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.

To compute class priors, it uses same hyperparameter alpha for all the classes from 0 to 9. And it uses MAP parameter estimates, the priors for class c is

$$\bar{\pi}_c - \frac{(n_c+\alpha_c-1)}{(n+\ \alpha_0-C)}\ \text{, where }\ \alpha_0 \text{ - } \textstyle\sum_{c=1}^{C}\alpha_c$$

To compute class-conditional density, in each class C, D means features, and K means values of features.

$$\widehat{\theta_{cjk}} - \frac{n_{cjk} + \alpha_{cjk} - 1}{n_c - \sum_k(\alpha_{cjk} - 1)}$$

The return value of the function is used by logarithm, then it returns to the log prior and the class-conditional log likelihood. By using Dirichlet prior, it can address model overfitting problem and zero-count problem.

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 (confidence) of that label. A description of the parameters and expected result can be found in the Python file.

Using naïve bayes assumption is

$$p(x|\mathrm{y}) = \Pi_j^D p(x_j|y)$$

To compute unnormalized log joint probability for class c,

$$P(y-c,x) - P(y-c)\prod_j^D P(x_j|y-c) - \bar{\pi}_c\prod_j^D[\hat{\theta}_{cj]}]_x$$

3 Experiments on MNIST digits data

a) Train your model with α = 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 of model results in 0.8361.

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

(https://en.wikipedia.org/wiki/Confusion_matrix, code provided). Discuss the errors the model makes.

The confusion matrix is to evaluate the accuracy of a classification. In the plot which is made by model, it shows how many get true-predicted values in class labels. The lighter the color, the more correctly the predicted value for the class label appears. However, it is not known which handwriting numbers in the MNIST data are well distinguished by the model because simple count values without normalization come out. Because the class distribution is imbalanced and the number of tested inputs is different, creating a confusion matrix after the normalization process will be a more convincing indicator of model evaluation.

5 Generating data

a) Implement a function nb generate that generates digits for a given class label. A description of the parameters and expected result can be found in the Python file.

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

The hyperparameter is predefined values for good prediction and have a great impact on performance. From repetition of data generation with varying hyperparameter,

If it keeps increasing the alpha value, then naive bayes model will bias towards the class which has more records, and it could encounter underfitting. so, by choosing small alpha value is good idea.

And as the value of the hyperparameter alpha changes, the accuracy is affected. For example, when the alpha value is 1, the values are 0.4083, when the alpha value is 3, 0.8288, when the alpha value is 4, 0.819, and when the alpha value is 5, the values are 0.8104. The lower the accuracy, the smaller the confidence is.

Reference

Murphy, Ch.9.3, Naïve Bayes Classifiers

Improving Multi-class Text Classification with Naive Bayes, AI Technical Report 2001-004, Jason D.M. Rennie (https://dspace.mit.edu/bitstream/handle/1721.1/7074/AITR-2001-004.pdf?sequence=2&isAllowed=y)