COMP 343: Programming Project 2

Fall 2016

Your second project involves building and evaluating a decision tree learner for the classic iris flower data set. This problem generalizes the book's algorithm in two ways: continuous inputs and greater than two classes. The project is broken into three milestones each containing a distinct coding and analysis component.

Iris Data

The iris flower data set is so well known that it has its own wikipedia page[1]. The data set contains 150 examples of iris flowers from three distinct classes, 50 samples per class. Flowers are measured by four continuous valued features. By modern standards, this data set is small in size and low in dimension, but it remains a good place to cut your teeth on learning methods if for no other reason than its place in history. You may download the data set as comma-separated values from the UCI Machine Learning Repository[2].

[1] `https://en.wikipedia.org/wiki/Iris_flower_data_set`

[2] `https://archive.ics.uci.edu/ml/datasets/Iris`

Continuous Valued Features

One basic strategy for dealing with continuous valued data is to discretize it, make it discrete, by breaking the range of values into discrete intervals. The simplest way of doing this is to choose some threshold value t and partition data by those less than or equal to t and those greater than t. The key is, obviously, to choose a good threshold. Once again, entropy can at as a guide and provide a greedy method for selecting threshold values from a set of candidate values. This process and various performance optimazations for the computation involved is discussed Fayyad and Irani's paper from 1992[3].

[3] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Mach. Learn.*, 8(1):87–102, January 1992

Stratified Random Sampling

Our initial data set has equal representation from each class. Assuming these are representative samples, then this kind of balance is generally ideal for learning. To properly evaluate and train a learner we need to split some proportion of the data into training data and leave the rest for testing. This split should be done randomly to avoid bias with respect to a specific subset of our data. If we'd like to maintain the equal representation of all three classes in our training data, then we need to carry out *stratified* sampling. In stratified sampling we split the data by class into groups called strata and then randomly sample from each strata. Those samples are then combined to form the training data. The remaining data is combined to form the test set.

Project Milestones

The project is organized by three milestones. The core programming tasks are accumulated from one milestone to the next. Formal tests must be given for the core functionality listed in each milestone. Each milestone has an analysis task associated with it. Analysis tasks are *not* cumulative, but doing all the analysis tasks up to your target milestone can net you *extra credit*. Plan your time accordingly.

C-level Project

To achieve the C-level milestone you must have code that can:

1. compute the entropy of an arbitrary set of Iris data
2. compute the average entropy of an arbitrary set of sets of Iris data[4].

[4] called Remainder in the text

You're not required to read the actual iris data from the file at this point, but you must be able to mock up data with the same structure[5].

[5] think $n \times 5$ matrix

In addition to the above coding requirement you must do an analysis of the entropy of discrete systems with three possible values like the iris flower data. This analysis will result in producing what is known as a set of small multiples[6].

[6] multiple, small visualizations of a data set

For a random variable with three outcomes, c_1, c_2, and c_3, we'd like to know how the entropy of the variable changes as the probabilities of each outcome changes. This is a three dimensional problem. We have two degrees of freedom in the probability distribution and then we want to plot that against entropy. We can flatten this by doing a column of 2D plots. Each plot in the column has a fixed value for $P(c_1)$ and as we progress down the column we step through values for $P(c_1)$. For example, if the step size were $\frac{1}{9}$, then the first row would be a plot for $P(c_1) = 0$, then for $\frac{1}{9}$, $\frac{2}{9}$, $\frac{1}{3}$, etc. such that the i^{th} plot has a fixed probability $P(c_1) = \frac{i}{9}$. We can then plot the entropy for the distribution $[\frac{i}{9}, k, (1 - \frac{i}{9} - k)]$ for $0 \leq k \leq (1 - \frac{i}{9})$ as a 2D plot with k on the x-axis and entopy on the y.

B-level Project

To achieve the B-level milestone you must have completed the C-level code as well as the following:

1. import the data from a file to an appropriate data structure
2. stratified random sampling given a specified percentage p such that p percent of the data is training and $(1 - p)$ will be used for testing.
3. compute the average entropy of a partitioning induced by a feature threshold. There are several ways of organizing this logic. One possibility is to have a feature index and threshold parameters, i.e. $remainder(S, 2, .5)$ returns the remainder on set S when partitioned by feature 2 with a threshold of 0.5.
4. select the best threshold value for a given feature with respect to a given set of data.

For the analysis task of this milestone you'll look at the variability in threshold values with respect to random sampling. To do this you'll take 20 random training sets for training sets covering 90% of the original data set down to 20% in 10% increments. For each of these sets you'll compute the optimal threshold value for each Iris feature. From this data you'll produce 4 point plots, one per feature. The x-axis for the plot is the size of the training sample as a percent of the total data. The y-axis is the threshold value. For each of the sizes you tested there should be a set of 20 points in a vertical line. Alternatively, you can replace the points with a box and whiskers plot. In either case, we should be able to see the variability of the computed threshold for a given size by examining the data at each size point and the effect of smaller samples by scanning left to right in the plot.

A-level Milestone

To achieve the A-level milestone you must complete the coding tasks for the other milestones and complete the code necessary to compute and evaluate a decision tree for Iris classification. This includes:

1. constructing (by hand) and utilizing decision trees
2. the algorithm for learning a decision tree from training data
3. code for computing classification error on an arbitrary set of labeled data points

The analysis task is to compute learning graphs by constructing trees from stratified training sets whose proportions range from 20% to 80% of the total data. For each training set size you should split the data 20 times and average the training and testing error across all 20 splits. You then graph average training and testing error (or similarly correctness) as a function of training set size.

References

[1] Usama M. Fayyad and Keki B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Mach. Learn.*, 8(1):87–102, January 1992.