Separation of data points
Example: Outdoor activity

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Wind</th>
<th>Workday</th>
<th>Active ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sun</td>
<td>8</td>
<td>25</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sun</td>
<td>3</td>
<td>27</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>25</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>22</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>18</td>
<td>6</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>9</td>
<td>12</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>14</td>
<td>3</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sun</td>
<td>6</td>
<td>17</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sun</td>
<td>19</td>
<td>4</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>23</td>
<td>6</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sun</td>
<td>25</td>
<td>8</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>24</td>
<td>7</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>16</td>
<td>7</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>7</td>
<td>19</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Discriminative Classifiers: Support Vector Machines (SVM), Binary Classification

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 $\delta$ of the (positive and negative) training samples closest to the hyperplane is maximized. (Vectors with distance $\delta$ 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 $\mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$ such that

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

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

Find $\alpha_1, \ldots, \alpha_n \in \mathbb{R}_0^+$ such that

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

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

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

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

$$b = C_j - \mathbf{w} \cdot \mathbf{x}_j$$
+ Very efficient implementations available (e.g., SVM-Light at http://svmlight.joachims.org/):
  with training time empirically found to be
  \( \approx \) 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? (implicitly 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
Clustering: Classification based on Unsupervised Learning

given:
n \textbf{m-dimensional data records} \ d_j \in D \subseteq \text{dom}(A_1) \times \ldots \times \text{dom}(A_m)
with attributes \ A_i \ (e.g. \ \text{term frequency vectors} \ \subseteq \mathbb{N}_0 \times \ldots \times \mathbb{N}_0)
or \ n \ \textbf{data points} with pair-wise \ \textit{distances (similarities)} \ \text{in a metric space}

wanted:
k \ \textbf{clusters} \ c_1, \ldots, c_k \ \text{and an assignment} \ \text{D} \rightarrow \{c_1, \ldots, c_k\} \ \text{such that the}
average \ \textbf{intra-cluster similarity} \ \frac{1}{k} \sum_k \left( \frac{1}{|c_k|} \sum_{\tilde{d} \in c_k} \text{sim}(\tilde{d}, \bar{c}_k) \right)
is high and
the average \ \textbf{inter-cluster similarity} \ \frac{1}{k(k-1)} \sum_{i,j \neq j} \text{sim}(\bar{c}_i, \bar{c}_j)
is low,
where the \ \textbf{centroid} \ \bar{c}_k \ \text{of} \ \text{c}_k \ \text{is:}
\bar{c}_k = \frac{1}{|c_k|} \sum_{\tilde{d} \in c_k} \tilde{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)
Alternatives:

- **Centroid method**:\[ \text{sim} (c, c') = \text{sim}(d, d') \text{ with centroid } d \text{ of } c \text{ and centroid } d' \text{ of } c' \]

- **Single-Link method**:\[ \text{sim}(c, c') = \text{sim}(d, d') \text{ with } d \in c, d' \in c', \text{ such that } d \text{ and } d' \text{ have the highest similarity} \]

- **Complete-Link method**:\[ \text{sim}(c, c') = \text{sim}(d, d') \text{ with } d \in c, d' \in c', \text{ such that } d \text{ and } d' \text{ have the lowest similarity} \]

- **Group-Average method**:\[ 1 \left\| c \cdot c' \right\| \sum_{d \in c, d' \in c'} \text{sim}(d, d') \]

For hierarchical clustering the following axiom must hold:
\[ \max \{ \text{sim}(c, c'), \text{sim}(c, c'') \} \geq \text{sim}(c, c' \cup c'') \text{ for all } c, c', c'' \in 2^D \]
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**

$max_{\nu=1..g} \left\{ d \in c_j \mid L(d) = L_\nu \right\} / \mid c_j \mid$

Complete clustering has purity

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

**Alternatives:**

- **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:

\[
\frac{1}{k} \sum_k \left( \frac{1}{|c_k|} \sum_{d \in c_k} \text{sim}(\tilde{d}, \tilde{c}_k) \right) / \left( \frac{1}{k(k-1)} \sum_{i,j \neq j} \text{sim}(\tilde{c}_i, \tilde{c}_j) \right)
\]

or other \textit{cluster validity measures} of this kind
(e.g. considering variance of intra- and inter-cluster distances)
Flat Clustering: Simple Single-Pass Method

given: data records \( d_1, \ldots, d_n \)
wanted: (up to) \( k \) clusters \( C = \{c_1, \ldots, c_k\} \)

\[ C := \{d_1\}; \quad /* \text{random choice for the first cluster} */ \]

for \( i := 2 \) to \( n \) do

\[ \text{determine cluster } c_j \in C \text{ with the largest value of} \]
\[ \text{sim}(d_i, c_j) \text{ (e.g. sim}(d_i, \bar{c}_j) \text{ with centroid } \bar{c}_j); \]

\[ \text{if } \text{sim}(d_i, c_j) \geq \text{threshold} \]
\[ \text{then assign } d_i \text{ to cluster } c_j \]
else if \( |C| < k \)

\[ \text{then } C := C \cup \{d_i\}; \quad /* \text{create new cluster} */ \]
else assign \( d_i \) to cluster \( c_j \)
\fi
\fi

\od
K-Means Method for Flat Clustering (1)

Idea:
• determine \textbf{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)
• \textbf{iterate} until clusters are sufficiently stable

randomly choose \( k \) prototype vectors \( \bar{c}_1, \ldots, \bar{c}_k \)
while not yet sufficiently stable do
  for \( i := 1 \) to \( n \) do
    assign \( d_i \) to cluster \( c_j \) for which \( \text{sim}(\bar{d}_i, \bar{c}_j) \) is minimal
  od;
  for \( j := 1 \) to \( k \) do
    \( \bar{c}_j := \frac{1}{|c_j|} \sum_{\bar{d} \in c_j} \bar{d} \)
  od;
Example for K-Means Clustering

K=2

after 1st iteration

after 2nd iteration
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