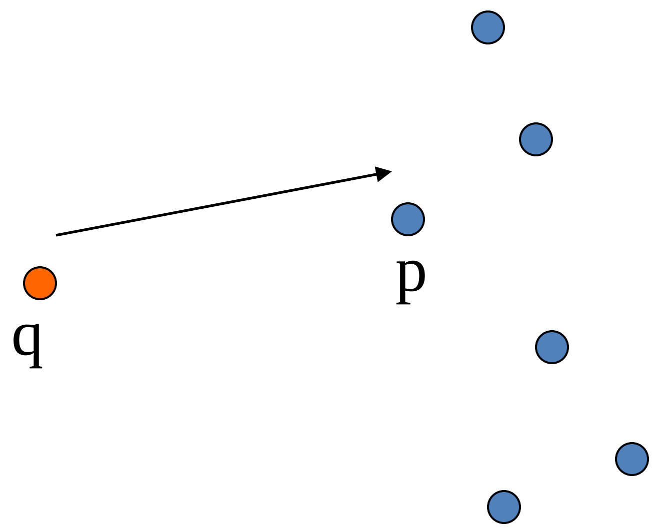
Classification Using K-Nearest Neighbor

Nearest Neighbor and Exemplar

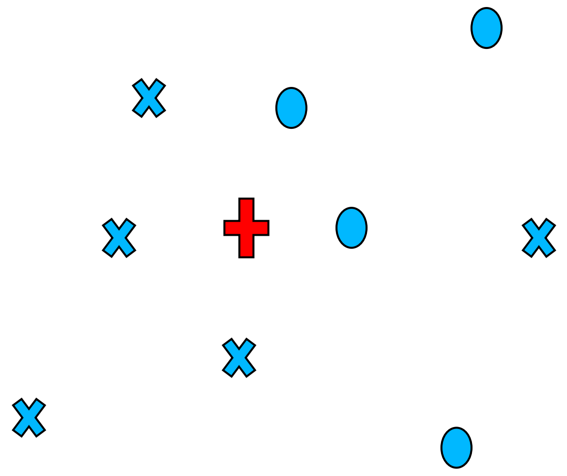
Nearest Neighbor Search

- Given: a set P of n points in R^d
- Goal: a data structure, which given a query point q, finds the *nearest neighbor* p of q in P



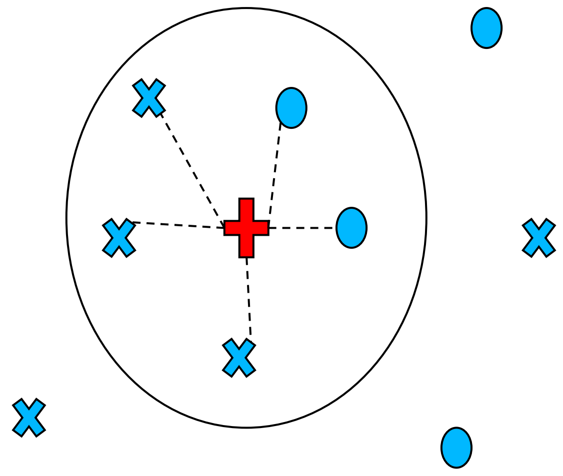
K-NN

K=5



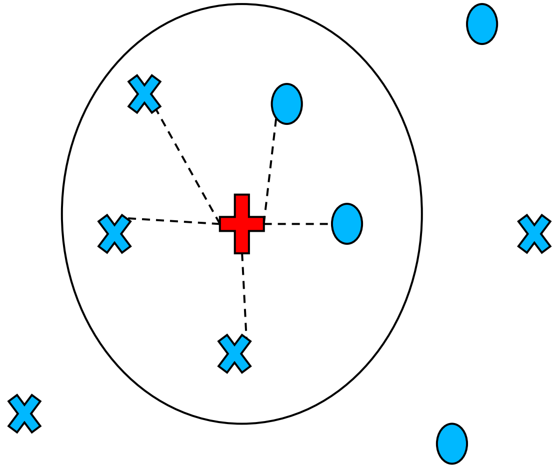
K-NN

- Select 5 Nearest Neighbors as Value of K=5 by Taking their Euclidean Distances



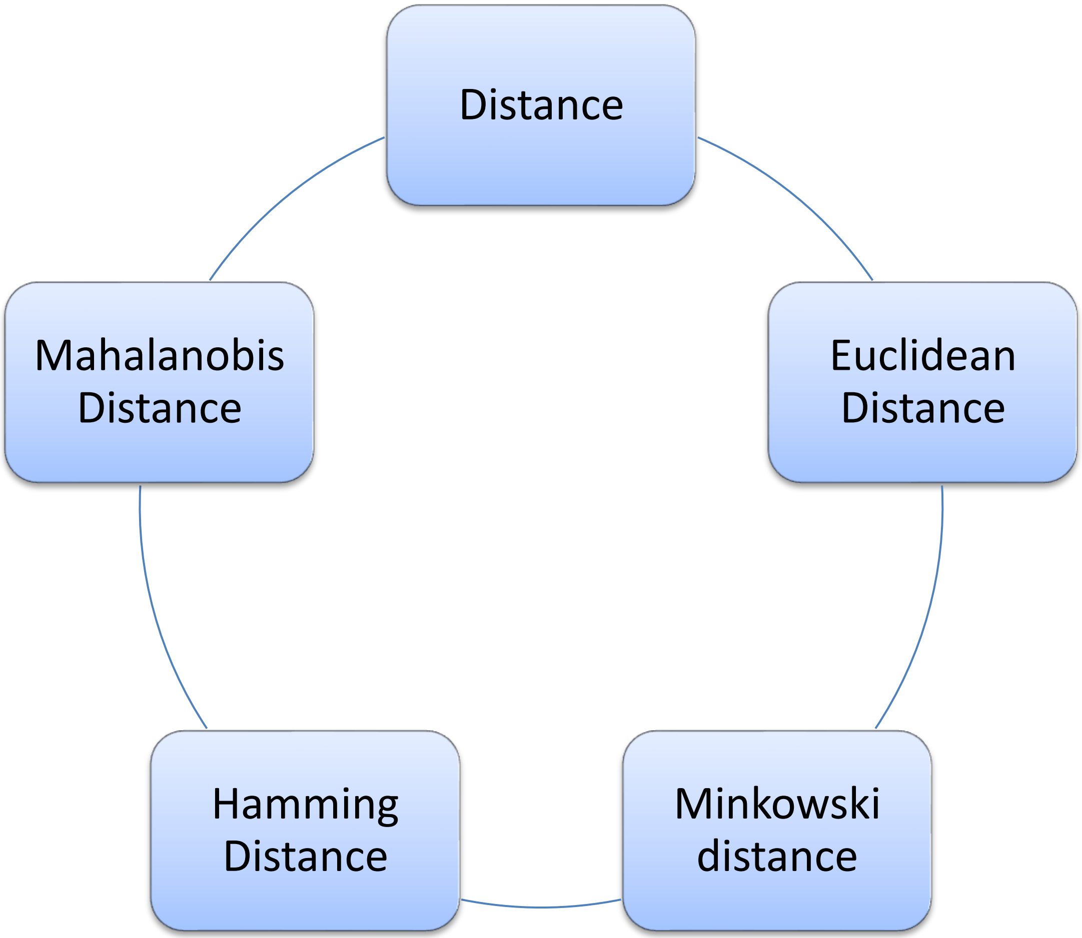
K-NN

- Decide if majority of Instances over a given value of K Here, K=5.



Distances

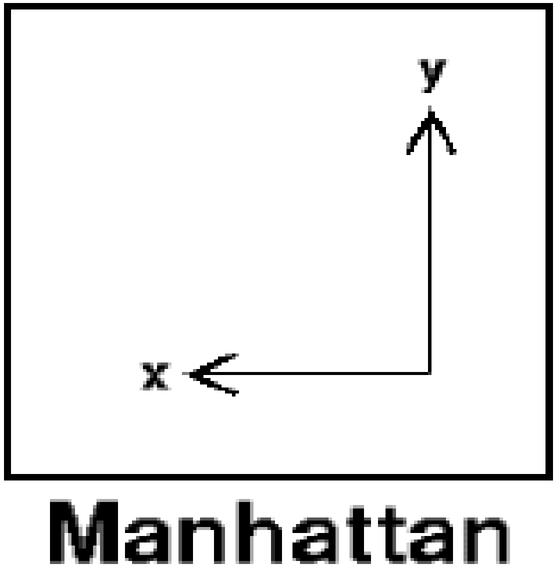
- Distance are used to measure similarity
- There are many ways to measure the distance s between two instances



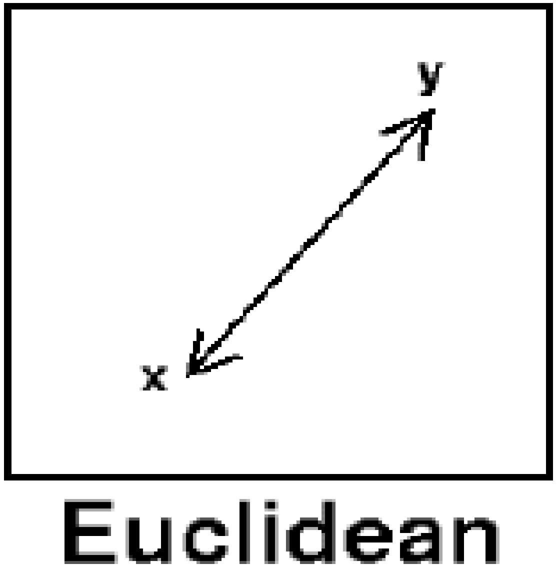
Distances

- Manhattan Distance

|X1-X2| + |Y1-Y2|



- Euclidean Distance
- $\sqrt{(x1-x2)^2} + \sqrt{(y1-y2)^2}$



Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \left(\sum_{k=1}^{p} |a_k - b_k|^r \right)^{1/r}$$

Where r is a parameter, *p* is the number of dimensions (attributes) and a_k and b_k are, respectively, the k-th attributes (components) or data objects a and b

Minkowski Distance: Examples

- r = 1. City block (Manhattan, taxicab, L1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- r = 2. Euclidean distance
- r $\rightarrow \infty$. "supremum" (L_{max} norm, L_{∞} norm) distance.
  - This is the maximum difference between any component of the vectors
- Do not confuse r with p, i.e., all these distances are defined for all numbers of dimensions.

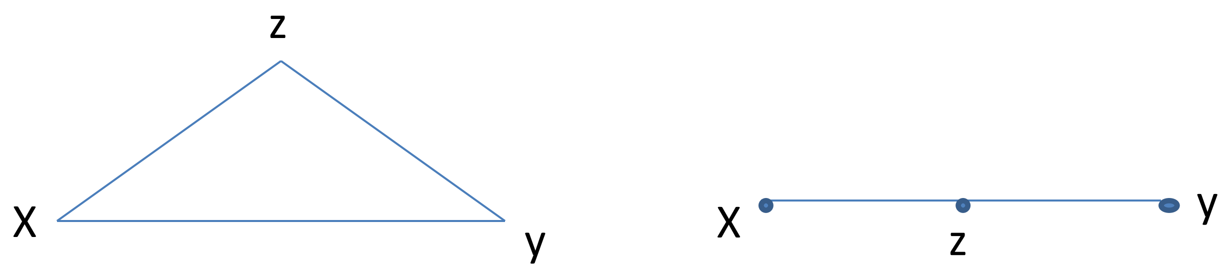
Cosine Similarity

- $\cos(d_1, d_2) = \begin{cases} 1: \text{exactly the same} \\ 0: \text{orthogonal} \\ -1: \text{exactly opposite} \end{cases}$

Properties of Distance

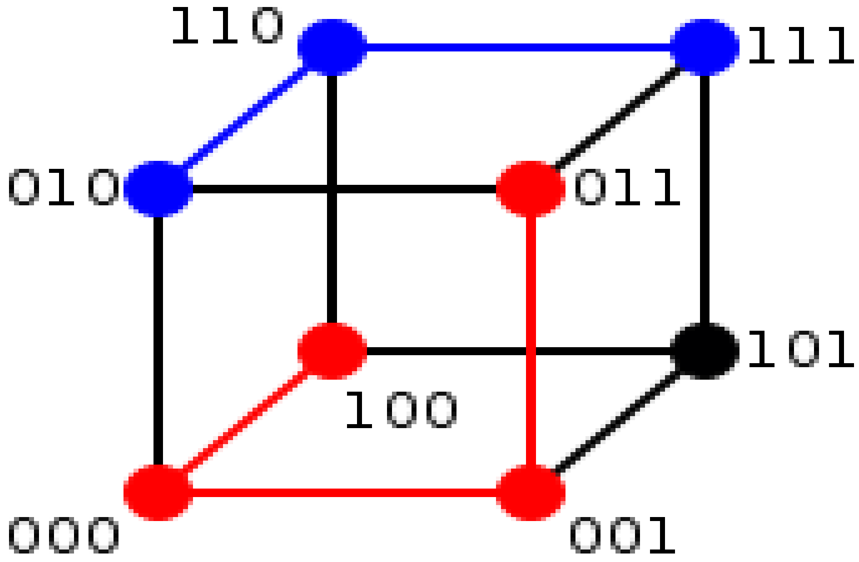
- Dist (x,y) >= 0
- Dist (x,y) = Dist (y,x) are Symmetric
- Detours can not Shorten Distance

Dist(x,z) <= Dist(x,y) + Dist (y,z)



Distance

Hamming Distance



Distances Measure

- Distance Measure – What does it mean “Similar"?
- Minkowski Distance
  - Norm: $$d(x,y) = \| x - y \|_m = \left[\sum_{i=1}^{N} (x_i - y_i)^m \right]^{1/m}$$
  - Chebyshew Distance
  - Mahalanobis distance:

$$d(x\,,y) = |x - y|^T S_{xy}^{-1} |x - y|$$

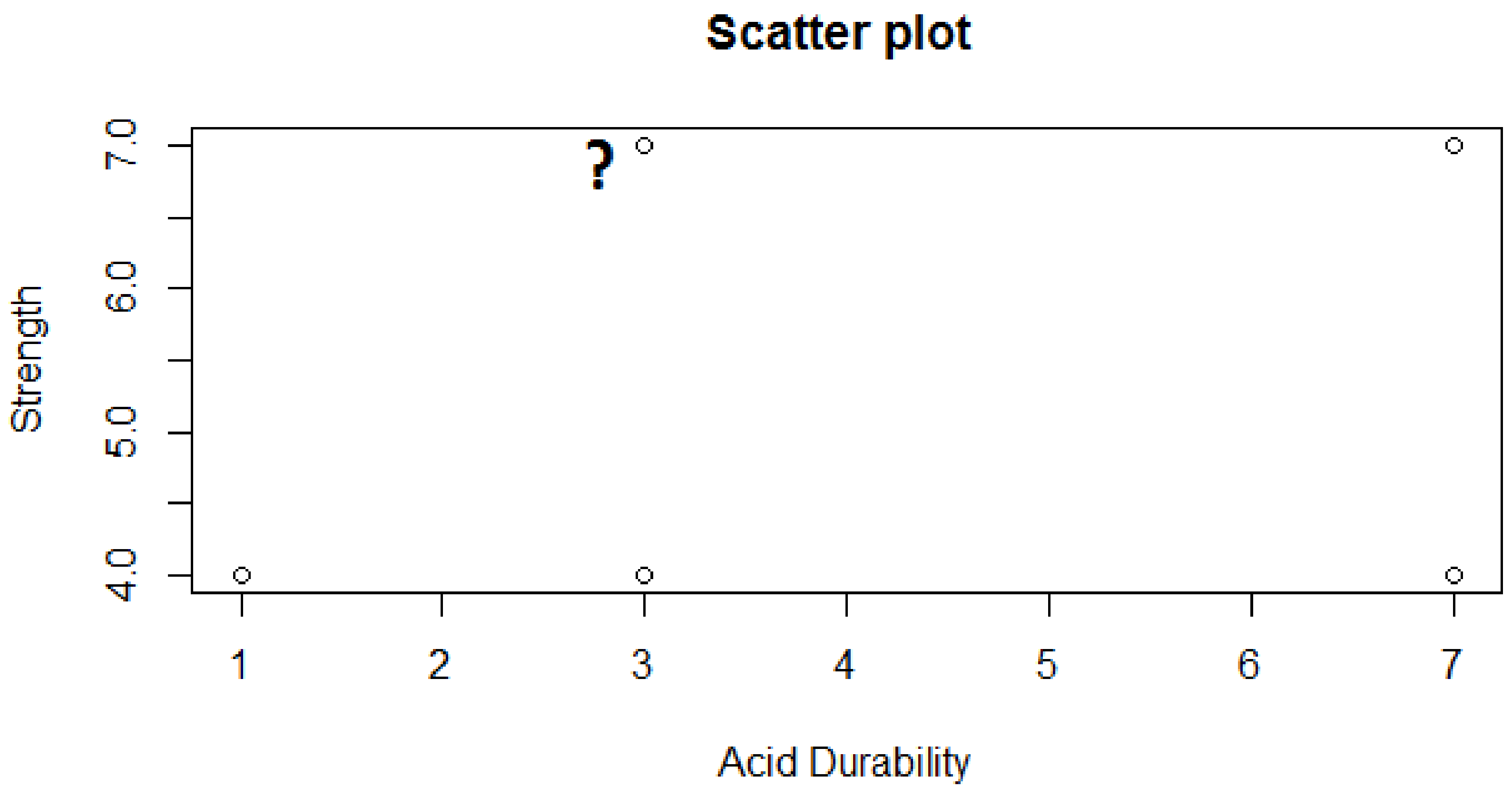
Example

Points	X1 (Acid Durability)	X2(strength)	Y=Classification
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD

KNN Example

Points	X1(Acid Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	?

Scatter Plot



Euclidean Distance From Each Point

KNN				
Euclidean Distance of P5(3,7) from	P1	P2	P3	P4
	(7,7)	(7,4)	(3,4)	(1,4)
	Sqrt($(7-3)^2 + (7-7)^2$) = $\sqrt{16}$ $= 4$	Sqrt($(7-3)^2 + (4-7)^2$) = $\sqrt{25}$ $= 5$	Sqrt($(3-3)^2 + (4-7)^2$) = $\sqrt{9}$ $= 3$	Sqrt($(1-3)^2 + (4-7)^2$) = $\sqrt{13}$ $= 3.60$

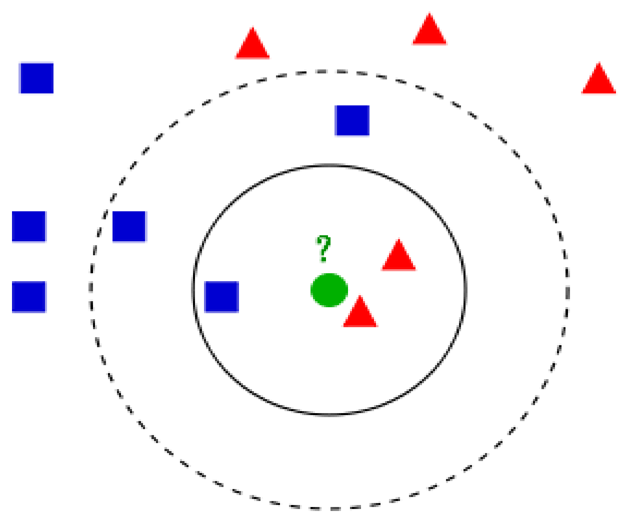
3 Nearest NeighBour

Euclidean Distance of P5(3,7) from	P1	P2	P3	P4
	(7,7)	(7,4)	(3,4)	(1,4)
	Sqrt((7-3) ^{2} + (7-7)^2) = $\sqrt{16}$ $= 4$	Sqrt((7-3) ^{2} + (4-7)^2) = $\sqrt{25}$ $= 5$	Sqrt((3-3) ^{2} + (4-7)^2) = $\sqrt{9}$ $= 3$	Sqrt((1-3) ^{2} + (4-7)^2) = $\sqrt{13}$ $= 3.60$
Class	BAD	BAD	GOOD	GOOD

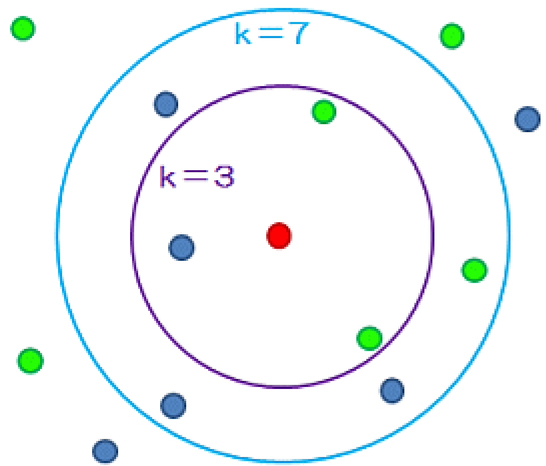
KNN Classification

Points	X1(Durability)	X2(Strength)	Y(Classification)
P1	7	7	BAD
P2	7	4	BAD
P3	3	4	GOOD
P4	1	4	GOOD
P5	3	7	GOOD

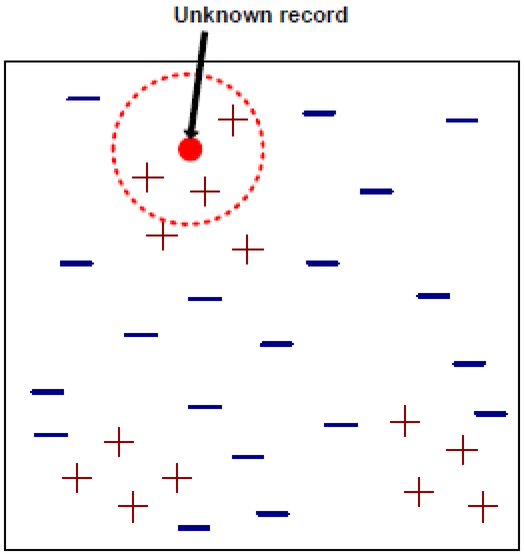
Variation In KNN



Different Values of K



Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

The KNN classification algorithm

Let k be the number of nearest neighbors and D be the set of training examples.

1. for each test example z = ($\mathbf{x}'$,y') **do**

2. Compute d($\mathbf{x}'$,$\mathbf{x}$), the distance between z and every example, ($\mathbf{x}$,y) ϵ D

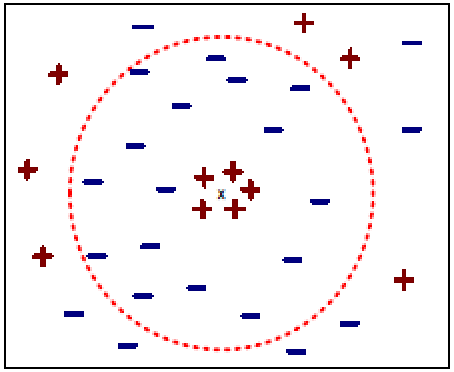
3. Select $D_z \subseteq D$, the set of k closest training examples to z.

4. y' = $\underset{v}{\mathrm{argmax}} \sum_{(x_i,y_i) \in D_z} I(v = y_i)$

5. end for

Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 60 KG to 100KG
    - income of a person may vary from Rs10K to Rs 2 Lakh

Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality: all vectors are almost equidistant to the query vector
  - Can produce undesirable results

1 1 1 1 1 1 1 1 1 1 1 0	vs	1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1		0 0 0 0 0 0 0 0 0 0 0 1
d = 1.4142		d = 1.4142

◆ Solution: Normalize the vectors to unit length

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

Choosing appropriate k

- Deciding how many neighbors to use for kNN determines how well the mode will generalize to future data.
- The balance between overfitting and underfitting the training data is a problem known as the bias-variance tradeoff.
- Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner such that it runs the risk of ignoring small, but important patterns.

Choosing appropriate k

- In practice, choosing k depends on the difficulty of the concept to be learned and the number of records in the training data.
- Typically, k is set somewhere between 3 and 10. One common practice is to set k equal to the square root of the number of training examples.
- In the classifier, we might set k = 4, because there were 15 example ingredients in the training data and the square root of 15 is 3.87.

The kNN Algorithm

Strengths	Weaknesses
• Simple and effective • Makes no assumptions about the underlying data distribution • Fast training phase	• Does not produce a model, which limits the ability to find novel insights in relationships among features • Slow classification phase • Requires a large amount of memory • Nominal features and missing data require additional processing