Statistical Signal Processing
A.A. 2020/2021

Computer Lab 2 – Model fitting and classification

Duration: 9 hours
- approx. 3 hours for Exercises 1 and 2
- approx. 3 hours for Exercise 3 without optional part
- approx. 1 hour for Exercise 4
- approx. 2 hours for Exercise 5

Exercise 1 – Model fitting for continuous distributions: multivariate Gaussian

In this exercise, you will employ a real dataset (file `heightWeigth.mat`), containing labelled data for two classes, i.e. males and females. Every row of the dataset contains three numbers: the gender (1=male, 2=female), the height (cm) and the weight (kg) of each person in the dataset.

Task: you have to fit class-conditional Gaussian multivariate distributions to these data, and visualize the probability density functions. In particular, you should perform the following:

- Divide the dataset in two parts (males and females). Then work only on one gender at a time.
- Plot the data of each class (use the `scatter( )` function)
- Visualize the histogram of weight and height (use e.g. the `hist( )` function)
- Calculate the maximum likelihood estimate of the mean and covariance matrix under a multivariate Gaussian model, independently for each class (these are the parameters of the class-conditional distributions). Note: is the Gaussian model good for these data?
- Visualize the 2-D joint pdf of weight and height. Note: inside a Matlab figure, you can use the "Rotate 3D button" to change the viewpoint.

For visualizing the joint pdf, the following example code may be useful (the limits on x1 and x2 are only indicative):

```
figure
x1 = 160:1:205; x2 = 50:1:130;
[X1,X2] = meshgrid(x1,x2);
F = mvnpdf([X1(:) X2(:)],mean_males,sigma_males);
F = reshape(F,length(x2),length(x1));
surf(x1,x2,F);
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);
axis([160 205 50 130 0 max(F(:))])
xlabel('weight'); ylabel('height'); zlabel('Probability Density - males');
```

Exercise 2 - Model fitting for discrete distributions: Bag of Words

In this exercise, you will employ a real dataset (file `XwindowsDocData.mat`), containing features for two classes, i.e. documents talking about Microsoft Windows, and documents talking about X Windows. The features are the presence/absence (1/0) of a word of the vocabulary in each document. The vocabulary has D=600 words. The training set contains the features for 900 documents (`xtrain`) as well as the specific class (`ytrain`). The test set is organized in the same way, but it will not be used for this exercise.

Task: you have to fit the parameters employed by a Naïve Bayes Classifier, using a Bernoulli model. Under this model, the parameters are:
- π_c, the prior probabilities of each class.
- θ_{jc}, the probability that feature j is equal to 1 in class c.

Model fitting can be done using the pseudocode on slide 37 in Lecture 3.

Display the class-conditional densities θ_{j1} and θ_{j2}. Try to identify "uninformative" features (i.e., features j such that $\theta_{j1} \simeq \theta_{j2}$.

Exercise 3 - Classification – discrete data

In this exercise, you will design a Naïve Bayes Classifier (NBC) for the Bag of Words (BoW) features for document classification that have been prepared in Exercise 2 (there is no new dataset for this exercise). In particular, in exercise 2, you have already estimated the following parameters:
- The prior probabilities of each class, $\pi_c = p(c)$.
- The class-conditional probabilities of each feature, $\theta_{jc} = p(x_j = 1|c)$.

These parameters have been estimated from the training data. In this exercise, you will use the *test* data, and classify each test vector using an NBC whose model has been fitted in Exercise 2. In particular, you will do the following:
- For each test vector, calculate the MAP estimate of the class the test vector belongs to. Remember: the MAP classifier chooses the class that maximizes
  $\max_c \log p(y = c|\boldsymbol{x}) \propto \log p(\boldsymbol{x}|y = c) + \log p(c)$. In the NBC, the features (i.e. each entry of $\boldsymbol{x}$) are assumed to be statistically independent, so $p(\boldsymbol{x}|y = c) = \prod_{j=1}^{D} p(x_j|y = c)$. This formula allows you to calculate $p(\boldsymbol{x}|y = c)$ for a given test vector $\boldsymbol{x}$ using the parameters θ_{jc} already calculated in exercise 2. Note that, after the logarithm, the product becomes a summation. It is *much* better to use the logarithm in order to avoid underflow.
- After classifying a test vector using the NBC, the obtained class can be compared with the truth (vector `ytest`).
- The accuracy of the classifier can be computed as the percentage of times that the NBC provides the correct class for a test vector.
- Repeat the same operations using the training data as test data, and compare the accuracy of the classifier on the training and test data.
- Note: It is expected that students write the NBC *without* using Matlab's function for the Naïve Bayes Classifier.

Optional:
If you plot the class-conditional densities as done at the end of Exercise 2, you will see that many features are uninformative; e.g., words that appear very often (or very rarely) in documents belonging to either class re nit very helpful to classify a document. The NBC can performa a lot better if these uninformative features are disregarded during the classification, i.e. only a subset of the features, chosen among the most informative ones, are retained. To rank the features by "significance", one can employ the mutual information between feature x_j and class y (see Sec. 3.5.4 of the textbook):

$$I(X, Y) = \sum_{x_j} \sum_y p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

For binary features, the mutual information of feature j can be written as:

$$I_j = \sum_c \left[\theta_{jc} \pi_c \log \frac{\theta_{jc}}{\theta_j} + (1 - \theta_{jc}) \pi_c \log \frac{1 - \theta_{jc}}{1 - \theta_j} \right]$$

with $\theta_j = p(x_j = 1) = \sum_c \pi_c \theta_{jc}$. For this part, you should:
- Calculate I_j for all features. Note: try to avoid divisions by zero adding the machine precision constant `eps` to the denominators.
- Rank the features by decreasing values of I_j, and keep only the K most important ones. (You may also want to look up what words represent the most important features – see variable `vocab`).
- Run the classifier employing only the K most important features, and calculate the accuracy.
- Plot the accuracy as a function of K.

Exercise 4 – Plotting the ROC curve

For the discrete data classification problem of Exercise 3, analyse the performance of the classifier plotting the complete ROC curve, instead of simply measuring the accuracy. This requires to do the following.
- Instead of classifying the documents choosing class 1 if $p(y = 1|x) > p(y = 2|x)$, now you can generalize this to choosing class 1 if $\frac{p(y = 1|x)}{p(y = 2|x)} > \tau \in [0, \infty)$ for some threshold τ that determines the compromise between true positive rate (TPR) and false positive rate (FPR).
- Choose a **reasonable** range of values for τ. For each value of τ, compute the TPR and FPR on the dataset.
    - Hint: determine suitable minimum and maximum values for τ, and sample densely enough in that range.
- Plot a curve of TPR as a function of FPR – this is the ROC curve for this classifier.
- Determine an estimate of the Equal Error Rate (EER), i.e. the point of the ROC curve such that TPR+FPR=1.

Exercise 5 – Classification – continuous data

This exercise employs the height/weight data already employed in Exercise 1, and performs model fitting and classification using several versions of **Gaussian discriminative analysis**. However, for this exercise the available data have to be divided into two sets, namely *training* and *test* data. For example, pick around 20-25 males and 35-40 females to be used as test samples, and remove them from the training set (you would typically choose a smaller percentage of test samples, but they would be too few in this case).

You will have to 1) re-fit the training data to the specific model (see below), 2) classify each of the test samples, and 3) calculate the accuracy of each classifier.

Classifiers to be employed:
- Two-class quadratic discriminant analysis (fitting: both mean values and covariance matrices are class-specific – same as in exercise 1).
- Two-class quadratic discriminant analysis with diagonal covariance matrices (fitting: as in the previous case, but you must set to zero the off-diagonal entries of the class-specific covariance matrices).
- Two-class linear discriminant analysis (fitting: class-specific mean values as in the previous case. Shared covariance matrix is calculated putting together male and female training examples; the mean values should also be recalculated accordingly).