ML 4375 – Intro to Machine Learning – Mazidi

Project 1 Overview

Worth 200 points

For this project you will be implementing two machine learning algorithm in C++ and comparing the results and performance to using the equivalent function in R.

Turn in your R scripts, cpp files, and report, zipped together.

Project Component	Details	Max Points
Logistic Regression R	2+ graphs; 4+ data exploration functions; algorithm; metrics	30
Logistic Regression C++	Algorithm from scratch; metrics	60
Naïve Bayes R	2+ graphs; 4+ data exploration functions; algorithm; metrics	30
Naïve Bayes C++	Algorithm from scratch; metrics	60
Report	3+ pages; For each algorithm: compare/contrast R/C++ run times; include screen shots of timing runs; include screen shots of R graphs	20

Notes:

- *This is an individual project, your code must be uniquely your own*
- More details for each part in the following pages

Timing Algorithms in R/C++:

- Indicate in your Report how you computed run times. Here are some suggestions:
    - For the R script you can use Sys.time() at the start and end of the machine learning part of the script and subtract the difference.
    - For the C++ program, you may calculate it with something like chrono, shown below

```
#include <chrono>
```

…

```
auto start = high_resolution_clock::now();
```

// run the algorithm

```
auto stop = high_resolution_clock::now();
std::chrono::duration<double> elapsed_sec = stop-start;
cout << "Time:" << elapsed_sec.count() << endl;
```

Note: The timing for the R code should be only that portion running the algorithm, not parts that run data exploration functions or create graphs.

Project Component: Logistic Regression

- Data: Titanic data set "titanic_project.csv" on Piazza. Use the first 900 observations for train, the rest for test.
- R script:
  - train a logistic regression model on the training data, survived~pclass, using glm()
  - print the coefficients of the model
  - test on the test data
  - print metrics for accuracy, sensitivity, specificity
  - create 2+ graphs and run 4+ data explorations functions
  - only include the training portion of the script in your timing
- C++ program:
  - implement in C++ the same steps for logistic regression from scratch (see p. 116-117)
  - feel free to use whatever data structures you like: arrays, vectors, etc.
  - For matrix multiplication, you may want to check out the Armadillo library or the Eigen library (links below), although I've noticed that if students write the matrix multiplication from scratch it runs much much faster
  - http://arma.sourceforge.net/
  - https://eigen.tuxfamily.org/index.php?title=Main_Page
  - feel free to use whatever programming paradigm you like: oo, procedural, etc., but oo will probably slow down the results
  - calculate and output metrics for accuracy, sensitivity, specificity
- Report section for logistic regression
  - Write a summary of the two implementations, R and C++. Did you get the same results? How do the run times compare? How did you measure execution time?
  - Include screen shots of the output of each program
  - Include screen shots of the run times of each program
  - Include screen shots of your R graphs

Project Component: Naïve Bayes

- Data: Titanic data set "titanic_project.csv" on Piazza. Use the first 900 observations for train, the rest for test.
- R script:
  - train a naïve Bayes model on the train data, survived~pclass+sex+age
  - print the model, which will show all the probabilities learned from the data
  - test on the test data
  - print metrics for accuracy, sensitivity, specificity
  - create 2+ graphs and run 4+ data explorations functions
  - only include the training portion of the script in your timing
- C++ program:
  - implement naïve Bayes in C++; the code on pages 140-144 should help
  - train/test on the same data as in the R script

- calculate and output metrics for accuracy, sensitivity, specificity
- Here is a great video that gives a conceptual picture of naïve Bayes with Gaussian predictors: https://www.youtube.com/watch?v=r1in0YNetG8
- The following formula shows how to calculate the likelihood of a continuous predictor. Section 7.10.3 can serve as pseudocode for implementing this.

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

Component: Report

- For each algorithm, Logistic Regression and Naïve Bayes:
  - Write a summary of the two implementations, R and C++. Did you get the same results? How do the run times compare? How did you measure execution time?
  - Include screen shots of the output of each program
  - Include screen shots of the run times of each program
  - Include screen shots of your R graphs for each algorithm
  - No required format for the report