Introduction to

# Information Retrieval

Hinrich Schütze and Christina Lioma

Lecture 14: Vector Space Classification

Introduction to

# Information Retrieval

# Outline

1. Recap
2. Feature selection
3. Intro vector space classification
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

# 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*.
- 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

SelectFeatures($\mathbb{D}, c, k$)
$V \leftarrow$ ExtractVocabulary($\mathbb{D}$)
$L \leftarrow []$
**for each** $t \in V$
**do** $A(t, c) \leftarrow$ ComputeFeatureUtility($\mathbb{D}, t, c$)
    Append($L, \langle A(t, c), t\rangle$)
**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.
  - $X^2$ feature selection

# Frequency based feature selection

- Select terms that are most common in class.
- Frequency can be of two types:
  - Document frequency – number of documents of class c that contain t
  - Collection frequency – The number of times t occurs in the documents of 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.
- 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:

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

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

# MI example for *poultry*/EXPORT in Reuters

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

Plug these values into formula:

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

# MI feature selection on Reuters

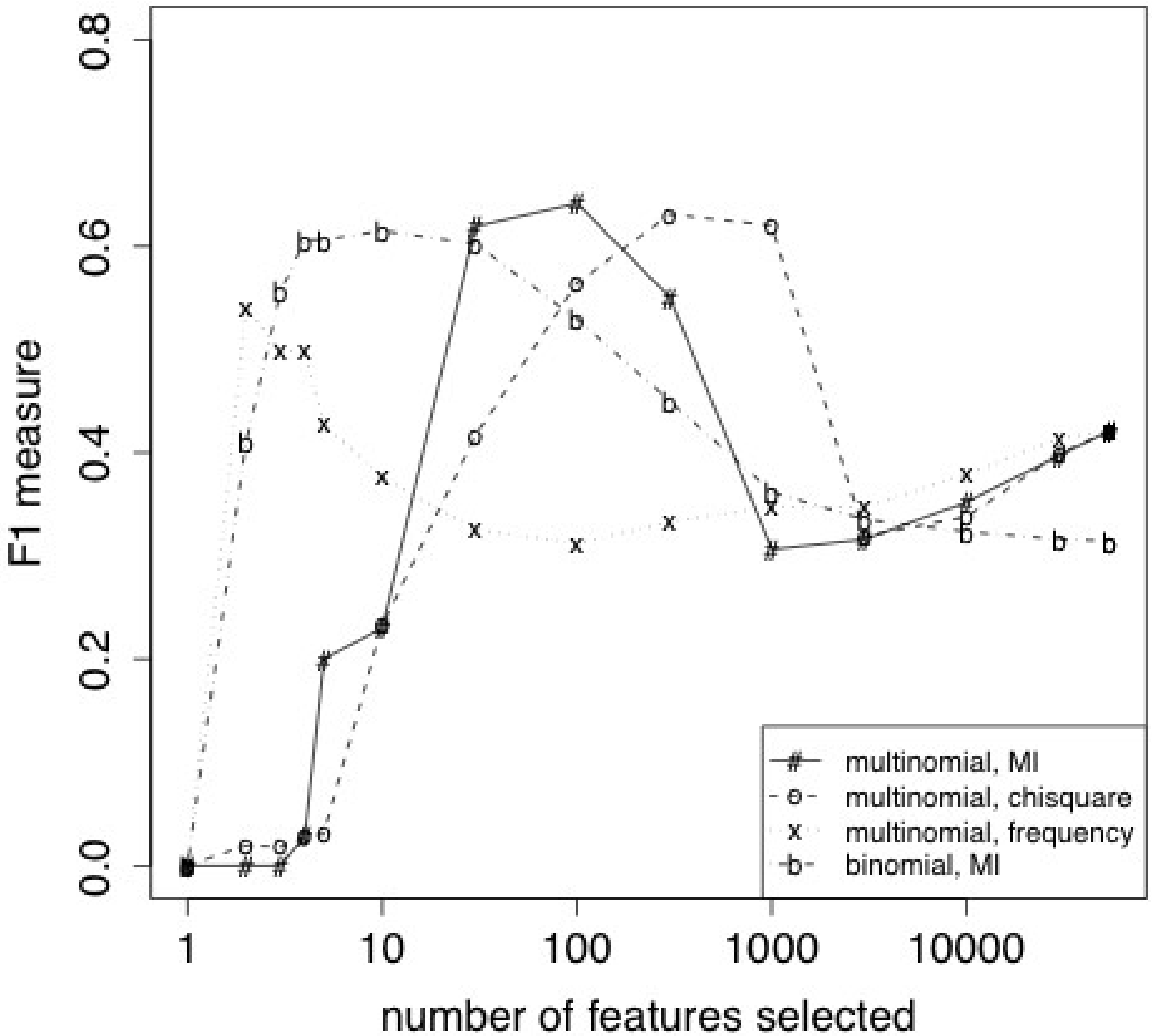
Class: *coffee*

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

Class: *sports*

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

# Naive Bayes: Effect of feature selection




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

# Feature selection for Naive Bayes

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

# $\chi^2$ Feature selection

- $X^2$ test is applied to test independence of two events.
  - P(AB)=P(A)P(B) or P(A|B)=P(A) or P(B|A)=P(B)
- Two events are
  - Occurrence of term
  - Occurrence of class
- Rank terms with respect to the following quantity

$$X^2(\mathbb{D},t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

- N: Observed frequency in D, E: Expected frequency

# $\chi^2$ Feature selection

$$\begin{aligned} E_{11} &= N \times P(t) \times P(c) = N \times \frac{N_{11}+N_{10}}{N} \times \frac{N_{11}+N_{01}}{N} \\ &= N \times \frac{49+141}{N} \times \frac{49+27652}{N} \approx 6.6 \end{aligned}$$

where $N$ is the total number of documents as before.
We compute the other $E_{e_t e_c}$ in the same way:

| | $e_{poultry} = 1$ | | $e_{poultry} = 0$ | |
|---|---|---|---|---|
| $e_{\text{export}} = 1$ | $N_{11} = 49$ | $E_{11} \approx 6.6$ | $N_{10} = 27{,}652$ | $E_{10} \approx 27{,}694.4$ |
| $e_{\text{export}} = 0$ | $N_{01} = 141$ | $E_{01} \approx 183.4$ | $N_{00} = 774{,}106$ | $E_{00} \approx 774{,}063.6$ |

Plugging these values into Equation (13.18), we get a $X^2$ value of 284:

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \approx 284$$

# $\chi^2$ Feature selection

- $X^2$ test is the measure of how much E and N deviates from each other.
- $X^2 = 284$ ($X^2 > 10.83$) => we can reject the the hypothesis that *poultry* and *export* are independent with only a 0.001 chance of being wrong.
- ▶ **Table 13.6** Critical values of the $\chi^2$ distribution with one degree of freedom. For example, if the two events are independent, then $P(X^2 > 6.63) < 0.01$. So for $X^2 > 6.63$ the assumption of independence can be rejected with 99% confidence.

| $p$ | $\chi^2$ critical value |
|---|---|
| 0.1 | 2.71 |
| 0.05 | 3.84 |
| 0.01 | 6.63 |
| 0.005 | 7.88 |
| 0.001 | 10.83 |

# Outline

1. Recap
2. Feature selection
3. Intro vector space classification
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

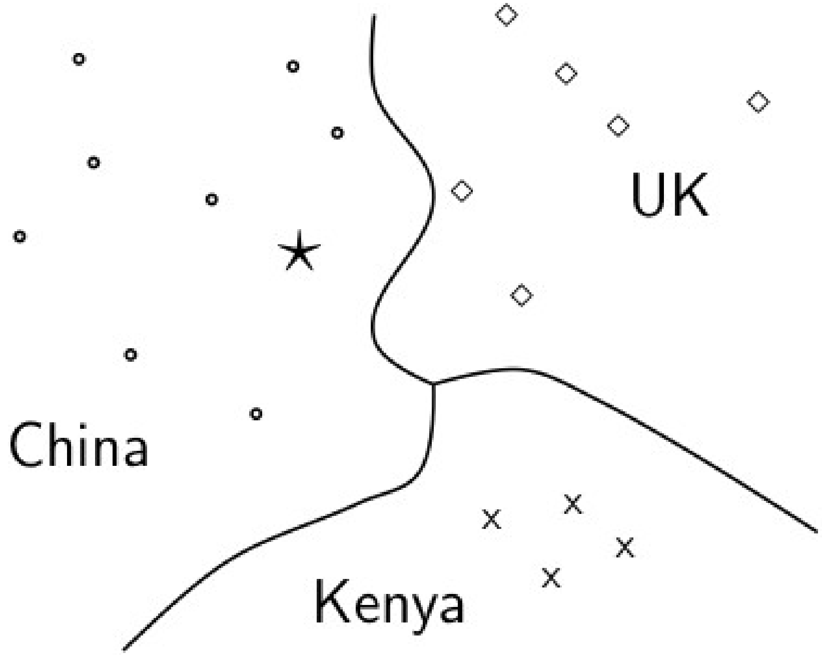
# Recall vector space representation

- 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

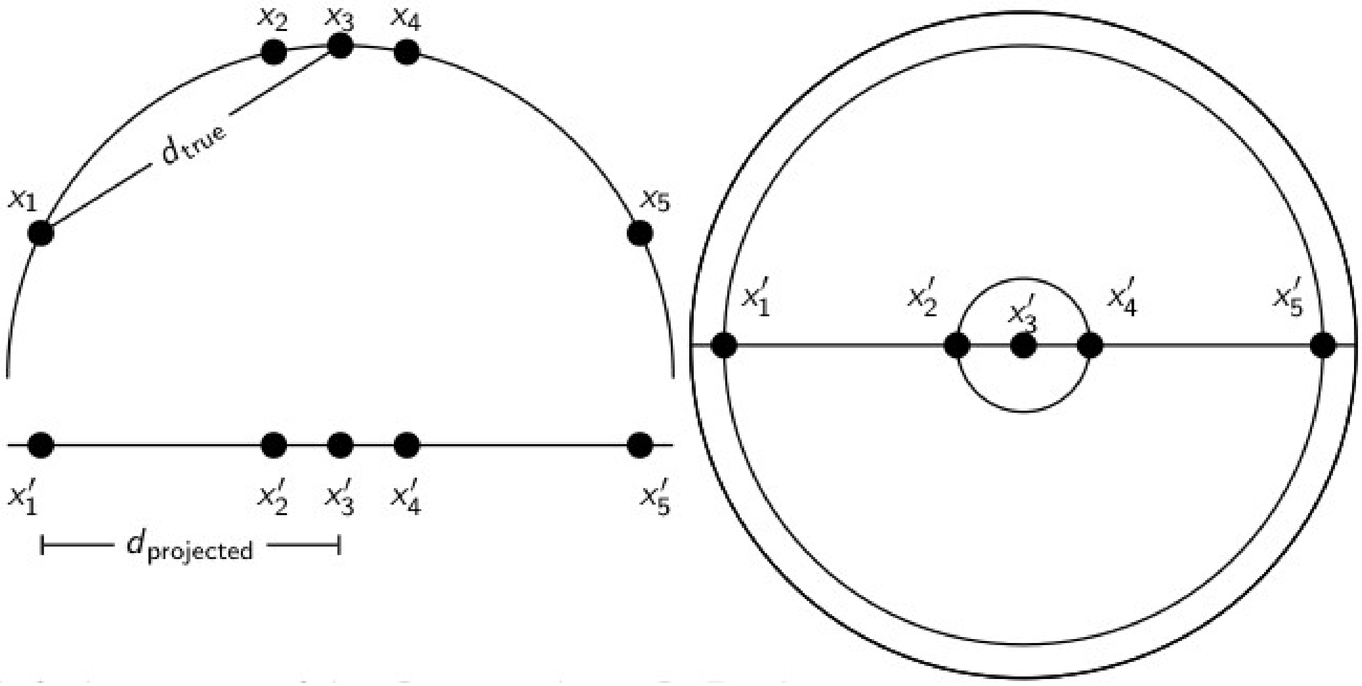
- As before, the training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a contiguous region.
- Premise 2: Documents from different 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 ⋆? – 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 $|x_1x_3|/|x'_1x'_3| = d_{true}/d_{projected} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area. *Right*: The corresponding projection of the 3D hemisphere to 2D.

# Outline

1. Recap
2. Feature selection
3. Intro vector space classification
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

# Relevance feedback

- In relevance feedback, the user marks documents as relevant/nonrelevant.
- Relevant/nonrelevant can be viewed as classes or categories.
- For each document, the user decides which of these two classes is correct.
- The IR system then uses these class assignments to build a better query (“model”) of the information need . . .
- . . . and returns better documents.
- Relevance feedback is a form of text classification.

# Using Rocchio for vector space classification

- 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
  - 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 algorithm

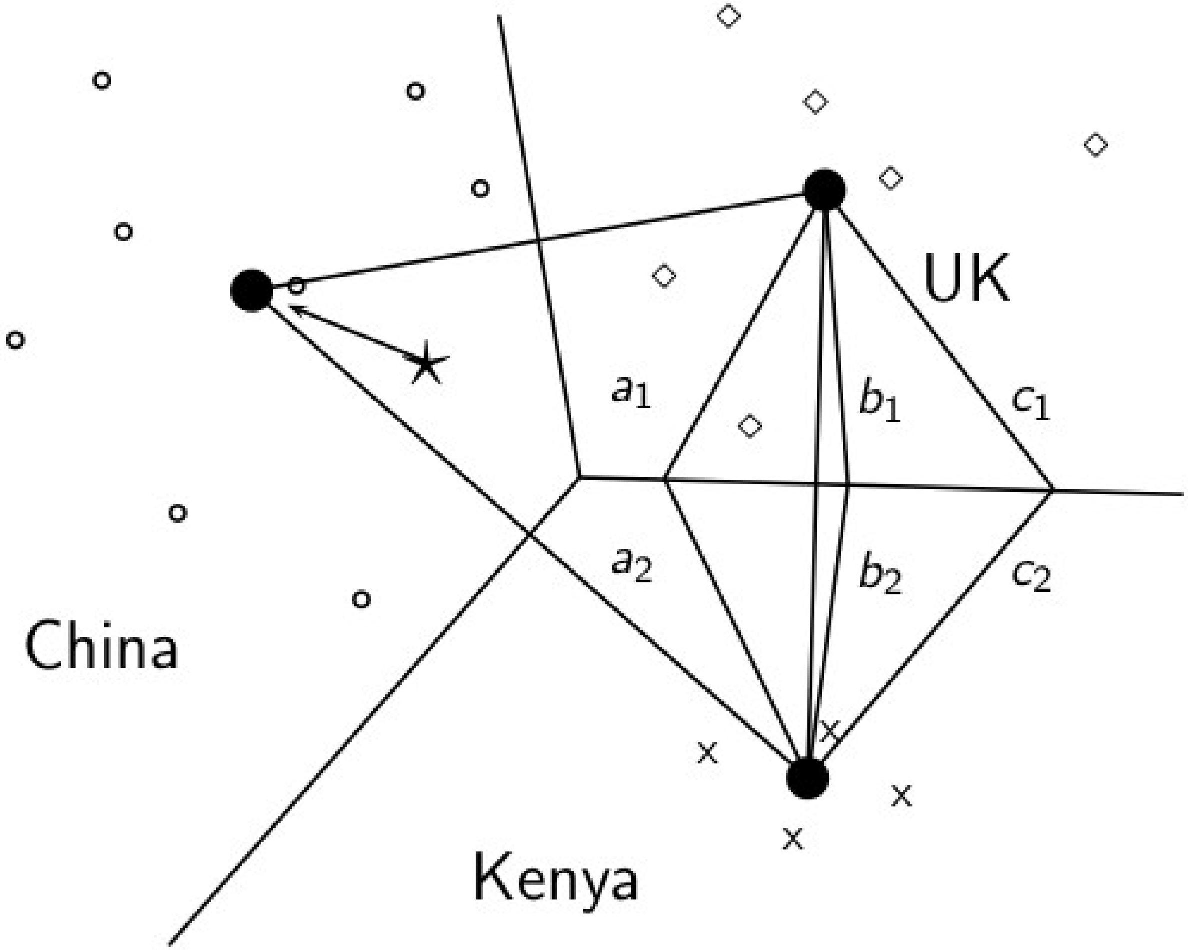
$\text{TrainRocchio}(\mathbb{C}, \mathbb{D})$

**for each** $c_j \in \mathbb{C}$
**do** $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$
$\vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
**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 illustrated : a1 = a2, b1 = b2, c1 = c2



# 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!

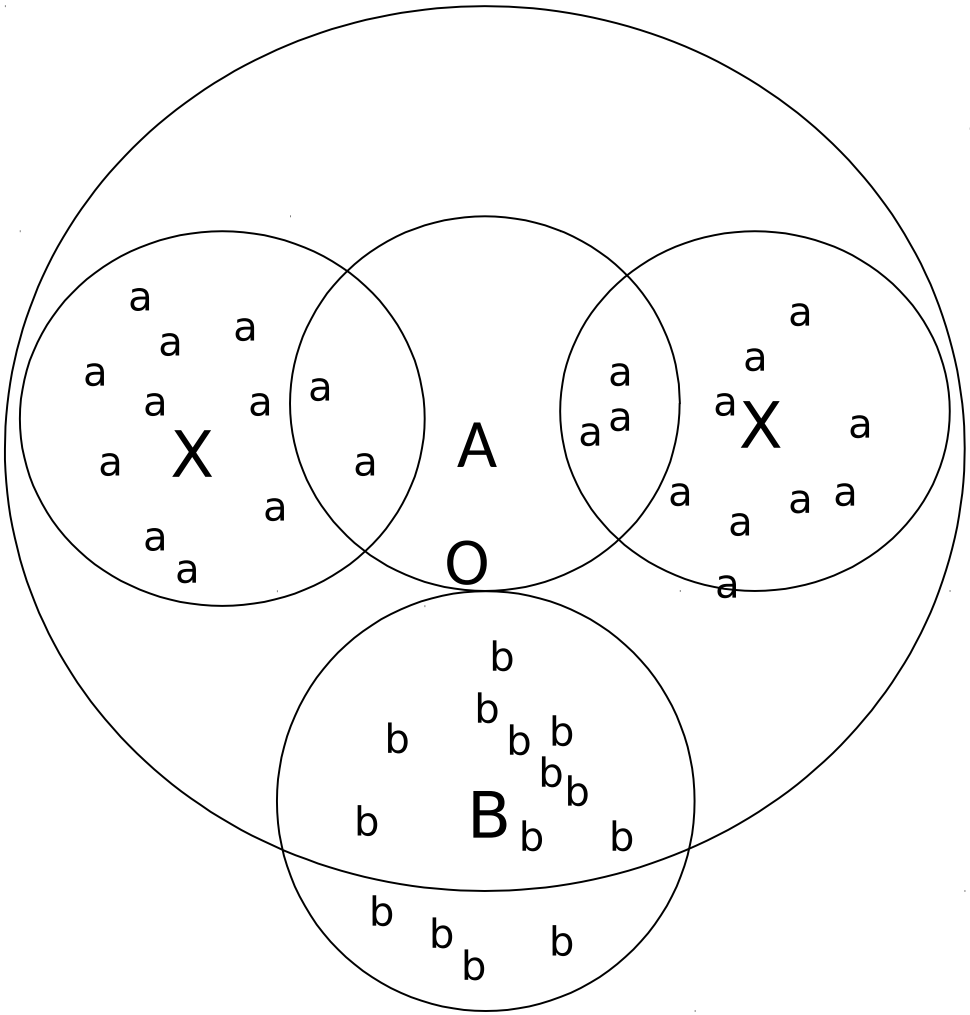
# Time complexity of Rocchio

| mode | time complexity |
|---|---|
| training | $\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert) \approx \Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$ |
| testing | $\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) \approx \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$ |

# 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




Exercise: Why is Rocchio not expected to do well for the classification task a vs. b here?

- 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.

# Outline

1. Recap
2. Feature selection
3. Intro vector space classification
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

# 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.

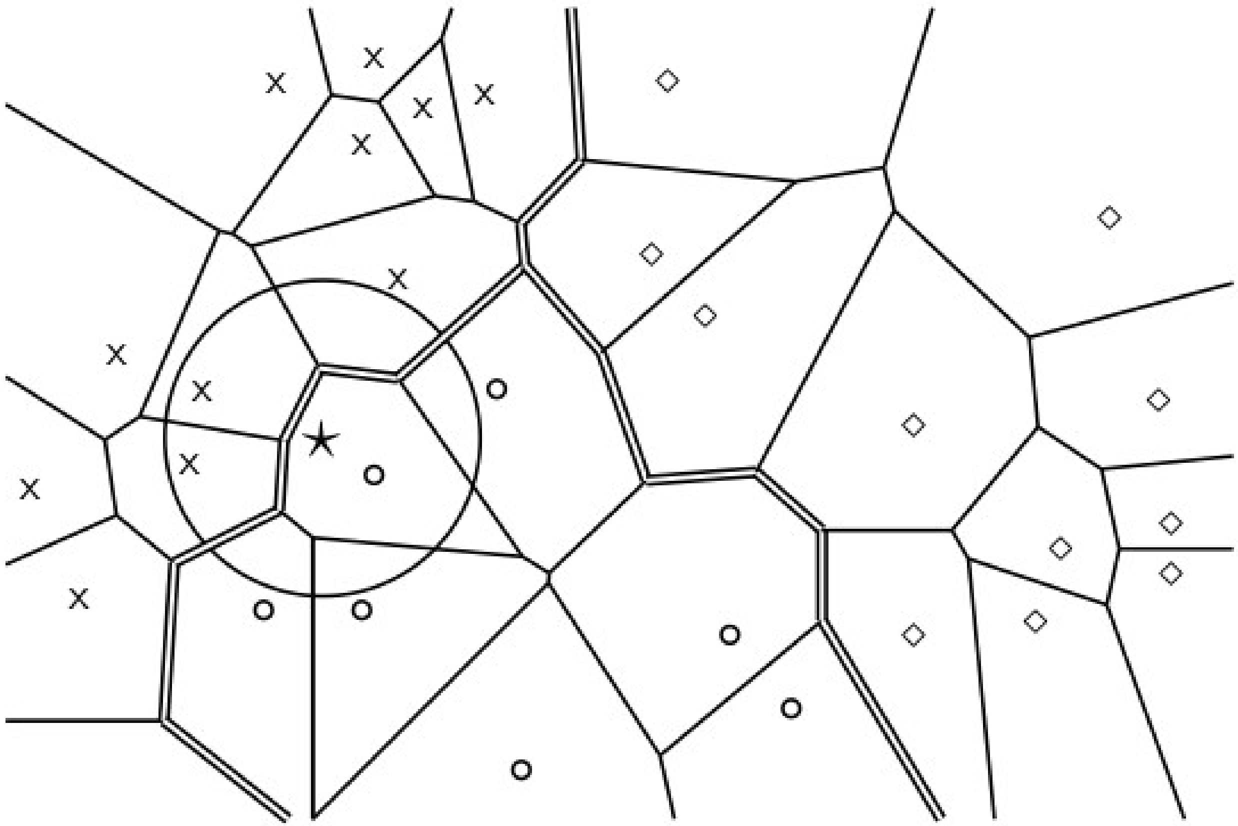
# kNN classification

- kNN = *k* nearest neighbors
- 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*.

# Probabilistic kNN

- Probabilistic version of kNN: $P(c|d)$ = fraction of $k$ neighbors of $d$ that are in $c$
- kNN classification rule for probabilistic kNN: Assign d to class $c$ with highest $P(c|d)$

# Probabilistic kNN



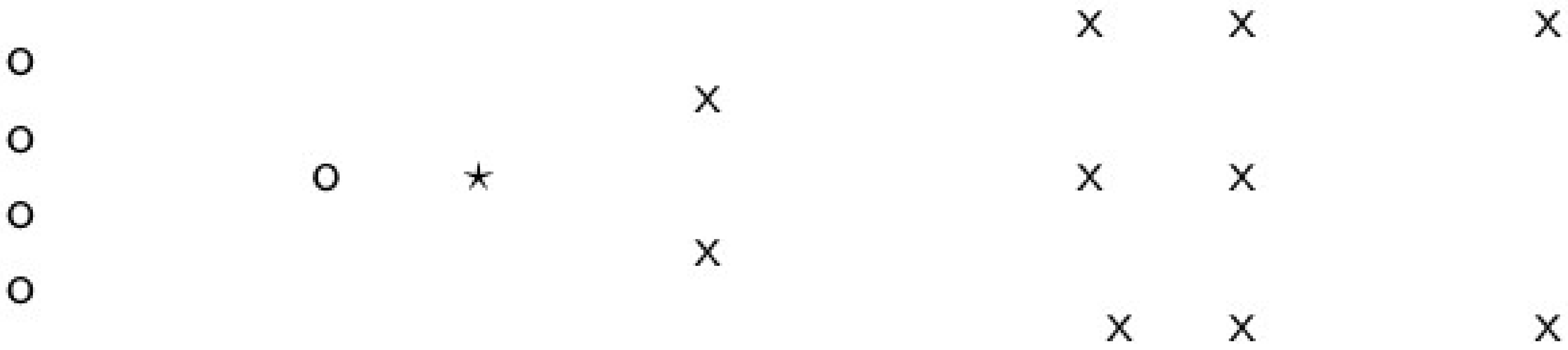
1NN, 3NN classification decision for star?

# kNN algorithm

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

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

# Exercise



How is star classified by:

(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN (v) Rocchio?

# 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 test time proportional to the size of the training set!
- The larger the training set, the longer it takes to classify a test document.
- kNN is inefficient for very large training sets.

# kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
  - We always preprocess the training set, so in reality training time of kNN is linear.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.
- But kNN can be very inaccurate if training set is small.

# Outline

1. Recap
2. Feature selection
3. Intro vector space classification
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

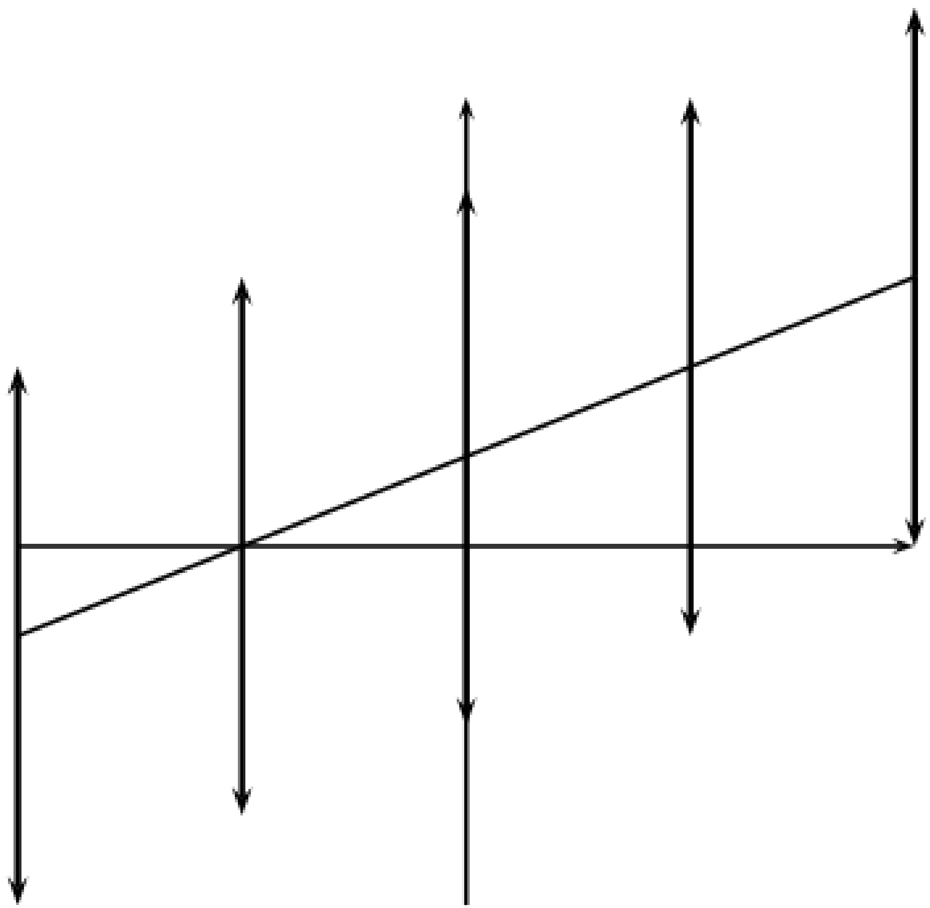
# Linear classifiers

- 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 $\theta$ (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, Naïve Bayes – as we will explain on the next slides
- Assumption: The classes are linearly separable.

# A linear classifier in 1D

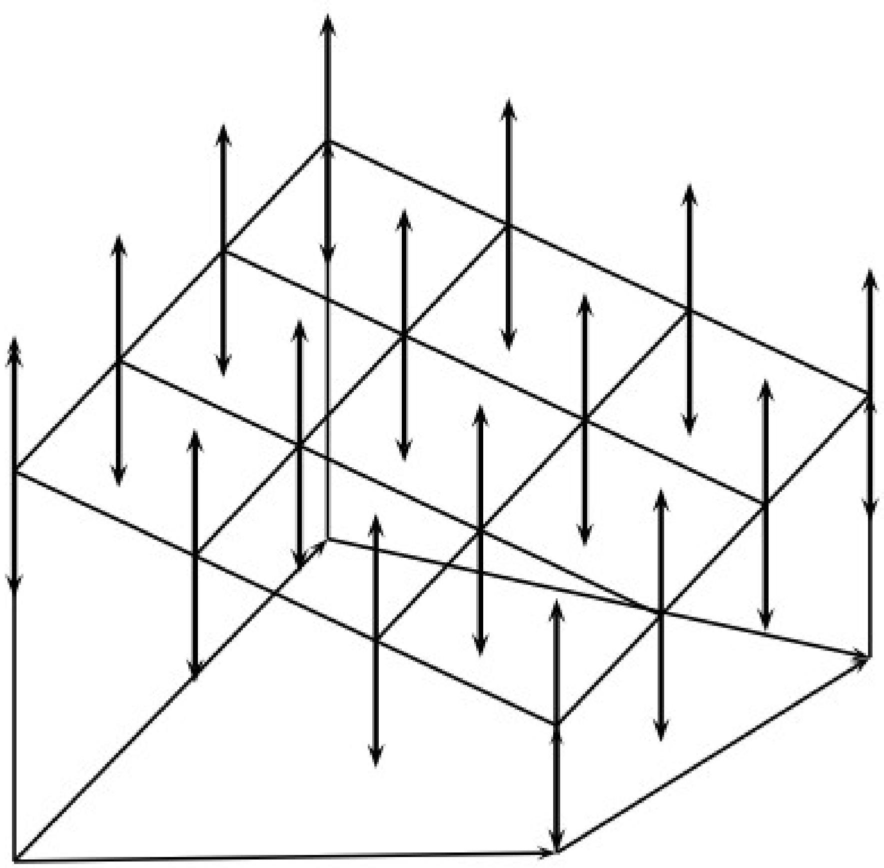
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at $\theta / w_1$
- Points $(d_1)$ with $w_1 d_1 \geq$ are in the class *c*.
- Points $(d_1)$ with $w_1 d_1 < \theta$ are in the complement class $\bar{c}$.

# 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 $\overline{c}$.

# A linear classifier in 2D



- A linear classifier in 3D is a plane described by the equation $w_1d_1 + w_2d_2 + w_3d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1\, d_2\, d_3)$ with $w_1d_1 + w_2d_2 + w_3d_3 \geq \theta$ are in the class $c$.
- Points $(d_1\, d_2\, d_3)$ with $w_1d_1 + w_2d_2 + w_3d_3 < \theta$ are in the complement class $\overline{c}$.

# Rocchio as 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 $\theta = 0.5 * (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$.

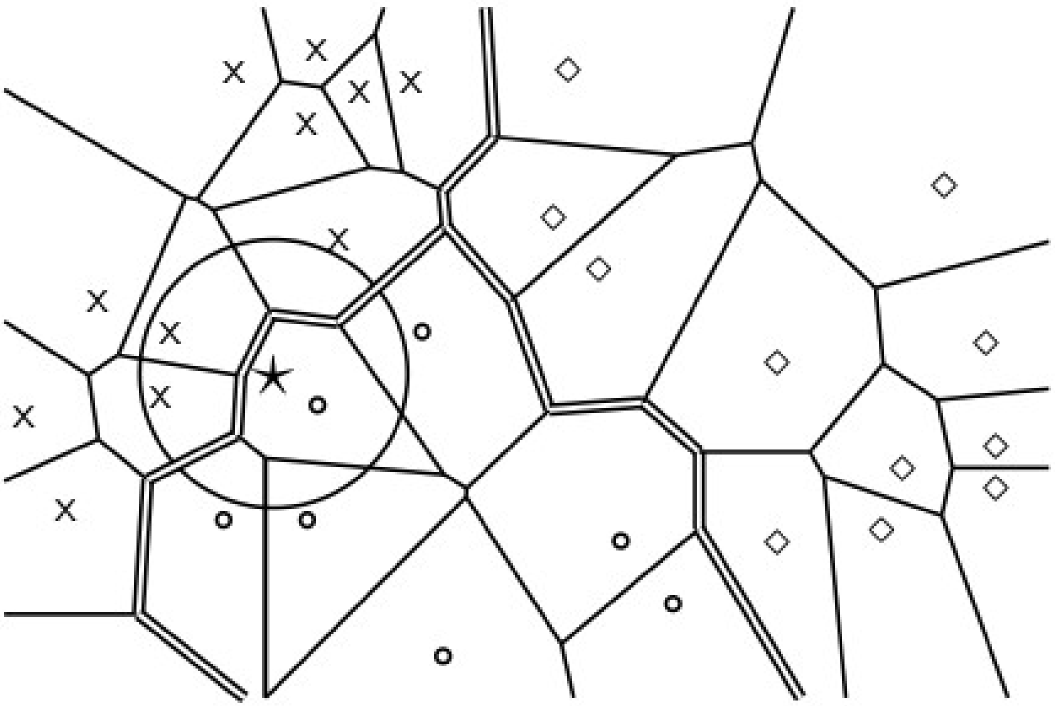
# Naive Bayes as a linear classifier

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

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

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 ≤ *i* ≤ *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



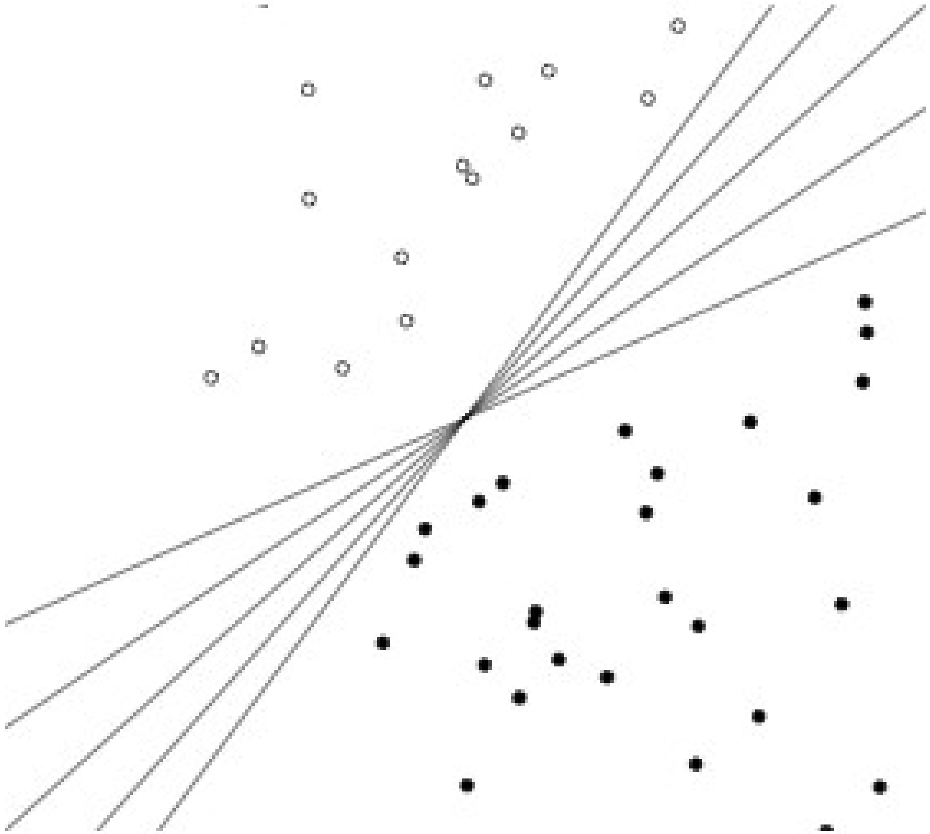
- Classification decision based on majority of *k* nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .
- . . . but they are in general not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta.$

# 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
- $d_1$: "rate discount dlrs world"
- $d_2$: "prime dlrs"
- $\theta = 0$
- Exercise: Which class is $d_1$ assigned to? Which class is $d_2$ assigned to?
- We assign document $\vec{d}_1$ "rate discount dlrs world" to *interest* since
- $\vec{w}^T \vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta$
- We assign $\vec{d}_2$ "prime dlrs" to the complement class (not in *interest*) since
- $\vec{w}^T \vec{d}_2 = -0.01 \leq \theta$.

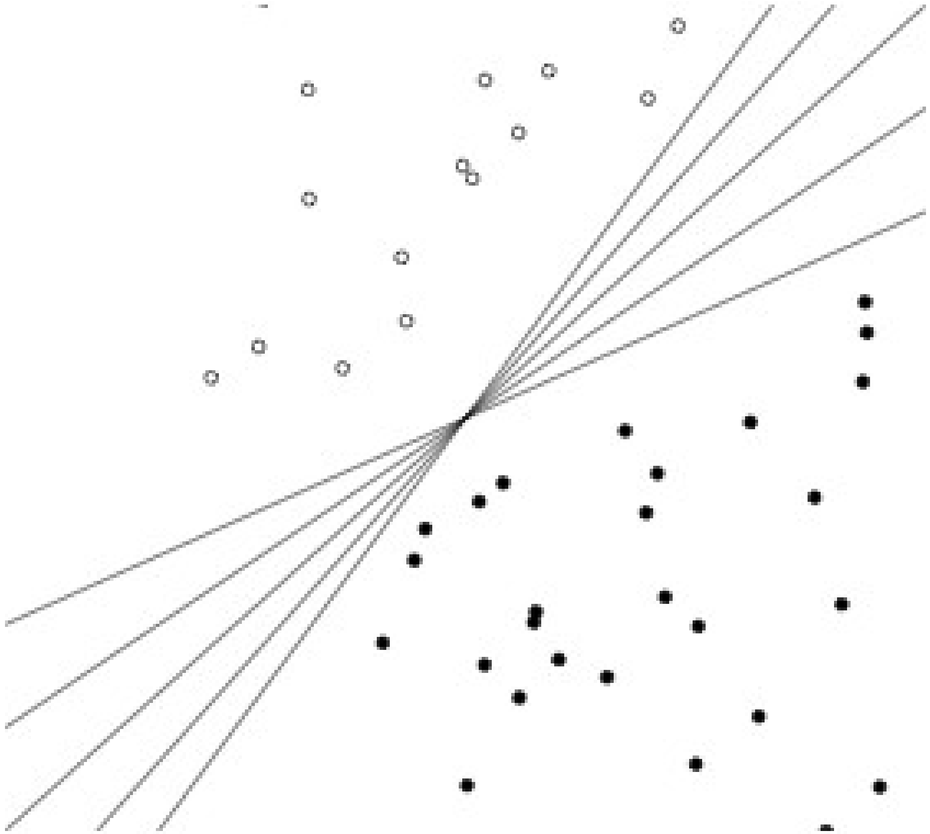
# Which hyperplane?



# 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
  - Perceptron (example available as PDF on website: http://ifnlp.org/ir/pdf/p.pdf)
- The best performing learning algorithms usually require iterative learning.

# Which hyperplane?



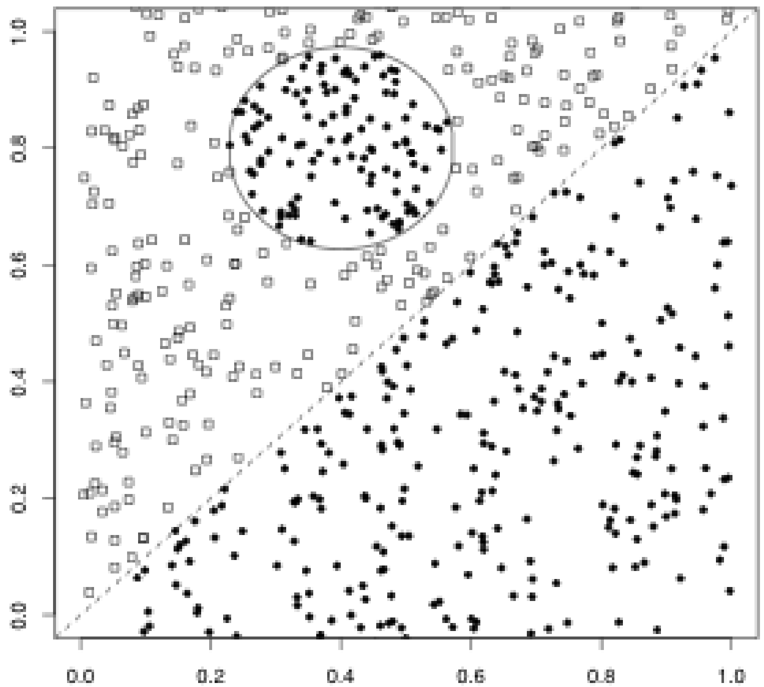
# Which hyperplane?

- 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

# 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 on test documents
- 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



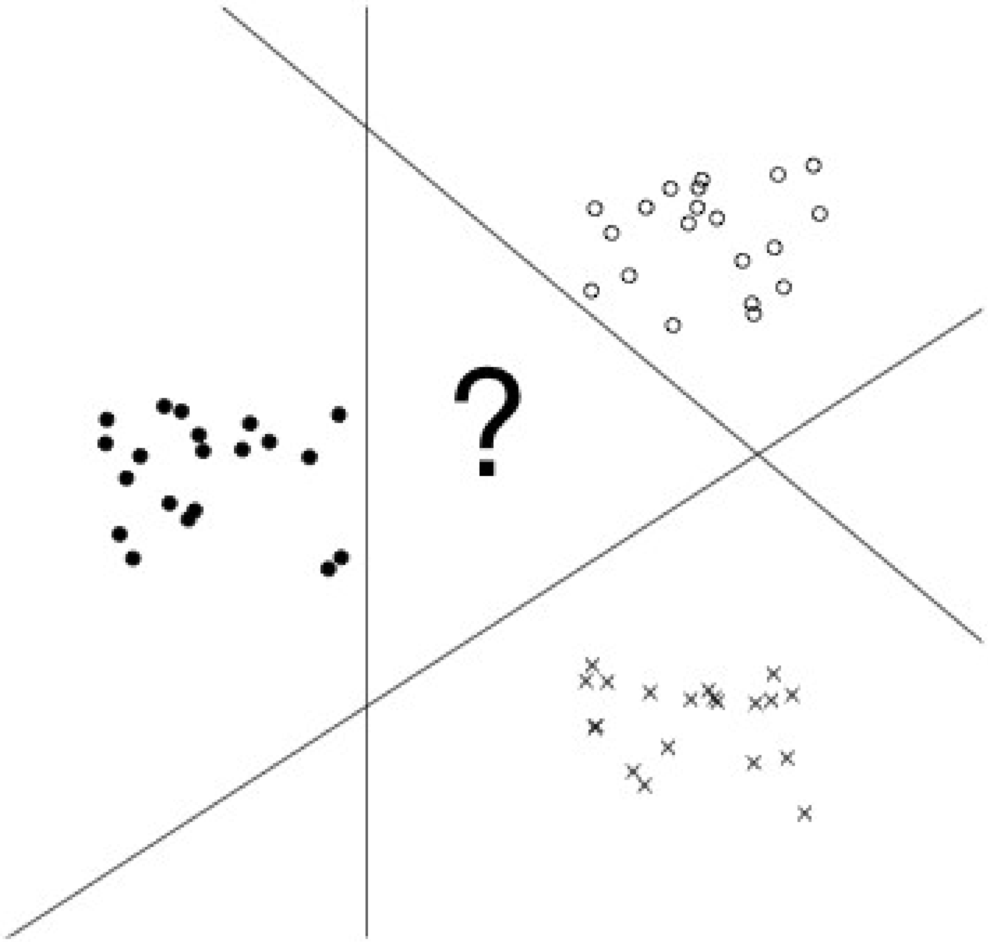

- Linear classifier like Rocchio does badly on this task.
- kNN will do well (assuming enough training data)

# Which classifier do I use for a given TC problem?

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



# One-of problems

- 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)

# One-of classification with linear classifiers

- Combine two-class linear 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

# Any-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 decisions open on all other classes.
  - A type of "independence" (but not statistical independence)
  - Example: topic classification
  - Usually: make decisions on the region, on the subject area, on the industry and so on "independently"

# Any-of classification with linear classifiers

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

# Take-away today

- Feature selection for text classification: How to select a subset of available dimensions
- Vector space classification: Basic idea of doing text classification for documents that are represented as vectors
- Rocchio classifier: Rocchio relevance feedback idea applied to text classification
- *k* nearest neighbor classification
- Linear classifiers
- More than two classes

# Resources

- Chapter 13 of IIR (feature selection)
- 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)