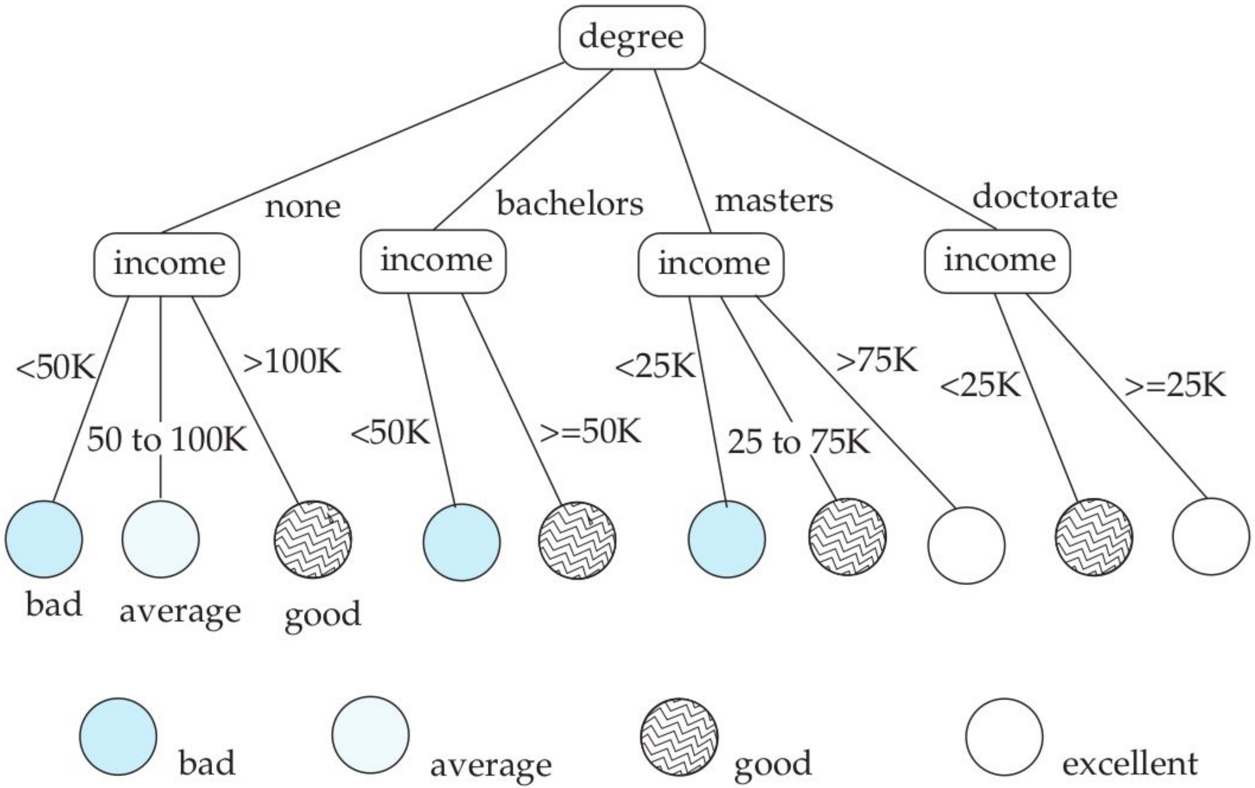
Databases and Information Systems CS303

Data Mining
08-11-2023

Recap : Decision-Tree Classifiers

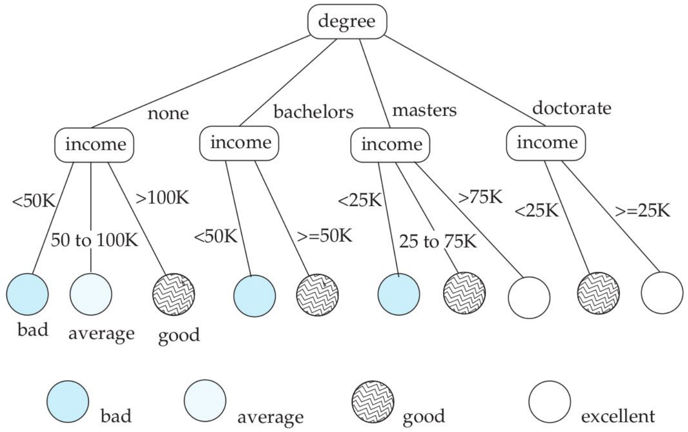


Building Decision Tree Classifiers

- Build a decision-tree classifier, given a set of training instances
- Greedy algorithm
    - Works recursively, starting at the root and building the tree downward
    - Initially there is only one node, the root, and all training instances are associated with that node
    - At each node, if all, or "almost all" training instances associated with the node belong to the same class, then the node becomes a leaf node associated with that class
    - Otherwise, a partitioning attribute and partitioning conditions must be selected to create child nodes
    - The data associated with each child node is the set of training instances that satisfy the partitioning condition for that child node.

Building Decision Tree Classifiers

- In the example,
  - The attribute degree is chosen, and four children, one for each value of degree, are created.
  - The conditions for the four children nodes are
    - degree= none
    - degree = bachelors
    - degree = masters
    - degree = doctorate
  - At the node corresponding to masters, the attribute income is chosen with the range of values partitioned into intervals
    - 0 to 25K, 25K to 50K, 50K to 75K, and over 75K
  - Since the class for the range 25K to 50K and the range 50K to 75K is the same under the node degree = masters, the two ranges have been merged into a single range 25K to 75K.



Building Decision Tree Classifiers : Best split

- We start with the set of all training instances, which is impure
  - It contains instances from many classes

- Ends up with leaves which are pure
  - All most all training instances belong to only one class

- Pick a particular attribute and condition for partitioning of the data at a node:
  - measure the purity of the data at the children resulting from partitioning by that attribute
  - The attribute and condition that result in the maximum purity are chosen.

Building Decision Tree Classifiers : Best split

- The purity of a set S of training instances can be measured quantitatively in several ways.

- Suppose there are k classes, and of the instances in S the fraction of instances in class i is p_i .
    - Gini measure, is defined as: $$\text{Gini}(S) = 1 - \sum_{i-1}^{k} p_i^2$$
        - Gini value is 0 (minimum) if there is a single class in S
        - Maximum ? (Exercise)
    - Entropy is defined as: $$\text{Entropy}(S) = -\sum_{i-1}^{k} p_i \log_2 p_i$$
        - Entropy is 0 (minimum) if there is a single class in S
        - Maximum ? (Exercise)

Building Decision Tree Classifiers : Best split

- When S is split into S_1 S_2..... S_r (purity can be either entropy or Gini value)

$$\text{Purity}(S_1, S_2, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \text{purity}(S_i)$$

$$\text{Information\_gain}(S, \{S_1, S_2, \ldots, S_r\}) = \text{purity}(S) - \text{purity}(S_1, S_2, \ldots, S_r)$$

Building Decision Tree Classifiers : Best split

- When S is split into S_1 S_2..... S_r (purity can be either entropy or Gini value)

$$\text{Purity}(S_1, S_2, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \text{purity}(S_i)$$

$$\text{Information\_gain}(S, \{S_1, S_2, \ldots, S_r\}) = \text{purity}(S) - \text{purity}(S_1, S_2, \ldots, S_r)$$

- The number of elements in each S_i should also be taken into account
    - Whether S_i has 0 or non-zero element makes a big difference

$$\text{Information\_content}(S, \{S_1, S_2, \ldots, S_r\}) = -\sum_{i-1}^{r} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- Best split has maximum information gain ratio

$$\frac{\text{Information\_gain}(S, \{S_1, S_2, \ldots, S_r\})}{\text{Information\_content}(S, \{S_1, S_2, \ldots, S_r\})}$$

Building Decision Tree Classifiers : Finding Best split

- Depends on the type of the attribute
  - Can be either continuous valued : ordered in a fashion meaningful to classification
    - such as age or income
  - Can be categorical : have no meaningful order
    - such as department names or country names

- Splitting continuous attributes:
  - Consider binary splits (Multiway splits is more complicated)
    - First sort the attribute values in the training instances.
    - Then compute the information gain obtained by splitting at each value
      - Example: if the training instances have values 1, 10, 15, and 25 for an attribute, the split points considered are 1, 10, and 15
      - In each case values less than or equal to the split point form one partition and the rest of the values form the other partition.
    - Best binary split for the attribute is the split that gives the maximum information gain.

Building Decision Tree Classifiers : Finding Best split

- Depends on the type of the attribute
  - Can be either continuous valued : ordered in a fashion meaningful to classification
    - such as age or income
  - Can be categorical : have no meaningful order
    - such as department names or country names

- Splitting Categorical attributes:
  - One child for each value of the attribute.
  - Works well with only a few distinct values, such as degree or gender
  - If the attribute has many distinct values
    - creating a child for each value is not a good idea
  - In such cases, try to combine multiple values into each child, to create a smaller number of children

Decision Tree construction Algorithm

- Evaluate different attributes and different partitioning conditions

- Pick the attribute and partitioning condition that results in the maximum information-gain ratio

- Works recursively
    - On each of the sets resulting from the split, recursively construct a decision tree

- The parameters δ_p and δ_s define cutoffs for purity and size

procedure GrowTree(S)
    Partition(S);

procedure Partition (S)
    if ($purity(S) > \delta_p$ **or** $|S| < \delta_s$) **then**
        return;
    for each attribute A
        evaluate splits on attribute A;
    Use best split found (across all attributes) to partition
        S into $S_1, S_2, \ldots, S_r$;
    for $i = 1, 2, \ldots, r$
        Partition(S_i);

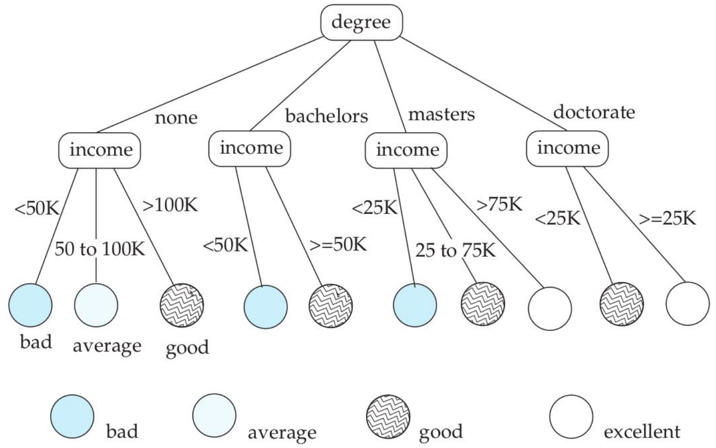
Decision Tree construction Algorithm

- Several of the algorithms also prune subtrees of the generated decision tree to reduce overfitting
  - A subtree is overfitted if it has been so highly tuned to the specifics of the training data that it makes many classification errors on other data
  - A subtree is pruned by replacing it with a leaf node
  - There are different pruning heuristics
    - uses part of the training data to build the tree and another part of the training data to test
  - The heuristic prunes a subtree if it finds that misclassification on the test instances would be reduced if the subtree were replaced by a leaf node

Decision Tree construction Algorithm

- We can generate classification rules from a decision tree
- For each leaf we generate a rule as follows:
  - Left-hand side is the conjunction of all the split conditions on the path to the leaf,
  - Class is the class of the majority of the training instances at the leaf.
- Example:

degree = masters and income > 75000 ⇒ excellent



Bayesian Classifiers

- Find the distribution of attribute values for each class in the training data

- When given a new instance d:
  - Use the distribution information to estimate, for each class c_j , the probability that instance d belongs to class c_j
  - The class with maximum probability becomes the predicted class for instance d

- Formula to find the probability of instance d belonging to class c_j

$$p(c_j|d) = \frac{p(d|c_j)p(c_j)}{p(d)}$$

  - $p(d \mid c_j)$ is the probability of generating instance d in the class c_j
  - $p(c_j)$ is the probability of the occurrence of the class c_j
  - $p(d)$ is the probability of occurrence of the instance d

Bayesian Classifiers

- Example:
  - One attribute, income, is used for classification; input person has income is 76000
  - Assume that income values are broken up into buckets and the bucket containing 76000 contains values in the range (75000, 80000)
  - Suppose
    - Among instances of class excellent, the probability of income being in (75000, 80000) is 0.1
    - Among instances of class good, the probability of income being in (75000, 80000) is 0.05
    - Among instances of class bad and average, the probability of income being in (75000, 80000) is 0
  - Suppose
    - Overall 0.1 fraction of people are classified as excellent
    - Overall 0.3 are classified as good.
- Then,
  - $p(d \mid c_j)\, p(c_j)$ for class excellent is 0.01
  - $p(d \mid c_j)\, p(c_j)$ for class good is 0.015

$$p(c_j|d) = \frac{p(d|c_j)p(c_j)}{p(d)}$$

- The person would therefore be classified in class good.

Bayesian Classifiers

- Calculating $p(d \mid c_j)$ with multiple attributes is complex
- With a limited training set used to find the distribution:
  - Most combinations would not have even a single training set matching them
  - Leads to incorrect classification decisions
- Naive Bayesian classifiers assume attributes have independent distributions

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \cdots * p(d_n|c_j)$$

  - Probability of the instance d occurring is the product of the probability of occurrence of each of the attribute values d_i of d, given the class is c_j

Bayesian Classifiers

- Benefits of Bayesian classifiers
    - They can classify instances with unknown and null attribute values
    - unknown or null attributes are just omitted from the probability computation.

- Decision-tree classifiers cannot handle situations where an instance to be classified has a null value for a partitioning attribute used to traverse further down the decision tree

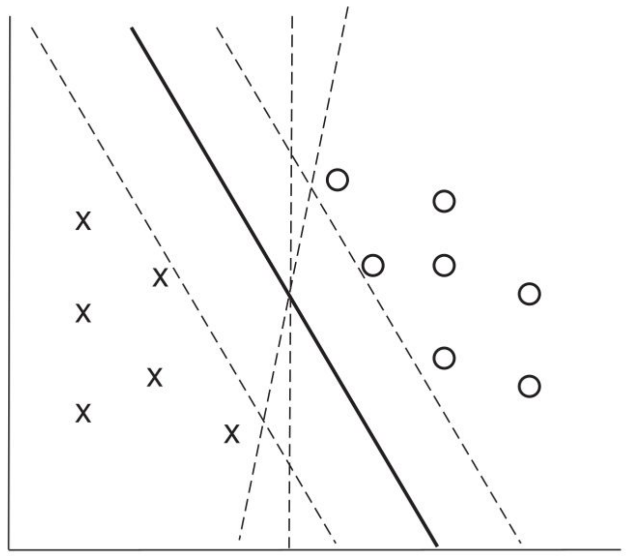
Support Vector Machines

- Gives accurate classification across a range of applications
  - Geometric classification

- In the simplest case, consider a set of points in a two-dimensional plane, some belonging to class A, and some belonging to class B

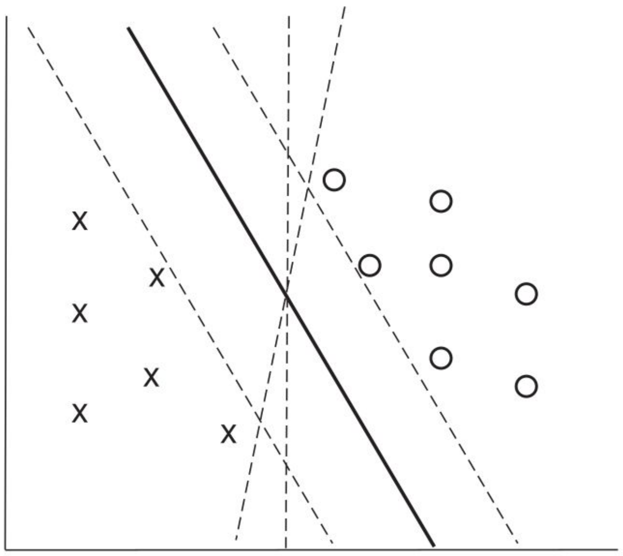
- We are given a training set of points whose class (A or B) is known, and we need to build a classifier of points, using these training points

- Draw a line on the plane, such that all points in class A lie to one side and all points in line B lie to the other
  - Which line to choose?



Support Vector Machines

- Choose the line whose distance from the nearest point in either class (from the points in the training data set) is maximum
    - maximum margin line
- Can be generalized to more than two dimensions, allowing multiple attributes to be used for classification
    - Classifier finds a dividing hyperplane
- Can find nonlinear separating curves also
- Can be used for non-binary classification too



Regression

- Deals with the prediction of a value (as supposed to classification)
- Given values for a set of variables $X_1, X_2, \ldots, X_n$
  - Predict the value of a variable Y
- Example : Treat the level of education as a number and income as another number
  - on the basis of these two variables, predict the likelihood of default
- Find a curve that fits the data (Curve fitting)
- Linear regression: Find values of a_o a_1 ..... a_n using the training data

$$Y = a_0 + a_1 * X_1 + a_2 * X_2 + \cdots + a_n * X_n$$

Validating a Classifier

- Measure the classification error rate, before deciding to use it for an application
- A set of test cases where the outcome is already known is used to measure the quality of the classifier
- For binary classifiers we can assume that input is classified into positive / negative
  - True positive (t_pos) : Positive instance classified as positive (correct decision)
  - False positive (f_pos) : Negative instance classified as positive (wrong decision)
  - True negative (t_neg) : Negative instance classified as negative (correct decision)
  - False negative (f_neg) : Positive instance classified as negative (wrong decision)
- Quality of a classifier: Let pos = t_pos + f_neg and neg = t_neg + f_pos
  - Accuracy : (t_pos + t_neg) / (pos + neg) fraction of time when classifier is correct
  - Recall : t_pos / pos how many of the actual positive classes are classified as positive
  - Precision : t_pos / (t_pos + f_pos) how often is the positive prediction correct
  - Specificity : t_neg / neg Dual notion of Recall

Validating a Classifier

- It is a bad idea to use exactly the same set of test cases to train as well as to measure the quality of the classifier
  - Can lead to artificially high measures of quality.
  - The quality of a classifier must measured on test cases that have not been seen during training
- A subset of the available test cases is used for training and a disjoint subset is used for validation.
- In cross validation, the available test cases are divided into k parts numbered 1 to k, from which k different test sets are created as follows:
  - Test set i uses the i-th part for validation, after training the classifier using the other k −1 parts.
  - The results from all k test sets are added up before computing the quality measures.
- Cross validation provides much more accurate measures than merely partitioning the data into a single training and a single test set

Association Rules

- Books frequently bought together
- Movies frequently watched together

- Bread => Milk

  (A person who buys bread has a high probability of buying milk)
  - Place them close in the store (customer time is reduced)
  - Place them far away (customer has to look at many other things before buying what he wants)

- An association rule must have an associated population

  The population consists of a set of instances.
  - In the example, population is the grocery purchases, every purchase being an instance

Association Rules

- Rules have a support and confidence
  - Support : Measure of what fraction of the population satisfies both the antecedent and the consequent of the rule
    - 0.001 percent of purchases include milk and stapler
      (so there is very less support for Milk => Stapler)
    - Minimum degree of support is considered desirable depends on the application
  - Confidence : Measure of how often the consequent is true when the antecedent is true
    - if 80 % of the purchases that include milk also include bread
      (Milk => Bread has 0.8 confidence)
    - Rule with a low confidence is not meaningful
    - In business applications, rules usually have confidences significantly less than 100 %
    - In other domains, such as in physics, rules may have high confidences.

Association Rules

- How to discover association rules of the form
    - $i_1 i_2 \dots i_n \Rightarrow i_0$
- First find items with sufficient support, called large itemsets
- For each large itemset output all rules with sufficient confidence that involve all and only the elements of the set.
    - For each large itemset S, output a rule $S - s \Rightarrow s$ for every subset $s \subset S$
        - If $S - s \Rightarrow s$ has sufficient confidence
    - the confidence of the rule is given by support of s divided by support of S

Association Rules

- How to generate large itemsets
  - If the number of possible sets of items is small
    - Single pass over the data suffices
    - A count, initialized to 0, is maintained for each set of items
    - When a purchase record is fetched, the count is incremented for each set of items such that all items in the set are contained in the purchase.
  - Example : If a purchase included items a , b, and c
    - counts would be incremented for {a }, {b}, {c}, {a , b}, {b, c}, {a , c}, and {a , b, c}.
    - Those sets with a sufficiently high count at the end of the pass correspond to items that have a high degree of association

  - The number of sets grows exponentially,
  - But most of the sets would normally have very low support
    - Optimizations have been developed to eliminate most such sets from consideration.
    - Use multiple passes on the database, considering only some sets in each pass.

Association Rules

- How to generate large itemsets
    - For generating large itemsets
        - only sets with single items are considered in the first pass.
        - In the second pass, sets with two items are considered, and so on
    - At the end of each pass, all sets with sufficient support are output as large itemsets.
    - Sets found to have too little support at the end of a pass are eliminated
    - Once a set is eliminated, none of its supersets needs to be considered
        - In pass i count only supports for sets of size i such that all subsets of the set have been found to have sufficiently high support
        - Suffices to test all subsets of size i – 1 to ensure this property.
    - At the end of some pass i, if no set of size i has sufficient support then computation terminates.

Other types of Association

- Many associations are not very interesting
    - They can be predicted
    - Example : if many people buy shoes and many people buy milk, we can predict that a fairly large number of people would buy both, even if there is no connection between the two purchases
- Is there is a deviation from the expected co-occurrence of the two?
    - Look for correlations between items;
        - correlations can be positive
    - There are standard measures of correlation used in statistics

Other types of Association

- Sequence associations (or sequence correlations) in Time-series data
  - Stock prices on a sequence of days
    - Find associations among stock-market price sequences.
      - “Whenever bond rates go up, the stock prices go down within 2 days”

- Find deviations from what one would have expected on the basis of past temporal or sequential patterns
  - If sales of winter clothes go down in summer, it is not surprising

Clustering

- Finding clusters of points in the given data
  - Grouping points into k sets (for a given k) so that the average distance of points from the centroid of their assigned cluster is minimized
  - Grouping points so that the average distance between every pair of points in each cluster is minimized

- Scalable clustering algorithms can cluster very large data set (Birch clustering algorithm):
  - Data points are inserted into a multidimensional tree structure (based on R-trees)
  - Nearby points are clustered together in leaf nodes
    - summarized if there are more points than fit in memory
  - Result of first phase of clustering is to create a partially clustered data set that fits in memory
  - Standard clustering techniques can be executed on the in-memory data to get the final clustering

Clustering

- Example of clustering : Predict what new movies a person is likely to be interested in
  - The person’s past preferences in movies
  - Other people with similar past preferences
  - The preferences of such people for new movies

- Find people with similar past preferences
- Create clusters of people based on their preferences for movies
  - The accuracy can be improved by previously clustering movies by their similarity
    - If people have seen similar movies they would be clustered together.

- Given a new user, find a cluster of users most similar to that user
  - On the basis of the user’s preferences for movies already seen.
  - Predict movies in movie clusters that are popular with that user’s cluster as likely to be interesting

Other forms of Data Mining

- Text mining applies data-mining techniques to textual documents.
    - Form clusters on pages that a user has visited
    - Helps users when they browse the history of their browsing to find pages they have visited earlier

- Data-visualization systems help users to examine large volumes of data, and to detect patterns visually.
    - Maps, charts ...
    - Example: if the user wants to find out whether production problems at plants are correlated to the locations of the plants
        - Problem locations can be encoded in a special color on a map.
        - User can quickly discover locations where problems are occurring
        - The user may form hypotheses about why problems are occurring in those locations, and verify the hypotheses quantitatively against the database.

    - Data-visualization systems do not automatically detect patterns
        - Provide support for users to detect patterns

Reference:

Database System Concepts by Silberschatz, Korth and Sudarshan

(6th edition)

Chapter 20