Classification Algorithms

UCSB 293S, 2017. T. Yang
Some of slides based on R. Mooney (UT Austin)
Table of Content

• Problem Definition
• Rocchio
• K-nearest neighbor (case based)
• Bayesian algorithm
• Decision trees
• SVM
Classification

• Given:
  – A description of an instance, \( x \)
  – A fixed set of categories (classes): \( C=\{c_1, c_2, \ldots c_n\} \)
  – Training examples

• Determine:
  – The category of \( x \): \( h(x) \in C \), where \( h(x) \) is a classification function

• A training example is an instance \( x \), paired with its correct category \( c(x) \): \( <x, c(x)> \)
Sample Learning Problem

- Instance space: <size, color, shape>
  - size ∈ \{small, medium, large\}
  - color ∈ \{red, blue, green\}
  - shape ∈ \{square, circle, triangle\}
- \(C = \{\text{positive, negative}\}\)
- \(D:\)

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>
General Learning Issues

• Many hypotheses are usually consistent with the training data.

• Bias
  – Any criteria other than consistency with the training data that is used to select a hypothesis.

• Classification accuracy (% of instances classified correctly).
  – Measured on independent test data.

• Training time (efficiency of training algorithm).

• Testing time (efficiency of subsequent classification).
Text Categorization/Classification

- Assigning documents to a fixed set of categories.
- Applications:
  - Web pages
    - Recommending/ranking
    - Category classification
  - Newsgroup Messages
    - Recommending
    - Spam filtering
  - News articles
    - Personalized newspaper
  - Email messages
    - Routing
    - Prioritizing
    - Folderizing
    - Spam filtering
Learning for Classification

• Manual development of text classification functions is difficult.

• Learning Algorithms:
  – Bayesian (naïve)
  – Neural network
  – Rocchio
  – Rule based (Ripper)
  – Nearest Neighbor (case based)
  – Support Vector Machines (SVM)
  – Decision trees
  – Boosting algorithms
Illustration of Rocchio method
Rocchio Algorithm

Assume the set of categories is \( \{c_1, c_2, \ldots c_n\} \)

**Training:**
Each doc vector is the frequency normalized TF/IDF term vector.
For \( i \) from 1 to \( n \)
   Sum all the document vectors in \( c_i \) to get prototype vector \( p_i \)

**Testing:** Given document \( x \)
   Compute the cosine similarity of \( x \) with each prototype vector.
   Select one with the highest similarity value and return its category
Rocchio Anomaly

- Prototype models have problems with polymorphic (disjunctive) categories.
Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in $D$.
- Testing instance $x$:
  - Compute similarity between $x$ and all examples in $D$.
  - Assign $x$ the category of the most similar example in $D$.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based
  - Memory-based
  - Lazy learning
K Nearest-Neighbor

• Using only the closest example to determine categorization is subject to errors due to:
  – A single atypical example.
  – Noise (i.e. error) in the category label of a single training example.

• More robust alternative is to find the $k$ most-similar examples and return the majority category of these $k$ examples.

• Value of $k$ is typically odd to avoid ties, 3 and 5 are most common.
Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous $m$-dimensional instance space is *Euclidian distance*.
- Simplest for $m$-dimensional binary instance space is *Hamming distance* (number of feature values that differ).
- For text, cosine similarity of TF-IDF weighted vectors is typically most effective.
3 Nearest Neighbor Illustration
(Euclidian Distance)
K Nearest Neighbor for Text

Training:
For each training example \( <x, c(x)> \in D \)
   Compute the corresponding TF-IDF vector, \( d_x \), for document \( x \)

Test instance \( y \):
Compute TF-IDF vector \( d \) for document \( y \)
For each \( <x, c(x)> \in D \)
   Let \( s_x = \text{cosSim}(d, d_x) \)
Sort examples, \( x \), in \( D \) by decreasing value of \( s_x \)
Let \( N \) be the first \( k \) examples in \( D \) \( \text{(get most similar neighbors)} \)
Return the majority class of examples in \( N \)
Illustration of 3 Nearest Neighbor for Text
Bayesian Classification
Bayesian Methods

• Learning and classification methods based on probability theory.
  – Bayes theorem plays a critical role in probabilistic learning and classification.

• Uses prior probability of each category
  – Based on training data

• Categorization produces a posterior probability distribution over the possible categories given a description of an item.
Basic Probability Theory

• All probabilities between 0 and 1

\[ 0 \leq P(A) \leq 1 \]

• True proposition has probability 1, false has probability 0.

\[ P(\text{true}) = 1 \quad P(\text{false}) = 0. \]

• The probability of disjunction is:

\[ P(A \lor B) = P(A) + P(B) - P(A \land B) \]
Conditional Probability

• \( P(A \mid B) \) is the probability of \( A \) given \( B \)
• Assumes that \( B \) is all and only information known.
• Defined by:

\[
P(A \mid B) = \frac{P(A \cap B)}{P(B)}
\]
Independence

• $A$ and $B$ are independent iff:

$P(A | B) = P(A)$
$P(B | A) = P(B)$

• Therefore, if $A$ and $B$ are independent:

$P(A | B) = \frac{P(A \land B)}{P(B)} = P(A)$

$P(A \land B) = P(A)P(B)$

These two constraints are logically equivalent
Joint Distribution

- Joint probability distribution for $X_1, \ldots, X_n$ gives the probability of every combination of values: $P(X_1, \ldots, X_n)$
  - All values must sum to 1.

<table>
<thead>
<tr>
<th>Category=positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color\shape</td>
<td>circle</td>
</tr>
<tr>
<td>red</td>
<td>0.20</td>
</tr>
<tr>
<td>blue</td>
<td>0.02</td>
</tr>
</tbody>
</table>

- Probability for assignments of values to some subset of variables can be calculated by summing the appropriate subset

\[
P(red \land \text{circle}) = 0.20 + 0.05 = 0.25
\]

\[
P(red) = 0.20 + 0.02 + 0.05 + 0.3 = 0.57
\]

- Conditional probabilities can also be calculated.

\[
P(\text{positive} \mid red \land \text{circle}) = \frac{P(\text{positive} \land red \land \text{circle})}{P(red \land \text{circle})} = \frac{0.20}{0.25} = 0.80
\]
Computing probability from a training dataset

<table>
<thead>
<tr>
<th>Ex</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

Test Instance $X$: <medium, red, circle>

<table>
<thead>
<tr>
<th>Probability</th>
<th>$Y=\text{positive}$</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(Y)$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{small} \mid Y)$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{medium} \mid Y)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$P(\text{large} \mid Y)$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{red} \mid Y)$</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{blue} \mid Y)$</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{green} \mid Y)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$P(\text{square} \mid Y)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$P(\text{triangle} \mid Y)$</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{circle} \mid Y)$</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Bayes Theorem

\[ P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} \]

Simple proof from definition of conditional probability:

\[ P(H \mid E) = \frac{P(H \land E)}{P(E)} \quad \text{(Def. cond. prob.)} \]

\[ P(E \mid H) = \frac{P(H \land E)}{P(H)} \quad \text{(Def. cond. prob.)} \]

\[ P(H \land E) = P(E \mid H)P(H) \]

Thus:

\[ P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} \]
Bayesian Categorization

• Determine category of instance $x_k$ by determining for each $y_i$

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)}$$

• $P(X=x_k)$ estimation is not needed in the algorithm to choose a classification decision via comparison.

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)}$$

• If really needed:

$$\sum_{i=1}^{m} P(Y = y_i \mid X = x_k) = \sum_{i=1}^{m} \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} = 1$$

$$P(X = x_k) = \sum_{i=1}^{m} P(Y = y_i)P(X = x_k \mid Y = y_i)$$
Bayesian Categorization (cont.)

• Need to know: $P(Y = y_i | X = x_k) = \frac{P(Y = y_i)P(X = x_k | Y = y_i)}{P(X = x_k)}$
  
  – Priors: $P(Y = y_i)$
  – Conditionals: $P(X = x_k | Y = y_i)$

• $P(Y = y_i)$ are easily estimated from training data.
  – If $n_i$ of the examples in training data $D$ are in $y_i$ then
    $P(Y = y_i) = n_i / |D|$

• Too many possible instances (e.g. $2^n$ for binary features) to estimate all $P(X = x_k | Y = y_i)$ in advance.
Naïve Bayesian Categorization

• If we assume features of an instance are independent given the category (conditionally independent).

\[
P(X \mid Y) = P(X_1, X_2, \cdots X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)
\]

• Therefore, we then only need to know \(P(X_i \mid Y)\) for each possible pair of a feature-value and a category.
  – \(n_i\) of the examples in training data \(D\) are in \(y_i\)
  – \(n_{ij}\) of the examples in \(D\) with category \(y_i\)
  – \(P(x_{ij} \mid Y=y_i) = \frac{n_{ij}}{n_i}\)

Underflow Prevention:
Multiplying lots of probabilities may result in floating-point underflow. Since \(\log(xy) = \log(x) + \log(y)\), it is better to perform all computations by summing logs of probabilities.
Computing probability from a training dataset

<table>
<thead>
<tr>
<th>Ex</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
<th>Probability</th>
<th>Y=positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
<td>P(Y)</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(small</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(medium</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(large</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(red)</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(blue)</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(green)</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(square)</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(triangle)</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P(circle)</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Test Instance $X$: 
<medium, red, circle>
## Naïve Bayes Example

<table>
<thead>
<tr>
<th>Probability</th>
<th>$Y=\text{positive}$</th>
<th>$Y=\text{negative}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(Y)$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(\text{small} \mid Y)$</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>$P(\text{medium} \mid Y)$</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$P(\text{large} \mid Y)$</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>$P(\text{red} \mid Y)$</td>
<td>0.9</td>
<td>0.3</td>
</tr>
<tr>
<td>$P(\text{blue} \mid Y)$</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>$P(\text{green} \mid Y)$</td>
<td>0.05</td>
<td>0.4</td>
</tr>
<tr>
<td>$P(\text{square} \mid Y)$</td>
<td>0.05</td>
<td>0.4</td>
</tr>
<tr>
<td>$P(\text{triangle} \mid Y)$</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>$P(\text{circle} \mid Y)$</td>
<td>0.9</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Test Instance: $<\text{medium},\text{red},\text{circle}>$
Naïve Bayes Example

<table>
<thead>
<tr>
<th>Probability</th>
<th>Y=positive</th>
<th>Y=negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(Y)</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>P(medium</td>
<td>Y)</td>
<td>0.1</td>
</tr>
<tr>
<td>P(red</td>
<td>Y)</td>
<td>0.9</td>
</tr>
<tr>
<td>P(circle</td>
<td>Y)</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Test Instance: <medium, red, circle>

P(positive | X) = P(Positive)*P(X/Positive)/P(X)
       = P(positive)*P(medium | positive)*P(red | positive)*P(circle | positive) / P(X)
       = 0.5 * 0.1 * 0.9 * 0.9
       = 0.0405 / P(X) = 0.0405 / 0.0495 = 0.8181

P(negative | X) = P(negative)*P(medium | negative)*P(red | negative)*P(circle | negative) / P(X)
       = 0.5 * 0.2 * 0.3 * 0.3
       = 0.009 / P(X) = 0.009 / 0.0495 = 0.1818

P(positive | X) + P(negative | X) = 0.0405 / P(X) + 0.009 / P(X) = 1

P(X) = (0.0405 + 0.009) = 0.0495
Error prone prediction with small training data

<table>
<thead>
<tr>
<th>Ex</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probability</th>
<th>Y=positive</th>
<th>negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(Y)</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>P(small</td>
<td>Y)</td>
<td>0.5</td>
</tr>
<tr>
<td>P(medium</td>
<td>Y)</td>
<td>0.0</td>
</tr>
<tr>
<td>P(large</td>
<td>Y)</td>
<td>0.5</td>
</tr>
<tr>
<td>P(red</td>
<td>Y)</td>
<td>1.0</td>
</tr>
<tr>
<td>P(blue</td>
<td>Y)</td>
<td>0.0</td>
</tr>
<tr>
<td>P(green</td>
<td>Y)</td>
<td>0.0</td>
</tr>
<tr>
<td>P(square</td>
<td>Y)</td>
<td>0.0</td>
</tr>
<tr>
<td>P(triangle</td>
<td>Y)</td>
<td>0.0</td>
</tr>
<tr>
<td>P(circle</td>
<td>Y)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Test Instance $X$:  
<medium, red, circle>

\[
P(\text{positive} \mid X) = 0.5 \times 0.0 \times 1.0 \times 1.0 = 0
\]

\[
P(\text{negative} \mid X) = 0.5 \times 0.0 \times 0.5 \times 0.5 = 0
\]
Smoothing

• To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
• Laplace smoothing using an $m$-estimate assumes that each feature is given a prior probability, $p$, that is assumed to have been previously observed in a “virtual” sample of size $m$.

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk} + mp}{n_k + m}$$

• For binary features, $p$ is simply assumed to be 0.5.
Laplace Smoothing Example

- Assume training set contains 10 positive examples:
  - 4: small
  - 0: medium
  - 6: large

- Estimate parameters as follows (if $m=1$, $p=1/3$)
  - $P(\text{small} \mid \text{positive}) = (4 + 1/3) / (10 + 1) = 0.394$
  - $P(\text{medium} \mid \text{positive}) = (0 + 1/3) / (10 + 1) = 0.03$
  - $P(\text{large} \mid \text{positive}) = (6 + 1/3) / (10 + 1) = 0.576$
  - $P(\text{small or medium or large} \mid \text{positive}) = 1.0$
Bayes Training Example

Category

spam
legit
spam
spam
spam
legit
spam
legit
spam
legit

spam

Viagra
win
hot!!
Nigeria!
deal
lottery
nude!
Viagra
$

legit

science
PM
computer
Friday
test
homework
March
score
May
exam
Evaluating Accuracy of Classification

• Evaluation must be done on test data that are independent of the training data
  – **Classification accuracy**: the number of test instances correctly classified divided by total number of test instances
  – Average results over multiple training and test sets (splits of the overall data) for the best results.

• Not enough labeled data? N-fold cross-validation

• Partition data into $N$ equal-sized disjoint segments.
  – Run $N$ trials, each time using a different segment of the data for testing, and training on the remaining $N-1$ segments.
  – This way, at least test-sets are independent.
  – Report average classification accuracy over the $N$ trials.
  – Typically, $N = 10$. 
Sample Learning Curve
(Yahoo Science Data)
Classification with Decision Trees
Decision Trees

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless \(f\) nondeterministic in \(x\)) but it probably won't generalize to new examples.

- Prefer to find more compact decision trees: we don’t want to memorize the data, we want to find structure in the data!
Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range ($, $$, $$$)
7. **Raining**: is it raining outside?
8. **Reservation**: have we made a reservation?
9. **Type**: kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate**: estimated waiting time (0-10, 10-30, 30-60, >60)
Training data: Restaurant example

- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Target Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₂</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>X₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₄</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>X₅</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₆</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₇</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₈</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>X₉</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>X₁₀</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>X₁₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>X₁₂</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

- Classification of examples is positive (T) or negative (F)
A decision tree to decide whether to wait

- imagine someone talking a sequence of decisions.
Decision tree learning

• If there are so many possible trees, can we actually search this space? (solution: greedy search).

• **Aim**: find a small tree consistent with the training examples

• **Idea**: (recursively) choose "most significant" attribute as root of (sub)tree.
Choosing an attribute for making a decision

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

To wait or not to wait is still at 50%.
Information theory background: Entropy

- **Entropy** measures uncertainty
  
  \[-p \log (p) - (1-p) \log (1-p)\]

Consider tossing a biased coin. If you toss the coin VERY often, the frequency of heads is, say, \( p \), and hence the frequency of tails is \( 1-p \).

Uncertainty (entropy) is zero if \( p=0 \) or \( 1 \) and maximal if we have \( p=0.5 \).
Using information theory for binary decisions

- Imagine we have $p$ examples which are true (positive) and $n$ examples which are false (negative).

- Our best estimate of true or false is given by:
  
  $P(\text{true}) \approx \frac{p}{p + n}$
  
  $P(\text{false}) \approx \frac{n}{p + n}$

- Hence the entropy is given by:

  $$\text{Entropy}(\frac{p}{p+n}, \frac{n}{p+n}) \approx -\frac{p}{p+n} \log \frac{p}{p+n} - \frac{n}{p+n} \log \frac{n}{p+n}$$
Using information theory for more than 2 states

- If there are more than two states $s=1, 2, \ldots, n$ we have (e.g. a die):

\[
\text{Entropy}(p) = -p(s = 1) \log[p(s = 1)] - p(s = 2) \log[p(s = 2)] - p(s = n) \log[p(s = n)]
\]

\[
\sum_{s=1}^{n} p(s) = 1
\]
ID3 Algorithm: Using Information Theory to Choose an Attribute

- How much information do we gain if we disclose the value of some attribute?
- ID3 algorithm by Ross Quinlan uses information gained measured by maximum entropy reduction:
  - IG(A) = uncertainty before – uncertainty after
  - Choose an attribute with the maximum IA
Before: Entropy = \(- \frac{1}{2} \log(1/2) - \frac{1}{2} \log(1/2) = \log(2) = 1\) bit:
There is “1 bit of information to be discovered”.

After: for Type: If we go into branch “French” we have 1 bit, similarly for the others.

\[
\begin{align*}
\text{French: } & 1 \text{ bit} \\
\text{Italian: } & 1 \text{ bit} \\
\text{Thai: } & 1 \text{ bit} \\
\text{Burger: } & 1 \text{ bit}
\end{align*}
\]

\{ 
On average: 1 bit and gained nothing!
\}

After: for Patrons: In branch “None” and “Some” entropy = 0!,
In “Full” entropy = \(-\frac{1}{3} \log(1/3) - \frac{2}{3} \log(2/3) = 0.92\)

So Patrons gains more information!
Information Gain: How to combine branches

- 1/6 of the time we enter “None”, so we weight “None” with 1/6. Similarly: “Some” has weight: 1/3 and “Full” has weight ½.

\[
\text{Entropy}(A) = \sum_{i=1}^{n} \frac{p_i + n_i}{p + n} \cdot \text{Entropy} \left( \frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \right)
\]

weight for each branch

entropy for each branch.
Choose an attribute: Restaurant Example

For the training set, \( p = n = 6 \), \( I(6/12, 6/12) = 1 \) bit

\[
IG(Patrons) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6}) \right] = 0.0541 \text{ bits}
\]

\[
IG(Type) = 1 - \left[ \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) \right] = 0 \text{ bits}
\]

*Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root
Example: Decision tree learned

- Decision tree learned from the 12 examples:
Issues

- When there are no attributes left:
  - Stop growing and use majority vote.
- Avoid over-fitting data
  - Stop growing a tree earlier
  - Grow first, and prune later.
- Deal with continuous-valued attributes
  - Dynamically select thresholds/intervals.
- Handle missing attribute values
  - Make up with common values
- Control tree size
  - pruning
Classification with SVM
Two Class Problem: Linear Separable Case with a Hyperplane

Many decision boundaries can separate classes using a hyperplane. Which one should we choose?

Example of Bad Decision Boundaries
Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- *Quadratic programming* problem
### Training examples for document ranking

Two ranking signals are used (Cosine text similarity score, proximity of term appearance window)

<table>
<thead>
<tr>
<th>Example</th>
<th>DocID</th>
<th>Query</th>
<th>Cosine score</th>
<th>Judgment</th>
<th>ω</th>
</tr>
</thead>
<tbody>
<tr>
<td>Φ₁</td>
<td>37</td>
<td>linux operating system</td>
<td>0.032</td>
<td>3</td>
<td>relevant</td>
</tr>
<tr>
<td>Φ₂</td>
<td>37</td>
<td>penguin logo</td>
<td>0.02</td>
<td>4</td>
<td>nonrelevant</td>
</tr>
<tr>
<td>Φ₃</td>
<td>238</td>
<td>operating system</td>
<td>0.043</td>
<td>2</td>
<td>relevant</td>
</tr>
<tr>
<td>Φ₄</td>
<td>238</td>
<td>runtime environment</td>
<td>0.004</td>
<td>2</td>
<td>nonrelevant</td>
</tr>
<tr>
<td>Φ₅</td>
<td>1741</td>
<td>kernel layer</td>
<td>0.022</td>
<td>3</td>
<td>relevant</td>
</tr>
<tr>
<td>Φ₆</td>
<td>2094</td>
<td>device driver</td>
<td>0.03</td>
<td>2</td>
<td>relevant</td>
</tr>
<tr>
<td>Φ₇</td>
<td>3191</td>
<td>device driver</td>
<td>0.027</td>
<td>5</td>
<td>nonrelevant</td>
</tr>
</tbody>
</table>

...
Proposed scoring function for ranking

$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c,$$
Formalization

- \( w \): weight coefficients
- \( x_i \): data point \( i \)
- \( y_i \): class result of data point \( i \) (+1 or -1)
- Classifier is:
  \[
f(x_i) = \text{sign}(w^T x_i + b)
  \]
Linear Support Vector Machine (SVM)

- Hyperplane
  \[ w^T x + b = 0 \]
  \[ w^T x + b = 1 \]
  \[ w^T x + b = -1 \]

- Support vectors
  datapoints that the margin pushes up against

- \( \rho = \|x_a - x_b\|_2 = 2/\|w\|_2 \)

- \( \|w\|^2 = w^T w \)
Linear SVM Mathematically

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set \( \{(x_i, y_i)\} \)

\[
\begin{align*}
    w^T x_i + b &\geq 1 \quad \text{if } y_i = 1 \\
    w^T x_i + b &\leq -1 \quad \text{if } y_i = -1
\end{align*}
\]

- For support vectors, the inequality becomes an equality
- Then, each example’s distance from the hyperplane is

\[
r = y \frac{w^T x + b}{\|w\|}
\]

- The margin of dataset is:

\[
\rho = \frac{2}{\|w\|}
\]
The Optimization Problem

- Let \( \{x_1, \ldots, x_n\} \) be our data set and let \( y_i \in \{1,-1\} \) be the class label of \( x_i \)
- The decision boundary should classify all points correctly \( \Rightarrow \)
- A constrained optimization problem
  
  \[
  \text{Minimize} \quad \frac{1}{2} ||w||^2 \\
  \text{subject to} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i
  \]
Classification with SVMs

• Given a new point \((x_1,x_2)\), we can score its projection onto the hyperplane normal:
  – In 2 dims: score = \(w_1 x_1 + w_2 x_2 + b\).
    • I.e., compute score: \(wx + b = \sum \alpha_i y_i x_i^T x + b\)
  – Set confidence threshold \(t\).

Score > \(t\): yes
Score < \(-t\): no
Else: don’t know
Soft Margin Classification

- If the training set is not linearly separable, *slack variables* $\xi_i$ can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane “far” from each class (large margin)
Soft margin

- We allow “error” $\xi_i$ in classification; it is based on the output of the discriminant function $w^Tx+b$
- $\xi_i$ approximates the number of misclassified samples

New objective function:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$$

$C$: tradeoff parameter between error and margin; chosen by the user; large $C$ means a higher penalty to errors
Soft Margin Classification
Mathematically

- The old formulation:

Find \( \mathbf{w} \) and \( b \) such that
\[
\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \quad \text{is minimized and for all } \{(\mathbf{x}_i, y_i)\}
\]
\[
y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1
\]

- The new formulation incorporating slack variables:

Find \( \mathbf{w} \) and \( b \) such that
\[
\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \quad \text{is minimized and for all } \{(\mathbf{x}_i, y_i)\}
\]
\[
y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i
\]

- Parameter \( C \) can be viewed as a way to control overfitting – a regularization term
Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:

- But what are we going to do if the dataset is just too hard?

- How about … mapping data to a higher-dimensional space:
Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \phi(x) \]
Transformation to Feature Space

• “Kernel tricks”
  – Make non-separable problem separable.
  – Map data into better representational space
Example Transformation

• Consider the following transformation

\[
\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)
\]

\[
\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)
\]

\[
\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)\rangle = (1 + x_1y_1 + x_2y_2)^2
\]

\[
= K(x, y)
\]

• Define the kernel function \(K(x, y)\) as

\[
K(x, y) = (1 + x_1y_1 + x_2y_2)^2
\]

• SVM computation involves pair-wise vector product. The inner product \(\phi(\cdot)\phi(\cdot)\) can be computed by \(K\) without going through the map \(\phi(\cdot)\) explicitly!
Choosing a Kernel Function

- Active research on kernel function choices for different applications

- Examples:
  - Polynomial kernel with degree $d$: $K(x, y) = (x^T y + 1)^d$
  - Radial basis function (RBF) kernel:
    
    $$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

  - or sometime $K(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$

- Closely related to radial basis function neural networks

- In practice, a low degree polynomial kernel or RBF kernel is a good initial try
Example: 5 1D data points

Value of discriminant function

class 1

class 2

class 1

1 2 4 5 6

We use the polynomial kernel of degree 2

\[ K(x,y) = (xy+1)^2 \]
Software

• A list of SVM implementation can be found at http://www.kernel-machines.org/software.html
• Some implementation (such as LIBSVM) can handle multi-class classification
• SVMLight is among one of the earliest implementation of SVM
• Several Matlab toolboxes for SVM are also available
Evaluation: Reuters News Data Set

- Most (over)used data set
- 21,578 documents
- 9,603 training, 3,299 test articles (ModApte split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories (#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)
New Reuters: RCV1: 810,000 docs

- Top topics in Reuters RCV1
Dumais et al. 1998: Reuters - Accuracy

<table>
<thead>
<tr>
<th>Category</th>
<th>Rocchio</th>
<th>NBayes</th>
<th>Trees</th>
<th>LinearSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>earn</td>
<td>92.9%</td>
<td>95.9%</td>
<td>97.8%</td>
<td>98.2%</td>
</tr>
<tr>
<td>acq</td>
<td>64.7%</td>
<td>87.8%</td>
<td>89.7%</td>
<td>92.8%</td>
</tr>
<tr>
<td>money-fx</td>
<td>46.7%</td>
<td>56.6%</td>
<td>66.2%</td>
<td>74.0%</td>
</tr>
<tr>
<td>grain</td>
<td>67.5%</td>
<td>78.8%</td>
<td>85.0%</td>
<td>92.4%</td>
</tr>
<tr>
<td>crude</td>
<td>70.1%</td>
<td>79.5%</td>
<td>85.0%</td>
<td>88.3%</td>
</tr>
<tr>
<td>trade</td>
<td>65.1%</td>
<td>63.9%</td>
<td>72.5%</td>
<td>73.5%</td>
</tr>
<tr>
<td>interest</td>
<td>63.4%</td>
<td>64.9%</td>
<td>67.1%</td>
<td>76.3%</td>
</tr>
<tr>
<td>ship</td>
<td>49.2%</td>
<td>85.4%</td>
<td>74.2%</td>
<td>78.0%</td>
</tr>
<tr>
<td>wheat</td>
<td>68.9%</td>
<td>69.7%</td>
<td>92.5%</td>
<td>89.7%</td>
</tr>
<tr>
<td>corn</td>
<td>48.2%</td>
<td>65.3%</td>
<td>91.8%</td>
<td>91.1%</td>
</tr>
</tbody>
</table>

Avg Top 10 | 64.6% | 81.5% | 88.4% | 91.4% |
Avg All Cat | 61.7% | 75.2% | na    | 86.4% |

**Recall**: % labeled in category among those stories that are really in category  
**Precision**: % really in category among those stories labeled in category  
**Break Even**: $(\text{Recall} + \text{Precision}) / 2$
## Results for Kernels (Joachims 1998)

<table>
<thead>
<tr>
<th></th>
<th>Bayes</th>
<th>Rocchio</th>
<th>C4.5</th>
<th>k-NN</th>
<th>SVM (poly) degree $d =$</th>
<th>SVM (rbf) width $\gamma =$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>earn</td>
<td>95.9</td>
<td>96.1</td>
<td>96.1</td>
<td>97.3</td>
<td>98.2</td>
<td>98.4</td>
</tr>
<tr>
<td>acq</td>
<td>91.5</td>
<td>92.1</td>
<td>85.3</td>
<td>92.0</td>
<td>92.6</td>
<td>94.6</td>
</tr>
<tr>
<td>money-fx</td>
<td>62.9</td>
<td>67.6</td>
<td>69.4</td>
<td>78.2</td>
<td>66.9</td>
<td>72.5</td>
</tr>
<tr>
<td>grain</td>
<td>72.5</td>
<td>79.5</td>
<td>89.1</td>
<td>82.2</td>
<td>91.3</td>
<td>93.1</td>
</tr>
<tr>
<td>crude</td>
<td>81.0</td>
<td>81.5</td>
<td>75.5</td>
<td>85.7</td>
<td>86.0</td>
<td>87.3</td>
</tr>
<tr>
<td>trade</td>
<td>50.0</td>
<td>77.4</td>
<td>59.2</td>
<td>77.4</td>
<td>69.2</td>
<td>75.5</td>
</tr>
<tr>
<td>interest</td>
<td>58.0</td>
<td>72.5</td>
<td>49.1</td>
<td>74.0</td>
<td>69.8</td>
<td>63.3</td>
</tr>
<tr>
<td>ship</td>
<td>78.7</td>
<td>83.1</td>
<td>80.9</td>
<td>79.2</td>
<td>82.0</td>
<td>85.4</td>
</tr>
<tr>
<td>wheat</td>
<td>60.6</td>
<td>79.4</td>
<td>85.5</td>
<td>76.6</td>
<td>83.1</td>
<td>84.5</td>
</tr>
<tr>
<td>corn</td>
<td>47.3</td>
<td>62.2</td>
<td>87.7</td>
<td>77.9</td>
<td>86.0</td>
<td>86.5</td>
</tr>
<tr>
<td>microavg.</td>
<td>72.0</td>
<td>79.9</td>
<td>79.4</td>
<td>82.3</td>
<td>84.2</td>
<td>85.1</td>
</tr>
</tbody>
</table>

combined: 86.0
combined: 86.4
If we have more than one class, how do we combine multiple performance measures into one quantity?

- **Macroaveraging:** Compute performance for each class, then average.
- **Microaveraging:** Collect decisions for all classes, compute contingency table, evaluate.
Micro- vs. Macro-Averaging: Example

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Truth: yes</th>
<th>Truth: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: yes</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>10</td>
<td>970</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class 2</th>
<th>Truth: yes</th>
<th>Truth: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: yes</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>10</td>
<td>890</td>
</tr>
</tbody>
</table>

- Macroaveraged precision: \((0.5 + 0.9)/2 = 0.7\)
- Microaveraged precision: \(100/120 = .83\)
- Why this difference?

Micro.Av. Table

<table>
<thead>
<tr>
<th></th>
<th>Truth: yes</th>
<th>Truth: no</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: yes</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Classifier: no</td>
<td>20</td>
<td>1860</td>
</tr>
</tbody>
</table>
The Real World

- 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) then categorize as grain
  - With careful crafting (human tuning on development data) performance is high:
    - 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
  - Amount of work required is huge
    - Estimate 2 days per class … plus maintenance
Which methods to use?

- A reasonable amount of data
  - Good with SVM, Trees
  - Be prepared with the “hybrid” solution.

- A huge amount of data
  - SVMs (train time) or kNN (test time) can be too expensive.
  - Naïve Bayes, logistic regression
  - Trees including boosting trees, random forests