Gaussian Naïve Bayes vs. Logistic Regression for Image Classification of Handwritten Digits

Jaehyuk Choi

CSE575 Statistical Machine Learning
Arizona State University

Abstract

Image classification techniques has been widely applied at a rapid growth rate. As a simple task of image classification, the data of image 7 and image 8 from the MNIST database were extracted and manipulated for training and testing with two techniques of Gaussian Naïve Bayes and Logistic Regression. The algorithms used for classifying the given test data and the accuracies resulted in these two techniques were compared and discussed.

I. INTRODUCTION

Gaussian Naïve Bayes belongs to generative classifier whereas Logistic Regression belongs to discriminative classifier. A generative classifier learns a model based on joint probability p(X, Y), where X is input data and Y is class label, to predict a class by calculating p(Y|X) from Bayes theorem. However, a discriminative classifier learns p(Y|X) directly from a model.

The data of the MNIST database represents images of a handwritten digits from 0 to 9. Each image, as grayscale, has 28 x 28 pixel. The number of the given training data samples are 12116 (6265 samples for image 7 and 5851 samples for image 8) and the number of the given testing data samples are 2002 (1028 samples for image 7 and 974 samples for image 8). The following figure shows an image 7 and an image 8; to classifying unknown data (testing data) after learning a model by training from training data is our goal.

Figure 1. Image 7 and Image 8 as examples of handwritten digit from the MNIST database.

To come back to the physical meaning of probability p(X, Y), X represents a random variable of 784 features (28x28 = 784) and Y represents a class label as label 0 for image 7 and label 1 for image 8. To simplify the problem, the dimension of the data was reduced by calculating mean and standard deviation of pixel values for every samples; training dataset (12116 x 784 => 12116 x 2) and testing dataset (2002 x 784 => 2002 x 2). The following figure shows the data distributed based on mean and standard deviation of all pixels (features).

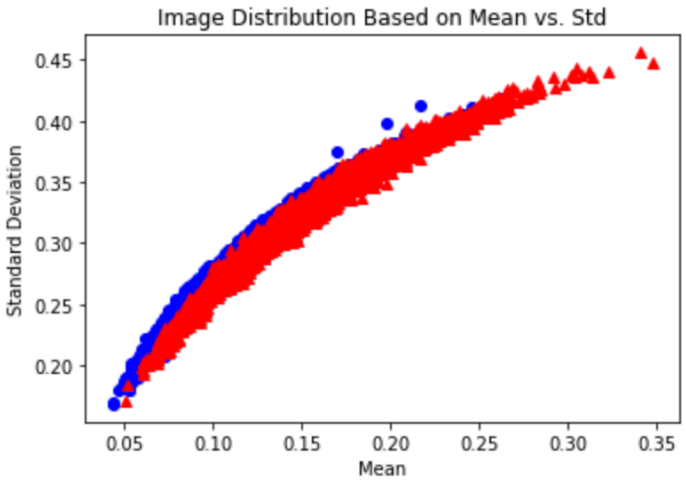

Figure 2. Blue dots are for image 7 and red triangles are for image 8 in 2-dimensional space (mean and standard deviation) from the MNIST database.

II. COMPARISON

Since both classifiers are linear, these techniques may not be accurate for the data that are not clearly and linearly separable. Thus, we can expect the accuracy would be low. To compare these two techniques, the algorithms used for handwritten digit recognition and their results will be compared. The algorithms of Gaussian Naïve Bayes and Logistic Regression were

1) Algorithm Analysis of Gaussian Naïve Bayes

Since the given images are grayscale (meaning that the feature of the data is continuous), Gaussian Naïve Bayes was used. That is, we assume that the prior distribution over the data is gaussian.

In the Gaussian Naïve Bayes model, we have two assumptions: individual features are independent, and the images are distributed as a Multivariate Gaussian Distribution. From the first assumption, we can say that each individual probability distribution P_j over X = (x_1 as mean, x_2 as standard deviation) where j indicates a class. From the second assumption, the correlation between features can be found.

Under the second assumption, the equation for the Multivariate Gaussian Distribution was used.

$$P_j(X) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(X-\mu)^{\mathrm{T}}\Sigma^{-1}(X-\mu)\right]$$

Where d is the number of dimension (feature), Σ is a d x d covariance matrix, X is a vector of the features from testing dataset, and μ is a mean vector of the features from training dataset.

Since X is a random variable with Gaussian distribution $N(\mu, \Sigma)$, the expected value of X must be equal to μ.

Now consider two independent Univariate Gaussian random variables as $X_1 \sim N(\mu_1, (\sigma_1)^2)$ and $X_2 \sim N(\mu_2, (\sigma_2)^2)$. Since those two random variables are independent, their covariance is zero. Then, the covariance matrix Σ can become following.

$$Cov(X) = E[(X-\mu)(X-\mu)^{\mathrm{T}}]$$

$$Cov(X_1, X_2) = E[(X_1-\mu_1)(X_2-\mu_2)] = 0$$

$$Cov(X) = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_2, X_1) & Var(X_2) \end{bmatrix}$$

$$\mathrm{Cov(X)} = \Sigma = \begin{bmatrix} (\sigma_1)^2 & 0 \\ 0 & (\sigma_2)^2 \end{bmatrix}$$

The probability density of X is given by joint density of (X_1, X_2) that is just multiplication of those two individual densities.

$$P(X) = P(X_1)P(X_2)$$

$$P(X_1) = \frac{1}{(2\pi)^{\frac{1}{2}} \sigma_1^{\frac{1}{2}}} \exp\left[-\frac{\frac{1}{2}(X_1-\mu_1)^2}{2\sigma_1^2}\right]$$

$$P(X_2) = \frac{1}{(2\pi)^{\frac{1}{2}} \sigma_2^{\frac{1}{2}}} \exp\left[-\frac{\frac{1}{2}(X_2-\mu_2)^2}{2\sigma_2^2}\right]$$

$$P_j(X) = \frac{1}{2\pi |\Sigma|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(X-\mu)^{\mathrm{T}}\Sigma^{-1}(X-\mu)\right]$$

Similarly, two random variables are applied to X and μ. The testing data vector X and mean vector μ are expressed as following:

$$X = \begin{bmatrix} data\ from\ feature1 \\ data\ from\ feature2 \end{bmatrix}$$

$$\mu = \begin{bmatrix} \mu\ from\ feature1 \\ \mu\ from\ feature2 \end{bmatrix}$$

The Bivariate Gaussian Distribution equation represents the conditional probability of getting image X given the image of digit j.

$$P(X|Y = j) = P_j(X)$$

Where j is a class, X is a random variable, and Y is a class label.

That is, by using this BGD equation shown above and calculating the prior probability, we can calculate the posterior probability.

$$P(Y = j|X) = \frac{P(Y = j)P(X|y = j)}{P(X)} = \frac{\pi_j P_j(X)}{\sum \pi_i P_i(X)}$$

However, in this experiment, among two classes (image 7 and image 8), we need to compare those posterior probabilities, so normalization was not executed (i.e. no calculation for the marginal probability $P(X)$ because the posterior probability is proportional to the product of the prior and conditional probabilities $P(Y = j|X) \sim \pi_j P_j(X)$).

The prior probability can be calculated as number of samples divided by total number of samples from a training dataset. In this particular experiment, the training dataset from MNIST provides 6265 samples for image 7 out of 12116 samples. Then, the prior probability for image 7 can be calculated.

Since we have only two classes, the prior probability of image 8 will be calculated as $1 - P(Y = j = 0)$.

NOTE: j = 0 means the labels. 0 labels indicate image 7 and 1 labels indicate image 8.

After calculating the prior and conditional probabilities, the term $\pi_j P_j(X)$ proportional to the posterior probabilities were calculated for image 7 and image 8 by using the Multivariate Normal Distribution equation to check which probability is greater. By comparing those two probabilities from image 7 and image 8, if a posterior probability of 7 is greater, a predicted label was classified as 0. Otherwise, predicted label was set as 1.

Once all the testing samples are classified, these estimated labels are compared with the actual labels for calculating the accuracy. The accuracy can be mathematically expressed following.

$$acc = \frac{\#\ of\ matches\ between\ est.\ and\ act.\ labels}{\#\ of\ total\ testing\ data\ samples}$$

- **Result for Gaussian Naïve Bayes**

After reducing the original dimension of pixels, the mean and standard deviation of feature1are 0.1145 and 0.0306 respectively. Similarly, values for feature 1 and feature 2 can be found in the following table.

	$Cov(X_1)$	$Cov(X_2)$
Feature 1	0.1145	0.0306
Feature 2	0.2876	0.0382

Table1. Mean and standard deviation for feature 1 and feature 2.

Since the Multivariate Gaussian Distribution formula requires covariance matrix and its determinant, the calculated values can be found in Table2.

Cov Matrix for Feature 1		Cov Matrix for Feature 2	
0.00094	0.00000	0.00149	0.00000
0.00000	0.00146	0.00000	0.00160
det	1.37E-06	det	2.38E-06

Table2. Covariance matrices and corresponding determinants for feature 1 and feature 2.

The prior probability of image 7 is number of images 7 divided by total number of images. 6265 samples for image 7 out of 12116 samples was 0.517 (i.e. $P(Y = j = 0) = 0.517$). Similarly, the prior probability of image 8 was calculated as 0.483.

By using the Bivariate Gaussian Distribution equation, 2002 testing samples were classified, and the accuracy of the algorithm was calculated as 69.53%.

2) Algorithm Analysis of Logistic Regression

Logistic Regression technique uses a logistic sigmoid function for modeling directly p(X|Y). From a given parameters ω, conditional probability is defined as following.

$$P_j(X) = \sigma(XW)^{\mathrm{Y}}[1-\sigma(XW)]^{1-Y}$$

Where $\sigma(.)$ is a sigmoid function, W is a weight vector, X is a random variable of a training dataset, and Y is the labels of a training dataset.

Conditional likelihood is the conditional probability of getting the data that we see. In this experiment, the data we want to see is image 7 or image 8. If conditional likelihood can be expressed as a function of parameter, we want the parameter which maximize the likelihood. Like the assumption for Gaussian Naïve Bayes, we assume the features are independent. Then, the conditional likelihood can be expressed as following.

$$P_j(X) = \prod_{j=0}^{d} \sigma\left(X_j W\right)^{\mathrm{Y}_j}\left[1-\sigma\left(X_j W\right)\right]^{1-Y_j}$$

Where d is a number of features.

This conditional likelihood can be taken by logarithm to facilitate finding a maximum value because the logarithm is a monotonically non-decreasing function. That is, the function itself has the maximum value at a parameter, the log of the function has the maximum value at the same parameter. By differentiating the log of the likelihood equation, the term indicates the slope of the function. Once the slope is zero, the value is the maximized likelihood. Then, we should find the one that has the maximum likelihood for the data.

$$\log[L(W)] = \log\{\sigma(XW)^{Y}[1-\sigma(XW)]^{1-Y}\}$$

$$= Y\log\{\sigma(XW)\} + (1-Y)\log\{1-\sigma(XW)\}$$

$$\frac{\partial}{\partial W}\log[L(W)] = 0$$

NOTE: this is called a cost (error) function J.

$$J = Y\log\{\sigma(XW)\} + (1-Y)\log\{1-\sigma(XW)\}$$

For finding the optimal weight vector that provides the maximum likelihood of the data, we cannot use the equation shown above because the sum of random number of terms make itself have no closed form.

Thus, we can use a gradient ascent approach with a concave cost function. Once a weight vector is randomly selected as a certain point, the cost function $f(W)$ and its slope (gradient) $f'(W)$ at that point can be calculated. Here, the slope will help you to find which direction to go finding the maximum on the concave cost function $f(W)$. The gradient ascent algorithm is shown below.

$$W_i \leftarrow W_{i-1} + \eta\,(grad)$$

Where i is time, η is a learning rate, and grad is a slope of the cost function as $f'(W)$. The grad can be expressed as following.

$$grad = X^{\mathrm{T}}\left(Y - \sigma(XW)\right)$$

Where X is the vector of the dimensionally reduced training data from MNIST, Y is the vector of the training labels, and W is a weight vector.

Within a loop of iterations, these factors of a gradient and a weight vector are updated; the initial weight vector can be updated with a gradient which let it climb to the peak on the cost function.

Since X is a 12116 x 3 matrix (because of bias for weights, the data was augmented by adding a vector of 1s), a weight vector W was initialized as a 3 x 1 vector. The following matrix forms shows X and W.

$$X = [1, feature1, feature2]$$

$$W = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}$$

Then, by using the gradient ascent algorithm, a weight vector was updated until the cost function J has small changes (the difference was set up as 0.001).

Once a new weight vector was found, using a sigmoid function, the posterior probability was directly calculated with the 2002 testing data samples.

Those individually calculated posterior probabilities were stored in an array. Using a threshold of 0.5, a posterior probability in the array is less than 0.5, the corresponding data was classified as 0. Otherwise, it was classified as 1 (as a reminder, label 0 indicates image 7 and label 1 indicated image 8).

As the same fashion of calculating an accuracy in Gaussian Naïve Bayes, an accuracy was calculated by using the following formula.

$$acc = \frac{\#\ of\ matches\ between\ est.\ and\ act.\ labels}{\#\ of\ total\ testing\ data\ samples}$$

- **Result for Logistic Regression**

The initial weight vector was randomized, a number of iterations was defined as 10,000, and the learning rate was set 0.0009. Within a while loop, a gradient and a weight vector were updated for achieving the maximum likelihood. Once the difference of cost function J in time i from previous one (i.e. J in time i-1) is smaller than 0.001, a new weight vector was finally updated as $[\omega_0, \omega_1, \omega_2]^T = [2.934, 87.025, -43.043]^T$. As the number of iterations increases, the value of cost function J converges to 68.023.

Then, the inner (dot) product of the newly updated weight vector with the testing data samples was computed in a sigmoid function to calculate a posterior probability directly.

These posterior probabilities were checked by a threshold of 0.5; the test data samples of a posterior probability which is less than 0.5 were labeled as 0, and the others were labeled as 1. In this approach, by modifying parameters such as a learning rate or number of iteration, the best accuracy of the algorithm was calculated as 67.28%.

III. DISCUSSION

Both Gaussian Naïve Bayes and Logistic Regression techniques are linear, so those learning models are limited to the non-linearly separable data. The given data shown in Figure1 is not linearly separable.

Because of the limitation, both accuracies are not high enough (69.53% and 68.33% respectively). The accuracies without any packages are compared with those with SciKit-Learn package (sklearn accuracies: 70.21% and 70.49% respectively). The accuracy can be improved with some additional approaches; 1) calculating posterior probability pixel by pixel without reducing the dimension of the original training set, 2) discretizing the pixel values as 0 or 1 not continuous values as a gray-scale for Naïve Bayes and 1) using stochastic gradient, 2) using a batch gradient, and 3) using a posteriori estimate.

For the comparison of computational time, Logistic Regression requires much more time than Gaussian Naïve Bayes depending on number of iterations set up for convergence of the cost function. Also, normalization would help complicated numerical computations. Thus, the computational time can be improved if 1) minimal number of iterations is known 2) the data is normalized.