INF5830 – 2013 FALL
NATURAL LANGUAGE PROCESSING

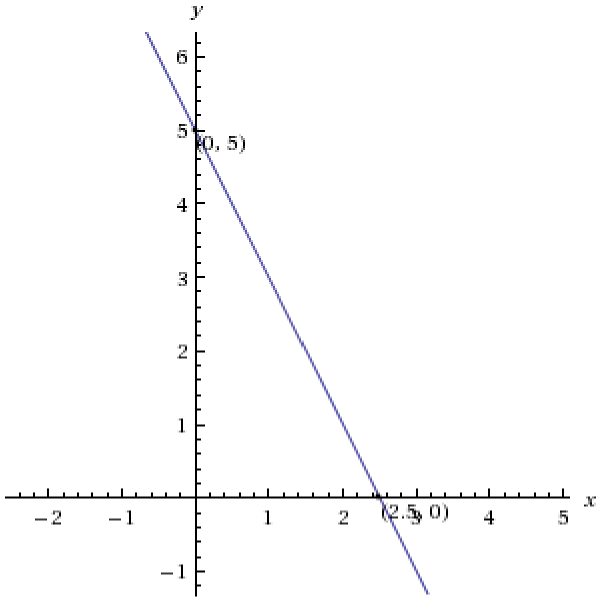
Jan Tore Lønning, Lecture 13, 7.11

Today

- Geometrical background:
  - Lines
  - Planes
  - Vectors
- Vector spaces for
  - Information retrieval
  - Distributional semantics
- Vector spaces and classification
  - Rocchio
  - *K* Nearest Neighbors
  - Linear classifiers

Geometry: lines

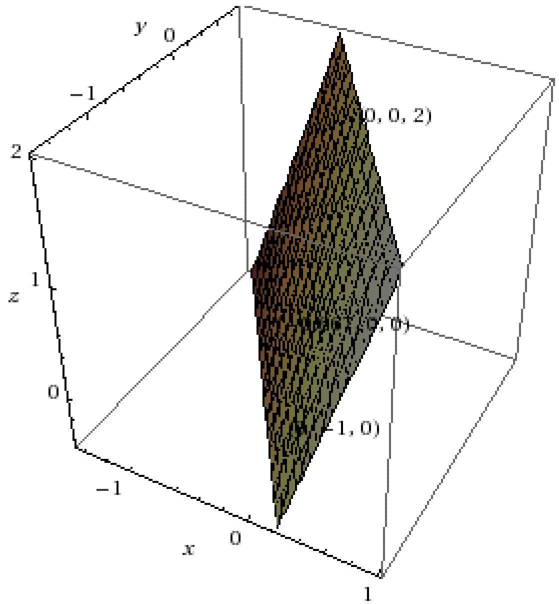
- Descartes
  - (1596-1650)
- Line:
- $ax + by + c = 0$
- If $b \neq 0$:
  - $y = mx + n$
  - $n = -c/b$ is the intercept with the y-axis
  - $m = -a/b$ is the slope
- A point = intersection of two lines

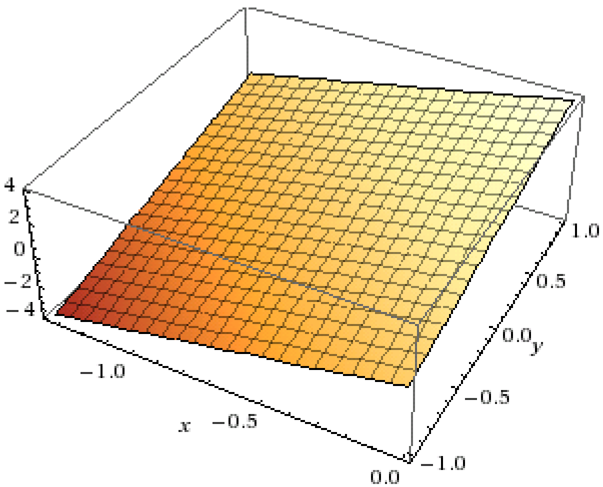


- $y = -2x + 5$
- $2x + y - 5 = 0$

Geometry: planes

- Plane:
- $ax + by + cz + d = 0$
- If $c \neq 0$:
  - $z = mx + ny + n$
- A line is
  the intersection of two planes





- $3x + 2y - z + 2 = 0$
- $z = 3x + 2y + 2$

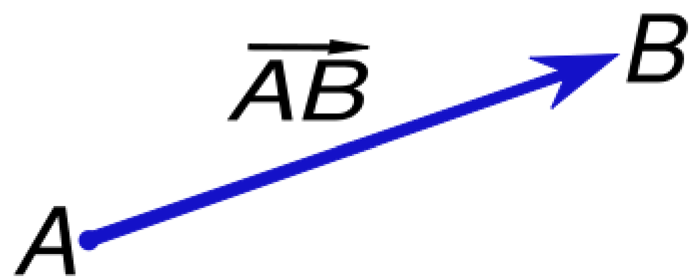
http://www.univie.ac.at/future.media/moe/galerie/geom2/geom2.html#eb

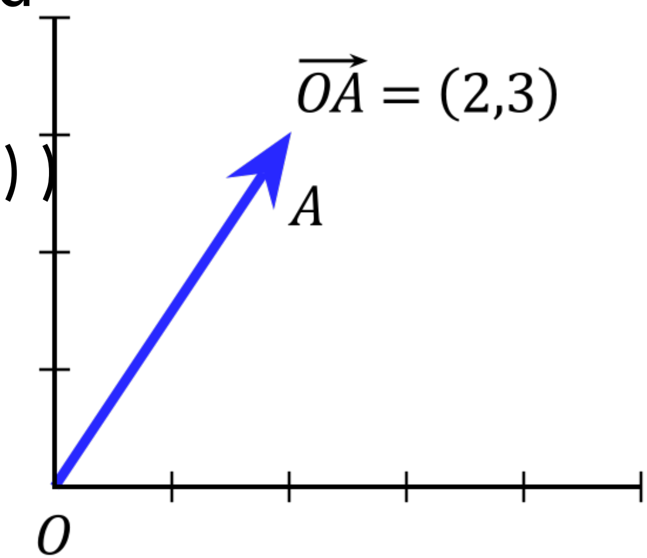
Hyperplanes

- Generalizes to higher dimensions
- In n-dimensional space ($x_1, x_2, \ldots, x_n$):
  - Points satisfying:
- $w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = 0$
  - for any choice of $w_0, w_1, w_2, \ldots w_n$
  - where not all of $w_1, w_2, \ldots w_n = 0$
- is called a hyper-plane
- (In machine learning) the same as the intersection of two hyper-planes in n+1 dimensional space:
  - $w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$
  - $x_0 = 1$

Vectors

- From physics an mathematics
- Here in n-dimensional Euclidean space
- Dual nature:
  - Arrows in plane with start and endpoint
  - Points (with start point in (0,0))





Vector algebra

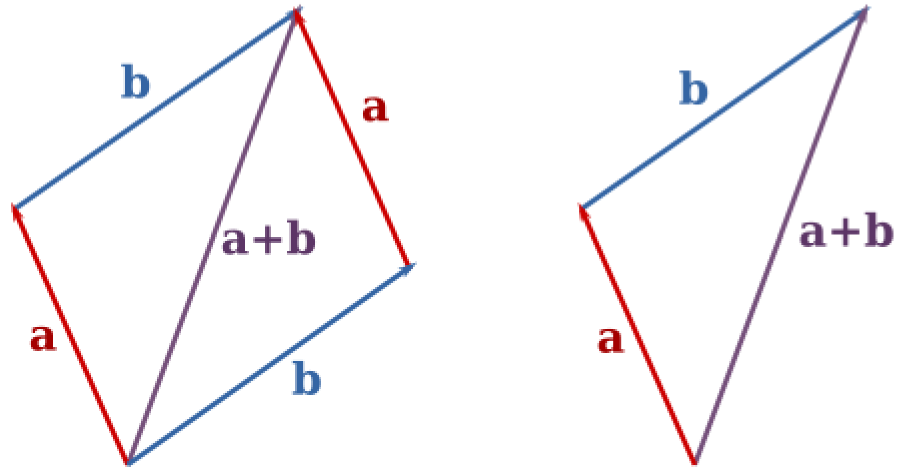
- **a** = $(a_1,a_2,\ldots,a_n)$
- **b** = $(b_1,b_2,\ldots,b_n)$
- **a** + **b** = $(a_1+b_1,\ a_2+b_2,\ \ldots,\ a_n+b_n)$
- Mean: 0.5(**a** + **b**)
- Length

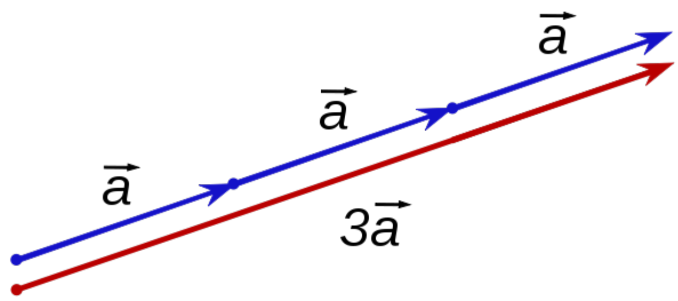
$$\|\mathbf{a}\| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2}$$

- Dot product (inner product):

$$\mathbf{a}\cdot\mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos\theta$$

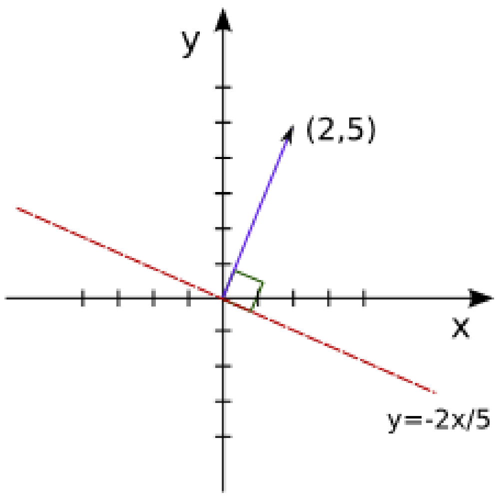
$$\mathbf{a}\cdot\mathbf{b} = a_1b_1 + a_2b_2 + a_3b_3.$$





Normal vector of a line

- $\cos(\pi/2) = 0$
- If P passes through (0,0) there is an **n** = (x_n, y_n) s.t.
- (x,y) is on P iff
  - $(x,y) \bullet (x_n, y_n) = 0$
  - $x \times x_n = - y \times y_n$
  - If $(a,b) \neq (0,0)$ is on P:
    - **n** $= s \times(b, -a)$ for some s

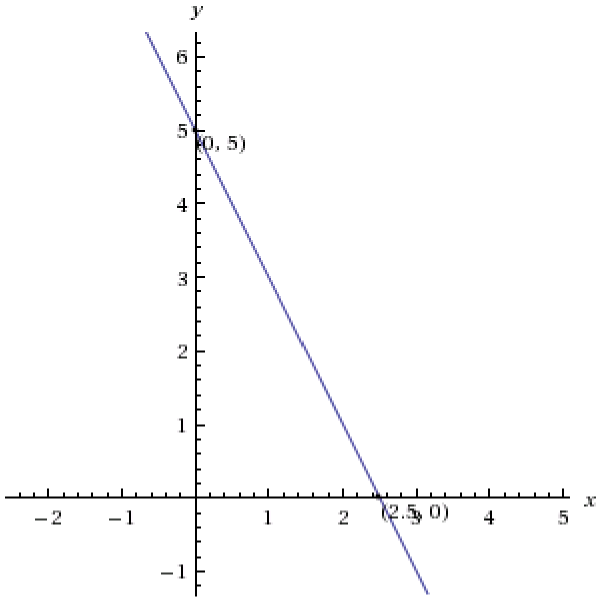


Vector (2,5) is normal to the line y=-2x/5

- Example:
  - $y = -2x/5$
  - $2x + 5y = 0$
  - $(x,y) \bullet (2,5) = 0$

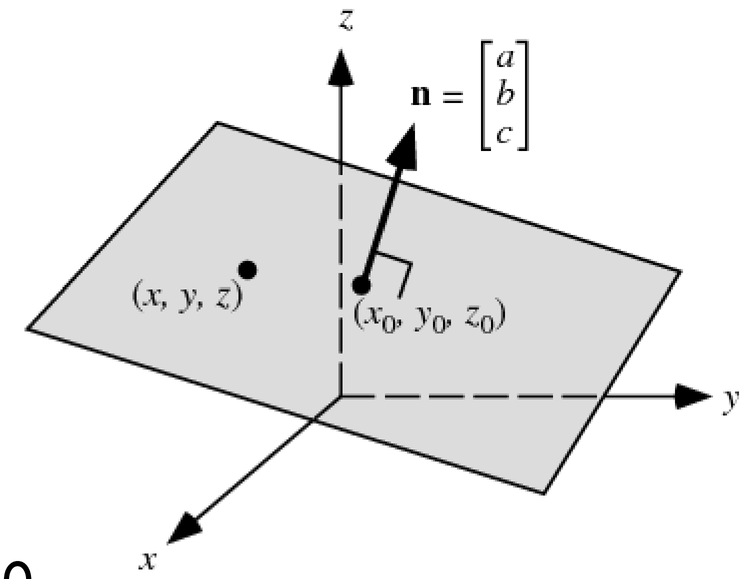
Lines not through (0,0)

- $y = -2x + 5$
- $2x + y - 5 = 0$
- $(x,y) \bullet (2,1) = 5$



Normal vector of a plane

- All points (x,y,z) where
- $((x,y,z)-(x_0,y_0,z_0))\bullet(a,b,c) = 0$
- $(x,y,z) \bullet (a,b,c) = d$
  - $(d = a\, x_0 + b\, y_0 + c\, z_0)$



- Hyperplane
  - $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$
  - $(w_1, w_2, \ldots, w_n) \bullet (x_1, x_2, \ldots x_n) = -w_0$
- Sometimes (n+1 dimensions):
  - $(w_0, w_1, w_2, \ldots, w_n) \bullet (1, x_1, x_2, \ldots x_n) = 0$

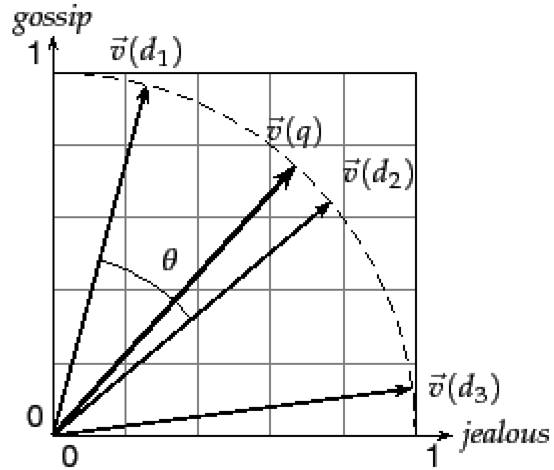
Dot product and cosine

- Dot product:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3.$$

$$\cos(\vec{q}, \vec{d}) = \text{SIM}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$



► **Figure 6.4** Cosine similarity illustrated. $\text{sim}(d_1, d_2) = \cos\theta$.

- Normalized vector (length 1):

$$\hat{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \qquad \|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dot product measures how similar normalized vectors are

Today

- Geometrical background:
  - Lines
  - Planes
  - Vectors
- Vector spaces for
  - Information retrieval
  - Distributional semantics
- Vector spaces and classification
  - Rocchio
  - *K* Nearest Neighbors
  - Linear classifiers

Vector space model

Information retrieval:

- A document is represented by a vector where each entry corresponds to a term
- $tf_{t,d}$ = frequency of t in d
- $tf_{jealous, SaS} = 10$
- $v(d) = (tf_{t1,d}, tf_{t2,d}, \ldots, tf_{tn,d})$
- $V(SaS) = (115, 10, 2)$
- $v(PaP) = (58, 7, 0)$

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

- SaS: *Sense and Sensibility*
- PaP: *Pride and Prejudice*,
- WH: *Wuthering Heights*
- (Manning et al: IIR)

Cosine similarity

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

term	SaS	PaP	WH
affection	0.9961	0.9928	0.8474
jealous	0.0866	0.1198	0.4661
gossip	0.0173	0	0.2542

Normalized

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- SIM(**v**(SaS),**v**(PaP)) = 0.9993
- SIM(**v**(SaS),**v**(WH)) = 0.8889
- SIM(**v**(PaP),**v**(PaP)) = 0.8972

- This may also be used to measure the similarity between a query and a document.

Vector space semantics

A well defined class of objects: O

1. We extract features associated with the objects
2. We turn the features into real valued vectors
3. Apply a similarity measure between the objects

Application:

- Determine which objects are similar and dissimilar
  - For an object o, find similar o'; search
- Classification
  - Given a set of classes S and training data from OxS,
  - construct a classifier which maps an object o to a class *s*
- Cluster together similar objects (flat or hierarchical)

Vector space semantics

1. Features
   - how do we select them
   - reduce their numbers?
2. How do we turn features into vectors
   - Association measures
3. How do we measure the similarities between the features?

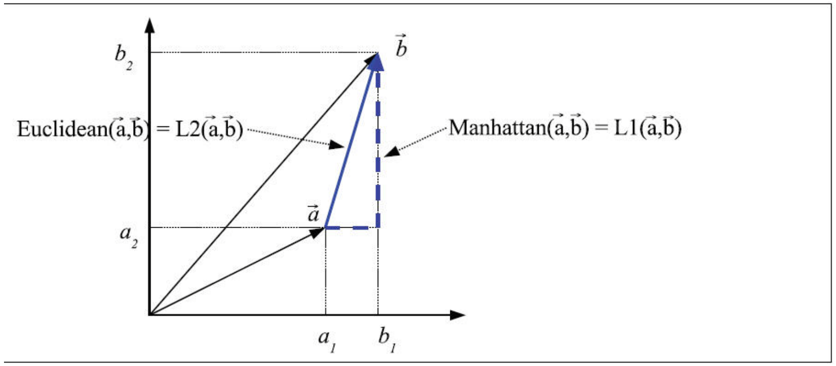
Information retrieval

1. Terms (words) that occur in documents
   - (Reduce: latent semantic indexing)
2. (Td-idf weighing, variants)
3. Cosine measure

Different applications – different optimal choices:
IR, text classification, semantic similarities of words, etc.

3. Similarity measures

- Cosine: $\cos(\vec{a},\vec{b}) = \frac{\vec{a} \bullet \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$
- Euclidean: $\|\vec{a}-\vec{b}\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
- Manhattan: $\|\vec{a}-\vec{b}\|_1 = \sum_i |a_i - b_i|$



- FSNLP, tab 8.7: Several alternatives for binary vectors
- For normalized vectors: Euclid and cos yield same result
  - (Obs: Centroids of normalized vectors are not normalized)

2. Association measures

- IR:
  - Td-idf
- Text classification
- Semantic similarity of words:
  - The various association measures for collocations
  - Some more:
    - Odds Ratio
    - Log Odds Ratio:

$$\log \theta(f,o) = \log\left(\frac{P(f,o)/P(f,\neg o)}{P(\neg f,o)/P(\neg f,\neg o)}\right)$$

Examples

Task	Objects	Features	Vectors	Similarity
IR	Documents	Words in the documents	td-idf weighting	cosinus
Text classification	Text/documents	same		
WSD	Occurrences of a word	Words in the context of the occ.		
Word similarity clustering	Words (lexemes)	Words or other features collected from all the occurrences of the word		

Today

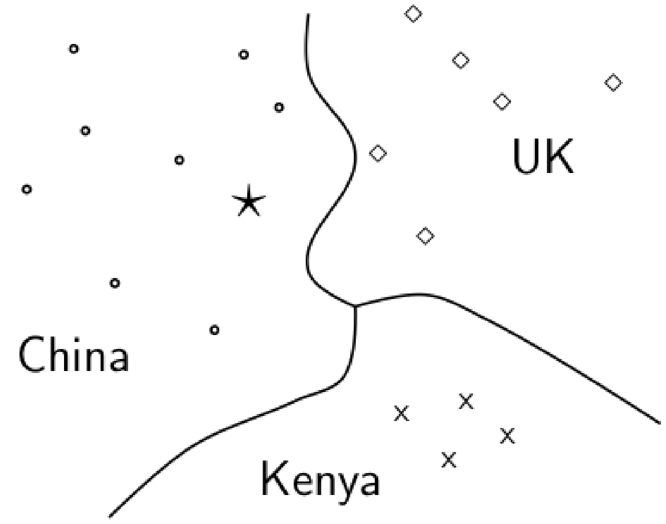
- Geometrical background:
  - Lines
  - Planes
  - Vectors
- Vector spaces for
  - Information retrieval
  - Distributional semantics
- Vector spaces and classification
  - Rocchio
  - *K* Nearest Neighbors
  - Linear classifiers

Vector space classification

- Main idea: Objects that are represented by similar vectors belong to the same class
- Two different algorithms:
  - Rocchio, centroid-based
  - *k* nearest neighbors
- Assumptions:
  - Classes form contiguous regions (not strictly for *k*NN)
  - Classes don't overlap. They may be separated

Task

- Should * be assigned to China, UK or Kenya?
- What are good separators between the classes?



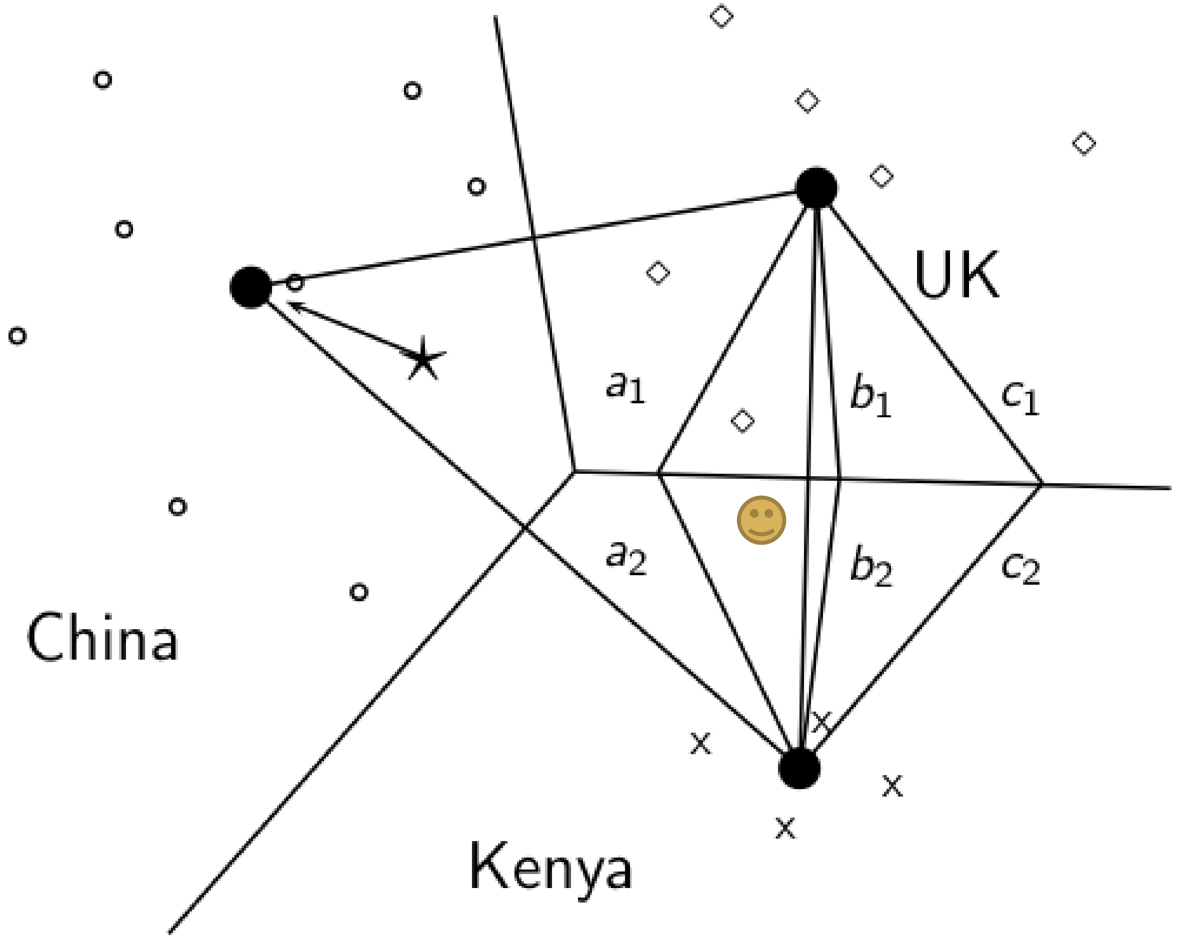
Roochio for classification

- For each class of training vectors, construct a prototype, the centroid (average) of the class

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

- Assign each test document to the class of its closest centroid.

Rocchio



- a1 = a2
- b1 = b2
- c1 = c2

Where does ☺ belong?

Rocchio algorithm

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

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

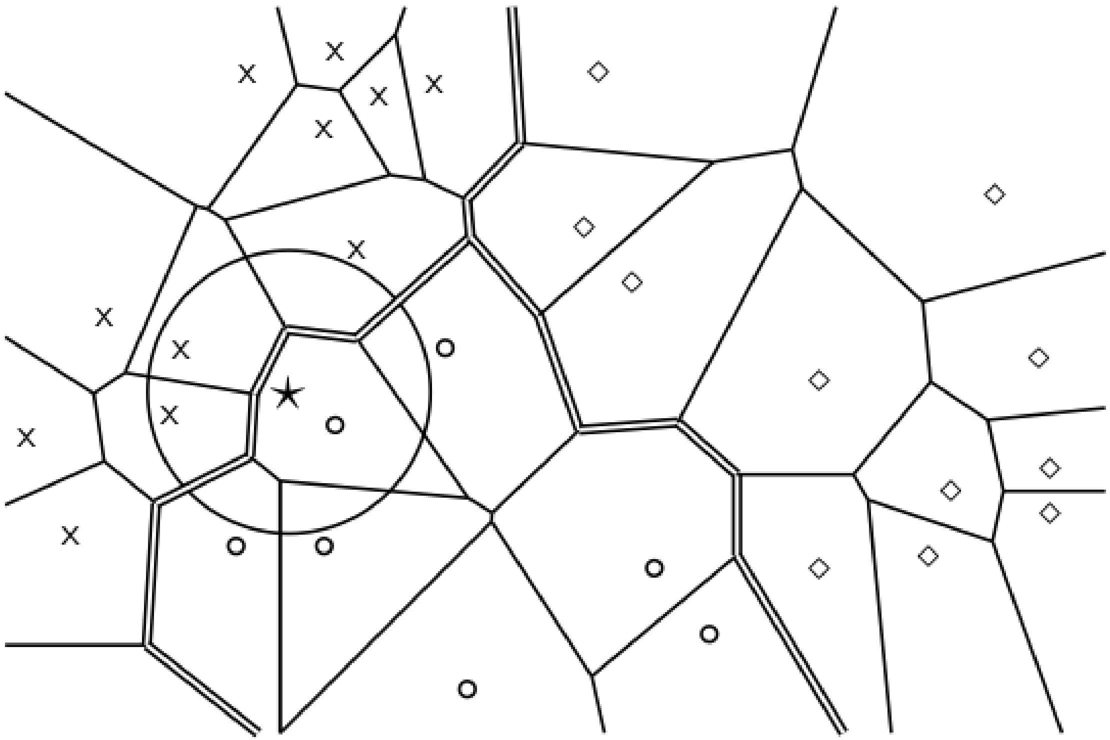
return $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$

- Does not guarantee that classifications are consistent with the training data!
- In many cases, Rocchio performs worse than Naive Bayes
- Because:
  - Assumes all classes have the same radius
  - Assume classes are convex

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 rule for $k > 1$ (kNN):
  - Assign each test document to the majority class of its k nearest neighbors in the training set.

Example



- 1NN classification of *?
- 3NN classification of *?

*k*NN algorithm

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

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

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

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

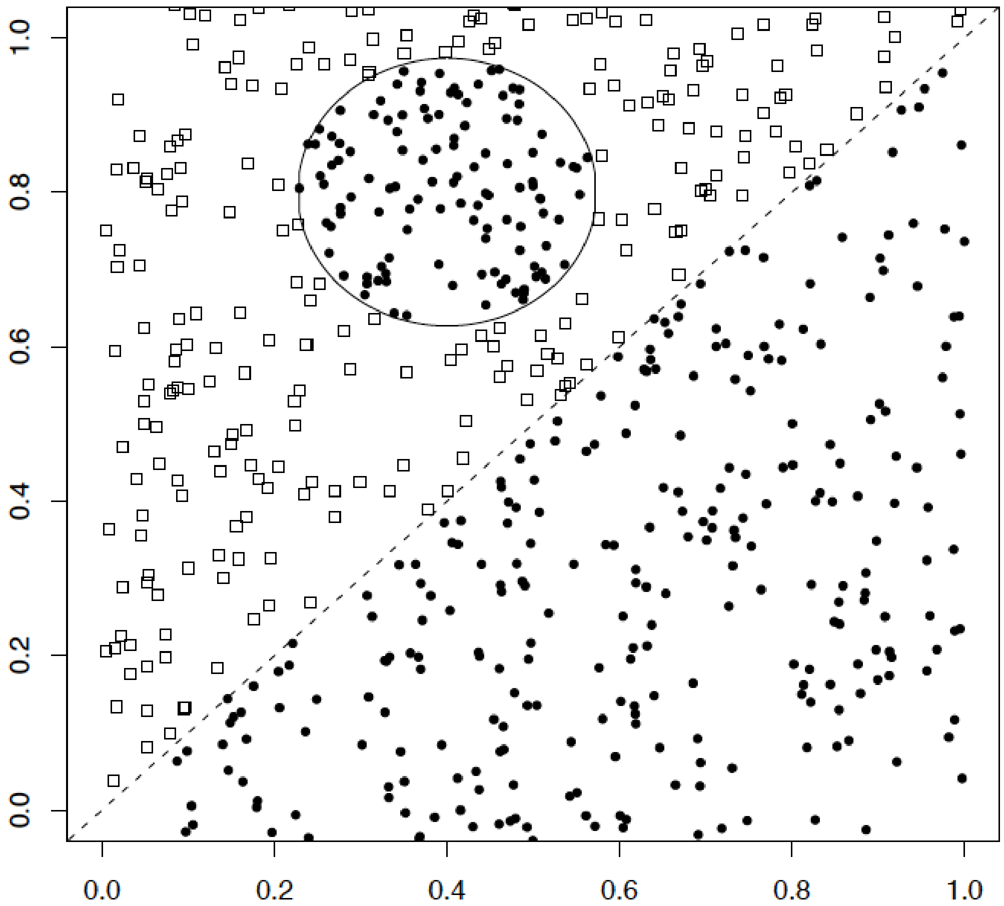
*k*NN properties

- No training is necessary
  - But preprocessing is linear (same as training NB)
- Classification is slow for large training set

- kNN is very accurate for large training sets
- But inaccurate for small training set's

- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning

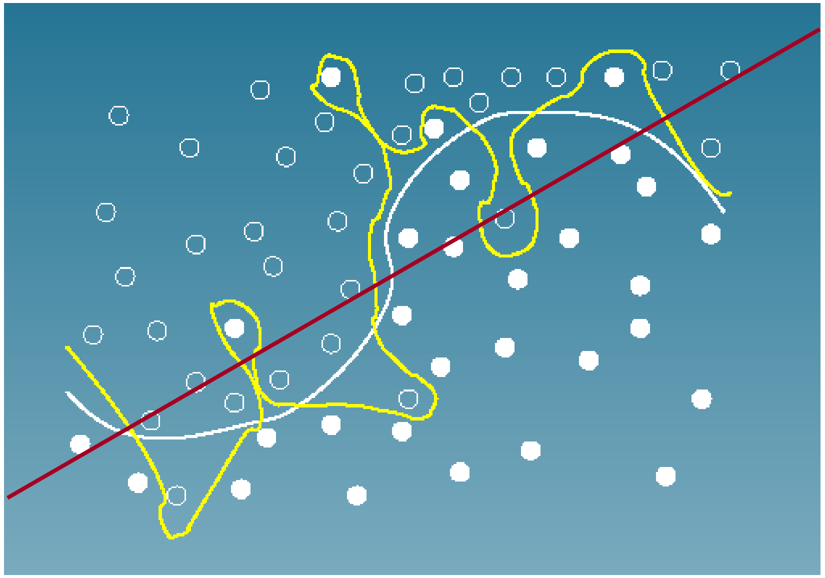
A nonlinear problem



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

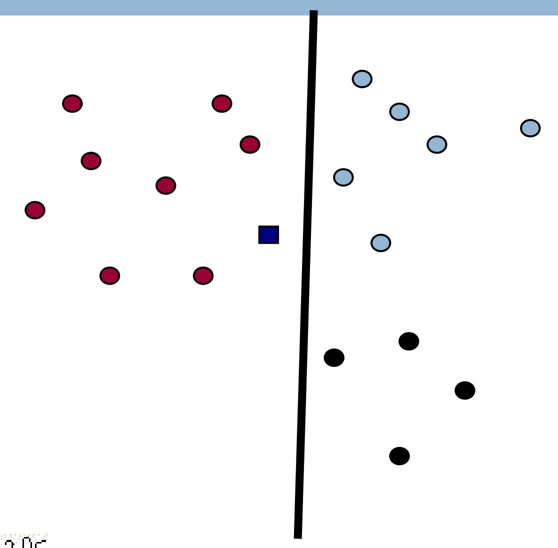
Bias vs. variance

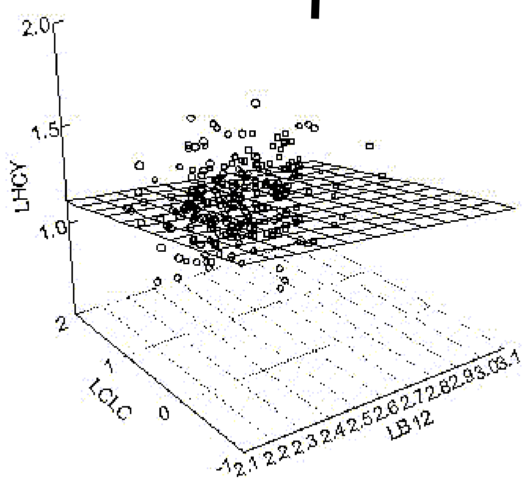
- kNN has high variance and low bias
  - (in particular for small *k*)
- NBB has low variance and high bias
  - (linear classifier)
- Goal is to strike the right balance



Linear classifiers

- Consider binary classifiers:
  - pos – neg
  - Jane Austen – not Jane Austen
  - (Return to more than two classes later)
- Assume linear separability:
  - The two classes as set of points in n-space can be separated by a hyperplane
- In 2 dimensions that is a line:
  - $ax + by > c$ for red points
  - $ax + by < c$ for blue points





Linear classifiers – general case

- The classes can be separated by a hyperplane

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

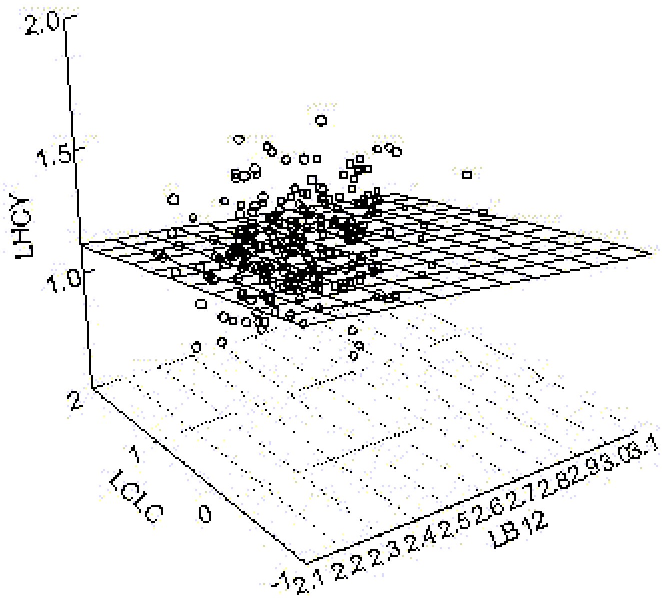
- (equivalently

$$\vec{w} \bullet \vec{x} = \sum_{i=0}^{M} w_i x_i = 0$$

  - taking $w_0 = -\theta$ and $x_0 = 1$)
- The object represented by

$$(x_1, x_2, \ldots, x_n)$$

  - is in C if and only if $\sum_{i=1}^{M} w_i x_i > \theta$
  - And in –C if $\sum_{i=1}^{M} w_i x_i < \theta$
  - Or the other way around: Check >< in each case!



Linear classifiers

- Rocchio
- Naive Bayes
- Logistic regression
- (SVM – with linear kernel)
- Perceptron

- Non-linear:
  - *k*NN

Rocchio is a linear classifier

- The decision is considering the equivalent expressions

$$\cos(\vec{x}, \vec{\mu}(C_1)) > \cos(\vec{x}, \vec{\mu}(C_2))$$

$$\frac{\vec{x} \bullet \vec{\mu}(C_1)}{\left\|\vec{\mu}(C_1)\right\|} > \frac{\vec{x} \bullet \vec{\mu}(C_2)}{\left\|\vec{\mu}(C_2)\right\|}$$

$$\vec{x} \bullet \left(\frac{1}{\left\|\vec{\mu}(C_1)\right\|} \vec{\mu}(C_1) - \frac{1}{\left\|\vec{\mu}(C_2)\right\|} \vec{\mu}(C_2) \right) > 0$$

Also linear with Euclidean dist. as sim. measure

Naive Bayes is a linear classifier

$$\hat{c} = \underset{c \in \{c_1, c_2\}}{\arg\max}\, P(c) \prod_{j=1}^{n} P(f_j \mid c)$$

$$P(c_1) \prod_{j=1}^{n} P(f_j \mid c_1) > P(c_2) \prod_{j=1}^{n} P(f_j \mid c_2)$$

$$\frac{P(c_1) \prod_{j=1}^{n} P(f_j \mid c_1)}{P(c_2) \prod_{j=1}^{n} P(f_j \mid c_2)} > 1$$

$$\frac{P(c_1)}{P(c_2)} \prod_{j=1}^{n} \frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} > 1$$

$$\log\left(\frac{P(c_1)}{P(c_2)} \prod_{j=1}^{n} \frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right) > 0$$

$$\log\left(\frac{P(c_1)}{P(c_2)} \right) + \sum_{j=1}^{n} \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right) > 0$$

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

$$w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} \right)$$

$$\theta = -w_0 = -\log\left(\frac{P(c_1)}{P(c_2)} \right)$$