Open Lab 3

Supervised Learning

CSCI 4350 - Introduction to Artificial Intelligence

Due: Nov. 15 @ 11:00pm

Overview

Develop a software agent in Python to learn an ID3 decision tree from labeled classification data.

Procedure

1. Create a Python program (`id3.py`) which creates an ID3 decision tree to classify a set of input training data and then reports the classification performance on a separate set of input validation data.
   - The program should take 2 command-line arguments: (**string**: input training data filename, **string**: input validation data filename).
   - The program should read in the **training data** file (one training example per line, see below).
   - The program should read in the **validation data** file (one validation example per line, see below).
   - *Each line* will contain the **real-valued features** for several attributes and a single integer **class label** at the end.
   - The program should build an ID3 decision tree by:
     - **Sorting** the *training data* along **each attribute**,
     - Determining all **potential *binary* split points** based on **attribute value changes** (average the two values to make the split: those examples less than the split value **or** those examples greater than or equal to the split value),
     - **Calculating** the associated **information gain** for **all** of the potential split points,
     - Choose to make a split node for the potential split with the **maximum information gain**,
     - **Ties** (in maximum information gain) should be broken by attribute order (left to right) and then attribute value (smallest to largest) as found in the input file,
     - **Terminal nodes** are created instead of split nodes when:
       - the probability of one of the class labels is 1 (all others zero) or
       - there are no more potential split points found among the attributes (in this case, use a majority class label vote, with ties in favor of the smaller integer label).
   - **After the decision tree has been created**, each of the *validation* examples should be classified using the resulting decision tree.
   - The program should then output **only** a single integer value (on one line): the **number of validation examples** classified **correctly** by the decision tree.

2. Utilize your program to perform **cross-validation analysis** (specifically, repeated random sub-sampling a.k.a. monte-carlo sampling) on the **iris and cancer** data sets (see below).
    - Use bash scripting tools (see `split.bash`) to create $n = 100$ different training and validation sets of size $(m - v)$ and v, respectively, where m is the **total** number of examples in the entire data set, v is the desired number of validation examples, and $m - v$ is the desired number of training examples.
    - For the iris data set use $v = [1, 5, 10, 25, 50, 75, 100, 125, 140, 145, 149]$ and for the cancer data set use $v = [1, 5, 10, 25, 50, 75, 90, 100, 104]$.
    - Calculate the **mean**, μ, and **standard error of the mean**, $\hat{\sigma}_\mu$, of the percentage of testing examples correctly classified by your decision trees for each data set, where μ is the arithmetic mean, and $\hat{\sigma}_\mu = \frac{\sigma}{\sqrt{n}}$ where σ is the standard deviation.
    - Use an iPython Notebook, Matplotlib, and/or other tools to **plot** $\mu \pm 1.96 * \hat{\sigma}_\mu$ across all validation set sizes, v.
3. Write a report (at least 2 pages, single spaced, 12 point font, 1 inch margins, no more than 4 pages) describing:
    - the ID3 method,
    - the code you developled to implement ID3,
    - the performance of the code under cross-validation (using the statistics above for justification),
    - any limitations of the overall approach,
    - and describe any additional implementation details that improved the performance of your code

Requirements

- Additional tools/scripts for this assignment may be found here: OLA3-support.zip
- You should utilize the Iris data set to build your ID3 agent: iris-data-txt
    - A link to the original data set, with additional information can be found here: Iris@UCI
    - **DO NOT** use the original data set from the UCI link as input; I have re-formatted it to match the specifications above.
- You should also utilize the Breast Cancer data set to analyze the performance of the ID3 agent: cancer-data.txt
    - A link to the original data set, with additional information can be found here: BreastTissue@UCI
    - **DO NOT** use the original data set from the UCI link as input; I have re-formatted it to match the specifications above.
- Include a header in the source code with relevant information for assignments (your name, course number/name, etc).
- Your code should **only** print the number of correctly classified testing examples **followed by a newline** character.
- Example Training Data (training.txt):

```
4.9 2.5 4.5 1.7 2
5.6 2.8 4.9 2.0 2
7.7 3.0 6.1 2.3 2
4.6 3.2 1.4 0.2 0
6.0 2.9 4.5 1.5 1
```

- Example Validation Data (validation.txt):

```
6.1 2.8 4.0 1.3 1
5.5 4.2 1.4 0.2 0
6.3 3.3 4.7 1.6 1
```

- Example Run Command: `python id3.py training.txt validation.txt`
- Example Output:

```
1
```

- Write your report such that a peer NOT taking this course would understand the problem, your approach to solving it, justification of various choices, and your final comments.
- Include **all** of the plots requested above in your report.
- Include at least one figure to illustrate the ID3 method.
- All sources must be properly cited; failure to do so may result in accusations of plagiarism.
- Your report should be submitted in PDF format.

Submission

- A zipped file (.zip) containing (with **exact** filenames):
  - `id3.py`
  - `report.pdf`
- Typical command to zip your lab: `zip OLA3.zip id3.py report.pdf`
- Download your zip file and then use your PipelineMT credentials to log in and submit your zip file to the Open_Lab_3 dropbox: https://jupyterhub.cs.mtsu.edu/azuread/services/csci4350-assignments/