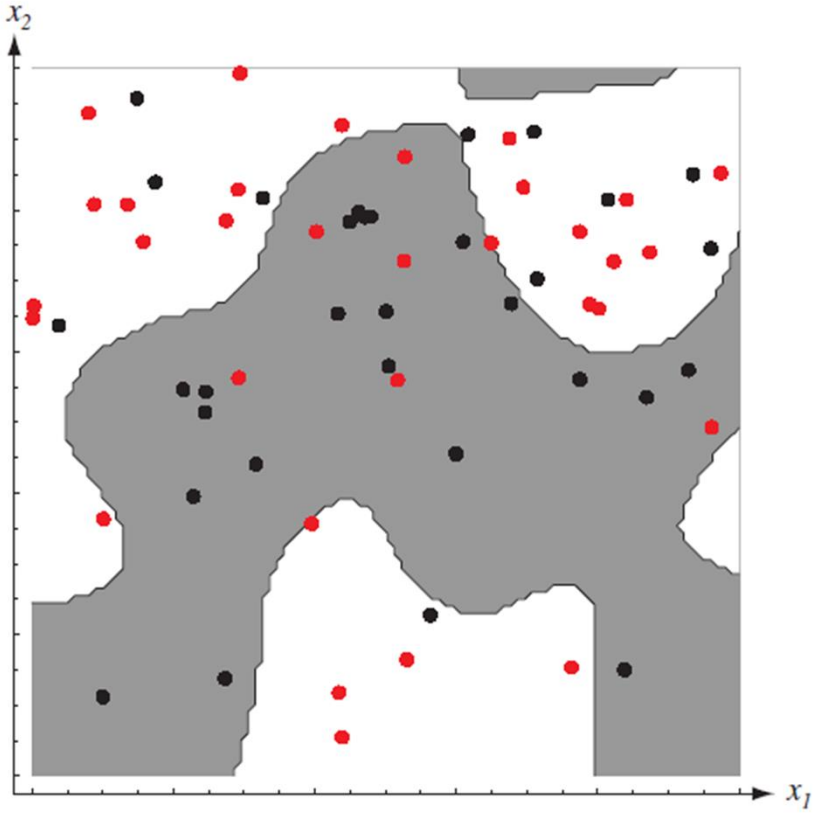


Parzen Window Dichotomizer

- Decision Boundaries for Two class classification



Small window size (*h*)



Large window size (*h*)

Convergence of Parzen Window DE

- Let for a $\mathcal{R}_n$

$$p_n(\mathbf{x}) = \frac{k_n/n}{V_n}$$

- Case 1 (finite n)

$$\lim_{V_n \to 0, k_n = 0} p_n(\mathbf{x}) = 0$$

- Case 2 (finite n)

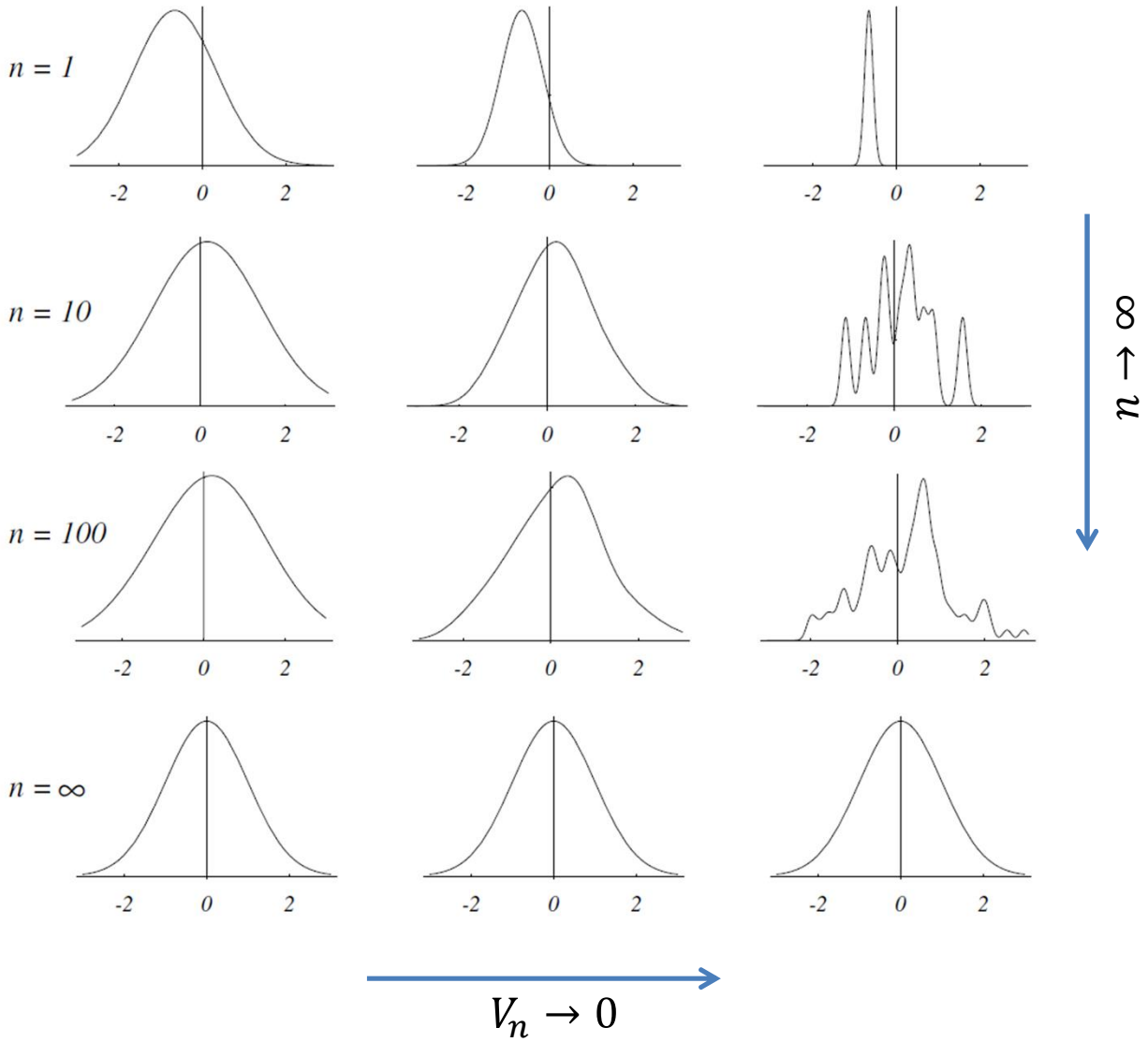
$$\lim_{V_n \to 0, k_n \neq 0} p_n(\mathbf{x}) = \infty$$

- Case 3 (desired)

$$\lim_{n \to \infty} V_n = 0$$

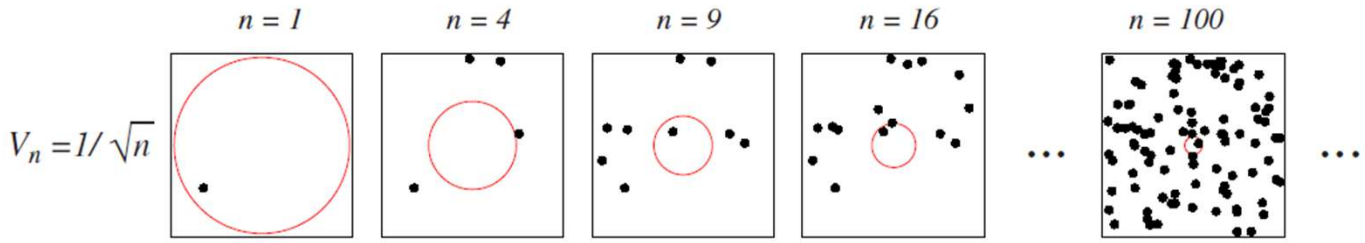
$$\lim_{n \to \infty} k_n = \infty$$

$$\lim_{n \to \infty} k_n/n = 0$$

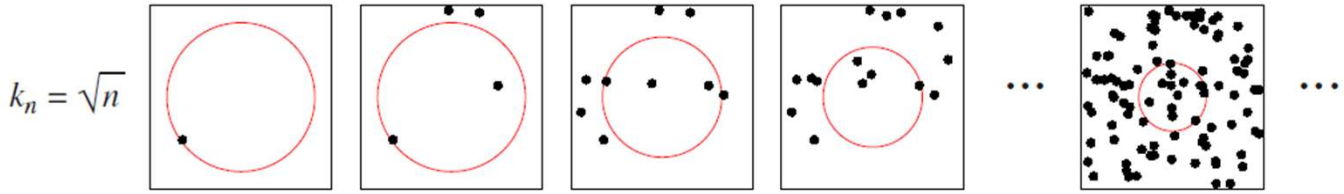


Parzen window v/s KNN Density Estimation

- Parzen Window: Reduce the volume of window as the number of samples increases.



- KNN: Reduce the volume of window by fixing the number of samples enclosed by it.



KNN Density Estimation

- Goal: a solution for the problem of the unknown "best" window function.
- Approach: Estimate density using real data points.
- Let the window volume be a function of the training data.
- Center a window about $\mathbf{x}$ and let it grow until it captures k_n samples: $k_n = f(n)$
- These samples inside this window are called the k_n nearest-neighbors of $\mathbf{x}$.

KNN Density Estimation

- Two possibilities can occur:
  - Density is high near $\mathbf{x}$; therefore the window will be small which provides good resolution.
  - Density is low; therefore the window will grow large and stop until higher density regions are reached.
- We can obtain a family of estimates by setting $k_n = \frac{\beta}{\sqrt{n}}$ and choosing different values for β.

Estimation of A Posteriori Probabilities

- Goal: Estimate $p(\omega_i|\mathbf{x})$ from a set of n labeled samples.
- Let's place a window of volume V around $\mathbf{x}$ and capture k samples.
- Let k_i samples amongst k turned out to be labeled with ω_i then: $p(\mathbf{x}, \omega_i) = \frac{(k_i/n)}{V}$
- A reasonable estimate for $p(\omega_i|\mathbf{x})$ is:

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x},\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x},\omega_j)} = \frac{k_i}{k}$$

Estimation of A Posteriori Probabilities

- $\frac{k_i}{k}$ is the fraction of the samples within the window that are labeled as ω_i .
- For minimum error rate, the most frequently represented category within the window is selected.
- If k is large and the window is sufficiently small, the performance will approach the best possible.

Nearest Neighbor (NN) Classifier

- Two data points (samples) from the same class should have similar features/attributes.
- Similarity in feature space should be aligned with similarity between among data points in the real-world scenarios.
- The easiest way to classify a data point in the test data is to find a very similar data point in the training data.

Nearest Neighbor (NN) Classifier

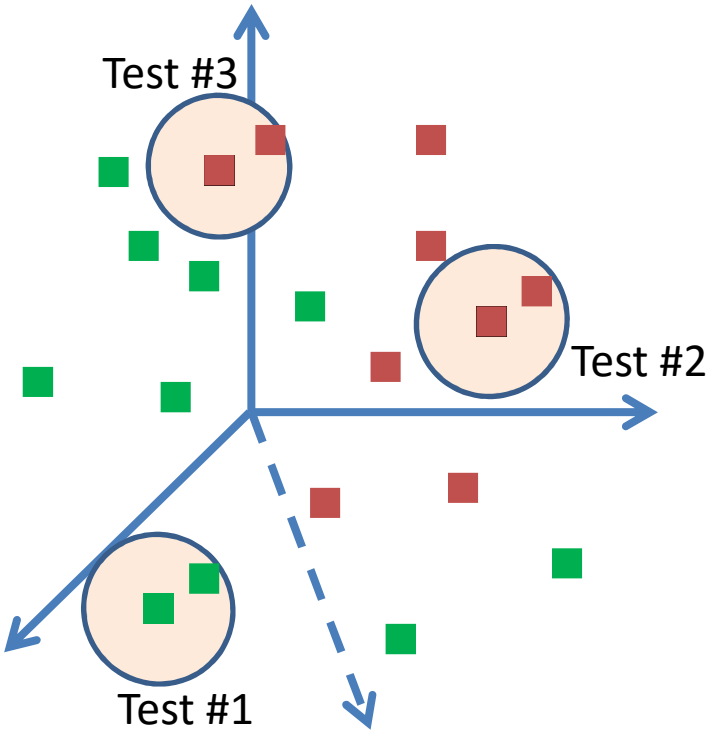
Eager v/s Lazy learning

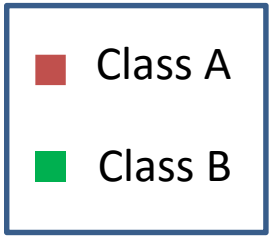
- **Eager learning**: Learn a classifier using training data before receiving a test data sample to classify.
- **Lazy learning:** Simply stores training data and waits until it is given a test data sample.

NN Classifier

- A lazy learning approach where each data sample is represented as point in a Euclidean space.
- Any new test data sample is assigned with the label of closest data point in the training data using Euclidean distance metric.

Nearest Neighbor (NN) Classifier



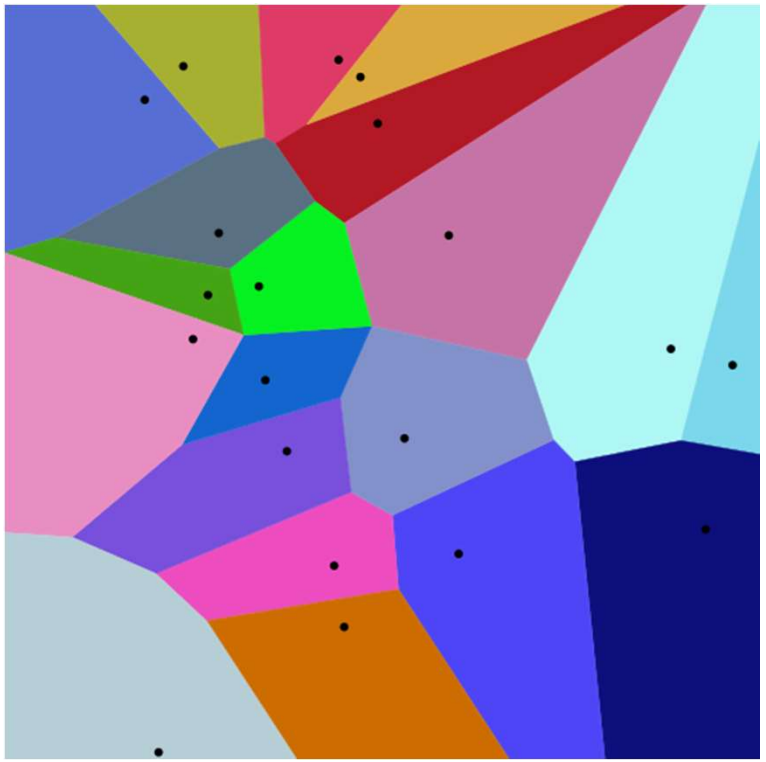


■ $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

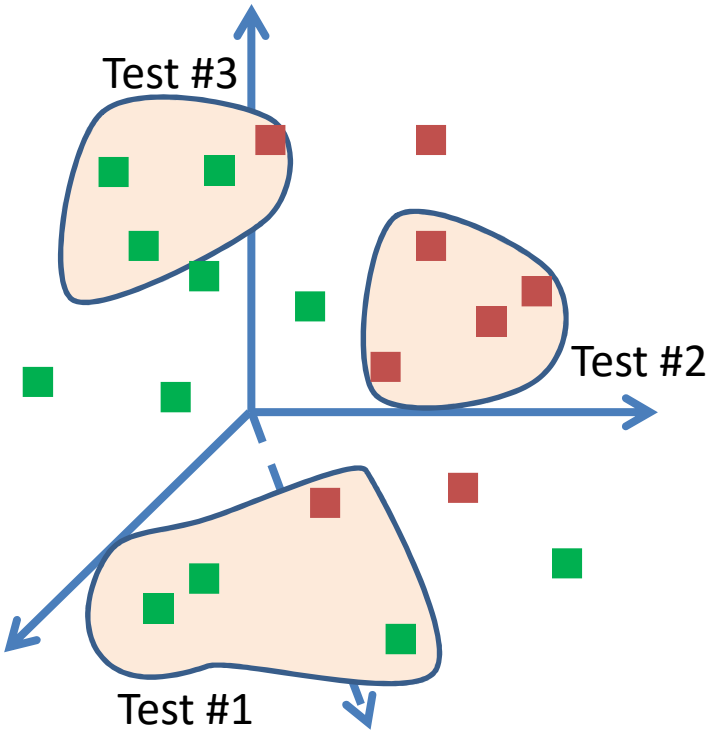
$$dist(\boldsymbol{x}^{train}, \boldsymbol{x}^{tes}) = \sqrt{\sum_{i=1}^{d} \left(x_i{}^{train} - x_i{}^{test}\right)^2}$$

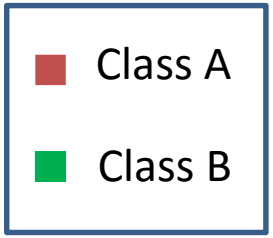
Nearest Neighbor (KNN) Classifier

- NN rule leads to partition of the Euclidean space into cells (Vornoi cells)



K-Nearest Neighbor (KNN) Classifier





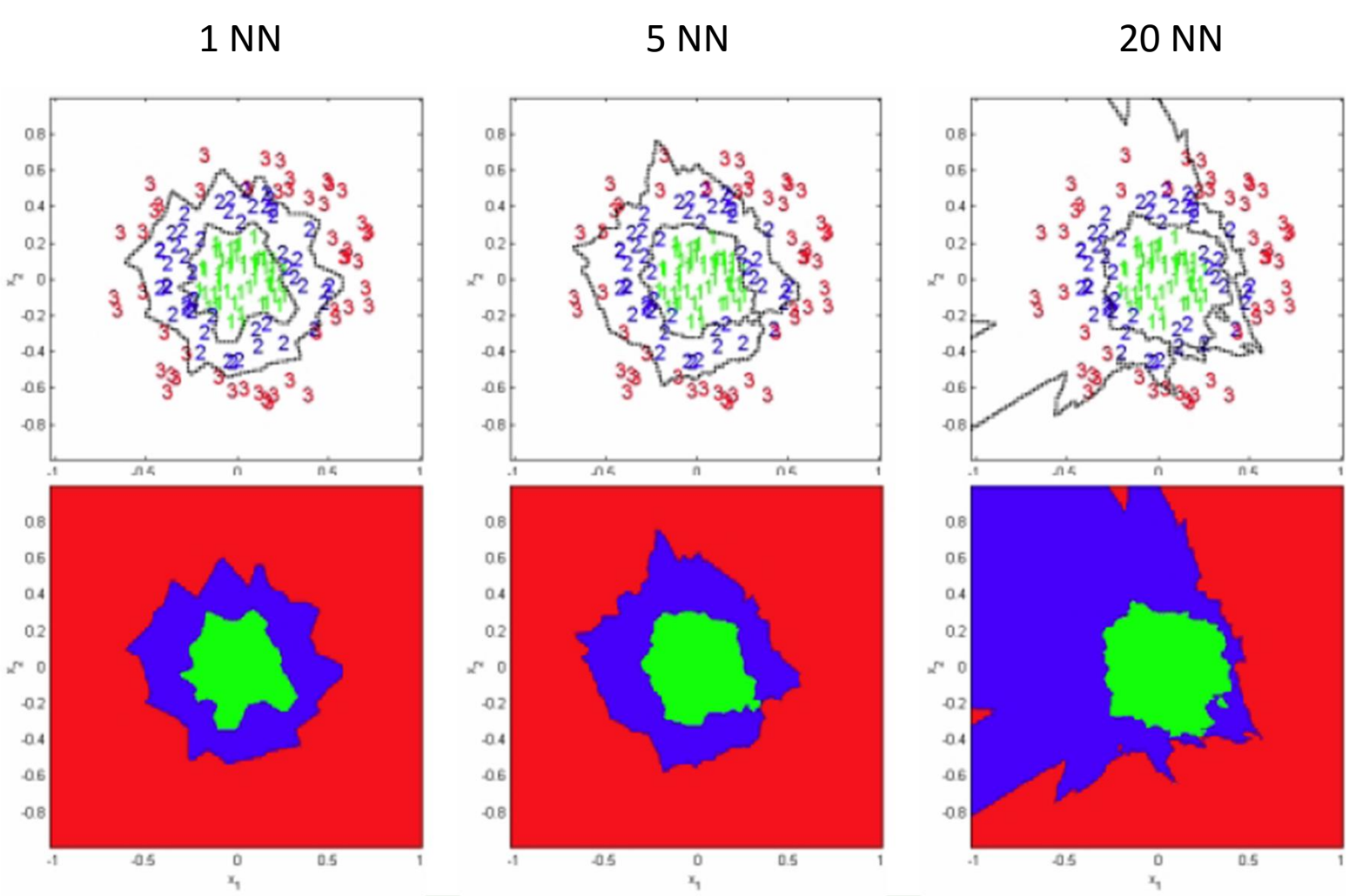
■ $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

$$dist(\boldsymbol{x}^{train}, \boldsymbol{x}^{test}) = \sqrt{\sum_{i=1}^{d} \left(x_i{}^{train} - x_i{}^{test}\right)^2}$$

Practical Aspects

- K should be chose empirically and preferably odd to avoid tie situation.
- KNN can have both *discrete-value* and *continuous-value* **target functions**.
- Weighted contributions from different neighbors can be used to compute final label.
- Distance based weighting can be used for giving higher importance to closer data points.
- Performance of NN classification typically degrades when data is high-dimensional.
- This can be avoided by assigning feature weights inside Euclidean distance computation.

Effect of K on decision boundaries



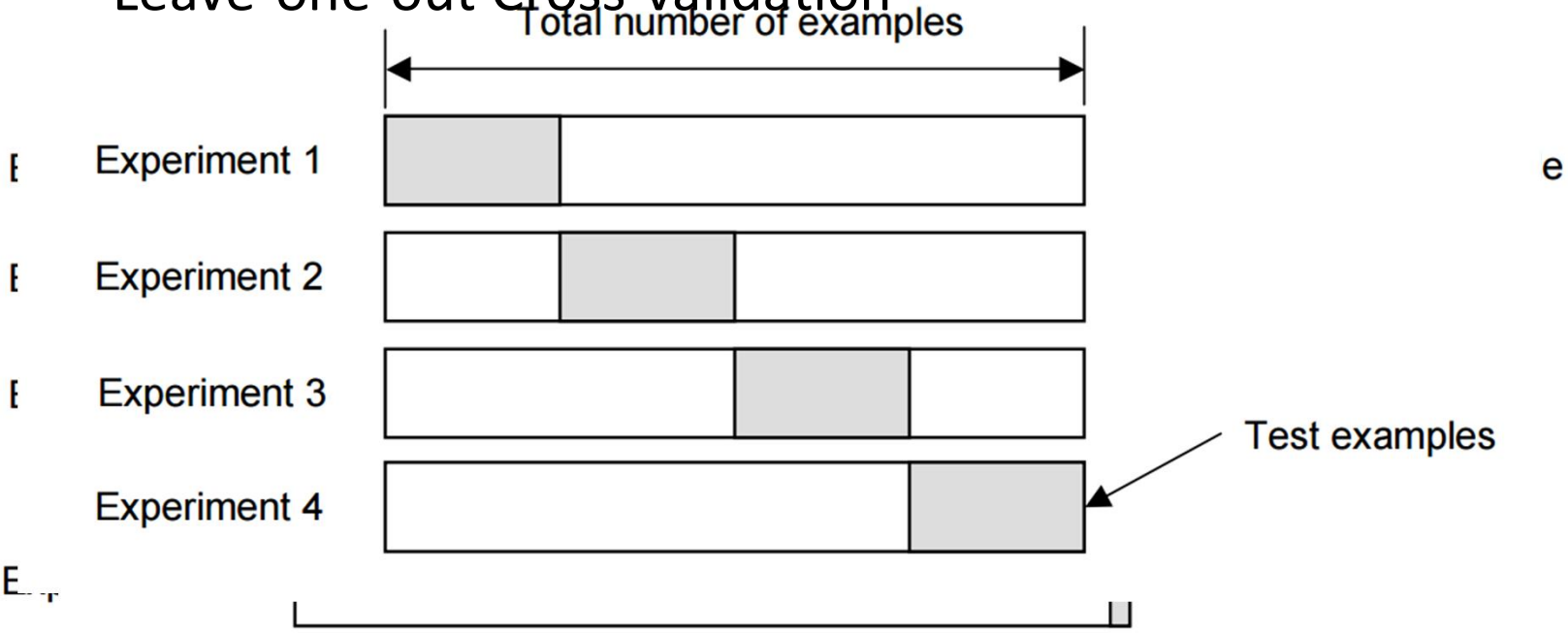
KNN Classifier

- Advantages:
  - Learn complex target functions
  - Training is very fast
  - Zero loss of information

- Disadvantages:
  - Classification cost for new instances can be very high
  - Major computation takes place at classification time

Classifier Evaluation

- *Cross Validation* is an important means of evaluating classifiers. Types of cross validation techniques are:
  - Random Subsampling
  - K-fold Cross-Validation
  - Leave-one-out Cross-validation



Assignment 0

- Iris data or other standard dataset
- KNN classifier
- Matlab/C++ pipeline
- K-fold cross-validation