INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Progressive retrieval on hierarchical associative memories

João António Rodrigues Sacramento

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Presidente:	Prof. Ana Maria Severino de Almeida e Paiva
Orientador:	Prof. Andreas Miroslaus Wichert
Vogal:	Prof. Jan Gunnar Cederquist

Outubro 2009

to Matilde

Acknowledgements

I cannot be more grateful towards Prof. Andreas Wichert who, like a friend, has guided me out of trouble and into the world of science. He has opened my eyes to the beauty and creativity of applied mathematics, brain theories and philosophy. I would also like to thank Prof. Ana Paiva for her kindness and an enlightening talk during a difficult phase of my life.

A big thank you to my friends António Bebianno, Diogo Rendeiro and Luís Tarrataca for following every stage of this work and always helping through.

Of course, I cannot finish without thanking my parents and my sister, and especially Matilde, to whom I dedicate this thesis, for all her love and support.

Abstract

In this work we explore an alternative structural representation for Steinbuch-type binary associative memories. These neural networks offer very generous information storage capacities (both asymptotic and finite) at the expense of sparse coding. However, the original retrieval prescription performs a complete search on a fully-connected network, whereas only a small fraction of units will eventually contain desired results due to the sparse coding requirement.

Instead of modelling the network as a single layer of neurons we suggest a hierarchical organization where the information content of each memory is a successive approximation of one another. With such a structure it is possible to enhance retrieval performance using a progressively deepening procedure. Collected experimental evidence backing our intuition is presented, together with comments on eventual biological plausibility.

Key words: Associative memory, Steinbuch model, structural representation, hierarchical neural network, sparse coding

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Men possess fascinating innate cognitive capabilities in many senses completely unrivalled by any other system. They have puzzled many scientists of the most diverse fields, especially since the exponential escalation in computational power began. With ever-enhancing information processing technologies at reach, it has been believed by many notables since as far as the late forties that artificially intelligent beings would finally evolve.

However, after decades of improvements on the underlying technology, the list of grand open challenges remains almost unchanged. We have yet to implement the first truly capable cognitive computer system; either influenced by neurobiology or not. Even a basic and foundational task such as memorizing and recalling information is still at a sub-par level when compared to our brain. A computer may possess the ability of mass exact storage, but when it comes to approximative matching or content-based retrieval, difficulties arise.

Engineering and science sometimes tend to overlook the simplest of structures and solutions in favour of more complex ones. It has been stated repeatedly that simplicity and elegance walk hand in hand with each other and that such designs, although quite obvious once presented, are within the most difficult to conceive or discover.

On this work we focus on a set of remarkably simple architectures, with their roots set on the first artificial brain-inspired computational models (McCulloch and Pitts, 1943). Mathematical realizations of artificial neural networks have found widespread adoption as function prediction or general statistical tools, but since their inception they have also been used to model associative or mapping memories.

Unlike random access memories, ubiquitous these days thanks to the popularity of the John von Neumann sequential architecture (Burks et al., 1946), associative memories are simultaneously a model for data storage and compu-

tation. Their general task is to establish an approximative mapping between pairs of strings of information, where there is no distinction between address or content as in the common sense. Both can be served as a key to fetch one another, and both strings convey useful information. Also not alike random access memories, their approximative nature confers them interesting properties such as some degree of noise and incompleteness tolerance.

We will devote particular attention to one of the simplest and oldest formulations of this kind of device. Steinbuch- or Willshaw-type associative memories, named in honour to the separate groundbreaking works of Steinbuch (1961) and Willshaw et al. (1969), were among the first neural networks to introduce a nonlinearity on their retrieval process. As we will see in greater detail, they deserve special treatment thanks to their impressive storage capacity. When operating on certain regimes, they can store information nearly as densely (about 69 per cent as densely) as trivial listing or random access memories with no mapping abilities.

Steinbuch memories are also quite attractive for brain modelling due to their biological plausibility. Of course, they correspond to overly abstract structures where many important factors such as chemical binding or neuron diversity were left out. But if synaptic plasticity, a form of inter-neuronal connectivity closely related to the Hebbian cell assembly theory (Hebb, 1949, Braitenberg, 1978), is ever proved as the basic functioning pillar of memory, these simple memories could be a definite starting point for implementing an artificial neural device. In fact, they have been used as the basis for more complex systems, e.g., highly fault tolerant networks of spiking neurons with temporal sensitivity (Sommer and Wennekers, 2001, Knoblauch, 2003b).

Throughout this thesis we will also revisit and present in a comparative fashion a series of prominent models of neural associative memory, possibly unknown to a computer science audience. Some of them have nonetheless achieved fame and high interest within other communities; especially noteworthy has been the case of Hopfield networks and the high-end physics community of the eighties and nineties.

Our major contribution, however, will be centered around the retrieval efficiency of Steinbuch memories, a classical research aspect of the model which has been subject of much former discussion (Schwenker et al., 1996, Sommer and Palm, 1999, Knoblauch, 2005). Even if their retrieval performance is already above par, improvements are more than welcome not only for the obvious benefits for large-scale technical applications but also because *a*) even local cortical networks can easily reach sizes of millions of neurons (Braitenberg and Schüz, 1998); *b*) their storage efficiency increases with size (Palm, 1980); and *c*) metabolic energy constraints mandate so (Lennie, 2003).

Structure and organization of this work

This work can be divided roughly into two parts. The first one corresponds to chapter 2, where we present a detailed survey of the classical models of associative memory. Readers already familiar with the subject should probably skip directly to chapter 3, but not without first reading section 2.2 where we introduce the terminology which will be used throughout the rest of the thesis.

On the remaining chapters we are concerned with our novel retrieval method. Chapter 3 presents both an intuitive view and a rigorous mathematical formulation. We also discuss its optimal working regimes on sequential implementations. Further along on chapter 4, a collection of experimental evidence is considered, reflecting the performance impact and benefits of our contribution. We finish with a discussion on chapter 5 on the open edges and more intrincate implications, with a special regard to neurobiology.

Chapter 2

Neural associative memory

Associative memory models have been around for about half a century now. They have attracted many researchers thanks to their concealed promise of artificially replicating some functions of the human brain. Even if current results are nowhere near the full processing capabilities of their biological counterpart, associative memories have already been successfully applied to engineering fields as vast as image processing, pattern recognition, natural speech or mathematical programming.

Despite countless scientific contributions available in the literature, associative memories still pose interesting questions to be addressed. As we will see, their performance when dealing with natural, non-random data is extremely low. Without special data coding prescriptions, mostly undiscovered yet, their full potential as a useful engineering tool remains out of reach.

The first part of this thesis will be written in the style of a survey, highlighting and reviewing the most relevant advances and open challenges, hopefully presenting them in a simple fashion to an outside reader. We will then proceed to introduce and explain the basic intuitions and concepts behind our current research work on the performance enhancement of associative memories.

2.1 From the human memory to the first artificial model

The memory function of the human brain has long been subject of study, garnering the efforts of philosophers, mathematicians, physicists, biologists and thinkers alike. Credited as one of the fathers of western philosophy, Aristotle stated around 350 BC a set of observations on human remembrance and recall later regarded as the *Classical Laws of Association*. A book of modest

dimension, *On Memory and Reminiscence* (Sorabji, 1972) was nevertheless a breakthrough, spawning innumerous subsequent work. A simple, modern enunciation of Aristotle's four laws commonly found in the literature is:

The law of contiguity Things or events occurring close to each other in space or time tend to get linked together (*spatial contact*).

The law of frequency The more often two things or events are linked, the stronger will be that association (*temporal contact*).

The law of similarity Similar things will tend to trigger the thought of one another.

The law of contrast Opposite mental items also tend to trigger the thought of one another.

These laws were revisited by Kohonen on his classic work *Self-Organization and Associative Memory* (Kohonen, 1989a), where he suggested the existence of a sense of *locality* or *context*. Despite being a recurrently neglected factor, this notion of *background* in which the perceptions take place is a key concept behind the selectivity and high capacity achieved by the human memory.

Kohonen also sums up the major human associative memory operational features in a simple fashion:

1. Information search is performed on the basis of some measure of *similarity* relating to the input pattern.

2. Memory recalls and stores representations as *structured sequences*.

3. Information is recollected according to *dynamic processes*.

A well-informed reader with a background on computer science might have already noticed a few key differences when comparing the human memory to a standard computer one. In fact, a conventional John von Neumann serial computer is based on very fast processing elements — much faster than the neurons found in the brain. However, as powerful as modern computers may be, they still do not rival many of the capabilities of the human brain.

This divergence has led to the creation of *biologically-inspired* computational models (Rojas, 1996) less than a decade after the inception of the Turing machine and Alonzo Church's lambda calculus (Rosser, 1982). More precisely, in 1943, Warren McCulloch and Walter Pitts proposed a very simple yet ingenious model (McCulloch and Pitts, 1943) which is regarded as the first successful artificial neural network architecture. They argued that simple *two-state* artificial neurons sufficed for representing the neurophysiological findings which

had recently been discovered. Acting as *binary threshold logic gates*, the elementary units which compose McCulloch-Pitts networks were proven capable of synthesising and computing the value of any Boolean function.

The original model was quickly superseded by more powerful ones, especially since the complexity of a McCulloch-Pitts network grew very rapidly, in spite of the simplicity of the underlying units. The resulting networks were also not very relevant from a biological viewpoint — the computing units were too similar to conventional logic gates, and the lack of free parameters made it difficult to change the behaviour without *manually* rearranging the whole network design. In common neurocomputing jargon, these networks were uncapable of *learning*.

Wiener, one of the pioneers of the then called cybernetics field, took an especially interesting conclusion while analyzing the McCulloch-Pitts model. He studied pulse coding and arrived at the conclusion that binary signals were an *efficient coding* for representing information transmission on the nervous system (Wiener, 1948). This result is of particular interest to our work, as the artificial neural network architecture we shall use is also based on a binary representation.

Of the following models, which appeared within the newly born research field, the most notable is perhaps Rosenblatt's perceptron (Rosenblatt, 1958), which gathered a lot of attention and controversy greatly thanks to the famous pessimistic analysis of Minsky and Papert (Minsky and Papert, 1969). The original perceptron, being a linear classifier, could only recognize a restricted class of patterns. Although Minsky and Papert correctly stated this on their book, citing the Boolean XOR as an example of a non-computable function, they also speculated that the statement would hold true for all kinds of perceptrons. This was later proved wrong, as a perceptron with *multiple* layers and a *non-linear, differentiable* activation function could indeed be trained to approximate the values of any function.

Early researchers were primarily interested in developing a general means of function computation which retained the principal properties and structure of the brain. However, a new trend was set in 1961, when Karl Steinbuch invented the *Lernmatrix* (Steinbuch, 1961) — german for "learning matrix". The *Lernmatrix* is held as the first *artificial neural associative memory*. The term NAM was then coined to refer to a family of artificial neural networks which addressed the issue of establishing a mapping between *discrete* input and output spaces (Palm, 1980). In other words, NAMs are neural networks capable of implementing the functions of classification and memory.

Unlike their conventional computer memory counterparts which employ *location addressing*, associative memories support *content-addressability*. While they do not share internal similarities with traditional content-addressable mem-

ory hardware designs, they do share the notion of *address* and *data* possibly being one and the same thing. Kohonen provides an elegant explanation to this fact stating that one cannot regard each neural cell as a specific storage location; in fact, it is *the set of responses* which must be memorized as a whole, making the information *spatially distributed* across the whole network.

Although Steinbuch's original description of the *Lernmatrix* model was fairly incomplete, especially from a theoretical viewpoint, the model itself was a breakthrough. Its non-formal approach, however, has led many researchers in the literature to refer instead to an equivalent model devised by noteworthy british neurocomputing researcher David Willshaw. Sharing fundamentally the same working principles, Willshaw's *Non-holographic Memory* (Willshaw et al., 1969) was released together with a clever optical physics analogy which set the basis for the first formal studies of associative memory models. Throughout this work we will however refer to this model as the *Lernmatrix* or as Steinbuch-type memory, giving credit to its pioneering value. Despite its original simplicity it is still valid today and still presents both interesting features and research challenges as we will discuss on the following section.

2.2 An associative memory theory

Before proceeding any further into more detailed descriptions of NAM models, we shall first introduce an associative memory theory. We will do so after Günther Palm, a leading authoritative reference on associative memories, which provided a concise and rigorous mathematical and biological approach to the capabilities and properties of neural associative memories. We will now briefly introduce his notation (Palm et al., 1997) and then follow it along our comparative study of the different NAM models.

We start by providing a concise, even if somewhat simplified, theoretical definition of a neural network. According to Palm et al. (1997) an artificial neural network realizes a mapping F between an input space X and an output space Y, being F approximative in some sense and specifiable through a learning process from a finite set of examples. We call *retrieval* or *performance* to the execution of the mapping F and *training* or *learning* to its establishment. We can also identify common applications of F attending to the properties of X and Y:

1. Continuous X and Y — function approximation or interpolation
2. Continuous X and finite Y — classification and recognition
3. Discrete X and Y — classification and memory

As we will see, associative memory models (particularly the *Lernmatrix* one) generally fall into the third category, where a mapping $x \mapsto y$ is established. If $x \neq y$ then F realizes a *hetero-association* memory task and x is called the *key* or *address* string and y the *content* string. The special case of content-addressability is achieved when $x = y$. We then say F performs an *auto-association* memory task, which is obviously only useful if the approximative property of F is guaranteed. This way $\tilde{x} \mapsto x$ for all $\tilde{x}$ that are close to x in accordance to a defined metric in X.

2.2.1 Input and output patterns

Without losing generality we shall focus on the case where the strings x and y are binary patterns. As it has been stated before, it has been shown in the literature that the binary representation is an efficient one (Wiener, 1948), and it is the one chosen by most NAM models. We define and henceforth treat a binary pattern as a binary vector: an n-vector x where all x_i components contain either one of two possible elements. Different models will use different values for these elements; they may be thought of in generic terms as "B" and "W" (for black and white pixels in respect to image processing tradition). In practice they will often assume the values "0" and "1" or "-1" and "1", but they could assume any two literals a and b; a and b are not restricted to numbers. For the sake of simplicity we will choose the more familiar hardware-related "0" and "1" notation whenever possible. A pattern x can be classified (Palm and Sommer, 1996) as:

- *Distributed* if it contains more than one "0" component and also more than one "1" component.
- *Singular* if it has only one "1" element out of $m - 1$ "0" elements, being m the dimension of the vector.
- *Sparse* if the ratio p between the number of "0" and "1" elements satisfies $p << 1/2$. Authors may also use the infinite model where $m \to \infty$, in which case a *sparse* pattern obeys $\lim_{m\to\infty} p = 0$. A finite model especially useful for our work defines a pattern as sparse where the number of "1" components is of logarithmic order $O(\log m)$ with the dimension of the vector. Even other notations may be preferred; for instance, S. Amari, which has studied sparse patterns in detail (Amari, 1989), employs a special sparseness exponent e defined by $a_n = n^{-e}$ with $0 < e < 1$, where a_n is, like p, the probability that a component x_i of x is a "1" component.
- *Non-sparse* if the fraction p is away from zero, or, in the infinite model,

$\lim_{m\to\infty} p = K$ for some constant K. For unbiased patterns, we would have $K = 1/2$ (Nadal and Toulouse, 1990).

Since we are working on n-dimensional spaces of binary vectors, we can use the natural distance relation between two vectors x and y as the error detection metric in order to provide fault-tolerant addressing. This relationship can be defined using the *Hamming distance* (Hamming, 1950, Kohonen, 1989b), a classic information theory similarity measure. Let $h(x, \hat{x})$ be the number of different bits between the two patterns x and $\hat{x}$; notice how one can use h to retrieve a whole set of similar patterns using the relation $h(x, \hat{x}) < h(\hat{x}, x^k)$ for all $x^k \neq x$, with x and x^k belonging to the set of address patterns.

Palm and Sommer (1996) suggest assigning the real numbers "a" and "1" with $a \in [-1, 0]$ to the two possible signal representations since not all models use the typical $a = 0$ value; some of them — in particular those inspired on Ising spin physics (Little, 1974), as we will further see — use $a = -1$.

2.2.2 Synaptic weight matrix

Most neural network models accomplish learning through the modification of special free parameters contained in every neuron called *weights*. These weights correspond to the strength of synaptic connections on the biological model. Considering a network of n neurons, with m weights within each processing element, we can then introduce the concept of a global network weight matrix (Hecht-Nielsen, 1989a):

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix} \tag{2.1}$$

where each column represents the *weight vector* of each processing element from 1 to n.

2.2.3 Associative learning

Let S be the set of learning patterns presented to a neural associative memory, where x denotes the address vector, y the content vector and M the number of patterns to be stored:

$$S = \{(x^\mu, y^\mu) \mid \mu = 1, \ldots, M\}. \tag{2.2}$$

When the artificial neurons are fed with the learning signals represented by each pair (x^μ, y^μ) synaptic changes may occur, which will result in the calculation of new synaptic weights. We denote this change behaviour through a *local learning rule*, which can be simply represented by a four-dimension vector — remember we are dealing solely with binary patterns and as such the number of possible different pre- and post-synaptic combinations is $2 \times 2 = 4$. We will therefore use a vector in the form of $R = (r_{aa}, r_{a1}, r_{1a}, r_{11})$ where r_{xy} is the synaptic weight update expression and x is the pre-synaptic signal and y is the post-synaptic one. This way, we are able to specify through a compact form the neural learning behaviour for every possible combination of input and output values.

The learning process activity of a NAM can then be summed by a synaptic weight change equation in the form of

$$W = (w_{ij}) = \Big(\sum_{\mu=1}^{M} R(x_i^\mu, y_j^\mu) \Big). \tag{2.3}$$

Many different learning rules may be employed as we will see, but an important basic distinction which can be immediately done is between *additive* (also known as *incremental*) and *binary* learning rules. Equation 2.3, for instance, describes an incremental storage procedure, where the resulting weight matrix W is the sum of successive learning rule applications. However, NAMs may also just store a reduced binary version of W, obtained via a highly non-linear operation called *clipping* to W:

$$\bar{w}_{ij} = \text{sgn}(w_{ij}) \tag{2.4}$$

where sgn is the *sign function*, which can be defined as

$$\text{sgn}(x) = \begin{cases} -1 \text{ if } x < 0, \\ 0 \text{ if } x = 0, \\ 1 \text{ if } x > 0. \end{cases} \tag{2.5}$$

2.2.4 Associative retrieval

When we apply an input pattern $\tilde{x}$ to the network input, each neuron of the NAM will perform a weighted sum of the input values, calculating the *dendritic potential* s_j:

$$s_j = \sum_i w_{ij} \tilde{x}_i \ \forall j. \tag{2.6}$$

Using the dendritic potentials of every neuron the network is able to compute the output vector y, through the *neural update equation*

$$y_j = f(s_j - \theta_j) \ \forall j, \tag{2.7}$$

where we call f the *transfer function* and θ_j the *threshold value* of neuron j. Threshold setting strategies may vary; values may be adjusted globally becoming $\theta_j = \Theta$ or they can be individually stored on each neuron (Graham and Willshaw, 1995). Binary output patterns are obtained by choosing an appropriate transfer function. The *Heaviside* two-valued step-function $H(x)$ is the classic choice, carefully modified to return the real constant a instead of 0:

$$f(x) = \begin{cases} 1 \text{ if } x \geq 0, \\ a \text{ otherwise.} \end{cases} \tag{2.8}$$

The retrieval algorithm may run in parallel independently for each neuron if the NAM is a feedforward neural network; in this case we call the process *one-step retrieval*. As we will see some NAM models use instead feedback networks which require an *iterative retrieval* procedure, where the algorithm loops until the system reaches a stable state.

2.3 Steinbuch-type memories

Several mathematical descriptions and formal analyses of the original *Lernmatrix* or simply *Steinbuch* model have arisen in the literature; it has been subject of extensive study from not only but most relevantly Palm (1980), Willshaw et al. (1969), Amari (1989), Nadal and Toulouse (1990), Willshaw and Dayan (1990), Buckingham and Willshaw (1992). Even as of today researchers are still motivated by formal and performance aspects of the model — for instance, Sanchez-Garfias et al. (2005), Cruz et al. (2005) have successfully applied a novel theoretical approach to Steinbuch-type memories in 2005.

Simply put, Steinbuch memories can be seen as a cluster of units modelled in order to resemble some features of a biological neuron (see Fig. 2.1). Mathematically, they are more precisely described as single-layer feedforward (i.e., no loops in the network connections) neural networks consisting of *threshold logic* neurons. The typical representation of such a network is a matrix as shown in Fig. 2.2 where rows correspond to biological axons, columns to dendrites and weight values to modifiable synaptic strengths.

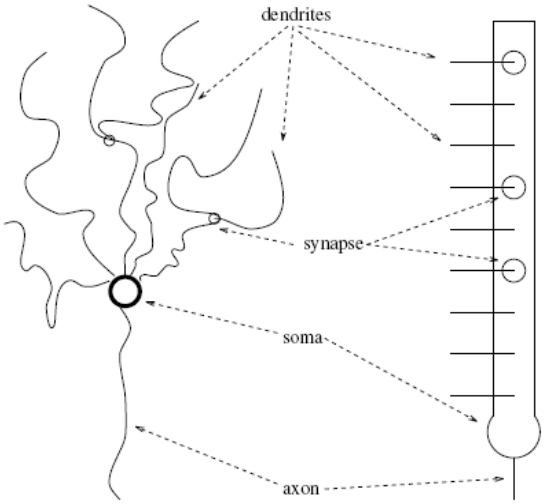


Figure 2.1: A unit as an abstract model of a real biological neuron (Hertz et al., 1991).

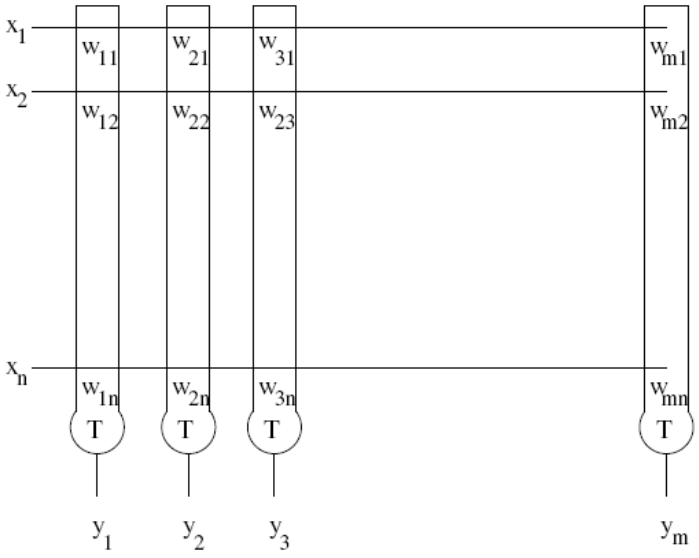


Figure 2.2: The associative memory as a cluster of units.

2.3.1 Hebbian learning

Psychologist Donald Hebb conjectured a plausible explanation for the biological learning processes taking place within brain cells. According to his theory, whenever neural synaptic activity occurs through a certain incoming signal, changes in the synaptic connections take place so that the neuron will respond with *greater efficacy* to that specific axonal input signal in the future (Hebb, 1949). This postulate falls on the category of *coincidence learning* — completely local rules based on connection strength changes in response to events that happen *simultaneously*.

Hebb's statement, later confirmed at least to some extent by neurobiologists, has been so influential in the neurocomputing field it became known as *Hebbian learning* at some point, honouring its original creator. Considering the input vectors x and y, one can translate it into the mathematical expression

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + x_i y_j. \tag{2.9}$$

Using the formal R-vector notation we presented in the last section, notice (2.9) can simply be rewritten in a compact form as

$$H := (0, 0, 0, 1) \text{ with } a = 0 \tag{2.10}$$

because it corresponds to a synaptic increase only when both pre- and post-synaptic activity is "1".

Like many other neural network models, the *Lernmatrix* uses Hebbian learning, even if a modified version of the general rule presented in (2.9). Weight values are updated during the training process according to a special binary *clipped* version of *Hebbian learning* (Palm, 1982):

$$w_{ij}^{\text{new}} = \begin{cases} 1 \text{ if } w_{ij}^{\text{old}} = 1, \\ 1 \text{ if } x_i y_j = 1, \\ 0 \text{ otherwise.} \end{cases} \tag{2.11}$$

Notice how applying the clipping operation over binary data is equivalent to performing a special Boolean sum (in fact, the Boolean OR). A Boolean rewrite of (2.11) such as

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} \vee (x_i \wedge y_j) \tag{2.12}$$

could be efficiently implemented in hardware in a straightforward fashion.

2.3.2 Anti-hebbian rule

Making the NAM forget a vector association (i.e., undoing a learning step) may not be trivial if it uses the *clipped* version of Hebbian learning. Therefore implementations which ought to support such a feature should use the classic *unclipped* law (see equation 2.9), where the memory weights store a frequency correlation value and not only a binary state. This way one can use the following anti-hebbian rule to forget an association pair (x, y):

$$w_{ij}^{\text{new}} = \begin{cases} w_{ij}^{\text{old}} - x_i y_j \text{ if } w_{ij}^{\text{old}} > 0, \\ w_{ij}^{\text{old}} \text{ if } w_{ij}^{\text{old}} = 0. \end{cases} \tag{2.13}$$

2.3.3 Thresholded retrieval

When an input (or "question") vector $\tilde{x}$ is presented to the network the most similar learned content vector y^μ is determined using the retrieval rule

$$\hat{y}_j = \begin{cases} 1 \text{ if } \sum_{i=1}^{m} \delta(w_{ij} x_i) \geq \Theta, \\ 0 \text{ otherwise.} \end{cases} \tag{2.14}$$

Note that this rule does not require the frequency correlations calculated by the incremental storage procedure; if such learning process is being used one should define δ as the *sign function* with a domain restriction to $\mathbf{R}_0^+$ (as there can be no negative weights on W):

$$\delta(x) = \begin{cases} 1 \text{ if } x > 0, \\ 0 \text{ if } x = 0. \end{cases} \tag{2.15}$$

If W is calculated according to a clipped learning rule one can effectively discard the δ function or simply define it as the identity function $\delta(x) = x$.

Note that a well-chosen Θ is a crucial for high-quality retrieval; too low values will result in *add-errors* and too high values will result in *miss-errors*. We define an add-error as an additional "1"-component in the output vector $\hat{y}$ which was not present in the originally learnt y^μ. Similarly, a miss-error occurs when there is a missing "1"-component in the output vector y.

We use the ℓ_0 zero norm of a pattern x to denote its "activity level" $|x|_0$, i.e., the number of active elements. The classic threshold setting regime due to Willshaw et al. (1969), where $\Theta = |\tilde{x}|_0 = \sum_i \tilde{x}_i$, is an optimal solution when the input cue $\tilde{x}$ is possibly incomplete but not noisy. For the general case where $\tilde{x}$ may contain additional or mispositioned "1"-components, the threshold must be chosen using an approximation strategy with the goal value set at $\Theta \approx \sum_i \tilde{x} x^\mu$,

which may or may not be easy to accomplish as it is not a simple function of $\tilde{x}$ (Graham and Willshaw, 1995).

Several variations on the original retrieval process have been proposed (see Sommer and Palm (1999) for a careful analysis) yielding greater error tolerance at the expense of additional computational costs. We will not consider them in this work, as we aim for the lowest possible retrieval effort.

A possible solution to the Willshaw threshold problem would be to use the so called "soft" or *winners-take-all* (WTA) threshold strategy where the neurons with the highest dendritic potential are selected

$$\Theta := \max \sum_{i=1}^{m} \delta(w_{ij} x_i). \tag{2.16}$$

This way, the retrieval process will only fail to return an answer if all of the components $\tilde{x}_i$ of the question vector $\tilde{x}$ are not correlated with any $x_i^\mu \in x^\mu$, for all $x^\mu \in S$.

If S is comprised of patterns with a fixed activity k, a simple modification of the WTA threshold strategy has been shown to deliver better results. This *L-max* strategy prescribes that instead of selecting the neurons corresponding to the single highest (winner) dendritic potential, the L maximum values are chosen until exactly k neurons are firing. It has been recently subject to experimentation (Hobson and Austin, 2009), leading to interesting results.

2.3.4 Storage capacity

The results on how much information can be stored on a *Lernmatrix* NAM are held as one of the major conclusions of Palm's essay on associative memories. Using elementary probability and information theory, Palm not only calculated the asymptotic bounds of the information capacity of Steinbuch-type memories, but also showed the importance of vector *sparseness* (recall section 2.2.1 for a definition).

Basically, Palm arrived at the conclusion that the *Lernmatrix* could store much more information while maintaining *high-fidelity* (i.e., error-free) results if the vectors $x^\mu, y^\mu \in S$ were *sparse*. This is one of the principal distinguishing properties of the *Lernmatrix* when compared to other NAM models. It is discouraging in the sense of being a restriction; but it can also be extremely positive because if one can employ a *sparse coding scheme* that maps densely-represented information into a set of sparse codes, then the model will provide unrivalled storage capacity.

Jagota et al. (1998) have questioned if the use of sparse codes is a real necessity and pushed the theoretical information capacity limit to an even higher

bound; however, no experimental evidence has been provided, and it seems that using currently discovered learning rules, Palm's results are still the best reachable limit. We will now provide the mathematical relations and briefly cover how they were found; for a full proof see the original article (Palm, 1980) or Hecht-Nielsen's explanation (Hecht-Nielsen, 1989b). For simplicity's sake and without loss of generality we will deal with square memories where $n = m$.

Using Shannon's information measure I (Shannon, 1948), Palm derived an equation where the ratio of *realized* information storage capacity to *available* information storage capacity $C := I/n^2$ is expressed in terms of the activity level of the stored patterns, provided that this level is fixed $k := |x^\mu|_0 = |y^\mu|_0 \ \forall x^\mu, y^\mu$. Willshaw et al. (1969) and Palm (1980) defined an asymptotic upper bound $C = \log 2$ bits/synapse for Steinbuch memories, achievable only when $n \to \infty$ and the stored patterns are sparse and uncorrelated.

With the aid of a numerical computer algorithm, an approximate optimum value of k can be calculated, rendering the expression

$$k_{\text{opt}} = \log_2(\frac{n}{4}) \tag{2.17}$$

from which Palm also derived an approximate equation for the maximum number of vector pairs M which can be stored:

$$M_{\max} = (\log 2)(\frac{n^2}{k^2}). \tag{2.18}$$

Notice how M is much greater than the number of procesing units n, if we use the optimum sparse value k_{opt}. In this case, as we will see while covering other NAM models, Steinbuch-type memories offer a very generous storage capacity relation.

2.3.5 Efficient implementation

The *Lernmatrix* is particularly well apt for massively parallel hardware implementations such as the Parallel Associative Network system design and the respective specialized BACCHUS processor as described in the literature (Palm and Palm, 1991, Strey, 1993). These specific hardware solutions are obviously interesting thanks to the inexistence of feedback connections and to the *locality* property of the Hebbian learning rule, i.e., each processing element may compute its output independently. It is shown that an efficient parallel implementation is able to realize the retrieval process in $O(1)$ constant time, granted that both address and content patterns are sparse. Young et al. (1999) have also provided a fast hardware architecture called PRESENCE, up to 50-fold faster than its software counterpart, and successfully applied it to pattern clas-

sification (Hodge et al., 2004). Wichert (2006) successfully applied a Hebbian learning *Lernmatrix* on a production, real-time diagnostics embedded system.

However, not much has yet been studied when it comes to implementing associative memories on conventional serial computers. Note that an immediate naïve solution would require $m \times n$ sequential steps to compute an answer, where m is the dimension of the pattern vector and n is the number of processing units.

As shown by Bentz et al. (1989) an immediate optimization arises if one takes into consideration the pattern *sparseness* factor. Due to the fact that on a sparse pattern we have $k << m$, where k is the number of "1" components on a vector, a special representation where only the "1"-components are stored instead of the whole vector could be used to improve space and time complexity. The so called *pointer representation* $\dot{x}$ of a given sparse pattern x would then be a k-dimensional vector where its components $\dot{x}_1, \dot{x}_2, ..., \dot{x}_k$ would contain coordinates c meeting the condition $\{x_c = 1 : x_c \in x\}$. We will see later in greater detail on chapter 3 the usefulness of this representation format from the key performance perspectives of minimizing storage space and retrieval time.

2.4 Other NAM models

Alongside and after the inception and the study of the *Lernmatrix* memory model other researchers pursued similar goals, albeit in different tracks. As we will see, these parallel (yet subsequent) efforts resulted in models which resemble in many ways Steinbuch's original associative memory, even if employing different physical analogies or mathematical tools to derive the final results. We will now present the most relevant competing neural network architectures in a comparative fashion so that a conclusion can be drawn on why we have chosen the *Lernmatrix* as the basis of our research.

2.4.1 The Linear Associator

Perhaps the simplest of all NAM models was simultaneously invented by independent researchers (see Anderson, 1968, Kohonen, 1972, Anderson, 1972, Nakano, 1972) towards the end of the 1960s and the beginning of the 1970s. Heavily inspired on Rosenblatt's Perceptron, this single-layer feedforward neural network architecture is commonly referred to as the *linear associator* (although other names such as *correlation matrix memory* may as well be found in the literature).

The linear associator is based on elementary linear algebra operations. When presented with an input vector x, the output vector y is calculated through

matrix multiplication:

$$y = Wx \tag{2.19}$$

where W is the $m \times n$ *weight matrix* with each row i being the weight vector of the i^{th} processing element.

Note that unlike the *Lernmatrix*, the input and output spaces X, Y of the linear associator are not limited to binary vectors of the form $x \in \{0,1\}^m$ or $y \in \{0,1\}^n$. In the latter, x and y belong to $\mathbf{R}^m$ and $\mathbf{R}^n$ and may thus contain real values. X, Y may even be composed of complex vectors given that a slightly modified model is used (Nemoto and Kubono, 1996).

The process of *learning* is then that of adjusting the weights so that when presented with a question vector $x+\epsilon$, the network will return the closest stored $y+\delta$, according to the pairs of previously learnt vectors $(x, y) \in S$ (cf. equation (2.2)). Due to the linear relation between the errors ϵ and δ, the linear associator is said to be *interpolative*.

Just like the *Lernmatrix*, the original linear associator used Hebbian learning to perform weight updates. For each pair of vectors (x, y) to be associatively stored, one ought to update the weight matrix W using the rule

$$W^{\text{new}} = W^{\text{old}} + yx^{\text{T}}. \tag{2.20}$$

As simple as it may be, this NAM model is however of limited interest, especially in its original form. Because of the simple linear transformation involved in the retrieval process combined with the Hebbian learning rule, the linear associator can only effectively recognize vector pairs (x, y) without introducing a significant error factor if all stored input vectors x are *orthonormal* (for a proof see the work of Hecht-Nielsen (1989a)). Furthermore, the capacity of a linear associator network is also much lower than that of a sparse *Lernmatrix*: it follows the former proof that $M_{\max} = n$.

Much research has been carried out in order to improve the linear associator. For instance, error reduction may be attained using special pattern encoding techniques (Leung et al., 1998). It has also been shown that the network may provide perfect recall for vectors that are not *orthonormal* but only linearly separable (Murakami and Aibara, 1987) if instead of employing Hebbian learning one uses the weight update equation

$$W = YX^{+} \tag{2.21}$$

which is called the *pseudoinverse formula*, where X and Y are matrices containing all previously learnt x and y vectors and the $+$ operator is the Moore-Penrose generalized matrix inverse. However, the limited capacity $M_{\max} = n$

upper bound is still not lifted. A possible solution is to group together K linear associators in parallel and distribute input vectors across the pool. This will increase the capacity and allow for $M_{\max} = Kn$ vector pairs to be stored, but at the expense of extra computational steps and possibly introducing errors in the output (Haque and Cheung, 1994).

Even if the model has been successfully used for 2-D object recognition in images (Wechsler and Zimmerman, 1988), research results have not been particularly impressive because of the low capacity bound and the linear separability restriction.

2.4.2 The Hopfield Net

Physicist John Hopfield was one of the key figures who led the resurge of interest in the neurocomputing field which occurred during the 1980s. Following the works of Amari (1972), Grossberg (1972), Little (1974), who had already hinted the idea of using energy functions to describe and analyze the *dynamical behaviour* of neural networks, Hopfield (1982) established a physical analogy between ensembles of neurons and Ising spin systems and invented what would become known as the Hopfield net.

This *single-layer* network is of the *recurrent* type, allowing *feedback connections* between its n processing elements. Input and output (which may only assume binary or bipolar values) operations are performed through the same connectors and as such the network is *strictly* autoassociative. Due to the existance of feedback connections, the neurons of Hopfield nets perform local random asynchronous state changes at a given average update rate which is the same for all neurons. Notice how this behaviour is unlike the *Lernmatrix* or the linear associator which imply a synchronous one-step learning and retrieval process.

When the network is presented with a new input vector x, each neuron will perform a series of changes to its state x_i described through the equation:

$$x_i^{\text{new}} = \begin{cases} 1 \text{ if } \sum_{j=1}^{n} w_{ij} x_j^{\text{old}} > \theta_i, \\ -1 \text{ if } \sum_{j=1}^{n} w_{ij} x_j^{\text{old}} < \theta_i, \\ x_i^{\text{old}} \text{ otherwise.} \end{cases} \tag{2.22}$$

Note that this equation refers to the *discrete* variant of Hopfield nets, which is the most relevant one concerning associative memories. A *continuous* version which uses a bounded, continuous output function (generally *sigmoid* or the *hyperbolic tangent*) was also proposed by Hopfield (1984). We will not discuss this variation, which has similar behaviour and properties, but slightly more

complex mathematics and no advantages as a NAM. The equations in this section will thus refer to discrete Hopfield networks.

One may still apply the W weight matrix notation to combine the internal states of all neurons. For Hopfield nets we define W as a square $n \times n$ real-valued matrix where

$$w_{ij} = w_{ji} \text{ for all } i, j \in \{1, 2, ..., n\}, \tag{2.23}$$

and

$$w_{ii} = 0 \text{ for all } i \in \{1, 2, ..., n\}. \tag{2.24}$$

Granted that these restrictions (matrix symmetry and 0 diagonal) are verified, the network is guaranteed to reach a stable state. Notice how equations (2.23) and (2.24) in practice mean that each neuron must be connected to every other one, except for itself (albeit with varying weights). This fact may be taken from the study of a *Lyapunov function* E, which expresses the dynamical properties of a Hopfield net:

$$E = -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + 2\sum_{i=1}^{n}\theta_i x_i. \tag{2.25}$$

Analysis of this lower bounded, monotonically decreasing *energy function* guarantees the network will stabilize into one of its *local minima* whenever inputs x_i change (for interested readers, a proof and further discussion may be found on J. Hopfield's original article (Hopfield, 1982) and on the mathematical analysis by Amit et al. (1985a)). Notice that one should not assume that the network will always stabilize on a *global minimum*; in fact, state changes will often lead to a *local optimum trap* (Kennedy and Chua, 1988). Several researchers have attempted to overcome this limitation but still no perfect solution is known (Wen et al., 2009). As an example, see the work of Chen (2002).

Hopfield nets may be used as neural autoassociative memories and the weight matrix W may be updated using associative learning. The idea is that the weights are changed in a way that the network *stabilizes* into the desired binary pattern when presented with an incomplete or distorted version of it. This may be accomplished through Hebbian learning, where W is updated according to the *agreement rule*, defined using Palm's rule notation (recall from section 2.2) as

$$A := (1, -1, -1, 1) \tag{2.26}$$

where the null state a of the binary vectors $x_1, x_2, ..., x_n$ to be learnt is set to -1 instead of the typical $a = 0$ definition. Using rule A weights will *increase* if pre- and post-synaptic states agree, and *decrease* otherwise. One should note that this is not the only available learning procedure; other methods have

further been discovered and can be found in the literature. For instance, Liou and Yuan (1999) have provided an algorithm to achieve *greater error tolerance* while performing distorted pattern retrieval.

Amit et al. (1985b, 1987) provided seminal contributions to the study of Hopfield nets as associative memories, especially on the topic of their capacity. They have shown that the memory becomes full very quickly, introducing *spurious states* and behaving unexpectedly after the limit $M_{\max} = 0.14n$. This value was achieved considering the *imperfect* retrieval of *uncorrelated* patterns, which is a severely constraining factor for realistic data, as well as unnatural in a biological sense.

Several other authors have also performed further investigation on the theoretical capacity limits of the network (McEliece et al., 1987) and tried to extend it (Bruce et al., 1986); on the other hand, François (1996) has conducted rigorous mathematical proofs backing Amit's original results, resorting to classical statistical mechanics analysis. These authors agree that the maximum theoretical limit for perfect retrieval on a Hopfield network is

$$M_{\max} = \frac{n}{2 \log n}. \tag{2.27}$$

In 1998, Löwe (1998) finally carried on a full mathematical analysis on the behaviour of Hopfield nets for the case of correlated data, proving results near equation (2.27) could also be reached if the correlation was generated by a homogeneous Markov chain.

Of all these studies, perhaps the most relevant conclusion in the light of our work was taken by Gardner (1988), who was able to theoretically relate the capacity limit with the *sparseness factor* of the input patterns. Precisely alike Steinbuch-type memories, Gardner has shown that if pattern activity is extremely low (recall our definition of sparseness in section 2.2.1), then the capacity limit may be pushed well over the former barrier, leading it to the increase factor of $-(p \log p)^{-1}$ (recall p as the ratio between active and total number of elements). Using our notation of $M_{\max}$ as a function of the number of "1" components k of the stored vectors and their length n will render the formula

$$M_{\max} = -\frac{n^2}{k \log(k/n)}. \tag{2.28}$$

Buhmann et al. (1989) later experimentally showed these analytical results were right and suggested an extremely sparse optimum ratio p of the order of 10^{-7}.

Gardner also achieved important results when comparing the Hopfield net with the classic perceptron model. Instead of using Hebbian learning, Gardner experimented with the early perceptron learning rule instead, and was able to

push $M_{\max}$ to $2n$ for uncorrelated random data, and to an even higher figure for a correlated set of patterns. This study was particularly relevant because it showed the storage process in Hopfield nets may be enhanced through the use of alternative learning rules, unlike *Lernmatrix* memories, which present *optimal* or *near-optimal* storage capacity using Hebbian learning as shown by Palm (1980). The asymptotic limits of this perceptron-learning Hopfield, sometimes referred to as the *generalized Hopfield net*, were studied in detail by Ma (1999), formally backing Gardner's original results.

Hopfield nets, originally envisioned as a means of implementing associative neural memories, have been extensively used as a tool for mathematical programming problem solving as well. Hopfield himself first suggested their applicability to the optimization problem domain in his highly cited work with D. W. Tank (Hopfield and Tank, 1985), and much work was carried in that direction ever since. A thorough and up-to-date invited review has been written by Wen et al. (2009) summarizing the field work while highlighting still open, interesting problems.

Several variations on the original Hopfield net (commonly called *Hopfield-type* networks) may be found on the literature, such as the Bidirectional Associative Memory (which we will cover on the next section), the Inverted Neural Network (Shouval et al., 1991) or the Hopfield-clique network (Jagota, 1994). The INN is a modified version for easy optical hardware implementation, while the latter, named and modelled after the Max-Clique problem, is especially suitable for graph-based problems thanks to its graph-theoretic characterization. Incidentally, the authors of both networks have joined efforts and wrote a deeper analysis and comparison of the two architectures (Grossman and Jagota, 1993).

Some of the existing variations are inspired on Gardner's prior experimentations with different learning rules and network structures. A discussion of the performance of these variants can be found on Davey and Adams (2004). Promising results were published in 2009 combining perceptron-learning Hopfield nets with *asymmetric* connections (i.e., disrespecting equation (2.23)) and a genetic algorithm to investigate the most efficient weight matrices (Adams et al., 2009).

As interesting as Hopfield's model may be, thanks to its easiness of use in the sense that it allows storing non-sparse information, it is after all less adequate than Steinbuch-type memories to pursuit our goal due to its strictly autoassociative behaviour, worse achievable information capacity and increased underlying model complexity. Research on Hopfield nets has been decreasing also due to their minimal biological value. Dense synaptic connectivity and bipolar weights are quite hardly justifiable and unlikely to happen on real biological systems

(Palm and Sommer, 1992).

2.4.3 The Bidirectional Associative Memory

In 1988, Bart Kosko introduced the *Bidirectional Associative Memory*, an extension to the original Hopfield model which removed the strictly autoassociative behaviour constraint and allowed for heteroassociations to be performed (Kosko, 1988). Just like classic Hopfield nets, the BAM network architecture is also made of units with symmetric feedback weighted connections. However, BAM nets have two *layers* or *slabs* of interconnected neurons (recall Hopfield nets were *single-layer*).

Again, in the same fashion of Hopfield nets, BAM nets use *binary* or *bipolar* units on the *discrete* variant, or real-valued units on the *continuous* variant. We will once again discuss and present the equations for the discrete version, inviting the avid reader to consult the proofs and the equations for the continous version on Kosko's original paper (Kosko, 1988).

Let us denote the *input-output* signals of the first set as $x_1, x_2, ..., x_n$ and the signals of the second one as $y_1, y_2, ..., y_n$. We will follow the same notation we used when describing previous network models. Hence, we have the similar (but now for both layers of units) state update equations, presented in the bipolar form:

$$x_i^{\text{new}} = \begin{cases} 1 \text{ if } \sum_{j=1}^{m} w_{ij} y_j^{\text{old}} > \theta_i, \\ -1 \text{ if } \sum_{j=1}^{m} w_{ij} y_j^{\text{old}} < \theta_i, \\ x_i^{\text{old}} \text{ otherwise,} \end{cases} \tag{2.29}$$

and

$$y_j^{\text{new}} = \begin{cases} 1 \text{ if } \sum_{j=1}^{m} w_{ij} x_i^{\text{old}} > \theta_j, \\ -1 \text{ if } \sum_{j=1}^{m} w_{ij} x_i^{\text{old}} < \theta_j, \\ y_j^{\text{old}} \text{ otherwise.} \end{cases} \tag{2.30}$$

These equations use non-zero thresholds θ as proposed by Haines and Hecht-Nielsen (1988) to maximize storage capacity; technically, we are then dealing with a *non-homogeneous* BAM network, following the nomenclature suggested by the authors (the original *homogeneous* version of the former equations used a constant value $\Theta = 0$).

The stability of discrete BAM networks is ensured by the monotonically decreasing Lyapunov state energy function

$$E = -\sum_{i=1}^{n} \sum_{j=1}^{m} x_i w_{ij} y_j \tag{2.31}$$

provided by Kosko (1988).

The original BAM model employed Hebbian learning to perform the heteroassociations, where the correlations are stored in the weight matrix W using equation (2.20), just like the linear associator. However, as Tanaka et al. (2000) have shown, the BAM architecture coupled with Hebbian learning provides a very low capacity limit, set at $M_{\max} = 0.2n$ even if we allow for a small error ϵ (*imperfect* retrieval). To counter this fact, Haines and Hecht-Nielsen (1988) applied Palm's information theory approach to their *non-homogeneous* BAM architecture and reached the very interesting value of

$$M_{\max} \approx 0.68 \, \frac{n^2}{\log_2 n + 4} \tag{2.32}$$

when using the optimal sparseness factor $k = (\log_2 n) + 4$. Comparing equations (2.32) and (2.18) reveals that the capacity limit of a BAM network is even better than the one offered by a *Lernmatrix*-type memory ($O(n^2/\log n)$ vs. $O(n^2/\log^2 n)$). This result is in accordance with our research; delivering an effective sparse coding (even if for a restricted domain) will result in a dramatic increase of performance for BAM networks.

The extra layer of units adds interesting capabilities to the BAM model when opposed to single-layer recursive networks. It allows for bidirectional traversal of associated patterns by using either the weight matrix W or its transpose W^{T}, i.e., if a set of patterns (x^μ, y^μ) has been stored, one might perform a query using either $\tilde{x}$ or $\tilde{y}$ as the address pattern.

Variations of the BAM network may be found in the literature. For instance, Kosko (1987) pointed out that a BAM network is well suited for adaptive unsupervised learning if paired with an appropriate rule; Humpert (1990) developed a special BAM_q variant which allowed for the heteroassociation of q-tuples of patterns (a conventional BAM network would thus be called BAM_2 according to the author's notation). However, concerning network performance, the use of sparse codes remains unbeatable.

2.4.4 The Sparse Distributed Memory

In 1984, Pentti Kanerva published through his PhD. thesis a mathematical theory which tried to encompass the main properties of long-term human memory using *high-dimensional spaces* (Kanerva, 1984). The basic intuition behind his theory is that distances between concepts in the human mind correspond to distances between vectors on a high-dimensional space.

Kanerva's original work led to the inception of the influential Sparse Distributed Memory model (Kanerva, 1988), which featured a physical hardware

implementation comparable to a traditional random-access memory. Since his first papers did not present the SDM as a neural network architecture but as a statistical memory, we will instead use an alternative notation which became standard in later analyses. This way, the SDM can be easily related and compared to other NAM models (Kanerva, 1993). If the reader is interested on the original physical memory formulation, an easy introductory explanation may be found in Wasserman (1993).

The SDM may be seen as a two-layer feedforward artificial neural network, capable of performing both heteroassociative or autoassociative memory tasks. The innermost layer represents the *memory contents* and is simply a classic correlation matrix memory. Let C be its $U \times S$ unclipped ($\mathbf{Z}$-valued) synaptic weight matrix, where U is the dimension of the data vectors to be stored and S is a fixed special parameter of high dimension. Note that this layer's input signals $y_1, y_2, ..., y_S$ are connected to the outputs of the first layer, and that its output signals $z_1, z_2, ..., z_U$ are the output of the SDM. This weight matrix can be trained according to a desired learning prescription such as Hebbian learning (which was the rule originally employed by Kanerva).

The first layer, however, is where the key difference of the SDM model lies. It has a fixed (i.e., not trainable) set of synaptic weights we shall denote by a $S \times N$ weight matrix A, whose purpose is to perform a blow-up expansion on the input address vectors from their original N-dimensional space to an S-dimensional one, where $S >> N$. Ideally, this layer would generate a sparse, high-dimensional vector y, which would then activate a very small fraction of neurons on the next layer for each input vector x and thus generate an optimal correlation matrix. An optimally-defined A would solve the *sparse-coding problem* for a given problem domain and yield an efficient, scalable associative memory.

Using the bipolar form (where $a = -1$) for input and output vectors x and z, the retrieval process can then be expressed through two equations, where the first one calculates the outputs y_s for the neurons on the first layer

$$y_s = \text{sgn}(\sum_{n=1}^{N} A_{sn} x_n - (n - 2r)) \tag{2.33}$$

whose values are then fed onto the inputs of the second layer, finally rendering the output vector z calculated by second equation

$$z_u = f(\sum_{s=1}^{S} C_{su} y_n) \tag{2.34}$$

where f is the bipolar step function as defined in equation (2.8). Notice that

equation (2.33) performs an implicit thresholding with the term $n-2r$, where the parameter r is a previously defined Hamming distance radius of activation. In other words, all memory locations within a hypersphere of r bits are activated.

To perform a mathematical analysis of the model, Kanerva considered vectors which were uniformly sampled from their spaces. This way it would be trivial to find an adequate matrix A (also set according to a random uniform distribution) and the analysis would be eased while resorting to statistics. He derived an approximate capacity bound for perfect-retrieval of

$$M_{\max} \approx 0.1S. \tag{2.35}$$

This relation was later theoretically expanded by Chou (1989), which confirmed that under optimal conditions the capacity of the SDM model was asymptotically independent of the dimension of the input vectors, being only related to the size S of the internal high dimensional memory. However, as many researchers have pointed out, the problem is that when dealing with real-world data (which is not uniform), a badly chosen matrix A would result on an extremely degraded capacity. Ryan and Andreae (1995) have suggested the use of an enhanced iterative retrieval rule, which would improve $M_{\max}$ to $1.77S$, but still only under optimal conditions.

A possible workaround for this serious shortcoming has been provided by Keeler (1988). He has suggested that if the distribution of the training set is known in advance, one could sample the weights forming matrix A according to the same distribution and then achieve near optimal performance, provided that a correct radius r would be selected.

The SDM model also suffers from its rigidness; having to define the parameters r and S in advance requires knowledge over the training set. Hely et al. (1997) tried to address this issue by removing the fixed parameters and introducing a signal decay factor which is basically equivalent to a classic threshold factor, making the SDM even closer to a conventional associative memory.

Interestingly enough, studying the SDM model has once again emphasized the severeness of the sparse coding problem. Either through a matrix transformation or through more complex algorithms, it is clear that the performance of an associative memory is closely tied to whether a solution to it is found or not. Nevertheless, the SDM remains as an interesting alternative formulation of high biological interest, with plausible connections to many models of the cerebellum (Kanerva, 1993).

2.5 Sparse similarity-preserving codes

Inumerous neuroscience studies suggest that *sparse coding* of information may be an explanation to the *selective neuron firing* phenomena that occur in the mammalian visual cortex. Within the billions of biological processing elements that make up our brain, sensory stimuli only activate a very small fraction of interconnected neurons. (Olshausen and Field, 1996)

As we have shown through our former analyses of various associative memory models, the performance of these neural network architectures which try to mimic and implement certain features of the human brain is also highly related with information sparseness. In fact, as information is naturally dense, without successful sparse encoding prescriptions, their capacity will grow atmost linearly with the dimension of the input space, severely restricting their real-world applicability.

This so-called *sparse coding problem* has already been formally specified and addressed within particular domains by some researchers. Although we are especially interested in storing and performing fault tolerant retrieval of images, we will now describe a classic successful result for sparse representation of textual objects, which illustrates the concept of *feature extraction*.

A typical representation of a string of characters will employ an *any-out-of-N* encoding (Hecht-Nielsen, 1989c), where each ASCII character corresponds to a dense, compact binary code. Trivially, if we wanted to encode a single character, at the expense of additional space we could build a sparse *1-out-of-S* code, where $S >> N$, and each possible ASCII character would correspond to a single active bit. Considering the standard 128 possible different ASCII characters, a compact representation would only need $N = \log_2(128) = 7$ bits to encode it. The sparse *1-out-of-S* variant would need $S = 2^N = 128$ bits to encode the same information, and would thus correspond to an *exponential space blow-up*. Hence, for instance, in order to represent a 7-character word one would need 1024 bits of information. A similar approach for string encoding has been employed by the famous NETtalk neural network (Sejnowski and Rosenberg, 1987).

Building up on the previous example, how could one sparsely represent an entire word using a more efficient and robust code? Wickelgren's psychology results led to a possible solution in accordance to our mind's behaviour (Wickelgren, 1977). By mapping to each string position a context-sensitive triple (a "*Wickelphone*") containing the previous, current and next character, a word could be represented by a set of features. Considering a minimal set of 27 possible different characters (the entire alphabet plus a special delimiter), a word could be encoded by activating the positions corresponding to present features

on a 27^3-dimensional vector.

Similarity-preserving sparse coding prescriptions are especially useful for pattern recognition tasks. They perform transformations into high-dimensional spaces while preserving certain desired properties such as the ordering between elements or their relative distance. The *Wickelphone* mapping described earlier is one such example, as it is order-preserving. We define ϕ as order-preserving iff ϕ is a metric space mapping $\phi : (A, d_A) \longrightarrow (B, d_B)$ such that the following relation holds true:

$$\begin{aligned} & d_B(\phi(x), \phi(y)) \leq d_B(\phi(x), \phi(z)) \\ & \forall x, y, z \in A : d_A(x, y) \leq d_A(x, z). \end{aligned} \tag{2.36}$$

As in our case A, B are vector spaces of the form $\{0, 1\}^N$, metrics $d_A(x, y)$ and $d_B(x, y)$ are naturally the Hamming distance $h(x, y)$. We can then easily define a similarity measuring function s_A:

$$s_A(x, y) = dim(A) - h(x, y). \tag{2.37}$$

This function satisfies desirable properties which hold true $\forall a, b$:

$$\begin{aligned} s(a, b) \quad & \geq 0 \\ s(a, b) \quad & \leq s(a, a) \\ s(a, b) \quad & = s(a, a) \Rightarrow a = b \\ s(a, b) \quad & = s(b, a) \\ s(a, b) \quad & < \infty \end{aligned} \tag{2.38}$$

As Hecht-Nielsen points out, *1-out-of-N* codes are very inefficient information-wise, with their information content per word being fixed at the low limit $\log_2(N)$. In his 1987 article, he discusses the existance of *combinatorial hypercompression codes* (Hecht-Nielsen, 1987), which could be used to create a mapping between generic *any-out-of-N* codes and *K-out-of-N* ones, where exactly K components of an N-dimensional vector are active, with $K << N$ in order to provide an acceptable sparseness factor. Notice that these codes would contain a higher information capacity of $\log_2 C(N, K)$.

Hecht-Nielsen suggested the use of a K-nearest neighbour classifier to automatically determine and assign K-out-of-N codes. However, the question remains of how could such a classifier be trained in order to provide a successful mapping for a generic set of input codes.

Currently, several computer vision researchers are investigating how could the *sparse coding problem* be solved for the imaging domain. This solution

would be technically and biologically relevant, as it has been shown that sparse coding not only offers a good feature extraction representation, but also matches very well current theories on primary visual cortex V1 neurons (Olshausen and Field, 1997).

These studies often formulate the *sparse coding problem* generically as the task of finding a representation $Z \in \mathbf{R}^m$ for a given signal $Y \in R^n$ by linear combination of an overcomplete set of basis vectors, columns of matrix $B \in \mathbf{R}^{m \times n}$ with $m > n$ (Kavukcuoglu et al., 2008):

$$\min ||Z||_0 \text{ subject to } Y = BZ \tag{2.39}$$

which employs the zero norm ℓ^0 notation to represent the number of non-zero elements in Z.

Diverse alternative mathematical formulations have been proposed, since solving equation (2.39) means performing a prohibitively expensive combinatorial search. Best-in-class results have been achieved when (2.39) is relaxed into a ℓ^1-norm convex optimization problem:

$$\mathcal{L}(Y, Z; B) = \frac{1}{2} ||Y - BZ||_2^2 + \lambda ||Z||_1 \tag{2.40}$$

which has been shown to have an equivalent solution (Donoho and Elad, 2003). However, solving this unconstrained optimization problem is still a computationally heavy task, unsuitable for real-time systems or where large training sets are to be dealt with in practical time.

2.6 Final remarks on Steinbuch-type memories

Although sparse coding has been effectively classified as a (very) hard problem, especially when applied to sound, images, or other naturally dense data, current neuroscience trends point favourably towards the existence of such a biological mapping system to encode sensory information (Baddeley, 1996, Olshausen and Field, 1996). We think it is interesting and exciting how a simple computational model such as a Steinbuch associative memory can mimic the behaviour of some portions of the human brain and pose similar open challenges, establishing a link between biology and computer science.

Physiological measurements on the primary visual cortex which show that only a small part of the neuron population is simultaneously active at a given time, together with new findings on synaptic plasticity supporting Hebb's now sixty-year old learning theory (Dayan and Abbott, 2001) have pushed forward the interest on binary associative memories and sparse coding techniques.

While offering substantial advantages when compared to early competing models such as the original Hopfield net (Palm, 1991), on the end of the eighties and the beginning of the nineties these alternative models eventually evolved and integrated the concept of low activity levels into their specification to make them more realistic and to increase their information storage capacity bounds. Advanced features such as the greater noise tolerance of BAM's iterative retrieval can be attractive, but come with the price of higher computational complexity and reduced biological significance. So far, when it comes to implementing brain-inspired memories in neurocomputing, the Steinbuch model has been the architecture of election.

Chapter 3

An alternative structural representation

The misterious gap between the asymptotic storage capacity bound of $C = \log 2$ initially defined by Willshaw et al. (1969) and the theoretical maximum of $C = 1$ for binary associative memories, which has been considered unsolvable for decades, has recently been worked around. While the traditional measure of network capacity still remains valid and fixed at $\log 2$, Knoblauch et al. (2009) have argued that perhaps the ideal means of storing the synaptic connectivity information on a computer is not the binary weight matrix itself.

In fact, this simple act of questioning the basic premises of an underlying model can open interesting new possibilities. In this chapter we revisit the classical one-step retrieval prescription of Steinbuch associative memories, originally envisioned for dedicated parallel architectures where the cost of activating one, two or one thousand of neurons was merely a question of energy, but not time. This is of course not the case of sequential implementations and, at least according to current neuroscience theories, neither of mammalian brains where such metabolic costs appear to be unsustainable (Lennie, 2003).

We will present a simple realization of a sub-symbolic hierarchical organization, where we establish within a set of unlabelled inputs an approximative relation which serves as a decision function to accelerate retrieval. Our progressive retrieval procedure navigates a tree-like structure of increased resolution until arriving at the leaf level, which corresponds to the original full content neurons. At each level, unnecessary nodes are pruned from the process, avoiding computationally and energetically expensive operations.

3.1 Intuition

As we have seen on the previous chapter, Steinbuch-type associative memories are single-layer networks in form. Thus, the straight-forward representation for a given memory state is simply a matrix where the synaptic connection strengths are encoded according to a specified learning prescription. This abstract mathematical representation has obvious physical implementations in dedicated hardware or conventional computers; in fact it is remarkably compact and easily mapped to low-level efficient constructs thanks to the synaptic weight restriction to binary values.

For correct behaviour and ideal storage capacities, the sparse coding constraint must be met and to reach optimal results the set of stored associations $S = (x^\mu, y^\mu)$ should be comprised of patterns with logarithmic activity levels, i.e., $|x^\mu|_0 \sim \log m$ and $|y^\mu|_0 \sim \log n$ for networks of size $m \times n$. Loading a memory with such a set of associations will result on a *balanced* synaptic potentiation, where the number of active ("1") synapses is of the same order as the number of silent ("0") synapses (Knoblauch et al., 2009), eventually reaching a capacity maximum up to the theoretical limit of $C = 0.69$ bits/synapse.

A balanced weight matrix created from sparse and equally distributed patterns is efficient information-wise because a single synaptic location will be reused by several patterns at the same time without introducing undesired recall noise. However, this also implies that during the retrieval process, the dendritic potential (recall equation 2.6) of the majority of neurons will be $0 < s_j < \Theta$ and consequently will be left out on the threshold cut step. Therefore, on the classical associative memory model, unnecessary computation steps are performed on every recall operation. Of course, these superlative comparisons lead to an increased computational cost which could be avoided.

Besides being an efficient alternative model capable of performing pattern recognition and completion tasks, Steinbuch-type memories may also replicate interesting biological properties. They are the simplest realization of the theory due to Hebb (1949), and may correspond to an abstract simplified model of the basic building blocks of a mammalian brain, where local cortical representations can be seen as Hebbian cell assemblies (Braitenberg, 1978, Palm, 1982, Wickelgren, 1992). However, if this is the case, several issues arise. One of the most common critiques resides on the energy cost of performing a retrieval on a fully connected net, i.e., a network where the synaptic connection weight matrix W is balanced. The metabolic costs for sustaining a complete retrieval have been proven to be unrealistic on a biological system (Lennie, 2003). Furthermore, experimental evidence has revealed that brain areas performing functions similar to associative memory, such as the hippocampus, display degrees of connectivity

no greater than 5% (Amaral et al., 1990).

As we have seen, an associative memory will quickly reach a balanced synaptic state when approaching higher capacity levels. While other authors such as Graham and Willshaw (1994), Bosch and Kurfess (1998) have focused on artificially limiting the synaptic connectivity while maintaining the original model, our proposal suggests the use of an alternative hierarchical structural representation. Recent advances on the study of associative memories have shed light on the importance of their structural representation concerning achievable network performance (Knoblauch et al., 2009). For instance, as a means to enhance storage capacity, an alternative representation scheme based on Huffman or Goulomb coding has been proposed (Knoblauch, 2003a). Instead of encoding synaptic connectivity through its immediate weight matrix form, it has been shown that if a compressed variant is used, the classic asymptotic upper bound on storage capacity may be lifted.

Whereas in conventional Steinbuch memories information is stored within a single neural network, in our method it is spread across an ensemble of hierarchically disposed networks of increased resolution. These additional networks are successive approximated (or "compressed") versions of each other and allow for early selective filtering of relevant neurons, pruning unnecessary units from the query process while progressively reaching a final result.

Hierarchical memories are already a well-known concept in neural network architecture theory but have been employed for different purposes. A trend of research has been using such structures (Kakeya and Kindo, 1996, Hirahara et al., 2000, Štanclová and Zavoral, 2005) to store naturally correlated patterns, which would otherwise quickly saturate the network and introduce untolerable errors in the retrieval process.

3.2 On the computational complexity of retrieval

On this section we will briefly cover the computational cost and asymptotic complexity of the original retrieval process to provide a means of comparison with our new approach. Steinbuch-type memories naturally benefit from specialized massively parallel hardware implementations (see for instance Palm and Palm (1991)), as each neuron may independently perform both learning and retrieval, given that the computation is synchronous. In such a computer equipped with n processors, one for each neuron, the retrieval process will be proportional in time to the number of "1" elements of the input pattern $\tilde{x}$. As the activity level $|\tilde{x}|_0$ is close to $|x|_0$ which is of logarithmic order $O(\log m)$ due to the sparseness constraint, the time complexity of a parallel retrieval is also $O(\log m)$.

On a serial computer, the results of each neuron have to be calculated se-

quentially, for every active element on the input vector $\tilde{x}$. Generally, the so-called "pointer representation" format is employed, where $\tilde{x}$ is represented as a $|\tilde{x}|_0$-sized vector containing the indices of its "1" components (Bentz et al., 1989), avoiding their determination every time the retrieval process occurs, which would cost m additional steps. This way, n units must perform $|\tilde{x}|_0$ comparisons and a threshold cut, resulting in

$$t = n * |\tilde{x}|_0 + n \approx n * |\tilde{x}|_0 \tag{3.1}$$

operations. Assuming the classic Willshaw threshold setting strategy is used, its computational cost may be neglected. This yields a *quasilinear* time complexity of $O(n \log m)$. Notice that t is independent from the number of stored associations μ, contrarily to the traditional list-based "brute-force" solution. In this case, the input cue must be compared through a measure function χ to every other address pattern to determine the closest match, rendering a total cost of

$$t^{\text{list}} = \sum_{i=1}^{\mu} \chi(\tilde{x}, x_i) \approx \mu * (|x|_0 + |\tilde{x}|_0). \tag{3.2}$$

The list-based solution is however the only viable option when a sparse coding prescription is not available for a given problem domain, as no general solution to the sparseness constraint has been found so far.

3.3 Tree-like hierarchical memories

By inspection of the Hierarchical Subspace Tree due to Wichert (2009) we were led to an interesting question: could the properties of a tree-like structure be applied to binary associative memories in order to improve retrieval performance and minimize energy consumption?

In pursuit of this premise we conceived a hierarchical structural representation suitable for the general associative memory task. The simple intuition behind our method can be grasped through the analysis of equation 3.1. Whenever an input cue is presented to the network, every neuron will have to perform a recall procedure. However, as the stored patterns are sparse, only very few of them will eventually possess useful information for a given query — most units will be left out during the threshold cut. As such, if a memory containing approximated versions of the stored associations performs a recall first, the retrieved result may be used as an indicator of which neurons will fire positively. Of course, the method will work especially well if the employed approximation function does not lead to false negatives, otherwise undesired miss-errors could be introduced.

By applying the same process as a recursion, introducing additional layers of memories which are successive approximations of one another, a tree-like structure is built. It is not a tree of associative memories, as there is only one memory at each level; it is rather the output of the selective lookup process at step r which resembles the nodes at the level r of a tree.

Formally, we are dealing with an ordered set of R Steinbuch-type associative memories with a fixed address pattern space dimension m but a variable number of neurons $n_1, n_2, ..., n_R$ with $n_R = n$. Thus, the full $m \times n$ uncompressed associative memory can be found at depth R.

Our compression technique differs from the one found in Knoblauch (2003a) as we are interested in creating approximated versions retaining a "no false-negatives" property rather than solely maximizing space usage and information efficiency. The easiest way to achieve this goal is to apply a Boolean OR based transform, which is somehow the binary equivalent of the arithmetic mean employed in Wichert (2009).

Whenever a pair of patterns (x, y) is presented to the full memory for learning, a transformed version $(x, \zeta_r(y))\ \forall r : 1 \leq r < R$ is also presented to the r-th memory. We define ζ_r as a family of functions $\zeta_r : \{0,1\}^n \longrightarrow \{0,1\}^{n_r}$ where the elements of $z = \zeta_r(x)$ are given by equation:

$$z_i = \bigvee_{j=i\times a_r-(a_r-1)}^{i\times a_r-1} x_j. \tag{3.3}$$

The dimensions $n_1 < n_2 < ... < n_R = n$ are inversely proportional to the aggregation window factors $a_1, a_2, ..., a_R$, where $a_R = 1$, and may be expressed by the recursive relation

$$n_r = \begin{cases} n_{r+1}/a_r & \text{if } 1 \leq r < R, \\ n & \text{if } r = R. \end{cases} \tag{3.4}$$

Notice that in practice ζ_r carries on a partition of x onto n/a_r subvectors and then aggregates each of them using an a_r-ary Boolean OR.

The retrieval process is performed when a distorted or incomplete pattern $\tilde{x}$ is presented to the memory at $r = 1$, which corresponds to the smallest and most approximated version of the hierarchy. Likewise, on the following memories ranging from $r = 2$ to $r = R$, $\tilde{x}$ is used as the recall cue. However, the dendritic sum at level $r + 1$ will only have to be calculated for a subset of neurons.

Let $y(r)$ denote the output pattern returned by the r-th memory. Note that for a given index j of a "1" component of $y(r)$ corresponds an index set Y_j with

a_r elements which identify the original uncompressed units at level $r+1$:

$$Y_j = \{j * a_r, j * a_r + 1, ..., j * a_r + a_r\}. \tag{3.5}$$

These $|y(r)|_0$ sets can then be merged to form the complete set $\mathcal{Y}_{r+1}$ of indices for which the dendritic sum must be calculated at level $r+1$:

$$\mathcal{Y}_{r+1} = \bigcup_j Y_j \quad \forall j : y_j(r) = 1. \tag{3.6}$$

Hence, equation 2.6 is kept unchanged, but should be restricted to the members of $\mathcal{Y}_{r+1}$:

$$s_j(r+1) = \sum_i W_{ij}\tilde{x}_i \quad \forall j : j \in \mathcal{Y}_{r+1}. \tag{3.7}$$

Moreover, the output transfer function should also be modified in order to update only the relevant positions for which the dendritic potential has been calculated:

$$y_j(r+1) = \begin{cases} H(s_j(r+1) - \Theta) & \text{if } j \in \mathcal{Y}_{r+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

This retrieval process is nicely illustrated through Figure 3.1, where a hetero-associative task is carried on by a hierarchy of two memories.

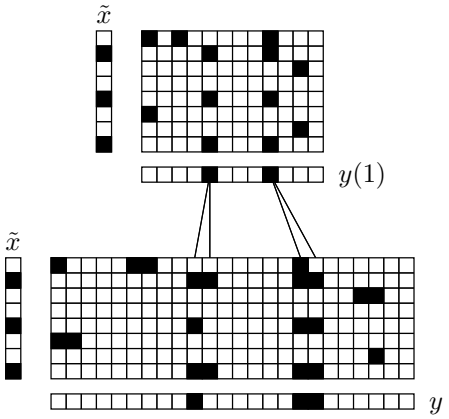


Figure 3.1: A tree-like hierarchical associative memory structure with $R = 2$ and $a_1 = 2$ during the retrieval process. The original synaptic weight matrix at $r = 2$ is 8×24, which yields a compressed $8 \times 24/a_1 = 12$ memory at $r = 1$. Black squares represent "1" entries.

3.4 Vertical aggregation and excitatory regimes

A thoughtful reader might have posed a simple question while reading the previous section: why is the aggregation operation performed solely over the horizontal dimension? Through equation 3.1 it is quite obvious that when $n \to \infty, m \to \infty$ then $t \to n$ because $|\tilde{x}|_0 \sim \log m$, if $n = m$ or $n \approx m$ which is generally the case. Thus it follows that n is the most impacting factor and a reduction on the number of traversed neurons from n to a smaller value will positively affect the total retrieval time. However, it also seems apparent that when dealing with finite memories an aggregation over each neuron's weight vector would reduce the second cost factor, even if with a lower impact on the total computational time.

Before we proceed with our analysis it is useful to define a measure of how many synapses are active at a given moment and provide a precise definition of memory load. Thanks to the simple Hebbian learning storage prescription, an exact value can be easily calculated. The fraction of "1" entries p_1 on the weight matrix W or the *memory load* can be defined as

$$p_1 = \sum_i \sum_j \frac{W_{ij}}{nm}, \tag{3.9}$$

and is a monotonic function of the number of stored patterns M. We assume that the pattern activity levels are either fixed or their mean value is fixed. Let k and l denote the fixed or mean activity levels $|x^\mu|_0$ and $|y^\mu|_0$, respectively. When performing an association during the storage phase, the probability that a synapse is *not* activated is $1 - kl/mn$. Hence, the probability that a synapse is silent after storing M patterns is

$$p_0 = \left(1 - \frac{kl}{mn}\right)^M, \tag{3.10}$$

and we can rewrite p_1 as a function of k, l and M:

$$p_1 = 1 - \left(1 - \frac{kl}{mn}\right)^M. \tag{3.11}$$

Ideal storage capacities are reached only when the loaded associations (x^μ, y^μ) are comprised of sparse vectors with *logarithmic* activity levels $k \sim c \log m$ and $l \sim d \log n$, being c and d constant and dependent on network sizes m and n. When M is large enough, this optimal memory load translates into a *balanced* synaptic potentiation, i.e., where $0 < p_1 < 1$. Ideally, $p_1 = 0.5$ while still allowing for perfect (error-free) retrieval.

Analogously to equation 3.11, the probability p_{1r} of a synapse being active

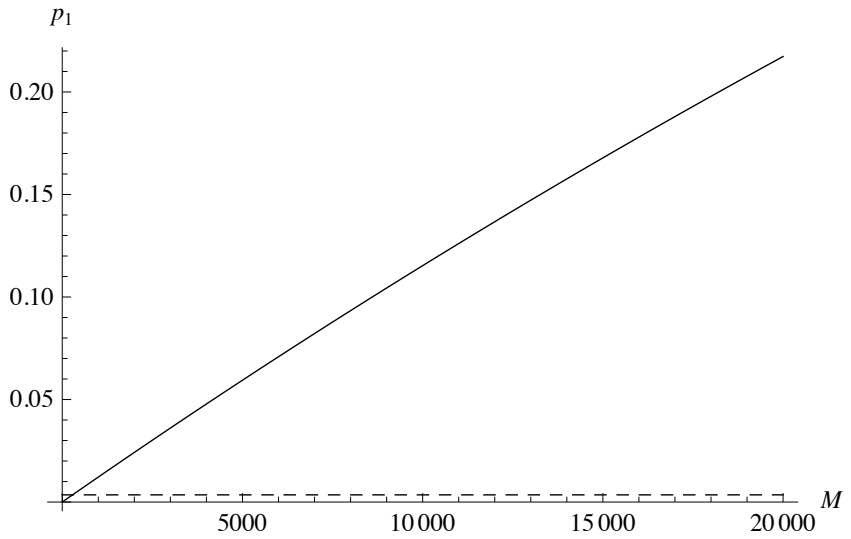


Figure 3.2: The probability p_1 of finding a "1" component related to the number of stored patterns M, when $k = l = 7$, $m = n = 2000$ and $M \in [0, 20000]$. The full line represents the variation of equation 3.11. Dashes depict the constant probability $\tilde{p}_1 = k/m$ that a component is active on an address pattern x^μ. Incomplete or noisy recall cues will have similar activity levels $|\tilde{x}_0| \approx k$. Notice how the two probabilities quickly diverge as the memory gets fuller — also quickly rendering vertical aggregation useless.

at a given depth level r can be derived when vertical aggregation is applied. The recursive aggregation operations $a_r, a_r + 1, ..., a_R$ will successively divide the original n-sized weight vectors onto smaller dimension ones:

$$p_{1r} = \quad 1 - \Big(1 - \frac{kl}{m * n/(a_r * a_{r+1} * ... * a_R)}\Big)^M = \tag{3.12}$$

$$= \quad 1 - \Big(1 - \frac{a_r * a_{r+1} * ... * a_R * k * l}{mn}\Big)^M. \tag{3.13}$$

This relation is valid for small aggregation window factors $a_r \sim a_{r+1} \sim ... \sim a_R << n$, useful when $0 < p_1 < 1$. Notice that the activity level k remains constant; as the address patterns are sparse and the positions of the "1" components are generated by a uniform random variable from 1 to n, they are not affected by the small window aggregation operations.

So, in this case, we have logarithmic k and l and supra-logarithmic weight vectors, as Figure 3.2 illustrates. We now arrive to the answer for the initial question. The problem of aggregating over the vertical dimension is that any choice of values for the window factors $a_1, a_2, ..., a_r$ will be unappropriate. Too high values, which will effectively compress the question patterns

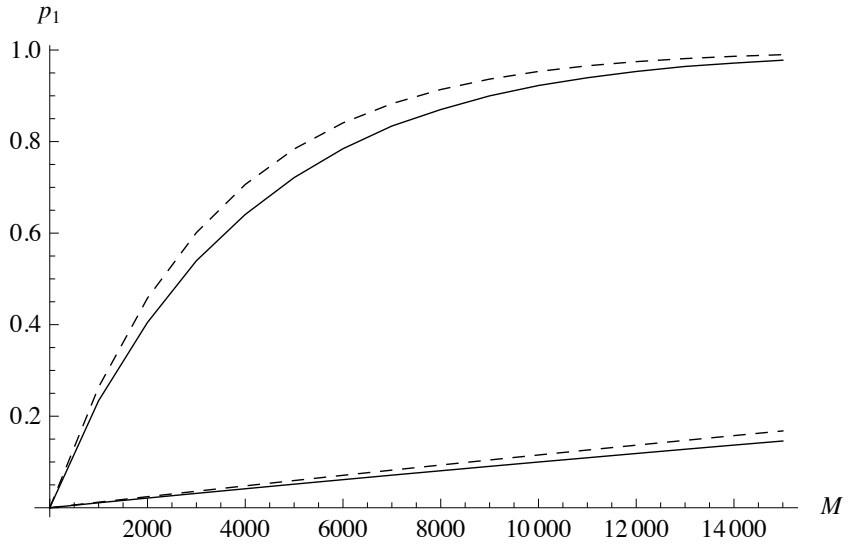


Figure 3.3: Analytical (dashed lines) and experimentally measured (full lines) values for the memory loads p_{1r} (upper lines) and p_1 (lower lines), concerning a hierarchical memory where $k = l = 7$, $m = n = 2000$, $M = 0, 1000, 2000, ..., 15000$, $R = 2$ and $a_1 = 25$. Equations 3.11 and 3.12 were used to determine the analytical values; experimental evidence was collected using the prescription described by equation 3.9 for both the aggregated matrix and the uncompressed one. Notice how the expected values (dashed lines) are very close to the measured ones (full lines), and how the aggregated matrix gets saturated much faster.

$\tilde{x}$, will introduce untolerable errors on the weight matrix, as $p_{1r} \to 1$ when $a_\rho = a_r * a_{r+1} * ... * a_R \to \infty$ converging for very small values of a_r, especially when the original load p_1 is already high (see Figure 3.3). This would result in useless lookups on the approximated memories. On the other hand, too low values will not reduce the activity level of the question patterns since $k << m$ and the "1" components are equally distributed across the whole vector. Hence, the hierarchy of memories would also be pointless as the cost determining factor $|\tilde{x}|_0 \approx k$ would not be reduced.

Excluding the obvious case of near empty memories, i.e., near the intersection point shown in Figure 3.2, performing a bidirectional aggregation could be useful for memories operating on the sparse synaptic potentiation regime, which corresponds to $p_1 \to 0$. Sparse weight matrices are obtained with sub-logarithmic pattern activity levels, i.e., when $k/\log m \to 0$ and $l/\log n \to 0$. However, besides the limited practical use for technical applications, this regime is also known to produce asymptotically vanishing storage capacities C when $m = n \to \infty$, peaking at some finite value and then tending to zero. This result

can be derived from the original capacity analysis due to Palm (1980).

3.5 Comments on pointer-based representations

As already briefly discussed on section 2.3.5, representing patterns using a pointer format where only the indices of the active components are physically stored is extremely convenient, not only because it is a compact notation thanks to the sparseness requirement but also because it allows for immediate access to the relevant positions of the synaptic weight matrix during both learning and retrieval.

On a sequential computer implementation, a related alternative representation encodes the weight vector of each *address neuron* i (i.e., each row of a synaptic weight matrix) by a singly-linked list of indices addressing a set of targets, the active synapses. This can be interesting for the sparse synaptic potentiation regime or for emptier memory states, as each row W_i will contain few active locations.

Considering a common 32-bit sequential computer is used, this so-called *pointer representation* or target list compression will typically require two words per each active synapse, viz. $s_{\text{index}} = 32$ bit to store the actual index (generally using an unsigned integer) and another $s_{\text{pointer}} = 32$ bit to store a memory pointer linking to the neighbour index. To calculate the turning point over which the use of such a representation sheme is no longer worthwhile, one can use an equation which compares the costs of representing a synapse (active or not):

$$c_{\text{target}} = c_{\text{original}} \quad \Leftrightarrow \tag{3.14}$$

$$p_1 s_{\text{index}} s_{\text{pointer}} = 1 \quad \Leftrightarrow \tag{3.15}$$

$$\Big(\sum_i \sum_j \frac{W_{ij}}{nm} \Big) s_{\text{index}} s_{\text{pointer}} = 1 \quad \Leftrightarrow \tag{3.16}$$

$$\Big(1 - \Big(1 - \frac{kl}{mn} \Big)^M \Big) s_{\text{index}} s_{\text{pointer}} = 1, \tag{3.17}$$

where s_{index} and s_{pointer} are the implementation-specific space requirements for an index and a pointer, respectively. c_{original} is fixed as the standard representation always consumes 1 bit for each synaptic location. c_{target} can either use the exact value (as expressed by equation 3.16) when calculating the ratio of active locations, useful for technical applications where a switch-over could occur, or an approximation (as suggested by equation 3.17). The latter equation is especially interesting as an analysis tool, and is solved graphically in Figure 3.4 for a set of sample parameters.

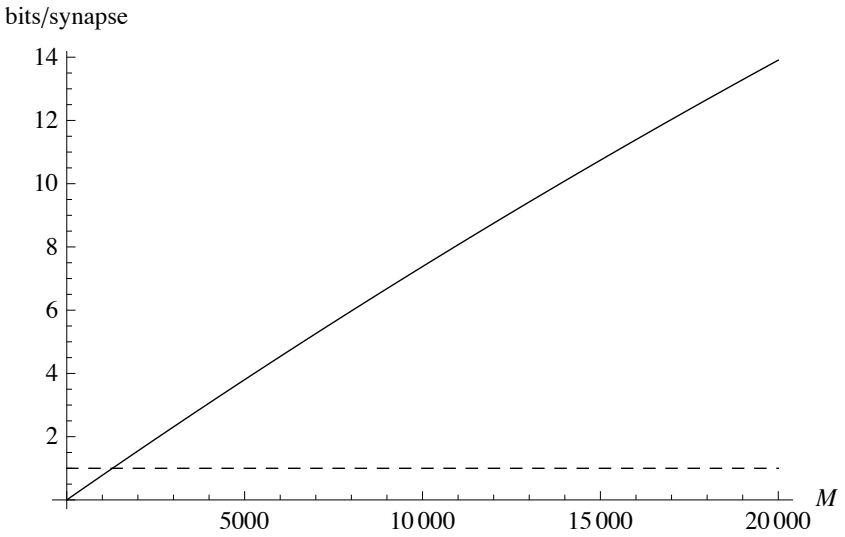


Figure 3.4: The number of required bits to represent a synapse, when $k = l = 7$, $m = n = 2000$ and $M \in [0, 20000]$, derived using equation 3.17. The full line represents c_{target} and the dashed one c_{original}. As already pointed out, the target list compression method is only useful for emptier memories when patterns with logarithmic activity levels are stored.

One should note that even when target list compression saves physical space it has an associated cost. In order to maintain the retrieval performance unaltered and avoid space waste, storing a pattern using Hebbian learning will now have to ensure that the weight rows do not contain duplicate entries. To achieve this, the new indices should be inserted in place, which is an operation of the order $O(kn) = O(m \log n)$ (instead of the classical magnitude $O(kl) = O(\log m \log n)$). Alternatively, the entire weight matrix could be cleaned from time to time, at the cost of an unnecessary temporary waste of computational resources.

Quite possibly another simple question might have popped on the reader's mind: why does the target list compression method encode the synaptic weights of the address neurons, as opposed to encoding the columns of the matrix? The answer lies on the dendritic potential calculation phase of the retrieval process, described by equation 2.6. As the potential must be determined for every position $1 < j < n$ of the address neuron population respecting $i : x_i = 1$, representing the weight columns W_j in pointer representation would involve a linear $O(m)$ search on every neuron to verify if its i-th synapse is active or not. Performing a retrieval would then be of quadratic order $O(mn)$ instead of the quasi-linear $O(n \log m)$.

As useful as it may be to encode near empty memories on a compact form,

target list compression is however not an adequate representation when combined with our hierarchical arrangement. Our modified progressive retrieval procedure will have to select individual matrix columns when it advances within the hierarchy $1 < r \leq R$. The dendritic potential at level 1 is computed using the ordinary prescription (recall equation 2.6); however, for the remaining levels, equation 3.6 implies a filtering process where only a subset of content neurons will be queried. Using target list compression would require a linear search within each address neuron's weight vector, disarming the whole purpose of our method.

Vieira (2009) suggested combining a hierarchical retrieval prescription similar to ours and representing each weight matrix column W_j in pointer format. However, the results were an order of magnitude slower than that of a classical Steinbuch-type associative memory, due to the linear search which had to be carried out over each column, rendering a total retrieval time of worst-case quadratic order $O(mn) \approx O(n^2)$. Using this representation, the performance of the retrieval process depended on the memory load p_1, as storing more patterns would lead to an increase on the number of index pointers on each weight matrix column. In fact, one of the greatest advantages of associative memories, viz. the ability of performing distributed storage with contant random access time, was being discarded.

Chapter 4

Experimental evidence

Now that a theory around our hierarchical associative memory model has been established, we may proceed and present a collection of empirical evidence to assess its merit. For the purpose we have implemented in ANSI C an efficient simulation platform on a conventional sequential computer, capable of simulating both the original and the hierarchical models. Numerical optimization methods were employed to determine ideal network parameters such as the tree height R and the corresponding aggregation window factors $a_1, ..., a_R$. For fast, unbiased uniform random pattern generation we resorted to the multipurpose GNU Scientific Library (GSL).

We have measured the effective retrieval costs with varying hierarchical dispositions and compared the results with the reference single-layer Steinbuch model. We have also experimented with a combination of our hierarchical method and target list compression, which was proven inadequate — confirming the insights revealed by the former analysis of section 3.5.

4.1 Network configuration

Every experiment has been conducted on mid-sized square associative memories with $m = n = 2000$ neurons. Without loss of generality, the memories performed an auto-associative task, as similar results should be observed at equivalent (i.e., with a higher number of stored associations M) capacity loads for hetero-association. If an ideal sparseness factor of $k = 8$ is used, the network size of $m = n = 2000$ is sufficiently high to achieve reasonable capacity rates around $C_{\text{eff}} \approx 0.5$ (Hecht-Nielsen, 1989b) for hetero-association (still far from the asymptotic upperbound $C = \log 2 \approx 0.69$) and half of it $C_{\text{eff}} \approx 0.25$ for auto-association. However, larger memories would imply prohibitive costs for the numerical optimization process, without offering significant advantage to

the task of proving the performance enhancement of our method.

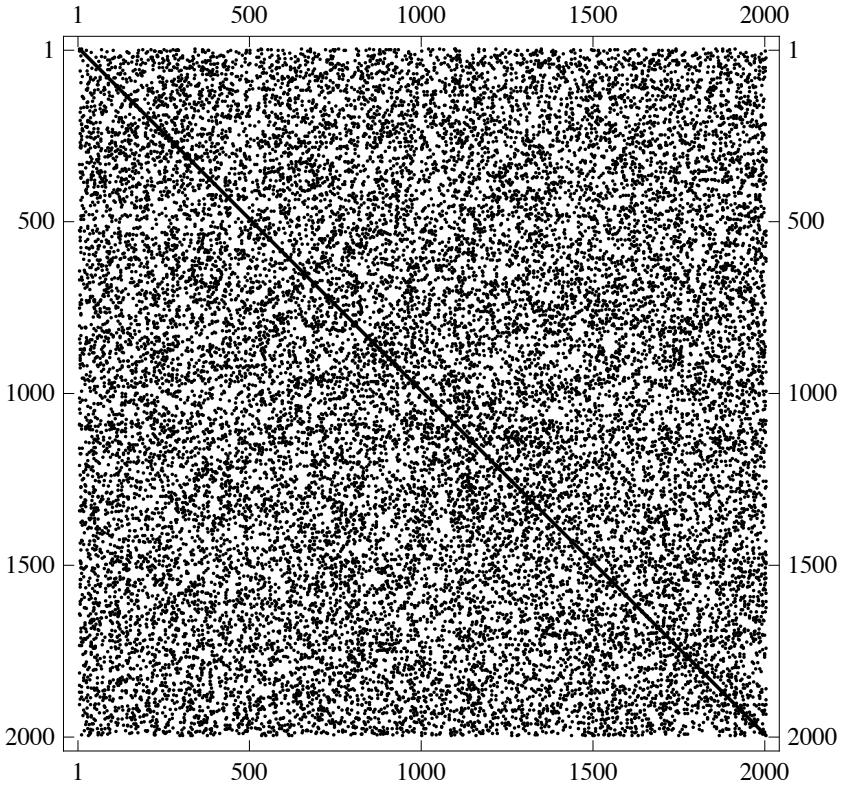


Figure 4.1: Uniform random data sets will produce evenly distributed weight matrices. Black dots represent active synapses and white dots represent silent ones. On auto-associative memories the diagonal is quickly filled with "1" entries, thanks to the Hebbian learning storage procedure. As the associations are of the form (x^μ, x^μ), each stored pair will necessarily activate $|x^\mu|_0 = k$ entries on the diagonal. For the same reason, even if not perceptible while inspecting the figure, on auto-associative memories the corresponding weight matrix should as well be symmetric.

To avoid the use of unnecessary statistical tools while analysing the results, the set of stored associations was comprised of patterns with fixed activity levels, following the style of Palm (1980). Every pattern was crafted using a high-quality pseudo-random number generator from the GSL suite according to a uniform random distribution. This way, highest capacity levels were ensured thanks to the minimal observed pattern correlation factor (see Figure 4.1).

In order to experiment with a varied range of memory load levels, we have used several data sets with increasing loads, as shown in Table 4.1. On every case, the aim of the experiment was to perform pattern completion using a noise-free incomplete question vector $\tilde{x}$ where $|\tilde{x}|_0 = k - 1$. This way, the classic

Willshaw "hard" thresholding strategy could be safely employed on an error-free regime. More complicated thresholding techniques could thus be avoided, once again for simplicity's sake.

Data set	M	$\|\tilde{x}\|_0$	$k = l$	p_1	C_{eff}
A	2000	3	4	0.007	0.016
B	2000	7	8	0.032	0.032
C	8000	7	8	0.120	0.128
D	15000	7	8	0.213	0.239

Table 4.1: Test data sets and respective pattern configurations. Notice that for $n = m = 2000$ and $k = 8$, $L \approx 0.25 * 2000^2/8^2 \approx 15625$ (recall section 2.3.4). Thus, data set D will result in a near-full memory state, even though $p_{1D} = 0.213$. The upper bound value of $p_1 = 0.5$ and a regime of perfect retrieval are possible only when $C = \log 2$, on the infinite model where $n \to \infty$ and zero correlation exists between the stored patterns (Willshaw et al., 1969, Palm, 1980). In practice, for small finite network sizes such a value would result in highly degraded retrieval quality.

An interesting memory load visualization method, perhaps more useful than simply observing the weight matrix as in Figure 4.1, is to measure the frequency distribution of the active synapse counts for each content neuron j (weight matrix column) or each address neuron (weight matrix row) i (Marcinowski, 1987). A summation ρ_j over the weights of each content neuron j (for the case of counting over the address neuron population a trivial switch from j to i is required) is carried out

$$\rho_j = \sum_{i=0}^{m} W_{ij}, \tag{4.1}$$

and the resulting values $\rho_1, ..., \rho_n$ are then arranged through sorting in order to determine the set of π groups with equal synapse count values and the number of elements on each group:

$$\underbrace{\rho_1 = \rho_2 = \rho_3}_{\varsigma_1=3} < \underbrace{\rho_4 = ...}_{\varsigma_2=...} < \underbrace{... = \rho_n}_{\varsigma_\pi=...}.$$

These $\varsigma_1, ..., \varsigma_\pi$ values can then be plotted with $x = 1, ..., \pi$, as shown in Figure 4.2. When the stored patterns are uniform random and M is large enough, the synaptic count frequency distribution should approach a Gaussian shape. On auto-associative memories calculating the distributions over the address neuron population or the content neuron should be equivalent as the weight matrices are symmetric.

4.2 Memory load impact on hierarchical retrieval

Choosing the right aggregation window factors $a_1, a_2, ..., a_R$ will depend on the memory load p_1. As we have seen on the analysis of section 3.4, on the balanced excitatory regime (i.e., when the stored patterns (x^μ, y^μ) are sparse and of logarithmic activity order) the weight matrix W will quickly become non-sparse. On finite networks some fixed bound $0 < p_1 < 0.5$ will be reached as M increases, after which perfect retrieval will no longer be possible.

Sparser weight matrices will benefit from larger aggregation factors — as the probability of finding a "1" component within a row is small, selecting a higher window size will result in higher compression efficiency, as the horizontal dimension of the compressed version will be lower and the probability of finding more than one active synapse within the window will still remain small. Likewise, for fuller memories the ideal aggregation factors will be smaller.

For each data set listed in Table 4.1 we have experimentally measured the total retrieval costs on both the original Steinbuch single-layer model and on a two-step ($R = 2$) hierarchical memory while spanning the aggregation factor a_1 across a wide range of possible values $1 < a_1 < 100$. Figures 4.3, 4.4, 4.5, 4.6 and 4.7 present the total operation count (excluding threshold cuts) t_{measured} for each aggregation factor value a_1, where $a_1 = 1$ refers to the original non-hierarchical model. Further discussion on the cost of thresholding can be found on section 4.4.

As the plots highlight, noticeable gains may be attained with a single-step hierarchy of solely one additional memory, viz. when $R = 2$. Unsurprisingly, fuller memories tend to be more sensitive to aggregation factor choice, and an incorrect selection will easily lead to uninteresting results. This phenomenom occurs due to the successively increasing memory saturation introduced when traversing the hierarchy from $r = R$ to $r = 1$ during the learning stage. The aggregation process employed to create the smaller approximated memories will also render a capacity overload, as the resulting number of neurons will be smaller and each one will contain a higher number of active synapses. The aggregation factor should of course not be too low, as it would then deliver a sub-optimal compression, but it also should not be too high, at the expense of introducing undesired add-errors in the output patterns of the approximated memories due to excessive load.

It is interesting to notice that while data set C corresponds to a memory load of roughly half the achievable limit, approximately equivalent to loading the memory with data set D, the curves of both plots resemble each other in shape. Even if the memory is still far from completely full, careful aggregation window choice can be decisive on the performance impact of using our

hierarchical retrieval prescription.

4.3 Hierarchical structures with $R > 2$

	Minimum t^{measured} excluding threshold cuts					
Memory	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$
A	6000	465	222	177	168	168
B	14000	1708	1071	973	917	931
C	14000	2674	2065	1995	2023	2065
D	14000	3710	3122	3024	3066	3129

Table 4.2: The impact of the hierarchy depth factor R on the retrieval cost. Notice that $R = 1$ corresponds to the original Steinbuch associative memory. For each subsequent value of R the minimum measured number of comparisons (again, excluding threshold cuts) is shown. To reach this result, numerical optimization was employed in order to determine the ideal $a_1, ..., a_R$ aggregation factors.

Applying the recursive aggregation through hierarchies with a depth factor $R > 2$ has also been shown useful. Table 4.2 illustrates achieved performance gains when optimum aggregation factors are chosen for each level $r = 1, ..., r = R - 1$. Hierarchy heights of logarithmic order $O(\log n)$ of network size resembling the properties of a tree (such as the Subspace Tree) appear to be beneficial to the retrieval process.

This fact matched our expectations, since our hierarchical arrangement can be described as a tree where the leaf nodes at level R are the n content neurons and each following level $r = R-1, ..., r = 1$ contains the n/a_r compressed content neurons, with increasing degrees of compression as r approaches 1. As with the Subspace Tree and in general with most tree search data structures, the biggest improvements are attained with the first hierarchical division (corresponding to $R = 2$ in our model); however, even if smaller, additional improvements can be drawn from the creation of further levels within the hierarchy.

4.4 On the cost of thresholding

Although asymptotically vanishing, the $n/a_1, n/a_2, ..., n/a_R = n$ threshold operations for each memory in the hierarchy may not be neglectable on finite size memories, especially if the number of neurons is not sufficiently large. On small- or mid-sized networks, the $O(n \log m)$ retrieval cost factor will still remain close to the $O(n)$ threshold cost. Hence, when determining the ideal architecture of

	Minimum t^{measured} including threshold cuts					
Memory	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$	$R = 6$
A	8000	2537	2383	2385	2389	2483
B	16000	3832	3412	3393	2684	3417
C	16000	4960	4566	4598	4614	4654
D	16000	6110	5914	5962	5994	6042

Table 4.3: Repetition of the experiments performed for Table 4.2, but with t^{measured} including the threshold operation count. Notice that the actual ideal depths have decreased. The effective performance gains have also decreased, especially on memory A, where the weight matrix sparseness factor had provided the ideal conditions for error-free large window aggregation. Improvements of the same order of Table 4.2 should however still be attainable on larger networks where the cost of thresholding becomes discardable.

such hierarchical memories, the relation between depth and extra threshold cuts must be taken into account. An increased depth factor R implies extra threshold operations due to the introduced additional memories, and the effective total retrieval cost might actually be lower on shallower trees, as shown on Table 4.3.

The series of measured times t^{measured} presented in Table 4.2 discard the number of threshold cuts on purpose; the impact of these will be near irrelevant on large memories as the asymptotic growth of the dendritic potential operation count will outpace the threshold's one. Our experiments were performed on $n = m = 2000$ mid-sized memories where we could afford to apply numerical optimization, which would take rather too long to complete on larger networks. On these networks the hierarchical retrieval process should benefit from higher R factors, achieving better effective costs, as the extra threshold cuts will be neglectable. This fact is particularly interesting since Steinbuch-type memories present optimal capacity relations for very large values of n and m (Palm, 1980).

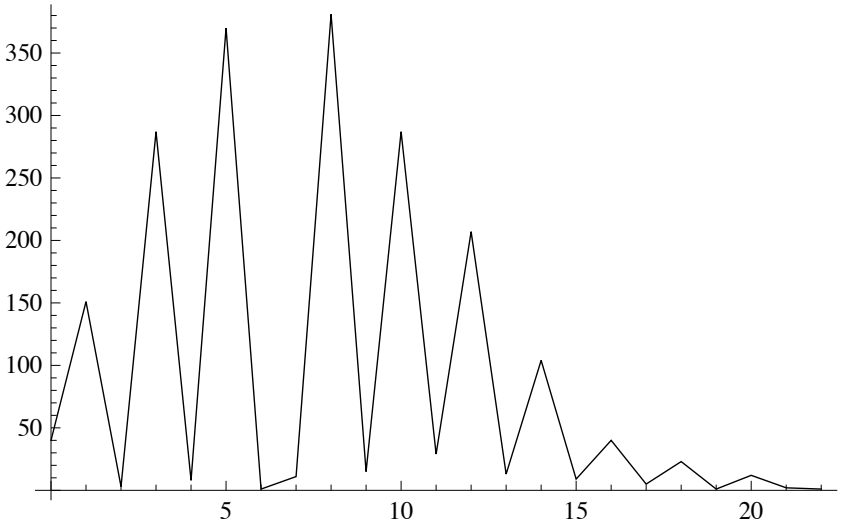
4.5 Content neuron target list compression

For the sake of completeness and in the light of our argument on target list compression and vertical aggregation of chapter 3, we have reimplemented the hierarchical model proposed by Vieira (2009). On his solution, the weight vectors of the content neuron population (i.e., the weight matrix columns) were represented in pointer format, and then subject to (vertical) aggregation on each level of the hierarchy.

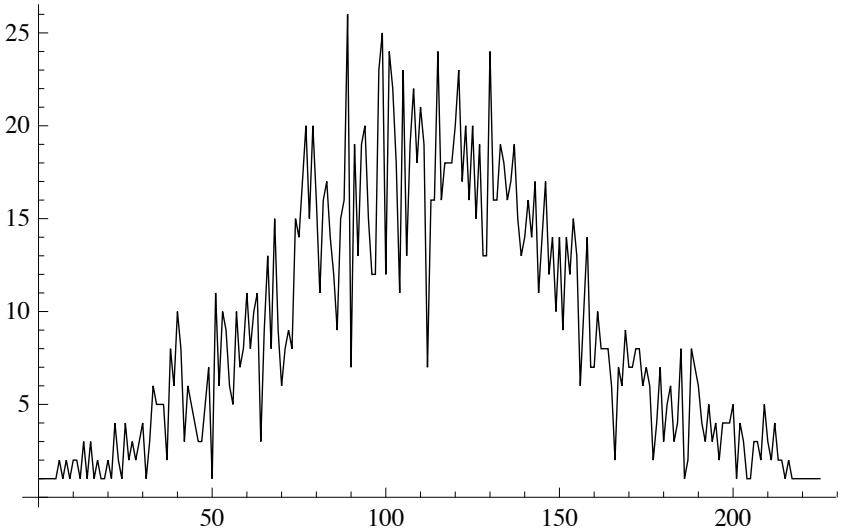
As we have seen, in most scenarios not only does this representation actually increase space usage (recall equation 3.14), but in fact also does affect quite

negatively the total retrieval time in one order of magnitude, from constant quasi-linear $O(n \log m)$ to worst-case quadratic $O(nm)$. As the retrieval process now implies a linear search over each content neuron's index pointers, its performance will depend on the memory load. When the weight matrix is sparse, the negative impact of vertical target list compression is not so noticeable, as shown by Figure 4.8. However, even for the case of data set A where the memory load $p_1 = 0.007$ is very low, the representation is already slowing down the lookup process by a factor slightly greater than twice the time taken by the original Steinbuch model.

Most prominent is the delay depicted by Figure 4.9, due to the high memory load introduced by data set D. It is noteworthy, however, how on either case the benefits of the hierarchical aggregation process are still visible, if we take apart from our consideration the negative performance factor introduced by vertical target list compression. Even if an order of magnitude slower when compared with the classical model, within its own possibilities the progressive retrieval method still applies and still allows for retrieval time improvement.



(a) Data set A



(b) Data set D

Figure 4.2: Synaptic weight count distribution for a near-empty (first plot) and a near-full (second plot) memory. When a memory is loaded with data set A, the distribution is still far from gaussian and each content neuron still has very few possible synaptic count variations due to the low value of M. As the number of stored associations increases, the range of different synaptic count values will drastically increase (in this example, from 20 to 200) and the plotted values will assume the normal bell curve.

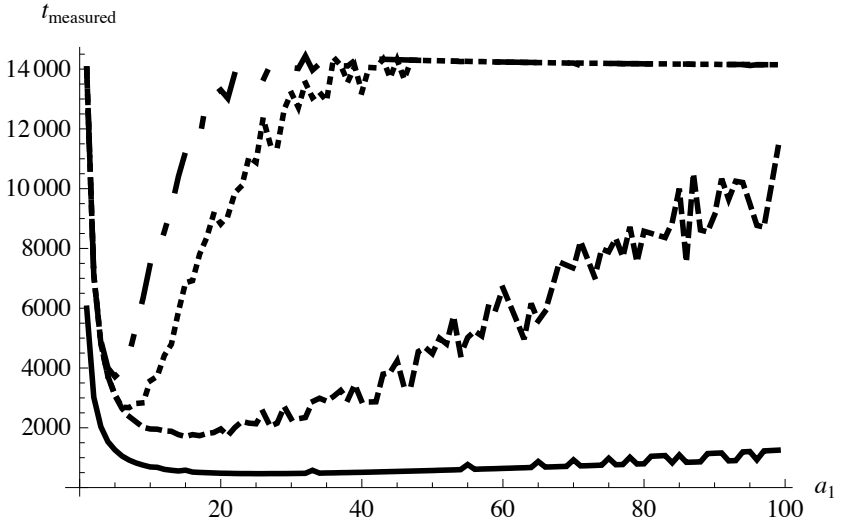


Figure 4.3: Combined view of the measured number of computation steps (excluding threshold cuts) on a two-step aggregation ($R = 2$) hierarchy with a varying a_1 factor. The series of points were drawn from an auto-associative mid-sized memory where $n = m = 2000$ loaded with uniform random sparse patterns. The full line depicts the retrieval cost corresponding to data set A; the dashed line corresponds to data set B; the dotted line to data set C; and finally, the dot-dashed line to data set D. On every case, $a_1 = 1$ corresponds to the classic associative memory model retrieval process, where no aggregation is performed. Notice how the costs $t^{\text{measured}} = 14000$ and $t^{\text{measured}} = 6000$ confirm equation 3.1.

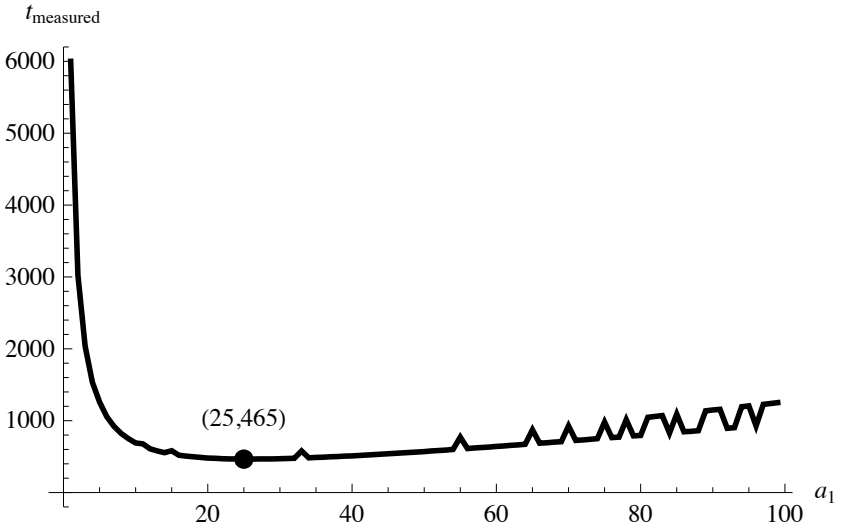


Figure 4.4: Isolated view of the retrieval costs when the test memory is loaded with data set A. Corresponds to the full line in Figure 4.3. Learning data set A generates a sparse weight matrix (recall from Table 4.1 that $p_1 = 0.007$) which allows for large aggregation windows and high improvements on the computational retrieval time. The labelled point reports the optimal aggregation value $a_{1\text{opt}}$ and the associated t_{best} cost.

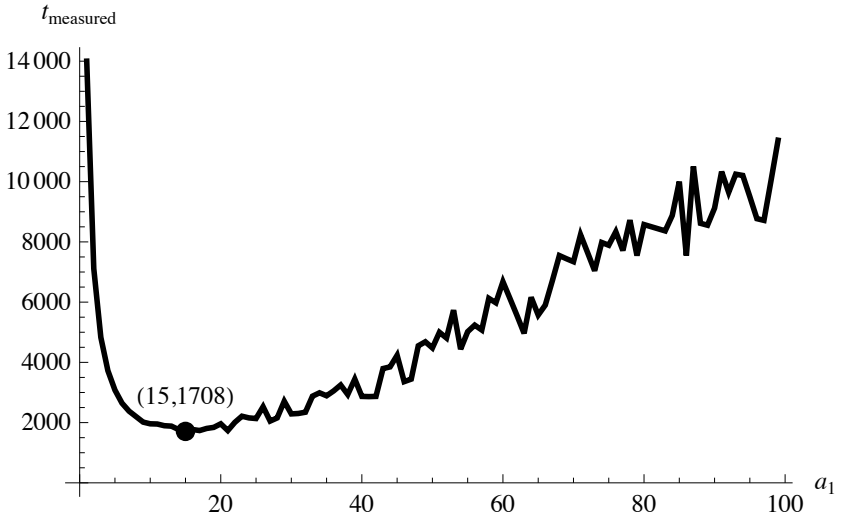


Figure 4.5: Isolated view of the retrieval costs when the test memory is loaded with data set B. Corresponds to the dashed line in Figure 4.3. The generated matrix is still far from balanced as $p_1 = 0.032$ but aggregation factor sensitivity starts to show.

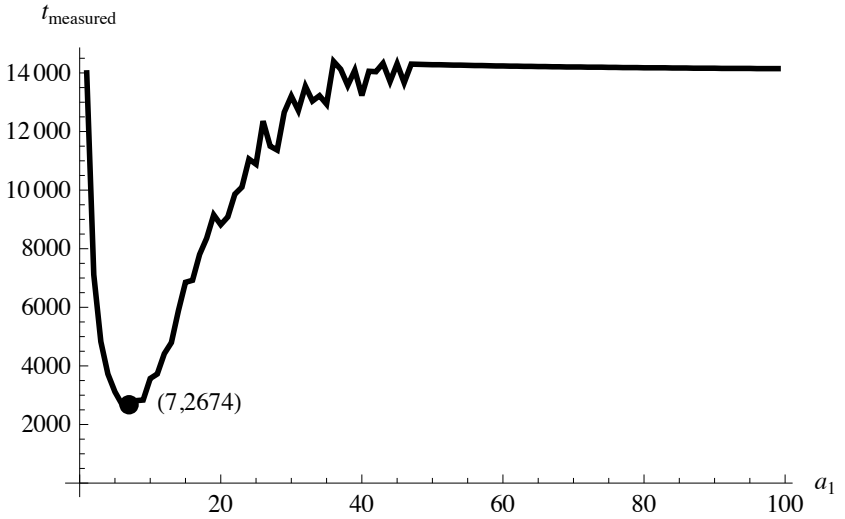


Figure 4.6: Isolated view of the retrieval costs when the test memory is loaded with data set C. Corresponds to the dotted line in Figure 4.3. High aggregation factor sensitivity is already at hand thanks to the balanced weight matrix ($p_1 = 0.128$).

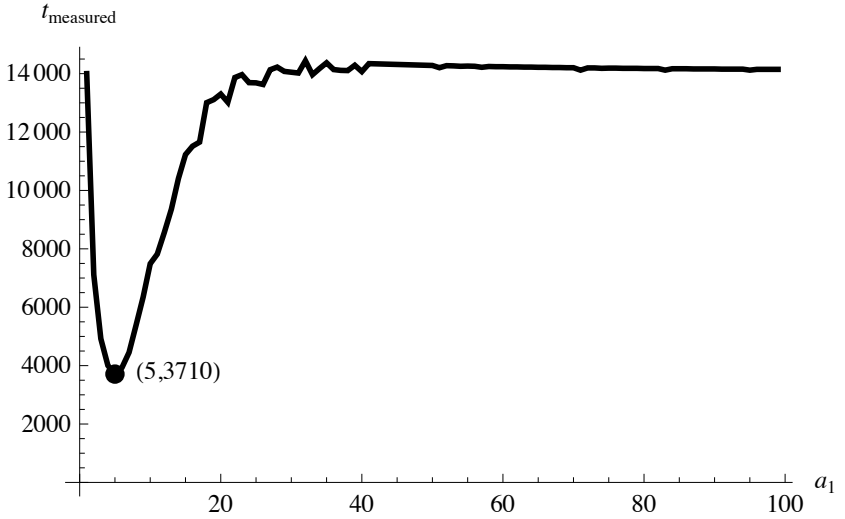


Figure 4.7: Isolated view of the retrieval costs when the test memory is loaded with data set D. Corresponds to the dot-dashed line in Figure 4.3. Although the memory load is roughly double when compared to data set C, the aggregation factor sensitivity is similar as the plot shape is near identical.

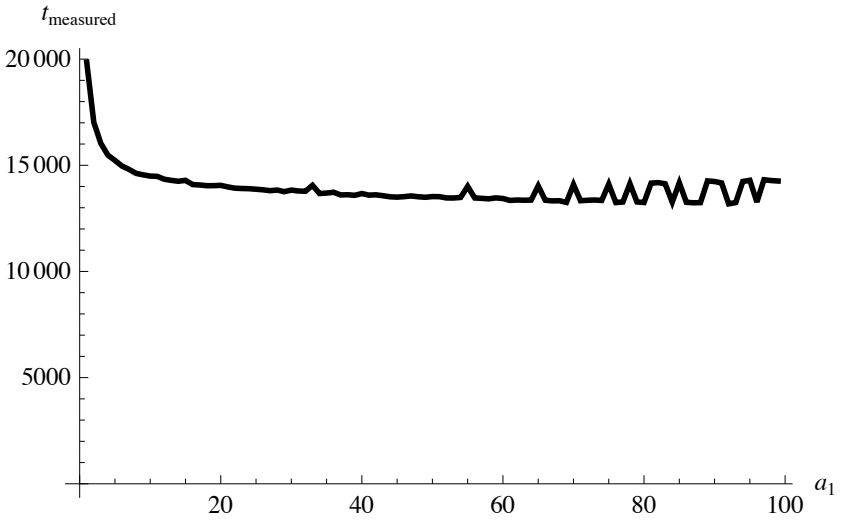


Figure 4.8: Total retrieval time t^{measured} when performing a two-step aggregation ($R = 2$) on a vertically target list compressed associative memory, when loaded with data set A. When $a_1 = 1$ we have plotted the time taken by a non-hierarchical memory, still using vertical target list compression. Notice how t^{measured} never even reaches the value of the original Steinbuch memory of 6000 (see Figure 4.4). However, it is interesting to note how the benefits of hierarchical approximation still apply.

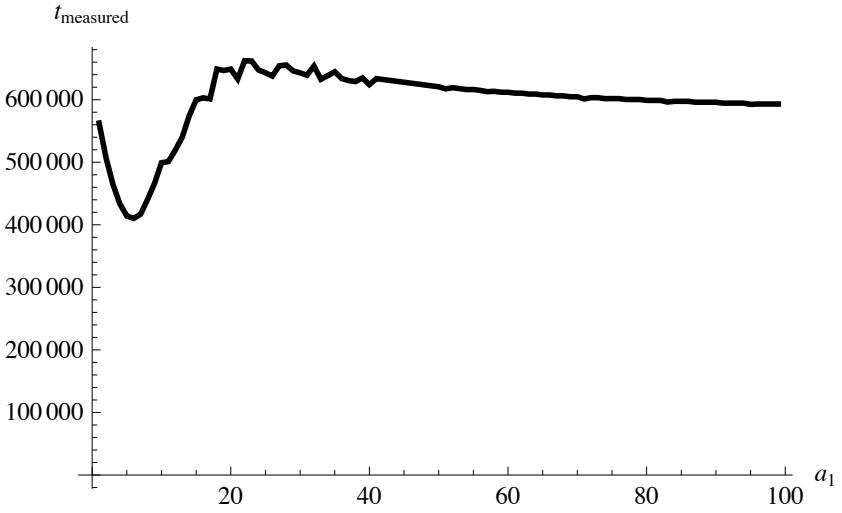


Figure 4.9: Total retrieval time t^{measured} when performing a two-step aggregation ($R = 2$) on a vertically target list compressed associative memory, when loaded with data set D. When $a_1 = 1$ we have plotted the time taken by a non-hierarchical memory, still using vertical target list compression. As expected, retrieval times are much closer to the quadratic bound of mn, as p_1 is nearer to its maximum value of 1. The disadvantages of this representation scheme are clear if we recall that a classic Steinbuch memory would take only 14000 steps to compute an answer.

Chapter 5

Conclusions

When searching through a set of structured or naturally hierarchical data it is often useful to store it and manipulate it on a tree or some similar data structure. Tremendous performance gains can be achieved when a decision function can be defined and then used to prune unnecessary computational steps from the lookup process. However, applying such a representation to neural networks performing associative memory tasks at the sub-symbolic level can be challenging as the decision function cannot be drawn from symbolic or domain-specific properties (such as posing the question "is the person taller than 1.80m?").

Inspired on the hierarchical decomposition approach of the Subspace Tree we have been able to exploit lower-dimensional mapping as a means to establish an organization on otherwise unstructured data. We think it is rather interesting how a simple function such as an n-ary (more precisely, an a-ary according to our notation) Boolean-OR is capable of creating approximations which convey useful decision information. At each step we are in fact pruning unnecessary content neurons from the search process, virtually without risk of information loss, thanks to the Boolean-OR function.

Through numerical simulations we have shown the potential of these tree-like memory structures. From the sequential computer point-of-view they have been proven useful: provided that correct parameters are chosen for the hierarchical network configuration, relevant performance gains can be achieved. Exciting conclusions might also be drawn from these results concerning real biological systems; the reader ought to appologize the hypothetical nature of our assertions. Quite possibly such a mechanism does not exist on the human brain, but until it is proved otherwise, interesting theories can still be speculated and analyzed.

5.1 Hierarchical retrieval cost analysis

From our set of collected experimental evidence we have established the importance of correctly predicting optimal (or near-optimal) aggregation window factors $a_1, a_2, ..., a_R$. Dependence of performance gains upon network configuration and memory load has been observed, as optimal or near-optimal results were restricted to a subset of depth and aggregation factor combinations. Besides depending upon obvious parameters such as the network dimensions n and m, we have seen that there is a close relationship between the memory load p_1 and the ideal aggregation factors. We may attempt to analytically sketch this relation through the study of a single-step ($R = 2$) hierarchical retrieval process.

When considering the classical analysis scenario where the stored patterns are uniform random and display fixed activity levels $k = |x^\mu|_0$ and $l = |y^\mu|_0$, performing a retrieval with an input pattern $\tilde{x}$ where $z := |\tilde{x}|_0 \leq k$ will return an output pattern $\hat{y}$ with an activity level $|\hat{y}|_0 = l$, granted that the associative memory is not overloaded. As we have seen on section 3.2, on a sequential implementation the memory will take $n * z$ steps to calculate the membrane potentials plus n final threshold operations.

Applying our progressive hierarchical retrieval prescription to the base case where $R = 2$ will (trivially) involve performing $z * n/a_1$ comparisons on the dendritic potential phase at $r = 1$, plus n/a threshold cuts. This initial lookup will result in an n/a-dimensional compressed output vector $\hat{y}(1)$. The problem arises when analysing the process at stage $r = 2$, because our filtered dendritic sum (recall equations 3.6 and 3.7) depends on the firing components of the pattern $\hat{y}(1)$. To derive an expression for the retrieval cost at $r = 2$ we only care about its activity level $w := |\hat{y}(1)|_0$, which will determine the number of content neurons we will have to traverse. The retrieval time at this stage is $z * a_1 * w$ (excluding the n final threshold operations), as we will have to verify the potentials of $a_1 * w$ (instead of n on the classic model) neurons. This renders the total cost function of

$$t_{\text{hierarchical}}(n, a_1, z, w) = z\frac{n}{a_1} + \frac{n}{a_1} + za_1w + n \approx z\frac{n}{a_1} + za_1w, \tag{5.1}$$

which could be easily analyzed to derive an ideal a_1 factor, provided that w would be known. The approximation where the threshold cuts are neglected is valid for large z (occuring at very large n).

Unfortunately, w is not a linear function of l. Compressing the memory matrix will result in overloading it (recall equation 3.12 and Figure 3.3), generally beyond the practical upper bound of $p_1 \approx 0.2$ (Knoblauch, 2003b). This memory overload introduces add-errors in the retrieval process at $r = 1$, which

translates into $w > l$. Ideally, in order to minimize equation 5.1, it would be most interesting to produce a highly compressed (large a_1) version of the weight matrix where the retrieval process would return vectors with unchanged or nearly unchanged activity levels $w \approx l$. However, a_1 and w cannot be optimized at the same time, since increasing a_1 will lead to extra add-errors and therefore an increase in w.

One open possibility to estimate w is to resort to probability and information theory and follow the classical analyses due to Willshaw et al. (1969) and Palm (1980). Although not yet done, for sparse uniform random patterns it should be possible to relate the increased overload introduced by the windowed aggregation operation with the output add-errors w at each level r. Similar work has been carried out by Sommer (1993) on his extension of the classical information storage results.

5.2 Biological implications of our contribution

One of the most interesting properties of Steinbuch-type associative memories is the model's biological richness. Its generous storage properties combined with its high biological relevance have ensured continuous received attention. When sparse coding, partial connectivity and noise-tolerant thresholding strategies are combined, Steinbuch memories become very attractive from a biological viewpoint, while maintaining high computational efficiency (Graham and Willshaw, 1994). As it has been pointed out recently, these memories are still the only known way of implementing computational systems capable of simulating very large Hebbian cell assemblies resembling the mammalian cortical structure (Wennekers, 2009).

Our extension to the original single-layer formulation, although not directly inspired on biological systems, can actually display interesting links to current theories on the brain cortex organization. Synaptic activity on the mammalian brain has been related to increased energy costs which may not be sustainable by the underlying metabolic processes. Our structural representation, still in the form of an abstract mathematical model, may be a simple explanation to the biological phenomena occurring on the visual cortex. Rather different hierarchical organizations have been proposed (see Lennie (1998) for an interesting discussion), but at least there is some consensus over the benefits of hierarchical structures within the human visual system.

On one hand, neural signalling efficiency is generally understood as a requirement for sparse codes (Laughlin, 2001), which is in respect with the information theoretic aspects of the Steinbuch model. On the other hand, even with sparse codes, traversing a fully-connected network might be too expensive energy-wise.

The synaptic connectivity levels of most brain areas do not correspond to that of an infinite Steinbuch memory at maximal load, where $p_1 \to 0.5$. However, if partial connectivity is employed, as suggested by Graham and Willshaw (1994), Bosch and Kurfess (1998), an asymtoptic bound of $0.53 \leq C \leq 0.69$ bits/synapse can still be reached.

Applying our hierarchical organization to this biologically plausible associative memory would correspond to a highly sought decrease in metabolic costs. In fact, it is believed that a similar mechanism might exist in the retinal system, where some sort of filter chaining and hierarchical compression would take place (Vincent and Baddeley, 2003). We finish this work without resisting the temptation of stating our intention to go deeper on neuroscience theories and physiological measurement results elsewhere, to ascertain to greater detail the possibilities of our model.

Bibliography

Adams, R., Calcraft, L., Davey, N., 2009. Using a genetic algorithm to investigate efficient connectivity in associative memories. Neurocomputing 72 (4-6), 732 – 742.

Amaral, D. G., Ishizuka, N., Claiborne, B., 1990. Neurons, numbers and the hippocampal network. Progress in Brain Research 83, 1–11.

Amari, S., Nov. 1972. Learning patterns and pattern sequences by self-organizing nets of threshold elements. Computers, IEEE Transactions on C-21 (11), 1197–1206.

Amari, S., 1989. Characteristics of sparsely encoded associative memory. Neural Networks 2 (6), 451 – 457.

Amit, D. J., Gutfreund, H., Sompolinsky, H., Aug 1985a. Spin-glass models of neural networks. Phys. Rev. A 32 (2), 1007–1018.

Amit, D. J., Gutfreund, H., Sompolinsky, H., Sep 1985b. Storing infinite numbers of patterns in a spin-glass model of neural networks. Phys. Rev. Lett. 55 (14), 1530–1533.

Amit, D. J., Gutfreund, H., Sompolinsky, H., Mar 1987. Information storage in neural networks with low levels of activity. Phys. Rev. A 35 (5), 2293–2303.

Anderson, J. A., Sep. 1968. A memory storage model utilizing spatial correlation functions. Biological Cybernetics 5 (3), 113–119.

Anderson, J. A., 1972. A simple neural network generating an interactive memory. Mathematical Biosciences 14, 197–220.

Baddeley, R., June 1996. An efficient code in V1? Nature 381, 560–561.

Bentz, H. J., Hagstroem, M., Palm, G., 1989. Information storage and effective data retrieval in sparse matrices. Neural Networks 2 (4), 289–293.

Bosch, H., Kurfess, F. J., 1998. Information storage capacity of incompletely connected associative memories. Neural Networks 11 (5), 869 – 876.

Braitenberg, V., 1978. Cell assemblies in the cerebral cortex. Vol. 21 of Lecture Notes in Biomathematics. Springer-Verlag, Ch. Theoretical Approaches to Complex Systems, pp. 171–188.

Braitenberg, V., Schüz, A., 1998. Cortex: statistics and geometry of neuronal connectivity. Springer, Berlin, Germany, iSBN: 3-540-63816-4.

Bruce, A. D., Canning, A., Forrest, B., Gardner, E., Wallace, D. J., 1986. Learning and memory properties in fully connected networks. AIP Conference Proceedings 151 (1), 65–70.

Buckingham, J., Willshaw, D., 1992. Performance characteristics of the associative net. Network: Computation in Neural Systems 3 (4), 407–414.

Buhmann, J., Divko, R., Schulten, K., Mar 1989. Associative memory with high information content. Phys. Rev. A 39 (5), 2689–2692.

Burks, A., Goldstine, H., von Neumann, J., 1946. Preliminary discussion of the logical design of an electronic computing instrument. Tech. rep., U.S. Army Ordnance Department.

Chen, Y., 2002. Global stability of neural networks with distributed delays. Neural Networks 15 (7), 867 – 871.

Chou, P., Mar 1989. The capacity of the kanerva associative memory. IEEE Transactions on Information Theory 35 (2), 281–298.

Cruz, A. A., de Leon, J. D., Yanez-Marquez, C., Nieto, O. C., 2005. Pattern recognition and classification using Weightless Neural Networks (WNN) and Steinbuch's Lernmatrix. In: Proc. SPIE. Vol. 5916. SPIE.

Davey, N., Adams, R., 2004. High capacity associative memories and connection constraints. Connection Science 16 (1), 47–65.

Dayan, P., Abbott, L., 2001. Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT Press.

Donoho, D. L., Elad, M., Mar. 2003. Optimally sparse representation in general (nonorthogonal) dictionaries via minimization. Proceedings of the National Academy of Sciences of the United States of America 100 (5), 2197–2202.

François, O., 1996. New rigorous results for the Hopfield's neural network model. Neural Networks 9 (3), 503 – 507.

Gardner, E., 1988. The space of interactions in neural network models. Journal of Physics A: Mathematical and General 21 (1), 257.

Graham, B., Willshaw, D., Mar. 1995. Improving recall from an associative memory. Biological Cybernetics 72 (4), 337–346.

Graham, B., Willshaw, D. J., 1994. Capacity and information efficiency of a brain-like associative net. In: NIPS. pp. 513–520.

Grossberg, S., 1972. Pattern learning by functional-differential neural networks with arbitrary path weights. In: Schmitt, K. (Ed.), Delay and functional-differential equations and their applications. New York: Academic Press.

Grossman, T., Jagota, A., 1993. On the equivalence of two Hopfield-type networks. In: IEEE International Conference on Neural Networks. Vol. 2. pp. 1063–1068.

Haines, K., Hecht-Nielsen, R., Jul 1988. A BAM with increased information storage capacity. In: IEEE International Conference on Neural Networks. pp. 181–190 vol.1.

Hamming, R., 1950. Error detecting and error correcting codes. Bell System Technical Journal 26 (2), 147–160.

Haque, A. L., Cheung, J. Y., 1994. Preprocessing of the input vectors for the linear associator neural networks. In: IEEE International Conference on Neural Networks. Vol. 2. pp. 930–933.

Hebb, D. O., June 1949. The organization of behavior: a neuropsychological theory. Wiley, New York.

Hecht-Nielsen, R., 1987. Combinatorial hypercompression. In: IEEE International Conference on Neural Networks. IEEE Press, New York.

Hecht-Nielsen, R., 1989a. Neurocomputing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Ch. Learning Laws: Self-Adaptation Equations.

Hecht-Nielsen, R., 1989b. Neurocomputing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Ch. Associative Networks: Data Transformation Structures.

Hecht-Nielsen, R., 1989c. Neurocomputing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Ch. Spatiotemporal, Stochastic, and Hierarchical Networks.

Hely, T., Willshaw, D., Hayes, G., May 1997. A new approach to Kanerva's sparse distributed memory. IEEE Transactions on Neural Networks 8 (3), 791–794.

Hertz, J., Palmer, R. G., Krogh, A. S., 1991. Introduction to the theory of neural computation. Perseus Publishing.

Hirahara, M., Oka, N., Kindo, T., 2000. A cascade associative memory model with a hierarchical memory structure. Neural Networks 13 (1), 41 – 50.

Hobson, S., Austin, J., 2009. Improved storage capacity in correlation matrix memories storing fixed weight codes. In: International Conference on Artificial Neural Networks 2009. pp. 728–736.

Hodge, V. J., Lees, K. J., Austin, J. L., Apr. 2004. A high performance k-NN approach using binary neural networks. Neural Networks 17 (3), 441–458.

Hopfield, J. J., Apr. 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America 79 (8), 2554–2558.

Hopfield, J. J., May 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proc Natl Acad Sci U S A 81 (10), 3088–3092.

Hopfield, J. J., Tank, D. W., Jul. 1985. Neural computation of decisions in optimization problems. Biological Cybernetics 52 (3), 141–152.

Humpert, B., Jun 1990. Bidirectional associative memory with several patterns. In: IJCNN International Joint Conference on Neural Networks. pp. 741–750 vol.1.

Jagota, A., 1994. A Hopfield-style network with a graph-theoretic characterization. Journal of Artificial Networks 1 (1), 145–166.

Jagota, A., Narasimhan, G., Regan, K. W., 1998. Information capacity of binary weights associative memories. Neurocomputing 19 (1-3), 35 – 58.

Kakeya, H., Kindo, T., 1996. Hierarchical concept formation in associative memory composed of neuro-window elements. Neural Networks 9 (7), 1095 – 1098.

Kanerva, P., 1984. Self-propagating search: a unified theory of memory (address decoding, cerebellum). Ph.D. thesis, Stanford University, Stanford, CA, USA.

Kanerva, P., 1988. Sparse Distributed Memory. MIT Press.

Kanerva, P., 1993. Associative Neural Memories. Oxford University Press, Ch. Sparse Distributed Memory and Related Models, pp. 50–76.

Kavukcuoglu, K., Ranzato, M., LeCun, Y., 2008. Fast inference in sparse coding algorithms with applications to object recognition. Tech. rep., Computational and Biological Learning Laboratory.

Keeler, J. D., 1988. Comparison between Kanerva's SDM and Hopfield-type neural networks. Cognitive Science 12 (3), 299 – 329.

Kennedy, M., Chua, L., May 1988. Neural networks for nonlinear programming. IEEE Transactions on Circuits and Systems 35 (5), 554–562.

Knoblauch, A., 2003a. Optimal matrix compression yields storage capacity 1 for binary willshaw associative memory. In: International Conference on Artificial Neural Networks and Neural Information Processing. pp. 176–176.

Knoblauch, A., 2003b. Synchronization and pattern separation in spiking associative memory and visual cortical areas. Ph.D. thesis, Department of Neural Information Processing, University of Ulm, Germany.

Knoblauch, A., 2005. Neural associative memory for brain modeling and information retrieval. Information Processing Letters 95 (6), 537 – 544, applications of Spiking Neural Networks.

Knoblauch, A., Palm, G., Sommer, F. T., 2009. Memory capacities for synaptic and structural plasticity. Neural Computation (in press).

Kohonen, T., 1972. Correlation matrix memories. IEEE Transactions on Computers C-21 (4), 353–359.

Kohonen, T., 1989a. Self-organization and associative memory: 3rd edition. Springer-Verlag New York, Inc., New York, NY, USA, Ch. Various Aspects of Memory.

Kohonen, T., 1989b. Self-organization and associative memory: 3rd edition. Springer-Verlag New York, Inc., New York, NY, USA, Ch. Pattern Mathematics.

Kosko, B., 1987. Adaptive bidirectional associative memories. Appl. Opt. 26 (23), 4947–4960.

Kosko, B., 1988. Bidirectional associative memories. Applied Optics 18, 49–60.

Laughlin, S. B., 2001. Energy as a constraint on the coding and processing of sensory information. Current Opinion in Neurobiology 11 (4), 475 – 480.

Lennie, P., 1998. Single units and visual cortical organization. Perception 27 (8), 889–935.

Lennie, P., 2003. The cost of cortical computation. Current Biology 13 (6), 493 – 497.

Leung, Y., Dong, T.-X., Xu, Z.-B., 1998. The optimal encodings for biased association in linear associative memories. Neural Networks 11 (5), 877 – 884.

Liou, C.-Y., Yuan, S.-K., Oct. 1999. Error tolerant associative memory. Biological Cybernetics 81 (4), 331–342.

Little, W. A., 1974. The existence of persistent states in the brain. Mathematical Biosciences 19 (1-2), 101 – 120.

Löwe, M., 1998. On the storage capacity of Hopfield models with correlated pattern. The Annals of Applied Probability 8, 1216–1250.

Ma, J., 1999. The asymptotic memory capacity of the generalized Hopfield network. Neural Networks 12 (9), 1207 – 1212.

Marcinowski, M., 1987. Codierungsprobleme beim assoziativen speichern. Master's thesis, Fakultät für Physik der Eberhard-Karls-Universität Tübingen.

McCulloch, W., Pitts, W., Dec. 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology 5 (4), 115–133.

McEliece, R., Posner, E., Rodemich, E., Venkatesh, S., Jul 1987. The capacity of the Hopfield associative memory. Information Theory, IEEE Transactions on 33 (4), 461–482.

Minsky, M. L., Papert, S. A., 1969. Perceptrons: An Introduction to Computational Geometry. MIT Press.

Murakami, K., Aibara, T., July 1987. An improvement on the moore-penrose generalized inverse associative memory. Systems, Man and Cybernetics, IEEE Transactions on 17 (4), 699–707.

Nadal, J.-P., Toulouse, G., 1990. Information storage in sparsely coded memory nets. Network: Computation in Neural Systems 1, 61–74(14).

Nakano, K., July 1972. Associatron - a model of associative memory. Systems, Man and Cybernetics, IEEE Transactions on 2 (3), 380–388.

Nemoto, I., Kubono, M., 1996. Complex associative memory. Neural Networks 9 (2), 253 – 261.

Olshausen, B. A., Field, D. J., Jun. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381 (6583), 607–609.

Olshausen, B. A., Field, D. J., 1997. Sparse coding with an overcomplete basis set: a strategy employed by V1. Vision Research 37, 3311–3325.

Palm, G., Feb. 1980. On associative memory. Biological Cybernetics 36 (1), 19–31.

Palm, G., 1982. Neural Assemblies, an Alternative Approach to Artificial Intelligence. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Palm, G., 1991. Memory capacities of local rules for synaptic modification: a comparative review. Concepts in Neuroscience 2, 97–128.

Palm, G., Palm, M., 1991. Parallel Associative Networks: The PAN-system and the Bacchus-Chip. In: Ramacher, U., Rückert, U., Nossek, J. (Eds.), International Conference on Microelectronics for Neural Networks. Kyrill & Method, pp. 411–416.

Palm, G., Schwenker, F., Sommer, F. T., Strey, A., 1997. Neural associative memory. In: Krikelis, A., Weems, C. C. (Eds.), Associative Processing and Processors. IEEE Computer Society Press, pp. 307–326.

Palm, G., Sommer, F., 1992. Information capacity in recurrent McCulloch-Pitts networks with sparsely coded memory states. Network: Computation in Neural Systems 3, 177–186(10).

Palm, G., Sommer, F. T., 1996. Associative data storage and retrieval in neural networks. Models of Neural Networks: Association, Generalization, and Representation (Physics of Neural Networks) 3, 79–118.

Rojas, R., July 1996. Neural Networks: A Systematic Introduction. Springer, Ch. The biological paradigm.

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65 (6), 386–408.

Rosser, J. B., 1982. Highlights of the history of the lambda-calculus. In: LFP '82: Proceedings of the 1982 ACM symposium on LISP and functional programming. ACM, New York, NY, USA, pp. 216–225.

Ryan, S., Andreae, J., Jan 1995. Improving the performance of Kanerva's associate memory. IEEE Transactions on Neural Networks 6 (1), 125–130.

Sanchez-Garfias, F. A., de Leon, J. L. D., Yanez-Marquez, C., Aug. 2005. A new theoretical framework for the Steinbuch's Lernmatrix. In: Proc. SPIE. Vol. 5916. SPIE, San Diego, CA, USA, pp. 59160N–9.

Schwenker, F., Sommer, F. T., Palm, G., Apr. 1996. Iterative retrieval of sparsely coded associative memory patterns. Neural Networks 9 (3), 445–455.

Sejnowski, T. J., Rosenberg, C. R., 1987. Parallel networks that learn to pronounce english text. Complex Systems 1, 145–168.

Shannon, C. E., 1948. A mathematical theory of communication. CSLI Publications.

Shouval, H., Shariv, I., Grossman, T., Friesem, A. A., Domany, E., 1991. An all-optical Hopfield network: theory and experiment. International Journal of Neural Systems 1 (4), 355–360.

Sommer, F. T., 1993. Theorie neuronaler Assoziativspeicher — Lokales Lernen und iteratives Retrieval von Information. Verlag Hänsel-Hohenhausen.

Sommer, F. T., Palm, G., 1999. Improved bidirectional retrieval of sparse patterns stored by Hebbian learning. Neural Networks 12 (2), 281 – 297.

Sommer, F. T., Wennekers, T., 2001. Associative memory in networks of spiking neurons. Neural Networks 14 (6-7), 825 – 834.

Sorabji, R., 1972. Aristotle on Memory. Brown University Press.

Steinbuch, K., 1961. Die Lernmatrix: Automat und Mensch. Springer-Verlag.

Strey, A., Oct 1993. Implementation of large neural associative memories by massively parallel array processors. In: Dadda, L., Wahl, B. (Eds.), International Conference on Application-Specific Array Processors. IEEE Computer Society Press, pp. 357–368.

Tanaka, T., Kakiya, S., Kabashima, Y., 2000. Capacity analysis of bidirectional associative memory. In: Proc. Seventh Int. Conf. Neural Information Processing (ICONIP). Vol. 2. pp. 779–784.

Vieira, R., June 2009. Fast associative memory. Master's thesis, Department of Computer Science and Engineering, Technical University of Lisbon.

Vincent, B. T., Baddeley, R. J., 2003. Synaptic energy efficiency in retinal processing. Vision Research 43 (11), 1285 – 1292.

Štanclová, J., Zavoral, F., 2005. Hierarchical associative memories: the neural network for prediction in spatial maps. In: International Conference on Image Analysis and Processing. Vol. 3617. Springer Berlin / Heidelberg, pp. 786–793.

Wasserman, P. D., 1993. Advanced Methods in Neural Computing. John Wiley & Sons, Inc., New York, NY, USA, Ch. Sparse Distributed Memory, pp. 56–73.

Wechsler, H., Zimmerman, G. L., 1988. 2-d invariant object recognition using distributed associative memory. IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (6), 811–821.

Wen, U.-P., Lan, K.-M., Shih, H.-S., 2009. A review of Hopfield neural networks for solving mathematical programming problems. European Journal of Operational Research 198 (3), 675 – 687.

Wennekers, T., Jun. 2009. On the natural hierarchical composition of cliques in cell assemblies. Cognitive Computation 1 (2), 128–138.

Wichert, A., 2006. Cell assemblies for diagnostic problem-solving. Neurocomputing 69, 810–824.

Wichert, A., 2009. Subspace tree. In: International Workshop on Content-Based Multimedia Indexing Conference Proceedings, IEEE. pp. 38–43.

Wickelgren, W. A., 1977. Cognitive Psychology. Prentice Hall.

Wickelgren, W. A., 1992. Webs, cell assemblies, and chunking in neural nets. Concepts in Neuroscience 3, 1–53.

Wiener, N., 1948. Cybernetics. MIT Press, Cambridge, MA.

Willshaw, D., Dayan, P., 1990. Optimal plasticity from matrix memories: what goes up must come down. Neural Computation 2 (1), 85–93.

Willshaw, D. J., Buneman, O. P., Longuet-Higgins, H. C., Jun. 1969. Non-holographic associative memory. Nature 222 (5197), 960–962.

Young, J., Lees, K., Austin, J., 1999. Performance comparison of correlation matrix memory implementations. In: International Conference on Neural Information Processing. Vol. 2. pp. 570–575 vol.2.