Taking on the Curse of Dimensionality in Joint Distributions Using Neural Networks

Samy Bengio and Yoshua Bengio

Abstract— **The curse of dimensionality is severe when modeling high-dimensional discrete data: the number of possible combinations of the variables explodes exponentially. In this paper we propose a new architecture for modeling high-dimensional data that requires resources (parameters and computations) that grow at most as the square of the number of variables, using a multi-layer neural network to represent the joint distribution of the variables as the product of conditional distributions. The neural network can be interpreted as a graphical model without hidden random variables, but in which the conditional distributions are tied through the hidden units. The connectivity of the neural network can be pruned by using dependency tests between the variables (thus reducing significantly the number of parameters). Experiments on modeling the distribution of several discrete data sets show statistically significant improvements over other methods such as naive Bayes and comparable Bayesian networks, and show that significant improvements can be obtained by pruning the network.**

Keywords— **density estimation, high dimensionality, data mining, curse of dimensionality, probabilistic models, multi-layer neural networks, Bayesian networks, graphical models**

I. Introduction

The curse of dimensionality hits particularly hard on models of high-dimensional discrete data because there are many more possible combinations of the values of the variables than can possibly be observed in any data set, even the large data sets now common in data-mining applications. In this paper we are dealing in particular with multivariate discrete data, where one tries to build a model of the distribution of the data. What motivated this research in the context of data-mining was the need to model the distribution of high-dimensional data for two types of applications. First, in order to detect anomalous cases in customer databases (e.g. for fraud detection or to identify atypical customers) one needs a probabilistic model of high-dimensional data that does not assign zero probability - or a fixed small probability - to unseen cases, and yet that does take into account high-order dependencies between the variables. Second, in data-mining classification applications such as those requiring to identify "target customers" (e.g. those customers most likely to respond positively to a marketing campaign, or most likely to churn out and move to a competitor), the fraction of positive cases can be many orders of magnitude smaller than that of negative cases. Whereas ordinary neural networks are difficult to train as classifiers in this context, we have found it convenient to apply probabilistic models (one per class) to this problem, since the way in which each model is built does not depend on the relative frequency of classes.

When modeling the joint distribution of many discrete variables, a simple multinomial maximum likelihood model would give zero probability to all of the combinations that were not encountered in the training set, i.e., it would most likely give zero probability to most of the out-of-sample test cases. Smoothing the model by assigning the same non-zero probability for all the unobserved cases would not be satisfactory either because it would not provide much generalization from the training set. For example, such smoothing would be obtained with a multivariate multinomial model whose parameters θ are estimated by the maximum a-posteriori (MAP) principle, i.e., those that have the greatest probability, given the training data D, and using a diffuse prior $P(\theta)$ (e.g. Dirichlet) on the parameters.

The approach proposed here uses a neural network to represent the joint distribution of a set of random variables (taken in a given but possibly arbitrary order) as the product of conditional distributions (the probability of the i-th one given the previous ones). To simplify, one can see the network as a kind of auto-encoder in which the i-th variable is predicted based on the previous $i-1$ variables, but unlike in previously proposed neural network auto-encoders, the predictor for one variable does not use this variable in input. The architecture is such that hidden units are shared across the outputs while preserving the constraint that the conditional probability for the i-th variable only depends on the value of the previous variables. This structure allows the whole joint distribution to be automatically correctly normalized for any value of the parameters, as long as the probabilities associated to each conditional distribution are properly normalized (which is easily obtained with a sigmoid or a softmax). The model uses the approximating power of multi-layer neural networks to take into account dependencies of any order while requiring a number of parameters that grows like the square of the number of variables (unlike models which explicitly represent the joint or conditional probabilities and require an exponential number of parameters). The number of parameters can be further reduced using pruning, and this may help generalization, as shown in the experiments described in this paper. The experiments compare various alternative and more classical models such as naive Bayes and comparable Bayesian networks on four publically available data sets with many discrete variables, where the task is to model the distribution of the data as well as possible, using out-of-sample log-likelihood to compare performance.

A *graphical model* or Bayesian network [1], [2] represents

Samy Bengio was with CIRANO, 2020 University, Montréal, Canada. He is now with IDIAP, Martigny, Switzerland (bengio@idiap.ch).

Yoshua Bengio is with the department of computer science, Université de Montréal, Canada (bengioy@iro.umontreal.ca).

the joint distribution of random variables $Z_1, \ldots, Z_n$ with

$$P(Z_1, \ldots, Z_n) = \prod_{i=1}^{n} P(Z_i|\text{Parents}_i)$$

where Parents_i is the set of random variables which are called the *parents* of variable Z_i in the graphical model because they directly condition Z_i, and an arrow is drawn, in the graphical model, from each of its parents to Z_i. A fully connected "left-to-right" graphical model is illustrated in Figure 1, which corresponds to the model

$$P(Z_1, \ldots, Z_n) = \prod_{i=1}^{n} P(Z_i|Z_1, \ldots, Z_{i-1}). \qquad (1)$$

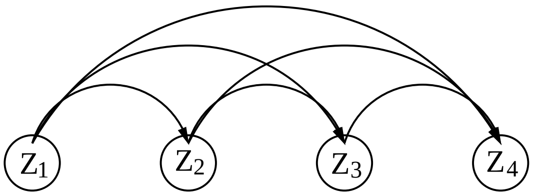

Fig. 1. *A fully connected "left-to-right" graphical model.*

Note that this representation depends on the ordering of the variables (in that all previous variables in this order are taken as parents). We call each combination of the values of the variables in Parents_i a *context* for Z_i. In the "exact" model (with the full table of all possible contexts) all the orders are equivalent, but if approximations are used, different predictions could be made by different models based on different orders. In this paper we do not study the effect of the choice of order and the experiments use the arbitrary order in which the variables are given in the data set. See the experiments of [3] in which the choice of order did not change the results significantly.

In graphical models, the *curse of dimensionality* shows up in the representation of conditional distributions $P(Z_i|\text{Parents}_i)$ where Z_i has many parents. If $Z_j \in \text{Parents}_i$ can take n_j values, there are $\prod_j n_j$ different contexts which can occur in which one would like to estimate the distribution of Z_i. This exponential number of contexts yields to an exponential number of parameters for the graphical model. This serious problem has been addressed in the past by two types of approaches, which are sometimes combined:

1. *Not modeling all the dependencies between all the variables*: this is the approach mainly taken with most graphical models or Bayes networks [1], [2]. The set of independencies can be assumed using a-priori or human expert knowledge or can be learned from data using search algorithms [4], [5], [6], [7]. See also [8] in which the set $Parents_i$ is restricted to at most one element, which is chosen to maximize the correlation with Z_i.
2. *Approximating the mathematical form of the joint distribution* with a form that takes only into account dependencies of lower order, or only takes into account some of the possible dependencies, e.g., with the Rademacher-Walsh expansion or multi-binomial [9], [10], which is a low-order polynomial approximation of a full joint binomial distribution (and is used in the experiments reported in this paper). The LARC model [3] described below is also of this type, but with first order dependencies only.

The approach we are putting forward in this paper is mostly of the second category, although we are using simple non-parametric statistics of the dependency between pairs of variables to further reduce the number of required parameters, in the spirit of the first of the above two categories.

In the multi-binomial model [10], the joint distribution of a set of binary variables is approximated by a polynomial. Whereas the "exact" representation of $P(Z_1 = z_1, \ldots, Z_n = z_n)$ (or its logarithm) as a function of $z_1, \ldots, z_n$ is a polynomial of order n, it can be approximated with a lower order polynomial, and this approximation can be easily computed using the Rademacher-Walsh expansion [9] (or other similar expansions, such as the Bahadur-Lazarsfeld expansion [9]). The Rademacher-Walsh expansion is defined as follows, letting $\mathbf{Z} = (Z_1, \ldots, Z_n)$ and $\mathbf{z} = (z_1, \ldots, z_n)$:

$$f(\mathbf{z}) = \sum_{i=0}^{2^n-1} a_i \phi_i(\mathbf{z}) \qquad (2)$$

with real-valued coefficients a_i, and where the ϕ_i are bases formed as follows, letting $\tilde{z}_i = 2z_i - 1$ (which take values -1 and 1),

- 0th order: $\phi_0(\mathbf{z}) = 1$,
- 1st order: for $i = 1$ to n, $\phi_i(\mathbf{z}) = \tilde{z}_i$,
- 2nd order: for $i = n+1$ to $n+1+n(n-1)/2$, all the products of pairs of the form $\tilde{z}_i\tilde{z}_j$ with $i \neq j$, i, j from 1 to n,
- k-th order: all the products of k-tuples, of the form $\tilde{z}_{i_1}\tilde{z}_{i_2}\ldots\tilde{z}_{i_k}$, with $i_p \neq i_q$ for $p \neq q$ and $i_p, i_q \in \{1 \ldots n\}$. These correspond to all the $\binom{n}{k}$ possible k-tuples.

These bases are orthogonal, and the corresponding coefficients a_i are moments of the corresponding order,

$$a_i = \frac{1}{2^n} E[\phi_i(\mathbf{Z})],$$

and they can be estimated by the corresponding sample averages. The expansion has the property that if it is truncated to k-th order, the above a_i's are still optimal in the mean-squared sense to approximate f. The function $f(\mathbf{z})$ in the expansion (2) could either be $P(\mathbf{Z} = \mathbf{z})$ or $log(P(\mathbf{Z} = \mathbf{z}))$. The first choice guarantees that the probabilities sum to 1 while the second that they are positive. We have preferred the second for the obvious reason that our measure of performance is the total log-probability of the test data. The k-th order approximation, instead of having 2^n parameters, requires only $O(n^k)$ parameters. Typically, order $k = 2$ is used (and was used in our experiments).

The model proposed in this paper also requires $O(n^2)$ parameters (at most, with no pruning), but it allows to model dependencies between arbitrary tuples of variables, with more than 2 variables at a time.

In previous related work [3], a fully-connected graphical model is used (see Figure 1) but each of the conditional distributions is represented by a logistic, taking into account only first-order dependencies between binary variables:

$$P(Z_i=1|Z_1=z_1,\ldots,Z_{i-1}=z_{i-1})=\frac{1}{1+\exp(-w_0-\sum_{j<i} w_j z_j)}.$$

Frey has named this model a *Logistic Auto-Regressive Classifier* or LARC (one such model is used for each class). He argues that the prior variances on the logistic weights (which correspond to inverse weight decays) should be chosen inversely proportional to the number of conditioning variables (i.e. the number of inputs to the particular output neuron). The model was tested on a task of learning to classify digits from 8x8 binary pixel images. Models with different orderings of the variables were compared and did not yield significant differences in performance. When averaging the predictive probabilities from 10 different models obtained by considering 10 different random orderings, Frey obtained small improvements in likelihood but not in classification. The model performed better or equivalently to other models tested: CART, naive Bayes, K-nearest neighbors, and various graphical models with hidden variables (Helmholtz machines). These results are impressive, taking into account the simplicity of the LARC model. In this paper, we basically extend Frey's idea to using a neural network with a hidden layer, with a particular architecture, allowing multinomial or continuous variables, and we propose to prune down the network weights.

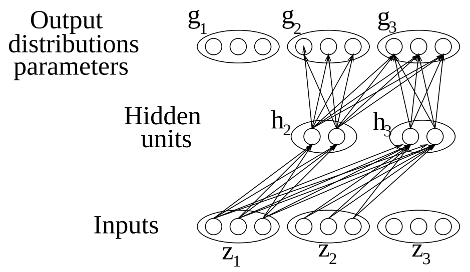

Fig. 2. *The architecture of a neural network that represents a fully connected "left-to-right" graphical model. For each variable Z_i, the observed value z_i is encoded in the corresponding input unit group. h_i is a group of hidden units. g_i is a group of output units (whose value depend only on $z_1, \ldots, z_{i-1}$). They represent the parameters of a distribution over Z_i, e.g. the probabilities associated with each possible value of Z_i. By reading the particular value that this distribution gives for $Z_i = z_i$ we obtain the conditional probabilities $P(Z_i = z_i|Z_1 = z_1, \ldots, Z_{i-1} = z_{i-1})$, and multiplying them we obtain the joint distribution.*

II. Proposed Architecture

The proposed architecture is a "neural network" implementation of a graphical model where all the variables are observed in the training set, with the hidden units playing a significant role to share parameters across different conditional distributions. Figure 2 illustrates the model in the simpler case of a fully connected (left-to-right) graphical model (Figure 1). The neural network represents the parameterized function

$$f_\theta(z_1,\ldots,z_n) = log(\hat{P}_\theta(Z_1 = z_1,\ldots,Z_n = z_n)) \quad (3)$$

approximating the joint distribution of the variables, with parameters θ being the weights of the neural network. The architecture has three layers, with each layer organized in *groups* associated with each of the variables. The above log-probability is computed as the sum of conditional log-probabilities

$$f_\theta(z_1,\ldots,z_n) = \sum_{i=1}^{n} log(P(Z_i = z_i|g_i(z_1,\ldots,z_{i-1})))$$

where $g_i(z_1,\ldots,z_{i-1})$ is the vector-valued output of the i-th group of output units, and it gives the value of the parameters of the distribution of Z_i, when $Z_1 = z_1, Z_2 = z_2, \ldots, Z_{i-1} = z_{i-1}$. In the case of binary variables, the output group can consist of a single unit that gives the conditional probability of the i-th variable Z_i being 1 given the values z_1 to z_{i-1} of the variables Z_1 to Z_{i-1}. In the multinomial case (output variable Z_i taking n_i possible discrete values), the i-th output group would have n_i outputs corresponding to the probabilities for the value $i' = 1$ to n_i for the i-th variable Z_i. By evaluating the probability of the value z_i under this distribution we obtain $P(Z_i = z_i|g_i(z_1,\ldots,z_{i-1}))$, and by multiplying all these numbers (for i from 1 to n) we obtain the value of the joint distribution at $(z_1,\ldots,z_n)$. In the discrete case, we have

$$P(Z_i = i'|g_i) = g_{i,i'}$$

where $g_{i,i'}$ is the i'-th output element of the vector g_i. In this example, a *softmax* output for the i-th group may be used to force these parameters to be positive and sum to 1, i.e.,

$$g_{i,i'} = \frac{e^{g'_{i,i'}}}{\sum_l e^{g'_{i,l}}}$$

where $g'_{i,i'}$ are linear combinations of the hidden units outputs, with i' ranging over the number of values that Z_i can take, i.e., n_i.

To guarantee that the functions $g_i(z_1,\ldots,z_{i-1})$ only depend on $z_1,\ldots,z_{i-1}$ and not on any of $z_i,\ldots,z_n$, the connectivity structure of the hidden units is constrained as follows:

$$g'_{i,i'} = b_{i,i'} + \sum_{j\le i}\sum_{j'=1}^{m_j} w_{i,i',j,j'} h_{j,j'}$$

where $h_{j,j'}$ is the output of the j'-th unit (out of m_j such units) in the j-th group of hidden layer nodes, the b's are corresponding biases and the $w_{i,i',j,j'}$'s are weights of the output layer, (connecting hidden unit (j,j') from hidden group j to the output unit (i,i') from output group i). Hidden units activations may be computed as follows (see Figure 2):

$$h_{j,j'} = \tanh(c_{j,j'} + \sum_{k<j}\sum_{k'=1}^{n_k} v_{j,j',k,k'} z'_{k,k'})$$

where the c's are biases and the $v_{j,j',k,k'}$'s are the weights of the hidden layer (from input unit (k,k') to hidden unit (j,j')), and $z'_{k,k'}$ is the k'-th element of the vectorial input representation z'_k for the value $Z_k = z_k$. We used a hyperbolic tangent non-linearity, but other non-linearities could have been used as long as the universal approximator properties of the network are preserved. For example, in the binary case, we have used only one input node per input group, i.e., we have

$$Z_i \in \{0,1\} \ \rightarrow z'_{i,0} = z_i$$

and in the general discrete case we use the "one-hot" encoding:

$$Z_i \in \{0,1,\ldots,n_i-1\} \ \rightarrow z'_{i,i'} = \delta_{z_i,i'}$$

where $\delta_{i,i'} = 1$ if $i = i'$ and 0 otherwise. The input layer has $n-1$ groups because the value $Z_n = z_n$ is not used as an input. The hidden layer also has $n-1$ groups corresponding to the variables $j = 2$ to n (because $P(Z_1)$ is represented unconditionally in the first output group, which does not need any associated hidden units or inputs, but just has biases).

A. Discussion

The number of free parameters of the model is $O(n^2H)$ where $H = \max_j m_j$ is the maximum number of hidden units per hidden group (i.e., associated with one of the variables). This is basically quadratic in the number of variables, like the multi-binomial approximation, that uses a polynomial expansion of the joint distribution. As the order k of the multi-binomial approximation increases, it can better approximate the true joint distribution (if one was able to estimate the parameters). Similarly, as H is increased, representation theorems for neural networks suggest that we should be able to approximate with arbitrary precision the true joint distribution. Of course in both cases the true limiting factor is the amount of data, and H should be tuned according to the amount of data. In our experiments we have used cross-validation to choose H (with the same value of $m_j = H$ for all the hidden groups). In this sense, this neural network representation of $P(Z_1,\ldots,Z_n)$ is to the polynomial expansions (such as the multi-binomial) what ordinary multi-layer neural networks for function approximation are to polynomial function approximators. The neural network can capture high-order dependencies, but not all of them (with a reasonable capacity). On the other hand, the polynomial approximation captures only low-order dependencies (e.g., second order), but it captures them all. For the neural network, it is the number of hidden units per hidden group, H, that controls "how many" such dependencies will be captured, and it is the data that "chooses" which of the actual dependencies are most useful in optimizing the training criterion.

There are really two kinds of high dimensional discrete data: (1) a large number of variables each taking few values (e.g. as in many data-mining applications), and (2) a few variables each taking a huge number of values (e.g. in textual data and natural language models). Note that the approach presented in this paper is well suited to modeling data of type (1). One interesting idea would be to attempt to convert data of type (2) into data of type (1) and apply the methodology of this paper. This could be achieved by representing for example each word in a text by a possibly large set of low-dimensional features.

B. MAP Criterion and Weight Decay

Unlike Bayesian networks with hidden random variables, learning with the proposed architecture is very simple, even when there are no conditional independencies. To optimize the parameters we have simply used gradient-based optimization methods, using conjugate or stochastic (on-line) gradient, to maximize a MAP (maximum a posteriori) criterion, that is the sum of a log-prior and the total log-likelihood (which is the sum of values of f (eq. 3) for the training examples). In our experiments we have used a "weight decay" log-prior, which gives a quadratic penalty to the parameters θ:

$$logP(\theta) = constant + \sum_i \gamma_i \theta_i^2.$$

Inspired by the analysis of [3], the inverse variances γ_i are chosen proportional to the number of weights incoming into a neuron

$$\gamma_i = \gamma_0 F_i$$

where F_i is the "fan-in" or number of inputs into the neuron to which the i-th weight contributes. The shared hyper-parameter γ_0 is chosen by cross-validation.

C. Marginalization and Missing Values

An important question one might ask about this model is how it could be marginalized. In general this is going to be expensive, requiring to sum over possibly many combinations of the values of variables not in the desired marginal, using

$$P(Z_1,\ldots,Z_{i-1},Z_{i+1},\ldots,Z_n) = \sum_j P(Z_1,\ldots,Z_{i-1},Z_i=j,Z_{i+1},\ldots,Z_n).$$

Another related question is whether one could deal with missing values: if the total number of values that the missing variables can take is reasonably small, then one can sum over these values in order to obtain a marginal probability. If the example with missing values is in the training set, this marginal probability can then be maximized within the MAP criterion. If it is a test example, this marginal probability can be used in taking a decision. If some variables have more systematically missing values, they can be put at the end of the variable ordering, and in this case the marginal distribution can be very efficiently computed (by taking only the product of the output probabilities up to the missing variables). Similarly, one can easily compute the predictive distribution of the last variable given the first $n-1$ variables. $P(Z_n|Z_1 = z_1,\ldots,Z_{n-1} = z_{n-1})$ can be simply read off the last output group when feeding $z_1,\ldots,z_{n-1}$ to the inputs of the neural network.

D. Extension to Continuous Variables and Conditional Distributions

The framework can be easily extended to hybrid models involving both continuous and discrete variables. In the case of continuous variables, one has to choose a parametric form for the distribution of the continuous variable when all its parents (i.e., the conditioning context) are fixed. Note that if there are significant dependencies between the variables, the conditional distribution of Z_i given $Z_1, \ldots, Z_{i-1}$ will have much less entropy (will be much more peaked), and it can probably be well modeled with simpler distribution classes than the unconditional distribution of Z_i. For example one could use a normal, log-normal, or mixture of normals. Instead of having softmax outputs, the i-th output group would compute the parameters of this continuous distribution. For example, let us consider the simple case of a univariate Normal variable:

$$Z_i \sim N(\mu, \sigma^2).$$

The corresponding output group could compute the 2-element vector $(\mu, \log \sigma)$, which is unconstrained. In the d-dimensional multivariate normal case, with

$$Z_i \sim N(\mu, \Sigma),$$

the covariance matrix can be represented by the $d(d+1)/2$ lower-diagonal and diagonal elements of L in the following Cholesky decomposition of Σ:

$$\Sigma = LL'$$

with $L_{i,j} = 0$ for $j > i$. The corresponding output group could compute a vector of $d + d(d+1)/2$ unconstrained elements, representing μ and the non-zero elements of L. Similarly, the model could be extended to other continuous distributions by appropriately transforming the unconstrained outputs of the neural network (i.e., weighted sums) into the parameters of the desired distribution.

Another type of extension allows to build a conditional distribution, e.g., to model $P(Z_1, \ldots, Z_n | X_1, \ldots, X_m)$, in a way that is similar to the approach already proposed by Bishop [11]. With the model proposed here, one just adds extra input units to represent the values of the conditioning variables $X_1, \ldots, X_m$. Finally, an architectural extension that we have implemented is to allow direct input-to-output connections (still following the rules of connections ordering which allow g_i to depend only on $z_1, \ldots, z_{i-1}$). Therefore in the case where the number of hidden units is 0 ($H = 0$), we obtain the LARC model studied by Frey [3].

E. Choice of topology

Another type of extension of this model which we have found very useful in our experiments is to allow the user to choose a topology that is not fully connected (left-to-right). In our experiments we have used non-parametric tests to heuristically eliminate some of the connections in the network, but one could also use expert or prior knowledge, just as with regular graphical models, in order to cut down on the number of free parameters.

In our experiments we have used for a pairwise test of statistical dependency the Kolmogorov-Smirnov statistic. The statistic for variables X and Y is

$$s = \sqrt{l} \sup_i |\hat{P}(X \leq x_i, Y \leq y_i) - \hat{P}(X \leq x_i)\hat{P}(Y \leq y_i)|$$

where l is the number of examples and $\hat{P}$ is the empirical distribution (obtained by counting over the training data). Under the null hypothesis of no statistical dependency for the 1-dimensional case (but note that here we are in the two-dimensional case), the asymptotic distribution of s [12] is

$$\lim_{l \to \infty} P(s < \epsilon) = 1 - 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2\epsilon^2 k^2}.$$

The sum can be closely approximated with its first few terms, because of its exponential convergence. As a heuristic, we have ranked the pairs according to their value of the statistic s, and we have chosen those pairs for which the value of statistic is above a threshold value s^*. This threshold value was chosen by cross-validation. When the pairs $\{(Z_i, Z_j)\}$ are chosen to be part of the model, and assuming without loss of generality that $i < j$ for those pairs, then the only connections that are kept in the network (in addition to those from the k-th hidden group to the k-th output group) are those from hidden group i to output group j, and from input group i to hidden group j, for every such (Z_i, Z_j) pair.

III. Experiments

In the experiments we have compared the neural network model (with and without pruning) to other models on four data sets obtained on the web from the UCI Machine Learning and STATLOG databases. Most of the data sets are meant to be for classification tasks but we have instead ignored the classification and used the data to learn a probabilistic model of all the input features. In all cases except Audiology, the train/test split used to measure performance is the one described in the documentation of the data set.

- DNA (from STATLOG): there are 180 binary features. 2000 cases are used for training and cross-validation, and 1186 for testing.
- Mushroom (from UCI): there are 22 discrete features (taking each between 2 and 12 values). 4062 cases are used for training and cross-validation, and 4062 for testing.
- Audiology (from UCI): there are 69 discrete features (taking each between 2 and 7 values). In our experiments, the first 113 cases are used for training and the remaining 113 for testing (the original train-test partition was 200 + 26 but we concatenated and re-split the data to obtain more significant test figures).
- Soybean (from UCI): there are 35 discrete features (taking each between 2 and 8 values). 307 cases are used for training and 376 for testing.

The compared models are the followings:

- *Naive Bayes*: the likelihood is obtained as a product of multinomials (one per variable). Each multinomial is smoothed with a Dirichlet prior.

• *Multi-Binomial*: the Rademacher-Walsh expansion of order 2 described in the introduction [10]. Since this only handles the case of binary data, it was only applied to the DNA data set.
• *Ordinary BN (Bayes Net, or graphical model)* with the same pairs of variables and variable ordering as selected for the pruned neural network, but in which each of the conditional distributions is modeled by a separate multinomial for each of the conditioning contexts. Because of the exponential explosion in the number of parameters, this works only if the number of conditioning variables is small. So in the Mushroom, Audiology, and Soybean experiments we had to reduce the number of conditioning variables (following the order given by the Kolmogorov-Smirnov tests). The multinomials are also smoothed with a Dirichlet prior.
• *Fully connected NN (neural network)*: there are no direct input to output connections but there are H hidden units per hidden unit group, as described in the previous section. The order of the variables is taken as is from the data set.
• *Pruned NN (neural network)*: the Kolmogorov-Smirnov statistic with threshold p-value chosen by 5-fold cross-validation was used to prune the connectivity. Five different pruning levels were tried, including no pruning. The order of the variables is taken as is from the data set.
• *LARC*: this corresponds to a fully connected (left-to-right) neural network with no hidden units (direct input to output connections). The LARC model from [3] was in fact extended to deal with the case of multinomials just like the neural network.
• *Pruned LARC*: using the same methodology as for the neural networks, the connections are pruned, using a threshold chosen by 5-fold cross-validation.

K-fold cross-validation with $k = 5$ was used to select the number of hidden units per hidden group (values from 2 to 20) and the weight decay for the neural network and LARC (values from 0.0001 to 1). The same technique was also used to choose the amount of smoothing in the Dirichlet priors (various values from 0.0001 to 10000) for the multinomials of the naive Bayes model and the ordinary graphical model. The same technique was used to choose the pruning threshold (p-values of 20%, 15%, 10% or 5%, or no pruning). To maximize the MAP criterion, conjugate gradients optimization was used for the smaller data sets (Soybean and Audiology) while stochastic gradient descent was used for the larger ones (Mushroom and DNA).

A. Results

Table I summarizes the experimental results, using out-of-sample average log-likelihood as a yardstick. Tests of significance allow a comparison of the average performance of the pruned neural network with each of the other models, using the average of the difference in log-likelihood for each of the test cases. Using these differences is important because there are strong correlations between the errors made by two models on the same test pattern, so the variance of the difference is usually much less than the sum of the variances. Note that looking only at the standard deviations (in parentheses in the table) would suggest that

	DNA		Mushroom	
	μ (σ)	p-value	μ (σ)	p-value
naive Bayes	100.4 (.18)	<1e-9	47.00 (.29)	<1e-9
multi-Binomial	117.8 (.01)	<1e-9		
ordinary BN	108.1 (.06)	<1e-9	44.68 (.26)	<1e-9
LARC	83.2 (.24)	7e-5	42.51 (.16)	<1e-9
pruned LARC	91.1 (.15)	<1e-9	43.45 (.14)	<1e-9
full-conn. NN	120.0 (.02)	<1e-9	33.58 (.01)	<1e-9
pruned NN	82.9 (.21)		31.25 (.04)	
	Audiology		**Soybean**	
	μ (σ)	p-value	μ (σ)	p-value
naive Bayes	36.40 (2.9)	<1e-9	34.74 (1.0)	<1e-9
multi-Binomial				
ordinary BN	16.56 (.48)	6.8e-4	43.65 (.07)	<1e-9
LARC	17.69 (.65)	<1e-9	16.95 (.35)	5.5e-4
pruned LARC	16.66 (.41)	0.20	19.07 (.44)	<1e-9
full-conn. NN	17.39 (.58)	<1e-9	21.65 (.43)	<1e-9
pruned NN	16.37 (.45)		16.55 (.27)	

TABLE I

Average out-of-sample negative log-likelihood (μ) obtained with the various models on four data sets (standard deviations (σ) of the average in parenthesis and p-value to test the null hypotheses that a model has same true generalization error as the pruned neural network). The pruned neural network was better than all the other models in all cases, and the pair-wise difference is always statistically significant, except with respect to the pruned LARC on Audiology.

many of these differences are not significant, whereas the more complete analysis that takes the covariances into account by first computing the differences in individual errors yields very significant differences: the pruned neural network was superior to all the other models in all 4 cases, and the pairwise differences with the other models are statistically significant in all 4 cases ,except Audiology, where the difference with the network without hidden units, LARC, is not significant. In some cases the difference with LARC is quite large while in others it is not significant, in terms of out-of-sample log-likelihood. This is not very surprising as the need for hidden units may be very task-dependent. Pruning seems to help the network with hidden units in all cases but helped LARC only in 1 out of 4 cases (maybe because the capacity of LARC was already too small). It should also be noted that log-likelihood reflects how well the model does at assigning high probability to events actually occurring out-of-sample, but it does not necessarily indicate how useful the model would be for doing tasks such as classification or selecting outliers. Although in principle having the "right" or true model (the one with the largest expected out-of-sample expected log-likelihood over all possible models) would allow us to perform the best decision according to a given utility function using Bayesian decision theory, all the above models are approximations.

Since the number of test cases is always more than a hundred and these log-likelihoods are independent (given the learned models), we assume that the **average** log-likelihood difference is Normally distributed, with a variance that is estimated unbiasedly by the sample variance of the difference in log-likelihoods divided by the number of test cases. We perform one-sided Z-tests to compute p-values as $P(\text{true difference} > \text{observed difference}|H_0)$, under the null hypothesis H_0 that the expectation of the difference is 0 (i.e., that the two algorithms are really of the same level of performance and the observed difference is due to sampling noise). Each model is compared to the pruned neural network (which had the best average performance on all four tasks). Note however that these tests do not take into account the variability due to the choice of training set (see [13] and [14] for a discussion).

IV. Conclusion

In this paper we have proposed a new application of multi-layer neural networks to the modeling of high-dimensional distributions, in particular for discrete data (but the approach could also be applied to continuous or mixed discrete / continuous data). Such a model could be used in data-mining applications where the number of variables is large. Like the polynomial expansions [10] that have been previously proposed for handling such high-dimensional distributions, the model approximates the joint distribution with a reasonable (at most $O(n^2)$) number of free parameters but unlike these it allows to capture high-order dependencies even when the number of parameters is small. The model can also be seen as an extension of the previously proposed auto-regressive logistic classifier [3], using hidden units to capture some high-order dependencies.

Experimental results on four data sets with many discrete variables are very encouraging. The comparisons were made with a naive Bayes model, with a multi-binomial expansion, with the LARC model and with an ordinary graphical model, showing that the pruned neural network did significantly better in terms of out-of-sample log-likelihood in almost all cases.

Pruning appears to be very helpful in getting such good results, at least in the case of the neural network with hidden units. The approach to pruning the neural network used in the experiments, based on pairwise statistical dependency tests, is highly heuristic and better results might be obtained using approaches that take into account the higher order dependencies when selecting the conditioning variables. Methods based on pruning the fully connected network (e.g., with a "weight elimination" penalty) should also be tried. A probabilistic model can also be used to perform the pruning, e.g. using an entropic prior [15] on the weights that encourages zero values. Also, we have not tried to optimize the order of the variables, or combine different networks obtained with different orders, like [3].

One of the particularities of the model proposed here is that the hidden units may compute functions that are shared and used for several of the conditional probabilities estimated by the model. An alternative would simply have been to have a different neural network for each conditional distribution. In general, which works better will depend on the underlying distribution of the data, but the concept of sharing functionalities with hidden units has been much used in the past (with success) with multi-layer neural networks having several outputs. It would be interesting to test these differences experimentally. However, the most important next step to follow the work presented in this paper is to test how the proposed model, which works well in terms of out-of-sample likelihood, will perform in terms of classification error (on classification tasks), when the model is used to learn one conditional density per class.

To summarize, the contributions of this paper are the following: first, showing how neural networks can be used as approximators for the joint distribution of data sets with many variables (as found in many data-mining applications) and how this helps to deal with the curse of dimensionality; second, this is achieved using an extension of Frey's LARC graphical model, by including hidden units to represent high-order dependencies and share functionality across the different conditional distributions; third, we have found pruning of the neural network weights using a non-parametric test of dependency and cross-validation to significantly help performance; fourth, comparative experiments on four data sets with discrete variables show this approach to work better than five alternatives.

References

[1] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[2] Steffen L. Lauritzen, "The EM algorithm for graphical association models with missing data," *Computational Statistics and Data Analysis*, vol. 19, pp. 191–201, 1995.

[3] B. Frey, *Graphical models for machine learning and digital communication*, MIT Press, 1998.

[4] P. Spirtes and C. Glymour, "An algorithm for fast recovery of sparse causal graphs," *Social Science Computing Reviews*, vol. 9, no. 1, pp. 62–72, 1991.

[5] J. Pearl and T.S. Verma, "A theory of inferred causation," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, J.A. Allen, R. Fikes, , and E. Sandewall, Eds. 1991, pp. 441–452, Morgan Kaufmann, San Mateo, CA.

[6] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, Springer-Verlag, New York, 1993.

[7] R. Scheines, "Inferring causal structure among unmeasured variables," in *Selecting Models from Data: Artificial Intelligence and Statistics IV*, P. Cheeseman and R.W. Oldford, Eds., pp. 197–204. Springer-Verlag, 1994.

[8] C.K. Chow, "A recognition method using neighbor dependence," *IRE Trans. Elec. Comp.*, vol. EC-11, pp. 683–690, October 1962.

[9] R.R. Bahadur, "A representation of the joint distribution of responses to n dichotomous items," in *Studies in Item Analysis and Predictdion*, ed. H. Solomon, Ed., pp. 158–168. Stanford University Press, California, 1961.

[10] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[11] Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, London, UK, 1995.

[12] A.N. Kolmogorov, "Sulla determinazione empirica di una leggi di distribuzione," *G. Inst. Ital. Attuari*, vol. 4, 1933, translated in English in *Breakthroughs in Statistics*, by Kotz and Johnson (editors), Springer-Verlag, 1992.

[13] T.G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10 (7), pp. 1895–1924, 1998.

[14] Claude Nadeau and Yoshua Bengio, "Inference for the generalization error," in *Advances in Neural Information Processing Systems 12*. 2000, p. to appear, MIT Press.

[15] Matthew Brand, "Structure learning in conditional probability models via an entropic prior and parameter extinction," *Neural Computation*, vol. 11, no. 5, pp. 1155–1182, 1999.

Samy Bengio is a research director and the machine learning group leader at the *Dalle Molle Institute for Perceptual Artificial Intelligence* (IDIAP) since 1999. He has obtained his PhD in computer science from *Université de Montréal* (1993), and spent three post-doctoral years at CNET, the research center of France Telecom, and INRS-Télécommunications (Montréal). He then worked as a researcher for CIRANO, an economic and financial academic research center, applying learning algorithms to finance. Before joining IDIAP, he was also research director at Microcell Labs, a private research center in mobile telecommunications. His current interests include learning algorithms applied to time series and data mining problems.

Yoshua Bengio is associate professor in the department of computer science and operations research at the *Université de Montréal* since 1993, and he is deputy-director of the *Centre de Recherches Mathématiques* in Montreal. He has obtained his PhD in computer science from McGill University (1991), and spent two post-doctoral years at M.I.T. and AT&T Bell Labs. Author of numerous articles on artificial neural networks and other learning algorithms and their applications, he is associate editor of the Neural Computing Reviews as well as of the IEEE transactions on neural networks. His current interests include generic methods to improve generalization, how to deal with non-stationarity and other sequential structure, and applications in data-mining, financial time-series, language and vision.