

**NGSSC Course in  
Data mining and applications in  
science and technology  
Lecture Notes: Clustering 3**  
June 3-12, 2002 Linköping University

Timo Koski  
Department of mathematics  
LiTH

This part of the lectures deals with various select topics some of which can be described as more 'recent' ideas in clustering/classification. Most of the topics are treated very superficially, mainly due to lack of time.

## Clustering 3: Various topics

### OUTLINE OF CONTENTS:

- CEM-clustering, CML
- Schnell's method of unsupervised clustering
- LVQ
- MCMC approach to classifying, Bayes Networks
- Support Vector Machines
- Overfitting, Generalization, Validation, Model choice

## A Method of Unsupervised Classification: The CEM-algorithm for Gaussian Mixture Models (1)

Suppose that we have performed iteration at step  $r$  and have thus obtained initial estimates  $\mu_j^{(r)}$  and  $\Sigma_j^{(r)}$  and proceed to step  $r + 1$ . The E- and M-steps are as before, we recall the E-step:

E-step The posterior probabilities  $p^{(r+1)}(j|\mathbf{x}^{(l)})$  of each cluster/class for the data point  $\mathbf{x}^{(l)}$  are computed as

$$\begin{aligned} & p^{(r+1)}(j|\mathbf{x}^{(l)}) \\ &= \frac{w_j^{(r)} p(\mathbf{x}^{(l)} | u_j = 1; \mu_j^{(r)}, \Sigma_j^{(r)})}{\sum_{j=1}^k w_j^{(r)} p(\mathbf{x}^{(l)} | u_j = 1; \mu_j^{(r)}, \Sigma_j^{(r)})} \end{aligned}$$

## The CEM-algorithm for Gaussian Mixture Models (2): Classification

C-step After the E and M-steps have converged, we let

$$\mathbf{P}^* = [p^{(*)}(j|\mathbf{x}^{(l)})]_{j=1}^k_{l=1}^t.$$

We classify  $\mathbf{x}^{(l)}$  to  $c_{j_*}$  if

$$j_* = \arg \max_{1 \leq j \leq k} p^*(j|\mathbf{x}^{(l)})$$

for  $l = 1, \dots, t$ .

Here we note a difference to the learning methods of supervised learning outlined in a previous lecture: all of the data is used to estimate all of the statistical parameters.

A Method of Unsupervised Classification:  
Classification Maximum Likelihood  
(CML) (1)

Let us again consider the class-conditional Gaussian densities for  $\mathcal{X}^t = \{\mathbf{x}^{(l)}\}_{l=1}^t$ . A loglikelihood function is defined as

$$l(\mu, \Sigma, \Lambda, U)$$
$$= \sum_{j=1}^k \sum_{l=1}^t u_j^{(l)} \left[ \log \lambda_j p(\mathbf{x}^{(l)} \mid u_j = 1; \mu_j, \Sigma_j) \right],$$

where

$$u_j^{(l)} := \begin{cases} 1 & \text{if } x^{(l)} \in c_j \\ 0 & \text{otherwise,} \end{cases}$$

## Classification Maximum Likelihood (CML) (2)

We estimate the statistical parameters and the clusters by maximum likelihood:

$$\left(\hat{\mu}, \hat{\Sigma}, \hat{\Lambda}, \hat{U}\right) = \arg \max_{\mu, \Sigma, \Lambda, U} l(\mu, \Sigma, \Lambda, U).$$

Algorithmically this can be done using an alternating algorithm as follows:

## Classification Maximum Likelihood (CML) (3)

1. Assign randomly  $k$  clusters for  $\mathcal{X}^t$ .
2. Find  $\widehat{\Sigma}_j$ ,  $\widehat{\mu}_j$ ,  $\widehat{\Lambda}$  using maximum likelihood with the data in each cluster.
3. Calculate new class memberships using the plug-in discriminant functions.

## Classification Maximum Likelihood (CML) (4)

4. Check if the cluster memberships have changed. If yes, start again from the step 2. If not, clustering is finished and all  $\mathbf{x}^{(l)}$  have their cluster memberships defined by CML.

## Schnell's method of unsupervised clustering (1)

Schnell's method (to be presented) is not a recent idea, the original paper was published in 1964\*. We choose to present the method, since it is quite different from the methods presented so far and does not seem to have gained much attention in the literature or in the textbooks or in practice (?).

\*For this reference and the results below we refer to J. Grim (1981): An algorithm for maximizing a finite sum of positive functions and its applications to cluster analysis. *Problems of Control and Information Theory*, 10, pp. 427–437.

## Schnell's method of unsupervised clustering (2)

The data matrix is

$$\mathcal{X}^t = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}\}.$$

We introduce a recursive algorithm

$$\mathbf{z}_{s+1} = \sum_{l=1}^t p(l | \mathbf{z}_s) \mathbf{x}^{(l)}$$

which can be proved to possess a finite number ( $= k$ ) of points of convergence denoted by

$$\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}.$$

## Schnell's method of unsupervised clustering (3)

In the recursion  $\mathbf{z}_{s+1} = \sum_{l=1}^t p(l | \mathbf{z}_s) \mathbf{x}^{(l)}$  we take

$$p(l | \mathbf{z}_s) = \frac{f_l(\mathbf{z}_s)}{f(\mathbf{z}_s)},$$

and

$$f_l(\mathbf{z}_s) = \frac{1}{(2\pi)^{d/2} \sqrt{\det h \cdot \Sigma}} e^{-\frac{1}{2} D_{h \cdot \Sigma}(\mathbf{z}_s, \mathbf{x}^{(l)})^2},$$

where  $h > 0$  and

$$f(\mathbf{z}_s) = \frac{1}{t} \sum_{l=1}^t f_l(\mathbf{z}_s)$$

We can think of  $\Sigma = I_d$ .

## Schnell's method of unsupervised clustering (4)

$$\mathbf{z}_{s+1} = \sum_{l=1}^t p(l | \mathbf{z}_s) \mathbf{x}^{(l)}$$

has as its limit points the vectors

$$\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}.$$

Then the method of Schnell is to cluster the data matrix into  $k$  clusters  $c_j$  defined by

$$c_j = \left\{ \mathbf{x}^{(l)} \mid \lim_{s \rightarrow \infty} \mathbf{z}_s = \mathbf{a}_j, \mathbf{z}_0 = \mathbf{x}^{(l)} \right\}.$$

In words this means that the algorithm takes each  $\mathbf{x}^{(l)}$  as its initial value and then  $\mathbf{x}^{(l)}$  is clustered to  $c_j$  if the algorithm converges to  $\mathbf{a}_j$ .

## Schnell's method of unsupervised clustering (5)

$$\mathbf{z}_{s+1} = \sum_{l=1}^t p(l | \mathbf{z}_s) \mathbf{x}^{(l)}$$

$$c_j = \left\{ \mathbf{x}^{(l)} \mid \lim_{s \rightarrow \infty} \mathbf{z}_s = \mathbf{a}_j, \mathbf{z}_0 = \mathbf{x}^{(l)} \right\}.$$

The points  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$  are local minima of an unknown density  $p(\mathbf{x})$  generating the data.

We approximate  $p(\mathbf{x})$  by  $f(\mathbf{x})$  which is a sum of Gaussian kernels.

Does Schnell's method produce reasonable results ??? (I do not know 'i skrivande stund')

## Learning Vector Quantization (LVQ)

We have a training set

$$\mathcal{X}^t = \left\{ \mathbf{x}^{(l)} \right\}_{l=1}^t, U = \left\{ u_j^{(l)} \right\}_{j=1, l=1}^{k, t}.$$

We have also a set of reference vectors,  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$  for each of the classes, respectively.

LVQ is a kind of on-line version of  $k$ -means clustering.

LVQ gained popularity during the 1980's, but a first version of an algorithm like this was given by

McQueen, J.B.(1967): Some methods of classification and analysis of multivariate observations, *Proc. 5th Berkeley Symposium in Mathematics, Statistics and Probability*, vol 1., pp. 281-296, Univ. of California, Berkeley

1. Assign each  $\mathbf{x}^{(l)}$  to the closest  $\mathbf{a}_c$  using, e.g., Euclidean distance.

2. • If  $\mathbf{x}^{(l)}$  is correctly classified, i.e., its label in  $U$  is the one corresponding to  $\mathbf{a}_c$  Update  $\mathbf{a}_c(r)$  to form  $\mathbf{a}_c(r+1)$  using  $\mathbf{x}^{(l)}$  as follows

$$\mathbf{a}_c(r+1) = \mathbf{a}_c(r) + \alpha(k) [\mathbf{x}^{(l)} - \mathbf{a}_c(r)]$$

• If  $\mathbf{x}^{(l)}$  is incorrectly classified, then update  $\mathbf{a}_c(r)$  as follows

$$\mathbf{a}_c(r+1) = \mathbf{a}_c(r) - \alpha(k) [\mathbf{x}^{(l)} - \mathbf{a}_c(r)]$$

•  $\mathbf{a}_i(r+1) = \mathbf{a}_i(r)$  for  $i \neq c$ .

3. Repeat step 2, and let  $\alpha(k)$  decrease at each step toward zero.

## MCMC for Classification

Assume that we have a data matrix  $\mathcal{X}^t$  and want a classification, which makes the components independent. Define a class assignment randomly, and compute the probability of data, given the model with independent attributes. The MCMC (=Markov Chain Monte Carlo) will now implement a *move function* proposing a changed class in some case. The move is accepted if the posterior probability increases, or otherwise by a probability given by the ratio of new to old data probability.

A reference for MCMC:

W.R. Gilks, S. Richardson and D.J. Spiegelhalter (1997): Markov Chain Monte Carlo in Practice, Chapman and Hall.

## Probabilistic networks a.k.a Graphical Models (1)

Probabilistic graphical models are graphs in which nodes represent random variables, and the (lack of) arcs represent conditional independence assumptions. Hence they provide a compact representation of joint probability distributions.

Directed graphical models also called Bayesian Networks or Belief Networks (BNs), have a more complicated notion of independence, which takes into account the directionality of the arcs.

A well-written tutorial on the subject:

D. Heckerman (1997): Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery*, 1, pp. 79–119.

According to Heckerman (loc.cit) Bayesian Networks is a technique of modelling multivariate distributions that has the following advantages to offer (when compared to clustering, classification, neural networks, decision trees etc.). :

- Bayesian Networks can readily handle incomplete data sets
- Bayesian Networks allow one to learn causal relationships, which is important for predictive modelling.
- Bayesian Networks facilitate combination of domain knowledge with data using statistical learning techniques
- Bayesian Networks can be trained avoiding **overfitting**

## Probabilistic networks a.k.a Graphical Models (2) : DAG

Let  $\mathcal{G} = (V, E)$  be a directed acyclic graph (DAG) with the set of nodes  $V = \{1, \dots, d\}$  and the edges  $E$ . For each node there is a discrete random variable  $X_i$  the instantiations of which are designated by  $x_i$ . Each edge  $(j, i)$  in  $E$  is a statement telling that  $X_j$  is influencing  $X_i$ . The absence of an edge indicates lack of direct influence.

Since the graph is acyclic and directed, the *structure* of the graph  $\mathcal{G}$  is given by the parent sets

$$\mathbf{\Pi} = (\Pi[1], \Pi[2], \dots, \Pi[d]),$$

where  $\Pi[j]$  is the set of parents of the node  $j$ .

We assume that the joint distribution of  $(X_1, \dots, X_d)$  is factorized along  $\mathcal{G}$  in the sense that

$$P(X_1 = x_{i_1}, \dots, X_d = x_{i_d}) = \prod_{j=1}^d P(X_j = x_{j_1} | \pi(j)),$$

where  $\pi(j)$  denotes the *parent configuration*. The parent configuration is the set of instantiations assumed by the variables in the parent nodes

$$\pi(j) = \times_{l \in \Pi[j]} \{X_l = x_{i_l}\} \in \mathcal{X}_j,$$

where  $\mathcal{X}_j$  is the notation for the space of all possible parent configurations at node  $j$ . In addition to the structure we need to specify for each node the table of conditional probabilities  $P(X_j = x_j | \pi(j))$  in order to define a Bayesian network.

## Factorizing a multivariate probability along a DAG

In the graph depicted in the figure below  
we have

$$P(x_1, x_2, x_3, x_4) = \\ P(x_4|x_3) P(x_3|x_2, x_1) P(x_2|x_1) P(x_1)$$

## Regression for DAG

Suppose that continuous variable  $\gamma$  with parent nodes  $\pi(\gamma)$  has the distribution

$$X_\gamma | \pi(\gamma) \in N(\alpha + \beta^T \underline{z}, \sigma).$$

Here  $\pi(\gamma)$  stands for the combination of discrete  $i$  and continuous variables  $\underline{z}$  in the parent set of  $\gamma$ . Here  $\alpha$  is a number,  $\sigma > 0$  and  $\beta$  is a vector with dimension equal to the dimension of the continuous component  $\underline{z}$  so that  $\beta^T \underline{z}$  is an inner product. Thus the conditional density is equal to

$$\phi(i, z, x_\gamma) = \frac{1}{\sqrt{2\pi\sigma(i)}} e^{-\frac{(x_\gamma - (\alpha + \beta^T \underline{z}))^2}{2\sigma}}$$

## SUPPORT VECTOR MACHINES

- a technique of supervised learning, not a model based method in the same sense as the preceding ones
- also called a kernel method
- is based on a well-developed mathematical theory due to N. Vapnik\*.
- uses the idea of an optimal separating hyperplane

\*surveyed by the originator of SVM himself in N.V. Vapnik (1995): The Nature of Statistical Learning Theory, Springer Verlag, New York

## LINEAR SEPARABILITY

There are two classes, whose members are given the labels  $\times$  and  $\circ$ , respectively. These two classes are called **separable** (by a hyperplane), since there is hyperplane  $w^T x + w_0 = 0$  so that the classes are wholly on the positive or negative side, respectively, of the hyperplane.

## A LINEARLY NONSEPARABLE SITUATION

There are two classes, whose members are given the labels  $\times$  and  $\circ$ , respectively. These classes cannot be separated by a hyperplane without a considerable number of errors.

## SUPPORT VECTOR MACHINES

The support vector machine maps the vectors  $x$  into a higher-dimensional feature space by an a priori chosen map. In this higher-dimensional space an optimal linear hyperplane is constructed.

A separating hyperplane  $\mathbf{w}^T \mathbf{x} + w_o = 0$  is optimal, if it has maximal distance to the closest vector. If  $y^{(l)} \in \{-1, 1\}$  for  $l = 1, \dots, t$  and the separating hyperplane can be written as

$$y^{(l)} [\mathbf{w}^T \mathbf{x}^{(l)} + w_o] \geq 1, \quad l = 1, \dots, t.$$

The optimal hyperplane satisfies these inequalities and minimizes the function

$$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2$$

In a nonseparable case one introduces slack variables  $\xi^{(l)}$  that allow data to cross boundaries such that

$$y^{(l)} [\mathbf{w}^T \mathbf{x}^{(l)} + w_0] \geq 1 - \xi^{(l)} \quad (*)$$

The optimal nonseparable hyperplane that minimizes training errors minimizes the function

$$\Phi(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{l=1}^t \xi^{(l)}$$

under the constraints (\*). By some mathematics it is found that (next slide please):

The previous optimization problem is equivalent to maximizing

$$W(\alpha, C) = \sum_{l=1}^t \alpha_l - \frac{1}{2} \sum_{l=1, j}^t \alpha_l \alpha_j y^{(l)} y^{(j)} \mathbf{x}^{(j)T} \mathbf{x}^{(l)}$$

subject to the constraints

$$(C - \alpha_l) y^{(l)} = 0, 0 \leq \alpha_l \leq C.$$

$$\alpha_l \left[ y^{(l)} \left[ \mathbf{w}^T \mathbf{x}^{(l)} + w_0 \right] - 1 + \xi^{(l)} \right] \geq 0$$

Only some of the coefficients  $\alpha_l$  differ from zero. They determine the *support vectors*.



In the preceding picture we note the following:

- When the data lie strictly outside the margin boundaries,  $\alpha_i = 0$ . These data points are not included in the support vectors.
- When the data lie on the margin boundary, we have  $0 \leq \alpha_i \leq C$ . These are considered as support vectors, but their maximum contribution is bounded by  $C$ .
- For the data points that lie inside the boundary, and on the wrong side of separating hyperplane, we have  $\alpha_i = C$ . All such points are support vectors and they contribute a maximum weight of  $C$  no matter how far they are from the boundary.

## SUPPORT VECTOR MACHINES & KERNEL METHOD

The crucial idea is that the larger feature space is dealt with as follows: Find the maximum of

$$W(\alpha, C) = \sum_{l=1}^t \alpha_l - \frac{1}{2} \sum_{l,j=1}^t \alpha_l \alpha_j y^{(l)} y^{(j)} K(\mathbf{x}^{(j)}, \mathbf{x}^{(l)})$$

under the same constraints as previously. Here  $K(\mathbf{x}^{(j)}, \mathbf{x}^{(l)})$  is a kernel (a symmetric, non-negative definite function), it can be, e.g, the Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}^{(l)}) = e^{-\gamma \|\mathbf{x} - \mathbf{x}^{(l)}\|^2}.$$

This SVM is more or less similar to a *radial basis function neural network*. The optimal hyperplane will be

$$\sum_{l=1}^N \alpha_l y^{(l)} \left( e^{-\gamma \|\mathbf{x} - \mathbf{x}^{(l)}\|^2} - w_o \right) = 0$$

where  $N$  is the number of supporting vectors.

## How do SVM:s work (1)

Setting:

- training examples  $\mathbf{x}^{(l)}, y^{(l)}, l = 1, \dots, t$ ,  
 $y^{(l)} \in \{-1, 1\}$ .
- A larger feature space generated by the kernel  $K(\mathbf{x}^{(j)}, \mathbf{x}^{(l)})$ .
- Parameter  $C$  for trade-off between training error and margin size.

## How do SVM:s work (2)

Training:

- Finds a hyperplane in feature space generated by the kernel  $K(\mathbf{x}^{(j)}, \mathbf{x}^{(l)})$ .
- The hyperplane has maximum margin in feature space with minimal training error given  $C$ .
- The result of training are  $\alpha_1, \alpha_2, \dots, \alpha_t$ , they determine the  $\mathbf{w}$  and  $w_o$ .

Classification for a new item: Either side of the hyperplane

$$\sum_{l=1}^N \alpha_l y^{(l)} K(\mathbf{x}, \mathbf{x}^{(l)}) - w_o = 0$$

## SUPPORT VECTOR MACHINES & DATA MINING

E.G., Technical reports from the Data Mining Institute, Computer Sciences Department, University of Wisconsin-Madison:

<http://www.cs.wisc.edu/dmi/>

## SUPPORT VECTOR MACHINES: generalization

If the model is not overfitted, it will be able to generalize. It can be proved that SVM:s can generalize since the empirical training error will converge in probability.

## STATISTICS AND DATA MINING

- Algorithms plays a much more central role in data mining.
- There is little of *inference* in data mining, statistical learning is more important.
- In data mining modelling is predictive: we search for models with good predictive power, even if one does not understand why the predictive power is good.

D.J. Hand (1999): Statistics and Data Mining: Intersecting Disciplines. *SIGKDD Explorations*, vol. 1, pp. 16–19.