Introduction to

CS60092: Information Retrieval

Sourangshu Bhattacharya

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

[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

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

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: "" <takworlld@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:

    $C = \{c_1, c_2,\ldots, c_J\}$
- 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 is 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
                                 /author = "fsmith"
topic definition modifiers       /date   = "30-Dec-01"
                                 /annotation = "Topic created
                                                by fsmith"
subtopic topic               * 0.70 performing-arts ACCRUE
  evidence topic             ** 0.50 WORD
  topic definition modifier      /wordtext = ballet
  evidence topic             ** 0.50 STEM
  topic definition modifier      /wordtext = dance
  evidence topic             ** 0.50 WORD
  topic definition modifier      /wordtext = opera
  evidence topic             ** 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, which was bought by HP ...]

Classification Methods (3): Supervised learning

- Given:
  - A document *d*
  - A fixed set of classes:
    $C = \{c_1, c_2, \ldots, c_J\}$
  - 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 *γ*
  - For a test document *d*, we assign it the class
    $\gamma(d) \in C$

Classification Methods (3)

- Supervised learning
  - Naive Bayes (simple, common)
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (newer, generally more powerful)
  - … plus many other methods
  - No free lunch: requires hand-classified training data
  - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture 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)

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

Mutual information

- Compute the feature utility $A(t, c)$ as the expected mutual information (MI) of term t and class c.
- MI tells us "how much information" the term contains about the class and vice versa.
- For example, if a term's occurrence is independent of the class (same proportion of docs within/without class contain the term), then MI is 0.
- Definition:

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U=e_t, C=e_c) \log_2 \frac{P(U=e_t, C=e_c)}{P(U=e_t)P(C=e_c)}$$

How to compute MI values

- Based on maximum likelihood estimates, the formula we actually use is:

$$
\begin{aligned}
I(U;C) = {} & \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} \\
& + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}}
\end{aligned}
$$

- N_{10}: number of documents that contain t ($e_t = 1$) and are not in c ($e_c = 0$); N_{11}: number of documents that contain t ($e_t = 1$) and are in c ($e_c = 1$); N_{01}: number of documents that do not contain t ($e_t = 1$) and are in c ($e_c = 1$); N_{00}: number of documents that do not contain t ($e_t = 1$) and are not in c ($e_c = 1$); $N = N_{00} + N_{01} + N_{10} + N_{11}$.

MI example for *poultry*/EXPORT in Reuters

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{\text{EXPORT}} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{\text{EXPORT}} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

Plug these values into formula:

$$
\begin{aligned}
I(U;C) &= \frac{49}{801{,}948}\log_2\frac{801{,}948\cdot 49}{(49+27{,}652)(49+141)} \\
&+\frac{141}{801{,}948}\log_2\frac{801{,}948\cdot 141}{(141+774{,}106)(49+141)} \\
&+\frac{27{,}652}{801{,}948}\log_2\frac{801{,}948\cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\
&+\frac{774{,}106}{801{,}948}\log_2\frac{801{,}948\cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\
&\approx 0.000105
\end{aligned}
$$

MI feature selection on Reuters

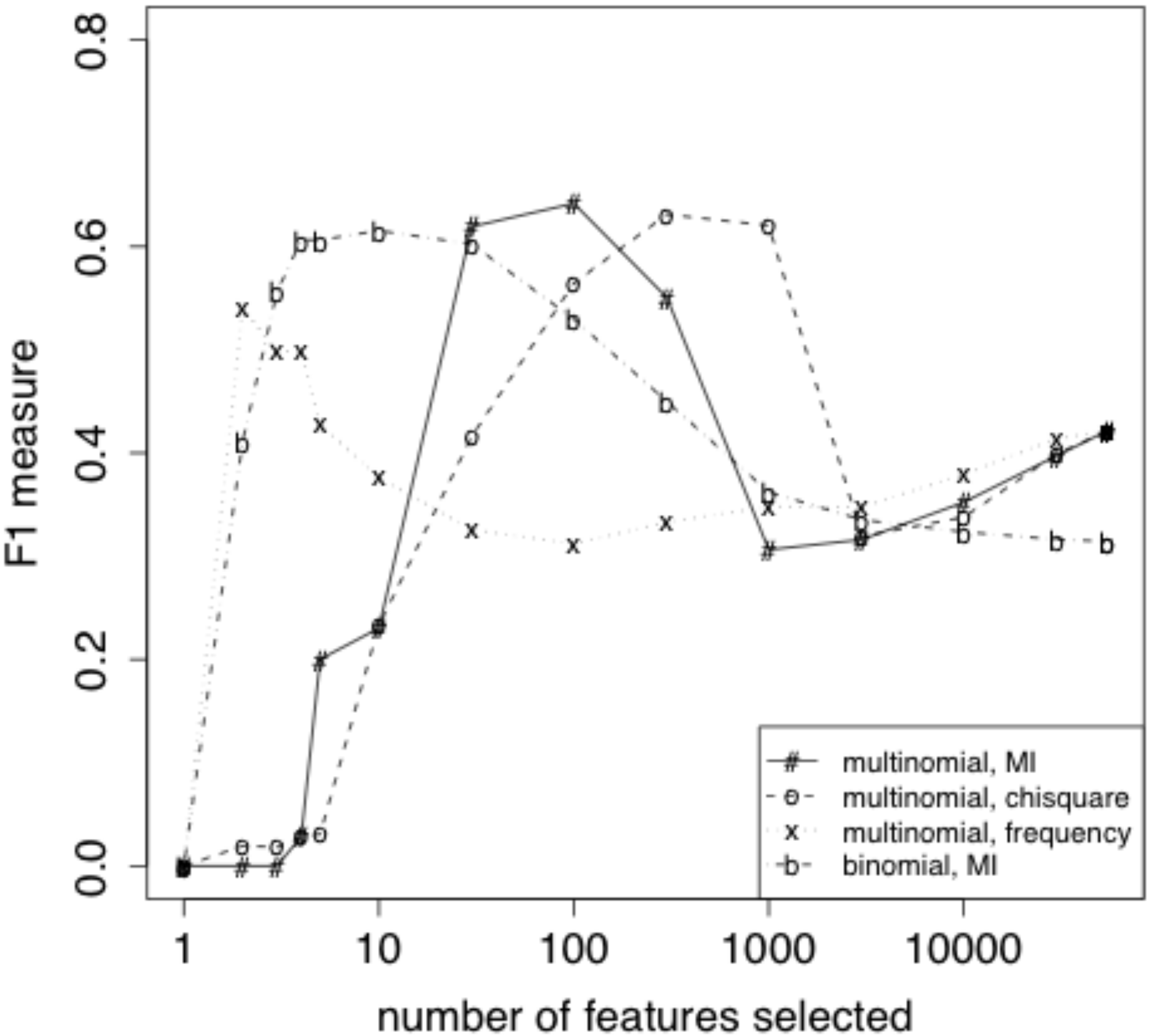
Class: *coffee*

term	MI
COFFEE	0.0111
BAGS	0.0042
GROWERS	0.0025
KG	0.0019
COLOMBIA	0.0018
BRAZIL	0.0016
EXPORT	0.0014
EXPORTERS	0.0013
EXPORTS	0.0013
CROP	0.0012

Class: *sports*

term	MI
SOCCER	0.0681
CUP	0.0515
MATCH	0.0441
MATCHES	0.0408
PLAYED	0.0388
LEAGUE	0.0386
BEAT	0.0301
GAME	0.0299
GAMES	0.0284
TEAM	0.0264

Naive Bayes: Effect of feature selection



(multinomial = multinomial Naive Bayes, binomial = Bernoulli Naive Bayes)

Feature selection for Naive Bayes

- In general, feature selection is necessary for Naive Bayes to get decent performance.
- Also true for most other learning methods in text classification: you need feature selection for optimal performance.

Outline

1. Recap
2. Text classification
3. Naive Bayes
4. NB theory
5. Evaluation of TC

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document *d* being in a class *c* as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k\ |c)$ is the conditional probability of term t_k occurring in a document of class *c*
- $P(t_k\ |c)$ as a measure of how much evidence t_k contributes that *c* is the correct class.
- $P(c)$ is the prior probability of *c*.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the “best” class.
- The best class is the most likely or maximum a posteriori (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \ \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

22

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since log(*xy*) = log(*x*) + log(*y*), we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

23

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)\right]$$

- Simple interpretation:
  - Each conditional parameter log $\hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior log $\hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
  - We select the class with the most evidence.

24

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

$$\hat{P}(c) = \frac{N_c}{N}$$

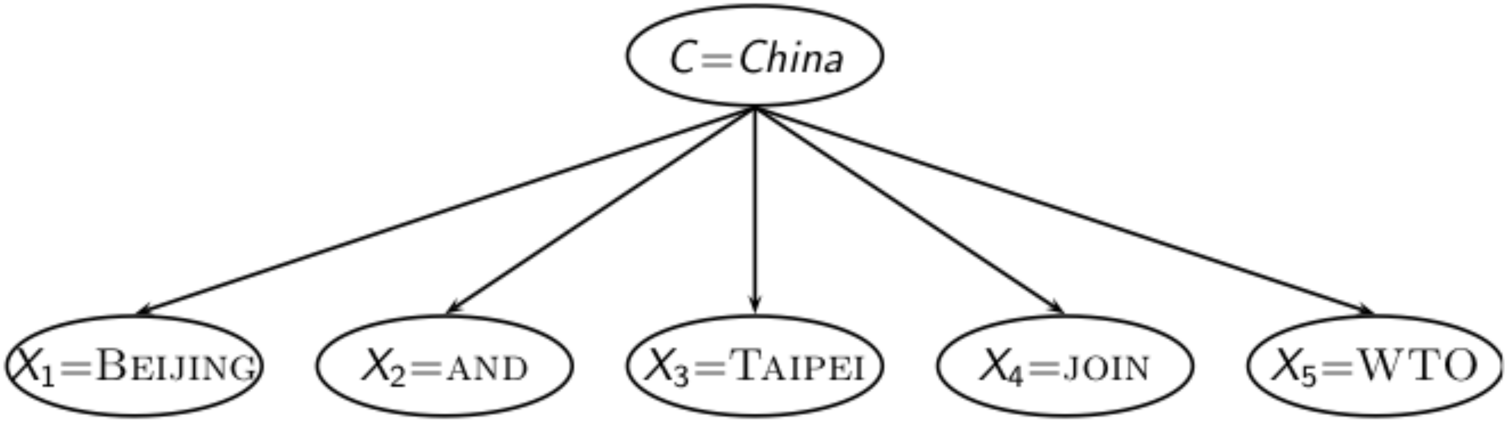
- N_c : number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)
- We've made a Naive Bayes independence assumption here:

$$\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$$

The problem with maximum likelihood estimates: Zeros



$$P(China|d) \propto P(China) \cdot P(\texttt{BEIJING}|China) \cdot P(\texttt{AND}|China) \cdot P(\texttt{TAIPEI}|China) \cdot P(\texttt{JOIN}|China) \cdot P(\text{WTO}|China)$$

- If WTO never occurs in class China in the train set:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

26

The problem with maximum likelihood estimates: Zeros (cont)

- If there were no occurrences of WTO in documents in class China, we'd get a zero estimate:

$$\hat{P}(\mathrm{WTO}|China) = \frac{T_{China,\mathrm{WTO}}}{\sum_{t' \in V} T_{China,t'}} = 0$$

- → We will get P(China|d) = 0 for any document that contains WTO!
- Zero probabilities cannot be conditioned away.

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

- B is the number of different words (in this case the size of the vocabulary: $|V| = M$)

28

To avoid zeros: Add-one smoothing

- Estimate parameters from the training corpus using add-one smoothing
- For a new document, for each class, compute sum of (i) log of prior and (ii) logs of conditional probabilities of the terms
- Assign the document to the class with the largest score

29

Naive Bayes: Training

$\text{TrainMultinomialNB}(\mathbb{C}, \mathbb{D})$

$V \leftarrow \text{ExtractVocabulary}(\mathbb{D})$

$N \leftarrow \text{CountDocs}(\mathbb{D})$

for each $c \in \mathbb{C}$

do $N_c \leftarrow \text{CountDocsInClass}(\mathbb{D}, c)$

$\quad prior[c] \leftarrow N_c / N$

$\quad text_c \leftarrow \text{ConcatenateTextOfAllDocsInClass}(\mathbb{D}, c)$

$\quad$ **for each** $t \in V$

$\quad$ **do** $T_{ct} \leftarrow \text{CountTokensOfTerm}(text_c, t)$

$\quad$ **for each** $t \in V$

$\quad$ **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$

return $V, prior, condprob$

30

Naive Bayes: Testing

```
ApplyMultinomialNB(ℂ, V, prior, condprob, d)
  W ← ExtractTokensFromDoc(V, d)
  for each c ∈ ℂ
  do score[c] ← log prior[c]
      for each t ∈ W
      do score[c] += log condprob[t][c]
  return arg max_{c∈ℂ} score[c]
```


31

Exercise

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify test document

Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$
\begin{aligned}
\hat{P}(\textsc{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\textsc{Tokyo}|c) = \hat{P}(\textsc{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textsc{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textsc{Tokyo}|\overline{c}) = \hat{P}(\textsc{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}
$$

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Classification

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Thus, the classifier assigns the test document to c = *China*. The reason for this classification decision is that the three occurrences of the positive indicator `CHINESE` in d_5 outweigh the occurrences of the two negative indicators `JAPAN` and `TOKYO`.

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V : vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$ is the time it takes to compute all counts.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the parameters from the counts.
- Generally: $\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\mathrm{ave}}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes is linear in the size of the training set (training) and the test document (testing). This is optimal.

35

Outline

1. Recap
2. Text classification
3. Naive Bayes
4. NB theory
5. Evaluation of TC

Naive Bayes: Analysis

- Now we want to gain a better understanding of the properties of Naive Bayes.
- We will formally derive the classification rule . . .
- . . . and state the assumptions we make in that derivation explicitly.

Derivation of Naive Bayes rule

We want to find the class that is most likely given the document:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(c|d)$$

Apply Bayes rule $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)}$$

Drop denominator since P(d) is the same for all classes:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(d|c)P(c)$$

38

Too many parameters / sparseness

$$
\begin{aligned}
c_{\text{map}} &= \arg\max_{c \in \mathbb{C}} \; P(d|c)P(c) \\
&= \arg\max_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)P(c)
\end{aligned}
$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$ one for each unique combination of a class and a sequence of words.
- We would need a very, very large number of training examples to estimate that many parameters.
- This is the problem of data sparseness.

39

Naive Bayes conditional independence assumption

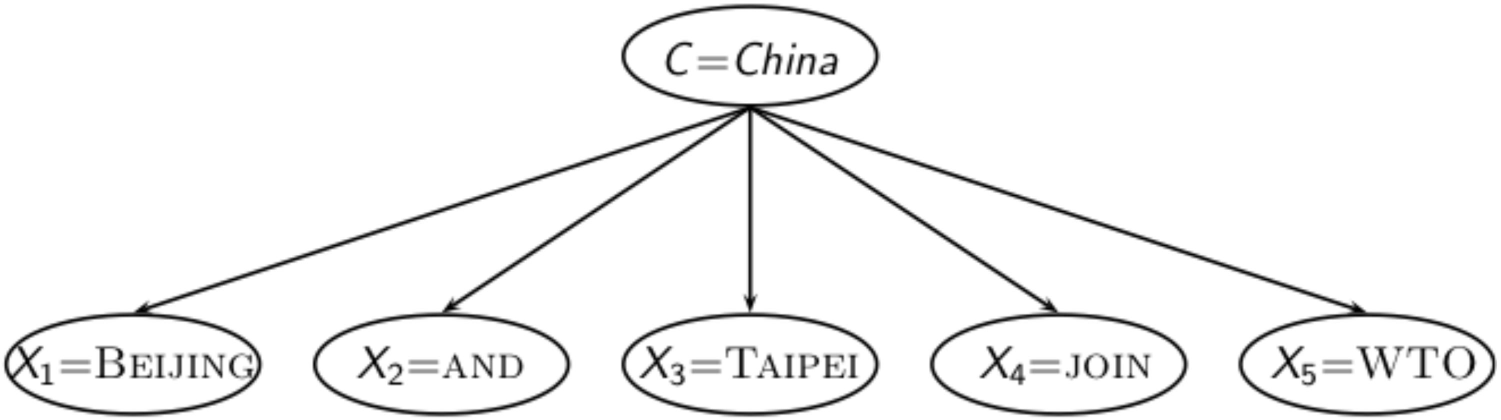
To reduce the number of parameters to a manageable size, we make the Naive Bayes conditional independence assumption:

$$P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$$

We assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities P($X_k = t_k \mid c$). Recall from earlier the estimates for these priors and conditional probabilities: $\hat{P}(c) = \frac{N_c}{N}$ and $\hat{P}(t|c) = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B}$

40

Generative model



$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- Generate a class with probability $P(c)$
- Generate each of the words (in their respective positions), conditional on the class, but independent of each other, with probability $P(t_k\,|c)$
- To classify docs, we "reengineer" this process and find the class that is most likely to have generated the doc.

Second independence assumption

- $\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$
- For example, for a document in the class *UK*, the probability of generating QUEEN in the first position of the document is the same as generating it in the last position.
- The two independence assumptions amount to the bag of words model.

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

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 1st and 2nd 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

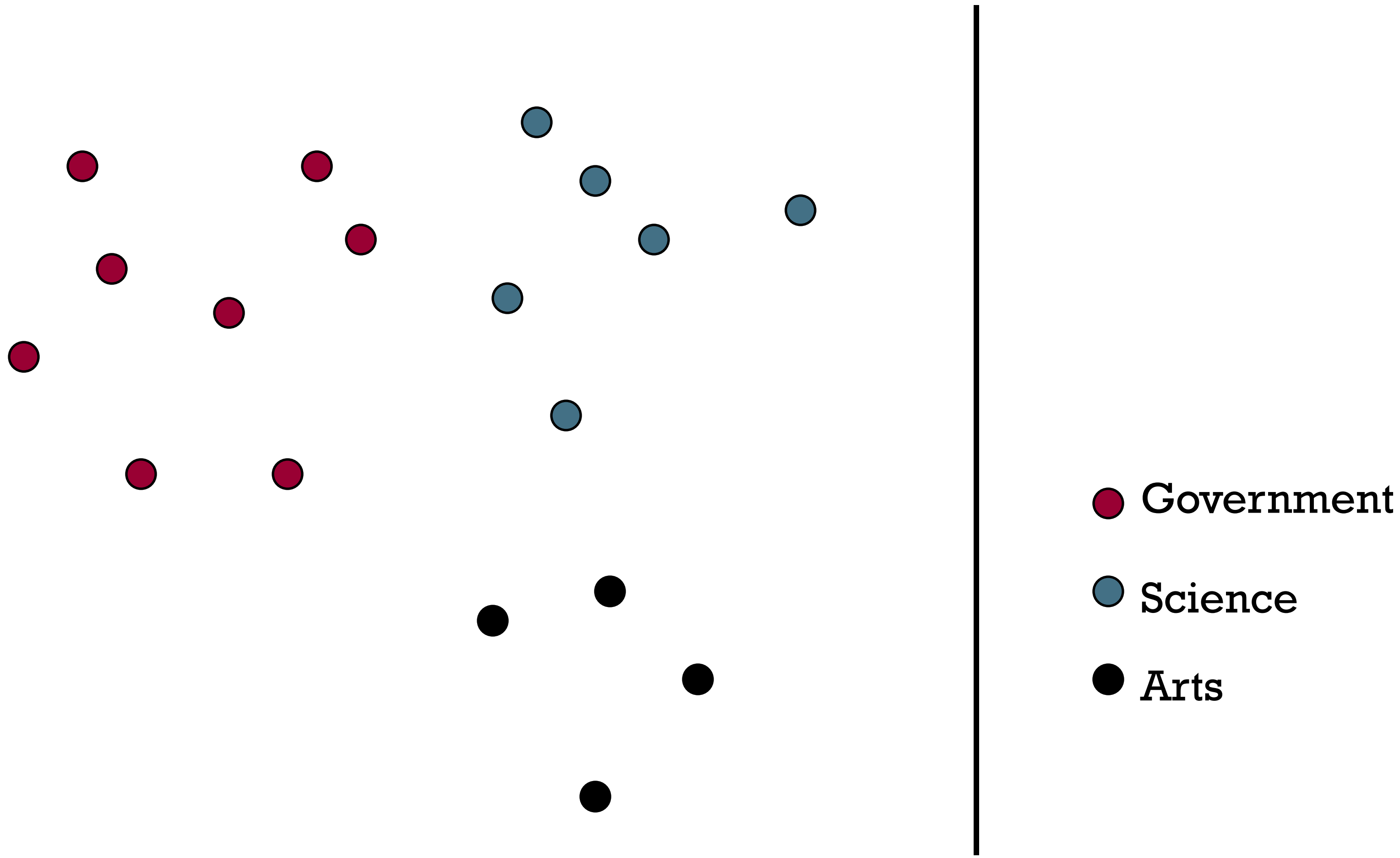
Recall: Vector Space Representation

- 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?

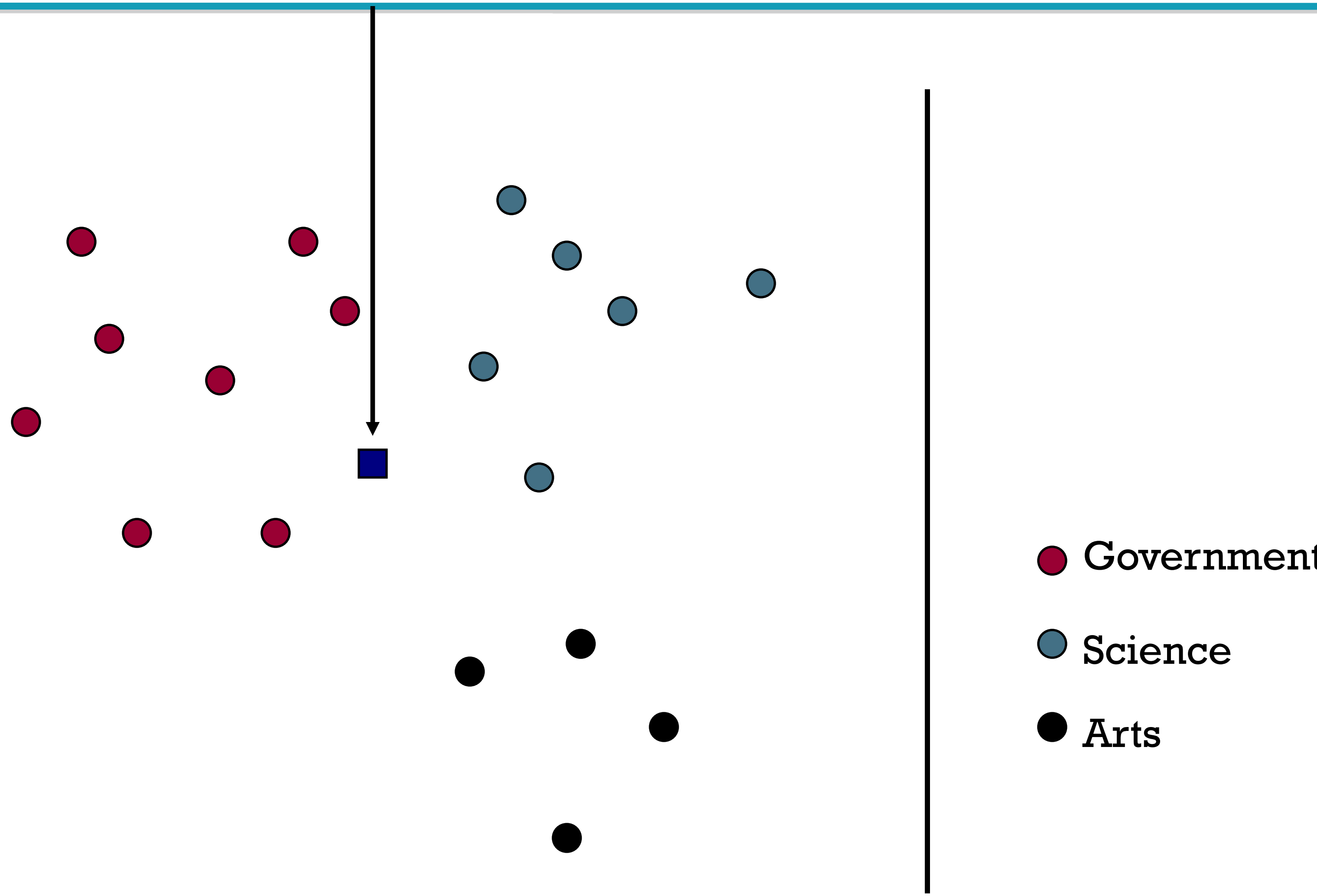
Classification Using Vector Spaces

- 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

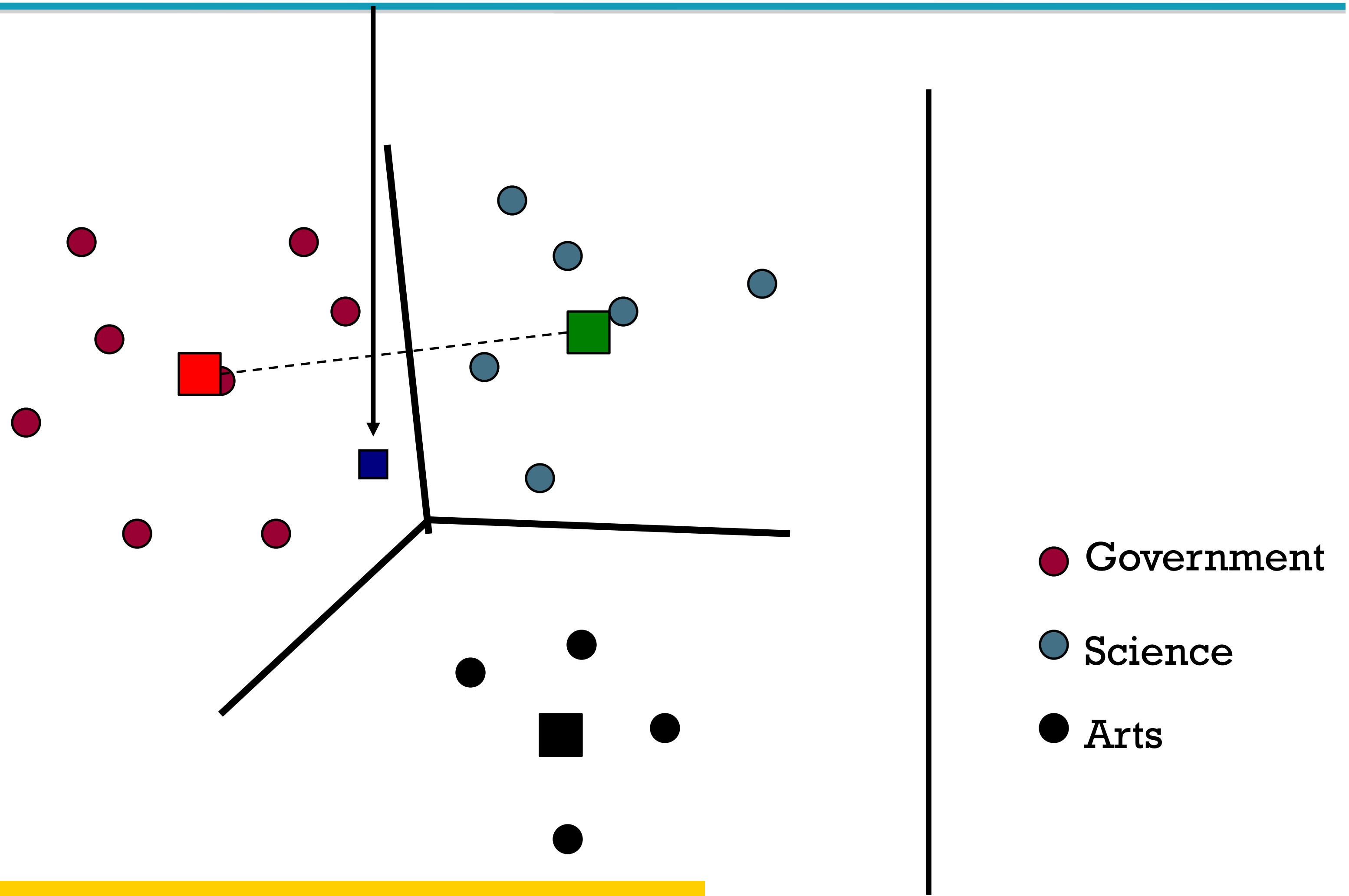
Documents in a Vector Space



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 $v(d)$ 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	w_i	t_i
0.70	prime	−0.71	dlrs
0.67	rate	−0.35	world
0.63	interest	−0.33	sees
0.60	rates	−0.25	year
0.46	discount	−0.24	group
0.43	bundesbank	−0.24	dlr

- To classify, find dot product of feature vector and weights

Rocchio classification

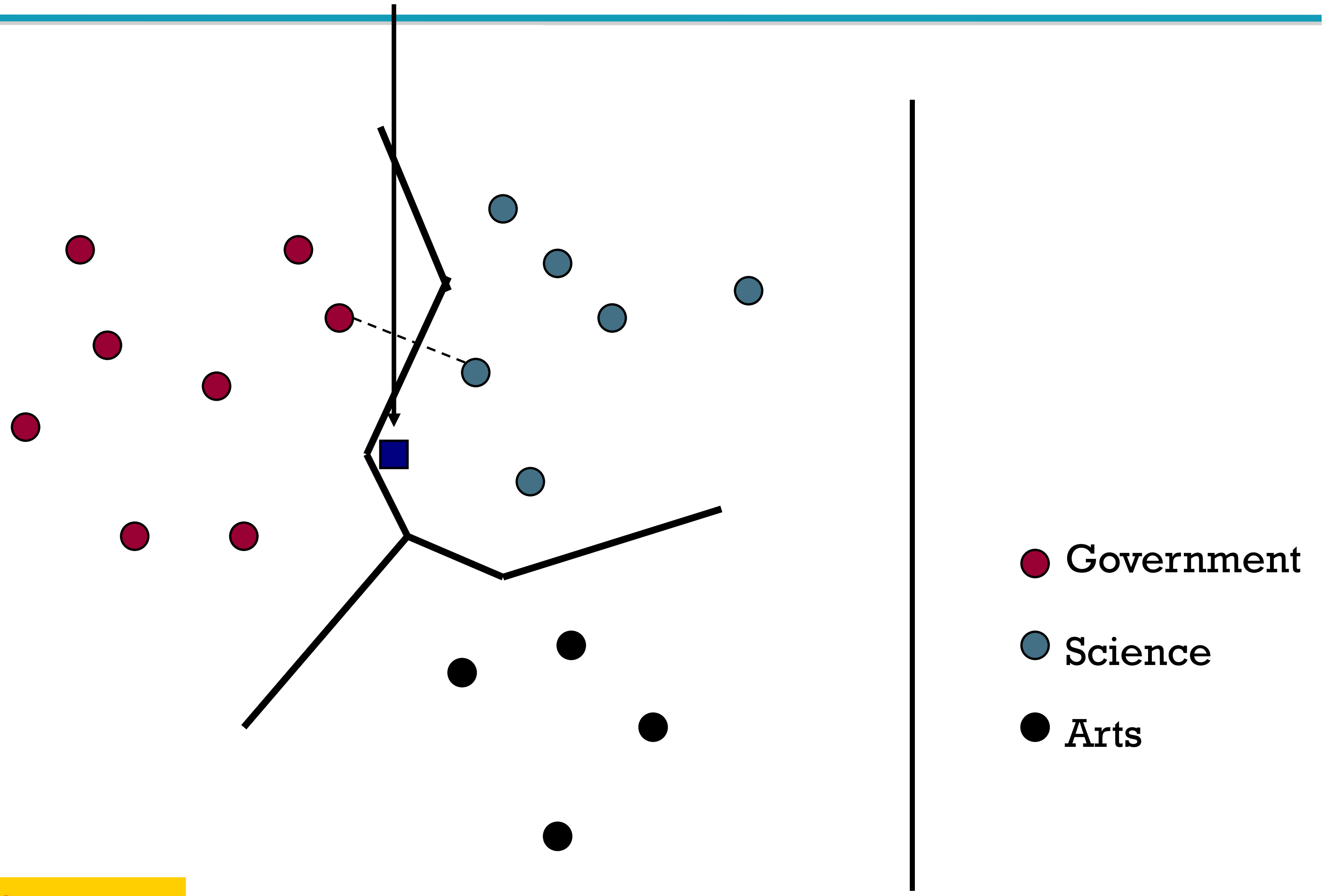
- A simple form of Fisher's linear discriminant
- Little used outside text classification
  - It has been used quite effectively for text classification
  - But in general worse than Naïve Bayes
- Again, cheap to train and test documents

k Nearest Neighbor Classification

- kNN = *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) 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 $B << |D|$

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

Bias vs. capacity – notions and terminology

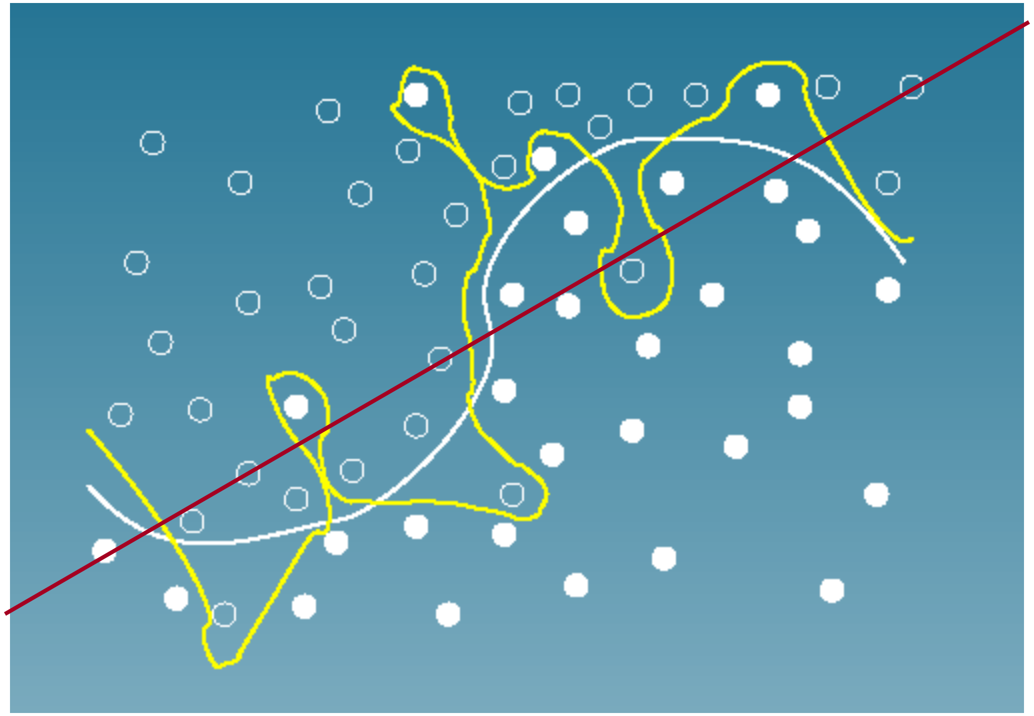
- Consider asking a botanist: Is an object a tree?
  - Too much *capacity*, low *bias*
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., different # of leaves)
  - Not enough capacity, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - You want the middle ground

(Example due to C. Burges)

kNN vs. Naive Bayes

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- Rocchio/NB has low variance and high bias.
  - Linear decision surface between classes

Bias vs. variance:
Choosing the correct model capacity



Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional
  - "The curse of dimensionality"
- High-bias algorithms should generally work best in high-dimensional space
  - They prevent overfitting
  - They generalize more
- For most text categorization tasks, there are many relevant features and many irrelevant ones

Which classifier do I use for a given text classification problem?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the data?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.