ICANN'99
Edinburgh, UK

Multi-Layer Perceptrons as Nonlinear Generative Models for Unsupervised Learning: a Bayesian Treatment

Harri Lappalainen
Neural Networks Research Centre
Helsinki University of Technology, Finland
Harri.Lappalainen@hut.fi

Xavier Giannakopoulos
IDSIA
Lugano, Switzerland
xavier@idsia.ch

Abstract

In this paper, multi-layer perceptrons are used as nonlinear generative models. The problem of indeterminacy of the models is resolved using a recently developed Bayesian method called ensemble learning. Using a Bayesian approach, models can be compared according to their probabilities. In simulations with artificial data, the network is able to find the underlying causes of the observations despite the strong nonlinearities of the data.

1 Introduction

Many types of unsupervised learning can be viewed as generative learning where the goal is to find a model which explains how the observations were generated. The hypothesis is that there are latent variables which have generated the observations by an unknown mapping. The goal of the learning is to identify the latent variables and the unknown mapping.

The success of the model depends on how well it can capture the structure of the phenomena underlying the observations. Sometimes the process is well characterised by assuming a discrete latent variable which produces different observations at different states. Then the generative model used in vector quantisation is appropriate. If there is reason to assume that several independent latent variables have generated the observations via a linear mapping, then the model used in independent component analysis suits the problem well.

In many realistic cases it is reasonable to assume that there are several latent variables which affect the observations nonlinearly. One example could be the effect of pressure and temperature on the properties of the end product of a chemical process. Although many effects in real world are locally linear, the overall effects are almost always nonlinear. Also, there are usually several factors whose nature and effect on the observations are completely unknown and whose direct measurement is impossible for practical reasons.

The goal of this work is to develop methods for inferring the hidden causes, the latent variables, from the observations alone. The nonlinear mapping from the unknown latent variables to the observations is modelled with the familiar multi-layer perceptron network (MLP).

2 Bayesian learning

Given the observed data, there are usually more than one way to explain it. With a flexible model family — like MLP-networks — there is always an infinite amount of explanations and it could be difficult to choose among them. Choosing too complex a model would result in overlearning, a situation where one not only finds the underlying causes of the observations but also makes up meaningless explanations for the noise always present in real signals. Choosing too simple a model results in underlearning, i.e., would leave some of the true causes hidden.

The solution to the problem is that no single model should, in fact, be chosen. Probability theory tells that all the explanations should be taken into account and weighted according to their posterior probabilities. This approach, known as Bayesian learning, optimally solves the tradeoff between under- and overlearning.

The posterior probability densities of too simple models are low because they leave much of the data unexplained while the peaks of the posterior probability density function (pdf) of too complex models are high but also very narrow. This is because a complex model is very sensitive to changes in its parameters. Due to the narrow peaks, too complex models occupy little probability mass and therefore contribute little to expectations weighted by the probabilities.

2.1 Parametric approximation of the posterior pdf

In practice, exact treatment of the posterior pdf of the models is impossible and the posterior pdf needs to be approximated. The existing methods for doing this can be roughly divided into stochastic sampling and parametric approximation. Stochastic sampling typically yields better approximations but is also computationally much more expensive. We therefore opt for the computationally efficient parametric approximation which usually yields satisfactory results.

A standard approach for parametric approximation is the Laplace's method. One variation was introduced to the neural networks community by MacKay, who called his method the evidence framework: one first finds a (local) maximum point of the posterior pdf and then applies a second order Taylor's series approximation for the logarithm of the posterior pdf. This amounts to applying the Gaussian approximation to the posterior pdf.

Unfortunately, also Laplace's method can suffer from overlearning. Recall that too complex models have very high posterior probability densities. Therefore finding the maximum point of the posterior pdf focuses the search on too complex models. In the end, the second order Taylor's series approximation will reveal that the peak is narrow, but then it is already too late.

2.2 Ensemble learning

Ensemble learning [3, 6], also known as variational learning, is a recently developed method for parametric approximation of posterior pdfs where the search takes into account the probability *mass* of the models. Therefore, it does not suffer from overlearning. The basic idea is to minimise the misfit between the posterior pdf and its parametric approximation.

Let P denote the exact posterior pdf and Q its parametric approximation. The misfit is measured with the Kullback-Leibler divergence between P and Q and thus the cost function $C_{\mathrm{K}L}$ is

$$C_{\mathrm{K}L} = E_Q \left\{ \log \frac{Q}{P} \right\}. \tag{1}$$

Notice that the Kullback-Leibler divergence involves an expectation over a distribution and, consequently, is sensitive to probability mass rather than probability density.

3 Model structure

The basic model structure is the ordinary MLP network. It has a slight improvement, however, and to motivate this, let us first consider how the learning with generative models looks like in practice.

As usually, we start with a random initialisation of the weights. Then we take the first observed data vector and find those values of the latent variables which best explain the observed data. In vector quantisation, for instance, this is very easy. The latent variable is the index of the model vector which is used for representing the data, and therefore the best value for the latent variable is simply the index of the closest model vector.

Inverting an MLP network is harder, however, and we are going to use gradient descent for doing it. Usually back propagation is used for updating the weights, but it can be used for adapting the unknown inputs, the latent variables as well. Typically the gradient descent has to be iterated several times in order to find the optimal latent variables.

To put the same thing differently, in supervised learning we are presented with the inputs and desired outputs. In our case the latent variables play the role of inputs and the observations play the role of outputs. The difference is that the latent variables are unknown and to find them, the model has to be inverted.

Once the optimal latent variables are found, the inputs of the MLP network are known and the learning proceeds as in supervised learning: the weights are adapted so as to make the mapping from the found latent variables to the observed data even better. Again we can draw a parallel from vector quantisation: in most learning algorithms the best matching model vector is moved even closer to the input.

Then we take the next data vector, find the latent variables that best describe the data by iterating the gradient descent a few times, adapt the weights, and so on.

In the beginning the learning can be slow, however. When the model is random, no values of the latent variables are able to explain much of the data. The optimal latent variables can be more or less random, and without sensible inputs it is difficult to adapt the mapping. It would therefore seem that learning is bound to be slower with unsupervised generative models than with supervised models.

Luckily, it turns out that the situation is much better because the mapping can be learned layer by layer starting from the layers closest to the observations. The point is to create parts of the network only when they may have meaningful input. In this process the model is refined when getting closer to the solution.

In the beginning, only the linear layer is created, together with first layer latent variables which act as training wheels for the network. At this point the mapping is linear and the network quickly finds some meaningful values for the first layer weights.

After the first layer has found a rough representation, the second, nonlinear layer is added on top of the first layer. Since the first layer weights already have reasonable values, the second layer learns much faster. Initially the data is represented mainly by the first layer latent variables, but gradually the second layer latent variables take over and the first layer latent variables become silent.

We can now formalise the model used in this work. Let $\mathbf{x}(t)$ denote the observed data vector at time t; $\mathbf{s}_1(t)$ and $\mathbf{s}_2(t)$ the vectors of latent variables of the first and the second layer at time t; $\mathbf{A}$ and $\mathbf{B}$ the matrices containing the weights on the first and the second layer, respectively; $\mathbf{b}$ the vector of biases for the second layer and $\mathbf{f}$ the vector of nonlinear activation functions. As all real signals contain noise, we shall assume that observations are corrupted by Gaussian noise denoted by $\mathbf{n}(t)$. Using this notation, the model for the observations passes through the three phases described below:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}_1(t) + \mathbf{n}(t) \quad (2)$$

$$\mathbf{x}(t) = \mathbf{A}\left[\mathbf{f}\left(\mathbf{B}\mathbf{s}_2(t) + \mathbf{b}\right) + \mathbf{s}_1(t)\right] + \mathbf{n}(t) \quad (3)$$

$$\mathbf{x}(t) = \mathbf{A}\left[\mathbf{f}\left(\mathbf{B}\mathbf{s}_2(t) + \mathbf{b}\right)\right] + \mathbf{n}(t). \quad (4)$$

The nonlinearity most often used in MLP networks is the hyperbolic tangent, but it has a disadvantage from the point of view of this application. It saturates for large values of its inputs which can cause problems during the inversion of the model. For the sake of numerical stability, the inverse hyperbolic sine, $\sinh^{-1}$, is chosen instead. It is also a sigmoidal function but instead of saturating it behaves logarithmically for large values.

The latent variables are assumed to be independent and Gaussian. The independence assumption is natural as the goal of the model is to find the underlying independent causes of the observations. If the latent variables were dependent, then they would presumably have a common cause which should be modelled by yet another latent variable.

Even the Gaussianity assumption is usually not unrealistic. The network has nonlinearities which can transform the Gaussian distributions to virtually any other regular distribution. This is why with linear models it makes a difference whether the latent variables are assumed to have Gaussian, as in PCA, or non-Gaussian distributions, as in ICA, but for nonlinear models these assumptions do not make such a great difference. It may, of course, sometimes be that an explicit model of a non-Gaussian distribution, e.g., by mixtures of Gaussians as in [6], is simpler than an implicit model with nonlinearities.

The parameters of the network are: (1) the weight matrices $\mathbf{A}$ and $\mathbf{B}$ and the vector of biases $\mathbf{b}$; (2) the parameters of the distributions of the noise, latent variables and column vectors of the weight matrices; and (3), hyperparameters which are used for defining the distributions of the biases and the parameters in the group (2). For simplicity, all the parametrised distributions are assumed to be Gaussian.

This kind of hierarchical description of the distributions of the parameters in the model is a standard procedure in probabilistic modelling. Its strength is that knowledge about equivalent status of different parameters can be easily incorporated. All the variances of the noise components, for instance, have a similar status in the model and this is reflected by the fact that their distributions are assumed to be governed by common hyperparameters. Often there is some vague prior information about the distributions of the hyperparameters, but the amount of information is, in any case, very small compared to the amount of information in the data. Here the hyperparameters are assigned flat priors.

4 Cost function

The cost function was already outlined in section 2.2. We can now go into more detail. Let us denote $X = \{\mathbf{x}(t)|t\}$ and $S = \{\mathbf{s}_1(t), \mathbf{s}_2(t)|t\}$ and let $\boldsymbol{\theta}$ denote all the unknown parameters of the model. For notational simplicity, let us denote all the unknown variables by $\boldsymbol{\xi} = \{S, \boldsymbol{\theta}\}$. The cost function is then

$$C_{\mathrm{KL}} = \int d\boldsymbol{\xi} Q(\boldsymbol{\xi}) \log \frac{Q(\boldsymbol{\xi})}{P(\boldsymbol{\xi}|X)}. \tag{5}$$

The two things needed for equation 5 are the exact formulation of the posterior density $P(\boldsymbol{\xi}|X)$ and its parametric approximation $Q(\boldsymbol{\xi})$.

According to the Bayes' rule, the posterior pdf of the unknown variables S and $\boldsymbol{\theta}$ is

$$P(S, \boldsymbol{\theta}|X) = \frac{P(X|S, \boldsymbol{\theta})P(S|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(X)}.$$

The term $P(X|S, \boldsymbol{\theta})$ is obtained from equations 2–4; the distribution of the data is the same as for the noise $\mathbf{n}(t)$ except that its mean is corrected by the prediction given by the model. Let us denote the vector of the means of $\mathbf{n}(t)$ by $\boldsymbol{\mu}$ and the vector of the variances by $\boldsymbol{\sigma}^2$. The distribution $P(x_i(t)|\mathbf{s}_1(t), \mathbf{s}_2(t), \boldsymbol{\theta})$ is thus Gaussian with mean $\mathbf{a}_{i\cdot}[\mathbf{f}(\mathbf{B}\mathbf{s}_2 + \mathbf{b}) + \mathbf{s}_1] + \mu_i$ and variance σ_i^2. Here $\mathbf{a}_{i\cdot}$ denotes the ith row vector of $\mathbf{A}$. As usually, the noise components $n_i(t)$ are assumed to be independent and therefore $P(X|S, \boldsymbol{\theta}) = \prod_{t,i} P(x_i(t)|\mathbf{s}_1(t), \mathbf{s}_2(t), \boldsymbol{\theta})$.

The terms $P(S|\boldsymbol{\theta})$ and $P(\boldsymbol{\theta})$ are also products of simple Gaussian distributions and they are obtained directly from the definition of the model structure. The term $P(X)$ is not a function of any of the parameters of the model and can be neglected.

The approximation $Q(S, \boldsymbol{\theta})$ needs to be simple for mathematical tractability and computational efficiency. We assume that it is Gaussian density with a diagonal covariance matrix. This means that the approximation is a product of the independent distributions: $Q(\boldsymbol{\xi}) = \prod_i Q_i(\xi_i)$. The parameters of each $Q_i(\xi_i)$ are the mean and variance which will be denoted by $\hat{\xi}_i$ and $\tilde{\xi}_i$, respectively.

Both the posterior density $P(S, \boldsymbol{\theta}|X)$ and its approximation $Q(S, \boldsymbol{\theta})$ are products of simple Gaussian terms, which simplifies the cost function considerably: it splits into expectations of many simple terms. The terms of the form $E_Q\{\log Q_i(\xi_i)\}$ are the negative entropies for Gaussians and have the values $-(1 + \log 2\pi\tilde{\xi}_i)/2$. The most difficult terms are of the form $-E_Q\{\log P(x_i(t)|\mathbf{s}_1(t), \mathbf{s}_2(t), \boldsymbol{\theta})\}$. They are approximated by applying the second order Taylor's series expansions of the nonlinear activation functions as explained in [5]. The rest of the terms are expectations of simple Gaussian terms, whose expectations can be computed as in [6].

The cost function C_{KL} is a function of $\hat{\xi}_i$ and $\tilde{\xi}_i$, i.e., the posterior means and variances of the latent variables and the parameters of the network. This is because instead of finding a point estimate, a whole distribution will be estimated for the latent variables and the parameters during learning.

5 Simulations

5.1 Data

Artificial data was generated with a randomly initialised MLP network having one hidden layer with tanh-nonlinearities and a linear output layer. The network had a 2-10-5 structure and the data thus consisted of a two-dimensional manifold nonlinearly wrapped in five dimensions. The inputs had Gaussian distributions with unit covariance matrix. The five-dimensional outputs of the random network were further linearly embedded in a 10-dimensional space and finally, Gaussian noise was added to the data which consisted of 1000 vectors.

5.2 Learning scheme

Learning was done in batches which had two nested loops. The outer loop went through all the data vectors. For each data vector, the posterior distribution Q of the latent variables was adapted in the inner loop of length 15. The distributions of the rest of the model parameters were updated at the end of each batch. The whole learning consisted of 200 batches.

Only the first, linear layer was generated in the beginning. The second, nonlinear layer was generated after 20 batches when the first layer had already found a rough representation for the data. To encourage the growth of the second layer, the standard deviations of the latent variables of the first layer were reduced by a factor of three after each batch during 10 batches starting from the creation of the second layer. If this phase is left out, the network easily gets stuck in a local minimum where the training wheels, the latent variables of the first layer, represent the data while the second layer remains silent.

The posterior variance $\tilde{\theta}_i$ of a parameter contains information about how certain the network is about the value of the parameter. This gives the network the very interesting property of being able to effectively prune away useless weights by increasing their posterior variances and thus decreasing the complexity of the network.

Some care has to be taken in the beginning of the learning since the network might get stuck in some unwanted local minima. When the weights of the network are random and the network is not able to find any structure in the data, the weights can get prematurely pruned away. To prevent this, the posterior variances of the weights and the latent variables were bounded for the first 50 batches.

5.3 Results

Several different structures were tested for the network and the best one was chosen according to the probability mass it occupied in the posterior pdf of all models. The most probable model had four latent variables and ten hidden units. Despite the difficulty of the problem, the network was able to discover the underlying causes of the observations. Out of the four latent variables,

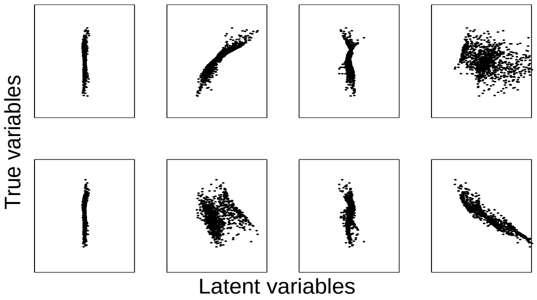

Figure 1: Scatter plots of the two original inputs and the posterior means of the four estimated latent variables.

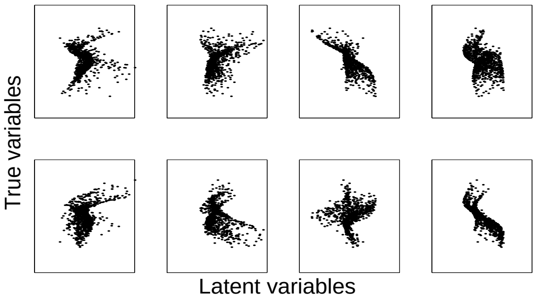

Figure 2: Scatter plots of the original inputs and the four sources estimated by linear ICA.

two corresponded to the original inputs that had generated the data while the other two had much smaller variances and were used by the network to represent the slight discrepancies between the original and the estimated nonlinear generative models. Notice that the two models used different nonlinearities and therefore can never be exactly the same.

Figure 1 shows the scatter plots of the four estimated latent variables (x-axes) versus the two original inputs (y-axes). The first original input (upper row) correlated with the second latent variable while the second original input correlated with the fourth latent variable. The figure shows that the first and third latent variables had much smaller variance than the two others.

For comparison, four independent components extracted by linear ICA [1] are shown in similar scatter plots in figure 2. As the data is severely nonlinear, linear ICA performs very poorly. None of the retrieved sources correspond to the original inputs and all four sources are used for representing the data. These results show that the nonlin-

[1]The FastICA MATLAB-toolbox, available at `http://www.cis.hut.fi/projects/ica/fastica/`.

earity was quite strong and, consequently, the problem of finding the underlying latent variables very difficult.

6 Discussion

The combination of MLP networks, generative learning and full Bayesian analysis is novel — to the best of our knowledge — although the individual parts have been published earlier. For instance, MLP networks were used as generative models in [7]. The model was inverted by gradient descent as in this work, but Bayesian analysis was not applied.

Rectified Gaussian belief networks were used as generative models in [2], but the Bayesian analysis was restricted to the posterior distributions of the latent variables and stochastic sampling was used instead of parametric approximation. Also [1] neglects the Bayesian treatment of the parameters of the network. With flexible models having a large number of parameters, it is important to take into account also the complexity of the nonlinear mapping. Restricting the Bayesian approach to the latent variables can lead to problems with overlearning.

Methods based on minimising the description length of the model are closely related, often equivalent, to Bayesian learning [3, 4, 6] since the description length is by definition the minus logarithm of the probability mass of the model. The description length of an auto-associative MLP model was minimised in [4]. The disadvantage of auto-associative models is that they need to learn both the generative mapping and its inversion and the learning can thus be slow.

7 Conclusion

Using MLP networks as generative models was proven feasible in simulations with artificial data. The network was able to retrieve the original inputs which had generated the data. The difficulty of the problem is apparent from the results obtained with the linear ICA-model. Bayesian learning was used for solving the indeterminacy of the unknown mapping.

The Bayesian approach is particularly valuable for unsupervised learning due to its robustness against overlearning and the ability to compare models. Other techniques, such as cross-validation, are available for supervised learning but they are not applicable for unsupervised learning.

The Bayesian approach was implemented using ensemble learning, which is an efficient method for approximating the posterior distributions. Its main advantage over the traditional Laplace's method is that the search for good models is focused on those areas of the model space which occupy large probability mass, as opposed to searching for large probability densities.

References

[1] Hagai Attias. Hierarchical ICA belief networks. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999. In press.

[2] Zoubin Ghahramani and Geoffrey E. Hinton. Hierarchical non-linear factor analysis and topographic maps. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 486–492. MIT Press, 1998.

[3] Geoffrey E. Hinton and Drew van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the COLT'93*, pages 5–13, Santa Cruz, California, 1993.

[4] Sepp Hochreiter and Jürgen Schmidhuber. LOCOCODE performs nonlinear ICA without knowing the number of sources. In *Proceedings of the ICA'99*, pages 149–154, Aussois, France, 1999.

[5] Harri Lappalainen. Using an MDL-based cost function with neural networks. In *Proceedings of the IJCNN'98*, pages 2384–2389, Anchorage, Alaska, 1998.

[6] Harri Lappalainen. Ensemble learning for independent component analysis. In *Proceedings of the ICA'99*, pages 7–12, Aussois, France, 1999.

[7] Jong-Hoon Oh and H. Sebastian Seung. Learning generative models with the up-propagation algorithm. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 605–611. MIT Press, 1998.