

# Introduction to **Information Retrieval**

CS276: Information Retrieval and Web  
Search

Text Classification 1

Chris Manning and Pandu Nayak

# Standing queries

---

- The path from IR to text classification:
  - You have an information need to monitor, say:
    - Unrest in the Niger delta region
  - You want to rerun an appropriate query periodically to find new news items on this topic
  - You will be sent new documents that are found
    - I.e., it's not **ranking** but **classification** (relevant vs. not relevant)
- Such queries are called **standing queries**
  - Long used by “information professionals”
  - A modern mass instantiation is **Google Alerts**
- Standing queries are (hand-written) text classifiers

**From:** Google Alerts

**Subject:** **Google Alert - stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal**

**Date:** May 7, 2012 8:54:53 PM PDT

**To:** Christopher Manning

---

**Web**

3 new results for stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal

[Twitter / Stanford NLP Group: @Robertoross If you only n ...](#)

@Robertoross If you only need tokenization, java -mx2m edu.stanford.nlp.process.PTBTOKENIZER file.txt runs in 2MB on a whole file for me.... 9:41 PM Apr 28th ...

[twitter.com/stanfordnlp/status/196459102770171905](https://twitter.com/stanfordnlp/status/196459102770171905)

[\[Java\] LexicalizedParser lp = LexicalizedParser.loadModel\("edu ...](#)

loadModel("edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz");. String[] sent = { "This", "is", "an", "easy", "sentence", "." };. Tree parse = lp.apply(Arrays.

[pastebin.com/az14R9nd](http://pastebin.com/az14R9nd)

[More Problems with Statistical NLP || kuro5hin.org](#)

Tags: nlp, ai, coursera, stanford, nlp-class, cky, nltk, reinventing the wheel, ... Programming Assignment 6 for Stanford's nlp-class is to implement a CKY parser .

[www.kuro5hin.org/story/2012/5/5/11011/68221](http://www.kuro5hin.org/story/2012/5/5/11011/68221)

---

Tip: Use quotes ("like this") around a set of words in your query to match them exactly. [Learn more.](#)

[Delete](#) this alert.

[Create](#) another alert.

[Manage](#) your alerts.

# Spam filtering

## Another text classification task

---

From: "" <takworld@hotmail.com>

Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

=====

Click Below to order:

<http://www.wholesaledaily.com/sales/nmd.htm>

# Categorization/Classification

---

- Given:
  - A representation of a document  $d$ 
    - Issue: how to represent text documents.
    - Usually some type of high-dimensional space – bag of words
  - A fixed set of classes:
$$C = \{c_1, c_2, \dots, c_J\}$$
- Determine:
  - The category of  $d$ :  $\gamma(d) \in C$ , where  $\gamma(d)$  is a classification function
  - We want to build classification functions (“classifiers”).

# Classification Methods (1)

---

- Manual classification
  - Used by the original Yahoo! Directory
  - Looksmart, about.com, ODP, **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

# Classification Methods (2)

---

- Hand-coded rule-based classifiers
  - One technique used by news agencies, intelligence agencies, etc.
  - Widely deployed in government and enterprise
  - Vendors provide “IDE” for writing such rules

# Classification Methods (2)

---

- Hand-coded rule-based classifiers
  - Commercial systems have complex query languages
  - Accuracy can be high if a rule has been carefully refined over time by a subject expert
  - Building and maintaining these rules is expensive



# A Verity topic

## A complex classification rule: art

```

comment line      # Beginning of art topic definition
top-level topic   art ACCRUE

topic definition modifiers {
    /author = "fsmith"
    /date  = "30-Dec-01"
    /annotation = "Topic created
                    by fsmith"

subtopic topic    * 0.70 performing-arts ACCRUE
evidencetopic     ** 0.50 WORD
topic definition modifier /wordtext = ballet
evidencetopic     ** 0.50 STEM
topic definition modifier /wordtext = dance
evidencetopic     ** 0.50 WORD
topic definition modifier /wordtext = opera
evidencetopic     ** 0.30 WORD
topic definition modifier /wordtext = symphony
subtopic          * 0.70 visual-arts ACCRUE
                  ** 0.50 WORD
                  /wordtext = painting
                  ** 0.50 WORD
                  /wordtext = sculpture
subtopic          * 0.70 film ACCRUE
                  ** 0.50 STEM
                  /wordtext = film
subtopic          ** 0.50 motion-picture PHRASE
                  *** 1.00 WORD
                  /wordtext = motion
                  *** 1.00 WORD
                  /wordtext = picture
                  ** 0.50 STEM
                  /wordtext = movie
subtopic          * 0.50 video ACCRUE
                  ** 0.50 STEM
                  /wordtext = video
                  ** 0.50 STEM
                  /wordtext = vcr
                  # End of art topic

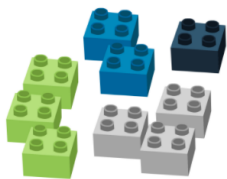
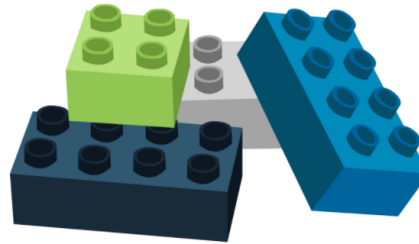
```

### ■ Note:

- maintenance issues (author, etc.)
- Hand-weighting of terms

[Verity was bought by Autonomy in 2005, which was bought by HP in 2011 – a mess; I think it no longer exists ...]

At the heart of eContext is its taxonomy of the commercial and social web. This taxonomy provides a consistent framework that groups conversations, behavior, and digital interactions into topical hierarchies up to 21 tiers, with clear connections between those topics. Clients push large amounts of text data from multiple sources through eContext, creating a single data set for measurement and analytics. With this holistic perspective, they can distill more comprehensive, meaningful insights, in less time.



As clients feed information through our API, our custom pre-processing parses the text into linguistically appropriate units that eContext can work with efficiently.

Then eContext checks each unit against its 55 million positive and negative rules to determine the correct topic, or topics, that content exhibits. This happens hundreds of thousands of times per second, as we enrich the highest velocity data streams available in our industry — everything up to and including the full firehose of Tweets. Thanks to the rules that govern eContext's decision-making, and the 450,000 different concepts it recognizes, eContext is able to deliver deep, broad, and accurate results.



Schedule a Demo



Enhancing Virtual Agents with Structured Knowledge

Download White Paper

# Classification Methods (3): Supervised learning

---

- Given:
  - A document  $d$
  - A fixed set of classes:  
 $C = \{c_1, c_2, \dots, c_J\}$
  - A training set  $D$  of documents each with a label in  $C$
- Determine:
  - A learning method or algorithm which will enable us to learn a classifier  $\gamma$
  - For a test document  $d$ , we assign it the class  
 $\gamma(d) \in C$

# Classification Methods (3)

---

- Supervised learning
  - Naive Bayes (simple, common) – see video, cs229
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (newer, generally more powerful)
  - Decision trees → random forests → gradient-boosted decision trees (e.g., xgboost)
  - ... plus many other methods
  - No free lunch: need hand-classified training data
  - But data can be built up by amateurs
- Many commercial systems use a mix of methods

# Features

---

- Supervised learning classifiers can use any sort of feature
  - URL, email address, punctuation, capitalization, dictionaries, network features
- In the simplest bag of words view of documents
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)

# The bag of words representation

---

$Y($

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

$) = C$

# The bag of words representation

$Y($

|           |     |
|-----------|-----|
| great     | 2   |
| love      | 2   |
| recommend | 1   |
| laugh     | 1   |
| happy     | 1   |
| ...       | ... |

$) = C$

# Feature Selection: Why?

---

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words ... and more
- Selection may make a particular classifier feasible
  - Some classifiers can't deal with 1,000,000 features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Makes runtime models smaller and faster
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting



# Feature Selection: Frequency

---

- The simplest feature selection method:
  - Just use the commonest terms
  - No particular foundation
  - But it make sense why this works
    - They're the words that can be well-estimated and are most often available as evidence
  - In practice, this is often 90% as good as better methods
- Smarter feature selection:
  - chi-squared, etc.

# Naïve Bayes: See //R 13 or cs124 lecture on Coursera or cs229

- Classify based on prior weight of class and conditional parameter for what each word says:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \left[ \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

- Training is done by counting and dividing:

$$P(c_j) \leftarrow \frac{N_{c_j}}{N} \qquad P(x_k | c_j) \leftarrow \frac{T_{c_j x_k} + \alpha}{\sum_{x_i \in V} [T_{c_j x_i} + \alpha]}$$

- Don't forget to smooth

# SpamAssassin

---

- Naïve Bayes has found a home in spam filtering
  - Paul Graham's A Plan for Spam
  - Widely used in spam filters
  - But many features beyond words:
    - black hole lists, etc.
    - particular hand-crafted text patterns

# SpamAssassin Features:

---

- Basic (Naïve) Bayes spam probability
- Mentions: Generic Viagra
- Regex: millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- Phrase: impress ... girl
- Phrase: 'Prestigious Non-Accredited Universities'
- From: starts with many numbers
- Subject is all capitals
- HTML has a low ratio of text to image area
- Relay in RBL, [http://www.mail-abuse.com/enduserinfo\\_rbl.html](http://www.mail-abuse.com/enduserinfo_rbl.html)
- RCVD line looks faked
- [http://spamassassin.apache.org/tests\\_3\\_3\\_x.html](http://spamassassin.apache.org/tests_3_3_x.html)

# Naive Bayes is Not So Naive

---

- Very fast learning and testing (basically just count words)
- Low storage requirements
- Very good in domains with many equally important features
- More robust to irrelevant features than many learning methods

Irrelevant features cancel out without affecting results

# Naive Bayes is Not So Naive

---

- More robust to concept drift (changing class definition over time)
- Naive Bayes won 1<sup>st</sup> and 2<sup>nd</sup> place in KDD-CUP 97 competition out of 16 systems

Goal: Financial services industry direct mail response prediction: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- A good dependable baseline for text classification (but not the best)!

# Evaluating Categorization

---

- Evaluation must be done on test data that are independent of the training data
  - Sometimes use cross-validation (averaging results over multiple training and test splits of the overall data)
- Easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set)

# Evaluating Categorization

---

- Measures: precision, recall, F1, classification accuracy
- **Classification accuracy**:  $r/n$  where  $n$  is the total number of test docs and  $r$  is the number of test docs correctly classified



# Remember: Vector Space Representation

---

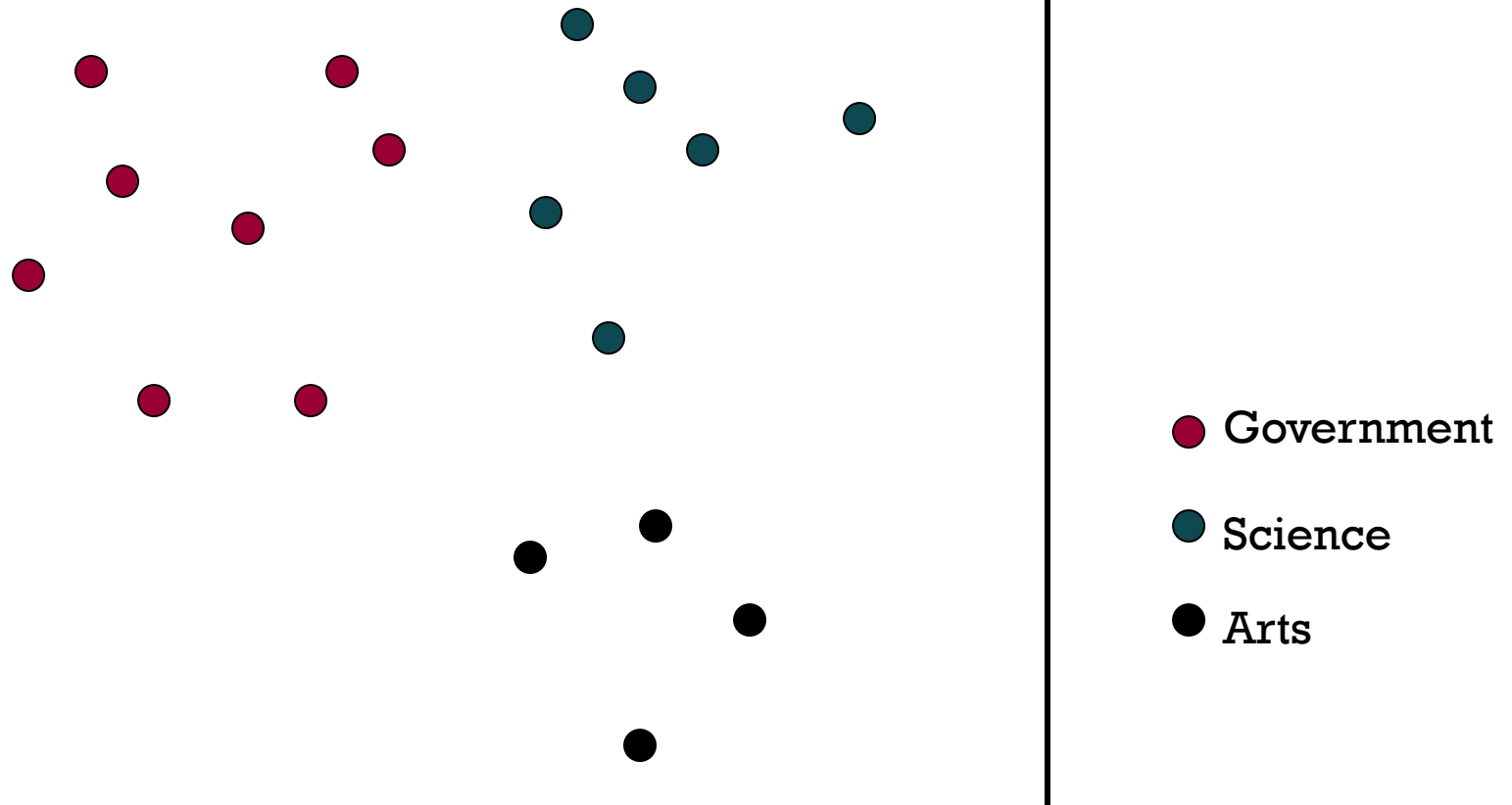
- Each document is a vector, one component for each term (= word).
- Normally normalize vectors to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space
- How can we do classification in this space?

# Classification Using Vector Spaces

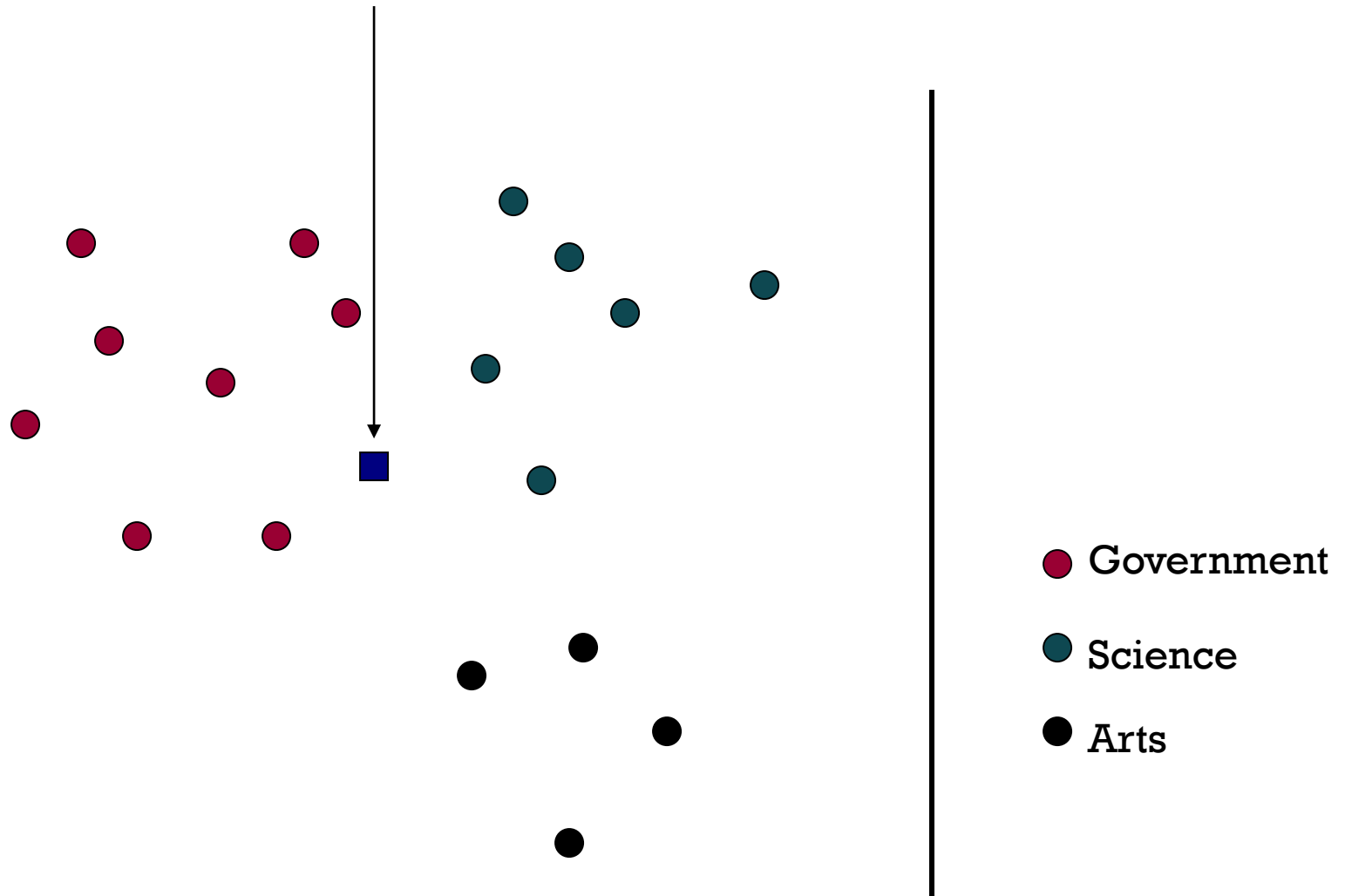
---

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)
- **Premise 1:** Documents in the same class form a contiguous region of space
- **Premise 2:** Documents from different classes don't overlap (much)
- Learning a classifier: build surfaces to delineate classes in the space

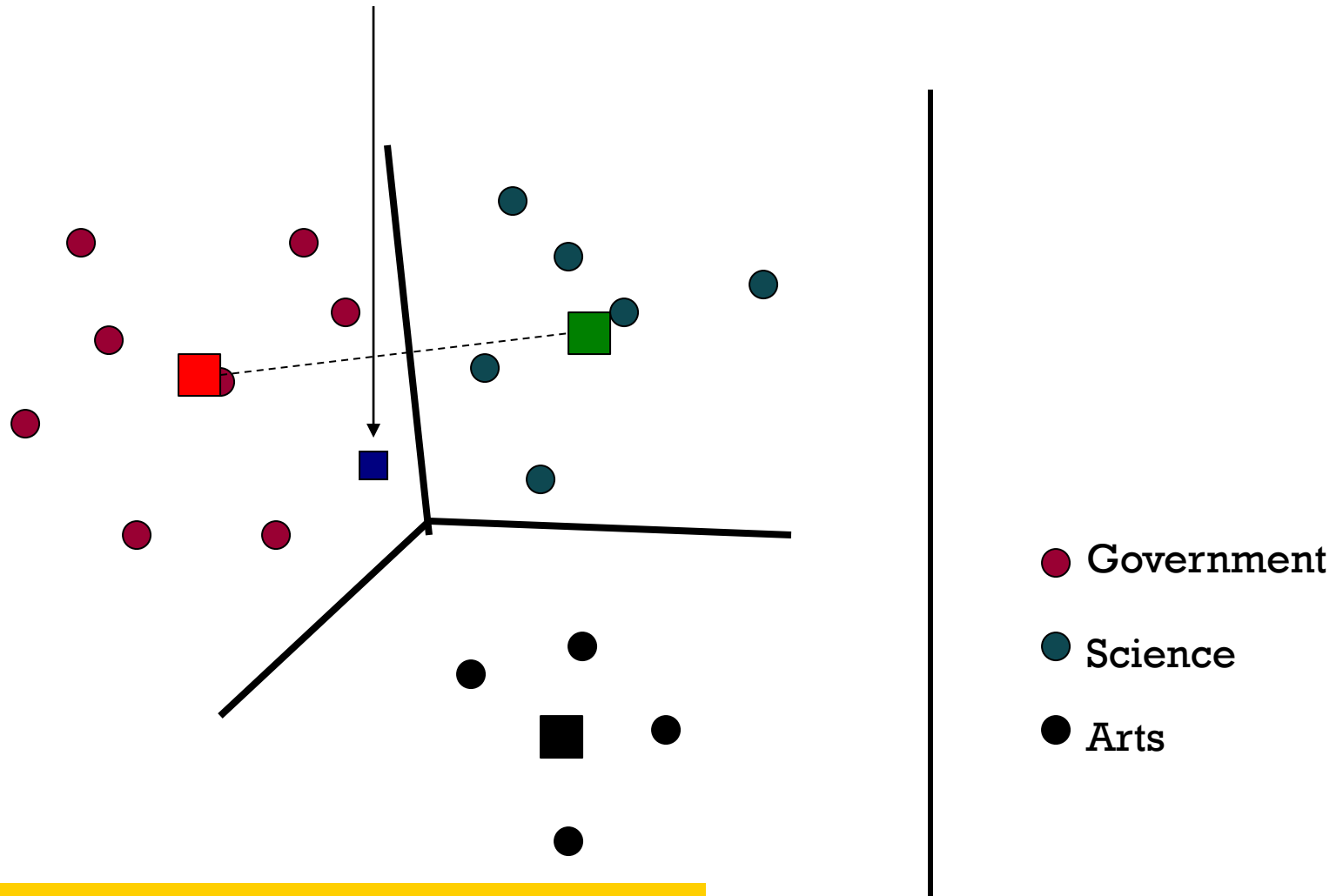
# Documents in a Vector Space



# Test Document of what class?



# Test Document = Government



Our focus: how to find good separators

# Definition of centroid

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where  $D_c$  is the set of all documents that belong to class  $c$  and  $\vec{v}(d)$  is the vector space representation of  $d$ .
- *Note that centroid will in general not be a unit vector even when the inputs are unit vectors.*

# Rocchio classification

- Rocchio forms a simple representative for each class: the centroid/prototype
- Classification: nearest prototype/centroid
- It does not guarantee that classifications are consistent with the given training data

Why not?

# Two-class Rocchio as a linear classifier

- Line or hyperplane defined by:

$$\sum_{i=1}^M w_i d_i = \theta$$

- For Rocchio, set:

$$\vec{w} = \vec{\mu}(c_1) - \vec{\mu}(c_2)$$

$$\theta = 0.5 \times (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$$



# Linear classifier: Example

- Class: “interest” (as in interest rate)
- Example features of a linear classifier
  - | $w_i$ | $t_i$      |
|-------|------------|
| 0.70  | prime      |
| 0.67  | rate       |
| 0.63  | interest   |
| 0.60  | rates      |
| 0.46  | discount   |
| 0.43  | bundesbank |
  - | $w_i$ | $t_i$ |
|-------|-------|
| -0.71 | dlrs  |
| -0.35 | world |
| -0.33 | sees  |
| -0.25 | year  |
| -0.24 | group |
| -0.24 | dlr   |
- To classify, find dot product of feature vector and weights

# Rocchio classification

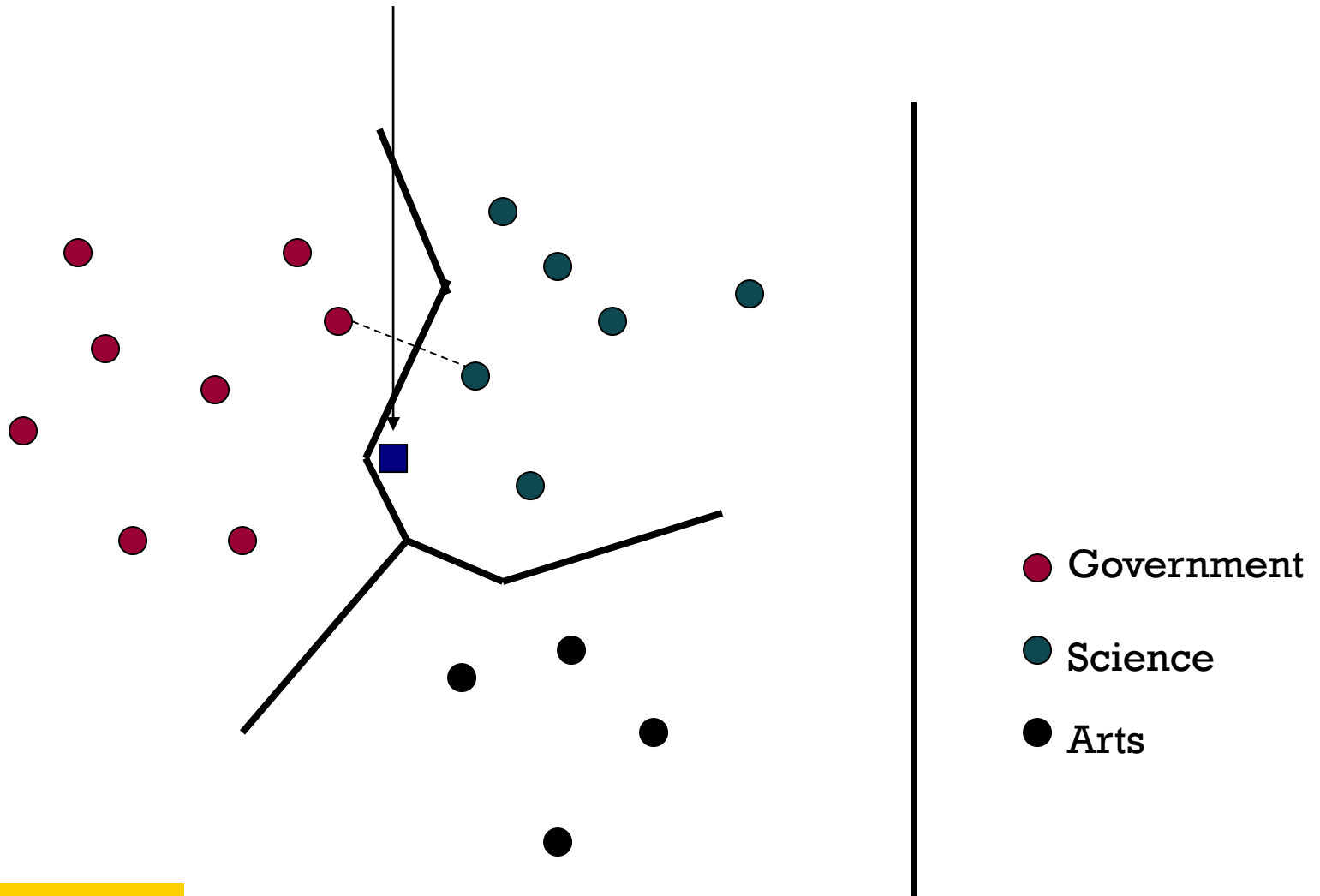
- Little used outside text classification
  - It has been used quite effectively for text classification
  - But in general worse than Naïve Bayes
- Again, cheap to train and test documents

# $k$ Nearest Neighbor Classification

---

- $k$ NN =  $k$  Nearest Neighbor
- To classify a document  $d$ :
- Define  $k$ -neighborhood as the  $k$  nearest neighbors of  $d$
- Pick the majority class label in the  $k$ -neighborhood
- For larger  $k$  can roughly estimate  $P(c | d)$  as  $\#(c)/k$

# Test Document = Science



Voronoi diagram

# Nearest-Neighbor Learning

- Learning: just store the labeled training examples  $D$
- Testing instance  $x$  (*under 1NN*):
  - Compute similarity between  $x$  and all examples in  $D$ .
  - Assign  $x$  the category of the most similar example in  $D$ .
- Does not compute anything beyond storing the examples
- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning
- Rationale of kNN: contiguity hypothesis

# k Nearest Neighbor

- Using only the closest example (1NN) is subject to errors due to:
  - A single atypical example.
  - Noise (i.e., an error) in the category label of a single training example.
- More robust: find the  $k$  examples and return the majority category of these  $k$
- $k$  is typically odd to avoid ties; 3 and 5 are most common

# Nearest Neighbor with Inverted Index

---

- Naively finding nearest neighbors requires a linear search through  $|D|$  documents in collection
- But determining  $k$  nearest neighbors is the same as determining the  $k$  best retrievals using the test document as a query to a database of training documents.
- Use standard vector space inverted index methods to find the  $k$  nearest neighbors.
- **Testing Time:**  $O(B|V_t|)$  where  $B$  is the average number of training documents in which a test-document word appears.
  - Typically  $B \ll |D|$

# kNN: Discussion

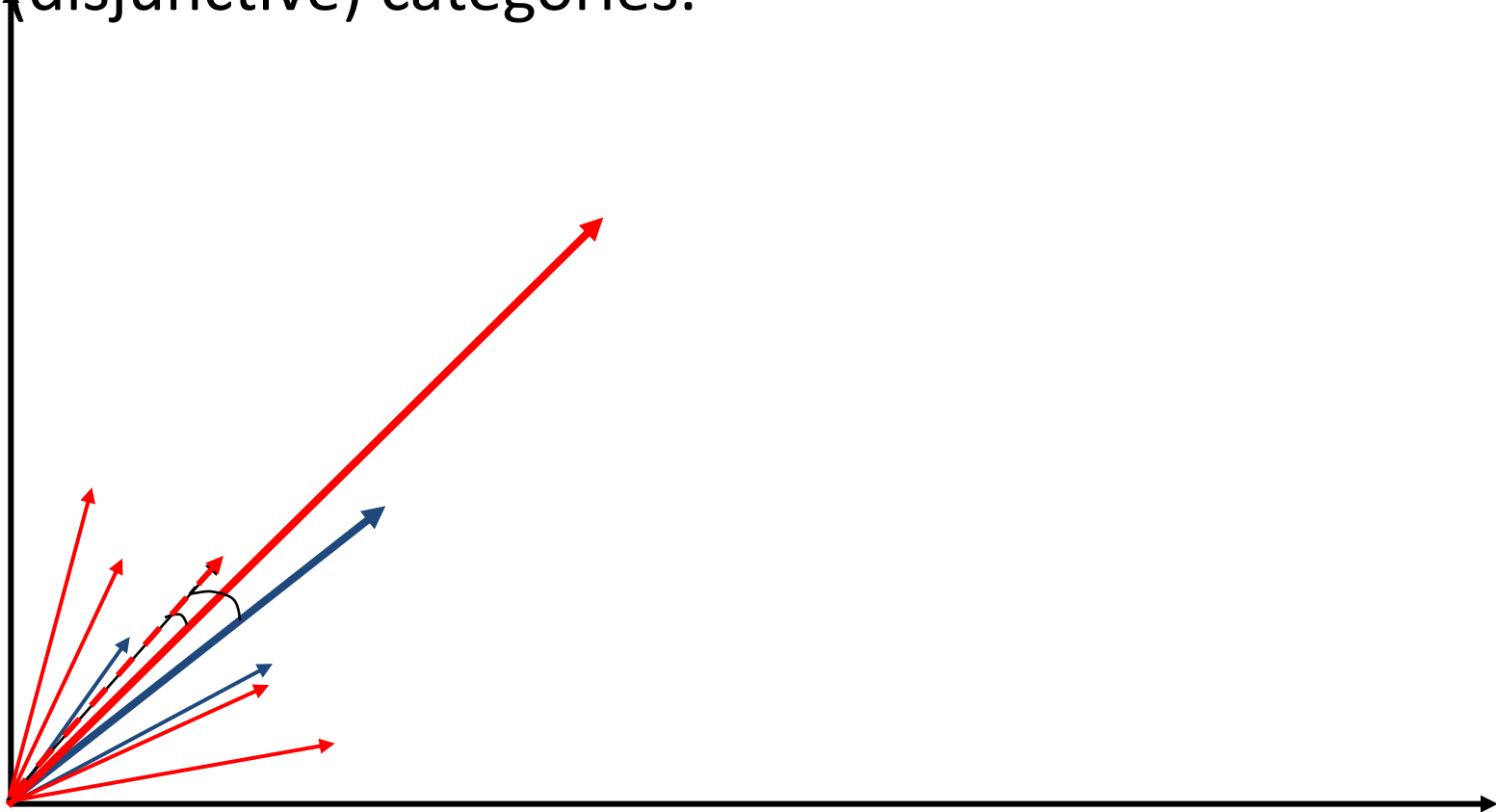
---

- 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
- Done naively, very expensive at test time
- In most cases it's more accurate than NB or Rocchio
  - As the amount of data goes to infinity, it has to be a great classifier! – it's "Bayes optimal"



# Rocchio Anomaly

- Prototype models have problems with polymorphic (disjunctive) categories.



### 3 Nearest Neighbor vs. Rocchio

- Nearest Neighbor tends to handle polymorphic categories better than Rocchio/NB.

