Programming the Harmonium[*]

Petri Myllymäki Pekka Orponen
Department of Computer Science, University of Helsinki
SF–00510 Helsinki, Finland

Abstract

We show how to synthesize, given a Bayesian network description of a probability distribution, an instance of Smolensky's harmony network that computes maximum-likelihood completions to partial value assignments, according to the given distribution. As an application, we present a scheme for translating a high-level description of a conceptual hierarchy, with default values and exceptions, to a harmony network representation, which is then inherently capable of quite complicated query processing. Finally, we discuss our experience from implementing such a translation scheme.

1 Introduction

The work presented in this paper has been motivated by the goal of developing *hybrid neural–symbolic* programming systems. Such systems aim at combining the benefits of two complementary approaches to knowledge representation and reasoning: a neural knowledge representation is used to provide a basic robustness to the system, whereas standard symbolic techniques are used for complicated high-level inferential tasks, at which current neural network techniques perform poorly. Some of the related recent work has been reported in the collections [2, 4].

A pivotal issue in building a hybrid neural–symbolic system is designing the interface between the two components. In this, we have found the conceptual framework of *Bayesian networks*, as developed in particular by Pearl in his book[10], most useful. In an earlier paper [9], we presented a neural implementation of Pearl's *stochastic simulation* method for estimating posterior probabilities on the basis of a given Bayesian network. In this paper, we show how to explicitly set up a stochastic network conforming to Smolensky's [12] *harmony theory*, or *"harmonium"* model, corresponding to a given Bayesian network. Such a network can then be used to answer queries on the basis of the given probabilistic information, in the sense of computing maximum-likelihood completions to partial value assignments to the variables under consideration.

As an application, we show how conceptual hierarchies, described in a high-level language, can be translated into a harmony network representation, amenable to processing queries of a quite general form. In spirit this work is similar to, e.g., Shastri's [11], but whereas Shastri's networks are deterministic, and consequently require fairly complicated computing elements and control regimes, the stochastic harmony networks are structurally very simple and uniform.

2 Bayesian Networks, Harmony Networks, and Markov Random Fields

We approach the problem of realizing Bayesian networks on the harmonium somewhat indirectly, by first discussing the relationship of *Markov random fields* [3, 6] to harmony networks.

Let $\mathcal{F} = \{X_1, \ldots, X_n\}$ be a set of binary (± 1) valued random variables that form a Markov random field (MRF) with potential function U. In other words, assume that the joint distribution of the variables X_i is

[*]This work has been partially supported by the Academy of Finland, and by the Finnish Technology Development Center (TEKES) under its FINSOFT programme.

Figure 1: A harmony network.

the Gibbs distribution determined by U:

$$P(\boldsymbol{x}) = Z^{-1}e^{U(\boldsymbol{x})},[1]$$

where Z is a normalization constant; and that there is defined on $\mathcal{F}$ a system of *cliques* $\mathcal{C} \subseteq \mathcal{P}(\mathcal{F})$ such that U can be decomposed:

$$U(\boldsymbol{x}) = \sum_{C \in \mathcal{C}} V_C(\boldsymbol{x}),$$

where each *clique potential* V_C depends only on the variables in the clique C. For each clique C, the potential V_C may be decomposed further by considering all the different value assignments α to variables in the clique, $\alpha : C \rightarrow \{+1, -1\}$, and writing

$$V_C(\boldsymbol{x}) = \sum_{\alpha} \lambda_\alpha \chi_\alpha(\boldsymbol{x}),$$

where the constant λ_α is the value of the clique potential for the assignment α, and the function $\chi_\alpha(\boldsymbol{x})$ has value 1 or 0 according to whether the value assignment α "matches" state vector $\boldsymbol{x}$ or not. A value assignment α can also be thought of as a vector $\boldsymbol{\alpha} \in \{-1, 0, +1\}^n$, whose 1-norm $|\boldsymbol{\alpha}|$ is equal to the size of the underlying clique C. In this formulation, the definition of the function χ_α can be written as

$$\chi_\alpha(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{\alpha} \cdot \boldsymbol{x}/|\boldsymbol{\alpha}| = 1, \\ 0, & \text{otherwise.} \end{cases}$$

We shall henceforth mostly forget about the functions V_C, and represent the potential U directly in terms of the elementary potentials associated to the value assignments:

$$U(\boldsymbol{x}) = \sum_{\alpha} \lambda_\alpha \chi_\alpha(\boldsymbol{x}).$$

A Markov random field determined by a clique structure and a potential function U can be given an approximate realization as a harmony network as follows. Consider a network consisting of two layers of stochastic binary-valued units, a set of *feature* nodes corresponding to the variables $X_1, \ldots, X_n$, and a set of *pattern* nodes (Smolensky's "knowledge" nodes) corresponding to the value assignments α (Figure 1). The possible activity values for the feature nodes are $\{-1, +1\}$, and the values for the pattern nodes are $\{0, 1\}$. A pattern node corresponding to a value assignment α for a clique C is connected to each of the feature nodes representing variables in C. The interconnections are undirected, and the weight of the connection between pattern node α and feature node i is given by

$$w_{\alpha i} = \alpha_i \sigma_\alpha / |\boldsymbol{\alpha}|,$$

where α_i, the *sign* of the connection is $+1$ or -1, according to which value the assignment α assigns to variable X_i; $|\boldsymbol{\alpha}|$ is the number of connections at node α; and σ_α is the *strength* of the node α, defined as

$$\sigma_\alpha = \lambda_\alpha / (1 - \kappa_\alpha),$$

[1]Note that we are deviating from standard practice and following Smolensky [12] in the choice of the sign of the function U.

where κ_α is a parameter that can be given any value satisfying $1 - 2/|\boldsymbol{\alpha}| < \kappa_\alpha < 1$.

The nodes in the network are updated asynchronously in a random order[2], according to the following rules: if the node selected for updating is a pattern node α, and the current states of the feature nodes are given by x_i, then a net input value of

$$I_\alpha = \sum_i w_{\alpha i} x_i - \sigma_\alpha \kappa_\alpha$$

is computed, and the node obtains value 1 with probability $P(a_\alpha = 1) = (1 + e^{-I_\alpha/T})^{-1}$, where T is the current *computational temperature*. For a feature node i the net input is computed as

$$I_i = \sum_\alpha w_{\alpha i} a_\alpha,$$

where the a_α are the current states of the pattern nodes; the node then obtains value 1 with probability $P(x_i = 1) = (1 + e^{-2I_i/T})^{-1}$.

Smolensky [12] proves the following fundamental result about the computational behavior of harmony networks. Consider a partial assignment $\boldsymbol{y}$ to some of the variables, say $X_{i_1}, \ldots, X_{i_k}$. Then a state vector $\boldsymbol{x}$ is a *maximum-likelihood completion* of $\boldsymbol{y}$ if it extends $\boldsymbol{y}$, i.e., if it agrees with $\boldsymbol{y}$ on the variables $X_{i_1}, \ldots, X_{i_k}$, and if for any state vector $\boldsymbol{x}'$ extending $\boldsymbol{y}$ it is the case that $P(\boldsymbol{x}) \geq P(\boldsymbol{x}')$.

Let H be a harmony network constructed as above from a representation of the distribution

$$P(\boldsymbol{x}) \propto \exp(\sum_\alpha \lambda_\alpha \chi_\alpha(\boldsymbol{x})),$$

where all the parameters λ_α are *positive.*[3]. Smolensky proves that such a network H computes maximum-likelihood completions according to the distribution P, in the following sense. Let a partial assignment $\boldsymbol{y}$ to some variables $X_{i_1}, \ldots, X_{i_k}$ be given. Initialize H so that the feature nodes $i_1, \ldots, i_k$, corresponding to variables $X_{i_1}, \ldots, X_{i_k}$, are assigned the values determined by $\boldsymbol{y}$, and the other nodes are initialized randomly. During the subsequent computation by H, the values of nodes $i_1, \ldots, i_k$ are kept fixed, while the other nodes update their values stochastically, according to the rules given above. If the temperature parameter T is lowered to zero sufficiently slowly during the computation [1, 3], the network will with high probability converge to a state that, projected to the feature nodes, represents a maximum-likelihood completion of the initial assignment $\boldsymbol{y}$.

Let us then consider Bayesian networks. We present here a brief summary of the basic definitions; for details see [10]. Let P be a probability distribution over the set of variables $\mathcal{F} = \{X_1, \ldots, X_n\}$, and let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subseteq \mathcal{F}$ be sets of variables. Then $\boldsymbol{X}$ is *conditionally independent of* $\boldsymbol{Y}$, *given* $\boldsymbol{Z}$, if the condition

$$P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z}) = P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Z} = \boldsymbol{z})$$

holds for all vectors of values $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ such that $P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{Z} = \boldsymbol{z}) > 0$. (Intuitively, the variables in $\boldsymbol{Z}$ intercept any causal connections between the variables in $\boldsymbol{X}$ and the variables in $\boldsymbol{Y}$: knowing the values of $\boldsymbol{Z}$ renders information about the values of $\boldsymbol{Y}$ irrelevant to determining the distribution of $\boldsymbol{X}$.)

A *Bayesian network* for a probability distribution P is a directed acyclic graph D whose nodes correspond to the variables $X_1, \ldots, X_n$, and whose topology satisfies the following: each variable X is conditionally independent of all its non-descendants in D, given its set of parents $\boldsymbol{F}_X$, and no proper subset of $\boldsymbol{F}_X$ satisfies this condition. (Here the parent variables $\boldsymbol{F}_X$ may intuitively be thought of as the immediate causes of variable X.) In a Bayesian network representation, we are typically given for each variable X the matrix of conditional probabilities $P(X = x | \boldsymbol{F}_X = \boldsymbol{z})$, from which the joint distribution may then be reconstructed.

The importance of Bayesian networks lies in the way such a structure supports probabilistic reasoning about many naturally occurring distributions, where the dependencies between variables arise from a relatively sparse network of causal influences.

It has been noted by several authors [5, 7, 8, 13] that a Markov random field with the same joint distribution as a given Bayesian network can be constructed as follows. Take as the cliques of the MRF all

[2] Actually, since the network is bipartite, any nodes belonging to the same layer may be updated in parallel.

[3] This condition is not stated explicitly by Smolensky in [12], but it is required in his proof.

collections of variables of the form $C_i = \{X_i\} \cup \boldsymbol{F}_{X_i}$, i.e., a variable together with all its parents, and define the potential function for clique C_i as

$$V_{C_i}(\boldsymbol{x}) = \ln P(X_i = x_i | X_j = x_j \ \forall X_j \in \boldsymbol{F}_{X_i}) + K_{C_i},$$

where K_{C_i} is an arbitrary constant.

It follows that the realization of MRF's on harmony networks described above can also be used for realizing Bayesian networks. Given a Bayesian network D, let the harmony network contain a pattern node for each value assignment α to a clique of variables C_i; and assign to the parameter λ_α for this pattern node the value

$$\lambda_\alpha = \ln P(X_i = \alpha_i | X_j = \alpha_j \ \forall X_j \in \boldsymbol{F}_{X_i}) - \lambda_{C_i},$$

where λ_{C_i} is some value smaller than any of the $\ln P$ terms for an assignment α associated to C_i. (The shift by λ_{C_i} of all the λ_α's associated to a clique C_i is required to make the parameters positive.) By Smolensky's theorem, the resulting harmony network computes (with high reliability) maximum likelihood completions to given initial partial value assignments. We emphasize that, in contrast to Smolensky's original work [12], there is no "learning" process involved in our construction: both the structure of the harmony network and the values of the λ_α parameters are explicitly determined from the given Bayesian network. Conceivably, though, the learning algorithm suggested by Smolensky could be used for fine-tuning the parameters on the basis of observed data.

3 A Translation Scheme for Conceptual Hierarchies

Let us then apply the technique of implementing Bayesian networks on the harmonium to the problem of translating high-level descriptions of concepts to a neural representation. As a simple example, consider the following description of the inhabitants of a zoo, presented in a generic high-level knowledge representation language:

```
concept animal is basic (100) with
    offspring : [ living (20), eggs (80) ];
    can       : { swim (70), fly (29), walk (49) };
concept mammal is animal (20) with
    offspring : [ living (20), eggs (0) ];
    can       : { swim (10), fly (0), walk (19) };
concept bird is animal (30) with
    offspring : [ living (0), eggs (30) ];
    can       : { swim (10), fly (29), walk (30) };
concept fish is animal (50) with
    offspring : [ living (0), eggs (50) ];
    can       : { swim (50), fly (0), walk (0) };
concept dolphin is mammal (1) with
    offspring : [ living (1), eggs (0) ];
    can       : { swim (1), fly (0), walk (0) };
concept penguin is bird (1) with
    offspring : [ living (0), eggs (1) ];
    can       : { swim (1), fly (0), walk (1) }.
```

Here, a description of a concept consists of a reference to its immediate ancestor (if any), together with a list of attributes and their value distribution for objects belonging to this conceptual class. There are two types of attributes: exclusive (indicated by the square brackets "[ ]" enclosing the list of possible values) and set-valued (indicated by the curly braces "{ }"). For any object, an exclusive attribute must be assigned exactly one value from its list of possible values; a set-valued attribute may possess any number of values (including zero) from its list. The parenthesized numbers indicate the "frequency" of a given value for an attribute, or at the header of a concept declaration, the "frequency" of objects falling into that concept

Figure 2: A Bayesian network for a part of the "zoo" description.

class. These numbers may either be actual objective frequencies, or subjective estimates of the "typicality" of certain contingencies.

One way of representing such a description as a Bayesian network was presented in our earlier paper [9]; here we consider another approach. Based on the given conceptual hierarchy, we generate a complete classification of objects into terminal classes (i.e., leaf nodes in the hierarchy tree). This may require creating new terminal classes; thus, for instance, parallel to the dolphin we create a new class

concept other_mammal **is** mammal (19) **with**
offspring : [living (19), eggs (0)];
can : { swim (9), fly (0), walk (19) }.

A Bayesian network corresponding to the mammalian part of this enhanced classification is shown in Figure 2. In the network, a binary variable is assigned to each concept class and attribute–value pair. To save space in the figure, variable names have been abbreviated in an obvious manner: d for "dolphin", $o\!:\!l$ for "offspring:living" etc. If variable X is a parent of variable Y in the network, the value of $P(Y|X)$ is indicated next to the arc representing the dependency.

As can be seen from the figure, groups of mutually exclusive variables, such as the immediate descendants of a concept variable (m, b, and f for a), or the value variables for an exclusive attribute ($o\!:\!l$ and $o\!:\!e$) give rise to zero-probability dependencies in the Bayesian network. In theory, such dependencies are not allowed, since they give rise to infinitary λ_α values. We have circumvented this problem by simply introducing a small positive constant η, and replacing all the probability 0 dependencies in the network by probability η dependencies.

For lack of space, we do not show the detailed harmonium implementation of the zoo network. In addition to what has been said above, the only notable feature of the implementation is that we are leaving out the very large number of pattern nodes that correspond to conditions with probability 0. Thus, we have nodes corresponding to the patterns *(offspring:living | dolphin)* and *(offspring:living | penguin)*, but not for *(offspring:living | dolphin, penguin)*. The resulting network has 13 feature nodes and 53 pattern nodes.

4 Computational Experience

To empirically test the method developed, we have implemented a simulator, together with a compiler that realizes a given set of high-level concept descriptions as a harmony network. For the compilation, the pattern

Figure 3: The behavior of selected "zoo" network nodes during query processing.

node parameters κ_α were set uniformly to the value κ,

$$\kappa = 1 - 2/(\max_{\alpha} |\boldsymbol{\alpha}| + 1),$$

where α ranges over all the pattern nodes. The parameters λ_α were first initialized to negative values as $\lambda'_\alpha = \ln P(X_i = \alpha_i | X_j = \alpha_j \; \forall X_j \in \boldsymbol{F}_{X_i})$, and the smallest value λ'_{min} of these was stored. The parameters were then rescaled to positive numbers according to the rule

$$\lambda_\alpha = \lambda'_\alpha - 1.1 * \lambda'_{min}.$$

The value of the parameter η, corresponding to the zero probabilities in the Bayesian network, was set to 0.001.

After the compilation the user may perform queries against the knowledge coded in the network. A query consists of a set of initial values for some subset of feature nodes. The query is performed by clamping the chosen nodes to the given values, and letting the network settle down to a final state. For example, the query

> **assume** offspring:living, **not** can:walk.
> **query concept**.

clamps the feature node corresponding to the variable *offspring:living* to the value 1, whereas the node corresponding to the variable *can:walk* is clamped to the value -1. The parameter after the command "query" is used only to limit the set of variable values displayed to the user from the final state. Hence the given query represents the question "which animal in the zoo has living offspring, but can not walk?". An example of the behavior of the feature nodes during one simulation run of the "zoo" network, given the initial state derived from the query above, can be seen in Figure 3. Each data point shows the mean of the calculated state 1 probabilities for the variables, averaged over 500 iteration steps. The network converges to the state corresponding to the right classification *mammal, dolphin, not other mammal, not bird*, etc. In addition to this, it was concluded that the object in question can swin, but it can not fly.

During the simulation, the computational temperature was lowered according to the simple, but frequently used annealing schedule [1]:

$$T_{t+1} = cT_t,$$

where T_t is the temperature after t iterations. In the annealing scheme used, the temperature was lowered from an initial value of 1000 to a final temperature of 0.1 in 9205 steps, using the value 0.999 for the constant c.

The proper choice of the cooling schedule is one of the remaining problems in our approach. Lowering the temperature too fast leads to final states that are not optimal in the sense of the maximum-likelihood measure. On the other hand, lowering the temperature too slowly leads to computationally intractable simulations, at least in a non-neural environment. We are currently investigating this problem further.

References

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* John Wiley & Sons, Chichester, 1989.

[2] J. A. Barnden and J. B. Pollack (eds.), *Advances in Connectionist and Neural Computation Theory, Vol. I: High Level Connectionist Models.* Ablex Publ. Co., Norwood, NJ, 1991.

[3] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

[4] G. E. Hinton (ed.), Special Issue on Connectionist Symbol Processing. *Artificial Intelligence* 46:1–2 (1990).

[5] T. Hrycej, Gibbs sampling in Bayesian networks. *Artificial Intelligence* 46 (1990), 351–363.

[6] R. Kinderman and J. L. Snell, *Markov Random Fields and their Applications.* American Mathematical Society, Providence, RI, 1980.

[7] K. B. Laskey, Adapting connectionist learning to Bayesian networks. *Int. J. of Approximate Reasoning* 4 (1990), 261–282.

[8] S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Stat. Soc., Ser. B* 1989. Reprinted as pp. 415–448 in: *Readings in Uncertain Reasoning* (G. Shafer and J. Pearl, eds.). Morgan Kaufmann, San Mateo, 1990.

[9] P. Orponen, P. Floréen, P. Myllymäki, H. Tirri, A neural implementation of conceptual hierarchies with Bayesian reasoning. Pp. 297–303 in: *Proc. of the International Joint Conf. on Neural Networks (San Diego, CA, June 1990), Vol. I.* IEEE, New York, NY, 1990.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[11] L. Shastri, *Semantic Networks: An Evidential Formalization and Its Connectionist Realization.* Pitman, London, 1988.

[12] P. Smolensky, Information processing in dynamical systems: Foundations of Harmony Theory. Pp. 194-281 in: *Parallel Distributed Processing, Vol. I* (D. E. Rumelhart and J. L. McClelland, eds.). The MIT Press, Cambridge, MA, 1986.

[13] D. J. Spiegelhalter, Probabilistic reasoning in predictive expert systems. Pp. 47–67 in: *Uncertainty in Artificial Intelligence* (L. N. Kanal and J. F. Lemmer, eds.). Elsevier–North-Holland, Amsterdam, 1986.