CS4445 Data Mining and Knowledge Discovery in Databases. B Term 2012

Solutions Exam 2 - December 11, 2012

By Prof. Carolina Ruiz
Department of Computer Science
Worcester Polytechnic Institute

Problem I: **(/35 points)** Rule-Based Classification

Problem II: **(/25 points)** Association Analysis

Problem III: **(/30 points)** Clustering Analysis

Problem IV: **(/20 points)** Anomaly Detection

TOTAL SCORE: **(/100 points)**

Instructions:

- Show your work and justify your answers
- Use the space provided to write your answers
- Ask in case of doubt

Problem I. Rule-Based Classification [35 Points]

Consider the following general sequential covering algorithm used to construct classification rules:

Sequential Covering Algorithm:

1. Let D be a dataset of training data instances with *n* predictive attributes $A_1,\ldots A_n$, and a target attribute C with possible values $c_1,\ldots,c_k$.
2. Let RuleSet = {} be the initial rule list.
3. **for** each class c_i in C **do**
4. **while** stopping criterion is not met **do**
5. R ← Learn-One-Rule(D,c_i)
6. D ← D – data instances covered by R (i.e., remove training data instances from D that are covered by R)
7. RuleSet ← RuleSet U R (i.e., add R at the bottom of the rule list in RuleSet)
8. **end-while**
9. **end-for**

Answer the following questions:

1. **[5 Points]** In what order does the RIPPER algorithm consider the class values $c_1,\ldots,c_k$ in line 3 when constructing rules? Explain.

RIPPER considers the class values in increasing order of frequency. That is, it considers (i.e., constructs rules for) the least frequent class first. Note also that RIPPER constructs rules for each of the *k-1* least frequent classes, and uses a default rule for the most frequent (majority) class.

2. Consider the LearnOneRule function in line 5 of the algorithm.
   2.1. **[2 Points]** What rule for c_i does the LearnOneRule function in RIPPER start from? Show the rule.

It starts from the most general rule for c_i , namely {} → c_i

   2.2. **[5 Points]** If LearnOneRule decides to refine the current rule by adding a condition to the left-hand side of the rule, what candidate conditions does it consider? Explain.

It considers all possible attribute value pairs as candidate conditions. (Note that it's enough to consider all attribute value pairs that occur in data instances of class c_i.)

   2.3. **[8 Points]** What metric does RIPPER use to select which among the candidate conditions to add to the rule? Explain the notation in your formula.

It uses FOIL's information gain: $p_1 \times \left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right)$ where:

p_0 (resp. p_1) is the number of positive data instances (i.e., data instances with class = c_i) covered by the rule before (resp. after) adding the candidate condition.

n_0 (resp. n_1) is the number of negative data instances (i.e., data instances with class ≠ c_i) covered by the rule before (resp. after) adding the candidate condition.

   2.4. **[5 Points]** What termination criteria does RIPPER use to stop refining a rule?

RIPPER stops adding conditions to a rule when: (1) the information gain of adding the best candidate condition is below a given threshold; (2) the rule covers only positive examples (note that this criterion is a particular case of the 1st criterion); or (3) there are no more candidate conditions available (i.e., RIPPER runs out of attribute value pairs to add to the rule).

3. **[5 Points]** What does it mean for a data instance to be covered by rule R in line 6 of the algorithm? Explain.

It means that the data instance satisfies the antecedent (left-hand side) of the rule.

4. **[5 Points]** What stopping criterion does RIPPER use in line 4 of the algorithm above?

RIPPER stops constructing rules for class c_i when: (1) no data instances with class=c_i remain in dataset D; (2) the rule to be added to RuleSet increases the minimum description length of RuleSet by at least a given number of bits; or (3) the error rate of RuleSet over the validation set is greater than 50%.

Problem II. Association Analysis [25 Points]

Consider the Apriori algorithm that constructs association rules.

1. **[5 Points]** Is the Apriori algorithm a sequential covering method? If your answer is yes, explain why. If you answer is no, describe 3 differences between the Apriori algorithm and a sequential covering algorithm.

No, Apriori is not a sequential covering algorithm. Here are just a few differences:

- Apriori constructs association rules, not classification rules
- Apriori constructs rules in parallel (through the construction of frequent itemsets), rather than one by one.
- Apriori doesn't attempt to generate rules that cover all data instances.
- Apriori doesn't remove data instances once that they are covered by a rule.
- Apriori can generate rules that have several conjuncts in the consequent (right hand side) of the rule.

2. **[5 Points]** State the Apriori Principle.

Here are 2 equivalent formulations of the Apriori principle:

- If an itemset is frequent, then all its subsets are frequent.
- If an itemset is infrequent, then all its supersets are infrequent.

3. **[5 Points]** Describe 2 places in the Apriori algorithm where the Apriori Principle is used.

- The level-by-level construction of frequent itemsets uses the Apriori principle to remove from consideration supersets of itemsets when these itemsets are known to be infrequent.
- Candidate generation (join condition) prevents generating some itemsets that have infrequent subsets.
- Candidate pruning (subset condition) removes from consideration all candidate itemsets that have infrequent subsets.

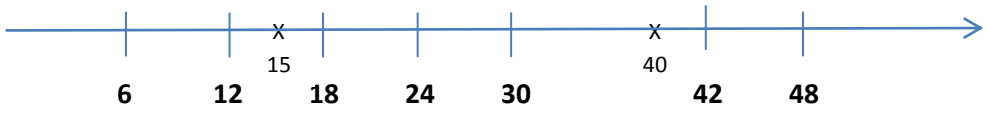
4. **[10 Points]** The traditional version of the Apriori algorithm receives three inputs: A dataset of instances, a minimum support threshold, and a minimum confidence threshold. It then outputs all association rules over the dataset whose support and confidence are no lower than the given thresholds. However, the Apriori algorithm implemented in Weka does not follow this specification. Describe the process followed by Weka's Apriori algorithm precisely and the output (rules) that it produces in terms of its input parameters: *dataset of instances, minimum confidence, number of rules, upper bound for minimum support, lower bound for minimum support*, and *delta*.

Weka's Apriori mines association rules with min. support equal to the *upper bound for minimum support* and the given *minimum confidence*. If it obtains at least the required *number of rules*, it outputs this rules and stops. If it obtains less than the required *number of rules*, it decreases the min. support by *delta.* It repeats this process until it obtains enough rules or the min. support reaches the given *lower bound for minimum support.*

Problem III. Clustering Analysis [30 Points]

Consider the following dataset of one-dimensional instances: {6, 12, 18, 24, 30, 42, 48}.

1. **[9 Points]** Using 15 and 40 as the initial centroids, follow the K-means algorithm to partition the dataset into 2 clusters. Show your work.



Distance to 15:	9	3	3	9	15	27	33
Distance to 40:	34	28	22	16	10	2	8
Goes to cluster:	C1	C1	C1	C1	C2	C2	C2

Clusters after the first iteration of K-Means: C1 = {6, 12, 18, 24} and C2 = {30,42, 48}

New centroids of these clusters: C1: (6+12+18+24)/4 = 15; and C2: (30+42+48)/3 = 40

Since the centroids didn't change, K-Means stops and outputs these 2 clusters.

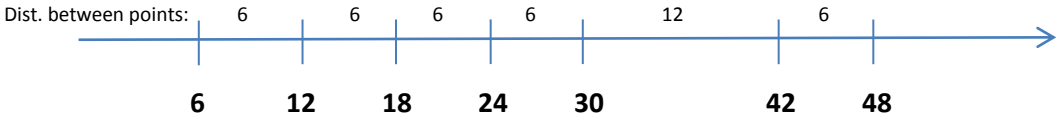
2. **[9 Points]** Calculate the SSE of the clustering you obtained above. Show your work.

SSE for Cluster C1: $(6 -15)^2 + (12 -15)^2 + (18 -15)^2 + (24 -15)^2 = 9^2 + 3^2 + 3^2 + 9^2 = 81 + 9 + 9 + 81 = 180$

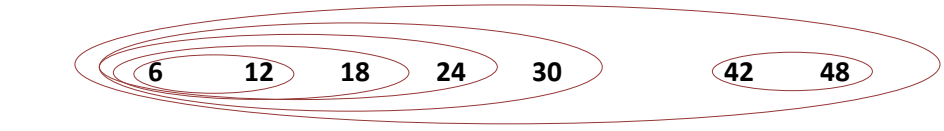
SSE for Cluster C2: $(30 - 40)^2 + (42 - 40)^2 + (48 - 40)^2 = 10^2 + 2^2 + 8^2 = 100 + 4 + 64 = 168$

Total SSE of the clustering: 180 + 168 = 348

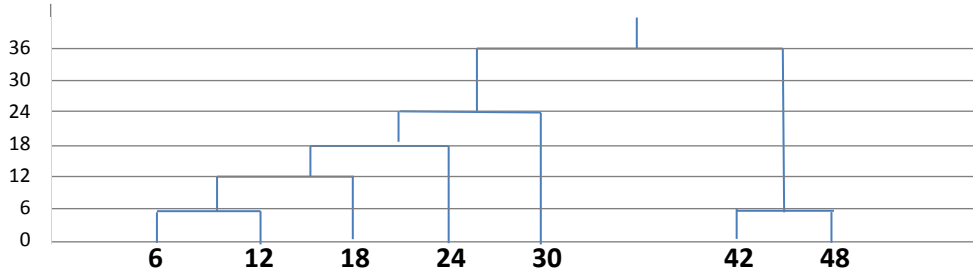
3. **[9 Points]** Follow the single link (= MIN) version of hierarchical clustering to cluster this dataset. Show your work and the resulting dendrogram.



Here we'll find several ties when choosing the pair of points/clusters with min distance. I'll select the 1st pair (from left to right) when breaking the tie, but you could select another pair with same min distance:



Clustering process followed by single link



Resulting dendrogram

4. **[4 Points]** Assume that you want to cluster the dataset into exactly 2 clusters. Which 2 clusters would single link clustering output from the dendrogram above? C1: {6,12,18,24,30} , C2: {42,48}.

Problem IV. Anomaly Detection [20 Points]

1. **Unsupervised anomaly detection.** The typical unsupervised approach to detect anomalies in a dataset receives as input an anomaly scoring function *f(x)* and a threshold *t*, and it returns all the data instances in the dataset with anomaly score greater than *t*.

   1.1. **[5 Points]** Given a scoring function *f(x),* describe a method to determine a good value for the threshold *t*.

Here are several possible answers to this problem:

- Plot the values of *f(x)* sorted in increasing order and find a knee in the plot. Take *t* = *f(x)* of the knee.
- Or look at the distribution of *f(x)* values, and select:
  - a certain (small) percentage of the top values. Take *t* = the smallest *f(x)* among these values.
  - or a certain (small) number of the top values. Take *t* = the smallest *f(x)* among these values.
  - or a certain number *k* of standard deviations σ over the mean. Take *t* = mean *f(x)* + *k* σ.

   1.2. **[5 Points]** Provide an example of a specific scoring function *f(x),* and state if it can be used for statistical-based, proximity-based, clustering-based, or density-based anomaly detection.

You can select any *f(x)* discussed in class or in the textbook.
For example, for clustering-based anomaly detection, take *f(x)* = *distance(x, c_x),* where c_x is the closest centroid to *x* in the clustering.

2. **Supervised anomaly detection.** Assume that the dataset contains a Boolean attribute "*isAnomaly?*" with value "yes" if the data instance is an anomaly and "no" if it is not.

   2.1. **[5 Points]** Describe how you would determine if a given new data instance (for which you don't know its *isAnomaly?* value) is an anomaly or not. Explain your answer.

Construct a classifier (e.g., decision tree, Bayesian net, classification rules, …) using *isAnomaly?* as the target attribute. Then use the classifier to classify (= predict the class of) the given new data instance.

   2.2. **[5 Points]** One expects anomalies to be relatively rare (i.e., infrequent) with respect to normal instances in a dataset. How does this affect the construction and/or the evaluation of the supervised method you proposed above? Explain.

Since likely there are much fewer anomalies than normal data instances in the dataset, the classification model constructed over the training set may end up preferring the "no" class over the "yes" class, as this strategy would tend to increase its accuracy. In other words, the model constructed might end up being equivalent to ZeroR. Or even if the model is not ZeroR, the evaluation of the model using accuracy might not be appropriate. Some approaches to deal with this issue (see also Section 5.7 of the textbook on class imbalance):

During Construction of the model:

- Sample the dataset to make the frequencies of the two classes more similar.
- Don't allow for pruning of the model to avoid removing parts of the model that deal with the most infrequent class.
- If the model allows, assign higher (resp. lower) weights to anomaly (resp. normal) instances.

During Evaluation of the model:

- Use metrics other than accuracy. For instance, precision and recall.
- Use cost-sensitive classification. That is penalize misclassifications (true positive and false positive) differently.