MEEM 5990 Applied Machine Learning

Spring 2021

Project 2

Total Points: 100

Part 1: Naive Bayes

The objective of this part of the project is to use Naive Bayes classifier for digit classification. We have a training set and test set of data for digits 0-9. We would be using Naive Bayes for recognizing the digits.

Data Set Information:

The image data for this consist of 28 × 28 pixel images stored as text. Each image has 28 lines of text with each line containing 28 characters. The values of the pixels are encoded as ' ' for white and '1' for black. You can view the images by simply looking at them with a fixed width font in a text editor. We will be classifying images of the digits 0-9, meaning that given a picture of a number we should be able to accurately label which number it is. Since computers can't tell what a picture it is we're going to have to describe the input data to it in a way that it can understand. We do this by taking an input image and extracting a set of features from it. A feature simply describes something about our data. In the case of our images, we can simply break the image up into the grid formed by its pixels and say each pixel has a feature that describes its value.

We will be using these images as a set of binary features to be used by our classifier. Since the image is 28 × 28 pixels we will have 28 × 28 = 784 features for each input image. To produce binary features from this data we will treat pixel *i, j* as a feature and say that *Fi,j* has a value of 0 or 1.

We will be using the training images to train our classifier specifically to compute probabilities that we will use and then try to recognize the test images.

Following files are provided:

- readme.txt - description of the files in the zip
- testimages - 1000 images to test
- testlabels - the correct class for each test image
- trainingimages - 5000 images for training
- traininglabels - the correct class for each training image

Overview of Naive Bayes Classifier

Naive Bayes classifiers are based on Bayes' theorem which describes the probability of an event based on prior knowledge of conditions that might be related to the event. In our case we are computing the probability that an image might belong to a class based on the value of a feature. This idea is called conditional probability and is the likelihood of event A occurring given event B and is written as $P(A|B)$. From this we get Bayes' theorem.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

This says that if A and B are events and $P(B)$ not equal to 0 then

- $P(A|B)$ is the probability that A occurs given B
- $P(B|A)$ is the probability that B occurs given A
- $P(A)$ the probability A occurs independently of B
- $P(B)$ the probability Boccurs independently of A

Applying this to our case we want to compute for each feature the probability that given the feature $F_{i,j}$ has a value $f \in \{0,1\}$ that the image will belong to a given class $c \in \{0, 1, \ldots 9\}$. To do this we will train our model on a large set of images where we know the class that they belong to. Once we have computed these probabilities we will be able to classify images by computing the class with the highest probability given all the known features of the image and assign it to that class.

From this we get two phases of the algorithm.

- Training - compute the conditional probability that for each feature that an image is part of each class.
- Classifying - for each image compute the class that it is most likely to be a member of given the assumed independent probabilities computed in the training stage.

Training

The goal of the training stage is to teach the computer the likelihoods that a feature $F_{i,j}$ has value $f \in \{0, 1\}$ as we have binary features, given that the current image is of class $c \in \{0, 1, \ldots, 9\}$, which we can write as $P(F_{i,j} = f \,|\, class = c)$

This can be simply computed as follows:

$$P(F(i,j) = f|class = c) = \frac{Number\ of\ times\ F(i,j) = f\ when\ class = c}{Total\ number\ of\ training\ examples\ where\ class = c}$$

You should also estimate the priors $P(class = c)$ or the probability of each class independent of features by the empirical frequencies of different classes in the training set.

$$P(class = c) = \frac{Number\ of\ training\ examples\ where\ class = c}{Total\ number\ of\ training\ examples}$$

With the training done you will have an empirically generated set of probabilities that can be referred to as the model that we will use to do classification on unknown data. This model once generated can be used in the future without needing to be regenerated.

Classification

To classify the unknown images using the trained model you will perform maximum a posteriori (MAP) classification of the test data using the trained model. This technique computes the posterior probability of each class for the particular image. Then classifies the image as belonging to the class which has the highest posterior probability.

Since we assume that all the probabilities are all independent of each other we can compute the probability that the image belongs to the class by simply multiplying all the probabilities together and dividing by the probability of the feature set. Finally, since we don't actually need the probability but merely to be able to compare the values we can ignore the probability of the feature set and thus compute the value as follows:

$$P(class) * P(f_{1,1}|class) * P(f_{1,2}|class) * \ldots * P(f_{28,28}|class)$$

Unfortunately, when computers do floating point arithmetic, they are not quite computing as we do by hand and they can produce a type of error called underflow. To avoid this problem, we will compute using the log of each probability rather than the actual value. This way the actual computation will be the following:

$$\log(P(class)) + \log(P(f_{1,1}|class)) + \log(P(f_{1,2}|class)) + \ldots + \log(P(f_{28,28}|class))$$

Where $f_{i,j}$ is the feature value of the input data. You should be able to use the priors and probabilities you calculated in the training stage to calculate these posterior probabilities.

Once you have a posterior probability for each class from 0-9 you should assign the test input to the class with the highest posterior probability.

Deliverables:

- A single code file which should run on a click when the trainingimages, testimages, traininglabels, testlabels files are placed in the working directory.
- A pdf file of all the results

Following tasks are to be performed:

1. Obtain test data accuracy. (Trick: You can use Laplace smoothing for Naïve Bayes to increase the

accuracy. Use of Laplace smoothing is not mandatory for this project. However if you face issues of zero probability you need Laplace smoothing.)

2. Obtain confusion matrix. This is a 10 × 10 matrix whose entry in row r and column c is the percentage of test images from class r that are classified as class c. Use the true class labels of the test images from the testlabels file to obtain the confusion matrix.

Part 2: Logistic Regression

The objective of this part is to use logistic regression on Sonar data set available at UCI Machine Learning Dataset Repository.

Dataset information

Information for this dataset is available at
http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)

Deliverables:

- A single code file which should run on a click when the sonar5841.mat file is placed in the working directory. (Functions are to be written at the end of file in MATLAB)
- A pdf file of all the results

Following tasks are to be performed:

1. Loading the data in MATLAB

   ```
   load('sonar.mat')
   ```

   After loading the data, you will have 2 variables in your workspace: X and Y. X is the data matrix where rows are observations. Y contains the class labels, either -1 or +1. There are 208 observations in this dataset.

   Select random 25% of the data for testing. Remember to select respective data labels from Y matrix. The remaining data is training data.

   You should have following matrices after this step,

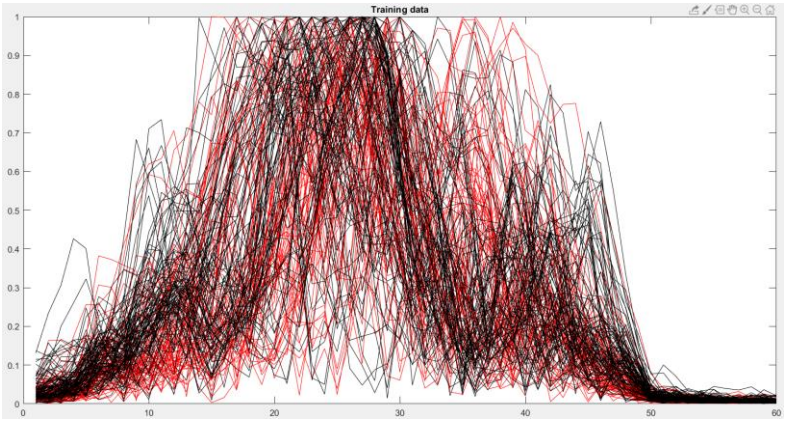
   trainfeatures (156*60)

   trainlabels (156*1)

   testfeatures (52*60)

testlabels(52*1)

2. Plot training data and test data in **2 different plots**. Use different colors for different colors. Here is my training data with red color for label==-1 and black color for label==+1. Don't forget to add titles to the plot. Legend to the plot can be in the plot or can be added as a caption.



3. Write 3 functions.

```
function g = sigmoid(z)
```

Which returns sigmoid of z.

```
function p = pred(theta, X)
```

Which predicts for x given the theta

```
function [J, grad] = costFunction(theta, X, y)
```

which returns cost J and gradient.

Then you can use fminunc in MATLAB to find theta

```
options = optimset('GradObj', 'on', 'MaxIter', 1000);
[theta, cost] = fminunc(@(t)(costFunction(t, x, y)), initial_theta,…
options);
```

Once you obtain theta use your pred function to predict and return Percentage test accuracy.

4. Increase the training data from 75% to 85% and report the testing accuracy.

Point distribution:

Part 1: 50 points (10 for accuracy+40 for confusion matrix)

Part 2: 40 points (5+5+20+10 for the listed 4 tasks)

Readability of the code: 10 points

Check this blog to know about readability of the code
https://blog.pragmaticengineer.com/readable-code/

Acknowledgements:

The digit data was obtained from illinois.edu. The sonar data was obtained from UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Appendix

TO LOAD THE DATA ON MATLAB USE THE FOLLOWING:

```
a = string(fileread('trainingimages'));
data = split(a,newline);
```

For logistic Regression

Cost Function	$\sum_{i=1}^{n} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i})$
	n: number of training samples
Labels	**{-1,1}**
Prediction	$y_i = -1$ if $p(y_i \mid x_i, \mathbf{w}) < 0.5$ $y_i = 1$ if $p(y_i \mid x_i, \mathbf{w}) >= 0.5$