BITS Pilani
Pilani Campus

Instance-based Learning

Dr. Sugata Ghosal

sugata.ghosal@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

Text Book(s)

T1	Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
T2	Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc..

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore and many others who made their course materials freely available online.

Topics to be covered

- Instance based learning
- K-Nearest Neighbour Learning
- Locally Weighted Regression (LWR) Learning
- Radial Basis Functions

k-Nearest Neighbor Classifier

- Nearest Neighbour classifier is an instance based classifier
- ‘lazy learning’, as learning is postponed until a new instance is encountered
- Constructs a local approximation to the target function, applicable in the neighbourhood of new instance
- Suitable in cases where target function is complex over the entire input space, but easily describable in local approximations
- Real world applications found in recommendation systems (amazon).
- Caveat is the high cost of classification, which happens at the time of processing rather than before hand (there’s no training phase)

Instance Based Learning

Key idea: just store all training examples $<x_i,f(x_i)>$

Nearest neighbor:

- Given query instance x_q, first locate nearest training example x_n, then estimate $f^*(x_q)=f(x_n)$

K-nearest neighbor:

- Given x_q, take vote among its k nearest neighbors (if discrete-valued target function)
- Take mean of f values of k nearest neighbors (if real-valued) $f^*(x_q)=\sum_{i=1}^{k} f(x_i)/k$

When to Consider Nearest Neighbors

- Instances map to points in R^N
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

k-Nearest Neighbor Classifier

- Considers all instances as members of n-dimensional space
- Nearest neighbours of an instance is determined based on Euclidean distance
- Distance between two n-dimensional instances x_i and x_j is given by:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

- For nearest neighbour classifier, target function can be discrete or continuous

Distance Measures : Special Cases of Minkowski

- *h* = 1: Manhattan (city block, L_1 norm) distance

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

- *h* = 2: (L_2 norm) Euclidean distance

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h\to\infty}\left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h\right)^{\frac{1}{h}} = \max_{f}^{p} |x_{if} - x_{jf}|$$

BITS Pilani, Pilani Campus

Standardizing Numeric Data

- X: raw score to be standardized, μ: mean of the population, σ: standard deviation
- the distance between the raw score and the population mean in units of the standard deviation
- negative when the raw score is below the mean, "+" when above

Where

$$z = \frac{x - \mu}{\sigma}$$

BITS Pilani, Pilani Campus

Multiple Features

$$y_n = w_0 + w_1 f_{n1} + w_2 f_{n2} + w_3 f_{n3} + w_4 f_{n4}$$

Size (feet²) f1	Number of bedrooms f2	Number of floors f3	Age of home (years) f4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Feature scaling

Feature scaling of features x_i consists of rescaling the range of features to scale the range in [0, 1] or [−1, 1] (Do not apply to $x_0 = 1$)

E.g. $x_1 = \frac{size - 1000}{2000}$

Average value of x1

Maximum value of x1 – min value of x1

$$x_2 = \frac{\#bedrooms - 2}{5}$$

Discrete and Continuous-valued function

- **discrete-valued target function:**
  - $f : \Re^n \rightarrow V$ where V is the finite set $\{v_1, v_2, ..., v_s\}$
  - the target function value is the most common value among the k nearest training examples

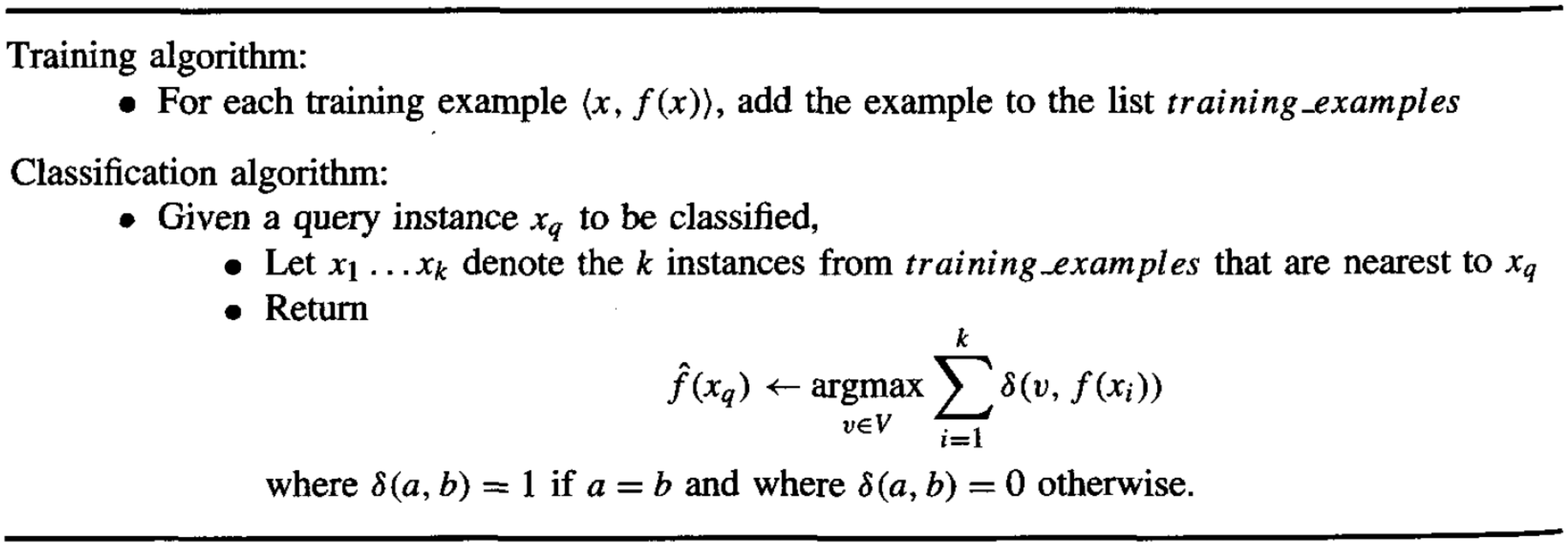
$$\hat{f}(x_q) \leftarrow \underset{v \in V}{argmax} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a, b) = (a == b)$

- **continuous-valued target function:**
  - algorithm has to calculate the mean value instead of the most common value
  - $f : \Re^n \rightarrow \Re$

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

k-Nearest Neighbor Classifier



Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list $training\_examples$

Classification algorithm:

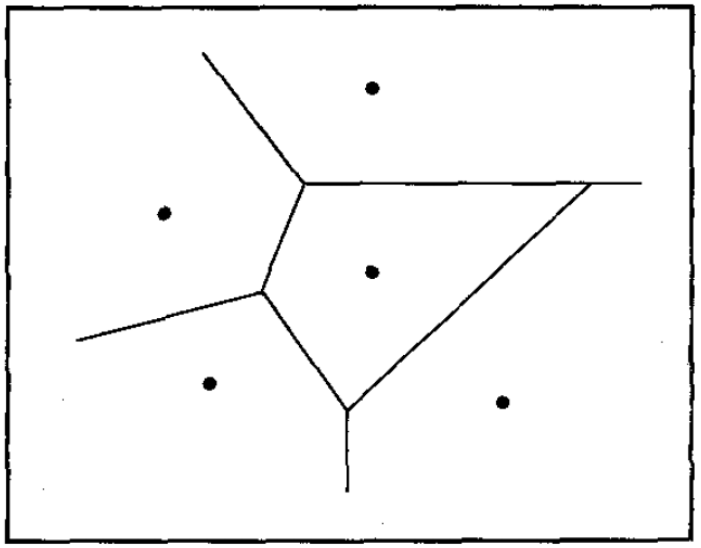
- Given a query instance x_q to be classified,
    - Let $x_1 \ldots x_k$ denote the k instances from $training\_examples$ that are nearest to x_q
    - Return

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

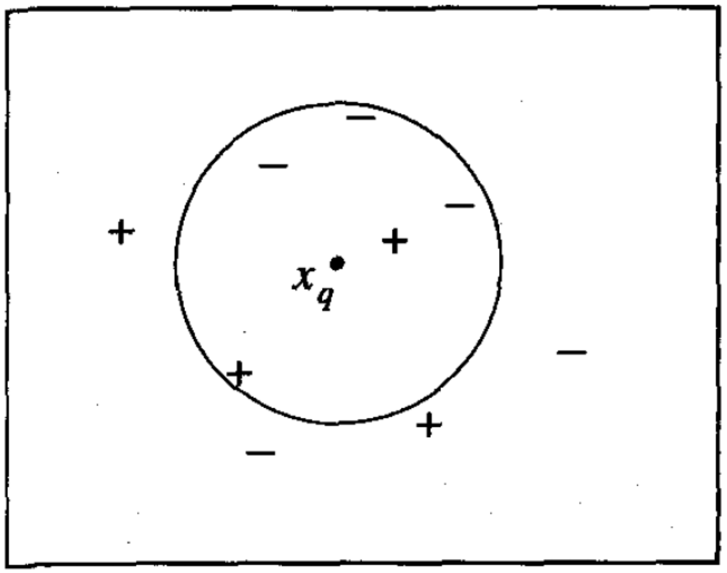
where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

* It can be used for Regression as well.

k-NN examples



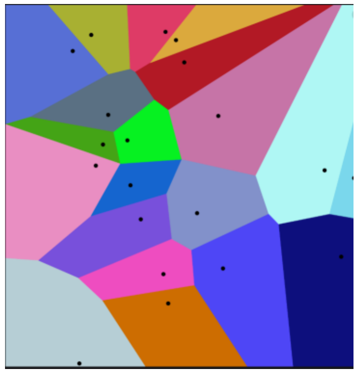
K=1



K=5

Voronoi Diagram

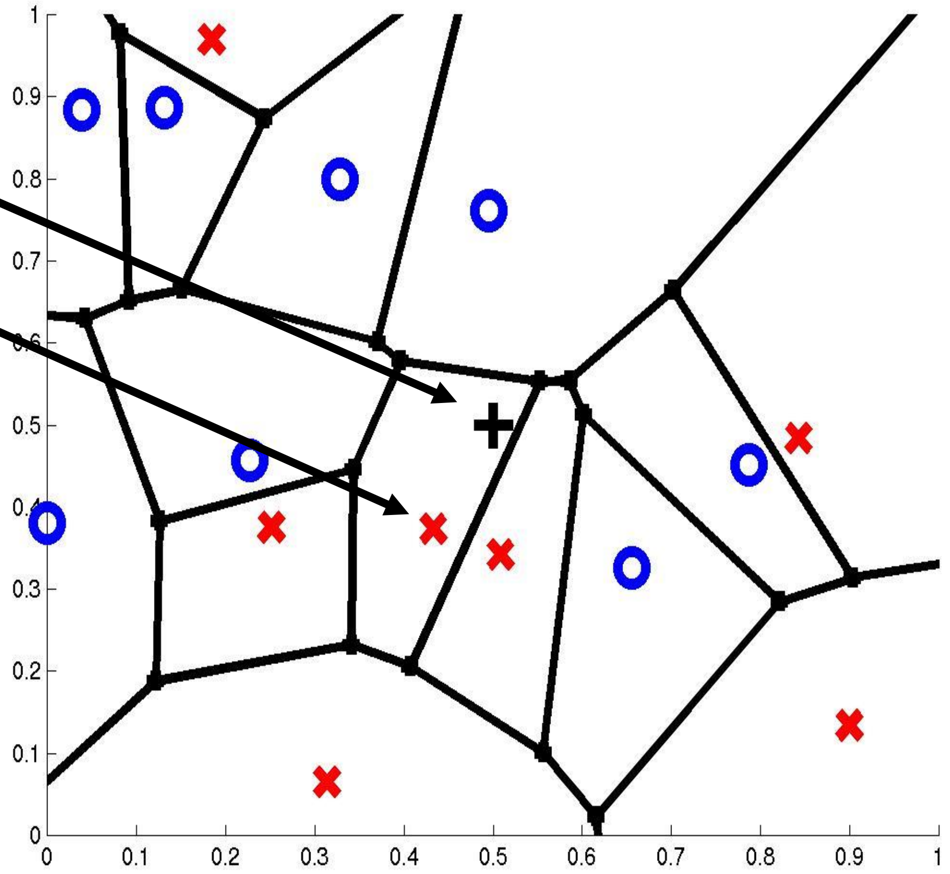
- It is a partition of a plane into regions close to each of a given set of objects.



Voronoi Diagram

query point q_f

nearest neighbor q_i

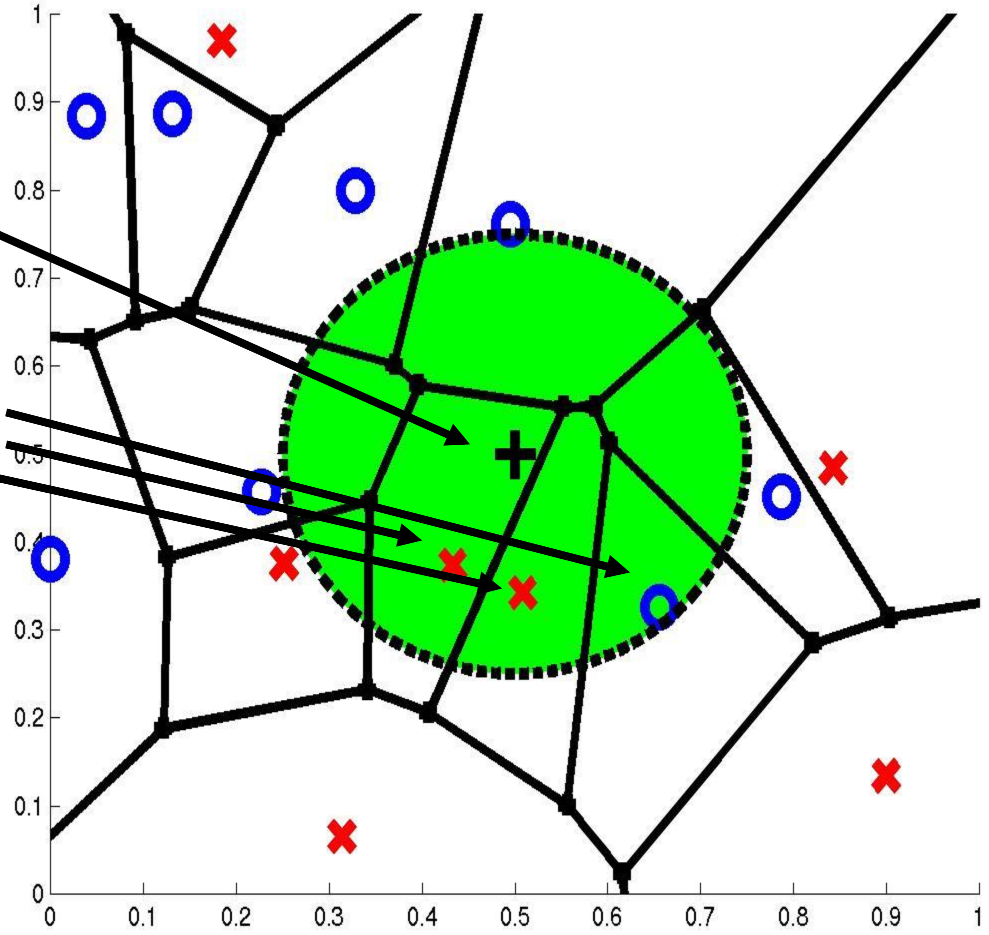


3-Nearest Neighbors

query point q_f

3 nearest neighbors

2x,1o

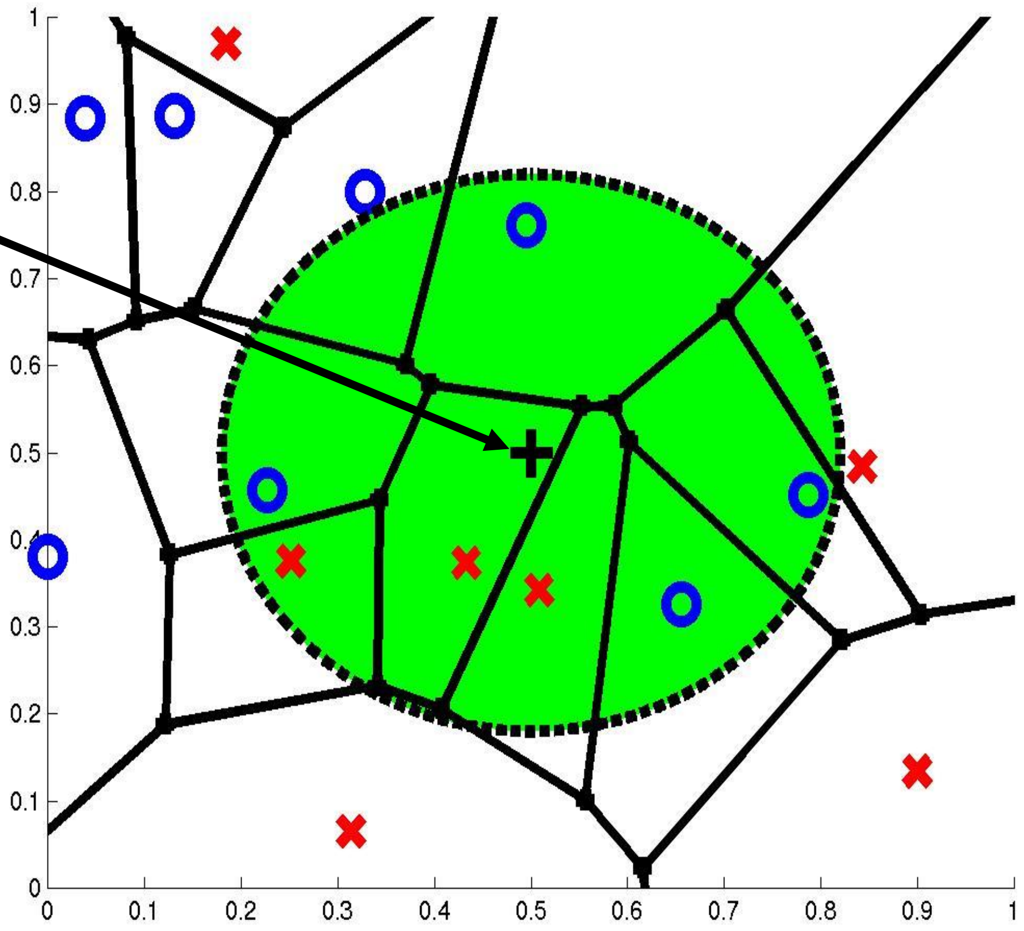


7-Nearest Neighbors

query point q_f

7 nearest neighbors

3x,4o

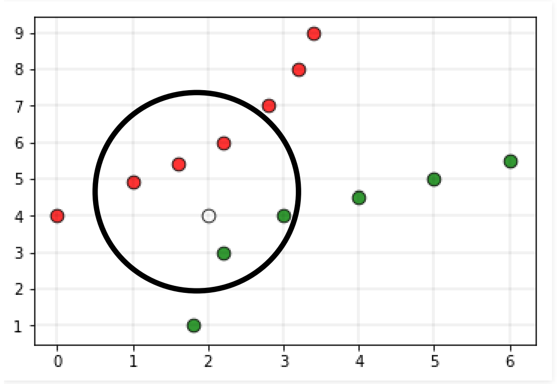


Various issues that affect the performance of kNN:

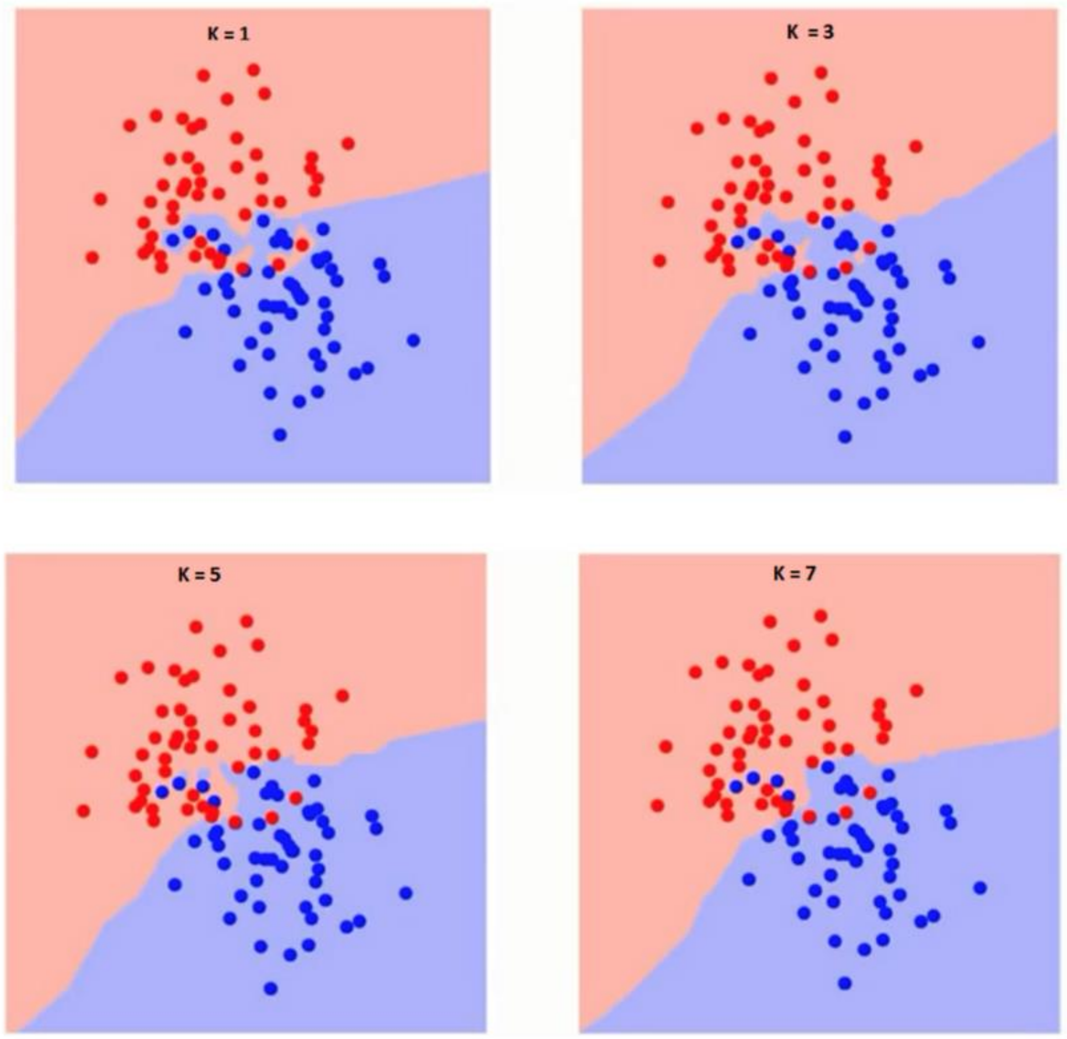
Performance of a classifier largely depends on the of the hyperparameter k

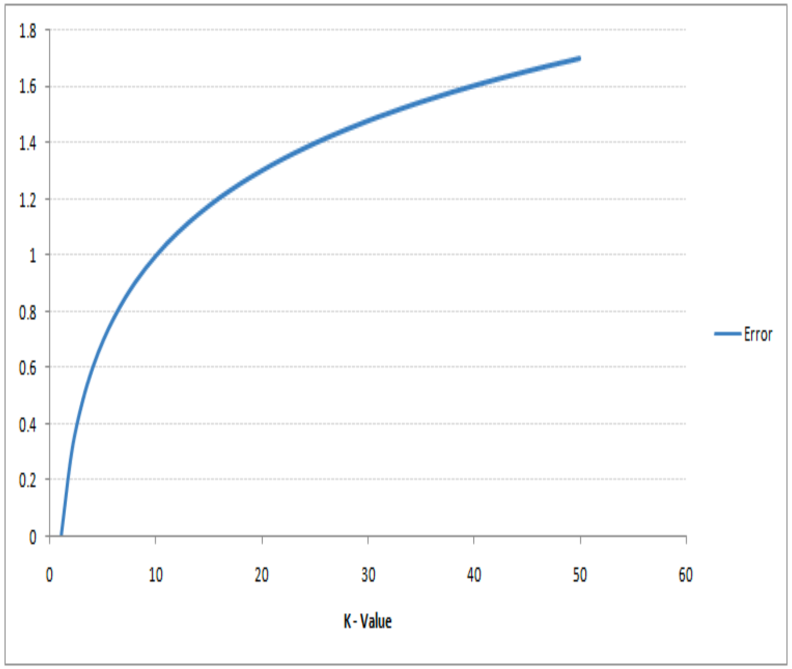
- Choosing smaller values for K, noise can have a higher influence on the result.
- Larger values of k are computationally expensive

Assigning the class labels can be tricky. For example, in the below case, for (k=5) the point is closer to ‘green’ classification, but gets classified as ‘red’ due to higher red votes/majority voting to ‘red’



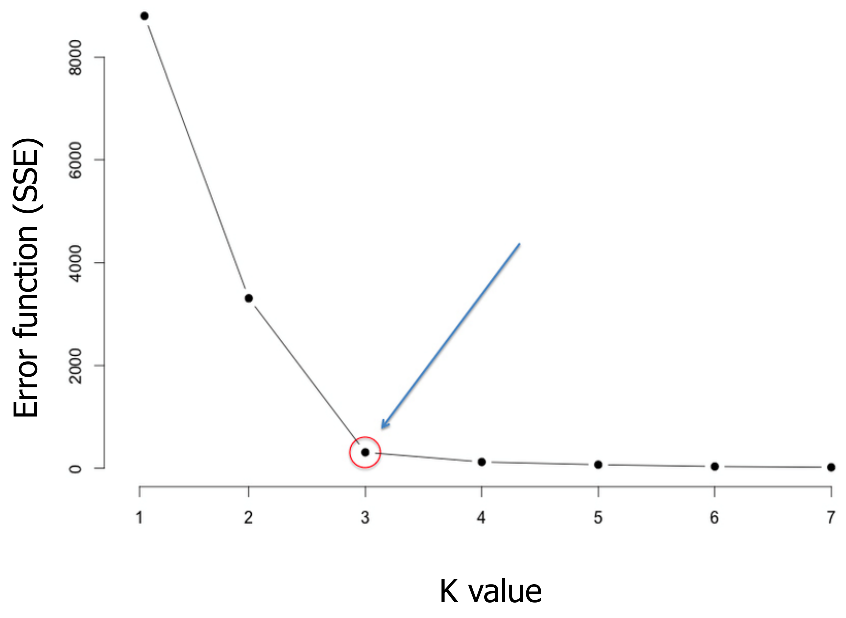
Finding optimal k for kNN classifiers





Finding K - Elbow method

- Compute sum of squares error (SSE) or any other error function for varying values of K (1 to a reasonable X) and plot against K
- In the plot, the elbow (see pic) gives the value of K beyond which the error function plot almost flattens
- As K approaches the total number of instances in the set, error function drops down to '0', but beyond optimal K, the model becomes too generic



Distance weighted nearest neighbor

- contribution of each of the k nearest neighbors is weighted accorded to their distance to x_q
  - **discrete-valued target functions**

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

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

  - **continuous-valued target function:**

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

Distance Weighted k-NN

Give more weight to neighbors closer to the query point

- $f^*(x_q) = \Sigma_{i=1}^{k} w_i\, f(x_i) \,/\, \Sigma_{i=1}^{k} w_i$;
- $w_i=K(d(x_q,x_i))$

and

- $d(x_q,x_i)$ is the distance between x_q and x_i

Variation: Instead of only k-nearest neighbors use all training examples (Shepard's method)

Distance Weighted Average

Weighting the data

- $f^*(x_q) = \Sigma_i\, f(x_i)\, K(d(x_i,x_q)) / \Sigma_i\, K(d(x_i,x_q))$

Relevance of a data point (x_i, $f(x_i)$) is measured by calculating the distance $d(x_i, x_q)$ between the query x_q and the input vector x_i

Weighting the error criterion

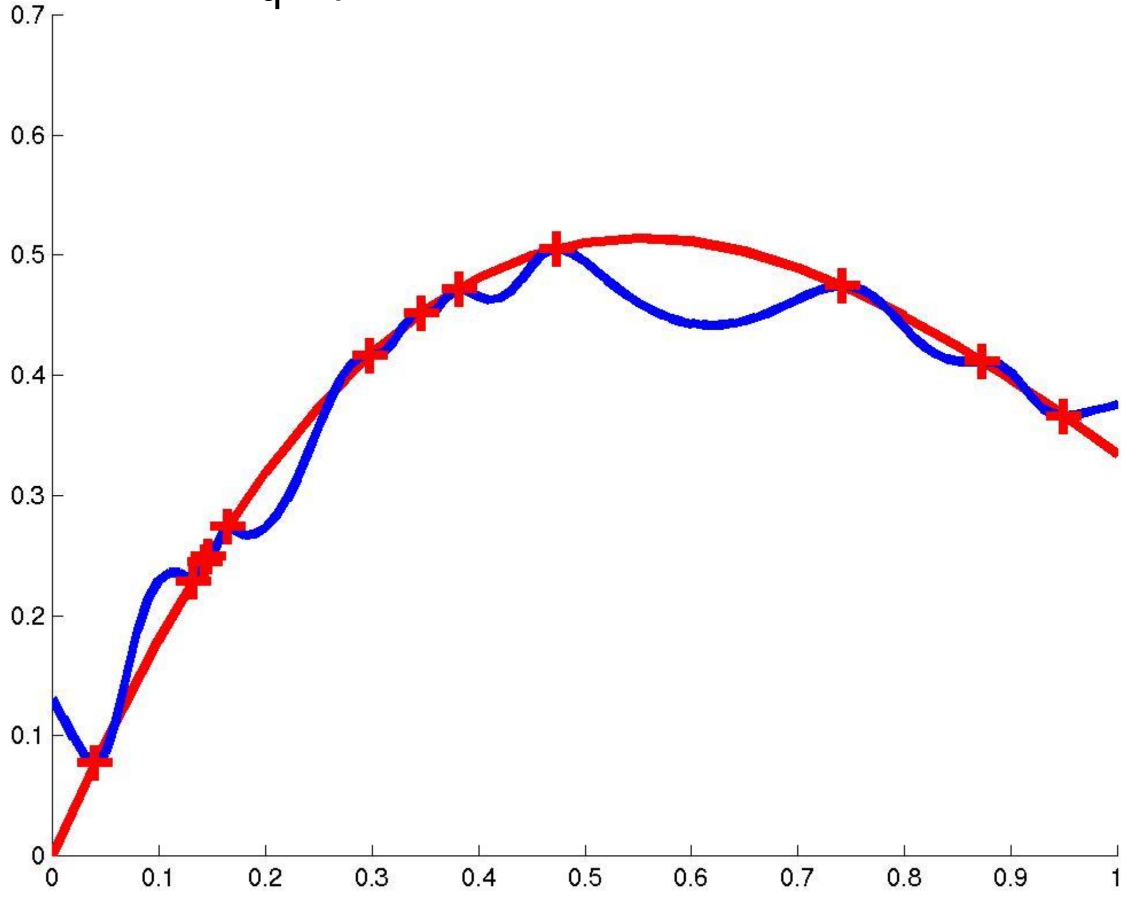
- $E(x_q) = \Sigma_i\, (f^*(x_q)-f(x_i))^2\, K(d(x_i,x_q))$

Best estimate $f^*(x_q)$ will minimize the cost $E(x_q)$, therefore

$$\partial E(x_q)/\partial f^*(x_q)=0$$

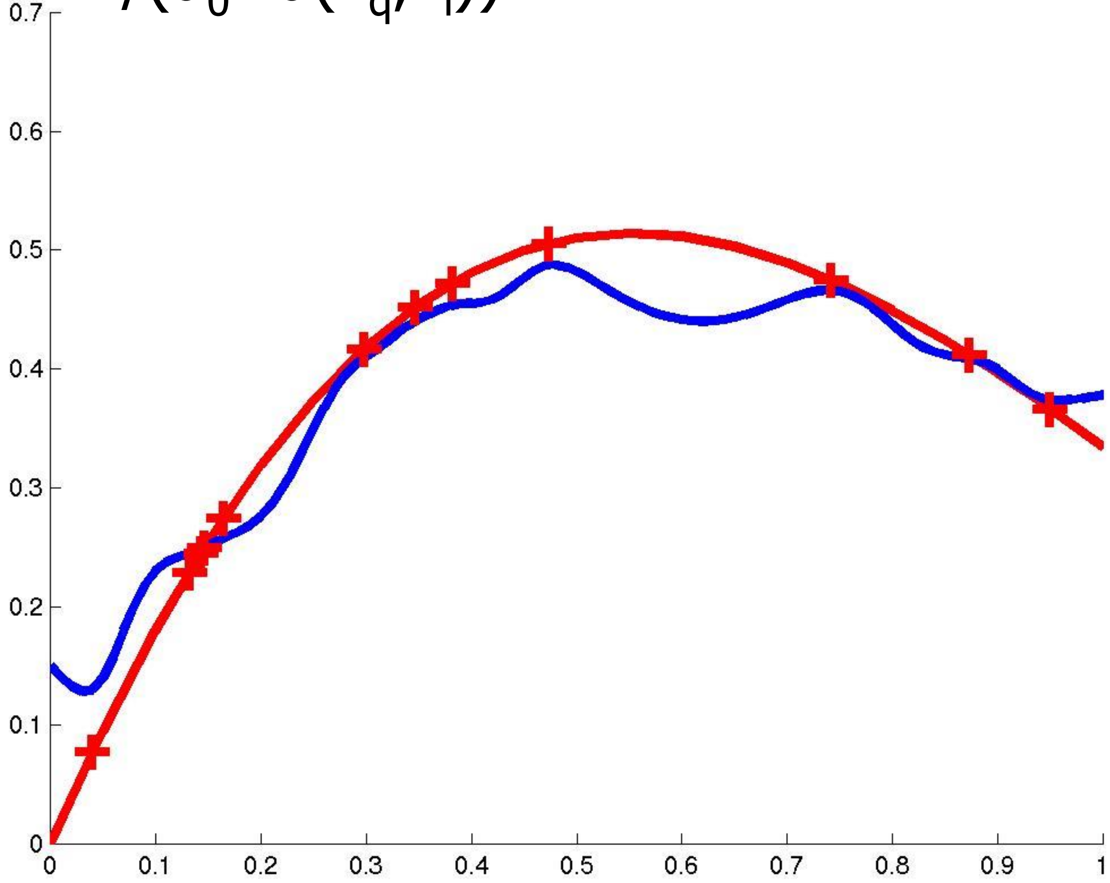
Distance Weighted NN

$K(d(x_q,x_i)) = 1/\ d(x_q,x_i)^2$



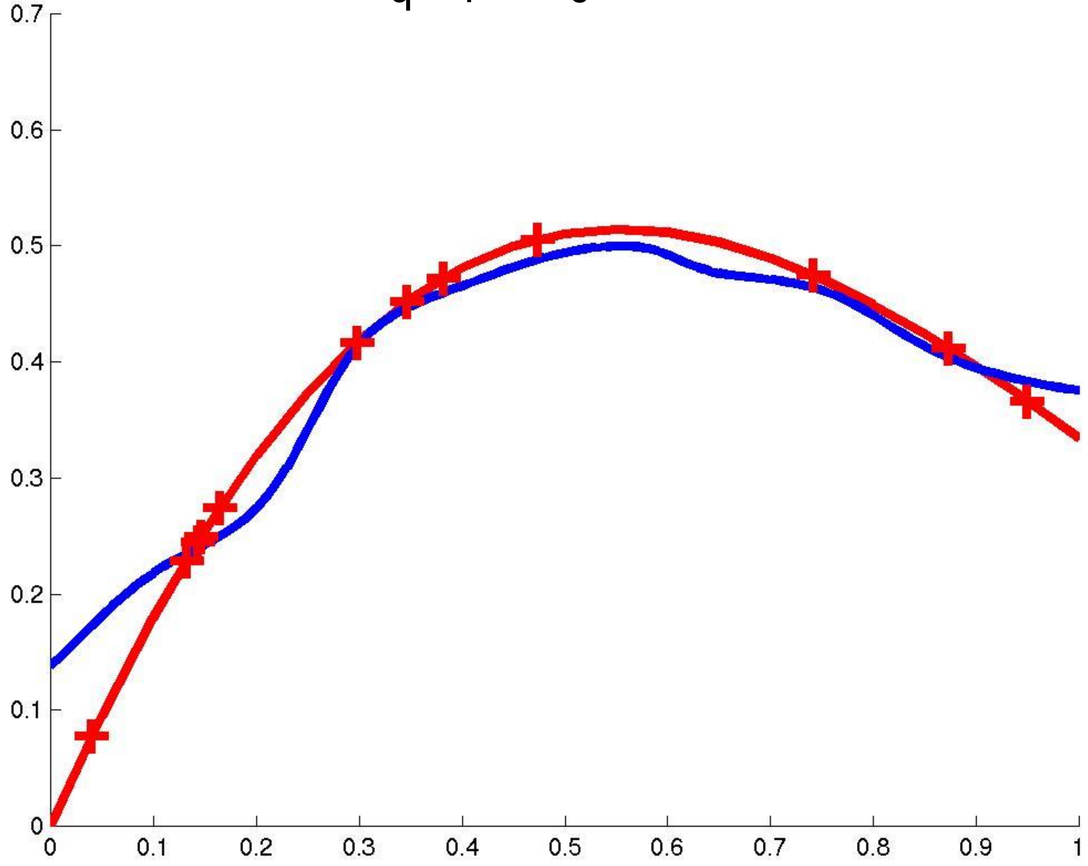
Distance Weighted NN

$K(d(x_q,x_i)) = 1/(d_0+d(x_q,x_i))^2$



Distance Weighted NN

$K(d(x_q,x_i)) = \exp(-(d(x_q,x_i)/\sigma_0)^2)$ – ***Gaussian weighted***



Curse of Dimensionality

Imagine instances described by 20 attributes but only a few are relevant to target function

Curse of dimensionality: nearest neighbor is easily misled when instance space is high-dimensional

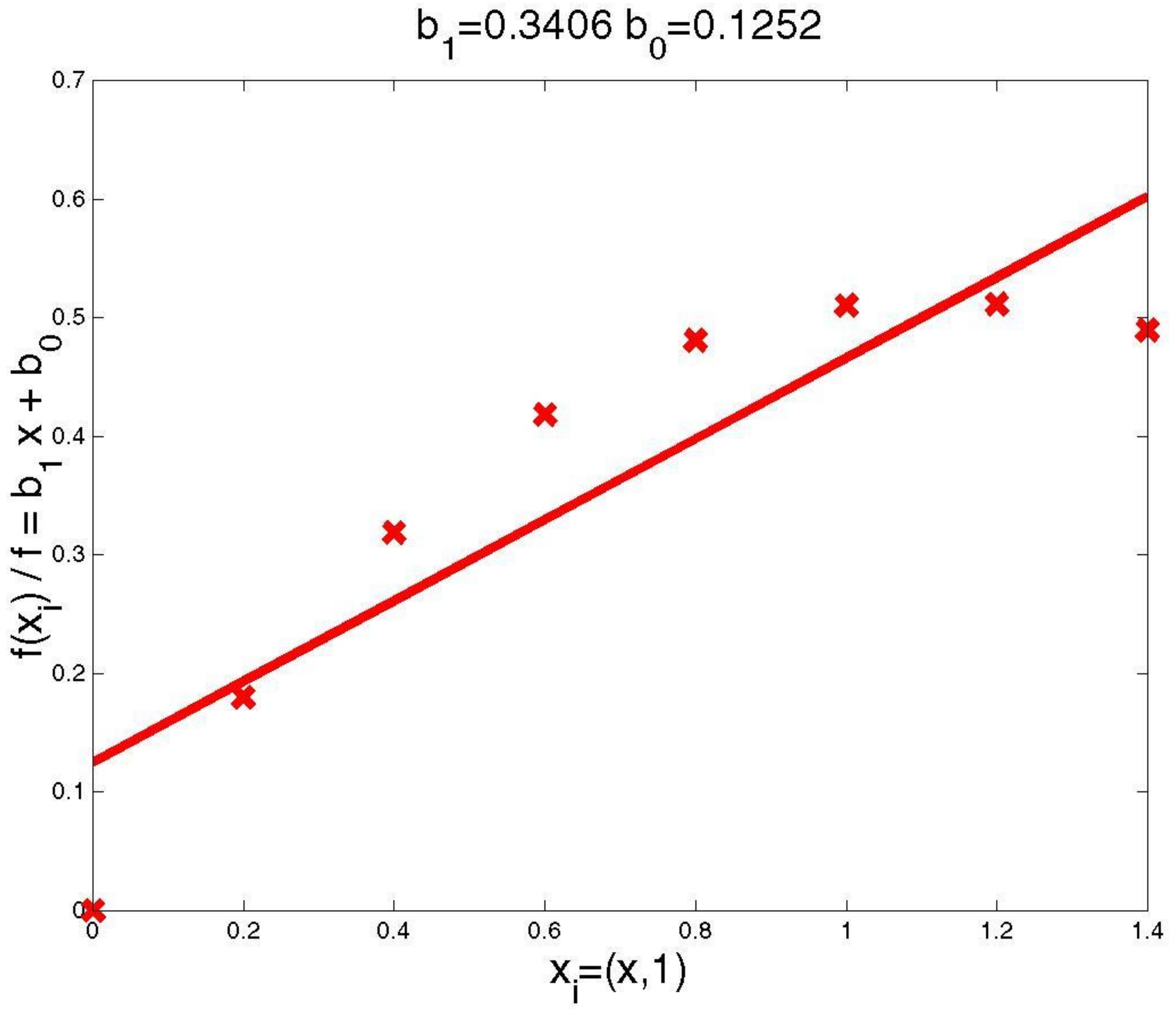
One approach:
Stretch j-th axis by weight z_j, where $z_1,\ldots,z_n$ chosen to minimize prediction error
Use cross-validation to automatically choose weights $z_1,\ldots,z_n$
Note setting z_j to zero eliminates this dimension alltogether (feature subset selection)

Locally Weighted Regression

Locally Weighted Regression

- Locally – Function approximated based on data near query point
- Weighted – Contribution by each training example is weighted by its distance from query point
- Regression- Approximates real-valued target function

Linear Regression Example



Locally weighted regression

- a note on terminology:
  - *Regression* means approximating a real-valued target function
  - *Residual* is the error $\hat{f}(x) - f(x)$ in approximating the target function
  - *Kernel function* is the function of distance that is used to determine the weight of each training example. In other words, the kernel function is the function K such that $w_i = K(d(x_i, x_q))$
- nearest neighbor approaches can be thought of as approximating the target function at the single query point x_q
- locally weighted regression is a generalization to this approach, because it constructs an explicit approximation of f over a local region surrounding x_q

Locally weighted regression

- target function is approximated using a **linear function**

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

- methods like **gradient descent** can be used to calculate the coefficients $w_0, w_1, \ldots, w_n$ to minimize the error in fitting such linear functions

- ANNs require a global approximation to the target function
- here, just a local approximation is needed

$\Rightarrow$ the error function has to be redefined

Locally weighted regression

- possibilities to redefine the error criterion E
  1. Minimize the squared error over just the k nearest neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neigbors}} (f(x) - \hat{f}(x))^2$$

  2. Minimize the squared error over the entire set D, while weighting the error of each training example by some decreasing function K of its distance from x_q

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

  3. Combine 1 and 2

$$E_3(x_q) \equiv \frac{1}{2} \sum_{x \in k \text{ nearest neighbors}} (f(x) - \hat{f}(x))^2 \cdot K(d(x_q, x))$$

Locally weighted regression

- choice of the error criterion
  - E_2 is the most esthetically criterion, because it allows every training example to have impact on the classification of x_q
  - however, computational effort grows with the number of training examples
  - E_3 is a good approximation to E_2 with constant effort

$$\Delta w_j = \eta \sum_{x \in k \text{ nearest neighbors}} K(d(x_q, x))(f(x) - \hat{f}(x))a_j$$

- Remarks on locally weighted linear regression:
  - in most cases, constant, linear or quadratic functions are used
  - costs for fitting more complex functions are prohibitively high
  - simple approximations are good enough over a sufficiently small subregion of X

Design Issues in Local Regression

- Local model order (constant, linear, quadratic)
- Distance function d
  - feature scaling: $d(x,q)=(\Sigma_{j=1}^{d} m_j(x_j-q_j)^2)^{1/2}$
  - irrelevant dimensions $m_j=0$
- kernel function K

See paper by Atkeson [1996] "Locally Weighted Learning"

Radial Basis Function

Radial Basis Function

- closely related to distance-weighted regression and to ANNs
- learned hypotheses have the form

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u \cdot K_u(d(x_u, x))$$

where

- each x_u is an instance from X and
- $K_u(d(x_u, x))$ decreases as $d(x_u, x)$ increases and
- k is a user-provided constant

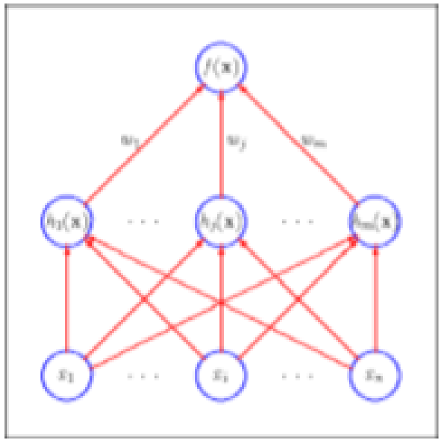
- though $\hat{f}(x)$ is a global approximation to $f(x)$, the contribution of each of the K_u terms is localized to a region nearby the point x_u

Radial Basis Function

- it is common to choose each function $K_u(d(x_u, x))$ to be a Gaussian function centered at x_u with some variance σ^2

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

- the function of $\hat{f}(x)$ can be viewed as describing a two-layer network where the first layer of units computes the various $K_u(d(x_u, x))$ values and the second layer a linear combination of the results



Training Radial Basis Function Networks

How to choose the center x_n for each Kernel function K_n?

- scatter uniformly across instance space
- use distribution of training instances (clustering)

How to train the weights?

- Choose mean x_n and variance σ_n for each K_n
  - say, by using non-linear optimization or EM
- Hold K_n fixed and use local linear regression to compute the optimal weights w_n

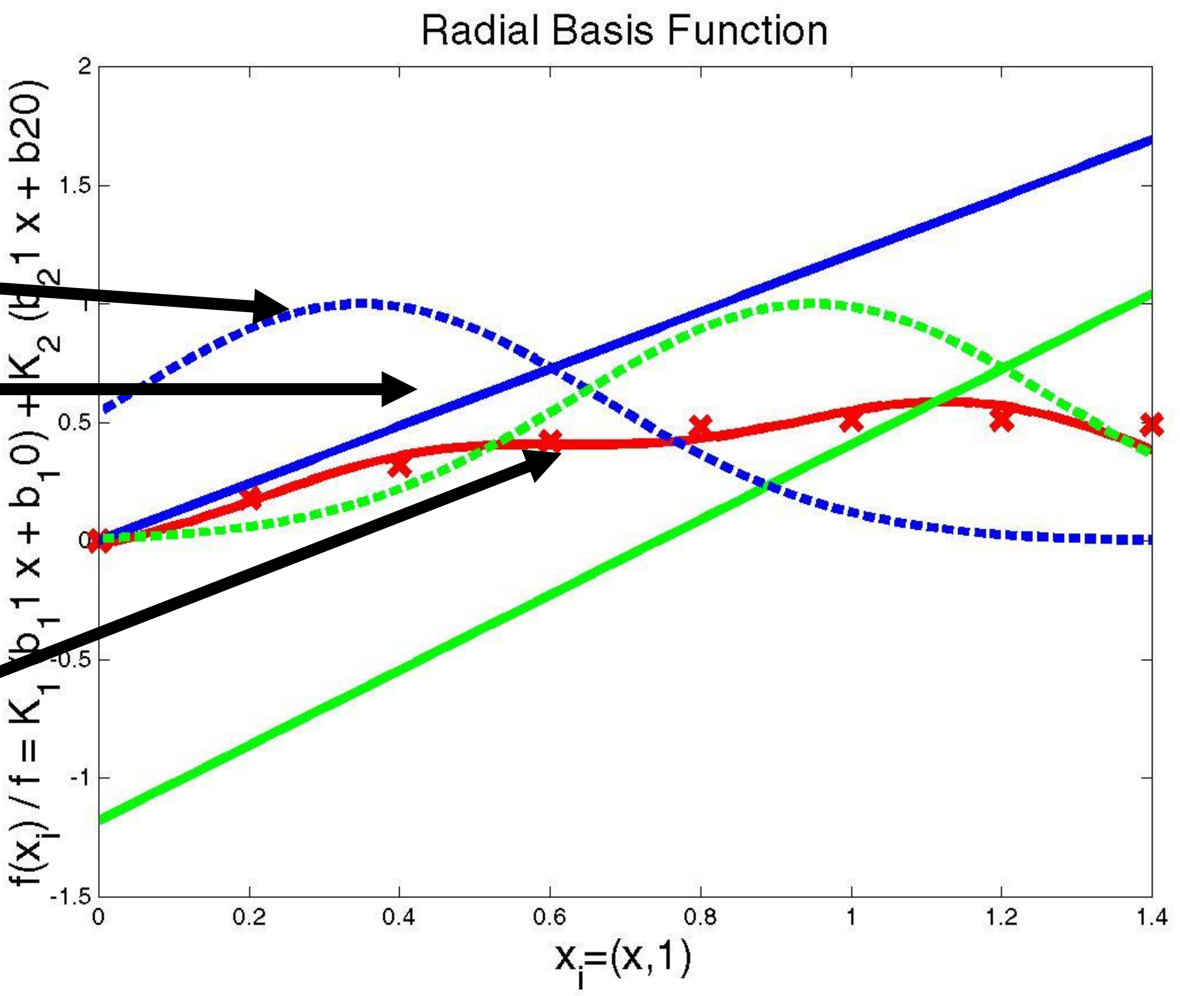
Radial Basis Network Example

$K_1(d(x_1,x))=$

$\exp(-1/2\ d(x_1,x)^2/\sigma^2)$

$w_1\ x+ w_0$

$\hat{f}(x) = K_1\ (w_1\ x+ w_0)$
$\quad + K_2\ (w_3\ x + w_2)$



Lazy and Eager Learning

Lazy: wait for query before generalizing

– k-nearest neighbors, weighted linear regression

Eager: generalize before seeing query

– Radial basis function networks, decision trees, back-propagation

– Eager learner must create global approximation

Lazy learner can create local approximations
If they use the same hypothesis space, lazy can represent more complex functions (H=linear functions)

Literature & Software

- T. Mitchell, “Machine Learning”, chapter 8, “Instance-Based Learning”
- “Locally Weighted Learning”, Christopher Atkeson, Andrew Moore, Stefan Schaal
- R. Duda et al., “Pattern recognition”, chapter 4 “Non-Parametric Techniques”

Netlab toolbox

- –k-nearest neighbor classification
- –Radial basis function networks

Exercise - 1

Consider the following training set in 2-dimensional Euclidean space What is the class of the point (1, 1) if 7NN classifier is considered? If the value of K is reduced whether the class will change? (Consider K=3 and K=5). What should be the final class if the above 3 values of K are considered?

Point	Coordinate	Class
X1	(-1, 1)	Negative
X2	(0, 1)	Positive
X3	(0, 2)	Negative
X4	(1, -1)	Negative
X5	(1, 0)	Positive
X6	(1, 2)	Positive
X7	(2, 2)	Negative
X8	(2, 3)	Positive

Exercise -1

Point	Coordinate	Class	Distance from 1,1
X1	(-1, 1)	Negative	2
X2	(0, 1)	Positive	1
X3	(0, 2)	Negative	1.414
X4	(1, -1)	Negative	2
X5	(1, 0)	Positive	1
X6	(1, 2)	Positive	1
X7	(2, 2)	Negative	1.41
X8	(2, 3)	Positive	2.236

- class of the point (1, 1) if 3NN classifier is considered x2, x5, x6 - Positive
- class of the point (1, 1) if 5NN classifier is considered?

x2, x5, x6, x3, x7 - Positive

- class of the point (1, 1) if 7NN classifier is considered?

x2, x5, x6, x3, x7, x1, x5- Negative

Final class value to be considered as Positive

Exercise - 2

discrete-valued target functions

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

where $w_i \equiv \frac{1}{d(x_q, x_i)^2}$ and $\hat{f}(x_q) = f(x_i)$ if $x_q = x_i$

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use distance weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Locality	Category
C0	60000	Doctor	Hindi	Village	**L1**
C1	70000	Doctor	Bengali	Village	**L2**
C2	60000	Carpenter	Hindi	Suburban	**L2**
C3	80000	Doctor	Bhojpuri	Metropolitan	**L2**
C5	80000	Data Scientist	Hindi	Small Town	**L1**
C4	**50000**	**Data Scientist**	**Hindi**	**Small Town**	

Proximity Measures for Attributes of Mixed Type

- Approach
  - Process all attribute types together, performing a single analysis
  - Combine the different attributes into a single dissimilarity matrix, bringing all of the meaningful attributes onto a common scale of the interval [0.0, 1.0]
- Suppose that the data set contains *p* attributes of mixed type
- The dissimilarity *d(i, j)* between objects *i* and *j* is defined as:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

- Where indicator $\delta_{ij}^{(f)} = 0$, if either
  - x_{if} or x_{jf} is missing (i.e., there is no measurement of attribute *f* for object i or j), OR
  - $x_{if} = x_{jf} = 0$ and attribute *f* is asymmetric binary
- Otherwise
  - $\delta_{ij}^{(f)} = 1$

Proximity Measures for Attributes of Mixed Type

– $d_{ij}^{(f)}$ is the contribution of attribute *f* to the dissimilarity between i and j. This is computed as:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

- **If f is ordinal : convert ordinal attribute to Z_{if}**

1. Compute rank r_{if} and M_f = number of ordered states, calculate $z_{if} = \frac{r_{if} - 1}{M_f - 1}$
2. Treat Z_{if} as numeric

- **If f is numeric then d_{ij} is calculated as**

1. apply min-max normalization
2. Euclidian distance between i and j

- **If f is nominal then apply simple matching :** $d(i,j) = \frac{p-m}{p}$

–*p* = total number of attributes describing the objects

–*m* = number of *matches* (i.e., the number of attributes for which *i* and *j* are in the same state),

Exercise - 2

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use locally weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 → 0	**L1**	**0.97**	**1.06**
C1	0.67	Doctor	Bengali	1 → 0	**L2**	**1.75**	**0.33**
C2	0.33	Carpenter	Hindi	3 → 0.67	**L2**	**0.97**	**1.06**
C3	1	Doctor	Bhojpuri	4 → 1	**L2**	**2.2**	**0.21**
C5	1	Data Scientist	Hindi	2 → 0.33	**L1**	**1**	**1**
C4	**0**	**Data Scientist**	**Hindi**	**2 → 0.33**		**0 (NA)**	

d(C4, C2)
= distance w.r.t to numerical attributes + distance w.r.t Nominal attributes
= Euclidean distance on (Income, Scaled Locality) + Categorical distance on attributes (Profession, Region)

$= \sqrt{(\text{Income}_{c4} \_ \text{Income}_{c2})^2 + (\text{Locality}_{c4} \_ \text{Locaility}_{c2})^2} + (No.of.Categorical\ features - No.of.Matches\ between\ C4\ \&\ C2)/(No.of.Categorical\ Features)$

$= \sqrt{(0-0.33)^2 + (0.33-0.67)^2} + \frac{2-1}{2}$

= 0.47 + 0.5 = 0.97

Exercise - 2

Based on the information given in the table below, find the customer type of x_q=C4 using K-NN. In given example consider all the instances and use distance weighted K-NN algorithm with kernel $K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$.

Apply min-max normalization on income to obtain [0,1] range. Consider profession and Region as nominal. Consider :Locality as ordinal variable with ranking order of [Village, Small Town, Suburban, Metropolitan]. Give equal weight to each attribute.

Customer	Income	Profession	Region	Scaled Locality	Category	Distance	Weight
C0	0.33	Doctor	Hindi	1 → 0	**L1**	**0.97**	**1.06**
C1	0.67	Doctor	Bengali	1 → 0	**L2**	**1.75**	**0.33**
C2	0.33	Carpenter	Hindi	3 → 0.67	**L2**	**0.97**	**1.06**
C3	1	Doctor	Bhojpuri	4 → 1	**L2**	**2.2**	**0.21**
C5	1	Data Scientist	Hindi	2 → 0.33	**L1**	**1**	**1**
C4	**0**	**Data Scientist**	**Hindi**	**2 → 0.33**		**0 (NA)**	

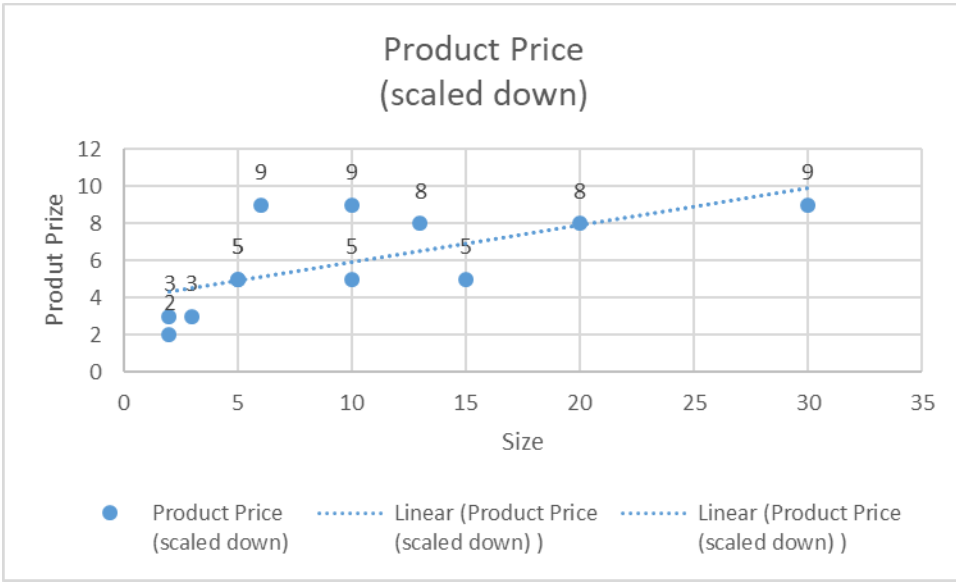
Point	Weight	Class
C5	1	L1
C0	1.06	L1
C2	1.06	L2
C1	0.33	L2
C3	**0.21**	L2

Inference : Without weighted KNN, L2 is the majority voted class. (take sum of count of each class)
But with weighted distances, L1 class instances are more similar to C4 (take sum of weights of each class)

Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(-d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form $y = 3.5 + 0.8\ \text{Size}$ and the learning rate of 0.5 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8

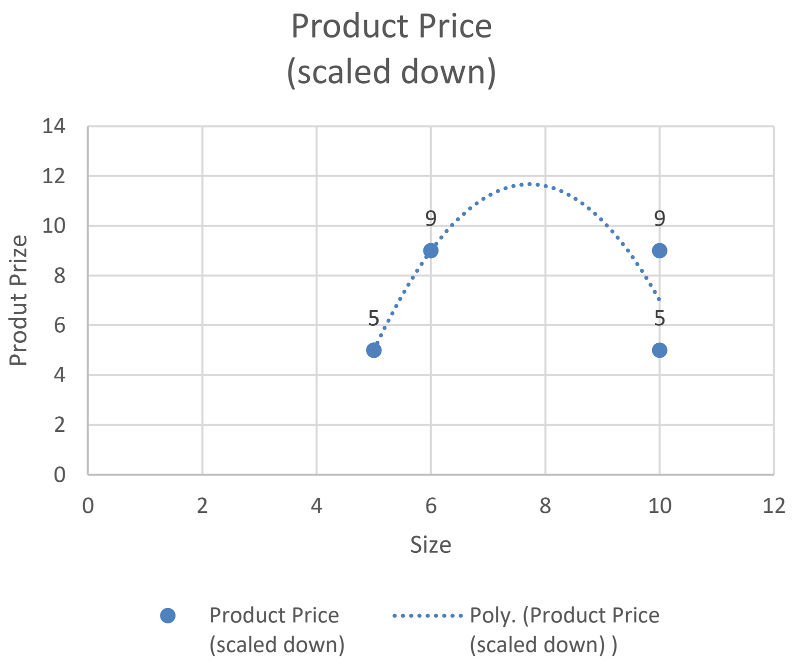


Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(-d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form y = 3.5 + 0.8 Size and the learning rate of 0.5 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)
10	5
5	5
2	2
3	3
5	5
15	5
30	9
20	8
6	9
2	3
10	9
13	8

Considering the 5-NN of the query point x= size = 7



Locally weighted linear regression

regression

- For a given query point $\mathbf{x}_q$ we solve the following weighted regression problem using gradient descent training rule:

$$\boxed{\Delta w_j = \eta \sum_{x \in \, k \; nearest \; nbrs \; of \; x_q} K(d(x_q, x)) \; (f(x) - \hat{f}(x)) \; a_j(x)}$$

This is used in GD formula for calculation of Delta Wi

- Note that we need to solve a new regression problem for *every* query point—that's what it means to be *local*.
- In ordinary linear regression, we solved the regression problem once, globally, and then used the same h_w for any query point.

$$w_i \leftarrow w_i + \Delta w_i$$

Note: + sign in weight update rule as in Δw_i (actual-prediction) is used.

If we use (prediction – actual) then instead of '+', '-' sign will be used.

Exercise - 3

Consider the following training set in 2-dimensional Euclidean space. What is the prize of the product with size 7 if 5-NN is considered? Use the kernel of the form $K(d(x_q,x_i)) = \exp(-d(x_q,x_i)^2 / 2b^2)$ where b=1. Use the linear regression of the form Price = 3 + 0.8 Size and the learning rate of 0.2 to apply the gradient descent and update the weights at the end of first iteration

Size	Product Price (scaled down)	d(xq, xi)	$K(d(x_q,x_i))$
10	5	3	0.01
5	5	2	0.14
2	2	5	
3	3	4	
5	5	2	0.14
15	5	8	
30	9	23	
20	8	13	
6	9	1	0.61
2	3	5	
10	9	3	0.01
13	**8**	6	

Apply gradient descent where the gradient is weighed by the kernel output. Initial values w0=3, w1=0.8

Answer :
Updated weights after 1st iteration
w1 = 0.95, w0= 3.02

Price Prediction	9.699758	After 1st iteration of GD
	8.6	Before GD

Thank You

31 August 2023