ACM Word Template for SIG Site

Bulat Gabbasov

[bgabbaso@andrew.cmu.edu](mailto:bgabbaso@andrew.cmu.edu)

Ivana Stevens

[istevens@andrew.cmu.edu](mailto:istevens@andrew.cmu.edu)

**ABSTRACT**

In this paper, we analyzed Kaggle data from the Otto Group product classification challenge. We compared different methodologies like Naïve Bayes Classifiers, Decision Trees, Random Forest and Principle Component Analysis to classify data.

**General Terms**

Algorithms, Measurement, Theory.

**Keywords**

Naïve Bayes classifier, PCA.

# INTRODUCTION

For this competition we were provided data that corresponded to products sold on the Otto Group website. Features were obfuscated but pertained to characteristics of each product. The goal of thee challenge was to create a classifier to separate the data into 9 classes. We addressed the problem by comparing four methodologies. First we attempt to use Principle component Analysis, but the yielded poor results. We were able to best classify the data using Random Forest Approach. And finally, we improved our results by applying calibration to Random Forest algorithm.

The data has 93 integer features, which are occurrences of events, and each instance belongs to the one out of 9 classes. The training set contains 61878 samples.

The goal is to predict probabilities of an instance belonging to the classes. According to the competition, the quality of estimation is evaluated based on logloss function:

where is the number of samples, if and only if sample is in class .

However, in addition to logloss we evaluated our algorithms in terms of accuracy, which is the ratio of correctly classified samples to the total number of samples.

We used cross-validation technique to evaluate our methods before submitting them to Kaggle. Thus, we splitted the training set into 10 folds and we run our algorithm 10 times and then we took average accuracy and logloss values.

As a baseline, we decided to use a simple algorithm which assigns probabilities to all classes. In that case .

In our work we used MATLAB, Python and skicit-learn library, Weka.

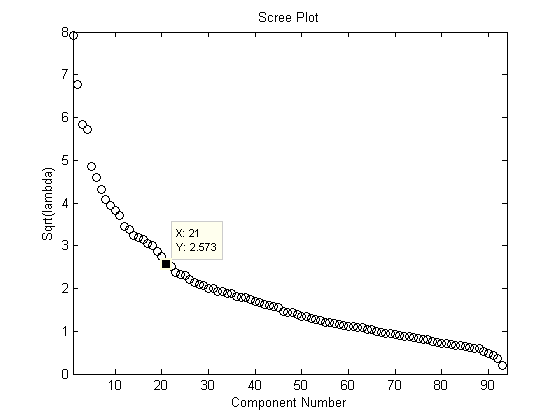
# Principle Component Analysis

Given the high dimensionality of our data (93 factors) we attempted to analyze our data using a reduced number of features.

We considered several different stopping methods to choose the number of factors: the Kaiser-Guttman rule, and Cattell’s “elbow” method. From there we determined the likelihood that a sample belonged to a particular class by calculating the percentage of its ten nearest neighbors also belonged to a given class.

Principle component analysis works by projecting data in the feature space to a lower dimensional space, maintaining only the features, which within that space account for the most variance. The reason choosing the number of factors is important is that you run the risk of losing information with too many factors, or incorporating too much noise in your system, if you use too many.

In 1966, Cattell proposed using a scree plot to find the optimal number of data points. This method requires finding the “elbow” or the greatest difference in variation between values. Essentially we are looking for where the variation describes from the eigenvalues (𝜆k) begins to level-off. This methodology has been critiqued for its subjectivity. We identified the elbow from the scree plot to be at about 21 data points.



The next method we tested was the Kaiser-Guttman rule. Under this rule we assume that all of the factors are independent. Therefore their eigenvalues should equal 1. This means we limit the number of factors to only those that are greater than or equal to 1. Under this assumption we reduced the number of components to 65 from 93. These 63 factors accounted for 97% of the variance.

We also considered using the Karlis-Saporta-Spinaki Rule. This rule relates the number of reduced features to both the original number of features (p) and the number of samples in the data set (n).

However this rule provided a value similar to the Kaisser-Guttman rule (63), so it was left out of this analysis.

In general each PCA stopping method provided little change to the accuracy of the data classifier as we found that the k-nearest neighbors did not change.

# Naïve Bayes classifier

Multinomial Naïve Bayes classifier was a starting point in the project. The classification is performed according to the following model:

This method was chosen because the features are occurrences of events and the classes are categories.

However, the model makes very strong assumptions on the data. The model assumes that occurrences of events are independent from each other and the features are also independent from each other. Since there is no additional information provided by the Otto group about the data we were not able to make any decisions if these assumptions are correct or not based on the information about the data.

To estimate parameters we used maximum likelihood method with Laplace smoothing, which could be expressed by the following formula:

where is number of times of occurrences of event *i* in class *c* and is total number of events occurrences in class *c*. is the smoothing parameter.

We tuned the method by adjusting smoothing parameter. We found that the method gives the best results with smoothing parameter . The table 1 shows results obtained by using Multinomial Naïve Bayes classifier.

Table . Multinomial Naive Bayes

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Multinomial Naïve Bayes ( | 0. 632860 | 3. 799616 |
| Multinomial Naïve Bayes ( | 0.673406 | 1.219739 |

The results show that while Multinomial Naïve Bayes performed relatively well in terms of accuracy, the logloss is not very good.

# Modified Naïve Bayes Classifier

In order to improve accuracy of Naïve Bayes Classifier we decided to tune it using the techniques described in [1]. However, the techniques did not give us significant improvement. Therefore, we modified Naïve Bayes classifier by generalization of it. According to [1] a Naïve Bayes classifier can be expressed by the following formula:

We modified this formula to estimate probabilities of a data point belonging to a class:

Furthermore, if we assume that the first component of each is 1, then the formula can be simplified to:

As it can be seen from the formula sum of probabilities by all classes is always 1.

In order to estimate parameters and , we minimized logloss function by solving following optimization problem:

where is 1 iff equals .

To solve the problem we used gradient descent method. The gradient of the expression is:

The optimality of the solution can be proved by showing that the function is a convex function. The function is a convex function because the second derivative is positive:

And . Therefore Jacobian of the function can be expressed as:

The method improved logloss to 0.64861 and accuracy improved up to 0.7616.

Table 2. Modified Multinomial Naive Bayes

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Modified Multinomial Naïve Bayes | 0.7616 | 0.64861 |

We tried to improve the results by preprocessing the data. First, we noticed that any linear transformation of the data does not improve the results, because it is mathematically the same to the original optimization problem. Suppose that is a transformation of the data. It can be easily seen that by applying the same transformation to the parameter vector the optimization problem remain the same:

So it is meaningful to perform only non-linear transformations. The first transformation that we tried was using relative number of occurrences instead of absolute:

The transformation made the results worse:

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Modified Multinomial Naïve Bayes with relative occurrences | 0.706029 | 0.906848 |

The second thing that we tried was performing kernel trick. However, with amount of data that we had and given our computational power (we used only our laptops), it was not possible to do it. Instead, we decided to directly add to features interconnections between them of size 2. So new feature space was:

The new number of features is , which our laptops were able to process. However, it didn’t give any significant improvement in results.

# Decision Trees

Decision tree learning is another technique for classification problems. After not much luck with Naïve Bayes Classifier we decided to try some other techniques and we started from decision trees.

Decision tree learning tries to approximate a function by constructing a decision tree. In each step a feature that maximizes information gain, i.e. difference between entropies when the selected feature is know and when the selected feature is not known [2]:

The main challenge in decision tree was selecting appropriate parameters to avoid overfitting. We used a greedy algorithm to construct decision trees and squared maximal number of features.

The best result obtained by use of decision trees was worse than in modified Naïve Bayes approach.

Table 3. Decision Tree

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Decision Tree | 0.714868 | 9.848110 |

# Random Forest

Random forest belongs to the ensemble methods family. The idea is to combine several estimators by taking average. This reduces variance of estimators and in general can provide better results.

The main 2 parameters of the random forest classifier are number of trees and number of features based on which each tree will be constructed.

Accuracy of the random forest classifier depends on correlation between trees and accuracy of each individual tree. Increasing accuracy of individual trees increases overall accuracy. Increasing correlation between the trees decreases accuracy.

Increasing number of features leads to increasing of accuracy of individual trees and increasing correlation between trees. So the optimal value is somewhere in between.

Increasing number of trees increases accuracy, however, starting from some value new trees are correlated to each other and do not give significant improvement in accuracy.

We used random forest algorithm with number of trees equals to 600 and number of features equals to 20. The result showed significant improvement in both accuracy and logloss.

Table . Random Forest

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Random Forest | 0. 810558 | 0. 541102 |

# Calibration

Calibration may improve accuracy in predicting probabilities of classifiers. According to [3] and [4] accuracy in predicting probabilities for methods like Naïve Bayes Classifier, Decision Tree and Random Forest is poor, because they were not intended to estimate probabilities. Therefore instead of optimizing logloss function they optimize 0-1 loss function (number of correctly predicted instances). During calibration output estimates of classifiers are calibrated using logloss function, which may significantly improve quality of predicted probabilities.

There two types of calibration: the one that is based on sigmoid function and the one that is based on isotonic functions.

According to [2] in case of 2 classes sigmoid calibration is performed according to the following formula:

where is the output of base classifier, and *A* and *B* are parameters. Then the function is optimized to minimize logloss function:

The sigmoid calibration can be extended to multiclass classifier using the same technique as we used in section 4 for Modified Multinomial Naïve Bayes Classifier.

According to [2] isotonic calibration assumes that true target probabilities can be expressed as:

where is an isotonic (monotonically increasing) function. After than the problem can be expressed as finding an optimal isotonic function that decreases squared error:

The optimization problem is than solved using pair-adjacent violators algorithm.

We achieved the best results using isotonic calibration with Random Forest classifier described in the previous section.

Table . Random Forest

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| Naïve Bayes Calibrated | 0.706464 | 0.831676 |
| Decision Tree Calibrated | 0.769151 | 0.837796 |
| Random Forest Calibrated | 0.820704 | 0.47751 |

# Summary

We tried to estimate probabilities using several techniques. The table 6 summarizes the results obtained by algorithms. As it can be seen the best results was obtained using Calibrated Random Forest Algorithm (Accuracy = 0.820704 and Logloss = 0.47751). In general, improving logloss led to improving in accuracy, which we think is surprising, because original methods were developed to optimize accuracy.

Table . Summary

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Logloss |
| PCA and Kaiser-Guttman stopping rule. (K = 65) |  | 5.24029 |
| PCA and Scree Plot (K = 21) |  | 5.24029 |
| Multinomial Naïve Bayes ( | 0. 632860 | 3. 799616 |
| Multinomial Naïve Bayes ( | 0.673406 | 1.219739 |
| Modified Multinomial Naïve Bayes | 0.7616 | 0.64861 |
| Modified Multinomial Naïve Bayes with relative occurrences | 0.706029 | 0.906848 |
| Decision Tree | 0.714868 | 9.848110 |
| Random Forest | 0. 810558 | 0. 541102 |
| Naïve Bayes Calibrated | 0.706464 | 0.831676 |
| Decision Tree Calibrated | 0.769151 | 0.837796 |
| Random Forest Calibrated | 0.820704 | 0.47751 |

# ACKNOWLEDGMENTS

Our thanks to Alex Smola and TAs for allowing us to take incomplete grade and submit the project later.

# REFERENCES

1. Jason D. M. Rennie and Lawrence Shih and Jaime Teevan and David R. Karger, 2003. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. *In Proceedings of the Twentieth International Conference on Machine Learning*, 616-623
2. Machine Learning, [Tom Mitchell](http://www.cs.cmu.edu/%7Etom), McGraw Hill, 1997
3. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers, B. Zadrozny & C. Elkan, ICML 2001
4. Predicting Good Probabilities With Supervised Learning, Alexandru Niculescu-Mizil, Rich Caruana