Chalmers-University of Gothenburg Department of Mathematical Sciences

Probability, Statistics and Risk, MVE240

Project 6

In the project description we sketch the analysis of the problem we expect you to do. (Obviously you are welcome to do more.) To pass the project a short report should be written and handed in to the project supervisor. In addition the group should present their results in class. The presentation should take about 15 minutes. Please include a short introduction which will facilitate for other students to understand the results of the project. (Do not assume that the audience knows the subject.)

1 Introduction - Pattern recognition and classification

Pattern recognition is a part of what is called machine learning. It could be called the art of decision making based on patterns found in some kind of raw data. One topic of particular interest in pattern recognition is the classification of data (patterns) into one of several predefined classes based on information extracted from the data. This is commonly referred to as supervised learning, and has almost infinitely many applications in technology such as optical character recognition, prediction of gene function, a variety of problems in image analysis, etc.

There are many different statistical tools to perform the classification procedure. We give a simplified presentation of some basic ideas of the statistical classification in order to relate it to what you studied in the course.
In Section 2 you have been studied different applications of Bayes formula. We started with a partition of the sample space into n excluding alternatives $A_1,\dots,A_n$, for example $A_1 =$"It is dog", $A_2 =$"It is cat" etc. Now an animal is chosen and one wishes to classify it, i.e. decide which of A_i is true. Obviously we may have "a priori" q_i^{prior} odds representing likelihood off A_i. Next suppose that one has a vector $\mathbf{x}$ of measurements taken on the object to be classified. One assumes that it is an outcome of a random experiment and we wish to include the information into the odds. This can be done using Bayes Theorem whenever the likelihood function $L(A_i) = P(\mathbf{X} \approx \mathbf{x}|A_i)$ is known. Then the posteriori odds are derived by

$$q_i^{post} = L(A_i)q_i^{prior}.$$

Finally one chooses the alternative i (classifies an object to be of type i) if the odds for i (q_i^{post}) is highest.
However the problem is that the likelihood function is unknown and hence it has to be estimated from $\mathbf{x}$ measured on objects for which we know which of the A_i's is true, called the training set. In this project you will use a classifier based on the assumption that $\mathbf{X}$ is a normally distributed random vector with the same covariance function for all alternatives A_i but different expected values.

2 Fisher's Iris data set

We will apply supervised learning to a data set called Fisher's Iris data set, which consists of four series of measurements on flowers. First download the file `irisdata.mat` and type

```
>> load irisdata.mat
```

The matrix `features` consists of four columns, corresponding to Fisher's measurement of 1) sepal length, 2) sepal width, 3) petal length, and 4) petal width, respectively, four features of Iris flowers that can be used to discriminate between species. The vector `class` represents the species of the flowers: 1) setosa, 2) versicolor, and 3) virginica. Thus we have feature vectors in $\mathbf{R}^4$ of known class membership (species).

3 Classification

The objective is now to construct an algorithm that takes feature vectors of known class membership as input, and using that, classifies data of unknown class membership into one of the three classes 1,2, or 3. In plain English this means, based on (some combination of) sepal length, sepal width, petal length, and petal width, we let the algorithm decide whether these measurements are (in some sense) most likely from a setosa, a versicolor or a virginica flower. Of course, the classification should be in as much accordance as possible to actual class labels (which we have access to, but not the algorithm).

4 Explore the data

Type the following lines:

```
>> figure(1), hold on
>> plot(features(class==1,3),features(class==1,2),'rx')
>> plot(features(class==2,3),features(class==2,2),'gx')
>> plot(features(class==3,3),features(class==3,2),'bx')
>> xlabel('Petal length');
>> ylabel('Sepal width');
>> legend('Setosa','Versicolor','Virginica')
```

You now see a scatter plot illustrating graphically the relation between the features petal length and sepal width. You are encouraged to look at more combinations of two features in this way by changing the columns of the matrix features that you plot.
Can you sense any classification rule already? Perhaps just draw straight lines in the scatter plot and classifying new feature vectors depending on which side of the lines they occur?

5 A classification rule

As you have now seen in the data, there appear to be some clustering in the data, i.e. feature vectors belonging to the same class (species) are closely grouped together. This can be used to design a classification rule. Even though it seems not entirely true we assume that the feature vectors corresponding to flowers in class i, $i \in \{1,2,3\}$, follows a (multivariate) normal distribution with probability density $f_i(\mathbf{x})$,

$$f_i(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right), \tag{1}$$

where μ_i is the mean vector for class i and Σ_i is the covariance matrix for class i. For simplicity we will assume that all classes have the same covariance matrix, and will estimate that by the pooled covariance. Then, in a maximum likelihood fashion, we assign a given feature vector $\mathbf{x}$ to the class i which maximizes $f_i(\mathbf{x})$ over all i. For example, if $f_1(\mathbf{x}) > f_2(\mathbf{x})$, it is in a sense more probable that $\mathbf{x}$ belongs to class 1 than class 2. The same principle holds for any number of classes.

6 Training data and validation data

Any supervised classifier needs data of known class membership for training, i.e. it needs to learn the relation between a data pattern and a particular class. This is known as training data. To know how well a classifier performs on previously unseen data, we need to feed it with validation data, and compare the alleged class (the response) with the actual class membership that we know, but not the classifier. Type the following lines of code:

```
>> randindex = randperm(length(class));
>> index1 = randindex(1:75);
>> index2= randindex(76:150);
>> features_training = features(index1,:);
>> features_validation = features(index2,:);
>> class_training = class(index1);
>> class_validation = class(index2);
```

Now we have, in a randomized fashion, divided the data into two equal parts, on for training and one for validation. Convince yourself that this is the case, and why `randperm` does the job.

7 Classify the data

Now, we are about to actually use the classifier described above. It is implemented in Matlab in the function `classify`, so there is no need for any advanced coding. Type

```
>> help classify
```

and read at least the first and second paragraph to learn more. Now we want to try out the performance of the classifier using all four features. Type

```
>> allegedclass = classify(features_validation,...
            features_training,class_training,'linear');
>> sum(allegedclass==class_validation)/length(allegedclass)
```

What result does the last line yield? What does this mean? Is it a good result? Now, replace both the matrices above with their second to fourth columns, leaving out the first column in both. What does this mean? What is the result now? What does this say about the importance of using sepal length as a feature? Now try different subsets of the original four features. Are the results what you expected?
Now, type

```
>> allegedclass = classify(2*features_validation,...
            2*features_training,class_training,'linear');
>> sum(allegedclass==class_validation)/length(allegedclass)
```

What does this mean? Is there any change as compared to before? Why/why not?
Now, type

```
>> features_training2 = features_training.^2;
>> features_validation2 = features_validation.^2;
>> allegedclass = classify(features_validation2(:,2:4),
            features_training2(:,2:4),class_training,'linear');
>> sum(allegedclass==class_validation)/length(allegedclass)
```

Now think through carefully what the first two commands above does. Is this result surprising? Using this new data, try some other subsets of the features and evaluate the performance.