Wine Quality Prediction with Single Layer Neural Network

Barışcan Tunalı
Big Data Masters Student
Istanbul Bahcesehir University,
bariscantunali@outlook.com

ABSTRACT

In this paper, I will explain my study on Wine classification using Single Layer Neural Network and compare the results with K-NN, Nearest Cluster, multi layer neural network (SPSS).

Our problem was predicting categorical value (quality) from continious variables which will be explained later. Quality score was given by some Wine Expert to those wines. Main dataset has no test set so we need to extract our test set myself.

My dataset had 2 datasets one of them was for red wine and the other is for white wine. So since datasets were small, instead of combining them programmaticaly I ran algorithm on them differently, Only transformed them to common wine class not to duplicate functions.

Keywords

K Nearest Neighbour, K-NN,KNN, Single Layer Neural Network, Artificial Neural Network, Nearest Cluster, Data Mining, Classification, Prediction, Wine, ANN

1. INTRODUCTION

My first challenge was domain itself, Wine quality is very hard to decide because there are lots of other qualifications other than chemical properties such as grape type, location, sun, wait time in barrels, wait time in bottle etc. So the prediction will be on only chemical attributes and based on first wine expert(s) who has qualified them. Another wine expert might not like the result.

Or since even slightest changes can change quality very much, test wine may be very bad wine even ANN value gives high score. Such advanced prediction requires very advanced mathematical modelling if we include every parameters starting from grape seed to wine in your table.

The paper is organized as follows: In Section 2, we briefly describe the dataset and data preparation. Next, in Section 3, we explain our prediction approach in detail. Experimental results and comparisons with other algorithms are provided in Section 4. Finally, Section 5 concludes the paper.

2. DATASET AND FEATURES

Red Wine dataset constists of 1599 values, White Wine dataset consists of 4898 values. I had 12 attributes, one of them was quality. But after looking data I eleminated one of them because it had very same values and I removed it because it wouldn't useful on prediction (Citric Acid).

My attributes are : [FixedAcidity], [VolatileAcidity], [CitricAcid], [ResidualSugar], [Chlorides], [FreeSulfurDioxide], [TotalSulfurDioxide], [Density], [PH], [Sulphates], [Alcohol], [Quality]

Attribute name	MAX	MIN	DISCINT
[FixedAcidity]	15,9	4,6	96
[VolatileAcidity]	1,58	0,12	143
[CitricAcid]	1	0	2
[ResidualSugar]	15,5	0,9	91
[Chlorides]	0,611	0,012	153
[FreeSulfurDioxide]	72	1	60
[TotalSulfurDioxide]	289	6	143
[Density]	1,00369	0,99007	436
[PH]	4,01	2,74	89
[Sulphates]	2	0,33	96
[Alcohol]	14,9	8,4	65
[Quality]	8	3	6

Table 1 : Basic Attribute Statisticts

Except quality all of the attributes were continious so I standardized them with mean stardardization ($x\text{-}x_{mean}$ / x_{stddev}).

In order to make my program more flexible I created a class and handled all data operations, test train seperations, standardizations etc. On presentation layer you only create class and call `Initialize()` function. After that you only access necessary train and test sets and send them to appropriate classification algorithms.

```
RedWineTrainSet = dprp.RedWineTrainSet;
RedWineTestSet = dprp.RedWineTestSet;
Perceptron pct = new Perceptron ();
pct.RedWineTrain(RedWineTrainSet, RedWineTestSet)
```

Data preparation class structure as below :

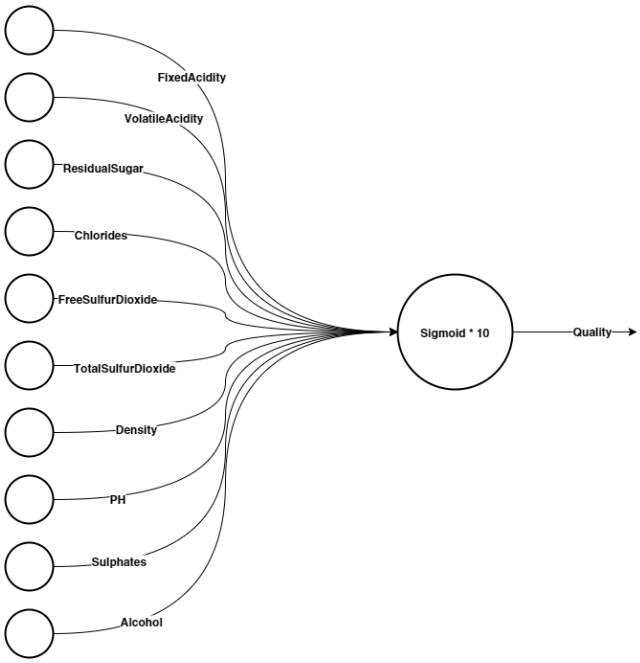
```
public void Initialize()
{
  ExtractData();
  TransformData();
  NormalizeData(RedWineOriginalSetTransformed);
  NormalizeData(WhiteWineOriginalSetTransformed);
  SeperateTestData(RedWineOriginalSetTransformed);
  SeperateTestData(WhiteWineOriginalSetTransformed);
}
```

ExcractData() : Dataset was in .csv format but to access and get data faster I inserted them to Microsoft SQL Server 2014. I have two tables as Red and White. This function uses ADO.NET Entity

Framework to get data and inserts them to my data lists for further processes.
TransformData() : Gets list of different data classes and converts them to common Wine.cs class.
NormalizeData() : Calculates standart deviations, means and standardizes each column except quality score.
SeperateTestData() : Gets normalized data and randomly seperates %20 of data as test and %80 as train.

3. METHOD

Basic structure of algorithm is shown below :



Algorithm has 3 functions :

```
public string WineTrain(List<Wine> TrainSet,List<Wine> TestSet);
private double Output(double[] weights, double FixedAcidity,
                      double VolatileAcidity,double ResidualSugar,
                      double Chlorides, double FreeSulfurDioxide,
                      double TotalSulfurDioxide, double Density,
                      double PH, double Sulphates, double Alcohol,
                      double Quality);

public double LogSigmoid(double x);
```

`LogSigmoid :` This is our activation function. Gets computed output and gives logsigmoided result. But I needed to distribute my output from 0 to 10, LogSigmoid gives 0 to 1 range. So I multiplied output with 10.

`Output :` Gets attributes and computes output value depending on neural weights.

`WineTrain :` Main function. Iteration count is 10000, learning rate is 0.01, local error limit is 3. These are my general parameters of neural network.

More general algorithm of neural network :

```
Weights <- Random
do
{
   for (int p = 0; p < TrainCount; p++)
   {
         Output()
         CalculateLocalError()
         if (Math.Abs(localError) > Limit)
                  UpdateWeights()

         }
}while ( iteration!= iterationLimit)
```

I didn't use any c# library or dll to implement ann itself but I used lambda expressions to avoid code complexity. Especially on data preparation part.

I assumed if (prediction-actual)<1 its correct else wrong. And summed all error differences and calculate model error average.

4. EXPERIMENTS

I implemented 10 run loop, which creates new class, new train and test sets. On presentation layer I only call knn class and print out results into a file. On results I counted correct predictions, I decided that if Abs(prediction-real score)<=1 then I took it as correct prediction else false prediction.

5. CONCLUSIONS

Here are 2 run results for knn and cluster for red and white wine sets to compare with our neural network :

Red Wine Individual
Average Euclidean Error : 0,495297805642633,
Correct Count : 303(94%),
False Count : 16(5%)
Red Wine Individual
Average Manhattan Error : 0,495297805642633,
Correct Count : 303(94%),
False Count : 16(5%)
Red Wine Individual
Average Minkowski Error : 0,70846394984326,
Correct Count : 297(93%),
False Count : 22(6%)

Red Wine Nearest Cluster
Average Euclidean Error : 0,70846394984326,
Correct Count : 280(87%),
False Count : 39(12%)
Red Wine Nearest Cluster
Average Manhattan Error : 0,70846394984326,
Correct Count : 283(88%),
False Count : 36(11%)
Red Wine Nearest Cluster
Average Minkowski Error : 0,70846394984326,
Correct Count : 280(87%),
False Count : 39(12%)

As we see here normal knn can guess with more than %95 accuracy averagely, and clustering can guess with about little less than %90 accuracy. There is a trade off between these algorithms knn's runtime is very high, it compares test wine with every wine in train set and then sorts them so run time is about $O(n + O_{sort}(n))$ so on real time applications or even nightly jobs its cost very high on large datasets.

Single layer network results :
Red Run 1
Average Error Per Item : 0,798549056507714, Correct Count : 202(63%), False Count : 117(36%)

Initial Weights :
FixedAcidity: 0,877040329332017
VolatileAcidity : 0,850316700921541
ResidualSugar : 0,414968327346709
Chlorides : 0,781091955388473
FreeSulfurDioxide : 0,350516948546524
TotalSulfurDioxide : 0,263109272002759
Density : 0,232024765215826
PH : 0,682838269361685
Sulphates : 0,245234213883632
Alcohol : 0,962397660576924

Final Weights :
FixedAcidity: 0,11766870232296
VolatileAcidity : -0,0479458257476297
ResidualSugar : 0,0541758139095643
Chlorides : -0,0532465206575621
FreeSulfurDioxide : -0,0455423229417415
TotalSulfurDioxide : 0,0524096915977025
Density : -0,247003210889526
PH : -0,0344735771456827
Sulphates : 0,148567852357593
Alcohol : 0,0138604545806748

White Run 1
Average Error Per Item : 0,605525163429125, Correct Count : 804(82%), False Count : 175(17%)

Initial Weights :
FixedAcidity: 0,412438684335183
VolatileAcidity : 0,129253830820906
ResidualSugar : 0,0323225939796877
Chlorides : 0,661892240709575
FreeSulfurDioxide : 0,581307428228346
TotalSulfurDioxide : 0,993666601364346
Density : 0,124531182984138
PH : 0,998324991668726
Sulphates : 0,105786901947943
Alcohol : 0,200814217888198

Final Weights :
FixedAcidity: 0,0267088473564369
VolatileAcidity : -0,0703350599589893
ResidualSugar : 0,0442362763684888
Chlorides : -0,0325790471239372
FreeSulfurDioxide : 0,104757377619064
TotalSulfurDioxide : -0,0508428262758203
Density : -0,00096413283934417
PH : 0,0327336279450971
Sulphates : 0,0387684984799634
Alcohol : 0,348156302318621

Red Run 2
Average Error Per Item : 0,738616522288756, Correct Count : 472(73%), False Count : 167(26%)

Initial Weights :
FixedAcidity: 0,304247384566929
VolatileAcidity : 0,754045191106408
ResidualSugar : 0,850976142962918
Chlorides : 0,894636550403497
FreeSulfurDioxide : 0,373753710358289
TotalSulfurDioxide : 0,391533844820938
Density : 0,210282569383403
PH : 0,337329601094746
Sulphates : 0,173992725170214
Alcohol : 0,089364736848215

Final Weights :
FixedAcidity: 0,0591792626196787
VolatileAcidity : -0,0822420583883561
ResidualSugar : -0,0611031199669275
Chlorides : -0,0488443348993188
FreeSulfurDioxide : -0,00193145554306245
TotalSulfurDioxide : -0,0378886125839179
Density : 0,00338113851058149
PH : -0,0160917710596649
Sulphates : 0,114164277156144
Alcohol : 0,0564125773429771

White Run 2
Average Error Per Item : 0,849456040135093, Correct Count : 1298(66%), False Count : 661(33%)

Initial Weights :
FixedAcidity: 0,643185375092172
VolatileAcidity : 0,486567771754492
ResidualSugar : 0,226907891327007
Chlorides : 0,141846130202453
FreeSulfurDioxide : 0,188597562810684
TotalSulfurDioxide : 0,200772424787643
Density : 0,264936369035829
PH : 0,0708625680165657
Sulphates : 0,358513366132282
Alcohol : 0,795233904288725

Final Weights :
FixedAcidity: -0,207375249212474
VolatileAcidity : -0,0456766431119792
ResidualSugar : 0,144669596885105
Chlorides : -0,0641394295172005
FreeSulfurDioxide : -0,231442558120058
TotalSulfurDioxide : -0,144659909361244
Density : 0,162450810151663
PH : 0,0324924401484647
Sulphates : 0,0125509247164137
Alcohol : 0,41697336126659

Multi layer network results (SPSS) :
SPSS doesn't allow to build single layer networks, the simplest network details has shown below.
Correct Percent : %62,3 Wrong Percent : %37,7
SPSS Neural network for Red Wine :

Hidden layer activation function: Sigmoid

Output layer activation function: Sigmoid

Weight matrix for network :

		Output Layer					
		[Quality=3]	[Quality=4]	[Quality=5]	[Quality=6]	[Quality=7]	[Quality=8]
Hidden Layer 1	(Bias)	-1,348	-1,357	,228	-1,503	-1,155	-1,439
	H(1:1)	-1,018	-,770	-2,267	2,480	1,506	,098
	H(1:2)	-1,115	-,226	,646	-,524	-1,388	-1,127
	H(1:3)	-,963	-,546	-,375	-,563	-,928	-,804
	H(1:4)	-1,030	-,363	-,103	-1,033	,980	-,317
	H(1:5)	-,473	-,282	-,246	,506	-,152	-,456
	H(1:6)	-,955	-,715	,744	1,207	-2,030	-1,007
	H(1:7)	-,290	-,208	1,534	-,863	-,931	-,376
	H(1:8)	-,820	-,663	-,678	-,422	,302	-,614

		Hidden Layer 1							
Predictor		H(1:1)	H(1:2)	H(1:3)	H(1:4)	H(1:5)	H(1:6)	H(1:7)	H(1:8)
Input Layer	(Bias)	1,096	,638	,189	-1,073	-,032	1,886	-1,239	,602
	FixedAcidity	1,371	-,318	-,270	,564	-,169	-,671	,504	,610
	VolatileAcidity	-,835	,816	,533	-,892	-,262	,797	,031	,263
	ResidualSugar	,669	-,255	-,146	,052	-,050	-1,152	,825	-,039
	Chlorides	-1,524	,008	,050	-,175	,693	,415	-,644	-,060
	Alcohol	2,938	-,731	-,269	,949	-,491	-,172	,297	,475
	Sulphates	,441	,079	-,700	-,094	,470	-1,247	-,925	,418
	FreeSulfurDioxide	,394	-,563	,266	-,289	,710	,374	,022	,172
	PH	-,193	,188	,484	-,244	,329	,406	-1,593	,129
	TotalSulfurDioxide	-1,542	,345	,470	,400	,112	,406	1,179	-,707
	Density	-,251	,904	-,359	-,555	,459	,883	-,187	-,146

Comparison neural network with spss and knn. We take our accuracy as %95 for knn and %90 for clustering.
As we see here if we increase complexity of prediction algorithms our prediction rate decreases. On single layer neural network our error mean is similar with clustering but count of prediction is lower than both of them. If we predict them using spss results are worse. We clearly see Occam's razor principle on this particular dataset. Simpler, better.

6. REFERENCES

Wine Quality Data Set : UCI Machine Learning Repository