CIS600 Fundamentals of Data and Knowledge Mining

Final Exam

NAME Anthony Redamonti

SU ID 253765329

Problem 1 (10 pts): Please circle the right variable type for the below attributes.

Celsius temperature values in ^oC	[nominal / ordinal / interval / ratio] INTERVAL
Weight in pounds	[nominal / ordinal / interval / ratio] RATIO
Birth date	[nominal / ordinal / interval / ratio] INTERVAL
SU ID	[nominal / ordinal / interval / ratio] NOMINAL
Income level [high/middle/low income]	[nominal / ordinal / interval / ratio] ORDINAL

Problem 2 (28 pts): Answer the following questions by circling the most likely choice. (2pts for each question)

[TRUE / FALSE] Decision tree algorithm (without pruning) is sensitive to noisy data. TRUE

[TRUE / FALSE] Naïve Bayes method is intolerant of irrelevant attributes. FALSE

[TRUE / FALSE] Type II errors, instead of type I errors, should be minimized when design a spam email detection algorithm (where spams are considered as the positive examples). FALSE

[TRUE / FALSE] Holdout method tends to produce more stable estimate of the model performance accuracy than cross validation method. FALSE

[TRUE / FALSE] Convenience sampling is one type of probability sampling method. FALSE

[TRUE / FALSE] Artificial neural network can be successfully trained using gradient descent for global optimum. TRUE

[TRUE / FALSE] K-means clustering algorithms can find clusters of arbitrary shape. FALSE

[TRUE / FALSE] Assume everything else the same, the decision tree induction algorithm produces predictions with higher variance than the Random Forest algorithm. TRUE

[TRUE / FALSE] Lift corrects for high confidence in rule X-> Y when item X is bought regularly by customers. TRUE

[TRUE / FALSE] The best centroid for minimizing the SSE of a cluster is the mean of the points in the cluster. TRUE

[TRUE / FALSE] Apriori principle indicates if an itemset is infrequent, then all of it subsets must also be infrequent. FALSE

[TRUE / FALSE] ROC (Receiver Operating Curve) is generated by plotting sensitivity (y-axis) against specificity (x-axis). FALSE

[TRUE / FALSE] K Nearest Neighbor is considered as a non-parametric method. TRUE

[TRUE / FALSE] Boosting method produces an ensemble of classifiers through random sampling the training data set with replacement. TRUE

Problem 3 (10 pts): Explain why decision tree algorithm based on impurity measures such as entropy and Gini index tends to favor attributes with larger number of distinct values. How would you overcome this problem?

The decision tree induction algorithm is a supervised learning algorithms that is generally robust to noisy data. It is used in classification and regression analysis. The decision tree algorithm is also a greedy algorithm that is going try to maximize the information gain when deciding which attributes to split when building the tree. Greedy algorithms do not guarantee a globally optimum solution. However, the locally optimum solution usually yields acceptable results.

The whole point of building a decision tree is to reduce the data impurity so that the dataset will be continually purified as the tree is built. Attributes with the highest information gain will be those that have a large number of distinct values. Therefore, these attributes will be favored by the greedy decision tree algorithm when building the tree.

Data purity can be measured using the Entropy, defined below (assuming binary classification).

$$Entropy(Dataset, Class\ Label) = -\sum_{i}^{n} P_i \log_2 P_i$$

P represents the probability of the class. The sum of the probability of the positive and negative classes is multiplied by -1 because the range of entropy must be between 0 and 1. An entropy of 0 represents a perfectly pure class label in the dataset (probability = 1; $\log_2(1) = 0$), meaning that all the data in the dataset *with respect to the class label* is the same. If all the data in the dataset with respect to the class label is unique, the entropy would be 1. The Gini Index is a similar measurement of the purity of the dataset with respect to a class label, but the range is between 0 and 0.5.

Information gain is calculated using entropy:

$$Information\ Gain(Attribute, Dataset, Class\ Label) = Entropy(D_{BeforeSplitting}) - Entropy(D_{AfterSplitting})$$

Information gain is the differential between the entropy of the dataset before and after splitting. But what if the attribute with high information gain does not make sense to keep in the decision tree: i.e. the SU ID of every student at Syracuse University? These types of attributes will have a high information gain but will obviously lead to overfitting.

We can overcome this issue by adjusting the information gain using the gain ratio. The gain ratio is the ratio of information gain to the intrinsic information.

$$GainRatio(Attr) = \frac{InformationGain(Attr)}{-\sum_{j}^{m} \frac{n_j}{n} \log_2 \frac{n_j}{n}}$$

The gain ratio divides the information gain by a formula that considers the possible number of values. As the possible number of values of an attribute increases, the gain ratio decreases. By considering the gain ratio, the algorithm is no longer bias towards attributes with a high number of distinct values.

Problem 4 (10 pts):

(a) Explain what is the *anti-monotone property* of the support measure and how you can incorporate this property directly into the mining algorithm to effectively prune the exponential search space of candidate itemsets. (5pts)

As a percentage, support is defined as the percent of the transactional dataset that includes both the antecedent and consequent of the association rule. The range of support is 0 to 1. Intuitively it means how often we can apply the association rule.

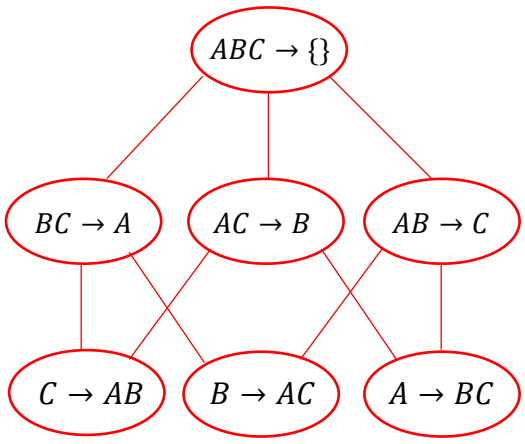
The anti-monotone property of support states that if an item is removed from an itemset, the resulting itemset will have the same or greater support than the original itemset. The anti-monotone property is the basis of Apriori principle, which states that all subsets of a frequent itemset must also be frequent, and all supersets of an infrequent item must also be infrequent. Apriori principle is the building block of the Apriori Algorithm, which prunes association rules whose antecedent and consequent (left- and right-hand sides) do not meet the minimum support input into the algorithm. Therefore, any association rule containing the item(s) that do not meet the minimum support can also be pruned.

(b) Unlike the support measure, confidence does not have any monotone property. Please explain how does confidence-based pruning work and provide a proof why it works. (5pts)

Confidence measures how often the association rule holds true (i.e. how often the antecedent implies the consequent).

$$Confidence(X \rightarrow Y) = \frac{Support(X,Y)}{Support(X)}$$

Confidence-based pruning in the Apriori Algorithm takes the pruning one step further by removing the search space based on a minimum confidence level. Consider the rule generation heuristic below.



If $BC \rightarrow A$ does not meet the minimum confidence, then not only can it be removed but also $C \rightarrow AB$ and $B \rightarrow AC$.

Proof:

Given:
$Confidence(BC \rightarrow A) = \frac{Support(A,B,C)}{Support(BC)} < Confidence\ Threshold.$

$$Confidence(C \rightarrow AB) = \frac{Support(A,B,C)}{Support(C)}$$

$$Confidence(B \rightarrow AC) = \frac{Support(A,B,C)}{Support(B)}$$

The numerators are the same for all three equations. They will be disregarded as a constant. Given that confidence of $BC \rightarrow A$ is less than the confidence threshold, the denominator, $Support(BC)$, must be a high enough value to make the confidence too low.

$$\frac{Constant}{Confidence\ Threshold} < Support(BC)$$

The Apriori Principle states that all subsets of a frequent itemset must also be frequent. Therefore,

$$\frac{Constant}{Confidence\ Threshold} < Support(B)$$

$$\frac{Constant}{Confidence\ Threshold} < Support(C)$$

Using the associativity property of multiplication,

$$\frac{Constant}{Support(B)} < Confidence\ Threshold$$

$$\frac{Constant}{Support(C)} < Confidence\ Threshold$$

Substituting back the numerator yields:

$$Confidence(B \rightarrow AC) = \frac{Support(A,B,C)}{Support(B)} < Confidence\ Threshold$$

$$Confidence(C \rightarrow AB) = \frac{Support(A,B,C)}{Support(C)} < Confidence\ Threshold$$

Problem 5 (12 pts):

We will build a naïve Bayes classifier based on the below weather dataset in order to determine whether to play golf or not. There are four categorical attributes (outlook, temperature, humidity, windy) and one binary target (play).

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

(a) For a day when it is sunny, hot, windy with high humidity, please use naïve Bayes to predict if we should play golf or not. (10pts)

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

$$X = (outlook: sunny, temperature: hot, windy: true, humidity: high)$$

$$y = play: yes$$

First, find the class probability of "play." (Prior probabilities)

		Class Probability
play: yes	9	9/14
play: no	5	5/14
Total	14	

Next, find the posterior probabilities. Find the probability of playing golf when the outlook is sunny.

	(Play:yes)	(Play:no)	Prob(yes)	Prob(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5		

$$Prob(outlook: sunny|play: yes) = \frac{2}{9}$$

Next, find the probability of playing golf when the temperature is hot.

	(Play:yes)	(Play:no)	Prob(yes)	Prob(no)
Cool	3	1	3/9	1/5
Mild	4	2	4/9	2/5
Hot	2	2	2/9	2/5
Total	9	5		

$$Prob(temperature: hot|play: yes) = \frac{2}{9}$$

Next, find the probability of playing golf when it is windy.

	(Play:yes)	(Play:no)	Prob(yes)	Prob(no)
Windy	3	3	3/9	3/5
Not windy	6	2	6/9	2/5
Total	9	5		

$$Prob(windy: true|play: yes) = \frac{3}{9}$$

Next, find the probability of playing golf when the humidity is high.

	(Play:yes)	(Play:no)	Prob(yes)	Prob(no)
Normal	6	1	6/9	1/5
High	3	4	3/9	4/5
Total	9	5		

$$Prob(humidity: high|play: yes) = \frac{3}{9}$$

Next, compute the probability of playing golf given the input conditions and compare it to the probability of not playing golf given the input conditions.

$$Prob(play: yes) = (Prob(outlook: sunny|play: yes) * Prob(temperature: hot|play: yes) * Prob(windy: true|play: yes) * Prob(humidity: high|play: yes) * Prob(play: yes))$$

$$Prob(play: yes) = \frac{2}{9} * \frac{2}{9} * \frac{3}{9} * \frac{3}{9} * \frac{9}{14} = 0.003527$$

$$Prob(play: no) = (Prob(outlook: sunny|play: no) * Prob(temperature: hot|play: no) * Prob(windy: true|play: no) * Prob(humidity: high|play: no) * Prob(play: no))$$

$$Prob(play: no) = \frac{3}{5} * \frac{2}{5} * \frac{3}{5} * \frac{4}{5} * \frac{5}{14} = 0.04114$$

The two probabilities can be summed to 1, so they can be normalized as well.

$$Prob(play: yes) = \frac{0.003527}{0.003527 + 0.04114} = 0.0789$$

$$Prob(play: no) = \frac{0.04114}{0.003527 + 0.04114} = 0.921$$

The probability of play:no is higher than that of play:yes. Therefore, the prediction is that golf will not be played in these conditions.

(b) Does the prediction agree with the classification provided in the training data set? (2pts)

The naïve bayes classification model was trained using the training dataset (supervised learning). After it learned, it was used to predict the target attribute of a single entry (unsupervised learning).

Problem 6 (10 pts):

Use the same training dataset (as in Problem 5) for golf playing and calculate the better splitting attribute (between OUTLOOK and HUMIDITY) to use as the first level attribute in constructing decision tree with entropy.

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

The information gain will be used to determine which attribute to choose as the first level in constructing a decision tree with entropy. The information gain of outlook and humidity will be calculated.

Information gain is defined as follows:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where S represents the dataset, A represents an attribute.

Outlook has three possible values: rainy, overcast, or sunny.

$$S = [9+, 5-]$$

$$S_{rainy} \leftarrow [3+, 2-]$$

$$S_{overcast} \leftarrow [4+, 0-]$$

$$S_{sunny} \leftarrow [2+, 3-]$$

$$Gain(S, Outlook)$$
$$= Entropy(S) - \left(\frac{5}{14}\right) Entropy(S_{rainy}) - \left(\frac{4}{14}\right) Entropy(S_{overcast})$$
$$- \left(\frac{5}{14}\right) Entropy(S_{sunny})$$

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = 0.94$$

$$Entropy(S_{rainy}) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.971$$

$$Entropy(S_{overcast}) = 0$$

$$Entropy(S_{sunny}) = -\left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) = 0.971$$

$$Gain(S, Outlook) = 0.94 - \left(\frac{5}{14}\right) * (0.971) - 0 - \left(\frac{5}{14}\right) * (0.971) = \boxed{0.2464}$$

Now the information gain of the humidity will be calculated.

$$S_{normal} \leftarrow [6+, 1-]$$

$$S_{high} \leftarrow [3+, 4-]$$

$$Gain(S, Humidity) = Entropy(S) - \left(\frac{7}{14}\right) Entropy(S_{normal}) - \left(\frac{7}{14}\right) Entropy(S_{high})$$

$$Entropy(S_{normal}) = -\left(\frac{6}{7}\right)\log_2\left(\frac{6}{7}\right) - \left(\frac{1}{7}\right)\log_2\left(\frac{1}{7}\right) = 0.5917$$

$$Entropy(S_{high}) = -\left(\frac{3}{7}\right)\log_2\left(\frac{3}{7}\right) - \left(\frac{4}{7}\right)\log_2\left(\frac{4}{7}\right) = 0.985$$

$$Gain(S, Humidity) = 0.94 - \left(\frac{7}{14}\right) * (0.5917) - \left(\frac{7}{14}\right) * (0.985) = \boxed{0.152}$$

Because the information gain of the outlook attribute is higher than humidity, it is the better splitting attribute between outlook and humidity. The **outlook** attribute should be chosen as the first level in constructing a decision tree with entropy.

Problem 7 (10 pts):

A dataset of 800 cases was partitioned into a training set of 650 cases and a validation set of 150 cases. Classifier *A* predicts class label for the validation case based on the closest training case (a.k.a.1 Nearest Neighbor classifier). Classifier *A* has a misclassification error rate of 4% on the validation data. It was later discovered that the partitioning had been done incorrectly so that 80 cases from the training data set had been accidentally duplicated and had overwritten 80 cases in the validation dataset. What is the true misclassification error rate for the validation data?

The dataset was partitioned into 650 cases for the training data and 150 cases for the validation data. However, 80 cases in the training dataset had been accidentally duplicated, so the correct number of cases in the training dataset is 650 + 80 = 730.
Also, the mistake affected the testing dataset by overwriting 80 cases. Therefore, the correct number of cases in the validation dataset is 150 – 80 = 70.
The misclassification error rate is 4%. Therefore, 4% of the 70 cases in the validation dataset are incorrect. 70 cases * 0.04 misclassification error rate = 2.8 incorrect cases.
Therefore, the number of correct cases in the validation dataset is 70 – 2.8 = 67.2.
Hence, the misclassification error rate for the validation data is 67.2 cases correct / 150 validation cases total = 0.448 = 44.8%.

Problem 8 **(10 pts):**

Please provide at least **FIVE** hyper-parameters we discussed in the class for a deep learning algorithm and discuss in details how each one of them impacts the performance of the model.

The learning rate is the hyperparameter for the gradient descent algorithm. It determines how many updates are made to the weights of the neural network with respect to the loss gradient descent. The gradient descent algorithm is used to find the minimum value of error in the model. An optimally weighted neural network will be able to mitigate error in the quickest route possible. If the error is not decreased, it could grow exponentially across each layer of the neural network. If the learning rate is set too high, the learning process may overstep the optimal weights for the neural network. If it is too low, the training process will take too long to find. Each step would produce a small adjustment and the solution would take a long time to converge.

The number of epochs is the hyperparameter that determines the number of times we allow our training dataset to pass through our neural network. One time forward and backward is referred to as one epoch. Because it is so large, an epoch is made up of one or more batches. Example: if a dataset contains 10,000 entries, and it is divided into batches of 1,000 entries each, then the number of iterations through the gradient descent algorithm to complete one epoch is 10 iterations.

Batch size is an important hyperparameter that is used to set the size of the batch. It represents the number of entries that pass through the network before the weights of the neural network are updated. The weights of the nodes are not updated after every entry passes through as that would be too frequent. They also do not update after every epoch as that would be too infrequent. The batch size is used to define a middle ground so that the weights are being properly updated at the correct frequency.

Other hyperparameters define the architecture of the neural network such as the number of the hidden layers and the number of the nodes in each hidden layer. The hidden layer provides the capability to model complex systems of inputs and outputs (i.e. the relationship of the I/O is non-linear). Increasing the number of hidden layers in a neural network can be dangerous as it increases the networks susceptibility to overfitting. It also increases the overall training time and computational resources required. It is advised that all other hyperparameters be tested and tuned before attempting to add a hidden layer to a neural network.

The dropout rate is also a hyperparameter. Dropout is a form of regularization in deep learning. Regularization is a way to limit the magnitude of the error and improve model performance by preventing overfitting on the training dataset. The dropout method is performed by randomly selecting nodes to drop during the model training so that the model does not overfit.