Machine Learning
University of Mannheim
Winter 2019
Prof. Dr. Rainer Gemulla

Assignment 1: Naive Bayes

Sudhanva Rao

MNIST Dataset
We will use a preprocessed variant of the MNIST digits dataset in this assignment. The task is to classify hand-written digits. There is one class for each digit (i.e., classes 0,1,2,...,9). The features represent a scanned image (2828 pixels, values in 0,1,...,255). The dataset contains both training data ($\approx$ 6000 images per class) and test data ($\approx$ 1000 images per class)

1. **Training**

   Provide a function nb train that trains a Naive Bayes classifier for categorical data using a symmetric Dirichlet prior and MAP parameter estimates. A description of the parameters and expected result can be found in the Python file. For example, you may assume that all features take values in 0,1,...,K$-$1, where K is the number of possible values.

   The dataset contains 10 classes (C), 784 features(D) which can take up values between 0 to 255.

   The logarithm of the prior probability of the class is evaluated by evaluating the relative frequency of each class. If N_c is the number of instances of class C and N is the total number of samples, then the logarithm of prior probability, logprior is evaluated by

   $$logprior = \log(\frac{N_c + \alpha - 1}{N + (\alpha - 1)C})$$

   The class conditional log-likelihood of value v in feature j given a class c is similarly calculated by taking the logarithm of the relative frequency of a feature value v of feature j in class c.

2. **Prediction**

   Provide a function nb predict that takes your model and a set of examples, and outputs the most likely label for each example as well as the log probability of that label.

   The unnormalized log joint probability for each class given an input X_i is evaluating by adding the log probability of each feature taking a feature value given by the new input. Thus obtained log probability is then added with the prior class probability to get the joint probability of each class given an input.

   The joint probability is then normalized to obtain the most probable class and its corresponding probability(again log-ed).

3. **Experiments on MNIST digits data**

   (a) Train your model with $\alpha = 2$ on the MNIST training dataset, then predict the labels of the MNIST test data using your model. What is accuracy (= 1 − misclassification rate) of your model?

   The accuracy achieved = 0.8363

	precision	**recall**	**f1-score**
0	0.91	0.89	0.90
1	0.86	0.97	0.91
2	0.89	0.79	0.84
3	0.77	0.83	0.80
4	0.82	0.82	0.82
5	0.78	0.67	0.72
6	0.88	0.89	0.89
7	0.91	0.84	0.87
8	0.79	0.78	0.79
9	0.75	0.85	0.80
micro avg	0.84	0.84	0.84
macro avg	0.84	0.83	0.83

Table 1: MNIST: Classification report

   (b) Plot some test digits for each predicted class label. Can you spot errors? Then plot some misclassified test digits for each predicted class label (code provided). Finally, compute the confusion matrix. Discuss the errors the model makes.

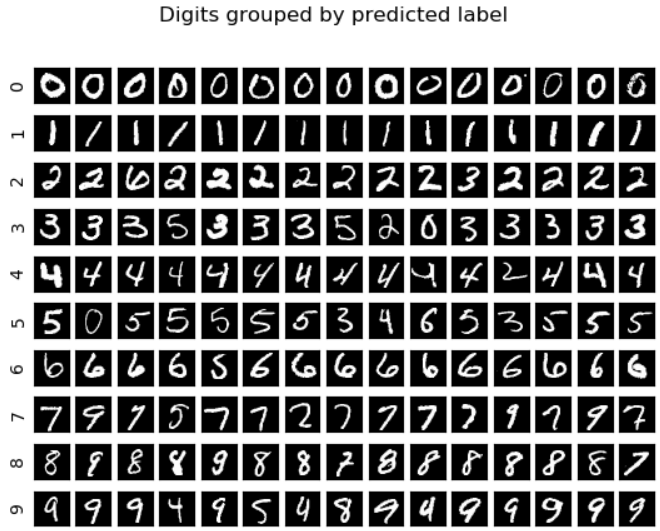

Figure 1: Test digits for each predicted class label

As shown in the figure 1, we can see that there are errors in some of the classes, especially class 5,3 and 9.

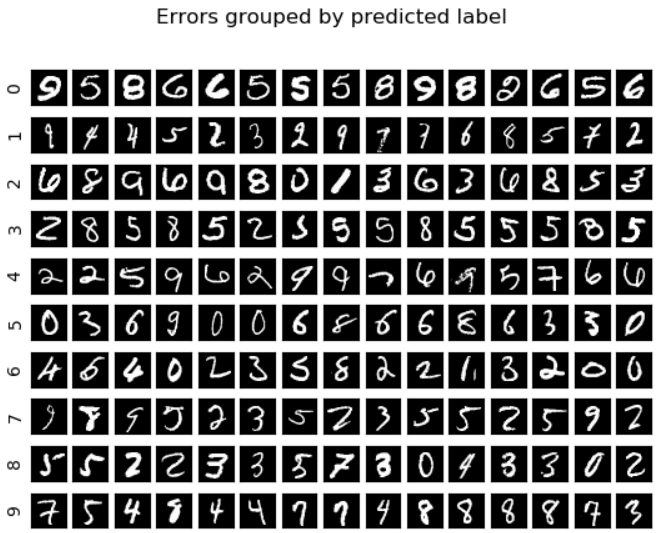

Figure 2: Test digits for each misclassified class label

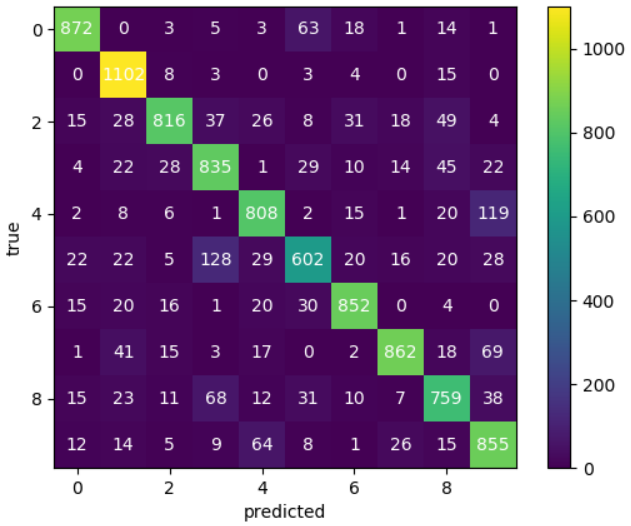

Figure 3: MNIST: Confusion matrix

From the figure 2 and 3 we can infer following:

- The model has difficulty classifying numbers with curves as can be seen by the accuracy of class 3,5,8 and 9.
- From the confusion matrix it can be seen that the false positives are high for the class which looks similar.

4. **Model selection**

   Use cross-validation to find a suitable value of the hyperparameter α (of the symmetric Dirichlet prior). Also plot the accuracy (as estimated via crossvalidation) as a function of α. Discuss.

   Most suitable alpha = 1.1

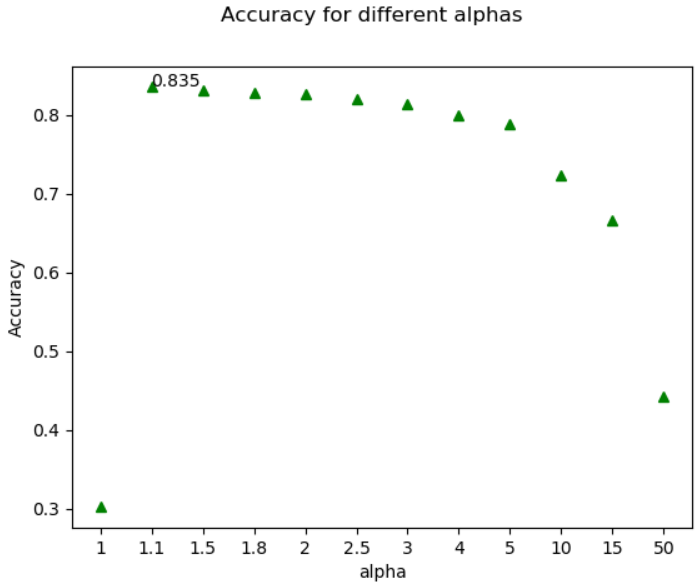

Figure 4: Accuracy for different alphas

From the figure 4, we can see that the accuracy drops with increase in alpha. However without smoothening, the accuracy is very low. This can be attributed to the fact that with increase in alpha, we solve zero frequency problem. But the distribution across the values becomes flatter and this can affect the accuracy.

5. **Generating data**

   (a) Implement a function nb generate that generates digits for a given class label. The feature values for each feature of the class is randomly picked from the distribution of the feature values. The higher the probability of a feature value, more likely is it to be picked.

   (b) Generate some digits of each class for your trained model and plot. Interpret the result. Repeat data generation for different models by varying the α. How does α influence the results? Discuss.

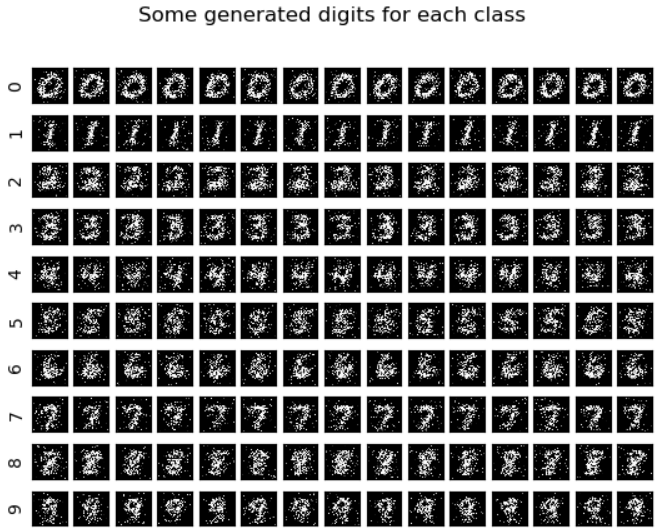

Figure 5: Generated data given class (alpha=2)

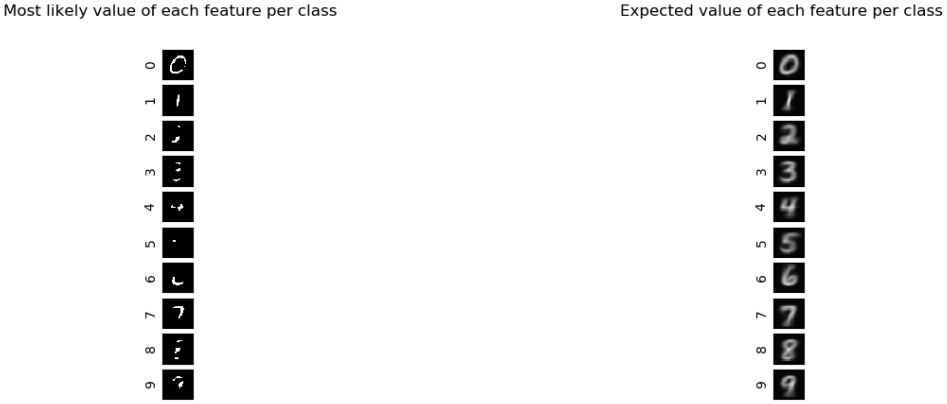

Figure 6: Dominant feature of each class

From the figure 5 and 6, we can see that the class pictures reflect the accuracy results. This can be seen especially in the case of class 5.

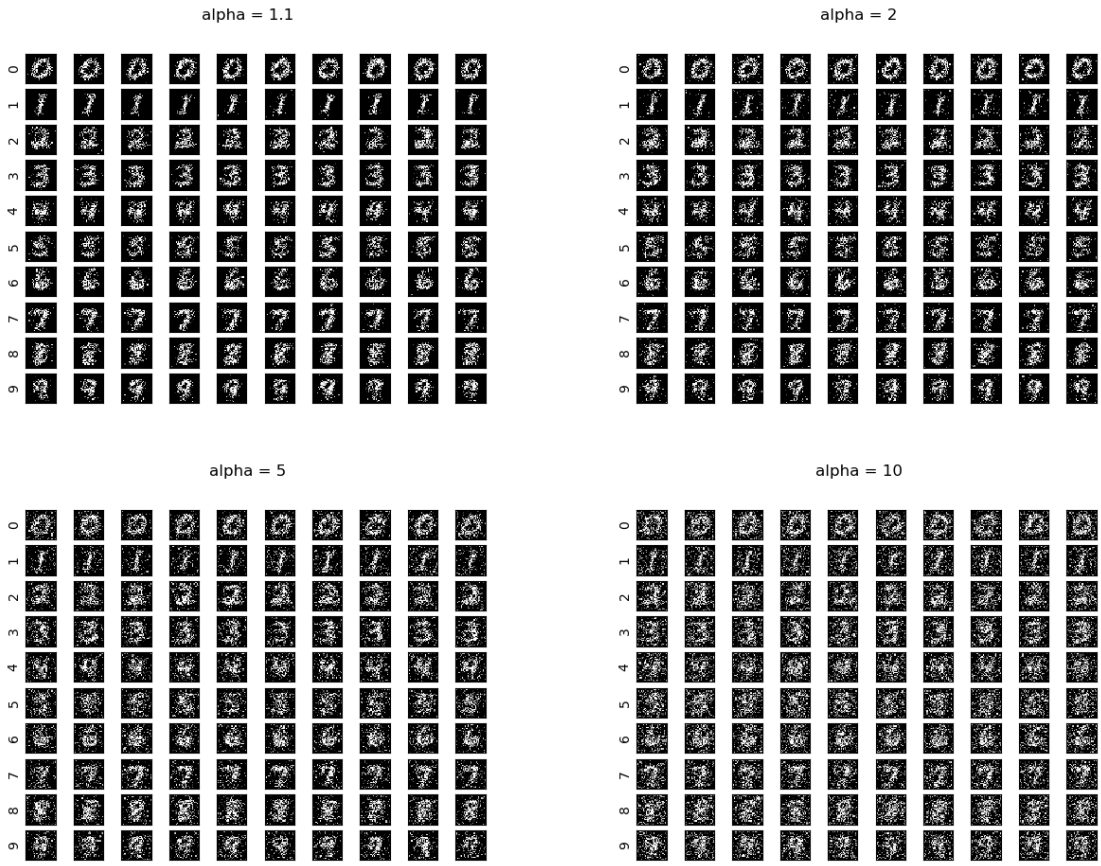

Figure 7: Generated data for all classes for different alphas

From the figure 7, we can see that as alpha increases, the "resolution" of the class pictures resuces. This can be attributed to the fact that as we increase smoothening(alpha), the distribution becomes flatter. This can be seen with multiple white dots spread in the images when alpha is increased.