Introduction to Information Retrieval

Vector Space Classification

Outline

1. **Intro vector space classification**
4. Rocchio
5. kNN
6. Linear classifiers
7. > two classes

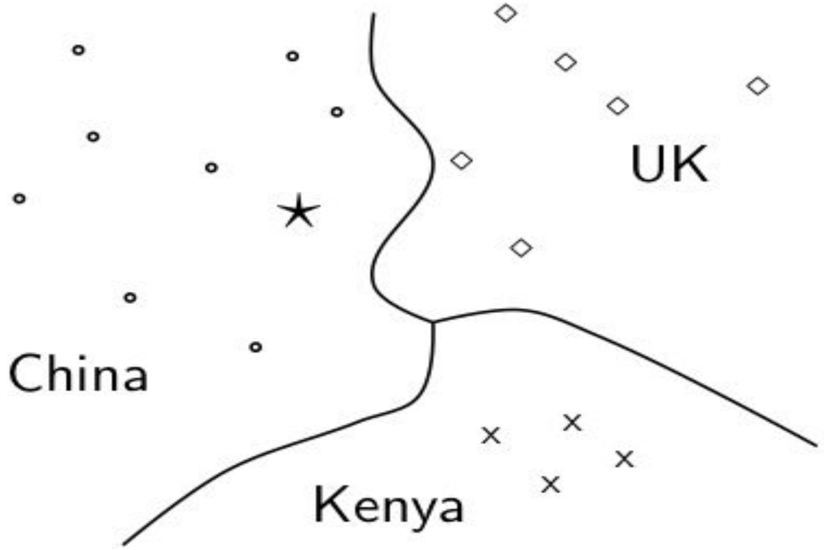
Recall vector space representation

- 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**

Outline

❶ Intro vector space classification

④ **Rocchio**

❺ kNN

❻ Linear classifiers

❼ > two classes

Using Rocchio for vector space classification

- The training set is given as part of the input in text classification.
- 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.

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

where *Dc* is the set of all documents that belong to class c and $\vec{v}(d)$ is the vector space representation of *d*.

Rocchio algorithm

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

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

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

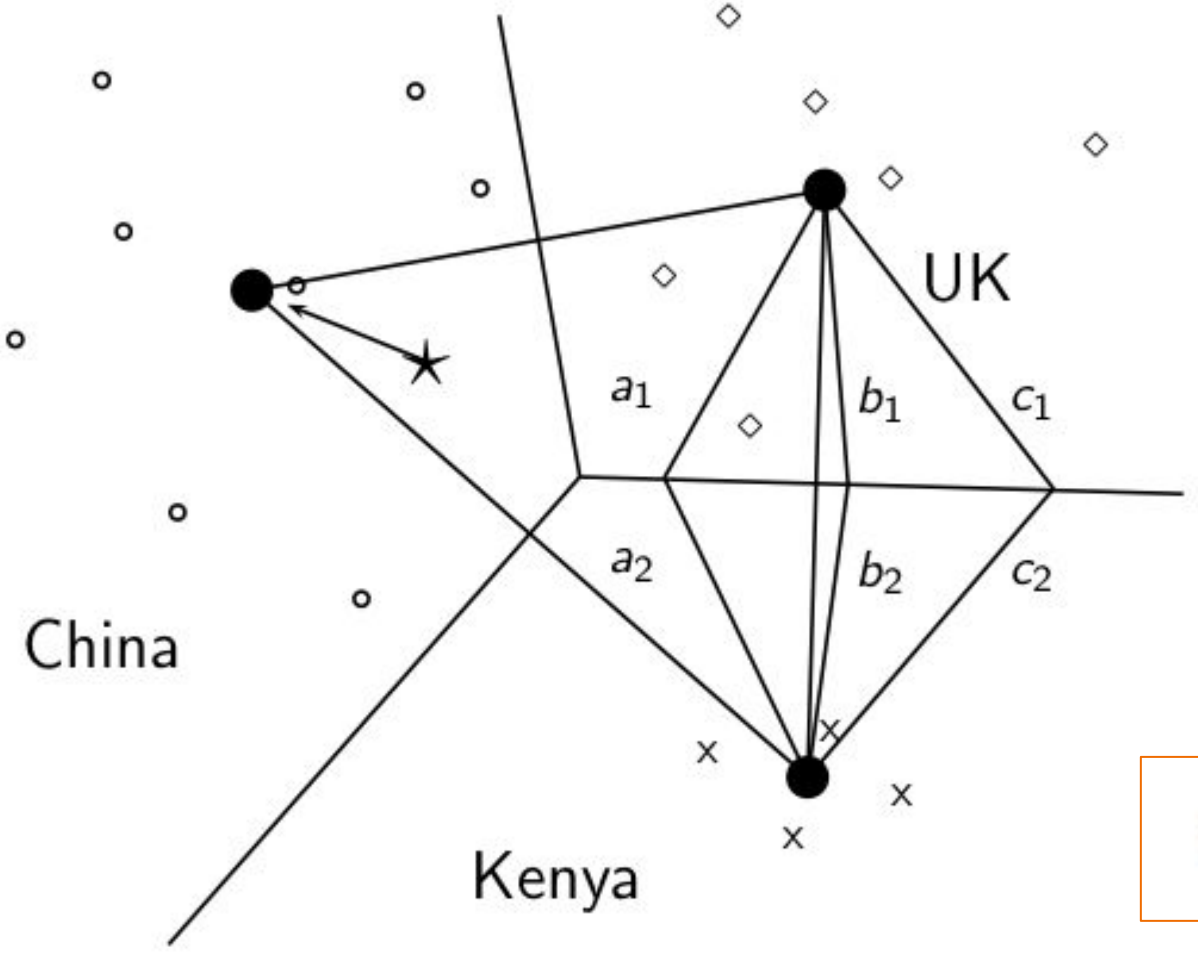
$\quad \vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$

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

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

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

Rocchio illustrated : a1 = a2, b1 = b2, c1 = c2



$$\vec{w}^T\vec{x} = b$$

Rocchio properties

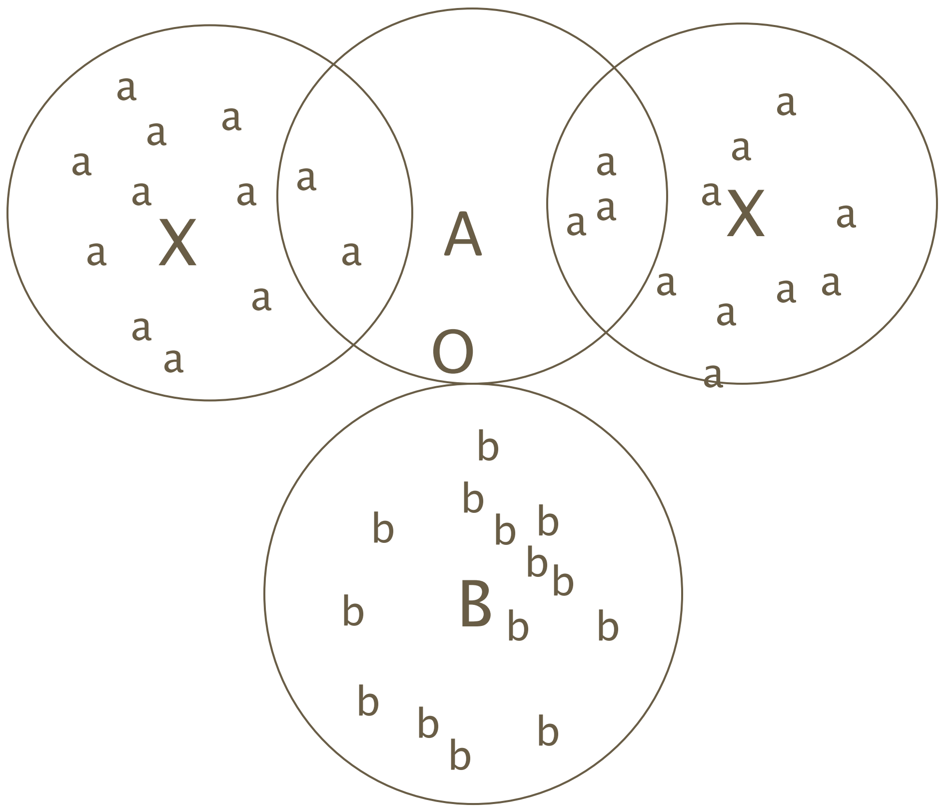
- Rocchio forms a simple representation for each class: **the centroid**
  - We can interpret the centroid as the prototype of the class.
- Classification is based on similarity to / distance from centroid/prototype.
- A hyperplane separates two classes

Time complexity of Rocchio

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

L_{ave}: average length of a training doc, L_a: length of the test doc, M_a: number of distinct terms in the test doc, $\mathbb{D}$: training set, V : vocabulary, $\mathbb{C}$: set of classes

Rocchio cannot handle non-convex, 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

❶ **Intro vector space classification**

❹ Rocchio

❺ kNN

❻ Linear classifiers

❼ > 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

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

- 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 $P(c|d)$

Weight the votes of the k-nearest neighbors by their cosine similarities.

$$\text{score}(c,d) = \sum_{d' \in S_k(d)} I_c(d') \cos(\vec{v}(d'), \vec{v}(d))$$

$S_k d$ – d's closest neighbor. $I_c(d') = 1$ if d' is in class c

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 $P(c|d)$

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 $\mathbb{D}', k$

$\textsc{Apply-kNN}(\mathbb{D}', k, d)$

$S_k \leftarrow \textsc{ComputeNearestNeighbors}(\mathbb{D}', k, d)$

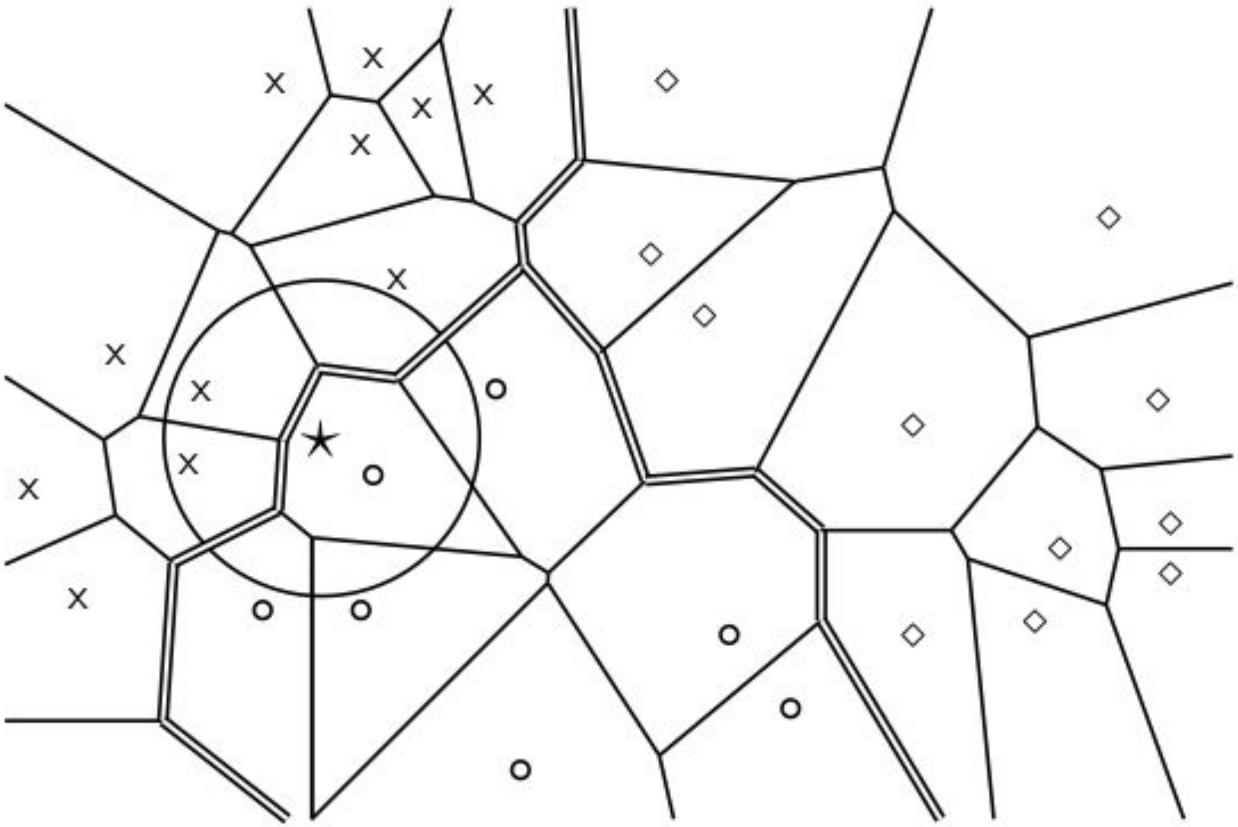
for each $c_j \in \mathbb{C}(\mathbb{D}')$

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

return $\arg\max_j p_j$

Probabilistic kNN

- 1NN, 3NN classification decision for star?



Time complexity of kNN

- kNN with preprocessing of training set
  - training $\Theta(|\mathbb{D}| L_{ave})$
  - testing $\Theta(L_a + |\mathbb{D}| M_{ave} M_a) = \Theta(|\mathbb{D}| M_{ave} M_a)$

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, M_{ave}: average number of distinct terms training set, V: vocabulary, $\mathbb{D}$: training set

- 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
- But kNN can be very inaccurate if training set is small.

Outline

❶ Intro vector space classification

❹ Rocchio

❺ kNN

❻ **Linear classifiers**

❼ > two classes

Linear classifiers

- Definition:
  - A linear classifier computes a linear combination or weighted sum of the feature values. $\sum_i w_i x_i$
  - Classification decision: $\sum_i w_i x_i > \theta?$

    . . .where θ (the threshold) is a parameter.
- We only consider binary classifiers.

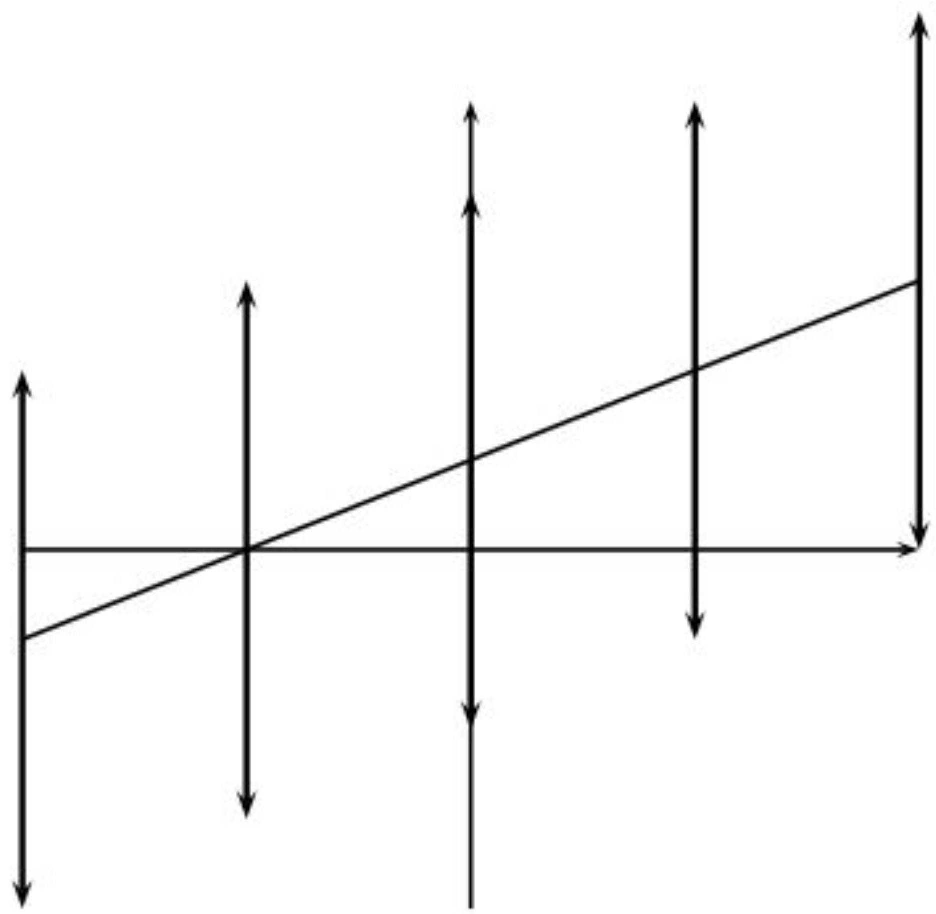
Linear classifiers

- 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

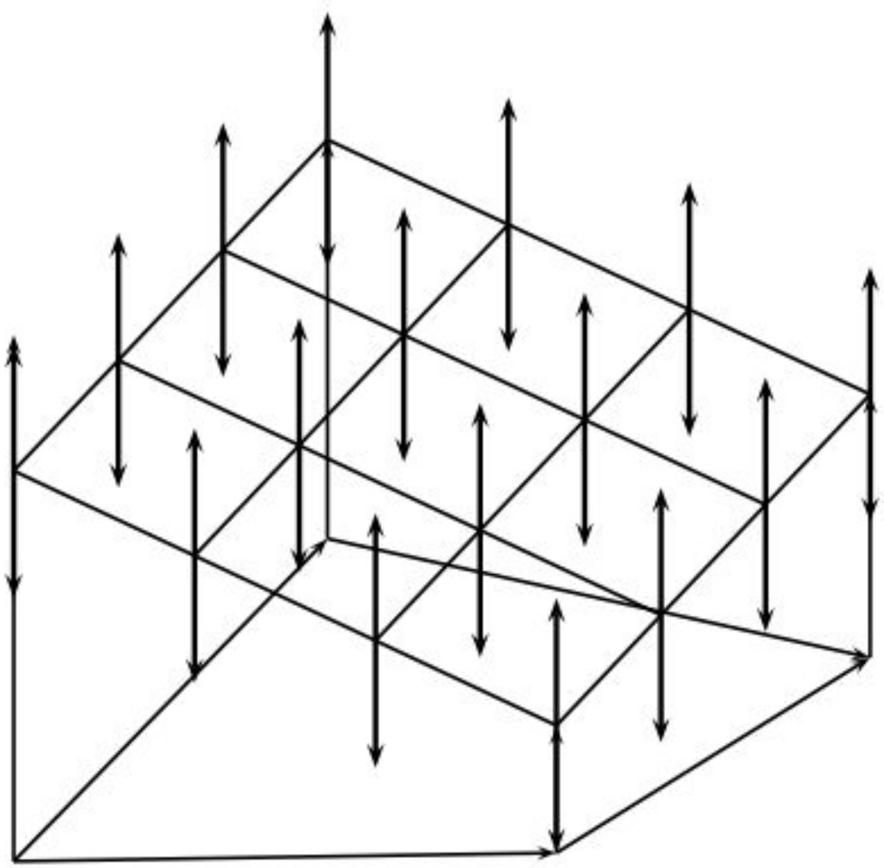
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1\ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w1d1 + w2d2 + w3d3 = \theta$
- Example for a 3D linear classifier
- Points ($d1$ $d2$ $d3$) with $w1d1 + w2d2 + w3d3 \geq \theta$ are in the class c.
- Points ($d1$ $d2$ $d3$) with $w1d1 + w2d2 + w3d3 < \theta$ are in the complement class $\overline{c}$.

Rocchio as a linear classifier

- By definition of decision boundary,

$$|\vec{x} - \vec{\mu}(c_1)| = |\vec{x} - \vec{\mu}(c_2)|$$

- Let us assume,

$$\vec{x} =< x_1, .., x_n >$$
$$\vec{\mu}(c_1) =< \mu_{11}, .., \mu_{1n} >$$
$$\vec{\mu}(c_2) =< \mu_{21}, .., \mu_{2n} >$$

Rocchio as a linear classifier

- Then we find,

$$
\begin{aligned}
&|\vec{x} - \vec{\mu}(c_1)| = |\vec{x} - \vec{\mu}(c_2)| \\
\Longrightarrow\ & \sqrt{\sum_{i=1}^{n}(x_i - \mu_{1i})^2} = \sqrt{\sum_{i=1}^{n}(x_i - \mu_{2i})^2} \\
\Longrightarrow\ & \sum_{i=1}^{n}(x_i - \mu_{1i})^2 = \sum_{i=1}^{n}(x_i - \mu_{2i})^2 \\
\Longrightarrow\ & \sum_{i=1}^{n}(x_i^2 - 2x_i\mu_{1i} + \mu_{1i}^2) = \sum_{i=1}^{n}(x_i^2 - 2x_i\mu_{2i} + \mu_{2i}^2) \\
\Longrightarrow\ & -\sum_{i=1}^{n} 2x_i\mu_{1i} + \sum_{i=1}^{n} \mu_{1i}^2 = -\sum_{i=1}^{n} 2x_i\mu_{2i} + \sum_{i=1}^{n} \mu_{2i}^2 \\
\Longrightarrow\ & 2\sum_{i=1}^{n} x_i(\mu_{2i} - \mu_{1i}) = \sum_{i=1}^{n} \mu_{2i}^2 - \sum_{i=1}^{n} \mu_{1i}^2 \\
\Longrightarrow\ & \vec{x} \cdot (\vec{\mu}(c_2) - \vec{\mu}(c_1)) = \frac{1}{2}|\vec{\mu}(c_2)|^2 - |\vec{\mu}(c_1)|^2
\end{aligned}
$$

Rocchio as a linear classifier

- Finally we can write it as $\overrightarrow{x} \cdot \overrightarrow{w} = b$

  where, $\overrightarrow{w} = (\overrightarrow{\mu}(c_2) - \overrightarrow{\mu}(c_1))$

  and $b = \frac{1}{2}|\overrightarrow{\mu}(c_2)|^2 - |\overrightarrow{\mu}(c_1)|^2$

Naive Bayes as a linear classifier

- Let us recall the decision rule of Naive Bayes
- For a document d, choose category c with largest $\hat{P}(c|d)$ where

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

- Here t_k are the tokens appearing in the document
- Let $\bar{c}$ be the complementary category

Naive Bayes as a linear classifier

- We will classify a document d to class c if

$$
\begin{aligned}
& \hat{P}(c|d) \geqslant \hat{P}(\bar{c}|d) \\
\implies & \frac{\hat{P}(c|d)}{\hat{P}(\bar{c}|d)} \geqslant 1 \\
\implies & \log \frac{\hat{P}(c|d)}{\hat{P}(\bar{c}|d)} \geqslant 0 \\
\implies & \log \frac{\hat{P}(c)}{\hat{P}(\bar{c})} + \sum_{k=1}^{n_d} \log \frac{\hat{P}(t_k|c)}{\hat{P}(t_k|\bar{c})} \geqslant 0
\end{aligned}
$$

- Let n_d be the number of token present in document d

Naive Bayes as a linear classifier

- Now let us assume x=<x_1,..,x_N> is vector representation of the document d, where each x_i denotes the number of time i-th term in the vocabulary is present in document d
- So we can write the decision rule as

$$\log \frac{\hat{P}(c)}{\hat{P}(\bar{c})} + \sum_{i=1}^{N} x_i \cdot \log \frac{\hat{P}(\tilde{t}_i|c)}{\hat{P}(\tilde{t}_i|\bar{c})} \geqslant 0$$

- N denotes the number of terms present in vocabulary
- $\tilde{t}_i$ represent i-th term in vocabulary

Naive Bayes as a linear classifier

- We can write the classifier as

$$\overrightarrow{x} \cdot \overrightarrow{w} = b$$

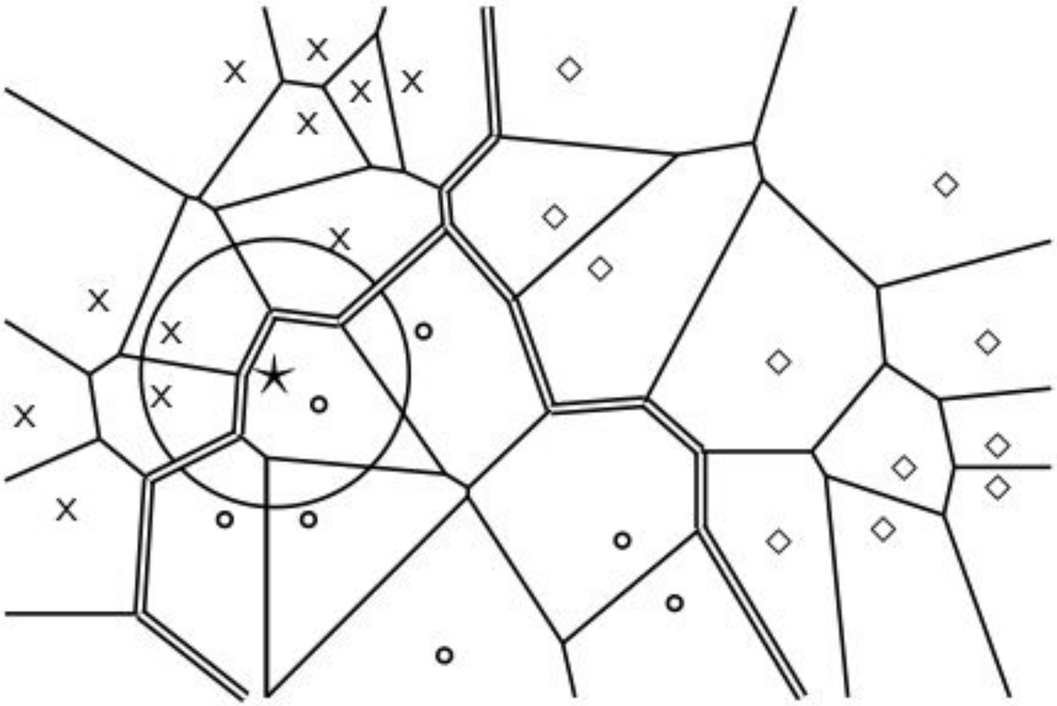
where x_i denotes number of occurrence of i-th term of vocabulary in d

$$w_i = \log \frac{\hat{P}(\tilde{t}_i|c)}{\hat{P}(\tilde{t}_i|\bar{c})}$$

$$b = -\log \frac{\hat{P}(c)}{\hat{P}(\bar{c})}$$

So, in log space, Naive Bayes is a linear classifier.

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”
- b = 0

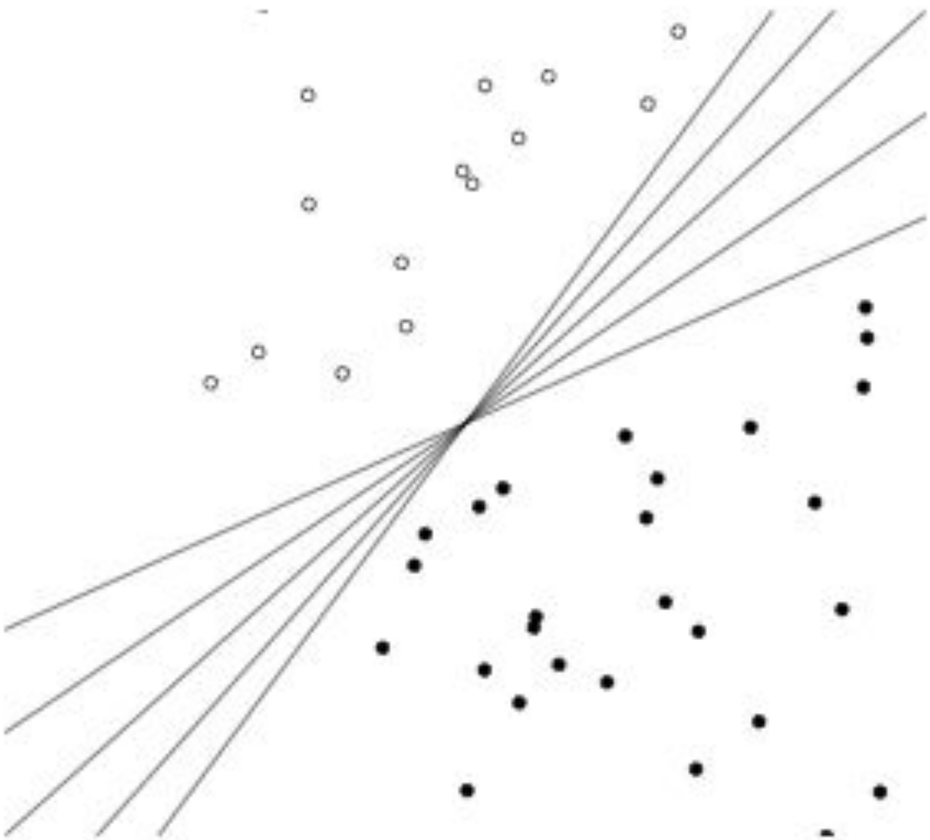
Exercise: Which class is d_1 assigned to? Which class is d_2 assigned to?

- We assign document d_1 “rate discount dlrs world” to *interest* since

  $\mathbf{w}^T\mathbf{d_1}$ = 0.67 · 1 + 0.46 · 1 + (−0.71) · 1 + (−0.35) · 1 = 0.07 > 0 = b.
- We assign d_2 “prime dlrs” to the complement class (not in *interest*)

  since $\mathbf{w}^T\mathbf{d_2}$ = 0.70 - 0.71 = −0.01 ≤ 0 = b.

Which hyperplane?

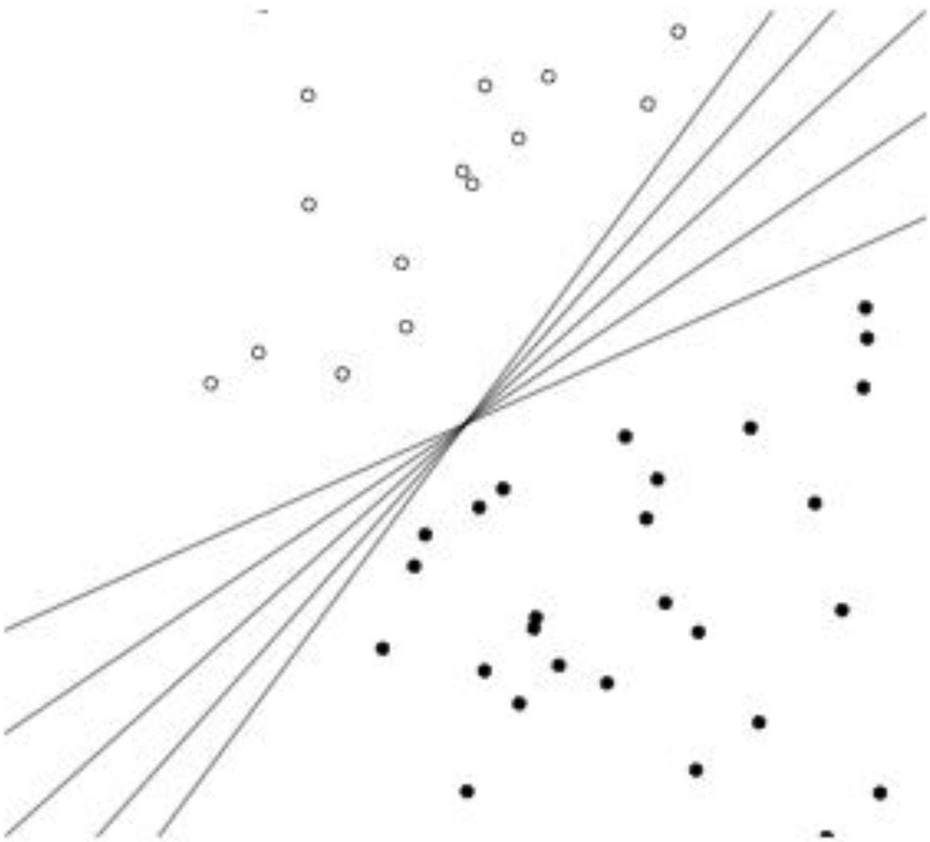


Learning algorithms for vector space classification

- In terms of actual computation, there are two types of learning algorithms.
    a. Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass.
        i. Naive Bayes, Rocchio, kNN are all examples of this.
    b. Iterative algorithms
        i. Support vector machines
        ii. Perceptron

The best performing learning algorithms usually require iterative learning.

Which hyperplane?



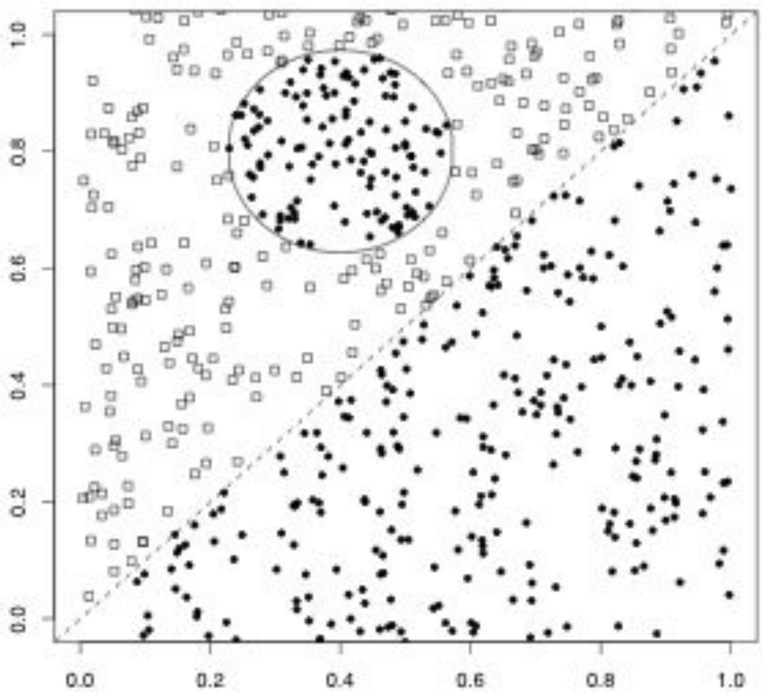
Which hyperplane?

- 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
- Non-linear classifier requires .huge amount of training data.

A nonlinear problem



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

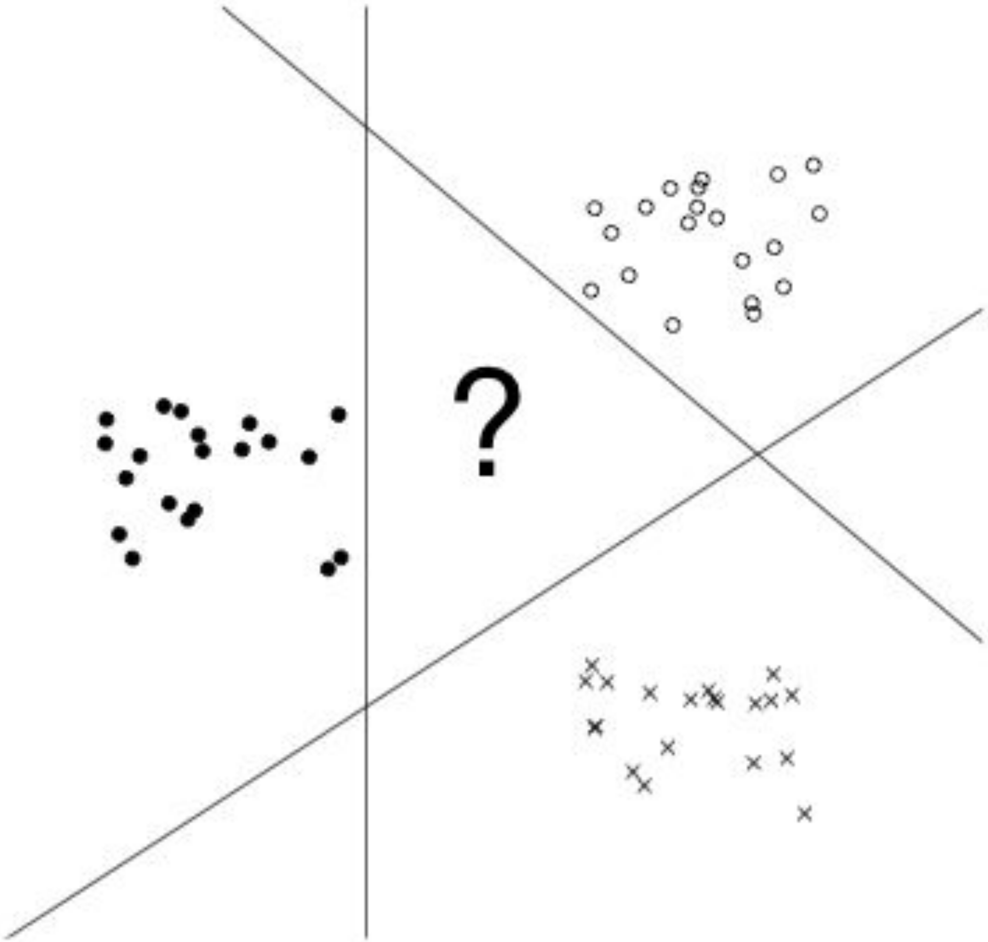
Which classifier do I use for a given TC problem?

- Is there a learning method that is optimal for all text classification problems? . .
- 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 unstable problem, it’s better to use a simple and robust classifier.

Outline

1. **Intro vector space classification**
4. Rocchio
5. kNN
6. Linear classifiers
7. **> two classes**

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

- Build a classifier for each class, where the training set consists of the set of documents in the class (positive labels) and its complement (negative labels)
- Combine the two-class linear classifiers as follows for one-of classification:
    - Run each classifier separately
    - Rank classifiers (eg., according to score, confidence value, probablity)
    - 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

- 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

Thank you

Questions?