Introduction

This assignment explores both the Pima Indians diabetes and MNIST dataset.

**Pima Indians diabetes**

*Dataset details* [1]

Features Include:

Pregnancies: how many times person in questions was pregnant

Glucose: Plasma glucose concentration over 2 hours in an oral glucose tolerance test

BloodPressure: Diastolic blood pressure (mm Hg)

SkinThickness: Triceps skin fold thickness (mm)

Insulin: 2-Hour serum insulin (mu U / ml)

BMI: Body mass index / body mass index (weight in kg / (height in m) ²)

Diabetes PedigreeFunction : Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)

Age: age of responder

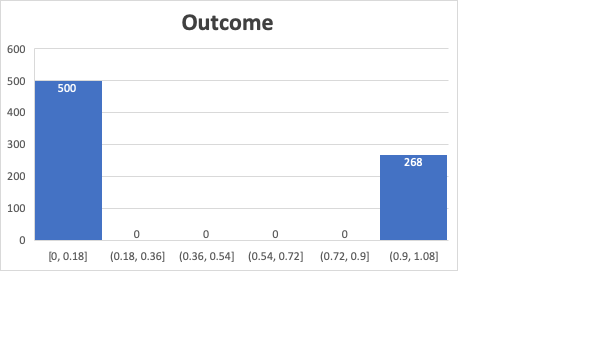
Outcome: Class variable (0 or 1), 0 for not having diabetes, and 1 with diabetes.

Also, from lecturer slides, used suggestion of training data (%80 of data), validation data (%10 of data) plus test data (%10 of data).

Graphs:

Above shows glucose level based those with diabetes.

Above shows BMI based those with diabetes.



Both graphs shows age breakdown of those with diabetes and the amount of persons living with vs without diabetes.

Algorithm Description:

There were a number of values in dataset that should not have been zero (such as Glucose, bloodPressure, SkinThickness, Insulin and Diabetes PedigreeFunction). These were replaced by the mean of the rest of the dataset for each feature. The mean was a total of non-zero values divided by the count of those non-zero values (calculated for each feature).

Feature scaling formula:

trainset[j][x] – low[x] / (high[x]-low[x]) ,*where j is an entry in train dataset and x is set of features*, was used to get features within [0,1] range.

Algorithm Results:

Euclidean:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Algorithim Classified | | |
| Actual Outcome | 19 | | 10 |
| 8 | | 40 |
| Manhattan |  | |  |
|  | Algorithim Classified | | |
| Actual Outcome | 0 | 29 | |
| 0 | 48 | |

Minkowski (with p = 2)

|  |  |  |
| --- | --- | --- |
|  | Algorithim Classified | |
| Actual Outcome | 15 | 14 |
| 8 | 40 |

Runtime:

T(n) = i\*n\*m where i is the validation/test dataset; n is size of training data and m is size of feature set

Wall clock: 222.29s user 1.88s system

**MNIST Dataset**

*Dataset details*[2]:

Features noted below:

Each pixel column in the training set, represents a feature, has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

Also, from lecturer slides, went with suggestion of training data (%80 of data), validation data (%10 of data) plus test data (%10 of data).

Graphs:

Above shows the numbers of digits in training set per digit.

Above shows the rgb total for each digit.

Algorithm Description:

There was no preprocessing done on dataset has it seemed that values were within valid range 0 (dark/low) to 255 (light/high).

feature scaling formula :

trainset[j][x] – low[x] / (high[x]-low[x]) where j is an entry in train dataset and x is set of features

Algorithm Results

Confusion matrix

((0.0, 5.0), 2), ((0.0, 6.0), 2), ((1.0, 7.0), 2), ((1.0, 8.0), 2), ((1.0, 9.0), 2), ((4.0, 2.0), 2), ((4.0, 7.0), 2), ((5.0, 8.0), 2), ((6.0, 3.0), 2), ((7.0, 9.0), 2), ((8.0, 9.0), 2), ((1.0, 2.0), 4), ((1.0, 5.0), 4), ((3.0, 5.0), 4), ((6.0, 5.0), 4), ((9.0, 5.0), 4), ((9.0, 7.0), 4), ((9.0, 8.0), 4), ((3.0, 8.0), 6), ((9.0, 3.0), 6), ((9.0, 4.0), 10), ((5.0, 5.0), 60), ((4.0, 4.0), 64), ((8.0, 8.0), 66), ((3.0, 3.0), 68), ((0.0, 0.0), 70), ((7.0, 7.0), 72), ((9.0, 9.0), 74), ((2.0, 2.0), 80), ((6.0, 6.0), 90), ((1.0, 1.0), 104)

Runtime:

T(n) = i\*n\*m where I is the validation/test dataset; n is size of training data and m is size of feature set (Note feature set was considerably larger for dataset causing algorithm to be costly)

Wall clock: 5774.08s user 58.36s (using all of dataset took too long to run for all features - used 4100 records of datatset instead)

References

[1]

<https://towardsdatascience.com/end-to-end-data-science-example-predicting-diabetes-with-logistic-regression-db9bc88b4d16>

[2]

<https://www.kaggle.com/c/digit-recognizer/data>