Information Retrieval

BITS Pilani
Pilani Campus

Abhishek
February 2020

innovate achieve lead

BITS Pilani

Pilani Campus

CS F469, Information Retrieval

Lecture topics: Text Classification

innovate achieve lead

BITS Pilani

Pilani Campus

Most of these slides are based on:

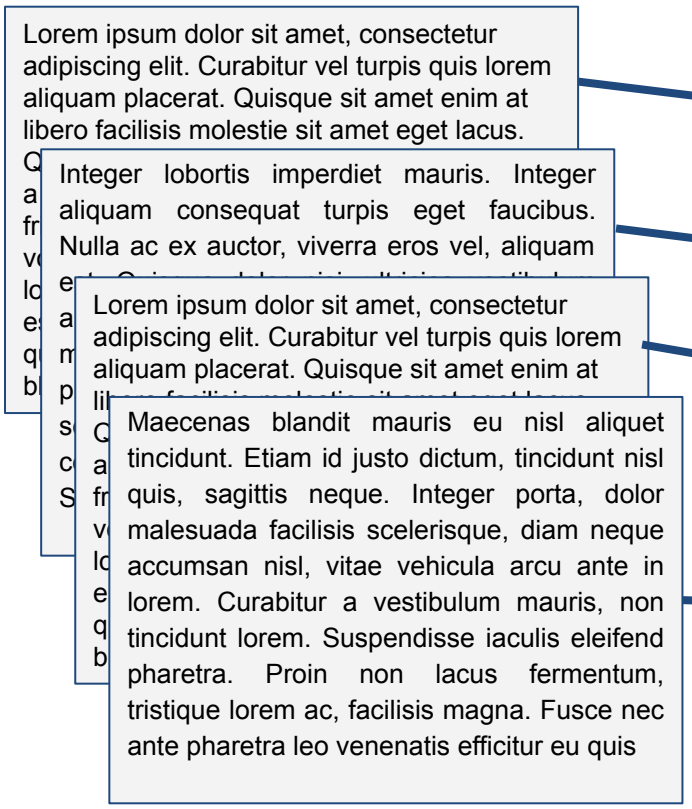
https://web.stanford.edu/class/cs276/

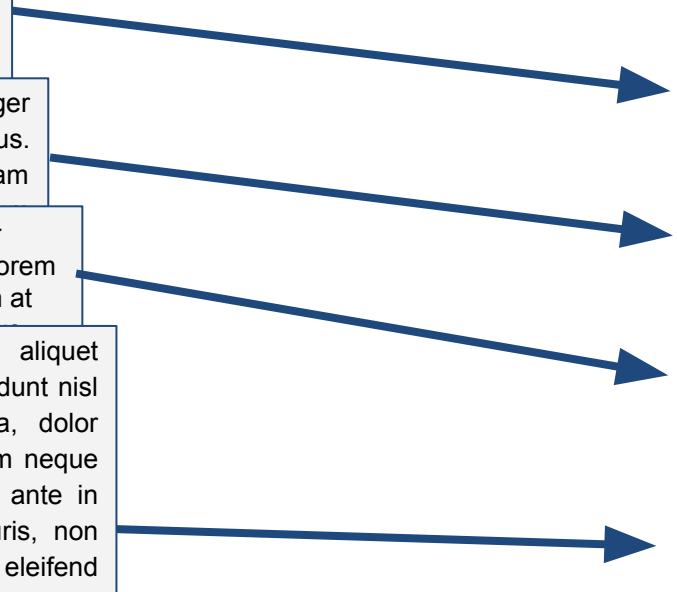
https://www.cis.uni-muenchen.de/~hs/teach/14s/ir/

This Lecture

- Text Classification
  - Naive Bayes Classifier
  - Rocchio Text Categorization
  - KNN

Text Classification Problem





For every input assign an output. Outputs are called as labels or categories.

How Search Engines Uses Text Classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. non spam).
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative).
- Topic-specific or vertical search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not) (Text classification).

Formal Definition of Text Classification

Given:

- A document space **X**
  - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes **C** = $\{c_1, c_2, \ldots, c_J\}$.
  - The classes are human-defined for the needs of an application (e.g., spam vs. non spam).
- A training set **D** of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbf{X} \times \mathbf{C}$

- Using a learning method or learning algorithm, we then wish to learn a classifier γ that maps documents to classes:

$$\gamma : \mathbf{X} \rightarrow \mathbf{C}$$

Classification methods:

- Manual
- Rule Based
- Supervised Learning (Statistical/Probabilistic classifiers)

Naive Bayes Classifier

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We are interested in computing :
  - P(class | document) or P(c|d)
- Apply Bayes Rule:

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

- Now:

$$P(c \mid d) \propto P(d \mid c)P(c)$$

- How to compute P(d | c) and P(c)?

Naive Bayes Assumptions

Naive Bayes Assumptions

- A document consists of sequence of terms:
  - Document = $<t_1, t_2, t_3, \ldots t_{nd}>$
  - Thus, $P(d \mid c) = P(<t_1, t_2, t_3, \ldots t_{nd}> \mid c)$

Two assumptions:

- Conditional Independence assumption
  - Occurance of term in a document in independent of occurrence of any other term.
- Positional Independence assumption
  - A term can be present in any i^{th} position independently.

Naive Bayes Assumptions

- A document consists of sequence of terms:
  - Document = $<t_1, t_2, t_3, \ldots t_{nd}>$
  - Thus, $P(d \mid c) = P(<t_1, t_2, t_3, \ldots t_{nd}> \mid c)$

Two assumptions:

- Conditional Independence assumption
  - Occurance of term in a document in independent of occurrence of any other term.
- Positional Independence assumption
  - A term can be present in any i^{th} position independently.

Using assumptions, we can simplify:

$$P(d \mid c) = P(<t_1, t_2, t_3, \ldots t_{nd}> \mid c) = \prod_{1 \leq k \leq n_d} P(t_k \mid c)$$

The Naive Bayes classifier

- So, we can write:

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

- $\mathbf{n_d}$ is the length of the document. (number of tokens).
- P($\mathbf{t_k}$ | **c**) is the conditional probability of term $\mathbf{t_k}$ occurring in a document of class **c.**
- P($\mathbf{t_k}$ | **c**) as a measure of how much evidence $\mathbf{t_k}$ 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

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}:

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

- We use here P (hat) instead of P as we do not know the true values of parameters P(c) and $P(t_k|c)$, but we estimate it from training dataset as we will see in a moment.

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:

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

Naive Bayes classifier: Summary

- Classification rule:

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

- 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.

Parameter estimation take 1: Maximum likelihood

Parameter estimation take 1: Maximum likelihood

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

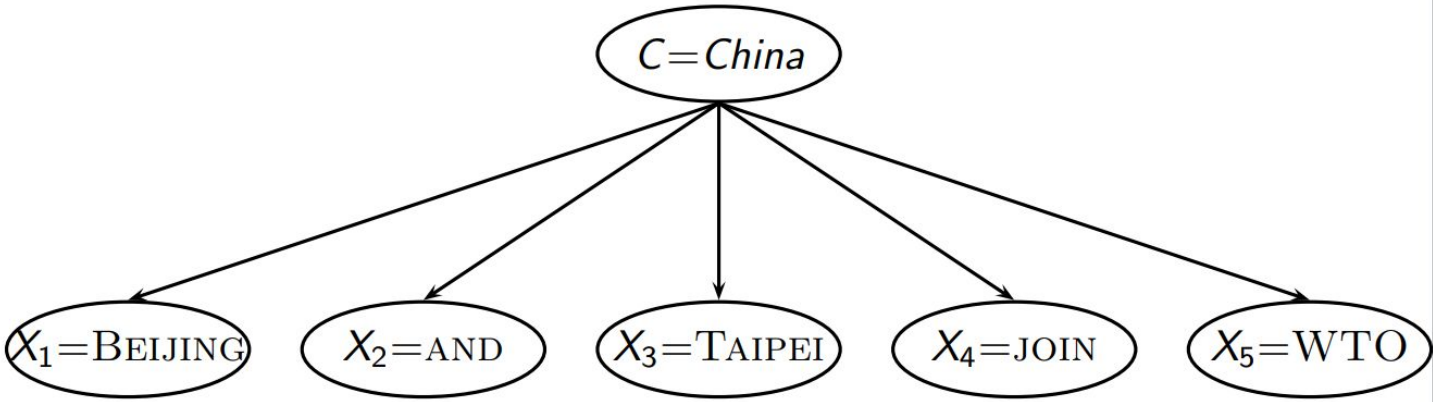
$$\hat{P}(c) = \frac{N_c}{N}$$

- 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)

The problem with maximum likelihood estimates: Zeros



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

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

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

The problem with maximum likelihood estimates: Zeros (cont)

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

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

- $\rightarrow$ We will get $P(China|d) = 0$ for any document that contains WTO!

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 bins – in this case the number of different words or the size of the vocabulary $|V| = M$

Naive Bayes: Summary

- 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

Exercise: Estimate parameters, classify test set

	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	?

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

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

(B is the number of bins – in this case the number of different words or the size of the vocabulary $|V| = M$)

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

Example: Parameter estimates

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

$$\hat{P}(\text{CHINESE}|c) = (5+1)/(8+6) = 6/14 = 3/7$$

Example: Parameter estimates

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

$$\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

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 $c = \textit{China}$. 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.

Vector Space Classification Methods

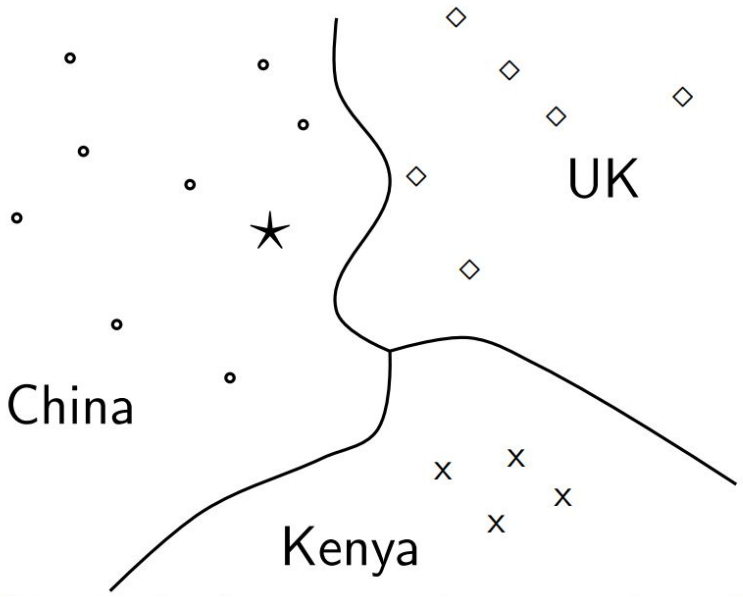
Recall: Vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions.
- Normalize vectors (documents) to unit length.
- How can we do classification in this space?

Vector Space Classification

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

Classes in the vector space



Should the document ⋆ be assigned to China, UK or Kenya?

Find separators between the classes.

Based on these separators: ⋆ should be assigned to China.

How do we find separators that do a good job at classifying new documents like ⋆?

Rocchio classification

Rocchio classification: Basic idea

- Compute a centroid for each class
  - The centroid is the average of all documents in the class.
- Assign each test document to the class of its closest centroid.

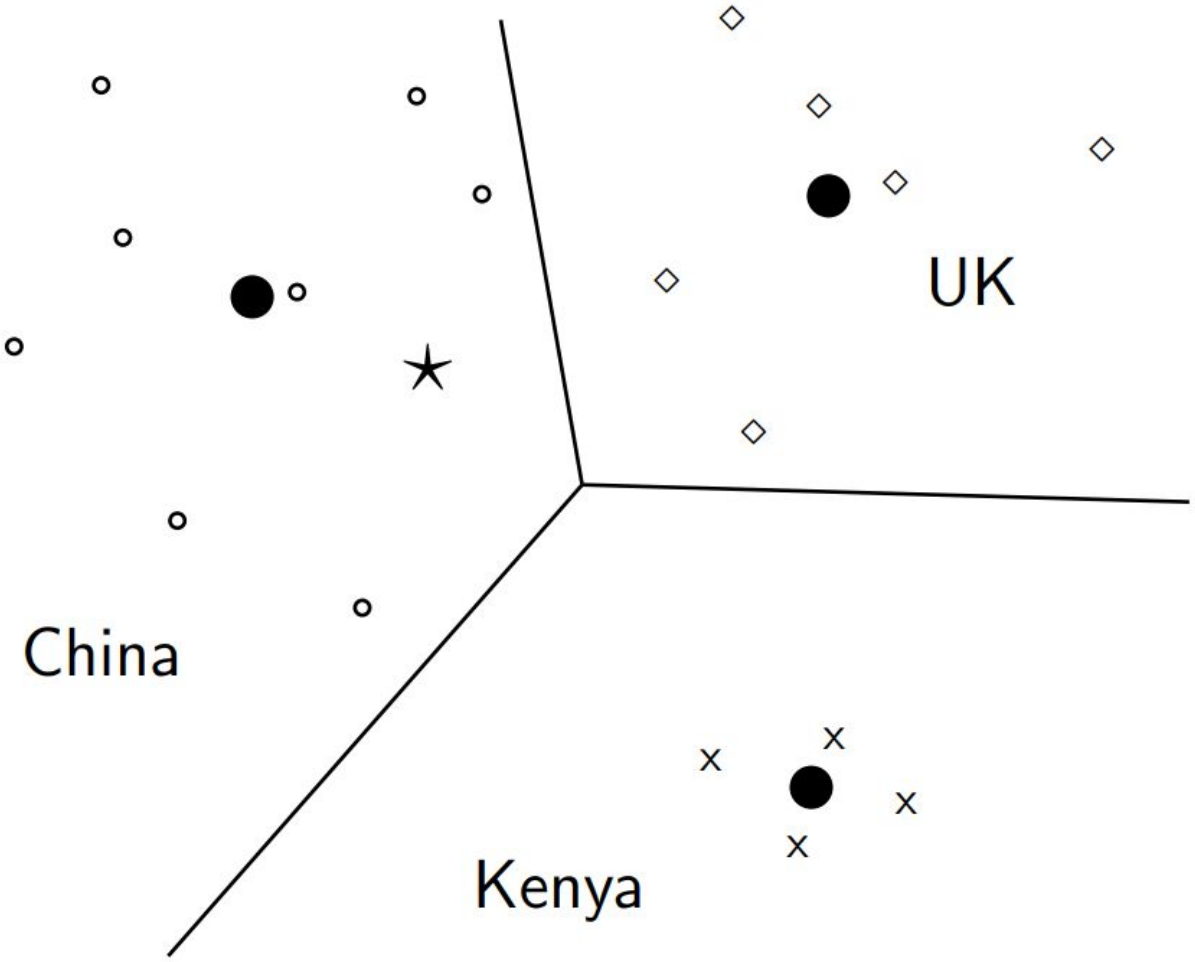
Recall definition of centroid

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

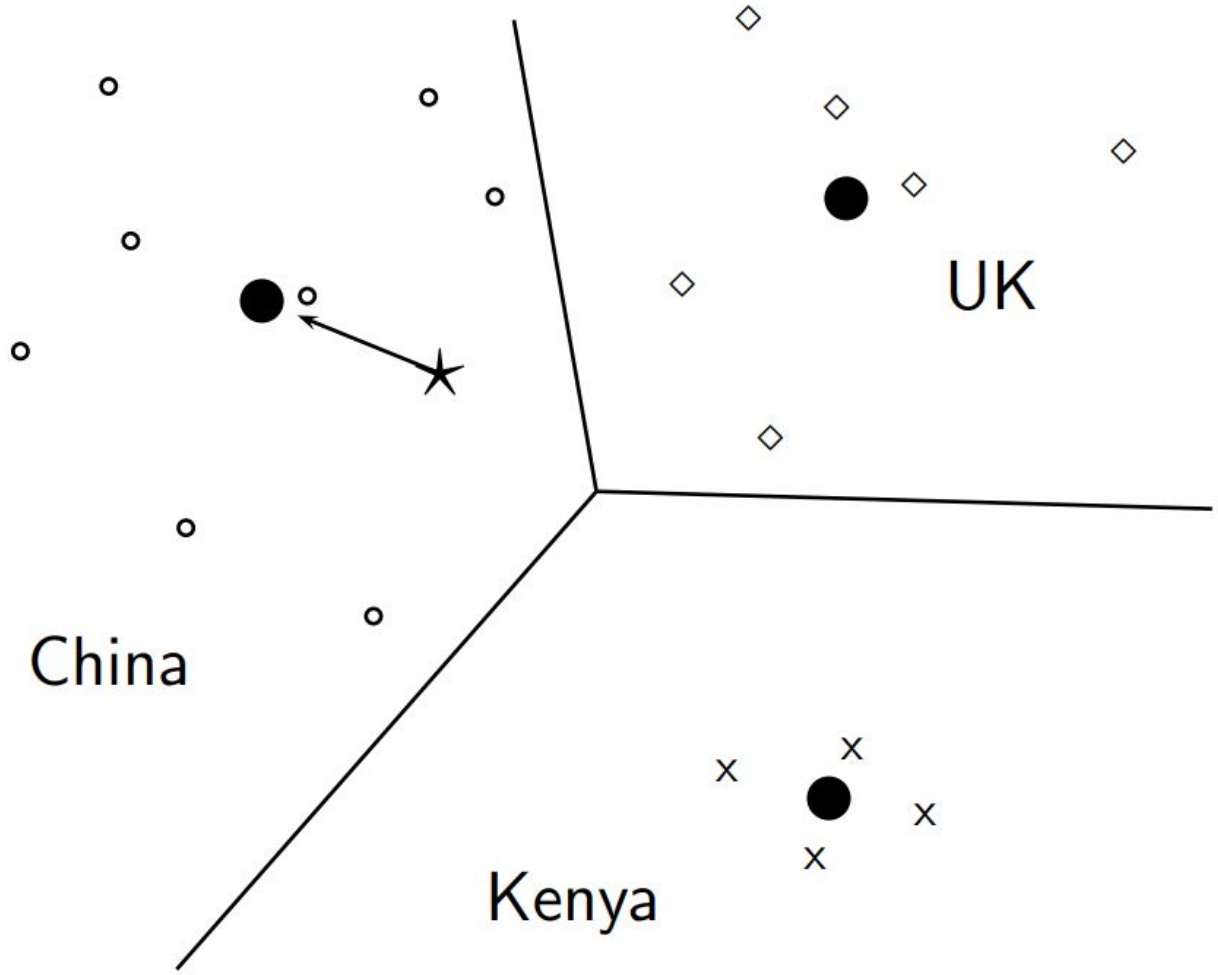
where D_c is the set of all documents that belong to class c and

$\vec{v}(d)$ is the vector space representation of d.

Rocchio illustrated



Rocchio illustrated



Rocchio algorithm

TRAINROCCHIO($\mathbb{C}, \mathbb{D}$)

1 **for each** $c_j \in \mathbb{C}$
2 **do** $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$
3 $\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
4 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

APPLYROCCHIO($\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}, d$)

1 **return** $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$

Rocchio properties

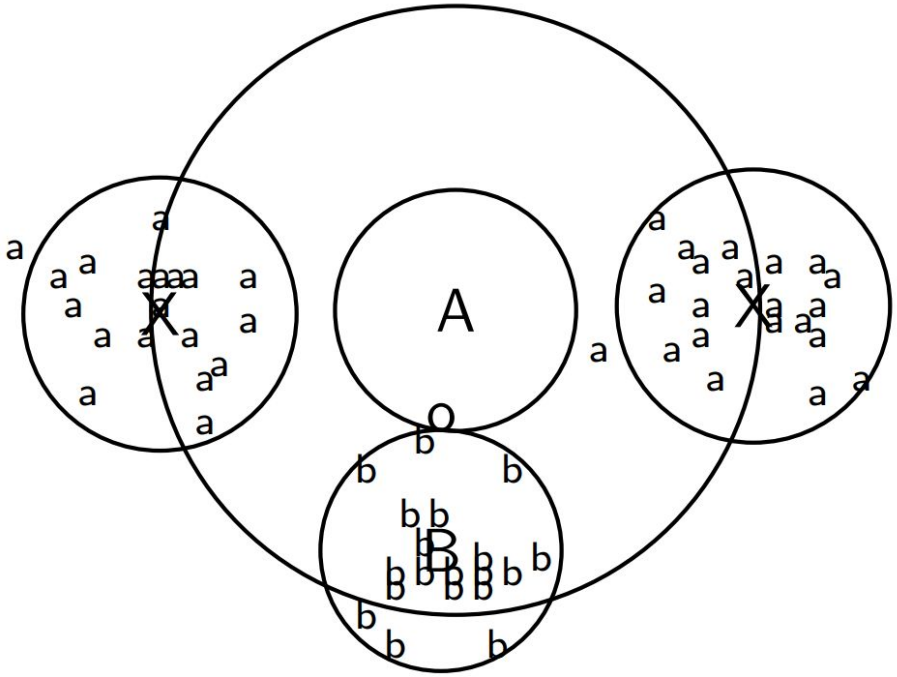
- Rocchio forms a simple representation for each class: the centroid.
  - We can interpret the centroid as the prototype of the class.

- Classification is based on similarity to / distance from centroid/prototype.
- Does not guarantee that classifications are consistent with the training data!

Rocchio vs. Naive Bayes

- In many cases, Rocchio performs worse than Naive Bayes.
- One reason: Rocchio does not handle nonconvex, multimodal classes correctly.

Rocchio cannot handle nonconvex, multimodal classes



- A is centroid of the a's, B is centroid of the b's.
- The point o is closer to A than to B.
- But o is a better fit for the b class.
- A is a multimodal class with two prototypes.
- But in Rocchio we only have one prototype.

k Nearest Neighbour (kNN) classification

kNN classification

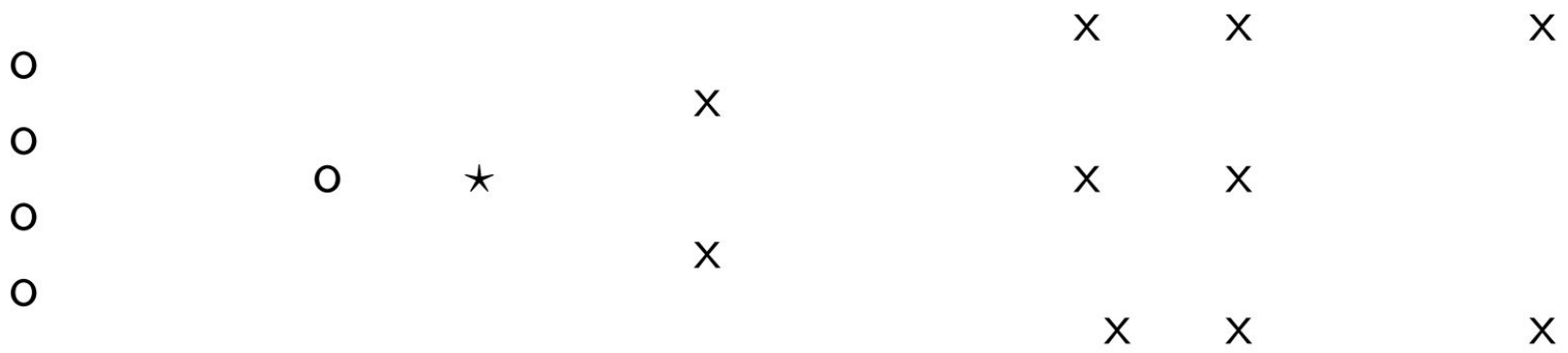
- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.
- kNN is more accurate (in most cases) than Naive Bayes and Rocchio.
- If you need to get a pretty accurate classifier up and running in a short time . . .
- . . . and you don’t care about efficiency that much . . .
- . . . use kNN.

kNN classification

kNN classification

- **kNN classification rule for k = 1 (1NN):**
  - Assign each test document to the class of its nearest neighbor in the training set.
  - 1NN is not very robust – one document can be mislabeled or atypical.
- **kNN classification rule for k > 1 (kNN):**
  - Assign each test document to the majority class of its **k** nearest neighbors in the training set.
- **Rationale of kNN: contiguity hypothesis**
  - We expect a test document **d** to have the same label as the training documents located in the local region surrounding **d**.

Exercise



How is star classified by:
(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN (v) Rocchio?

kNN Algorithm

TRAIN-KNN($\mathbb{C}, \mathbb{D}$)
1 $\mathbb{D}' \leftarrow$ PREPROCESS($\mathbb{D}$)
2 $k \leftarrow$ SELECT-K($\mathbb{C}, \mathbb{D}'$)
3 **return** $\mathbb{D}', k$

APPLY-KNN($\mathbb{D}', k, d$)
1 $S_k \leftarrow$ COMPUTENEARESTNEIGHBORS($\mathbb{D}', k, d$)
2 **for each** $c_j \in \mathbb{C}(\mathbb{D}')$
3 **do** $p_j \leftarrow |S_k \cap c_j|/k$
4 **return** $\arg\max_j p_j$

kNN Discussion

- kNN is very accurate if training set is large.
- But kNN can be very inaccurate if training set is small
- Scales well with large number of classes
- Don’t need to train n classifiers for n classes
- May be expensive at test time
- In most cases it’s more accurate than NB or Rocchio.

Till Now

- Working of Naive Bayes Classifier
- Working of Rocchio Classifier
- Working of kNN Classifier

Today will discuss some properties of these algorithm and other related topics to text classification

Properties of Naive Bayes

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:

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

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.

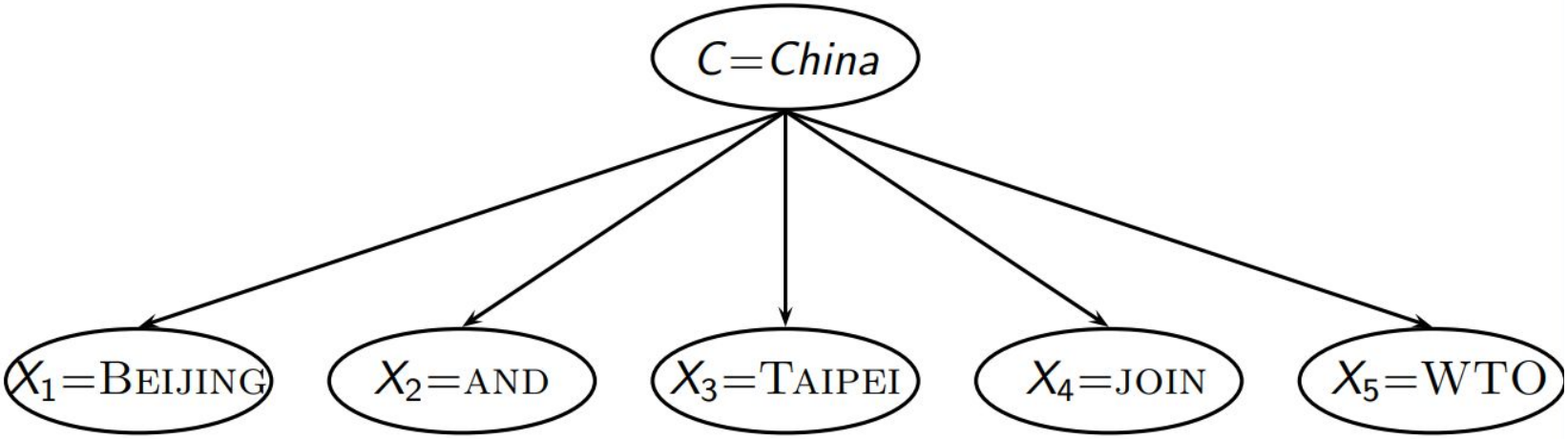
Naive Bayes conditional independence assumption

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 $P(X_k = t_k|c)$. Recall from earlier the estimates for these conditional probabilities: $\hat{P}(t|c) = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B}$

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

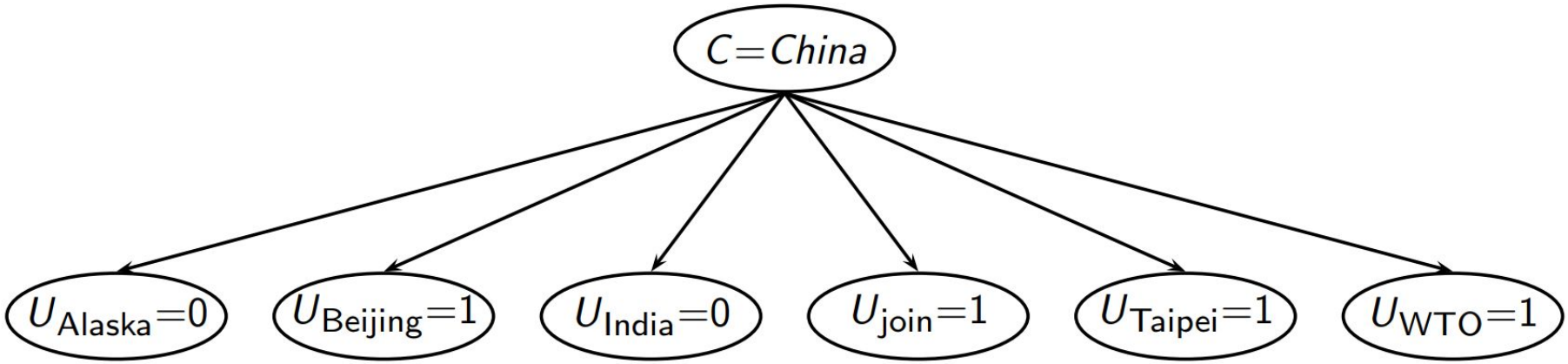
innovate achieve lead

- $\hat{P}(X_{k_1} = t|c) = \hat{P}(X_{k_2} = t|c)$
- 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.

Doc 1: China sues France

Doc 2: France sues China

A different Naive Bayes model: Bernoulli model



Multinomial vs Bernoulli model

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

Violation of Naive Bayes independence assumptions

- Conditional independence:

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

- Positional independence:
- $\hat{P}(X_{k_1} = t|c) = \hat{P}(X_{k_2} = t|c)$
- The independence assumptions do not really hold of documents written in natural language.
- Exercise
  - Examples for why conditional independence assumption is not really true?
  - Examples for why positional independence assumption is not really true?
- How can Naive Bayes work if it makes such inappropriate assumptions?

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class and not about accurately estimating probabilities.

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class and not about accurately estimating probabilities.
- Naive Bayes is terrible for correct estimation ...

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class and not about accurately estimating probabilities.
- Naive Bayes is terrible for correct estimation ...
- ... but if often performs well at accurate prediction (choosing the correct class).

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to non relevant features than some more complex learning methods.
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods.
- Better than methods like decision trees when we have many equally important features.
- A good dependable baseline for text classification (but not the best).
- Optimal if independence assumptions hold (never true for text, but true for some domains).
- Very fast.
- Low storage requirements.

Naive Bayes: Training

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

Naive Bayes: Testing

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

Time complexity of Naive Bayes

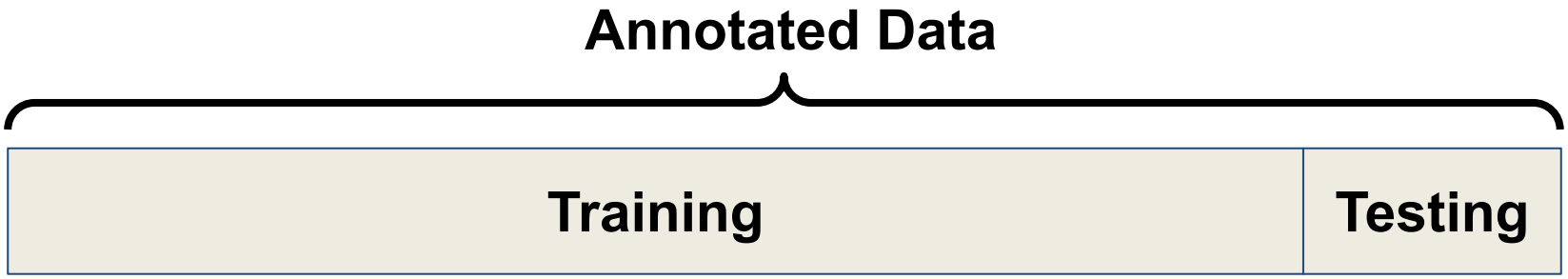
mode	time complexity
training	$\Theta(\|\mathbb{D}\|L_{ave} + \|\mathbb{C}\||V|)$
testing	$\Theta(L_a + \|\mathbb{C}\|M_a) = \Theta(\|\mathbb{C}\|M_a)$

- 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
- $\Theta(|\mathbb{D}|L_{ave})$ is the time it takes to compute all counts.
- $\Theta(|\mathbb{C}||V|)$ is the time it takes to compute the parameters from the counts.
- Generally: $|\mathbb{C}||V| < |\mathbb{D}|L_{ave}$
- 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.

Evaluating classification

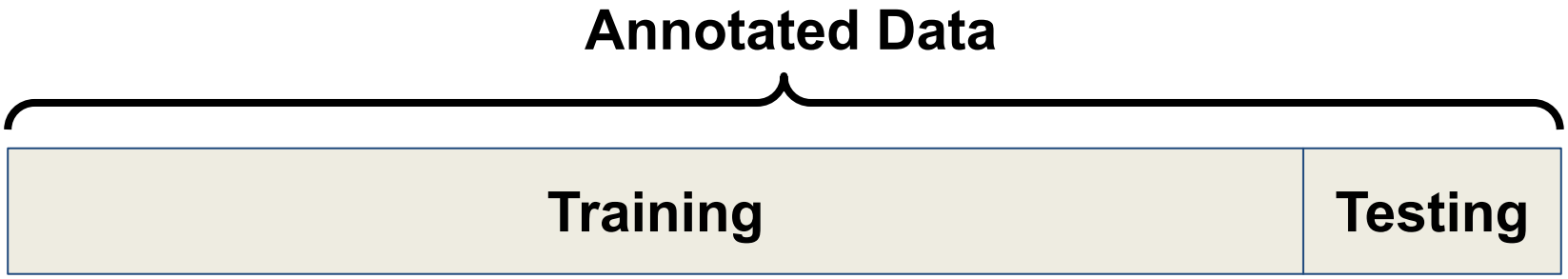
Evaluating Classifiers

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.



Evaluating Classifiers

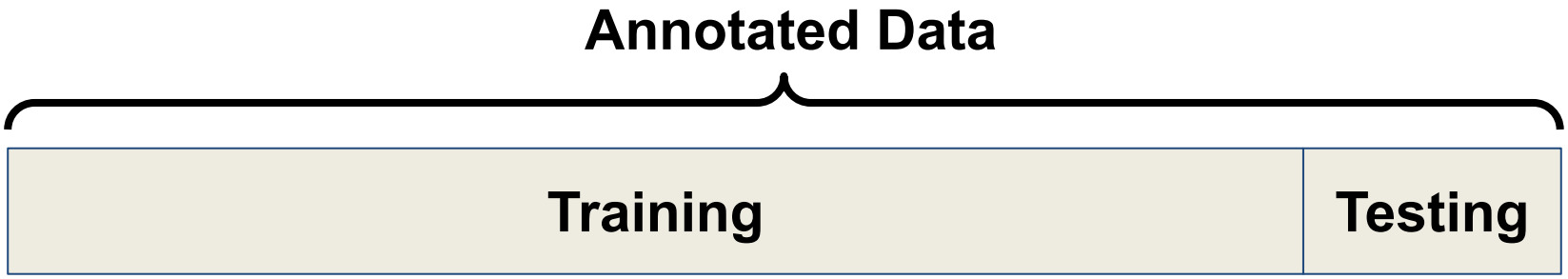
- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.



- A small (10-20%) of training data is used for the validation and hyper-parameter tuning, and the rest for actual model training.

Evaluating Classifiers

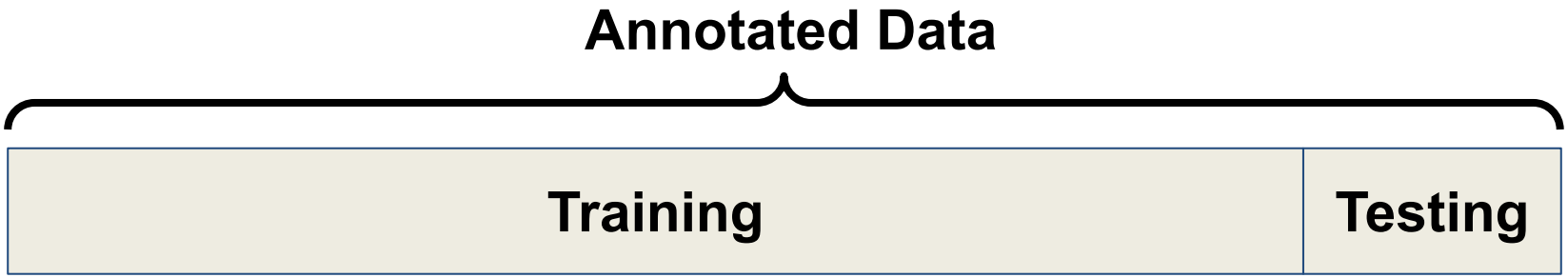
- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.



- A small (10-20%) of training data is used for the validation and hyper-parameter tuning, and the rest for actual model training.
- It's 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 Classifiers

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.



- A small (10-20%) of training data is used for the validation and hyper-parameter tuning, and the rest for actual model training.
- It's easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: Precision, recall, F1, classification accuracy

Precision P and recall R

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

TP, FP, FN, TN are counts of documents. The sum of these four counts is the total number of documents.

$$\text{precision:} P = TP/(TP + FP)$$
$$\text{recall:} R = TP/(TP + FN)$$

Averaging: Micro vs. Macro

- We now have an evaluation measure (F1) for one class.
- But we also want a single number that measures the aggregate performance over all classes in the collection.
- Macro averaging
  - Compute F1 for each of the C classes
  - Average these C numbers
- Micro averaging
  - Compute TP, FP, FN for each of the C classes
  - Sum these C numbers (e.g., all TP to get aggregate TP)
  - Compute F1 for aggregate TP, FP, FN

Macro-average and Micro-average: Example

- **Class 1:**
  - True positive (TP1) = 12, False positive (FP1) = 9, False negative (FN1) = 3
  - Then precision (P1) and recall (R1) will be 57.14 and 80.
- **Class 2:**
  - True positive (TP2) = 50, False positive (FP2) = 23, False negative (FN2) = 9
  - Then precision (P2) and recall (R2) will be 68.49 and 84.75
- **Macro Average:**
  - Macro-average precision = (P1+P2)/2 = (57.14+68.49)/2 = 62.82
  - Macro-average recall = (R1+R2)/2 = (80+84.75)/2 = 82.25
- **Micro Average:**
  - Micro-average of precision = (TP1+TP2)/(TP1+TP2+FP1+FP2) = (12+50)/(12+50+9+23) = 65.96
  - Micro-average of recall = (TP1+TP2)/(TP1+TP2+FN1+FN2) = (12+50)/(12+50+3+9) = 83.78

Source: http://rushdishams.blogspot.com/2011/08/micro-and-macro-average-of-precision.html

Macro-average vs Micro-average: Example 2

- Class A: 1 TP and 1 FP
- Class B: 10 TP and 90 FP
- Class C: 1 TP and 1 FP
- Class D: 1 TP and 1 FP

You can see easily that:

$P_A = P_C = P_D = 0.5, P_B = 0.1$

Macro-average precision = (0.5 + 0.1 + 0.5 + 0.5)/4 = 0.4

Micro-average precision = (1 + 10 + 1 + 1) / (2 + 100 + 2 + 2) = 0.123

- Macro-average: Equal importance to each class
- Micro-average: Equal importance to each sample

Source: https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin

Text classification effectiveness numbers on Reuters-21578

(a)

	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).

Feature Selection

Document as a vector: Feature Vector

	Document 1	Document 2	Document 3	Document 4	...	Document N
Word 1	**.25**	**.5**	**.30**	**0**		**4.3**
Word 2	**0**	**.1**	**.12**	**0**		**.1**
Word 3	**.12**	**0**	**2.1**	**0**		**0**
Word 4	**.1**	**.12**	**1.3**	**0**		**0**
Word 5	**1.3**	**4.5**	**.4**	**.1**		**0**
Word 6	**0**	**.6**	**.7**	**.2**		**.1**
:						
Word M	**0**	**1.2**	**.1**	**2.3**		**.3**

Each document is represented by a vector $\in R^{|V|}$.

Document as a vector: Feature Vector: Issues

- The length of feature vector will be very large: size of the vocabulary: in order of millions.
- In this lecture: axis = dimension = word = term = feature
- Many dimensions correspond to rare words.
- Rare words can mislead the classifier.
- Rare misleading features are called noise features.
- Some classifiers can’t deal with 1,000,000 features

Feature Selection

- Selection of a subset of relevant features.
- Advantages:
  - 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

Example of a Noise Feature

- Let's say we're doing text classification for the class China.
- Suppose a rare term, say arachnocentric, has no information about China . . .
- . . . but all instances of arachnocentric happen to occur in China documents in our training set.
- Then we may learn a classifier that incorrectly interprets arachnocentric as evidence for the class China.
- Such an incorrect generalization from an accidental property of the training set is called overfitting.
- Feature selection reduces overfitting and improves the accuracy of the classifier.

Basic feature selection algorithm

innovate achieve lead

SELECTFEATURES($\mathbb{D}, c, k$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $L \leftarrow []$
3 **for each** $t \in V$
4 **do** $A(t, c) \leftarrow$ COMPUTEFEATUREUTILITY($\mathbb{D}, t, c$)
5 APPEND($L, \langle A(t, c), t \rangle$)
6 **return** FEATURESWITHLARGESTVALUES(L, k)

How do we compute A, the feature utility?

Different feature selection methods

- A feature selection method is mainly defined by the feature utility measure it employs
- Feature utility measures:
  - Frequency – select the most frequent terms
  - Mutual information – select the terms with the highest mutual information
  - Mutual information is also called information gain in this context.
  - Chi-square

Using Mutual Information for Feature Selection

innovate achieve lead

- **Mutual information** (**MI**) of two random variables is a measure of the mutual dependence between the two variables.

- It quantifies the "amount of information" (in units such as shannons, commonly called bits) obtained about one random variable through observing the other random variable.

Mutual Information

- Compute the feature utility A(t, c) as the 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:

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

Linear Vs Non-Linear Classifier

Linear Classifier

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - . . . where θ (the threshold) is a parameter.

Linear Classifier

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ... where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)

Linear Classifier

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ...where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.

Linear Classifier

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ...where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.

Linear Classifier

- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - ... where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides

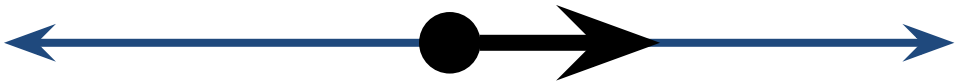
A linear classifier in 1D

- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$

A linear classifier in 1D

- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1

A linear classifier in 1D

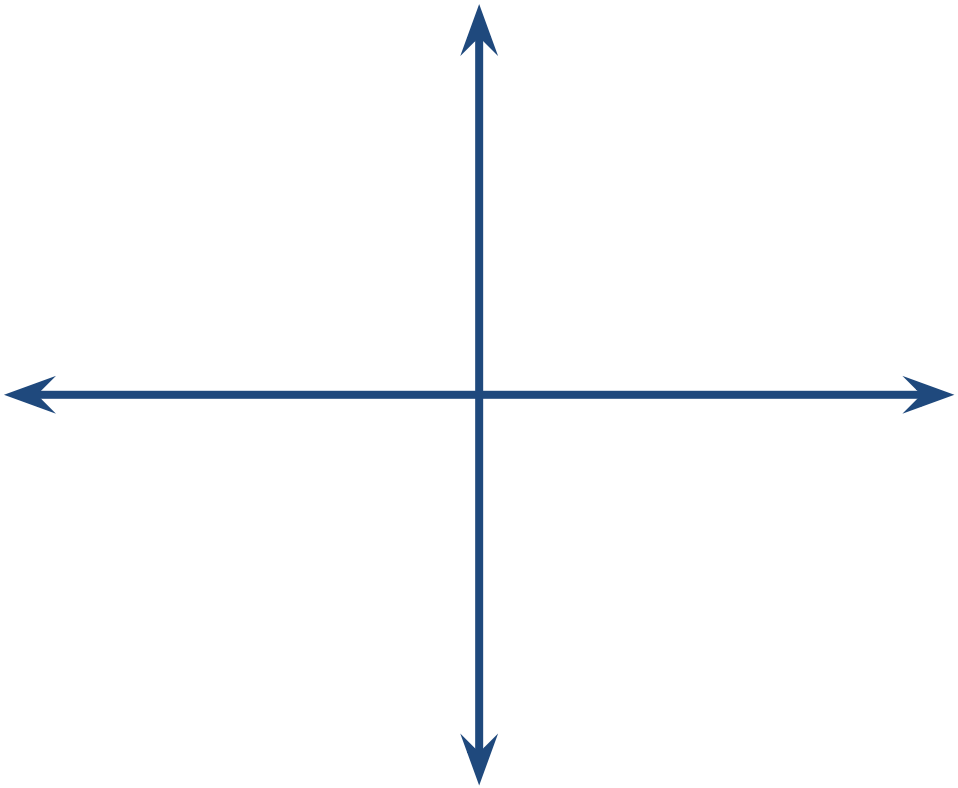


- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.

A linear classifier in 1D

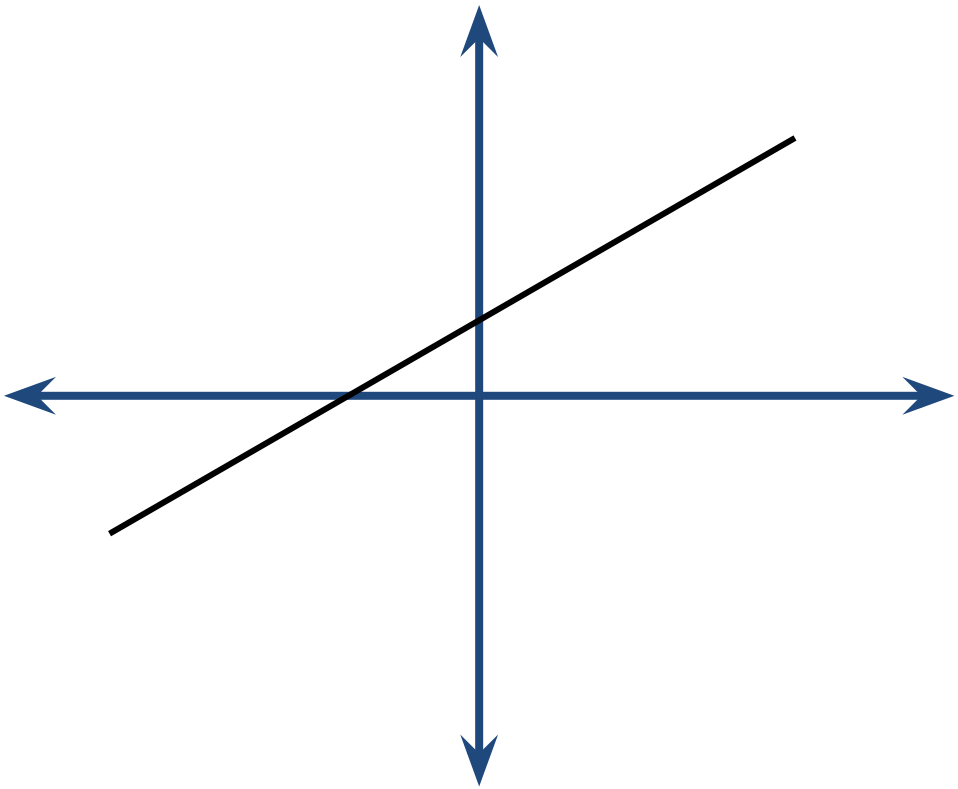
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class not c.

A linear classifier in 2D



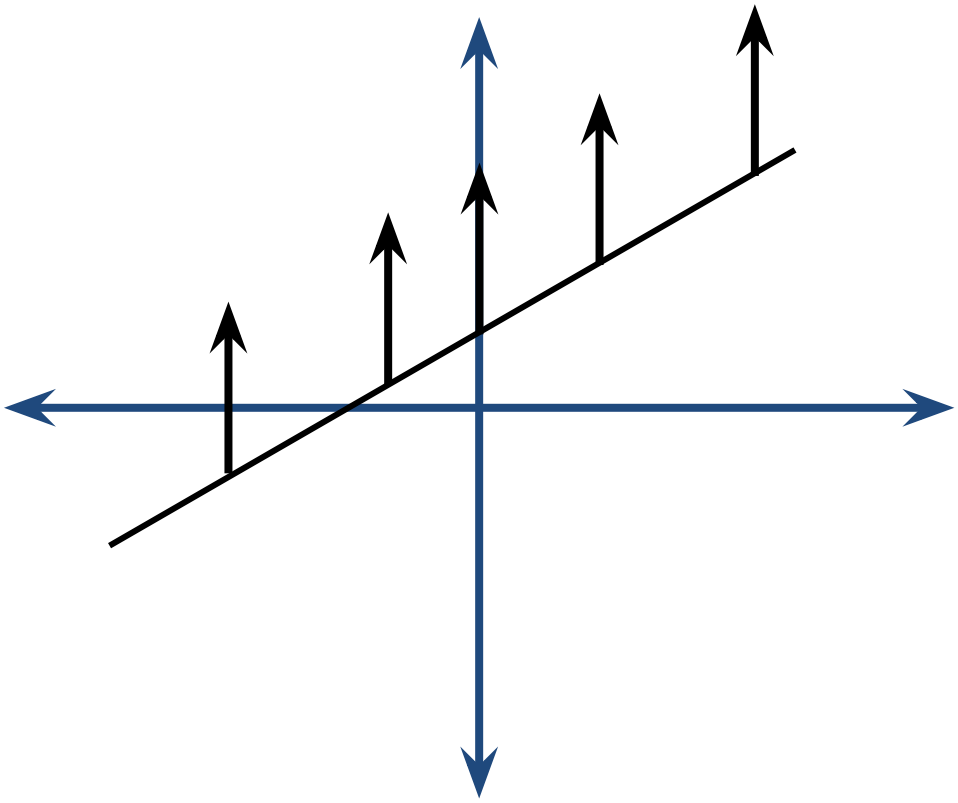
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$

A linear classifier in 2D



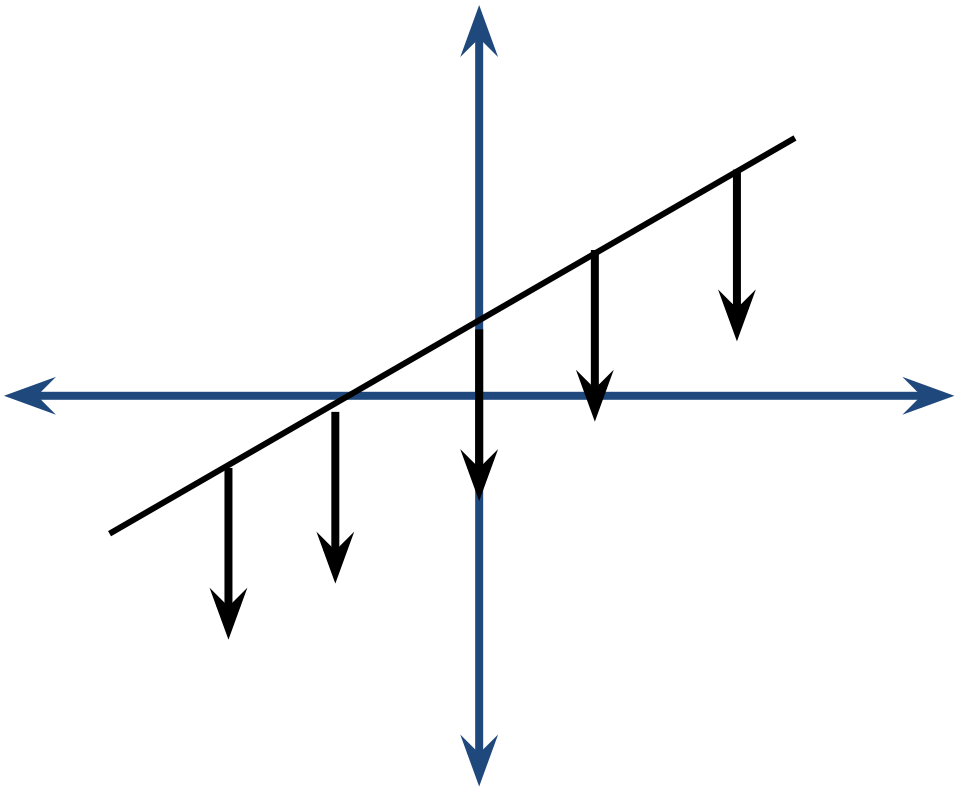
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1d_1 + w_2d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1\ d_2)$ with $w_1d_1 + w_2d_2 \geq \theta$ are in the class c.

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1d_1 + w_2d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1\ d_2)$ with $w_1d_1 + w_2d_2 \geq \theta$ are in the class c.
- Points $(d_1\ d_2)$ with $w_1d_1 + w_2d_2 < \theta$ are in the complement class not c.

Rocchio is a Linear Classifier

Rocchio is a linear classifier defined by:

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

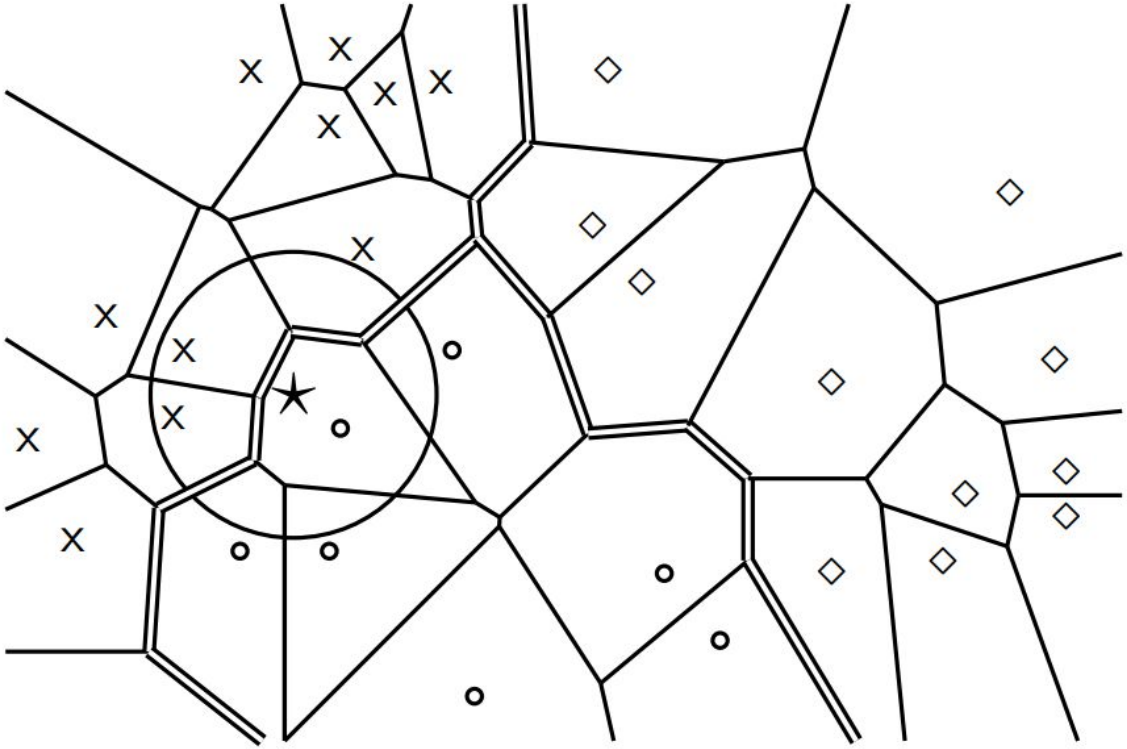
where $\vec{w}$ is the normal vector $\vec{\mu}(c_1) - \vec{\mu}(c_2)$ and $\theta = 0.5 * (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$.

Multinomial Naive Bayes is a linear classifier (in log space)

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

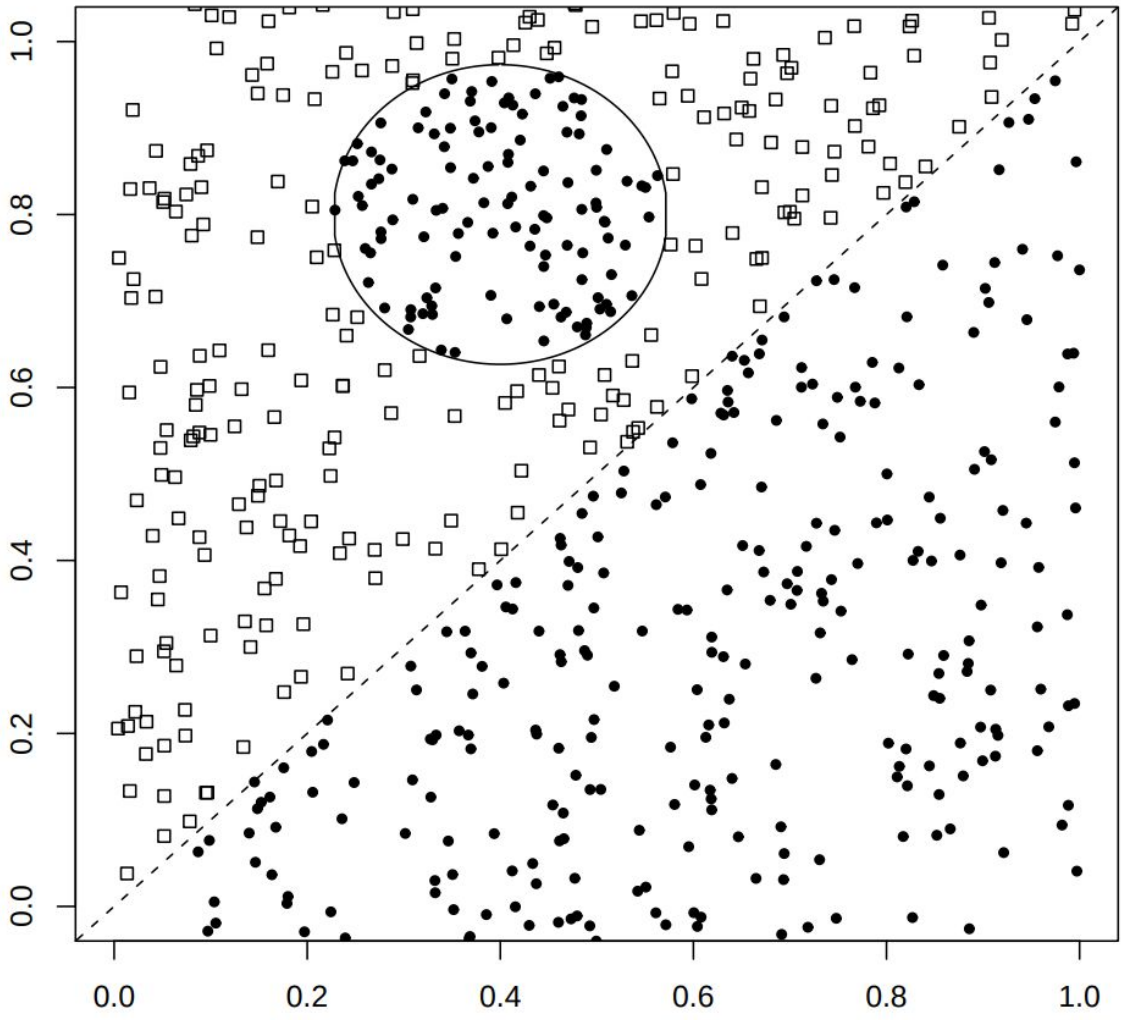
where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, d_i = number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in d as k did in our original definition of Naive Bayes)

kNN is not a Linear Classifier



- Classification decision based on majority of k nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .
- . . . but they are in general not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta$.

A nonlinear problem

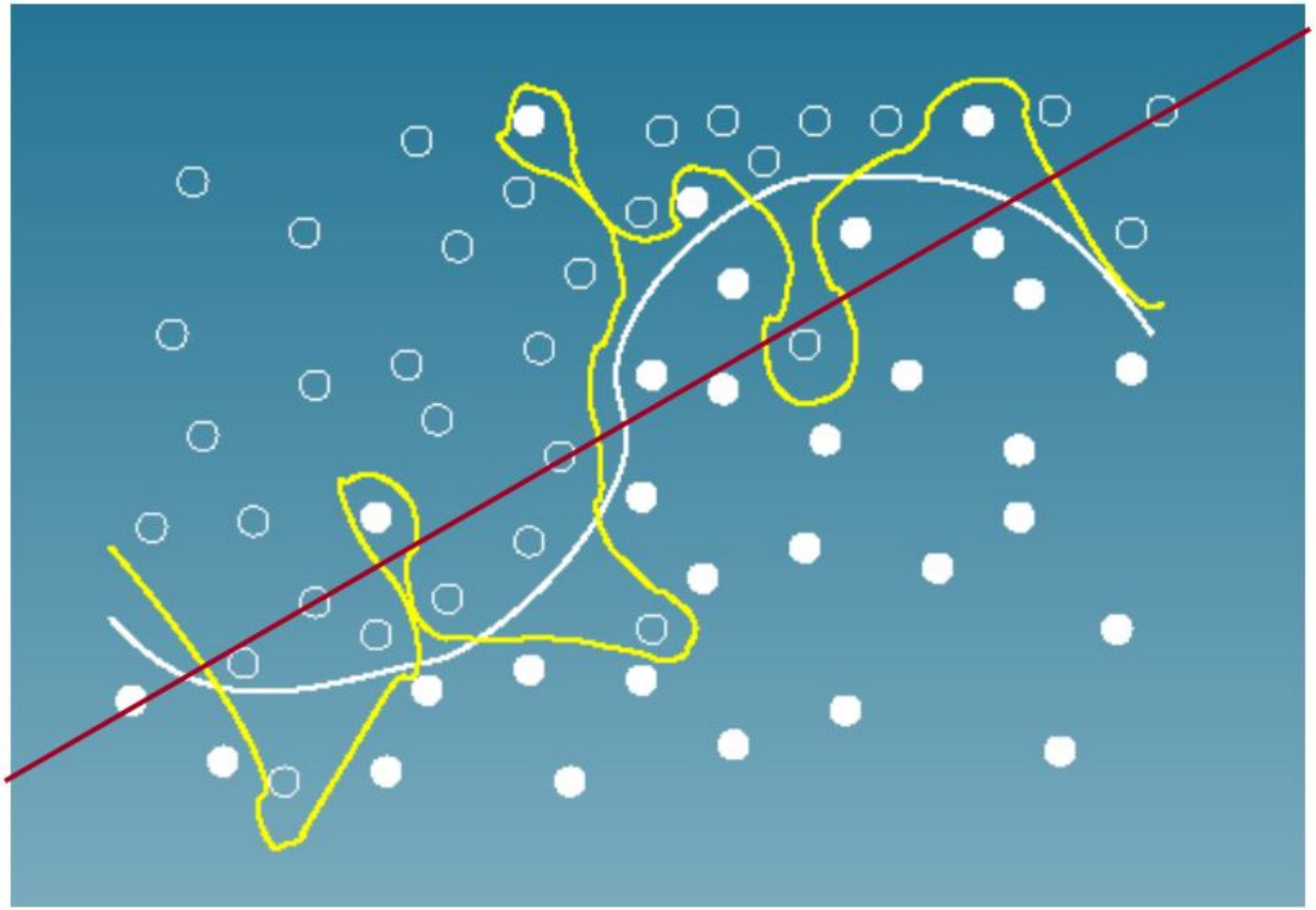


Bias-variance Tradeoff

- The *bias error* is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

- The *variance* is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Bias-variance Tradeoff



Learning algorithms for vector space classification

- In terms of actual computation, there are two types of learning algorithms.
- (i) Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass.
  - Naive Bayes, Rocchio, kNN are all examples of this.
- (ii) Iterative algorithms
  - Support vector machines
  - Neural Networks
- The best performing learning algorithms usually require iterative learning.

Which classifier do I use for a given TC 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 problem?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.

Take-away

- Text-classification Problem
- Naive Bayes, Rocchio and kNN Algorithms
- Evaluation Measures for Classification
- Feature Selection
- Linear vs Non-linear classifier
- Bias-variance tradeoff

Thank You!