Data mining components

- Task specification: Prediction
- Data representation: Homogeneous IID data
- Knowledge representation
- Learning technique
- Prediction technique
Descriptive vs. predictive modeling

• Descriptive models **summarize** the data
  – Provides insights into the domain
  – Focus on modeling joint distribution $P(X)$
  – May be used for classification, but prediction is not the primary goal

• Predictive models **predict** the value of one variable of interest given known values of other variables
  – Focus on modeling the conditional distribution $P(Y|X)$ or on modeling the decision boundary for $Y$

Example: SPAM

• I was reading a little more about Tsalling entropy and trying to figure out whether it would be appropriate for relational learning problems. One possibility is to use it for exponential random graph models, which have features like the number of triangles in the graph. Since these grow with graph size, it seems to be an "extensive" property that the Tsalling entropy is trying to model…

• Don't Be Silly To Pay Hundred$ Or Thousand$ You Can Have The Exactly Same Licensed Software At 5%-10% Of The Retail Price : All popular softwares for PC & MAC : Language available: English, Deutsch, French, Italian, Spanish : Buy & start downloading right after you paid : You will be given your dedicated PERSONAL LICENSE right after you paid….
Data representation

• Class label: isSpam \{+, -\}

• Attributes?
  – Convert email text into a set of attributes

<table>
<thead>
<tr>
<th>isSpam</th>
<th>word₁</th>
<th>Word₂</th>
<th>Word₃</th>
<th>...</th>
<th>Wordₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Predictive modeling

• Data representation:
  – Training set: Paired attribute vectors and class labels \(<y(i), x(i)>\)
    or \(n \times p\) tabular data with class label \(y\) and \(p-1\) attributes \(x\)

• Task: estimate a predictive function \(f(x; \theta) = y\)
  – Assume that there is a function \(y = f(x)\) that maps data instances \(x\) to class labels \(y\)
  – Construct a model that approximates the mapping
    • Classification: if \(y\) is categorical
    • Regression: if \(y\) is real-valued
Modeling approaches: Classification

- In its simplest form, a classification model defines a decision boundary (h) and labels for each side of the boundary.
- Input: $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is a set of attributes, function $f$ assigns a label $y$ to input $\mathbf{x}$, where $y$ is a discrete variable with a finite number of values.

What is the decision boundary: KNN
What is the decision boundary: **Perceptron**

\[
f(x) = \begin{cases} 
1 & \sum w_j x_j > 0 \\
0 & \sum w_j x_j \leq 0 
\end{cases}
\]

**Model space:** weights \( w \), for each of \( j \) attributes

---

What is the decision boundary: **Classification tree**
Parametric vs. non-parametric models

- **Parametric**
  - Particular functional form is assumed (e.g., Binomial)
  - **Number of parameters is fixed in advance**
  - Examples: Naive Bayes, perceptron

- **Non-parametric**
  - Few assumptions are made about the functional form
  - **Model structure is determined from data**
  - Examples: classification tree, nearest neighbor

Classification output

- Different classification tasks can require different kinds of output
  - Each requires progressively more accurate models (e.g., a poor probability estimator can still produce an accurate ranking)

- **Class labels** — Each instance is assigned a single label
  - Model only need to decide on crisp class boundaries

- **Ranking** — Instances are ranked according to their likelihood of belonging to a particular class
  - Model implicitly explores many potential class boundaries

- **Probabilities** — Instances are assigned class probabilities $p(y|x)$
  - Allows for more refined reasoning about sets of instances
Discriminative classification

- Model the decision boundary directly
- Direct mapping from inputs $x$ to class label $y$
- No attempt to model probability distributions
- May seek a discriminant function $f(x; \theta)$ that maximizes measure of separation between classes
- Examples:
  - Perceptrons, nearest neighbor classifiers, support vector machines, decision trees

Probabilistic classification

- Model the underlying probability distributions
  - Posterior class probabilities: $p(y|x)$
  - Class-conditional and class prior: $p(x|y)$ and $p(y)$
- Maps from inputs $x$ to class label $y$ indirectly through posterior class distribution $p(y|x)$
- Examples:
  - Naive Bayes classifier, logistic regression, probability estimation trees
Knowledge representation

- **Underlying structure of the model or patterns that we seek from the data**
  - Defines space of possible models for algorithm to search over

- **Model:** high-level global description of dataset
  - “All models are wrong, some models are useful”
    - *G. Box and N. Draper (1987)*
  - Choice of model family determines space of parameters and structure
  - Estimate model parameters and possibly model structure from training data

---

**Classification tree**

**Model space:**
- all possible decision trees
Model space

- How large is the space?
- Can we search exhaustively?
- Simplifying assumptions
  - Binary tree
  - Fixed depth
  - 10 binary attributes

<table>
<thead>
<tr>
<th>Tree depth</th>
<th>Number of trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>$8 \times 10^2$</td>
</tr>
<tr>
<td>3</td>
<td>$3 \times 10^6$</td>
</tr>
<tr>
<td>4</td>
<td>$2 \times 10^{13}$</td>
</tr>
<tr>
<td>5</td>
<td>$5 \times 10^{25}$</td>
</tr>
</tbody>
</table>

Perceptron

$$f(x) = \begin{cases} 
1 & \sum w_j x_j > 0 \\
0 & \sum w_j x_j \leq 0 
\end{cases}$$

Model space:
weights $w_j$ for each of $j$ attributes
Naive Bayes classifier

\[ p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{\prod_i p(x_i|y) p(y)}{\sum_j p(x|y_j)p(y_j)} \]

**Model space:**
- parameters in conditional distributions \( p(x_i|y) \)
- parameters in prior distribution \( p(y) \)

Decision rule

**Model space:**
- all possible rules formed from conjunctions of features
Nearest neighbor

**Rule**: find k closest (training) points to the test instance and assign the most frequently occurring class

Learning predictive models

- Choose a **data representation**
- Select a **knowledge representation** (a “model”)  
  - Defines a **space** of possible models $M = \{M_1, M_2, \ldots, M_k\}$
- Use **search** to identify “best” model(s)
  - Search the space of models (i.e., with alternative structures and/or parameters)
  - Evaluate possible models with **scoring function** to determine the model which best fits the data
What space are we searching?

- Consider a space of possible models $M=\{M_1, M_2, \ldots, M_k\}$
- Search could be over model structures or parameters, e.g.:
  - **Parameters**: In a linear regression model, find the regression coefficients ($\beta$) that minimize squared loss on the training data
  - **Model structure**: In a decision trees, find the tree structure that maximizes accuracy on the training data
Decision Trees

- A hierarchical data structure (tree) that represents data by implementing a divide and conquer strategy
- **Nodes** are tests for feature values
  - There is one branch for every value that the feature can take
- **Leaves** of the tree specify the class labels
- Given a collection of examples, **learn a decision tree that represents it.**
- Use this representation to **classify new examples**
  - The tree can be used as a non-parametric classification and regression method

Tree learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select **best** attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Other issues:
  - How to construct features
  - When to stop growing
  - Pruning irrelevant parts of the tree
Choosing an attribute/feature

- Idea: a good feature splits the examples into subsets that distinguish among the class labels as much as possible... ideally into pure sets of "all positive" or "all negative"

Which one is a better feature?

Information gain

- How much does a feature split decrease the entropy?

\[
Gain(S, A) = Entropy(S) - \sum_{v \in \text{values}(A)} \frac{|S_A|}{|S|} \times \text{Entropy}(S_A)
\]

\[
\text{Entropy}(D) = -\frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} = \frac{5}{14} = 0.9400
\]