Ch8. Instance-Based Learning 8.1, 8.2

S. Visa

Instance-Based Learning

- = stores all ex. or some representatives
- Classification of a new instance x → based on its relation with other observed ex. ➔ lazy methods

- Ex.
  - Prototype methods (Min.Dist.Classif., k-Means clustering, k-Prototypes)
  - k-Nearest Neighbors
  - Locally weighted regression
  - Radial basis function

- Simple and model free

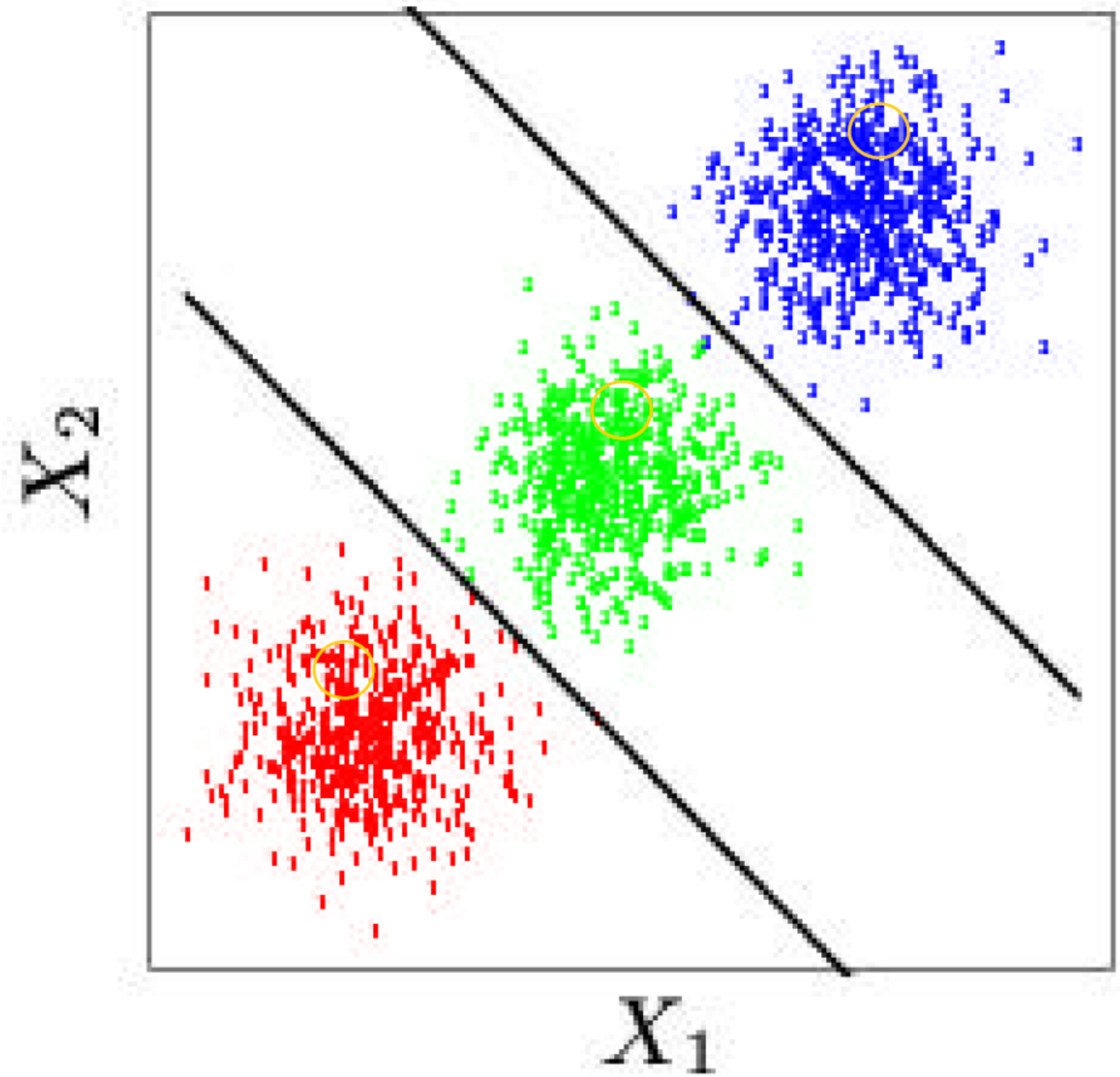
- Very effective for classification in real world data

- Not useful for understanding the nature of the relationship between the features and class outcome

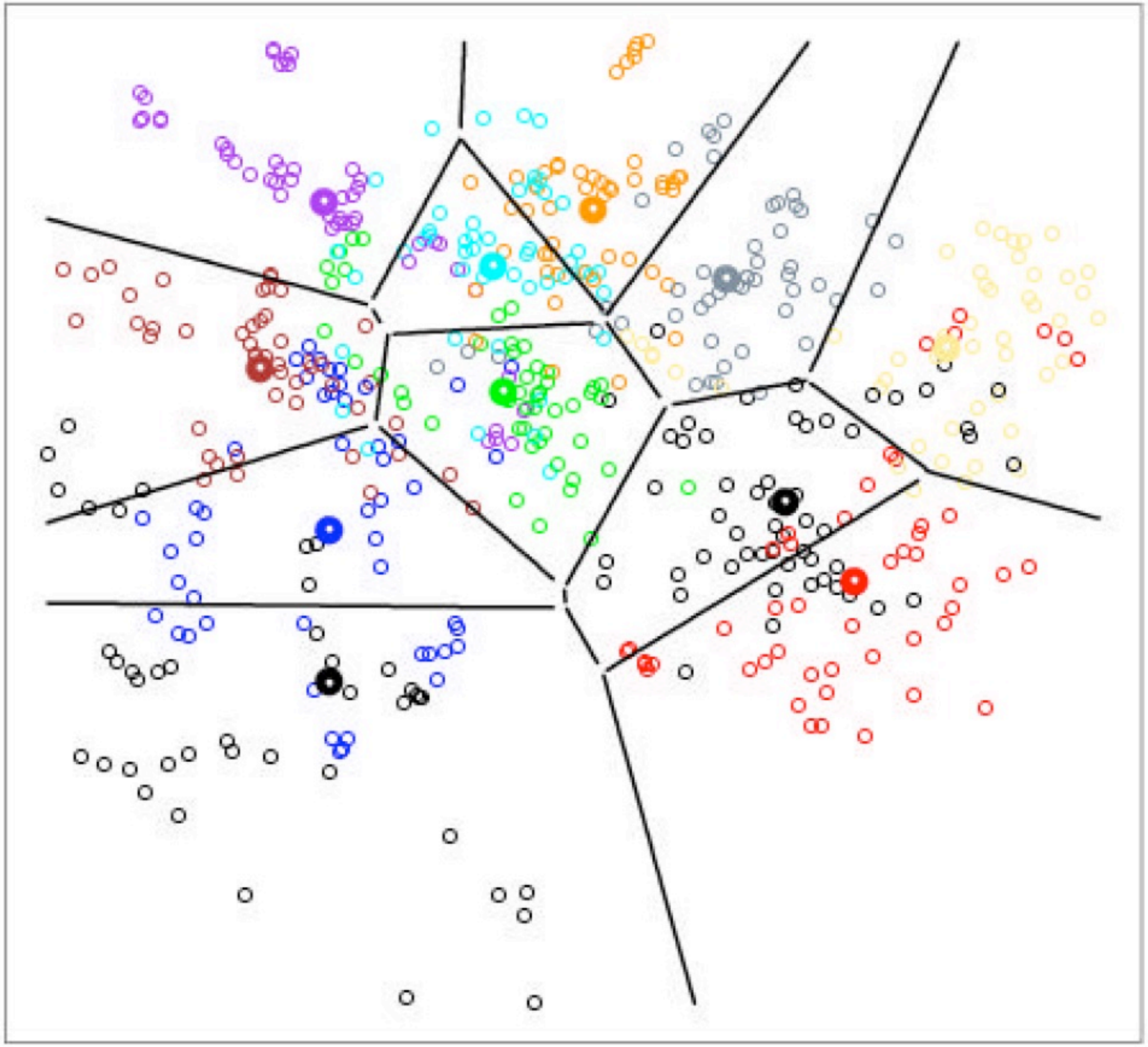
1) Prototype methods

- Training data: (x1,y1), (x2,y2), … , (xn,yn)
- Compute one prototype for each class
  - E.g. class center (Min. dist. classifier): $Ci = \Sigma_i x / |Ci|$
- Classification: a new instance is classified to the class of its "closest" prototypes
- ➔ very effective when prototypes are well positioned + data linearly separable

Example

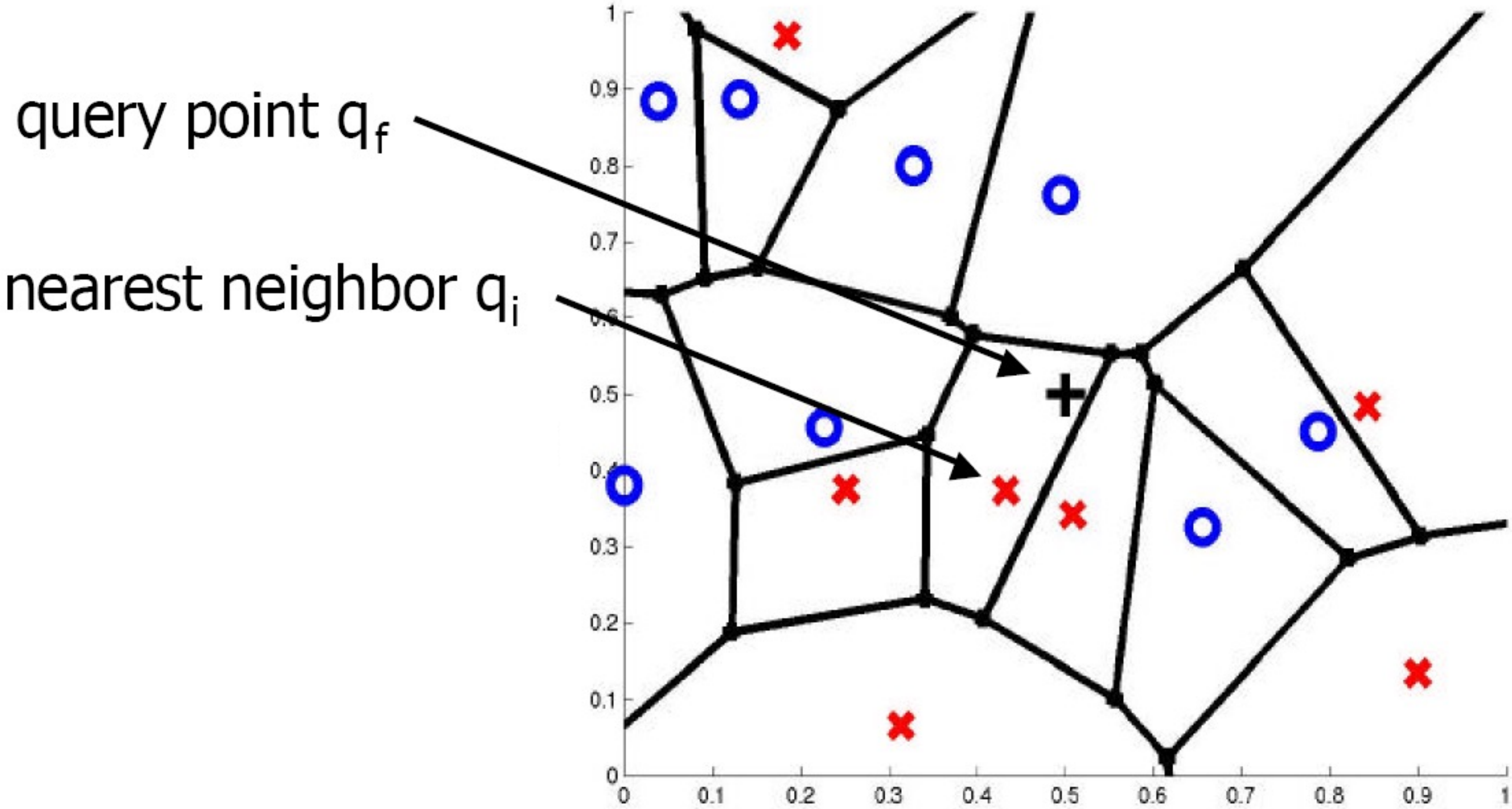


Example: vowel data



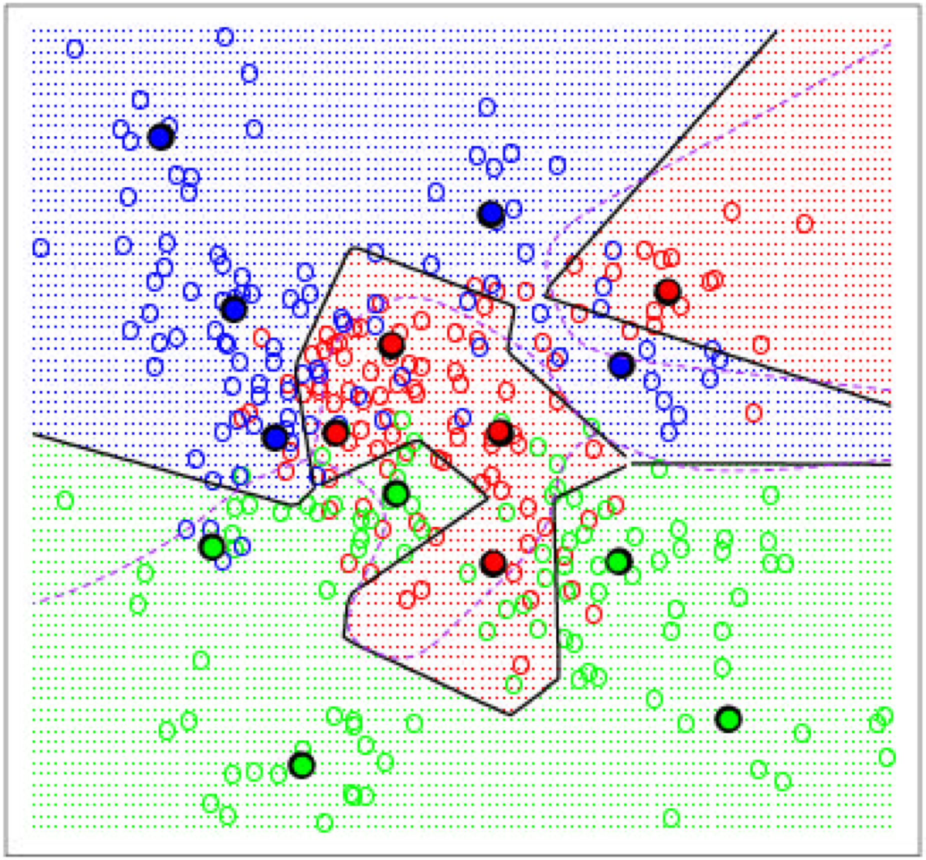
Voronoi diagram

- =decomposition of space det. by distances to a specific set of points
- Segments in a V. diagram = pts. equidistant to 2 prototypes



Modeling irregular class boundaries

- k (>1) prototypes for each class
- How many prototypes and where to put them?
  - ➔ k-means clustering



K-means clustering

- Delivers k prototypes for a given class

- See demo in class (1st demo is really nice)

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

https://stanford.edu/class/engr108/visualizations/kmeans/kmeans.html

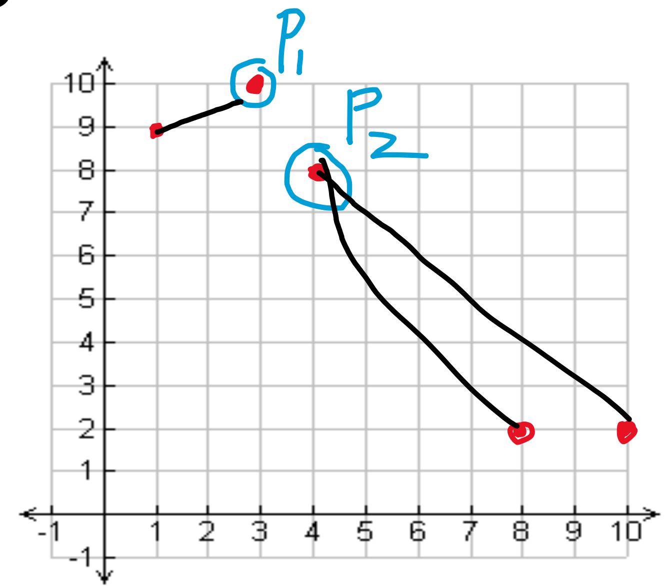
https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/

http://shabal.in/visuals/kmeans/4.html

- Algorithm
  - Start with an initial set of centers
  - Iterate until convergence
    - For each center, identify tr. points (= its cluster) that are closer to it than any other center
    - For each cluster, compute its mean vector that becomes its new center

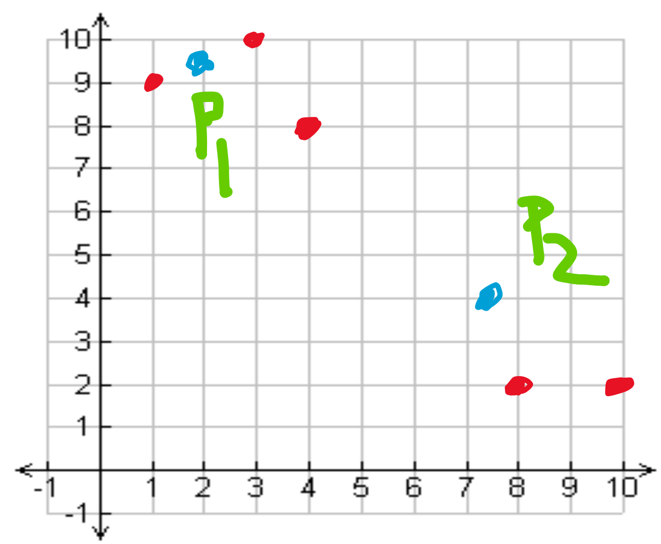
K-means clustering -demo

- Set-up: 5 red points; k=2 random prototypes shown in blue



Iteration = 1

- Black lines show closest prototype for all points
  - dist((1,9),(3,10)) = sqrt[(1-3)^2 + (9-10)^2] = sqrt[5] = **2.23**
  - dist((1,9),(4,8)) = sqrt[(1-4)^2 + (9-8)^2] = sqrt[10] = 3.16
  - Point (1,9) is closest to prototype (3,10) hence will belong to that cluster, at least at this iteration
  - ……
- Compute new prototypes
  - (1,9), (3,10) ➔ ((1+3)/2, (9+10)/2) = (2, 9.5)
  - (4,8), (8,2), (10,2) ➔(4+8+10)/3, (8+2+2)/3) = (7.3, 4)



Iteration = 2

- Draw lines to show closest prototype for all points
  - dist((x1,y1),(xp,yp)) = sqrt[(x1-xp)^2 + (y1-yp)^2], where (xp,yp) is a prototype
  - ….
- Compute new prototypes
  - (x1,y1), (x2,y2), … (xn, yn) ➔ ((x1+x2+…+xn)/n, (y1+1y2+…yn)/n)

Iteration = p

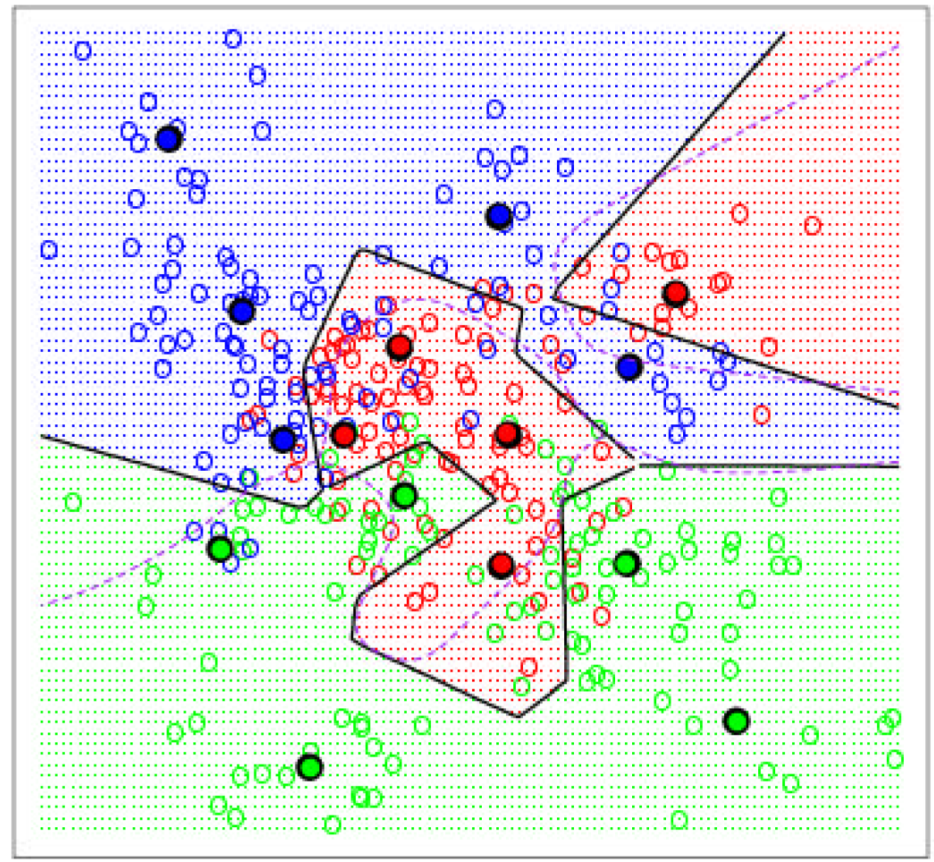
(repeat until convergence, i.e. no changes in location of prototypes)

K-prototypes method

- Apply k-means clustering to tr. data in each class separately, using k prototypes per class
- Assign a class label to each of the c*k prototypes
- Classify x to the class of the closest prototype

Obs. - K-prototypes method

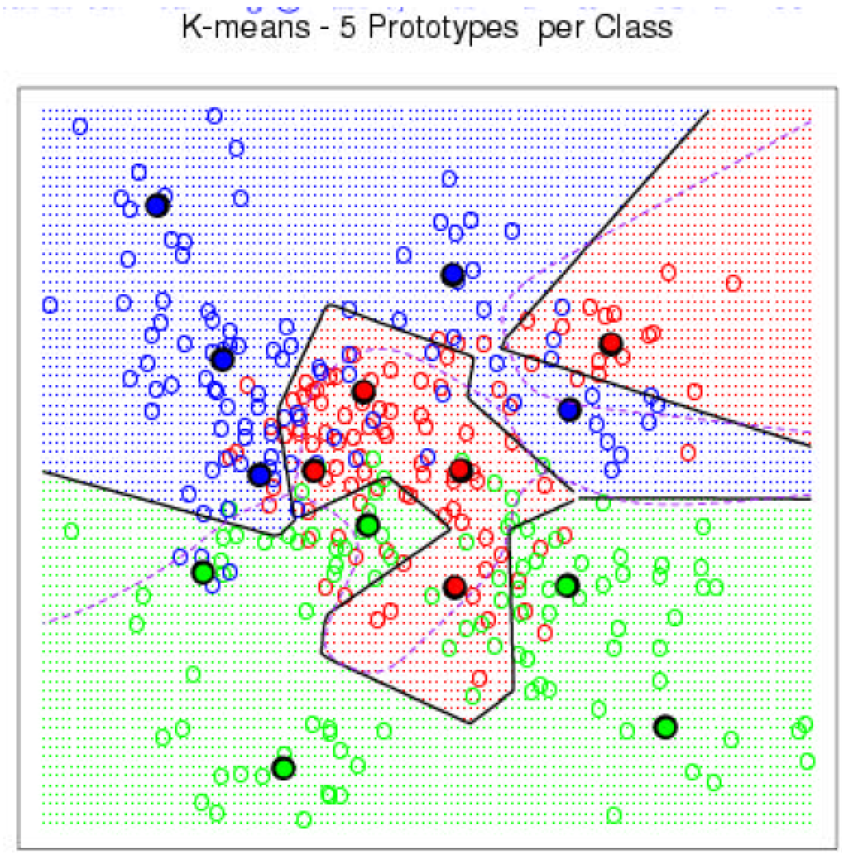
- “Negative” ex. not considered
- Many prototypes near class boundary
- ➔ potential misclassification for points near boundary
- Solution→ use LVQ

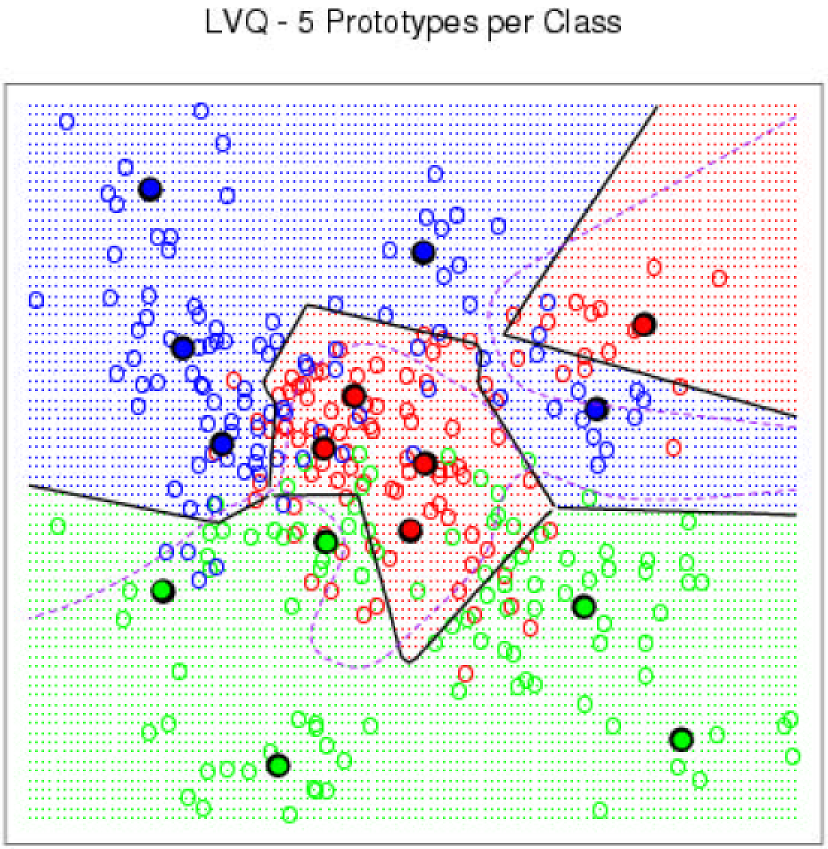


Learning vector quantization (LVQ)

- Choose k prototypes – e.g. use K-means
- Sample (rand.) an ex. (xi,vi); let (mj,gj) be the closest prototype
  - If vi=gj, move prototype towards (xi,vi)
    - mj=mj+ε(xi-mj)
  - else move prototype away from (xi,vi)
    - mj=mj-ε(xi-mj)
- Repeat above step, decreasing learning rate ε with each iteration until zero

Learning vector quantization





- Difficult to understand prototypes' properties

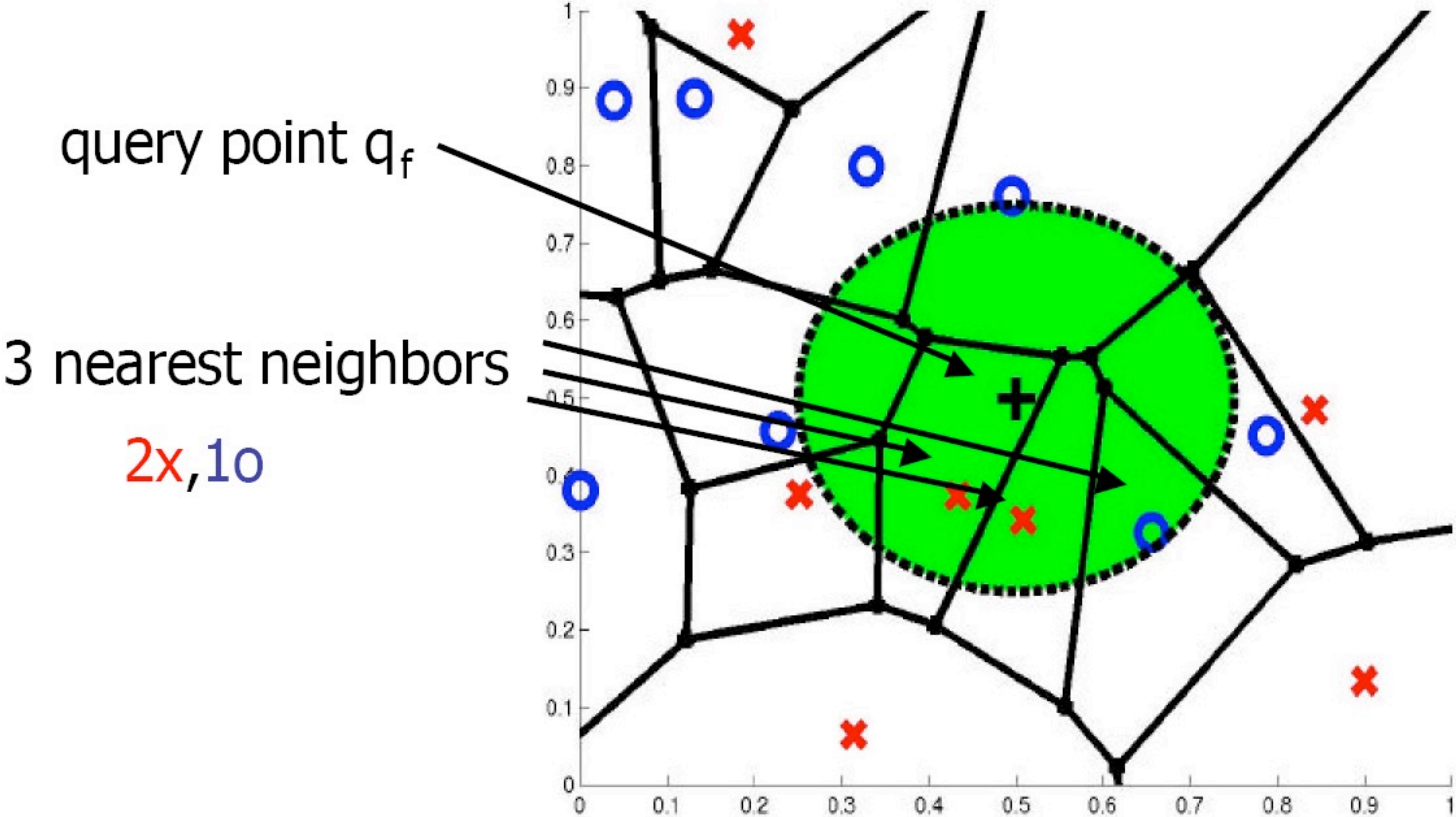
2) K-nearest neighbor (k-NN)

- Key idea: just store all training examples
  - Memory based, model free classifier

- K=1 ➔ Nearest neighbor
  - For test point x, find its closest point from tr. set
  - x has same class as its neighbor

- K-Nearest neighbor
  - For test point x. find its k closest points from tr. set
  - x has same class as the majority of its k nearest points

Ex.: 3-nearest neighbor

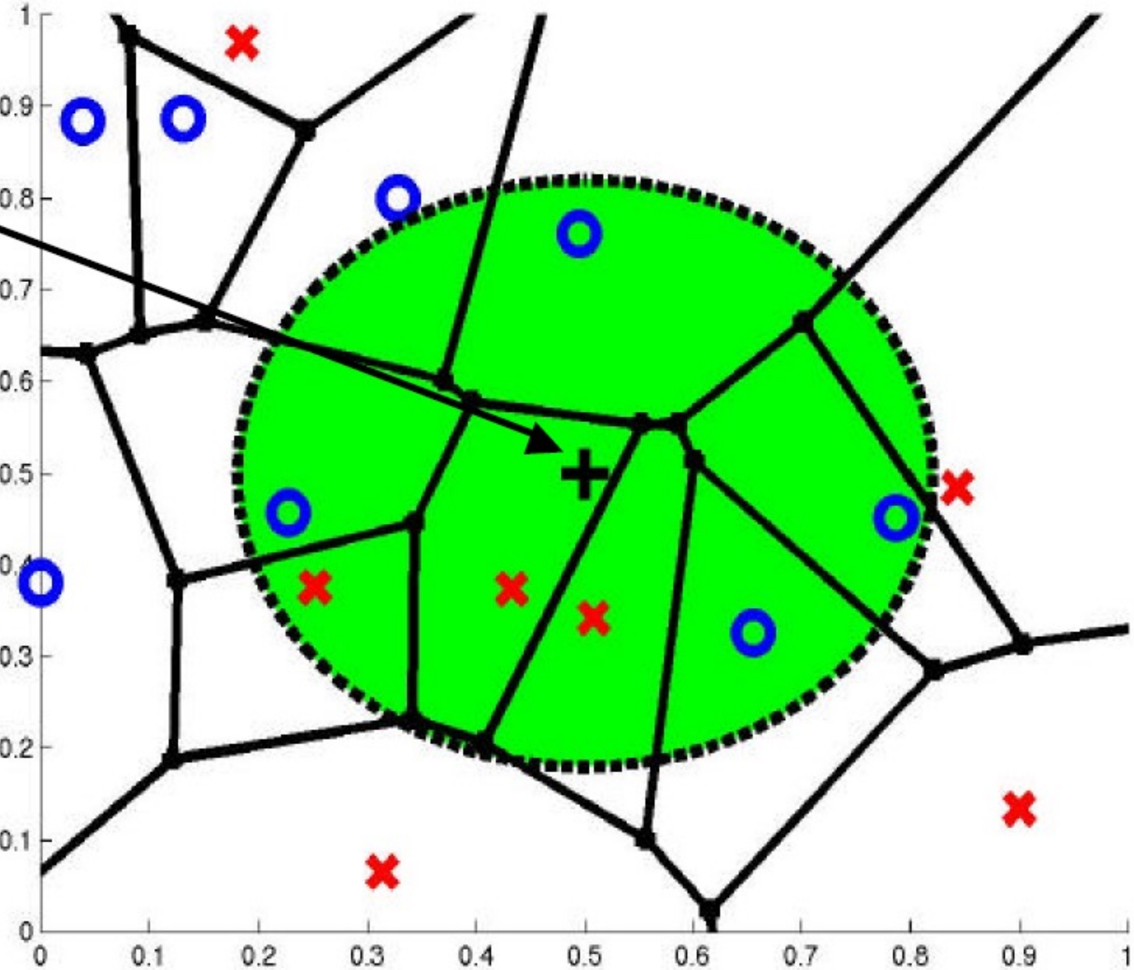


Ex.: 7-nearest neighbor

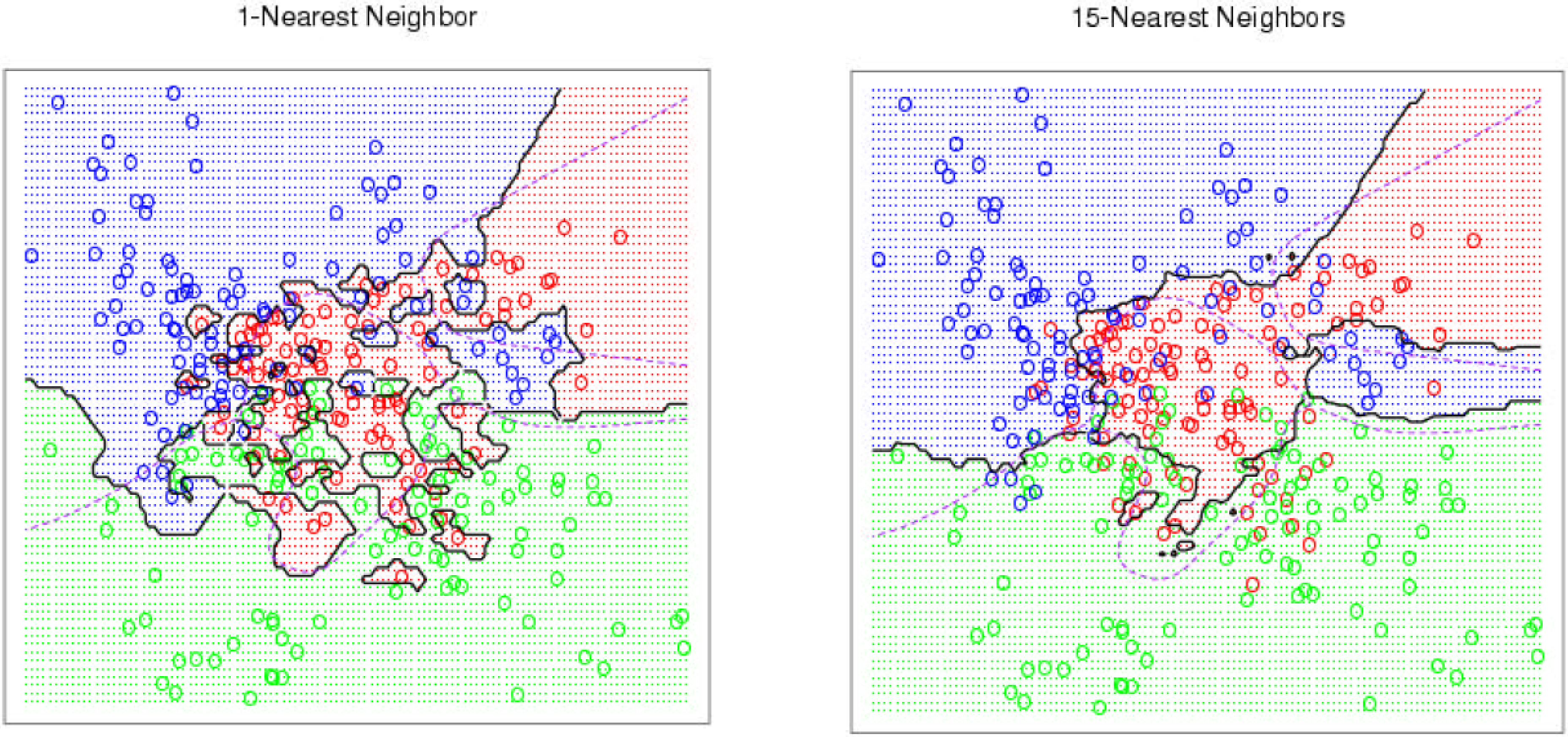
query point q_f

7 nearest neighbors

3x,4o

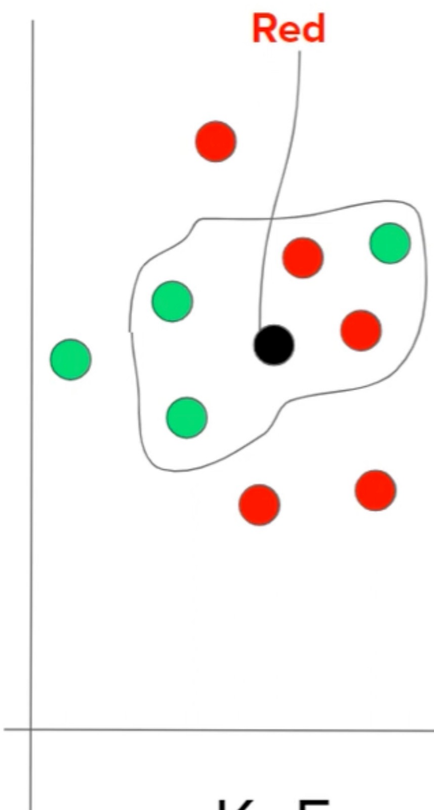


k-NN for classification



Ex. of dist. Weighted 5-NN

Point	Label	Distance	Weight
(x1,y1)	Red	0.2	5
(x2,y2)	Red	0.5	2
(x3,y3)	Green	0.7	1.4
(x4,y4)	Green	1.2	0.8
(x5,y5)	Green	1.5	0.6



K=5

Calculate Weight

Based on a **Weighing Function**

Distance Increases, Weight decreases

Simplest Weighing function

$\mathbf{w_i=1/d_i}$

1.4 + 0.8 + 0.6 = 2.8

5 + 2 = 7

Obs.
1) If considering ONLY labels of 5 neighbors -> green
2) If considering weighted dist of 5 neighbors -> red

Weight → shows importance of that neighbor
Weight → here is 1/d but usually is 1/d^2

Distance-weighted k-NN

1) Weight nearer neighbors more heavily – formula for **regression**

$$\hat{f}(x_q) \leftarrow \frac{\Sigma_{i=1}^{k} w_i f(x_i)}{\Sigma_{i=1}^{k} w_i}$$

where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

- (x1,f(x1)), (x2, f(x2)), …(xk, f(xk)) – k data nearest to test data xq
- f(xq) – regression value computed for xq

2) Formula for **classification**

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

- δ(v, f(xi)) = 1 if v = f(x_i), and 0 otherwise
- f(xq) – class for xq
- V - set of all classes

- → you may use all training examples instead of just k (Shepard's method)
  - OBS. very distant points will contribute very little in the decision
  - Disadvantage: slower

Obs. on k-NN

- When to use it?
  - Less than 20 attrib.
  - Lots of training data
  - Good for image classification (in some cases better than C4.5 and NN, see Statlog project)

- Adv.
  - Learn complex target functions
  - Training very fast = zero
  - kNN with majority vote approximates Bayes classifier

- Disadv.
  - Slow at query time
  - Easily fooled by irrelevant attributes
  - Difficulties in high dimensions ➔ curse of dimensionality

Irrelevant attributes and the curse of dimensionality

- Ex. 20 attr. but only 2 are relevant for classification
  - ➔ instances with **identical** 2 most relevant attributes may be **distant** in 20-dimensional space!
  - ooops → a major drawback

- As no. of attrib.↑
  - ➔ data becomes sparse
  - To capture 10% of data, we must cover 80% of the range of each attrib.
  - Data closer to edges of the sample space than to each other

- ➔ Rescale
  - E.g. a1:[0,1]; a2: [-10,10]
    - Distance w.r.t. a1 always small, hence a1 has small influence

- ➔ Attribute weighting

K-means clustering vs k-nearest neighbors

- unsupervised learning (class label **not** known)
- delivers prototypes, or even classes

- supervised learning (class label **are** known)
- used for classification of new data