COMP90051 Statistical Machine Learning

Semester 2, 2015

Lecturer: Ben Rubinstein

THE UNIVERSITY OF MELBOURNE

COMP90049 Revision

Covered Knowledge

- **un/supervised learning**
- **probability theory**; entropy
- association rule mining
- k-**means clustering**
- **naive Bayes**
- instance-based learning (IB1)
- feature selection (mutual information)
- **decision stump/tree induction** (0R, 1R, ID5)
- **basic sampling** (hold-out, cross-validation)
- **evaluation** (precision/recall/F, ROC)
- Seen: SVMs, bit of Bayes nets

Entropy

- The entropy of a discrete random event x with possible states $1, ..n$ is:

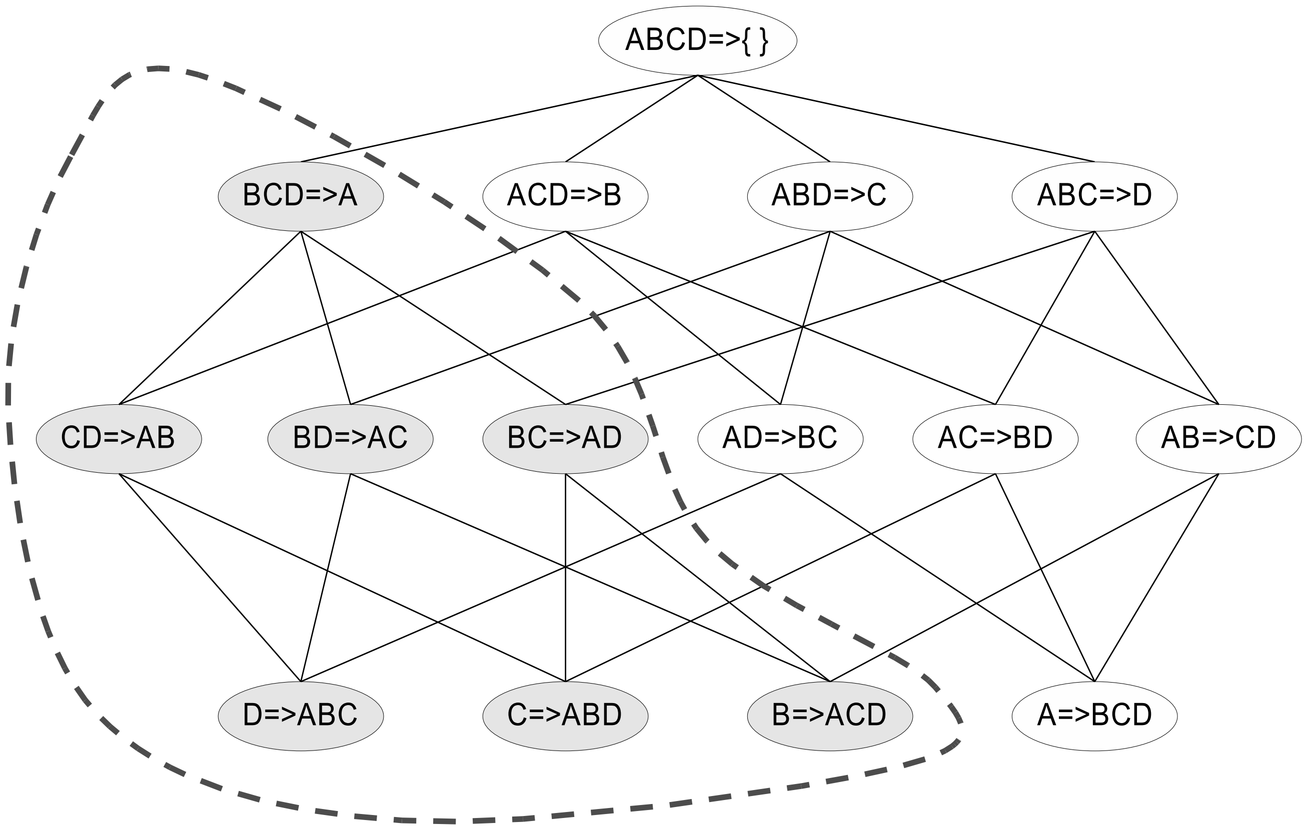
$$H(x) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

where $0 \log_2 0 =^{def} 0$

Association Rules: Definitions

- An association rule is an implication $A \rightarrow B$, where A and B are disjoint itemsets
  - ⋆ A = **antecedent**
  - ⋆ B = **consequent**
- N.B. in association mining parlance:
  - ⋆ **item** = attribute–value pair (I = set of items)
  - ⋆ **itemset** = set of attribute–value pairs
  - ⋆ k-**itemset** = set of k attribute–value pairs
  - ⋆ **transaction** = exemplar (T = set of transactions)

APriori Algorithm (Rule Generation)



k-means Clustering

- Given k, the k-means algorithm is implemented in four steps:
  1. Select k points at random to act as seed clusters
  2. Compute seed points as the centroids of the clusters of the current partition (the **centroid** is the centre, i.e., <u>mean</u> point, of the cluster)
  3. Assign each instance to the cluster with the nearest centroid
  4. Go back to 2, stop when no reassignments

- Exclusive, deterministic, partitioning, batch clustering method

Naive Bayes (NB) Classifiers

- Classify instance $D = \langle x_1, x_2, ..., x_n \rangle$ as class $c_j \in C$

$$
\begin{aligned}
c &= \arg\max_{c_j \in C} P(c_j | x_1, x_2, ..., x_n) \\
&= \arg\max_{c_j \in C} \frac{P(x_1, x_2, ..., x_n | c_j) P(c_j)}{P(x_1, x_2, ..., x_n)} \\
&= \arg\max_{c_j \in C} P(x_1, x_2, ..., x_n | c_j) P(c_j) \\
&= \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{n} P(x_i | c_j)
\end{aligned}
$$

- Model trained using frequencies

Nearest Neighbour Classification

- Combining training–test instance scores to form an overall categorisation function:
- **Method 1:** index all training documents, and query the training document set with each test document; classify the test document according to the class of the top-ranked training document **[1-NN]**
- **Method 2:** index all training documents, and query the training document set with each test document; classify the test document according to the **majority class** within the k top-ranked training documents **[k-NN]**

Similarity/Distance Metrics

- Cosine similarity:

$$sim(x,y) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

- Relative entropy:

$$D(x \mid\mid y) = \sum_i x_i(\log_2 x_i - \log_2 y_i)$$

or alternatively **skew divergence**:

$$s_\alpha(x,y) = D(x \mid\mid \alpha y + (1-\alpha)x)$$

Feature Selection

- **Mutual information**:

$$MI(T;C) = \sum_{t \in \{0,1\}} \sum_{c} P(t,c) \log_2 \frac{P(t,c)}{P(t)P(c)}$$

Constructing Decision Trees: ID3

- **Basic method:** construct decision trees in recursive divide-and-conquer fashion

  FUNCTION ID3 (Root)

  IF all instances at root have same class

  THEN stop

  ELSE Select a new attribute to use in partitioning root node instances

  Create a branch for each attribute value and partition up root node instances according to each value

  Call ID3(LEAF$_i$) for each leaf node LEAF$_i$

- Note: we may not end up with pure leaves

Split Criteria

- The **information gain** for attribute R_A (with values $x_1, ...x_m$) at a given root node R is:

$$IG(R_A|R) = H(R) - \sum_{i=1}^{m} P(x_i)H(x_i)$$

- The corresponding **gain ratio** is:

$$\begin{aligned} GR(R_A|R) &= \frac{IG(R_A|R)}{H(R_A)} \\ &= \frac{H(R) - \sum_{i=1}^{m} P(x_i)H(x_i)}{-\sum_{i=1}^{m} P(x_i)\log_2 P(x_i)} \end{aligned}$$

Evaluation

- **Classification accuracy**: is the proportion of

$$\mathrm{ACC} = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Error rate**:

$$\mathrm{ER} = \frac{FP + FN}{TP + FP + FN + TN}$$

- **Error rate reduction**:

$$\mathrm{ERR} = \frac{\mathrm{ER}_0 - \mathrm{ER}}{\mathrm{ER}_0}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F-score**:

$$\text{F-score} = (1 + \beta^2)\frac{PR}{R + \beta^2 P}$$

Sampling

- **Holdout** $=$ train a classifier over a fixed training dataset, and evaluate it over a fixed held-out test dataset
- **Random Subsampling** $=$ perform holdout over multiple iterations, randomly selecting the training and test data (maintaining a fixed size for each dataset) on each iteration
- **Cross Validation** $=$ partition data into N folds, and use $N-1$ as training data and 1 as test $\times N$ iterations
- **Stratified Cross Validation** $=$ partition the data so as to maintain the overall class distribution within individual partitions

ROC Curves

1. Sort the test instances in ascending order of "rating" $t_1, t_2, ..., t_k$

2. Initialise $TP_{k+1} = FP_{k+1} = 0$, and set FN_{k+1} and TN_{k+1} to the number of positive and negative instances in the dataset, resp.

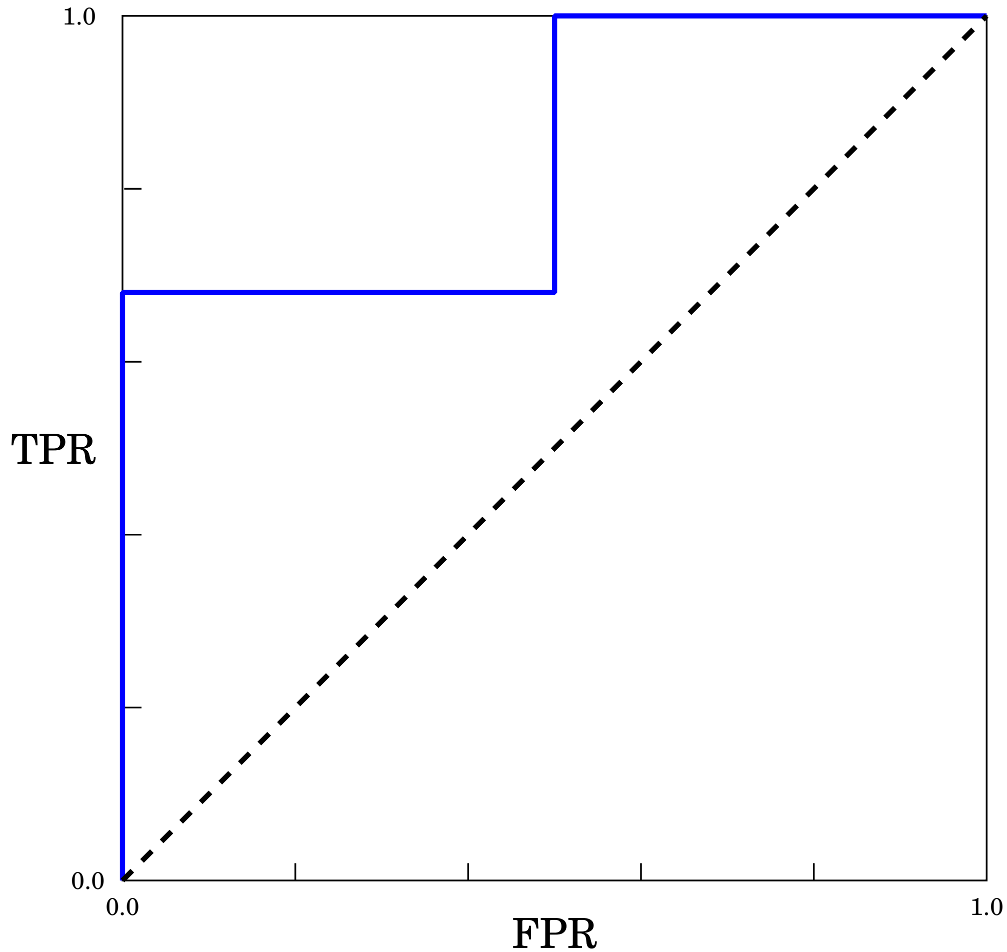
3. For each $i = k, ..., 2, 1$
   i. update TP_i, FP_i, FN_i and TN_i assuming positive classification of instance i, based on the actual class of t_i and TP_{i+1}, FP_{i+1}, FN_{i+1} and TN_{i+1}
   ii. calculate TPR and FPR at t_i

4. Plot the TPR and FPR values for each t_i

Generating ROC Curves: Example

Class	$-$	$+$	$-$	$+$	$+$	
Score	0.85	0.85	0.87	0.93	0.95	
i	1	2	3	4	5	6
TP	3	3	2	2	1	0
FP	2	1	1	0	0	0
FN	0	0	1	1	2	3
TN	0	1	1	2	2	2
TPR	$\frac{3}{3}$	$\frac{3}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	0
FPR	$\frac{2}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0

Generating ROC Curves: Example



What other topics in ML have you seen?