ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

Accommodating relevance in neural networks

Hui Wang, David Bell

School of Information and Software Engineering
University of Ulster at Jordanstown
N.Ireland, BT37 0QB, UK
e-mail: {h.wang, d.bell}@ulst.ac.uk

Abstract

It is becoming increasingly recognised in AI and elsewhere [3] that it is important to understand and account for relevance in systems for computational and functional reasons. In this paper we give an account of relevance formally, and address the basic issue of how to preserve relevant information and ignore irrelevant information from the perspective of neural networks, with emphasis on the application to pattern recognition. This can provide a basis for further discussion of relevance as well as having immediate significance for neural networks. Our approach is based on a novel associative network model, Moving Around Landscape (MAL) [5], which is based informally on kinetics and is concerned mainly with information transition under constraints. MAL facilitates accommodating relevance in its dynamics to evolve computations that ignore irrelevant aspects of the environments and preserve relevant information. Experiments on its application to character recognition show that accommodating one kind of relevance (topological structure) into MAL enables it to learn relevant internal representations of character images that are comparable in terms of ordinary metrics, such as Hamming distance and Euclidean distance. Thereby character recognition with physical variations is facilitated and the performance of recognition may be improved.

Category: Models and architectures.

1 Introduction

Physical variations abound in many recognition domains, and they are usually irrelevant to the recognition process. In these occasions, it is important to determine what type of information is relevant and what is irrelevant, so that we can design methods to preserve the relevant information and ignore the irrelevant information. Classification can then be made on the basis of the preserved relevant information. Similar problems arise in many other areas, such as knowledge-based systems and machine learning [3].

There has been a recent flurry of interest in explicitly working with relevant information. One focus has been on finding techniques for *improving the performance of embedded agents by ignoring or de-emphasizing irrelevant and superfluous information and making use of relevant information* [3]. This aims at reducing the search space, reducing the training space, taking out physical variations, or improving accuracy.

The term "relevance" is so broad that it can be associated with many different concepts in a number of disciplines including the AI fields of knowledge representation, probabilistic reasoning, machine learning and neural computation, as well as communities that range from statistics and operations research through database and information retrieval to cognitive science [3]. Here we give a mathematical formulation of relevance for the purpose of this paper and further investigation into relevance.

There are two basic issues to be faced when dealing with relevance. One is how to determine what is relevant to a given task. The other is how to preserve relevant information and ignore irrelevant information, thereby, for example, improving the performance of embedded agents. In this paper we are concerned with the second of these issues from the perspective of neural networks (NNs). So we aim at *accommodating relevance in NN* to evolve computations that ignore aspects of the environments and preserve relevant information. We address this issue particularly for the task of pattern recognition.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

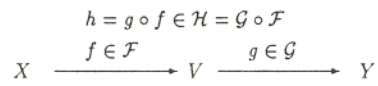

Figure 1: *Scenario of relevance accommodating internal representation*

Generally speaking relevance is a kind of constraint. In order for a NN to be able to accommodate relevance, it must be able to facilitate imposing constraints on its computation. Very few reports can be found in the literature on neural networks capable of accommodating explicit constraints on its computation. In [5] we proposed a novel NN model, Moving Around Landscapes (MAL), which is a kind of associative network addressing the basic problem of *associating from distorted knowledge*. This is in contrast with that addressed by Hopfield networks, which can be characterised as *associating from partial knowledge*. MAL can be intuitively imagined as a set of *particles*[1] moving around a *landscape*. The landscape is formed in the learning phase in response to a given problem. Particles move around according to forces of two types: landscape-oriented and task-dependent. The movement of particles induces the state transition of the network, and hence accomplishes the computation as required. The relevance of interest comes in a form of constraint on the task-dependent forces which affect the state transition of the network, and the computation in the sequel. Therefore MAL is employed here as a platform to accommodate relevance.

Section 2 is committed to the formulation of relevance mathematically. In Section 3 the MAL we use to accommodate relevance is briefly outlined, and Section 4 shows how useful information (i.e., relevant internal representations) can be obtained through accommodating relevance in MAL, and presents some experimental results concerned. The last section concludes with some discussion and directions of future research.

2 Mathematical formulation of relevance

Basically, relevance is a measure of the contribution of a piece of data, a feature, or a collection of data to the performance of a given task. Formally, consider the mapping: $h \in \mathcal{H} : X \to Y$. X and Y are any two spaces called *instance space* and *outcome space* respectively, and $\mathcal{H}$ is a space of functions, where each $h \in \mathcal{H}$ is called a *decision rule* on $X \times Y$. To indicate context, $\mathcal{H}$ is sometimes referred to as $\mathcal{H}_{X \times Y}$.

Definition 2.1 A *generator* of any space X is a set or vector of variables, $\Lambda = (\lambda_1, \cdots, \lambda_k)$, such that when $\lambda_1, \cdots, \lambda_k$ go through all their possible values, Λ traverses X. Any subset of Λ is called a *feature* of X. Denote Ψ_X^Λ as the space of all possible features of X for a given generator Λ. Where there is no ambiguity, Ψ_X^Λ is simplified as Ψ_X or even Ψ.

A generator of a space can be regarded as a *representation scheme* for this space. Just as a space can be represented in different ways, a space can also have many different generators. Since generators determine spaces, and in particular the relevance of spaces to tasks, generators are also referred to as *relevance labels*.

Definition 2.2 Let X be the instance space, Y be the outcome space, Ψ_X be one feature space of X, and $\mathcal{H}_{X \times Y}$ be the decision rule space. *Point relevance* is defined as a function $r_p : X \times \Psi_X \times \mathcal{H}_{X \times Y} \to \{0, 1\}$ such that for any $x \in X$, for any $\psi \in \Psi_X$, and for any $h \in \mathcal{H}_{X \times Y}$, $r_p(x, \psi, h) = 1$ if $\frac{\partial h}{\partial \psi}|_x \neq 0$; $r_p(x, \psi, h) = 0$ otherwise. Let P be the probability distribution of X. *Feature relevance* is then defined as a function $r_f : \Psi_X \times \mathcal{H}_{X \times Y} \to [0, 1]$ such that $r_f(\psi, h) = \int_X r_p(x, \psi, h) dP(x)$. Assume further that Q is the probability distribution of Ψ_X. Then *space relevance* of X to any $h \in \mathcal{H}_{X \times Y}$ is defined as a function $r_s^X : \mathcal{H}_{X \times Y} \to [0, 1]$ such that for any $h \in \mathcal{H}_{X \times Y}$, $r_s^X(h) = \int_{\Psi_X} r_f(\psi, h) dQ(\psi)$.

When X has very low space relevance to the decision rule, the decision making may be inefficient, and therefore we need to transform X to V which has higher space relevance value. A revised scenario accommodating V is shown in Figure 1, where, X and Y are any two spaces as before, and $\mathcal{H}$ is a space of functions, where each $h \in \mathcal{H}$ is now called a *task* on $X \times Y$. h is decomposed as $g \circ f$ [2], where $f \in \mathcal{F}$ is called a *representation rule* or simply *R-rule* of $\mathcal{H}$, and $g \in \mathcal{G}$ is called a *decision rule* of $\mathcal{H}$. V is called a

[1] We use the term *particle* just to facilitate accounting for our model, and do not assume any physical meaning.
[2] $\circ$ is the function *composition* operation: $g \circ f(x) = g(f(x))$.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

representation of X determined by f. Suppose V is generated by Λ, and Ψ_V^Λ is the corresponding feature space. Then the R-rule f may be regarded as Λ-type *feature extraction* operation, denoted f_Λ where needed for emphasis, and V can be regarded as the set of instantiations of the feature space, denoted V_Λ where needed. The expectation that V_Λ has higher space relevance to the decision rule leads to the following formal definition of *relevant representation*.

Definition 2.3 Let P be the probability distribution of X, and P_V be the probability distribution of V which is obtained by $P_V(v) = \sum_{f(x)=v} P(x)$. Let $r_s^X(h)$ be the space relevance of X to $h \in \mathcal{H}$, and $r_s^V(g)$ be the space relevance of V to $g \in \mathcal{G}$. If $r_s^V(g) > r_s^X(h)$, then we say that V is a *relevant representation* of X with respect to $h = g \circ f$, and f is then said to be a *relevant R-rule*.

Next we will give some examples of this formulation.

Example 1: Weather Forecasting. Consider the task h of forecasting whether or not there is rain in London based on the database X in the Meteorological Office in London. In this case the X is taken as the instance space which is the set of meteorological data at different times in London area. Each $x \in X$ consists of many different types of meteorological information. Assume for present purposes that these are temperature, humidity, wind speed and air pressure, which together form the generator of X. However, not all the meteorological information will affect the task h. In order to reduce the search space and speed up decision making, we need to extract from X the information relevant to the task. The extraction results in V, which is a subset of X *relevant to* h. The generator of V is assumed to include humidity and air pressure. The representation rule f is used to extract that relevant information from X. The decision rule g is used to forecast whether or not there is rain in London based on the relevant information. In this case we are mainly concerned with the *space relevance*, and it is easy to know that $r_s^V(g) > r_s^X(h)$. If we are asked "what data contribute to the forecast?" we will probably use the *feature relevance*, and pick up those features that have higher *feature relevance* values.

Example 2: Statistical Learning. Using the same setting as in Example 1, suppose further we are to let computer learn how to accomplish the task h. If we use X as the working data set, and pick up m successful examples from the previous forecast experience as training sample, then it seems obvious that the learning result can not match, in terms of the generalisation ability, similar result in the case where we use V as the working data set and pick up same number of examples as training sample. This is because V is more relevant to the task than X.

This example can be generalised to a general question in statistical learning: **Given a fixed number of training examples, can we improve the performance of the learning results for a given learning algorithm? And by how much can we improve?** To answer these questions, theoretically we could use the *feature relevance* and *space relevance* and transform a less relevant instance space to a more relevant one, and then apply learning algorithm to the samples based on the more relevant instance space.

Example 3: Character Recognition. Consider the task of learning to recognise character, which is mentioned at the beginning of Section 1. Assume that the instance space X is the set of all possible physical image vectors of a given character set, which is the usual case in practice. Since the size of X is usually very large (even infinite in some cases) due to irrelevant physical variations, in order to get certain level of generalisation ability we normally need a large number of training examples [4]. Sometimes this is impractical. Therefore we normally need to transform X into some representations that are much smaller in size but are more relevant to recognition. The relevant representations could be, for example, a set of instantiations of the number of cross-points and the number of end-points, or a set of *topological structures* as used in the example presented in Section 4. Based on these relevant representations, the learned character recogniser will have better generalisation ability [2]. Structural approach to character recognition is one example in this respect. The somewhat implicit concepts used in these approaches are the *feature relevance* and *space relevance*. The generators of relevant representations tend to be invariant of physical variations, and to certain extent capture the most distinctive characteristics of the characters.

The general research into relevance can provide guidelines on how to represent a universe of objects better in order to facilitate decision making and to improve the performance of embedded agents. But in this paper we focus on a more specific problem, that is, how to accommodate prior knowledge of relevance label in MAL, and as an example, by doing so, how to learn relevant representations for character recognition. A more general discussion on relevance from the perspective of machine learning and neural networks is presented in [2].

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

3 Moving Around Landscapes (MAL) network: an outline

MAL is an information transition system which can be perceived intuitively as a set of particles moving around a landscape, and we use this model in the discussions below. A pattern is represented by a set of particles with the same topology as the pattern, placed in a network of units – the landscape. The fact that a particle is placed at a unit corresponds to the activation of this unit, while the fact that no particle is placed at a unit corresponds to the inhibition (deactivation) of this unit. What we are basically concerned with here is the movement of the particles in the landscape due to the relationships between the particles and the characteristics of the landscape.

An *MAL* is a tuple $\mathcal{N} = (\mathcal{M}, \mathcal{L})$, where $\mathcal{M}$ is a finite set of *logical units*, and $\mathcal{L}$ is a landscape (see below) over which the logical units can move.

A *landscape* is a tuple $\mathcal{L} = (\mathcal{U}, \mathcal{C})$, where $\mathcal{U}$ is a finite set of N *physical units*, which is a subset of an n-dimensional Euclidean space; $\mathcal{C}$ is a set of ordered pairs of elements of $\mathcal{U}$ denoting the connections between the elements.

The *mass* of logical unit a, denoted m_a, can be understood as its information content. The *state* of physical unit i, denoted m_i, is the sum of the masses of all the logical units at this unit, which is a measure of the local information at this physical unit.

With each connection $\{i, j\} \in \mathcal{C}$ is associated a *slope* $\mathbf{k}_{ij} \in R^n$, where $\mathbf{k}_{ij} = k_{ij}\mathbf{I}_{ij}$, and $\mathbf{I}_{ij}$ is a unit vector directed from physical unit i to j. k_{ij} is a quantitative measure of the local possibility that information moves from physical unit i to j, or intuitively, a quantitative measure of the physical slope from physical unit i to physical unit j.

A *state configuration* or simply *state* of an MAL is uniquely defined by a point in the N-dimensional Euclidean space, $\mathbf{M}=(m_1, \cdots, m_N)$, whose i^{th} component m_i denotes the state of physical unit i. The *stable state* or *attractor* is a state $\mathbf{M}$ satisfying $\mathbf{M}(t + \Delta t) = \mathbf{M}(t)$.

Where there is no ambiguity we use *unit* for *physical unit*, and for simplicity, we use *particle* for *logical unit*.

The structure of MAL depends on the structure of the landscape. The landscape is usually arranged as a finite *geometrical* regular grid, typically of 2D or 3D, over which the particles can move.

The forces inducing the movement of a particle are determined by the following factors: (1) the features of the landscape; (2) the states of a unit's surrounding units; (3) the neighbourhood relationship between individual particles; and (4) the grouping coefficients of the network. Specifically the forces are divided into *gravitational force*, *resistance force*, and *binding force*. The first two forces arise from the landscape of the network and are independent of application, while the third force is application-oriented. These forces are usually constrained by *grouping coefficients*, which are also related to specific applications.

All the forces above have to be reformulated in different applications, but typically they have the following forms:

- Gravitational force: landscape

  The gravitational force (**GF**) is due to the landscape which is formed in the learning phase. The **GF** exerted on particle a at unit i at time t is typically $\mathbf{GF}_{(a,i)}(t) = m_a \sum_{j \in N(i)} \mathbf{k}_{ij} G_j(t)$, where $G_j(t)$ is the *grouping coefficient* defined below, and $N(i)$ is the set of neighbours of unit i. For brevity, in some cases we denote $\mathbf{GF}_{(a,i)}(t)$ as $\mathbf{GF}_a$ or $\mathbf{GF}_i$. This notation applies to other forces as well.

- Resistance force: landscape

  The resistance force (**RF**) is usually inertial in nature. It is always in the opposite direction to the gravitational force. It is a function of the mass, and typically of the grouping coefficient G_i. The resistance force exerted on particle a at unit i is $|\mathbf{RF}_{(a,i)}(t)| = \kappa m_a G_i(t)$, where κ is a constant.

- Binding force: application-dependent

  The binding force (**BF**) is used to preserve some application-oriented information as the particles move around the landscape. It is actually a kind of constraint on the movement of the particles. For an example of the binding force, see Equation 1 below.

- Grouping coefficient: application-dependent

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

The grouping coefficient (G) is used to guide the movements of the particles. In different applications it may have different forms. Conceptually the grouping coefficient imposed on unit i, G_i, is defined as the probability of unit i being at the *activation* state in the final stable state after initialising the network. It specifies the attractiveness of each individual unit. G_i may change with time.

The grouping coefficients are not used individually; rather they are incorporated into the gravitational forces, resistance forces and even binding forces. The grouping coefficient can appear in two forms: ***plain*** and ***non-plain***. For classification type applications, we do not know in advance which category a pattern falls into, and therefore we simply set $G_i = 0.5$ for $i = 1, 2, \cdots, N$; while for verification type of applications (e.g., combinatorial optimisation or signature verification) we know in advance which category we are interested in, and therefore we can pre-set the G_i's.

With these forces, the dynamics of MAL may be formulated as follows:

The force exerted on particle a at unit i at time t is:

$$\mathbf{F}_{(a,i)}(t) = \alpha \mathbf{G}\mathbf{F}_{(a,i)}(t) + \beta \mathbf{B}\mathbf{F}_{(a,i)}(t) + \gamma \mathbf{R}\mathbf{F}_{(a,i)}(t) = F_{(a,i)}(t)\mathbf{J}_i.$$

If $F_{(a,i)}(t) > \phi$ and $F_{(a,i)}(t) > F_{(a,i')}$ then particle a will move from unit i to unit i', where unit i' is one of unit i's closest neighbours (i.e., one grid away) in the direction closest to $\mathbf{J}_i$. Note: α and γ are constants, while β will usually be scaled down with time until it reaches zero. ϕ is a constant and usually $\phi \approx 0$.

The learning of MAL (i.e., how to make any prescribed set of states, $\{\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_L\}$, as the stable states of the network or in other words how to form the landscape of the network) is accomplished by the gradient descent method. The target function E is defined as follows:

$$E = \frac{1}{2} \sum_{l=1}^{L} \sum_{i=1}^{N} |\mathbf{F}_i^l|^2 = \sum_{l=1}^{L} E^l,$$

where $\mathbf{F}_i^l = \sum_a \mathbf{F}_{(a,i)}^l(0)$ is the force exerted on unit i due to pattern l when the network is initialised. The target function is a measure of the stability of the network. If no particle has potential to move, then the network is stable and H has come to a local minimum; otherwise the network is unstable and H is relatively high.

The learning algorithm is based on differentiation of the target function with respect to the slopes. The differentiation result is:

$$\frac{\partial E^l}{\partial k_{i_0 j_0}} = m_{i_0}^l(0)^2 G_{j_0}^l(0) |\mathbf{F}_{i_0}^l| (\mathbf{J}_{i_0}^l \cdot \mathbf{I}_{i_0 j_0}).$$

Note: $\mathbf{J}_{i_0}^l$ is the unit vector in the direction of the force applied to unit i_0 due to pattern l.

Based on these results it is easy to design an algorithm to undertake the learning task.

4 Accommodating relevance in MAL: learning internal representations

In this section we discuss how to accommodate relevance label in MAL to learn relevant internal representations. In the scenario of Figure 1, we assume that the relevance label Λ of the relevant instance space for the task h is known *a priori*, but the available instance space X is not generated or labelled by Λ, therefore we need a representation rule f_Λ to transform X to V_Λ that can be generated or labelled by Λ and hence it is relevant to the task. In other words, the embedded agent knows in advance what kind of feature is important to the decision, or the agent knows the required V_Λ; however the available data set is not satisfactory. What we are going to do here is to find ways of approximating the f_Λ in the framework of MAL by using the knowledge of Λ and V_Λ – namely, learning internal representations by MAL.

The task of learning internal representations based on MAL is accomplished by training the network with the relevance label Λ specified as a constraint on the binding forces. The R-rule f_Λ is the trained

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 185-192

Figure 2: *Some physical images of 'A' and 'B'*

Figure 3: *Internal representations of the physical images in the figure above learnt using MAL*

(committed) MAL network, the computation of f_Λ is through the dynamic evolution of the network, and the internal representation is the set of all stable states of the network.

For any $x \in X$, to compute the internal representation of x with f_Λ, we must first map x onto the network and then let the network evolve. The evolution preserves the Λ-type of information, which is assumed to be relevant to the decision making and is embodied by the binding forces, and is directed towards a stable state. The final stable state is the internal representation of x.

We have experimented on accommodating relevance in MAL to learn internal representations for the task of character recognition. The instance space X we used in the experiment is composed of the physical images of the 26 English capital letters, with multiple variations. These images are generated using the BITMAP facility in Unix rather than by scanning in order to focus on the effect of internal representation learning and to save effort in pre-processing (e.g., normalisation, skeletonization, and noise reduction). Some of the images of 'A' and 'B' are shown in Figure 2. The outcome space Y is the set of labels of the 26 letters.

The instance space X is rich in physical variations which are obviously irrelevant to the task h of recognition, therefore the relevance of X to h is low. Using examples of this instance space as training sample will result in poor generalisation ability [2]. To improve the efficiency of recognition, we use MAL to learn an internal representation for X that has relatively high relevance. In this case the relevance label Λ we use is *topological structure*, i.e., the neighbourhood relationship among image components, which determines the form of the binding force below; the R-rule f_Λ, i.e., the committed MAL network, extracts the topological structure for each image through its dynamics; the decision rule is to classify character images based on topological structures, and therefore the task is topology-based classification.

The network we use for this purpose is a 32×32 grid, and the binding force takes the following form:

$$\mathbf{BF}_a(t) = m_a \sum_{b \in N(a)} m_b |D(\mathbf{P}_a(t), \mathbf{P}_b(t)) - 1| \mathbf{I}_{ab} \tag{1}$$

where $N(a) = \{b \in \mathcal{M} : D(\mathbf{P}_a(0), \mathbf{P}_b(0)) \leq \delta\}$ is the set of neighbours of particle a, δ is the neighbourhood radius, and $D(x, y)$ is the distance measure in grids between x and y. The grouping coefficients are *plain*. The updating of the state configuration is done sequentially.

We trained the network with one image for each letter. In Figure 2 the first 'A' and 'B' images were used in training. After training, the training examples are made attractors of the network, which are the principal internal representations of the instance space. Other internal representations centre around these principal attractors. Figure 3 shows the corresponding internal representations of the images in Figure 2, reached by accommodating relevance in MAL.

Now we discuss the performance of the internal representation learned. We will discern if the internal representation is a *relevant representation* of X. Suppose the decision rule for recognition is the Hamming distance based discriminant function. Then the condition for a representation to be relevant is equivalent to the following condition

$$\sum_{x,y \in X} [d(f_\Lambda(x), f_\Lambda(y)) - \rho_h(x, y)]^2 < \sum_{x,y \in X} [d(x, y) - \rho_h(x, y)]^2 \tag{2}$$

where d is the normalised Hamming distance metric, ρ_h is the *canonical metric* [1]. In the present case, ρ_h comes as: $\rho_h(x, y) = 0$ if x and y are images of one character, and 1 otherwise. In this particular application

this condition can be explained as saying that relevant representation should be able to make instances in X cluster in terms of ordinary metrics (Hamming distance, Euclidean distance).

The images in Figure 2 vary moderately. It is obvious, however, that the Hamming distance and the Euclidean distance could not be directly used to classify them. For example, in the sense of Hamming distance, the third 'B' is close to the first 'A', but quite distant from any other 'B' except the last. The internal representations in Figure 3 are comparable in the desired sense of Hamming distance, since all the images for 'A' are close together, but far away from those for 'B'. Based on these internal representations, classification can proceed simply by calculating the Hamming distance. Thus we can say that the internal representations reached in this way are *relevant representation* of the instance space, thereby recognization performance can be improved. We believe that this approach to character recognition (i.e., first representing the instance space as another space more relevant to the decision rule, and then making classification with simpler discriminant function based on the relevant information) is a promising direction of research, though a lot of work lies ahead.

5 Conclusion

We have presented a mathematical account of relevance, which can cover many situations where relevance is frequently used, and we have demonstrated its value with a novel NN model, MAL, that can accommodate relevance to improve the performance of applications. Experiments on its application to character recognition show that accommodating space relevance in MAL can facilitate learning relevant internal representations of character images. Thereby recognition with reasonable accuracy can proceed simply by calculating the Hamming distances between character image vectors.

There is little work in the literature on directly accommodating relevance in NNs. Our work is primarily based on the MAL. Yet MAL itself, being novel, is a less studied subject in the NN community, and some theoretical issues remain open. We hope that this paper can draw attention to relevance as well as to the MAL.

References

[1] J. Baxter. *Learning Internal Representation.* PhD thesis, Department of Mathematics and Statistics, Faculty of Science, The Flinders University of South Australia, December 1994.

[2] D. A. Bell and H. Wang. Relevance in learning and neural networks. Technical report, Faculty of Informatics, University of Ulster, April 1995.

[3] Russell Greiner and Devika Subramanian, editors. *Relevance (Papers from the 1994 AAAI Fall Symposium).* The AAAI Press, 1994. AAAI Technical Report FS-94-02.

[4] D. Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Inform. Comput.*, 100:78–150, 1992.

[5] Hui Wang and David Bell. Moving around landscape: a novel associative network accomodating pattern deformation. Technical report, Faculty of Informatics, University of Ulster, 1995. To be presented in World Congress on Neural Networks'95, Washington.