CSCI 4350 - Open Lab 3

Supervised Learning

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 testing data.
    - The program should take **2 command-line arguments** (string: input training data filename, string: input testing data filename)
    - The program should read in the **training data** file (one training example per line, see below)
    - The program should read in the **testing data** file (one testing example per line, see below)
    - Each line will contain the **real-values features** for several attributes and a single interger **class label** at the end
    - The program should build an ID3 decision tree by:
        - **Sorting** the training data along **each attribute**,
        - Determining **potential *binary* split points** based on **attribute value changes** (average the two values to make the split: those examples less than the split value and those examples greater than or equal to the split value),
        - **Calculating** the associated **information gain** for **all** of these potential split points, and chose 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
            1. the probability of one of the class labels is 1 (all others zero) or
            2. there are no more potential split points found among the attributes (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 testing examples should be classified using the resulting decision tree.
    - The program should then output only a single integer value: the **number of testing examples** classified **correctly** by the decision tree.
2. Utilize your program to perform **cross-validation analysis** (specifically, repeated random sub-sampling) on the **iris and cancer** data sets (see below)
    - Use bash scripting tools (see split.bash) to create 100 training and testing sets of size (m-n) and n, respectively, where **m** is the **total** number of training examples in the data set, and **n** is the desired number of testing examples.
    - For the iris data set use n=[1,5,10,25,50,75,100,125,140,145,149] and for the cancer data set use n=[1,5,10,25,50,75,90,100,104].
    - Calculate the **mean and standard error** (stddev / sqrt(100)) of the percentage of testing examples correctly classified by your decision trees for each data set.
    - Use an iPython Notebook and/or Matplotlib or other tools to **plot** the mean (+/- 1.96*standard error) for each test set size (n).
3. Write a report (at least 2 pages, single spaced, 12 point font, 1 inch margins, no more than four 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 are here: OLA3-support.zip
- You should utilize the Iris data set to build your ID3 agent (download: 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 (download: 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 the relevant information for assignments as defined in the syllabus.
- 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 Testing Data (testing.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: `./id3.py training.txt testing.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 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

- **Due Date: Thursday. Nov, 11 by 11:00pm**
- Use your PipelineMT credentials to submit your assignment at: https://jupyterhub.cs.mtsu.edu/azuread/services/www/csci4350/assignment-system/public_html/

- A zipped file (.zip) containing:
  - id3.py
  - report.pdf

Joshua L. Phillips
Home Page | CS | MTSU

Last Modified: November 02 2021 17:49:01