# 1
# Density Networks

**David J.C. MacKay and Mark N. Gibbs**
*Cavendish Laboratory, Madingley Road,*
*Cambridge, CB3 0HE. United Kingdom.*
mackay@mrao.cam.ac.uk, mng10@mrao.cam.ac.uk

A density network is a neural network that maps from unobserved inputs to observable outputs. The inputs are treated as latent variables so that, for given network parameters, a non–trivial probability density is defined over the output variables. This probabilistic model can be trained by various Monte Carlo methods. The model can discover a description of the observed data in terms of an underlying latent variable space of lower dimensionality. We review results of the application of these models to toy problems with categorical and real–valued observables and to protein data.

## 1 Density Modelling

The most popular supervised neural networks, multilayer perceptrons, are well established as probabilistic models for *regression* and *classification*, both of which are *conditional* modelling tasks: the *input* variables are assumed given, and we *condition* on their values when modelling the distribution over the *output* variables; no model of the density over input variables is constructed. Density modelling (or generative modelling), on the other hand, denotes modelling tasks in which a density over *all* the observable quantities is constructed. Multi–layer perceptrons have not conventionally been used to create density models (though belief networks and other neural networks such as the Boltzmann machine do define density models). This paper discusses how one can use an multilayer perceptron as a density model. This definition of a full probabilistic model with a multilayer perceptron may prove also useful for other interesting problems, for example, the 'missing inputs' problem (Tresp, Ahmad and Neuneier 1994).

### 1.1 Traditional density models

A popular class of density models are *mixture models*, which define the probability distribution over observables $\mathbf{t}$ as a sum of simple densities. These models are 'latent variable' models (Everitt 1984); each observation $\mathbf{t}^{(n)}$ has an associated *categorical* latent variable $c^{(n)}$ that states which *class* of the mixture that observation comes from. These latent variables $\{c^{(n)}\}$ are not observed, but their

values are inferred during the modelling process.

Mixture models might however be viewed as inappropriate models for high–dimensional data spaces such as images or genome sequences. If we imagine modelling a sequence of densities with increasing numbers of independent degrees of freedom, the number of required components in a mixture model has to scale exponentially. Consider, for example, a protein family in which there are two independent correlations: one pair of amino acids in the protein chain are either both hydrophobic, or both hydrophilic, say, and two other amino acids elsewhere in the chain have anticorrelated size. A mixture model would have to use four categories to capture the four valid combinations of these binary attributes, whereas only two independent degrees of freedom are really present. Thus a *combinatorial* representation of underlying variables would seem more appropriate. [Luttrell's (1994) partitioned mixture distribution is motivated similarly, but is a different form of quasi–probabilistic model.]

These observations motivate the development of density models that have *components* rather than *categories* as their latent variables (Hinton and Zemel 1994). Let us denote the observables by $\mathbf{t}$. If a density is defined on the latent variables $\mathbf{x}$, and a parameterized mapping is defined from these latent variables to a probability distribution over the observables $P(\mathbf{t}|\mathbf{x},\mathbf{w})$, then when we integrate over the unknowns $\mathbf{x}$, a non–trivial density over $\mathbf{t}$ is defined, $P(\mathbf{t}|\mathbf{w}) = \int d\mathbf{x}\, P(\mathbf{t}|\mathbf{x},\mathbf{w})P(\mathbf{x})$. Simple linear models of this form in the statistics literature are called 'factor analysis' models. In a 'density network' (MacKay 1995a) $P(\mathbf{t}|\mathbf{x},\mathbf{w})$ is defined by a more general non–linear parameterized mapping, and interesting priors on $\mathbf{w}$ may be used.

This paper reviews work on density networks where the observables are categorical (MacKay 1995a, MacKay 1996), and describes preliminary research on the problem of real observables. In section 2 the model is defined. In section 3 an implementation of the model using a crude importance sampling method is reviewed, and its application to modelling of protein sequences is described in section 4. In section 5, a more sophisticated implementation using the hybrid Monte Carlo method (Neal 1993) is described and applied to a toy problem with real–valued observables.

## 2 The Density Network Model

The 'latent inputs' of the model form a vector $\mathbf{x}$ indexed by $h = 1 \ldots H$ ('$h$' is mnemonic for 'hidden'). The dimensionality of this hidden space is $H$ but the effective dimensionality of the density in the output space may be smaller, as some of the hidden dimensions may be effectively unused by the model. The relationship between the latent inputs and the observables has the form of a mapping from inputs to outputs $\mathbf{y}(\mathbf{x};\mathbf{w})$, parameterized by $\mathbf{w}$, and a probability of targets given outputs, $P(\mathbf{t}|\mathbf{y})$. The observed data are a set of target vectors $D = \{\mathbf{t}^{(n)}\}_{n=1}^{N}$. To complete the model we assign a prior $P(\mathbf{x})$ to the latent inputs (an independent prior for each vector $\mathbf{x}^{(n)}$) and a prior $P(\mathbf{w})$ to the unknown parameters. [In the applications that follow the priors over $\mathbf{w}$ and $\mathbf{x}^{(n)}$ are

assumed to be spherical Gaussians with widths controlled by hyperparameters $\alpha$; other distributions could easily be implemented and compared, if desired.] In summary, the probability of everything is:

$$P(D, \{\mathbf{x}^{(n)}\}, \mathbf{w}|\mathcal{H}) = \prod_n \left[P(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}, \mathcal{H})P(\mathbf{x}^{(n)}|\mathcal{H})\right] P(\mathbf{w}|\mathcal{H}). \tag{2.1}$$

It will be convenient to define 'error functions' $G^{(n)}(\mathbf{x}; \mathbf{w})$ as follows:

$$G^{(n)}(\mathbf{x}^{(n)}; \mathbf{w}) \equiv \log P(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}) \tag{2.2}$$

The function $G$ depends on the nature of the problem. If $\mathbf{t}$ consists of real variables then $G$ might be a sum–squared error between $\mathbf{t}$ and $\mathbf{y}$; in a 'softmax' classifier where the observations $\mathbf{t}$ are categorical, In general we may have many output groups of different types. The following derivation applies to all cases.

As an example, the following one–layer model may be useful to have in mind. Each observation $\mathbf{t} = \{t_s\}_{s=1}^S$ (*e.g.*, a single protein sequence) consists of $S$ categorical attributes that are believed to be correlated ($S$ will be the number of columns in the protein alignment). Each attribute can take one of $I$ discrete values (*e.g.*, $I = 20$), a probability over which is modelled with a softmax distribution,

$$P(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \prod_{s=1}^{S} \left\{ y_{t_s}^{(s)}(\mathbf{x}; \mathbf{w}) \right\}, \tag{2.3}$$

where

$$y_i^{(s)}(\mathbf{x}; \mathbf{w}) = \exp(a_i^{(s)}(\mathbf{x}; \mathbf{w})) \Big/ \textstyle\sum_{i'} \exp(a_{i'}^{(s)}(\mathbf{x}; \mathbf{w})) \ . \tag{2.4}$$

The parameters $\mathbf{w}$ form a matrix of $(H + 1) \times (S \times I)$ weights from the $H$ latent inputs $\mathbf{x}$ (and one bias unit) to the $S \times I$ outputs:

$$a_i^{(s)}(\mathbf{x}; \mathbf{w}) = w_{i0}^{(s)} + \sum_{h=1}^{H} w_{ih}^{(s)} x_h \tag{2.5}$$

The parameter $w_{i0}^{(s)}$ is called the bias of output unit $(s, i)$. The data items $\mathbf{t}$ and their associated latent inputs $\mathbf{x}$ are labeled by an index $n = 1 \ldots N$, not included in the above equations, and the error function $G^{(n)}$ is

$$G^{(n)}(\mathbf{x}^{(n)}; \mathbf{w}) = \sum_s \log y_{t_s^{(n)}}(\mathbf{x}^{(n)}; \mathbf{w}). \tag{2.6}$$

### 2.1 Learning in a density network

Having written down the probability of everything (equation 2.1) we can now make any desired inferences by turning the handle of probability theory. Let us aim towards the inference of the parameters $\mathbf{w}$ given the data $D$, $P(\mathbf{w}|D, \mathcal{H})$. We can obtain this quantity conveniently by distinguishing two levels of inference.

**Level 1**: Given $\mathbf{w}$ and $\mathbf{t}^{(n)}$, infer $\mathbf{x}^{(n)}$. The posterior distribution of $\mathbf{x}^{(n)}$ is

$$P(\mathbf{x}^{(n)}|\mathbf{t}^{(n)},\mathbf{w},\mathcal{H}) = \frac{P(\mathbf{t}^{(n)}|\mathbf{x}^{(n)},\mathbf{w},\mathcal{H})P(\mathbf{x}^{(n)}|\mathcal{H})}{P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H})}, \tag{2.7}$$

where the normalizing constant is:

$$P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H}) = \int d^H\mathbf{x}^{(n)}\ P(\mathbf{t}^{(n)}|\mathbf{x}^{(n)},\mathbf{w},\mathcal{H})P(\mathbf{x}^{(n)}|\mathcal{H}). \tag{2.8}$$

**Level 2**: Given $D = \{\mathbf{t}^{(n)}\}$, infer $\mathbf{w}$.

$$P(\mathbf{w}|D,\mathcal{H}) = \frac{P(D|\mathbf{w},\mathcal{H})P(\mathbf{w}|\mathcal{H})}{P(D|\mathcal{H})} \tag{2.9}$$

The data–dependent term here is a product of the normalizing constants of the level 1 inferences:

$$P(D|\mathbf{w},\mathcal{H}) = \prod_{n=1}^{N} P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H}). \tag{2.10}$$

The evaluation of the evidence $P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H})$ for a particular $n$ is a problem similar to the evaluation of the evidence for a supervised neural network (MacKay 1992). There, the inputs $\mathbf{x}$ are given, and the parameters $\mathbf{w}$ are unknown; we obtain the evidence by integrating over $\mathbf{w}$. In the present problem, on the other hand, the hidden vector $\mathbf{x}^{(n)}$ is unknown, and the parameters $\mathbf{w}$ are conditionally fixed. For each $n$, we wish to integrate over $\mathbf{x}^{(n)}$ to obtain the evidence.

### 2.2 The derivative of the evidence with respect to w

The derivative of the log of the evidence (equation 2.8) is:

$$\begin{aligned}\frac{\partial}{\partial\mathbf{w}}\log P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H}) &= \frac{1}{P(\mathbf{t}^{(n)}|\mathbf{w},\mathcal{H})}\int d^H\mathbf{x}^{(n)}\ \exp(G^{(n)})P(\mathbf{x}^{(n)}|\mathcal{H})\frac{\partial}{\partial\mathbf{w}}G^{(n)} \\ &= \int d^H\mathbf{x}^{(n)}\ P(\mathbf{x}^{(n)}|\mathbf{t}^{(n)},\mathbf{w},\mathcal{H})\frac{\partial}{\partial\mathbf{w}}G^{(n)}(\mathbf{x}^{(n)};\mathbf{w}). \end{aligned}\tag{2.11}$$

This gradient can thus be written as an expectation of the traditional 'backpropagation' gradient $\frac{\partial}{\partial\mathbf{w}}G^{(n)}(\mathbf{x};\mathbf{w})$, averaging over the posterior distribution of $\mathbf{x}^{(n)}$ found in equation (2.7).

### 2.3 Higher levels — priors on w

We can continue up the hierarchical model, putting a prior on $\mathbf{w}$ with hyperparameters $\{\alpha\}$ which are inferred by integrating over $\mathbf{w}$. These priors are important from a practical point of view to limit overfitting of the data by the parameters $\mathbf{w}$. The complexity of a model is controlled by, among other factors, the number of latent variables that make a significant contribution to the generative model; with a hierarchical prior on $\mathbf{w}$ making use of hyperparameters $\{\alpha\}$, we can automatically infer the appropriate complexity. These priors will also be used to bias the solutions towards ones that are easier for humans to interpret.

## 3 A Simple Monte Carlo Implementation

The evidence and its derivatives with respect to $\mathbf{w}$ both involve integrals over the hidden components $\mathbf{x}$. For a hidden vector of sufficiently small dimensionality, a simple Monte Carlo approach to the evaluation of these integrals can be effective. We use importance sampling with the sampler being defined by the prior, $P(\mathbf{x})$.

Let $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ be random samples from $P(\mathbf{x})$. Then we can approximate the log evidence by:

$$\log P(\{\mathbf{t}^{(n)}\}|\mathbf{w},\mathcal{H}) = \sum_n \log \int d^H\mathbf{x}\ \exp(G^n(\mathbf{x};\mathbf{w}))P(\mathbf{x}) \tag{3.1}$$

$$\simeq \sum_n \log \left[\frac{1}{R}\sum_r \exp(G^n(\mathbf{x}^{(r)};\mathbf{w}))\right]. \tag{3.2}$$

Similarly the derivative can be approximated by:

$$\frac{\partial}{\partial \mathbf{w}} \log P(\{\mathbf{t}^{(n)}\}|\mathbf{w},\mathcal{H}) \simeq \sum_n \frac{\sum_r \exp(G^n(\mathbf{x}^{(r)};\mathbf{w}))\frac{\partial}{\partial \mathbf{w}}G^n(\mathbf{x}^{(r)};\mathbf{w})}{\sum_r \exp(G^n(\mathbf{x}^{(r)};\mathbf{w})).} \tag{3.3}$$

This simple Monte Carlo approach loses the advantage that we gained when we rejected mixture models and turned to componential models: this implementation requires a number of samples $R$ that is exponential in the dimension of the hidden space $H$. More sophisticated methods using stochastic dynamics (Neal 1993) are described in section 5.

## 4 Modelling a Protein Family

A protein is a sequence of *residues*; each residue is one of the twenty amino acids. A protein family is a set of proteins believed to have the same physical structure but not necessarily having the same sequence of amino acids. In a *multiple sequence alignment*, residues of the individual sequences which occupy structurally analogous positions are aligned into columns. Columns can often be characterized by a predominance of particular amino acids. Lists of marginal frequencies over amino acids in different structural contexts are given in (Nakai, Kidera and Kanehisa 1988). Such frequencies correspond to a first order description of a protein family in which correlations between residues are not modelled.

The development of models for protein families is useful for two reasons. The first is that a good model might be used to identify new members of an existing family, and discover new families, in data produced by genome sequencing projects. The second reason is that a sufficiently complex model might be able to give new insight into the properties of the protein family; for example, properties of the proteins' tertiary structure might be elucidated by a model capable of discovering suspicious inter–residue correlations.

The principal probabilistic model that has been applied to protein families is a hidden Markov model (HMM) (Krogh, Brown, Mian, Sjolander and Haussler 1994). Assuming that each state of the HMM corresponds to one column of a

multiple sequence alignment, this model is not inherently capable of discovering long–range correlations, as Markov models, by definition, produce no correlations between the observables, given a hidden state sequence.

The next–door neighbour of proteins, RNA, has been modelled with a 'covariance model' capable of capturing correlations between base–pairs in anti–parallel RNA strands (Eddy and Durbin 1994). Density networks offer a model capable of discovering general correlations between multiple arbitrary columns in a protein family. Steeg (1997) has developed an efficient statistical test for discovering correlated groups of residues. The density network approach is complementary to Steeg's in that (1) in the density network, a residue may be influenced by more than one latent variable; whereas Steeg's test is specialized for the case where the correlated groups are non–overlapping; (2) the density networks developed here define full probabilistic models rather than statistical tests.

Here we model the protein families using a density network containing one softmax group for each column (see equations 2.3–2.6). The network has only one layer of weights connecting the latent variables $\mathbf{x}$ directly to the softmax groups. We have optimized $\mathbf{w}$ by evaluating the evidence and its gradient and feeding them into a conjugate gradient routine. The random points $\{\mathbf{x}^{(r)}\}$ are kept fixed, so that the objective function and its gradient are deterministic functions during the optimization. This also has the advantage of allowing one to get away with a smaller number of samples $R$ than might be thought necessary, as the parameters $\mathbf{w}$ can adapt to make the best use of the empirical distribution over $\mathbf{x}$.

### 4.1 Regularization schemes

A human prejudice towards comprehensible solutions gives an additional motivation for regularizing the model, beyond the usual reasons for having priors. Here we encourage the model to be comprehensible in two ways:

(1) There is a redundancy in the model regarding where it gets its randomness from. Assume that a particular output is actually random and uncorrelated with other outputs. This could be modelled in two ways: its weights from the latent inputs could be set to zero, and the biases could be set to the log probabilities; or alternatively the biases could be fixed to arbitrary values, with appropriate connections to unused latent inputs being used to create the required probabilities, on marginalization over the latent variables. In predictive terms, these two models would be identical, but we prefer the first solution, finding it more intelligible. To encourage such solutions we use a prior which only very weakly regularizes the biases, so that they are 'cheap' relative to the other parameters.

(2) If the distribution $P(\mathbf{x})$ is rotationally invariant, then the predictive distribution is invariant under corresponding transformations of the parameters $\mathbf{w}$. If a solution can be expressed in terms of parameter vectors aligned with some of the axes (*i.e.* so that some parameters are zero), then we would prefer that representation. Here we create a non–spherical prior on the parameters by using multiple undetermined regularization constants

**Table 1** Toy data for a protein family

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EEAB | EECB | EEBC | EECC | EEAA | EEBA | EEBB | EECD |
| EEDC | EEDD | AACD | DDDC | CBDD | CCAB | BDCB | ABBC |
| CBCC | EDAA | ABBA | BCBB | DBAB | AECB | EBBC | BDCC |
| BCAA | DABA | BCBB | | | | | |

$\{\alpha_c\}$, each one associated with a class of weights (*c.f.* the automatic relevance determination model (Neal 1996, MacKay 1994, MacKay 1995b)). A weight class consists of all the weights from one latent input to one softmax group, so that for a protein with $S$ columns modelled using $H$ latent variables, we introduced $SH$ regularization constants, each specifying whether a particular latent variable has an influence on a particular column. Given $\alpha_c$, the prior on the parameters in class $c$ is Gaussian with variance $1/\alpha_c$. This prior favours solutions in which one latent input has non–zero connections to all the units in some softmax groups (corresponding to small $\alpha_c$), and negligible connections to other softmax groups (large $\alpha_c$). The resulting solutions can easily be interpreted in terms of correlations between columns.

### 4.2 Method for optimization of hyperparameters

For given values of $\{\alpha_c\}$, the parameters $\mathbf{w}$ were optimized to maximize the posterior probability. The hyperparameters $\{\alpha_c\}$ were adapted during the optimization of the parameters $\mathbf{w}$ using a cheap and cheerful method motivated by Gaussian approximations (MacKay 1992), thus:

$$\alpha_c := f \frac{k_c}{\sum_{i \in c} w_i^2}. \tag{4.1}$$

Here $k_c$ is the number of parameters in class $c$ and $f$ is a 'fudge factor' incorporated to imitate the effect of integrating over $\mathbf{w}$ (set to a value between 0.1 and 1.0).

This algorithm could be converted to a correct 'stochastic dynamics' Monte Carlo method (Neal 1993) by adding an appropriate amount of noise to gradient descent on $\mathbf{w}$ and setting $f = 1$.

### 4.3 Toy data

A toy data set was created imitating a protein family with four columns each containing one of five amino acids (A–E). The 27 data (table 1) were constructed to exhibit two correlations between the columns: the first and second columns have a tendency both to be amino acid E together. The third and fourth columns are correlated such that if one is amino acid A, then the other is likely to be A or B; if one is amino acid B, then the other is likely to be A, B or C; if one is C, then the other is likely to be B, C or D; and if one is D then the other is likely to be C or D; so that a single underlying dimension runs through the amino acids A,B,C,D (E doesn't occur in these two columns). The model is given no prior

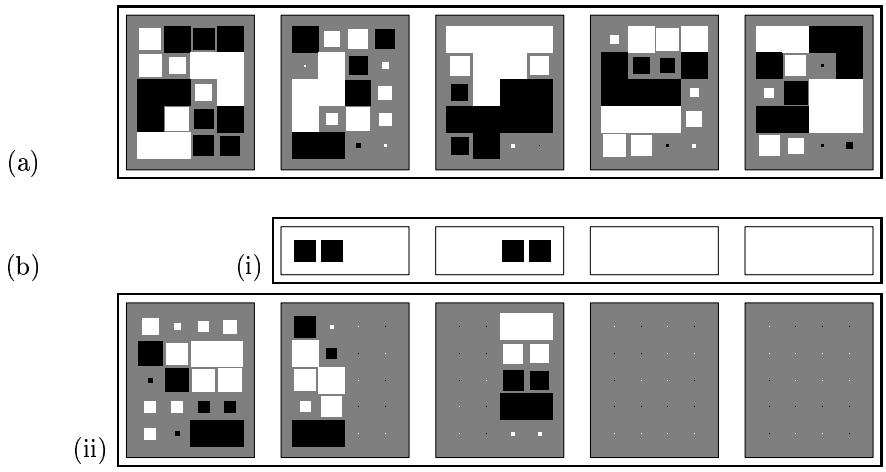


FIG. 1. Parameters and Hyperparameters inferred for the toy protein family (a) without regularization; (b) with adaptive regularizers. (a) Hinton diagram showing parameters $\mathbf{w}$ of model optimized without adaptive regularizers. Positive parameters are shown by black squares, negative by white. Magnitude of parameter is proportional to square's area. This diagram shows, in the five grey rectangles, the *projective fields* from the bias and the four latent variables to the outputs. In each grey rectangle the influences of one latent variable on the twenty outputs are arranged in a 5×4 grid: in each column the 5 output units correspond to the 5 amino acids. It is hard to interpret these optimized parameters. (b) The hyperparameters (i) and parameters (ii) of a hierarchical model with adaptive regularizers. The results are more intelligible and show a model that has discovered the two dimension that underlie the data. Hyperparameters: Each hyperparameter controls all the influences of one latent variable on one column. Square size denotes the value of $\sigma_w^2 = 1/\alpha$ on a log scale from 0.001 to 1.0. The model has discovered that columns 1 and 2 are correlated with each other but not with columns 3 and 4, and vice versa. Parameters: Note the sparsity of the connections, making clear the two distinct underlying dimensions of this protein family.

knowledge of the 'spatial relationship' of the columns, or of the ordering of the amino acids. A model that can identify the two correlations in the data is what we are hoping for.

Both regularized and unregularized density networks having four latent inputs were adapted to this data. Unregularized density networks give solutions that successfully predict the two correlations, but the parameters of those models are hard to interpret (fig. 1(a)). There is also evidence of overfitting of the data leading to overconfident predictions by the model. The regularized models,

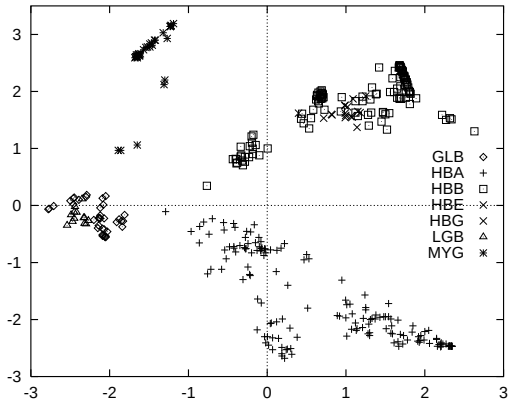


FIG. 2. Two–dimensional representation of globins. From (MacKay 1995a).

in which all the parameters connecting one input to one softmax group are put in a regularization class with an unknown hyperparameter $\alpha_c$, give interpretable solutions that clearly identify the two correlated groups of columns. Figure 1(b) shows the hyperparameters and parameters inferred in a typical solution using a regularized density network. Notice that two of the latent inputs are unused in this solution. Of the other two inputs, one has an influence on columns 1 and 2 only, and the other has an influence on columns 3 and 4 only. Thus this model has successfully revealed the underlying 'structure' of the proteins in this family. The parameters are straightforward to interpret also; the first latent variable, when negative, makes E more probable in columns 1 and 2, and makes all other amino acids less probable, with the possible exceptions `AExx` and `EBxx` having been memorized; the second latent variable's parameters clearly show the one dimensional ordering of the amino acids A, B, C, D.

### 4.4 Results on real data: globins

In MacKay (1995a) a rather ambitious attempt was made to model the joint density of the amino acids in an entire aligned protein sequence using a low–dimensional latent variable space.

Data describing 400 proteins in the globin family was received in aligned form courtesy of Sean Eddy (modelling of unaligned data is possible in principle, but harder), with $S = 208$ columns each containing one of $I = 21$ symbols (twenty amino acids and 'deletion'), or else a 'no measurement' symbol. The density network maps from $H$ latent inputs to $SI$ outputs, grouped in $S$ softmax groups of $I$ units each. With $H = 20$ latent dimensions this model has about 80,000 parameters. The special case of $H = 0$ latent inputs creates a model with independent probabilities over amino acids at each column, which is roughly equivalent to the hidden Markov model.

Results for the case of $H = 2$ latent variables are shown in fig. 2. The

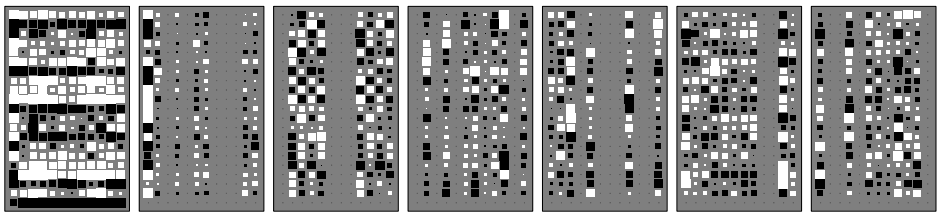

FIG. 3. Parameters $\mathbf{w}$ of an optimized density network modelling aligned antiparallel beta strands. From (MacKay 1996). In each grey rectangle the twelve columns represent the twelve residues surrounding a beta hydrogen bond, the first six on one strand and the second six on the other, with the hydrogen bond lying between residues 3, 4, 9 and 10. The twenty rows represent the twenty amino acids, in alphabetical order (A,C,D,...). Each rectangle shows the influences of one latent variable on the $12 \times 20$ probabilities. The top left rectangle shows the biases of all the output units. There is an additional 21st row in this rectangle for the biases of the output units corresponding to 'no amino acid'. The latent variables were defined to have no influence on these outputs to inhibit the wasting of latent variables on the modelling of dull correlations. The other six rectangles contain the influences of the 6 latent variables on the output units, of which the second and fifth are discussed in the text.

choice of two dimensions makes the globins easy to visualize. The posterior mean of the latent components, estimated by importance sampling, is displayed for each globin. The globin sub–families (GLB, HBA, HBB, etc.) are identified by the point style. The sub–families are seen to be cleanly separated in this representation.

### 4.5 Results on real data: beta sheets

Beta sheets are structures in which two parts of the protein engage in a particular hydrogen–bonding interaction. It would greatly help in the solution of the protein folding problem if we could distinguish correct from incorrect alignments of beta strands.

$N = 1000$ examples were taken from aligned antiparallel beta strand data provided by Tim Hubbard. Density networks with $H = 6$ latent inputs were used to model the joint distribution of the twelve residues surrounding a beta hydrogen bond. Our prior expectation is that if there is any correlation among these residues, it is likely to reflect the spatial arrangement of the residues, with nearby residues being correlated. But this prior expectation was not included in the model. The hope was that meaningful physical properties such as this would be learned from the data.

The parameters of a typical optimized density network are shown in fig. 3. The parameter vectors were compared, column by column, with a large number of published amino acid indices (Nakai et al. 1988) to see if they corresponded

to established physical properties of amino acids. Each index was normalized by subtracting the mean from each vector and scaling it to unit length. The similarity of a parameter vector to an index was then measured by the magnitude of their inner product.

Two distinctive patterns reliably emerged in most adapted models, both having a meaningful physical interpretation. First, an alternating pattern can be seen in the influences of the second latent variable (third rectangle from the left). The influences on columns 2, 4, 9 and 11 are similar to each other, and opposite in sign to the influences on columns 3, 5, 10 and 12. This dichotomy between the residues is physically meaningful: residues 2, 4, 9 and 11 are on the opposite side of the beta sheet plane from residues 3, 5, 10 and 12; when these influence vectors were compared with Nakai *et. al.*'s (1988) indices, they showed the greatest similarity to indices 57, 17, 7 and 42, which respectively describe the amino acids' polarity, the proportion of residues 100% buried, the transfer free energy to surface, and the consensus normalized hydrophobicity scale. This latent variable has clearly discovered the inside–outside characteristics of the beta sheet structure: either one face of sheet is exposed to the solvent (high polarity) or the other face, but not both.

Second, a different pattern is apparent in the second rectangle from the right. Here the influences on residues 4, 5, 6, 7, 8 are similar and opposite to the influences on 11, 12, 1, 2. For five of these residues the influence vector shows greatest similarity with index number 21, the normalized frequency of beta–turn. What this latent variable has discovered, therefore, is that a beta turn may happen at one end or the other of two anti–parallel beta strands, but not both.

Both of these patterns have the character of an 'exclusive–or' problem (Rumelhart and McClelland 1986). One might imagine that an alternative way to model aligned beta sheets would be to train a *discriminative* model such as a neural network binary classifier to distinguish 'aligned beta sheet' from 'not aligned'. However, such a model would have difficulty learning these exclusive–or patterns. Exclusive–or *can* be learnt by a neural network with one hidden layer and two layers of weights, but it is not a natural function readily produced by such a network. In contrast these patterns are easily captured by the density networks presented here, which have only one layer of weights.

It is interesting to note that the two effects discovered above involve competing correlations between large numbers of residues. The inside–outside latent variable produces a positive correlation between columns 4 and 11, for example, while the beta turn latent variable produces a negative correlation between those two columns. These results, although they do not constitute new discoveries, suggest that this technique shows considerable promise.

### 4.6 Future work

More complex models under development will include additional layers of processing between the latent variables and the observables. The present model has

no way of knowing that the 21 categories within a softmax group have the same meaning for all softmax groups. Imagine, for example, that two amino acids are functionally indistinguishable, and always occur with equal probability. With the present model, this relationship would have to be learned $S$ times over, once for each softmax group. But if some of the parameters of a second layer were communal to all columns of the protein, the model would be able to generalize amino acid equivalences from one column to another. This would reduce overfitting and improve predictive performance.

It is hoped that a density network adapted to beta sheet data will eventually be useful for discriminating correct from incorrect alignments of beta strands. The present work is not of sufficient numerical accuracy to achieve this.

## 5 Real Density Networks and Stochastic Dynamics

A major weakness of the importance sampling method described in the preceding sections is that it does not scale well with an increasing number of latent variables.

In this section we describe a more complex Monte Carlo implementation with favourable scaling properties. We sample from the joint posterior probability of the latent variables and the parameters,

$$P(\{\mathbf{x}^{(n)}\}, \mathbf{w}|D, \mathcal{H}) \propto \prod_n \left[P(\mathbf{t}^{(n)}|\mathbf{x}^{(n)}, \mathbf{w}, \mathcal{H})P(\mathbf{x}^{(n)}|\mathcal{H})\right] P(\mathbf{w}|\mathcal{H}), \qquad (5.1)$$

by a Gibbs/Metropolis method in which alternately the latent variables $\{\mathbf{x}^{(n)}\}$ are sampled using a Metropolis method that converges to $P(\{\mathbf{x}^{(n)}\}|\{\mathbf{t}^{(n)}\}, \mathbf{w})$; then the parameters are sampled from their distribution given the data and the latent variables, $P(\mathbf{w}|\{\mathbf{t}^{(n)}\}, \{\mathbf{x}^{(n)}\})$. The sampling is done using the hybrid Monte Carlo method (Neal 1993) which is reviewed below. The step in which the parameters are sampled is identical to ordinary Bayesian stochastic training of a neural network with known inputs.

One could also implement a simultaneous stochastic sampler which updated the parameters and the latent variables at the same time as each other. Such an approach, in which all $N$ latent variable vectors were simultaneously altered, would have the advantage of reducing random walk behaviour caused by correlations in the joint posterior distribution, but it would have the disadvantage that in order to maintain a reasonable acceptance probability for these more complex proposals, one would have to reduce the step size of the dynamical simulations.

### 5.1 A brief review of the hybrid Monte Carlo method

Radford Neal's 'Hybrid Monte Carlo' method for neural networks is a sophisticated Metropolis method, applicable to continuous state spaces, which makes use of gradient information to reduce random walk behaviour. It will be described here in the context of the general problem of sampling a continuous vector $\mathbf{x}$

from a probability $P(\mathbf{x})$ which can be written in the form

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} \tag{5.2}$$

where not only $E(\mathbf{x})$, but also its gradient with respect to $\mathbf{x}$ can be readily evaluated. It seems wasteful to use a simple random–walk Metropolis method when this gradient is available — the gradient indicates which direction one should go in to find states with higher probability!

In the hybrid Monte Carlo method, the state space $\mathbf{x}$ is augmented by *momentum variables* $\mathbf{p}$, and there is an alternation of two types of proposal. The first proposal density randomizes the momentum variable, leaving the state $\mathbf{x}$ unchanged. The second proposal density changes both $\mathbf{x}$ and $\mathbf{p}$ using reversible Hamiltonian dynamics as defined by the Hamiltonian

$$H(\mathbf{x}, \mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p}), \tag{5.3}$$

where $K(\mathbf{p})$ is a 'kinetic energy' such as $K(\mathbf{p}) = \mathbf{p}^{\mathsf{T}}\mathbf{p}/2$. Under these dynamics, the momentum variable determines where the state $\mathbf{x}$ goes, and the *gradient* of $E(\mathbf{x})$ determines how the momentum $\mathbf{p}$ changes. The net effect is that during each of the dynamical proposals, the state of the system moves a distance that goes *linearly* with the computer time, rather than as the square root.

If the simulation of the Hamiltonian dynamics is numerically perfect then the proposals are accepted every time; if the simulation is imperfect, because of finite step sizes for example, then some of the dynamical proposals will be rejected. The rejection rule makes use of the change in $H(\mathbf{x}, \mathbf{p})$, which is zero if the simulation is perfect.

Asymptotically, we obtain samples $(\mathbf{x}^{(t)}, \mathbf{p}^{(t)})$ from the joint density

$$P_H(\mathbf{x}, \mathbf{p}) = \frac{1}{Z_H} \exp[-H(\mathbf{x}, \mathbf{p})] = \frac{1}{Z_H} \exp[-E(\mathbf{x})] \exp[-K(\mathbf{p})]. \tag{5.4}$$

This density is separable, so it is clear that the marginal distribution of $\mathbf{x}$ is the desired distribution $\exp[-E(\mathbf{x})]/Z$. So, simply discarding the momentum variables, we obtain a sequence of samples $\{\mathbf{x}^{(t)}\}$ which asymptotically come from $P(\mathbf{x})$.

The source code in fig. 4 describes a hybrid Monte Carlo method which uses the 'leapfrog' algorithm to simulate the dynamics on the function `findE(x)`, whose gradient is found by the function `gradE(x)`.

### 5.2 Demonstration on a toy real–valued data set

We created a simple two–dimensional data set with a non–linear dependence on a single underlying variable, shown in fig. 5. The points are in fact noisy samples from a semicircle.

We modelled this data with a density network with two latent inputs. The parameters were put in just three weight classes (hidden unit biases, input weights

```
g = gradE ( x ) ;                    # set gradient using initial x
E = findE ( x ) ;                    # set objective function too

for l = 1:L                          # loop L times
   p = randn ( size(x) ) ;           # initial momentum is Normal(0,1)
   H = p' * p / 2 + E ;              # evaluate H(x,p)

   xnew = x
   gnew = g ;
   for tau = 1:Tau                   # make Tau 'leapfrog' steps

      p = p - epsilon * gnew / 2 ;   # make half-step in p
      xnew = xnew + epsilon * p ;    # make step in x

      gnew = gradE ( xnew ) ;        # find new gradient
      p = p - epsilon * gnew / 2 ;   # make half-step in p

   endfor

   Enew = findE ( xnew ) ;           # find new objective function
   Hnew = p' * p / 2 + Enew ;        # evaluate new value of H

   dH = Hnew - H ;                   # Decide whether to accept

   if ( dH < 0 )                accept = 1 ;
   elseif ( rand() < exp(-dH) ) accept = 1 ;
   else                         accept = 0 ;
   endif

   if ( accept )
      g = gnew ;   x = xnew ;    E = Enew ;
   endif
endfor
```

FIG. 4. `Octave` source code for the hybrid Monte Carlo method.

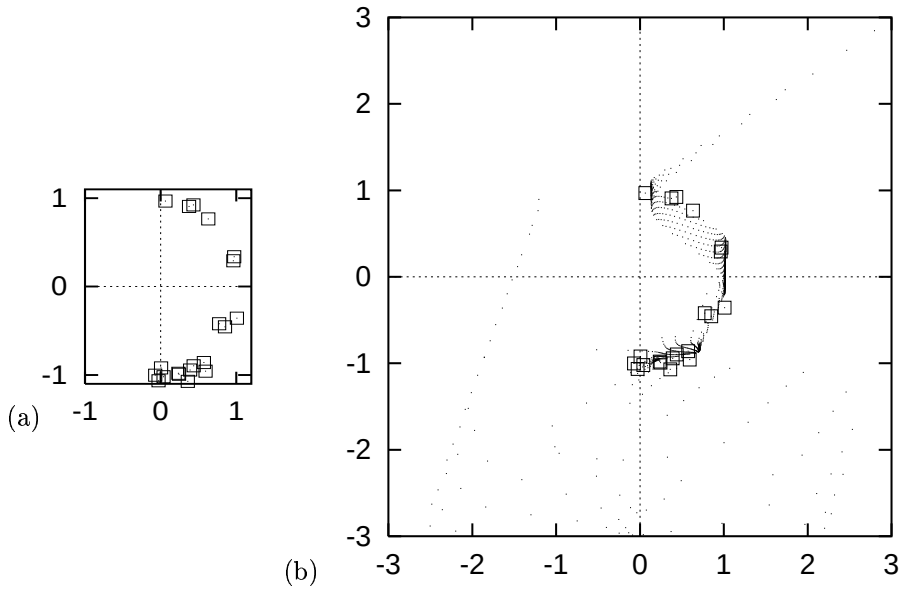


FIG. 5. Density network with real–valued outputs. (a) Data alone — the squares show the twenty data points. (b) State of the network after ten iterations. The network had two latent inputs, 16 hidden units and two outputs. The dots show the output of the network as its latent inputs vary in the range ± two standard deviations around their mean.

and output weights). The output noise level was fixed to $\sigma_x = 0.05$, and the latent variables, parameters and hyperparameters were updated iteratively as follows. First, the latent variables associated with each data point were updated using two hundred proposals generated by the hybrid Monte Carlo method. Each proposal used 30 leapfrog steps. Second, conditional on the latent variables, the weights were updated using two hundred proposals generated by the hybrid Monte Carlo method. Third, the hyperparameters (the mean and variance of the latent variables, and the variance of the three weight classes) were updated; this step might ideally have used Gibbs sampling, but here, the hyperparameters were reset to their maximum likelihood values conditional on the parameters. Figure 5 shows the situation after ten iterations. For typical values of the latent inputs, the output of the network is indeed in the semicircular region where the data are located.

## 6 Discussion

Density networks have shown some success as non–linear latent variable models. It is particularly encouraging that in applications to beta sheet protein sequences,

a natural, separable description of a complex distribution was found, with one latent variable capturing a hydrophobic/hydrophilic effect, and another discovering a beta–turn/no–beta–turn effect. This separation would not have been found if we had not used a hierarchical prior with multiple hyperparameters.

The biggest difficulty with density networks is that they require computer–intensive Monte Carlo methods. Importance sampling methods scale very badly with increasing dimensionality; stochastic dynamics methods scale better, but still worse than linearly with the number of variables.

A new latent variable model which can be implemented by local Monte Carlo methods with better scaling properties has recently been introduced by Hinton and Ghahramani (1997); this exciting model is able to discover both categorical and real latent variables. It has been applied to problems with high–dimensional real observables, but not yet to problems with categorical observables.

**Acknowledgements**

We thank Radford Neal, Geoff Hinton, Sean Eddy, Richard Durbin, Tim Hubbard and Graeme Mitchison for invaluable discussions. DJCM gratefully acknowledges the support of this work by the Royal Society Smithson Research Fellowship.

## Bibliography

A. Krogh, M. Brown, I. S. Mian, K. Sjolander and D. Haussler (1994). Hidden Markov models in computational biology: Applications to protein modeling, *Journal of Molecular Biology* **235**: 1501–1531.

B. S. Everitt (1984). *An Introduction to Latent Variable Models*, Chapman and Hall, London.

D. E. Rumelhart and J. E. McClelland (1986). *Parallel Distributed Processing*, MIT Press, Cambridge Mass.

D. J. C. MacKay (1992). A practical Bayesian framework for backpropagation networks, *Neural Computation* **4**(3): 448–472.

D. J. C. MacKay (1994). Bayesian non-linear modelling for the prediction competition, *ASHRAE Transactions, V.100, Pt.2*, ASHRAE, Atlanta Georgia, pp. 1053–1062.

D. J. C. MacKay (1995a). Bayesian neural networks and density networks, *Nuclear Instruments and Methods in Physics Research, Section A* **354**(1): 73–80.

D. J. C. MacKay (1995b). Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks, *Network: Computation in Neural Systems* **6**: 469–505.

D. J. C. MacKay (1996). Density networks and their application to protein modelling, *in* J. Skilling and S. Sibisi (eds), *Maximum Entropy and Bayesian Methods, Cambridge 1994*, Kluwer, Dordrecht, pp. 259–268.

E. Steeg (1997). *Automated Motif Discovery in Protein Structure Prediction*,

PhD thesis, Department of Computer Science, University of Toronto.

G. E. Hinton and R. S. Zemel (1994). Autoencoders, minimum description length and Helmholtz free energy, *in* J. D. Cowan, G. Tesauro and J. Alspector (eds), *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, San Mateo, California.

G. E. Hinton and Z. Ghahramani (1997). Generative models for discovering sparse distributed representations, submitted to Proc. Roy. Soc.

K. Nakai, A. Kidera and M. Kanehisa (1988). Cluster analysis of amino acid indices for prediction of protein structure and function, *Prot. Eng.* **2**: 93–100.

R. M. Neal (1993). Bayesian learning via stochastic dynamics, *in* C. L. Giles, S. J. Hanson and J. D. Cowan (eds), *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, California, pp. 475–482.

R. M. Neal (1996). *Bayesian Learning for Neural Networks*, number 118 in *Lecture Notes in Statistics*, Springer, New York.

S. P. Luttrell (1994). The partitioned mixture distribution: an adaptive Bayesian network for low-level image processing, *Proc. IEE Vision, Image and Signal Processing* **141**(4): 251–260.

S. R. Eddy and R. Durbin (1994). RNA sequence analysis using covariance models, *Nucleic Acids Research* **22**: 2079–2088.

V. Tresp, S. Ahmad and R. Neuneier (1994). Training neural networks with deficient data, *in* J. D. Cowan, G. Tesauro and J. Alspector (eds), *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, San Mateo, California.