CSE 575 – STATISTICAL MACHINE LEARNING

Class # 18194 – Spring 2021 Instructor – **Nupur Thakur**

PROJECT Part – 1

Density Estimation and Classification using Fashion-MNIST

Report by - **Subba Raja Kashyap Saligrama** **(Asu Id:** 1219510851**)**

Dataset used:

The dataset given is a subset of the Fashion-MNIST dataset. It is provided in form of a single mat file. The file contains both training and testing datasets along with their labels.

Number of samples in training set: "Tshirt": 6000 ; "Trouser": 6000

Number of samples in testing set: "Tshirt": 1000 ; "Trouser": 1000

Algorithms Implemented:

- Naïve Bayes Classifier
- Maximum Likelihood Estimation for Naïve Bayes
- Logistic Regression
- Gradient Ascent method for Parameter Estimation in Logistic Regression.

Programming Language, workspace, software used:

- Python is used to implement the source code of Naïve Bayes and Logistic Regression from the scratch.
- Python 3.9 environment is used to execute the program.
- 'Scipy' library is used to import and read the mat file as the dataset is in form of a dictionary.
- 'Numpy' library is used to perform mathematical operations on the datasets.

<u>Importing and Reading Dataset:</u>

Initially, we first import and load the data (training and testing) form the mat file with help of 'Scipy' library into the following labels –

trX– training samples
trY – labels for training samples

tsX – testing samples
tsY – labels for testing samples

<u>Preparing training data:</u>

- In the given dataset, each sample is a grayscale image of size 28x28 pixels.
- Converting into feature vectors, we have a 784x1 feature vector for each sample.
- We assume that each image is drawn from a 2-D Gaussian distribution. Hence, we can use mean and standard deviation as features and reduce the feature size from 784 to 2.
- The new features are '*mean*' and '*standard deviation*' of the 784 pixel values of each sample.
- Hence, the new feature vector for each sample will be 2x1 : [μ σ] where,

  Mean – $\mu = \sum_{i=1}^{d} x_i / d$ Standard deviation – $\sigma = \sqrt{\sum_{i=1}^{d} \frac{(x_i - \mu)^2}{d}}$

  d – number of pixel values ; x_i – value of i^{th} pixel value
  In the program, these values are calculated for all 12000 training samples using '*mean( )*' and '*std( )*' methods of 'Numpy' library.

- Now, we update the *trX* with these feature values of all 12000 training samples and use them as input variables instead of 784 pixel values of each sample.
- We append the label values *trY* to training sample data *trX* for easy reference of given label values for each sample and call it *trData*. *trData* – $[\,\mu_i \;\; \sigma_i \;\; y_i\,]_{12000\ x\ 3}$

- We, then, split *trData* based on class labels.
  There are two classes here – Tshirt (0) ; Trouser (1) and there are 6000 samples in each class.
  trData0 – $[\,\mu_i \quad \sigma_i \quad y_i = 0\,]_{6000\ x\ 3}$ *trData1* – $[\,\mu_i \quad \sigma_i \quad y_i = 1\,]_{6000\ x\ 3}$

- Then, we calculate the prior probabilities of class '0' [$P(y = 0)$] & '1' [$P(y = 1)$].

  As, out of 12000 samples, 6000 belong to class '0' & '1' each respectively,

$$P(y = 0) = \frac{6000}{12000} = 0.5 \qquad P(y = 1) = 1 - P(y = 0) = 0.5$$

<u>Preparing testing data:</u>

- We perform the same operations done on training data on testing dataset also and get variables *tsData*, *tsData0*, *tsData1* in below formats –

  tsData – $[\,\mu_i \;\; \sigma_i \;\; y_i\,]_{2000\ x\ 3}$
  tsData0 – $[\,\mu_i \quad \sigma_i \quad y_i = 0\,]_{1000\ x\ 3}$
  tsData1 – $[\,\mu_i \quad \sigma_i \quad y_i = 1\,]_{1000\ x\ 3}$

Measuring Accuracy:

The prediction accuracy of algorithms is computed as follows –

$$\text{Accuracy for class 'k'} = \frac{\text{Number of correctly 'k' classified samples in test dataset}}{\text{Total number of 'k' classified samples in test dataset}} * 100$$

Naïve Bayes Classifier:

In Naïve Bayes algorithm, we make a naïve assumption that – the features of the given dataset are conditionally independent to each other. So, for a d-dimensional feature vector X,

$$p(X/y) = p(x_1, x_2, \ldots, x_d/y) = p(x_1|y).p(x_2|y) \ldots \ldots p(x_d|y)$$

Now, to find the probability of a sample belonging to a certain class, this algorithm uses the Bayes theorem –

$$P(Y = y_k|X) = \frac{P(Y=y_k)\, P(X|Y=y_k)}{P(X)} \qquad \text{for a class 'k'}$$

Here,

$P(Y = y_k|X)$ is called Posterior Probability $P(Y = y_k)$ is called Prior Probability

$P(X|Y = y_k)$ is called Likelihood $P(X)$ is called Evidence

We can ignore $P(X)$ here as it doesn't change for the given sample.

So, $$P(Y = y_k|X) \propto P(Y = y_k)\, P(X|Y = y_k)$$

$$\Rightarrow \quad P(Y = y_k|X) \propto P(Y = y_k)\; P(x_1, x_2, ..x_d \mid Y = y_k)$$

$P(Y = y_k)$ can be directly obtained from observing the given dataset. The only term we need to find now is the Likelihood term. But, as we assume that the features are conditionally independent, the above equation changes for our 2-D feature vector of each sample as follows –

$$P(Y = y_k|X) \propto P(Y = y_k)\, P(x_1|Y = y_k)\, P(x_2|Y = y_k) \qquad \text{--- } \mathbf{(i)}$$

As the prior probability is constant for given dataset, if a given sample X belongs to class 'k', then its $P(Y = y_k|X)$ will be higher among all classes.

This is possible if the likelihood of the sample X being in y_k is maximum. Hence, in order to find the best posterior probability of a test sample X', we need to estimate maximum likelihood of X'.

Maximum Likelihood Estimation:

As we assumed that the dataset is derived from a Gaussian distribution, we use a type of Parametric Density Estimation here called, Maximum Likelihood Estimation (MLE) to find the best likelihood. Here, we assume that all samples are independent & identical distributed (i.i.d).

Then, for Dataset with n samples, $D = \{x_1, x_2, \dots x_n\}$

Data Likelihood, $L(\theta) = P(D|\theta) = \prod_{i=1}^{n} P(x_i|\theta)$

Now, we perform MLE and find a specific value for parameter $'\theta'$ that maximizes $L(\theta)$.

$$\therefore \quad \hat{\theta} = \underset{\theta}{argmax} \prod_{i=1}^{n} P(x_i|\theta)$$

But, $P(x_i|\theta)$ value is ≤ 1. So, as the number of samples increase, $L(\theta)$might get undervalued. To avoid this, we apply log function to $L(\theta)$. As, log is a monotonic function, it won't affect our expression in any unwanted manner.

$$\therefore \quad \hat{\theta} = \underset{\theta}{argmax} \log\left(\prod_{i=1}^{n} P(x_i|\theta)\right) = \underset{\theta}{argmax} \sum_{i=1}^{n} \log\left(P(x_i|\theta)\right)$$

Now, coming to our program, the assumption is that the dataset comes from a Gaussian distribution and the features μ, σ are i.i.d [$\theta = \{\mu, \sigma\}$ here]. So, we can use the Probability Density Function (PDF) of Univariate Gaussian distribution :–

$$P(x|\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \qquad \text{--- } \mathbf{(ii)}$$

Now, applying log,

$$l(\theta) = \log(L(\theta)) = \log\left(\prod_{i=1}^{n} P(x_i \mid \mu, \sigma)\right) = \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \sum_{i=1}^{n} \frac{(x_i - \hat{\mu})^2}{2\hat{\sigma}^2}$$

applying derivative w.r.t μ,

$$\frac{\partial}{\partial\mu} l(\theta) = 0 - \sum_{i=1}^{n} \frac{2(x_i - \hat{\mu})}{2\hat{\sigma}^2} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \hat{\mu})$$

and equating to $'0'$,

$$\frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} (x_i - \hat{\mu}) = 0 \quad \rightarrow \quad \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \hat{\mu} = 0 \quad \rightarrow \quad \sum_{i=1}^{n} x_i - \hat{\mu}.n = 0$$

$$\therefore \quad \textbf{Sample mean,} \quad \hat{\boldsymbol{\mu}} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{x_i}$$

Similarly, applying log, applying derivative w.r.t σ, and equating to $'0'$, we get –

$$\textbf{Sample Standard Deviation,} \quad \hat{\boldsymbol{\sigma}}^{\mathbf{2}} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} (\boldsymbol{x_i} - \hat{\boldsymbol{\mu}})^{\mathbf{2}}$$

So, we will find $\hat{\mu}$ & $\hat{\sigma}$ for each of the 2 input features ($x_1 = \mu$, $x_2 = \sigma$) for each class label ('0', '1'). Then, we will get eight MLE parameters in total –

Let them be, $\hat{\mu}_{0_\mu}, \hat{\sigma}_{0_\mu}$ for feature 1, class '0' ; $\hat{\mu}_{0_\sigma}, \hat{\sigma}_{0_\sigma}$ for feature 2, class '0'

$\hat{\mu}_{1_\mu}, \hat{\sigma}_{1_\mu}$ for feature 1, class '1' ; $\hat{\mu}_{1_\sigma}, \hat{\sigma}_{1_\sigma}$ for feature 2, class '1'

Now, using equations **(i)** and **(ii)**, substituting these values will give –

$$P(Y=0\,|\,X) \;\propto\; P(Y=0)\left(\frac{1}{\sqrt{2\pi}\,\hat{\sigma}_{0_\mu}}\;e^{-\left(\frac{\left(x-\hat{\mu}_{0_\mu}\right)^2}{2\,\hat{\sigma}_{0_\mu}{}^2}\right)}\right)\left(\frac{1}{\sqrt{2\pi}\hat{\sigma}_{0_\sigma}}\;e^{-\left(\frac{\left(x-\hat{\mu}_{0_\sigma}\right)^2}{2\,\hat{\sigma}_{0_\sigma}{}^2}\right)}\right)$$

$$P(Y=1\,|\,X) \;\propto\; P(Y=1)\left(\frac{1}{\sqrt{2\pi}\hat{\sigma}_{1_\mu}}\;e^{-\left(\frac{\left(x-\hat{\mu}_{1_\mu}\right)^2}{2\,\hat{\sigma}_{1_\mu}{}^2}\right)}\right)\left(\frac{1}{\sqrt{2\pi}\hat{\sigma}_{1_\sigma}}\;e^{-\left(\frac{\left(x-\hat{\mu}_{1_\sigma}\right)^2}{2\,\hat{\sigma}_{1_\sigma}{}^2}\right)}\right)$$

Finally, for a sample X,

if $P(Y=0\,|\,X) > P(Y=1\,|\,X)$, it is classified as class $'0'$, else it is classified as class $'1'$.

Implementing the above whole process, the estimated parameter values in the **Naïve Bayes** program are –

Class ‘0’ :	$\boldsymbol{\hat{\mu}_{0_\mu} = 0.3256077}$	$\boldsymbol{\hat{\sigma}_{0_\mu} = 0.1133749}$
	$\boldsymbol{\hat{\mu}_{0_\sigma} = 0.32003608}$	$\boldsymbol{\hat{\sigma}_{0_\sigma} = 0.0879828}$
Class ‘1’ :	$\boldsymbol{\hat{\mu}_{1_\mu} = 0.2229053}$	$\boldsymbol{\hat{\sigma}_{1_\mu} = 0.0569510}$
	$\boldsymbol{\hat{\mu}_{1_\sigma} = 0.33394171}$	$\boldsymbol{\hat{\sigma}_{1_\sigma} = 0.0570322}$

And, the prediction accuracy is as follows –

Accuracy for class ‘0’	=>	78.4 %
Accuracy for class ‘1’	=>	87.9 %

Total accuracy using Naïve Bayes => 83.15 %

Logistic Regression:

Logistic Regression doesn't require any prior probabilities and likelihoods to calculate the posterior probability. It directly tries to find $P(Y|X)$. It makes the assumption that the samples are linearly separable. Hence, it uses a linear model (decision boundary) to classify them (denoted as W^TX).

For a d-dimensional vector, $W^TX = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d$

Here, W is the weighted vector & X is the input feature vector.

As, our input feature vector is 2-d, $W^TX = w_0 + w_1x_1 + w_2x_2$

A sigmoid function is applied on $W^T X$ which gives the probability $P(Y|X)$. The sigmoid function type used here is *'Logistic Function'* **:-** $\sigma(x) = \frac{1}{1+e^{-x}}$

This function is considered because –

- It restricts the output to $[0,1]$ range, which is perfect for probability functions.
- It returns a probability of **0.5** for input **0** , which is correct for a sample that lies on the model.

Now, we have to find the coefficients $\mathbf{w_i}$ such that $\boldsymbol{\sigma}(\mathbf{W^T X})$ is maximized for all samples.

The likelihood expression then becomes,

$$L(w) = \prod_{i=1}^{n} \mathrm{P(Y = y_i \mid X = x_i)} = \prod_{i=1}^{n} (w^T x_i)^{y_i} \, (1 - w^T x_i)^{1-y_i}$$

Let $\boldsymbol{z} = \boldsymbol{\sigma}(w^T x)$. Also, log is applied to make the process of solving easier.

Then, the log-likelihood equation to maximize becomes –

$$l(w) = \log(L(w)) = \sum_{i=1}^{n} \left(y^{(i)}.\log\left(z^{(i)}\right) + \left(1 - y^{(i)}\right).\log\left(1 - z^{(i)}\right) \right)$$

Now, we want the coefficients w_i such that $L(w)$ is maximized –

$$\hat{w} = \underset{w}{argmax} \; l(w)$$

We can't directly apply derivation to $l(w)$ and equate it to 0 to find the optimal coefficients w_i as it is not a closed-form equation. Hence, we use Gradient Ascent method to estimate the optimal coefficients.

Gradient Ascent method:

We use a learning rate $\boldsymbol{\eta} > 0$ to update the values of coefficients w_i in each iteration till the obtained coefficient values converge.

$$w_{t+1}^{i} = w_t^{i} + \eta \left(\frac{\partial\, l(w)}{\partial w^i} \right)$$

where $\boldsymbol{t}$ is the iteration number, $\boldsymbol{\eta}$ is the learning rate.

Using chain rule method,

$$\frac{\partial\, l(w)}{\partial w^i} = \frac{\partial\, l(w)}{\partial z} . \frac{\partial z}{\partial p} . \frac{\partial p}{\partial w^i} = \left(\frac{y}{z} - \frac{1-y}{1-z} \right) . z(1-z). x^i = (y - z)\, x^i$$

Substituting this back into Gradient Ascent equation, $w_{t+1}^{i} = w_t^{i} + \eta \left((y - z)\, x^i \right)$

Now, we do this operation on the entire training dataset for a certain number of iterations. To make it easy to perform this operation in program –

- for the training samples array (*of type* $[x_1 \;\; x_2]_{nx2}$), append a nx1 matrix of 1 's to make it a $[1 \;\; \mathrm{x}_1 \;\; \mathrm{x}_2]_{\mathrm{nx3}}$ matrix so that it will be easy to multiply with $[\mathrm{w}_0 \;\; \mathrm{w}_1 \;\; \mathrm{w}_2]^{\mathrm{T}}$ and get $\mathrm{W^T X}$ form.

Once, the parameters $\widehat{w_i}$ are estimated, we use them in $W^T X$ to find the probabilities $P(Y|X)$,

$$P(Y = y_k \mid X) \;=\; \sigma(W^T X) \;=\; \frac{1}{1 + e^{-(W^T X)}}$$

For class $y = 0$, $\quad P(Y = 0 \mid X) \;=\; \frac{1}{1+e^{W^T X}}$

For class $y = 1$, $\quad P(Y = 1 \mid X) \;=\; \frac{e^{W^T X}}{1+e^{W^T X}}$

Finally, classify all the testing samples using above formulas.

Here, as there are 2 classes, we can use $\mathbf{0.5}$ as the threshold for probabilities to classify –

- if $P(y|x) < 0.5$, the sample is classified as Class ' 0 '
- if $P(y|x) \geq 0.5$, the sample is classified as Class ' 1 '

This can be done easily to all samples at a time using $\boldsymbol{around}$ method in 'Numpy' library.

Implementing the above whole process –

Taking the values – *Learning Rate,* $\boldsymbol{\eta}$ = **0.01** *Number of iterations,* $\boldsymbol{e}$ = **1000**

the estimated coefficient values obtained in the **Logistic Regression** program are –

$$\boldsymbol{w_0} = \mathbf{-17.73994442}, \quad \boldsymbol{w_1} = \mathbf{-213.47014233}, \quad \boldsymbol{w_2} = \mathbf{226.98987124}$$

and, the prediction accuracy is as follows –

Accuracy for class '0' => 92.5 %

Accuracy for class '1' => 91.7 %

Total accuracy using Logistic Regression => 92.10 %

<u>Inference:</u>

- Logistic Regression (LR) (92.25 %) performed better than Naïve Bayes (NB) (83.15 %) on this dataset, as NB assumes conditional independency of features whereas LR takes no such assumptions.
- In LR, for η = 0.1, e =1000 – accuracy : 92.05 %
  for η = 0.1, e =10000 – accuracy : 92.20 %
  for η = 0.01, e =1000 – accuracy : 92.10 %
  for η = 0.01, e =10000 – accuracy : 92.25 % (highest of these)
  for η = 0.01, e =100000 – accuracy : 92.10 %
  Not much improvement after e = 1000 for η = 0.01. So, choosing these to show final accuracy.