Data Classification Algorithms

a presentation by
B.Sivaselvan
Associate Professor, CSE
IIITD&M Kancheepuram

Classification

- Human beings - learning from past experiences.
- A computer does not have "experiences
- Form of data analysis – extracts models describing important data classes.
- Predict Future Trends
- Model (Classifier) is built to predict categorical labels (class labels)
- Model generated from training data
- Classifier used to predict labels for test data
- Supervised Learning mechanism

Classification Approaches

- learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- 2 step process – Training and Testing Phase
- Training (Learning Phase) - Learn / Generate a model to describe predefined classes (labels) using tuples & attributes – associated labels.
- Classification a.k.a Learning from Examples
- Testing Phase – New / Live data to be classified using the learnt model
- Eg: Biometric Fingerprint System,

Classification Algorithms

- Decision Tree Induction
  - - learn decision trees from training data
  - CART, ID3, SLIQ, .. – famous DT classifiers
  - Some More Applications:
  - to predict high-risk patients and discriminate them from low-risk patients – more relevant in the covid era with critcal ICU resource mgmt.
  - to decide whether a bank loan application should approved, or to classify applications into two categories, approved and not approved

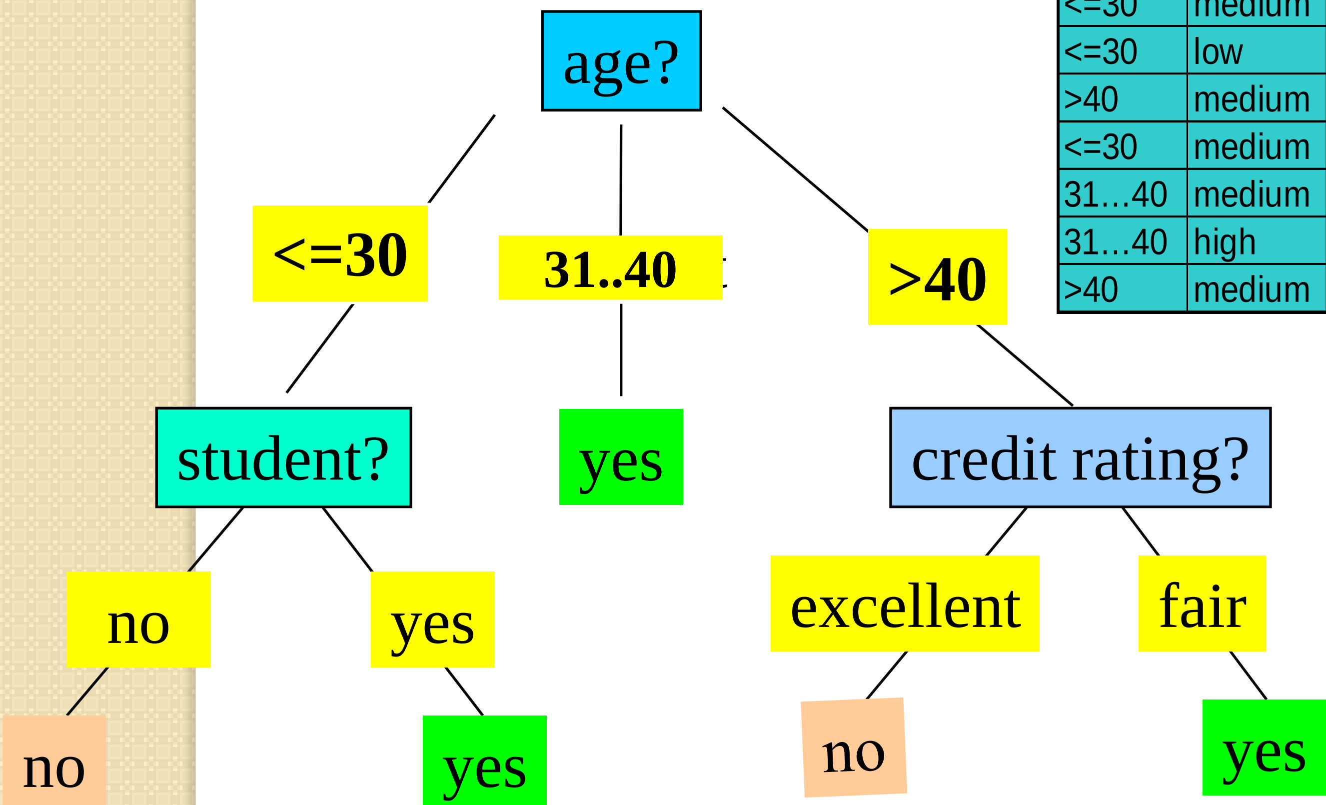
```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)     if D_j is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(D_j, attribute_list) to node N;
     endfor
(15) return N;
```

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conqu manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Decision Tree Induction: An Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$
$$= 0.443$$
$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Classifier Evaluation

- **Predictive accuracy**

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

where

TP: the number of correct classifications of the positive examples (**true positive**),

FN: the number of incorrect classifications of positive examples (**false negative**),

FP: the number of incorrect classifications of negative examples (**false positive**), and

TN: the number of correct classifications of negative examples (**true negative**).

Performance Measures contd…

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$p = \frac{TP}{TP + FP}. \qquad r = \frac{TP}{TP + FN}.$$

- Precision p is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
- Recall r is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

precision p = 100% and recall r = 1%

one positive example correctly and no negative examples wrongly.

Difficult to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p+r}$$

F_1-score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two. For F_1-value to be large, both p and r much be large.

Receive operating characteristics curve

- It is commonly called the ROC curve.
- It is a plot of the true positive rate (TPR) against the false positive rate (FPR).
- True positive ra

$$TPR = \frac{TP}{TP + FN}$$

- False positive ra

$$FPR = \frac{FP}{TN + FP}$$

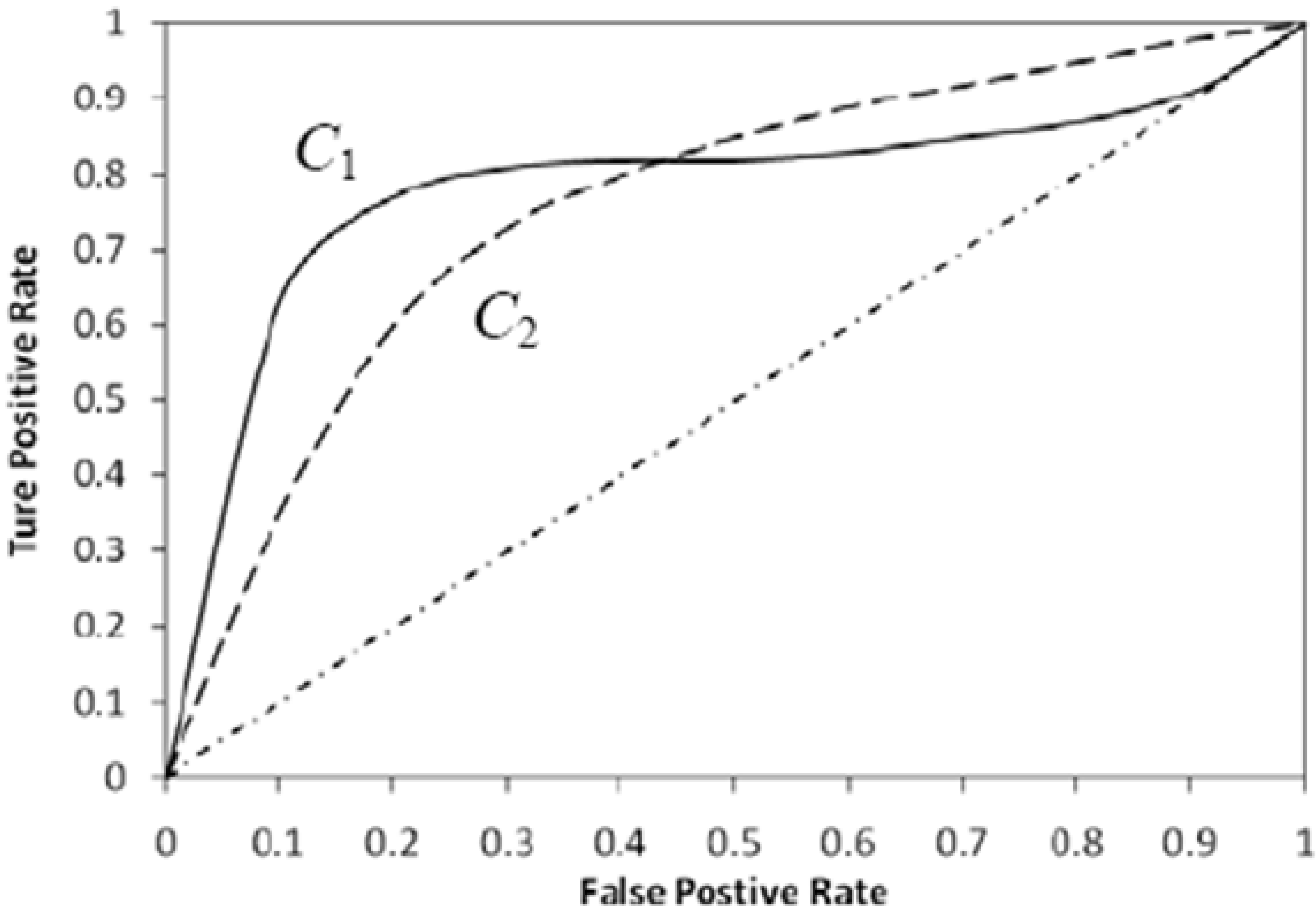
Sensitivity and Specificity

- In statistics, there are two other evaluation measures:
  - Sensitivity: Same as TPR
  - Specificity: Also called True Negative Rate (TNR)

$$TNR = \frac{TN}{TN + FP}$$

- Then we have $FPR = 1 - specificity$

Example ROC curves



Area under the curve (AUC)

- C_1 or C_2 ? Efficient dependent
  - It depends on which region you talk about.
- Can we have one measure?
  - Yes, we compute the area under the curve (AUC)
- If AUC for C_i is greater than that of C_j, it is said that C_i is better than C_j.
  - If a classifier is perfect, its AUC value is 1
  - If a classifier makes all random guesses, its AUC value is 0.5.