A Computational Study of Reconstruction (from Partial Data) and Anticipation Capabilities of an Associative Neural Net with Large Stored Data-Base

Mitja Peruš
BION Institute, Stegne 21, SLO-1000 Ljubljana, Slovenia
+386-1-513-1147 (phone & fax) – mitja.perus@uni-lj.si – http://www.bion.si/mitja.htm

Abstract
I simulated a large Hopfield neural net which had the signum instead of sigmoid activation function so that it could be naturally physically implemented, e.g. in spin systems. It has been used in computational simulations in order to analyze the following capabilities of processing very large and complex data sets (e.g., protein-structure data-bases): 1. completion of patterns; 2. recognition of patterns; 3. prediction of unknown parameters; 4. anticipation. While for tasks 1–3 we use a memory-base of previously-learned examples using "associations", task 4 is equivalent to case 3 re-interpreted for temporal (or time-series) prediction, i.e. prediction of unknown future parameter-values (instead of unknown present ones). For tasks 3 and 4 it is concluded that a generalization of the model used in simulation, like phase-Hebb processing or quantum-like information dynamics, is more promising. Data-structure conditions for success of tasks 1–4 are discussed in a complex "real-life" example.
Keywords: associative neural nets, pattern recognition, prediction, anticipation

1 Introduction

Computational performance of the following tasks will be discussed and simulated using an extraordinary-large physically-implementable Hopfield-like associative neural net:
• 1. *completion of* (relatively very big) *patterns*: the net completes a set or vector of parameters – it recalls them from a memory-base of previously-learned examples using "associations", i.e. activated correlations of the current partial input with the automatically-selected stored vector;
• 2. *recognition of* (very big) *patterns*: the net recognizes a pattern, indicated by a partial input, either as one of those previously-stored, or as one belonging to a class of (stored) patterns (that's an extension of case 1 by activation of contextual information and by corresponding re-arrangement of the output-vector / attractor);
• 3. *prediction of* (very many) *unknown parameters*: the net predicts unknown parameters based on known ones and on a memory-base of previously-learned examples which were similar to the current new pattern (an extrapolation of case 1, triggered by new-pattern's partial input);
• 4. *anticipation*: time-series prediction by artificial neural nets is a special case of prediction (interpolation, as in case 1 and 2, in a broad sense, and extrapolation, as in case 3, in a narrow sense) using the same mechanism of attractor-reconstruction and attractor re-shaping based on collective system dynamics, but with temporal interpretation of parameters which are encoded in the states of "neurons".

Anticipation is realized by "a system for which the present behavior is based on past and/or present events, but also on future events built from the past, present and future events" (Dubois, 2000a, p. 29). Thus, in an epistemic way, an anticipatory system evolves '*as if* it knows its future' (ibid., p. 5, quotes as in orig.).

As it will be shown in section 7, a Hopfield neural net that reconstructs a stored pattern from memory is an anticipatory system. The net's final state (pattern–attractor) is equal to the selected stored pattern, because the net's evolution towards the output pattern has been guided by the memory. The anticipation abilities could be better, i.e. (more) correct or reliable, or worse, i.e. producing an error, as the time flow will finally reveal. (Only precognition would be anticipation with certainty.)

In the case of net's pattern recognition from memory (task 2), anticipation is maximal and perfect (i.e., 100 % correct) if certain conditions, which will be discussed, are satisfied. In the case of predicting or anticipating values of some variables in *new* circumstances not faced before, i.e. not learned (tasks 3, 4), the prediction or anticipation about future state at time $t+1$ might turn out to be more or less wrong when time goes on to $t+1$. Results of testing such highly non-trivial cases will be presented. Since *anticipation is a special case of prediction, i.e. prediction of future states*, the presented prediction-method and computational tests are usable for anticipation also.

We will finally mention how anticipation capabilities could be enhanced by implementing such an algorithm, which is a *"natural"* (i.e., easily *physically-implementable*) one, in *quantum systems.*

2 Operation of the Basic Associative Neural Net

If all the neurons are bi-directionally connected with all the others, a so-called symmetric, self-consistent network is established – the Hopfield net (Amit, 1989; Palmer et al., 1991). It belongs to the so-called associative neural nets. Among numerous models of neural nets (Haykin, 1994; Peretto, 1992), it is *the most similar to physical complex systems and processes – spin or magnetic systems* (Dotsenko, 1994; Mezard et al., 1987), and even (with some differences discussed in: Peruš, 1998, 2000) to *quantum systems* and *holography.*

A state, or configuration, respectively, of a neural net is described by the vector $\vec{q} = (q_1, q_2, ..., q_N)$. The state of an individual formal neuron (or spin, respectively) i is given by q_i. N is the number of artificial neurons. A neuron can be "active" ($q_i = 1$) or "inactive" ($q_i = -1$, or in another notation: $q_i = 0$). The state of each neuron is determined as the sign $(+1, -1)$ of the sum of contributions from all the other neurons, weighted by the strengths of synaptic connections J_{ij}: $q_i = sgn(\sum_{j=1}^{N} J_{ij}\ q_j)$. (The function sgn gives the sign of its argument.)

Patterns $\vec{v}^k = (v_1^k, v_2^k, ..., v_N^k)$ are those neuronal configurations which *have a special informational significance.* Patterns can be *parallel-distributively* encoded in the system of neurons or stored in the system of connections (weights) which represent the *memory.*

The strengths of connections J_{ij} are determined by the so-called *Hebb learning rule* which comes from neurophysiology. Experiments show that a connection between two neurons with correlated activity is strengthened, but a connection between two neurons with un-correlated activity is weakened. Strengthening (weakening) of connections bases on increasing (decreasing) of the transmission-rate of a synapse. From this observation, the Hebb formula has been constructed: The connection-strength J_{ij} is proportional to the auto-correlations between equilibrium-configurations (stored patterns), i.e. (to be more specific) to the sum of correlation-rates between the i^{th} neuron and the j^{th} neuron in the framework of individual patterns:

$$J_{ij} = \frac{1}{N} \sum_{k=1}^{p} v_i^k v_j^k \qquad (1)$$

p is the number of patterns which are simultaneously stored in the same network; k is the pattern-index. v_i^k describes the role of the i^{th} neuron in the framework of (i.e., while co-realizing) the k^{th} pattern. Construction of matrix $\mathbf{J}$ made of all J_{ij}, which could be gradual or immediate, is equivalent to memory-storage, or the so-called "learning" process, respectively.

The Hopfield model with the Hebb connection-matrix produces patterns which are equivalent to attractors (Amit, 1989; Geszti, 1990; Peruš & Ečimovič, 1998). An *attractor* is a fixed-point of the net's collective dynamics. This is a state where the network stays "for-ever" after it has once occupied it. *Each attractor corresponds to a minimum of net's free energy* [1] (from the point of view of dynamics of the network in the case of physical implementation), or to a *minimum of the so-called net's energy function* (from the point of view of a simulation / model, used for information processing) (Amit, 1989; Geszti, 1990). Attractors are thus configurations which are especially dominant, stable, or usually stationary and act as eigen-states, respectively.

A pattern–attractor is a stable configuration in an energy-minimum, because the Hebb rule determines the connections so that the neurons in connection are, taking into account their "activity-sign" (+ or −), in optimal agreement (correlated). Between two active neurons or between two inactive neurons, there is a positive connection (meaning that the information they encode is in agreement); but a currently active neuron and a currently inactive neuron have a negative connection (i.e., information they encode is not in agreement).

The Hamiltonian of the Hopfield net with specific and symmetric (i.e., $J_{ij} = J_{ji}$) connections is like the usual spin-system or spin-glass Hamiltonian (Mezard et al., 1987):

$$H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{ij} q_i q_j - \sum_{i=1}^{N} B_i q_i \qquad (2)$$

where for connections J_{ij} the Hebb rule (eq. 1) is inserted. The first term incorporates local interactions. The second term describes the influence of an additional global field B onto a neuron (spin) i – environmental influence, for example. To this term, we can encode additional requirements and constraints during simulations.

Because the Hamiltonian (eq. 2) with the Hebb connections (eq. 1) incorporates information-patterns as attractors into the net-process, they regulate the net's dynamics

[1] The free energy F is defined by: $F = E - TS$. E is the (internal, "thermodynamic") energy, T is the "temperature" ("stochastic" neuronal activity), and S is the entropy. Minimization of the free energy is a "compromise" between minimization of energy and maximization of entropy.

so that the energy of the net keeps decreasing (proof in: Peruš & Ečimovič, 1998) and is minimal when the net reconstructs the patterns–attractors.

3 Recall from Memory and Pattern Recognition

3.1 "Holographic-like" Recall or Reconstruction Process

Convergence of the vector describing the net-state ($\vec{q}$) to the nearest attractor, i.e. those which is the most similar to the input-based pattern $\vec{v}^k$, corresponds to *recognition* of that pattern. Namely, this process includes intensive coordination between neurons, and with this, indirectly, also between the parallel-distributed patterns which extend virtually over their constitutive neurons (Peruš & Ečimovič, 1998). The network recognizes a pattern, usually input from environment, when the neurons find an optimal compromise. Neurons with mutual interactions and comparisons optimally re-distribute the information-traces which they encode. In the system of neurons, a superposition of the external (new input) pattern and the internal patterns (from their traces which are stored in the system of connections), is formed. The new pattern is in such a way physically and informationally connected with the context and is so recognized by the net.

This subtle process is executed by a computer using the following simple equation for the reconstruction or *recall of the pattern from memory*:

$$q_i^{output} = \sum_{j=1}^{N} J_{ij}\, q_j^{input} \qquad or \qquad \vec{q}^{output} = \mathbf{J}\, \vec{q}^{input} \tag{3}$$

The Hebb memory matrix $\mathbf{J}$ with elements J_{ij} acts as a projector into the system's subspaces corresponding to individual patterns $\vec{v}^k$ stored in those connections.

We worked with binary (i.e., bipolar) vectors: their components are only 1 or -1 (which is equivalent to 0 in another interpretation). To get the output bipolar as well, we had to use, instead of eq. 3, the following update-rule:

$$q_i^{output} = sgn\left(\sum_{j=1}^{N} J_{ij}\, q_j^{input}\right) \qquad or \qquad \vec{q}^{output} = sgn\left(\mathbf{J}\, \vec{q}^{input}\right) \tag{3b}$$

where the function sgn gives 1 when its argument is positive, else it gives -1. In general (with integer or real input and output values q_i), merely a suitable normalization is needed instead of sgn – if one wants to preserve linearity to enable natural implementation.

If a part of a stored vector (a part of $\vec{v}^1$, for example) is inserted as $\vec{q}^{input}$ (so-called "memory key"), the network reconstructs the whole bipolar vector $\vec{v}^1$ on the output. For example: $\vec{v}^1 = (1,0,1,1,0,1,1,0,1,1)$,
$\vec{q}^{input} = (1,0,1,1,0,0,0,0,0,0)$.
The first five "neurons" or "spins" were put, as an example, to the states which are equal to corresponding states in the chosen stored vector $\vec{v}^1$. The other "zeros" in $\vec{q}^{input}$ are signifying the lack of information. Assume that some conditions are fulfilled which will be discussed soon. After the recall-operation, i.e. $\vec{q}^{output} = sgn(\mathbf{J}\vec{q}^{input})$, the reconstructed whole $\vec{v}^1$ is obtained at the output: $\vec{q}^{output} = (1,0,1,1,0,1,1,0,1,1) = \vec{v}^1$.

From the “key” and **J**, where $\vec{v}^1$ is implicitly encoded, those (the same) $\vec{v}^1$ is reconstructed. Thus, when we can reconstruct a pattern $\vec{v}^k$ from memory **J**, it holds: $\vec{v}^k = \mathbf{J}\vec{v}^k$; or in our concrete example with bipolar vectors: $\vec{v}^k = sgn(\mathbf{J}\vec{v}^k)$. This is the equation which shows that all the patterns $\vec{v}^k$ are *eigen-vectors* of the matrix **J**, where the eigen-value is, in the limit case of the recall of a chosen $\vec{v}^k$, equal to 1 (Amari & Maginu, 1988). This is the result of competition between patterns, described by eigen-vectors $\vec{v}^k$, their relevance being specified by the corresponding *eigen-values* λ^k. The pattern with the biggest initial eigen-value (which is specified by the similarity with the "key" given in the input) pervades – this is $\vec{v}^1$ in our case. "Taking power" by $\vec{v}^1$ is accompanied by increase of λ^1 toward 1; other $\lambda^k, k \neq 1$, decrease toward 0 (Haken, 1991; Peruš & Ečimovič, 1998).

For the reconstruction of $\vec{v}^1$, the following condition, beside the similarity of the “key” with one of the chosen stored pattern ($\vec{v}^1$), must be satisfied: Pattern $\vec{v}^1$ has to be *orthogonal to other stored patterns* $\vec{v}^k, k \neq 1$. Why? We insert an input $\vec{q}\,'$ (the prime ['] denotes similarity of the input with the current net-state $\vec{q}$) and we calculate for each i:

$$q_i^{izhod} = \sum_{j=1}^{N} J_{ij} q'_j = \sum_{j=1}^{N} \left(\frac{1}{N} \sum_{k=1}^{p} v_i^k v_j^k\right) q'_j = \frac{1}{N} \left(\left(\sum_{j=1}^{N} v_j^1 q'_j\right) v_i^1 + \left(\sum_{j=1}^{N} v_j^2 q'_j\right) v_i^2 + \ldots + \left(\sum_{j=1}^{N} v_j^p q'_j\right) v_i^p \right) \quad (4).$$

If $\vec{v}^k, k \neq 1$, are orthogonal to $\vec{v}^1$, then all the terms from the second to the last are equal to 0. The first term, because of similarity of the “key” and $\vec{v}^1$, gives 1 in the bracket (vectors were normalized in the beginning) and so $\vec{v}^1$ is the final, output state.

3.2 Conditions for Undisturbed Recall and for Prediction

This was a trivial case – reconstruction of a stored pattern. Such a recall is not sufficient for prediction or anticipation. For prediction or anticipation, a “fuzzy” version of the same process is needed. The difference is merely in a different structure of correlations between the used "key" and the previously-learned / stored patterns, and in the "rate of orthogonality" (cf., Kainen & Kurková, 1993; Kainen, 1992) between the stored patterns. The memory recall of eq. 3 will not give any "non-sense" output if the stored patterns are *almost orthogonal* between each other (i.e., the scalar products should be close to 0, but not too close). Then, an optimal prediction or anticipation in new circumstances could be obtained – an interpolation or even some limited extrapolation.

The decomposition of eq. 4 can, in such a "fuzzy" case, give the following result: The first term gives a contribution which is *close to* $\vec{v}^1$, because the product in the bracket is close to 1. The other terms together contribute something to be described as a small "noise" – all their products in brackets are numbers that are just a bit bigger than 0. So, all the terms together produce an output-vector which has optimally taken into account the "key" as well as *all* the stored patterns, but mainly that one which is the most similar to the newly-produced (anticipated) pattern.

The recall is very simple, seemingly at least, but it is a very efficient process with implicit selection of relevant information. Such a procedure is very similar to *holography* – specifically, to reconstruction of a stored pattern from a hologram. A hologram (similar to **J**) is produced by an interference of so-called object (direct) wave and so-called reference (indirect) wave. At reconstruction, on the other hand, the hologram is illuminated-through with a "partial" reference-wave — so an image, corresponding to the "whole" reference-wave, is produced (Hariharan, 1996). Therefore, such a memory recall will be called "direct or holographic" recall to distinguish it from a gradual Monte Carlo adaptation.

4 Experience with Classification and Memory-Storage

For a Hopfield net, it is convenient that the learned patterns constitute a suitable number of classes, or attractors, respectively. The ratio of attractors and neurons should be somewhat less than 0.14 (this is my computational experience which confirms earlier theories and simulations described, for example, in: Amit, 1989; cf., Sompolinsky et al., 1985). Above this limit, "catastrophic melting-together" of patterns occurs, and they "average out". It is better if numbers of elements per class are approximately equal. The patterns which constitute an attractor should be more similar to each other than the patterns which belong to different attractors. My rough estimate is that data-similarity (measured, for instance, as a scalar product of binary vectors) inside a class should be, if possible, above 0.8–0.9; between the classes, on the other hand, should be below 0.1–0.2 (so, at least, my "protein case", to be presented later, suggests). In mathematical terms: patterns which belong to different classes–attractors should be such that their binary codes are *approximately-orthogonal* bipolar vectors (cf., Kainen, 1992). On the other hand, patterns which belong to the same class–attractor should be such that they are coded with similar binary (bipolar) vectors. If these conditions are not satisfied, mutual disturbances between patterns cause worse recall, and prediction or anticipation is not possible at all.

If the stored patterns do not form well-distinguishable classes, then there are no strong-enough attractors. The network in the worst of such cases "averages out" all the possible pattern-configurations – this results in a "mean pattern". In the case of inconvenient data-structure and exceeding of the memory capacity, there are, beside the real attractors ("good memory patterns") which are usually more global, also many more local unreal attractors ("spurious attractors" or "spin-glass states"). These are "bad, unclear, transitory memory patterns". This is because the rate of correlation is so high (or: they are "not enough orthogonal") that they disturb each other ("interfere") and so "the noise spoils the signal".

My simulations, which will be presented in the next sections, led to the following observations. The process of *completion, classification and recognition of stored patterns* and, on the other side, the process of *prediction or anticipation* are not coherent, but rather *exclusive and opposing.* These two sorts of processes arise from different tendencies of attractor dynamics, and usually a "subtle, compromising interplay" between these tendencies is needed for optimal performance. As already emphasized, a class or its prototype is formed by an attractor which is stronger if the patterns inside a class are more correlated, and if representatives of different classes are less correlated (it's the best *if they are orthogonal*). If attractors are not strong enough or if they "interfere" too much, the net gives an output that is an "average" ("compromise") over the "interfering" patterns. If the attractors are optimally strong, but not perfectly orthogonal, the net can classify input-data. If the net preserves some optimal "interference", it has associative capabilities necessary for prediction or anticipation based on learned examples. In an extreme case when there are orthogonal classes, but no interference, there are no abilities for prediction or anticipation, but there is a hundred-percent capability of reconstruction of a stored pattern from a partial "key" (i.e., pattern completion). An estimate for the rate of an optimal "creativity-stimulating freedom of interference-like interactions" (namely, prediction demands "some creativity" of the network – highly non-trivial!) depends on the actual data-structure. This estimate is chosen intuitively, after consideration of the correlation-structure of data.

If there are no correlations, the network cannot learn in Hebbian way so that it could even optimally predict or anticipate in new circumstances. It could merely rigidly reproduce a limited number of stored patterns. Note that these recipes are a result of many years of own computer simulation experience with various concrete data. In spite of their importance, it is often very hard to express these recipes mathematically in detail, because of complexity and multiplicity of the process. Articulated intuition helps most.

5 Prediction with "Holographic" Recall is much Faster than with Monte Carlo Adaptation

Modeling with Hopfield-(like) neural nets has two phases. In the phase of "learning", the patterns which have to be "learned" are encoded ("memorized") so that their joint Hebb correlation matrix is "calculated". This matrix describes / represents the "parallel-distributed" memory. It henceforth remains fixed and is used in the second phase – recall from memory, or prediction (anticipation) based on memory and the new "prediction key". For the recall phase, where the "learned cases" are applied in new circumstances, there are two possibilities: the direct recall, which is similar to holographic pattern-reconstruction, and the Monte Carlo adaptation (the net adapts to the new "key" as much as the Hebbian weights allow it).

When the Monte Carlo procedure is used for adaptation / learning, random changes of the "spin"-vector are made in order to achieve optimal agreement among encoded data. For this purpose, the cost-function called "energy" (proportional to disagreement) is tried to be maximally reduced as prescribed by Hamiltonian: $E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij}\, q_i\, q_j$. It implicitly incorporates Hebb's expression (eq. 1) for weights J_{ij} where correlations between all learned patterns are encoded. If eq. 1 is inserted into the above Hamiltonian, the latter becomes: $E = -\frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}(\sum_{k=1}^{p} v_i^k\, v_j^k)\, q_i\, q_j$. In Monte Carlo procedure we accept all those random changes of the elements of $\vec{q}$ which decrease the "energy" E or increase it a little bit only – up to a certain limit (the Metropolis algorithm – see reprint in: Mezard et al., 1987).

As already described in section 3.2, direct recall gives optimal output automatically and immediately. Although the "secret" is in *both cases* in the *well-known Hebb rule, which encodes patterns into the system of weights so that these patterns become the attractors of net-dynamics*, it turned out in my simulations that Monte Carlo adaptation produces virtually the same outputs, but needs very much more time than the "holography-like" direct recall.

Direct recall (multiplication of the Hebb memory matrix with a new input vector – "key") gives equally good results (in case of my concrete data and computer-power) as Monte Carlo in few seconds; Monte Carlo adaptation, on the other hand, lasts in the same case hours or even days. With direct recall it is possible to automatically avoid many problems of Monte Carlo adaptation: waste of time, getting trapped into intermediary and transitory local minima, need of artificial determination of an optimal "temperature", the number of Monte Carlo steps, etc.

TABLE 1a: "Learning" and recall scheme for task 1 or 2

	1D and 2D structure	3D structure
"learning"	learned for all 94 "proteins"	learned for all 94 "proteins"
recall	given for test "protein" (key)	reconstructed by net from memory

TABLE 1b: "Learning" and recall scheme for task 3 (or 4)

	1D and 2D structure	3D structure
"learning"	learned for 93 "proteins"	learned for 93 "proteins"
recall	given for test "protein" (key)	unknown to net, predicted by net

6 Results of Reconstruction (from Partial Data) and Prediction: a Complex Case Study

The *reconstruction* of a previously-memorized data-vector (pattern) from partial new input ("key"), and *prediction* based on the memory and the "key", are presented here in the case of encoded protein structures, i.e. from approximated (simplified) and encoded (binarized) quantitative data on proteins. [2] (See *Table 1.*) However, the purpose of this presentation is not to demonstrate a good model for prediction of the protein 3-dimensional structure! The aim is to present the processing capabilities of a physically-implementable, i.e. spin-glass-like (Mezard et al., 1987; Fisher & Hertz, 1991), associative neural net (*Appendix A*) and potentials for anticipation. (Numerous biochemical and modeling details, available in Peruš, 1999, are not important for our purpose.)

Table 2 presents evaluations of computational reconstructions of individual encoded protein patterns, named with standard codes in the left-side box "proteins" *(task 1, or 2, respectively).*

[2]Multi-level feed-forward neural nets and combined evolutionary (competing) neural nets (Rost & Sander, 1992, 1994) have the best prediction success for secondary structure and thus are better than all biochemical (Cantor et al., 1980, etc.) and artificial-intelligence methods. All attempts, using various methods, to predict *tertiary* structure have up to now been more or less unsuccessful. The previously-mentioned neural nets have merely a few outputs and are thus less suitable for prediction of many coordinates of the 3-dimensional (i.e., tertiary) structure. Therefore, attempts of modeling tertiary structure started with Hopfield-based nets (Bryngelson & Wolynes, 1987; Friedrichs & Wolynes, 1989; Goldstein et al., 1992a,b), in spite of frequent very serious obstacles such as astronomic complexity of protein structure and inconvenient concrete data-structure (classes, correlations between classes, correlations inside a class, etc.) (cf., Stein, 1992; Frauenfelder & Wolynes, 1994).

TABLE 2: Reconstruction of a memorized pattern						
"protein"	K	H	energy	RMS	category	commentary
bgd	37	35	-583	10.66	E	short "protein"
hlm	30	40	-285	11.74	E	short "protein"
gdk	29	29	-3334	11.67	G	-
lh4	29	32	-3334	12.83	G	-
emy	12	16	-6896	8.54	Y	-
mbc	35	33	-7609	9.75	Y	-
mlf	0	0	-9785	0	R	-
mll	0	0	-9778	0	R	-
mlq	1	1	-9760	2.59	R	-
moa	0	0	-9778	0	R	-
mtj	0	0	-9760	0	R	-
yma	14	15	-7060	7.04	Y	-
spe	0	0	-7659	0	Y	SP = min. 0.9
vxd	2	3	-7737	2.32	Y	-
2mgb	0	0	-9747	0	R	-
2mgh	0	0	-9717	0	R	-
2spn	1	1	-9766	2.59	R	-
mlh	0	0	-9785	0	R	-
bmn	37	28	-347	5.81	E	short "protein"
ilk	32	32	-772	7.90	E	short "protein"

First, all known data on all mentioned proteins were used in "learning" (i.e., insertion into the Hebb memory-matrix). After learning, each protein-pattern individually was tried to be reconstructed based on the partial "recall-key" (20 % of totally 593 "neurons"). Reconstruction means the global graphic presentation of a decoded 3-dimensional "protein", made from a "key" of aminoacid sequences and of data on local meso-structure such as α-helices and β-sheets.

Table 3 shows evaluations of responses which the neural net makes after learning of all protein-patterns *except* the one given in the left-side box, with the "recall-key" incorporating *partial* data on this given "protein" *(task 3, or 4 in the case of temporal interpretation of variables)*. The other, unknown data on this "protein" were then interpolated ("predicted") by my Hopfield-like net. The second experiment thus studied responses in somewhat new circumstances from those learned. A response was based on a new key interacting with memories of other "proteins". The network automatically selected the most relevant (similar) examples from memory to produce an optimal response-pattern.

The difference is also that in the case of reconstruction we can compare the net's result with the (tertiary) structure that we know about, but in the case of prediction or anticipation we cannot, since the result is "guessed" by the net.

Since the correlation structure of data is very important for a network with Hebb-type memory, let us see the *rates of structural similarity*, measured with the "INSIGHT"-program in RMS, between some arbitrary proteins from the DSSP-base. Classes are denoted as R, Y and G (see *Appendix B*):

1. **RMS-differences between pairs of "learned proteins" from two different classes:**
RMS (`gdl-hrm`: G–Y) = 5.10 Å / RMS (`gdl-mlf`: G–R) = 4.90 Å / RMS (`mlf-hrm`: R–Y) = 3.89 Å

TABLE 3: New response based on similar memories						
"protein"	K	H	energy	RMS	category	commentary
bgd	37	35	-354	10.66	E	short "protein"
aep	34	41	-818	12.95	E	short "protein"
hlm	30	40	-60	11.75	E	short "protein"
gdi	29	29	-3117	11.66	G	-
gdj	28	29	-3067	10.72	G	-
gdk	29	29	-3116	11.67	G	-
gdl	31	26	-3142	10.77	G	-
gdm	31	30	-3154	10.77	G	-
lh4	29	32	-3118	12.83	G	-
2gdm	31	30	-3154	10.77	G	into R-category
2lh3	30	33	-3116	12.45	G	-
2mgg	0	0	-9519	0	R	-
bvc	3	5	-7526	4.57	Y	-
emy	12	16	-6884	8.54	Y	-
fcs	0	0	-9390	0	R	-
fip	38	52	-852	12.42	E	short "protein"
hrm	43	40	-6898	15.13	Y	into R-category
hsy	12	14	-7048	7.33	Y	-
lhs	43	46	-5611	13.31	Y	-
lht	2	2	-5764	1.87	Y	-
mbc	35	33	-7414	9.75	Y	-
mbd	39	42	-7433	14.84	Y	-
mbo	40	41	-7434	14.84	Y	-
mbs	37	34	-6324	9.62	Y	into R-category
mcy	0	0	-9428	0	R	-
mgn	0	0	-9510	0	R	-
mlf	0	0	-9579	0	R	-
mlj	1	3	-9571	0	R	-
mlk	0	0	-9579	0	R	-
mll	0	0	-9572	0	R	-
mlm	1	1	-9557	2.59	R	-
mln	0	0	-9579	0	R	-
mlq	1	1	-9550	2.59	R	-
mlr	0	0	-9552	0	R	-
mls	1	1	-9759	2.59	R	-
mlu	35	32	-9278	10.37	R	SP = max. 0.8
moa	0	0	-9572	0	R	-
mob	34	34	-9272	10.13	R	SP = max. 0.8
moc	0	0	-9523	0	R	-
mod	0	0	-9523	0	R	-
moh	41	55	-1481	14.86	E	-
mtj	0	0	-9552	0	R	-
mtk	0	0	-9579	0	R	-
mym	0	0	-9559	0	R	-

continuation of the Table 3 on the next page

"protein"	K	H	energy	RMS	category	commentary
spe	0	0	-7447	0	Y	SP = min. 0.9
swn	1	2	-7526	2.32	Y	SP = min. 0.9
tes	0	0	-9272	0	R	-
vxc	1	2	-7526	2.32	Y	SP = min. 0.9
vxe	37	41	-7343	13.61	Y	-
vxf	39	42	-7433	14.84	Y	-
vxg	39	42	-7433	14.84	Y	-
vxh	39	40	-7380	13.74	Y	-
yma	14	15	-6849	7.04	Y	-
ymb	43	46	-7009	12.00	Y	-
ymb	43	39	-6858	13.31	Y	-
2hif	35	34	-382	10.96	E	short "protein"
2mb5	40	42	-7367	15.61	Y	-
2mga	33	35	-9283	10.46	R	SP = max. 0.8
2mgb	0	0	-9540	0	R	-
2mgc	1	1	-9489	2.59	R	-
2mgd	0	0	-9490	0	R	-
2mge	0	0	-9490	0	R	-
2mgf	0	0	-9523	0	R	-
2mgh	0	0	-9510	0	R	-
2mgi	0	0	-9523	0	R	-
2mgk	0	0	-9579	0	R	-
2mgl	0	0	-9579	0	R	-
2mgm	1	1	-9570	2.59	R	-
2mya	2	3	-7528	2.32	Y	SP = min. 0.3
2myc	39	42	-7433	14.84	Y	-
2spl	0	0	-9579	0	R	-
2spm	1	1	-9557	2.59	R	-
2spn	1	1	-9557	2.59	R	-
2spo	1	2	-9559	2.33	R	-
bvd	6	7	-7487	2.17	Y	SP = min. 0.3
ilk	32	32	-544	7.90	E	short "protein"
irc	0	0	-7386	0	Y	SP = min. 0.3
bmn	37	28	-104	5.81	E	short "protein"
mbw	6	8	-9473	3.06	R	-
mlg	0	0	-9577	0	R	-
mlh	0	0	-9577	0	R	-
mlo	1	1	-9555	2.59	R	-
mti	5	7	-9502	2.17	R	-
vxa	1	3	-7219	0.003	Y	SP = min. 0.3
vxb	11	13	-7125	4.32	Y	SP = min. 0.3
vxd	2	3	-7526	2.32	Y	SP = min. 0.3
2cmm	2	4	-7524	3.37	Y	SP = min. 0.3
2mgj	0	0	-9508	0	R	-
2mm1	37	41	-6865	14.19	Y	-
2myb	38	42	-7397	14.74	Y	-
2myd	1	2	-7526	2.32	Y	SP = min. 0.3
2mye	1	2	-7526	2.32	Y	SP = min. 0.3
4mbn	39	40	-7380	13.74	Y	-
5mbn	2	3	-7528	2.32	Y	SP = min. 0.3

2. **RMS between pairs of "learned proteins" from the same class:**

2a. class G: RMS (`gdl–gdm`) = 0.43 Å

2b. class Y: RMS (`hrm–yma`) = 0.69 Å / RMS (`hrm–vxd`) = 0.96 Å / RMS (`vxd–yma`) = 0.74 Å

2c. class R: RMS (`mlf–2mgh`) = 0.39 Å / RMS (`mlf–mlh`) = 0.10 Å / RMS (`mlh–2mgh`) = 0.41 Å

Proteins in our data-base are thus quite similar: mostly inside the same class, but they are less similar if belonging to different classes.

For our purposes, it is sufficient to note low values of the three measures of recognition-error: K (class error), H (number of wrong coding "neurons"), RMS (coordinate distance or discrepancy), and much negative values of "energy". (For detailed descriptions of these "distance measures" and details on the test-simulation in the case of "proteins" see *Appendix B.*) "Energy" refers to the Hopfield energy-function which has a more negative value if there is less reconstruction-failure. Several protein categories were found in the Brookhaven protein data-bank (cf., Murzin et al., 1995; Kabsch & Sander, 1983). The net's attractors were shown to correspond to categories with codes G, Y, R. E denotes all extra (usually short) protein-patterns not belonging to any category / attractor.

The prototype R was almost orthogonal (i.e., not similar) to other prototypes, therefore reconstruction of a R-type "protein" (its 3-dimensional structure) from partial input-data (2-dim. and 1-dim. structure, i.e. aminoacid sequences) was almost 100 % perfect (H, K, RMS zero or very small). Almost 60 % of all the "proteins" used in the computational experiment were reconstructed with such a high success — they were mainly members of the strongest R-category.

The prototypes G and Y were similar (as measured by the scalar product SP: minimum 0, maximum 1) to each-other, but not to R. Therefore, there were memory-disturbances ("interferences") between Y and G, which resulted in unclear ("mixed") responses, indicated by higher value of H, K and RMS, for Y and G. Some Y-type "proteins" converged to the R-attractor, because of having some similarity (denoted by "SP = min[imally] 0.3" in Table 3) with the R-prototype which constituted "far the strongest lobby". Some other Y-type "proteins" were similar to each-other (see "SP = min[imally] 0.9"), so they constituted an independent (small) Y-attractor. G-type "proteins" were, because of being too few, always attracted into "foreign" attractors.

There were only few R-type "proteins" being less similar to the R-prototype (see "SP = max[imally] 0.8"), so their reconstruction was not successful. These, and other "proteins" with H- and K-errors about 30 to 40, converged to a wrong attractor, called "spurious attractor". This was unavoidable in this case because of an "ambivalent" key shared by more than one attractor. The results demonstrate dependence of pattern recognition on categorization, i.e. on memory-based prototypes (attractors).

7 Conclusions: Improving Anticipation

For our systematic computational research of data-processing tasks 1–4, we used various large data-bases, including finally the encoded information on the structure of "proteins" (up to the 3-dimensional arrangement of protein constituents like α-helices and

β-sheets). Our Hopfield-like network has stored in its Hebbian content-addressable associative memory about hundred pattern-codes of protein's 1-, 2- and 3-dimensional structure. Then, capabilities 1, 2, 3 have been studied for individual unknown patterns (e.g., encoded proteins): the part of the new vector encoding the protein 3-dimensional structure was reconstructed ("protein was recognized or classified") or predicted.

Tasks 1 and 2 work perfectly, if certain conditions (see section 3.2) are satisfied, but task 3 is highly non-trivial (therefore, our results are good for correlated cases only, and are relevant for studying net's prediction-potentials rather than for biochemistry).

According to Dubois (2000a), an incursive anticipatory system is the one where its state q at time $t + \Delta t$ depends on the state at present time t (in other cases also at past times $t - \Delta t_n$, ...), but also at future time $t + \Delta t$. This is described, in general, by: $q(t + \Delta t) = f[q(t), q(t + \Delta t), p]$, where p is a parameter (unimportant for us here).

For pattern reconstruction in a Hopfield net or a similar one (like ours), the final state of dynamics is: $\vec{q}(t + \Delta t) = \vec{v}$, where $\vec{v}$ is a selected pattern. Indeed, after eq. 3, $\vec{v} = f[q(t), \vec{v}, p]$, where $\vec{v}$ is/are hidden in the weights as in the Hebbian equation – eq. 1. Thus, the process of Hopfield-(like) pattern reconstruction, or memory recall, respectively, *incorporates incursive anticipation* (cf., Dubois, 2000a; Nadin, 2000; et al. ibid.; Dubois, 2000b; Dubois & Nibart, 2000.)

In the case of our-net's prediction and anticipation in *new* circumstances (i.e., prediction of an *unknown* pattern or future state) the final state (pattern $\vec{v}$) is adaptable, i.e. changeable according to influences of the stored patterns, hidden in matrix $\mathbf{J}$, and the current "key". It depends on flexibility of the final attractor formation whether prediction or anticipation will be successful, i.e. correct, or not. In the case of anticipation, we can only wait and compare later.

To improve the rate of success, some sophisticated physical implementations of the net or their simulations could be tried. Our artificial neural net is roughly physically implementable (and that's why it was chosen!) in magnetic systems like spin glasses (Dotsenko, 1994; Fisher & Hertz, 1991) or, analogically, in quantum systems (Peruš, 1998, 2000). The quantum implementation would improve the prediction and anticipation abilities by introducing the processing of (oscillatory) phase, thus replacing the Hebb rule by the phase-Hebb rule, i.e. $J_{hj} = \sum_k v_h^k v_j^k e^{-i(S_h^k - S_j^k)}$ (phase-difference in the exponent!), as in Peruš (2000) (cf., Ezhov & Ventura, 2000). An analogical alternative, using phase-Hebbian processing, is so-called Holographic Neural Technology, a sort of simulated holography (Sutherland, 1990). However, success could not be great, except for simple cases, unless some more fundamental (necessarily quantum-field-based, I think) anticipation possibilities as debated at the CASYS conferences could be incorporated in the net-dynamics. For instance, the quantum forward and backward propagations (i.e., dynamics due to anti-parallel time-arrows), Ψ and Ψ^*, suggest new, subtle anticipation capabilities ($\Psi\Psi^* = |\Psi|^2 ...$) — see, especially, Dubois (2000a), Dubois & Nibart (2000), Nadin (2000) for review and further references. I will discuss them in a future work.

Appendix A: Algorithm for Prediction

Our computer program incorporates the following steps (details in Peruš, 1999):

– input of selected DSSP-bases and "learning" parameters;
– for each DSSP-base of an individual protein, DSSP-data are *"translated"* into binary (i.e., bipolar) spin values of $\vec{v}^k$ (as a sub-task, Cartesian coordinates of C_α-atoms of residua are transformed into so-called internal coordinates d, φ, τ);
– *calculation of the Hebb memory matrix* **J** following prescription eq. 1, i.e. $J_{ij} = \frac{1}{N}\sum_k v_i^k v_j^k$, based on the known examples $\vec{v}^k, k = 1, ..., p$ ($p = 93$ or 94 in our case);
– input of DSSP-base of a selected test protein and "trans-coding" of its primary plus secondary part into a simplified binary "key";
– "trans-coding" of the tertiary part of DSSP-file of the test protein into a set of lengths of "cylinders" [3] and a set of angles φ and τ created by the cylinders — these are the correct coordinates for testing the prediction capability (details in Peruš, 1999);
– *Monte Carlo adaptation* (tertiary part of the test vector randomly changes so that, on average, "energy" decreases) or *direct recall* (tertiary part obtained by multiplication of the Hebb matrix with the "key") – the result is a completed tertiary part of the test binary protein-pattern *(task 1)*;
– decoding of the completed tertiary part from a binary code into internal coordinates (angles φ in τ) or their classes, and their transformation into Cartesian coordinates;
– comparison of the correct and the predicted structure spin by spin (Hamming distance) and angle by angle; calculation of discrepancy between the correct and the predicted coordinates for the test "protein" (measured in "distances" K on the level of classes of internal coordinates and/or RMS-errors between Cartesian coordinates); visual comparison of rotatable 3D graphics (RASMOL).

Appendix B: Error Measures: K, H, RMS

In Table 2 (for reconstruction) and Table 3 (for prediction), from left to right, the columns incorporate values of the following quantities (described later): standard abbreviation of the reconstructed (Table 2) / predicted (Table 3) "protein"; class-error K, Hamming distance H, "energy" E, RMS-error in Å ($= 10^{-10}$ m) *between the correct polygon of cylinders and the reconstructed* (Table 2) */ predicted* (Table 3) *polygon of cylinders*; protein class (after a rough informal classification); commentary (sometimes).

In each row of Table 2, the result of reconstruction, or recall, respectively, of an individual protein-pattern (given in the left-most column) is shown after its primary plus secondary part has been used as the "key". All 94 proteins (all their parts) have been used for learning. Cf., Table 1a.

In each row of Table 3, the result of prediction is shown for a case where the primary plus secondary part of a given (in the left) protein was used as the "key" and *all the other* 93 proteins (all their parts) for learning. Cf., Table 1b.

Descriptions of the error-measures:

K: The correct values (calculated from the DSSP base) and the predicted values (decoded output of the net) of individual angles φ (angles between inter-atomic links) and dihedral angles τ were compared. If they differed for one class (less than 45 degrees), the "error" K increased for 1; if they differed for two classes (45-90 degrees), K increased for

[3] A cylinder is an approximative envelope of a protein chain.

2, etc. K was summed over all the angles φ and τ, and so the total discrepancy of the prediction from reality was obtained measured in the "class-distance" K.

H: The rate of similarity of binary vectors was measured with scalar product or with so-called Hamming distance. The Hamming distance is the number of differences in individual elements (corresponding to each other) between two vectors. The scalar product is more convenient if we would have real-valued "neurons" in the Hopfield or similar net (the theory is more developed for such a case). The Hamming distance is more convenient for binary coding.

RMS: The rate of structural similarity of stereo-forms is measured by so-called "root-mean-square error": $RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[(x_n - x_p)^2 + (y_n - y_p)^2 + (z_n - z_p)^2]}$, where (x_n, y_n, z_n) are the predicted coordinates of C_α-atoms (the out-most ones) of approximative polygons of cylinders, (x_p, y_p, z_p) are the correct coordinates from the DSSP-base, and N is the number of all C_α-atoms.

Acknowledgements: The reported simulations were done at the National Institute of Chemistry, Lab. for Molecular Modeling and NMR, in Ljubljana. I am grateful to collaborators for help. Special thanks to Dr. F. Avbelj. However, present interpretation of previously unpublished results has been finished later. Support of Asbl CHAOS and MŠZŠ is acknowledged.

References

- S. Amari, K. Maginu (1988). Statistical neurodynamics of associative memory. *Neural Net.* **1**, 63-73.
- D. Amit (1989). Modeling Brain Functions (The world of attractor neural nets). Cambridge University Press, Cambridge.
- J.D. Bryngelson, P.G. Wolynes (1987). Spin glasses and the statistical mechanics of protein folding. *Proceed. Natl. Acad. Sci. USA* **84**, 7524-7528.
- C. Cantor, P. Schimmel (1980). Biophysical Chemistry – part I: The Conformation of Biological Macromolecules. Freeman, San Francisco. G. Schulz, R. Schirmer (1979). Principles of Protein Structure. Springer, New York. F. Avbelj (1992). *Biochem.* **31**, 6290-6297. F. Avbelj, J. Moult (1995). *Biochem.* **34**, 755-764. F. Avbelj, L. Fele (1998). *Protein Sci.*, in press. J. Bascle, T. Garel, H. Orland (1993). *J. Phys. I (France)* **3**, 259-275.
- V. Dotsenko (1994). An Introduction to the Theory of Spin Glasses and Neural Networks. World Scientific, Singapore.
- D.M. Dubois (2000a). Review of incursive, hyperincursive and anticipatory systems – foundation of anticipation in electromagnetism. In: D. Dubois (Ed.). *AIP Conference Proceedings*, vol. **517**: Computing Anticipatory Systems – CASYS'99: 3rd Int. Conf. in Liege. American Institute of Physics, Melville (NY), pp. 3-30.
- D.M. Dubois (2000b). Non-locality property of neural systems based on incursive discrete parabolic equations. *Int. J. Comput. Anticip. Sys.* **7**, 233-244 (CASYS'99, ed. D. Dubois).
- D.M. Dubois, G. Nibart (2000). Towards a computational derivation of a dual relativity with forward-backward space-time shifts. *Int. J. Comput. Anticip. Sys.* **5**, 25-37 (CASYS'99, ed. D. Dubois).
- A. Ezhov, D. Ventura (2000). Quantum neural nets. Ch. 11 in: N. Kasabov (ed.). Future Directions for Intelligent Systems and Information Sciences. Physica-Verlag/Springer, Heidelberg.

– K.H. Fisher, J.A. Hertz (1991). Spin Glasses. Cambridge Univ. Press, Cambridge.
– H. Frauenfelder, P.G. Wolynes (1994). Biomolecules. *Phys. Today*, February no., 58-64.
– M.S. Friedrichs, P.G. Wolynes (1989). Toward protein tertiary structure recognition by means of associative memory Hamiltionians. *Science* **246**, 371-373.
– T. Geszti (1990). Physical Models of Neural Networks. World Scientific, Singapore.
– R. Goldstein, Z. Luthey-Schulten, P. Wolynes (1992). Optimal protein-folding codes from spin-glass theory. *Proceed. Natl. Acad. Sci. USA* **89**, 4918-4922.
– R. Goldstein, Z. Luthey-Schulten, P. Wolynes (1992). Protein tertiary structure recognition using optimized Hamiltonians with local interactions. *Proc. Natl. Acad. Sci. USA* **89**, 9029-33.
– H. Haken (1991). Synergetic Computers and Cognition (A Top-Down Approach to Neural Nets). Springer, Berlin.
– P. Hariharan (1996). Optical Holography. Cambridge Univ. Press, Cambridge.
– S. Haykin (1994). Neural Networks. MacMillan, New York.
– W. Kabsch, Ch. Sander (1983). Dictionary of protein secondary structure. *Biopolymers* **22**, 2577-2637.
– P. Kainen (1992). Orthogonal dimension and tolerance. Tech. report (Industrial Math., Washington, DC) IM-061592.
– P. Kainen, V. Kurková (1993). Quasiorthogonal dimension of Euclidean spaces. *Appl. Math. Lett.* **6** (3), 7-10.
– M. Mezard, G. Parisi, M.A. Virasoro (1987). Spin Glass Theory and Beyond. World Scientific, Singapore.
– A. Murzin, S. Brenner, T. Hubbard, C. Chothia (1995). SCOP: A classification of proteins database for the investigation of sequences and structures. *J. Molec. Biol.* **247**, 536-540.
– M. Nadin (2000). Anticipation – a spooky computation. *Int. J. Comput. Anticip. Sys.* **6**, 1-47 (CASYS'99, ed. D. Dubois).
– R. Palmer, J. Hertz, A. Krogh (1991). Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City.
– P. Peretto (1992). An Introduction to the Modeling of Neural Nets. Cambridge Univ. Press.
– M. Peruš (1998). Common mathematical foundations of neural and quantum informatics. *Zeitschr. angewan. Math. & Mech.* **78** (S 1), 23-26.
– M. Peruš (1999). Razporeditev α-vijačnic v proteinskih molekulah: Analiza z modelom spinskih stekel. Tech. report, NIC, L01.
– M. Peruš (2000). From neural to quantum associative networks: A new quantum "algorithm". In: D. Dubois (Ed.). *AIP Conference Proceedings*, vol. **517**: Computing Anticipatory Systems – CASYS'99: 3rd Int. Conf. in Liege. American Institute of Physics, Melville (NY), pp. 289-295.
– M. Peruš, P. Ečimovič (1998). Memory and pattern recognition in associative neural networks; *Int. J. Appl. Sci. & Comput.* **4**, 283-310.
– B. Rost, Ch. Sander (1992). Prediction of protein secondary structure at better than 70 % accuracy. *J. Molec. Biol.* **232**, 584-599.
– B. Rost, Ch. Sander (1994). Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins: Struc., Func. & Genet.* **19**, 55-72.
– H. Sompolinsky, D.J. Amit, H. Gutfreund (1985). Storing infinite numbers of patterns in a spin-glass model of neural networks. *Phys. Rev. Lett.* **55**, 428-431.
– D.L. Stein (1992). Spin Glasses and Biology. World Scientific, Singapore – especially: P.G. Wolynes: Spin Glass Ideas and the Protein Folding Problems; pp. 225.
– J.G. Sutherland (1990). Holographic model of memory, learning and expression. *Int. J. Neural Sys.* **1**, 256-267.