Mixtures-of-Experts

Robert Jacobs
Department of Brain & Cognitive Sciences
University of Rochester
Rochester, NY 14627, USA

August 8, 2008

The mixtures-of-experts (ME) architecture is a mixture model in which the mixture components are conditional probability distributions. Consequently, you should first understand the handout on mixture models before attempting to understand this handout.

ME architectures are generally used for regression or classification problems. They are interesting because they attempt to solve these problems using a "divide-and-conquer" strategy; that is, complex problems are decomposed into a set of simpler sub-problems. We assume that the data can be adequately summarized by a collection of functions, each of which is defined over a local region of the domain. The approach adopted here attempts to allocate different modules (experts) to summarize the data located in different regions.

ME architectures can be characterized as fitting a piecewise function to the data. The data are assumed to form a countable set of paired variables $\mathcal{X} = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_{t=1}^{T}$, where $\mathbf{x}$ is a vector of explanatory variables, also referred to as covariates, and $\mathbf{y}$ is a vector of responses. ME architectures divide the covariate space, meaning the space of all possible values of the explanatory variables, into regions, and fit simple surfaces to the data that fall in each region. Unlike many other piecewise approximators, these architectures use regions that are not disjoint. The regions have "soft" boundaries meaning that data points may lie simultaneously in multiple regions. In addition, the boundaries between regions are themselves simple parameterized surfaces whose parameter values are estimated from the data.

The ME architecture may be understood as an "associative mixture model" (or a conditional mixture model); that is, each distribution in a mixture model (e.g., Gaussian) is replaced by a conditional distribution (such as, for example, a neural network), and all the quantities in a mixture model are replaced by quantities that are conditional on the input vector $\mathbf{x}$.

It is assumed that the environment generating the data is decomposable into a set of sub-processes defined on (possibly overlapping) regions of the input space. For each data item, a sub-process is selected, based on the input $\mathbf{x}^{(t)}$, and the selected sub-process maps $\mathbf{x}^{(t)}$ to the output $\mathbf{y}^{(t)}$. More precisely, data are generated as follows. For each input $\mathbf{x}^{(t)}$,

1. a sub-process i is chosed from a multinomial distribution with probability $P(i|\mathbf{x}^{(t)})$ (we will denote this probability as $g_i^{(t)}$ for notational convenience);

2. an output $\mathbf{y}^{(t)}$ is generated by sub-process i with probability $P(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, i)$. Let's assume that this probability distribution is Gaussian (so what follows is appropriate for regression, but

not necessarily appropriate for other tasks such as classification), and let's also assume that the output is a scalar instead of a vector:

$$P(y^{(t)}|\mathbf{x}^{(t)}, i) = \frac{1}{\sqrt{2\pi}\sigma_i} \, e^{-\frac{1}{2\sigma_i^2}(y^{(t)} - \mu_i^{(t)})^2} \tag{1}$$

where $\mu_i^{(t)}$ is the mean of this distribution and σ_i^2 is its variance.

[Please go back and look at the handout on mixture models; steps 1 and 2 above should look very much like steps 1 and 2 on that handout. In fact, all the equations on this handout should look very much like the equations on that handout.] Note that the overall probability of the value $y^{(t)}$ is given by:

$$\begin{aligned} P(y^{(t)}|\mathbf{x}^{(t)}) &= \sum_i p(y^{(t)}, i|\mathbf{x}^{(t)}) && (2) \\ &= \sum_i P(i|\mathbf{x}^{(t)}) \, P(y^{(t)}|\mathbf{x}^{(t)}, i) && (3) \\ &= \sum_i g_i^{(t)} \frac{1}{\sqrt{2\pi}\sigma_i} \, e^{-\frac{1}{2\sigma_i^2}(y^{(t)} - \mu_i^{(t)})^2}. && (4) \end{aligned}$$

This is the situation for a particular data item $(\mathbf{x}^{(t)}, y^{(t)})$. In general, there will be many data items. The data items are independent and identically distributed so the probability of getting the entire sample $\mathcal{X}$ is:

$$\begin{aligned} P(\mathcal{X}) &= \prod_t \sum_i P(i|\mathbf{x}^{(t)}) \, P(y^{(t)}|\mathbf{x}^{(t)}, i) && (5) \\ &= \prod_t \sum_i g_i^{(t)} \frac{1}{\sqrt{2\pi}\sigma_i} \, e^{-\frac{1}{2\sigma_i^2}(y^{(t)} - \mu_i^{(t)})^2}. && (6) \end{aligned}$$

The product arises from the fact that the individual data items are generated independently.

Suppose that we consider data item $(\mathbf{x}^{(t)}, y^{(t)})$ and we want to know what subprocess it came from. That is, we want to know the probabilities $P(i|\mathbf{x}^{(t)}, y^{(t)})$. Using Bayes' rule, we get:

$$\begin{aligned} P(i|\mathbf{x}^{(t)}, y^{(t)}) &= \frac{P(i|\mathbf{x}^{(t)}) \, P(y^{(t)}|\mathbf{x}^{(t)}, i)}{\sum_j P(j|\mathbf{x}^{(t)}) \, P(y^{(t)}|\mathbf{x}^{(t)}, j)} && (7) \\ &= \frac{g_i^{(t)} \frac{1}{\sqrt{2\pi}\sigma_i} \, e^{-\frac{1}{2\sigma_i^2}(y^{(t)} - \mu_i^{(t)})^2}}{\sum_j g_j^{(t)} \frac{1}{\sqrt{2\pi}\sigma_j} \, e^{-\frac{1}{2\sigma_j^2}(y^{(t)} - \mu_j^{(t)})^2}}. && (8) \end{aligned}$$

Note that this is a powerful tool. We are not told what subprocess a particular data item was sampled from, but we are able to determine the probabilities that it came from the different subprocesses. Using Bayesian terminology, we will refer to $P(i|\mathbf{x}^{(t)})$ as the prior probability of subprocess

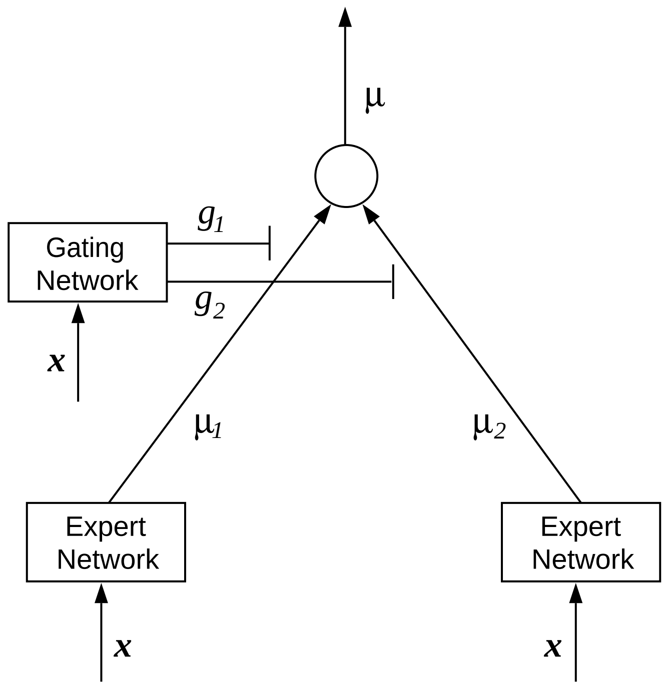

Figure 1: A mixtures-of-experts architecture.

i, and to $P(i|\mathbf{x}^{(t)}, y^{(t)})$ as its posterior probability (we will denote this posterior probability as $h_i^{(t)}$ for notational convenience).

Figure 1 presents a graphical representation of the ME architecture. The architecture consists of n modules referred to as *expert networks*. These networks approximate the data within each region of the input space: expert network i maps its input, the input vector $\mathbf{x}$, to an output vector $\boldsymbol{\mu_i}$. It is assumed that different expert networks are appropriate in different regions of the input space. Consequently, the architecture requires a module, referred to as a *gating network*, that identifies for any input $\mathbf{x}$, the expert or blend of experts whose output is most likely to approximate the corresponding target output $\mathbf{y}$. The gating network outputs are a set of scalar coefficients g_i that weight the contributions of the various experts. For each input $\mathbf{x}$, these coefficients are constrained to be nonnegative and to sum to one. The total output of the architecture, given by

$$\boldsymbol{\mu} = \sum_{i=1}^{n} g_i \boldsymbol{\mu_i}, \tag{9}$$

is a convex combination of the expert outputs for each $\mathbf{x}$.

From the perspective of statistical mixture modeling, we identify the gating network with the selection of a particular sub-process. That is, the gating outputs g_i are interpreted as the input-dependent, multinomial probabilities of selecting sub-process i. Different expert networks are identified with different sub-processes; each expert models the input-dependent distributions associated with its corresponding sub-process. The output units of the gating network must use the "softmax" activation function:

$$g_i = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{10}$$

where s_i is the weighted sum of unit i's inputs. The choice of activation function for the output units of the expert networks depends upon the nature of the problem: for regression problems, they should be linear; for binary classification problems, they should be logistic. The expert and gating networks may or may not have layers of hidden units.

Recall that for maximum likelihood estimation, we are interested in finding the values of the expert and gating network outputs that maximize the log likelihood of the probability of the data. That is, we want to adjust our parameters (via gradient ascent for the purposes of this note) so as to maximize the log likelihood function:

$$\begin{aligned}\log L &= \log p(\mathcal{X}) && (11)\\ &= \sum_t \log \sum_i P(i|\mathbf{x}^{(t)})\, P(y^{(t)}|\mathbf{x}^{(t)}, i) && (12)\\ &= \sum_t \log \sum_i g_i^{(t)} \frac{1}{\sqrt{2\pi}\sigma_i}\, e^{-\frac{1}{2\sigma_i^2}(y^{(t)}-\mu_i^{(t)})^2}. && (13)\end{aligned}$$

First we consider the partial derivative of the log likelihood with respect to the weighted sum s_i at the i^{th} output unit of the gating network:

$$\begin{aligned}\frac{\partial \log L}{\partial s_i} &= \sum_t p(i|\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) - p(i|\mathbf{x}^{(t)}) && (14)\\ & \sum_t h_i^{(t)} - g_i^{(t)}. && (15)\end{aligned}$$

Note that this has a strong Bayesian flavor; the derivative equals zero when the prior probabilities $[p(i|\mathbf{x}^{(t)})]$ and posterior probabilities $[h_i^{(t)} = p(i|\mathbf{x}^{(t)}, \mathbf{y}^{(t)})]$ are equal.

Consider now the derivatives of the log likelihood with respect to μ_i, the output of the i^{th} expert network:

$$\frac{\partial \log L}{\partial \mu_i} = \sum_t \frac{h_i^{(t)}}{\sigma_i^2}(y^{(t)} - \mu_i^{(t)}). \tag{16}$$

These derivatives weight the error term $y^{(t)} - \mu_i^{(t)}$ by the posterior probability $h_i^{(t)}$ associated with the i^{th} expert network. That is, expert network i's weights are adjusted to reduce the error between its output and the target vector, but only in proportion to its posterior probability. Typically only one expert network has a large posterior probability for each training pattern and, thus, only one expert network learns each training pattern. In general, different expert networks learn different training patterns.

Finally we present the derivative of the log likelihood with respect to the variance σ_i^2 associated with the i^{th} expert network:

$$\frac{\partial \log L}{\partial \sigma_i^2} = \sum_t \frac{h_i^{(t)}}{2\sigma_i^4}[(y^{(t)} - \mu_i^{(t)})^2 - \sigma_i^2]. \tag{17}$$

The variance σ_i^2 is adjusted toward the sample variance $(y^{(t)} - \mu_i^{(t)})^2$, but with a step size that is weighted by the posterior probability $h_i^{(t)}$.

It is possible to extend ME architectures to a hierarchical system known as hierarchical mixtures-of-experts (HME). In addition, it is possible to estimate the parameter values of ME and HME architectures via an Expectation-Maximization (EM) algorithm. See the references for details.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3, 79-87.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6, 181-214.

Peng, F., Jacobs, R. A., and Tanner, M. A. (1996). Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *Journal of the American Statistical Association*, 91, 953-960.