



USC Viterbi
School of Engineering

Introduction to

Relevance Feedback Query Expansion And Classification

Some slides are from Helen Yannakoudakis and Ronan Cummins of Univ. of Cambridge



- **Examples of Relevance Feedback**
- **Query Expansion examples and techniques for producing relevance feedback**
- **Rocchio Algorithm for Relevance Feedback (under ideal conditions, i.e. relevant and non-relevant documents are known)**
- **Rocchio Algorithm for Classification including an online version**
- **K-Nearest Neighbor Algorithm for Classification**

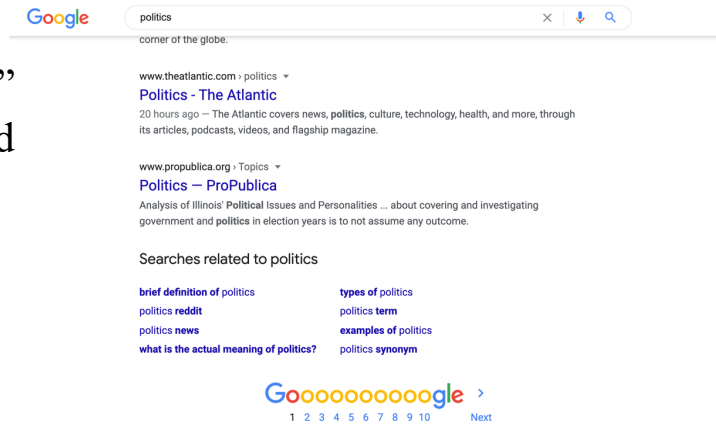


Relevance Feedback

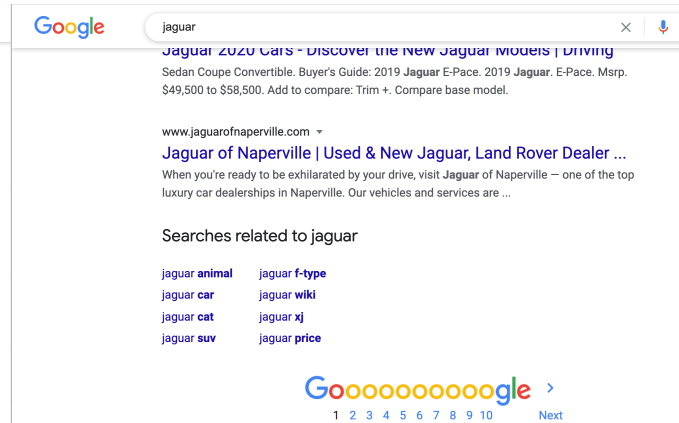
- **Idea:** The idea is to involve the user in the retrieval process by having the user give feedback about which results are most relevant
- The idealized process of user feedback of relevance is to improve recall
 - User issues a (short, simple) **query**
 - The user marks some results as relevant/non-relevant.
 - The system computes a better representation of the information need based on feedback.
 - New results have (hopefully) better **recall**.
 - Relevance feedback can go through one or more iterations
- **E.g. Google and Bing provide relevance feedback without user intervention**
 - However, their RSS feeds permit users to specify relevant and non-relevant results, thus improving the value of the feed

Examples where Relevance Feedback Would be Useful

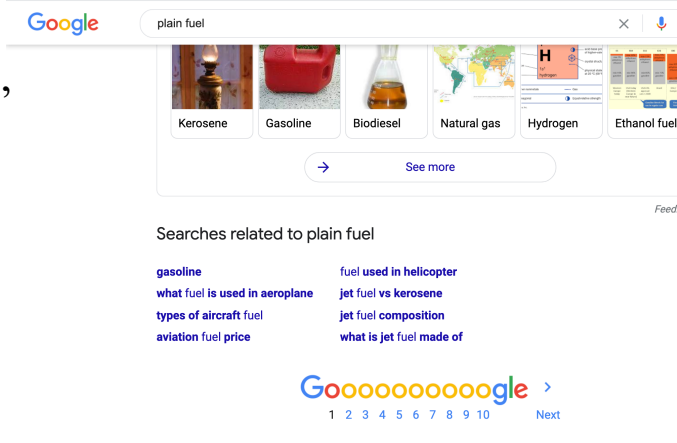
“politics”
unrefined
query



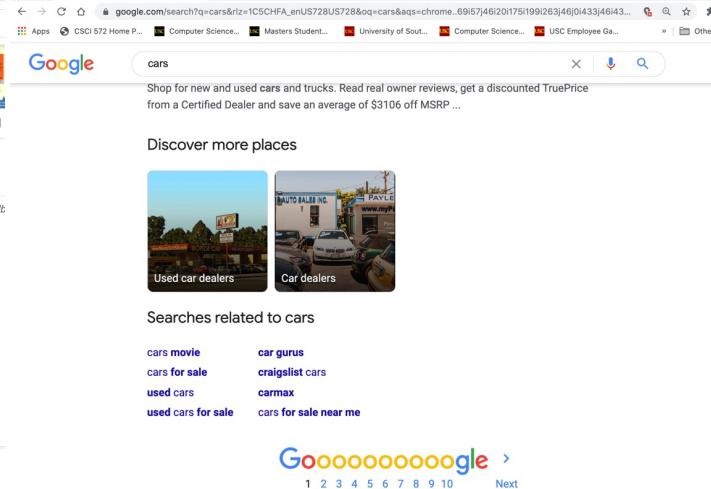
“jaguar”
ambiguous



“plain fuel”
Two
interpretations
are possible



“cars”
unrefined

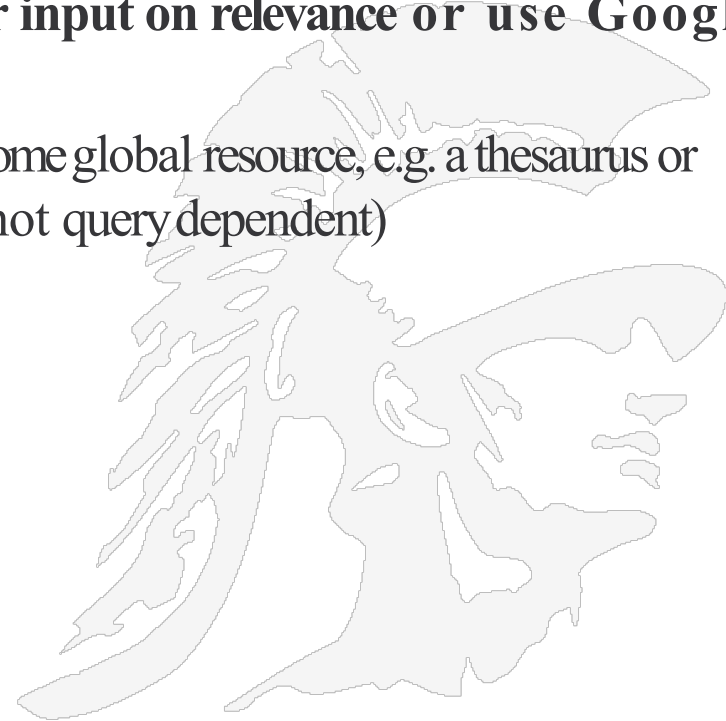


- Relevance feedback and query expansion aim to overcome these problems.



Relevance Feedback will Improve Recall

- **Methods for tackling this problem split into two classes**
 1. **Local methods:** adjust a query relative to the documents returned (query-time analysis on a portion of documents)
 - **Main local method: either get user input on relevance or use Google's approach of "Did you mean?"**
 2. **Global methods:** adjust query based on some global resource, e.g. a thesaurus or knowledge graph (i.e., a resource that is not query dependent)



Google Uses Query Expansion in its Search Results

- Google uses six techniques for expanding a query to make it more meaningful

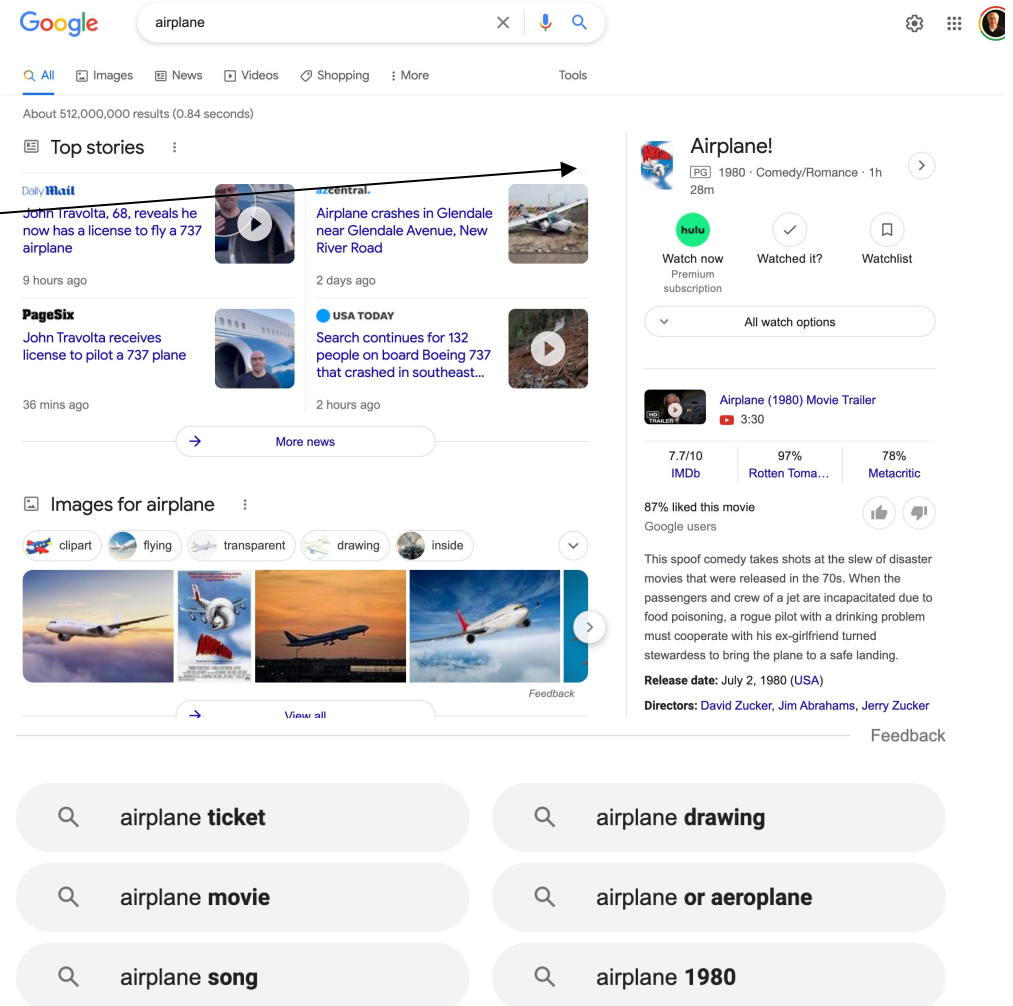
Google's Query Expansion	Explanation and observations	
Word stemming	The term searched is reduced to a root or 'stem'- words of the same stem can be ranked for the search containing only one of them.	
Acronyms	An abbreviation / acronym / initialism searched is automatically resolved to the full version. If there are several possible variants of one and the same acronym, Google will do its best to mention all variations on the first page in the following order: (1) most popular and hence most probable one (2) all the rest.	6
Misspellings	With misspelled words searched, Google will suggest the correct variant as well as list sites that use that correct variant. Note: based on common webmasters' <u>experience</u> , Google uses some other (unclear) algorithm for typos : sites ranked for the misspelled word (using it nowhere on the page or in the text of backlinks) may have low authority and rank nowhere if the correct spelling is searched.	1
Synonyms	Sometimes Google includes related words in the search results. Very often the query is extended this way when it is obvious the phrase was used improperly . The synonym substituting the part of the phrase is not bolded as a rule.	
Translations	In some instances, Google seems to translate search keywords into other languages and return results from that language.	
Ignored words	Occasionally some (" insignificant ") words appear to have been dropped completely from the query.	1



Query Expansion Example 1

- Query expansion is used to increase precision and recall.
- The query “airplane” primarily produces a set of movie results

- query expansion which Google calls Feedback will expand the query making use of the knowledge graph



The screenshot shows a Google search for "airplane". The search results are categorized into "Top stories", "Images for airplane", and "Airplane! (1980) Movie Trailer". The "Top stories" section includes news items from Daily Mail, PageSix, and USA TODAY. The "Images for airplane" section shows various images of airplanes. The "Airplane! (1980) Movie Trailer" section includes a video player and movie details. At the bottom, there are six query expansion suggestions: "airplane ticket", "airplane drawing", "airplane movie", "airplane or aeroplane", "airplane song", and "airplane 1980".

Google airplane

About 512,000,000 results (0.84 seconds)

Top stories

Daily Mail John Travolta, 68, reveals he now has a license to fly a 737 airplane 9 hours ago

PageSix John Travolta receives license to pilot a 737 plane 36 mins ago

USA TODAY Search continues for 132 people on board Boeing 737 that crashed in southeast... 2 hours ago

Images for airplane

clipart flying transparent drawing inside

airplane ticket airplane drawing airplane movie airplane or aeroplane airplane song airplane 1980

Airplane! 1980 · Comedy/Romance · 1h 28m

Watch now Watched it? Watchlist

Watch now Premium subscription

All watch options

Airplane (1980) Movie Trailer 3:30

7.7/10 IMDb 97% Rotten Toma... 78% Metacritic

87% liked this movie Google users

This spoof comedy takes shots at the slew of disaster movies that were released in the 70s. When the passengers and crew of a jet are incapacitated due to food poisoning, a rogue pilot with a drinking problem must cooperate with his ex-girlfriend turned stewardess to bring the plane to a safe landing.


Release date: July 2, 1980 (USA)

Directors: David Zucker, Jim Abrahams, Jerry Zucker



Feedback

Query Expansion: Example 2

- The query is “palm”
- The carousel of choices includes restaurants, hotels all named “palm”
- The related searches include references to palm trees, palm pdas and palm hotels



palm

✕  

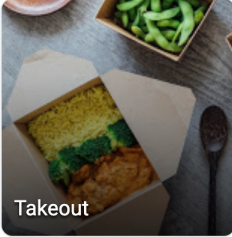
www.verizon.com > Connected Devices > Palm ▾

Palm Companion Phone with NumberShare - Small Size ...


Sync your **Palm** with any IOS or Android device to ensure you stay in touch while on the go. About the size of a credit card, **Palm** packs a huge punch. Unleash ...

★★★★★ Rating: 5 · 10 reviews · \$14.58 · In stock

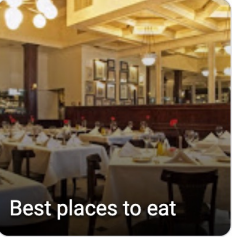
Discover more places



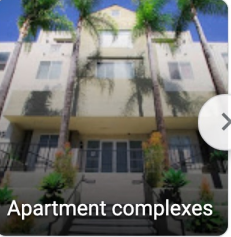
Takeout



Delivery



Best places to eat



Apartment complexes

Searches related to palm

palm **tree**

palm **hand**

palm **phone**


palm **device**

palm, inc

palm **companion**

palm **phone 2**

palm **pda**



1 2 3 4 5 6 7 8 9 10 [Next](#)



Once Again We Look at WordNet to Implement Query Expansion

- **One way to perform query expansion is to use WordNet**
- **A more detailed database of semantic relationships between English words.**
- **Developed by famous cognitive psychologist George Miller and a team at Princeton University.**
- **About 144,000 English words.**
- **Nouns, adjectives, verbs, and adverbs grouped into about 109,000 synonym sets called *synsets*.**





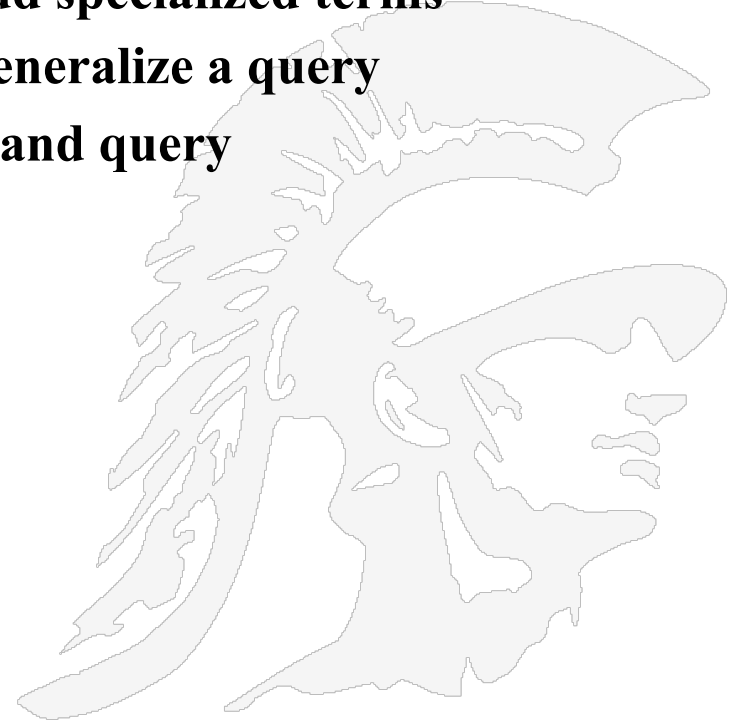
WordNet Synset Relationships

- **Antonym**: front → back (opposite)
- **Attribute**: benevolence → good (noun to adjective)
- **Pertainym**: alphabetical → alphabet (relating to; adjective to noun)
- **Similar**: unquestioning → absolute (resembling)
- **Cause**: kill → die
- **Entailment**: breathe → inhale (a necessary part)
- **Holonym**: chapter → text (part to whole)
- **Meronym**: computer → cpu (whole to part)
- **Hyponym**: plant → tree (specialization)
- **Hypernym**: apple → fruit (generalization)



Query Expansion Using WordNet

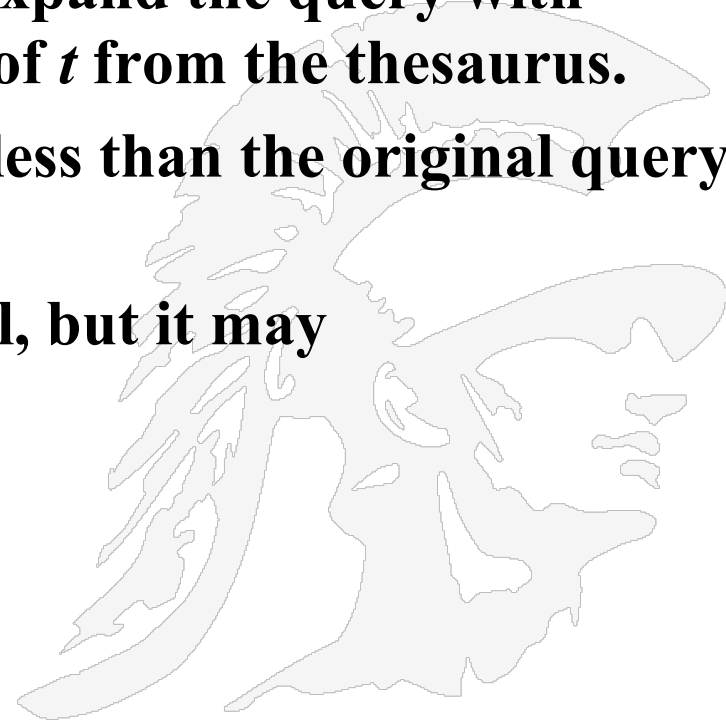
- **Form an expanded query by:**
 1. **Adding WordNet synonyms in the same synset**
 2. **Adding WordNet hyponyms to add specialized terms**
 3. **Adding WordNet hypernyms to generalize a query**
 4. **Adding other related terms to expand query**





Query Expansion Using Thesaurus

- Another way to do query expansion is to make use of a thesaurus
- For each term, t , in a query, expand the query with synonyms and related words of t from the thesaurus.
- One can weight added terms less than the original query terms.
- This generally increases recall, but it may significantly decrease precision





Query Expansion Summary of Points

- **Query Expansion is transparent in that it allows the user to see (select) expansion terms.**
- **Use WordNet to identify synonyms and related terms**
- **Use a thesaurus to develop a co-occurrence of related terms**
- **Query log mining approaches have also been shown to be useful**
- **However, query expansion still suffers from problems of polysemy (multiple meanings of a word or phrase)**
 - **E.g. “bank” is a financial institution, a building housing a financial institution, the process of saving money, a steep slope, something held in reserve, “you can rely (bank) on me”**



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

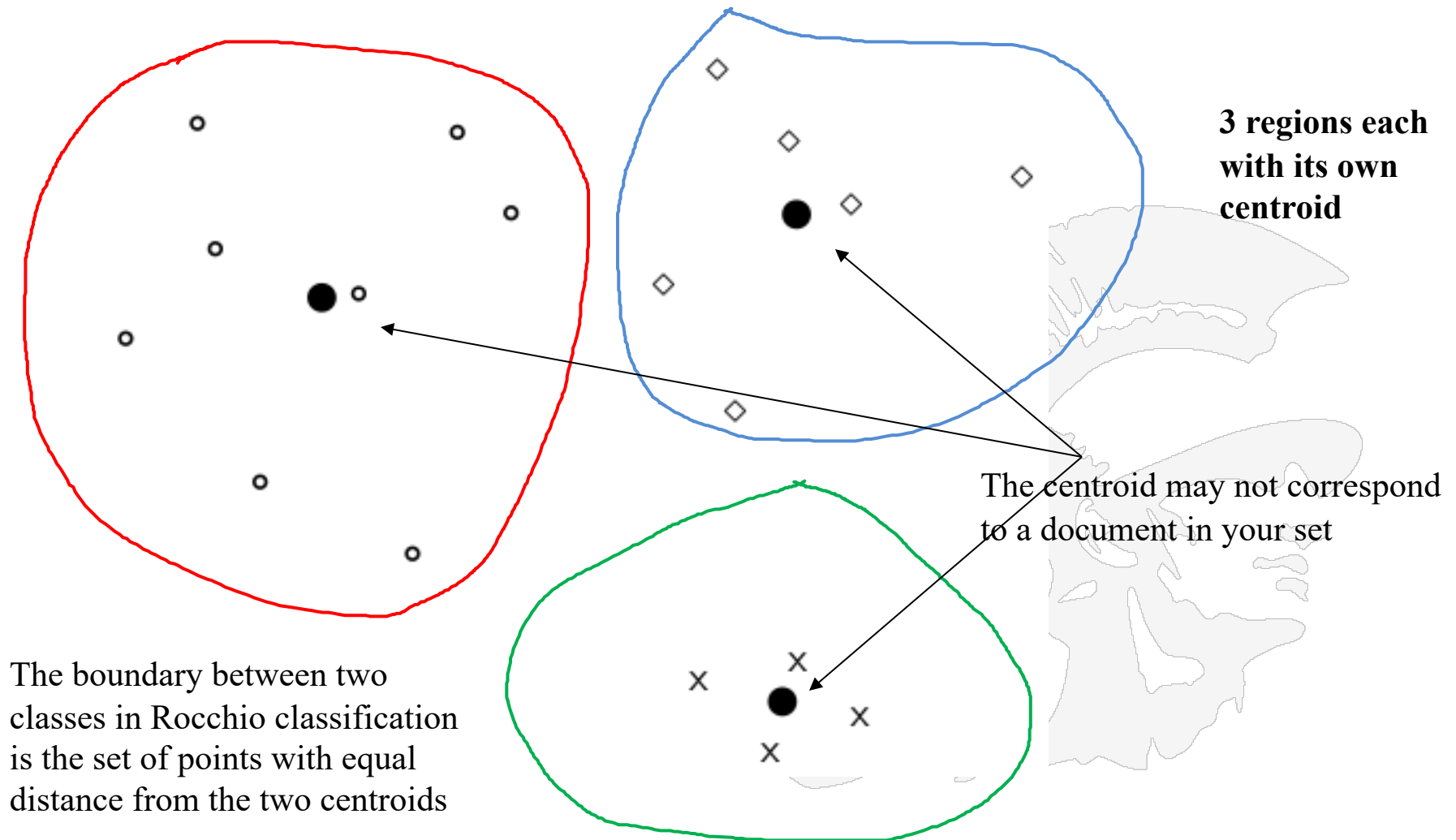
$$\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$$

where C is a set of documents, $|C|$ is the size of the set, and \vec{d} is the normalized vector representing document d

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



Centroid Example





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:

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

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

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.

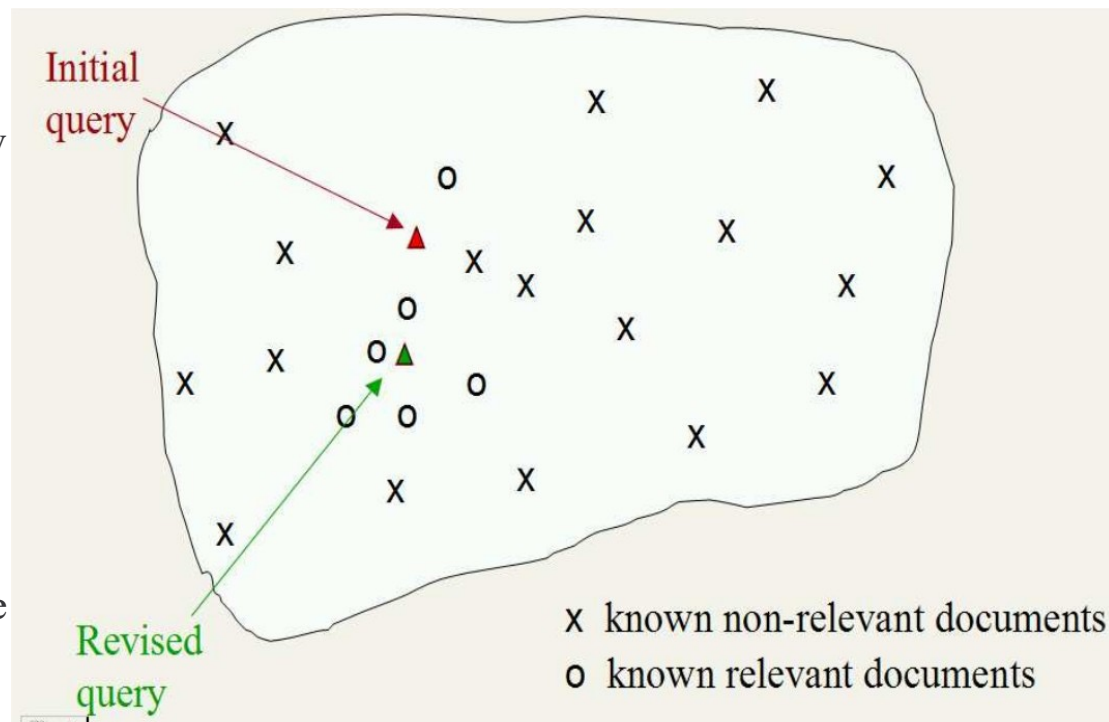
α , β , and γ are weight parameters attached to each component.

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

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

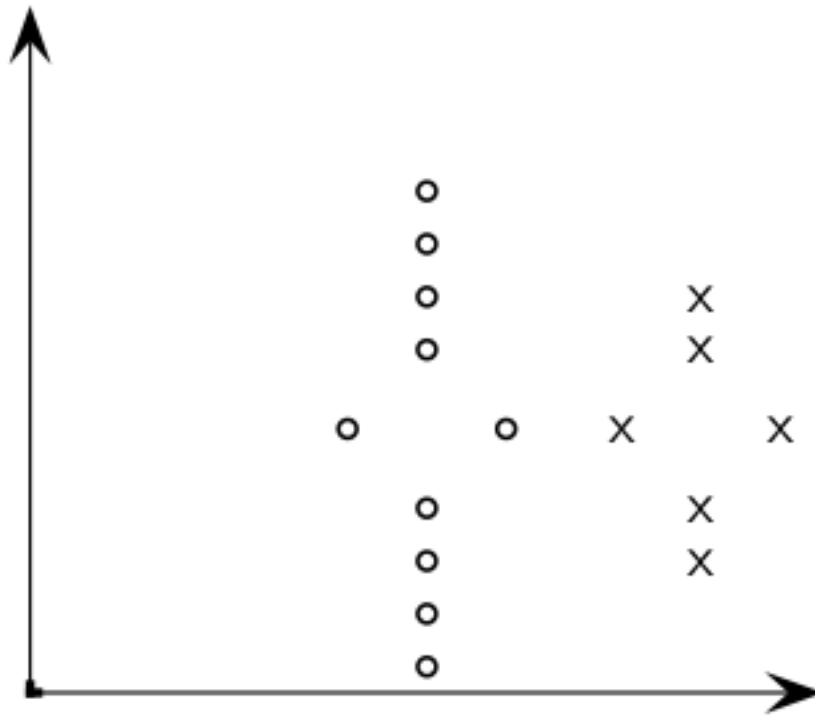
Rocchio in Practice

- 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 $\gamma = 0$.

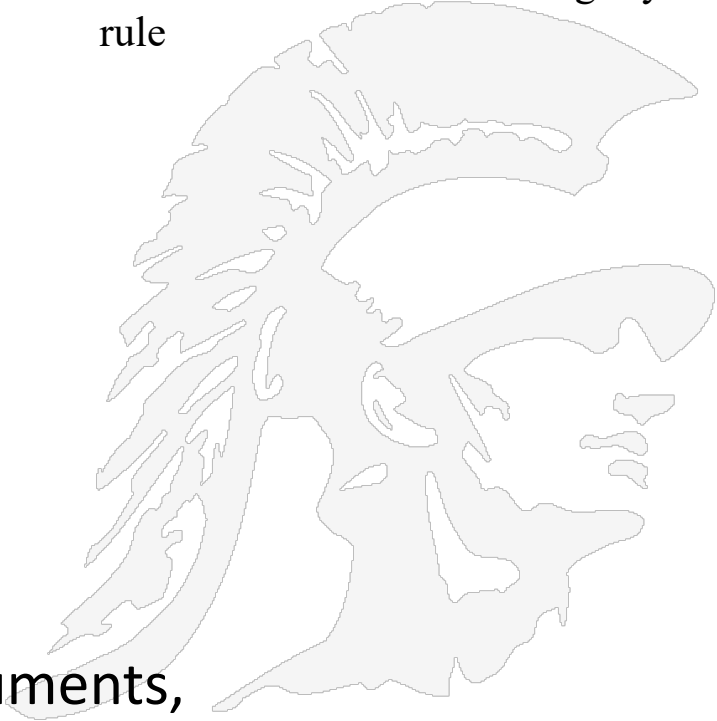




2D Rocchio Example



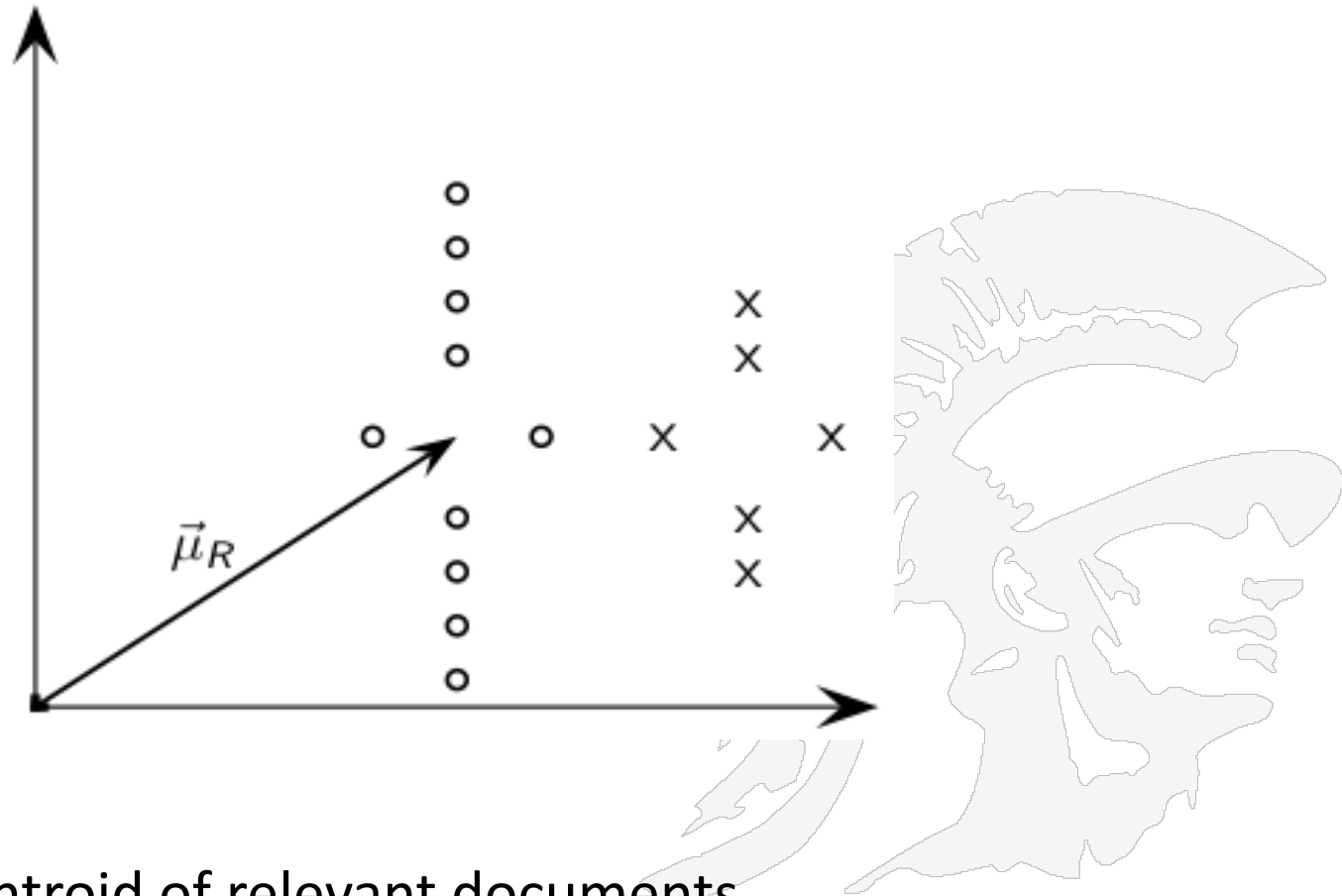
- For 2D examples the relevant set is generally much smaller than the non-relevant set;
- As a result we need a slightly modified rule



Let circles represent relevant documents,
Let Xs represent nonrelevant documents



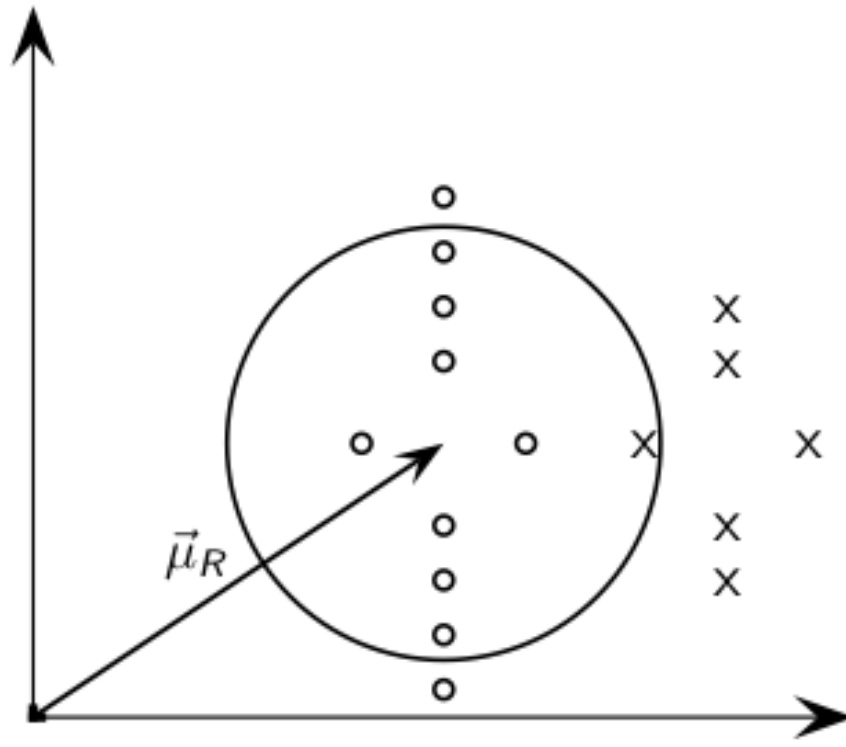
2D Rocchio Illustrated (1 of 9)



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



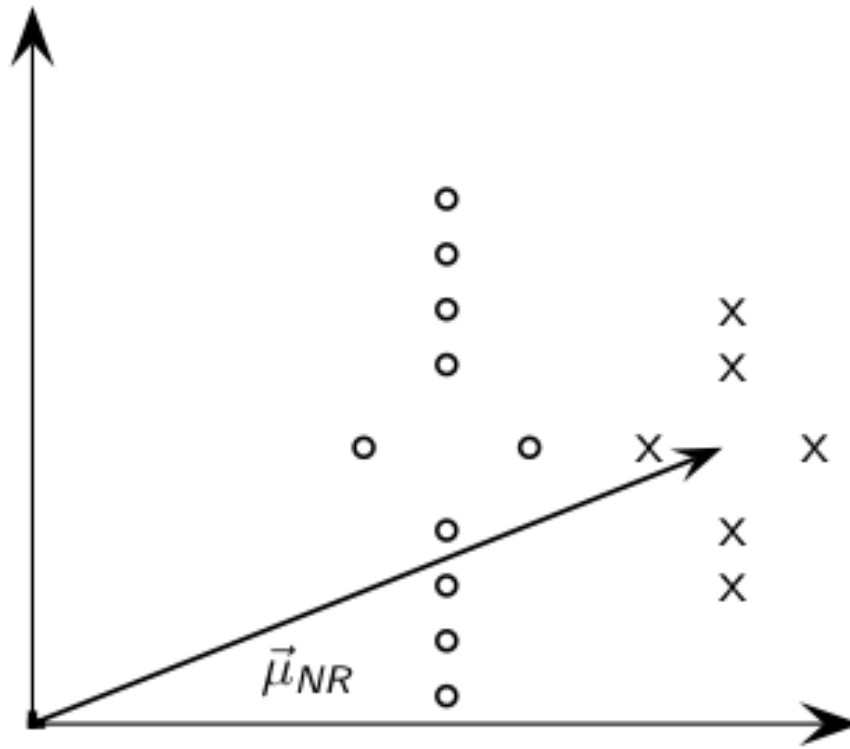
2D Rocchio Illustrated (2 of 9)



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



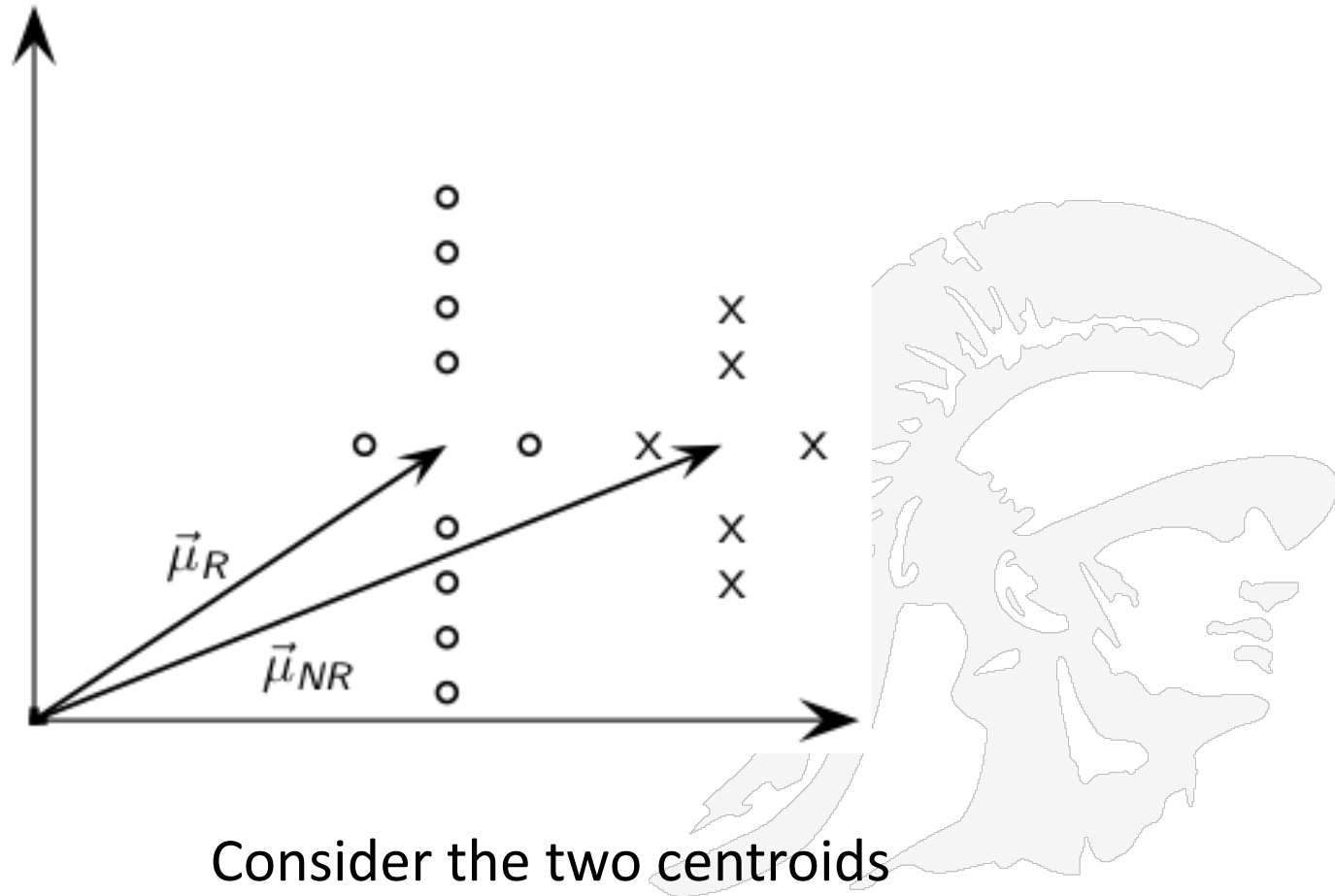
2D Rocchio Illustrated (3 of 9)



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

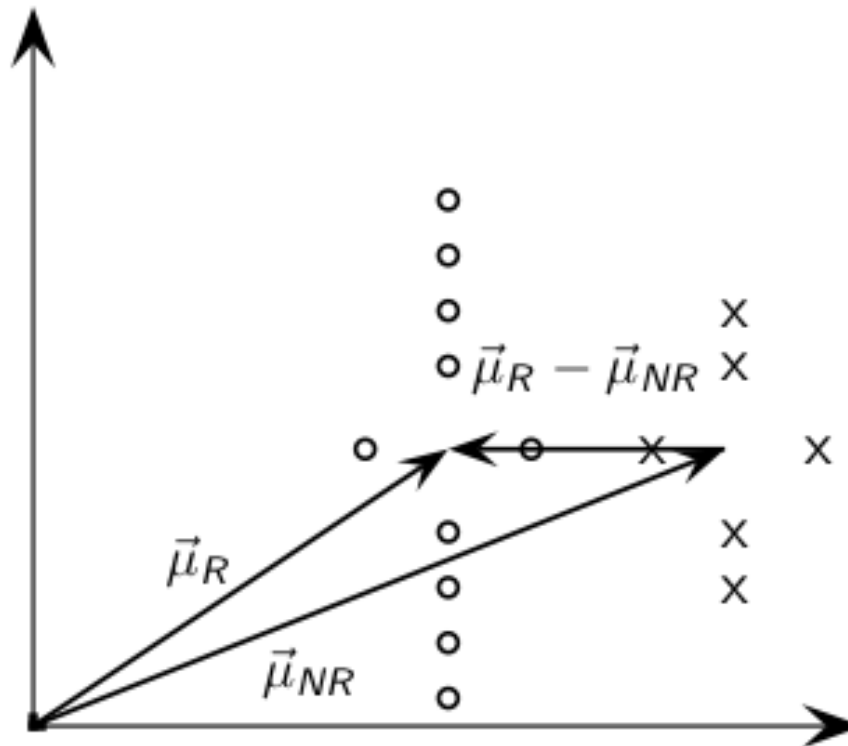


2D Rocchio Illustrated (4 of 9)





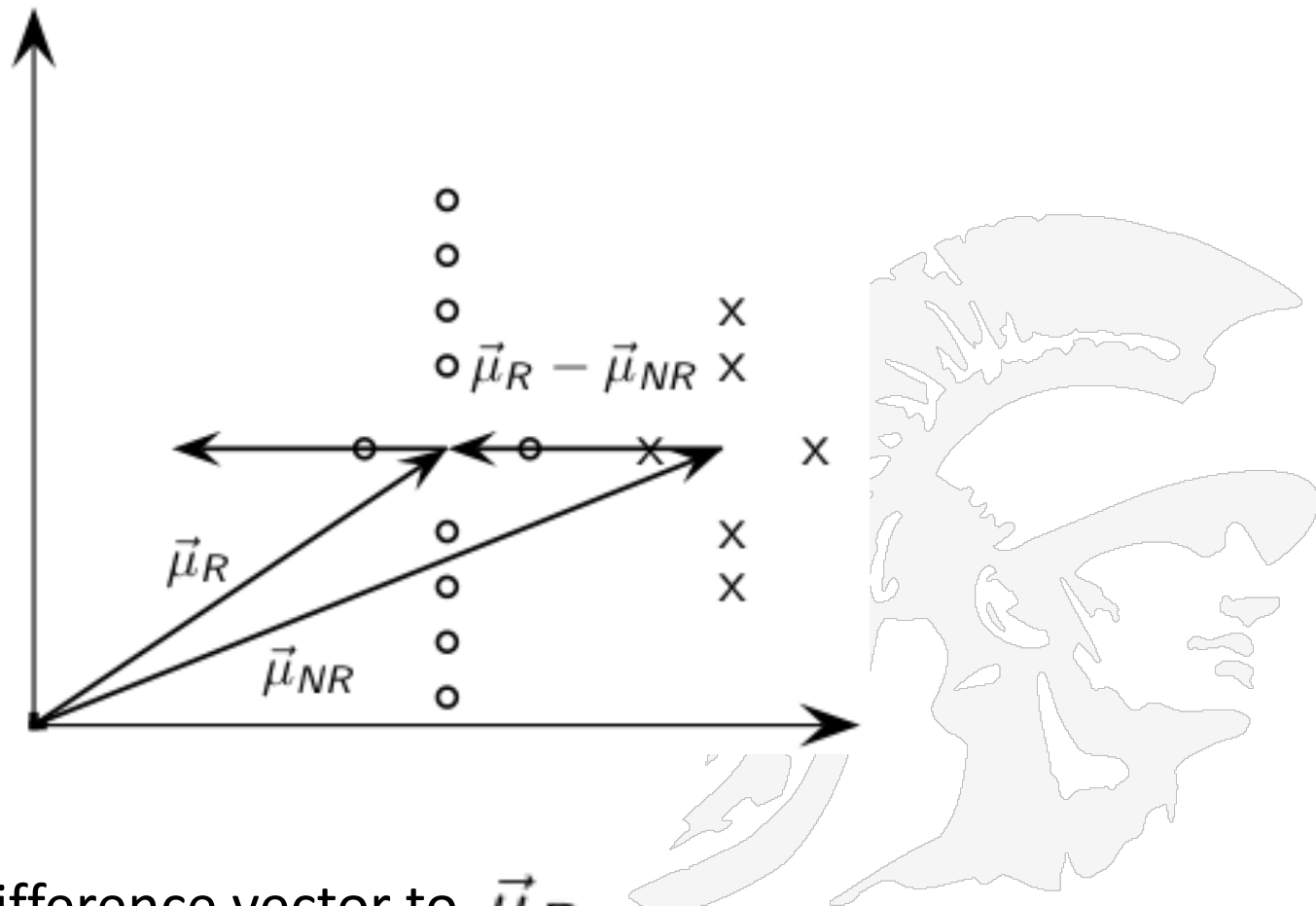
2D Rocchio Illustrated(5 of 9)



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

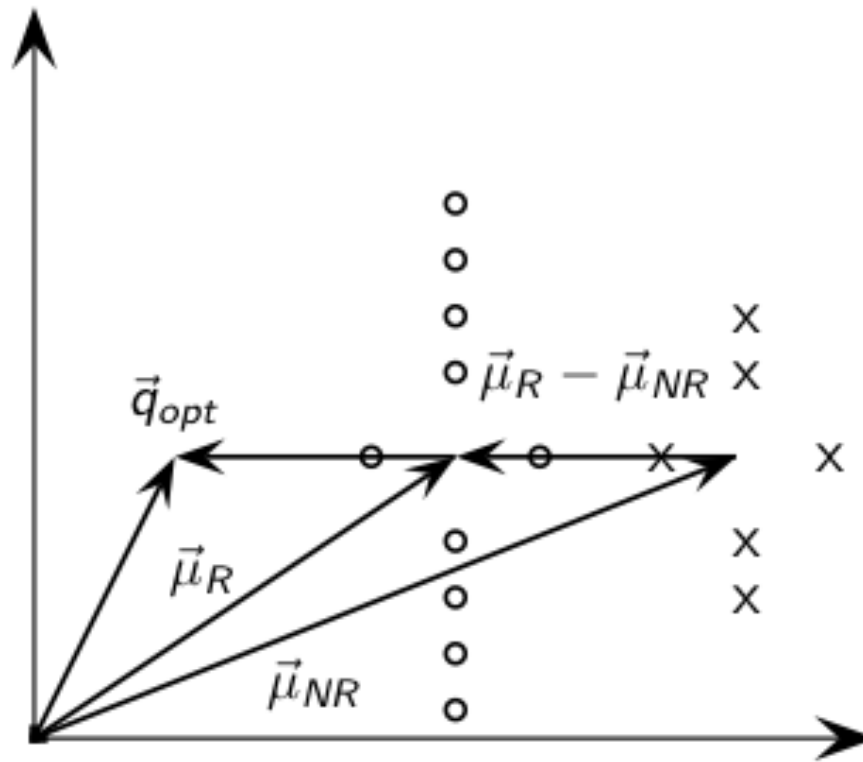


2D Rocchio Illustrated(6 of 9)



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

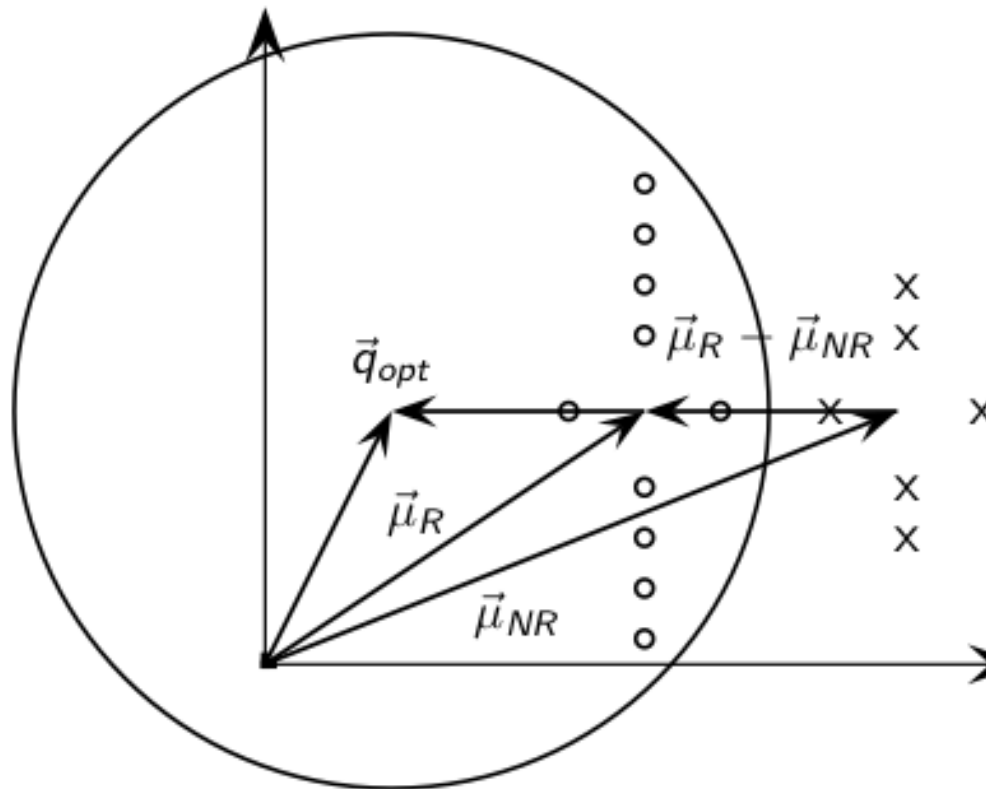
2D Rocchio Illustrated(7 of 9)



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



2D Rocchio Illustrated(8 of 9)



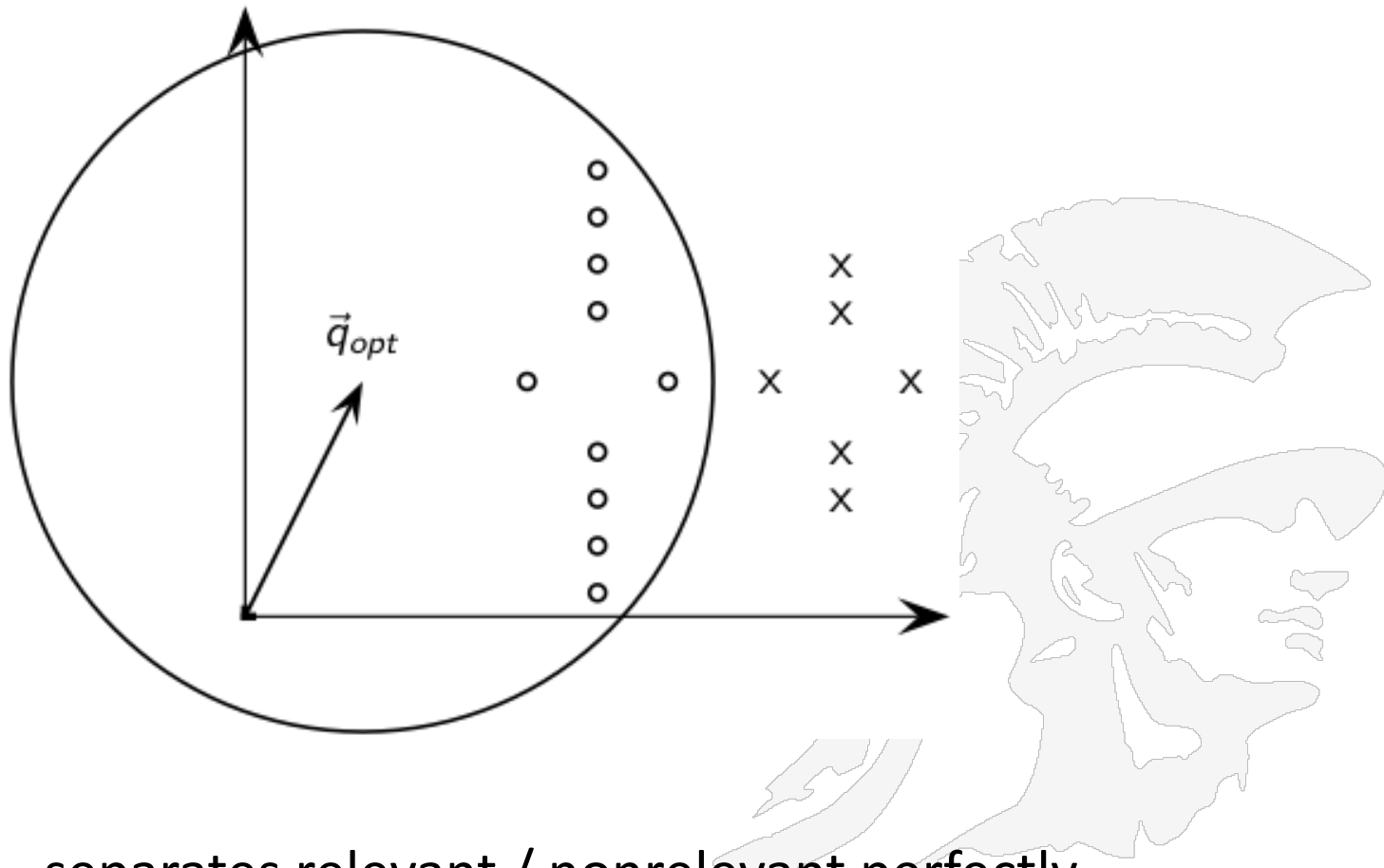
Note that the boundary computed during the Rocchio algorithm in This case is viewed as a circle;

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

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



2D Rocchio Illustrated(9 of 9)

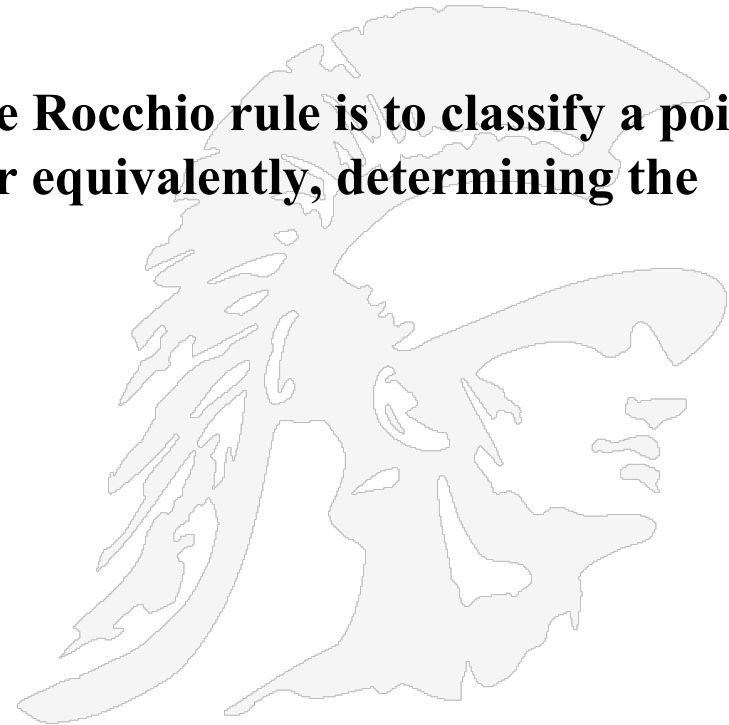


\vec{q}_{opt} separates relevant / nonrelevant perfectly.



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**





Rocchio Classification Algorithm

TrainRocchio(C, D)

For each c_j **in** C

do $D_j = \{d: \langle d, c_j \rangle \text{ is in } D\}$

$u_j = (1/|D_j|) * \sum d_j \text{ for } d_j \text{ in } D$

return $\{u_1, \dots, u_j\}$

ApplyRocchio($\{u_1, \dots, u_j\}, d$)

return $\arg \min |u_j - d_j|$

See Figure 14.4 in our textbook

D is the set of documents

C is the set of classes determined earlier

D_j is computed using Euclidean distance
and includes the set of documents in class c_j

u_j is the new centroid for documents in D_j

For each new document d and
centroids u_1 to u_j , place d in
the class whose centroid is the
closest to d



Computing Time for Rocchio

- | | | |
|-------------------|---------------------------------|---|
| • Mode | Complexity | |
| • Training | $O(D L_{ave} + C V)$ | ← Add all docs to their centroid,
And compute the centroid |
| • testing | $O(L_a + C M_a) = O(C M_a)$ | ← #centroids * #types |

- L_{ave} is the average number of tokens per document (*this is small*)
- L_a and M_a are the numbers of tokens and types respectively in the test document
- The time to compute the Euclidean distance between the class centroid and a document is $O(|C|M_a)$ (*the number of centroids times number of tokens*)
- Adding all documents to their respective (unnormalized) centroid is $\Theta(|D|L_{ave})$ (as opposed to $\Theta(|D||V|)$) since we need only consider non-zero entries.
- Dividing each vector sum by the size of its class to compute the centroid is $\Theta(|V|)$.

See Table 14.2 in our textbook

Conclusion: Overall training time is linear in the size of the collection



Rocchio Example

Simple Case

- Given two classes, C1 and C2 and the document vectors are
- C1: (2,7), (3,8), (1,6) (3 documents)
- C2: (6,4), (8,6), (5,2), (9,4) (4 documents)
- Which class, C1 or C2 does the document (74,36) fall into?
- Training phase of Rocchio
 - centroid of C1, $((2+3+1)/3, (7+8+6)/3) = (2,7)$
 - centroid of C2 = $((6+8+5+9)/4, (4+6+2+4)/4) = (7,4)$
- Actual phase to find into which class a new document fits:
 - compute the distance between the new document and the centroids or alternatively
 - draw the perpendicular divisor between the two centroids and then see where the new vector falls



Rocchio Example

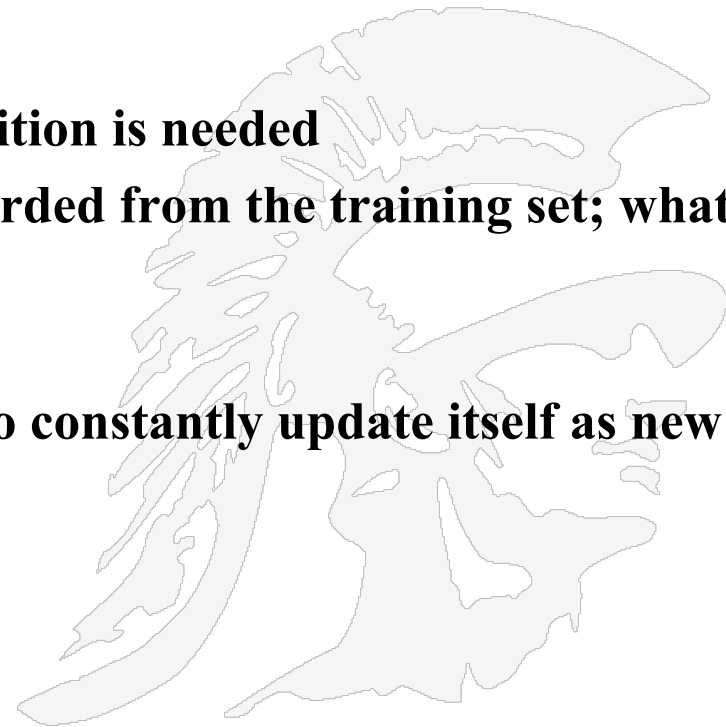
Simple Case

- To draw the perpendicular divisor between the two centroids and then see where the new vector falls recall:
 - The decision boundary will be a straight line, the perpendicular bisector of the two centroids;
 - The equation of the line defines the boundary
- Compute the perpendicular bisector between (2,7) and (7,4)
- if m_1 is the slope of the line between the two points, and m_2 is the slope of the perpendicular bisector, then $m_1 m_2 = -1$
- Slope $m_1 = (y_2 - y_1)/(x_2 - x_1) = (7 - 4)/(2 - 7) = -3/5$ and so the slope of m_2 is $5/3$
- The line is $y = mx + b$ where $m = 5/3$ and a point it passes through is the midpoint of the two centroids; the midpoint of the centroids is $((2+7)/2, (7+4)/2) = (9/2, 11/2)$
- So the value of b is $11/2 = (5/3)*(9/2) + b$ or $(11/2) - (45/6) = -12/6 = -2$
- $y = (5/3)x - 2$; returning to the original point (74,36) we see it falls into class C2



Rocchio Extension

- Suppose we have new documents arriving, how can we update the centroids in constant time?
 - Suppose we had N documents with centroid (x,y) and a new document with coordinates (a, b) arrives; then the new centroid is:
 - $((Nx + a)/(N+1), (Ny+b)/(N+1))$
 - One multiplication and one addition is needed
- Now suppose one document is discarded from the training set; what is the new centroid
 - $((Nx-a)/(N-1), (Ny-b)/(N-1))$
- The online Rocchio algorithm has to constantly update itself as new documents are added and deleted

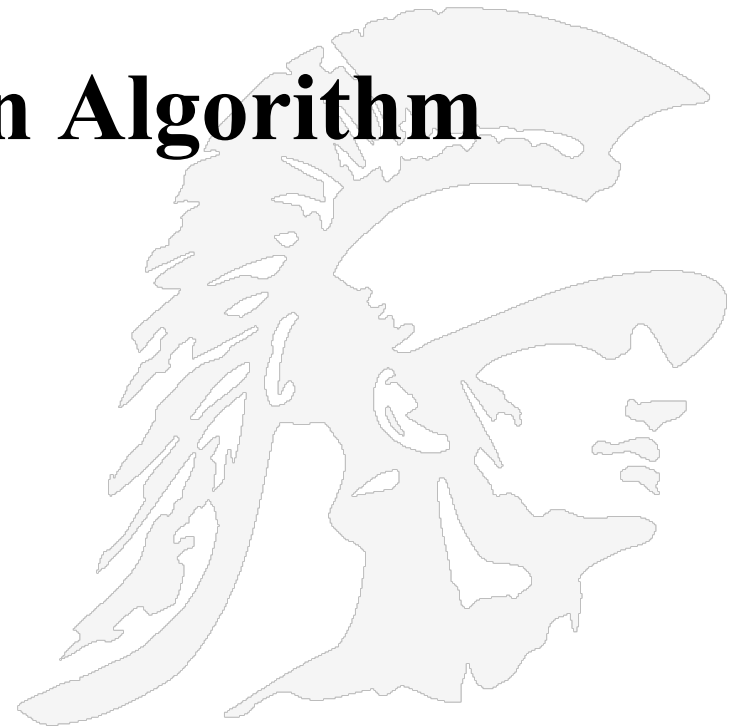




USC Viterbi
School of Engineering

***k*NN - *k* Nearest Neighbor Method**

Classification Algorithm





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



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



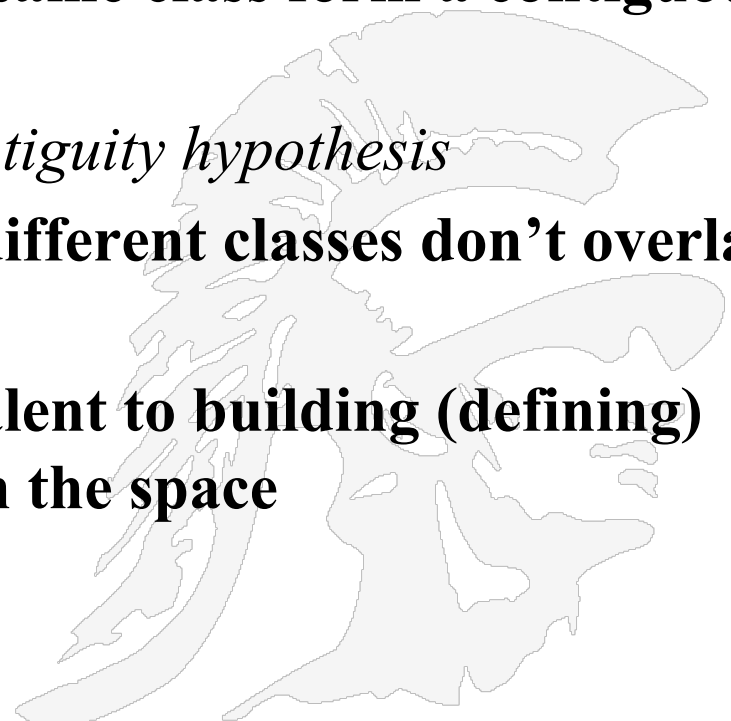
The Problem Statement for Classification

- **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:
$$C = \{c_1, c_2, \dots, c_n\}$$
- **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”**



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





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

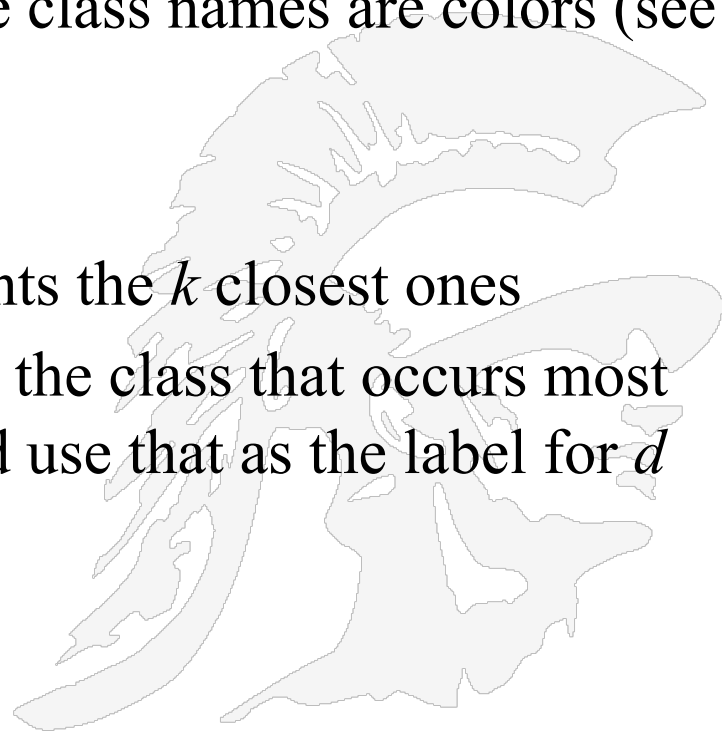
$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$





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

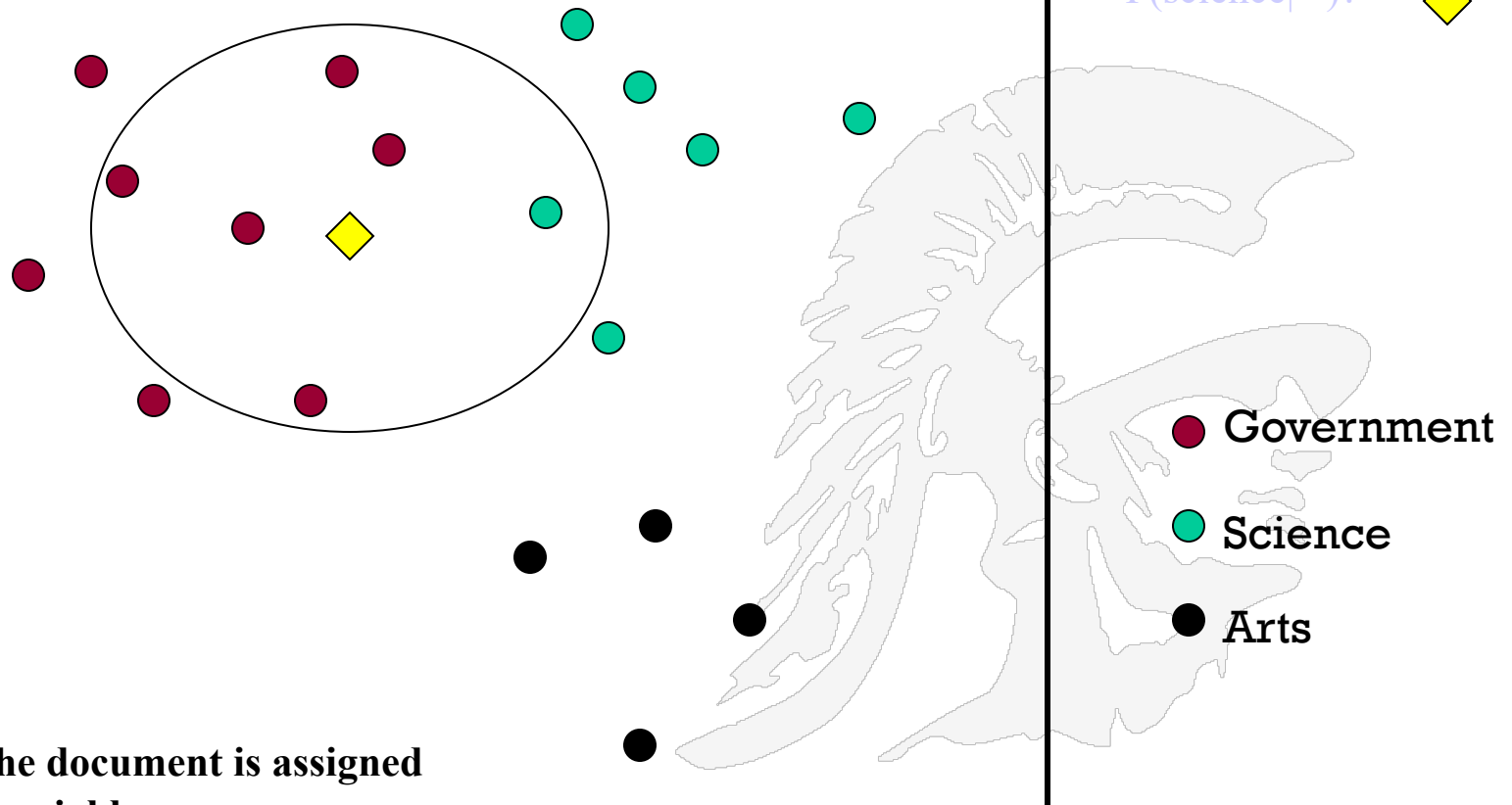




USC Viterbi
School of Engineering

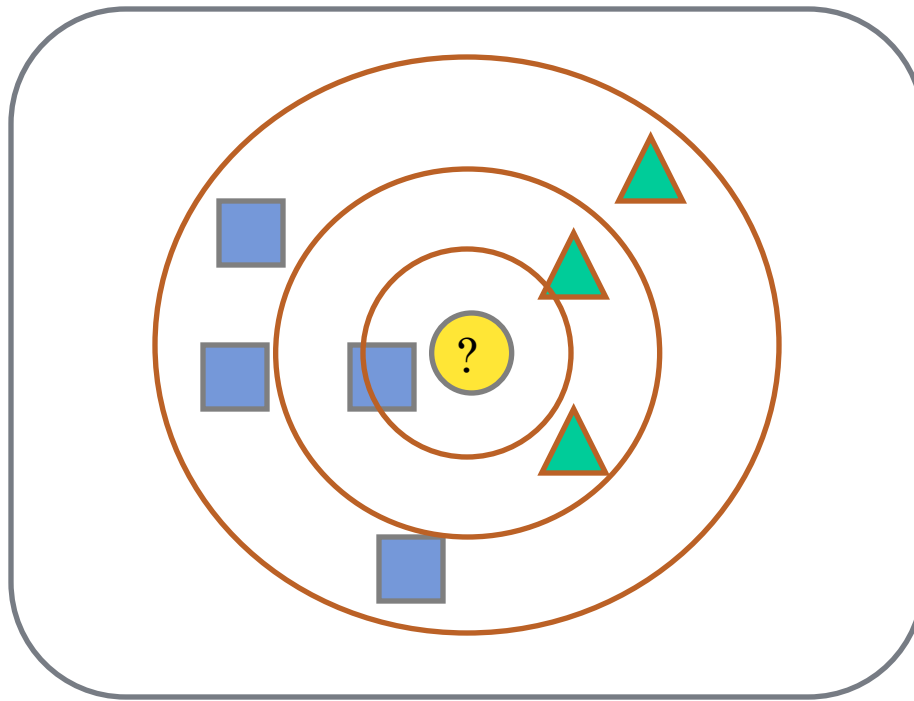
Example: $k=6$ (6NN)

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

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



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



Choice of K

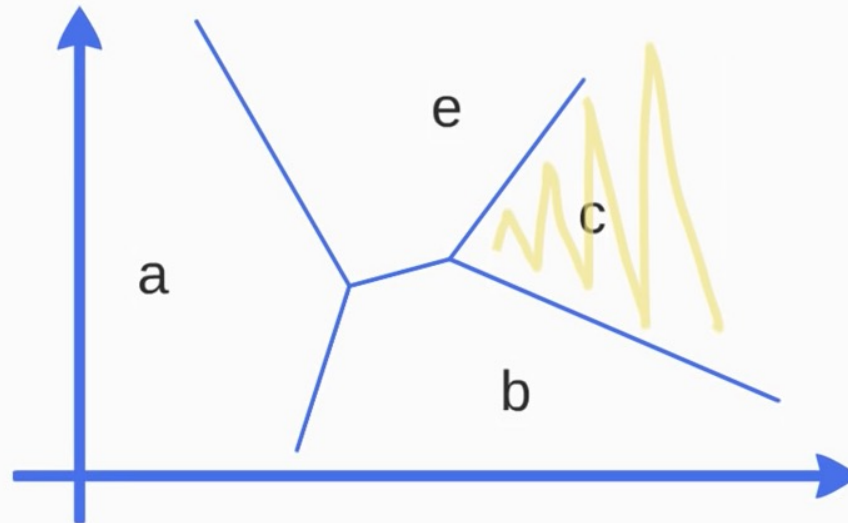
- **The best choice of k depends upon the data;**
 - generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct.
- The accuracy of the k -NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if 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



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



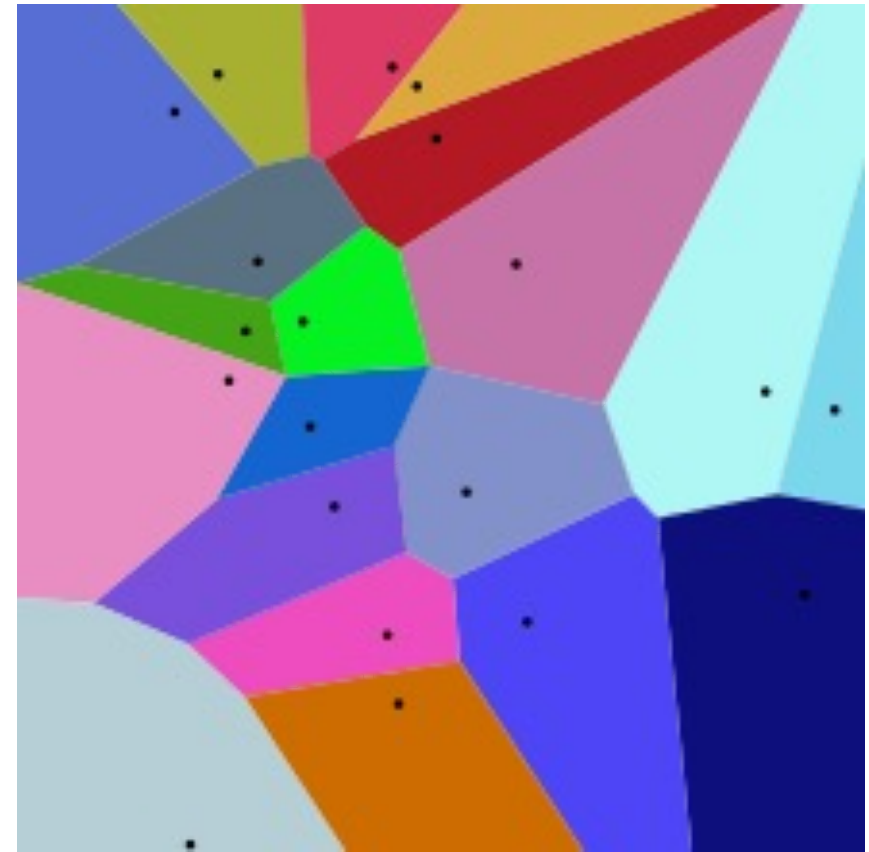
$$R_i = \{x : d(x, x_i) < d(x, x_j), i \neq j\}$$

R_i is Region i



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 Voronoi regions;
Line segments are all points equidistant to three
or more regions**

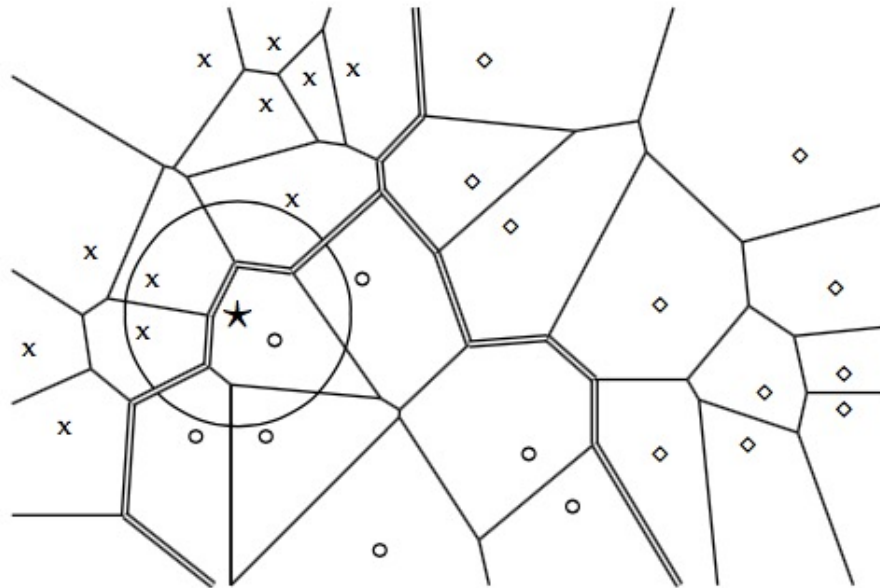


$K=1$ Nearest Neighbor Regions are Polygons

- For 1-NN we assign each document to the class of its closest neighbor
- For $k\text{-NN}$ 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

$K\text{-NN}$ is an example of a non-linear
classifier; (Rocchio is a linear classifier)





Computing Times

- Training a kNN classifier simply consists of determining k and pre-processing the documents
- If we preselect k and do not pre-process, then kNN requires no training at all
- It makes more sense to pre-process training documents once as part of the training phase rather than repeatedly every time we classify a new test document

kNN with preprocessing of training set

training $\Theta(|\mathcal{D}|L_{ave})$

testing $\Theta(L_a + |\mathcal{D}|M_{ave}M_a) = \Theta(|\mathcal{D}|M_{ave}M_a)$

kNN without preprocessing of training set

training $\Theta(1)$

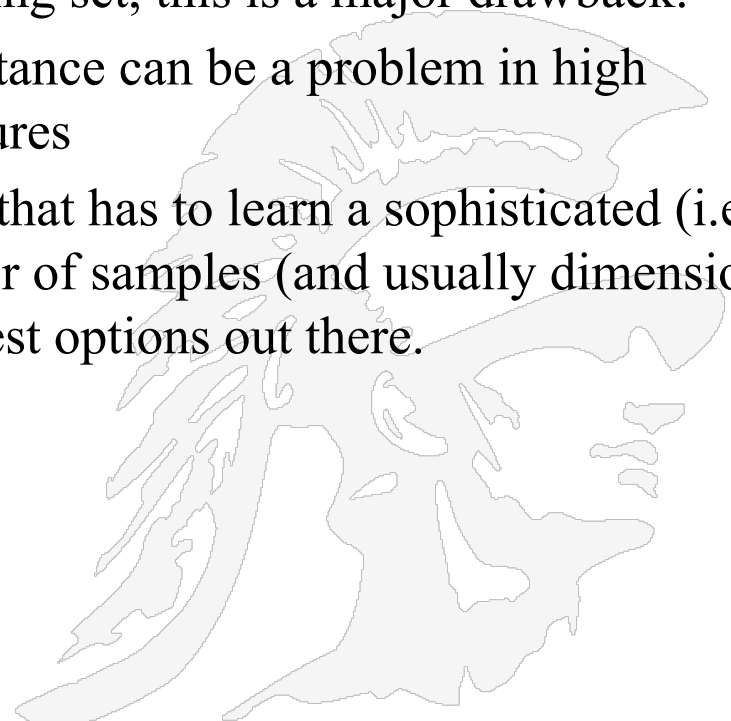
testing $\Theta(L_a + |\mathcal{D}|L_{ave}M_a) = \Theta(|\mathcal{D}|L_{ave}M_a)$

Recall from previous slide: L_{ave} is the average number of tokens per document
 L_a and M_a are the numbers of tokens and types respectively in the test document



Observations on k -NN

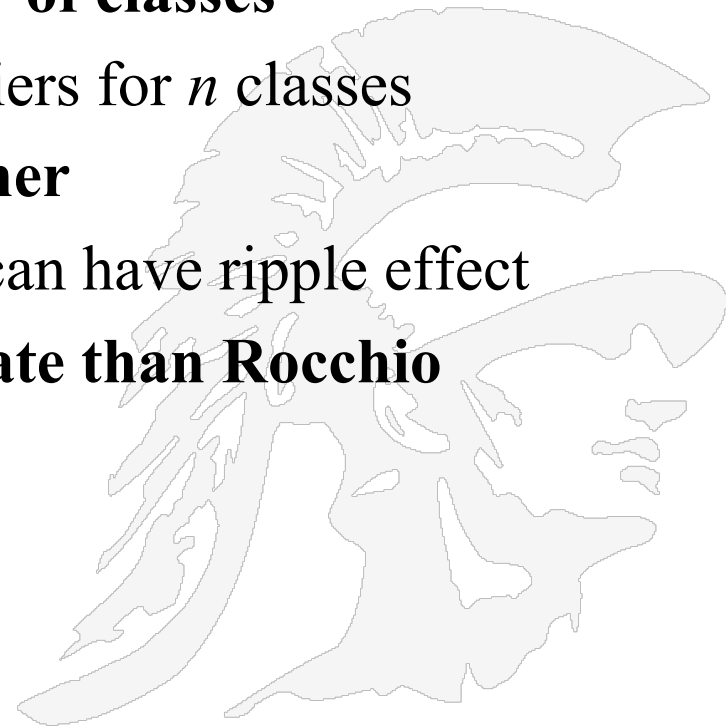
- **The K-Nearest-Neighbor model has two major drawbacks.**
 1. **performance.**
 - you have to load all of your training data and calculate distances to all training samples; Over a big training set, this is a major drawback.
 2. **distance metric.** Using Euclidean distance can be a problem in high dimensions as well as with noisy features
- **To summarize**, if you have a system that has to learn a sophisticated (i.e. nonlinear) pattern with a small number of samples (and usually dimensions), k -NN models are usually one of the best options out there.





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**





Algorithm Comparison

K-Means	K-Nearest Neighbors
Clustering algorithm	Classification Algorithm
Uses distance from data points to k-centroids to cluster data into k-groups.	Calculates k nearest data points from data point X. Uses these points to determine which class X belongs to
Centroids are not necessarily data points.	“Centroid” is the point X to be classified.
Updates centroid on each pass by calculations over all data in a class.	Data point to be classified remains the same.
Must iterate over data until center point doesn't move.	Only requires k distance calculations.