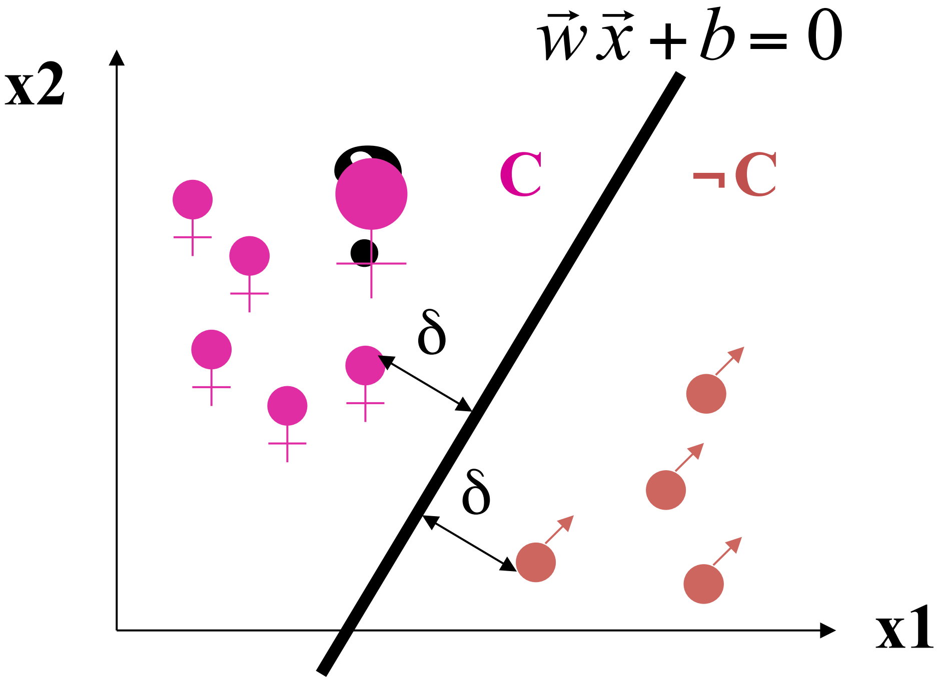
Separation of data points

Example: Outdoor activity

Day	Outlook	Temp	Wind	Workday	Active ?
D1	Sun	8	25	Yes	No
D2	Sun	3	27	Yes	No
D3	Overcast	25	4	Yes	Yes
D4	Rain	22	2	Yes	Yes
D5	Rain	18	6	No	Yes
D6	Rain	9	12	No	No
D7	Overcast	14	3	No	Yes
D8	Sun	6	17	Yes	No
D9	Sun	19	4	No	Yes
D10	Rain	23	6	No	Yes
D11	Sun	25	8	No	Yes
D12	Overcast	24	7	Yes	Yes
D13	Overcast	16	7	No	Yes
D14	Rain	7	19	Yes	No

Discriminative Classifiers: Support Vector Machines (SVM), Binary Classification



n training vectors
(x_1, ..., x_m, C)
with C = +1 or -1

large-margin separating hyperplane minimizes risk of classification error

Determine ***hyperplane*** $\vec{w}\vec{x}+b=0$ that optimally ***separates*** the training vectors in C from those not in C, such that the (Euclidean) distance δ of the (positive and negative) training samples closest to the hyperplane is maximized. (Vectors with distance δ are called ***support vectors***.)

Classify new test vector $\vec{y}$ into C if: $(\vec{w}\vec{y}+b) = \sum_{i=1}^{m} w_i y_i + b > 0$

Computation of the Optimal Hyperplane

Find $\vec{w} \in R^m$ and b $\in$R such that

1. $\delta \in$R is maximal and
2. $C_i \frac{1}{\|\vec{w}\|}(\vec{w}\vec{x}_i + b) \geq \delta$ for all i=1, ..., n

This is (w.l.o.g. with the choice $\|\vec{w}\| = 1/\delta$) equivalent to (V. Vapnik: Statistical Learning Theory, 1998):

Find $\alpha_1, \ldots, \alpha_n \in R_0^+$ such that

1. $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} C_i C_j \alpha_i \alpha_j (\vec{x}_i \vec{x}_j)$ is minimal
2. and $\sum_{i=1}^{n} C_i \alpha_i = 0$

(Quadratic programming problem)

Optimal vector $\vec{w}$ is a linear combination (where $\alpha_i > 0$ only for support vectors)

$$\vec{w} = \sum_{i=1}^{n} \alpha_i C_i \vec{x}_i$$

b is derived from any support vector $\vec{x}_j$ by:

$$b = C_j - \vec{w}\vec{x}_j$$

SVM Engineering

\+ Very efficient implementations available
(e.g., SVM-Light at http://svmlight.joachims.org/):
with training time empirically found to be
≈ quadratic in # training docs (and linear in # features)

\+ SVMs can and should usually consider all possible features
(no point for feature selection unless #features intractable)

\+ multi-class classification mapped to multiple binary SVMs:
one-vs.-all or combinatorial design of subset-vs.-complement

\- Choice of kernel difficult
and highly dependent on data and application

Classification: Practical Issues

Gee, I'm building a text classifier for real, now!

What should I do?

How much training data do you have?

- None
- Very little
- Quite a lot
- A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - Construe: 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Iterative labeling, co-training, …

- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you?? How can I get feedback? (implicitely or explicitely)

A reasonable amount of data?

Perfect!

We can use all our clever classifiers

Roll out the SVM!

But if you are using an SVM/NB etc., you should probably be prepared with the "hybrid" solution where there is a boolean overlay

- Or else to use user-interpretable Boolean-like models like decision trees
- Users like to hack, and management likes to be able to implement quick fixes immediately

A huge amount of data?

This is great in theory for doing accurate classification…

But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical

Naïve Bayes can come back into its own again!

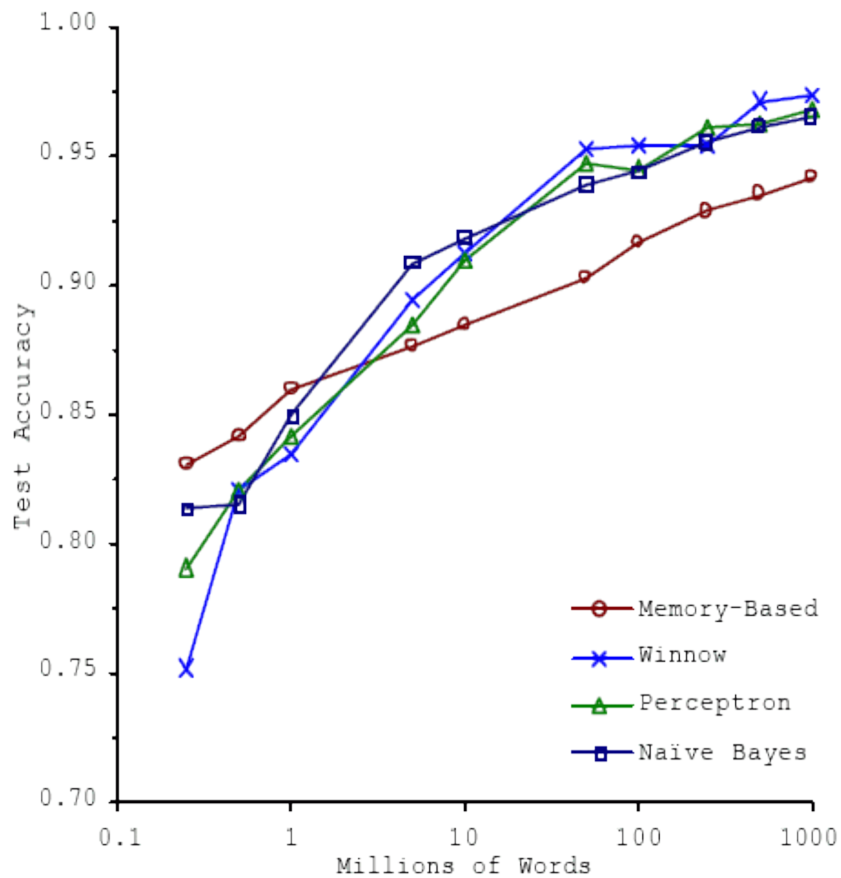
- Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)

A huge amount of data?

With enough data the choice of classifier may not matter much, and the best choice may be unclear

- Data: Brill and Banko on context-sensitive spelling correction

But the fact that you have to keep doubling your data to improve performance is a bit unpleasant



How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
      - Mileage fairly unclear
    - May need a hybrid automatic/manual solution

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn't be.
  - E.g., part numbers, chemical formulas

Does putting in "hacks" help?

You bet!

You can get a lot of value by differentially weighting contributions from different document zones:

- Upweighting title words helps (Cohen & Singer 1996)
  - Doubling the weighting on the title words is a good rule of thumb
- Upweighting the first sentence of each paragraph helps (Murata, 1999)
- Upweighting sentences that contain title words helps (Ko *et al,* 2002)

Two techniques for zones

- Have a completely separate set of features/parameters for different zones like the title
- Use the same features (pooling/tying their parameters) across zones, but upweight the contribution of different zones

- Commonly the second method is more successful: it costs you nothing in terms of sparsifying the data, but can give a very useful performance boost
  - Which is best is a contingent fact about the data

Automatic clustering

<u>given:</u>

n ***m-dimensional data records*** dj $\in$D $\subseteq$ dom(A1) × ... × dom(Am)
with attributes Ai (e.g. term frequency vectors $\subseteq N_0 \times \ldots \times N_0$)
or n ***data points*** with pair-wise ***distances (similarities)*** in a ***metric space***

<u>wanted:</u>

k **clusters** c1, ..., ck and an assignment D → {c1, ..., ck} such that the

average **intra-cluster similarity** $\frac{1}{k}\sum_k \left(\frac{1}{|c_k|} \sum_{\vec{d} \in c_k} sim(\vec{d}, \vec{c}_k) \right)$

is high and

the average **inter-cluster similarity** $\frac{1}{k(k-1)} \sum_{\substack{i,j \\ i \neq j}} sim(\vec{c}_i, \vec{c}_j)$

is low,

where the ***centroid*** $\vec{c}_k$ of ck is: $\vec{c}_k = \frac{1}{|c_k|} \sum_{\vec{d} \in c_k} \vec{d}$

Hierarchical vs. Flat Clustering

Hierarchical Clustering:

- detailed and insightful
- hierarchy built
  in natural manner
  from fairly simple algorithms
- relatively expensive
- no prevalent algorithm

Flat Clustering:

- data overview & coarse analysis
- level of detail depends
  on the choice of the
  number of clusters
- relatively efficient
- K-Means and EM are simple
  standard algorithms

Hierarchical Clustering: Agglomerative Bottom-up Clustering (HAC)

Principle:

- start with each d_i forming its own singleton cluster c_i
- in each iteration combine the most similar clusters c_i, c_j into a new, single cluster

for i:=1 to n do c_i := $\{d_i\}$ od;
C := $\{c_1, ..., c_n\}$; /* set of clusters */
while |C| > 1 do
  determine $c_i, c_j \in C$ with maximal inter-cluster similarity;
  $C := C - \{c_i, c_j\} \cup \{c_i \cup c_j\}$;
od;

Divisive Top-down Clustering

Principle:

- start with a single cluster that contains all data records
- in each iteration identify the least „coherent“ cluster and divide it into two new clusters

$c_1 := \{d_1, ..., d_n\};$
$C := \{c_1\};$ /* set of clusters */
while there is a cluster $c_j \in C$ with $|c_j|>1$ do
  determine c_i with the lowest intra-cluster similarity;
  partition c_i into c_{i1} and c_{i2} (i.e. $c_i = c_{i1} \cup c_{i2}$ and $c_{i1} \cap c_{i2} = \emptyset$)
  such that the inter-cluster similarity between c_{i1} and c_{i2}
  is minimized;
od;

For partitioning a cluster one can use another clustering method (e.g. a bottom-up method)

Alternative Similarity Metrics for Clusters

given: similarity on data records - sim: $D \times D \rightarrow R$ oder [0,1]
define: similarity between clusters – sim: $2^D \times 2^D \rightarrow R$ or [0,1]

Alternatives:

- **Centroid method**: sim (c,c‘) = sim(d, d‘) with centroid d of c and centroid d‘ of c‘
- **Single-Link method**: sim(c,c‘) = sim(d, d‘) with d ∈c, d‘∈c‘, such that d and d‘ have the highest similarity
- **Complete-Link method**: sim(c,c‘) = sim(d, d‘) with d ∈c, d‘∈c‘, such that d and d‘ have the lowest similarity
- **Group-Average method**: $$\frac{1}{|c| \cdot |c'|} \sum_{d \in c, d' \in c'} sim(d, d')$$

For hierarchical clustering the following axiom must hold:
max {sim(c,c‘), sim(c,c“)} ≥ sim(c, c‘∪ c“) for all c, c‘, c“ ∈ 2^D

Cluster Quality Measures (1)

With regard to **ground truth:**
known class labels $L_1, \ldots, L_g$ for data points $d_1, \ldots, d_n$:
$L(d_i) = L_j \in \{L_1, \ldots, L_g\}$

With cluster assignment $\Gamma(d_1), \ldots, \Gamma(d_n) \in c_1, \ldots, c_k$
cluster c_j has **purity** $\max_{\nu=1..g} |\{d \in c_j \mid L(d) = L_\nu\}| / |c_j|$

Complete clustering has purity $\sum_{j=1..k} purity(c_j)/k$

<u>Alternatives:</u>
- **Entropy** within cluster $\sum_{\nu=1..g} \frac{|c_j \cap L_\nu|}{|c_j|} \log_2 \frac{|c_j|}{|c_j \cap L_\nu|}$

- **MI** between cluster and classes

$$\sum_{c \in \{c_j, \bar{c}_j\}, L \in \{L_1, \ldots, L_g\}} \frac{|c \cap L|/n}{|c| \cdot |L|/n} \log_2 \frac{|c| \cdot |L|/n}{|c \cap L|/n}$$

Cluster Quality Measures (2)

Without any ground truth:

ratio of intra-cluster to inter-cluster similarities

$$\frac{1}{k}\sum_{k}\left(\frac{1}{|c_k|}\sum_{\vec{d}\in c_k} sim(\vec{d},\vec{c}_k)\right) \Bigg/ \left(\frac{1}{k(k-1)}\sum_{\substack{i,j\\ i\neq j}} sim(\vec{c}_i,\vec{c}_j)\right)$$

or other ***cluster validity measures*** of this kind
(e.g. considering variance of intra- and inter-cluster distances)

Flat Clustering: Simple Single-Pass Method

<u>given:</u> data records d1, ..., dn

<u>wanted:</u> (up to) k clusters C:={c1, ..., ck}

C := {{d1}}; /* random choice for the first cluster */
for i:=2 to n do
   determine cluster cj $\in$ C with the largest value of
   sim(di, cj) (e.g. sim(di, $\vec{c}_j$) with centroid $\vec{c}_j$);
    if sim(di, cj) ≥ threshold
    then assign di to cluster cj
    else if |C| < k
          then C := C $\cup$ {{di}}; /* create new cluster */
          else assign di to cluster cj
          fi
    fi
od

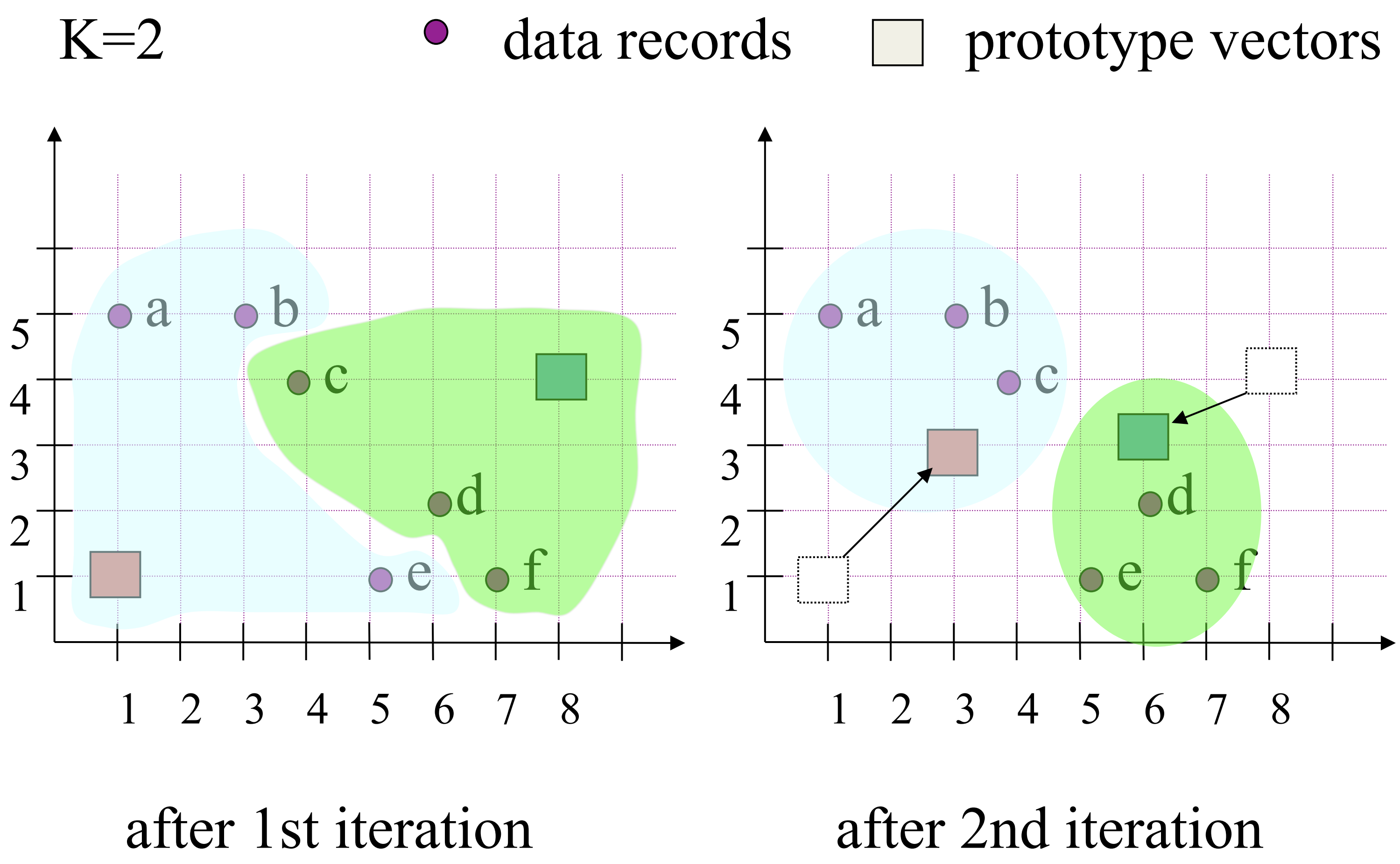
K-Means Method for Flat Clustering (1)

Idea:

- determine **k prototype vectors**, one for each cluster
- **assign each data record to the most similar prototype vector** and compute new prototype vector (e.g. by averaging over the vectors assigned to a prototype)
- **iterate** until clusters are sufficiently stable

randomly choose k prototype vectors $\vec{c}_1,\ldots,\vec{c}_k$
while not yet sufficiently stable do
  for i:=1 to n do
    assign di to cluster cj for which $sim(\vec{d}_i,\vec{c}_j)$ is minimal
  od;
  for j:=1 to k do $\vec{c}_j := \frac{1}{|c_j|}\sum_{\vec{d}\in c_j}\vec{d}$ od;
od;

Example for K-Means Clustering



K-Means Method for Flat Clustering (2)

- run-time is O(n) (assuming constant number of iterations)
- a suitable number of clusters, K, can be determined experimentally or based on the MDL principle
- the initial prototype vectors could be chosen by using another – very efficient – clustering method
  (e.g. bottom-up clustering on random sample of the data records).
- for sim any arbitrary metric can be used