T-61.5040
Learning Models and Methods, (5 cr) L

Spring 2007

Lecturer: **Petteri Pajunen**
Course Assistant: **Ville Viitaniemi**

LECTURE 10: 28.3.2007

LATENT VARIABLE MODELS

Latent Variable Models

- we have used regression as an example of a learning problem
- data includes both inputs x and corresponding outputs y
- if some or all inputs x are *unobserved*, then we end up with a latent variable model

- a toy example: learning a bit-to-bit function
- observe input $x(n) \in \{0,1\}$ and output $y(n) \in \{0,1\}$
- suppose you have $x(0) = 0, y(0) = 0$ and $x(1) = 1, y(1) = 1$
- what's $p(y(2)|x(2))$?

- it seems that the situation is symmetric wrt 0 and 1
- but if there is a latent input $z(n)$ so that

$$p(y(n) = 1 | x(n) = 1 \text{ or } z(n) = 1) = 1$$
$$p(y(n) = 0 | x(n) = 0 \text{ and } z(n) = 0) = 1$$

- then $y(2) = 1$ is more probable than $y(2) = 0$
- point: existence of the input $z(n)$ affects the prediction *even if* $z(n)$ is never observed

- some ways of coming up with a latent variable model:
  - such a model is realistic in the problem (e.g. ICA, image deblurring or superresolution)
  - model does not fit the data: perhaps some unobserved input causes this
  - adding latent variables helps solving the model (mixture models)

- in neurocomputing, unsupervised learning methods are latent variable models
- clustering can be thought of as a latent variable model
- the latent variable would define the cluster index for each observation
- PCA explains the outputs as linear combinations of unobserved principal components

- in the computer assignment the observed data is the output and the high-resolution image the unobserved input
- the pixel intensities of the high-res image are latent variables
- qualitatively speaking, latent variables are "data-like" parameters
- it is natural to think of pixel intensities as data instead of parameters

- unsupervised learning methods in general are taught in other courses in detail
- lets concentrate on *mixture models*, especially Normal Mixtures
- benefits: many latent variable models can be thought of as mixture models, computational benefits

Normal Mixtures

- consider data $y = \{y_1, \ldots, y_n\}$ where each y_i has a *multimodal distribution* with m modes
- lets attempt to model the data using m different Normal distributions $N(\mu_j, \sigma_j^2)$
- we'll construct the mixture model by assuming that each y_i has been generated by a specific *mixture component* $N(\mu_j, \sigma_j^2)$
- y_i is not Normally distributed, because *we don't know which component has generated it*

- what is the distribution of y_i?
- lets add *latent variables*, which indicate the guilty distribution
- *indicators* $L_{ij} \in \{0, 1\}$:

$$L_{ij} = 1 \text{ means that } y_i \text{ belongs to component } j$$

$$\sum_{j=1}^{m} L_{ij} = 1, \text{ each data point belongs to exactly one component}$$

- lets define λ_j as the probability that $L_{ij} = 1$ *when* y_i *is unknown*
- a benefit of the indicators is that they simplify the model
- the *complete-data likelihood* is (exercise)

$$p(y_i, L_{i\cdot} | \theta, \lambda) = \prod_{j=1}^{m} (\lambda_j N(y_i | \mu_j, \sigma_j^2))^{L_{ij}}$$

- integrate out the latent variables so that $L_{ik} = 1$ when $k = 1$:

$$
\begin{aligned}
p(y_i|\theta,\lambda) &= \sum_{k=1}^{m}\prod_{j=1}^{m}(\lambda_j N(y_i|\mu_j,\sigma_j^2))^{L_{ij}} \\
&= \sum_{k=1}^{m}\lambda_k N(y_i|\mu_k,\sigma_k^2)
\end{aligned}
$$

- observed data has distribution

$$
p(y|\theta,\lambda) = \prod_{i=1}^{n}\sum_{j=1}^{m}\lambda_j N(y_i|\mu_j,\sigma_j^2)
$$

- when the product is expanded, the result is a sum with m^n terms
- this cannot be even maximized in closed form

- full conditional posteriors are simpler (constant prior for θ):

$$
p(\theta|y,L) \propto \prod_j \prod_{i:L_{ij}=1} N(y_i|\mu_j,\sigma_j^2) \text{ (exercise)}
$$

$$
p(L_{ik}=1|\theta,y,\lambda) = \frac{\lambda_k N(y_i|\mu_k,\sigma_k^2)}{\sum_j \lambda_j N(y_i|\mu_j,\sigma_j^2)}
$$

- the probabilities λ_j are also unknown: the can be easily estimated given the other parameters (exercise)

- with enough mixture components, any distribution can be closely approximated
- θ and λ cannot be solved in closed-form
- Gibbs Sampler is possible due to full conditional posteriors
- but it has poor convergence properties

- example: simulate a two-component mixture model

$$p(y_i|\theta,\lambda) = \lambda_1 N(y_i|\mu_1,1) + \lambda_2 N(y_i|\mu_2,1)$$

using Gibbs Sampler

- demo shows that there are serious convergence problems
- demo: `gibbsmixture.R`

- the posterior mode can be approximated using Expectation Maximization
- first, lets motivate it by deriving something simpler
- consider the model

$$p(y|\theta) = \prod_{i=1}^{n} p(y_i|\theta) = \prod_{i=1}^{n} \left[\lambda_1 N(y_i|\mu_1,\sigma^2) + \lambda_2 N(y_i|\mu_2,\sigma^2)\right]$$

where σ^2 is known

- find μ_1 and μ_2 that maximize the log-likelihood

$$\log p(y|\theta,\lambda) = \sum_{i=1}^{n} \log\left(\lambda_1 N(y_i|\mu_1,\sigma^2) + \lambda_2 N(y_i|\mu_2,\sigma^2)\right)$$

- differentiate it (denoted by operator d):

$$\begin{aligned} d\log p(y|\theta,\lambda) &= \sum_{i=1}^{n} d\log p(y_i|\theta,\lambda) \\ &= \sum_{i=1}^{n} [p(y_i|\theta,\lambda)]^{-1} dp(y_i|\theta,\lambda) \end{aligned}$$

- to differentiate wrt μ_j, note that

$$\frac{\partial p(y_i|\theta,\lambda)}{\partial \mu_j} = \lambda_j N(y_i|\mu_j,\sigma^2)\sigma^{-2}(y_i-\mu_j)$$

- then we obtain as the derivative

$$\begin{aligned} \frac{\partial \log p(y|\theta)}{\partial \mu_j} &= \sum_{i=1}^{n} \sigma^{-2}(y_i-\mu_j)\tau_{ij} \\ \tau_{ij} &= \frac{\lambda_j N(y_i|\mu_j,\sigma^2)}{\sum_k \lambda_k N(y_i|\mu_k,\sigma^2)} \end{aligned}$$

- weight τ_{ij} is close to one when y_i is much closer to μ_j than any μ_k with $k \neq j$
- also, $\tau_{ij} = p(L_{ij} = 1|\theta, y, \lambda)$
- can easily be computed *if we know* $\mu_1, \mu_2, \lambda_1, \lambda_2$

- derivative

$$\frac{\partial \log p}{\partial \mu_j} = \sum_{i=1}^{n} \frac{y_i - \mu_j}{\sigma^2} \tau_{ij}$$

  is a weighted average of $(y - \mu_j)/\sigma^2$
- e.g. if y is on average larger than μ_j, the derivative will be positive
- the average is easy to compute *if all τ_{ij}:s are known*

- lets maximize $\log p(y|\theta)$ using *Newton-Rhapson iteration*

$$\mu_{new} = \mu_{old} - (\log p)' / (\log p)''$$

- first derivative was just computed
- the second derivative is approximately computed in the exercises

- start by selecting initial values for the means and λ:s
- then compute τ_{ij}:s
- then compute the derivatives and use Newton-Rhapson

$$\mu_{j,new} = \frac{\sum_i p(L_{ij} = 1|\theta, y_i) y_i}{\sum_i p(L_{ij} = 1|\theta, y_i)} = \frac{\sum_i \tau_{ij} y_i}{\sum_i \tau_{ij}}$$

- $\mu_{j,new}$ is an a weighted average of data
- the weights τ_{ij} emphasise some observations more than others
- for data not from component j, $\tau_{ij} \approx 0$
- average is mostly over data from component j
- demo: `twonormal.R`

Expectation-Maximization Algorithm

- the previous derivation suggests that the posterior mode is a computationally feasible approximation
- there is a general algorithm for finding the posterior mode for latent variable models
- it is called the *Expectation-Maximization* (EM) algorithm for reasons seen later

- the finite mixture model motivates the EM algorithm
- we saw that it is easy to deal with distributions

$$
\begin{aligned}
&p(L|\theta, y) \text{ (discrete distribution)} \\
&p(\theta|L, y) \propto \underset{\text{(Normal)}}{p(y|L,\theta)p(\theta|L)}
\end{aligned}
$$

- but it is difficult to deal with $p(\theta|y)$ directly: this is a product of a possibly multimodal or otherwise difficult likelihood and a prior

- we want to find θ that maximizes $p(\theta|y)$
- Jensen's inequality $\log \mathrm{E}(x) \geq \mathrm{E}(\log x)$ gives

$$
\begin{aligned}
\log p(\theta|y) &= \log \int p(\theta, L|y) dL \\
&= \log \int q(L) \frac{p(\theta, L|y)}{q(L)} dL \\
&\geq \int q(L) \log \frac{p(\theta, L|y)}{q(L)} dL = -D(q\|p) \\
&= F(q, \theta)
\end{aligned}
$$

- $F(q, \theta)$ bounds $\log p(\theta|y)$ from below
- it can be written as

$$F(q, \theta) = E_q(\log p(\theta, L|y)) - E_q(\log q(L))$$

- the EM-algorithm can be written as follows:

E-step maximize $F(q_n, \theta_{n-1})$ by choosing a maximizing distribution q_n given θ_{n-1}

M-step maximize $F(q_n, \theta_n)$ by choosing a maximizing θ_n

iterate increase n and go to E-step

E-step

- maximize $F(q_n, \theta_{n-1}) = -D(q_n \| p)$
- due to properties of KL divergence, $q_n(L) = p(\theta_{n-1}, L|y)$ maximizes $F(q_n, \theta_{n-1})$
- this step corresponds to the computation of τ_{ij}:s
- $q(L)$ allows us to compute

$$\mathrm{E}_q(\log p(\theta, L|y)) = \int \log p(\theta, L|y) q(L) dL,$$

M-step

- find θ_n which maximizes $F(q_n, \theta_n) = \mathrm{E}_q(\log p) - \mathrm{E}_q(\log q_n)$
- $\mathrm{E}_q(\log q(L))$ remains constant, so maximize

$$\mathrm{E}_q(\log p(\theta, L|y)) = \int \log p(\theta, L|y) q(L) dL,$$

- this corresponds to setting $\mu_{j,new} = \sum_i \tau_{ij} y_i / \sum_i \tau_{ij}$

- maximizing F makes intuitive sense since it is a lower bound to $\log p(\theta|y)$
- two problems: maximization of F is an iteration, and F is only a lower bound
- in general, global maximum of $p(\theta|y)$ is not found but each iteration of EM is guaranteed not to decrease the posterior (exercise)

Example: EM for Normal Mixture Model

- earlier example illustrates the results below
- the E-step gives the distribution $q(L)$ as

$$\tau_{ij} = p(L_{ij} = 1|\theta', y_i) = \frac{\lambda_j N(y_i|\mu'_j, \Sigma'_j)}{\sum_k \lambda_k N(y_i|\mu'_k, \Sigma'_k)}$$

- it tells the posterior probability that data y_i comes from mixture component j, given *old values* of parameters θ' (prime refers to old values)

- M-step gives the update equations

$$\lambda_j = \frac{1}{N}\sum_i \tau_{ij} \quad \text{(mixture proportions)}$$

$$\mu_j = \frac{\sum_i y_i \tau_{ij}}{\sum_i \tau_{ij}} \quad \text{(mean values)}$$

$$\Sigma_j = \frac{\sum_i \tau_{ij}(y_i - \mu_j)(y_i - \mu_j)^T}{\sum_i \tau_{ij}} \quad \text{(covariance matrices)}$$

- note that the values μ_j used in updating Σ_j are the *new values*, i.e. those that were just updated
- in the following demo, three clusters of 2D data are generated
- a Normal mixture model is solved using the EM-algorithm
- demo: `emnorm.R`

Variational Approximation for Normal Mixtures

- the derivation of EM-algorithm using KL-divergence can be generalized
- minimize the function

$$F(\lambda_L, \lambda_\theta) = \int\int q(L)q(\theta)\log\frac{q(L)q(\theta)}{p(\theta, L|y)}dLd\theta$$

which is $D(q(L)q(\theta)\|p(\theta, L|y))$

- substituting $q(\theta) = \delta(\theta - \theta_0)$ would give the function $-F(q, \theta_0)$ as in the EM-algorithm
- the variational approximation is more general, since it allows an arbitrary approximation distribution $q(\theta)$
- in the Normal mixture case, the approximating distributions can be obtained using free-form approximation

- the notation λ_L and λ_θ refer to parameters that define the approximating distributions $q(L)$ and $q(\theta)$
- for example, $q(\mu_j)$ is Normal so λ_θ contains the parameters of that Normal distribution
- the problem is to find the values of these parameters that minimize the KL-divergence

- the EM-like algorithm is now
  1. minimize F by choosing λ_L while keeping λ_θ fixed
  2. minimize F by choosing λ_θ while keeping λ_L fixed
  3. return to step 1 until convergence
- the full details of this example are quite complicated and are omitted here
- the details can be found in the paper *A Variational Bayesian Framework for Graphical Models*, H. Attias, NIPS-10, 2001, MIT Press.

- example: clustering data generated from a three-component Normal mixture
- the number of components in the model can be larger
- prior for the component means is such that the mean value $(0, 0)$ is most probable (circled in the demo)
- 'unused' components (those with very small precision) can be identified as those which converge to zero
- demo: `varmixt.R`

- compromise between representing the data well (many components) and avoiding overfitting (small number of components)
- variational approximation avoids overfitting automatically, since having too many components in the model makes the posterior very narrow at the mode
- there are numerous heuristics which attempt the same thing, such as MDL, AIC, and regularization

- the selection of components happens through making some components unused
- when some component is far from data, its effect to the posterior is mainly determined by the prior
- the prior is maximized by mean $(0, 0)$ and zero precision, so the "unused" components converge to these values