Introduction to Information Retrieval

http://informationretrieval.org

IIR 14: Vector Space Classification

Hinrich Schütze

Center for Information and Language Processing, University of Munich

2013-05-28

Overview

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

Outline

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

Feature selection: MI for *poultry*/EXPORT

Goal of feature selection: eleminate noise and useless features for better effectiveness and efficiency

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

Plug these values into formula:

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

Feature selection for Reuters classes coffee and sports

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

Using language models (LMs) for IR

- LM = language model
- We view the document as a generative model that generates the query.
- What we need to do:
- Define the precise generative model we want to use
- Estimate parameters (different parameters for each document's model)
- Smooth to avoid zeros
- Apply to query and find document most likely to have generated the query
- Present most likely document(s) to user

Jelinek-Mercer smoothing

- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- High value of λ: "conjunctive-like" search – tends to retrieve documents containing all query words.
- Low value of λ: more disjunctive, suitable for long queries
- Correctly setting λ is very important for good performance.

Take-away today

Take-away today

- Vector space classification: Basic idea of doing text classification for documents that are represented as vectors

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

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

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

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

Outline

Recall vector space representation

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

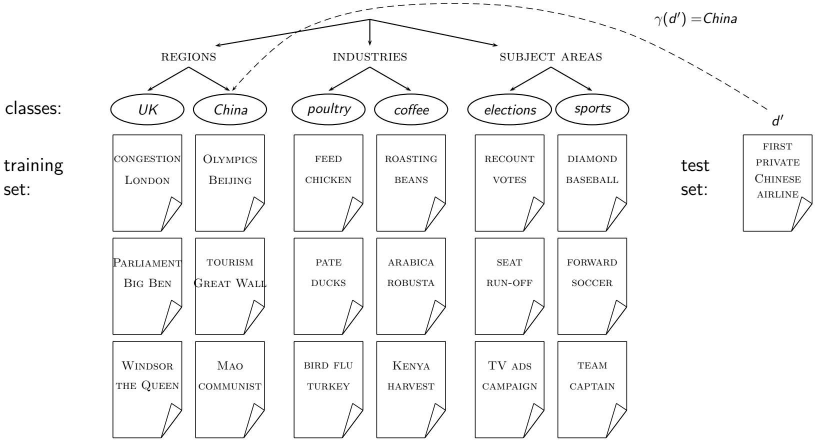
Recall vector space representation

- 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?

Basic text classification setup



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

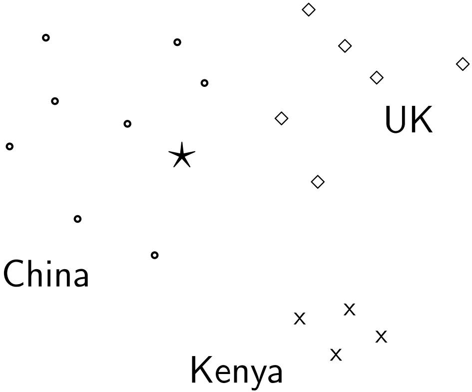
- As before, the training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a contiguous region.
- Premise 2: Documents from different 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.

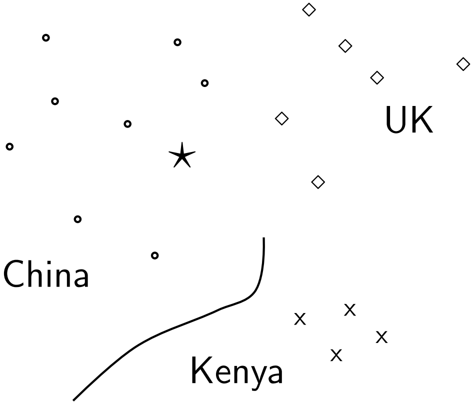
Classes in the vector space

Classes in the vector space



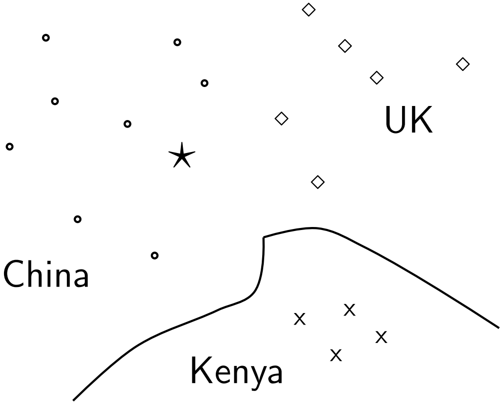
Should the document $\star$ be assigned to *China*, *UK* or *Kenya*?

Classes in the vector space



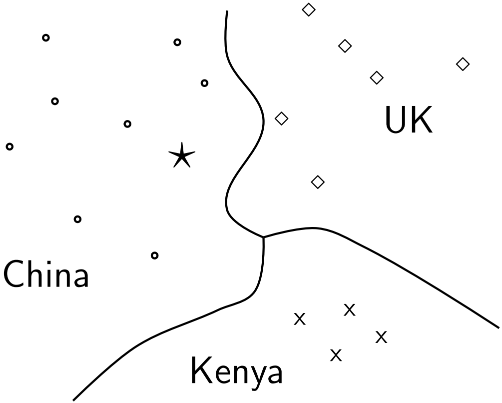
Find separators between the classes

Classes in the vector space



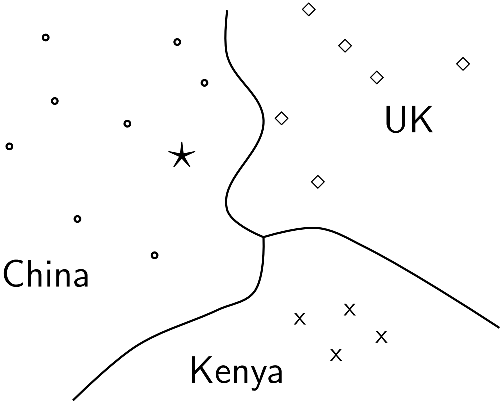
Find separators between the classes

Classes in the vector space



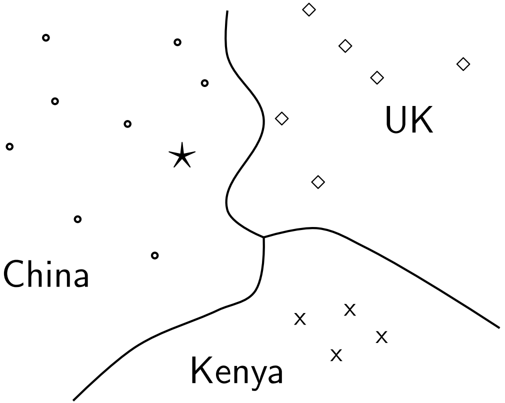
Find separators between the classes

Classes in the vector space



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

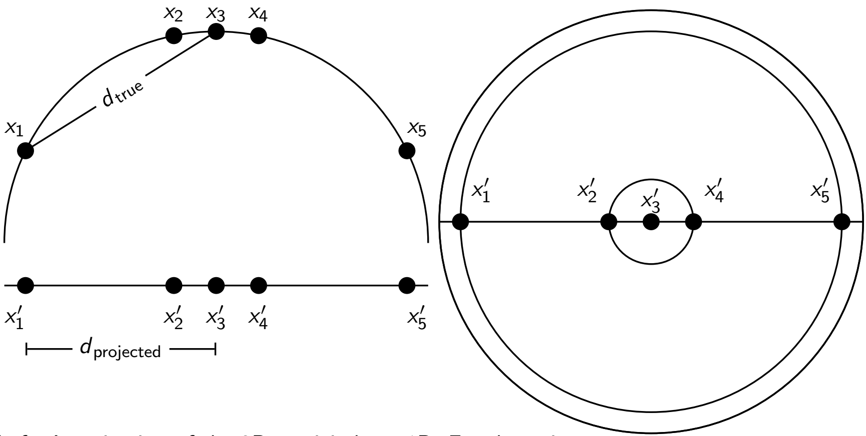
Classes in the vector space



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

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

1. Recap
2. Intro vector space classification
3. Rocchio
4. kNN
5. Linear classifiers
6. > 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:

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.

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

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.

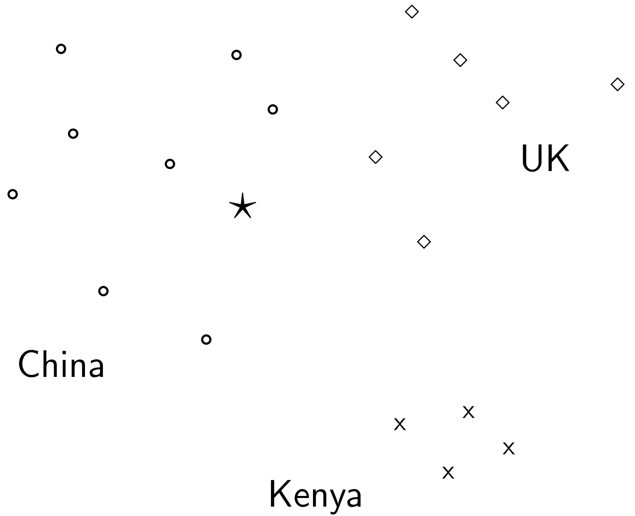
Recall definition of centroid

Recall definition of centroid

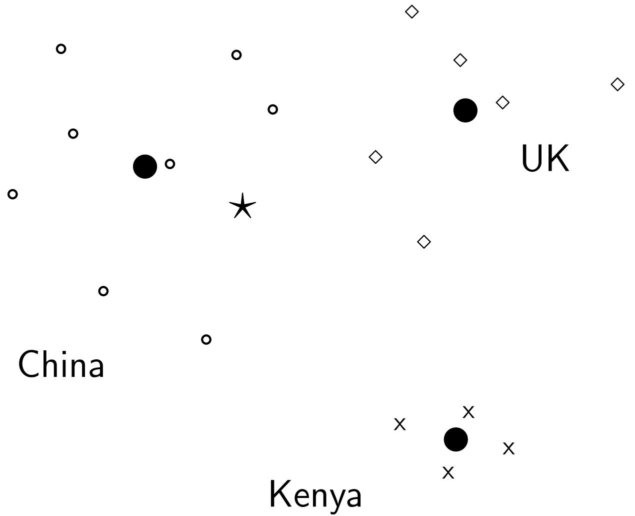
$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

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

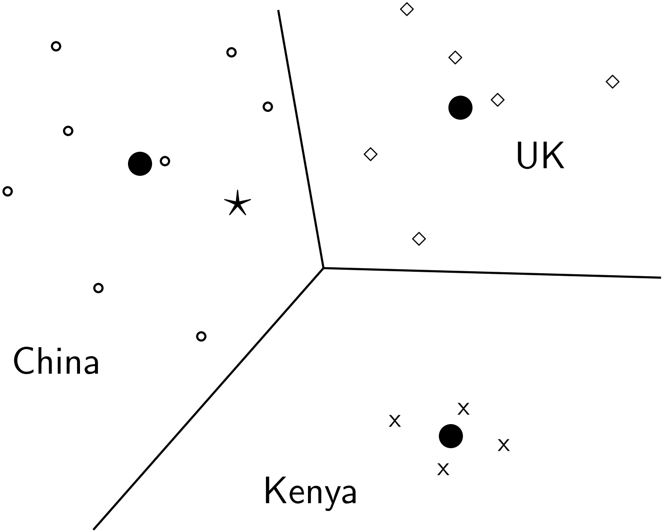
Rocchio illustrated



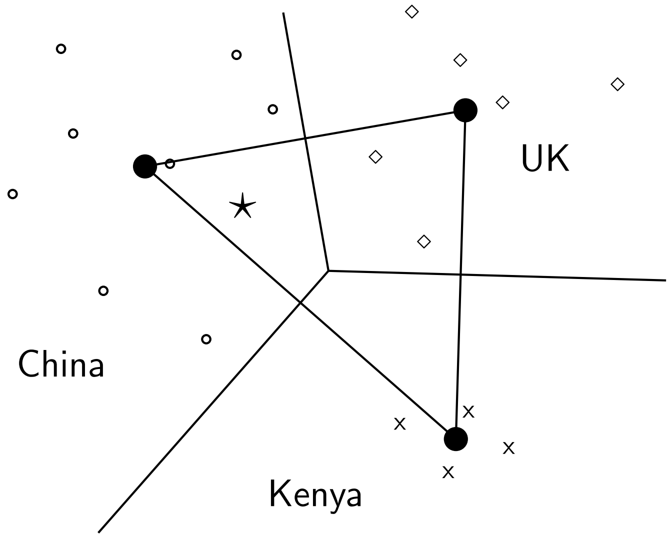
Rocchio illustrated



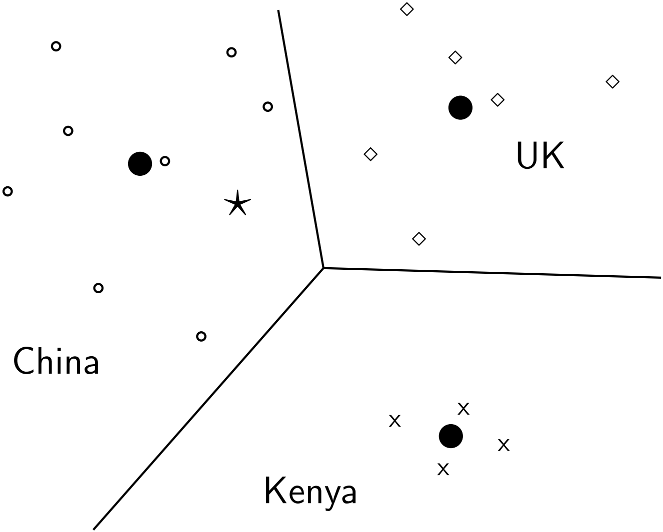
Rocchio illustrated



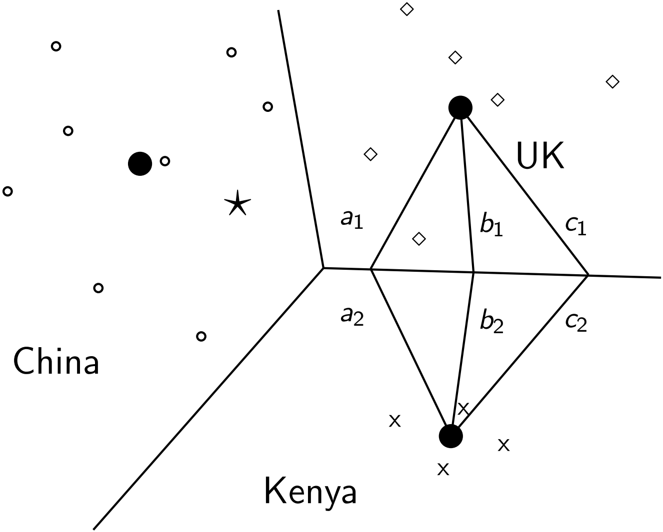
Rocchio illustrated



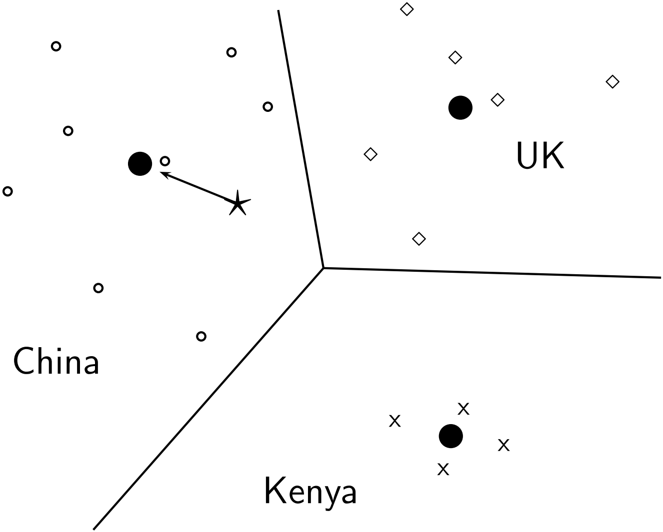
Rocchio illustrated



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



Rocchio illustrated



Rocchio algorithm

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 properties

Rocchio properties

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

Rocchio properties

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

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.

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

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

Rocchio vs. Naive Bayes

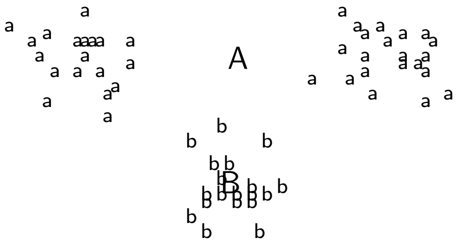
- In many cases, Rocchio performs worse than Naive Bayes.

Rocchio vs. Naive Bayes

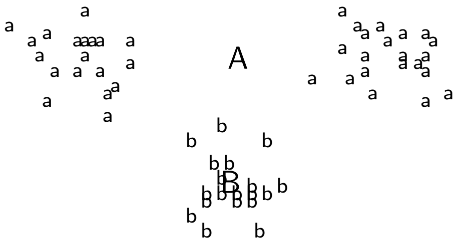
- In many cases, Rocchio performs worse than Naive Bayes.
- One reason: Rocchio does not handle nonconvex, multimodal classes correctly.

Rocchio cannot handle nonconvex, multimodal classes

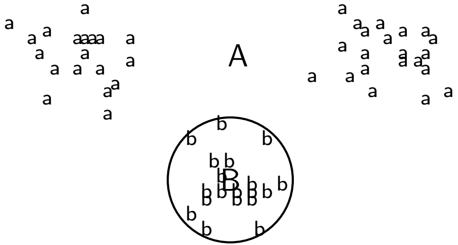
Rocchio cannot handle nonconvex, multimodal classes



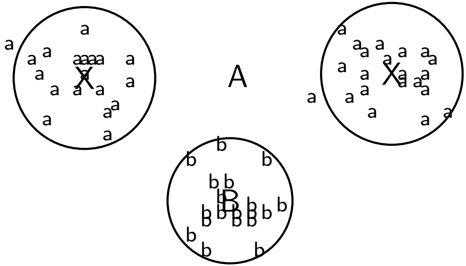
Rocchio cannot handle nonconvex, multimodal classes



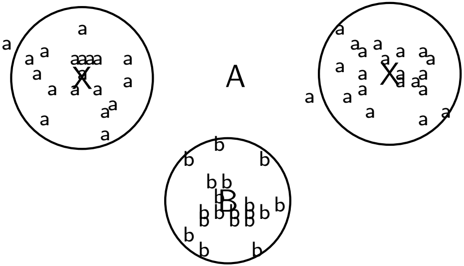
Rocchio cannot handle nonconvex, multimodal classes



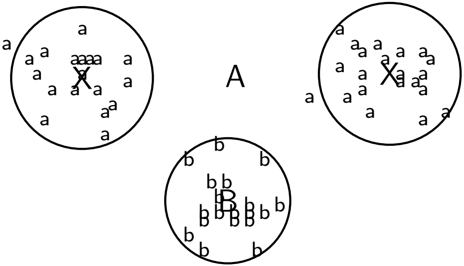
Rocchio cannot handle nonconvex, multimodal classes



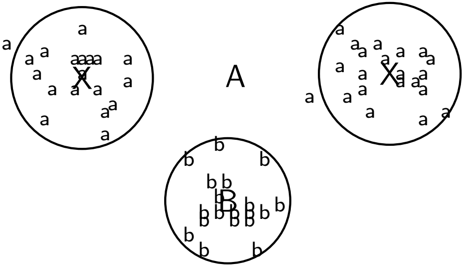
Rocchio cannot handle nonconvex, multimodal classes



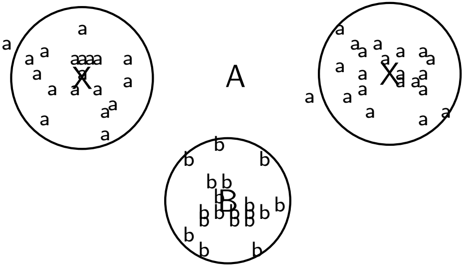
Rocchio cannot handle nonconvex, multimodal classes



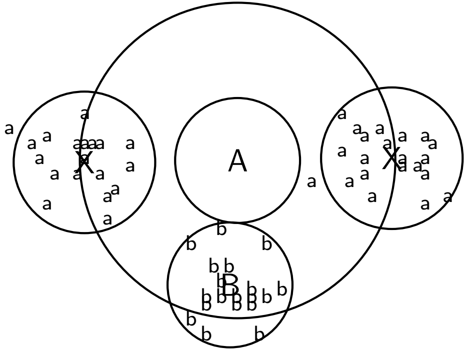
Rocchio cannot handle nonconvex, multimodal classes



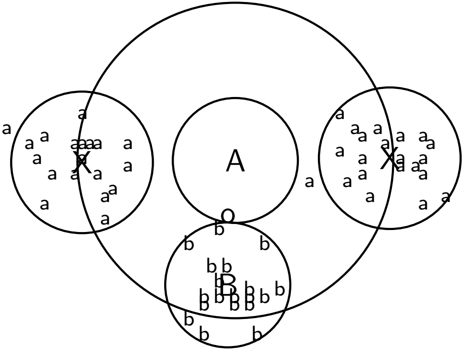
Rocchio cannot handle nonconvex, multimodal classes



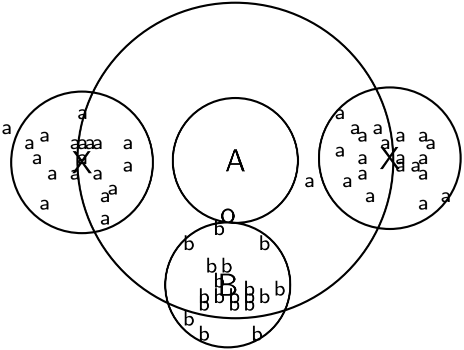
Rocchio cannot handle nonconvex, multimodal classes



Rocchio cannot handle nonconvex, multimodal classes



Rocchio cannot handle nonconvex, multimodal classes



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

Outline

kNN classification

kNN classification

- kNN classification is another vector space classification method.

kNN classification

- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.

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.

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 . . .

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 ...

kNN classification

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

kNN classification

kNN classification

- kNN = k nearest neighbors

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.

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 = 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.

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

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 kNN

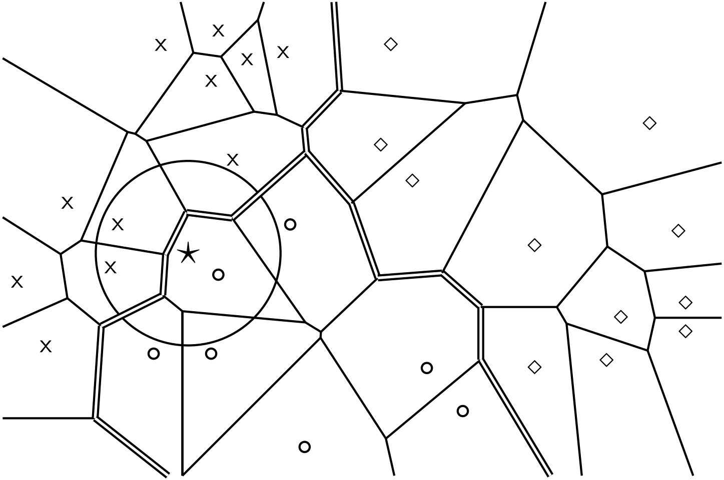
- Probabilistic version of kNN: $P(c|d) =$ fraction of k neighbors of d that are in c

Probabilistic kNN

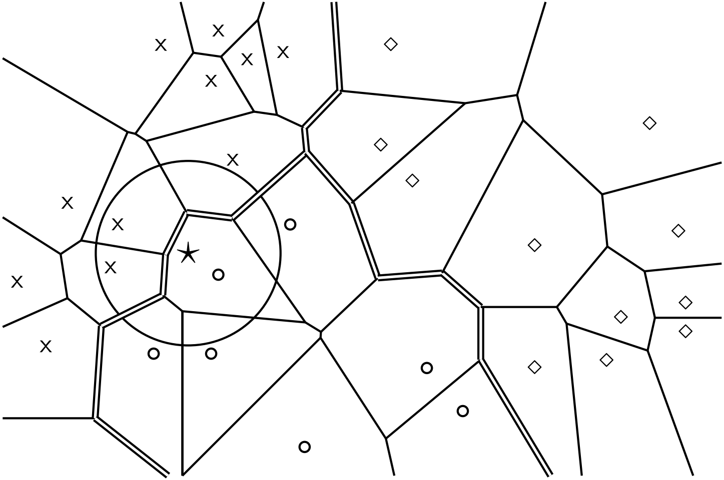
- Probabilistic version of kNN: $P(c|d) =$ fraction of k neighbors of d that are in c
- kNN classification rule for probabilistic kNN: Assign d to class c with highest $P(c|d)$

kNN is based on Voronoi tessellation

kNN is based on Voronoi tessellation



kNN is based on Voronoi tessellation



kNN algorithm

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 $\mathbb{D}', k$

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

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

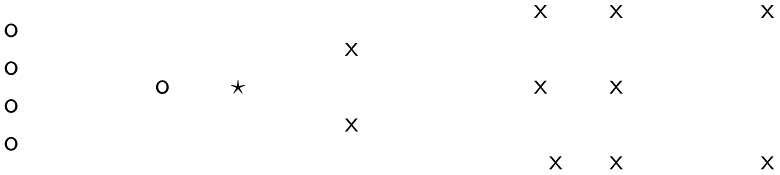
for each $c_j \in \mathbb{C}(\mathbb{D}')$

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

return $\arg\max_j p_j$

Exercise

Exercise



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

Time complexity of kNN

Time complexity of kNN

kNN with preprocessing of training set

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

Time complexity of kNN

kNN with preprocessing of training set

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

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

- kNN test time proportional to the size of the training set!

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.

Time complexity of kNN

kNN with preprocessing of training set

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

- 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.

Time complexity of kNN

kNN with preprocessing of training set

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

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

- 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.
- Question: Can we divide up the training set into regions, so that we only have to search in one region to do kNN classification for a given test document? (which perhaps would give us better than linear time complexity)

Curse of dimensionality

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- These two intuitions don't necessarily hold for high dimensions.

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- These two intuitions don't necessarily hold for high dimensions.
- In particular: for a set of k uniformly distributed points, let dmin be the smallest distance between any two points and dmax be the largest distance between any two points.

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- These two intuitions don't necessarily hold for high dimensions.
- In particular: for a set of k uniformly distributed points, let dmin be the smallest distance between any two points and dmax be the largest distance between any two points.
- Then

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

Curse of dimensionality: Simulation

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

Curse of dimensionality: Simulation

- Simulate
$$\lim_{d\to\infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$
- Pick a dimensionality d

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

- Pick a dimensionality d
- Generate 10 random points in the d-dimensional hypercube (uniform distribution)

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

- Pick a dimensionality d
- Generate 10 random points in the d-dimensional hypercube (uniform distribution)
- Compute all 45 distances

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

- Pick a dimensionality d
- Generate 10 random points in the d-dimensional hypercube (uniform distribution)
- Compute all 45 distances
- Compute $\frac{\text{dmax}-\text{dmin}}{\text{dmin}}$

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

- Pick a dimensionality d
- Generate 10 random points in the d-dimensional hypercube (uniform distribution)
- Compute all 45 distances
- Compute $\frac{\text{dmax}-\text{dmin}}{\text{dmin}}$
- We see that intuition 1 (some things are close, others are distant) is not true for high dimensions.

Intuition 2: Space can be carved up

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.
- To find the k closest neighbors of data point $< x_1, x_2, \ldots, x_d >$ do the following.

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.
- To find the k closest neighbors of data point $< x_1, x_2, \ldots, x_d >$ do the following.
- Using binary search find all data points whose first dimension is in $[x_1 - \epsilon, x_1 + \epsilon]$. This is $O(\log n)$ where n is the number of data points.

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.
- To find the k closest neighbors of data point $< x_1, x_2, \ldots, x_d >$ do the following.
- Using binary search find all data points whose first dimension is in $[x_1 - \epsilon, x_1 + \epsilon]$. This is $O(\log n)$ where n is the number of data points.
- Do this for each dimension, then intersect the d subsets.

Intuition 2: Space can be carved up

Intuition 2: Space can be carved up

- Size of data set $n = 100$

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?
- for $d = 4$: $1 - (1 - 0.1^4)^{100} \approx 0.0095$

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?
- for $d = 4$: $1 - (1 - 0.1^4)^{100} \approx 0.0095$
- In $d = 5$ dimensions?

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?
- for $d = 4$: $1 - (1 - 0.1^4)^{100} \approx 0.0095$
- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$

Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$

Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
- In other words: with enough dimensions, there is only one "local" region that will contain the nearest neighbor with high certainty: the entire search space.

Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
- In other words: with enough dimensions, there is only one "local" region that will contain the nearest neighbor with high certainty: the entire search space.
- We cannot carve up high-dimensional space into neat neighborhoods . . .

Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
- In other words: with enough dimensions, there is only one "local" region that will contain the nearest neighbor with high certainty: the entire search space.
- We cannot carve up high-dimensional space into neat neighborhoods ...
- ... unless the "true" dimensionality is much lower than d.

kNN: Discussion

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.
  - We always preprocess the training set, so in reality training time of kNN is linear.

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.

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.

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. Intro vector space classification
3. Rocchio
4. kNN
5. Linear classifiers
6. > two classes

Linear classifiers

Linear classifiers

- Definition:

Linear classifiers

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

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: $\sum_i w_i x_i > \theta$?

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 θ (the threshold) is a parameter.

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 θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)

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 θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.

Linear 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 θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.

Linear 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 θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides

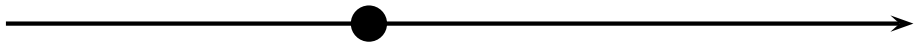
Linear classifiers

- 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 θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides
- Assumption: The classes are linearly separable.

A linear classifier in 1D

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

A linear classifier in 1D

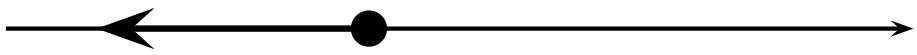


- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1

A linear classifier in 1D

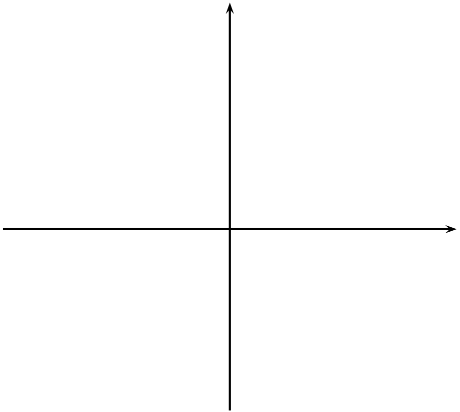
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.

A linear classifier in 1D



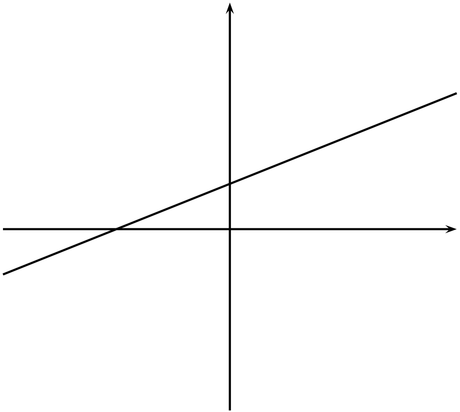
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



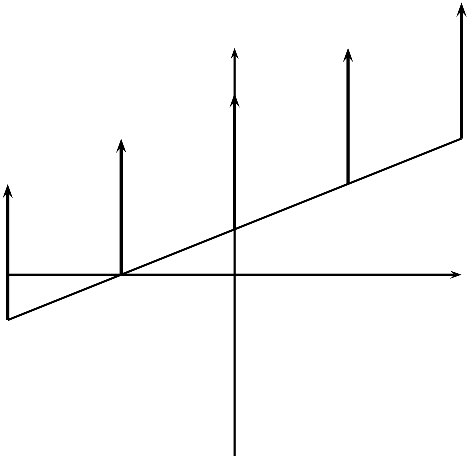
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$

A linear classifier in 2D



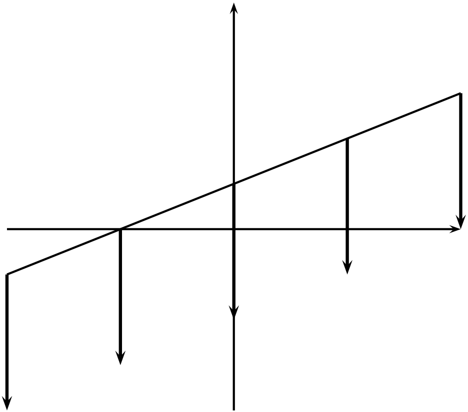
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier

A linear classifier in 2D



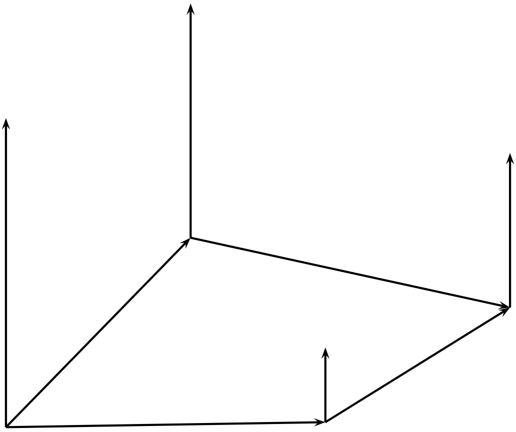
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.

A linear classifier in 2D



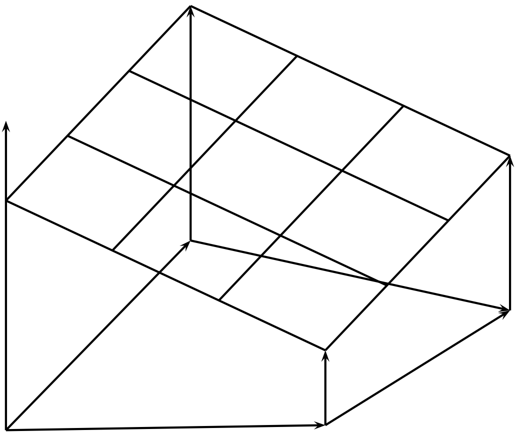
- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



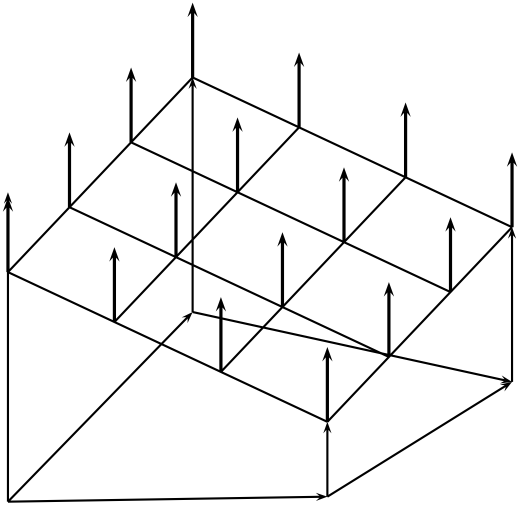
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$

A linear classifier in 3D



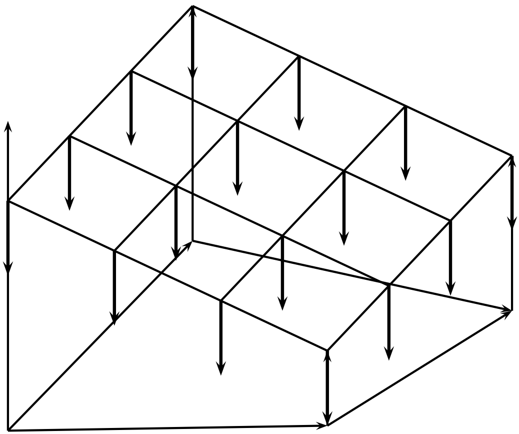
- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1 \ d_2 \ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1 \ d_2 \ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2 \ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\overline{c}$.

Rocchio as a linear classifier

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

Naive Bayes as a linear classifier

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

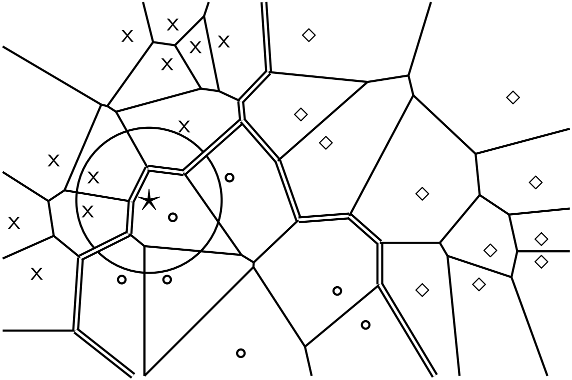
$$\sum_{i=1}^{M} w_i d_i = \theta$$

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, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in d as k did in our original definition of Naive Bayes)

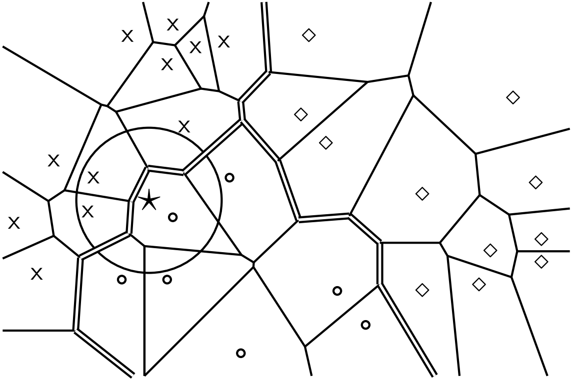
kNN is not a linear classifier

kNN is not a linear classifier

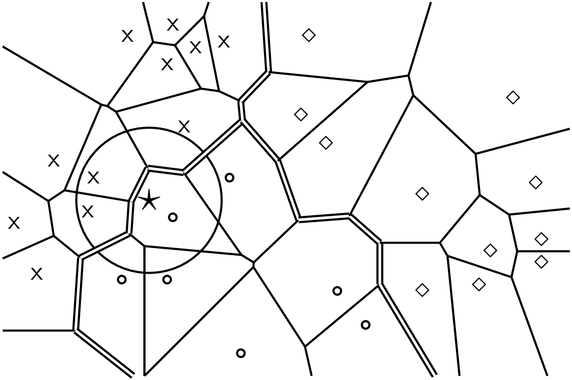
kNN is not a linear classifier



kNN is not a linear classifier

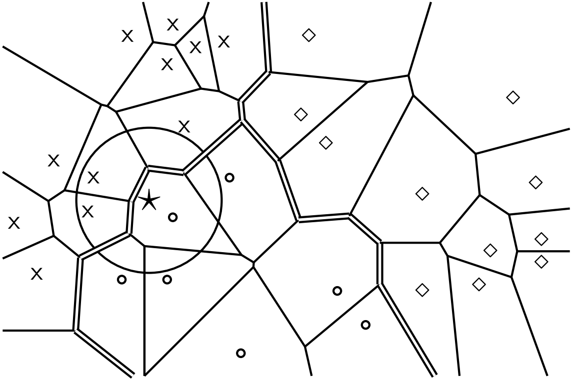


kNN is not a linear classifier



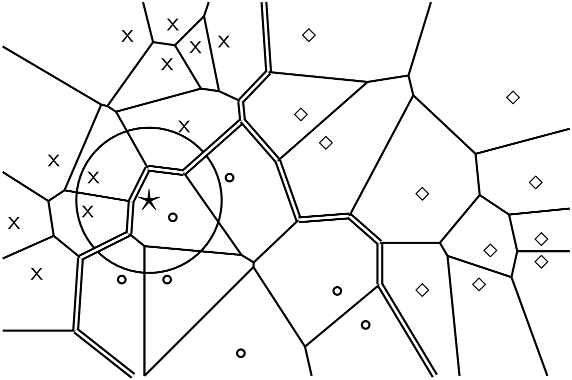
- Classification decision based on majority of *k* nearest neighbors.

kNN is not a linear classifier



- Classification decision based on majority of k nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .

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?

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?

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?

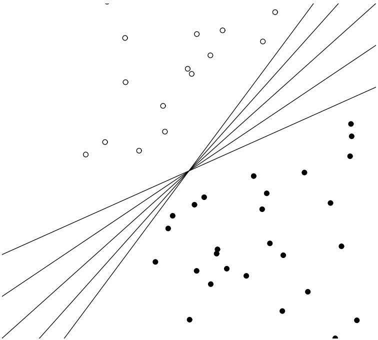
Example of a linear two-class classifier

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?

Which hyperplane?



Learning algorithms for vector space classification

Learning algorithms for vector space classification

- In terms of actual computation, there are two types of learning algorithms.

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.

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.

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

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

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://cislmu.org)

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://cislmu.org)
- The best performing learning algorithms usually require iterative learning.

Perceptron update rule

Perceptron update rule

- Randomly initialize linear separator $\vec{w}$

Perceptron update rule

- Randomly initialize linear separator $\vec{w}$
- Do until convergence:

Perceptron update rule

- Randomly initialize linear separator $\vec{w}$
- Do until convergence:
  - Pick data point $\vec{x}$

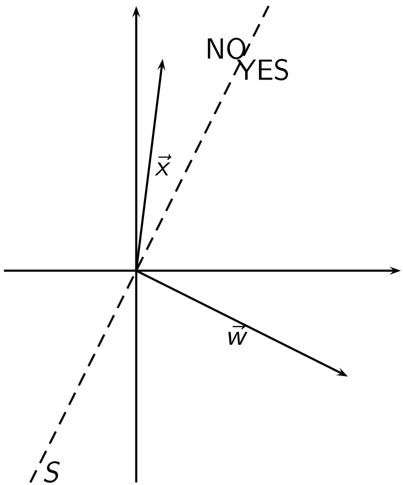
Perceptron update rule

- Randomly initialize linear separator $\vec{w}$
- Do until convergence:
  - Pick data point $\vec{x}$
  - If $\text{sign}(\vec{w}^T\vec{x})$ is correct class (1 or -1): do nothing

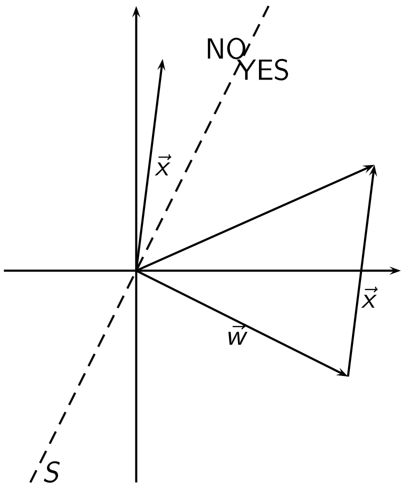
Perceptron update rule

- Randomly initialize linear separator $\vec{w}$
- Do until convergence:
  - Pick data point $\vec{x}$
  - If $\text{sign}(\vec{w}^T\vec{x})$ is correct class (1 or -1): do nothing
  - Otherwise: $\vec{w} = \vec{w} - \text{sign}(\vec{w}^T\vec{x})\vec{x}$

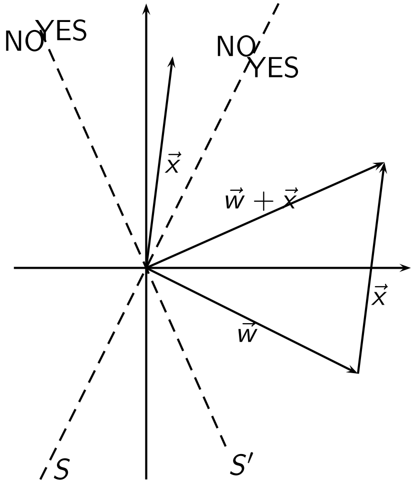
Perceptron (class of $\vec{x}$ is YES)



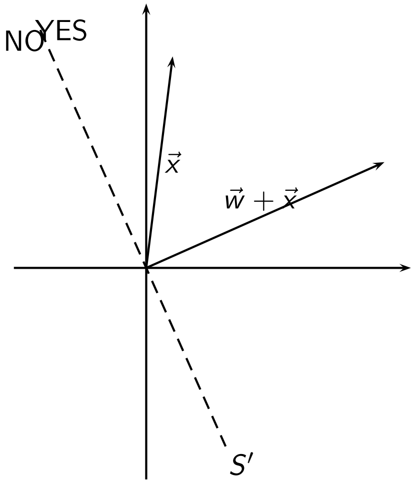
Perceptron (class of $\vec{x}$ is YES)



Perceptron (class of $\vec{x}$ is YES)

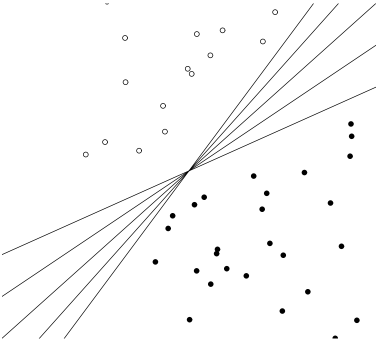


Perceptron (class of $\vec{x}$ is YES)



Which hyperplane?

Which hyperplane?



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

Linear classifiers: Discussion

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

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

Linear classifiers: Discussion

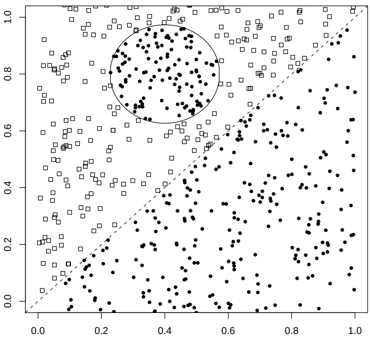
- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane
  - Huge differences in performance on test documents
- Can we get better performance with more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

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.

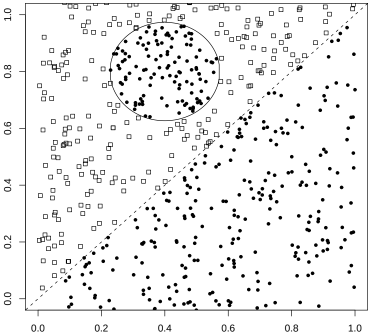
A nonlinear problem

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?

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?

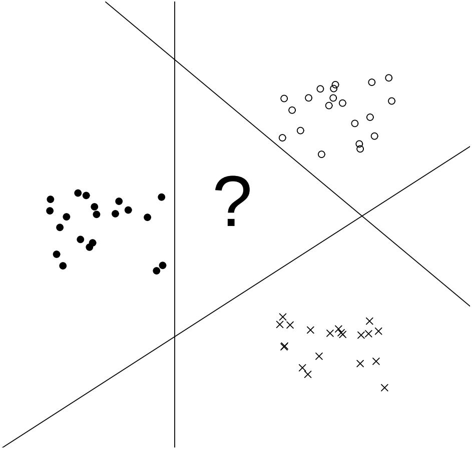
Which classifier do I use for a given TC problem?

- 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?

How to combine hyperplanes for > 2 classes?



One-of problems

One-of problems

- One-of or multiclass classification

One-of problems

- One-of or multiclass classification
  - Classes are mutually exclusive.

One-of problems

- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.

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

One-of classification with linear classifiers

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

One-of classification with linear classifiers

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

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)

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 problems

- Any-of or multilabel classification

Any-of problems

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

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.

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)

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

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

Any-of classification with linear classifiers

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

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

Resources

- Chapter 13 of IIR (feature selection)
- Chapter 14 of IIR
- Resources at `http://cislmu.org`
  - 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)