

Probability density estimation and self-organizing maps

©Signals and systems group, Uppsala University

Aug, 2003

In this laboratory work you will study some basic methods of probability density function estimation, such as, *histograms*, *kernel methods*, and *KNN-based estimation*. We will also see some examples of so-called *Gaussian mixture identification* using the *EM-algorithm* and compare this algorithm to the simple K-means clustering method to see the differences and similarities between the algorithms. Finally we will examine some simple examples of the self-organizing map (SOM). At your help you have a few matlab scripts and functions that act more or less as demos.

The demos are often interrupted and some messages indicate which the next operations will be. Take as a habit to predict the outcome of these operations. If your predictions are correct, you are likely to have understood the concepts correctly. You have freedom in choosing the way to proceed during the lab. Sometimes you will have to make slight modifications in the code in order to examine certain features of the algorithms. The main goal is to understand the methods and the ideas behind them.

No written lab report is required. However, you should be prepared to answer some questions during, or at the end of the lab exercise (or, if time is too short, during one of the following lab occasions).

1 Nonparametric probability density function (PDF) estimation

1.1 1-d examples

We start with a comparison between non-parametric PDF estimation methods. At your help you have three scripts `histill1` (for histogram illustration), `kernill1` (for kernel method illustration) and `knnill1` (for K-nearest neighbor (KNN) illustration). The data used comes from an ordinary 1-d normal zero mean, unit variance density.

In the histogram illustration you can modify the number of samples, N , and the box-width, u , by means of sliders in a graphical user interface. Remember to try to predict the outcome of your parameter settings.

In the kernel estimate illustration you can again choose N and a parameter u that here is the width of the kernel. The kernel can be set to “uniform” (a box) or “Gaussian”.

In the KNN illustration you can modify N and the parameter K .

1.2 Histogram PDF estimation in 2-d

For 2-d data we only consider histogram PDF estimation. The data is drawn from a 2-d normal distribution. The script to use is `histill2`. You can change the covariance matrix (line 15 in `histill2`) to vary the distribution. Try to generalize your conclusions

obtained by running this script to kernel based methods and KNN as well as to data spaces of dimension larger than 2.

2 Semi-parametric density estimation

2.1 A short review of Gaussian mixtures and the EM-algorithm

The probability density of a general M -component Gaussian mixture model can be written as

$$p(\mathbf{x}) = \sum_{k=1}^M p(\mathbf{x}|\mathbf{m}_k, \Sigma_k) P_k, \quad (1)$$

where the parameters P_1, P_2, \dots, P_M are the mixing probabilities, which must sum to one, and the density $p(\mathbf{x}|\mathbf{m}_k, \Sigma_k)$ is given by

$$p(\mathbf{x}|\mathbf{m}_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{d/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{m}_k)\right) \quad (2)$$

where $d = \dim(\mathbf{x})$, \mathbf{m}_k and Σ_k are the mean vector and the covariance matrix of the k th component, respectively.

The simplified model that we will use in this lab consists of a smaller number of free parameters. Instead of a number of general $d \times d$, covariance matrices, Σ_k , we use $\Sigma_k = \sigma_k^2 \mathbf{I}$, i.e., we assume ‘‘circular’’ distributions.

The EM-algorithm for estimating the parameters in such a model, using training data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is as follows:

1. Choose initial parameter values, $\hat{\mathbf{m}}_1^{old}, \dots, \hat{\mathbf{m}}_M^{old}, (\hat{\sigma}_1^{old})^2, \dots, (\hat{\sigma}_M^{old})^2, \hat{P}_1^{old}, \dots, \hat{P}_M^{old}$.
2. Calculate the posterior probabilities: $P^{old}(k|\mathbf{x}_n)$ for each pattern \mathbf{x}_n and each subclass $k = 1 \dots M$. This probability is the probability that \mathbf{x}_n was generated by component k , given that the old parameter values are correct. These posterior probabilities are obtained using Bayes theorem.
3. Update the mean vectors as the weighted average

$$\hat{\mathbf{m}}_k^{new} = \frac{\sum_n P^{old}(k|\mathbf{x}_n) \mathbf{x}_n}{\sum_n P^{old}(k|\mathbf{x}_n)}. \quad (3)$$

4. Update the variances as the weighted average

$$(\hat{\sigma}_k^{new})^2 = \frac{1}{d} \frac{\sum_n P^{old}(k|\mathbf{x}_n) \|\mathbf{x}_n - \hat{\mathbf{m}}_k^{new}\|^2}{\sum_n P^{old}(k|\mathbf{x}_n)}. \quad (4)$$

5. Calculate new mixing probabilities according to

$$\hat{P}_k^{new} = \frac{1}{N} \sum_n P^{old}(k|\mathbf{x}_n) \quad (5)$$

2.2 2-d Illustration of the EM-algorithm for identifying Gaussian mixture parameters

In the script `em11`, the algorithm runs on data from a two-component Gaussian mixture with known parameters. You can change the parameters in the m-code to find out which parameter settings that are easy or difficult to identify.

2.3 Comparison between KMC and mixture identification using EM-algorithm

The script `compkmc2em`, is similar to `emill`. The difference is that KMC runs on the same data and we can see how the two algorithms differ. To see different aspects of the problem, please modify the σ^2 -values in the code (lines 16-17). In particular, under what circumstances do KMC and EM yield very different solutions?

2.4 Illustration of mixture density approximation of non-Gaussian distribution

Run `circem1` a number of times to see how a mixture density can be fitted to a non-Gaussian PDF. In this example we consider 2-d data on a circle with a hole. Try to understand why the algorithm sometimes does not always find good solutions. (You may see some matlab-warnings which is an indication of something going wrong).

Also, run `circem2` a number of times. Here the initialization of the mean vectors has been improved. Can you come up with more suggestions for improvements?

3 Kohonens self-organizing map (SOM)

3.1 2-d data

Run `somex2d` to see a classical illustration of how the prototypes vectors corresponding to the nodes in a SOM evolve for a 2-d uniformly distributed training data set.

3.2 3-d data

In `somex3d` the data is generated from a cylindrical surface (points in 3d that essentially lie in a 2d subspace).

3.3 17-d data

Finally, in `somex17d` we examine if a SOM can identify a “natural coordinate system” for a problem where we have 17-dimensional data vectors that are determined by only two parameters, a (envelope width) and ω (frequency).

Each data vector, $\mathbf{x} = (x_{-8}, \dots, x_8)^T$ is simulated according to the equation

$$x_t = \exp(-at^2) \cos(2\pi\omega t) \quad (6)$$

with a and ω drawn uniformly from the intervals $[0, 1/4]$ and $[0, 2\pi/4]$, respectively.

After training you can present a number of vector to the SOM and examine how the $[a, \omega]$ -coordinate system is related to the coordinate system in the SOM.