Introduction to Data Science

Course 094201

Lab 7:

Data Normalization

Spring 2017

Data Normalization - Motivation

- Often the different variables (features) are measured in different units and thus have large differences in scale
  - For example, today we will work with a dataset where each item is a patient represented by:
    - Diastolic blood pressure (mm Hg)
    - Age in years
    - 2-Hour serum insulin (mu U/ml)
    - And more…
- Most distance metrics (e.g.: Euclidean distance, L1-norm) will be affected: larger scaled features will dominate the others
  - Unless there is some explicit feature weighting, all features should have the same importance before a model is learned

Data Normalization - Example

- Distance based classifiers will be biased towards features with larger scale
- Classifiers which use different numerical algorithms might exhibit problems
  - For example, the perceptron algorithm we learned uses gradient descent which might have convergence time issues as a result of different scales of features
- **Solution: we want to normalize all the variables (features) to the same scale**

Data Normalization – to unit L_p norm

Given a dataset where:

X is a quantitative variable (e.g., age), x_i is the value for item i (e.g., x_i=26)

n is the number of items in the dataset

We divide each value by the norm of the vector (x_1, x_2,…, x_n)

L1-norm - sum normalization	**L2-norm**
$x_i^{L_1} = \frac{x_i}{\sum_{1 \le i \le n} x_i}$	$x_i^{L_2} = \frac{x_i}{\sqrt{\sum_{1 \le i \le n} x_i^2}}$

Data Normalization – Z Norm

- We **standardize** all the points by using the mean and standard deviation of each feature.

$$x_i^{z-score} = \frac{x_i - \mu}{\sigma}$$

- μ is the population mean -> can be estimated using $\bar{x}$
- σ is the population standard deviation -> can be estimated using s
- The resulting values have the mean of 0 and quantify the distance between x_i and μ in units of σ

Data Normalization – Min-Max Norm

- Widely used is min-max normalization, also called range normalization
- For each variable we use its minimum and maximum values:
  - We first **shift x_i to the origin**
  - We **scale the shifted value by its range**. The result is in [0,1]

$$x_i^{min-max} = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}$$

The dataset and the code

- The code and the data can be found at:

/mnt/share/students/LAB7

- Copy everything to your **local folder** and unzip the code:

```
unzip lab7-students.zip
```

The only parameter of the code is the input file

This code is based on the last lab and emphasizes the need for data normalization.

You are asked to add **the normalization** functionality and check how the KNN classifier’s effectiveness is affected

The data - Pima Indians Diabetes Database

The data contains information regarding patients. All patients here are females at least 21 years old of Pima Indian heritage. The class we want to learn is the **presence of diabetes.**

Attribute Information:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Reference: Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

Assignment

- Implement an abstract class Normalizer which contains 2 **pure virtual** functions:
  - Init: a function that receives a vector of points and set up the normalizer parameters
    - E.g.: for sum-normalization we need a point that will hold the sum of each variable
  - Normalize: a function that receives a point and returns a normalized version of that point
- Implement the operator/ (division) in Point.cpp (look at operator- for reference)
- Compile and run the program
- Make sure you understand the flow in main()

Home assignment

- Implement the 2 remaining classes which are derived from Normalizer, one for each normalizing method:
  1. Sum-normalization (L_1)
  2. Min-max normalization
- Add the code that calls to the runKNN function with the two additional normalizers. How does polymorphism help reusing code?
- Run 5-NN and 7-NN on the data and report accuracy
  1. Without normalization
  2. With each of the 3 methods above
  3. What is your conclusion?
- Answer the moodle LAB7 quiz on C++