Iterative Retrieval of Sparsely Coded Associative Memory Patterns

F. Schwenker, F. T. Sommer and G. Palm
University of Ulm
Department of Neural Information Processing
Oberer Eselsberg, 89069 Ulm

May 22, 1995

Correspondence address:
Friedhelm Schwenker
University of Ulm
Department of Neural Information Processing
89069 Ulm
Germany

Running title: Iterative Retrieval

This work has been partially supported by the German Federal Ministry for Research and Technology (BMFT).

Iterative Retrieval of Sparsely Coded Associative Memory Patterns

F. Schwenker, F. T. Sommer and G. Palm

Abstract

We investigate the pattern completion performance of neural auto-associative memories composed of binary threshold neurons for sparsely coded binary memory patterns. Focussing on iterative retrieval, effective threshold control strategies are introduced. These are investigated by means of computer simulation experiments and analytical treatment. To evaluate the systems performance we consider the completion capacity C and the mean retrieval errors.

The asymptotic completion capacity values for the recall of sparsely coded binary patterns in one-step retrieval is known to be $\ln 2/4 \approx 17.32\%$ for binary Hebbian learning, and $1/(8 \ln 2) \approx 18\%$ for additive Hebbian learning [Palm, 1988]. These values are accomplished with vanishing error probability and yet are higher than those obtained in other known neural memory models. Recent investigations on binary Hebbian learning have proved that iterative retrieval as a more refined retrieval method does not improve the asymptotic completion capacity of one-step retrieval [Sommer, 1993]. In a finite size auto-associative memory we show that iterative retrieval achieves higher capacity and better error correction than one-step retrieval:

1. One-step retrieval produces high retrieval errors at optimal memory load. Iterative retrieval reduces the retrieval errors within a few iteration steps ($t \leq 5$).
2. Experiments with additive Hebbian learning show that in the finite model, binary Hebbian learning exhibits much better performance. Thus the main concern of this paper is binary Hebbian learning.
3. We examine iterative retrieval in experiments with up to $n = 20,000$ threshold neurons. With this system size one-step retrieval yields a completion capacity of about 16%, the second retrieval step increases this value to 17.9% and with iterative retrieval we obtain more than 19%.
4. The first two retrieval steps in the finite system have also been treated analytically. For one-step retrieval the asymptotic capacity value is approximated from below with growing system size. In the second retrieval step (and as the experiments suggest also for iterative retrieval) the finite size behaviour is different. The capacity value exceeds the asymptotic value, reaches an optimum value for finite system size, and decreases to the asymptotic bound.

Key words: neural auto-associative memory, Hebbian learning, iterative retrieval, threshold control, sparse coding.

1 Introduction

A neural associative memory is the realisation of an associative memory in a single layer artificial neural network, e.g. [Kohonen, 1983, Palm, 1982]. Two types of associative memory can be distinguished, namely *hetero-associative* and *auto-associative* memory. In hetero-association the memory stores a mapping $x \mapsto y$, a content pattern y is addressed by its input pattern x. This task is called *pattern mapping*.
This paper is about auto-associative memory, where the binary content pattern y is assumed to be equal to the corresponding input pattern x. Such memory models can be used for *pattern completion* tasks. The idea of pattern completion is that a noisy or incomplete version $\tilde{x}$ of a stored pattern x, should be completed to a pattern $\hat{x}$, which is close to x in some distance measure. As a measure for the model performance, we use the mean number of errors in the retrieved patterns and the *completion capacity* C which will be defined in section 2.
The simplest retrieval procedure for pattern completion is *one-step retrieval*, where the retrieval process stops with the output pattern that is recalled by a single memory quest. The analysis of one-step retrieval is based on elementary statistical tools, e.g. [Willshaw et al., 1969, Palm, 1980]. This analysis revealed that the highest completion capacity for one-step retrieval is attained for sparsely coded binary memory patterns. A binary vector $x \in B_n := \{0,1\}^n$ is *called sparsely coded*, if the number of components x_i with $x_i = 1$, is small in comparison to its vector lenght n. We concentrate on sparsely coded memory patterns in which the number of ones $k = \sum_i x_i$ is fixed and in the order of $\log n$.
Taking advantage of *feedback*, it should be possible to achieve better pattern completion results than with one-step retrieval. *Fixed point retrieval* from an auto-associative memory has been extensively studied some years ago. In particular, there are many results on *attractor networks* concerning the stability of networks, the number of fixed points, etc. [Little, 1974, Hopfield, 1982, Amit, 1989]. Motivated from the spin glass formalism in such networks the two neural activity values are modelled by $+1$ and -1, instead of 0 and 1. Furthermore, the coding scheme is different, the memory patterns are typically encoded by patterns with $prob[x_i = -1] = prob[x_i = 1] = 1/2$. None of these models achieve completion capacity values which can compete with the models using sparse $\{0,1\}$-patterns and one-step retrieval.
The properties of iterative retrieval of sparse patterns in neural auto-associative memory are the focus of this paper. It can be characterized by the following questions:

A Does iterative retrieval improve on one-step retrieval, and how many steps of iteration should be performed?

B How many patterns can be stored in the memory and retrieved from the memory?

C How many wrong bits in the initial pattern can be corrected during an iterative retrieval procedure?

D How many wrong bits have to be expected in the final retrieval result?

Recently, for binary biased $\{0,1\}$-patterns Gibson and Robinson presented a detailed statistical analysis and computer simulations of iterative retrieval [Gibson and Robinson, 1992]. However, their methods could not be applied to the very sparse patterns which yield the higher capacity values.
This paper is organized as follows: In section 2 we describe the memory model, explain the completion capacity C, and state the relevant asymptotic completion capacity values for different retrieval methods. Three different threshold control strategies for iterative retrieval are explained in section 3. In section 4 we present results of computer simulations in different

settings for auto-associative memory containing upto $n = 20,000$ threshold neurons. In section 5 the error probability for one-step and two-step retrieval is calculated and evaluated by computer simulation experiments. Finally we summarize and discuss our results.

2 Network model and evaluation

Single layer neural networks of n binary threshold neurons using a Hebbian learning rule [Hebb, 1949] to modify the synaptic connections have been proposed as associative memories, see e.g., [Steinbuch, 1961, Willshaw et al., 1969, Marr, 1971, Gardner-Medwin, 1976, Palm, 1980]. During a *one-step storage process* a set S of M *learning patterns* is transformed into the synaptic connectivity matrix W. We assume that these patterns are sparsely coded binary patterns each with exactly k ones:

$$S \subset B_{n,k} := \{x \in \{0,1\}^n | \sum_{i=1}^{n} x_i = k\}. \tag{1}$$

The *additive Hebbian learning* rule prescribes an incremention of the synaptic weight w_{ij}, provided that in a learning pattern x^μ both components x_i and x_j are $x_i^\mu = x_j^\mu = 1$. After learning, the synaptic weight matrix W is then given as

$$w_{ij} = \sum_{\mu=1}^{M} x_i^\mu x_j^\mu. \tag{2}$$

In *binary Hebbian learning* the synaptic weights are formed by

$$w_{ij} = \max_{\mu=1,\ldots,M} x_i^\mu x_j^\mu = \min(1, \sum_{\mu=1}^{M} x_i^\mu x_j^\mu). \tag{3}$$

In both storage models the elements of the diagonal can be set to $w_{ii} = 1$.
After the learning set S has been stored by one of the learning rules (2) or (3), a noisy version $\tilde{x}$ of a stored pattern x^μ (e.g. a subpattern of x^μ) is used as an input pattern to retrieve a completed version $\hat{x}$ of x^μ from the memory. In *one-step retrieval* we get the output pattern $\hat{x} = F\tilde{x}$ by means of the mapping $x \to Fx$, where

$$(F\tilde{x})_j = H(\sum_{i=1}^{n} w_{ij}\tilde{x}_i - \theta). \tag{4}$$

Here θ is a global threshold choice, and H is the *Heaviside function*, defined by $H(s) = 1$ for $s \geq 0$ and $H(s) = 0$ for $s < 0$. In *iterative retrieval* the output pattern $\hat{x}$ is fed back as a new input pattern. Thus a sequence $\hat{x}(t)$, $(t = 1, 2, 3, \ldots)$ is generated by iterating F. Here we concentrate on *synchronous* updating where all neurons perform their processing step (4) at the same time. Often in the literature *asynchronous* updating is considered, where a single neuron is randomly selected to perform processing step (4), see [Hopfield, 1982].
In the following we assume that the learning patterns $x^\mu \in S$ are drawn randomly and independently from the set $B_{n,k}$. Because of the fixed number of ones in each pattern x^μ, the values that can be taken by components x_i^μ, x_j^μ are not totally independent. But, for sparsely coded patterns this dependency is negligibly small. Thus, we can assume $prob[x_i^\mu] = k/n =: p$ for all $i = 1, \ldots, n$ and $\mu = 1, \ldots, M$.
Let $x, \tilde{x} \in B_n$ be binary patterns, where $\tilde{x}$ is a noisy version of x. Noise in a binary pattern results in the two types of confusion errors, described by the conditional error probabilities:

$$e_1 = prob[\tilde{x}_i = 0 \mid x_i = 1], \qquad e_0 = prob[\tilde{x}_i = 1 \mid x_i = 0]. \tag{5}$$

Let $\tilde{S} = \{\tilde{x}^\mu \,|\, x^\mu \in S\}$ be a set of noisy patterns which has been obtained from the learning set S, in such a way that in each pattern x^μ a component i with $x_i^\mu = 1$ changes with probability e_1 to $\tilde{x}_i^\mu = 0$ and a component j with $x_j^\mu = 0$ changes to $\tilde{x}_j^\mu = 1$ with probability e_0.
Because the sets S and $\tilde{S}$ are randomly generated, the *transinformation* $T(S, \tilde{S})$ of $\tilde{S}$ about S can be defined in terms of the transinformation of a single vector component:

$$T(S, \tilde{S}) := nM\Big(I(x_i) - I(x_i \,|\, \tilde{x}_i)\Big), \tag{6}$$

where $I(x_i \,|\, \tilde{x}_i)$ is the *conditional information* of a single component $\tilde{x}_i$ supposed the value of x_i is given. It is defined by

$$I(x_i \,|\, \tilde{x}_i) := \tilde{p}I(e_1) + (1 - \tilde{p})I(e_0), \tag{7}$$

where $\tilde{p}$ is given by $\tilde{p} := p(1 - e_0) + (1 - p)e_1$ and $I(\,\cdot\,)$ is the binary information function defined by $I(q) = -qldq - (1-q)ld(1-q)$ for probability $q \in [0, 1]$. In conclusion we achieve an expression for $T(S, \tilde{S})$ in terms of p, n, M (given by the set S) and the distortion probabilities e_0, e_1:

$$T(S, \tilde{S}) := nM\Big[I(p) - \tilde{p}I(e_1) - (1 - \tilde{p})I(e_0)\Big]. \tag{8}$$

Now, we fix the notation for iterative retrieval. Let $t \in \mathbb{N}$ be counting the iteration steps and $\hat{x}^\mu(0) = \tilde{x}$ be a initial input pattern derived from $x^\mu \in S$ according to the error probabilities $e_1(0), e_2(0)$ as defined in (9). The sets of output patterns which are recalled by t-step retrieval are given by $\hat{S}(t) = \{\hat{x}^\mu(t) \,|\, \mu = 1, \ldots, M\}$, where $\hat{x}^\mu(t)$ is the output at the t-th retrieval step, starting with the initial pattern $\hat{x}^\mu(0)$. For each time step $t = 0, 1, 2, \ldots$ the distortion of the patterns $\hat{x}^\mu(t)$ can be expressed by the probabilities:

$$e_1(t) = prob[\hat{x}_j^\mu(t) = 0 \,|\, x_j^\mu = 1], \qquad e_0(t) = prob[\hat{x}_j^\mu(t) = 1 \,|\, x_j^\mu = 0]. \tag{9}$$

The performance in a pattern completion task can be measured by the so-called *completion capacity* C. For iterative retrieval it is defined for $t \geq 1$ by

$$C := \frac{1}{n^2}\Big[T(S, \hat{S}(t)) - T(S, \tilde{S})\Big]. \tag{10}$$

The difference of these two transinformation values $T(S, \hat{S}(t))$ and $T(S, \tilde{S})$ is the information gain that is achieved by recall from the associative memory. Normalizing this value by the memory size n^2 (= number of synaptic weights) gives a size invariant capacity measure (bits/synapse).
In our setting, C can be written in terms of the conditional information values of the single components $x_i, \hat{x}_i(0), \hat{x}_i(t)$ of the corresponding vectors:

$$C = \frac{M}{n}\Big[I(x_i \,|\, \hat{x}_i(0)) - I(x_i \,|\, \hat{x}_i(t))\Big] \tag{11}$$

Using the relation (7) for the conditional information, the completion capacity C can be expressed in terms of the four error propabilities $e_0(t), e_1(t), e_0(0), e_1(0)$, and the three parameters k, n, M:

$$C = nM\Big[\Big(p(0)I(e_1(0)) + (1 - p(0))I(e_0(0))\Big) - \Big(p(t)I(e_1(t)) + (1 - p(t))I(e_0(t))\Big)\Big] \tag{12}$$

where the probability $p(t)$ is definied by $p(t) = p(1 - e_0(t)) + (1 - p)e_1(t)$.
For fixed $n \in \mathbb{N}$ and $t \geq 1$ the maximal completion capacity $C_n(t)$ which can be achieved with t-step retrieval from an auto-associative memory containing n neurons can be defined by

$$C_n(t) := \max_{S, e_1(0), e_0(0)} C(n, k, M, e_1(0), e_0(0), e_1(t), e_0(t)). \tag{13}$$

Here the maximum is taken over all possible learning sets S (over all k and M) and over all possible distortion settings $e_0(0)$ and $e_1(0)$. The asymptotic completion capacity $C = C(t)$ is defined by passing $C_n(t)$ to the limit $n \to \infty$.
For one-step retrieval in auto-associative memory the main results are [Willshaw et al., 1969, Palm, 1980, Palm, 1988, Palm, 1991]:

1. The asymptotic completion capacity is $C = \ln 2/4$ for binary Hebbian learning and $C = 1/(8 \ln 2)$ for additive Hebbian learning.

2. The patterns to be stored should be sparsely coded with $k \propto \log n$ ones.

3. These completion capacity values $C = \ln 2/4$ and $C = 1/(2 \ln 2)$ are obtained with initial input patterns containing $k/2$ ones of a stored pattern. Both values are achieved for sparse memory patterns with asymptotically vanishing error probability in the retrieved patterns.

For a different retrieval procedure, which extracts the set of stored patterns by a very large number of memory quests, exactly twice the capacity values of one-step retrieval can be obtained: for binary storage $F = \ln 2/2 \approx 0.346$ bits/synapse and $F = 1/(4 \ln 2) \approx 0.36$ bits/synapse for additive storage [Palm and Sommer, 1992]. This retrieval procedure cannot be used for the completion of a noisy pattern. However, these bounds limit the completion capacity C which can be expected from more refined retrieval procedures.
Since auto-association with pattern completion is the only subject of this paper we will always use the completion capacity as evaluation criterion, which will be simply called *capacity* in the following sections.

3 Threshold Control Strategies for Iterative Retrieval

For an iterative retrieval procedure it is necessary to determine a stopping criterion, and a strategy for setting the threshold in each iteration step, in order to suppress the noise from the inital input pattern. We call such a procedure a *threshold control strategy*. Basically, threshold setting in a neural network can be carried out by setting a

- global threshold $\theta = \theta_j$ [Palm, 1980, Gibson and Robinson, 1992], or by setting
- individual threshold values θ_j [Marr, 1971, Buckingham and Willshaw, 1993].

We concentrate on global threshold setting.
For a binary pattern $x \in B_n$ the set of one-components of x given by

$$A(x) = \{i \mid x_i = 1\}$$

can be identified with x, and vice versa. In particular, for sparsely coded binary patterns, in computer simulation it is convenient to use the set $A(x)$ instead of x. We also use this identification in the remainder of this paper, thus, we say that 'pattern x is part of pattern y', if $A(x) \subset A(y)$.
We assume the synaptic connectivity matrix W is formed either by additive (2) or binary (3) Hebbian learning from a learning set $S \subset B_{n,k} \subset B_n$ of M sparsely coded patterns each with exactly k ones.

3.1 Strategy lk

Let $\hat{x}^\mu$ be an input pattern containing exactly l ones of a learning pattern $x^\mu \in S$. Then, with a global threshold stetting $\theta = l$ it is guaranteed that the memory pattern x^μ is part of the retrieved pattern $\hat{x}^\mu(1)$. This, because storage of a pattern $x^\mu \in S$ leads to $w_{ij} \geq 1$ for all synaptic weights w_{ij} with $i, j \in A(x^\mu)$.
This observation suggests the following threshold setting: For an initial input pattern $\hat{x}(0)$ which contains l ones of $x^\mu \in S$, the threshold in the first retrieval step is put to $\theta(0) = l$ and to $\theta(t) = k$ in the following steps $t \geq 1$. The retrieval procedure can be stopped if the pattern $\hat{x}(t-1)$ is part of $\hat{x}(t)$ for iteration step $t \geq 2$.
In real applications the size of the common part $A(\hat{x}(0)) \cap A(x^\mu)$ is not known and has to be estimated. This can be achieved by testing different threshold values θ, and taking the largest value as $\theta(0)$ such that $A(\hat{x}(1)) \geq k$ holds. This testing procedure is similar to the subsequently described threshold control strategy CA.

3.2 Strategy $lk+$

Strategy $lk+$ is an improved version of the threshold control strategy lk. All threshold values $\theta(t)$ are defined as by strategy lk. For $t = 1$ the retrieved pattern $\hat{x}(1)$ is determined by strategy lk, whereas for $t \geq 2$ the retrieved pattern is given by boolean ANDing of the iterated output pattern with the previous pattern, i.e.,

$$\hat{x}(t) = \hat{x}(t-1) \wedge F(\hat{x}(t-1)). \tag{14}$$

Here the $\wedge$ denotes the boolean AND-function and the mapping F is defined as in (4).
Note that using strategy $lk+$, the iteration process always ends up in a fixed point $\hat{x}(t)$. This, because $\hat{x}^\mu(t)$ is part of $\hat{x}^\mu(t-1)$ for all $t \geq 2$ and additionally $\hat{x}^\mu$ is part of $\hat{x}^\mu(1)$.

3.3 Strategy CA (Constant Activity)

Because all learning patterns $x^\mu \in S$ containing exactly k ones, the threshold $\theta(t)$ could be adjusted in such a way that in each iteration step the number of ones $\hat{k}_\theta$ of the retrieved pattern $\hat{x}^\mu(t)$ is close to the expected activity k. This can be achieved by testing different threshold values θ and taking that threshold, which minimizes the absolute value of the difference between k and $\hat{k}_\theta$:

$$D(\theta) := |\hat{k}_\theta - k|. \tag{15}$$

In general, there may exist more than one threshold value θ minimizing this difference $D(\theta)$. In this case, $\theta(t)$ is set to be the smallest among these values. Typically, with this choice it is $A(\hat{x}^\mu(t+1)) \geq k$ and probably the whole learning pattern x^μ (or a large part of it) is part of the retrieved pattern $\hat{x}^\mu(t+1)$. The iteration process can be stopped, if a fixed point or a cycle in the sequence of output patterns $\hat{x}^\mu(t)$ is detected.

4 Results from computer simulations

In this section, we examine the behavior of an auto-associative memory with n threshold neurons by means of computer simulations. A set $S \subset B_{n,k}$ of sparse binary patterns, each containing exactly k ones has been generated. These k components are randomly and independently generated. Such a pattern set has been stored using a Hebbian storage rule (2) or (3), again, under the restriction $w_{ii} = 1$ for the diagonal elements.
The retrieval has been started from a noisy version $\hat{x}(0)$ of a learning pattern $x^\mu \in S$. Each initial input pattern has been generated from a learning pattern by changing a fixed number

of one-components into zero-components and vice versa. During the retrieval process both types of errors in $\hat{x}(t)$ have been observed and the completion capacity C, as defined in (12), has been calculated from the mean errors $e_0(t)$ and $e_1(t)$. These have been determined from the basis of 50 different learning sets and 500 test patterns per learning set.

4.1 Comparison of additive and binary Hebbian learning

In section 2 we already discussed that asymptotically the capacity for one-step retrieval with additive Hebbian learning is $C = 1/(8 \ln 2) \approx 18\%$, slightly higher than the asymptotic capacity $C = \ln 2/4 \approx 17.32\%$ for one-step retrieval with binary Hebbian learning. Computer simulations (for the finite system) show a completely different behavior.
In Fig. 1a the simulation results for an auto-associative memory with $n = 1900$ neurons for threshold control strategy $lk+$ are plotted. Learning sets of different sizes $M = 2000, \ldots, 15000$ have been generated, with $k = 13$ ones per pattern. These sets have been stored either with the binary or additive Hebbian learning rule. The set of initial input patterns has been generated for the retrieval phase, where each pattern $\hat{x}^\mu(0)$ is part of the corresponding $x^\mu \in S$ containing exactly $l = 6$ ones. This parameter setting has been chosen, because the maximal capacity for a network with $n = 1900$ neurons using threshold control strategy $lk+$ and binary Hebbian learning is achieved with the parameters $k = 13$ and $l = 6$ (see section 5).
In the range of $M \leq 6000$ memory patterns for additive, and in the range of $M \leq 11000$ for binary Hebbian learning, almost the same capacity results can be achieved with the strategy CA. Although the behavior of the two strategies $lk+$ and CA is very similar for these M values, it becomes more and more different as M grows (see section 4.2).
Additive Hebbian learning reaches a capacity of approximately 7% for one-step retrieval, and approximatively 9% for iterative retrieval. Almost twice of these capacity values, namely 14.5% and 18% can be obtained by binary Hebbian learning. For memory sizes up to $n = 10,000$ neurons the binary Hebbian learning rule shows always higher capacity values than the additive Hebbian learning rule.
It can be observed in Fig. 1a that in an intermediate range for M iterative retrieval yields higher capacity values. This effective storage range, here $2000 \leq M \leq 7000$ for additive and $5000 \leq M \leq 14000$ for binary learning, is a typical property of iterative retrieval. The reasons for this behavior are:

- If M is too small, one-step retrieval is sufficient, because the errors in $\hat{x}(1)$ are very low, therefore further iteration steps cannot improve the retrieval result.
- If M is too large, iterative retrieval does not improve the retrieval result, because $\hat{x}(1)$ contains too many errors, which cannot be corrected by further iteration steps.

These effective storage ranges of iterative retrieval can also be observed in Fig. 1b. Here we plot the iteration time until the threshold control strategy $lk+$ stops. Just in the effective storage range the mean iteration time rises from $t = 1$ upto $t \approx 3$ in both storage models.
Since additive Hebbian learning is apparently inferior to binary Hebbian learning on all interesting counts (capacity, iteration time and storage requirements), we will focus on binary Hebbian learning in the remainder of this paper.

4.2 Comparison of Different Threshold Control Strategies

In this section we discuss the three different threshold control strategies as described in section 3 for iterative retrieval and the binary Hebbian learning rule. In particular, we will compare their behavior in terms of the capacity C and iteration time.
We use the density of ones in the binary matrix W as a measure for the memory load. The

completion performance depends essentially on this density $p_1 = prob[w_{ij} = 1]$. It can be calculated in terms of $p = k/n$ and M:

$$p_1 = 1 - (1 - p^2)^M \approx 1 - \exp(-Mp^2) \tag{16}$$

The approximation in (16) is valid for sparsely coded patterns where p^2 is small. Asymptotically the optimal memory load for one-step retrieval is $p_1 = 0.5$ (see [Palm, 1980]).
In Fig. 2a the capacities for iterative retrieval of the three threshold strategies are depicted. The parameter setting $M = 10000$, $k = 13$ and $l = 6$, which has been used in the simulation, is almost optimal for this memory size $n = 1900$. The retrieval process started from an incomplete version of a memory pattern as initial input pattern. We observe the retrieval process until the 8th time step, because the capacity does not increase any more after that. For suboptimal *memory loads* $p_1 < 0.3$ (corresponding to $M \leq 8000$ stored patterns) we observe a linear increase of the capacity for all three threshold control strategies. The capacity increases proportional to the number of stored patterns M. In this range, the capacity values achieved by one-step, two-step and iterative retrieval are more or less the same – we practically have errorfree completion after the first retrieval step.
With control strategy $lk+$ as well as strategy CA the maximal capacity of approximately 18% is achieved for $M = 11000$ stored patterns, which corresponds to an optimal memory load of $p_{opt} \approx 0.4$ (optimal for memory size $n = 1900$). The capacity that can be achieved for iterative retrieval with the simplest strategy lk is nearly the same as with one-step retrieval.
In Fig. 2b we plot the number of iteration steps, until the iteration process is terminated. For memory loads p_1 exceeding the optimum of p_{opt} the iteration time of strategy $lk+$ decreases more or less rapidly from about $t = 3$ back to $t = 1$. Strategy CA behaves different for these memory loads; the iteration time still grows up, although the output pattern does not approach the desired memory pattern.

4.3 Optimal model

Now we consider the binary Hebbian learning rule (3) as storage procedure together with the threshold control strategy $lk+$. This has been the most promising model in our computer simulations.
In Fig. 1b the capacity values for the first and second retrieval step and for the fixed point (which is almost always reached until the 8th retrieval step) are shown. Iterated retrieval improves the capacity for large M values, where one-step retrieval produces already too many errors on average, see also Fig. 4b. For extremely high memory loads p_1 (corresponding to $M \geq 14000$ stored patterns) the performance of iterative retrieval drops back to that of one-step retrieval. For $M = 11000$, where the capacity takes its optimum, the capacity value for iterative retrieval is about 18% (see, Fig. 1a).
Depending on the amount and type of distortion in the initial input pattern, the capacity values for one-step, two-step, and iterative retrieval and binary Hebbian learning rule are shown in Fig. 3. Here $M = 10000$ patterns have been stored. The retrieval process has been started from an incomplete or noisy version of a stored pattern. The parameter l denotes the number of ones in the initial input pattern. Here $l < k$ corresponds to an input pattern which is part of a stored pattern, and $l > k$ to an input pattern which contains the whole stored pattern and additionally some extra ones. Therefore, just one type of error, namely missing or additional ones, occur in an input pattern.
The dashed line in Fig. 3 represents the upper bound for the completion capacity which could be achieved by errorfree retrieval. Near the errorfree input pattern the curves almost coincide. Obviously, the capacity is zero, if we use a memory pattern without any distortions as an input pattern, therefore the curves in Fig. 3 decrease to the capacity value $C = 0$ for $l = k = 13$. If

the errors in the input pattern increase the errors in the retrieved output pattern increase as well, and therefore the retrieval capacities stay below this upper bound.
For $l \approx k/2$ the capacity values reach their maximum. This type of diagram is 'M-shaped' (also for additive Hebbian learning) with two local maxima. The right maximum of the capacity curve for $l > k$ is always smaller than the left one due to the following reasons:

a. Because of the sparseness of the learning patterns, it is easier to correct an additional one (detection of a wrong one among the $k+1$ ones), than to correct a missing one (detection of a wrong zero among $n-(k-1)$ zeros) of a stored pattern. This is the reason for the asymmetry in the errorfree retrieval curve.

b. The models show more fault tolerance against missing ones than against additional ones in the input pattern. This can be observed in Fig. 3 here the capacity values are closer to the errorfree upper bound for input patterns with $l < k$ than for $l > k$.

For the binary storage procedure it is easy to see, that the error probability in the output pattern is higher if the input pattern contains additional ones than for missing ones in the input pattern. Let k be the number of ones in the stored pattern, and p_1 the memory load of the associative memory. Suppose the input pattern is part of a learning pattern with $l = k-m$ ones (m missing ones) and the threshold is set to $\theta = k - m$, then the error probability e_m after the first retrieval step is

$$e_m \approx p_1^{k-m} \tag{17}$$

For an input pattern containing a whole memory pattern and additionally m wrong ones, with threshold choice $\theta = k$ the error probability e_a is approximately

$$e_a \approx \sum_{j=0}^{m} \binom{k+m}{k+j} p_1^{k+j} (1-p_1)^{m-j}. \tag{18}$$

For a memory load of $p_1 = 0.5$, which is almost reached for memory sizes of $n = 20,000$ neurons, $k \geq 10$ and $m \leq k/2$ by an elementary calculation it can be shown that

$$\sum_{j=0}^{m} \binom{k+m}{k+j} p_1^{m+j} (1-p_1)^{m-j} > 1, \tag{19}$$

this shows that $e_a > e_m$.

5 Analysis of two-step retrieval

In the first part of this section we derive the error probabilities for one-step and two-step retrieval using the binary Hebbian learning rule (3) and the threshold control strategy $lk+$ as described in section 3. In section 5.2 we will compare these theoretical results with results from computer simulations.

5.1 Calculation of the retrieval errors

The calculation of the error probability for the binary Hebbian learning rule with learning patterns of constant activity is based on statistical methods introduced in [Palm, 1980]. Here one can also find the analysis of error probability for one-step retrieval.
As described above, we suppose that a set of sparse learning patterns $S \subset B_{n,k}$, is randomly generated. Using the binary Hebbian learning rule (3) the set S has been stored. As an initial input pattern an incomplete version $\tilde{x}$ with $l \leq k$ ones of a learning pattern $x^\mu \in S$ is used.

We use the threshold control strategy $lk+$, thus the threshold is set to $\theta(0) = l$ in the first retrieval step. This threshold choice guarantees that a retrieved output pattern $\hat{x}(1)$ consists of

- the whole learning pattern x^μ, and
- a number of additional (erroneous) ones.

Obviously, no one-component of the learning pattern x^μ becomes zero in the retrieved output pattern $\hat{x}(1)$. Thus, $e_1(1) = 0$ and the error probability $e_0(1)$ could be determined by

$$e_0(1) = p_1^l, \tag{20}$$

provided that $p_1 = prob[w_{ij} = 1]$ holds for each entry in the matrix W independently. A detailed consideration of the storage process show that the matrix entries of a column are dependent. Including this dependency we achieve a more realistic approximation for the error in the first retrieval step [Palm, 1980]:

$$e_0(1) = 1 + \sum_{h=1}^{l} \binom{l}{h} (-1)^h \left[\frac{n-k}{n} + \frac{k}{n-h} \cdot T(n,k,h) \right]^{M-1}, \tag{21}$$

where we use for brevity

$$T(n,k,i) := \prod_{h=0}^{i-1} \frac{n-k-h}{n-h}. \tag{22}$$

Therefore, on average $\hat{m}_{off} := (n-k)e_0(1)$ additional ones appear in the retrieved output pattern $\hat{x}(1)$. Using $\hat{x}(1)$ as input pattern for the second retrieval step the output pattern $\hat{x}(2)$ retrieved with threshold control strategy $lk+$ has the properties:

- For each component i with $\hat{x}_i(1) = 0$ it follows $\hat{x}_i(2) = 0$. This implies: $e_0(2) \geq e_0(1)$
- Provided that $x_i^\mu = 1$, it holds $\hat{x}_i(1) = \hat{x}_i(2) = 1$. Thus, it is $e_1(2) = e_1(1) = 0$.

Let us assume there are j errors (additional ones) in $\hat{x}(1)$ and $\hat{x}_q(1) = 1$ but $x_q^\mu = 0$, then the probability for $\hat{x}_q(2) = 1$ is approximately

$$\sum_{i=0}^{j} \binom{\hat{m}_{on} + j - 1}{\hat{m}_{on} + i - 1} (1-p_0)^{\hat{m}_{on}+i-1} p_0^{j-i} \tag{23}$$

where $\hat{m}_{on} := k - l$.
Because of the sparseness of the memory patterns, we can assume that the additional ones in $\hat{x}(1)$ are Poisson distributed around the mean $\hat{m}_{off} = (n-k)e_0(1)$. This leads to an approximation of the error probability $e_0(2)$:

$$e_0(2) = e^{-\hat{m}_{off}} \sum_{j=1}^{n-k} \frac{\hat{m}_{off}^j}{j!} \sum_{i=0}^{j} \binom{\hat{m}_{on} + j - 1}{j - i} (1-p_0)^{\hat{m}_{on}+i-1} p_0^{j-i}. \tag{24}$$

Again, we have to take into account the dependences between the entries of a matrix column by exchanging the expression $(1-p_0)^u p_0^v$ in formula (24) by the probability

$$p_{uv} := p\left[w_{1q} = w_{2q} = \cdots = w_{uq} = 1 \wedge w_{(u+1)q} = \cdots = w_{(u+v)q} = 0 \right]. \tag{25}$$

The probability p_{uv} can be approximated by $p_{uv} \approx p_A \cdot p_B$ with

$$p_A = 1 + \sum_{h=1}^{u} \binom{u}{h} (-1)^h \left[1 - \left(\frac{k\,T(n,k,v)}{n-v} - \frac{k\,T(n,k,v+h)}{n-v-h}\right)\right]^{M-1} \tag{26}$$

$$p_B = \left[1 - \frac{k}{n}\{1 - T(n-1,k-1,v)\}\right]^{M-l}. \tag{27}$$

Using this approximation for the terms $(1-p_0)^u p_0^v$ in (24) we obtain an estimate for the error probability $e_0(2)$ in the output pattern $\hat{x}^\mu(2)$.

5.2 Analytical and simulation results

The analytical equations (21—27) have been used in a numerical optimization procedure, in which for a given length k of the memory patterns the optimal input pattern length l, the optimal number of neurons n, and M the optimal number of memory patterns to be stored are determined. This yields a combination of parameters k, l, n, M which has been used for the generation of the set of learning patterns S and the initial input patterns $\tilde{S}$ in the computer simulations.

Based on the approximations of the error probability we found that optimal capacity values can be achieved with learning patterns containing an odd number of ones k and with input patterns containing about half of a learning pattern, precisely with $l = (k-1)/2$ ones. This has also been observed in computer simulations (compare Fig. 3).

Fig. 4a shows that the learning patterns have to be sparsely coded to reach high completion capacity values; in particular, it can be observed that the number of ones k grows with $\log n$. This result is also known for one-step retrieval (see [Palm, 1980]).

Using sparsely coded learning patterns, which means low information content per pattern, it is possible to store large learning pattern sets in the associative memory and retrieve these patterns with low error probability in order to reach high capacity values. In Fig. 4b the relation between storable learning patterns M and the number of neurons n is displayed. It shows that with growing memory size more and more patterns are storable. In particular, M the number of memory patterns exceeds n the number of neurons in the associative memory. In computer simulations the memory loads corresponding to the optimal capacity values, grow up to $p_1 \approx 0.42$ for $n = 20,000$, which is quite close to the asymptotic memory load of $p_1 = 0.5$. Fig. 5b shows that the error ratio, defined as the number of errors divided by the activity k, for one-step, two-step, and iterative retrieval observed in computer simulations, as well as, the error ratios calculated by formulae (21) for one-step and (24) to (27) and two-step retrieval, decrease with growing memory size. By very few further iterations (see Fig. 6b) it is possible to correct some of the errors which may appear in the first retrieval step. The error probability for one-step retrieval calculated by using formula (21) agree very well with the mean error observed in the computer simulations. This is also the case for the calculated error probability for two-step retrieval for memory sizes with $n \geq 5000$ neurons (compare Fig. 6a). Finally, in Fig. 5a the optimal capacity values are displayed for growing memory size. For one-step retrieval the capacity values monotonically increase to its asymptotic capacity value $C = \ln 2/4 \approx 17.32\%$. Because the error probability for one-step retrieval is vanishing in the limit $n \to \infty$, asymptotically, the capacity values of iterative, two-step and one-step retrieval are equal to $C = \ln 2/4$ [Sommer, 1993]. The capacity values for the second retrieval step and for fixed point retrieval exceeds this asymptotic value, even for resonably small memory sizes (for $n \geq 5000$). A capacity of about 19% is achieved with iterative retrieval. The capacity for two-step retrieval reaches its optimum of $C_n \approx 17.96\%$ with $n \approx 200,000$. Passing this optimum the capacity value slightly decreases to the asymptotic value.

6 Conclusion

In the finite model, binary Hebbian learning is the most effective storage procedure. Simulation experiments show that the rate of convergence to the asymptotic completion capacity seems to be much faster than for additive Hebbian learning. Even with one-step retrieval it is possible to achieve reasonable capacities, close to the asymptotic value of $C = \ln 2/4 \approx 17.32\%$ for memory sizes in the range from $n = 10^3$ to $n = 10^4$ neurons. With additive Hebbian learning the capacities are far below the asymptotic bound of $1/(8 \ln 2) \approx 18\%$ for these memory sizes. In the experiments we achieved only half of the completion capacity values obtained with binary storage. This result made a further investigation of binary Hebbian learning desireable.

With two of the proposed threshold strategies (CA and $lk+$) iterative retrieval yielded capacity values exceeding the asymptotic value for one-step retrieval. However, the results are far away from the theoretical upper bound $F = \ln 2/2 \approx 0.346$ discussed in section 2. For instance, an associative memory with $n = 20,000$ neurons achieved a capacity of about 17.9% for two-step and about 19% for iterative retrieval. In these simulations $M = 640,000$ patterns had been stored, each containing exactly $k = 19$ ones, coding about 218 Bit per pattern.

Our theoretical analysis revealed that the optimal capacity for two-step retrieval reaches its absolute maximum (which is slightly larger than the capacity value for $n = 20,000$) for $n \approx 200,000$ and than decreases towards the asymptotic value $\ln 2/4$. It appears that the same is true for iterative retrieval.

Although the optima of capacity for two-step and iterative retrieval are obtained in a range of high memory load p_1, these values are reached with a very small number of errors. The optimal capacity values are obtained by using about half of the learning pattern as initial input pattern.

For the threshold control strategy $lk+$, we observe that the iteration process takes a very short time (the mean iteration time was less than five steps). Starting the retrieval from an incomplete version of a memory pattern x, this strategy guarantees convergence to a fixed point containing the whole pattern x.

These properties together with the fact that only one bit per synapse is needed for the storage matrix, suggest the auto-associative memory with the binary Hebbian learning rule as most suitable for applications. As a consequence, the binary storage with iterative retrieval is highly recommendable in applications. For pattern mapping in bidirectional-associative memory models [Kosko, 1988] our treatment should be applicable, allowing a better exploitation of the stored information by iterative retrieval steps.

References

[Amit, 1989] Amit, D. J. (1989). *Modelling Brain Function: The world of attractor neural networks*. Cambridge University Press, Cambridge.

[Buckingham and Willshaw, 1993] Buckingham, J. and Willshaw, D. (1993). On setting unit thresholds in an incompletely connected accociative net. *Network*, 4:441–459.

[Gardner-Medwin, 1976] Gardner-Medwin, A. (1976). The recall of events through the learning of associations between their pairs. *Proceedings of the Royal Society of London B*, 194:375–402.

[Gibson and Robinson, 1992] Gibson, W. and Robinson, J. (1992). Statistical analysis of the dynamics of a sparse associative memory. *Neural Networks*, 5:645–662.

[Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behaviour*. Wiley, New York.

[Hopfield, 1982] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79.

[Kohonen, 1983] Kohonen, T. (1983). *Self-Organization and Associative Memory*. Springer, Berlin.

[Kosko, 1988] Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, man, and Cybernetics*, 18:49–60.

[Little, 1974] Little, W. (1974). The existence of persistent states in the brain. *Mathematical Biosciences*, 19:101–120.

[Marr, 1971] Marr, D. (1971). Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society of London B*, 262:23 – 81.

[Palm, 1980] Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36:19–31.

[Palm, 1982] Palm, G. (1982). *Neural Assemblies*. Springer, Berlin.

[Palm, 1988] Palm, G. (1988). *On the Asymptotic Storage Capacity on Neural Networks*, volume F41 of *NATA ASI Series*, pages 271–280. Springer-Verlag.

[Palm, 1991] Palm, G. (1991). Memory capacities of local rules for synaptic modification. *Concepts in Neuroscience*, 2:97–128.

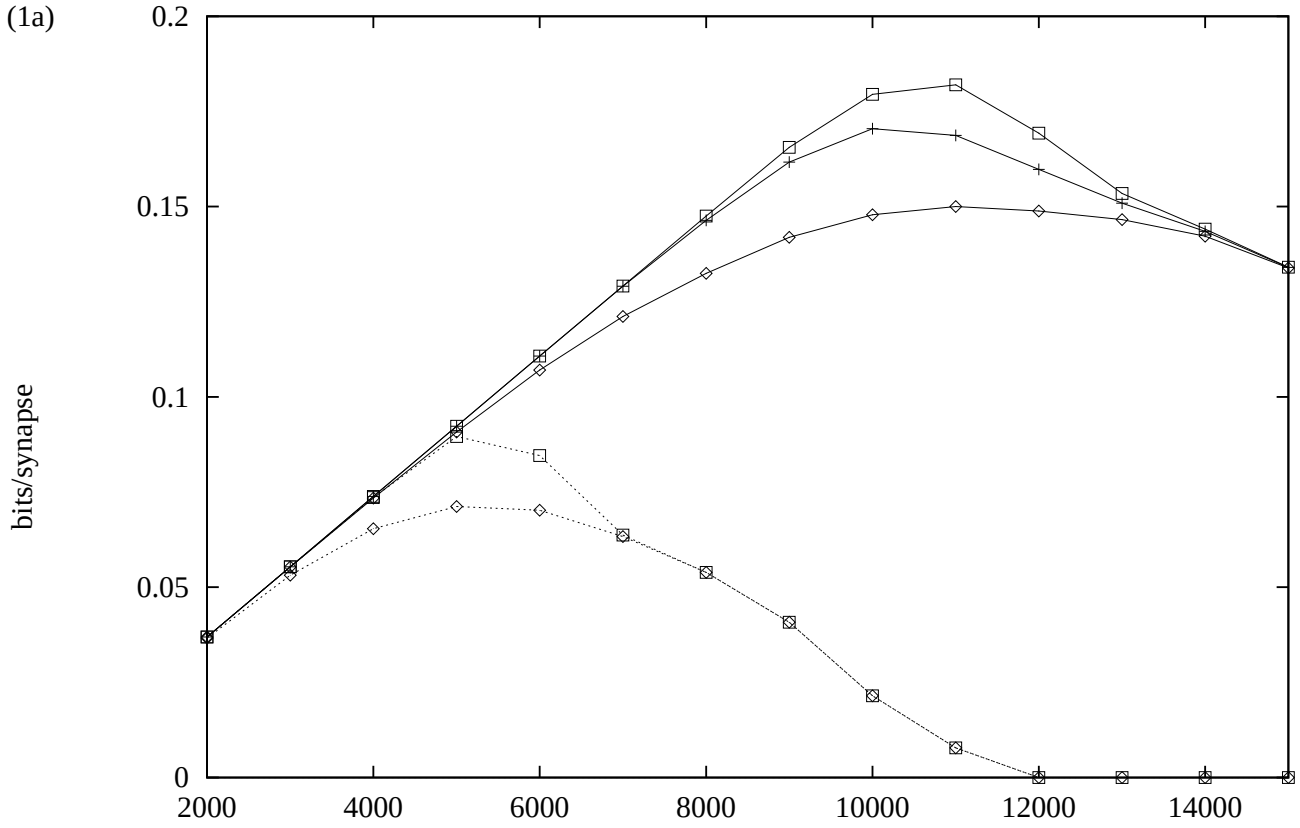
[Palm and Sommer, 1992] Palm, G. and Sommer, F. T. (1992). Information capacity in recurrent McCulloch-Pitts networks with sparsely coded memory states. *Network*, 3:1–10.

[Sommer, 1993] Sommer, F. T. (1993). *Theorie neuronaler Assoziativspeicher — Lokales Lernen und iteratives Retrieval von Information*. Hänsel-Hohenhausen.

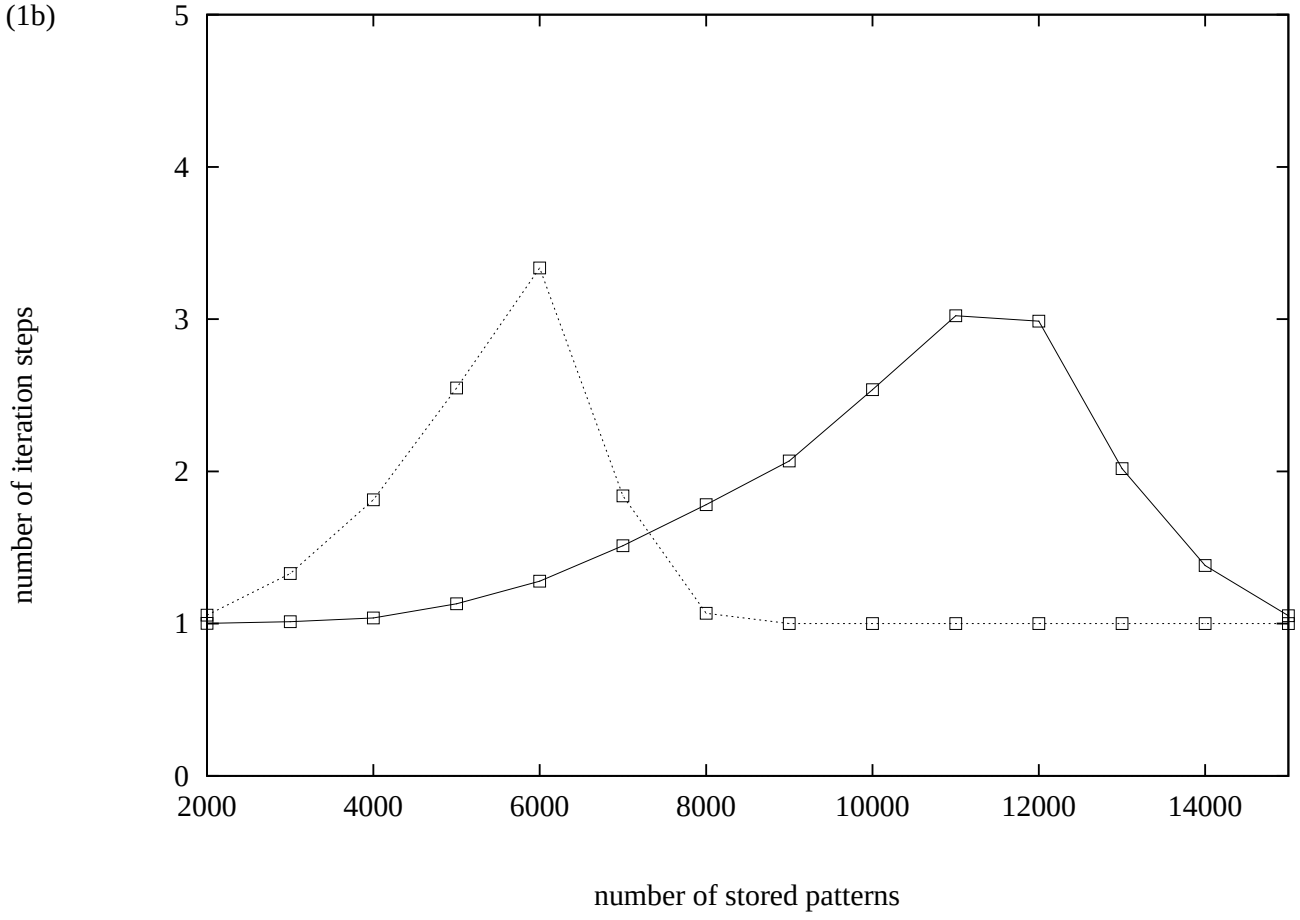
[Steinbuch, 1961] Steinbuch, K. (1961). Die Lernmatrix. *Kybernetik*, 1:36.

[Willshaw et al., 1969] Willshaw, D., Buneman, O., and Longuet-Higgins, H. (1969). Non-holographic associative memory. *Nature*, 222.

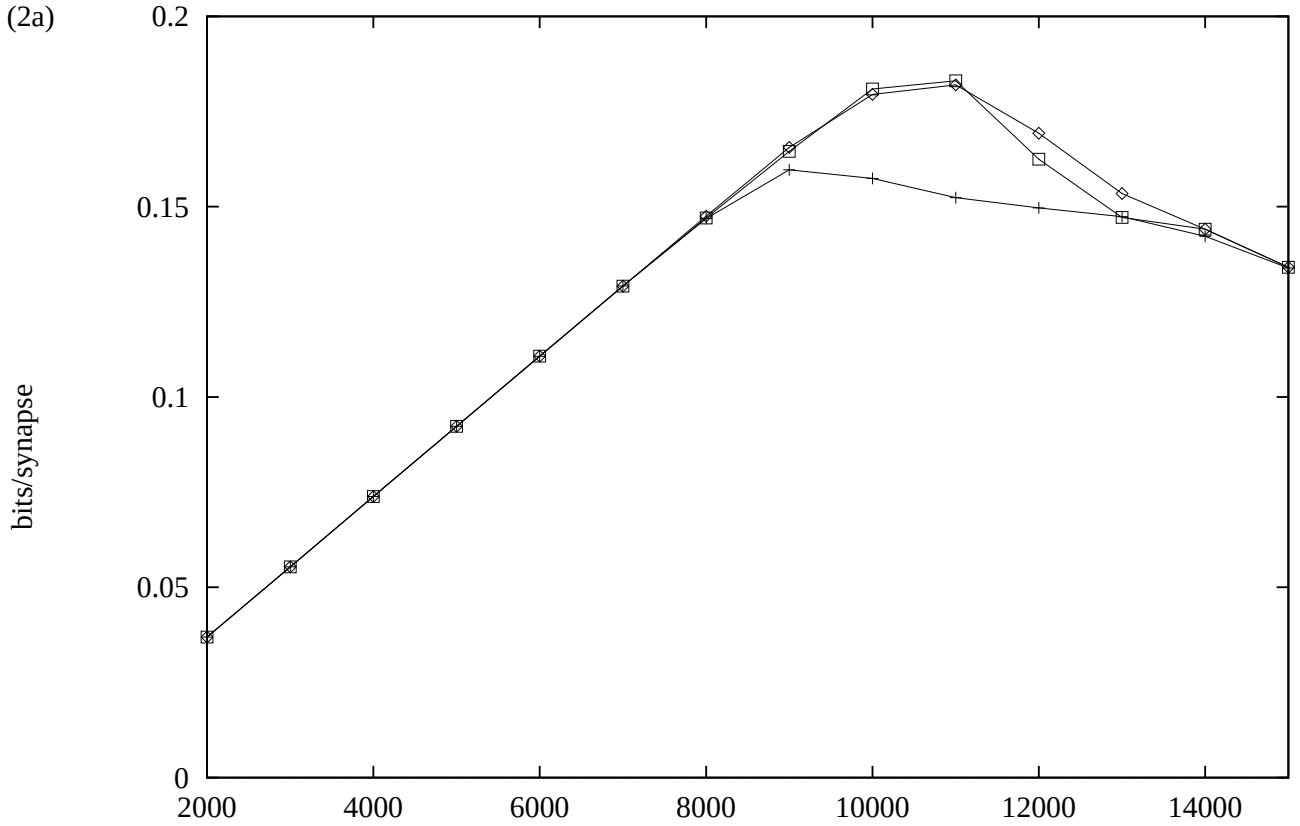
(1a)


0.2
0.15
0.1
0.05
0
bits/synapse
2000
4000
6000
8000
10000
12000
14000

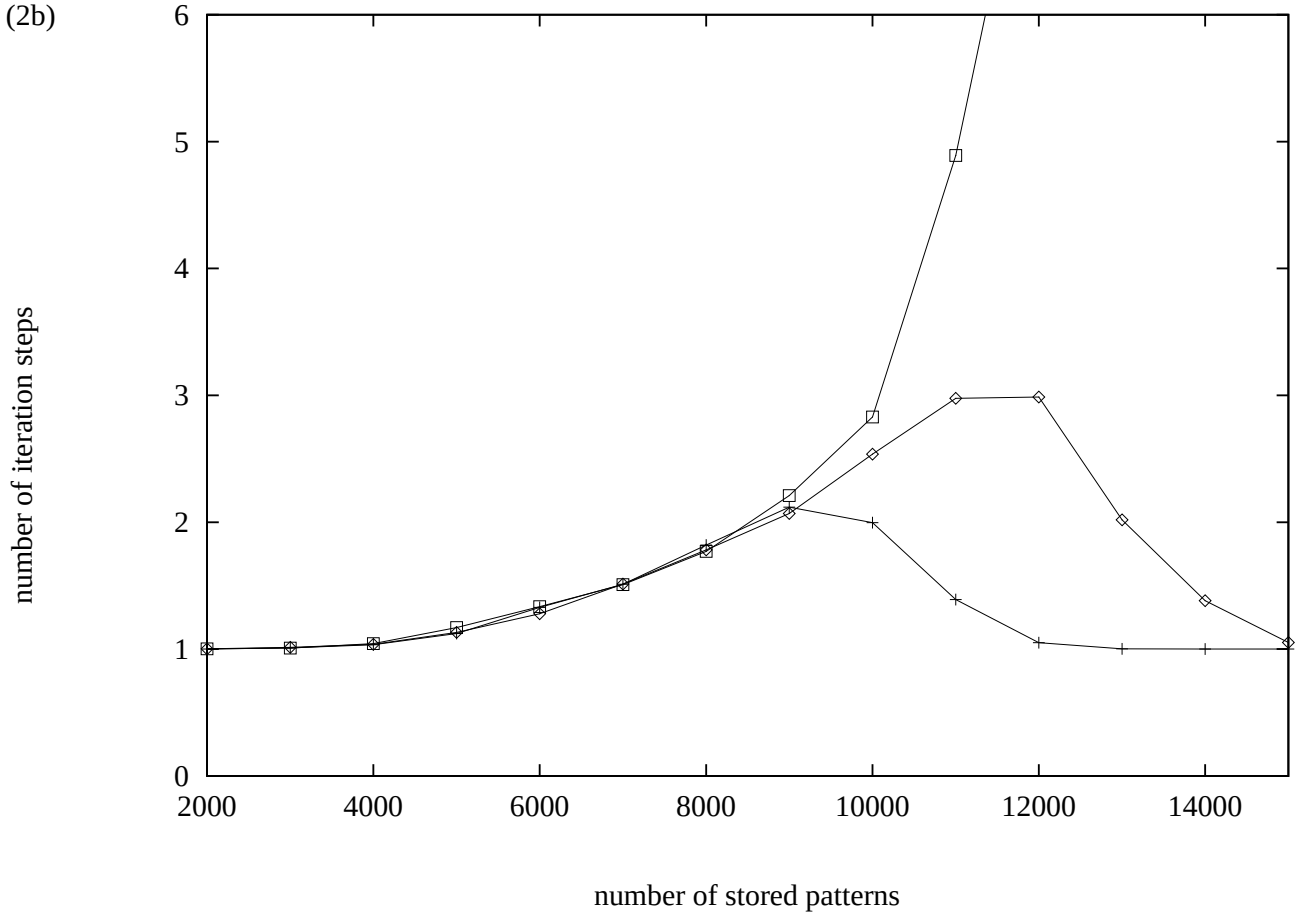
(1b)


5
4
3
2
1
0
number of iteration steps
2000
4000
6000
8000
10000
12000
14000
number of stored patterns

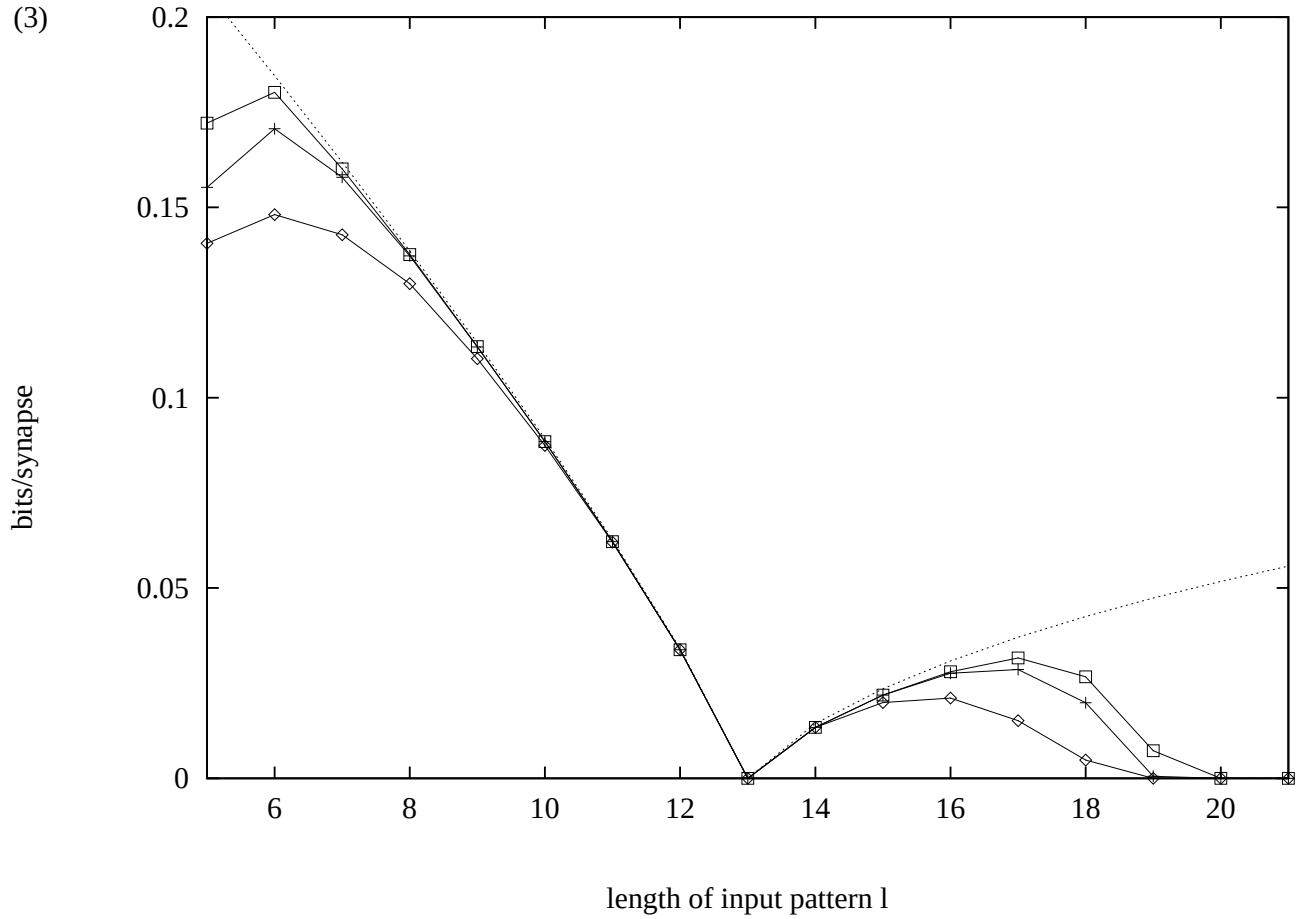
(2a)


0.2
0.15
0.1
0.05
0
bits/synapse
2000
4000
6000
8000
10000
12000
14000

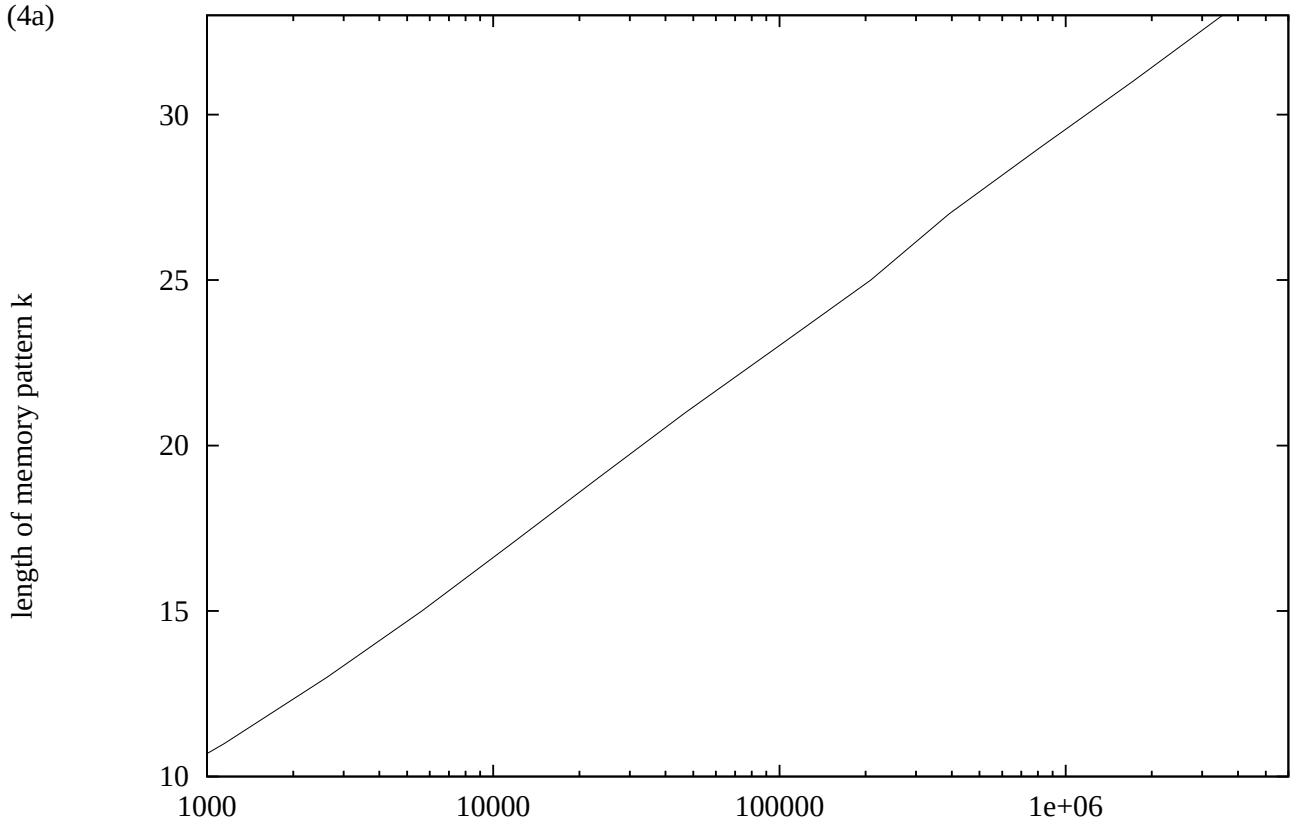
(2b)


6
5
4
3
2
1
0
number of iteration steps
2000
4000
6000
8000
10000
12000
14000
number of stored patterns

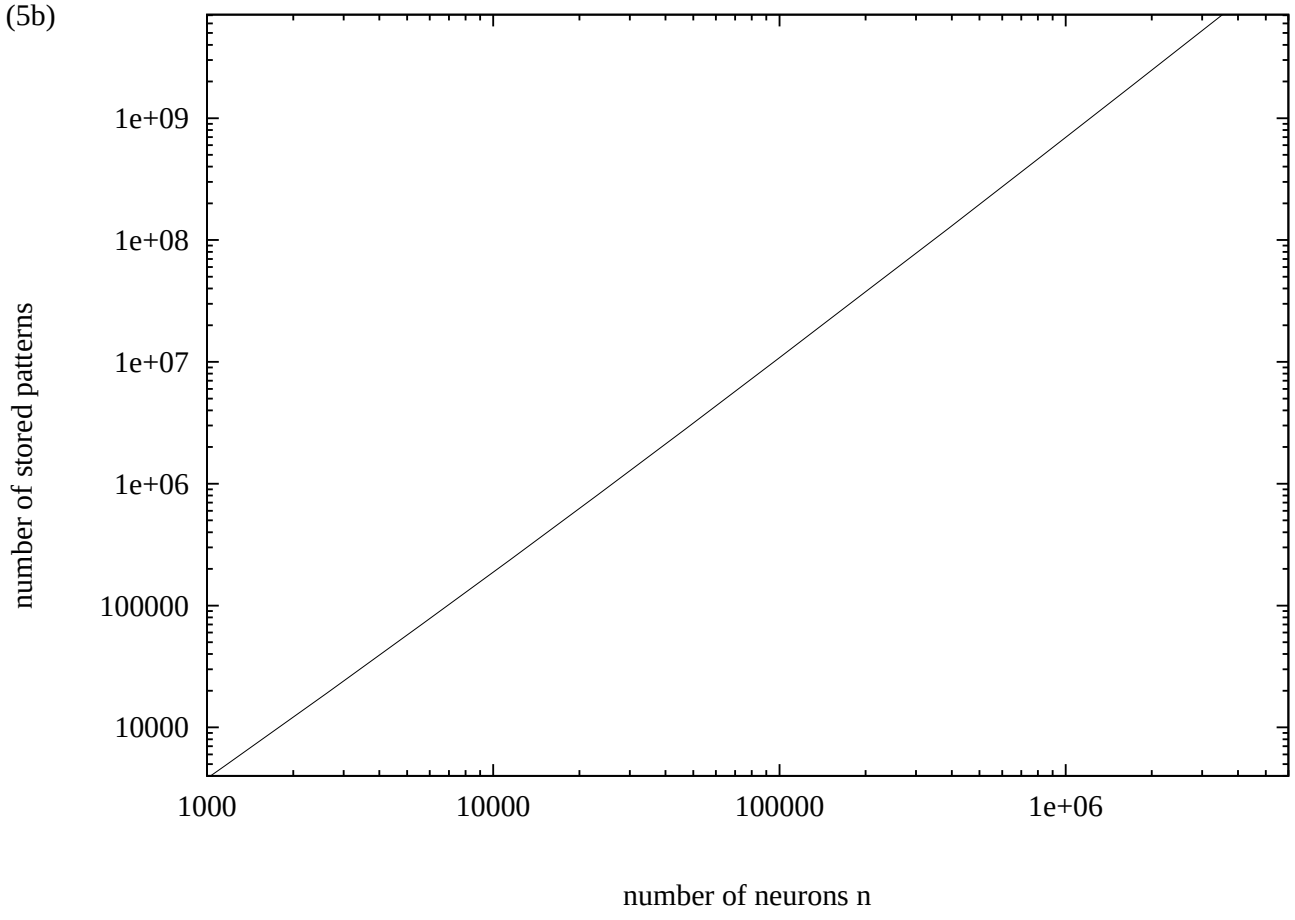
(3)


0.2
0.15
0.1
0.05
0
bits/synapse
6
8
10
12
14
16
18
20
length of input pattern l

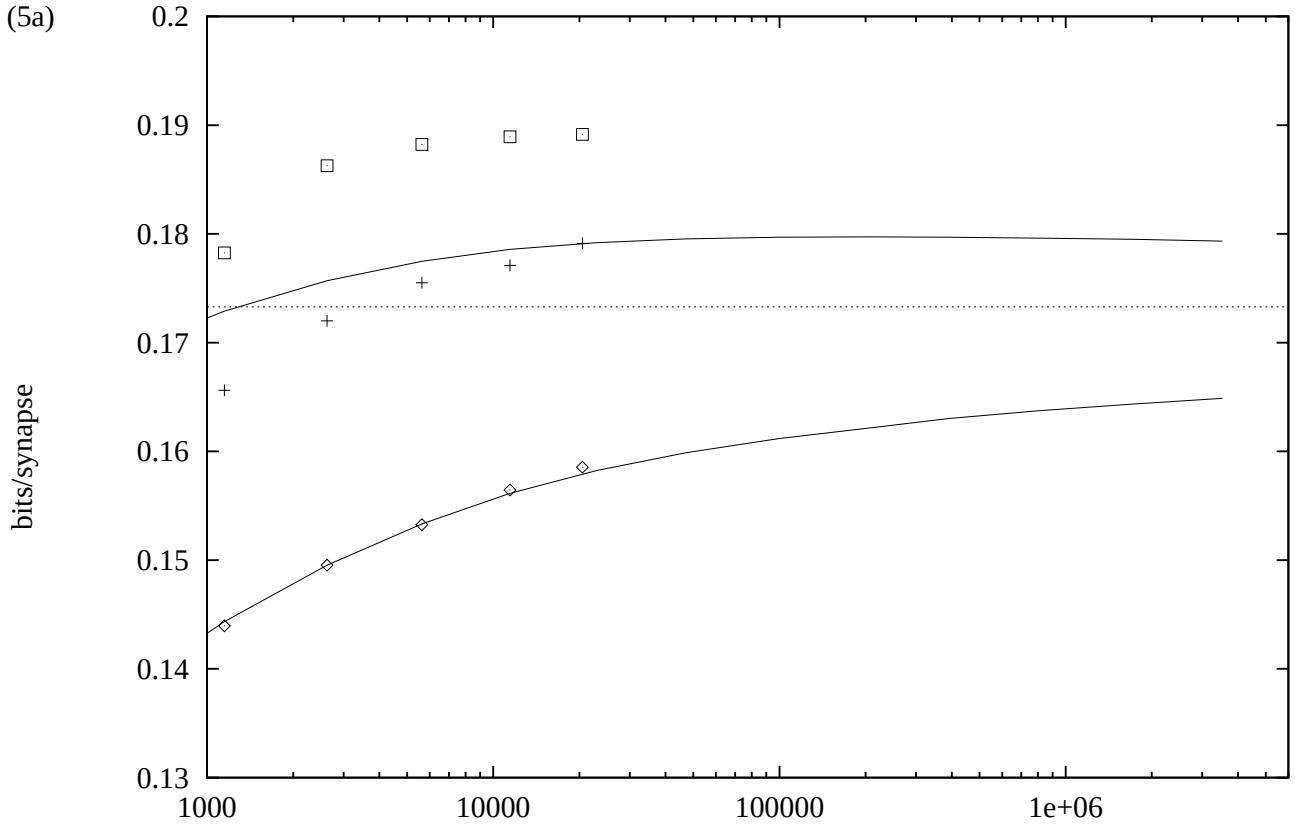
(4a)


length of memory pattern k
30
25
20
15
10
1000
10000
100000
1e+06

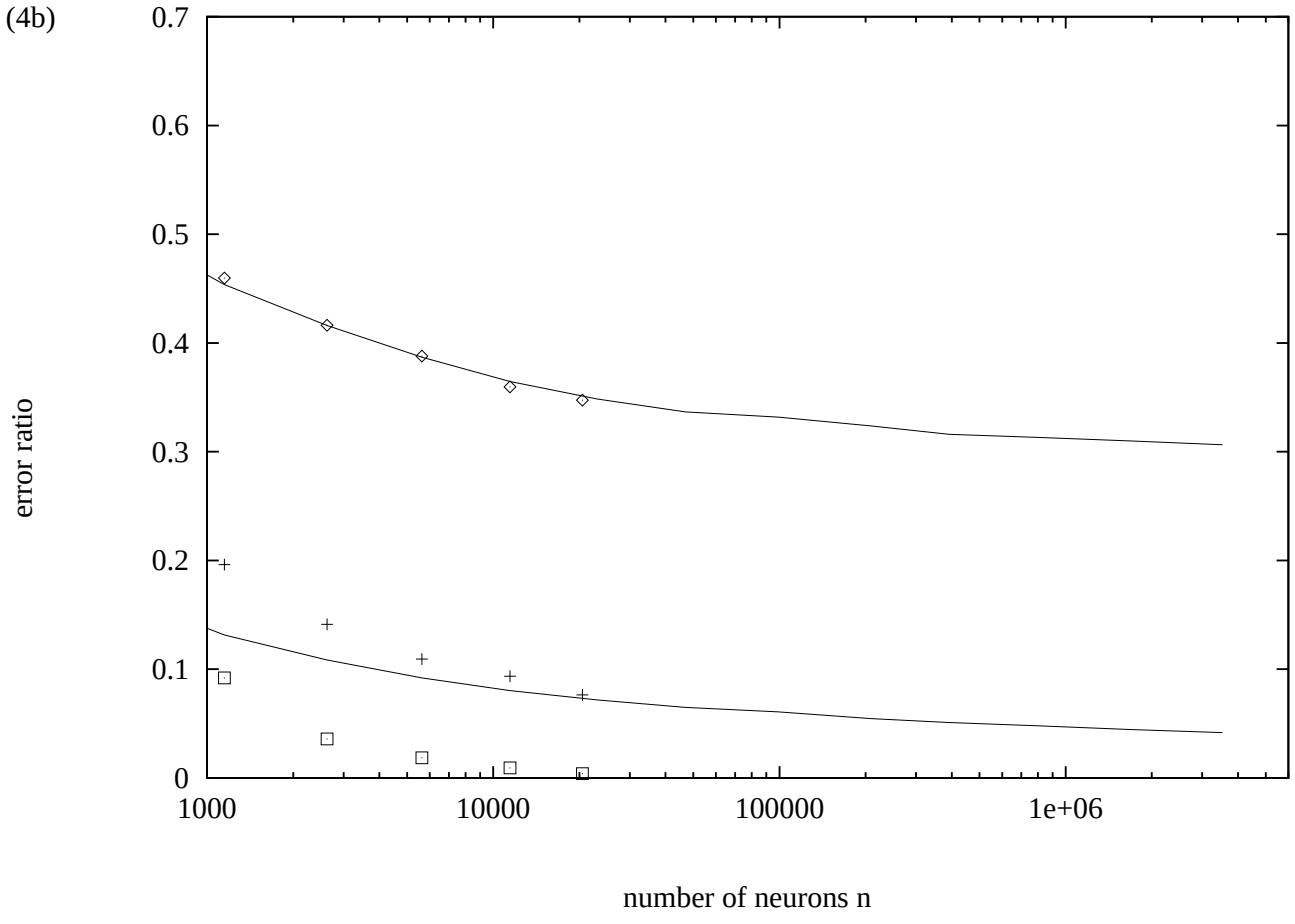
(5b)


number of stored patterns
1e+09
1e+08
1e+07
1e+06
100000
10000
1000
10000
100000
1e+06
number of neurons n

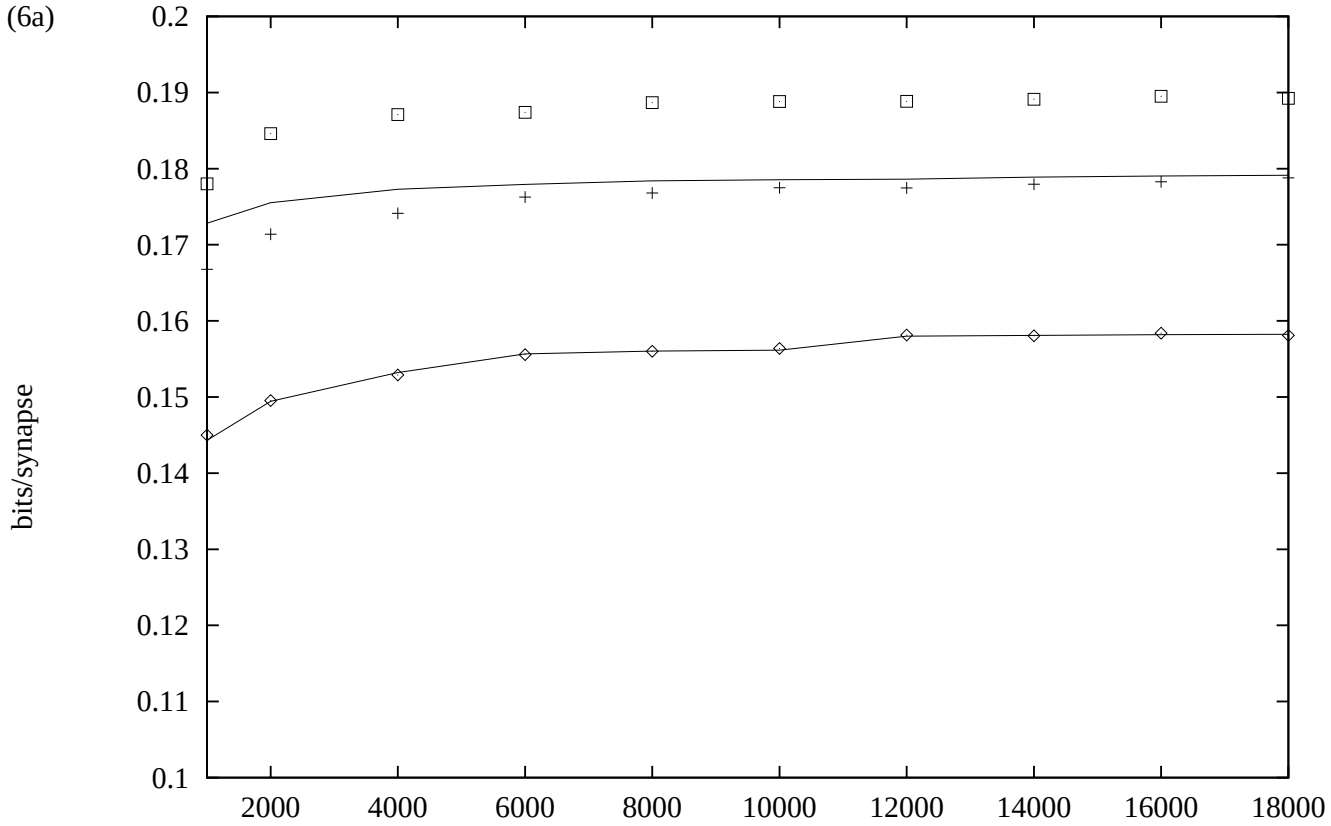
(5a)


0.2
0.19
0.18
0.17
0.16
0.15
0.14
0.13
bits/synapse
1000
10000
100000
1e+06

(4b)


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
error ratio
1000
10000
100000
1e+06
number of neurons n

(6a)


bits/synapse
0.2
0.19
0.18
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0.1
2000
4000
6000
8000
10000
12000
14000
16000
18000

(6b)

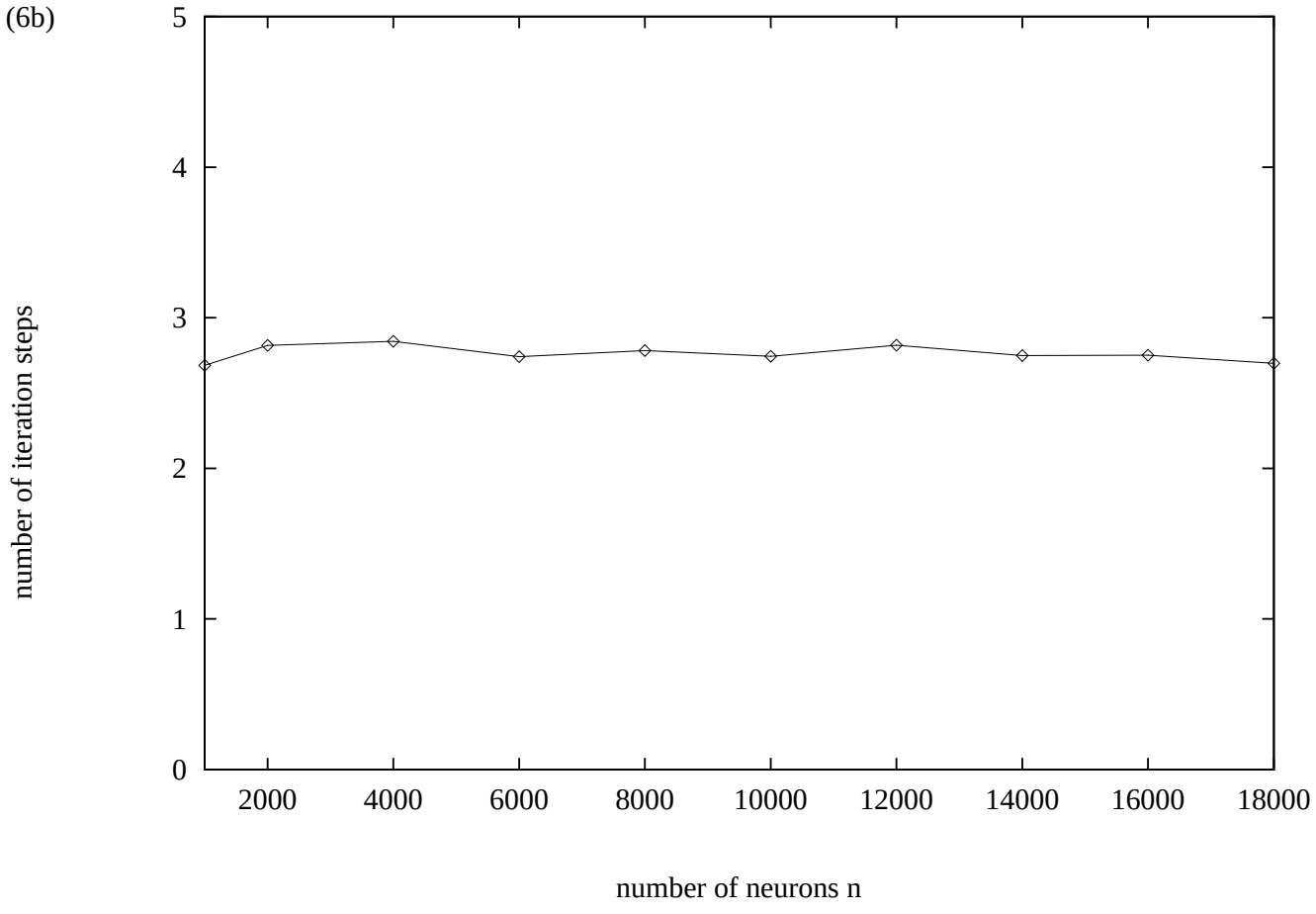

number of iteration steps
5
4
3
2
1
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
number of neurons n

Figure captions

Figure 1 (a): Capacity C of the additive (dashed line) and binary Hebbian learning rule (solid line). The results for one-step ($\diamond$), two-step ($+$) and iterative retrieval ($\Box$) using the threshold control strategy $lk+$ are shown. A memory with $n = 1900$ threshold neurons. Each learning pattern was sparsely coded with $k = 13$ ones, the retrieval started from an input pattern, which was a part of a memory pattern with $l = 6$ ones.
(b): Mean iteration time of the threshold control strategy $lk+$ of the additive (dashed line) and binary Hebbian learning rule (solid line) for the parameter setting $n = 1900$, $k = 13$ and $l = 6$. The effective storage range can be observed, $2000 \leq M \leq 7000$ for additive and $5000 \leq M \leq 14000$ for binary Hebbian learning.

Figure 2 (a): Capacity C achieved by iterative retrieval with the three different threshold control strategies ($CA = \Box$, $lk = +$ and $lk+ = \diamond$) for $n = 1900$ neurons, $k = 13$ ones per memory pattern and $l = 6$ ones per input pattern.
(b): The iteration times for the three threshold control strategies ($CA = \Box$, $lk = +$ and $lk+ = \diamond$).

Figure 3: Capacity C of one-step ($\diamond$), two-step ($+$), and iterative retrieval ($\Box$) with the threshold control strategy $lk+$ for different types of input pattern. Here $M = 10,000$ patterns each with $k = 13$ ones had been stored in the memory with $n = 1900$ neurons. The dashed line shows the upper bound representing errorfree retrieval.

Figure 4 (a): The number of ones k of the memory patterns corresponding to the optimal capacity C achieved by two-step retrieval and using threshold control strategy $lk+$.
(b): The optimal number of learning patterns M corresponding to the optimal capacity C with two-step retrieval and using threshold control strategy $lk+$.

Figure 5 (a): Optimal capacity values C for optimized parameters M, k and l. The curves are obtained from the calculated error probabilities of one-step and two-step retrieval. The simulation results are plotted for one-step ($\diamond$), two-step ($+$) and iterative retrieval ($\Box$). The asymptotic capacity value $C \approx 17.32\%$ for one-step, two-step, and iterative retrieval is shown by the dashed line. (b): The error ratio corresponding to the optimal capacity achieved by one-step ($\diamond$), two-step ($+$) and iterative retrieval ($\Box$).

Figure 6 (a): Optimal capacity values for different memories in the range from $n = 1000$ to $n = 18000$ threshold neurons (one-step ($\diamond$), two-step ($+$) and iterative retrieval ($\Box$)).
(b): Mean iteration time for the threshold control strategy $lk+$.