CENG 499 – Introduction to Machine Learning

Spring 2017 – Homework 1

Linear Regression and Weighted k-NN Classification

Selim Temizer

Feedback : Between April 17th and April 21st, 2017
Due date : April 23rd, 2017 (Submission through COW by 23:55)

Part 1. (30 points) Analytical Linear Regression Solver

Suppose that a single instance of a data set $\mathfrak{D}$ with n instances is described by attributes a_k, where $a_k \in \mathbb{R}, 1 \leq k \leq d$, and a class label b, where $b \in \mathbb{R}$. We first augment the attributes by a virtual attribute $a_0 = 1$, and then find weights w_k, such that $w_k \in \mathbb{R}, 0 \leq k \leq d$, and the ***mean squared error*** , $E(\mathfrak{D})$, that is defined as follows, is minimized:

$$E(\mathfrak{D}) = \frac{1}{n}\sum_{\mathfrak{D}}\left(b - \sum_{k=0}^{d} w_k a_k\right)^2$$

In other words, we would like to find the column vector $\boldsymbol{w}$ in the equation $\boldsymbol{Aw} = \boldsymbol{b}$, where $\boldsymbol{A}$ is the augmented attributes matrix and $\boldsymbol{b}$ is the column vector of real valued class labels. Write an application (in your favorite programming language and ***using any matrix library*** of your choice), which takes the name of a data file in comma-separated-values (CSV) format (a sample data file is attached to the homework bundle), solves and prints $\boldsymbol{w}$, and the ***root mean squared error*** (rms error) for the data set. A sample invocation is as follows, and a sample output is attached :

```
C:\499\HW1\Part1> AnalyticalSolver.exe  Data-Grades.csv
```

Part 2. (30 points) Iterative Linear Regression Solver

Solve the same problem as in Part 1, but this time ***don't use any matrix libraries and/or matrix operations***. Instead, initialize $\boldsymbol{w}$ to be the zero vector, and apply a gradient descent process, moving in the opposite direction of the gradient, with a step size of a constant factor $\mu \in \mathbb{R}$ multiplied by the negative of the gradient at $\boldsymbol{w}$. Note that the derivative of $E(\mathfrak{D})$ with respect to any w_j, where $0 \leq j \leq d$ is defined as follows:

$$\frac{\partial E(\mathfrak{D})}{\partial w_j} = \frac{2}{n}\sum_{\mathfrak{D}} -a_j\left(b - \sum_{k=0}^{d} w_k a_k\right)$$

And the update rule for $\boldsymbol{w}$ can be summarized as (update $\boldsymbol{w}$ after every iteration) :

$$w_j = w_j - \mu\frac{\partial E(\mathfrak{D})}{\partial w_j}$$

A sample invocation is as follows, and a sample output is attached :

```
C:\499\HW1\Part2> IterativeSolver.exe  Data-Grades.csv  0.00001  0.000001
```

The first argument is the data file as before, the second double value is μ, and the third double value is a continuation condition for the iterative process (minimum change in rms error). After each iteration, if the absolute difference between the rms error for the previous iteration and the current iteration is less than this value, then the iterative process should stop. Also, your solver should print useful information during the iterations.

Part 3. (30 points) Weighted k-NN Classifier

Write an application that takes a data file as before, the value for k for a k-NN predictor, and the attribute values of a query of interest, and computes and prints a weighted real valued answer for the query. Your application should also print out useful information during the solution process. A sample invocation is as follows, and a sample output is attached :

```
C:\499\HW1\Part3> WeightedKNNSolver.exe  Data-Grades.csv  5  91  92  93  94  95
```

As a technical note, the weights can just simply be taken as $(\text{EuclideanDistance})^{-1}$, and if the Euclidean distance between the query and some other instance turns out to be zero or a very small number, it can be taken as 0.01 (some small real number).

Part 4. (10 points) Financial Predictions

Convert the given financial data files for 2016 into appropriate formats, train the models in Parts 1 to 3 for them, and report your accuracies for each model using the data for 2017.

What to submit? (Use ***only ASCII characters*** when naming your files and folders)

1. Create a separate directory for each part (Part1 to Part4). Within each part, create a directory named "***Source***" that contains your source code, a second directory named "***Docs***" that contains special instructions on how to compile/run your application (if any) and your additional assumptions (if any) as a **WORD** or a **PDF** file, and a third directory named "***Bin***" that contains the compiled code that is ready to run. Note that when coding your applications, if you are using an object oriented language, DO NOT use name spaces and/or packages that may cause difficulty in compiling. Also, DO NOT submit IDE created project files or any irrelevant files.
2. For Part 4, include your converted input files, and include a short report as a **WORD** or a **PDF** file that summarizes your test procedure and your success (in terms of accuracies of results).

Zip the 4 directories, (tar also works, but I prefer Windows zip format if possible), name the compressed file as <ID>_<FullNameSurname> (with the correct extension of .zip or .tar) and submit it through COW. For example:

e1234567_SelimTemizer.zip

There are a number of design decisions and possible opportunities for visual illustrations and creative extensions for this homework. For example, you may plot the rms error versus the iteration number, or you may dynamically change the value of μ for a more efficient gradient descent process. There will be bonuses awarded for all types of extra effort that is documented appropriately. **Late submissions will not be accepted**, therefore, try to have at least a working baseline system by the deadline. Good luck.