CNSCC.361 Artificial Intelligence

Lab Session 7 Classification

This lab is intended for you to learn more about classification. You will see a MATLAB implementation of the KNN and Naïve Bayes classification methods. Even though MATLAB has built-in functions for KNN and Naïve Bayes classification methods, we will implement these methods from scratch to better understand their details and compare them with the MATLAB built-in functions. You are not expected to go through everything in this document during the 2-hour lab session. It will take you longer to understand the details of code, and you can do this in your own time.

Classification Problem

In classification, the data consists of a training dataset and a validation dataset. The training dataset is a set of feature/attribute vectors and their class labels; and a learning algorithm is used to train a classifier using the training dataset. The validation dataset is also a set of feature/attribute vectors and their class labels which are used for evaluating the performance of the learned classifier.

Iris Dataset

In this lab, we use "iris" dataset, which is a well-known dataset in machine learning. From Moodle, you can find this dataset in the "Lab 08" folder. The data is in the file called "iris_dataset.mat". "iris_dataset.mat" consists of four MATLAB variables including:

- "train_data.mat": There are 75 samples in this file, where each line in the file is one sample. Each sample has 4 features (sepal length, sepal width, petal length, petal width). For example, the first sample says that the sepal length is 5.1, the sepal width is 3.5, the petal length is 1.4, the petal width is 0.2.
- "train_label.mat": Class labels (represented by an integer) of the training data in "train_data.mat". The integer "1" is "Iris setosa", "2" is "Iris versicolor", and "3" is "Iris virginica". For example, the class of the first sample in "train_data.mat" is "Iris setosa", i.e. class 2.
- "validation_data.mat": Once you have trained a classifier, you can evaluate its performance by a new set of data called validation dataset. There are 75 samples in this file, where each line in the file is one validation sample.
- "validation_label.mat": Class labels (represented by an integer: 1,2,3) of the validation data in "validation_data.mat".

k-Nearest Neighbours (kNN) Classifier

An intuitive way to decide how to classify an unlabelled test item is to look at the training data points nearby, and make the classification according to the classes of those nearby labelled data points. This intuition is formalised in a classification approach called k-nearest neighbour (kNN) classification. The kNN approach looks at the k points in the training dataset that are closest to the

test point; the test point is then classified according to the class to which the majority of the K-nearest neighbours belong.

The training of the K-NN classifier is simple; we just need to store all the training set! However, testing is much slower, since it involves measuring the distance between each test point and every training point. Now, let's start implementing k-NN classification algorithm. We first use MATLAB built-in kNN classification function. Then, we implement kNN classifier from scratch to better understand the details of the algorithm.

kNN Classifier (MATLAB built-in function)

From Moodle, you can find the code for this in the "Lab 08" folder. The file "KNN_MATLAB_builtin.m" has all of the code you need. Look through the code to understand what each part of the code is doing. You have to add a few lines of code to this part of the file:

```
%% Call MATLAB built-in kNN Classifier
% need to add code here ...
classifier = fitcknn(train_data,train_label,'NumNeighbors',k);
YPred = predict(classifier,validation_data);
```

In the above code, "`fitcknn`" is a MATLAB built-in function for kNN classification. After adding the code, you should be able to execute it and see the results. You will see a number that is the accuracy of the validation data.

Some Questions about the Code

(1) Which part of the code defines the variable k in kNN algorithm? Try to change its value to 3, 5, 7, 9, 11 and see how the accuracy changes.

(2) Which part of the code implements the accuracy of validation dataset? Try to understand how the accuracy is calculated.

kNN Classifier (Implementing from scratch)

The file "KNN_MATLAB_scratch.m" has all of the code you need. Look through the code to understand what each part of the code is doing. You should be able to execute it and see the results. You will see the accuracy of the validation data as output. Compare this accuracy with the accuracy obtained from MATLAB built-in kNN classifier in "KNN_MATLAB_builtin.m". In order to have the same accuracy, you need to use the same k.

Some Questions about the Code

(1) Which part of the code defines the variable k in kNN algorithm? Try to change its value to 3, 5, 7, 9, 11 and see how the accuracy changes.

(2) Which part of the code compute distance between a validation example and training examples? What type of distance is?

Naïve Bayes Classifier (MATLAB built-in function)

The file “NB_MATLAB_builtin.m” has all of the code you need. Look through the code to understand what each part of the code is doing. You have to add a few lines of code to this part of the file:

```
%% Call MATLAB built-in Naive Bayes Classifier
% need to add code here ...
classifier =  fitcnb(train_data,train_label,'Distribution','normal');
YPred = predict(classifier,validation_data);
```

In the above code, “`fitcnb`” is a MATLAB built-in function for Naïve Bayes classifier. After adding the code, you should be able to execute it and see the results. You will see the accuracy of the validation data as output.

Since all attributes are continuous data, we use Normal/Gaussian distribution to estimate the conditional probability $P(A_i|C_j)$ as you can see in the above-mentioned code (shown in **BOLD**).

Naïve Bayes Classifier (Implementing from scratch)

The file “NB_MATLAB_scratch.m” has all of the code you need. Look through the code to understand what each part of the code is doing. Many of the ideas were explained in the lecture.

You have to add a few lines of code to this part of the file:

```
%% Prior probability
% need to add your own code here ...
priorProb_class1 = …;%% class probability for class 1 samples P(c1)
priorProb_class2 = …;%% class probability for class 2 samples P(c2)
priorProb_class3 = …;%% class probability for class 3 samples P(c3)
```

In the above code, you have to implement the class probability for each three classes in the dataset (it’s only a few lines of code!). After adding the code, you should be able to execute it and see the results. You will see the accuracy of the validation data. Compare this accuracy with the accuracy obtained from MATLAB built-in Naïve Bayes classifier in “NB_MATLAB_builtin.m”.

Some Questions about the Code

(1) Read what the “`prod`” and “`normpdf`” functions do.

(2) Which part of the code computes the conditional probability $(P(A_i \mid C_1),\ P(A_i \mid C_2),\ P(A_i \mid C_3))$ for each of the 3 classes in the dataset? Try to understand how we calculate these probabilities.