USC ID#: 3647897461

4/30/2017

Qilin You

UNIVERSITY OF SOUTHERN CALIFORNIA

EE559 Project

German Credit dataset

[qilinyou@usc.edu](mailto:qilinyou@usc.edu)

Qilin You

**Abstract**

Firstly, I need to preprocess dataset1 and dataset2 and convert qualitative type into numerical type. I replace missing data with the mean of corresponding features and take PCA and normalization to process data. After that, I divide dataset into test set, training set and validation set, I pick out optimal parameters via cross validation. To observe performance of different classifiers, I conduct minimum-distance classifier, discriminant analysis(Quadratic), naïve Bayes classifier, SVM, K-NN and MPL neural network. The performance of nonlinear classifier is similar and much better than linear classifier’. For dataset1, best performance is achieved by Naïve Bayes classifier and mean accuracy is 72.00%, F-measure on two classes are 0.82 and 0.39.SVM also achieve similar performance in dataset1. For dataset2, best performance is achieved by SVM and mean accuracy is 76.5%, F-measure on two classes are 0.84 and 0.58.

**Preprocessing**

As we can see that dataset includes qualitative type and numerical type data, the classification is based on numerical type data, thus, we need to convert them into numerical type.

There two situations: for categorical features having natural order, I transform them into numerical order; for categorical features which don’t possess natural order, I convert them in to a binary vector.

For example: (bases on dataset1)

Saving account has 4 different categories beside ‘NA’ representing missing data: ‘little’, ’moderate’, ’rich’ and ’quite rich’.

It’s obvious that this qualitative feature has a natural order, thus, I convert them according to followed rules:

‘little’ -> 1;

‘moderate’ -> 2;

‘rich’ -> 3;

‘quite rich’ ->4;

In this case, we should deal with missing data, for ‘NA’ exist. There are many other methods, such as using mode as a substitution, regarding it as new feature, applying k-NN to pick out neighborhoods and get their mean as substitution and so on. To simply the processing, I replace the missing with the mean of corresponding features.

Purpose has 8 different categories: ‘business’, ‘car’, ‘domestic appliances’, ‘education’, ‘furniture/equipment’, ‘radio/TV’, ‘repairs’, ‘vocation/others’.

We have to admit that we can find confident precedence relation among them, thus, I convert them into binary vector according to followed rules:

‘business’ -> [1,0,0,0,0,0,0,0];

‘car’ -> [0,1,0,0,0,0,0,0];

‘domestic appliances’ -> [0,0,1,0,0,0,0,0];

‘education’ -> [0,0,0,1,0,0,0,0];

‘furniture/equipment’ -> [0,0,0,0,1,0,0,0];

‘radio/TV’ -> [0,0,0,0,0,1,0,0];

‘repairs’ -> [0,0,0,0,0,0,1,0];

‘vocation/others’ -> [0,0,0,0,0,0,0,1];

In addition, the preprocessing is more complex on dataset2. As several special qua-litative type data exist in dataset2.

For instance:

Attribute 6: Saving account/bonds:

‘A61’: … < 100 DM;

‘A62’:100 <= … < 500 DM;

‘A63’:500 <= … < 1000 DM;

‘A64’: … >= 1000DM;

‘A65’: unknown/ no saving account

We can see that ‘A61’, ‘A62’, ‘A63’ and ‘A64’ have an obvious order except for ‘A65’. Thus, I re-cast this qualitative type data as following shown:

‘A61’ -> [1,0];

‘A62’ -> [2,0];

‘A63’ -> [3,0];

‘A64’ -> [4,0];

‘A65’ -> [0,1];

The reason is that I think ‘unknown/ no saving account’ can’t fitted into any order in this case, thus , I consider it as another binary feature indicating whether possessing saving account/bonds.

According above rules, we can re-cast all categories feature among dataset1 and dataset2.After re-casting data, we need to make use of dataset to get an appropriate evaluation of performance.

**Features analysis**

Before getting into analysis, I conduct an exploratory data analysis and data cleaning. In this case, I make a summary bases on dataset2, since dataset1 is extracted and modified from dataset2. Thus, analysis of dataset1 is similar with dataset2.

On consideration of the number of categorical, I only show analysis of modified proportions.

The blue and yellow shading indicates that these proportions have two few samples or merged among classes for final analysis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Categorical | Proportions | | | | |
| Attribute 1 | A11 | A12 | A13 | A14 |  |
| % | 27.40% | 26.90% | 6.30% | 39.40% |  |
| Attribute 6 | A61 | A62 | A63 | A64 | A65 |
| % | 60.30% | 10.30% | 6.30% | 4.80% | 18.30% |
| Attribute 3 | A30 | A31 | A32 | A33 | A34 |
| % | 4.00% | 4.90% | 53.00% | 8.80% | 29.30% |
| Attribute 7 | A71 | A72 | A73 | A74 | A75 |
| % | 6.20% | 17.20% | 33.90% | 17.40% | 25.30% |
| Attribute 17 | A171 | A172 | A173 | A174 |  |
| % | 2.20% | 20.0% | 63.00% | 14.80% |
| Attribute 9 | A91 | A92 | A93 | A94 |
| % | 5.00% | 31.00% | 54.8% | 9.2% |
| Attribute 16 | 1 | 2 | 3 | 4 |
| % | 63.30% | 33.30% | 2.80% | 0.06% |
| Attribute 10 | A101 | A102 | A103 |  |
| % | 90.7% | 4.10% | 5.20% |  |
| Attribute 14 | A141 | A142 | A143 |  |
| % | 13.90% | 4.70% | 81.40% |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute 4(Purpose) | | | | | | | | | |
| A40 | A41 | A42 | A43 | A44 | A45 | A47 | A48 | A49 | A410 |
| 10.30% | 18.10% | 28.00% | 1.20% | 2.20% | 5.00% | 0.90% | 9.70% | 1.20% | 23.40% |

(Above result can gained from Matlab code ‘observefeatures.m’)

Most of the proportions are categorical with several levels, the full cross-classification of all variables will lead to zero observations in many cells. Hence, we need to reduce the table size. The details are shown:

Attribute 1: A11, A12, (A13 and A14);

Attribute 6: A61, A62, (A63 and A64), A65;

Attribute 3: (A31 and A32), A32, (A33 and A34);

Attribute 7: (A71 and A72), A73, A74, A75;

Attribute 17: (A171 and A172), A173, A174;

Attribute 9: (A91 and A92), A93, A94;

Attribute 16: 1, (2, 3 and 4);

Attribute 10: A101, (A102 and A103);

Attribute 14: (A141 and A142), A143;

Attribute 4(Purpose): A40, A41, (A42, A43, A44 and A45), (A47, A48, A49 and A410);

**Dataset usage (training, validation ,test)**

I conduct the most rigorously correct way of doing this referring to lecture:

Use an outer loop for the "test" and cross-validation. Inside this outer loop, put all the decisions and trials that are made in coming up with a final best system.  All these decisions and trials are then repeated anew in each iteration of the outer loop.

Actual operations:

I randomly divide dataset into 5 bins where preserving percent representation of each class, thus, there will have 5 different combination groups to pick one bin (20%:200 samples) as testing dataset and the rest act learned dataset (80%:800 samples).

Then, I conduct same operation with the above for learned dataset, divide them into 5 bins and pick one bin (160 samples) as validation set and the rest (640 samples) as training set. Thus, there are 5 different combination groups in this cross-validation.

To pick out appropriate optimal parameter vis cross-validation, I set the iteration number of cross-validation as 20 and setting parameter refer to mean accuracy mat.

After setting parameter of model, I train the model with learned dataset and use trained model to predict testing dataset.

Finally, I take the mean accuracy and f1-scores from testing dataset as evaluation of performance of classifier.

**Features selection/dimensionality reduction/normalization**

After above operations, I conduct features selection via cross-validation. With help of cross-validation, I pick out optimal dimension of Principal component analysis (PCA) for corresponding classifier model. It’s a brutal method to accomplish dimensionality reduction.

What’s more, I also conduction normalization make all features contribute equally for classifiers based on Euclidean distance, such as K-nearest neighbors, logistical regression, SVMs, perceptron’s, neural network etc.

Standardization equation:



In addition, we must apply mean and standard deviation of train dataset to normalize validation dataset in cross-validation and apply mean and standard deviation of learned dataset to normalize testing dataset in evaluation to avoid corrupt evaluation dataset. The process is same for PCA, we should use coefficient matrix of training dataset to dimensionality reduction of validation dataset.

**Performance evaluation techniques**

To get a respective appropriate evaluation about classification, I set aside 20% dataset at first by dividing original dataset into 5 bins and pick out one bin as testing dataset, thus, there will be 5 different combination groups. Therefore, we can conduct 5 different experiment and get 5 different evaluation, then, I get the mean of result as the final evaluation of classification.

In order to increase the stability of evaluation, we also build several iterations on the evaluation process and get the mean of evaluation as final evaluation result.

**Approach**

I begin classification with a baseline classifier-**minimum-distance-classifier**, which is linear classier based on Euclidean distance.

The operation is that I pick out all same class data and calculate their mean of features as centroids, then, predict testing dataset with gained centroids based on Euclidean distance.

Please refer to file ‘classifier\_MDC.m’ for more information.

**Result:**

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 65.30% | 0.73 | 0.50 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 72.40% | 0.79 | 0.61 |

It’s obvious that the linear classifier doesn’t performance well on so complex feature space. Thus, the performance is bad. What’s more, it’s reasonable that the performance based on dataset2 is better than dataset1, since dataset2 concludes more rich information and no missing data to disturb the final performance.

**Discriminant analysis (Quadratic)**

I accomplish it via ‘fitcdiscr’ function in MATLAB and set discriminant type quadratic.

Please refer to MATLAB code ‘classifier\_MAQ.c’ for more information.

**Result:**

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 71.10% | 0.81 | 0.36 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 73.50% | 0.82 | 0.51 |

We can see that the performance has improved in general, since the nonlinear classifier can handle the complex features. But, the F1scores on class2 is so bad as those many merged features degrade the performance. Thus, I refer to Proj\_dataset\_2\_documentation and try to incorporate cost matrix. Although this project doesn’t consider cost matrix, I want to analyze the effect of cost matrix.

Cost matrix:

|  |  |
| --- | --- |
| 0 | 1 |
| 3 | 0 |

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 61.20% | 0.68 | 0.51 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 67.70% | 0.73 | 0.58 |

It’s obvious that the performance about F-measure gets improved compared with the former, but the accuracy is lower. It satisfies the assumption because the cost of misclassification on class 2 is higher, thus, the misclassification on class 2 will decrease, but misclassification on class 1 will increase. Meanwhile, the dataset is unbalance among class 1 and class 2, so the accuracy will have a big reduction.

In fact, it’s practical to incorporate the cost matrix as it is worse to class a customer as good when they are bad than it is to class a customer as bad when they are good. Meanwhile, features information can get a better utilization in this case, but we need do further features analysis to improve the accuracy since the model become more complex.

**Naïve Bayes classifier** is statistical classifier, which is based on probability (density function) for features. I accomplish it via ‘fitcnb’ function in MATLAB.

Please refer to file ‘classifier\_Bayes.m’ for more information.

**Result:**

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 72.00% | 0.82 | 0.39 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 76.30% | 0.84 | 0.56 |

Cost matrix:

|  |  |
| --- | --- |
| 0 | 1 |
| 3 | 0 |

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 63.50% | 0.71 | 0.49 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 68.40% | 0.74 | 0.59 |

The performance is better than linear classifier, but the result is not satisfied as volume of sample maybe not big enough to get a predicted density distribution. Dataset1 have less features and unstable predicted density function of features will degrade the performance.

Thus, the performance in dataset1 is much worse in term of F1scores.

**K-Nearest Neighborhoods** algorithmis statistical classifier, which is approximated locally and simplest of machine learning algorithm. The value of K is crucial for performance, I get optimal K via cross-validation. Thus, the K-NN is sensitive to the local structure of the data, but no explicit training step is required. I accomplish it via ‘fitcknn’ function in MATLAB.

Please refer to file ‘classifier\_KNN.m’ for more information.

**Result:**

For dataset1: (optimal K=31)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 71.00% | 0.81 | 0.32 |

For dataset2: (optimal K=23)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 75.30% | 0.84 | 0.47 |

Cost matrix:

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 0 |

For dataset1: (optimal K=32)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 71.50% | 0.79 | 0.54 |

For dataset2: (optimal K=15)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 75.15% | 0.80 | 0.58 |

By comparing with Naïve Bayes classifier, we can see that they have similar performance, but K-NN has a relative worse performance for the K-NN is sensitive to the local structure of the data. However, the performance of K-NN can boost with help of appropriate cost matrix, because cost matrix help classifier reduce misclassification on class2 dramatically, which guild K-NN with some global information to some degree and make up disadvantages of K-NN.

I won’t incorporate cost matrix in later classifiers, since time-cost is very high for setting optimal parameters via cross-validation, which is up to several hours.

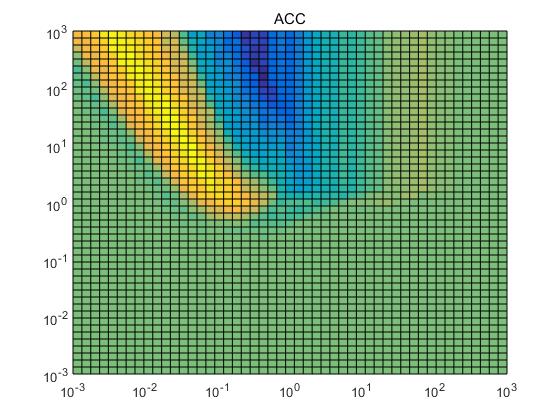
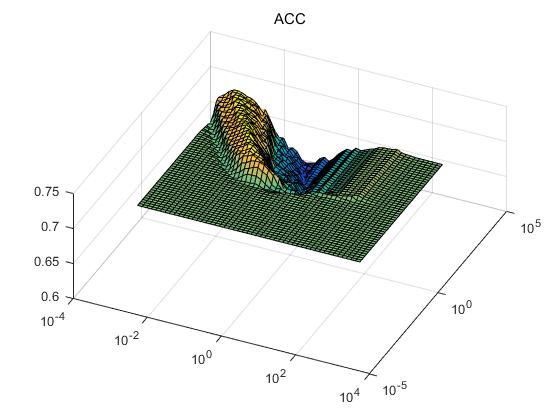
**Support Vector Machine (SVM)** is a distribution-free classifier, which can perform a non-linear classification using different kernels. In this case, I choose C-SVC model with radial basis kernel. Thus, I need to find out optimal C and gamma. I accomplish it via ‘svmtrain’ function in libsvm3.2.

Please refer to file ‘classifier\_SVM.m’ for more information.

**Result:**

For dataset1: (C=8.2864 gamma=0.0222)

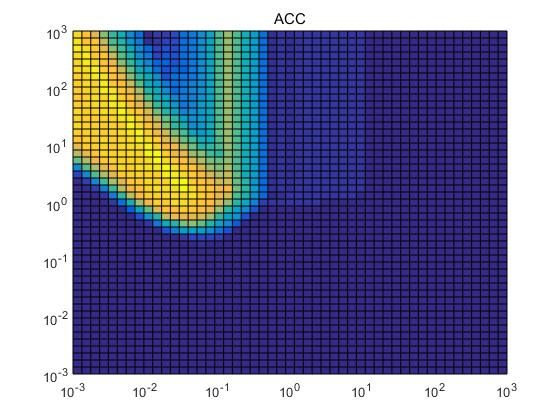
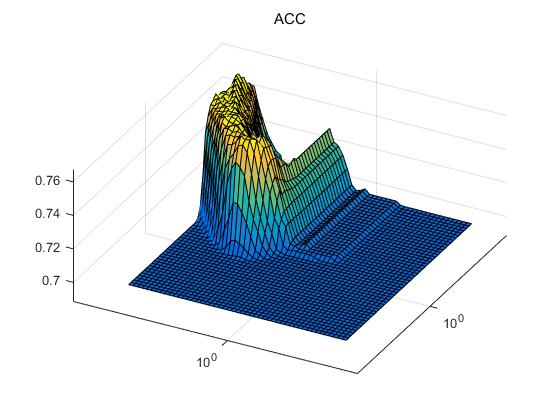
|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 72.00% | 0.82 | 0.35 |

**Figure 1:** ACC

For dataset2: (C=4.7149 gamma=0.0126)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 76.5% | 0.84 | 0.58 |

**Figure 2:** ACC

In general, SVM has best performance among these classifiers, since it makes use of finite samples and builds good decision boundary. However, we can see that the performance is highly depended on the appropriate parameters referring to figure 1 and 2. The time-cost on choosing appropriate parameters is very huge, cost efficient is much lower than K-NN and Naïve Bayes classifier even though it can achieve a better performance.

**Multiple layers neural network** is distribution-free classifier, which also take the responsibility for extracting features. Firstly, to make use of features, I prefer to set a lot of neurons in first layer. On consideration of parallel computation, I will set the number as 2n , thus, I set the number as 128. After trials, I build a relative good architecture:

|  |  |
| --- | --- |
| Number of neuron | Activation function |
| 128 | Tansig |
| 128 | Purelin |
| 128 | Tansig |
| 128 | Logsig |
| 64 | Tansig |

I accomplish it via ‘newff’ function in neural network tools 0.8 from MATLAB extension library. Please refer to file ‘classifier\_MPL.m’ for more information.

**Result:**

For dataset1:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 68.80% | 0.77 | 0.45 |

For dataset2:

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 73.90% | 0.82 | 0.56 |

The neural network doesn’t work well as expected, I think there are two main reasons. One of them is the limitation of sample volume, neural network usually need a huge amount of training data to make it work well. In this case, our sample volume is very limited, I observed the model has a high tendency to overfitting, which contribute a unsatisfied result. The other one is the architecture of model, it’s black box and I can’t give a specific good architecture to achieve an ideal performance in this case. Maybe, it will cost tons of time to find out a good architecture.

**Best result**

For dataset1:

**Naïve Bayes classifier**

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 72.00% | 0.82 | 0.39 |

**K-NN**

Cost matrix:

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 0 |

For dataset1: (optimal K=32)

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 71.50% | 0.79 | 0.54 |

For dataset2:

**Support Vector Machine (SVM)**

|  |  |  |
| --- | --- | --- |
| Accuracy | F1scores (class 1) | F1scores (class 2) |
| 76.5% | 0.84 | 0.58 |

**Tools:**

Matlab 2016a;

libsvm3.2;

Neural Network tools 0.8;

**Matlab code files:**

‘importdata.m’;

‘importdata2.m’;

‘Partition.m’;

‘redistribution.m’;

‘evaluation.m’

‘observefeatures.m’

‘normalize.m’

‘preprocessing.m’

‘preprocessing2.m’

‘classifier\_Bayes.m’

‘classifier\_KNN.m’

‘classifier\_MAQ.m’

‘classifier\_MDC.m’

‘classifier\_MPL.m’

‘classifier\_SVM.m’

**Reference:**

[1] https://www.r-bloggers.com/classification-on-the-german-credit-database/

[2] https://onlinecourses.science.psu.edu/stat857/node/215

[3] Sccot A. Zonneveldt, Kevin B. Korb and Ann E.Nicholson, Bayesian network classifiers for German credit data.

[4] Asuncion," A.," &" D." Newman" (2007)." UCI" machine" learning" repository." http://www.ics.uci.edu/~mlearn/MLRepository.html