Introduction to Information Retrieval

http://informationretrieval.org

IIR 14: Vector Classification

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2008.06.16

Overview

Outline

Naive Bayes classification rule

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

- 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

- Prior:

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

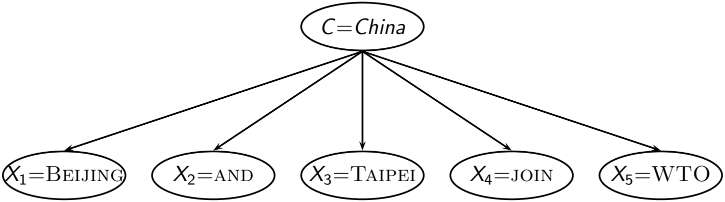
  where N_c is the number of docs in class c and N the total number of docs

- Conditional probabilities:

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

  where T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)

Add-one smoothing to avoid zeros



- In this example:

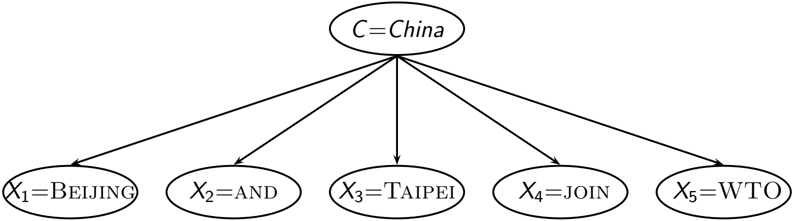
  $P(China|d) \propto P(China)P(\text{Beijing}|China)P(\text{and}|China)P(\text{Taipei}|China)P(\text{join}|China)P(\text{WT}$

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

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

- With this estimate: $[d$ contains $\text{WTO}] \rightarrow [P(China|d) = 0]$.
- We must smooth to get a better estimate $P(China|d) > 0$.

Naive Bayes Independence Assumption



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

- 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 $P(t_k|c)$

Outline

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- In this lecture: axis = dimension = word = term = feature

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- In this lecture: axis = dimension = word = term = feature
- Many dimensions correspond to rare words.

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- In this lecture: axis = dimension = word = term = feature
- Many dimensions correspond to rare words.
- Rare words can mislead the classifier.

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- 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.

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- 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.
- Eliminating noise features from the representation increases efficiency and effectiveness of text classification.

Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- 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.
- Eliminating noise features from the representation increases efficiency and effectiveness of text classification.
- Eliminating features is called feature selection.

Example for a noise feature

- Let's say we're doing text classification for the class *China*.

Example for 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* . . .

Example for 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.

Example for 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 the learning method can produce a classifier that misassigns test documents containing ARACHNOCENTRIC to *China*.

Example for 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 the learning method can produce a classifier that misassigns test documents containing ARACHNOCENTRIC to *China*.
- Such an incorrect generalization from an accidental property of the training set is called overfitting.

Example for 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 the learning method can produce a classifier that misassigns test documents containing ARACHNOCENTRIC to *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

```
SelectFeatures(𝔻, c, k)
V ← ExtractVocabulary(𝔻)
L ← []
for each t ∈ V
do A(t, c) ← ComputeFeatureUtility(𝔻, t, c)
    Append(L, ⟨A(t, c), t⟩)
return FeaturesWithLargestValues(L, k)
```

Basic feature selection algorithm

$\textsc{SelectFeatures}(\mathbb{D}, c, k)$

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

$L \leftarrow [\,]$

for each $t \in V$

do $A(t, c) \leftarrow \textsc{ComputeFeatureUtility}(\mathbb{D}, t, c)$

    $\textsc{Append}(L, \langle A(t, c), t\rangle)$

return $\textsc{FeaturesWithLargestValues}(L, k)$

How do we compute A, the feature utility?

Different feature selection methods

- Each definition of feature utility defines a different feature selection method.

Different feature selection methods

- Each definition of feature utility defines a different feature selection method.
- Frequency – select the most frequent terms

Different feature selection methods

- Each definition of feature utility defines a different feature selection method.
- Frequency – select the most frequent terms
- Mutual information – select the terms with the highest mutual information

Different feature selection methods

- Each definition of feature utility defines a different feature selection method.
- 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.

Mutual information

- Compute the feature utility $A(t, c)$ as the expected mutual information (MI) of term t and class c.

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.

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.

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:

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

How to compute MI values

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

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

- 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

UK

LONDON	0.1925
UK	0.0755
BRITISH	0.0596
STG	0.0555
BRITAIN	0.0469
PLC	0.0357
ENGLAND	0.0238
PENCE	0.0212
POUNDS	0.0149
ENGLISH	0.0126

China

CHINA	0.0997
CHINESE	0.0523
BEIJING	0.0444
YUAN	0.0344
SHANGHAI	0.0292
HONG	0.0198
KONG	0.0195
XINHUA	0.0155
PROVINCE	0.0117
TAIWAN	0.0108

poultry

POULTRY	0.0013
MEAT	0.0008
CHICKEN	0.0006
AGRICULTURE	0.0005
AVIAN	0.0004
BROILER	0.0003
VETERINARY	0.0003
BIRDS	0.0003
INSPECTION	0.0003
PATHOGENIC	0.0003

coffee

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

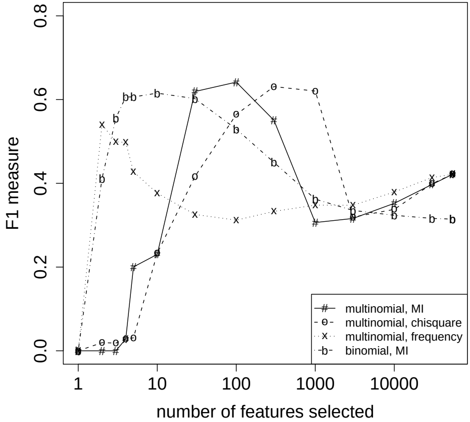
elections

ELECTION	0.0519
ELECTIONS	0.0342
POLLS	0.0339
VOTERS	0.0315
PARTY	0.0303
VOTE	0.0299
POLL	0.0225
CANDIDATE	0.0202
CAMPAIGN	0.0202
DEMOCRATIC	0.0198

sports

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

Evaluation of feature selection



Feature selection for Naive Bayes

- In general, feature selection is necessary for Naive Bayes to get decent performance.

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

Recall vector space representation

- Each document is a vector, one component for each term.

Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.

Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions

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

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.

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.

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.

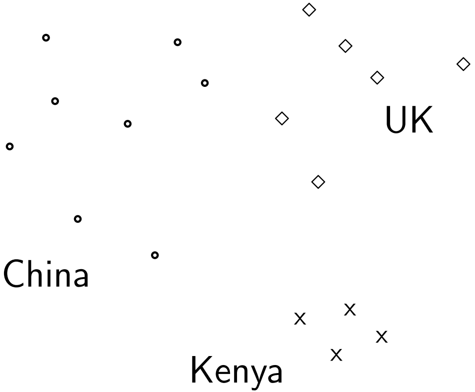
Vector space classification

- 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.

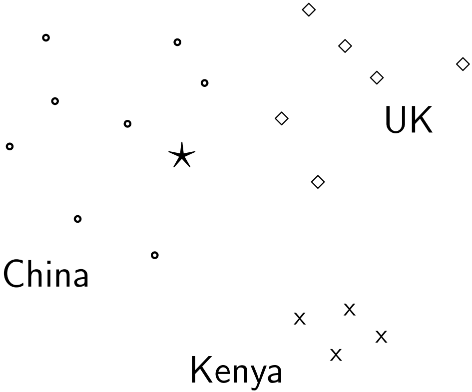
Vector space classification

- 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

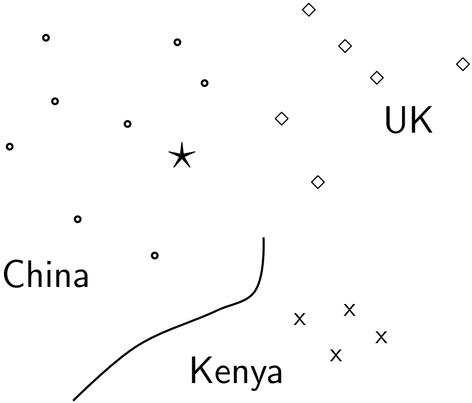


Classes in the vector space



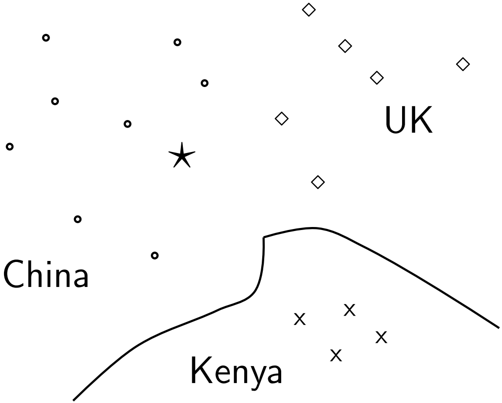
Should the document $\star$ be assigned to *China*, *UK* or *Kenya*?

Classes in the vector space



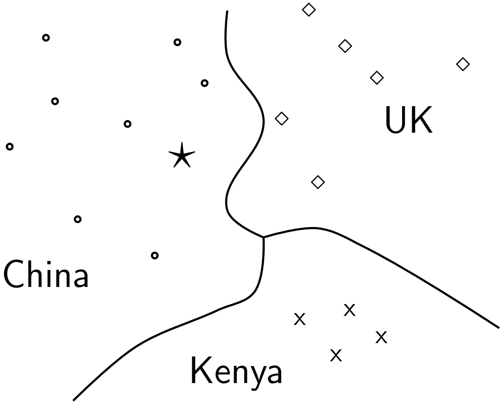
Find separators between the classes

Classes in the vector space



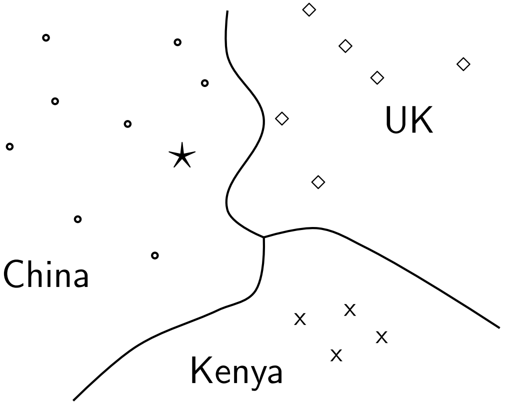
Find separators between the classes

Classes in the vector space



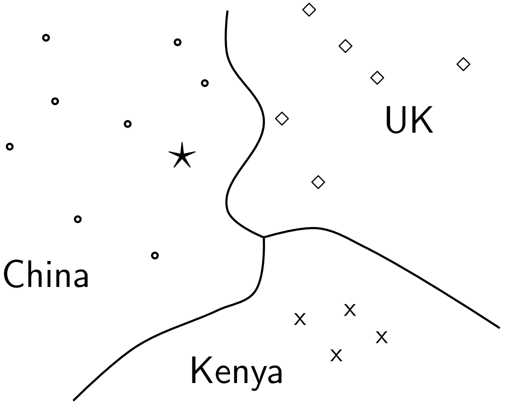
Find separators between the classes

Classes in the vector space



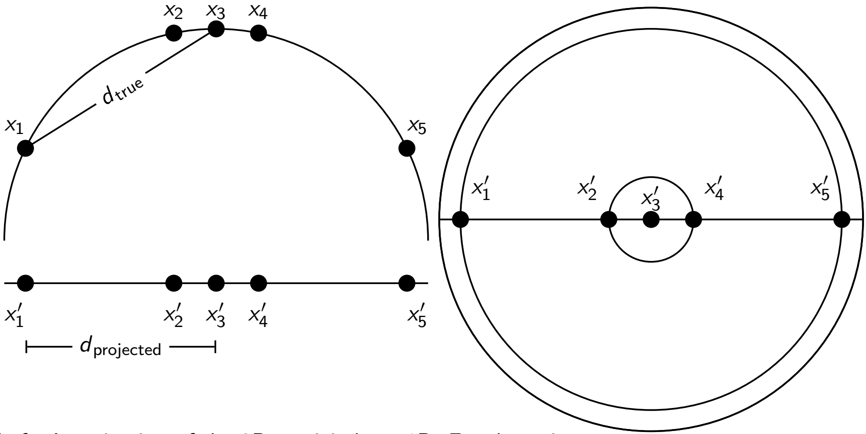
Based on these separators: $\star$ should be assigned to *China*

Classes in the vector space



How do we find separators that do a good job at classifying new documents like $\star$? – Main topic of today

Aside: 2D/3D graphs can be misleading



Left: A projection of the 2D semicircle to 1D. For the points x_1, x_2, x_3, x_4, x_5 at x coordinates $-0.9, -0.2, 0, 0.2, 0.9$ the distance $|x_2x_3| \approx 0.201$ only differs by 0.5% from $|x'_2x'_3| = 0.2$; but $|x_1x_3|/|x'_1x'_3| = d_{\text{true}}/d_{\text{projected}} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area. *Right:* The corresponding projection of the 3D hemisphere to 2D.

Outline

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.
- The two classes: the relevant documents and the nonrelevant documents.

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.
- The two classes: the relevant documents and the nonrelevant documents.
- The training set is the set of documents the user has labeled so far.

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.
- The two classes: the relevant documents and the nonrelevant documents.
- The training set is the set of documents the user has labeled so far.
- The principal difference between relevance feedback and text classification:

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.
- The two classes: the relevant documents and the nonrelevant documents.
- The training set is the set of documents the user has labeled so far.
- The principal difference between relevance feedback and text classification:
  - The training set is given as part of the input in text classification.

Using Rocchio for vector space classification

- We can view relevance feedback as two-class classification.
- The two classes: the relevant documents and the nonrelevant documents.
- The training set is the set of documents the user has labeled so far.
- The principal difference between relevance feedback and text classification:
    - The training set is given as part of the input in text classification.
    - It is interactively created in relevance feedback.

Rocchio classification: Basic idea

- Compute a centroid for each class

Rocchio classification: Basic idea

- Compute a centroid for each class
  - The centroid is the average of all documents in the class.

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.

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.

What can we say about the length of this centroid given that each $\vec{v}(d)$ is normalized?

Rocchio algorithm

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

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

do $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$

$\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$

return $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

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

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

Rocchio algorithm

$\textsc{TrainRocchio}(\mathbb{C}, \mathbb{D})$

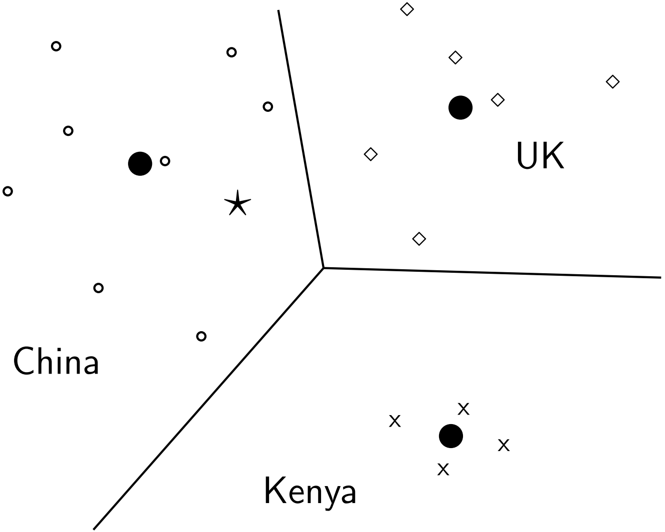
1 **for each** $c_j \in \mathbb{C}$
2 **do** $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$
3 $\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
4 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

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

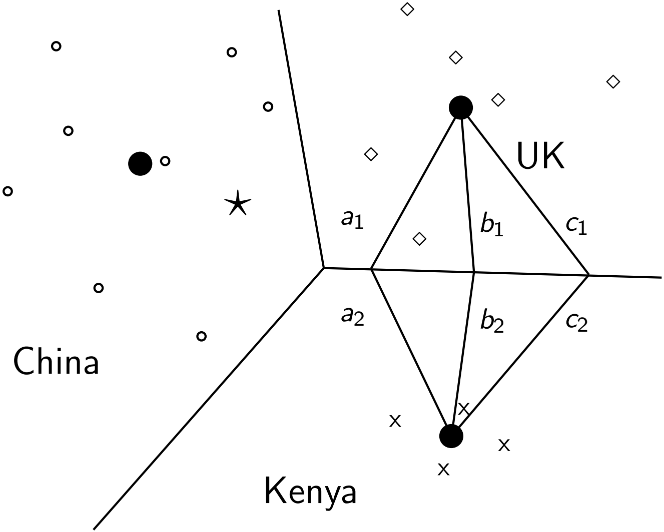
1 **return** $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$

Questions?

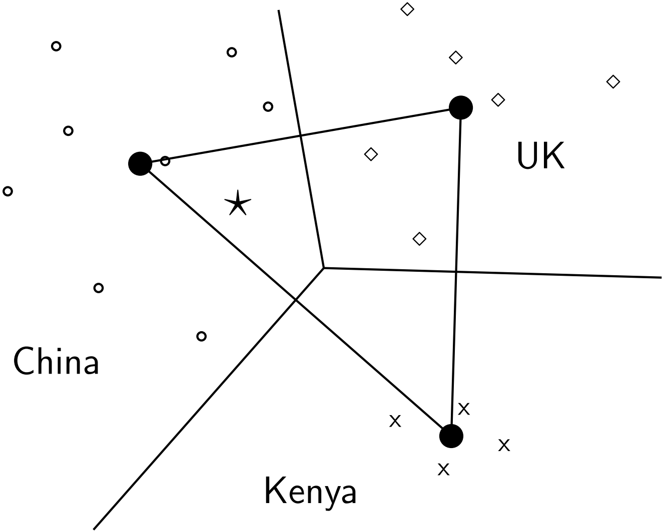
Rocchio illustrated



Rocchio illustrated: $a_1 = a_2, b_1 = b_2, c_1 = c_2$



Rocchio illustrated



Rocchio properties

- Rocchio forms a simple representation for each class: the centroid or prototype.

Rocchio properties

- Rocchio forms a simple representation for each class: the centroid or prototype.
- Classification is based on similarity to / distance from centroid/prototype.

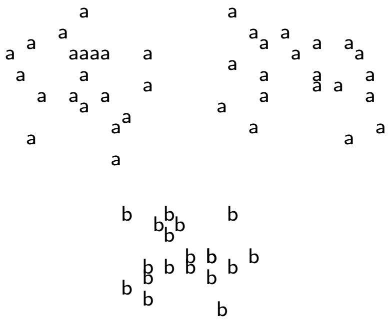
Rocchio properties

- Rocchio forms a simple representation for each class: the centroid or prototype.
- Classification is based on similarity to / distance from centroid/prototype.
- Does not guarantee that classifications are consistent with the given training data.

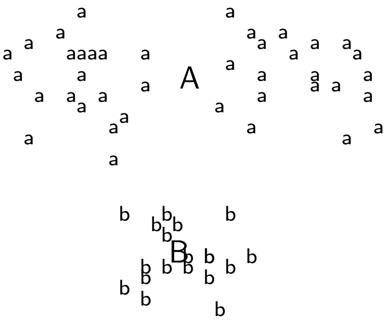
Time complexity of Rocchio

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

Rocchio cannot handle multimodal classes

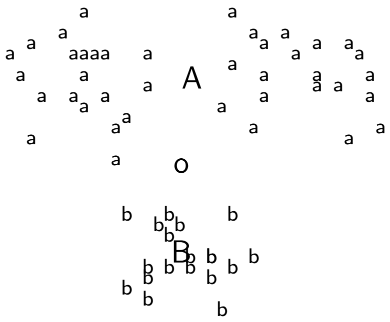


Rocchio cannot handle multimodal classes



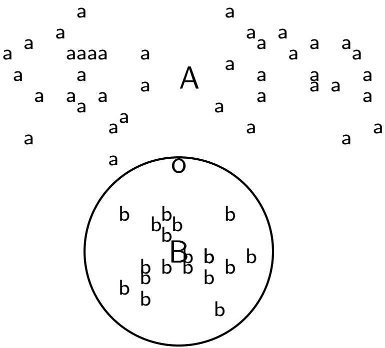
- A is centroid of the a's, B is centroid of the b's.

Rocchio cannot handle multimodal classes



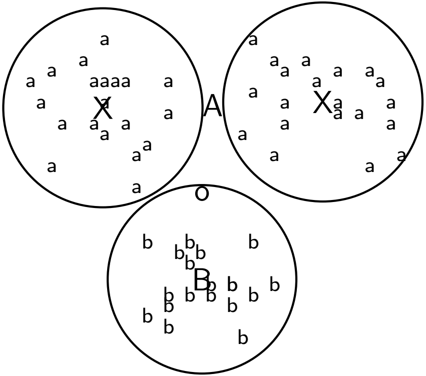
- A is centroid of the a's, B is centroid of the b's.
- The point o is closer to A than to B.

Rocchio cannot handle multimodal classes



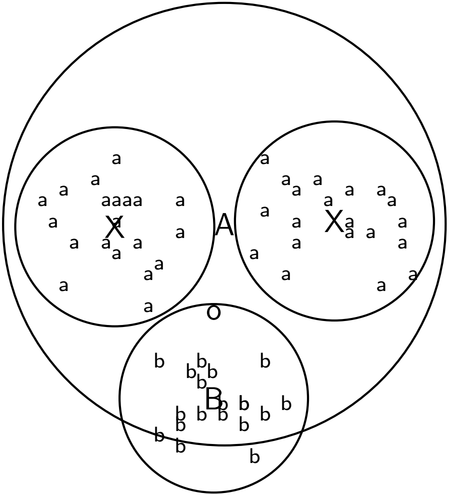
- A is centroid of the a's, B is centroid of the b's.
- The point o is closer to A than to B.
- But it is a better fit for the b class.

Rocchio cannot handle 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 it is a better fit for the b class.
- A is a multimodal class with two prototypes.

Rocchio cannot handle 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 it is a better fit for the b class.
- A is a multimodal class with two prototypes.
- But in Rocchio we only have one.

Outline

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- (First, we only consider binary classifiers.)

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities)

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities)
- Assumption: The classes are linearly separable.

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities)
- Assumption: The classes are linearly separable.
- Can find hyperplane (=separator) based on training set

Linear classifiers

- Linear classifiers compute a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
- Classification decision: $\sum_i w_i x_i > \theta$?
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities)
- Assumption: The classes are linearly separable.
- Can find hyperplane (=separator) based on training set
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides

Example of a linear two-class classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class *interest* in Reuters-21578.

Example of a linear two-class classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class *interest* in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation

Example of a linear two-class classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class *interest* in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- We assign document $\vec{d}_1$ "rate discount dlrs world" to *interest* since $\vec{w}^T\vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = b$.

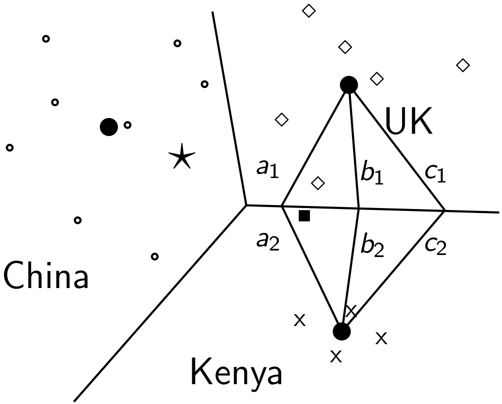
Example of a linear two-class classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class *interest* in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- We assign document $\vec{d}_1$ "rate discount dlrs world" to *interest* since $\vec{w}^T\vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = b$.
- We assign $\vec{d}_2$ "prime dlrs" to the complement class (not in *interest*) since $\vec{w}^T\vec{d}_2 = -0.01 \leq b$.

perceptron example: one way of finding a separator

Rocchio separators are linear classifiers that can be expressed as $\sum_i w_i x_i > \theta$



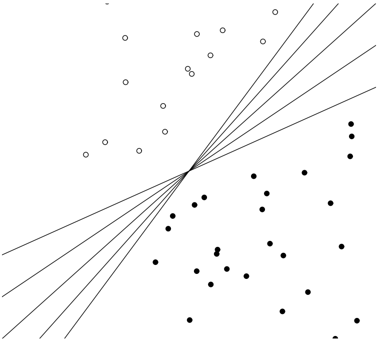
Two-class Rocchio as linear classifier

Line (or plane or hyperplane) defined by:

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

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

Which hyperplane?



Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly . . .

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly ...
- ... but they behave differently on test data.

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly . . .
- . . . but they behave differently on test data.
- Error rates on new data are low for some, high for others.

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly ...
- ...but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly ...
- ... but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Perceptron: generally bad; Naive Bayes, Rocchio: ok; linear SVM: good

Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly . . .
- . . . but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Perceptron: generally bad; Naive Bayes, Rocchio: ok; linear SVM: good
- Many more classification methods

Naive Bayes is also a linear classifier

We can derive the linearity of Naive Bayes from its decision rule, which chooses the category c with the largest $\hat{P}(c|d)$ where:

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

and n_d is the number of tokens in the document that are part of the vocabulary. Denoting the complement category as $\bar{c}$, we obtain for the log odds:

$$\log \frac{\hat{P}(c|d)}{\hat{P}(\bar{c}|d)} = \log \frac{\hat{P}(c)}{\hat{P}(\bar{c})} + \sum_{1 \leq k \leq n_d} \log \frac{\hat{P}(t_k|c)}{\hat{P}(t_k|\bar{c})}$$

We choose class c if the odds are greater than 1 or, equivalently, if the log odds are greater than 0. One can show that this is a linear classifier.

Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.

Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane – huge differences in performance.

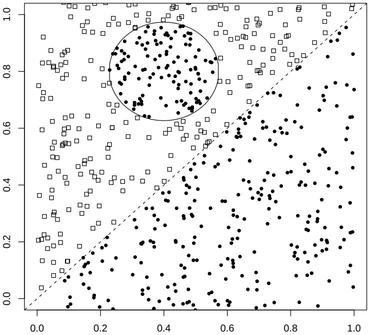
Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane – huge differences in performance.
- Can we get better performance with more powerful nonlinear classifiers?

Linear classifiers: Discussion

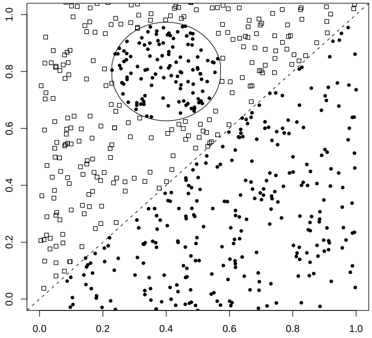
- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane – huge differences in performance.
- Can we get better performance with more powerful nonlinear classifiers?
- Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.

A nonlinear problem



- Linear classifier like Rocchio does badly on this task.

A nonlinear problem



- Linear classifier like Rocchio does badly on this task.
- kNN will do well (assuming enough training data)

Which classifier do I use for a given TC problem?

- Is there a learning method that is optimal for all text classification problems?

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.

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:

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?

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)

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?

Which classifier do I use for a given TC problem?

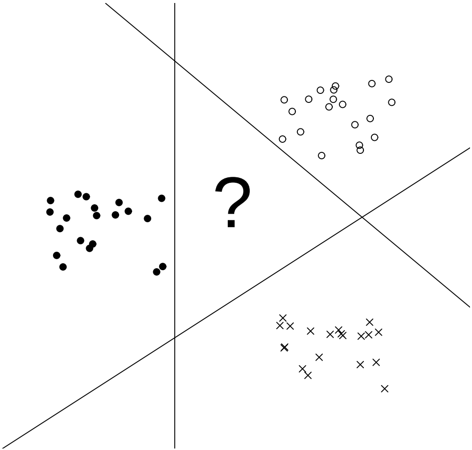
- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  -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

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.

Outline

How to combine hyperplanes for $>$ 2 classes?



Any-of vs. one-of problems

- Any-of or multilabel classification

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of “independence” (but not statistical independence)

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of “independence” (but not statistical independence)
  - Example: topic classification
- One-of or multiclass classification

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification
- One-of or multiclass classification
  - Classes are mutually exclusive.

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification
- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.

Any-of vs. one-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves all decisions open on other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification
- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.
  - Example: language of a document (assumption: no document contains multiple languages)

Any-of classification

- Combine two-class classifiers as follows for any-of classification:

Any-of classification

- Combine two-class classifiers as follows for any-of classification:
  - Simply run each two-class classifier separately on the test document and assign document accordingly

One-of classification

- Combine two-class classifiers as follows for one-of classification:

One-of classification

- Combine two-class classifiers as follows for one-of classification:
  - Run each classifier separately

One-of classification

- Combine two-class classifiers as follows for one-of classification:
  - Run each classifier separately
  - Rank classifiers (e.g., according to score)

One-of classification

- Combine two-class classifiers as follows for one-of classification:
  - Run each classifier separately
  - Rank classifiers (e.g., according to score)
  - Pick the class with the highest score

Outline

kNN classification

- kNN $= k$ nearest neighbors

kNN classification

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we assign each test document to the class of its nearest neighbor in the training set.

kNN classification

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we 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

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we 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.
- For $k > 1$, we assign each test document to the majority class of its k nearest neighbors in the training set.

kNN classification

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we 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.
- For $k > 1$, we assign each test document to the majority class of its k nearest neighbors in the training set.
- This amounts to locally defined decision boundaries between classes – far away points do not influence the classification decision. (different from Rocchio)

kNN classification

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we 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.
- For $k > 1$, we assign each test document to the majority class of its k nearest neighbors in the training set.
- This amounts to locally defined decision boundaries between classes – far away points do not influence the classification decision. (different from Rocchio)
- Rationale of kNN: contiguity hypothesis

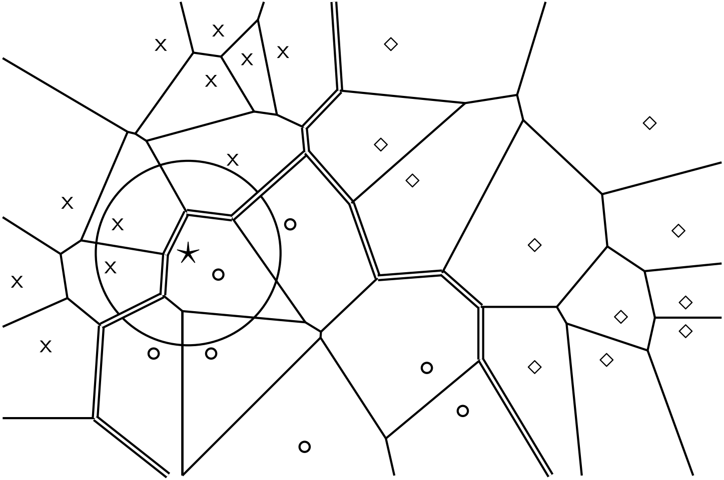
kNN classification

- kNN = k nearest neighbors
- For $k = 1$ (1NN), we 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.
- For $k > 1$, we assign each test document to the majority class of its k nearest neighbors in the training set.
- This amounts to locally defined decision boundaries between classes – far away points do not influence the classification decision. (different from Rocchio)
- 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.

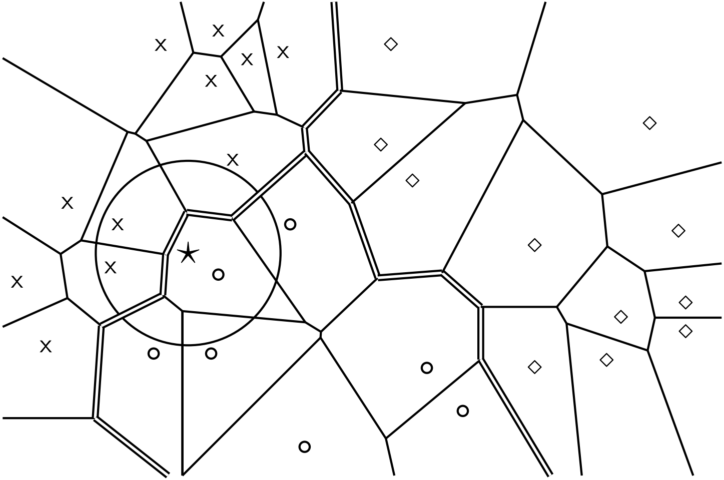
Probabilistic kNN

Probabilistic version of kNN: $P(c|d) =$ fraction of k neighbors of d that are in c

kNN is based on Voronoi tessellation



kNN is based on Voronoi tessellation



1NN, 2NN, 3NN classification decision for star?

kNN algorithm

$\text{TRAIN-KNN}(\mathbb{C}, \mathbb{D})$

$\mathbb{D}' \leftarrow \text{PREPROCESS}(\mathbb{D})$

$k \leftarrow \text{SELECT-K}(\mathbb{C}, \mathbb{D}')$

return $\mathbb{D}', k$

$\text{APPLY-KNN}(\mathbb{C}, \mathbb{D}', k, d)$

$S_k \leftarrow \text{COMPUTENEARESTNEIGHBORS}(\mathbb{D}', k, d)$

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

do $p_j \leftarrow |S_k \cap c_j|/k$

return $\arg\max_j p_j$

kNN algorithm

$\textsc{Train-kNN}(\mathbb{C}, \mathbb{D})$
$\mathbb{D}' \leftarrow \textsc{Preprocess}(\mathbb{D})$
$k \leftarrow \textsc{Select-k}(\mathbb{C}, \mathbb{D}')$
return $\mathbb{D}', k$

$\textsc{Apply-kNN}(\mathbb{C}, \mathbb{D}', k, d)$
$S_k \leftarrow \textsc{ComputeNearestNeighbors}(\mathbb{D}', k, d)$
for each $c_j \in \mathbb{C}$
do $p_j \leftarrow |S_k \cap c_j|/k$
return $\arg\max_j p_j$

Questions?

Time complexity of kNN

kNN with preprocessing of training set

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

testing $\Theta(L_{\mathrm{a}} + |\mathbb{D}|M_{\mathrm{ave}}M_{\mathrm{a}}) = \Theta(|\mathbb{D}|M_{\mathrm{ave}}M_{\mathrm{a}})$

kNN without preprocessing of training set

training $\Theta(1)$

testing $\Theta(L_{\mathrm{a}} + |\mathbb{D}|L_{\mathrm{ave}}M_{\mathrm{a}}) = \Theta(|\mathbb{D}|L_{\mathrm{ave}}M_{\mathrm{a}})$

kNN with inverted index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection.

kNN with inverted index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection.
- Finding 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.

kNN with inverted index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection.
- Finding 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 with inverted index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection.
- Finding 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: $O(|D|)$, that is, still linear in the number of documents. (Length of postings lists approximately linear in number of documents D.)

kNN with inverted index

- Naively finding nearest neighbors requires a linear search through $|D|$ documents in collection.
- Finding 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: $O(|D|)$, that is, still linear in the number of documents. (Length of postings lists approximately linear in number of documents D.)
- But constant factor much smaller for inverted index than for linear scan.

kNN: Discussion

- No training necessary

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
- kNN is very accurate if training set is large.

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.
- kNN test time proportional to the size of the training set.

kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.
- kNN test time proportional to the size of the training set.
  - kNN is inefficient for very large training sets.

Is kNN a linear classifier?

Resources

- Chapter 14 of IIR

Resources

- Chapter 14 of IIR
- Resources at `http://ifnlp.org/ir`

Resources

- Chapter 14 of IIR
- Resources at `http://ifnlp.org/ir`
- Perceptron example

Resources

- Chapter 14 of IIR
- Resources at `http://ifnlp.org/ir`
- Perceptron example
- General overview of text classification: Sebastiani (2002)

Resources

- Chapter 14 of IIR
- Resources at `http://ifnlp.org/ir`
- Perceptron example
- General overview of text classification: Sebastiani (2002)
- Text classification chapter on decision tress and perceptrons: Manning & Schütze (1999)

Resources

- Chapter 14 of IIR
- Resources at `http://ifnlp.org/ir`
- Perceptron example
- General overview of text classification: Sebastiani (2002)
- Text classification chapter on decision tress and perceptrons: Manning & Schütze (1999)
- One of the best machine learning textbooks: Hastie, Tibshirani & Friedman (2003)