IDIAP COMMUNICATION

IDIAP
Martigny - Valais - Suisse

Some Methods for Training Mixtures of Experts

Perry Moerland *

IDIAP–Com 97-05

November 97

Dalle Molle Institute for Perceptive Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

* e-mail: Perry.Moerland@idiap.ch

1 Introduction

Recently, a modular architecture of neural networks known as a *mixture of experts* (ME) [8][9] has attracted quite some attention. MEs are mixture models which attempt to solve problems using a divide-and-conquer strategy; that is, they learn to decompose complex problems in simpler subproblems. In particular, the *gating* network of a ME learns to partition the input space (in a soft way, so overlaps are possible) and attributes *expert* networks to these different regions. The divide-and-conquer approach has shown particularly useful in attributing experts to different regimes in piece-wise stationary time series [20], modeling discontinuities in the input-output mapping, and classification problems [6][13][19].

The ME error function is based on the interpretation of MEs as a mixture model [12] with conditional densities as mixture components (for the experts) and gating network outputs as mixing coefficients. The purpose of this note is to describe various existing methods for minimizing this ME error function and to do so in a unified notation. Learning algorithms treated are gradient descent, quasi-Newton methods, Expectation Maximization (EM) [5][20], and various one-pass solutions of the maximization step of the EM algorithm. The last section gives a short summary of how mixtures of experts can estimate local error bars [20].

2 Mixtures of Experts

In this section the basic definitions of the mixture of experts model are given which will be used in the rest of this note.

Figure 1 shows the architecture of a ME network, consisting of three expert networks and one gating network both having access to the input vector $\mathbf{x}$; the gating network has one output g_i per expert. The standard choices for gating and expert networks are generalized linear models [9] and multilayer perceptrons [20]. The output vector of a ME is the weighted (by the gating network outputs) mean of the expert outputs (the weights of the sub-networks have been left out):

$$\mathbf{y}(\mathbf{x}) = \sum_{j=1}^{m} g_j(\mathbf{x})\mathbf{y}_j(\mathbf{x}). \tag{1}$$

The gating network outputs $g_j(\mathbf{x})$ can be regarded as the probability that input $\mathbf{x}$ is attributed to expert j. In order to ensure this probabilistic interpretation, the activation function for the outputs of the gating network is chosen to be the soft-max function [4]:

$$g_j = \frac{\exp(z_j)}{\sum_{i=1}^{m} \exp(z_i)}, \tag{2}$$

where the z_i are the gating network outputs before thresholding. This soft-max function makes that the gating network outputs sum to unity and are non-negative; thus implementing the (soft) competition between the experts.

A probabilistic interpretation of a ME can be given in the context of mixture models for conditional probability distributions (see section 6.4 in [1]; again the dependence on the weights has been left implicit):

$$p(\mathbf{t}|\mathbf{x}) = \sum_{j=1}^{m} g_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x}), \tag{3}$$

where the ϕ_j represent the conditional densities of target vector $\mathbf{t}$ for expert j. The use of a soft-max function in the gating network and the fact that the ϕ_j are densities guarantee that the distribution is normalized: $\int p(\mathbf{t}|\mathbf{x})\, d\mathbf{t} = 1$.

This distribution forms the basis for the ME error function which can be optimized using gradient descent or the Expectation-Maximization (EM) algorithm [9]. Of course, a global least-squares

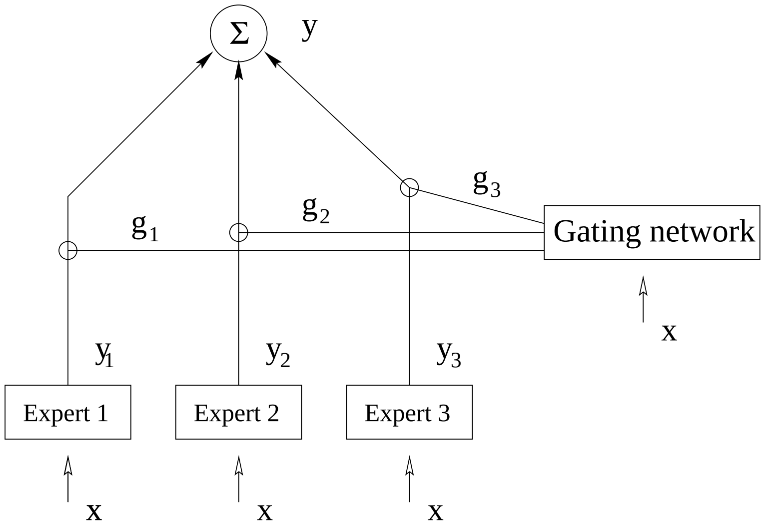

Figure 1: Architecture of a mixture of experts network.

approach could also be used and might be more appropriate when a division in subproblems is not feasible [3].

A standard way to motivate error functions is from the principle of maximum likelihood of the (independently distributed) training data with input vectors $\mathbf{x}^n$ and target vectors $\mathbf{t}^n$: $\{\mathbf{x}^n, \mathbf{t}^n\}$ (see section 6.1 in [1]):

$$\mathcal{L} = \prod_n p(\mathbf{x}^n, \mathbf{t}^n) = \prod_n p(\mathbf{t}^n|\mathbf{x}^n)p(\mathbf{x}^n),$$

where dependence of $p(\mathbf{x}^n, \mathbf{t}^n)$ and $p(\mathbf{t}^n|\mathbf{x}^n)$ on the network parameters has been left implicit. A cost function is then obtained by taking the negative logarithm of the likelihood (and dropping the term $p(\mathbf{x}^n)$ which does not depend on the network parameters):

$$E = -\sum_n \ln p(\mathbf{t}^n|\mathbf{x}^n). \tag{4}$$

The most suitable choice for the conditional probability density depends on the problem. For regression problems a Gaussian noise model is often used (leading to the sum-of-squares error function); for classification problems with a 1-of-c coding scheme, a multinomial density is most suitable (leading to the cross-entropy error function).

The *ME error function* is based on a mixture of conditional probability densities (substituting (3) in (4)):

$$E = -\sum_n \ln \sum_{j=1}^{m} g_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n),$$

the exact formulation of which depends on the choice for the conditional densities $\phi_j(\mathbf{t}^n|\mathbf{x}^n)$ of the experts. A useful definition for the next section is the per pattern error:

$$E' = -\ln \sum_{j=1}^{m} g_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x}). \tag{5}$$

3 Learning Algorithms for Mixtures of Experts

In this section a brief overview is given of different learning algorithms for minimizing the ME error function (4). The first approach consists of standard gradient-based learning and has been applied with some success in the training of (H)MEs [8][9]. The second approach is an instance of the Expectation-Maximization (EM) algorithm [5], which is often applied to unconditional mixture models [12] and has also been formulated for and applied to conditional mixtures of experts [9][10]. The advantage of the EM algorithm as compared with the gradient descent approach lies in the fact that EM nicely decouples the parameter estimation for the different components of a ME model.

3.1 Gradient Descent

Many standard optimization methods (back-propagation, line search, quasi-Newton) are based on the calculation of gradients. For feed-forward neural networks this involves in specific the partial derivatives of the error function with respect to the network outputs (before thresholding); these derivatives (commonly denoted as δ_j) form the basis for the back-propagation algorithm [17]. In section 6.4 of [1], the partial derivative for the gating network with respect to its outputs have been calculated in the context of a gradient descent algorithm for the mixture model (3). Bishop's outcomes are restated here (using the chain rule):

$$\frac{\partial E'}{\partial z_j} = \sum_k \frac{\partial E'}{\partial g_k}\frac{\partial g_k}{\partial z_j} = \sum_k -\frac{\pi_k}{g_k}(\delta_{jk} g_k - g_j g_k) = g_j - \pi_j. \tag{6}$$

where the posterior probability π_j is defined as:

$$\pi_j(\mathbf{x}, \mathbf{t}) = \frac{g_j \phi_j}{\sum_i g_i \phi_i}, \tag{7}$$

and δ_{jk} is the Kronecker delta. For the expert networks:

$$\frac{\partial E'}{\partial a_{jc}} = \sum_k \frac{\partial E'}{\partial y_{jk}}\frac{\partial y_{jk}}{\partial a_{jc}}, \tag{8}$$

the second term of which depends on the activation function in the output layer of the expert networks. If the activation function is the linear identity ($y_{jc} = a_{jc}$), then:

$$\frac{\partial y_{jk}}{\partial a_{jc}} = \delta_{ck}. \tag{9}$$

If the activation function is a soft-max function:

$$y_{jc} = \frac{\exp(a_{jc})}{\sum_i \exp(a_{ji})}, \tag{10}$$

then:

$$\frac{\partial y_{jk}}{\partial a_{jc}} = \delta_{ck} y_{jk} - y_{jc} y_{jk}. \tag{11}$$

Using the definition of E' (5), the first term in the summation of (8) is:

$$\frac{\partial E'}{\partial y_{jk}} = \frac{\partial \left(-\ln \sum_{i=1}^{m} g_i \phi_i\right)}{\partial y_{jk}} = -\frac{g_j}{\sum_{i=1}^{m} g_i \phi_i}\frac{\partial \phi_j}{\partial y_{jk}}, \tag{12}$$

where the last term depends on our choice for the noise model ϕ_j which in this note is either Gaussian or multinomial.

3.1.1 Gaussian Conditional Density

In section 6.4 of [1] mixture models are considered with multi-dimensional Gaussian conditional densities (where the covariance matrix is the identity matrix) as mixture components:

$$\phi_j(\mathbf{t}^n|\mathbf{x}^n) = \frac{1}{(2\pi)^{(d/2)}}\exp\left(-\frac{||\mathbf{t}-\mathbf{y}_j(\mathbf{x},\mathbf{w}_j)||^2}{2}\right), \tag{13}$$

where d is the dimensionality of $\mathbf{t}$ and $\mathbf{w}_j$ is the set of weight parameters of expert j. In the Gaussian case this leads to (using (13)):

$$\frac{\partial \phi_j}{\partial y_{jk}} = \phi_j(t_k - y_{jk}), \tag{14}$$

Recombining in the case of a Gaussian conditional density and linear activation function ((9), (12) and (14) in (8)):

$$\frac{\partial E'}{\partial a_{jc}} = \sum_k \delta_{ck}\pi_j(y_{jk}-t_k) = \pi_j(y_{jc}-t_c). \tag{15}$$

3.1.2 Multinomial Conditional Density

A suitable choice for the expert conditional density function in classification problems with 1-of-c coding is a multinomial density:

$$\phi_j(\mathbf{t}^n|\mathbf{x}^n) = \prod_{c=1}^{C}(y_{jc}(\mathbf{w}_j)^n)^{t_c^n}, \tag{16}$$

where $\mathbf{w}_j$ is the set of weight parameters of expert j. With multinomial conditional densities, a suitable choice for the activation function for the expert output units of is the soft-max function (2). For a multinomial conditional density this leads to (using (16)):

$$\frac{\partial \phi_j}{\partial y_{jk}} = \phi_j \frac{t_k}{y_{jk}}. \tag{17}$$

For a multinomial conditional density and soft-max activation function this gives for the output error terms of the expert networks ((11), (12) and (17) in (8))):

$$\frac{\partial E'}{\partial a_{jc}} = -\sum_k(\delta_{ck}y_{jk} - y_{jc}y_{jk})\pi_j\frac{t_k}{y_{jk}} = \pi_j(y_{jc}-t_c). \tag{18}$$

Thus, the output error terms of the expert networks are similar to the ones found for the well-known sum-of-squares and cross-entropy error functions but with the posterior probabilities π_j as an extra weighting factor. For example, if the expert and gating networks are perceptrons with a Gaussian conditional density and linear activation function for the experts (thus, expert and gating networks are generalized linear models [11]) the updates for the expert network weights w_j :

$$\Delta\mathbf{w}_j = \eta\pi_j(\mathbf{y}_j - \mathbf{t})\mathbf{x}^T,$$

and for the gating network weights v_j:

$$\Delta\mathbf{v}_j = \eta(g_j - \pi_j)\mathbf{x},$$

where η denotes the learning rate. Of course, the gradients obtained in this section could also be used in more powerful non-linear optimization techniques such as conjugate gradient algorithms and quasi-Newton methods.

3.2 Expectation Maximization

The basic idea of the EM algorithm is that the maximization of the likelihood $\mathcal{L}$ can often be simplified if a set of *missing* variables were known. The likelihood for the ME model would, for example, be considerably simplified if each pattern is associated with only one expert indicated by variables:

$$z_j^n = \begin{cases} 1 & \text{if pattern } \mathbf{t}^n \text{ is generated by expert } j \\ 0 & \text{otherwise} \end{cases}$$

Then the complete conditional probability density (including the missing variables) can be written as:

$$p(\mathbf{t}, \mathbf{z}|\mathbf{x}) = \sum_{j=1}^{m} z_j (g_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x})) = \prod_{j=1}^{m} (g_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x}))^{z_j}.$$

Substituting this in (4) gives the complete error function

$$E_c = -\sum_n \sum_{j=1}^{m} z_j^n \ln\left(g_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right). \tag{19}$$

Comparing the complete error function with the original ME error function (4) shows that the introduction of missing variables has allowed to move the logarithm inside resulting in a number of separate error minimization problems for each of the mixture components. The problem, however, is that one does not know the distribution of the missing variables z_j^n. The next important ingredient of the EM algorithm is therefore a (iterative) two-step approach consisting of an E-step in which the expectation of E_c with respect to the missing variables is calculated (based on the current parameter values) and of a M-step that minimizes this expected complete error function (or equivalently maximizes the likelihood). In [5] it has been shown that the decrease of the expected complete error function implies the decrease of the original error function E which guarantees convergence to a local minimum. A more detailed treatment of the convergence of the EM algorithm for mixture of experts can be found in [10].

E-step: The expectation of the complete error function is:

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} \mathcal{E}(z_j^n) \ln\left(g_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right), \tag{20}$$

where the expected values of the missing variables are (using Bayes' rule):

$$\mathcal{E}(z_j^n) = P(z_j^n = 1|\mathbf{t}^n, \mathbf{x}^n) = \frac{p(\mathbf{t}^n|z_j^n = 1, \mathbf{x}^n)P(z_j^n = 1|\mathbf{x}^n)}{p(\mathbf{t}^n|\mathbf{x}^n)},$$

which gives using the probabilistic interpretation of MEs (3) and the definition of π_j (7):

$$\mathcal{E}(z_j^n) = \frac{g_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n)}{\sum_{i=1}^{m} g_i(\mathbf{x}^n)\phi_i(\mathbf{t}^n|\mathbf{x}^n)} = \pi_j(\mathbf{x}^n, \mathbf{t}^n). \tag{21}$$

Substituting the expected values of the missing variables (21) in the expectation of the complete error function (20) gives:

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} \pi_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(g_j(\mathbf{x}^n)\right) \; - \; \sum_n \sum_{j=1}^{m} \pi_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right), \tag{22}$$

the terms of which can be minimized separately in each M-step. An interpretation of the first (cross-entropy) term is as the entropy of distributing a pattern $\mathbf{x}$ amongst the expert networks. This cost is minimal if experts are mutually exclusive and increases when experts share a pattern [20]. The second term has the general form of a weighted maximum likelihood problem; the weighting with π_j implies that the important experts are the ones with a large value for π_j. Thus, the error function nicely incorporates the soft splitting of the input space which is an essential characteristic of the ME model.

M-step: The term to be minimized for the gating network is:

$$E_{\text{gate}} = -\sum_n \sum_{j=1}^{m} \pi_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(g_j(\mathbf{x}^n)\right), \tag{23}$$

and for expert network j:

$$E_{\text{expert}} = -\sum_n \pi_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right). \tag{24}$$

This step of the EM algorithm can have different forms depending on the network architectures for gates and experts and which of the variants of the EM algorithm is chosen. With respect to the network architectures the two main options are to use either simple perceptrons (generalized linear models) or multilayer perceptrons. With simple perceptrons and exponential conditional densities (of which the Gaussian and multinomial density are a special case) for ϕ_j, the optimization problems reduce to (weighted) maximum likelihood problems for generalized linear models [9][11]. These can be solved efficiently with the iteratively weighted least-squares (IRLS) algorithm. A detailed description of the IRLS algorithm can be found in [7][9]. For Gaussian conditional densities the optimization of the parameters of the expert networks even reduces to a one-pass algorithm using pseudo-inverses (see section 3.2.1 below).

When the expert and gating networks are chosen to be multilayer perceptrons (MLPs) gradient-based optimization is most appropriate. The use of MLPs has the advantage that it adds non-linearity to the ME model; on the other hand convergence to a local minimum is no longer guaranteed [20]. A gradient approach (with multinomial or Gaussian conditional densities) leads to the same output error terms as before: (6) for the gating network and (18) for the expert networks.

The EM algorithm comes in several variants, the basic one being the one described above with a M-step that minimizes the various error functions. A variant that is often used is called Generalized EM (GEM) [5] that is based on the weaker assumption of decreasing (not necessarily minimizing) the error functions. A partial implementation of the E-step has been proposed in [14] which is basically an on-line EM algorithm. Also for these variants the convergence towards a local minimum is still guaranteed.

3.2.1 Weighted Least-Squares Algorithms for M-Step

In this section, a simple heuristic to reduce the M-step for MEs with perceptrons as gating and expert networks to a one-pass calculation [9] is described. For this purpose we investigate the maximization problems to be solved: eq. (23) for the gating network and eq. (24) for the expert networks. Their derivatives with respect to the weights $\mathbf{v}_j$ and $\mathbf{w}_j$ in the case of a Gaussian or multinomial conditional density follow directly from section 3.1; set to zero this gives for the gating network (with pattern index n):

$$-\sum_n (g_{j,n} - \pi_{j,n})\mathbf{x}_n = 0, \tag{25}$$

and for the expert networks:

$$-\sum_n \pi_{j,n}(\mathbf{y}_{j,n} - \mathbf{t}_n)\mathbf{x}_n^T = 0. \tag{26}$$

In the case of a Gaussian conditional density with a linear activation function for the expert networks, the solution of eq. (26) is a weighted least-squares problem that has an exact solution using pseudo-inverses (see, for example [1][16]). In that case eq. (26) is in matrix notation (using $Y_j = XW_j^T$):

$$X^T \Pi_j (XW_j^T - T) = 0,$$

where with N training patterns, I network inputs, and O network outputs, X is the pattern matrix of size $N \times I$, W_j is the weight matrix for expert j with dimensions $O \times I$, T is the target matrix of size $N \times O$, and Π_j is the diagonal matrix of the coefficients $\pi_{j,n}$ of size $N \times N$. The one-step solution of this equation is then:

$$W_j^T = (X^T \Pi_j X)^{-1} X^T \Pi_j T.$$

In order to avoid problems with singularity (of the matrix $X^T \Pi_j X$) and round-off errors, other possibilities are the use of QR decomposition or the even more suitable *singular value decomposition* (SVD) (§15.4 in [16]), that directly find the best solution in the least-squares sense of the system of linear equations associated with eq. (26):

$$\sqrt{\Pi_j} X W_j^T = \sqrt{\Pi_j} T.$$

For the gating network and in the case of a multinomial conditional density with a soft-max activation function for the expert networks, this least-squares approach cannot be applied directly because of the soft-max function giving the outputs $\mathbf{y}_{j,n}$ and $g_{j,n}$. However, from eqs. (25) and (26) it is clear that ordinary weighted least-squares can be used for the transformed equations where the inverse of the soft-max is applied to the network outputs, the posteriors $\pi_{j,n}$, and the targets $\mathbf{t}_n$ [9]. Inverting the soft-max:

$$y_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad \text{gives} \quad x_i = \ln y_i + \ln \sum_j \exp(x_j),$$

where the second term is constant for all x_i and disappears when the soft-max is applied. This means that eqs. (25) and (26) can be approximated by just taking the log of the targets:

$$-\sum_n (z_{j,n} - \ln(\pi_{j,n}))\mathbf{x}_n^T = 0,$$

and for the expert networks:

$$-\sum_n \pi_{j,n}(\mathbf{a}_{j,n} - \ln(\mathbf{t}_n))\mathbf{x}_n^T = 0.$$

The exact solution using pseudo-inverses is then, for the gating network:

$$V^T = (X^T X)^{-1} X^T \ln(\Pi_j),$$

and for the expert networks:

$$W_j^T = (X^T \Pi_j X)^{-1} X^T \Pi_j \ln(T).$$

Finally, a last obstacle for the application of this technique is that we have to avoid taking the log of zero values in T and Π_j. Therefore, a small positive value (in practice 10^{-3} has proven to work well) is added to the elements of these matrices. Of course, also in this case the problem can be solved with SVD instead of the more sensitive pseudo-inverses.

3.2.2 Gating Network with Gaussian Kernels

Another way to reduce the M-step for the gating network to a one-pass calculation has been proposed by Xu and Jordan [22]. Their idea is to use a modified gating network consisting of normalized kernels (by applying Bayes' rule):

$$g_j(\mathbf{x}) = P(j|\mathbf{x}) = \frac{\alpha_j P_j(\mathbf{x})}{\sum_i \alpha_i P_i(\mathbf{x})}, \tag{27}$$

where $\sum_i \alpha_i = 1$, $\alpha_i \geq 0$, and the P_i's are probability density functions. Thus the gating network outputs g_j sum to one and are non-negative. It is interesting to note that the numerators in eq. (27) can be interpreted as the components of a simple mixture model [12]; a fact which can be used to find a good initialization of the kernel parameters. This choice of the gating outputs leads to the following model (substituting (27) in (3)):

$$p(\mathbf{t}|\mathbf{x}) = \sum_{j=1}^{m} \frac{\alpha_j P_j(\mathbf{x})}{\sum_i \alpha_i P_i(\mathbf{x})} \phi_j(\mathbf{t}|\mathbf{x}). \tag{28}$$

To obtain a one-pass solution for the gating network parameters, maximum likelihood estimation is not performed on this conditional density, but on the joint density:

$$p(\mathbf{x}, \mathbf{t}) = p(\mathbf{t}|\mathbf{x})p(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j P_j(\mathbf{x})\phi_j(\mathbf{t}|\mathbf{x}),$$

which by maximum likelihood leads to the following error function:

$$E = -\sum_n \ln \sum_{j=1}^{m} \alpha_j P_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n).$$

In the EM framework, the complete error function is then (see eq. (19)):

$$E_c = -\sum_n \sum_{j=1}^{m} z_j^n \ln\left(\alpha_j P_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right),$$

which with a slight variation of the derivation in section 3.2 (basically replacing $g_j(\mathbf{x}^n)$ with $\alpha_j P_j(\mathbf{x}^n)$) gives for the E-step:

$$\mathcal{E}(z_j^n) = \frac{\alpha_j P_j(\mathbf{x}^n)\phi_j(\mathbf{t}^n|\mathbf{x}^n)}{\sum\limits_{i=1}^{m} \alpha_i P_i(\mathbf{x}^n)\phi_i(\mathbf{t}^n|\mathbf{x}^n)} = h_j(\mathbf{x}^n, \mathbf{t}^n).$$

The expectation of the complete error function is:

$$\mathcal{E}(E_c) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(\alpha_j P_j(\mathbf{x}^n)\right) \; - \; \sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(\phi_j(\mathbf{t}^n|\mathbf{x}^n)\right).$$

Comparing this equation with eq. (22) shows that expert error function (the second part of the equation) did not change and that consequently the M-step for the expert networks does not change.

In the rest of this section the probability density function P_j is chosen to be Gaussian with a local variance σ_j for each gating output:

$$P_j(\mathbf{x}) = \frac{1}{(2\pi\sigma_j^2)^{(d/2)}} \exp\left(-\frac{||\mathbf{x} - \boldsymbol{\mu}_j||^2}{2\sigma_j^2}\right).$$

Then the parameters of the gating network $\boldsymbol{\mu}_j$, σ_j, and α_j can be obtained by taking the partial derivatives of the gating error function:

$$E_{\text{gate}} = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln\left(\alpha_j P_j(\mathbf{x}^n)\right) = -\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln \alpha_j - \sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln P_j(\mathbf{x}^n). \tag{29}$$

This is exactly the error function that is minimized when applying the EM algorithm to a simple Gaussian mixture model (see, for example, section 2.6 in [1]). For the parameters α, the constraint $\sum_j \alpha_j = 1$ leads to a Lagrangian function using the first part of the gating error function (29):

$$L(\alpha, \lambda) = -(\sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln \alpha_j) + \lambda(\sum_j \alpha_j - 1) = 0,$$

and taking the partial derivatives with respect to α and λ gives a system of $m+1$ linear equations, the solution of which is:

$$\alpha_j = \frac{1}{N} \sum_n h_j(\mathbf{x}^n, \mathbf{t}^n),$$

where N is the number of patterns in the training set.

For the centers $\boldsymbol{\mu}_j$ of the Gaussian kernels, the partial derivative of the second part of the gating error function (29) gives:

$$\frac{\partial \left(- \sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln P_j(\mathbf{x}^n) \right)}{\partial \boldsymbol{\mu}_j} = - \sum_n \frac{h_j(\mathbf{x}^n, \mathbf{t}^n)}{P_j(\mathbf{x}^n)} \frac{\partial P_j(\mathbf{x}^n)}{\partial \boldsymbol{\mu}_j} = - \sum_n h_j(\mathbf{x}^n, \mathbf{t}^n) \frac{(\mathbf{x}^n - \boldsymbol{\mu}_j)}{\sigma_j^2}.$$

Setting this partial derivative to zero, we obtain a new estimate for the means:

$$\boldsymbol{\mu}_j = \frac{\sum_n h_j(\mathbf{x}^n, \mathbf{t}^n) \mathbf{x}^n}{\sum_n h_j(\mathbf{x}^n, \mathbf{t}^n)}.$$

For the local variances σ_j of the Gaussian kernels, the partial derivative of the second part of the gating error function (29):

$$\frac{\partial \left(- \sum_n \sum_{j=1}^{m} h_j(\mathbf{x}^n, \mathbf{t}^n) \ln P_j(\mathbf{x}^n) \right)}{\partial \sigma_j} = - \sum_n \frac{h_j(\mathbf{x}^n, \mathbf{t}^n)}{P_j(\mathbf{x}^n)} \frac{\partial P_j(\mathbf{x}^n)}{\partial \sigma_j} = - \sum_n h_j^n \left[\frac{||\mathbf{x}^n - \boldsymbol{\mu}_j||^2}{\sigma_j^3} - \frac{d}{\sigma_j} \right].$$

Setting this partial derivative to zero gives, the new estimate for the local variances is:

$$\sigma_j^2 = \frac{1}{d} \frac{\sum_n h_j(\mathbf{x}^n, \mathbf{t}^n) ||\mathbf{x}^n - \boldsymbol{\mu}_j||^2}{\sum_n h_j(\mathbf{x}^n, \mathbf{t}^n)},$$

which completes the M-step for the Gaussian kernels. It is important to note that, although we are actually optimizing the joint probability during training, recall is still based on the original conditional mixture model (28).

The idea of Gaussian kernels has been extended to the expert networks by Fritsch [7], resulting in a model that can be trained by a generalized EM algorithm. Other possible extensions are the use of exponential kernels and modeling the complete covariance matrix [22].

4 Adaptive Variances in Mixtures of Experts

The previous sections focused on MEs as single point estimators that predict the conditional average of the target data. This approach is quite suitable for target data that can be described with an input-dependent mean and one global variance parameter. However, for better data modeling it is often useful to have more higher-order information. One possibility that has been explored is to estimate local error bars (input-dependent variance) for non-linear regression. This gives an estimate of the confidence one can have in a prediction and the possibility to take into account input-dependent noise.

In a maximum-likelihood framework this has been proposed for a single isotropic Gaussian conditional probability density function [15] and has been generalized to an arbitrary covariance matrix [21]. A well-known disadvantage of maximum-likelihood estimations, however, is that it gives biased estimates and leads to under-estimation of the noise variance and overfitting on the training data. Therefore, Bayesian techniques have also been applied [2] which typically avoid these kind of problems.

For MEs it is usual to introduce a local variance for each expert [20], changing (13) to:

$$\phi_j(\mathbf{t}^n|\mathbf{x}^n) = \frac{1}{(2\pi\sigma_j^2)^{(d/2)}} \exp\left(-\frac{||\mathbf{t}^n - \mathbf{y}_j^n(\mathbf{x})||^2}{2\sigma_j^2}\right). \tag{30}$$

The effect of these expert variances is that the model can handle different noise levels which is for instance very useful when dealing with piece-wise stationary time series that switch between different regimes. It has been noted that it reduces overfitting and eases the subdivision of the problem among the experts [20]. The introduction of the expert variances necessitates some small changes in various equations of section 3.1. The weight changes for the expert networks (15) are proportional to:

$$\frac{\partial E'}{\partial a_{jc}} = \pi_j \frac{1}{\sigma_j^2}(y_{jc} - t_c).$$

The additional factor $1/\sigma_j^2$ can be seen as a form of weighted regression in which the focus is on low-noise regions and which discounts high noise regions (outliers for example). The updates for the variances are easily obtained by calculating the partial derivatives $\partial E'/\partial\sigma_j$ (like in (12)) using the definition of ϕ_j including the expert variances (30):

$$\sum_n \frac{\partial E'_n}{\partial \sigma_j} = \sum_n \left(-\frac{g_j^n}{\sum_{i=1}^m g_i^n \phi_i^n} \frac{\partial \phi_j^n}{\partial \sigma_j}\right) = \sum_n \left(-\frac{g_j^n}{\sum_{i=1}^m g_i^n \phi_i^n} \left[\phi_j^n \frac{||\mathbf{t}^n - \mathbf{y}_j^n(\mathbf{x}^n)||^2}{\sigma_j^3} - \frac{d\phi_j^n}{\sigma_j}\right]\right).$$

Using the definition of π_j (7) and setting the partial derivatives to zero, a direct solution (for the batch update) is:

$$\sigma_j^2 = \frac{1}{d}\frac{\sum_n \pi_j^n ||\mathbf{t}^n - \mathbf{y}_j^n(\mathbf{x}^n)||^2}{\sum_n \pi_j^n},$$

the weighted average of the squared errors, with the posteriors π_j^n as weights. Weigend et al. [20] also describe the incorporation of prior belief about the expert variances in a maximum likelihood framework. In order to avoid biased estimates and overfitting the Bayesian approach has also been applied to MEs [18] using ensemble learning.

The estimation of local error bars from the expert variances is straightforward (section 6.4 of [1]) using the definition of MEs (1) and of ϕ_j including the expert variances (30):

$$\sigma(\mathbf{x}) = \sum_j g_j(\mathbf{x}) \left(\sigma_j^2 + ||\mathbf{y}_j(\mathbf{x}) - \mathbf{y}(\mathbf{x})||^2\right).$$

In fact, Bishop [1] follows a more general approach where the expert variances are input-dependent and which allows modeling of conditional probability distributions.

Acknowledgments

The author gratefully acknowledges the Swiss National Science Foundation (FN:21-45621.95) for their support of this research.

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[2] C. M. Bishop and C. S. Qazaz. Regression with input-dependent noise: A Bayesian treatment. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge MA, 1997. MIT Press.

[3] N. P. Bradshaw, A. Duchâteau, and H. Bersini. Global least-squares vs. EM training for the Gaussian mixture of experts. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Artificial Neural Networks - ICANN'97*, number 1327 in Lecture Notes in Computer Science, pages 295–300. Springer-Verlag, Berlin, 1997.

[4] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. In F. Fogelman Soulié and J. Hérault, editors, *Neurocomputing: Algorithms, Architectures, and Applications*, pages 227–236. Springer Verlag, New York, 1990.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.

[6] J. Fritsch, M. Finke, and A. Waibel. Context-dependent hybrid HME/HMM speech recognition using polyphone clustering decision trees. In *Proceedings of ICASSP-97*, 1997.

[7] J. Fritsch. Modular neural networks for speech recognition. Master's thesis, Carnegie Mellon University & University of Karlsruhe, 1996. `ftp://reports.adm.cs.cmu.edu/usr/anon/1996/CMU-CS-96-203.ps.gz`.

[8] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[9] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[10] M. I. Jordan and L. Xu. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8(9):1409–1431, 1995.

[11] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, London, 2nd edition, 1989.

[12] Geoffrey J. McLachlan and Kaye E. Basford. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, Inc., New York, 1988.

[13] Perry Moerland. Mixtures of experts estimate a posteriori probabilities. IDIAP-RR 97-07, IDIAP, Martigny, Switzerland, `ftp://ftp.idiap.ch/pub/reports/1997/rr97-07.ps.gz`, 1997.

[14] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Unpublished manuscript from `ftp://ftp.cs.utoronto.ca/pub/radford/em.ps.Z`, 1993.

[15] A. D. Nix and A. S. Weigend. Learning local error bars for nonlinear regression. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 489–496, Cambridge MA, 1995. MIT Press.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press, 2nd edition, 1992.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1 : Foundations, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[18] S. R. Waterhouse, D. J. C. MacKay, and A. J. Robinson. Bayesian methods for mixtures of experts. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 351–357, Cambridge MA, 1996. MIT Press.

[19] S. R. Waterhouse and A. J. Robinson. Classification using hierarchical mixtures of experts. In *Proceedings 1994 IEEE Workshop on Neural Networks for Signal Processing*, pages 177–186, Long Beach CA, 1994. IEEE Press.

[20] Andreas S. Weigend, Morgan Mangeas, and Ashok N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6:373–399, 1995.

[21] P. M. Williams. Using neural networks to model conditional multivariate densities. *Neural Computation*, 8(4):843–854, 1996.

[22] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 633–640, Cambridge MA, 1995. MIT Press.