CS 411 — Artificial Intelligence I — Spring 2017

Assignment 2

Department of Computer Science, University of Illinois at Chicago

Instructor: Xinhua Zhang

Due: Mar 27, 2017 Mon by 5 PM (CST)
Out: Mar 3, 2017

Instructions

This is an **individual** assignment**.** Your mark will be out of 100, and it contributes 10% to your final course score. It consists of two parts: **programming** and **written questions**. The purpose of the programming part is to make you start experimenting with decision tree learning algorithms. We will build on the implementation of the algorithms from the textbook (called the AIMA code at http://aima.cs.berkeley.edu/code.html).

What to submit: You should submit to **Blackboard** a single zipped file (.zip or .tar) which includes the following files:

1. For the written problems, provide your answers in a single Portable Document Format (PDF) file, and name it as Written.pdf. It can be either typed using WORD or latex, or scanned copies of handwritten submissions provided that the handwriting is neat and legible.

2. **(read very carefully)** For the programming part, only typed submissions are accepted. Please submit:

a) **Code**: All the code you wrote – in such a form that the TA can run it. Please include plenty of comments. Put all code in one folder named Code/.
b) **ReadMe**: A ReadMe file that explains to your TA how to run your code. Name the file as ReadMe. In particular, highlight the **name of function and file** where you implemented for Q1, Q2 and Q4.
c) **A Report** in PDF format with filename Program.pdf documenting your answers to the questions.

How to submit: You'll submit your homework online, via Blackboard. Go to the class web site, and folder Assignments. The entry "Assignment 2" in this folder will have a link through which you can upload your homework. Submit a single zipped file named Surname_UIN.zip or Surname_UIN.tar, where Surname is your last name and UIN is your UIC UIN. Inside it, there should be two PDF files, one ReadMe, and one code folder as mentioned above.

Resubmission: The submission site will be open up to **5 PM of Mar 27, 2017**. You are allowed to resubmit (upload a new version) until the deadline. Grading with be conducted on the **last** version submitted **before** the deadline.

Programming Languages: You will need to invest a fair amount of time to understand how the AIMA code works, so that you can add to it and modify it. The book code comes in three languages: Java, Python and LISP (see http://aima.cs.berkeley.edu/code.html). You can choose either **Java or Python**.

Late Policy: A mark of 0 will be given if no submission has been made by the deadline (in the absence of medical evidence or other special permission).

Cheating and Plagiarism: All assignments must be done individually. Remember that plagiarism is a university offence and will be dealt with according to university procedures. Please refer to the corresponding UIC policies: http://dos.uic.edu/docs/Student%20Disciplinary%20Policy.pdf

Latex primer: http://ctan.mackichan.com/info/lshort/english/lshort.pdf

Part I. [50 points] Programming using decision tree

This homework will give you an opportunity to experiment with Decision Tree learning (DTL), using two different heuristics for choosing attributes: information gain (IG) and gain ratio (GR). You will also test feature engineering by discretizing (some) attributes.

Code

You are encouraged to use the AIMA code implementation. But if you prefer, you can of course go ahead and re-implement everything, but it will involve a substantial amount of work.

Java users: For Java, the implementation provides methods DecisionTreeLearner.train() and DecisionTreeLearner.test() that you can use for this purpose. Check the LearningDemo.java code for a demo of training and testing various learning algorithms on some sample data -- the learning demo is not part of aima core, but of aima gui. We are only interested in decisionTreeDemo (note the wrong println -- inducing DecisionList, it does induce the decision tree as expected). Also note that in this demo the tree is trained and tested on the same data set, the restaurant data set. In general, this is bad practice.

The JAVA code for DTL returns the number of data points correctly and incorrectly classified, not the error rate (see the method DecisionTreeLearner.test()).

Python users: For Python, it also contains code for cross-validation with random partitions of the data, and code to compute various types of errors. However, we will use stratified cross-validation.

Data

You need the following files, available from Blackboard under Assignment 2:

diabetes.csv: 768 data points on diabetes in Pima Indian women (768 lines, comma separated values)
diabetes.names: describes the meaning of attributes (also specifying attribution for the data).

The last column is the target class: 0 for no diabetes, and 1 for yes.

Note that we have discretized two continuous attributes: body mass index and diabetes pedigree function.

More information about the dataset can be found at
https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

Important Note: In your implementation, give the following names to the attributes in the diabetes data set (see the diabetes.names file):

pregnant glucose pressure triceps insulin bmi pedigree age diabetes

Questions

1. [5 pts] Stratified cross validation. To run your experiments, you must divide the data into N partitions to run cross-validation; then run each of the two learning models (DT with Information Gain and Gain Ratio) N times: in cross-validation, the algorithm trains on (N – 1) partitions and tests on the N-th partition. Finally, you will compute the average error rate across the N runs, for each algorithm.

If we randomly split the dataset in N subsets, we run the risk that the original distribution of the classes (in our case, yes / no), is not reflected in those subsets. In stratified cross-validation (SCV), one tries to use the same proportion of the classes in each subset. Implement a variation to the random split such that, in each of those N partitions, the distribution of the two classes is the same as that of the whole data set (up to rounding).

WRITE CODE that reads a number $N \in \{2, 3, \ldots, 5\}$ representing the number of partitions, randomly splits the data into N subsets, and then runs SCV, as discussed above. For your experiments, you will use N = 5, but N must be a parameter.

IMPORTANT: Save your partitions once you obtained them! You will use the same partitions for all experiments.

2. [3 pts] **WRITE CODE** that computes the error rate δ as the number of errors the model makes **on the test set** divided by the cardinality of the test set -- you will need to compute the error rate of each run of SCV, and then compute the average error rate at the end.

3. [8 pts] **RUN THE ALGORITHM** of DTL with information gain as given. Print (to standard output) the trees obtained by each run of SCV, the error rates for each run, and the average error rate.

4. [18 pts] Attributes with many different possible values can cause problems with the information gain. Such attributes tend to split the examples into numerous small classes or even singleton classes, thereby appearing to be highly relevant according to the gain measure. To address this problem, the gain ratio criterion selects attributes according to the ratio between their gain and their intrinsic information content – that is, the amount of information contained in the answer to the question, "What is the value of this attribute?" The gain ratio criterion therefore tries to measure how efficiently an attribute provides information on the correct classification of an example.

Mathematically, Gain Ratio is defined as:

$$GainRatio(E, A) = \frac{InformationGain(E, A)}{SplitInformation(E, A)}$$

where

$$SplitInformation(E, A) = -\sum_{k=1}^{d} \frac{|E_k|}{|E|} \log_2 \frac{|E_k|}{|E|}$$

Here E_1, …, E_d are the d subsets of examples resulting from partitioning E by the d-valued attribute A.

WRITE CODE to implement the Gain Ratio measure to choose the next attribute. **RERUN** the SCV experiments with Gain Ratio, by using the 5 partitions obtained previously. Print to standard inputs the trees, the error rates for each run, and the average error rate.

5. [8 pts] The trees you obtained are bound to be very bushy for certain attributes, i.e. to have lots of branches because the attributes have lots of different values. Pick two attributes and think about how to discretize them, which in this case means, how to reduce their values to symbolic intervals. E.g., suppose we had an additional attribute height, which could span (in inches) from 48 to 72, and that all possible values between 48 and 72 were present in the data. We could discretize it into 6 intervals, each spanning four inches, and that we'd call something like very-short, short, average, tall, very-tall, extremely-tall; or we could discretize it into 4 intervals, or into 3. Also note the division need not be uniform, i.e., the intervals could cover spans of different measures, e.g. 3 intervals, one from 48 to 55, one from 55 to 63, one from 63 to 72.

For each of the two attributes, decide how to discretize it, then write code that automatically rewrites your data. Rerun the two decision tree learners on the discretized new dataset. Print (to standard output) the derived trees and the error rates.

6. [8 pts] Your report will include:

(a) [points included in the above Q1-5] Portions of four trees: two for the original dataset and two for the discretized dataset. For both original and discretized datasets, one tree for one run with DTL with Information Gain, and one tree for one run with DTL with Gain Ratio (pick one of the 5 trees returned during SCV).

We ask to report portions of trees and not the whole trees because they're bound to be very large. For the four trees you chose, include the top portion -- the root, its children and grandchildren; and one of the terminal subtrees which bottom out with leaves, starting from the leaves, and going up two levels (parents / grandparents).

(b) [points included in the above Q1-5] The four average error rates for the two algorithms (DTL with Information Gain and Gain Ratio) on original/discretized dataset.

(c) [8 pts] Comments on the relative performance of the algorithms as follows:

(i) Does DTL with Information Gain behave better or worse than DTL with Gain Ratio? Consider performances to be the same if they are within 1% of each other (this is a shortcut to statistical significance).

(ii) A priori, what algorithm would you have expected to perform the best and why? Did the results confirm your expectations?

Several notes:

1. It's totally fine to have a decision tree where no leaf node has pure label. The splitting for cross validation should be random, as long as it satisfies the requirement of stratified CV.

2. Question (**for Java users only**): I was wondering if DecisionTreeLearner.train works for numeric values in Java. The restaurant data works fine for me. When I try to do anything that involves numeric values, I get "aima.core.learning.framework.NumericAttributeSpecification" cannot be cast to aima.core.learning.framework.StringAttributeSpecification".

Answer: One solution can be turning all attribute values into string. This will require defineStringAttribute used in a way like:dss.defineStringAttribute("pedigree", new String[]{"0", "1", "2"}); So you will need to write {"148", "85", "183", "89" ,…} in the code, i.e. enumerate all possible values of the "string". I guess that's fine, and the file can be read in two passes. For example, just store all values of the second column in an array, and then pull the distinct values in it. The following links will help:

http://stackoverflow.com/questions/14656208/array-of-unique-elements
http://stackoverflow.com/questions/26679860/pulling-distinct-values-from-a-array-in-java

After that, turn the array of numbers into an array of strings, and use it in defineStringAttribute. Or you may first turn the numbers into strings, and then pull the distinct values. This requires reading the file in two passes. But at least it automates the process.

Part II. [50 points] Written questions

7. [6 pts] True or false. Justify your answer: if you answer no, then provide a counter-example. If you answer yes, then provide a proof.

(a) [3 pts] Consider a knowledge base containing just two sentences: $P(a)$ and $P(b)$. Then this knowledge base entails $\forall x\ P(x)$.

(b) [3 pts] Assume the domain of the model is not empty. The following sentence is valid: $\exists x, y\ x = y$.

8. [18 pts] Suppose we know the following facts:
(1) Anyone who takes an AI course is smart.
(2) Any course that teaches an AI topic is an AI course.
(3) John takes CS411.
(4) CS411 teaches Inference.
(5) Inference is an AI topic.

(a) [4 pts] Represent these facts as first-order logic sentences. Use the following predicates:

- AI_course(x) is true if and only if (iff) x is an AI course,
- Takes(x,y) is true iff a person x takes a course y,
- Smart(x) is true iff x is smart,
- AI_topic(x) is true iff x is an AI topic,
- Teaches(x,y) is true iff a course x teaches a topic y.

(b) [7 pts] Convert all first order logic sentences into conjunctive normal form.

(c) [7 pts] Use resolution to prove that "John is smart".

9. [8 pts] A cab was involved at night in a hit and run accident. In this town there are only two cab companies, the Blue with 85% of the cabs and the Green company with the remaining 15%. As it happened there was a witness to the accident who identified the hit and run cab as Blue. The case came to trial. An important part of the testimony was a test of the reliability of the eyewitness to identify a cab as being either Blue or Green under the same circumstances as those that existed the night of the accident. The result of these tests was that the witness correctly identified a blue cab as blue 60% of the time and a green cab as green 80% of the time. On the basis of the claim of the eyewitness and the test conducted by the court, what is the probability that the cab involved in the accident was in fact Blue as the witness had claimed?

10. [8 pts] In what follows, suppose that X, Y are random variables with possible outcomes $\{0, 1\}$. For each of the following statements, write whether they are True (T) or False (F). Wherever $P(X = x \mid Y = y)$ appears, it is implicitly assumed that $P(Y = y) > 0$. Similarly for $P(Y = y \mid X = x)$. In each case, if you say True, then give a proof. If you say False, then construct a joint distribution on (X, Y) such that the assertion does not hold.

(a) (2 pt) It is always true that $P(X = 1 \mid Y = 1) + P(X = 1 \mid Y = 0) = 1$.

(b) (2 pt) It is always true that $P(X, Y) \leq P(X \mid Y)$. Note this actually involves four inequalities

corresponding to $X \in \{0, 1\}$ and $Y \in \{0, 1\}$.

(c) (2 pt) If $P(X \mid Y) = P(Y \mid X)$, then X and Y must be independent.

(d) (2 pt) Given the conditionals $P(X = x \mid Y = y)$ for every x and y, and the marginal $P(X = x)$ for every x, we can compute $P(Y = y \mid X = x)$ for every x and y.

11. [10 pts] The loans department of a bank has the following past loan processing records each containing an applicant's income, credit history, debt, and the final approval decision. These records can serve as training examples to build a decision tree for a loan advisory system.

Income	Credit History	Debt	Decision
$ 0 – $ 5K	Bad	Low	Reject
$ 0 – $ 5K	Good	Low	Approve
$ 0 – $ 5K	Unknown	High	Reject
$ 0 – $ 5K	Unknown	Low	Approve
$ 0 – $ 5K	Unknown	Low	Approve
$ 0 – $ 5K	Unknown	Low	Reject
$ 5K – $ 10K	Bad	High	Reject
$ 5K – $ 10K	Good	High	Approve
$ 5K – $ 10K	Unknown	High	Approve
$ 5K – $ 10K	Unknown	Low	Approve
Over $10K	Bad	Low	Reject
Over $10K	Good	Low	Approve

Use the above training examples to construct a decision tree based on information gain.