CS578 Empirical Methods in Machine Learning and Data Mining
Mid-Term Exam   Due: Tuesday, November 6, 2007 at 2:55pm

Please keep answers clear and concise. Some questions can be answered with a few words. Some questions admit alternate answers depending on how you think about the problem. Better answers receive more credit.

All work must be done individually. Discussion is not allowed. If you have questions see the instructor or one of the TAs. You may use references. The text for this mid-term is available through the course web page. Handwritten exams are acceptable if legible.

Sign your exam before turning it in indicating that you did the work without external help. Exams will not be accepted without signatures.

(18 points) COMPUTATIONAL COMPLEXITY OF LEARNING ALGORITHMS

For the following, assume a 2-class problem with N training cases and A attributes. Where appropriate, express answers with big-O notation in terms of A and N.

(3) What is the computational complexity of finding the best binary split for one continuous attribute (i.e., finding the threshold for that split)?

(3) Suppose attribute A has V discrete values (i.e., A is arity V), but we want to do a binary split on A. One way to do this is subsetting, where one subset of the values of A go down one branch, and the remaining values of A go down the other branch. What is the complexity of finding the optimal partitioning of the values V into the subsets for the two branches branches?

(3) What is the complexity of doing 1-step lookahead when installing the root attribute in a decision tree?

(3) How many multiplies are required to do forward propagation in a standard fully-connected feedforward neural net with A inputs, H hidden units, and O outputs?

(3) What is the complexity of finding the single nearest neighbor to a test point in 1-NN?

(3) What is the complexity of LOOCV (leave-one-out cross validation) using 1-NN?

(6 points) OVERFITTING AND GENERALIZATION

(2) Training with more training data usually improves generalization. Suppose we have a training set containing M training cases. If we duplicate the training cases D times each so that the training set now contains D*M cases (but still only M unique cases), will this improve the generalization performance of a neural net? Why?

(2) Suppose instead of a neural net we use k-NN on the duplicated training set. Will the optimal value of k be larger than, smaller than, or the same size as the optimal value before the training set was duplicated? Why?

(2) Suppose instead of a neural net or k-NN we prune a decision tree using the duplicated training set. Will the tree pruned using the duplicated data likely be larger, smaller, or the same size as the tree pruned on the original unduplicated data? Why? State any assumptions you make.

(21 points) DECISION TREES

We want to grow decsion trees to minimize squared error instead of using information gain or gain ratio. Let us define RMS_Gain as:

$$\text{RMS\_Gain}(\text{Attribute}) = \text{RMSE}(S) - \sum_{v} \left(\frac{|Sv|}{|S|} * \text{RMSE}(Sv) \right)$$

where S is the node being split, v ranges over the values of the Attribute, Sv is the subset of cases from S with Attribute value v, and |S| and |Sv| are the number of cases at the node S or in the subset Sv. RMS_Gain is similar to InfoGain, but we've replaced the entropy terms with RMSE terms. The RMSE(S) for node S is calculated by predicting probabilities for the node in the usual way (i.e. by dividing the fraction of cases in each class in the node by the total number of cases in the node) and then computing the root-mean-squared-error of these predictions for the node.

Calculate the Info_Gain, Gain_Ratio, and Gini_Score for the following two attributes:

Attribute A	Attribute B	True Class
1	1	2
1	2	1
2	1	1
2	2	2
3	1	1
3	2	2
3	2	1
3	3	2

(2) Info_Gain ___ ___

(2) Gain_Ratio ___ ___

(2) RMS_Gain ___ ___

(2) What is the largest and smallest possible Info_Gain that could be calculated for *any* data set? Do large or small values of Info_Gain represent good splits?

(2) What is the largest and smallest possible Gain_Ratio that could be calculated on any data set? Do large or small values of Gain_Ratio represent good splits?

(2) What is the largest and smallest possible RMS_Gain that could be calculated on any data set? Do large or small values of RMS_Gain represent good splits?

(3) Which attribute test above, A or B, should we install for each splitting criterion?

Info_Gain? _____
Gain_ration? _____
RMS_Gain? _____

One nice property of decision trees is that they can handle missing values. When a case is missing an attribute value that is tested at a node, we can handle the missing value several ways:

1) let the case can go down all branches
2) let the case go down all branches weighted by 1/attribute_arity
3) let the case go down the branch most other cases went down
4) let the case go down each branch weighted by the fraction of other cases that went down each branch
5) let the case go down one branch selected at random
6) create a new branch just for the missing values

(2) Which approach do you think is best? Briefly explain why.

(2) Which approach do you think is worst? Briefly explain why.

(1) Just between 1) and 2), which one do you think is more appropriate? why?

(1) Briefly describe a situation where 6) would be very inappropriate to use.

(18 points) K-NEAREST NEIGHBOR

(2) Does k-nearest neighbor converge to baseline performance as k approaches 1 or as k approaches the size of the training set? Why?

(3) If a data set has significant class noise, do we want to use a smaller or larger value of k? Why?

(3) Does kNN tend to overfit with smaller values of k or with larger values of k? Why?

(3) A training set has M instances and the optimal value of k for this training set is K_OPT. If the training set gets larger, do you expect the optimal value of k to increase, decrease, or stay the same? Why?

(3) The optimal value of k for unweighted kNN for a data set is K_OPT. If we switch to weighted kNN (where the weight of a case is inversely proportional to the distance to that case), do you expect the optimal value of k to increase, decrease, or stay the same? Why?

(2) Suppose we perform kNN on a large data set with many features, but give weight 1 to all features used in a decision tree trained on this data, and give weight 0 to all features not used in the decision tree. Do you think kNN will perform better, worse, or about the same with the features weighted this way? Why?

(2) Suppose a data set contains both nominal attributes and continuous attributes. Calculating distance on continuous attributes is easy. One way to calculate distance on nominal attributes is to define the distance between two nominal attributes as 0 when the attributes have the same value, and 1 whenever the attribute have different values. Another approach is to first convert each nominal attribute of arity V to V binary attributes (as we did with neural nets), and then calculate distance on these binary attributes. Does it matter which approach we use? If it does, what should we do to make the two approaches yield the same answers?

(10 points) PERCEPTRONS

In the following, assume inputs are +1 or -1 and that the threshold unit outputs +1 for inputs > 0 and -1 otherwise.

(2) Briefly explain why the threshold unit in perceptrons is both a blessing and a curse.

(2) Briefly explain why replacing threshold units with sigmoids (or any well-behaved non-linear squashing/activation function) makes neural nets more useful than perceptron nets.

(2) Draw a perceptron that computes the NAND (NOT AND) function for two inputs A and B. Be sure to show the weights for the inputs to A and B, as well as the bias weight. (This is not a network -- just a single perceptron.)

(2) Draw a perceptron that computes the OR function for three inputs A, B, and C. Be sure to show the weights for the inputs to A, B and C, as well as the bias weight. (This is not a network -- just a single perceptron.)

(2) Draw a network of perceptrons that computes the XOR function for three inputs A, B, and C. Be sure to show the weights for the inputs to A, B and C, as well as all bias weights and weights joining the

perceptrons.

(19 points) ARTIFICIAL NEURAL NETS (ANNs)

(2) Backpropagation is done with error functions on the outputs such as squared error or cross-entropy, even when the goal is to maximize classification accuracy. Why is it difficult to do backpropagation to maximize classification accuracy directly?

(2) When training a neural net with backpropagation on a training set with 0/1 targets, which do you think is likely to reach its early stopping point first, RMSE or accuracy? Why?

(2) Consider a neural net with one hidden layer containing N hidden units, and another neural net with N hidden layers each containing 1 hidden unit. If there are I inputs and O outputs, how many weights are in each neural net? Which net can learn more complex functions?

(3) Show that a neural net with two hidden layers consisting only of linear units is equivalent to a network with one hidden layer of linear units. That is, if we remove the non-linear activation functions (e.g., the sigmoids) from the nodes so that each node's activation is just the weighted sum of it's inputs, show that the expressive power of a network with two hidden layers is the same as an ANN with one hidden layer. To keep it simple, it is sufficient to show this for a simple ANN that has 2 inputs, 2 hidden units in each layer, and one output unit.

(3) Are the bias weights in an ANN really necessary? Suppose we eliminate the bias weight going to each node. This means each node's activation is the sigmoid of just the weighted sum of it's inputs, without an extra weight going to a constant "1" activation. The net is trained as usual with backpropagation. How well do you think it will perform compared to a regular ANN trained with bias weights. Why?

(3) If a neural net with one hidden unit is trained long enough on a large consistent training set containing N cases, will the training set error always go to zero? If a neural net with 2^N hidden units is trained on the same training set, will the training set error always go to zero? Explain your reasoning?

(4) Derive backprop for just the output unit of a neural net trained with -1*[target*log(output) + (1-target)*log(1-output)] instead of the usual squared error (Target - Output)^2. Assume a sigmoid output unit and 0/1 targets.

(8 points) CROSS VALIDATION

(3) Suppose you are doing m-fold cross-validation. Show that the average accuracy of the m folds is equal to the accuracy of the union of the m-fold test sets. In other words, compute the accuracy on each fold individually, and then take the average of these m accuracies, and compare this to calculating the accuracy on all the folds concatenated together after doing prediction on each of them with m-fold CV. Is the same true for RMSE (root-mean-squared error)?

(3) List as many reasons as you can why we usually prefer 10-fold cross validation to 2-fold cross validation?

(2) Suppose we use 3-fold cross validation to train neural nets. In trial 1 the net is trained on fold 1, early stopped on fold 2, and tested on fold 3. 3-fold CV yields three neural nets, N1, N2, and N3. Suppose we want to combine the three nets into one model M by averaging the predictions from the three nets. If Ni(Cj) is the prediction net Ni makes on test case Cj, the average prediction of the three nets is M(Cj) = (N1(Cj)+N2(Cj)+N3(Cj))/3 for case Cj. If we compare M to Ni on Ni's test fold, why isn't this a fair comparison?