Lecture 12: mixtures, hierarchies, and experts
Topics

- Density estimation
  - Mixture models in classification, example
  - Hierarchical mixture models, estimation

- Conditional density models
  - experts, mixtures of experts
Review: Mixture density

- Data generation process:

\[ P(x) = \sum_{j=1,2} P(y = j) \cdot P(x|y = j) \] (generic mixture)

\[ = \sum_{j=1,2} p_j \cdot P(x|\mu_j, \Sigma_j) \] (mixture of Gaussians)

(exclusive events, additive probabilities)

- Any data point \( x \) could have been generated in two ways...
Review: the EM algorithm

**E-step:** First we perform a soft reassignment of examples based on the current mixture distribution, i.e., we compute

\[
\hat{p}(j|i) \leftarrow P(y_i = j|x_i, \theta), \text{ for all } j = 1, 2 \text{ and } i = 1, \ldots, n
\]

**M-step:** Then we re-estimate the parameters (separately for the two Gaussians) based on the soft assignments.

For each \( j = 1, 2 \), we maximize the likelihood of the corresponding weighted training set

\[
\sum_{i=1}^{n} \hat{P}(j|i) \log P(x_i|\mu_j, \Sigma_j)
\]
Review: the EM algorithm

**E-step:** First we perform a soft reassignment of examples based on the current mixture distribution, i.e., we compute

\[ \hat{p}(j|i) \leftarrow P(y_i = j|x_i, \theta), \text{ for all } j = 1, 2 \text{ and } i = 1, \ldots, n \]

**M-step:** Then we re-estimate the parameters (separately for the two Gaussians) based on the soft assignments.

\[
\hat{n}_j = \sum_{i=1}^{n} \hat{p}(j|i) = \text{Soft # of examples labeled } j \\
\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i) x_i \\
\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i) (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T
\]

where \( j = 1, 2 \).
Classification example

• A digit recognition problem (8x8 binary digits)
  Training set \( n = 100 \) (50 examples of each digit).
  Test set \( n = 400 \) (200 examples of each digit).

• We estimate a mixture of Gaussians model separately for each type of digit (with varying numbers of mixture components).

\[
\text{Class 1: } P(x|\hat{\theta}_1), \quad \text{Class 0: } P(x|\hat{\theta}_0)
\]

• Classification rule is based on the posterior class probability, or, equivalently, based on the log-likelihood ratio:

\[
\text{Class} = 1 \text{ if } \log \frac{P(x|\hat{\theta}_1)}{P(x|\hat{\theta}_0)} > 0 \quad \text{and Class} = 0 \text{ otherwise}
\]

(we are assuming that each digit is equally likely a priori)
Classification example cont’d

- The figure gives the number of missclassified examples on the test set as a function of the number of mixture components in each digit model.

- Anything wrong with this figure?
Classification example cont’d

• A single covariance matrix has $64 \times 65/2 = 2080$ parameters, we have $n = 50$ training examples...

• We can regularize the model

  We assign a prior distribution ($\sim$ Wishart) over each covariance matrix

  $$P(\Sigma | S, n') \propto \frac{1}{|\Sigma|^{n'/2}} \exp \left( -\frac{n'}{2} \text{Trace}(\Sigma^{-1} S) \right)$$

  (written here in a bit non-standard way)

  $S = \text{“prior” covariance matrix}$

  $n' = \text{equivalent sample size}$
Classification example cont’d

• In the resulting M-step we maximize the penalized log-likelihood of the (weighted) training set

$$\sum_{i=1}^{n} \hat{P}(j|i) \log P(x_i|\mu_j, \Sigma_j) + \log P(\Sigma_j|S, n')$$

• Adding such a regularization penalty changes the estimation of the covariance matrix only slightly

$$\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j + n'} \left[ \sum_{i=1}^{n} \hat{p}(j|i) (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T + n'S \right]$$

The remaining parts of the M-step are as before. Note that the E-step is unaffected (though the resulting values for the soft assignments will change)
Hierarchical mixture models

- We have already used a hierarchical mixture model in the digit recognition problem.

Data generation model:

First level: class distinction

Second level: class contingent components (e.g., style)
Hierarchical mixture models cont’d

• The hierarchy may not in general represent class distinction ⇒ the top level division may also be unobserved for all training examples.

**E-step:** We need to compute posterior probabilities over the possible paths in the tree.

\[
\hat{P}(j, k | i) \leftarrow \frac{P(y = j | x)}{P(c = k | y = j, x)},
\]

where \( j = 1, 2 \) and \( k = 1, 2, 3 \). In general the tree need not be symmetric.

---

**Diagram:**

```
P(y=1)------P(y=0)
  |          |
P(c=1|y=1)------P(c=3|y=0)
  |          |
P(x|y=1,c=1) . . . P(x|y=0,c=3)
```

```
Hierarchical mixture models cont’d

- The posterior over the first division

\[ P(y = j|x) = \frac{P(x|y = j)P(y = j)}{\sum_{j' = 1}^{2} P(x|y = j')P(y = j')} \]

where the probability of generating \( x \) from the \( y = j \) branch is

\[ P(x|y = j) = \sum_{k=1}^{3} P(c = k|y = j)P(x|y = j, c = k) \]
Hierarchical mixture models cont’d

The conditional posterior over the second division

\[ P(c = k | y = j, x) = \frac{P(x | y = j, c = k) P(c = k | y = j)}{\sum_{k' = 1}^{3} P(x | y = j, c = k') P(c = k' | y = j)} \]

Note that the normalization term equals \( P(x | y = j) \), the probability of generating \( x \) from one of the branches after \( y = j \).

This is a term we needed for evaluating the previous posterior \( \Rightarrow \) perhaps we can evaluate these probabilities by propagating information in the tree?
Propagation in hierarchical mixture models

- Bottom-up phase:

\[
P(x|y = j) = \sum_{k=1}^{3} P(x|y = j, c = k) P(c = k|y = j), \quad j = 0, 1
\]

\[
P(x) = \sum_{j=0}^{1} P(x|y = j) P(y = j)
\]
Propagation in hierarchical mixture models cont’d

- **Top-down phase:**

  \[
  P(y = j | x) = \frac{P(x | y = j)P(y = j)}{P(x)}
  \]

  \[
  P(c = k, y = j | x) = P(c = k | y = j, x) \times P(y = j | x)
  \]

  \[
  = \left[ \frac{P(x | y = j, c = k)P(c = k | y = j)}{P(x | y = j)} \right] \times P(y = j | x)
  \]
Hierarchical mixture models

- Can this happen?
Mixtures of experts

- Many regression or classification problems can be decomposed into smaller (easier) sub problems

Examples:
1. Dealing with various styles in handwritten character recognition
2. Dealing with dialect/accent in speech recognition etc.

- Each sub-problem can be solved by a specific “expert”
- The selection of which expert to rely on must depend on the context (i.e., the input $x$)
Experts

• Suppose we have several “experts” or component regression models generating conditional Gaussian outputs

\[ P(y|x, \theta_i) = N(y; w_i^T x + w_i0, \sigma_i^2) \]

where

- mean of \( y \) given \( x \) = \( w_i^T x + w_i0 \)
- variance of \( y \) given \( x \) = \( \sigma_i^2 \)

We use \( \theta_i = \{w_i, w_i0, \sigma_i^2\} \) to denote the parameters of the \( i^{th} \) expert.

• We need to find an appropriate way of allocating tasks to these experts (linear regression models)
Mixtures of experts

Example:

- Here we need a switch or a gating network that selects the appropriate expert (linear regression model) as a function of the input $x$. 
**Gating network**

- A simple gating network is a probability distribution over the choice of the experts conditional on the input $x$.

- Example: in case of two experts (0 and 1), the gating network can be a logistic regression model.

  $$ P(\text{expert} = 1 | x, v, v_0) = g( v^T x + v_0 ) $$

  where $g(z) = (1 + e^{-z})^{-1}$ is the logistic function.

- In case of $m > 2$ experts, the gating network can be a softmax model.

  $$ P(\text{expert} = j | x, \eta) = \frac{\exp( v_j^T x + v_j0 )}{\sum_{j' = 1}^{m} \exp( v_{j'}^T x + v_{j'0} )} $$

  where $\eta = \{v_1, \ldots, v_m, v_{10}, \ldots, v_{m0}\}$ are the parameters in the gating network.
Gating network cont’d

\[
P(\text{expert} = j| x, \eta) = \frac{\exp(\mathbf{v}_j^T \mathbf{x} + v_{j0})}{\sum_{j'=1}^{m} \exp(\mathbf{v}_{j'}^T \mathbf{x} + v_{j'0})}
\]
Mixtures of experts model

- The probability distribution over the output $y$ given $x$ is

$$P(y|x, \theta, \eta) = \sum_{j=1}^{m} P(\text{expert} = j|x, \eta) P(y|x, \theta_j)$$

- The allocation of experts is made conditionally on the input

- Only a single expert is assumed to be responsible for any specific input output mapping