Chapter 4: Algorithms

CS 795
Inferring Rudimentary Rules

• 1R – Single rule – one level decision tree
  – Pick each attribute and form a single level tree without overfitting and with minimal branches
  – Pick that attribute that results in minimum total errors
• Missing values – Treat its as another attribute value (nominal)
• Numeric attributes---Sort the values and partition it into discrete ranges (i) Expect minimum # of occurrences to avoid overfitting (ii) Take a major class for each data partition
Bayes’ Rule: If you have a hypothesis $H$ and evidence $E$ that bears on that hypothesis, then

$$
\Pr[H|E] = \frac{\Pr[E|H] \cdot \Pr[H]}{\Pr[E]}
$$

For example, if $H$: “Play will be yes” and $E$: weather conditions (outlook ($E_1$), temperature ($E_2$), humidity ($E_3$), windy ($E_4$)) on a day, then $\Pr[H|E]$ is the probability that they would play today.

Let $E = \{\text{outlook=sunny, temp = cool, humid = high, windy = true}\}$

$$
\Pr[\text{yes}|E] = \Pr[E_1|\text{yes}] \cdot \Pr[E_2|\text{yes}] \cdot \Pr[E_3|\text{yes}] \cdot \Pr[E_4|\text{yes}] \cdot \Pr[\text{yes}] / \Pr[E] = \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} / \Pr[E] = 0.005291 / \Pr[E]
$$

$$
\Pr[E] = \frac{5}{14} \cdot \frac{4}{14} \cdot \frac{7}{14} \cdot \frac{6}{14} = 0.02187
$$

$$
\Pr[\text{yes}|E] = 0.242 \text{ or 24%}
$$

$$
\Pr[\text{no}|E] = 76% 
$$

It is called Naïve Bayes rule because it naively assumes that all attributes are independent.
Problems with Naïve Bayes

• If the attributes are not all independent, the conclusion is not valid.
• If the training data does not capture all values for an attribute, the computed probability would be zero---e.g., if outlook=sunny never appears in the training data, then $\text{prob}(\text{sunny}|\text{play}=\text{yes})$ would be zero, making the probability zero independent of other attribute values.
Bayes’ Method – cont.

- Missing values –
  - if on the day that we are testing the hypothesis, a particular attribute value (say outlook is not known), then calculations simply omit the outlook variable [This is because of normalization.]
  - If an attribute value is missing in the training instance, it is simply not included in the frequency count
Bayes’s Theorem

• Numerical attributes
  – Assume normal distribution
  – Compute mean and standard deviation for each numeric attribute when outcome is yes and when outcome is no.
  – Use the pdf given in page 93 and 94 -- example
Bayesian Model for Classification

- Naïve Bayes is popular here
- Suppose \( n_1, n_2, \ldots, n_k \) is the #of times words 1,2,\ldots, k, occur in a document, respectively. Let \( N = n_1+n_2+\ldots+n_k \)
- Let \( P_1, P_2, \ldots, P_k \) be the prob that words 1,2,\ldots,k occur in a document of class \( H \)
- Assuming that these words occur independently, we have a multinomial distribution
- \( \Pr[E|H] = N! \prod(P_i^{n_i}/n_i!) \)
- Example: Only two words: \{yellow, blue\}
- Suppose we are told in a document class \( H \), \( \Pr(\text{yellow}|H)=0.75; \  \Pr(\text{blue}|H)=0.25 \)
- If a document \( E \) has 3 words \{blue yellow blue\}: \( n_1=1; \ n_2=2; \ N=3 \)
- There are 4 possible bags of 3 words each with yellow and blue: \{yyy\}, \{yyb\}, \{ybb\}, \{bbb\}
- \( \Pr[\text{yyy}|H]=3!*0.75^3/3!; \  \Pr[\text{yyb}|H]=3!*(0.75^2/2!)*(0.25/1); \  \Pr[\text{ybb}|H]=3!*(0.75)*(0.25^2/2!); \  \Pr[\text{bbb}|H]=3!(0.25^3)/3! \)
- So the prob that the given document belongs to class \( H \) is \( \Pr[\text{ybb}] = 9/64 \) or 14\% (\( \Pr[\text{yyy}|H]=0.42; \  \Pr[\text{yyb}|H]=0.42; \  \Pr[\text{ybb}|H]=0.14; \  \Pr[\text{bbb}|H]=0.02; \  \Pr[\text{ybb}]=14\% \))
Divide-and-conquer: Constructing Decision Trees

- Select an attribute to place at the root node and make branches for each possible value. (works well for nominal values)
- Repeat this process for each attribute
- Any time all instances at a node are of the same class, stop that part of the tree.
- Question: determining the attribute order in which the tree could be built?
- Objective: Small trees are preferred to large trees
- Measuring information (in bits): Expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node.
- Info([2,3])---information value for a node that has 2 yes and 3 no nodes under it--- \((-2/5)\log_2{2/5} + (-3/5)\log_2{3/5} = 0.971\)
• Outlook → sunny info([2,3]) = 0.971 bits
• Outlook → overcast info([4,0])=0.0 bits
• Outlook → rainy info([3,2]) = 0.971 bits
• info([2+4+3,3+0+2])=info([9,5])=0.940
• info([2,3],[4,0],[3,2]) = 5/14*info([2,3])+4/14*info([4,0])+5/14*info([3,2]) = 0.693 bits
• gain(outlook) = 0.940-0.693 = 0.247 bits
• Similarly, gain(temperature)=0.029 bits
• Gain(humidity)=0.152 bits
• Gain(windy)=0.048 bits
• Hence, select outlook as the root of the decision tree.
• Repeat this process at each of the other nodes; E.g., outlook→sunny:
gain(temperature)=info([2,3])-info([0,2],[1,1],[1,0]) = 0.971-0.4=0.571 bits
How to calculate the information measure?

- Entropy \((p1,p2,\ldots,p_n)=-p_1\log p_1-p_2\log p_2-\ldots-p_n\log p_n\) where \(p_1, p_2, \ldots, p_n\) are the fraction of occurrence of value \(x_i\) for \(X\).

- For example, if a random variable \(X\) can have three values: blue, red, green. If \(X=\text{blue}\) has prob. of \(\frac{1}{2}\), \(X=\text{red}\) is \(\frac{1}{3}\), and \(X=\text{green}\) is \(\frac{1}{6}\); then Entropy of \(X\) is:
  
  \[-\frac{1}{2}\log(1/2)-\frac{1}{3}\log(1/3)-\frac{1}{6}\log(1/6) \quad [\log \text{ is base } 2] \approx 0.5+0.528+0.431 = 1.459 \text{ bits}\]

So info([2,3,4]) = information where 2 instances are of one value, 3 of 2nd, and 4 of the 3rd.

Info([2,3,4])=entropy(2/9,3/9,4/9) = \(-2/9\log(2/9)-3/9\log(3/9)-4/9\log(4/9)\)

Problem with selecting maximum gain attribute---see Table 4.6 example

Solution:
Covering Algorithms: Constructing Rules

- Alternate to divide-and-conquer; it is referred to as separate-and-conquer—"identify a rule that covers many instances in the class (and excludes ones not in the class), separate out the covered instances because they are already taken care by the rule, and continue the process on those that are left."
- Objective: Produce effective rules
- Approach: Take each class (e.g., yes, no) in turn and seek a way of covering all instances in it and excluding those not in the class
- See Fig 4.6 – example
Simple covering algorithm:

- Choose a rule that includes as many instances of the desired class as possible and exclude as many instances of other classes as possible---if the new rule covers a total of $t$ instances of which $p$ are of the desired class, then $p/t$ is the desired ratio; maximizing $p/t$ is the objective.

- E.g., Table 1.1: If we are considering hard lenses as a recommendation rule, age=young covers: 8 instances out of which 2 are hard (recommended lenses); so $p/t = 2/8$.

- Similarly, sunny=hot covers 3 instances out of which 1 is yes; $p/t = 1/3$; since this is still not accurate, we have to add a conjunctive rule: windy=true with $p/t=1/1$. Thus the final rule covering just this one is: sunny=hot and windy=true then play = yes.
Rules vs. trees

• Trees must select one attribute at a time to split; rules have no such restriction ➔ Trees can be larger than an equivalent set of rules
• Rule-generating method concentrates on one class at a time (say play-yes); a decision tree split takes all classes (e.g., yes and no) into account

Rules vs. decision lists
The rules developed above must be executed in order just as in a decision list.
Mining Association Rules

- Find association among the attributes
- Use a divide-and-conquer rule-induction procedure as in classification rules
- Coverage: Number of instances for which the conditions in a rule are true. E.g., if A>10 and B>5 $\rightarrow$ color=Red; then if there are 15 instances for which A>10 and B>5, then coverage is 15.
- Accuracy: Out of the 15, suppose color=Red only for 10, then accuracy = 10/15 or 67%
- Item sets: Fig. 4.10 (An attribute-value pair is an item)
  - One-item sets: (3 for outlook + 3 for Temp + 2 for Humid + 2 for Windy + 2 for Play = 12 sets)
  - Two-item sets: Take pairs of attributes: (outlook=sunny and Temp=mild)
  - Coverage and accuracy: Accuracy = 100%; minimum coverage = 2 (for example)
Mining Association Rules (cont.)

• Generating rules efficiently: Input: Data and minimum coverage accuracy

• Generate all 1-item sets with given minimum coverage
  – Use this to generate two-item sets, three-item sets, …
  – For example, suppose we have A, B, C, D as the initial 1-item set where A to D are conditions such as outlook=sunny
  – Check if (A B), (A C), (A D), (B C), (B D), and (C D) satisfy the coverage and accuracy.
  – Suppose (A B) (B C) and (C D) are selected.
  – Then check for (A B C) and (B C D) as the 3-item sets.
  – If both these are selected, check for (A B C D) as the 4-item set.

• Turn each into a rule, or set of rules, with at least the specified minimum accuracy. Some item sets may produce more than one rule; others may produce none.
Mining Association Rules (cont.)

• Example: (Humidity=normal, windy=false, play=yes) has a coverage of 4. So we will consider it.
• It leads to seven potential rules:
• LHS $\rightarrow$ RHS
  2 $\rightarrow$ 1; 1 $\rightarrow$ 2; 0 $\rightarrow$ 3
  3 ways + 3 ways + 1 = 7 ways
• Example: If humidity=normal and windy=false $\rightarrow$ play = yes
  Coverage =4 (instances where all three conditions are true); instances for which the antecedent is true= 4; accuracy = 4/4= 100%

However, for play =yes $\rightarrow$ humidity=normal and windy=false, instances for which play=yes is 9; so accuracy = 4/9.
See Page 117 for generating rules efficiently
Key --- what is the minimum coverage specified for a rule to be viable
Linear Models

- Relevant for numeric valued attributes
- Numeric prediction with **Linear regression**: \( x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \). Now the problem is to determine the weights so that the MSE is minimized
- Linear classification using linear regression: Perform a regression for each class (e.g., play=yes, or play=no), setting the output equal to 1 for training instances that belong to the class and 0 for others. This results in a linear regression for each class. Given a test example of unknown class, calculate the value of each expression and choose the one that is largest. This is Multi-response linear regression.
- Problem: Least square regression assumes that errors are statistically independent and normally distributed --- this may not be true in the classification case
Linear classification using perceptron

- Applicable where data is separable by a hyperplane (linearly separable)
- Hyperplane eqn: $w_0a_0 + w_1a_1 + \ldots + w_k a_k = 0$ ($a_0$ always has the value 1)
- If the sum is $> 0$, we will predict the unknown to be in class 1. Otherwise in class 2. Problem is to find $w_0, w_1, \ldots, w_k$ given the training instances.
- See Fig. 4.10a (page 125)
- After the corrections, the resulting hyperplane is called the perceptron Demo
Linear classification using Winnow

- For datasets with binary attributes, Winnow is the alternate tool---similar to perceptron---it is also mistake driven
- Winnow employs multiplicative method updates and alters weights individually by multiplying them by a user specified parameter $\alpha$ or $1/\alpha$
- If the attribute value is 0 --- nothing is done
- If attribute value is 1, then the weight is multiplied by $\alpha$ if it helps to make a correct decision and $1/\alpha$ if it does not.
- In addition, the user also specifies a threshold, $\theta$
- An instance belongs to class 1 if $w_0a_0+w_1a_1+\ldots+w_ka_k > \theta$
Instance-based Learning

- Idea is to store the training set as it is;
- Given an unknown instance, determine the instance that is closest to it and assign it to the class that the instance belongs
- We need to define a distance metric---Euclidean distance is the most commonly used one; Manhattan distance is another metric
- First, we need to normalize each attribute value so that they are between 0 and 1: \( a_i = \frac{[x_i - \min(x_i)]}{(\max(x_i) - \min(x_i))} \);
- For nominal values, the values are either 0 (same) or 1 (different); if values are missing, the difference is as large as it can possibly be.
- For missing values, for nominal attributes, assume maximum difference (i.e., 1); for numerical attributes, if both are missing, the difference is taken as 1; otherwise take it as the maximum possible difference given one value.
Finding nearest neighbors efficiently

• The naïve procedure to find the nearest neighbor is $O(n)$;
• kD-tree is one way to organize the training set as a tree:
  – This is a binary tree that divides the input space with a hyperplane and then splits each partition again, recursively. $K$ is the number of attributes and represents the dimensionality of the tree
  – All splits are made parallel to one of the axes---either vertically or horizontally in a 2D case.
  – It stores a set of points in k-dimensional space, where $k$ is # of attributes.
• Critical aspects---training points to split on and the direction (vertical or horizontal of the split)
• To find a good direction for the split---calculate the variance of points along each axis individually and select the axis with the greatest variance and create a splitting hyperplane perpendicular to it.
• To find a good place for the hyperplane, locate the median value along that axis and select the corresponding point.
Illustration of kD-tree

- K=2; Seven Points: (3,8), (3,4), (6,7), (2,2), (4,1), (7,5), (9,2)
- Variation along X-axis: 6.47; along Y-axis: 7.14;
- So split along axis perpendicular to Y-axis --- i.e., horizontally.
- Median value is (3,4) along the Y-axis. So (3,4) horizontal is the first split.
- To the left we will have the ones with y < 4; so (2,2), (4,1), (9,2); To the right we have (3,8), (6,7), (7,5)
- Now take the left tree: Variance along x-axis is: 13; along y-axis is 0.33; so select X-axis and split along perpendicular to X-axis; so vertically. Median along X-axis is (4,1), So we will have one to its left (2,2) and one to the right (9,2)
- Similarly, the right tree: Variance is largest along X-axis; so we choose a vertical split with a median point of (6,7)
- http://homes.ieu.edu.tr/~hakcan/projects/kdtree/kdTree.html
Clustering

- Applicable when the instances have to be simply divided into natural groups---census and population.
- Iterative distance-based clustering
  - K-means algorithm where k is the number of clusters we seek to form
  - Choose k points at random as cluster centers
  - All instances are assigned to their nearest cluster center using Euclidean distance
  - Calculate the centroid of the instances in each cluster; assign the centroid as the new cluster center
  - Repeat the above process until the clusters are stabilized
• Overall objective is to minimize the distance of the instances from their centers --- locally optimal
• Main problem---final clustering may depend on the initial random choice of k points as cluster centers; requires several iterations with several initial start points
• Faster distance calculations---using kD-trees