Web Information Retrieval

Lecture 14

Text classification

Text Classification

- Naïve Bayes Classification

- Vector space methods for Text Classification
  - K Nearest Neighbors
  - Decision boundaries
  - Linear Classifiers

Recall a few probability basics

- For events A and *B:*
- Bayes' Rule

$$P(A,B) = P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

Prior

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Posterior

Probabilistic Methods

- Our focus this lecture
- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Builds a *generative model* that approximates how data is produced
- Uses *prior* probability of each category given no information about an item.
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

Bayes' Rule for text classification

- For a document *d* and a class *c*

- P(c) = Probability that we see a document of class c
- P(d) = Probability that we see document d

$$P(c,d) = P(c \mid d)P(d) = P(d \mid c)P(c)$$

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Naive Bayes Classifiers

Task: Classify a new instance *d* based on a tuple of attribute values $d = \langle x_1, x_2, \ldots, x_n \rangle$ into one of the classes $c_j \in C$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

MAP is "maximum a posteriori" = most likely class

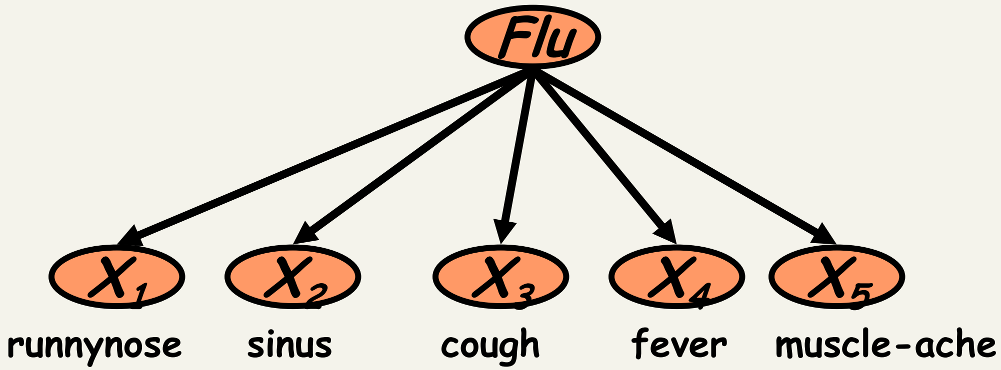
Naive Bayes Classifier: Naive Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \cdot |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naive Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

The Naive Bayes Classifier

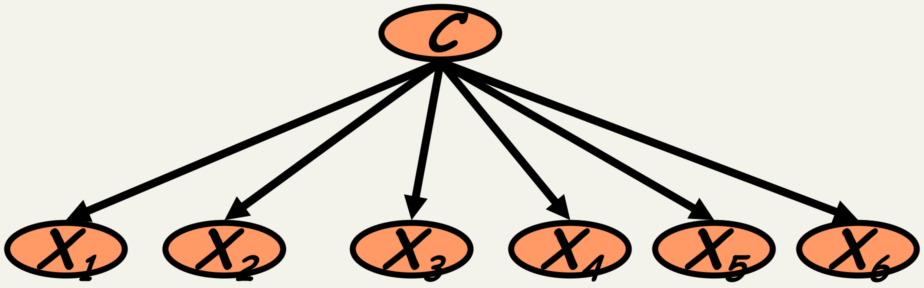


- **Conditional Independence Assumption:** features detect term presence and are **independent** of each other **given the class**:

$$P(X_1, \ldots, X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables
  - Multivariate Bernoulli model

Learning the Model

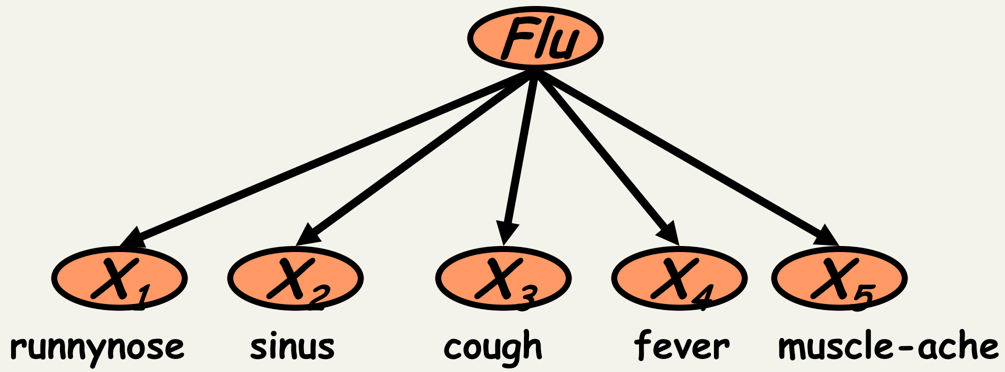


- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{\sum_{w \in \text{Vocabulary}} N(X_i = w, C = c_j)} = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Maximum Likelihood



$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- What if we have seen no training documents with the word ***muscle-ache*** and classified in the topic ***Flu***?

$$\hat{P}(X_5 = t \mid C = Flu) = \frac{N(X_5 = t, C = Flu)}{N(C = Flu)} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Smoothing

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + \alpha}{\sum_{w \in \text{Vocubulary}} (N(X_i = w, C = c_j) + \alpha)} = \frac{N(X_i = x_i, C = c_j) + \alpha}{N(C = c_j) + \alpha \cdot \left|\text{Vocabulary}\right|}$$

- More advanced smoothing is possible

Stochastic Language Models

- Model *probability* of generating strings (each word in turn) in a language (commonly all strings over alphabet ∑). E.g., a unigram model

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

Stochastic Language Models

- Model *probability* of generating any string

Model M1

0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

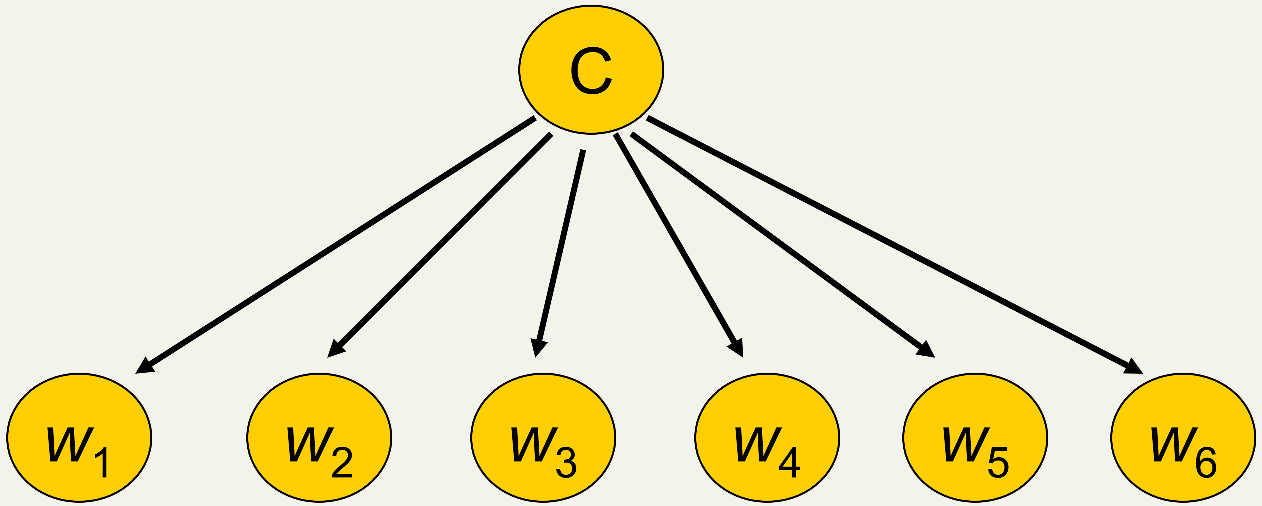
Model M2

0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Naive Bayes via a class conditional language model = multinomial NB



- Effectively, the probability of each class is done as a class-specific unigram language model

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is bag of words model

Naive Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$
  - $Text_j \leftarrow$ single document containing all $docs_j$
  - for each word x_k in *Vocabulary*
    - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
    - $$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Naive Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naive Bayes: Time Complexity

- Training Time: $O(|D|L_{ave} + |C||V|))$ where L_{ave} is the average length of a document in *D.*
  - Assumes all counts are pre-computed in $O(|D|L_{ave})$ time during one pass through all of the data.
  - Generally just $O(|D|L_{ave})$ since usually $|C||V| < |D|L_{ave}$

Why?

- Test Time: $O(|C|\ L_t)$ where L_t is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.

Underflow Prevention: using logs

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

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

- Note that model is now just max of sum of weights…

Naive Bayes Classifier

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}[\log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)]$$

- Simple interpretation: Each conditional parameter log $P(x_i|c_j)$ is a weight that indicates how good an indicator x_i is for c_j.
- The prior log $P(c_j)$ is a weight that indicates the relative frequency of c_j.
- The sum 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 for it

Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- May make using a particular classifier feasible
  - Some classifiers can't deal with 100,000 of features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting

Feature selection: how?

- Two ideas:
  - Hypothesis testing statistics:
    - Are we confident that the value of one categorical variable is associated with the value of another
    - Chi-square test (χ^2)
  - Information theory:
    - How much information does the value of one categorical variable give you about the value of another
    - Mutual information

- They're similar, but χ^2 measures confidence in association, (based on available statistics), while MI measures extent of association (assuming perfect knowledge of probabilities)

Violation of NB Assumptions

- The independence assumptions do not really hold of documents written in natural language.
  - Conditional independence
  - Positional independence

- Examples?

Naive Bayes is Not So Naive

- Naive Bayes won 1st and 2nd place in KDD-CUP 97 competition out of 16 systems
  - Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
- More robust to irrelevant features than many learning methods
  - Irrelevant Features cancel each other without affecting results
  - Decision Trees can suffer heavily from this.
- More robust to concept drift (changing class definition over time)
- Very good in domains with many equally important features
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: Bayes Optimal Classifier
  - Never true for text, but possible in some domains
- Very Fast Learning and Testing (basically just count the data)
- Low Storage requirements

Summary: Naïve Bayes classifiers

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

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

- Training is done by counting and dividing:

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

- Don't forget to smooth

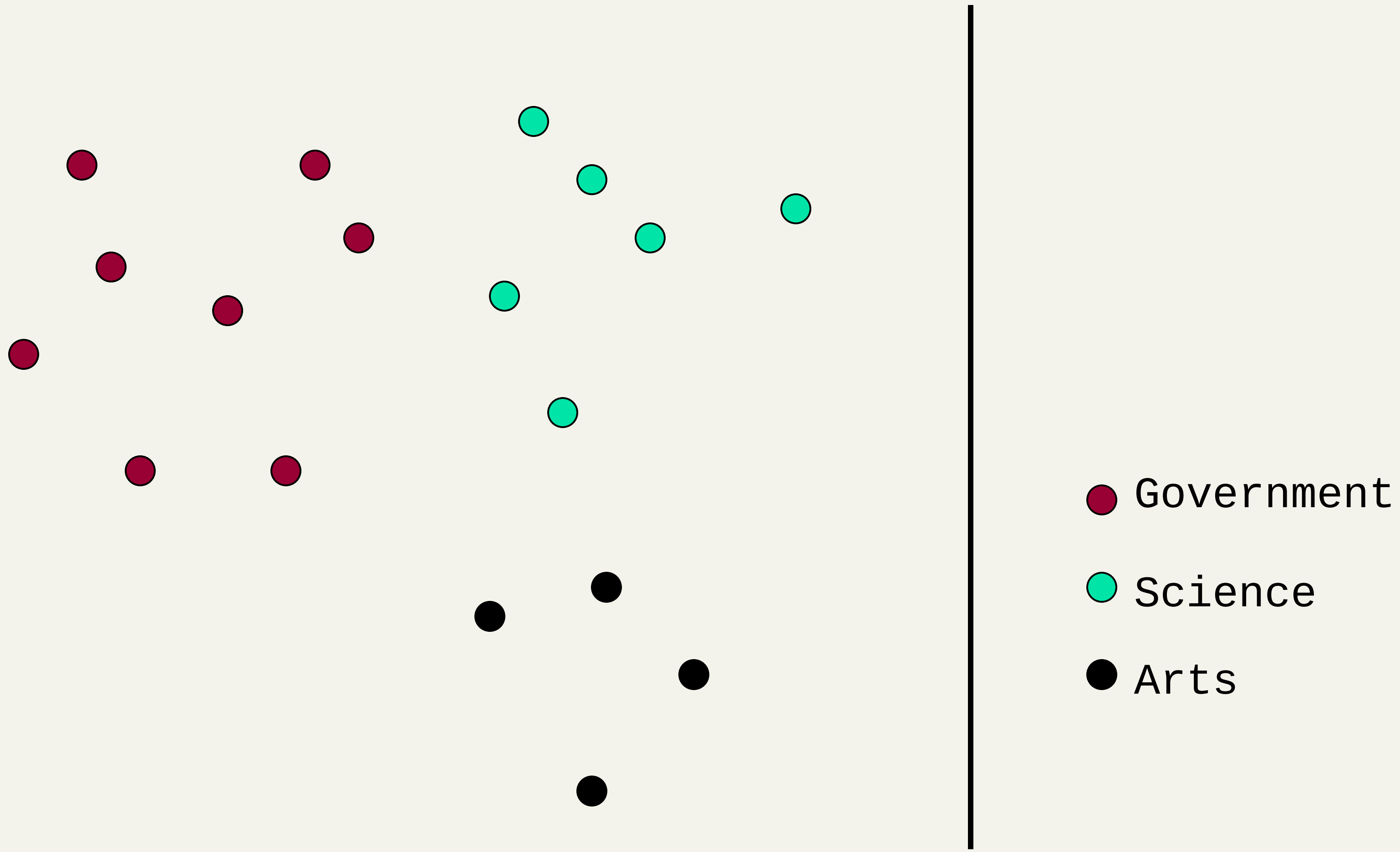
Recall: Vector Space Representation

- 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?

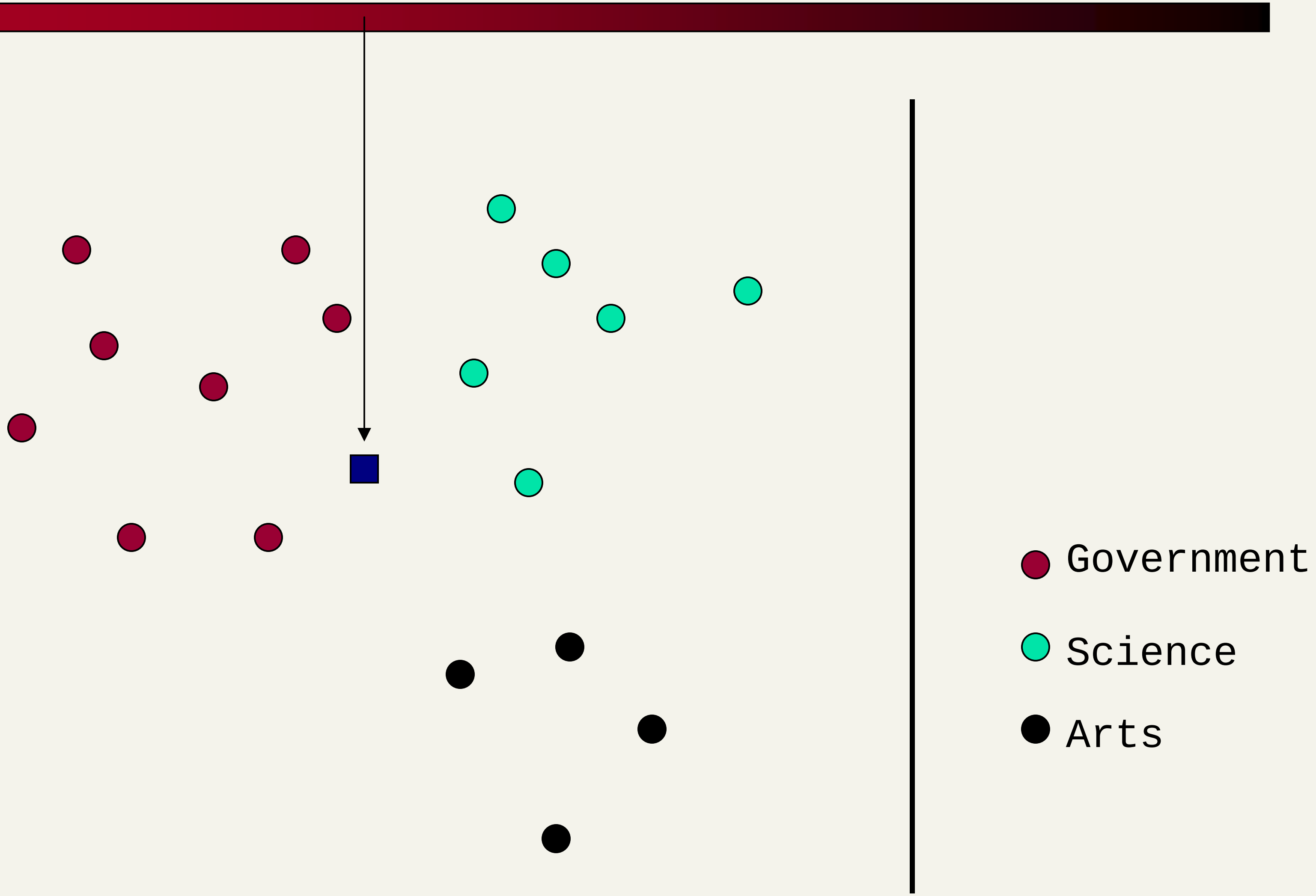
Classification Using Vector Spaces

- As before, the training set is a set of documents, each labeled with its class (e.g., topic)
- In vector space classification, this set corresponds to a labeled set of points (or, equivalently, vectors) in the vector space
- **Premise 1:** Documents in the same class form a contiguous region of space
- **Premise 2:** Documents from different classes don't overlap (much)
- We define surfaces to delineate classes in the space

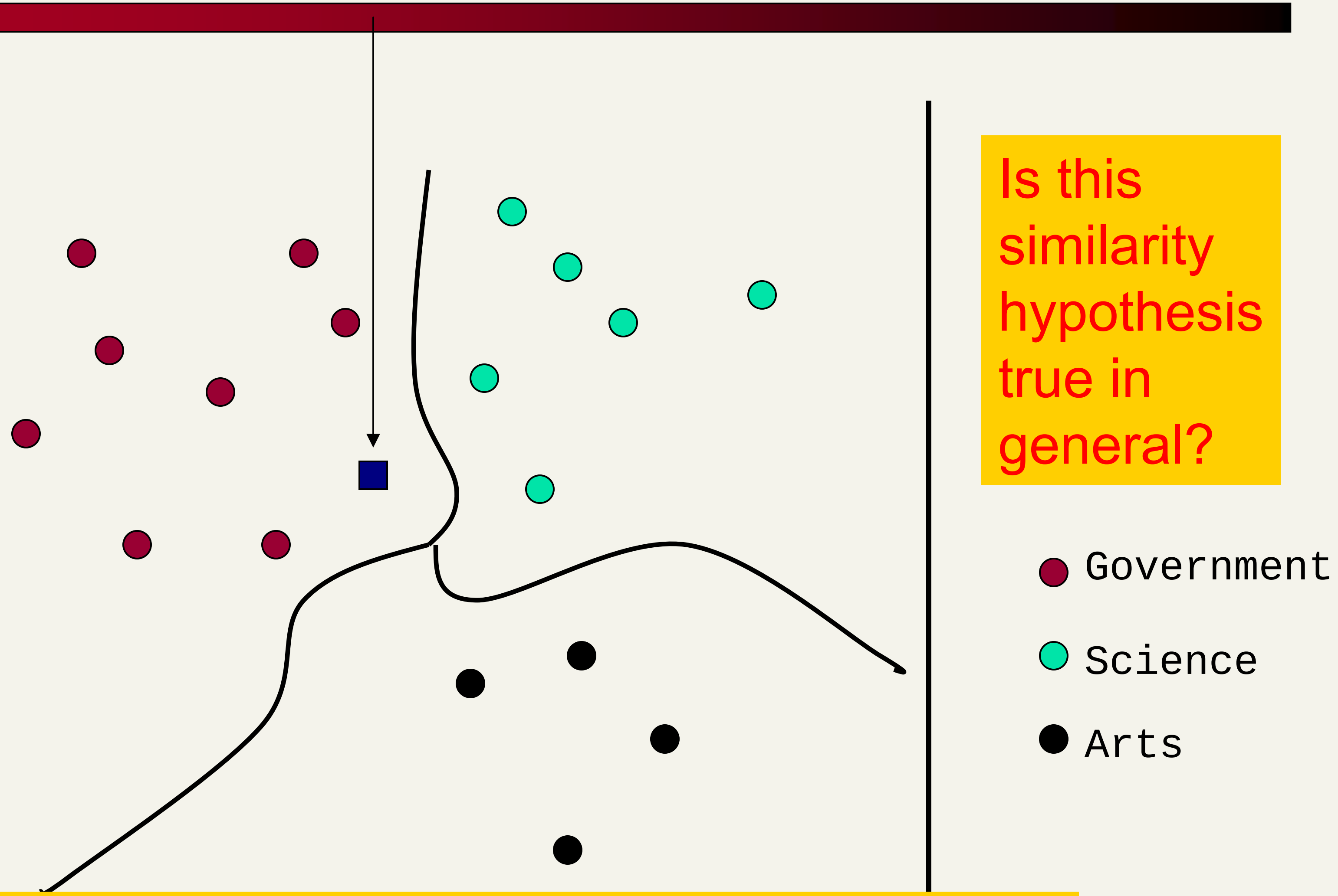
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Is this similarity hypothesis true in general?

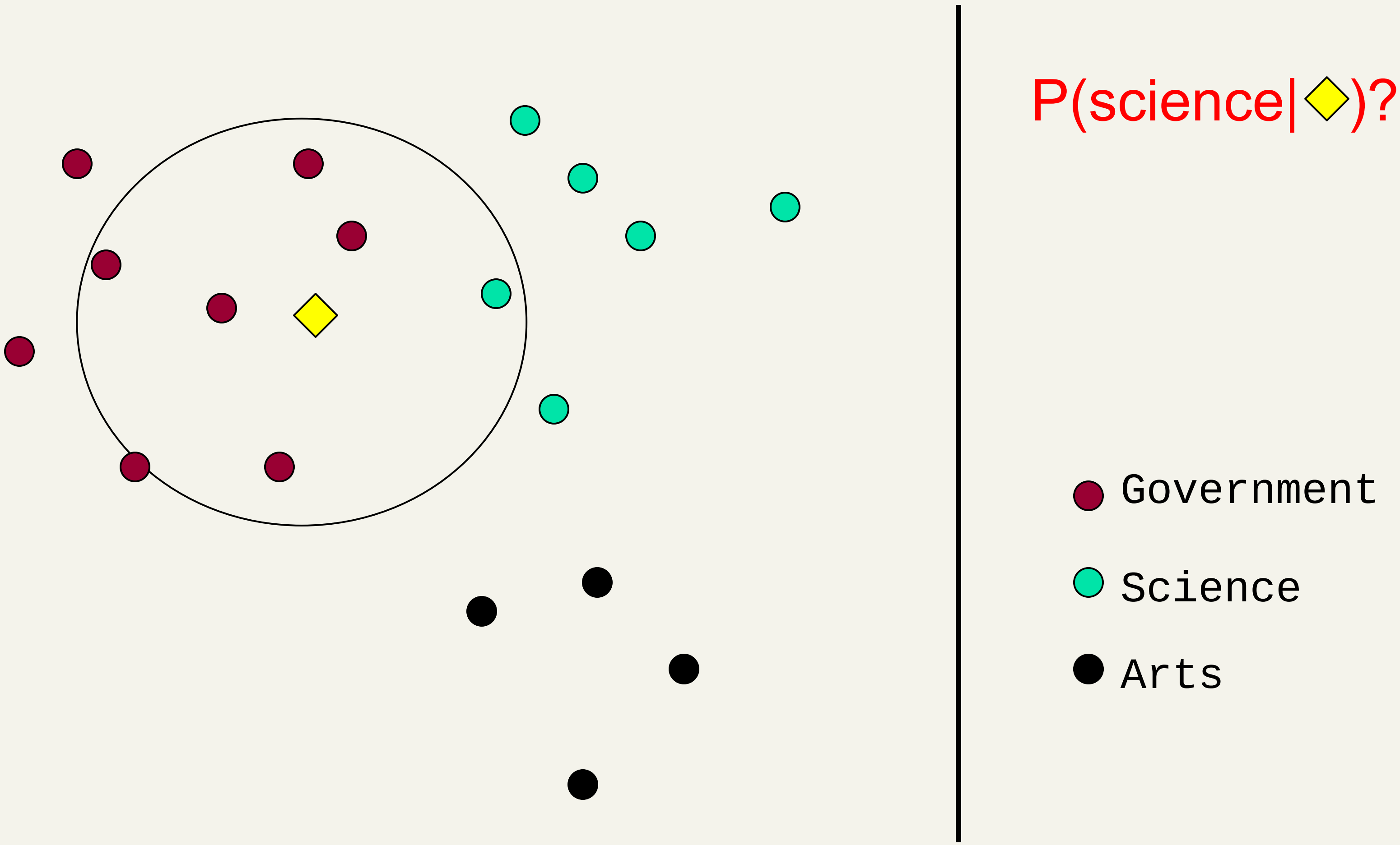
Our main topic today is how to find good separators

k Nearest Neighbor Classification

- kNN = k Nearest Neighbor

- To classify document *d* into class c:
- Define *k*-neighborhood N as *k* nearest neighbors of *d*
- Count number of documents i in N that belong to c
- Assign *d* to class *c* with most documents

Example: k=6 (6NN)



Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in *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 explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning
- Rationale of kNN: contiguity hypothesis

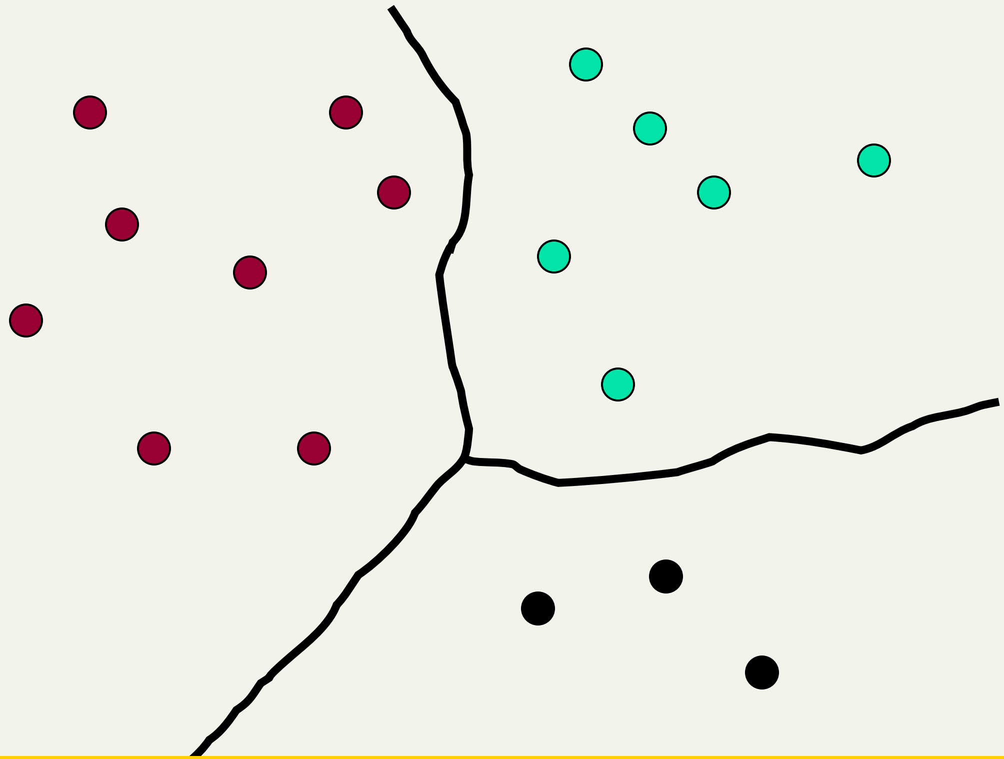
kNN Is Close to Optimal

- Cover and Hart (1967)
- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate [error rate of classifier knowing model that generated data]
- In particular, asymptotic error rate is 0 if Bayes rate is 0.
- Assume: query point coincides with a training point.
- Both query point and training point contribute error → 2 times Bayes rate

k Nearest Neighbor

- Using only the closest example (1NN) to determine the class 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 alternative is to find the *k* most-similar examples and return the majority category of these *k* examples.
- Value of *k* is typically odd to avoid ties; 3 and 5 are most common.

kNN decision boundaries



Government

Science

Arts

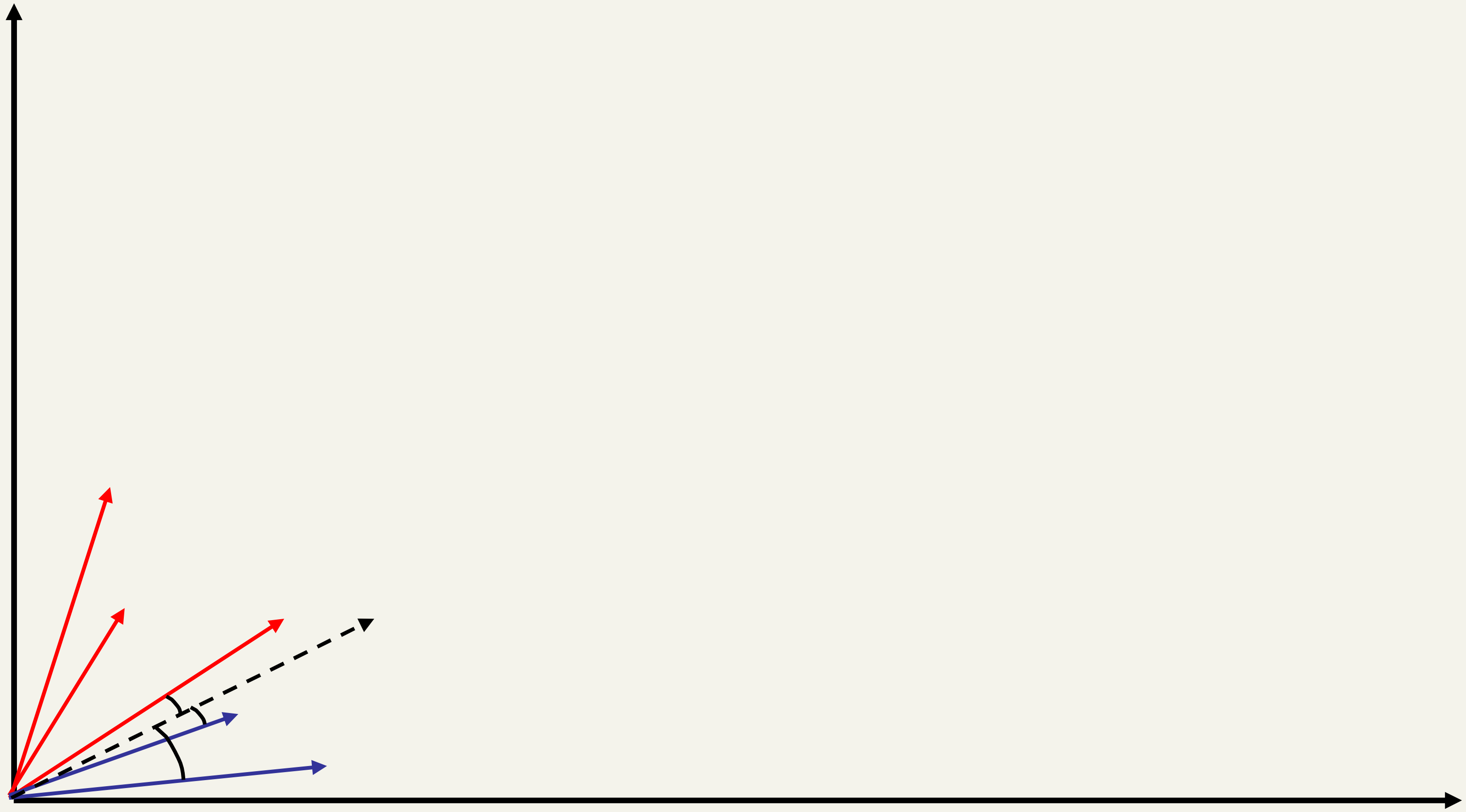
Boundaries are in principle arbitrary surfaces – but usually polyhedra

kNN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, etc.)

Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous *m*-dimensional instance space is *Euclidean distance*.
- Simplest for *m*-dimensional binary instance space is *Hamming distance* (number of feature values that differ).
- For text, cosine similarity of tf.idf weighted vectors is typically most effective.

Illustration of 3 Nearest Neighbor for Text Vector Space



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.

kNN: Discussion

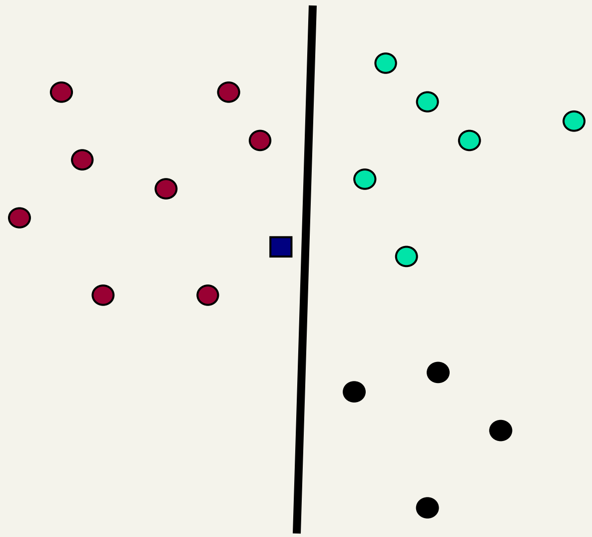
- No feature selection necessary
- Scales well with large number of classes
  - Don’t need to train *n* classifiers for *n* classes
- Scores can be hard to convert to probabilities
- No training necessary
- May be more expensive at test time

Linear classifiers and binary and multiclass classification

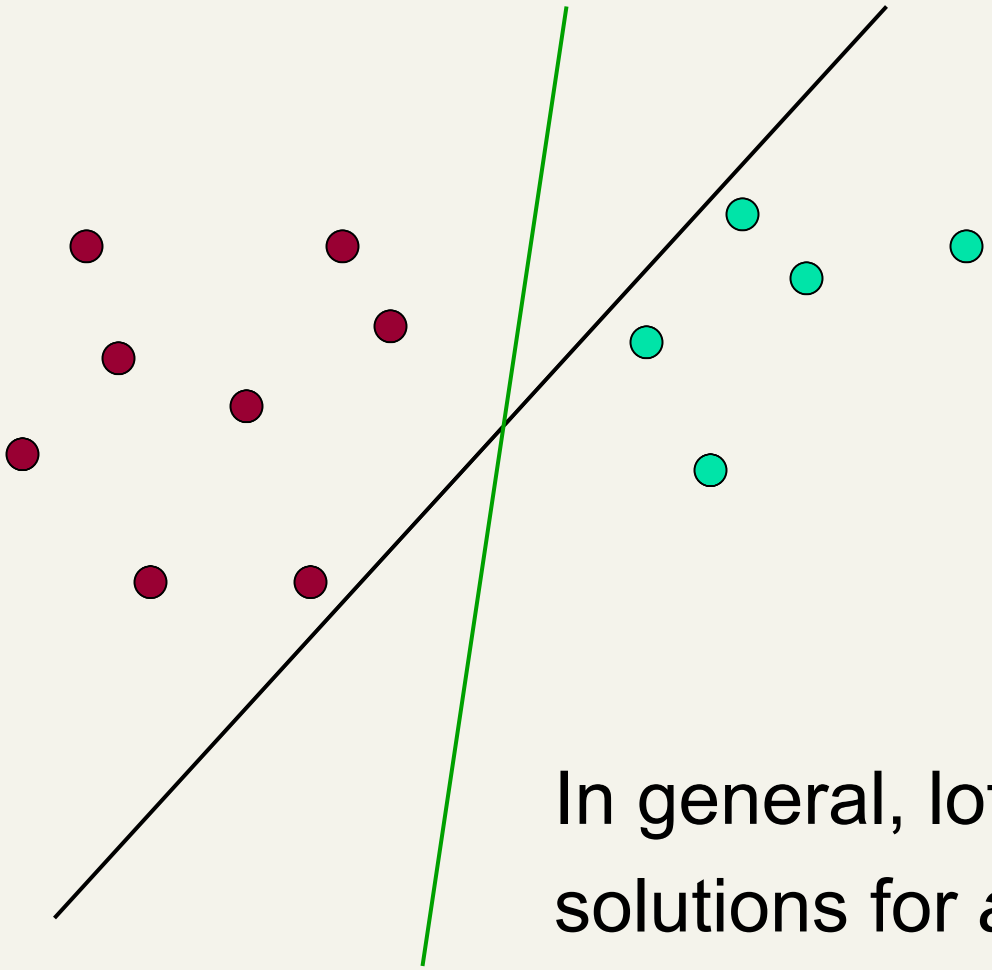
- Consider 2 class problems
  - Deciding between two classes, perhaps, government and non-government
    - One-versus-rest classification
- How do we define (and find) the separating surface?
- How do we decide which region a test doc is in?

Separation by Hyperplanes

- A strong high-bias assumption is *linear separability*:
  - in 2 dimensions, can separate classes by a line
  - in higher dimensions, need hyperplanes
- Can find separating hyperplane by *linear programming* (or can iteratively fit solution via perceptron):
  - separator can be expressed as $ax + by = c$



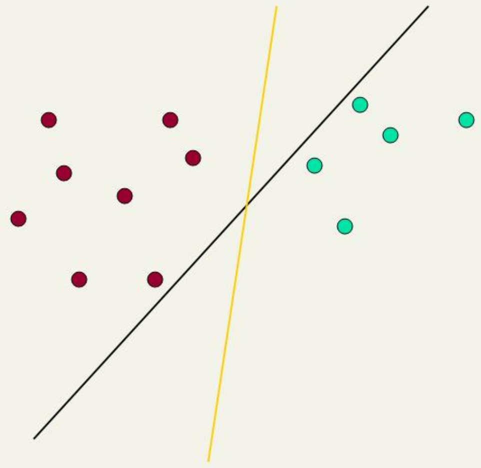
Which Hyperplane?



In general, lots of possible solutions for *a,b,c.*

Which Hyperplane?

- Lots of possible solutions for *a,b,c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Most methods find an optimal separating hyperplane
- Which points should influence optimality?
  - All points
    - Linear regression
    - Naïve Bayes
  - Only “difficult points” close to decision boundary
    - Support vector machines



Naive Bayes is a linear classifier

- Two-class Naive Bayes. We compute:

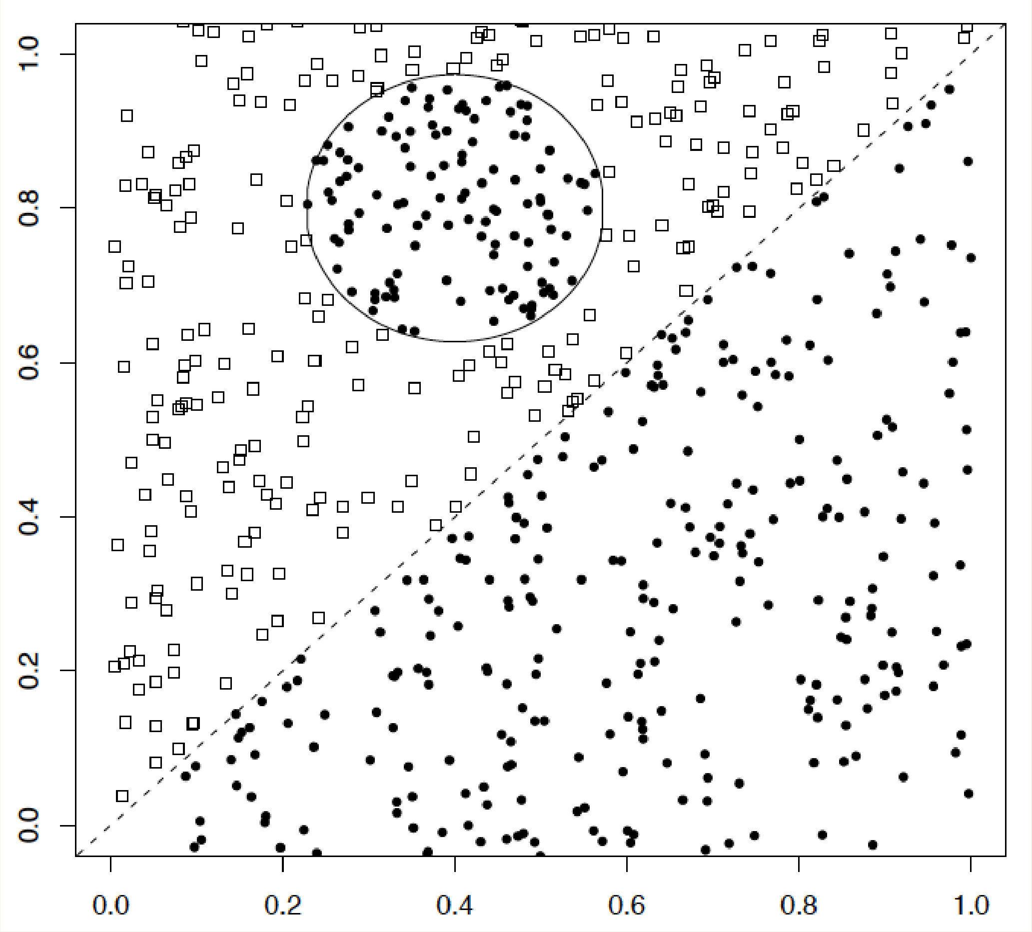
$$\log \frac{P(C \mid d)}{P(\overline{C} \mid d)} = \log \frac{P(C)}{P(\overline{C})} + \sum_{w \in d} \log \frac{P(w \mid C)}{P(w \mid \overline{C})}$$

- Decide class *C* if the odds is greater than 1, i.e., if the log odds is greater than 0.
- So decision boundary is hyperplane:

$$\alpha + \sum_{w \in V} \beta_w \times n_w = 0 \quad \text{where } \alpha = \log \frac{P(C)}{P(\overline{C})};$$

$$\beta_w = \log \frac{P(w \mid C)}{P(w \mid \overline{C})}; \quad n_w = \#\text{ of occurrences of } w \text{ in } d$$

A nonlinear problem



- A linear classifier like Naïve Bayes does badly on this task
- kNN will do very well (assuming enough training data)

Resources

- IIR Chapters 13 – 13.2, 13.5.0
- IIR Chapters 14 – 14.1, 14.3, 14.4