How to classify a document?

●●

USC Viterbi School of Engineering

Rocchio Algorithm: Basics

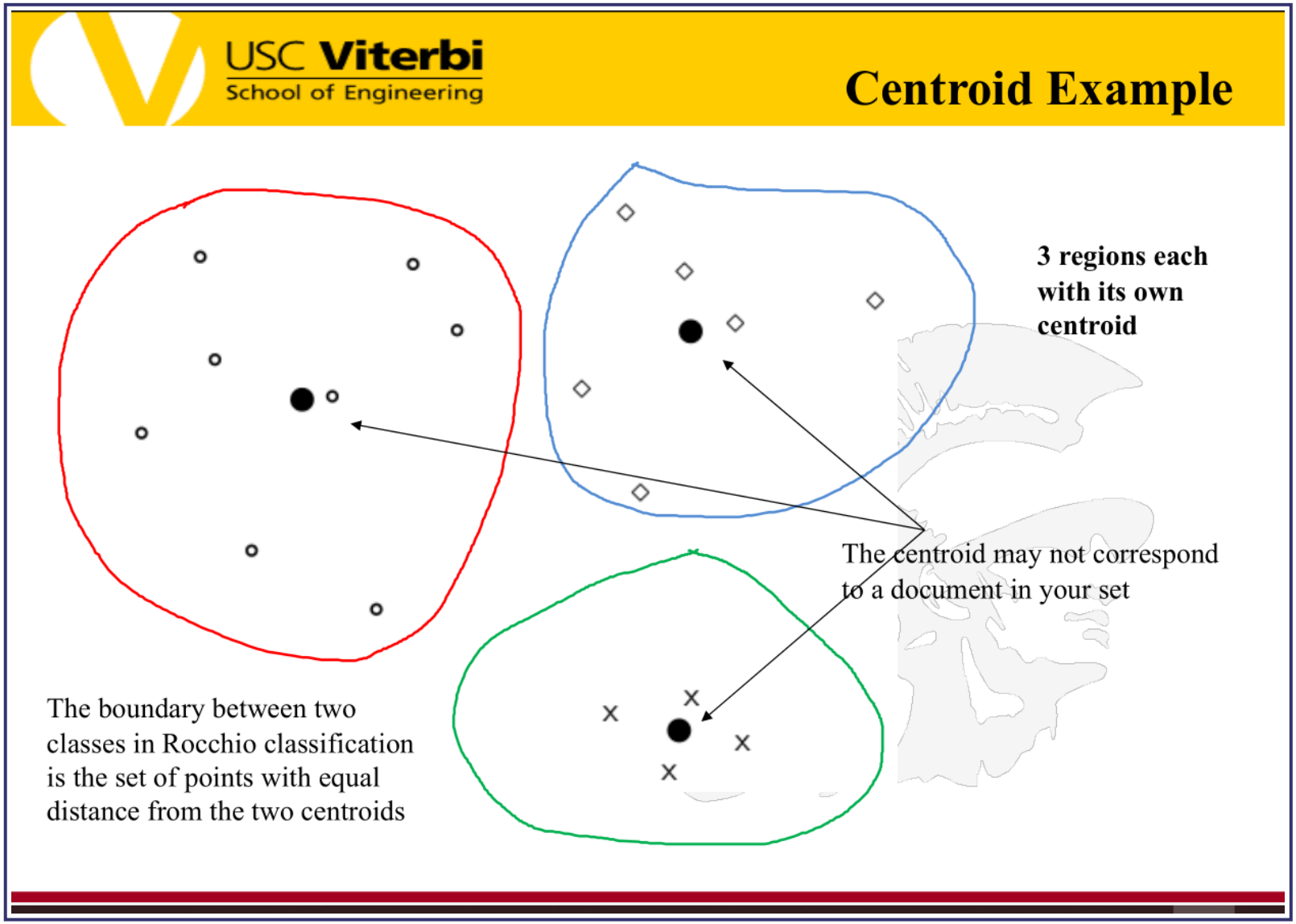
- The Rocchio algorithm is a method of relevance feedback
- It was initially developed by the SMART Information Retrieval System in 1960-1964.
- It assumes documents are represented using the vector space model
- The algorithm uses the notions of relevant/non-relevant documents and centroids
- Recall: the centroid is the center of mass (or vector average) of a set of points.
- *Definition*: Centroid

$$\vec{\mu}(C) = \frac{1}{|C|}\sum_{d \in C} \vec{d}$$

where C is a set of documents, |C| is the size of the set, and d is the normalized vector representing document d

- **Note**: We have seen centroids before in the k-means algorithm

●●


USC Viterbi
School of Engineering
Centroid Example
3 regions each with its own centroid
The centroid may not correspond to a document in your set
The boundary between two classes in Rocchio classification is the set of points with equal distance from the two centroids

●●

USC Viterbi School of Engineering

Rocchio Algorithm Derivation

Assuming someone has identified the set of relevant (D_r) and non-relevant (D_{nr}) documents, the algorithm aims to find the query q that maximizes similarity with the set of relevant documents D_r while minimizing similarity with the set of non-relevant documents D_{nr} :

$$q_{opt} = arg\ max\ [sim(q, D_r) - sim(q, D_{nr})]$$

Under cosine similarity, the optimal query for separating relevant and non-relevant documents is:

$$q_{opt} = 1/|D_r| \sum_{d_j\ in\ D_r} d_j - 1/|D_{nr}| \sum_{d_j\ in\ D_{nr}} d_j$$

which is the vector difference between the centroids of the relevant and non-relevant documents.

●●

USC Viterbi School of Engineering

Rocchio Algorithm for Relevance Feedback - in Practice

- In practice, however, we usually do not know the full set of relevant and non-relevant sets.
- For example, a user might only label a few documents as relevant / non-relevant.
- Therefore, in practice Rocchio is often parameterised as follows:

$$\vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

where q is the original query vector; D_r and D_n are the sets of known relevant and non-relevant documents.

α, β, and γ are weight parameters attached to each component.

Reasonable values are $\alpha = 1.0$, $\beta = 0.75$, $\gamma = 0.15$

•Note: if the final vale of q_m has negative term weights, set those to 0.

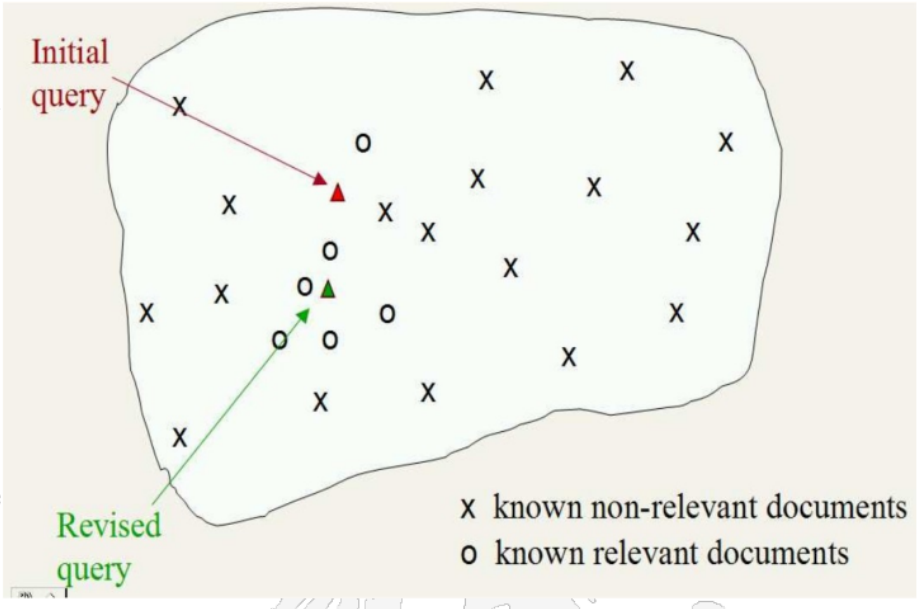
 17

●●

USC Viterbi School of Engineering

Rocchio in Practice

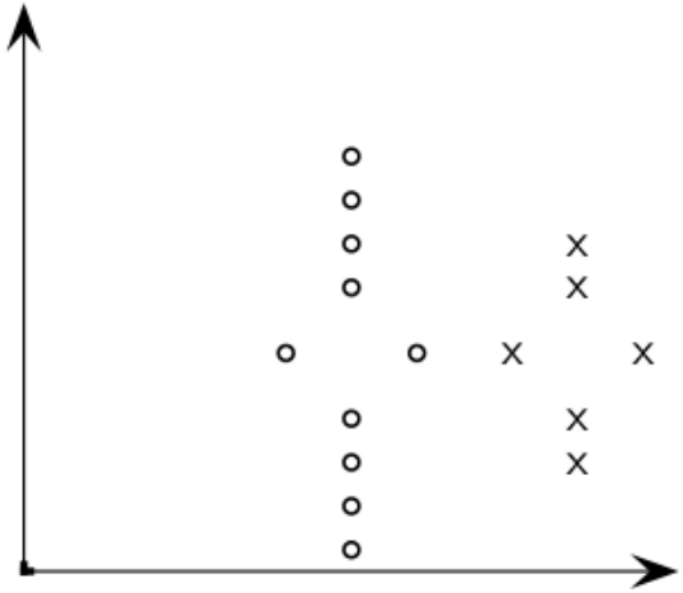
- Represent query and documents as weighted vectors (e.g., tf–idf).
- Use Rocchio formula to compute new query vector (given some known relevant / non-relevant documents).
- Calculate cosine similarity between new query vector and the documents.
- Rocchio has been shown useful for increasing both precision and recall because it contains aspects of positive and negative feedback.
- Positive feedback is much more valuable than negative (i.e., indications of what *is* relevant) so typically systems set $\gamma < \beta$ or even $\gamma = 0$.



●●

USC Viterbi School of Engineering

2D Rocchio Example



- For 2D examples the relevant set is generally much smaller than the non-relevant set;
- As a result we need a slightly modified rule

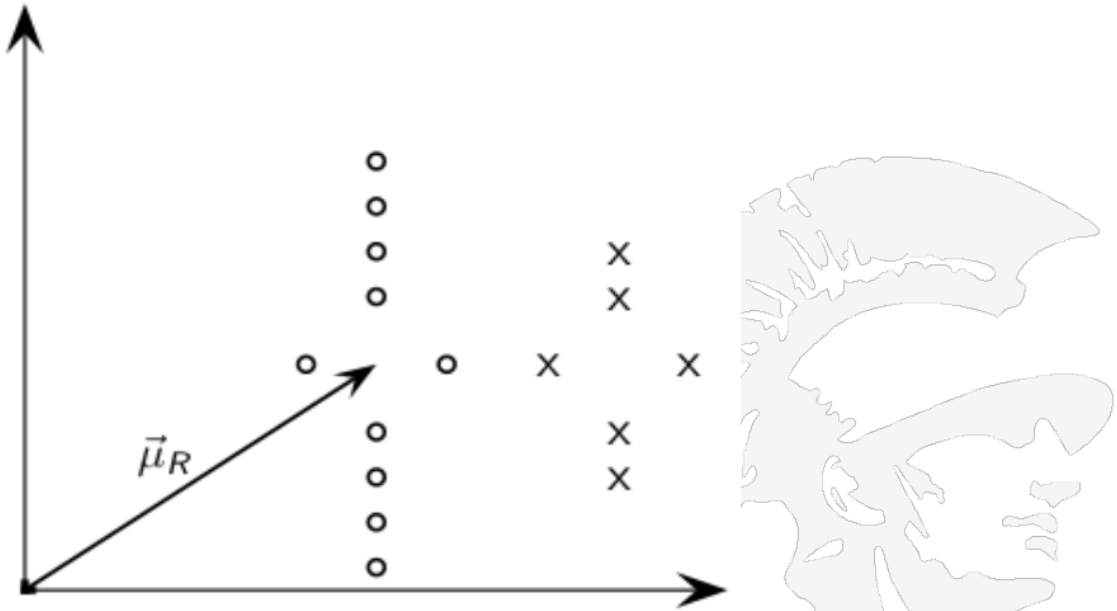
Let circles represent relevant documents,
Let Xs represent nonrelevant documents

19

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated (1 of 9)



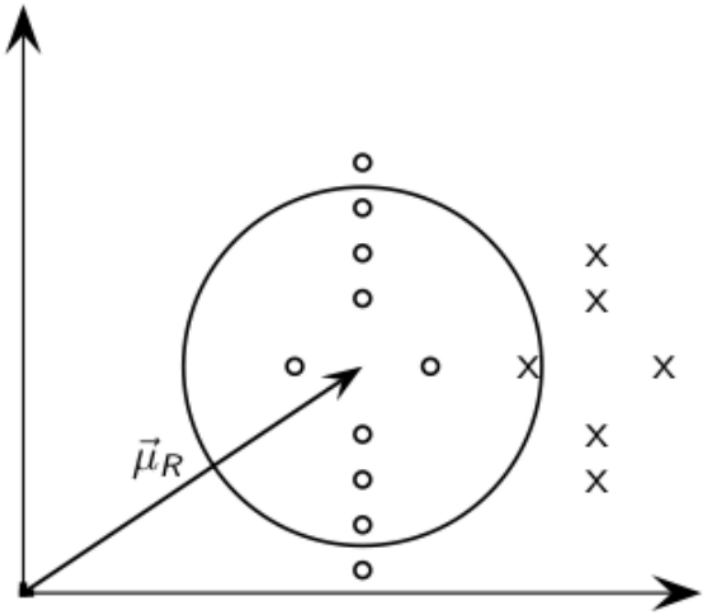
$\vec{\mu}_R$: centroid of relevant documents

20

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated (2 of 9)



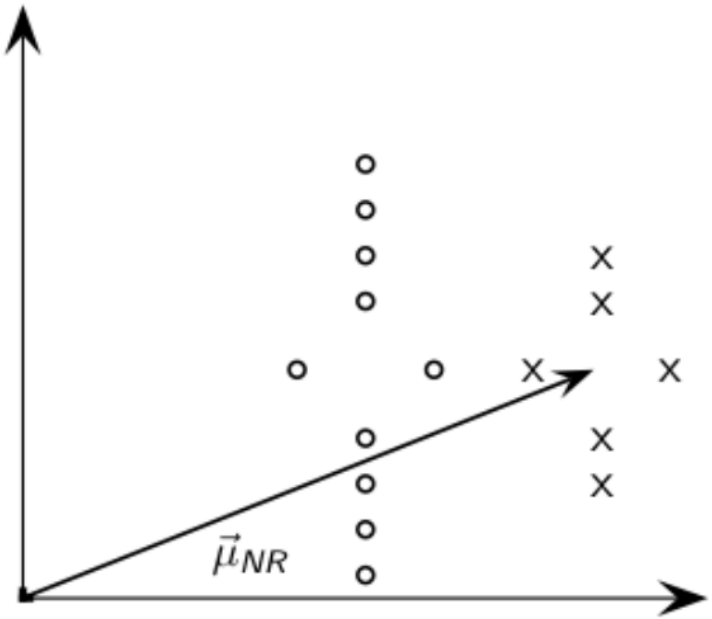
$\vec{\mu}_R$ does not separate relevant / nonrelevant.

21

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated (3 of 9)



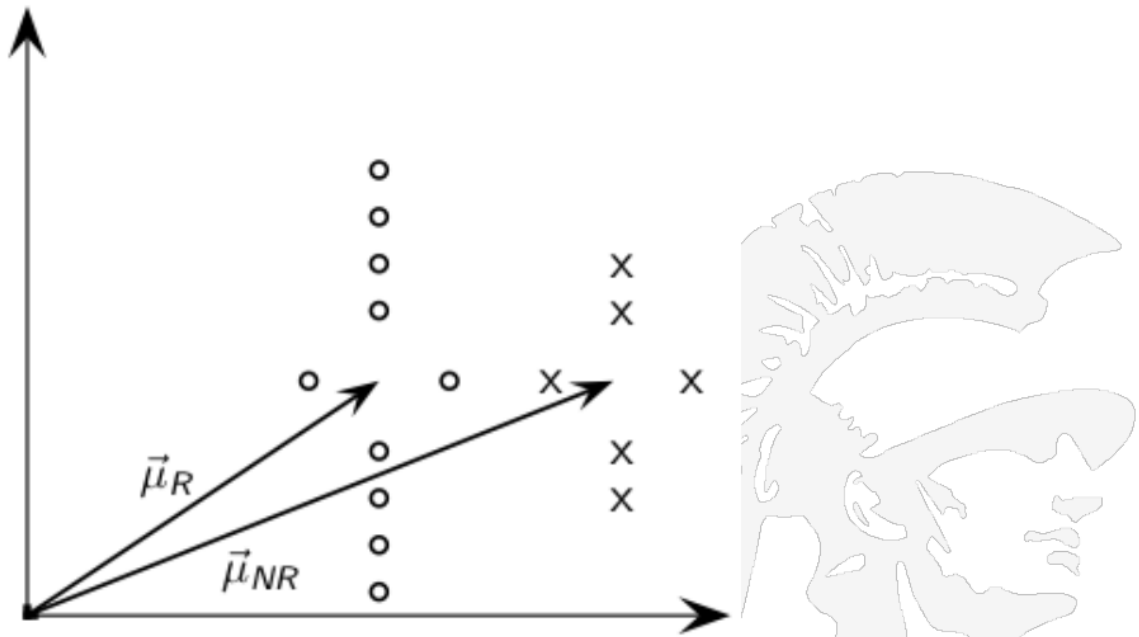
$\vec{\mu}_{NR}$: centroid of nonrelevant documents.

22

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated (4 of 9)



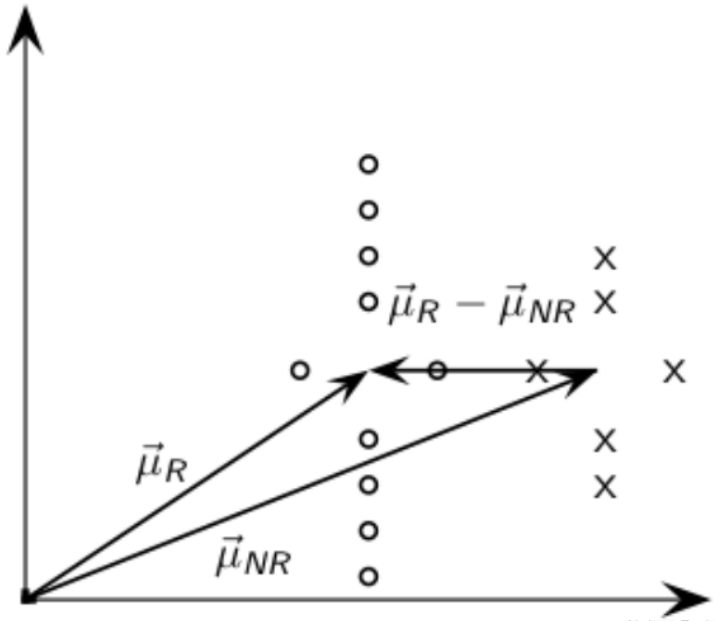
Consider the two centroids

23

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated(5 of 9)



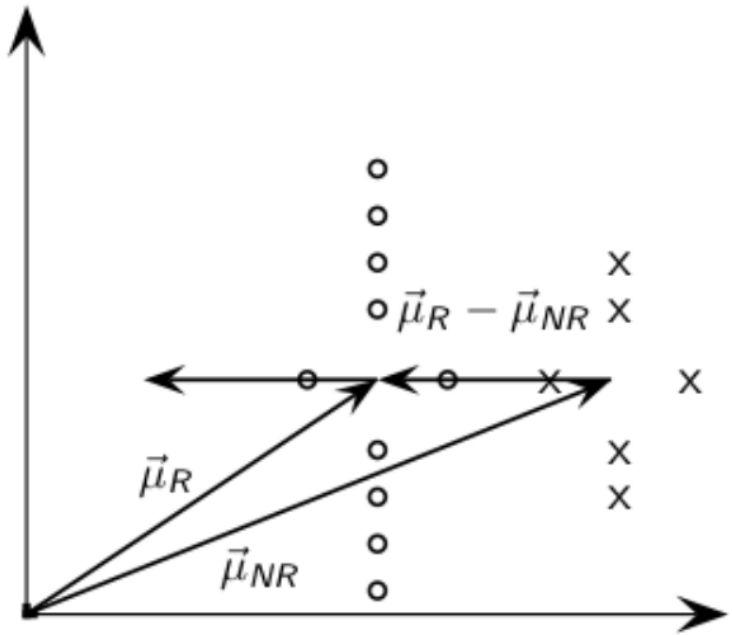
$\vec{\mu}_R$ - $\vec{\mu}_{NR}$: centroid difference vector

24

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated(6 of 9)



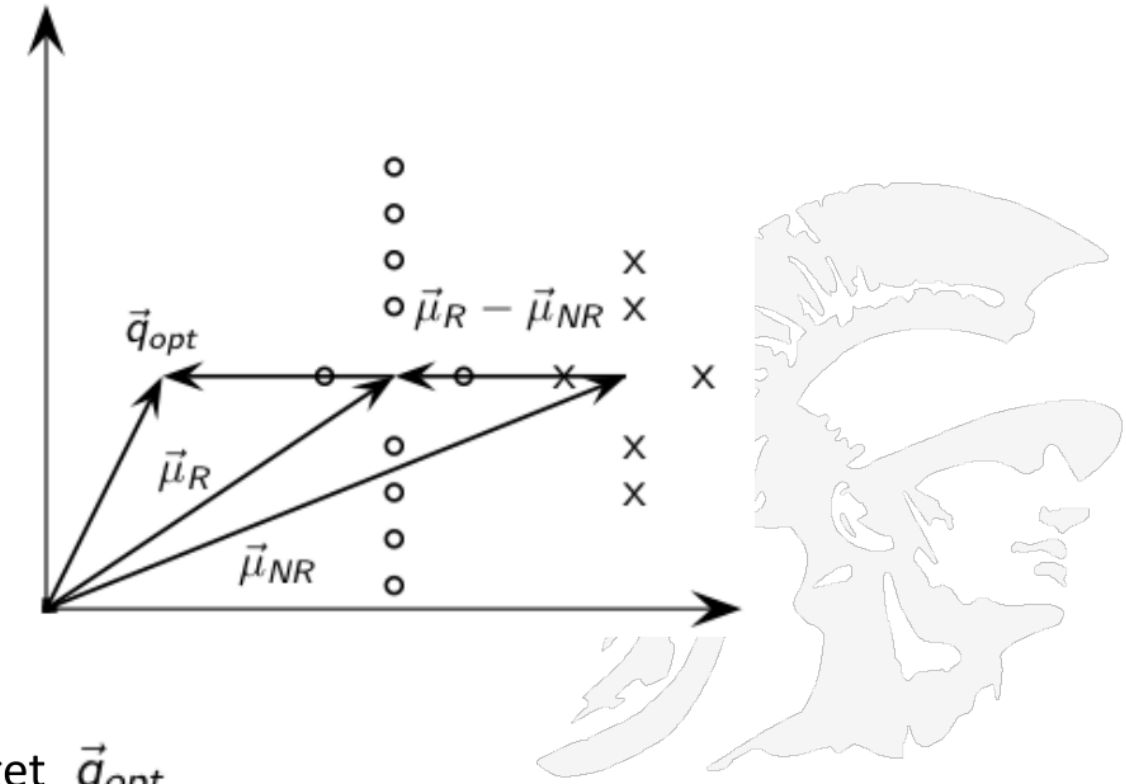
Add difference vector to $\vec{\mu}_R$ …

25

••

USC Viterbi School of Engineering

2D Rocchio Illustrated(7 of 9)



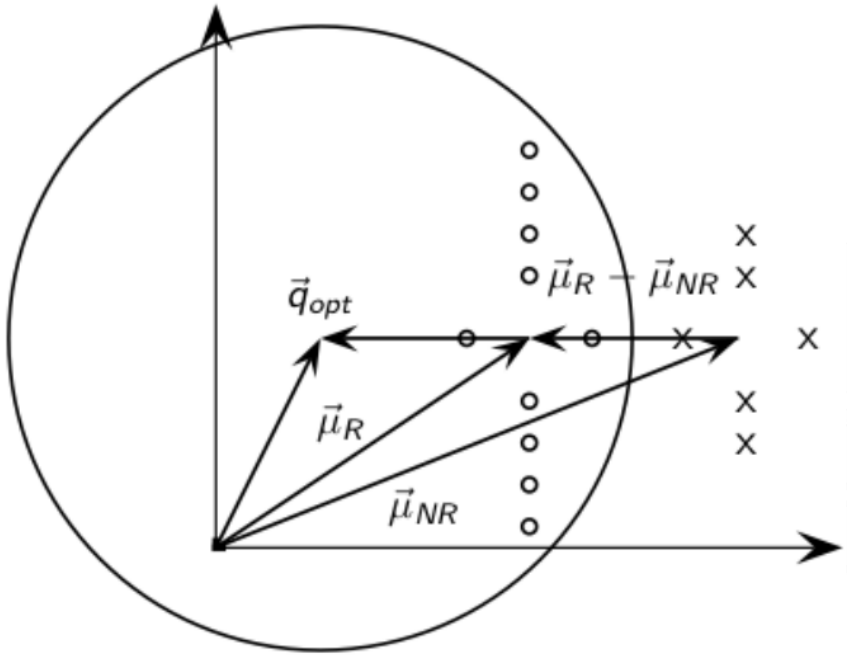
... to get $\vec{q}_{opt}$

26

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated(8 of 9)



Note that the boundary computed during the Rocchio algorithm in This case is viewed as a circle;

Tests of new documents are easily determined to either fit within the circle or not

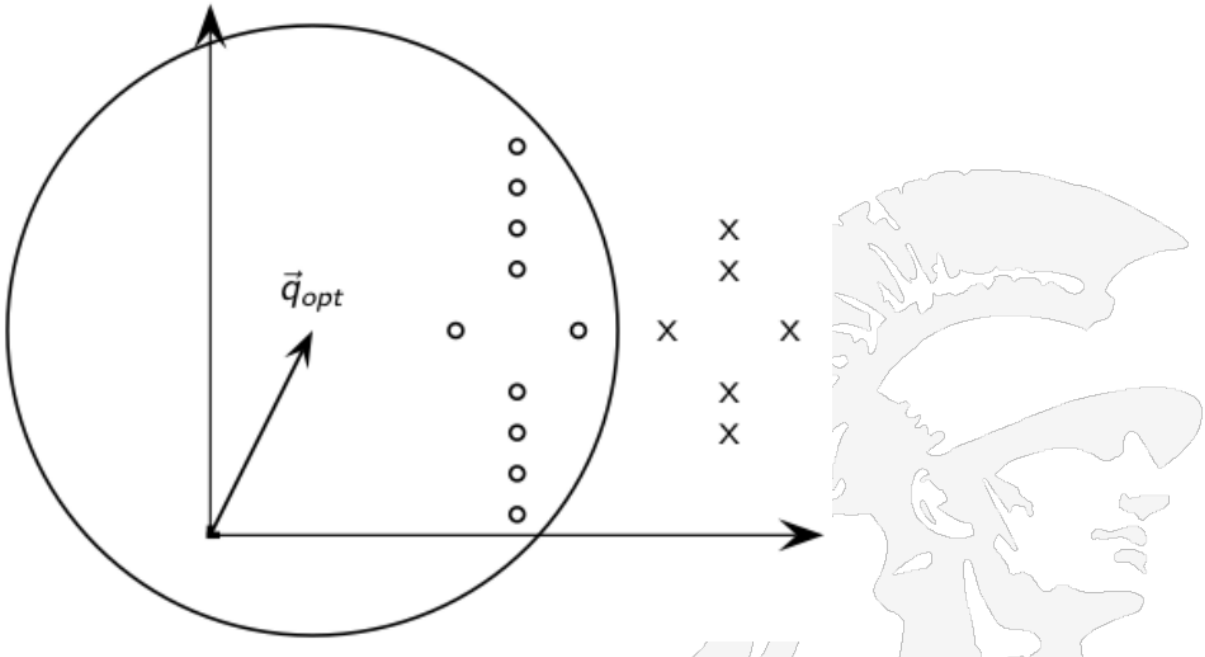
$\vec{q}_{opt}$ now separates relevant / nonrelevant perfectly.

27

●●

USC Viterbi School of Engineering

2D Rocchio Illustrated(9 of 9)



$\vec{q}_{opt}$ separates relevant / nonrelevant perfectly.

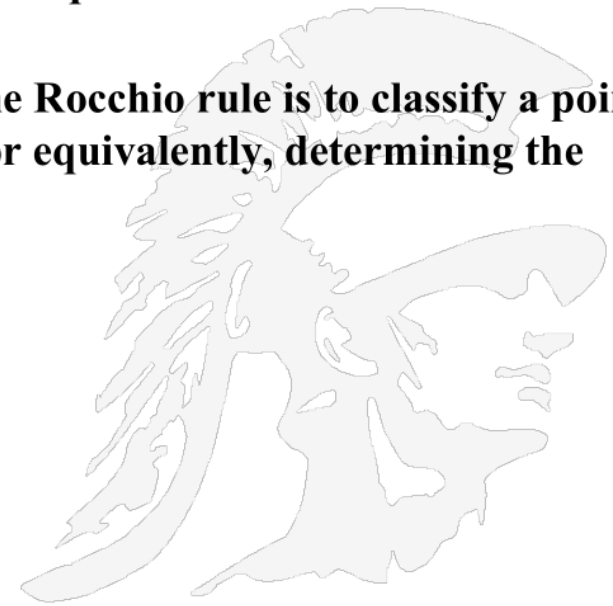
28

●●

USC Viterbi School of Engineering

Rocchio Algorithm Used for Classification

- More typically, the boundary determination in Rocchio is not a circle, but a hyperplane
- Given two centroids of two classes of documents, the boundary between the two classes is the set of points with equal distance from the two centroids
- Once the boundary is determined, the Rocchio rule is to classify a point according to the region it falls into, or equivalently, determining the centroid that the point is closest to



••

USC Viterbi School of Engineering

Classification is Different from Clustering

- In general, in ***classification*** you have a set of predefined classes and want to know which class a new object (document) belongs to.
- Remember, ***Clustering*** tries to group a set of objects and find whether there is *some* relationship between the objects.
  - we already saw two algorithms for clustering, K-Means Algorithm and Agglomerative Clustering algorithm
- In the context of machine learning, classification is *supervised learning* and clustering is *unsupervised learning*
  - **Clustering** requires a. an algorithm, b. a similarity measure, and c. a number of clusters
  - **classification** has each document labeled in a class and an algorithm that assigns documents to one of the classes

••

USC Viterbi School of Engineering

Classification Methods

- **Manual classification**
  - Used by the original Yahoo! Directory
  - **Other search engines did similar things, e.g.**
    - Looksmart, http://www.looksmart.com/
    - Open Directory Project, https://www.dmoz.org/
    - PubMed, http://www.ncbi.nlm.nih.gov/pubmed
  - **Accurate when job is done by experts**
  - **Consistent when the problem size and team is small**
  - **Difficult and expensive to scale**
    - Means we need automatic classification methods for big problems

••

USC Viterbi School of Engineering

The Problem Statement for Classification

- **Given two things:**
  1. **A description of an instance, $x \in X$, where X is the *instance language* or *instance space*, and**
  2. **A fixed set of categories:**

     $C = \{c_1, c_2, \ldots, c_n\}$
- **Determine:**
  - **The category of x: $c(x) \in C$, where $c(x)$ is a *categorization function* whose domain is X and whose range is C.**
    - **Functions that categorize are called "classifiers"**

••

USC Viterbi School of Engineering

Classification Using Vector Spaces

- **In vector space classification, the training set corresponds to a labeled set of document vectors**
- **Premise 1: Documents in the same class form a contiguous region of space**
  - This is referred to as the *contiguity hypothesis*
- **Premise 2: Documents from different classes don't overlap (much)**
- **Learning a classifier is equivalent to building (defining) surfaces to delineate classes in the space**

●●

USC Viterbi School of Engineering

Ways to Measure Distance

For normalized vectors Euclidean distance and cosine similarity correspond

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

••

USC Viterbi School of Engineering

k Nearest Neighbor Classification Algorithm

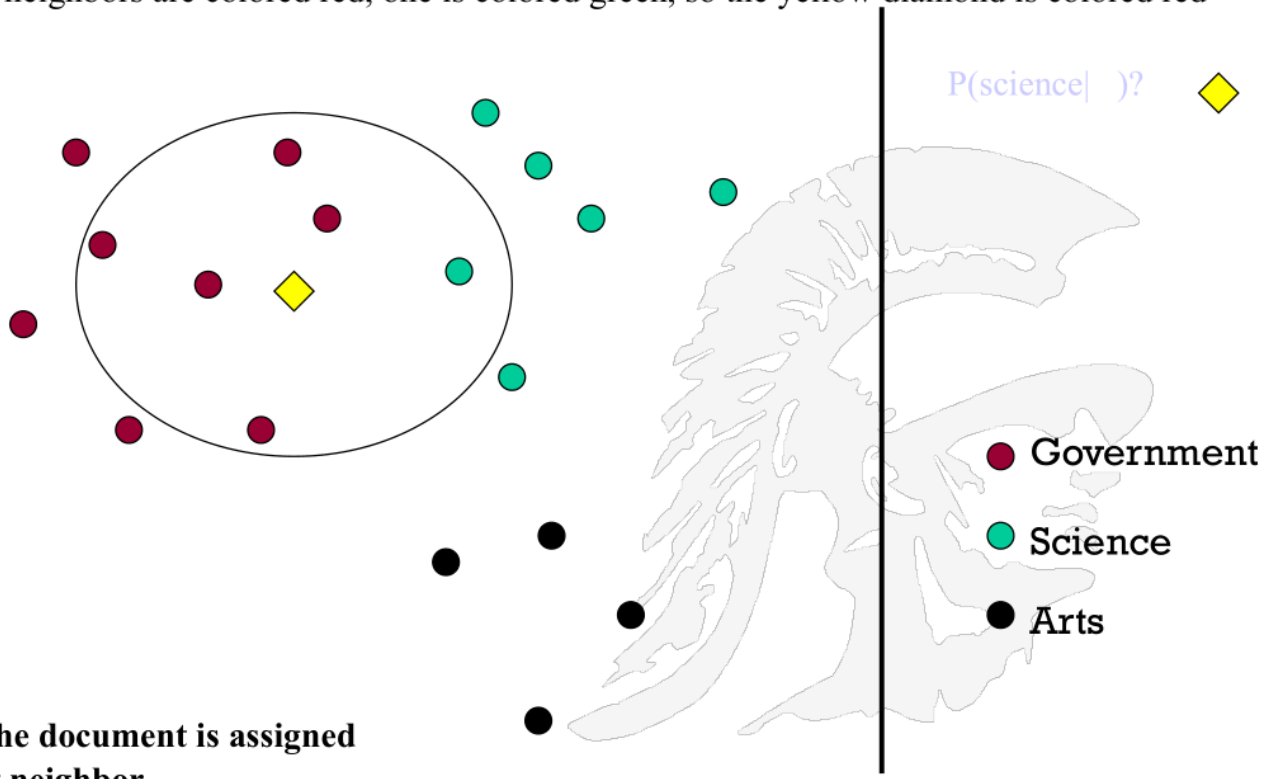
- **Initially we assume we have a set of *N* documents that have already been classified**
  - the WDM videos assume the class names are colors (see the Schedule of Lectures)
- **To classify a document *d***
  - locate among the *N* documents the *k* closest ones
  - from these *k* neighbors, pick the class that occurs most often, the majority class, and use that as the label for *d*

●●

USC Viterbi School of Engineering

Example: *k=6 (6NN)*

5 neighbors are colored red, one is colored green, so the yellow diamond is colored red

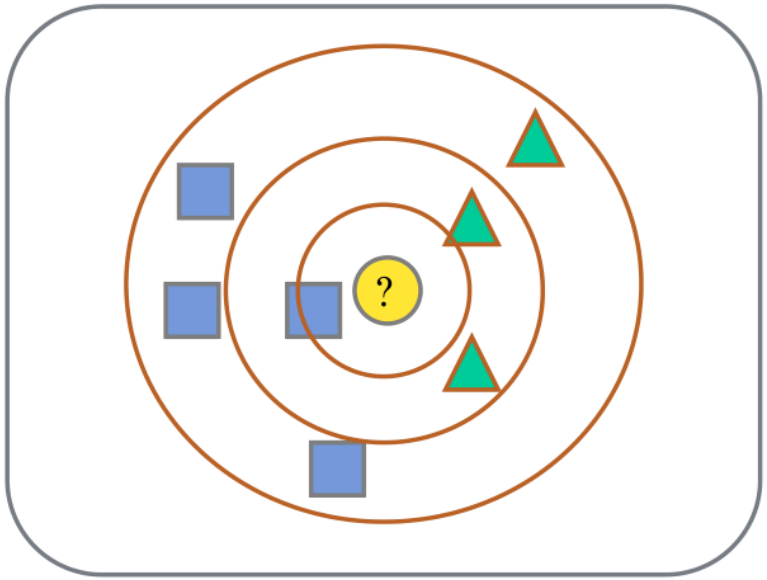


When k=1, the document is assigned to its nearest neighbor

●●

USC Viterbi School of Engineering

K-Nearest Neighbor Another Example



- $k = 1$:
  - Belongs to square class
- $k = 3$:
  - Belongs to triangle class
- $k = 7$:
  - Belongs to square class

- **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**
  - **Choose an odd value for *k*, to eliminate ties**

••

USC Viterbi School of Engineering

Nearest-Neighbor Without Learning

- **Learning: there is no learning step; just store the labeled training examples *D***
- **Testing instance *x (under 1-NN)*:**
  - Compute the distance between *x* and all examples in *D*.
  - Assign *x* the category of the closest example in *D*.
- **Rationale of *k-NN*: contiguity hypothesis**
  - if documents do form contiguous regions in space, then *k-NN* makes sense
- **Does not compute anything beyond storing the examples**
  - we are NOT determining any hierarchy or model
- ***K-NN* has also been called:**
  - Case-based learning
  - Memory-based learning
  - Lazy learning

●●

USC Viterbi School of Engineering

Choice of *K*

- **The best choice of *k* depends upon the data;**
  - generally, larger values of *k* reduce the effect of noise on the classification, but make boundaries between classes less distinct.
- The accuracy of the *k-NN* algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance
- In binary (two class) classification problems, (e.g. male or female) it is helpful to choose *k* to be an odd number as this avoids tied votes
- Choosing the optimal value for k is best done by first inspecting the data
- Cross-validation is another way to retrospectively determine a good value of K by using an independent dataset to validate the K value
- Historically, the optimal K for most datasets has been between 3-10

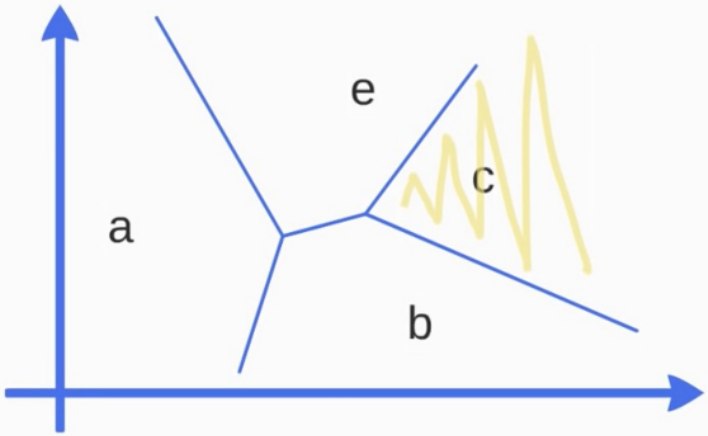
●●

USC Viterbi School of Engineering

Voronoi Diagram

For the k-Nearest Neighbor Algorithm, k = 1 is a special case

When *k* = 1, each training vector defines a region in space, defining a *Voronoi* partition of the space



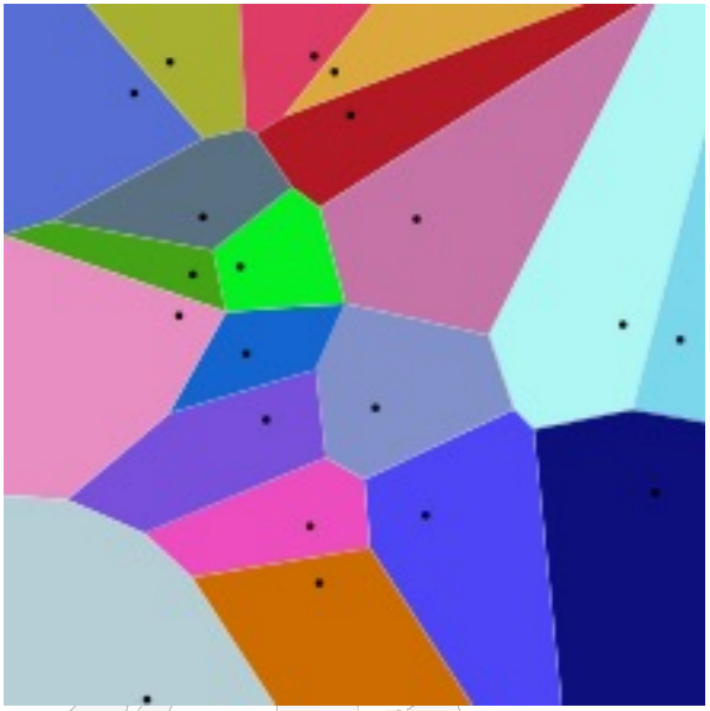
$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\}$$

●●

USC Viterbi School of Engineering

When k=1 – A Special Case

- A **Voronoi diagram** is a partitioning of a plane into regions based on distance to points in a specific subset of the plane
- Decision boundaries in *1-NN* are concatenated segments of a Voronoi tessellation (e.g. polygons)
- The set of points (called class labels) is specified beforehand
- For each class label there is a corresponding region consisting of all points closer to that class label than to any other. These regions are called Voronoi cells



20 points (class labels) and their Voroni regions; Line segments are all points equidistant to three or more regions

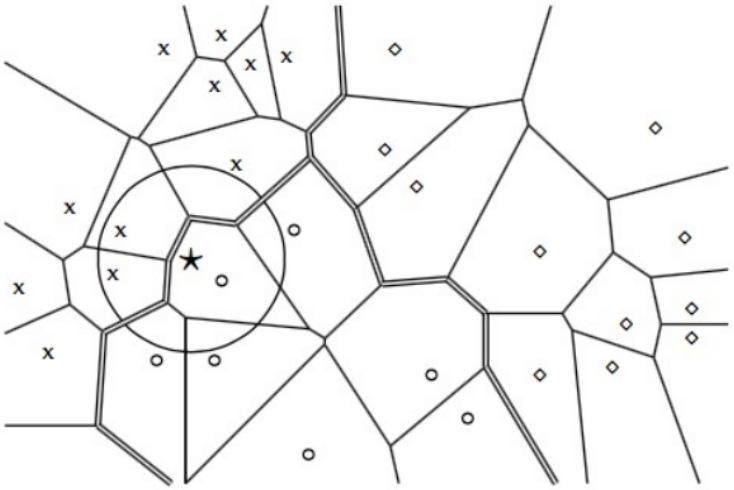
●●

USC Viterbi School of Engineering

K=1 Nearest Neighbor Regions are Polygons

- **For *1-NN* we assign each document to the class of its closest neighbor**
- **For *k-NN* we assign each document to the majority class of its *k* closest neighbors where *k* is a parameter**

The two classes are: X and circle, and the star document is falling into the circle area;
Double lines define the regions in space where documents are similar;
think of each region as defining a cellphone tower



***K-NN* is an example of a non-linear classifier; (Rocchio is a linear classifier)**

••

USC Viterbi School of Engineering

K-NN: Final Points

- **No feature selection necessary**
- **No training necessary**
- **Scales well with large number of classes**
  - Don't need to train *n* classifiers for *n* classes
- **Classes can influence each other**
  - Small changes to one class can have ripple effect
- **In most cases it's more accurate than Rocchio**