Products of Experts

Geoffrey E. Hinton
Gatsby Computational Neuroscience Unit
University College London
17 Queen Square, London WC1N 3AR, U.K.

`http://www.gatsby.ucl.ac.uk/`

Abstract

It is possible to combine multiple probabilistic models of the same data by multiplying the probabilities together and then renormalizing. This is a very efficient way to model high-dimensional data which simultaneously satisfies many different low-dimensional constraints. Each individual expert model can focus on giving high probability to data vectors that satisfy just one of the constraints. Data vectors that satisfy this one constraint but violate other constraints will be ruled out by their low probability under the other expert models. Training a product of models appears difficult because, in addition to maximizing the probabilities that the individual models assign to the observed data, it is necessary to make the models disagree on unobserved regions of the data space: It is fine for one model to assign a high probability to an unobserved region as long as some other model assigns it a very low probability. Fortunately, if the individual models are tractable there is a fairly efficient way to train a product of models. This training algorithm suggests a biologically plausible way of learning neural population codes.

Introduction

Given several different generative models of the same data, a common way to combine them is to use a mixture in which the combined probability distribution is a weighted arithmetic mean of the individual distributions. This is equivalent to assuming an overall generative model in which each data vector is generated by first choosing one of the individual generative models and then allowing that individual model to generate a data vector. Combining models by forming a mixture is attractive because it is easy to fit mixtures of tractable models to data using EM or gradient ascent and, if the individual models differ a lot, the mixture is likely to be a better fit to the true distribution of the data than a random choice among the individual models.

Unfortunately, mixture models are very inefficient in high-dimensional spaces. Consider, for example, the manifold of face images. It takes about 35 real numbers to specify the shape, pose, expression and illumination of a face and, under good viewing conditions, our perceptual systems produce a sharp posterior distribution on this 35-dimensional manifold. This cannot be done using a mixture of models each of which is tuned in the 35-dimensional space because mixing always makes distributions vaguer and each model must already be broadly tuned in order to cover the 35-dimensional space.

An alternative way to combine many individual expert models is to multiply the probabilities together and renormalize. Products of Experts (PoE) have the advantage that they can produce much sharper distributions than the individual expert models. For example, ~~each model can constrain different dimensions in a high-dimensional space and their product will then constrain all of the dimensions~~ (see figure 1). For modeling handwirtten digits, one low-resolution model can generate images that have the approximate overall shape of the digit and other more local models can ensure that small image patches contain segments of stroke with the correct fine structure. For modeling sentences, one expert can ensure that the tenses agree and another can ensure that there is number agreement between the subject and verb.

Fitting a PoE to data appears difficult because it is necessary to compute the derivatives of the partition function that is used in the renormalization. As we shall see, however, these derivatives can often be estimated very easily.

Learning products of experts

We consider individual expert models for which it is tractable to compute the derivative of the log probability of a data vector with respect to the parameters of the model. This includes mixtures of Gaussians, mixtures of a Gaussian and a uniform, and many other popular models including hidden markov models and linear dynamical systems. We combine n individual expert models as follows:

$$p(\mathbf{d}|\theta_1 ... \theta_n) = \frac{\Pi_m p_m(\mathbf{d}|\theta_m)}{\sum_i \Pi_m p_m(\mathbf{c}_i|\theta_m)} \qquad (1)$$

where $\mathbf{d}$ is a data vector in a discrete space, θ_m is all the parameters of individual model m, $p_m(\mathbf{d}|\theta_m)$ is the probability of $\mathbf{d}$ under model m, and i is an index over all possible vectors in the data space. For continuous data spaces the sum is replaced by the appropriate integral.

For an individual expert to fit the data well it must give high probability to the observed data and it must waste as little probability as possible on the rest of the data space. A PoE can fit the data well even if each expert gives high probability to unobserved regions of the data space provided the experts disagree on which unobserved regions are probable.

To fit a PoE to a set of observed *iid* data vectors [1], we need to compute the derivative of the log likelihood of each observed vector, $\mathbf{d}$, under the PoE. This is given by:

$$\frac{\partial \log p(\mathbf{d}|\theta_1 ... \theta_n)}{\partial \theta_m} = \frac{\partial \log p_m(\mathbf{d}|\theta_m)}{\partial \theta_m}$$

$$- \sum_i p(\mathbf{c}_i|\theta_1 ... \theta_n) \frac{\partial \log p_m(\mathbf{c}_i|\theta_m)}{\partial \theta_m} \qquad (2)$$

The second term on the RHS of Eq. 2 is just the expected derivative of the log probability of an expert on fantasy data, $\mathbf{c}$, that is generated from the PoE. So, assuming that each of the individual experts has a tractable derivative, the only difficulty in estimating the derivative of the log probability of the data under the PoE is generating correctly distributed fantasy data. This can be done in various ways. For discrete data it is possible to use rejection sampling: Each expert generates a data vector independently and this process is repreated until all the experts happen to agree. This is typically *very* inefficient. A Markov chain Monte Carlo method that uses Gibbs sampling is typically much more efficient. In Gibbs sampling, each variable draws a sample from its posterior distribution given the current states of the other variables. Given the data, the hidden states of all the experts can always be updated in parallel because they are conditionally independent. If the individual experts have the property that the components of the data vector are conditionally independent given the hidden state of the expert, it is also possible to update all of the components of the data vector in parallel given the hidden states of all the experts. So Gibbs sampling can alternate between parallel updates of the hidden and visible variables. To get an unbiased estimate of the gradient for the PoE it is necessary for the Markov chain to converge to the equilibrium distribution, but in practice very brief Gibbs sampling works remarkably well for reasons discussed at the end of the paper.

[1]For time series models, $\mathbf{d}$ is a whole sequence.

A simple example

PoE's should work very well on data distributions that can be factorized into a product of lower dimensional distributions. This is demonstrated in figure 1. There are 15 "unigauss" experts each of which is a mixture of a uniform and a single, axis-aligned Gaussian. In the fitted model, each tight data cluster is represented by the intersection of two Gaussians which are elongated along different axes. Using a conservative learning rate, the fitting required 2,000 updates of the parameters, but a single Gibbs iteration was sufficient to estimate the derivatives on the fantasy data. For each update of the parameters, the following computation is performed on every observed data vector:

1. Given the data, $\mathbf{d}$, calculate the posterior probability of selecting the Gaussian rather than the uniform in each

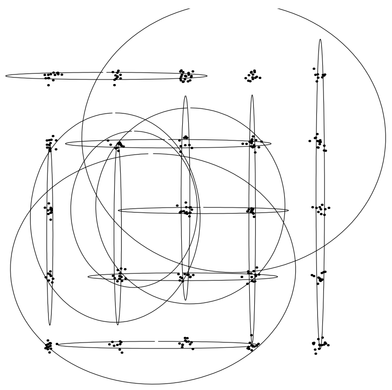

Figure 1: Each dot is a datapoint. The data has been fitted with a product of 15 experts. The ellipses show the one standard deviation contours of the Gaussians in each expert. The experts are initialized with randomly located, circular Gaussians that have about the same variance as the data. The five unneeded experts remain vague, but the mixing proportions of their Gaussians remain high.

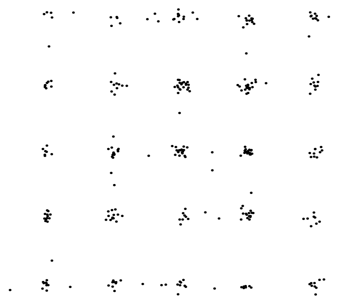

Figure 2: 300 datapoints generated by prolonged Gibbs sampling from the 15 experts fitted in figure 1. The Gibbs sampling started from a random point in the range of the data and used 25 parallel iterations with annealing. Notice that the fitted model generates data at the grid point that is missing in the real data.

expert and compute the first term on the RHS of Eq. 2.

2. For each expert, stochastically select the Gaussian or the uniform according to the posterior. Compute the normalized product of the selected Gaussians, which is itself a Gaussian, and sample from it is used to get a "reconstructed" vector in the data space.

3. Compute the negative term in Eq. 2 using the reconstructed vector as $\mathbf{c}$.

Logarithmic opinion pools

The idea of combining the opinions of multiple different expert models by using a weighted average in the log probability domain is far from new (Genest and Zidek, 1986), but research has focussed on how to find the best weights for combining experts that have already been learned separately rather than training the experts cooperatively. The ~~geometric~~ mean of a set of probability distributions has the attractive property that its Kullback-Liebler divergence from the true distribution, P, is smaller than the ~~average~~ of the Kullback-Liebler divergences of the individual distributions, Q:

$$D\left(P||\frac{\Pi_m Q_m^{w_m}}{Z}\right) \le \sum_m w_m D(P||Q_m) \quad (3)$$

where $\sum_m w_m = 1, \quad Z = \sum_i \Pi_m Q_m^{w_m}(c_i)$. When all of the individual models are identical, $Z = 1$. Otherwise, Z is less than one and the difference between the two sides of (3) is $-\log Z$. This makes it clear that the benefit of combining experts comes from the fact that they make $\log Z$ small by disagreeing on unobserved data.

Learning a population code

A PoE can also be a very effective model when each expert is quite broadly tuned on every dimension and precision is obtained by the intersection of a large number of experts. Figure 3 shows what happens when experts of the type used in the previous example are fitted to 100-dimensional synthetic images that contain edges. The edges varied in their orientation, position, and the intensities on each side of the edge. The intensity profile across the edge was a sigmoid. Each expert also learned a variance for each pixel and although these variances varied, individual experts did not specialize in a small subset of the dimensions. Given an image, about half of the experts have a high probability of picking their Gaussian rather than their

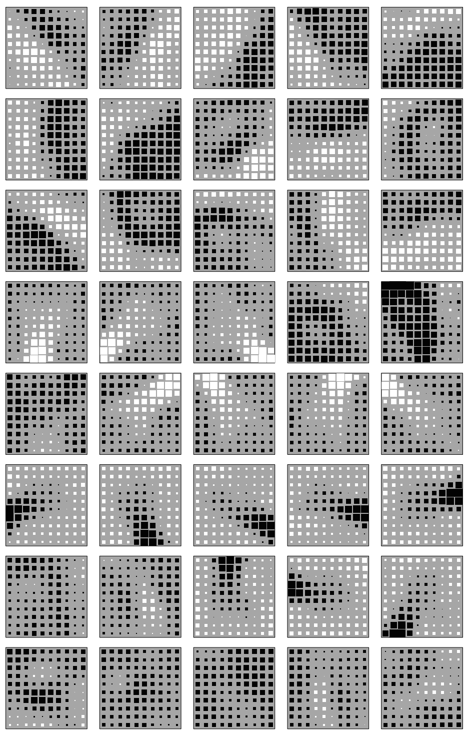

Figure 3: The means of all the 100-dimensional Gaussians in a product of 40 experts, each of which is a mixture of a Gaussian and a uniform. The PoE was fitted to 10×10 images that each contained a single intensity edge. The experts have been ordered by hand so that qualitatively similar experts are adjacent.

uniform. The products of the chosen Gaussians are excellent reconstructions of the image. The experts at the top of figure 3 look like edge detectors in various orientations, positions and polarities. Many of the experts further down have even symmetry and are used to locate one end of an edge. They each work for two different sets of edges that have opposite polarities and different positions. Again, a single Gibbs iteration was sufficient for learning.

Initializing the experts

An efficient way to initialize a PoE is to train each expert separately, forcing the experts to differ by giving them different training cases or by training them on different subsets of the data dimensions, or by using different model classes for the different experts. Once each expert has been initialized separately, the individual probability distributions need to be raised to a fractional power to create the initial PoE. This is not necessarily easy because when the probabilities assigned by an expert model are softened they need to be renormalized which involves summing over the entire data space. Fortunately, an expert does not really need to assign a probability to a data vector. So long as the expert assigns a positive number to each data vector, this number can be normalized to create a probability and since the normalization is the same on both real and fantasy data, the derivative of the log normalization term cancels out. When the experts are being trained cooperatively, the only real requirement on each expert is that it be easy to compute the derivative (with respect to the expert's parameters) of the log of the number that it assigns to a vector in the data space.

Comparison with directed acyclic graphical models

Inference in a PoE is trivial because the experts are individually tractable and the hidden states of different experts are conditionally independent given the data. This makes them relevant as models of biological perceptual systems, which must be able to do inference very rapidly. Alternative approaches based on directed acyclic graphical models suffer from the "explaining away" phenomenon. When such graphical models are densely connected exact inference is intractable, so it is necessary to resort to implausibly slow iterative techniques for approximate inference (Saul and Jordan, 1998) or to use crude approximations that ignore explaining away during inference and rely on the learning algorithm to find representations for which the shoddy inference technique is not too damaging (Hinton et al., 1995).

Unfortunately, the ease of inference in PoE's is balanced by the difficulty of generating fantasy data. This can be done trivially in one ancestral pass in a directed acyclic graphical model but requires an iterative procedure such as Gibbs sampling in a PoE. If, however, brief Gibbs sampling is sufficient for learning, the difficulty of generating unbiased fantasy data is not a major problem.

In addition to the ease of inference that results from the conditional independence of the experts given the data, PoE's have a more subtle advantage over generative models that work by first choosing values for the latent variables and then generating a data vector from these latent values. If such a model has a single hidden layer and the latent variables have independent prior distributions, there will be a strong tendency for the posterior values of the latent variables to be approximately marginally independent after the model has been fitted to data. Any lack of marginal independence can be viewed as a coding inefficiency if the data is communicated by first specifying the states of the latent variables under an (inappropriate) independent prior and then specifying the data given the hidden values. For this reason, there has been little success with attempts to learn such generative models in a greedy bottom-up way. With PoE's, however, even though the experts have independent priors the latent variables in different experts will be marginally *dependent*: They can have high mutual information even for fantasy data generated by the PoE itself. So after the first hidden layer has been learned greedily there may still be lots of statistical structure in the latent variables for the second hidden layer to capture.

PoE's and Boltzmann machines

The Boltzmann machine learning algorithm (Hinton and Sejnowski, 1986) is theoretically elegant but it is very slow in networks with interconnected hidden units and it suffers from strange effects in which the weights are driven away from regions in which the learning signal has zero mean but high variance. In an unsupervised Boltzmann machine with one visible layer, one hidden layer, and no intralayer connections, the probability of generating a visible vector is proportional to the product of the probabilities that the visible vector would be generated by each of the hidden units acting alone. This type of Boltzmann machine is therefore a PoE with one expert per hidden unit.

Inference is tractable in this restricted architecture because the states of the hidden units are conditionally independent given the data. The PoE learning algorithm is exactly equivalent to the Boltzmann learning algorithm in this case. Consider the derivative of the log probability of the data with respect to the weight w_{ij} between a visible unit i and a hidden unit j. Using the notation from Hinton and Sejnowski (1986), the first term on the RHS of Eq 2 is $< s_i s_j >^{+} - < s_i s_j >^{j-}$ and the second term is $< s_i s_j >^{-} - < s_i s_j >^{j-}$, where $< s_i s_j >^{j-}$, which cancels out, denotes the expected value of $s_i s_j$ when the visible units are unclamped and their states are being determined solely by the single expert whose only hidden unit is j.

One major advantage of a PoE over a Boltzmann machine is that the individual experts can be much more complex than a single, symmetrically connected hidden unit. This makes it possible to take advantage of the tractability of quite complicated experts (like hidden markov models) each of which can capture a lot of structure. Furthermore, each expert can be initialised sensibly, so the PoE learning algorithm can *start* from a logarithmic opinion pool of sensible experts.

Why one Gibbs iteration works

In the absence of a single convincing argument that shows why one Gibbs iteration is enough we will have to rely on a product of four less convincing arguments. First, simulations show that it works. In addition to the simulations described above that use unigauss experts, simulations have been run in which each expert is a mixture of many Gaussians or each expert is a Boltzmann machine with one hidden unit. In all simulations, a single Gibbs iteration was sufficient.

The second argument relies on the fact that, in high-dimensional datasets, the data nearly always lies on, or close to, a much lower dimensional, smoothly curved manifold. The PoE needs to find parameters that make a sharp ridge of log probability along the low dimensional manifold. By starting with a point on the manifold and ensuring that this point has higher log probability than the typical reconstructions from the latent variables of all the experts, the PoE ensures that the probability distribution has the right local curvature. It is possible that the PoE will accidentally assign high probability to other distant and unvisited parts of the data space, but this is unlikely if the log probabilty surface is smooth and if both its height and its local curvature are constrained at the data points. It is possible to find and eliminate such points by performing prolonged Gibbs sampling without any data, but this is just a way of improving the learning and not an essential part of it.

The third argument is that the individual experts will still behave quite sensibly even if the estimate of the derivative of the second term in Eq. 2 is just random noise. In this case, the derivative of the first term will force each expert to model the data and the random noise will simply make the experts differ randomly rather than in a more useful and systematic way.

The fourth argument is that a single Gibbs iteration will be a move in the direction of the equilibrium distribution and so it will tend to produce the right sign for the whole RHS of Eq. 2 even if the magnitude is a lot smaller.

Discussion

There have been previous attempts to learn representations by adjusting parameters to cancel out the effects of brief iteration in a recurrent network (Hinton and McClelland, 1987; Seung, 199?), but these were not formulated as approximate gradient descent in a full generative model.

PoE's have been presented as an unsupervised technique. They can also be used for classification by comparing the log probabilities under separate, class-specific PoE's. The normalization term in Eq. 1 is unknown but the difference in the log normalization terms of two PoE's is a single number which can easily be estimated. A PoE in which each of the 100 experts was a Boltzmann machine with a single hidden unit learned an excellent model of a thousand 16×16 real-valued images of the digit 2. The images, from the USPS Cedar ROM, were normalized but highly variable in style. The PoE learned localised features that yielded almost perfect reconstructions. Another PoE was trained on a thousand 3's and the two models were then used to label 200 separate test images. Each test image was represented by two coordinates which were its unnormalized log probabilities under the two models. Using the maximum margin 2-D linear separator on the training data, there were only two errors on test data and both were extremely close to the decision boundary.

It is also possible to define each expert to be a conditional probability model that produces a probability distribution over output vectors when given an input vector. If each expert has latent variables, the PoE learning algorithm can be applied.

PoE's seem to be particularly promising for sequential data because they allow each expert to use a tractable but fairly powerful model. Even for static images, it would be worth exploring the use of sophisticated but tractable individual experts. For example, each expert could be a tree-structured Gaussian belief net. The product of many experts that used different trees would then provide a proper probabilistic image model that would eliminate block boundary artifacts. The belief nets could first be trained separately and then trained together as a PoE.

Acknowledgements

This research was funded by the Gatsby Foundation. Thanks to Zoubin Ghahramani, Peter Dayan, David MacKay, Guy Mayraz. Andy Brown and David Lowe for helpful discussions.

References

Genest, C. & Zidek, J. V. (1986) Combining probability distributions: A critique and an annotated bibliography. *Statistical Science* **1**, 114-148.

Hinton, G., Dayan, P., Frey, B. & Neal, R. (1995) The wake-sleep algorithm for self-organizing neural networks. *Science*, **268**, 1158-1161.

Hinton, G. E. & McClelland, J. L. (1988) Learning representations by recirculation. In D. Z. Anderson, editor, *Neural Information Processing Systems*, 358–366, American Institute of Physics: New York.

Hinton, G. E. & Sejnowski, T. J. (1986) Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, MIT Press

Saul, L. K., Jaakkola, T. & Jordan, M. I. (1996) Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, **4** 61-76.

Seung, H. S. (1998) Learning continuous attractors in a recurrent net. *Advances in Neural Information Processing Systems 10.* M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.) MIT Press, Cambridge Mass.