Open your mind. LUT.
Lappeenranta University of Technology
LUT Machine Vision and Pattern Recognition 2015-10-11

BM40A0700 Pattern Recognition

Lasse Lensu

Exercise 6: Parameter estimation and classification

1. Maximum Likelihood (ML) versus Bayesian parameter estimation (2 points): Take the provided data set. Let us assume that the data is normally distributed with covariance matrix $\mathbf{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ but with an unknown mean $\boldsymbol{\mu}$. We also know a priori distribution for the mean,

$$\boldsymbol{\mu} \sim N(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0), \qquad \boldsymbol{\mu}_0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad \mathbf{\Sigma}_0 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

The Bayesian estimate of the predictive distribution for a normally distributed data with unknown mean is also a normal distribution, with parameters

$$\boldsymbol{\mu}_B = \mathbf{\Sigma}_0 \left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma} \right)^{-1} \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k + \frac{1}{N}\mathbf{\Sigma} \left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma} \right)^{-1} \boldsymbol{\mu}_0 \qquad (1)$$

$$\mathbf{\Sigma}_B = \mathbf{\Sigma}_0 \left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma} \right)^{-1} \frac{1}{N}\mathbf{\Sigma} + \mathbf{\Sigma} \qquad (2)$$

where $\boldsymbol{\mu}_B$ and $\mathbf{\Sigma}_B$ are the parameters of the Bayesian estimate.

   (a) Estimate the density function with ML estimation.
   (b) Estimate the density function with Bayesian estimation.

   Repeat the estimations four times: i) using only the first data sample, ii) using the first ten data samples, iii) using the first one hundred samples, and iv) using all (one thousand) samples. Compare the estimates for all the four sample sizes given above. What differences can be seen?

   Hints: You can use Matlab or any other tool for the calculations, but remember to mention all the used formulas.

   Additional files: `param_est_data1.mat`.

2. Multivariate Gaussian mixture (2 points): Construct Matlab function

```
[mu, Sigma, P] = em_gmm(traindata, J)
```

   which performs expectation maximization (EM) for multivariate Gaussian mixtures. The parameters and the outputs are as follows:

   - `traindata` contains training samples,
   - `J` is the number of components in the mixture,
   - `mu` is a matrix containing a mean vector for each of the generating Gaussians,
   - `Sigma` is a three-dimensional array of covariance matrices, where the covariance matrices are along the third dimension, and

- `P` is a vector of probabilities.

The distribution to model is thus

$$\sum_{j=1}^{J} N(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) P_j$$

and the update rules for each of the unknowns are:

$$\boldsymbol{\mu}_j(t+1) = \frac{\sum_{k=1}^{N} p(j|\mathbf{x}_k; \Theta(t))\mathbf{x}_k}{\sum_{k=1}^{N} p(j|\mathbf{x}_k; \Theta(t))}$$

$$\boldsymbol{\Sigma}_j(t+1) = \frac{\sum_{k=1}^{N} p(j|\mathbf{x}_k; \Theta(t))(\mathbf{x}_k - \boldsymbol{\mu}_j(t+1))(\mathbf{x}_k - \boldsymbol{\mu}_j(t+1))^T}{l \sum_{k=1}^{N} p(j|\mathbf{x}_k; \Theta(t))}$$

$$P_j(t+1) = \frac{1}{N} \sum_{k=1}^{N} p(j|\mathbf{x}_k; \Theta(t))$$

where

$$p(j|\mathbf{x}_k; \Theta(t)) = \frac{p(\mathbf{x}_k|j; \theta(t)) P_j(t)}{p(\mathbf{x}_k; \Theta(t))}$$

$$p(\mathbf{x}_k; \Theta(t)) = \sum_{j=1}^{J} p(\mathbf{x}_k|j; \theta(t)) P_j(t)$$

Try your algorithm using the given data. Plot the data using `plot(r(:,1), r(:,2), 'x');` and try your algorithm for one, two and three components in the mixture.

Additional files: `gmmdata.mat`.

3. Statistical classifier with training data (2 points): Construct a Matlab function

```
function C = bayes3(trainclass, traindata, data)
```

that performs Bayesian classification with a Gaussian distribution model. The parameters and the output are as follows:

- Matrix `traindata` contains training examples so that each column is a single example.
- Vector `trainclass` contains the classes of the examples, so that element i of `trainclass` is the class of the example in column i of `traindata`.
- Matrix `data` contains the data to be classified.
- Vector `C` contains the classification result for each column in `data`.

You have to estimate the means, covariance matrices and a priori probabilities for each class.

Test your classifier using the same given data both for training and testing, e.g.,

```
errors = sum(bayes3(classes, data, data) ~= classes);
```

You can also present the confusion matrix.

Hints:

(a) You can use the function `find` to get indices of a certain class from the training data. For example, `classi = traindata(find(trainclass == i));` outputs the samples that belong to class i.

(b) You can assume that the a priori probabilities of each class follow the distribution of the training examples (naturally, this cannot be assumed generally).

(c) You can use Matlab functions to calculate the mean value of each class and the corresponding covariance matrix. Note that you have to consider the meaning of rows and columns in the matrices and use transpose of the matrix as needed. You may want to put the covariance matrices into a single three-dimensional array where the third dimension is the class index.

Additional files: `bayes1data.mat, confmatr.m`.