Machine Learning

Non-parametric Algorithms: k-NN Classifier and Parzen Window

Indian Institute of Information Technology
Sri City, Chittoor

This week's Agenda

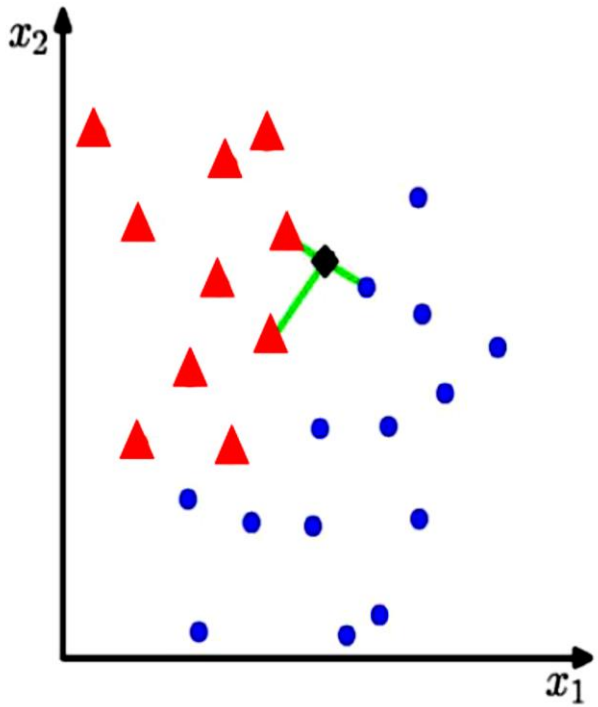
- Recap to KNN
- KNN Properties and Training
- Evaluation Metrics for a Classification model
- Advantages and Disadvantages of KNN
- r-fold cross validation for KNN
- Improving KNN
- K-NN from Computation Perspective
- Parzen Window
- Defining Rn
- Two different approaches - fixed volume vs. fixed number of samples in a variable volume
- Example 3D hypercunbde
- The window function and estimation
- Critical parameters of the Parzen-window technique: window width and kernel
- Selecting Window
- Selecting Kernel

K-Nearest Neighbour (KNN) Classifier

Algorithm

- For each test point, x, to be classified, find the K-nearest samples in the training data.
- Classify the point, x, according to the majority vote of their class labels.
- applicable to multi-class case

e.g. K = 3

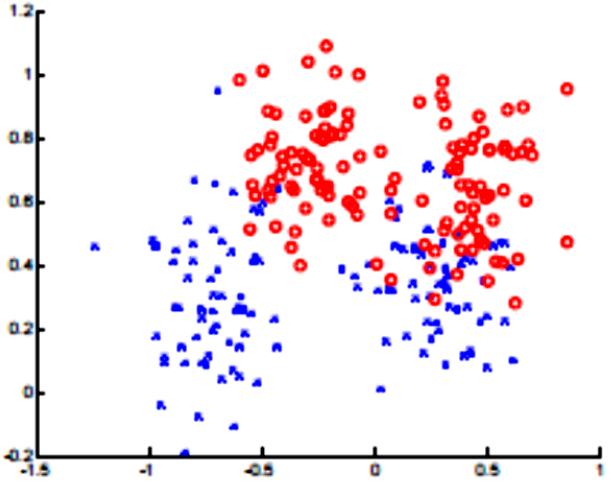


A sampling assumption: Train and Test Data

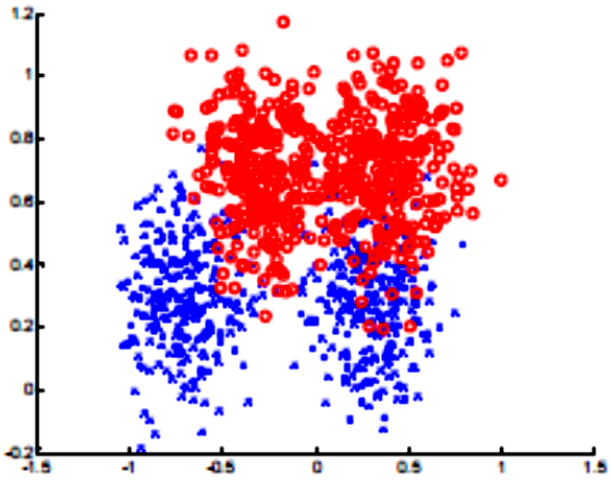
- Assume that the training examples are drawn independently from the set of all possible examples.

- This makes it very unlikely that a strong regularity in the training data will be absent in the test data.

- Measure classification error as $= \frac{1}{N} \sum_{i=1}^{N} \underbrace{[y_i \neq f(x_i)]}_{\text{loss function}}$ The "risk"



Training data



Testing data

KNN Properties and Training:

As K increases:

- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation:

- Split training data into training and validation
- Hold out validation data and measure error on this

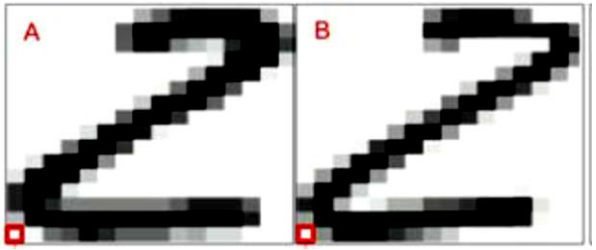
KNN Properties and Training:

- MNIST data set
- Distance = raw pixel distance between images
- 60K training examples
- 10K testing examples
- K-NN gives 5% classification error

Train Set

$$D(\mathbf{A},\mathbf{B}) = \sum_{ij} \sqrt{\left(a_{ij} - b_{ij}\right)^2}$$

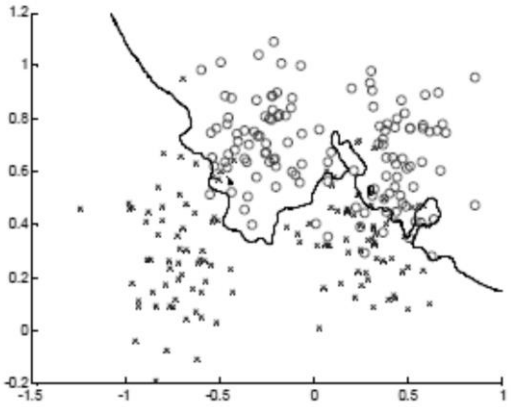
Distance Metric

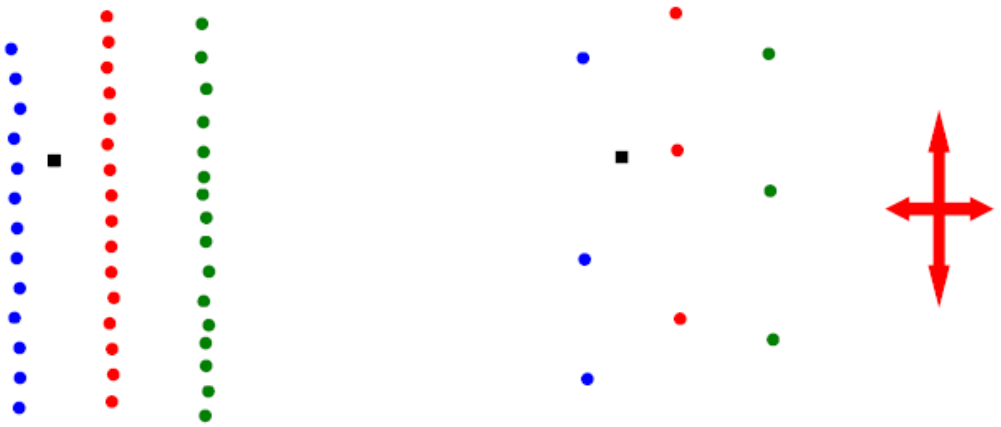


Test Samples

Advantages:

- K-NN is a simple but effective classification procedure.
- Applies to multi-class classification.
- Decision surfaces are non-linear.
- Quality of predictions automatically improves with more training data.
- Only a single parameter, K; easily tuned by cross-validation.





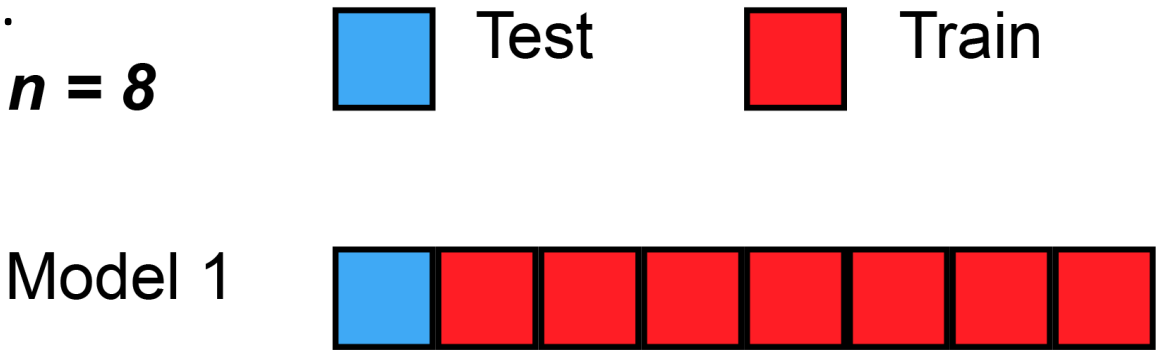
Disadvantages

- What does nearest mean? Need to specify a distance metric.
- Computational cost: must store and search through the entire training set at test time.
- Can alleviate this problem by thinning, and use of efficient data structures like KD trees.

r-fold Cross Validation

- **Cross Validation Method**: Cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (Validation set).

- To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance.



Data Sampling Methods

r-fold Cross validation: This procedure has a single parameter called “r” that refers to the number of groups that a given data sample is to be split into.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into r groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

r-fold Cross validation:

r-fold Cross validation for K-NN

- ***r-fold cross validation:***
  1. Partition the training set into r blocks. Let these are $D_1, D_2, \ldots, D_r$.
  2. For i = 1 to r do
     - I. Consider $D - D_i$ as the training set and D_i as the validation set.
     - II. For a range of k values (say from 1 to m) find the error rates on the validation set.
     - III. Let these error rates are $e_{i1}, e_{i2}, \ldots, e_{im}$
  3. Take e_i = mean of $\{e_{1i}, e_{2i}, \ldots., e_{ri}\}$, for i = 1 to m.
  4. k value = $\underset{j}{\text{argmin}} \{e_1, e_2, \ldots.,e_j, \ldots, e_m\}$

r-fold Cross validation for K-NN

- One should not use *the test set* to decide the value of k.
- Test set should be used only after fixing k, to get the final *error-rate* for the classifier.
- Cross validation is only to fix the value of parameters like k . So the error rates on validation sets should be called *validation error rates*.

Improving KNN

- k-NNC gives equal importance to the first NN and to the last NN.
- *S.A. Dudani (1976)* has given a method where we give *weights* to the NNs.
- Voting is done according to these weights.
- Let the distances (with given pattern) of k NNs be an ordered set = $\{ d_1, d_2, \ldots, d_k\}$
- For i th NN the weight is, $w_i = (d_k-d_i)/(d_k-d_1)$
- Use these weights as vote values and classify accordingly.
- This is called *modified k-NNC* or *weighted k-NNC*, and is found to improve the performance in almost all cases.

Improving KNN

- Another promising improvement is to regenerate the training set, so that the training patterns belonging to different classes are separated well.
- *Hamamoto(1997)* proposed the following:
- For each training pattern *y* do:
  1. Find *r* NNs of *y* in the training set that belongs to the same class as *y*.
  2. Find the mean of these *r* NNs. Let this is y_r
  3. Replace *y* by y_r

K-NN from Computation Perspective:

- Let *n* be the number of training patterns.
- Let *k* be a small constant when compared with *n*
- The time and space complexity of k-NNC are both equal to *O(n).*
- To reduce the computational burden of k-NNC is another important direction of research.
  - Prototype selection.
  - Not all training patterns are important for k-NNC, so remove those which are unimportant.

Parzen window

- The Parzen-window method (also known as Parzen-Rosenblatt window method) is a widely used non-parametric approach to estimate a probability density function p(x) for a specific point p(x) from a sample $p(x_n)$.

- It doesn't require any knowledge or assumption about the underlying distribution.

- A popular application of the Parzen-window technique is to estimate the class-conditional densities (or also often called 'likelihoods').

- Likelihoods, $p(x \mid \omega_i)$ in a supervised pattern classification problem from the training dataset (where p(x) refers to a multi-dimensional sample that belongs to particular class ω_i)).

Where would this method be useful?

- Imagine that we are about to design a Bayes classifier for solving a statistical pattern classification task using Bayes' rule:

$$P(\omega_i \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid w_i) \cdot P(\omega_i)}{p(\boldsymbol{x})}$$

$$\Rightarrow \textit{posterior probability} = \frac{\textit{likelihood} \cdot \textit{prior probability}}{\textit{evidence}}$$

- If the parameters of the the class-conditional densities (also called likelihoods) are known, it is pretty easy to design the classifier.
- Imagine we are about to design a classifier for a pattern classification task where the parameters of the underlying sample distribution are not known.
- Therefore, we wouldn't need the knowledge about the whole range of the distribution; it would be sufficient to know the probability of the particular point, which we want to classify, in order to make the decision.

Parzen Window

- In parzen window we are going to see how we can estimate this probability from the training sample.

- However, the only problem of this approach would be that we would seldom have exact values - if we consider the histogram of the frequencies for a arbitrary training dataset.
- Therefore, we define a certain region (i.e., the Parzen-window) around the particular value to make the estimate.

[1] *Parzen, Emanuel*. On Estimation of a Probability Density Function and Mode. The Annals of Mathematical Statistics 33 (1962), no. 3, 1065 1076.
[2] *Rosenblatt, Murray*. Remarks on Some Nonparametric Estimates of a Density Function. The Annals of Mathematical Statistics 27 (1956), no. 3, 832–837.

Defining the Region Rn

- The basis of this approach is to count how many samples fall within a specified region Rn (or "window" if you will). Our intuition tells us, that (based on the observation), the probability that one sample falls into this region is:

$$p(x) = \frac{\#\ of\ samples\ in\ R}{total\ samples}$$

- To tackle this problem from a more mathematical standpoint to estimate "the probability of observing k points out of n in a Region R " we consider **a binomial distribution:**

$$p_k = \begin{bmatrix} n \\ k \end{bmatrix} \cdot p^k \cdot (1-p)^{n-k}$$

- Make the assumption that in a binomial distribution, the probability peaks sharply at the mean

Defining the Region Rn

- The basis of this approach is to count how many samples fall within a specified region Rn (or "window" if you will). Our intuition tells us, that (based on the observation), the probability that one sample falls into this region is:

$$p(x) = \frac{\#\ of\ samples\ in\ R}{total\ samples}$$

- To tackle this problem from a more mathematical standpoint to estimate "the probability of observing k points out of n in a Region R " we consider **a binomial distribution:**

$$p_k = \begin{bmatrix} n \\ k \end{bmatrix} \cdot p^k \cdot (1-p)^{n-k}$$

- Make the assumption that in a binomial distribution, the probability peaks sharply at the mean

$$E[k] = n \cdot p \quad \sim \quad k = n \cdot p$$

Defining the Region Rn

- And if we think of the probability as a continuous variable, we know that it is defined as:

$$p(\boldsymbol{x}) = \int_R dx = p(\boldsymbol{x}) \cdot V,$$

where V is the volume of the region R, and if we rearrange those terms, so that we arrive at the following equation, which we will use later:

$$\frac{k}{n} = p(\boldsymbol{x}) \cdot V$$

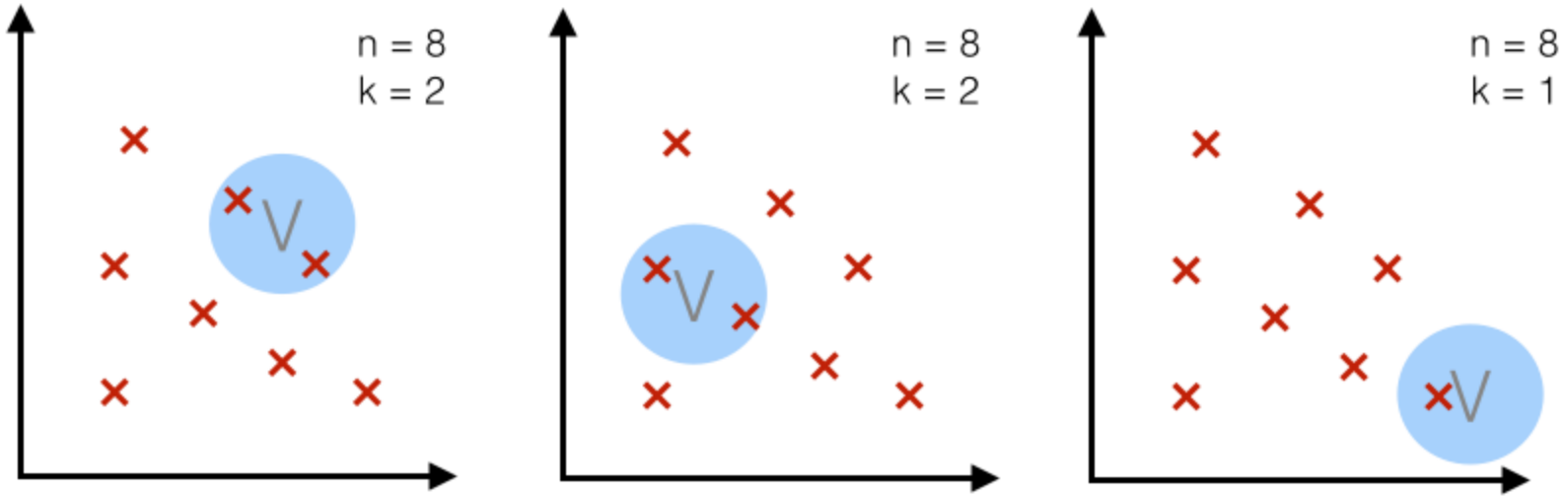
$$\Rightarrow p(\boldsymbol{x}) = \frac{k/n}{V}$$

- This simple equation above (i.e, the "probability estimate") lets us calculate the probability density of a point x by counting how many points k fall in a defined region (or volume).

Two different approaches - fixed volume vs. fixed number of samples in a variable volume

Case 1 - fixed volume:

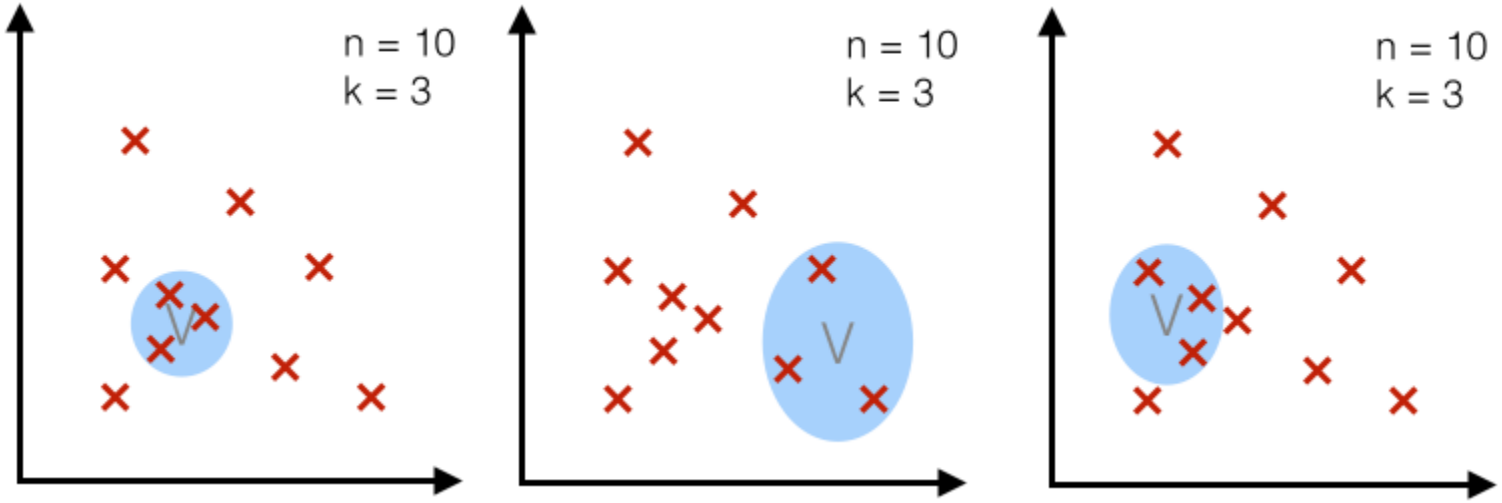
- For a particular number n (= number of total points), we use volume V of a fixed size and observe how many points k fall into the region.



Two different approaches - fixed volume vs. fixed number of samples in a variable volume

Case 2 - fixed *k*:

- For a particular number n (= number of total points), we use a fixed number k (number of points that fall inside the region or volume) and adjust the volume accordingly..



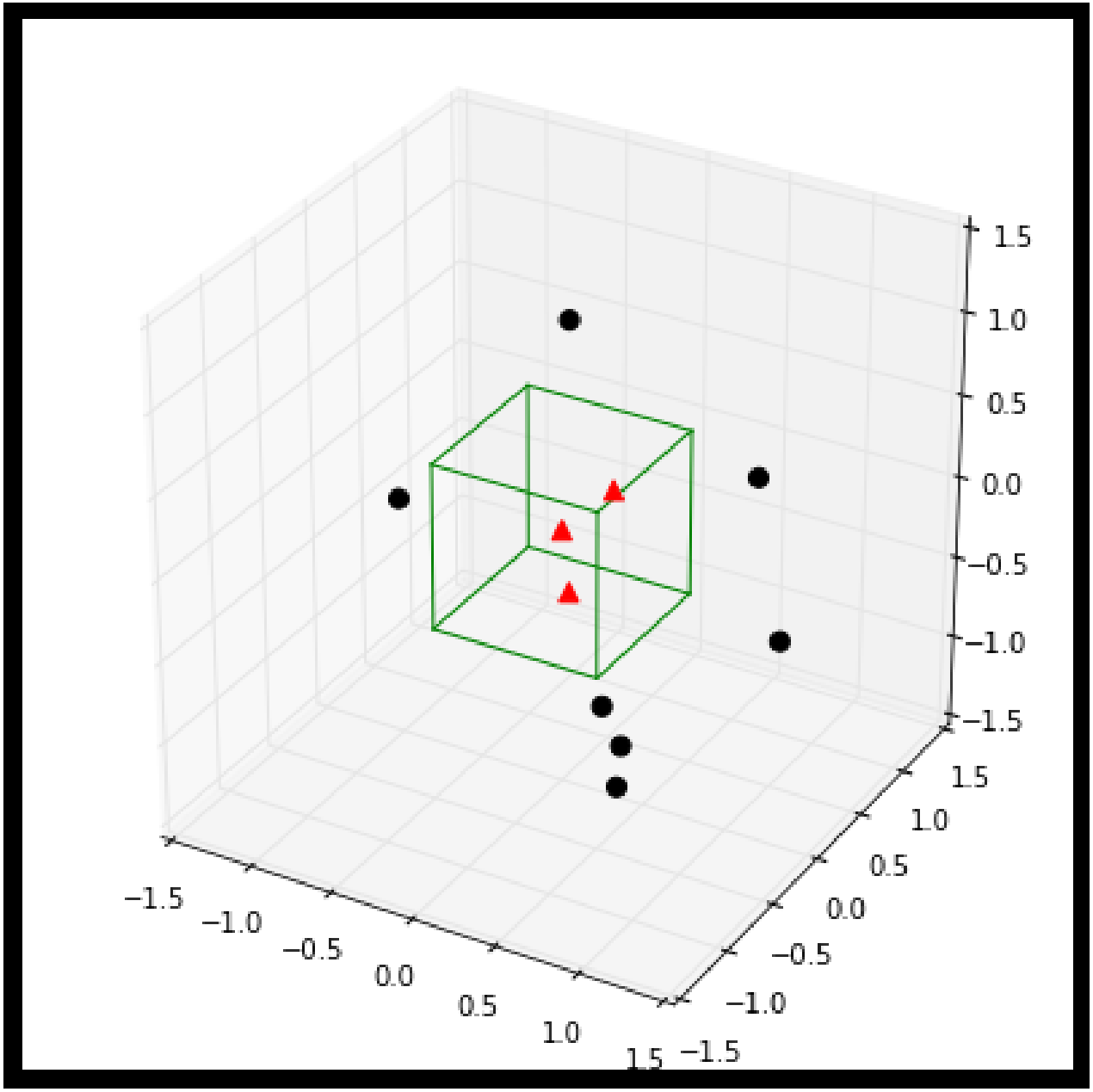
Example 3D-hypercubes

To illustrate this with an example and a set of equations, let us assume this region R_n is a hypercube.

The volume of this hypercube is defined by $V_n = h_n^d$, where h_n is the length of the hypercube, and d is the number of dimensions.

For an 2D-hypercube with length 1, for example, this would be $V_1 = 12$ and for a 3D hypercube $V_1 = 13$, respectively.

Example: A typical 3-dimensional unit hypercube ($h_1 = 1$) representing the region R_1, and 10 sample points, where 3 of them lie within the hypercube (red triangles), and the other 7 outside (black dots).


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

The window function

- Once we visualized the region R1 like above, it is easy and intuitive to count how many samples fall within this region, and how many lie outside.
- To approach this problem more mathematically, we would use the following equation to count the samples k_n within this hypercube, where φ is our so-called window function.

$$\phi(\boldsymbol{u}) = \begin{cases} 1 & |u_j| \leq 1/2\,; \qquad j = 1, \ldots, d \\ 0 & otherwise \end{cases}$$

for a hypercube of unit length 1 centered at the coordinate system's origin.

- If we extend on this concept, we can define a more general equation that applies to hypercubes of any length h_n that are centered at *x*:

$$k_n = \sum_{i=1}^{n} \phi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_n} \right)$$

$$\text{where } \boldsymbol{u} = \left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_n} \right)$$

Parzen-window estimation

- we can now formulate the Parzen-window estimation with a hypercube kernel as follows:

$$p_n(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} \phi \left[\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n} \right]$$

where

$$h^d = V_n \quad \text{and} \quad \phi \left[\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n} \right] = k$$

- And applying this to our unit-hypercube example above, for which 3 out of 10 samples fall inside the hypercube (into region *R*), we can calculate the probability *p(**x**)* that ***x*** samples fall within region *R* as follows:

$$\boldsymbol{x} = \frac{k/n}{h^d} = \frac{3/10}{1^3} = \frac{3}{10} = 0.3$$

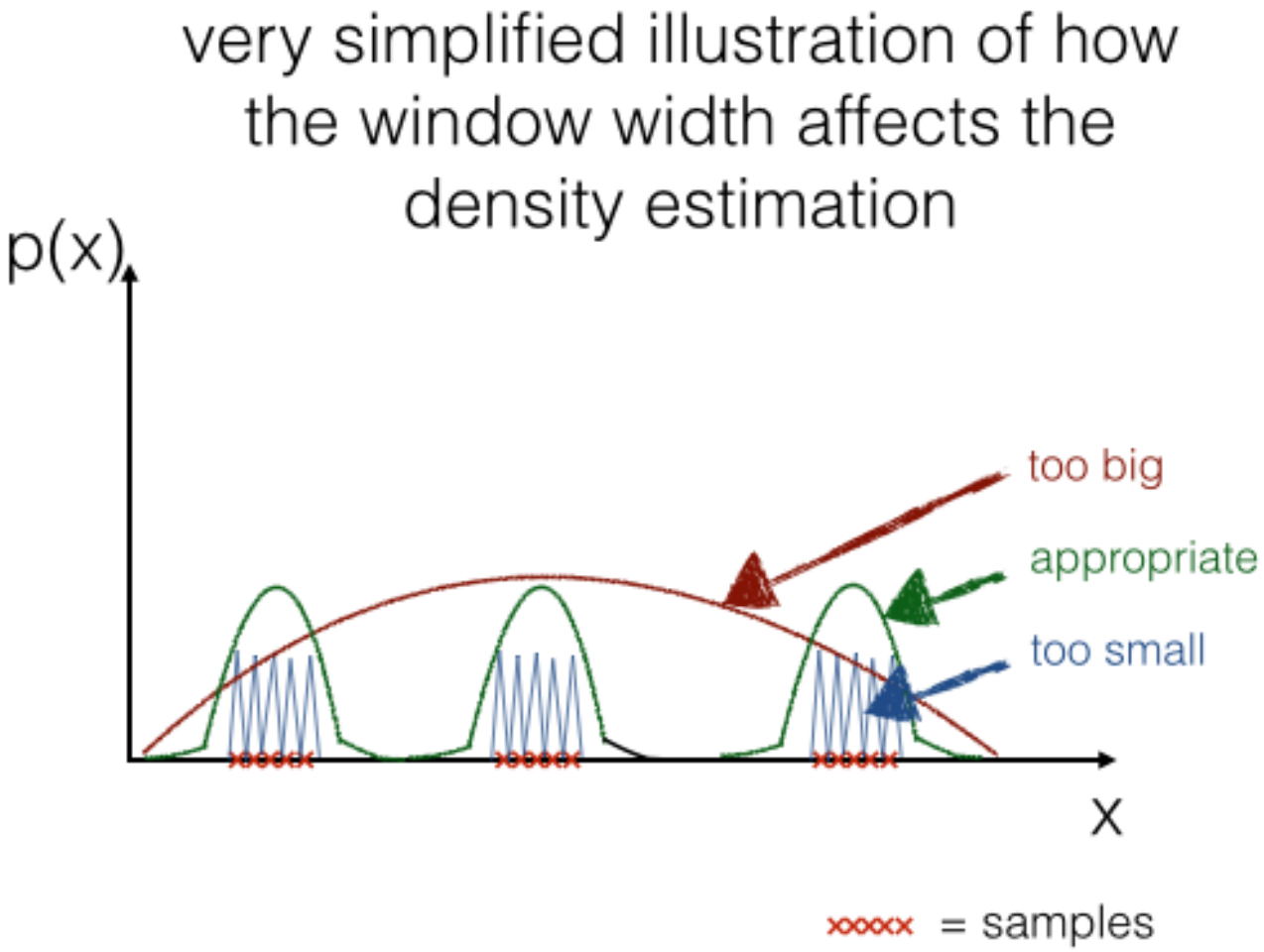
Critical parameters of the Parzen-window technique: window width and kernel

The two critical parameters in the Parzen-window techniques are

1) the window width: Which window size should we choose (i.e., what should be the side length h of our hypercube)?

2) the kernel: Most commonly, either a hypercube or a Gaussian kernel is used for the window function. But how do we know which is better?

Selecting the window width



If we would choose a window width that is “too small”, this would result in local spikes, and a window width that is “too big” would average over the whole distribution.

Selecting the kernel

- Hypercube or a Gaussian kernel?
  - It really depends on the training sample.

- Intuitively, it would make sense to use a Gaussian kernel for a data set that follows a Gaussian distribution.

  - *But remember, the whole purpose of the Parzen-window estimation is to estimate densities of a* ***unknown distribution!***

- ***Gaussian kernel instead of the hypercube:***
  - *simply swap the terms of the window function, which we defined above for the hypercube.*

Gaussian Kernel

- **The Parzen-window Gaussian kernel:**

$$\frac{1}{(\sqrt{2\pi})^d h_n^d} exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)^2\right]$$

- **The Parzen-window estimation would then look like this:**

$$p_n(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\phi\left[\frac{1}{(\sqrt{2\pi})^d h_n^d} exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)^2\right]\right]$$

Thank You: Question?