CS446 Class Project: Naive Baye Classifier for Handwritten Digit

Weijie Huo, NetID: whuo3

May 7, 2015

Abstract

The goal is to implement a Naive Byes classifier using Java for handwritten digits. I explored different features to build the classifier, which help improving the accuracy of digit classification step by step

1 Introduction

*Millions of historical documents are written by hand. As computer has become the main force of storing human knowledge. It would be of great convenient to human if we can use machine learning skills to help us classify the hand written documents and inputed it into computer automatically instead of typing characters by characters by hand. So far, I have learned several machine learning skills in the class. I want to use Java to implement a Naive Bayes classifier to help classify characters. For simplicity, the task will focus on classifying handwritten digit. Besides, by implementing the Naive Bayes classifier, I want to find what features are important to classify hand written characters and compare different features. In this experiment, we are assuming that each character has been separated from other characters and is located at a 28*28 cells. Besides, we ignore the importance of the relationship in the sequent of digits. To consider this case, we can use HMM algorithm to solve the problem. To sum up, I want to implement the classifier in two ways. One is to use single pixel as features. The other one is to group multiple pixels as a feature to train the model. I expect grouping pixels as feature to perform better than single pixel as feature. Besides, I will use comparisions of different models in terms of the accuracy to evaluate it. Last, I will introduce a unique feature to help improve the accuracy of classifying digit 8 for about 10 percent.*

2 Background

There are many new technologies used in classifying handwritten digit nowadays. One of the most well known technique is Convolutional Neural Network. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations. In fact convolutional neural network has been proven to be a better method than Naive Bayes to build model to classify hand written. So, why Naive Bayes rather than Convolutional Net? Because I want to keep it simple and write my own code for digit recognition. Besides, the whole theory behind Naive Bayes classifier is very fascinating. I can assume that the particular feature is unrelated to any other feature, given class variable, which never be true. However, we can still get satisfying solution for classification.

3 Task and Data

3.1 The Task

*I will perform maximum a posteriori(MAP) classification of test digits according to the learned Naive Bayes model. Assume a test image has feature values $f_{1,1}$, $f_{1,2}$, $f_{1,3}$ etc. According to this model, the posterior probability is P(class)*P($f_{1,1}$ | Class)*P($f_{1,2}$ | Class)*P($f_{1,3}$ | Class) ... To avoid underflow, I use the log to the equation. Therefore, log P(Class) + logP($f_{1,1}$ | Class) + logP($f_{1,2}$ | Class) + logP($f_{1,3}$ | Class). After use them for MAP classification. I will use maximum likelihood(ML) classification where I omit the prior term from the above, and compare the difference. Besides, I will compare the difference of picking different features. For his experiment, I will compare single pixel of digit as feature and group of pixels as feature. For evaluation, I will use different features to obtain classification rate for each digit. Then, by computing the Odds ratios to obtain pairs of digits that have the highest confusion rates and come up with new feature that help to solve the problem of it.*

3.2 The Data

*I have a file that contains 6000 digit images. Each digit is of size 28*28. The file is in ASCII format, with three possible characters. white space means a white pixel, others mean a black pixel. Since there is no label given, I need to classify each digit image manually by hand. After that, I randomly separate 5000 samples to train and 1000 to test. Besides, digit images from 0 to 9 are evenly distributed in the dataset.*

4 The Models

4.1 Baseline Models

The most trivial approach is to regard each single digit as feature to build the probabilistic model for Naive Bayes classifier. However, this would ignore the dependency between the target pixel and the neighbor pixels.

4.2 Existing Models

LeNet-5 is the latest convolutional network designed for hand written and machine-printed character recognition. It was developed by Yann LeCun, director of AI Research in Facebook. Nowadays, the method are embedded in several high speed scanners used by banks to read checks. Convolutional Neural Networks are a special kind of multi-layer neural network. It is trained with a version of the back propagation algorithm. It is designed to recognize visual patterns directly from pixel images with minimal pre-processing. Especially, it can recognize patterns with extreme variability and with robustness to distortions and simple geometric transformations.

4.3 Proposed Model(s)

*I will perform maximum a posteriori(MAP) classification of test digits according to the learned Naive Bayes model. First, I will build the model using single pixel as features. Therefore, there are 28*28 features in the model. Secondly, instead of the building each feature corresponding to a single pixel, the second model will be builded by grouping adjacent pixel as features. Especially, the second model has two versions, which are disjoint and overlapping. Besides, I will find out how many pixels grouped as a feature give the better performance. According to the corresponding model, the posterior probability is P(class)*P($f_{1,1}$ | Class)*P($f_{1,2}$ | Class)*P($f_{1,3}$ | Class) ... To avoid underflow, I use the log to the equation. Therefore, log P(Class) + logP($f_{1,1}$ | Class) + logP($f_{1,2}$ | Class) + logP($f_{1,3}$ | Class). To avoid no zero count, I use Laplace*

smoothing. Last, I will use maximum likelihood(ML) classification where I omit the prior term from the above, and compare the difference.

5 Experiments

5.1 Experimental Hypotheses

1.The hypotheses is designed to classify what is the digit given the digit image.
2.Grouping pixels as feature can perform better than just picking single pixel as feature.
3. Assuming each features conditionally independent can still get a good accuracy for solving this problem.
4.Adding feature that check whether the current pixel has been bounded in the horizontal line can help better classify digit 8.

5.2 Experimental setup

*There are 6000 digits image samples. Digits 1 to 9 are evenly distributed in the dataset. I first categorize each digit image by the digit label. Then shuffle each category, and randomly pick 100 samples from each category. Therefore, there are 1000 samples that forms the test dataset, and 5000 samples that forms the train dataset. As mentioned above, I first implemented the classifier using single pixel as feature. Then, to obtain a better performance, I implemented a classifier that use grouped pixels as feature. There are two version of it, one is disjoint grouping and the other is overlapping grouping. Finally, I add a feature that check whether the current pixel has been bounded in the horizontal line in disjoint grouping to help improve accuracy of classifying 8. For evaluation, I first find the classifier's accuracy of classifying each digit. And then I build the confusion table to help us better analyze our model. The confusion matrix is a 10*10 matrix whose entry in row and column is the percentage of test images from row class that are classified as column class. After that, I recode four pairs of the digits that have the highest confusion rate according to the confusion matrix and inspect the impact of each feature on the confusion digits. The odd ratio is defined as* $odds(F_{ij} = 1, class1, class2) = P(F_{ij} = 1 \mid class1 / P(F_{ij} = 1 \mid class2)$. *I will build the confusion rate and odd ratio according to the model that use single pixel as feature. Then, I use accuracy of models that group pixels as feature to show the overall improvement. Note that, I use Java to implement necessary methods for evaluation. The output of the evaluation is performed on terminal console.*

5.3 Experimental results

I first implemented a naive classifier using single pixel as feature, which gives the average performance of 77.1% accuracy over all test data. Especially, the classifier performs relatively good in classifying digit 1 reaching 96% accuracy, while only 60% accuracy for digit 8. Then I build the confusion table:

Confusion Table:

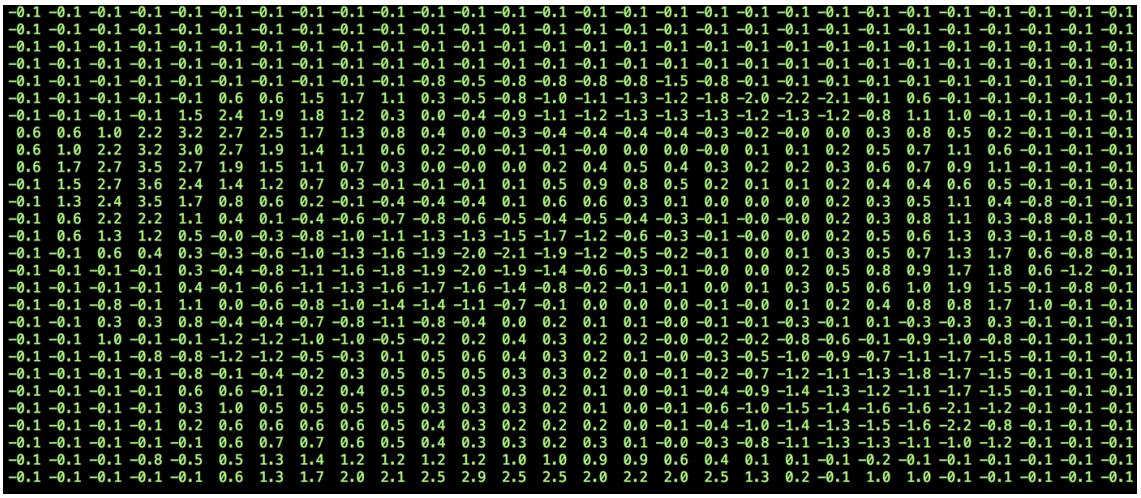
	0	1	2	3	4	5	6	7	8	9
0	0.844	0.000	0.011	0.000	0.011	0.056	0.033	0.000	0.044	0.000
1	0.000	0.963	0.009	0.000	0.000	0.019	0.009	0.000	0.000	0.000
2	0.010	0.029	0.777	0.039	0.010	0.000	0.058	0.010	0.049	0.019
3	0.000	0.020	0.000	0.790	0.000	0.030	0.020	0.060	0.020	0.060
4	0.000	0.009	0.000	0.000	0.766	0.000	0.028	0.009	0.019	0.168
5	0.022	0.022	0.011	0.130	0.033	0.674	0.011	0.011	0.022	0.065
6	0.011	0.066	0.044	0.000	0.044	0.055	0.758	0.000	0.022	0.000
7	0.000	0.057	0.028	0.000	0.028	0.000	0.000	0.726	0.028	0.132
8	0.019	0.010	0.029	0.136	0.019	0.058	0.000	0.010	0.602	0.117
9	0.010	0.010	0.010	0.030	0.090	0.020	0.000	0.020	0.010	0.800

As we can see from the confusion table, the highest confusion rate are: (5, 8), (7,9), (3,5), (4,8). The odds ratio is listed as below:

(5,8):

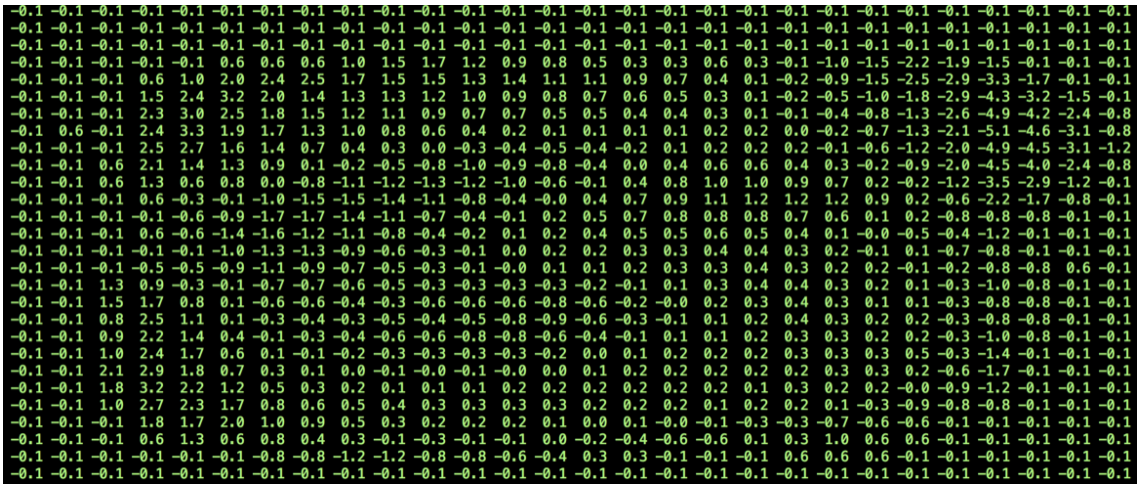
According to the table, the relatively most distinctive features that differ 5 and 8 is in the left center.

(7,9):



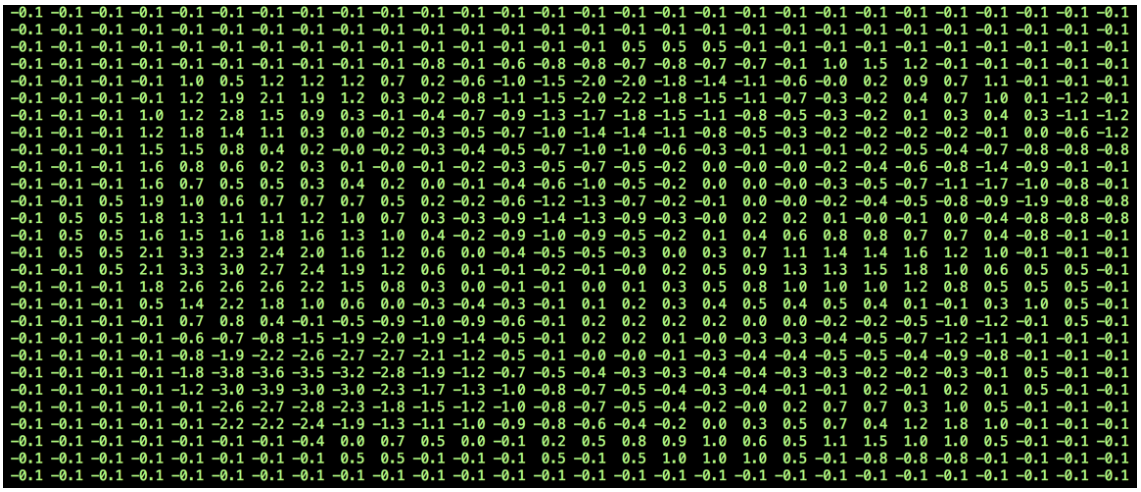
According to the table, the relatively most distinctive features that differ 7 and 9 is in the left.

(3,5):



According to the table, the relatively most distinctive features that differ 3 and 5 is in the upper left.

(4,8):



According to the table, the relatively most distinctive features that differ 4 and 8 is in the lower center.

*To improve the accuracy of the classifier, I group pixels as feature. To differentiate the impact of accuracy in terms of choosing number of pixels to group, I test accuracies for disjoint patches of size 2*2, 2*4, 4*2, 4*4 and accuracies for overlapping patches of size 2*2, 2*4, 4*2, 4*4, 2*3, 3*2, 3*3. As a result, disjoint give the accuracy ranging from 82%(In the case of size 2*2) to 86.1%(In the case of size 2*4), and OverLap give the accuracy give the accuracy ranging from 79.1%(In the case of size 2*2) to 84.3%(In the case of size 4*2). However, both models perform poorly when classifying 5 and 8, which reach only 36% to 60%. Therefore, in order to get a better accuracy of this two digits, I implemented a new classifier considering new feature that check whether the current pixel has been bounded in the horizontal line on Overlap model. The result is satisfying. Compared to the old overlap model using different grouping size, the overall performance get improved around 1% and the accuracy that classifying 8 get improved ranging from 4% to 9%.*

6 Conclusion

*Single pixel as a feature has lower accuracy than Overlap and Disjoint. Because single pixel fail to consider its dependency on the surrounding pixels, while overlap and disjoint does. As the result shown, the overall accuracy of selecting disjoint as features is slightly better than overlap. I believe the reason for that is overlap is a little bit overfitting(a little change of test case from train data will result in different result). Therefore the accuracy is worse. On the other hand, among disjoint, they are relatively the same. However, when n = 4, m = 2, the accuracy of the test data is the highest(86.1%). I think it is the case when our classifier?s variance is not too high and classifier?s bias is not too low, where it reach a relatively equilibrium point. In terms of running time of training, since the complexity of single pixel as feature is O(m*n), therefore it is the fastest. On the other hand, the complexity of disjoint as feature is O((2$^{(m*n)}$) * (28*28 / (m*n))) while the complexity of overlap is O((2$^{(m*n)}$) * (m*n)). Obviously, overlap take more time than disjoint in training. The most exciting things is that by creating a new feature that check whether the current pixel has been bounded in the horizontal line on Overlap model can improve the accuracy of classifying digit 8. For the experiment, all the training data stands vertical. The above classifiers will fail to recognize patterns with extreme variability. For future work, we need to consider the patterns with extreme variability. Also, we need to consider characters that appear in sequence, which will involve combining HMM model with the single digit classifier.*

Bibliography

Mladenic, D., & Grobelnik, M. (1999, June). Feature selection for unbalanced class distribution and naive bayes. In ICML (Vol. 99, pp. 258-267).

Sahiner, B., Chan, H. P., Petrick, N., Wei, D., Helvie, M. A., Adler, D. D., & Goodsitt, M. M. (1996). Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. Medical Imaging, IEEE Transactions on, 15(5), 598-610.

Sermanet, P., Chintala, S., & LeCun, Y. (2012, November). Convolutional neural networks applied to house numbers digit classification. In Pattern Recognition (ICPR), 2012 21st International Conference on (pp. 3288-3291). IEEE.

Zhang, M. L., Pea, J. M., & Robles, V. (2009). Feature selection for multi-label naive Bayes classification. Information Sciences, 179(19), 3218-3229.