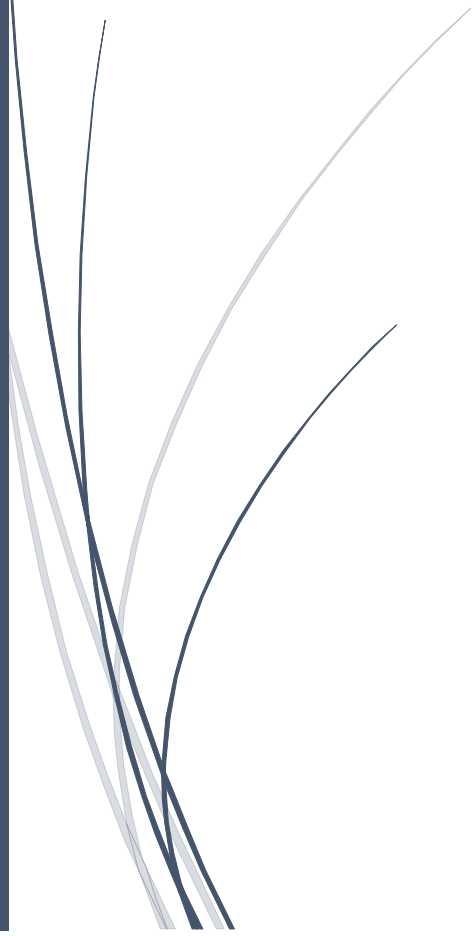
9/22/2019

Density Estimation and Classification

Raktim Mukhopadhyay
ASU ID - 1217167380

CS 575 – STATISTICAL MACHINE LEARNING



Objective

1. Feature Extraction from the provided dataset
2. Parameter Estimation for the Normal Distribution
3. Calculating the Model Parameters for Naïve-Bayes Classifier and Logistic Regression
4. Train the Naïve Bayes Classifier using the training data for classifying the test data
5. Train the Logistic Regression Classifier for classifying the test data
6. Calculating the classification accuracy for Naïve-Bayes and Logistic Regression Classifier

Dataset Description

The dataset used in this project is a subset of MNIST dataset consisting of Class – 7 and Class – 8 images. The test and training dataset are represented by “trX” and “tsX” respectively. Their corresponding labels or targets are stored in “trY” and “tsY” respectively. The total number of samples in the training dataset are: “7” :6265; “8”:5851. The number of samples in the testing dataset are "7": 1028; “8": 974.

Feature Extraction

The testing and training dataset have 784 features – which represent the pixel values of each image. We need to extract only two features – MEAN and STANDARD DEVIATION for each of these images. Hence, we need to extract mean and standard deviation for 12116 images of the training dataset and 2002 images of the training dataset. The extracted features are then stored in another column stacked NumPy array.

$$Mean_{Feature} = \frac{\sum_{i=0}^{783} pixel\_value}{784}\,;\; SD_{Feature} = \sqrt{\frac{\sum_{i=0}^{783}(pixel_{value} - Mean_{Feature})^2}{784}}$$

Parameter Estimation for Normal Distribution

The parameters for Normal Distribution are mean and SD of the features.

For the first feature i.e. Mean -

$$Mean_{Mean_{Feature}} = \frac{\sum_{i=0}^{n-1} Mean_{Feature}}{n}\,;$$

$$SD_{Mean_{Feature}} = \sqrt{\frac{\sum_{i=0}^{n-1}\left(Mean_{Feature} - Mean_{Mean_{Feature}}\right)^2}{n}}$$

For the second Feature i.e. Standard Deviation-

$$Mean_{SD_{Feature}} = \frac{\sum_{i=0}^{n-1} SD_{Feature}}{n}$$

$$SD_{Mean_{Feature}} = \sqrt{\frac{\sum_{i=0}^{n-1}\left(Mean_{Feature} - Mean_{Mean_{Feature}}\right)^2}{n}}$$

CLASS OF IMAGE	MEAN	STANDARD DEVIATION
7	Feature – Mean : 0.11452 Feature – SD : 0.28755	Feature – Mean : 0.03063 Feature – SD : 0.03820
8	Feature – Mean : 0.15015 Feature – SD : 0.32047	Feature – Mean : 0.03863 Feature – SD : 0.03996

Naïve-Bayes Classifier

The probability that an image belongs to Class -7 is represented as -

*P (Y=Class – 7 / X) = P (X/Y=Class 7) * P (Y= Class 7) / P(X)*

Where P (Y=Class – 7/X) is known as the posterior, P (X/Y=Class 7) is known as the likelihood, P (Y= Class 7) is known as Class Prior Probability and P(X) is known as Predictor Prior Probability.

The probability that an image belongs to Class -8 is represented as -

*P (Y=Class – 8 / X) = P (X/Y=Class 8) * P (Y= Class 8) / P(X)*

Where P (Y=Class – 8/X) is known as the posterior, P (X/Y=Class 8) is known as the likelihood, P(Y= Class 8) is known as Class Prior Probability and P(X) is known as Predictor Prior Probability.

The Naïve condition is that we consider that all the features are independent. Hence, we can rewrite the Posterior as –

*P (Y=Class 7/X) = P (Class -7/X_1) * P(Class – 7/X_2) where X_1 and X_2 are the two features respectively.*

Similarly, for the Class – 8, we can write -

*P (Y=Class 8/X) = P (Class -8/X_1) * P (Class – 8/X_2) where X_1 (Mean) and X_2 (SD) are the two features respectively.*

We assume that P(X/Y) follows a Normal Distribution. Hence, we can calculate P(X/Y) from the *p_x_given_y ()* function defined in code. The Normal Distribution function is defined as -

```
def p_x_given_y (x, mean, variance):
    p = (1/(np.sqrt(2*np.pi*variance)))*np.exp(-(x-mean)**2/(2*variance))
    return p
```

We need to calculate the numerator of the posterior probability and then compare the values to predict whether the image in the test dataset belongs to Class – 7 or Class - 8

Numerator of Class- 7 is calculated as -

```
numr_postr_class_7 = p_class_7*
p_x_given_y(test_mean_images,mean_images_7.mean(),mean_images_7.var())*
p_x_given_y(test_sd_images,sd_images_7.mean(),sd_images_7.var())
```

Similarly, we can calculate the numerator for Class-8

Probability of Class 7 = (Total number of Images in Training Dataset Belonging to Class 7/ Total Number of images in the Training Dataset)

Probability of Class 8 = 1 - Probability of Class 7

We calculate the numerator of posterior probability for class 7 and class 8 and compare the values. If the numerator of posterior probability of class 7 is greater than the numerator of posterior probability of class 8, then the image is classified as belonging to Class -7 or vice versa.

Logistic Regression Classifier

In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 (representing Class -7) and 1 (representing Class – 8).

The hypothesis is represented as $h_\theta(x) = g(\theta^T x)$

$z = \theta^T X$, θ represents the weight and X represents the inputs. We apply the sigmoid function to our hypothesis to get our final prediction for the logistic regression classifier. The sigmoid function is given by –

$$g(z) = \frac{1}{1 + e^{-z}}$$

The parameter of the logistic function is θ, which is the coefficient or the weight. We need to calculate the parameter θ, such that the log likelihood is maximized –.

$\mathbf{w}^* = \text{argmax}_{\mathbf{w}} l(\theta)$

$\mathbf{w}^* = \text{argmax}_{\theta} \sum_{i=1}^{n} [y^{(i)} \theta^t x^{(i)} - \log\left(1 + \exp\left(\theta^t x^{(i)}\right)\right)]$

We cannot really solve for w* analytically (no closed-form solution). We will use gradient ascent to maximize the equation.

Gradient Ascent in Logistic Regression -

The gradient ascent equation which is used to find the θ is given by –

$$\theta^{(k+1)} = \theta^{(k)} + \eta \nabla_{\theta^{(k)}} l(\theta)$$

$\eta > 0$ is a constant called the learning rate, $l(\theta)$ represents the log likelihood function for the logistic regression.

$\nabla_{\theta^*} l(\theta)$ is given by - $\nabla_{\theta^*} l(\theta) = \sum_{i=1}^{n} [X^{(i)}(Y^{(i)} - h^{(i)})]$

We will substitute the above equation in the gradient ascent equation. We set the *number of iterations as 100000 and the learning rate as 0.001.*

If the result of the sigmoid function is more than 0.5, then it is classified as belonging to Class-8., else Class – 7.

The value of the θ parameters determined by Logistic Regression are- **[23.0058898 246.35633793 -180.19141079]**

Accuracy of Naïve-Bayes Classifier and Logistic Regression

CLASSIFIER	OVERALL ACCURACY	CLASS - 7	CLASS - 8
NAÏVE BAYES	69.53%	75.97	62.73%
LOGISTIC REGRESSION	81.66%	77.72%	85.83%

It must be noted that Naïve Bayes is a very simple classifier. Even though we know that Mean and SD as feature are correlated, we consider them to be independent for the sake of simplicity. The computation time for fitting the Naïve Bayes classifier is substantially less as compared to Logistic Regression Classifier.

The accuracy of Logistic Regression is much better compared to Naïve-Bayes Classifier