VŠB - Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Boolean Factor Analysis by Attractor Neural Network

PHD THESIS

Pavel Y. Polyakov

Ostrava, October 2016

Thesis supervisor:
Ing. Dušan Húsek, CSc.
Institute of Computer Science
The Czech Academy of Sciences
Pod Vodárenskou věží 271/2
182 07 Praha 8
dusan@cs.cas.cz

Educational institution
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava–Poruba
Czech Republic
http://fei.vsb.cz http://www.vsb.cz

Study program
P1807 Computer Science, Communication Technology and Applied Mathematics

Field of study
1801V001 - Informatics

Assignment

Boolean Factor Analysis by Attractor Neural Network
The aim of the dissertation is to create effective methods for nonlinear Boolean factor analysis of signals of different properties, using the paradigm of neural networks, especially auto-associative memory. Dissertation will link up to the previous research performed by the supervisor and his collaborators and their expertise in attractor neural networks. A generative data model of Boolean factor analysis should be defined by the doctoral student. Proceeding from this, doctoral student should define a measure that allows for a comparison of this method with those targeted on similar applications: e.g. a special clustering methods, feed forward neural methods and classical statistical methods. Research is targeted on the current issues, finding underlying data structure, including extraction of information from large data files, such as Internet content, for internal data structure revealing, data dimensionality reduction, data compression, etc.
In Ostrava, 13th September, 2010 (Revised 2016).

Zadání disertace

Booleovská faktorová analýza atraktorovou neuronovou sítí
Cílem disertace je vytvoření efektivní metody pro nelineární Booleovskou faktorovou analýzu signálů různých vlastností s využitím paradigmatu neuronových sítí, speciálně asociativní paměti. Práce naváže na předchozí výzkum školitele a jeho spolupracovníků v oblasti atraktorových neuronových sítí. Doktorand by měl definovat generativní model dat pro tuto úlohu. Na jeho základě by měl doktorand definovat univerzální míru umožňující srovnání s existujícím metodami řešícími podobné úlohy: např. speciální shlukovací metody, metody založené na dopředených neuronových sítích, ale i klasické statistické metody. Výzkum se zaměřuje na jeden z aktuálních problémů strojového učení, speciálně na zkoumání nových metod pro zjištění skryté struktury dat, včetně extrakce informací z velkých datových souborů, jako je např. internetový obsah, pro kompresi rozsáhlých souborů dat, atd.
V Ostravě 13. září, 2010 (Revised 2016).

Prohlášení studenta

„Prohlašuji, že jsem tuto disertační práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě …………….. ………….……………….
datum podpis autora

Student statement

"*I hereby declare that this PhD Thesis was written by myself. I have quoted all the references I have drawn upon.*"

In Ostrava ………………….. ………………………………….
date signature of the author

Abstrakt

Jednou z nejdůležitějších výzev současnosti, která stojí před komunitou badatelů z oblasti strojového učení je výzkum metod pro analýzu vysoce-dimenzionálních binárních dat s cílem odhalení jejich skryté struktury. V literatuře můžeme nalézt mnoho přístupů, které se snaží tuto doposud poněkud vágně definovanou úlohu řešit. Booleovská Faktorová Analýza (BFA), jež je předmětem této práce, předpokládá, že skrytou strukturu binárních dat lze reprezentovat jako booleovskou superpozici binárních faktorů tak, aby co nejlépe odpovídala generativnímu modelu signálů BFA a danému kritériu optimálnosti. Za těchto podmínek je BFA dobře definovaná úloha zcela analogická lineární faktorové analýze. Hlavní přínosy disertační práce, jsou následující: Za prvé byl vyvinut efektivní způsob BFA založený na původní atraktorové neuronové síti s rostoucí aktivitou (ANNIA), která byla následně zlepšena kombinací s metodou expectation–maximization (EM). Dále byly provedeny analýzy charakteristik ANNIA, které jsou důležité pro fungování metody. Funkčnost obou metod byla také ověřena na uměle vytvořených souborech dat pokrývajících celou škálu parametrů generativního modelu. Dále je v práci ukázáno použití metod na reálných datech z různých oblastí vědy s cílem prokázat jejich přínos pro tento typ analýzy. A konečně bylo provedeno i srovnání metod BFA se podobnými metodami včetně analýzy jejich použitelnosti.

Klíčová slova

Booleovská faktorová analýza, dolování z dat, neuronové sítě, statistika, redukce dimenze, atraktorová neuronová síť, Hopfieldova neuronová síť, Hebbovo učení, metoda maximální věrohodnosti, expectation-maximization, informační zisk

Abstract

Methods for the discovery of hidden structures of high-dimensional binary data rank among the most important challenges facing the community of machine learning researchers at present. There are many approaches in the literature that try to solve this hitherto rather ill-defined task. The Boolean factor analysis (BFA) studied in this work represents a hidden structure of binary data as Boolean superposition of binary factors complied with the BFA generative model of signals, and the criterion of optimality of BFA solution is given. In these terms, the BFA is a well-defined task completely analogous to linear factor analysis. The main contributions of the dissertation thesis are as follows: Firstly, an efficient BFA method, based on the original attractor neural network with increasing activity (ANNIA), which is subsequently improved through a combination with the expectation-maximization method, has been developed. Secondly, the characteristics of the ANNIA that are important for method functioning were analyzed. Then the functioning of both methods was validated on artificially generated data sets. Next, the method was applied to real-world data from different areas of science to demonstrate their contribution to this type of analysis. Finally, the BFA method was compared with related methods, including applicability analysis

Keywords

Boolean factor analysis, data mining, statistics, dimension reduction, attractor neural network, Hopfield neural network, Hebbian learning rule, information gain, dimension reduction, likelihood-maximization, expectation-maximization.

Acknowledgments

First of all, I would like to express my gratitude to my dissertation thesis supervisor, Ing. Dušan Húsek, CSc. for his leadership and professional guidance during my research. I would like to express thanks to Prof Alexander A. Frolov, head of laboratory at IHNA RAS, my supervisor specialist in Moscow, with whom we three worked shoulder to shoulder on this topic. My thanks are also due to others who helped me connect my work to other scientific areas, namely to Prof. Ing. Hana Řezanková, CSc. who is an outstanding statistician. The same applies to the lecturers at the VSB TU Ostrava, among them mainly Professor Ivo Vondrák, Ph.D, the Rector of the university, and to the Dean of FEI, Professor Václav Snášel, CSc, who filled us with optimism and energy to persist at this work. Also to all the laboratory personnel I visited for their valuable comments and help in solving numerous problems. Special thanks go to the Department of Computer Science management for giving me the possibility to study in the Czech Republic.

Finally, my greatest thanks go to my wife because of her infinite patience and love, as well to my parents who have supported me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Factor analysis is one of the most effective methods of elucidation and removal of information redundancy of multidimensional signals. The essence of the idea of information redundancy is intuitively clear: during everyday life human, like all living beings, deals with regular in space and time information structure. Due to this regularity, some characteristics of the environment, which are referred to as features, are encountered concurrently, i.e., the probability of their concurrent observation is considerably higher than the probability of random observation. The group of concurrent features is called as a factor. Encoding of signals using factors requires smaller volume of information and this means that the signals that arrive to the brain carry redundant information.

The idea of Boolean factor analysis (BFA) follows from the studies by Barlow and Marr, the outstanding neurophysiologists of the last century. Marr believed that detection and elimination of information redundancy is inherent function of the brain because it gives two substantial evolutionary advantages. First, re-encoding of signals in terms of factors helps to save brain resources: "Excessive information may be effectively saved if some composition of coherent features is extracted from the incoming signal and added to lexicon of the brain experience as a new 'word' (entity). This lexicon is used by the brain for interpretation and recording of its experience . . . " [73]. By replacing of several features with one factor, we get a more compact encoding system and decrease resources spent on the recording of signal into memory. Second, we get an opportunity to operate not only with directly observed but also with other (implicit) valuable features. Marr's fundamental hypothesis was that if certain combinations of evident characteristics (traits, features) have a trend to be met coherently (at least, a number of them), it is possible that there are other implicit valuable characteristics that are associated with this combination [72]. These two statements exactly correspond to formal definition of factor analysis that has two goals: a decrease in the number of variables used for description of the data and recognition of latent (implicit) variables.

According to Barlow [9], as Foldiak [20] said, "objects (and also features, concepts or anything that deserves a name) are collections of highly correlated properties. For instance, the properties 'furry', 'shorter than a metre', 'has a tail', 'moves', 'animal', 'barks', etc. are highly correlated, i.e., the combination of these properties is much more frequent than it would be if they were independent (the probability of the conjunction is higher than the product of individual probabilities of the component features). It is these non-independent, redundant features, the 'suspicious coincidences' that define objects, features, concepts, categories, and these are what we should be detecting. While components of objects can be highly correlated, objects are relatively independent of one another... The goal of the sensory system might be to detect these redundant features and to form a representation in which these redundancies are reduced and the independent features and objects are represented explicitly". Obviously, objects are factors and the scene observed is incoming signal that is decomposed into factors. Both objects and scene are presented in binary code, therefore, operations with them may be performed in the frameworks of Boolean logics, i.e., use not linear but Boolean factor analysis.

Binary presentation of data is quite typical of many fields including sociology, marketing, zoology, genetics, and medicine [17]. However, in contrast, to linear presentation, Boolean factor analysis is very weakly developed. Those who need to use factor analysis for binary data have several possibilities. If data have small dimension, it is possible to find binary factors using brute force search. It is clear that in this case the problem is NP-complete and its solution within a reasonable time may be performed only in case of a small number of factors (< 50). Approximate methods are required for analysis of binary data of greater dimension. The more widespread approach is linear factor analysis followed by binarization of solution [10]. Of course, like all methods based on linear algebra, it uses completely different model of signals, hence, it may be used cautiously and in very narrow range of parameters when non-linearity of the initial data is small (for example, when one signal has only one factor or factors do not intersect).

A good alternative to linear approach is method of Boolean matrix factorization [12, 86]. Its advantage over linear methods is that initial data are decomposed into factors according to Boolean arithmetics. However, factor analysis could not be reduced to formal decomposition of data matrix. The most substantial limitation of this approach is that it cannot be used for analysis of noisy data.

Recently, several methods have been proposed for BFA including neural network based method of Spratling [87]. Unfortunately, efficacy of these methods for solution of BFA was not examined on data of large dimension when their use would be really practicable. The above suggests that, despite importance of task, there no good theoretically based BFA methods that can work with the data of large dimension.

Boolean matrix factorization and Boolean factor analysis are frequently confused although they are different tasks like linear matrix factorization and linear factor analysis. Presumably, this confusion appeared because formal definition to Boolean factor analysis is not widely known yet. In the articles on the BFA, it is formulated using examples of practical tasks that authors are going to solve [20, 74]. Or they are limited by general equation of decomposition of initial data into factors which valid for both Boolean factor analysis and Boolean matrix factorization [12, 86]. To avoid here any ambiguity, in Chapter 2 the formal definition of BFA based on the information criteria of optimality of solution is given.

Thus, on one hand, there no good methods for BFA; on the other hand, there are strong arguments that the brain have to somehow separate factors. In this situation it is reasonable to address neurophysiology and look for the brain structures that would be able to have this function. According to widespread two-stage model of learning, new information that comes to the brain is initially kept in temporal storage and, then, interpreted, processed, and filtered information is directed to the neocortex for long-term storage. Numerous experimental results led to conclusion that the hippocampus, a phylogenetically ancient brain structure, plays a key role in consolidation, processing and storage of information. Due to an extensive system of recurrent collaterals, which provide self-excitation of group of neurons of one layer, the CA3 field can keep associatively linked patterns of information which allows the hippocampus to function as temporal storage of incoming information. According to two-stage model of learning, the desired procedure of separation of factors should be located somewhere between recording of information into temporal storage and its transfer to the long-term memory. The first candidate for the structure that is responsible for separation of factors is CA3 field.

Studies of Hopfield network, whose natural prototype is CA3 field, led to conclusion that its capacities are beyond the frameworks of autoassociative memory. For example, if a set of patterns that are close to each other is recorded into the Hopfield network, a pattern-prototype will appear which is somewhat average of these patterns [26]. It appears that associative neural network, a Hopfield network with sparse coding, can reveal factors [26]. This work is focused on study of this question.

Chapter 2

Formal definition of Boolean factor analysis

2.1 Decomposition of signal in linear and Boolean factor analysis

Factor analysis is one of the most efficient methods to reveal the structure underlying the data (the term factor analysis was first introduced by Thurstone [90]). This analysis presumes that statistical correlations in the data analyzed are accounted for by some hidden causes that are referred to as factors. It is necessary to reveal all hidden factors and determine how they are present in the data. Linear factor analysis implies that each observed signal $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ of N real-valued components can be presented as a linear superposition of L factors:

$$\mathbf{x} = \sum_{i=1}^{L} s_i \mathbf{f}_i + \mathbf{u}, \tag{2.1}$$

where factor score s_i gives a contribution of ith factor to the signal, the vector of factor loadings $\mathbf{f}_i = [f_{i1}, f_{i2}, \ldots, f_{iN}]$ represents the ith factor in the original N-dimensional signal space and $\mathbf{u} = [u_1, u_2, \ldots, u_N]$ designates the row vector of residual part of the signal, namely the part of signal that is not described by factors. In matrix form, the decomposition of the data matrix $\mathbf{X}$ of dimension $M \times N$ (where M is the number of observations) into matrices of factor scores $\mathbf{S}$ of dimension $M \times L$ and factor loadings $\mathbf{F}$ of dimension $L \times N$ is given by

$$\mathbf{X} = \mathbf{SF} + \mathbf{U}, \tag{2.2}$$

where $\mathbf{U}$ designates the matrix of residuals. The purpose of factor analysis is finding of the smallest number of factors that can describe initial data with the maximum accuracy. The decomposition $\mathbf{X} \simeq \mathbf{SF}$ may be interpreted as mapping of initial N-dimensional space of observed features into the space of smaller dimension L where signals are described by matrix of factor scores $\mathbf{S}$. The matrix of factor loadings $\mathbf{F}$ gives a way of transformation of one space into the other. If $L < N$, we have more compact representation of signals.

Boolean factor analysis (BFA) is used to find hidden relationships among binary data; the factor scores and factor loadings are also binary and all arithmetic operations are performed according to the Boolean logics. Operator OR ($\vee$) is used in BFA instead of summation in linear factor analysis, operator AND ($\wedge$) is used instead of multiplication. The equation of representation of binary signal $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ has similar form

$$\mathbf{x} = [\bigvee_{i=1}^{L} s_i \wedge \mathbf{f}_i] \vee \mathbf{u}. \tag{2.3}$$

Here, instead of linear sum $\sum$, we used Boolean sum $\bigvee$: $\bigvee\limits_{i=1}^{L} x_i = x_1 \vee x_2 \vee \cdots \vee x_L$. In matrix form, binary matrix $\mathbf{X}$ of dimension $M \times N$ decomposes into binary matrix of factor scores $\mathbf{S}$ of dimension $M \times L$ and binary matrix of factor loadings $\mathbf{F}$ of dimension $L \times N$:

$$\mathbf{X} = \mathbf{S} \odot \mathbf{F} \oplus \mathbf{U}, \tag{2.4}$$

where $\mathbf{U}$ is binary matrix of residuals, $\odot$ denotes Boolean matrix multiplication and $\oplus$ denotes Boolean matrix summation. The operations of Boolean matrix multiplication and Boolean matrix summation are similar to matrix multiplication and matrix summation for linear algebra, except that bitwise AND and bitwise OR are used instead of common componentwise product and componentwise sum.

2.2 Statistical model of binary signals

The definition of Boolean factor analysis in the decomposition form 2.3 or 2.4 is too general. This may result in confusion with other tasks, for example, with Boolean matrix factorization described in 3.1. On the other hand, the factor analysis is always based on assumption on statistical model of the data analyzed [10]. On the basis of generative model for linear factor analysis [47], it is possible to define analogous model for binary signals. According to the model, every component of signal $\mathbf{x}$ takes 1 or 0, depending on the presence or absence of the related attribute (observed feature). Each factor can be represented by a binary raw vector of factor loadings $\mathbf{f}_i = [f_{i1}, f_{i2}, \ldots, f_{iN}]$ where one valued entries correspond to highly correlated attributes of the ith object and zero valued entries correspond to attributes not constituting the object. Although the probability of the object's attributes appearing in a pattern simultaneously with its other attributes is high, it is not necessarily equal to 1. For example, the attribute "has a tail" does not always appear with the appearance of the object "dog". Let us denote this probability by p_{ij}, where j is the index of the attribute and i is the index of the factor. For attributes constituting the factor, i.e., for attributes with $f_{ij} = 1$, the probability p_{ij} is high, and for the other attributes (with $f_{ij} = 0$), it is zero.

As in linear factor analysis, let us suppose that in addition to common factors $\mathbf{f}_i$ that influence more than one attribute, each signal also contains N specific or unique factors that influence only particular attributes. Specific factors are also called "specific noise." The contribution of specific factors is defined by a binary row vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_N]$ of dimension N. Each specific factor η_j is characterized by the probability q_j with which η_j takes on the value 1. In the same way, each common factor $\mathbf{f}_i$ is characterized by the probability π_i that it appears in a signal, i.e., factor score s_i takes 1 with probability π_i. The probabilities p_{ij}, q_j and π_i are parameters of BFA generative model of signals: $\boldsymbol{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \ldots, L, j = 1, \ldots, N)$.

As a result, instead of using equation 2.3, any observed signal $\mathbf{x}$ can be presented in the form

$$\mathbf{x} = [\bigvee_{i=1}^{L} s_i \wedge \mathbf{f}'_i] \vee \boldsymbol{\eta} \tag{2.5}$$

where $\mathbf{s} = [s_1, s_2, \ldots, s_L]$ is a binary row vector of factor scores of dimension L, L being the total number of factors, $\mathbf{f}'_i = [f'_{i1}, f'_{i2}, \ldots, f'_{iN}]$ is a distorted version of factor $\mathbf{f}_i$ and $\boldsymbol{\eta}$ is a binary row vector of specific factors. Factor distortion implies that one valued entries of $\mathbf{f}_i$ can transform to 0 with probability $1 - p_{ij}$ before mixing in the observed pattern but none of the zero valued entries $\mathbf{f}_i$ can take 1 in the distorted version of the factor because the probability for them to transform to 1 is zero ($p_{ij} = 0$).

It is supposed that each component of the common factor is distorted independently of the presence of other factors in the pattern and independently of specific noise. Thus, the probability of the jth component of $\mathbf{x}$ to take the value x_j is

$$\mathcal{P}(x_j|\mathbf{s}, \boldsymbol{\Theta}) = x_j - (2x_j - 1)(1 - q_j) \prod_{i=1}^{L} (1 - p_{ij})^{s_i}, \tag{2.6}$$

where scores s_i are assumed to be given. Different components of $\mathbf{x}$ (attributes) are also supposed to be statistically independent, and thus

$$\mathcal{P}(\mathbf{x}|\mathbf{s}, \boldsymbol{\Theta}) = \prod_{j=1}^{N} \mathcal{P}(x_j|\mathbf{s}, \boldsymbol{\Theta}). \tag{2.7}$$

BFA is performed on the set $\mathcal{X}$ of patterns $\mathbf{x}_m$ containing M representatives. Instantiation of vectors $\mathbf{f}'_i$, $i = 1, \ldots, L$ and $\boldsymbol{\eta}$ for some pattern $\mathbf{x}_m$ is assumed to be independent of other patterns, and thus

$$\mathcal{P}(\mathcal{X}) = \prod_{m=1}^{M} \mathcal{P}(\mathbf{x}_m). \tag{2.8}$$

In most cases, it is fair to assume that factors appear in patterns independently of each others, then

$$\mathcal{P}(\mathbf{s}|\mathbf{\Theta}) = \prod_{i=1,L} \pi_i^{s_i}(1-\pi_i)^{1-s_i}. \tag{2.9}$$

The aim of Boolean factor analysis is to find the parameters of the generative model $\mathbf{\Theta}$ and factor scores $\mathbf{s}_m (m = 1, \ldots, M)$ for all M patterns $\mathbf{x}_m$ of the observed dataset. However, it is supposed that the factors found could also be detected in any arbitrary pattern if generated by the same model. Note that the finding of p_{ij} implies the finding of factor loadings f_{ij} since $f_{ij} = \text{sgn}(p_{ij})$.

2.3 Criterion of solution of Boolean factor analysis

According to the definition of factor analysis, it is necessary to simultaneously minimize both number of factors L and number of non-zero components in the matrix of residuals $|\mathbf{U}|$ (see equations 2.2 and 2.4). However, in the general case, these aims are opposite: a decrease in the number of factors results in an increase in $|\mathbf{U}|$ and, vice versa, at a relatively large number of factors the data matrix could be completely decomposed into factors. Is it possible to reconcile these two requirements and use a common criterion for evaluation of quality of factor analysis? Encoding of signals with the use of factors decreases information redundancy of signals and helps to use smaller volume of information for their storage. It is easy to show that elimination of information redundancy of signals also occurs during minimization of the number of factors at fixed $|\mathbf{U}|$ and during minimization of the number of non-zero components in the residue matrix $|\mathbf{U}|$ at fixed L. This means that the condition of elimination of information redundancy includes each of the above requirements. Note that principle of optimal encoding which eliminates information redundancy serves as a basis for objective strategy of brain neurons [8].

Information gain is a general information theoretic measure that could be used for estimation of successfulness of information redundancy elimination. Since generative model of binary signals is specified in Section 2.2, information gain for BFA can be calculated as the difference of two entropies. The first is the entropy of a dataset when its hidden factor structure is unknown, and the second is the entropy when it is revealed and taken into account. If the factor structure of the signal space is unknown, then representing the jth component of vector

$\mathbf{x}$ requires $h(p_j)$ bits of information, where $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the Shannon function and p_j is the probability of the jth component's taking 1. Representing the whole dataset requires

$$H_0 = M\sum_{j=1}^{N} h(p_j) \tag{2.10}$$

bits of information. If the hidden factor structure of the signal space is detected, i.e., the parameters of the generative model $\mathbf{\Theta}$, factor scores $\mathbf{s}_m(m = 1,\ldots,M)$ and factor loadings $\mathbf{f}_i(i = 1,\ldots,L)$ are found, then representing the whole dataset requires

$$H = H_1 + H_2 + H_3 \tag{2.11}$$

bits of information. Here

$$H_1 = M\sum_{i=1}^{L} h(\pi_i) \tag{2.12}$$

is the information required to represent the factor scores,

$$H_2 = L\sum_{j=1}^{N} h(r_j) \tag{2.13}$$

is the information required to represent the factor loadings,

$$H_3 = \sum_{m=1}^{M}\sum_{j=1}^{N} h(\mathcal{P}(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})) \tag{2.14}$$

is the information required to represent all patterns of the dataset when factor scores and loadings are given, r_j is the probability of the j-th component's taking 1 in factor loadings, $\mathcal{P}(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})$ is given by (2.6). The information gain is determined by the difference between H_0 and H. Relative information gain defined as

$$G = (H_0 - H)/H_0 \tag{2.15}$$

is used as criterion of solution of BFA.

The magnitude G may vary in the range from $-\infty$ to 1. The positive G means that the encoding of signals with the use of factors is more effective as compared to the initial encoding. If G is close to zero or negative then either the selection of factors was wrong or factor analysis in the frameworks of the postulated generative model is senseless because this model is inadequate to the internal structure of the data analyzed.

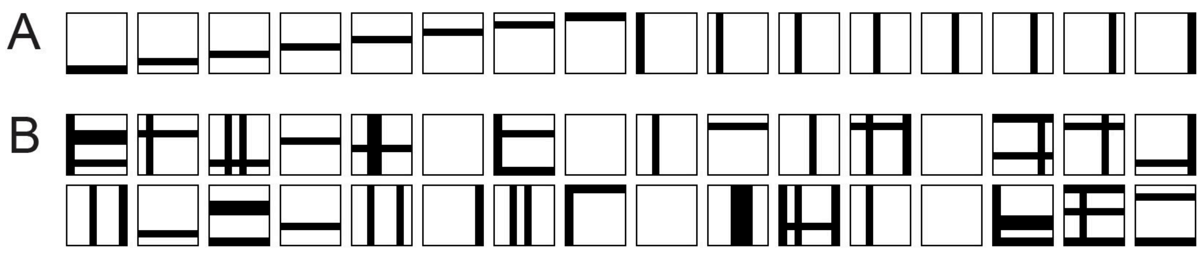


FIGURE 2.1: **A** Sixteen vertical and horizontal bars in 8-by-8 pixel images. **B** Examples of images in the standard bars problem. Each image contains two bars on average.

As shown in [35, 39, 45], the relative information gain reaches maximum when BFA solution is right, i.e., all scores and generative model parameters used for artificially generated dataset are found correctly. It decreases when both the difference between the resulting BFA solution and the right solution increases and noise in data increases. Thus, information gain has proved to be a reliable measure for detecting the presence of hidden factor structures in a given dataset and for comparing efficiency of different BFA methods as well.

In many methods appropriate for BFA the probabilities p_{ij} and q_j are not computed explicitly, they give only factor scores. In this case, for computing the information gain G, these probabilities can be estimated by maximizing the dataset likelihood for the proposed generative model that takes the form

$$\Lambda = \sum_{m=1}^{M} \Lambda_m, \tag{2.16}$$

where

$$\Lambda_m = \log[P(\mathbf{s}_m|\mathbf{\Theta})P(\mathbf{x}_m|\mathbf{s}_m, \mathbf{\Theta})], \tag{2.17}$$

$P(\mathbf{x}_m|\mathbf{s}_m, \mathbf{\Theta})$ and $P(\mathbf{s}_m|\mathbf{\Theta})$ are given by (2.7) and (2.9), correspondingly.

2.4 Bars problem — illustrative example of Boolean factor analysis

The well-known benchmark for learning of objects from complex patterns is the Bars Problem (BP) introduced by Foldiak [20]. The BP in various modifications has been considered in many papers (see Lucke and Sahani [71], for references). In this problem, each pattern of the dataset is n-by-n binary pixel image containing several of $L = 2n$ possible (one-pixel wide) horizontal and vertical bars (Fig. 2.1).

Pixels constituting a bar take the value 1 and pixels not constituting it take the value 0. For each image, each bar could be chosen with the probability C/L, where C is the mean number of bars mixed in an image. At the point of intersection of a vertical and a horizontal bar, the pixel takes the value 1. This Boolean summation of pixels belonging to different bars simulates the occlusion of objects. The task is to recognize all bars as individual objects, exploring a dataset containing M images consisting of bar mixtures. In most papers where the BP was used as benchmark, C was set to 2 and $n \leq 8$.

In terms of BFA, bars are factors. Factor loadings f_{ij} ($j = 1, \ldots, N$) take value 1 for pixels constituting the i^{th} bar and value 0 for pixels not constituting it. Each image is a Boolean superposition of factors, and the factor score takes the value 1 or 0 depending on the presence or absence of a bar in the image. Thus, the bars problem is a special case of BFA.

Chapter 3

Related works

BFA is NP-hard problem that was shown, for example, for Boolean matrix factorization [14, 75], the special case of BFA. Therefore the precise solution given by brute force search is unattainable for most real-world problems due to computational reasoning. Although there are strategies for filtering possible solutions of BFA to reduce computational complexity [65], they increase applicability of brute force search not cardinally [65].

Other approaches for solving BFA are approximate. They could be classified in the next groups. The first group contains neural network based methods [20, 46, 88] inspired by investigations of properties of natural neural networks. These methods propose some neural network architecture and algorithm that is able to simulate the mechanism of new feature detection inherent to brain functioning. The second group is matrix factorization methods that factorize binary matrices into binary or real-valued components under different assumptions [11, 13, 14, 64, 75]. Methods of the third group are based on some generation model of signal space [41, 71]. An appropriate likelihood function is defined for postulated generation model and solution is searched by maximizing likelihood function. BFA is also close to fuzzy and multi-assignment clusterings [22, 50] when appropriate condition of simultaneous assignment of object to several clusters is chosen [33]. The detailed description of these methods is presented below.

3.1 Boolean matrix factorization methods

Boolean matrix factorization implies representation of $M \times N$ binary data matrix $\mathbf{X}$ as a Boolean product of two binary factor matrices $\mathbf{A}$ of dimension $M \times L$ and $\mathbf{B}$ of dimension $L \times N$ with L as small as possible. In matrix presentation $\mathbf{X} = \mathbf{A} \odot \mathbf{B}$. In component-wise presentation

$$x_{mn} = \bigvee_{l=1}^{L} a_{ml} \cdot b_{ln}, \tag{3.1}$$

where $\bigvee$ denotes maximum (truth function of logical disjunction) and $\cdot$ is the usual product (truth function of logical conjunction).

When analysing non-artificial data (solving real-world problem), matrix $\mathbf{X}$ is usually large and may be noisy. In this case it is particularly appealing to look for approximate decompositions of $\mathbf{X}$. That is, given data matrix $\mathbf{X}$ and positive integer K, find an $M \times K$ binary matrix $\mathbf{A}'$ and $K \times N$ binary matrix $\mathbf{B}'$ that minimize

$$|\mathbf{X} - \mathbf{A}' \odot \mathbf{B}'| = \sum_{n=1}^{N} \sum_{m=1}^{M} |x_{mn} - (\mathbf{A}' \odot \mathbf{B}')_{mn}|.$$

In BFA interpretation, matrix $\mathbf{A}'$ is considered as matrix of factor scores $\mathbf{S}$ in (2.4), and parameters of the generative model $\mathbf{\Theta}$ can be estimated using likelihood maximization procedure: by the M-step of the likelihood maximization procedure presented in Section 4.2 in which likelihood Λ defined by (2.16) is gradually increased with respect to parameters of the model while holding factor scores s_{mi} fixed. The discussion about the difference between BFA and BMF approaches can be found in [25].

In the case of noisy data, the information gain G defined by (2.15) can reach a maximum value when $K < L$. The optimal value of parameter K could be found in two ways: by computing G for all possible K and choosing the value that maximize G, or by increasing K step by step while G is increasing. The first way is used in this work for estimating the performance of BMF in solving BFA.

3.1.1 BMF using formal concept analysis

Formal concept analysis (FCA) is based on a traditional understanding of concepts by which a concept is determined by its extent and its intent. The extent of a concept (e.g., "dog") is the collection of all objects covered by the concept (the collection of all dogs), while the intent is the collection of all attributes (e.g., "to bark", "to be a mammal") covered by the concept. The output of FCA consists of a hierarchically ordered collection formal concepts. Formal concept is defined as a pair (C, D) of subsets of objects $\mathcal{C}$ and attributes $\mathcal{D}$ that: 1. every object in C has every attribute in D; 2. for every object in $\mathcal{C}$ that is not in C, there is an attribute in D that the object does not have; 3. for every attribute in $\mathcal{D}$ that is not in D, there is an object in C that does not have that attribute. Efficient algorithms for computing formal concepts exist (see [64] for overview).

The main assumption of the method is that formal concepts are optimal factors, and so required binary factor matrices **A** and **B** are consist of subset of formal concepts [13]. There are various heuristics for optimal choosing of such subset: filtering concepts with too many attributes or objects (constraints on factor matrices sparseness) [66], excluding similar objects or attribute in data matrix **X** [66], finding mandatory factor concepts at first [12], using greedy approximation algorithm that step by step selects the concept that covers the largest number of entries with 1 in **X** that were not covered by the previously selected concepts [13].

The computational complexity is weak point of the approach. In spite of algorithms for computing formal concepts are rather fast, the number of produced formal concepts is too large when processing real data. For example, factorization of 8124×119 data matrix from "Mushroom" database [7] was performed in 18 min. and required 97 MB memory [11] while competitive greedy algorithm suggested by the same authors (see Section 3.1.2) was performed in 13 sec. and required 2 MB memory.

3.1.2 Greedy algorithm that maximize coverage of data matrix by factors

The algorithm was developed by Belohlavek and Vychodil [11, 14]. It avoids the necessity to compute all the concepts of **X**. Instead, it computes the candidate factors, i.e. concepts of **X**, on demand by the following greedy procedure. Each time a new factor is needed, one looks at the columns of **X** and selects the concept generated by a column which covers most of the yet uncovered 1s in **X**. Such a concept corresponds to a narrow but high rectangle in the data. Then one tries to see if such a rectangle may be extended to a wider rectangle by adding some attribute and deleting the objects so that one still has a rectangle. If so, one selects the best such a rectangle, i.e. the rectangle covering most of the yet uncovered 1s in **X**. One repeats the process of extension until no such extension yields a better rectangle. This way one obtains the new factor. The procedure is repeating until data matrix is covered completely or number of factors reaches the predefined value K for approximate decomposition.

The algorithm imposes constrains on matrices $\mathbf{A}'$ and $\mathbf{B}'$ in the case of approximate decomposition: the difference $x_{mn} - (\mathbf{A}' \odot \mathbf{B}')_{mn}$ is not negative for all components of matrices.

Greedy algorithm of computing candidate factors on demand gives slightly less accurate results in comparison with browsing through all candidates for factors obtained by FCA. For example, when these methods are applied to "Mushroom" database [7], first 4 factors cover about 38% of data matrix in the case of greedy algorithm versus about 41% in the case of browsing through all formal concepts [11]. At the same time, the first method outperforms the second in both the memory consumption and the time needed to perform factorization: 2 MB RAM versus 97 MB RAM, and about 13 sec. versus about 18 min. [11].

3.1.3 Greedy algorithm using association rules

The basic idea of the algorithm is to exploit the correlations between the columns of data matrix $\mathbf{X}$ [75]. First, the associations between each two columns are computed. The association c_{ij} is computed as $(\mathbf{x}_i, \mathbf{x}_j)/(\mathbf{x}_i, \mathbf{x}_i)$ where $(\cdot, \cdot)$ is the vector inner product operation and $\mathbf{x}_i$ and $\mathbf{x}_j$ are ith and jth columns of data matrix $\mathbf{X}$ (note that $c_{ij} \neq c_{ji}$). Second, the associations are used to form candidate basis vectors. The association coefficients are binarized and binary association matrix $\mathbf{C}'$ is formed with components c'_{ij}: $c'_{ij} = 1$ if $c \geq \tau$ and $c'_{ij} = 0$ if $c < \tau$ where τ is a parameter of the algorithm. The rows of association matrix $\mathbf{C}'$ are candidate basis vectors. The result matrix $\mathbf{B}'$ is constructed as a subset of rows of $\mathbf{C}'$ at third stage in a greedy way. Initially $\mathbf{A}' = 0^{M\times K}$ and $\mathbf{B}' = 0^{K\times N}$ where $K < \min(M, N)$ is a parameter of the algorithm. The matrices $\mathbf{A}'$ and $\mathbf{B}'$ are updated in the iteration $k = 1, \ldots, K$ by setting $b'_{kn} = c'_{in}$, $n = 1, \ldots, N$, where row index i and values a'_{mk}, $m = 1, \ldots, M$ are chosen maximizing

$$\begin{aligned}\text{cover}(A', B', X, \omega^+, \omega^-) = \quad & \omega^+|\{(i,j) : x_{ij} = 1, (\mathbf{A}' \odot \mathbf{B}')_{ij} = 1\}| \\ & - \omega^-|\{(i,j) : x_{ij} = 0, (\mathbf{A}' \odot \mathbf{B}')_{ij} = 1\}|,\end{aligned} \tag{3.2}$$

where ω^+ and ω^- are parameters of the algorithm and are used to reward for covering 1s and penalize for covering 0s, respectively.

As noted in the comments of the algorithm [75], there exist cases, where the algorithm is able to find only suboptimal solution. For example, if all 1s in some basis vector occur in some other basis vectors, then the algorithm is unable to find that basis vector. Parameter sweeping is another disadvantage of the algorithm, especially for parameter τ.

3.2 Neural network based methods

Neural network based methods that could be applied to BFA are developed for simulation of the brain activity related to concept detection problem. Visual cortex was the most attractive object for simulation, and so the most experiments for testing these methods were performed on images. The Bars Problem (BP) introduced by Foldiak [20] and described in Section 2.4 is a widely-accepted benchmark for learning of objects from complex patterns. Most of the methods considered in this section are illustrated on BP.

3.2.1 Feedforward neural network combining Hebbian and anti-Hebbian learning

The architecture of the network developed by Foldiak [20] is presented in Fig. 3.1. The network has two layers: input and output, $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ designate input and output

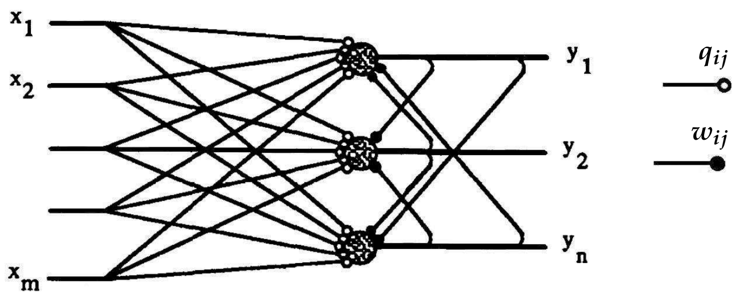


FIGURE 3.1: The architecture of feedforward neural network combining Hebbian and anti-Hebbian learning

activity of the network, correspondingly. Empty circles are feedforward Hebbian excitatory connection with weights q_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$. Filled circles are feedback anti-Hebbian inhibitory connections with weights w_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$. The patterns of images are presented to the network as inputs: the number of input neurons m is equal to the number of pixels in an image. If the network is trained successfully the output activity encodes the components constituting the image on the input layer (for BP the components are bars): the number of output neurons n is equal to the number of possible components. For each input pattern $\mathbf{x}$ the following differential equation for each ith output neuron is solving (initially, $y_i^* = 0$):

$$\frac{dy_i^*}{dt} = f(\sum_{j=1}^{m} q_{ij}x_j + \sum_{j=1}^{n} w_{ij}y_i^* - t_i) - y_i^*,$$

where t_i is activation threshold and $f(u) = 1/(1 + exp(-\lambda u))$ is nonlinear activation function. The network is guaranteed to converge into a stable state if the feedback weights are symmetric, i.e., $w_{ij} = w_{ji}$. The output activity of the network is calculated by rounding the values of y_i^* in the stable state to 0 or 1: $y_i = 1$ if $y^* > 0.5$, $y_i = 0$ otherwise.

After the output activity $\mathbf{y}$ has been determined for input pattern $\mathbf{x}$, the feedforward and feedback connections are calculated by the following equation:

$$\Delta q_{ij} = \beta y_i(x_j - q_{ij})$$
$$\Delta w_{ij} = -\alpha(y_i y_j - p^2), \quad w_{ij} = w_{ji}, \quad w_{ij} \leq 0,$$

where α and β are small positive constants, and constant p gives a desired level of neural network activity. Foldiak considered sparse coding as the important property of biological neural networks [21], and for that, he introduced a variable activation threshold:

$$\Delta t_i = \gamma(y_i - p),$$

where γ is a small positive constant. A neuron that has been inactive for a long time gradually lowers its threshold, while a frequently active neuron gradually becomes more selective by raising its threshold, maintaining the probability of activation of neuron at the same predefined level p.

Competition between neurons due to their mutual inhibition results in the state when only a few neurons are active that receive the largest synaptic excitation from the inputs. If the neural network is performing properly these active neurons correspond to factors. However, in order to keep the network working in the right mode, it is important to choose the right parameters $\alpha, \beta, \gamma, \lambda, p$. Exceptional problems arise with the selection of value for the desired level of the network activity that depends on the parameter p. The matter of the fact is that the selection of the parameter p requires a priory knowledge about a solution of the problem. Unfortunately, the author does not give any recommendations on the selection of the parameters. For the bars problem, the following parameters was chosen: α=0.1, β=γ=0.02, λ=10, p=1/8. In this case the problem was solved after the network was trained with 1200 patterns. More detailed testing of the method performed by Spratling et al. [89] shows that it is very sensitive to parameters selection: small variation in condition of the problem (and also in training set) results in drastically decreasing of the method performance, which could be retrieved by selection of another proper set of parameters.

3.2.2 Auto-associator with non-negative synaptic connections

Auto-associator that is a three-layer neural network with a small number of neurons in the hidden layer (Fig. 3.2) is suggested to solve the bars problem [46]. Input and output layers have the same number of neurons equal to the dimension of the signal space, i.e., to the number of components in a pattern of the learning set. The hidden layer consists of the lower number of neurons, which is a parameter of the method in dependence of conditions of the problem. The network is trained by the standard error back-propagation method, but with an important limitation on the non-negativity of synaptic weights w_{ij}:

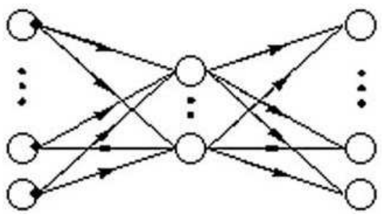


FIGURE 3.2: Three-layer neural network with a small number of neurons in the hidden layer

$$w^*_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t),$$

$$w_{ij}(t+1) = \begin{cases} 0, & \text{if } w^*_{ij}(t+1) < 0; \\ w^*_{ij}(t+1) & \text{if } w^*_{ij}(t+1) \geq 0, \end{cases}$$

where $\Delta w_{ij}(t)$ is the weight change at step t required by back-propagation algorithm. In addition, a limit of unity is applied to the connection between neurons of hidden and output layers, and thus the weights between them are kept within the range $[0, 1]$. To ensure the activity of neurons in the hidden layers are in the range $[0, 1]$ their activation function is the conventional logistic function:

$$f(a_i) = \frac{1}{1 + exp(-(a_i - \theta_i)/\rho)},$$

where a_i is the sum of all inputs to the ith neuron, θ_i is a bias, and ρ is a control parameter, for small values of which the activity of the neurons becomes binary. Neurons in the output layer are linear and involve no bias, hence their activity can be only positive.

The hidden layer is like a bottleneck that passes to the output layer just the information required for most accurately reproducing a pattern on the input layer. While training stage, the synaptic connections are set up in such a way that information passes through the hidden layer with minimal losses. If each pattern of the training set consists of some combination of finite number of independent factors, then the best way to pass information with minimal losses is to learn synaptic weights so that each neuron of the hidden layer corresponds to one factor and this neuron is activated when corresponding factor occurs in pattern on the input layer. The non-negativity restriction on synaptic connections means that the activity of a neuron of the hidden layer could only increase the activity of neurons of the output layer. Hence, after training, if pattern of activity of neurons on the input layer corresponds to some image with objects then active neurons of the hidden layer corresponds to these objects and the activity of neurons on the output layer encodes the image reconstructed as superposition of the objects.

The developers of the method showed the possibility for applying the method to bars problem and achieved good results in comparison with non-negative matrix factorization, but they didn't specified parameters that were used for testing.

3.2.3 Feed-forward neural network with pre-integration lateral inhibition

One of the best feed-forward neural network based method for solving BFA problem is developed by Spratling and Johnson [88]. It is based on the similar feed-forward neural network with lateral inhibition described in Section 3.2.1, but with important modification.

Diagram of a typical, postsynaptic lateral inhibition of the neuron is shown in Fig. 3.3(a): initially, excitations from the neurons of the input layer **x** are summed producing some activity of the neuron of the output layer, and then this activity is reduced due to the influence of inhibitory activity of other neurons of the same layer **y**. Such scheme leads to competition of output activities of the neurons on the principle "winner takes all", and it is hard to get the stable state when few neurons of the output layer are active at once because the

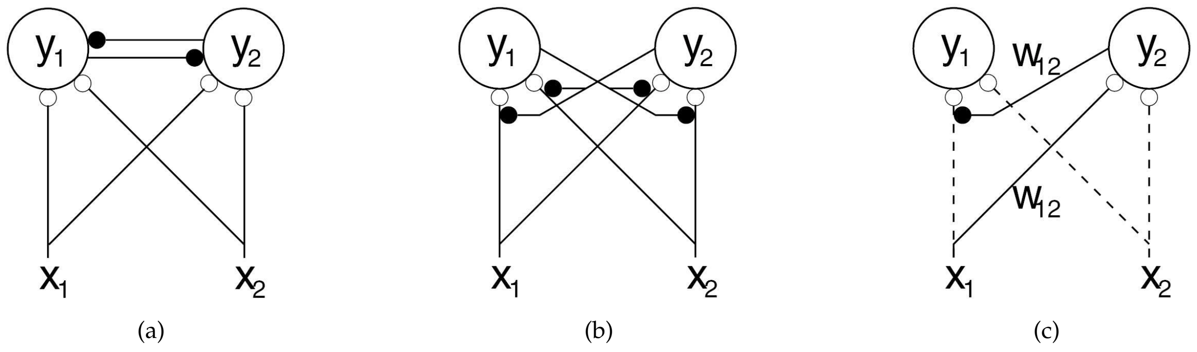


FIGURE 3.3: Models of lateral inhibition. Neurons are shown as large circles, excitatory synapses as small open circles, and inhibitory synapses as small filled circles. (a) The standard model of lateral inhibition provides competition between outputs. (b) The preintegration lateral inhibition model provides competition for inputs. (c) The preintegration lateral inhibition model with only one lateral weight shown. This lateral weight has an identical value (w_{12}) to the afferent weight shown as a solid line.

most active neuron inhibit all others. But in BFA problem the signal could contain several factors, and then several neurons of the output layer corresponding to these factors must be active at once. That is why Spratling and Johnson [88] propose the scheme of pre-integration inhibition, or inhibition of dendritic activity shown in Fig. 3.3(b). In accordance with this scheme, an excitation from a neuron of the input layer to a neuron of the output layer is inhibited separately before they are summed. If ith neuron of the input layer has high activity and excites jth neuron of the output layer then this jth neuron inhibits the excitations of ith input neuron to other neurons of the output layer. Thus we come to the model of competition for the inputs on principle "capture a source, keep it away from others". According to this model, only those neurons are active that successfully captured at least one active neuron of the input layer and insulated it from other neurons by dendritic inhibition. Even in the case of large lateral inhibition, several neurons could be active in the network at the same time if they successfully divide active inputs among themselves. This peculiarity gives the neural network a very useful property: contrast response to input activity with activation of several neurons. At last, for simplification of the network architecture and reduction of number of parameters the exciting and inhibiting synaptic connections were set to be equal, which is illustrated in Fig. 3.3(c).

Let's m to be the number of neurons of the input layer with activities $x_1, \ldots, x_m$, n – the number of neurons of the output layer with activities $y_1, \ldots, y_n$, w_{ij} – the weights of synaptic connections between ith neuron of the input layer and jth neurons of the output layer, then the network activity at the $(t+1)$ step is defined by equation

$$y_j(t+1) = \sum_{i=1}^{m} w_{ij} x'_{ij}$$

$$x'_{ij} = x_i \left(1 - \alpha(t) \max_{\substack{k=1 \\ k \neq j}}^{n} \left\{ \frac{w_{ik}\, y_k(t)}{(\max\limits_{l=1\ldots m} w_{lk})(\max\limits_{l=1\ldots n} y_l(t))} \right\} \right)^{+}$$

where $(v)^+$ is the positive half rectified value of v, α is a scale factor controlling the strength of lateral inhibition that gradually increases at each iteration step from 0 to some maximum value α_{max}. At first step $\alpha = 0$ and there is no inhibition, as a consequence many neurons are activated. But as the coefficient α grows, an increasing number of neurons cease to receive input excitation due to lateral inhibition and stop firing.

Finally, in the proper operating mode, when α is large only few neurons are active, each of which possesses its own group of input active neurons. However, testing of the network showed that there are cases when all neurons of the network continue to be active even for large values of α. Perhaps, it occurs when the value α_{max} is insufficiently large. Another problem is the choice of a step with which the value of α is decreasing. If it is insufficiently small the situation when all neurons of the network become not active during one step is possible. On the contrary, the smaller it is, the slower the algorithm converges to a stable point. For bars problem the developers of the algorithm recommend $\alpha_{max} = 6$ with step of increasing 0.25 [87].

Before training, the weight of synaptic connections are initialized by the Gaussian distributed numbers with mean $\frac{1}{m}$ and standard deviation $0.001\frac{n}{m}$, provided that $w_{ij} = w_{ji}$. The network is trained using unsupervised learning: pattern of the training set is presented to the input layer of the network, then, while the strength of lateral inhibition α increases, activities of neurons of the output layer are changed, and synaptic connections are modified when α amounts up to α_{max} by the rule:

$$\Delta w_{ij} = \begin{cases} \dfrac{(x_i - \bar{x})}{\sum x_k}(y_j - \bar{y})^{+} & \text{if } w_{ij} > 0; \\ \dfrac{(x'_{ij} - \frac{1}{2}x_i)^{-}}{\sum x_k}(y_j - \bar{y}) & \text{if } w_{ij} \leq 0, \end{cases}$$

where $(v)^-$ is the negative half rectified value of v, $\bar{x}$ is the mean level of the input activity, (i.e., $\bar{x} = \frac{1}{m}\sum x_k$), $\bar{y}$ is the mean activity of the output neurons (i.e., $\bar{y} = \frac{1}{n}\sum y_k$).

As noted in [87], we can consider the case of $w_{ij} > 0$. For this case, the performance of the method in solving bars problem would be worsen, but not by much. As shown in [87], even for this case, the performance of the method in solving bars problem surpasses the method of non-negative matrix factorization (in all modifications). The comparison was comprehensive, the only drawback is that testing was performed for standard bars problem, i.e., when the dimension of signal space is small, the number of factors in a signal is small, and without noise. The influence of noise was investigated in [88]. As one would expect, increasing the size of the training set resulted in improvement in the quality of the solution of bars problem, but even in the case of small noise, not less than 4000 training patterns are required to reach an accuracy of the order of 100%. The influence of the increasing number of factors mixed in a signal was investigated in [79]. It was shown that as it grows, the method loses the ability to solve BFA. At last, as shown in [89], use of pre-integration lateral inhibition provides a significant gain in the solution of bars problem in comparison with the ordinary mechanism of lateral inhibition described in Section 3.2.1.

In BFA interpretation, each neuron of the output layer corresponds to a factor. The activity of neurons of the output layer $y_1, \dots, y_n$ corresponds to the vector of factor scores $\mathbf{s}$ in (2.3) for the signal $\mathbf{x}$ activated at the input layer of the network. Since the values $y_1, \dots, y_n$ are not binary they are binarized using a binarization threshold to obtain $\mathbf{s}$. As in Sec. 3.1, the parameters of the generative model $\mathbf{\Theta}$ are estimated using M-step of the likelihood maximization procedure presented in Section 4.2. In the experiments presented in this work, the binarization threshold was the same for all signals $\mathbf{x}$, and it was chosen for each set of experiments to provide maximum of the information gain G defined by (2.15).

3.3 Expectation-maximization methods

The expectation-maximization (EM) method [18] allows for finding parameters of the generative model $\mathbf{\Theta} = \{p_{ij}, q_j, \pi_i\}$ of a given probabilistic generative model to maximize the likelihood of the observed dataset $\mathcal{X}$. The EM method maximizes the likelihood of the observed data by maximizing the free energy $\mathcal{F}(\mathbf{\Theta}, \mathbf{g})$. The iterations of EM alternatively increase $\mathcal{F}$ with respect to the distribution of factor scores $\mathbf{g}$, while holding $\mathbf{\Theta}$ fixed (the E-step), and with respect to parameters of the model $\mathbf{\Theta}$, while holding $\mathbf{g}$ fixed (the M-step). The procedure stops when $\mathcal{F}$ reaches maximum. EM iterative procedure terminates once increasing of $\mathcal{F}$ is small.

3.3.1 Expectation-maximization method for BFA generative model

Expectation-maximization method for BFA generative model (EMBFA) is introduced in [24, 41, 80]. For BFA generative model free energy $\mathcal{F}(\mathbf{\Theta}, \mathbf{g})$ takes the form

$$\mathcal{F} = \sum_{m=1}^{M} \left\{ \sum_{\mathbf{s}} g_m(\mathbf{s}) \big[\log P(\mathbf{x}_m | \mathbf{s}, \mathbf{\Theta}) + \log P(\mathbf{s} | \mathbf{\Theta}) \big] + H(g_m(\mathbf{s})) \right\}, \tag{3.3}$$

where $g_m(\mathbf{s})$ is the expected distribution of factor scores for the mth pattern, $H(g_m(\mathbf{s}))$ is the Shannon entropy of $g_m(\mathbf{s})$, $P(\mathbf{x}_m|\mathbf{s},\mathbf{\Theta})$ is given by (2.7), $P(\mathbf{s}|\mathbf{\Theta})$ is given by (2.9), and $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \ldots, L, j = 1, \ldots, N)$ are the model parameters.

At the E-step, when $\mathbf{\Theta}$ is fixed, the distributions g_m maximizing $\mathcal{F}$ are calculated according to the following equation

$$g_m(\mathbf{s}|\mathbf{\Theta}) = \frac{P(\mathbf{x}_m|\mathbf{s},\mathbf{\Theta})P(\mathbf{s}|\mathbf{\Theta})}{\sum\limits_{\mathbf{s}} P(\mathbf{x}_m|\mathbf{s},\mathbf{\Theta})P(\mathbf{s}|\mathbf{\Theta})}. \tag{3.4}$$

The obtained distributions g_m provide the expected likelihood of the observed data over the factor scores for the given parameters of the generative model [77].

At the M-step, when the distributions g_m are fixed, π_i are calculated as

$$\pi_i = (1/M)\sum_{m=1}^{M} \langle s_i \rangle_{g_m},$$

where $\langle s_i \rangle_{g_m}$ is the expectation of factor score s_i under the distribution $g_m(\mathbf{s}|\mathbf{\Theta})$:

$$\langle s_i \rangle_{g_m} = \sum_{\mathbf{s}} g_m(\mathbf{s}|\mathbf{\Theta}) s_i. \tag{3.5}$$

Respectively, p_{ij} and q_j are obtained by steepest ascent maximization of $\mathcal{F}$:

$$\Delta p_{ij} = \gamma \frac{\partial \mathcal{F}}{\partial p_{ij}}, \qquad \Delta q_j = \gamma \frac{\partial \mathcal{F}}{\partial q_j}, \tag{3.6}$$

where γ is a learning rate,

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial p_{ij}} &= \sum_{m=1}^{M}\sum_{\mathbf{s}} g_m(\mathbf{s}|\mathbf{\Theta}) P(x_{mj}|\mathbf{s},\mathbf{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s},\mathbf{\Theta})}{\partial p_{ij}} \\ \frac{\partial \mathcal{F}}{\partial q_j} &= \sum_{m=1}^{M}\sum_{\mathbf{s}} g_m(\mathbf{s}|\mathbf{\Theta}) P(x_{mj}|\mathbf{s},\mathbf{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s},\mathbf{\Theta})}{\partial q_j} \end{aligned} \tag{3.7}$$

and

$$\begin{aligned} \frac{\partial P(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})}{\partial p_{ij}} &= (x_{mj} - P(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})) \frac{\langle s_i \rangle_{g_m}}{1-p_{ij}} \\ \frac{\partial P(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})}{\partial q_j} &= (x_{mj} - P(x_{mj}|\mathbf{s}_m,\mathbf{\Theta})) \frac{1}{1-q_j}. \end{aligned} \tag{3.8}$$

The probabilities p_{ij} are assumed to be sufficiently high for the components constituting the ith factor ($f_{ij} = 1$) and equal to zero for the other components ($f_{ij} = 0$), at each iteration cycle of step M we put $p_{ij} = 0$ if

$$p_{ij} < 1 - \prod_{l \neq i} (1 - \pi_l p_{lj}), \tag{3.9}$$

where the right side of the inequality is the probability that the jth attribute appears in the pattern due to other factors besides $\mathbf{f}_i$.

The obtained values of p_{ij}, q_j and π_i are used as the input for the next E–step. EM iterative procedure continues while $\sum_{ij} |\Delta p_{ij}|/LN > 10^{-3}$. After the convergence of the procedure, the resulting values $\langle s_i \rangle_{g_m}$ are the estimates of the factor scores which are not binary but gradual. To satisfy the generative model, these values are binarized. The binarization threshold is chosen so that to maximize the BFA information gain [41].

Due to computational complexity, the developers of the method restricted the EMBFA algorithm to the case of sparse scores, when only a small number of factors (no more than three) are supposed to be mixed in the observed patterns. In this case, summation over $\mathbf{s}$ in the above formulas (3.3, 3.4, 3.5, 3.7) is reduced to

$$\sum_{\mathbf{s}} (\ldots) = (\ldots)_{\mathbf{s}=0} + \sum_{i} (\ldots)_{\mathbf{s}=\mathbf{s}_i} + \sum_{i<j} (\ldots)_{\mathbf{s}=\mathbf{s}_{ij}} + \sum_{i<j<k} (\ldots)_{\mathbf{s}=\mathbf{s}_{ijk}}, \tag{3.10}$$

where $\mathbf{s}_i$ is the vector of factor scores with all zeros except s_i, $\mathbf{s}_{ij}$ is the vector of factor scores with all zeros except s_i and s_j, and $\mathbf{s}_{ijk}$ is the vector of factor scores with all zeros except s_i, s_j and s_k. The EMBFA algorithm was also implemented using a neural network [38].

3.3.2 Expectation-maximization method for maximal causes analysis

Lucke and Sahani [71] have studied the bars problem with the expectation-maximization method. In the generative model studied by Lucke and Sahani [71], multiple active hidden causes (factors in terms of BFA) combine to determine the values of an observed variable through a max function. Each cause results in a set of observations given by a vector of generative influences (factor loadings in terms of BFA). Free-energy $\mathcal{F}$ has the same form as for Expectation-Maximization method for BFA generative model (3.3) but difference in generative model results in different formula of derivative of $\mathcal{F}$ and consequently formulas for both E and M steps are also different.

In Maximal Causes Analysis (MCA) generative model a weight of association $w_{ij} \in \mathcal{R}$ between ith factor and jth component of signal is used instead of probability p_{ij} in BFA generative model. The observed variables (components of signals) are not binary but non-negative integer values, i.e., $\mathbf{x} \in \mathcal{Z}_+^N$ where N is the dimension of signal space. Factor scores (active hidden causes in terms of MCA) are drawn from a multivariate Bernoulli distribution; and observed variables, given the causes, conditionally independent and Poisson-distributed:

$$P(\mathbf{s}|\text{ß}) = \prod_{i=1}^{L} P(s_i|\pi_i), P(s_i|\pi_i) = \pi_i^{s_i}\,(1-\pi_i)^{1-s_i}, \tag{3.11}$$

$$P(\mathbf{x}|\mathbf{s},\mathbf{W}) = \prod_{j=1}^{N} P(x_j|\bar{\mathbf{W}}_j(\mathbf{s},\mathbf{W})),\;\; P(x_j|w) = \frac{w^{x_j}}{x_j!}\,\mathrm{e}^{-w}, \tag{3.12}$$

where $\boldsymbol{\pi}$ is the vector of probabilities of factor scores π_i, $i = 1,\dots,L$, $\mathbf{W} \in \mathcal{R}^{L\times N}$ is a matrix of association weights between components of signals and factors, and $\bar{\mathbf{W}}_j(\mathbf{s},\mathbf{W}) = \max\limits_{i=1\dots L}\{s_i w_{ij}\}$ is *effective weights* on x_j.

The partial derivative of $\mathcal{F}$ with respect to association weights w_{ij} requires calculation of derivative of $\bar{\mathbf{W}}_j$, but $\bar{\mathbf{W}}_j$ is not differentiable, for this reason, the developers of the method defined a smooth function $\bar{\mathbf{W}}_j^{\rho}$ that converges to $\bar{\mathbf{W}}_j$ as ρ approaches infinity:

$$\bar{\mathbf{W}}_j^{\rho}(\mathbf{s},\mathbf{W}) = \left(\sum_{i=1}^{L}(s_i w_{ij})^{\rho}\right)^{\frac{1}{\rho}} \Rightarrow \lim_{\rho\to\infty} \bar{\mathbf{W}}_j^{\rho}(\mathbf{s},\mathbf{W}) = \bar{\mathbf{W}}_j(\mathbf{s},\mathbf{W}),$$

and replaced the derivative of $\bar{\mathbf{W}}_j$ by the limiting value of the derivative of $\bar{\mathbf{W}}_j^{\rho}$, which was written as $\mathcal{A}_{ij}$:

$$\mathcal{A}_{ij}(\mathbf{s},\mathbf{W}) = \lim_{\rho\to\infty}\left(\frac{\partial}{\partial w_{ij}}\bar{\mathbf{W}}_j^{\rho}(\mathbf{s},\mathbf{W})\right) = \lim_{\rho\to\infty}\frac{s_i(w_{ij})^{\rho}}{\sum\limits_{l=1}^{L} s_l(w_{lj})^{\rho}}.$$

Taking into account these tricks, the parameters of the generative model are estimated at the M-step by equations:

$$w_{ij} = \frac{\sum\limits_{m=1}^{M}\left(\sum\limits_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta})\mathcal{A}_{ij}(\mathbf{s},\mathbf{W})\right)x_{jm}}{\sum\limits_{m=1}^{M}\sum\limits_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta})\mathcal{A}_{ij}(\mathbf{s},\mathbf{W})}$$

$$\pi_i = (1/M)\sum_{m=1}^{M}\langle s_i\rangle_{g_m} \text{ with } \langle s_i\rangle_{g_m} = \sum_{\mathbf{s}} g_m(\mathbf{s}|\boldsymbol{\Theta})s_i.$$

At the E-step the distributions g_m are calculated by the same equation (3.4) as for EMBFA, but the probabilities $P(\mathbf{x}_m|\mathbf{s},\boldsymbol{\Theta})$ and $P(\mathbf{s}|\boldsymbol{\Theta})$ are given by (3.12) and (3.11), respectively. In order to reduce computational complexity, the summation over $\mathbf{s}$ in all formulas is restricted to the case when each pattern of the dataset contains not more than three factors. Note that the same restriction is applied in EMBFA (3.10). The maximal causes analysis method with this restriction is called as MCA3.

If all influences have the same value, then max function is equivalent to the Boolean summation of the influences and the generative model becomes almost equivalent to the BFA generative model. The difference is in the types of noise used in two generative models. In the BFA generative model, noise is supposed to be in the form of factor distortion and specific factors, while Lucke and Sahani [71] supposed noise in the form of a random choice of the observed variable according to the Poisson distribution with mean equal to the strongest influence.

The developers of the method compared it with several non-negative matrix factorization (NMF) methods and with feed-forward neural network with pre-integration lateral inhibition described in Section 3.2.3 and showed that for bars problem its performance is better than all others [71].

Since the generative model of signals in Maximal Causes Analysis is different from BFA generative model, a post processing is required when the method is used in solving BFA. The values of $\langle s_i \rangle_{g_m}$, $i = 1, \ldots, L$, are probabilities that m-th observation of the dataset contain i-th found factor. These probabilities are transformed to binary factor scores using a binarization threshold. Then the parameters of the generative model $\mathbf{\Theta}$ are estimated using M-step of the likelihood maximization procedure presented in Section 4.2. In the experiments presented in this work, the binarization threshold was the same for all observations of the dataset, and it was chosen for each set of experiments to provide maximum of the information gain G defined by (2.15).

Chapter 4

Neural network based method for Boolean factor analysis

The idea of using the Hopfield-like neural network for solving the BFA problem is based on the well known property of Hopfield network to create attractors of the network dynamics by assemblies of tightly connected neurons. If N-dimensional binary patterns of the signal space are interpreted as activities of N binary neurons (1 – active, 0 – nonactive) then neurons representing a factor are activated simultaneously each time when the factor appears in the patterns of the dataset, and neurons representing different factors are rather seldom activated simultaneously. Therefore, due to the Hebbian learning rule, the factor neurons become more tightly connected than the other neurons. So factors can be revealed as attractors of the network dynamics.

Attractor neural network with increasing activity (ANNIA) presented in Section 4.1 reveals factors as groups of tightly connected neurons. The results obtained by computer simulations in Chapter 5 show that ANNIA enhanced with Hebbian unlearning of found factors (5.20) is able to reveal all factors in the wide range of parameters of input data so long as restrictions derived in Sections 5.3 and 5.4 are satisfied. Although it provides an accurate estimation of the factor loadings (as set of active neurons in true attractor), but it provides only approximate estimation of factor scores, and no estimation of the parameters of the generative model p_{ij} and q_j. A way to overcome this drawback is to combine ANNIA with likelihood maximization (LM) described in Section 4.2. It is shown in Section 4.3 that the LM procedure itself is able to provide complete solution of the BFA problem but requires an appropriate initial approximation. If it starts from the random initial parameters it commonly fails. In the combination of ANNIA and LM the role of ANNIA is to provide LM with the initial approximation. Another aspect of the ANNIA and LM interaction is a suppression of the dominant attractors in ANNIA using the data provided by LM. The resulting hybrid ANNIA and LM procedure is presented in Section 4.3. The whole algorithm is presented in Appendix D.

4.1 Attractor neural network with increasing activity – ANNIA

The proposed neural network consists of N neurons of the McCulloch-Pitts type (integrate-and-fire binary neurons) with gradually ranged synaptic connections between them. Only a fully connected case is considered here. Each pattern of the learning set x_m is stored in the matrix of synaptic connections $\mathbf{J}'$ according to the modified Hebbian rule:

$$J'_{ij} = \sum_{m=1}^{M} (x_{mi} - a_m)(x_{mj} - a_m),\ i,j = 1,\dots,N, i{\neq}j,\ J'_{ii} = 0 \tag{4.1}$$

where M is the number of patterns in the learning set and bias $a_m = \sum_{i=1}^{N} x_{mi}/N$ is the total relative activity of the m-th pattern. This form of bias corresponds to the biologically plausible global inhibition being proportional to overall neuron activity.

Under conditions discussed in Section 5.2, two global spurious attractors dominate in the network dynamics. To suppress their dominance one special inhibitory neuron is added to the N principal neurons of the Hopfield network. The neuron is activated during the presentation of every pattern of the learning set and is connected with all the principal neurons by bidirectional connections. The patterns of the learning set are stored in column vector $\mathbf{j}''$ of the connections according to the Hebbian rule:

$$j''_i = \sum_{m=1}^{M} (x_{mi} - a_m) = M(b_i - \bar{b}), i = 1,\dots,N, \tag{4.2}$$

where $b_i = \sum_{m=1}^{M} x_{mi}/M$ is a mean activity of the i-th neuron in the learning set and $\bar{b}$ is a mean activity of all neurons in the learning set (note $< a_m >= \bar{a} = \bar{b}$). It is also supposed that the excitability of the inhibitory neuron decreases inversely proportional to the size of the learning set, being $1/M$ after all patterns stored. In the recall stage its activity is then:

$$A(t) = (1/M)\sum_{i=1}^{N} j''_i x_i(t) = (1/M)\mathbf{j}''^T\mathbf{x}(t)$$

where $\mathbf{j}''^T$ is transposed $\mathbf{j}''$, and $\mathbf{x}(t)$ is a column vector of activity of the network on step t. The inhibition produced in all principal neurons of the network is given by vector $\mathbf{j}''A(t) =$

$(1/M)\mathbf{j}''\mathbf{j}''^T\mathbf{x}(t)$. Thus, the inhibition is equivalent to the subtraction of $\mathbf{J}''$ from $\mathbf{J}'$ where

$$\mathbf{J}''_{ij} = M(b_i - \bar{b})(b_j - \bar{b}),\ i,j = 1,\dots,N, i\neq j,\ J''_{ii} = 0. \tag{4.3}$$

Adding the inhibitory neuron is equivalent to replacing the ordinary connection matrix $\mathbf{J}'$ by the matrix $\mathbf{J}$:

$$\mathbf{J} = \mathbf{J}' - \mathbf{J}''. \tag{4.4}$$

The following two-run recall procedure is used to reveal factors. Its initialization starts by the presentation of a random initial pattern $\mathbf{x}^{\text{in}}$ with $k_{\text{in}} = r_{\text{in}}N$ active neurons. Activity k_{in} is supposed to be smaller than the activity of any factor. On presentation of $\mathbf{x}^{\text{in}}$, network activity $\mathbf{x}$ evolves to some attractor. This evolution is determined by the synchronous discrete time dynamics. At each time step:

$$x_i(t+1) = \Theta(h_i(t) - T(t)),\ i = 1,\dots,N, \quad x_i(0) = x_i^{\text{in}} \tag{4.5}$$

where h_i are components of the column vector of synaptic excitations

$$\mathbf{h}(t) = \mathbf{J}\mathbf{x}(t), \tag{4.6}$$

Θ is a step function, and $T(t)$ is an activation threshold. At each time step of the recall process the threshold $T(t)$ is chosen in such a way that the level of the network activity is kept constant and equal to k_{in}. Thus, on each time step k_{in} "winners" (neurons with the greatest synaptic excitation) are chosen and only they are active on the next time step. As shown by Frolov et al. [26], this choice of activation threshold enables the network activity to stabilize in point or cyclic attractors of length two. The fixed level of activity at this stage of the recall process could be ensured by biologically plausible non-linear negative feed-back control accomplished by the inhibitory interneurons.

When activity stabilizes at the initial level of activity k_{in}, $k_{\text{in}} + 1$ neurons with maximal synaptic excitation are chosen for the next iteration step, and network activity evolves to an attractor at the new level of activity $k_{\text{in}} + 1$. The level of activity then increases to $k_{\text{in}} + 2$, and so on, until the number of active neurons reaches the final level $k_{\text{fin}} = r_{\text{fin}}N$ where $r = k/N$ is a relative network activity. Thus, one trial of the recall procedure contains $(r_{\text{fin}} - r_{\text{in}})N$ external steps and several internal steps (usually 2-3) inside each external step to reach an attractor for a given level of activity.

At the end of each external step, when the network activity stabilizes at the level of k active neurons, the Lyapunov function is calculated:

$$\lambda = \mathbf{x}^{\mathrm{T}}(t+1)\mathbf{J}\mathbf{x}(t)/k, \tag{4.7}$$

where $\mathbf{x}^{\mathrm{T}}(t+1)$ and $\mathbf{x}(t)$ are two network states in a cyclic attractor (for point attractor $\mathbf{x}^{\mathrm{T}}(t+1) = \mathbf{x}(t)$). In this definition, the value of the Lyapunov function gives a mean synaptic excitation of neurons belonging to an attractor at the end of each external step. The identification of factors is based on the analysis of the dynamics of the Lyapunov function $\lambda(k)$ and the activation threshold $T(k)$ in the recall procedure. Figure 4.1(a) demonstrates changes of the Lyapunov function along the trials of the recall procedure. The trajectories shown in Fig. 4.1 were obtained for a 8x8 bars problem dataset containing $M = 400$ patterns without noise ($p = 1$, $q = 0$), the number of ones in factors $n = 8$. Trajectories of the network dynamics in Fig. 4.1(a) form two separated groups. As shown in Fig. 4.2, the trajectories with higher Lyapunov function values are true and the lower ones are spurious. Each point in the Figure represents one of the trajectories. Its ordinate is the value of the Lyapunov function at the point $k = n$ while its abscissa is the maximal overlap of the pattern of network activity at this point across all factors, i.e., $\mathrm{Ov} = \max_{i=1,\ldots,L} m(\mathbf{x}, \mathbf{f}_i)$ where the overlap between two patterns $\mathbf{x}_1$ and $\mathbf{x}_2$ with $n = Np$ active neurons was calculated by the formula:

$$m(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{Np(1-p)} \sum_{i=1}^{N} (x_{1i} - p)(x_{2i} - p).$$

According to this formula the overlap between identical patterns is equal to 1 and the mean overlap between random patterns is equal to 0. Patterns with high Lyapunov function values have high overlaps with one of the factors, while patterns with low Lyapunov function values are spurious and distant from all the factors. This shows that true and spurious trajectories and attractors are separated by the values of their Lyapunov functions.

The trajectories with higher Lyapunov function values are called as true, they pass through network states coincided with factors. As illustrated in Fig. 4.1(a), at the initial part of the true recall trajectory, when $k < n$, stable states are factor fragments and their neurons are tightly connected. The joining of such neurons results in an increase of the Lyapunov function approximately proportionate to k. When $k > n$, the increase of k occurs due to the join of some random neurons that are connected with neurons of the factor by weak connections. Hence, the increase of the Lyapunov function for the trajectories sharply breaks at the point $k = n$.

The second characteristic feature of the trajectories lies in the behavior of their activation threshold obtained at the end of each external step (Fig. 4.1(b)). In fact, the activation threshold is the maximum synaptic excitation over non-active neurons. Initially, the activation threshold increases when k increases. This behavior stems from the well known fact that in

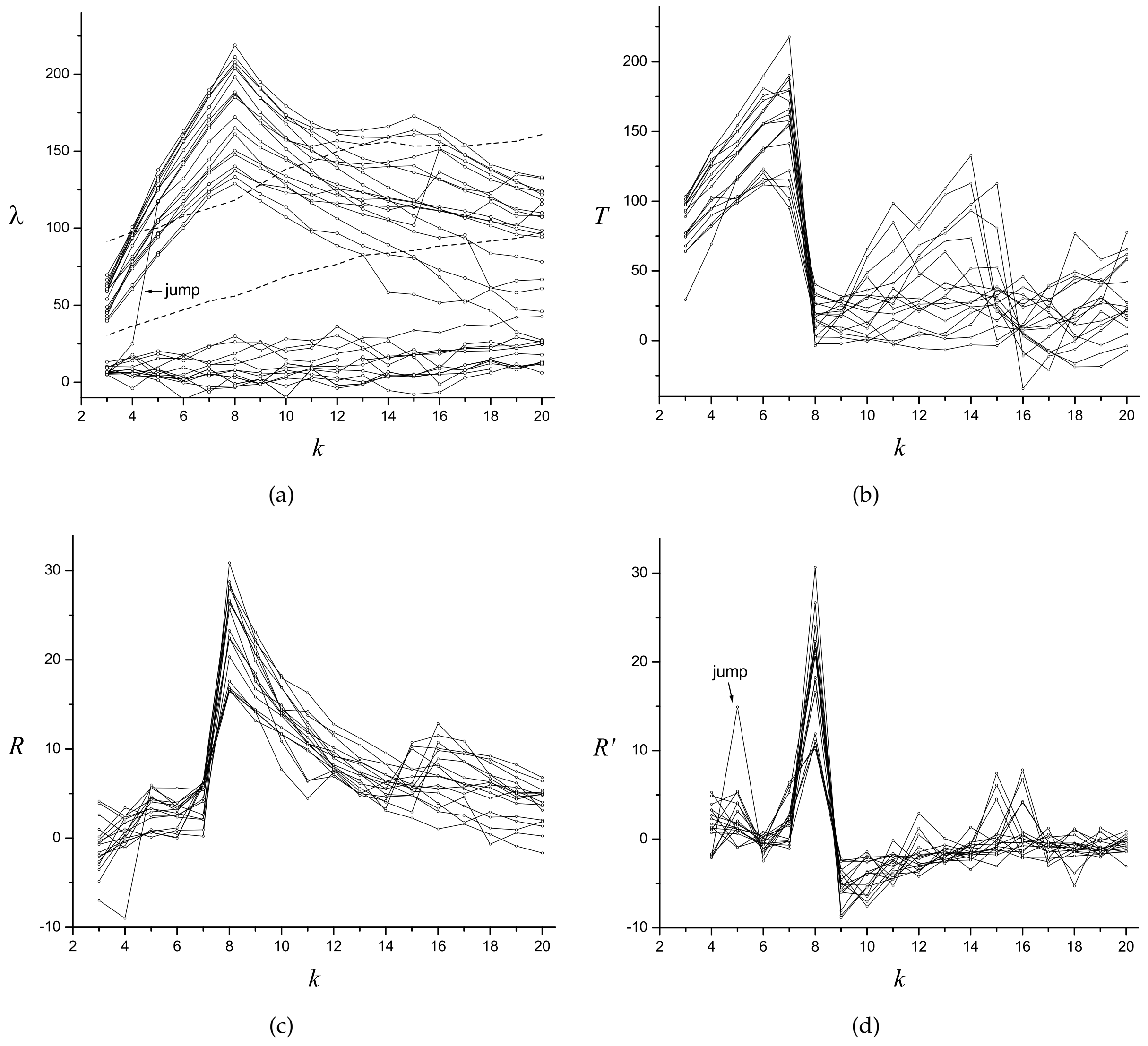


FIGURE 4.1: Lyapunov function λ (a), activation threshold T (b), function $R = \lambda/(k-1) - T/k$ (c) and its derivative (d) in dependence on the number of active neurons k. Dashed lines in (a) are thresholds for separating true and spurious trajectories at the beginning (upper line) and at the end (lower line) of the recall procedure. The example of a jump from one to another continuous trajectory is marked in (a) and (c). The results were obtained for the 8x8 bars problem dataset containing $M = 400$ patterns without noise ($p = 1$, $q = 0$).

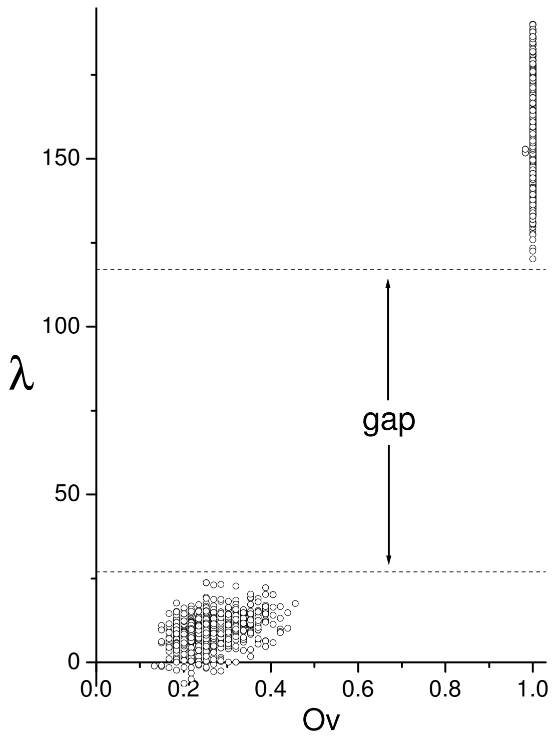


FIGURE 4.2: Values of the Lyapunov functions at the point $k = n$ in relation to overlaps with the closest factors for the trajectories in Fig. 4.1(a). It is shown that true and spurious attractors can be easily separated due to a large gap between the distributions of the Lyapunov function values.

the Hopfield network during activation of the fragment of a stored pattern, the distribution of synaptic excitations has two, separate, high and low modes: high - for neurons belonging to the pattern and low - for neurons not belonging to it (see [27]). The mean of the high mode increases proportionately to the size of the fragment and the mean of the low mode is close to zero. When $k < n$, a fragment of the factor is activated along the trajectory, then the activation threshold is inside the high mode, and hence, increases with k. However when the stored factor is activated totally ($k = n$), the activation threshold has to jump down to the low mode to activate an additional neuron not belonging to the factor.

To match the two properties of the trajectories we subtract the activation threshold from the Lyapunov function calculating $R(k) = \lambda(k)/(k-1) - T(k)/k$ and its derivative $R'(k) = R(k) - R(k-1)$. The changes of R and R' along the trajectories are shown in Figures 4.1(c) and 4.1(d). The curves for R' have distinctly exposed peak at the point $k = n$. Thus, the maximum of $R'(k)$ was used as an indicator of the factor on each recall trajectory. The state of the neural network activity at the maximum gives the factor loadings $\mathbf{f}$ for the found factor.

As shown in Fig. 4.1(a), sometimes Lyapunov function jumps up from one to another continuous trajectory. In this step, the network activity transits to an attractor far from the attractor at the previous step. As shown in Fig. 4.1(d), such a transition could also produce a peak of R'. To avoid falsely treating such transition as factors, at each point of each trajectory the similarity $Sim(k)$ between the patterns of the network activity in the current attractor $\mathbf{x}_{\mathrm{attr}}(k)$ and in the previous attractor $\mathbf{x}_{\mathrm{attr}}(k-1)$ is calculated as

$$Sim(k) = \frac{c - (k-1)k/N}{(k-1)(1-k/N)}, \tag{4.8}$$

where c is the number of common Ones in $\mathbf{x}_{\mathrm{attr}}(k)$ and $\mathbf{x}_{\mathrm{attr}}(k-1)$. If $\mathbf{x}_{\mathrm{attr}}(k)$ contains $\mathbf{x}_{\mathrm{attr}}(k-1)$, then $Sim(k) = 1$. If $\mathbf{x}_{\mathrm{attr}}(k)$ and $\mathbf{x}_{\mathrm{attr}}(k-1)$ are independent, then $Sim(k)$ is equal to zero on average. It is assumed that the pattern of the network activity changes smoothly along the trajectory if $Sim(k) \geq Sim_{\mathrm{thr}}$, where $Sim_{\mathrm{thr}} = 0.8$. In the opposite case, the transition from $\mathbf{x}_{\mathrm{attr}}(k-1)$ to $\mathbf{x}_{\mathrm{attr}}(k)$ is treated as a jump. Thus, the point on the trajectory with the largest peak for R' could be considered as related to the factor if only there was no jump at this point.

The network dynamics can converge not only to the true attractors corresponding to the factors, but also to spurious attractors far from all factors. The Lyapunov function for the spurious attractors is smaller than that for factors (Fig. 4.1(a)), otherwise spurious attractors become dominant and factors could not be revealed (see Sec. 5.4 for details). The trajectories with lower Lyapunov function values are called as spurious. To separate the true attractors from the spurious ones, the following heuristic method is used. For each k a random set of k neurons is activated and then the maximal synaptic excitation over all neurons of the network is calculated. After repeating this procedure 100 times, mean $m(k)$ and standard deviation $\sigma(k)$ of the maximal excitations are calculated. If the Lyapunov function in the peak of R' along the trajectory satisfies the following inequality

$$\lambda(k) > h_{\max}(k) = m(k) + 3\sigma(k), \tag{4.9}$$

the found point on the trajectory is treated as a factor, in the opposite case — as a spurious state. The borders $h_{\max}$ separating true and spurious trajectories are shown in Fig. 4.1(a) by the dashed lines. The upper curve corresponds to the beginning of the recall procedure when the first factor with the highest Lyapunov function was found. The lower curve corresponds to the end of the recall procedure when the last factor with the lowest Lyapunov function was found. Note that the border $h_{\max}(k)$ separating the true and spurious trajectories markedly decreases as the factor discovering proceeds.

4.2 Likelihood maximization – LM

Since the generative model is defined in terms of probabilities one can set up likelihood function and find generative model parameters by likelihood maximization. For BFA generative model, the dataset likelihood takes the form (2.16).

To maximize Λ the iterative procedure could be used that is very similar to the classic expectation maximization (EM) procedure [18]. The iterations alternatively increase Λ with respect to a set of factor scores s_{mi} ($m = 1, \dots, M$, $i = 1, \dots, L$), while holding $\mathbf{\Theta}$ fixed (the E-step), and with respect to parameters of the model $\mathbf{\Theta}$, while holding s_{mi} fixed (the M-step).

At the M-step, when scores are fixed, p_{ij} and q_j can be found by maximization of Λ according to the following iterative procedure:

$$\Delta p_{ij} = \gamma_{ij} \frac{\partial \Lambda}{\partial p_{ij}}, \qquad \Delta q_j = \gamma_j \frac{\partial \Lambda}{\partial q_j}, \tag{4.10}$$

where γ_{ij} and γ_j are positive learning rates and according to (2.16), (2.17) and (2.7),

$$\begin{aligned} \frac{\partial \Lambda}{\partial p_{ij}} &= \sum_{m=1}^{M} P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})}{\partial p_{ij}} \\ \frac{\partial \Lambda}{\partial q_j} &= \sum_{m=1}^{M} P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})^{-1} \frac{\partial P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})}{\partial q_j}, \end{aligned}$$

and

$$\begin{aligned} \frac{\partial P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})}{\partial p_{ij}} &= (x_{mj} - P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})) \frac{s_{mi}}{1 - p_{ij}} \\ \frac{\partial P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})}{\partial q_j} &= (x_{mj} - P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta})) \frac{1}{1 - q_j}. \end{aligned}$$

Since the probabilities p_{ij} are assumed to be sufficiently high for the components constituting the i-th factor ($f_{ij} = 1$) and equal to zero for the other components ($f_{ij} = 0$), at each iteration cycle of step M, $p_{ij} = 0$ if

$$p_{ij} < 1 - \prod_{l \neq i} (1 - \pi_l p_{lj}),$$

where the right side of the inequality is the probability that the j-th attribute appears in the pattern due to other factors except $\mathbf{f}_i$. It is worth noting that without threshold truncation of p_{ij}, the procedure does not converge at all because of uncertainties arising from the competition between common and specific factors. To eliminate this uncertainty most of the components of each common factor are put to be zero.

The learning rates in (4.10) are set to be

$$\gamma_{ij} = p_{ij}(1 - p_{ij})/(M\pi_i), \qquad \gamma_j = q_j(1 - q_j)/M. \tag{4.11}$$

The iterative procedure (4.10) at each M-step continues until $\sum_{i,j} |\Delta p_{ij}| / \sum_{i,j} p_{ij} < 10^{-5}$.

The generative model parameters p_{ij} and q_j obtained at the M-step are used as the input for the next E-step to find factor scores. For each individual signal $\mathbf{x}_m$ of the dataset, factor scores $\mathbf{s}_m$ can be found as those maximizing Λ_m. The global maximum of Λ_m can be provided only by the exhaustive search. However, the number of possible versions of $\mathbf{s}_m$ is usually large (equal to 2^L), then to use some iterative procedure providing at each iterative step the increase of likelihood function is more reasonable even if the procedure provides only local maximum. One of the possible procedures is following.

At each iterative step the values $\Lambda_m|_{s_{mi}=1}$ and $\Lambda_m|_{s_{mi}=0}$ obtained by substituting $s_{mi} = 1$ and $s_{mi} = 0$ into (2.17) given p_{ij}, q_j and π_i are compared. The value of s_{mi} that provides the greater $\Lambda_m|_{s_{mi}}$ is chosen and the procedure goes to another i until it converges. To compute $\mathbf{s}_m$ the following two-run iterative procedure is used. At each external cycle of the procedure all components of $\mathbf{s}_m$ are processed to maximize Λ_m. Then the sequence of their processing is randomly permuted at each cycle. The procedure is terminated when $\mathbf{s}_m$ remained the same at the next cycle. The procedure converges because at each iterative step the likelihood function does not decrease. The procedure starts with all $s_{mi} = 0$. After computing $\mathbf{s}_m$ the procedure is applied to the next signal $\mathbf{x}_{m+1}$ until the dataset is exhausted. The probabilities π_i are estimated at the end of E-step as frequencies of i-th factor appearance in the dataset.

The scores found at the E-step are used as input to the next M-step. If for some factor, all found loadings or scores are zeros, this factor is excluded from the list of found factors.

The LM-iterative procedure terminates when the increment of the Λ at the next step does not exceed $10^{-6}MN$ or the number of steps in the LM procedure reaches 10.

4.3 Hybrid ANNIA and likelihood maximization method – LANNIA

In hybrid method ANNIA and LM procedures are performing successively. It starts from ANNIA step. Then factors revealed by ANNIA in several trials (usually, 10–50) are used to initiate LM. The LM procedure is initiated from E-step. The probabilities p_{ij} required to start LM procedure are estimated as

$$p_{ij} = \begin{cases} h_{ij} / \max\limits_{j'=1\ldots N} \{h_{ij'}\} & \text{if } h_{ij} > 0 \\ 0 & \text{if } h_{ij} \leq 0 \end{cases}, \tag{4.12}$$

where $\mathbf{h}_i = \mathbf{f}_i\mathbf{J}$ is a row vector of synaptic excitations of neurons when the i-th found factor is activated in the network. This estimation seems to be reasonable because h_{ij} is proportional to p_{ij} [33].

After the LM convergence, each i-th found factor is unlearned from ANNIA by subtracting the matrix $\Delta\mathbf{J}^i$ from the matrix of synaptic connections $\mathbf{J}$ where

$$\Delta J^i_{jk} = M\pi_i(1-\pi_i)p_{ij}(1-p^0_{ij})p_{ik}(1-p^0_{ik}),\ k{\neq}j,\ \Delta J^i_{jj}{=}0, \tag{4.13}$$

where p^0_{ij} is a probability that the j-th component takes One in signals not containing i-th found factors. Probabilities π_i, p_{ij} and p^0_{ij} are obtained from the LM procedure. The modified matrix of synaptic connections is used in next ANNIA step for searching the other factors with lower Lyapunov function.

The parameters of the generative model and the factor scores obtained by LM are used to calculate the information gain provided by all found factors. The LANNIA continues until G stops to increase due to adding new found factors. This is the first criterion to terminate LANNIA. The border $h_{\max}$ defined in (4.9) separates true and spurious trajectories. The appearance of only spurious attractors in the recall procedure indicates that all factors are found. This is the second termination criterion of LANNIA.

On the whole, the main steps of LANNIA are the following:

1. Signals of a dataset are stored in the fully connected neural network according to Hebbian rule, forming the matrix of synaptic connections between neurons $\mathbf{J}'$ given by (4.1).

2. Two global attractors are excluded from the network dynamics by subtracting the matrix $\mathbf{J}''$ given by (4.3) from the matrix $\mathbf{J}'$ forming the matrix $\mathbf{J}$ according to (4.4).

3. 10 – 50 trajectories sequentially start from random states containing k_{in} active neurons and continue according to the two-run procedure until the number of active neurons reaches k_{fin}. The initial activity k_{in} is chosen to be lower and the final activity k_{fin} is chosen to be higher than the expected number of Ones in the factors. Usually we take $k_{\text{in}} = 5$ and $k_{\text{fin}} = 100$.

4. Only the trajectories satisfying the two following criteria are chosen from the obtained trajectories for a further analysis. First, they are smooth, according to the condition $Sim(k) \geq Sim_{\text{thr}} = 0.8$, where $Sim(k)$ is given by (4.8). Second, they are true, according to (4.9). The patterns of the network activity at peaks of R' at the chosen trajectories are treated as revealed factors. If neither trajectory satisfies both these criteria, LANNIA is terminated.

5. The probabilities p_{ij} for factors revealed at step 4 are estimated by (4.12). The probabilities π_i for these factors are set to be 0.5. The probabilities p_{ij} and π_i for factors revealed earlier and the probabilities q_j are taken from the output at the previous LANNIA cycle. At the

first cycle of LANNIA only the probabilities p_{ij} estimated by (4.12) and $\pi_i = 0.5$ are used as the input to the LM procedure, and the seed probabilities q_j are taken to be $1/M$.

6. The LM procedure alternates steps E and M until converges. The output of the procedure is a set of the generative model parameters and factor scores.
7. The information gain provided by found factors for this temporary BFA solution is calculated. If it is not higher than the gain obtained at the previous step, LANNIA is terminated. In the opposite case the parameters of the generative model obtained at step 6 are used for unlearning the new found factors from the neural network by the rule (4.13), and the procedure returns to step 3.

The execution time required for ANNIA is composed of two times. The first time T_1 is required to create connection matrix. In the limit of large M, L and N, $T_1 \simeq 10^{-9}MN^2$. The second time is required to find factors: $T_2 \simeq 10^{-8}LN\langle n_f\rangle^2$, where n_f is the mean number of ones in factors. For LANNIA additional time $T_3 \simeq 10^{-7}ML\langle n_f\rangle$ is required to perform the procedure of likelihood maximization. Coefficients in these formulas are calculated for PC Core2 6400, 2.13 GHz.

The LANNIA performance is illustrated when solving the so called bars problem presented in Section 2.4 in the case of noisy images. The noise is assumed to be distributed uniformly over observation components and factors so that $q_j = q$ for any j, and $p_{ij} = pf_{ij}$ for any i and j. This means that pixels constituting a bar can take 0 with the equal probabilities $1 - p$ and any pixel can take 1 with the probability q due to related specific factor. The method was tested using dataset containing $M = 800$ noisy images ($p = 0.7$ and $q = 0.2$). Several examples of images of the generated dataset are presented in Fig. 4.3(**A**).

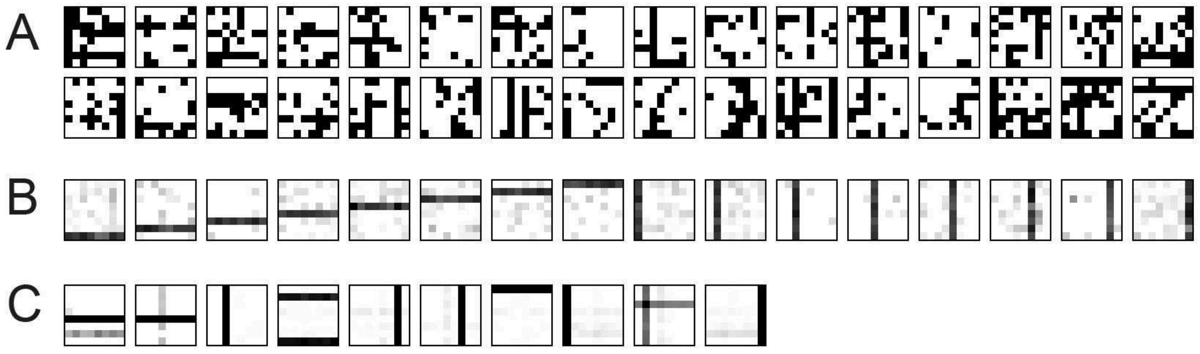


FIGURE 4.3: **A** Examples of noisy images ($p = 0.7$, $q = 0.2$). **B** Factors found by LANNIA when solving BP with the noisy images ($p = 0.7$, $q = 0.2$, $M = 800$). **C** Factors found by LM in the absence of noise in the images, $M = 800$. In **B** and **C**, the black pixels correspond to $p_{ij} = 1$, the white pixels correspond to $p_{ij} = 0$, and the gray pixels correspond to the intermediate values of p_{ij}.

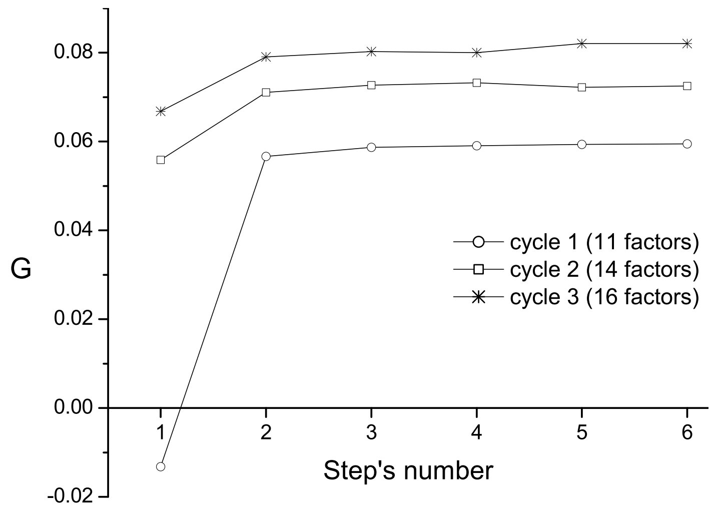


FIGURE 4.4: Increase of information gain at each cycle of LANNIA in solving the Bars Problem with noise (p = 0.7 and q = 0.2).

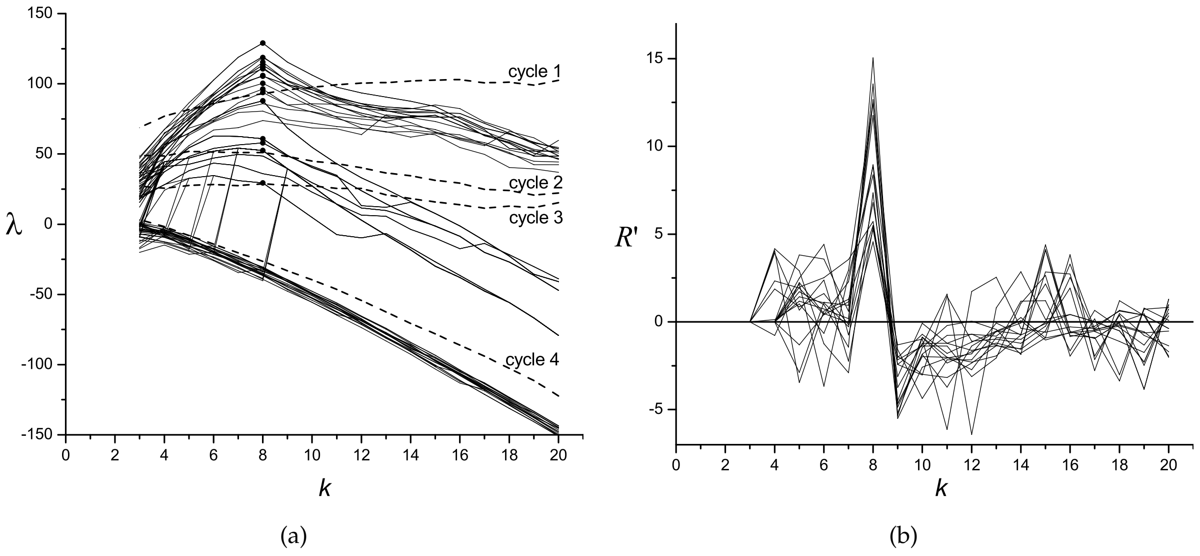


FIGURE 4.5: Lyapunov function λ (a) and function R' (b) in dependence on the number of active neurons k. Dashed lines in (a) are thresholds for separating true and spurious trajectories at four cycles of LANNIA. Results were obtained for dataset consisting of M = 800 noisy patterns (p = 0.7, q = 0.2).

Figure 4.4 demonstrates the dependence of the information gain G obtained in this procedure on the number of found factors. All factors were revealed for three full cycles. Twenty trajectories were run in ANNIA at the beginning of each LANNIA cycle. At the first cycle 11 trajectories were identified as true (Fig. 4.5(a)). These trajectories were continuous and had the values of the Lyapunov function at the peaks of R' (Fig. 4.5(b)) exceeding the separation border. The increment of the information gain provided by LM is the largest at the first cycle and amounted to 0.06.

After the factors found were unlearned from ANNIA at the end of the first LANNIA cycle, ANNIA found three factors at the second cycle. The increment of the gain amounted to 0.015, that is proportional to the number of the found factors. At the third full LANNIA cycle, ANNIA found two factors and the gain increased for about 0.01. After unlearning the factors found at the third cycle, all trajectories at ANNIA happened to be spurious and the procedure was terminated.

The factors found by LANNIA are shown in Fig. 4.3(**B**). They almost coincide with bars shown in Fig. 2.1(**A**) but contain also pixels with small probabilities p_{ij}. Nevertheless, the gain obtained by LANNIA that amounts to 0.081 was the same as for the information gain calculated for the precise generative model parameters.

The role of ANNIA in the hybrid procedure is illustrated in Fig. 4.3(**C**). There are shown the factors found by LM alone when it started from random initial scores, that is, without the approximate solution provided by ANNIA. The analyzed dataset contained $M = 800$ images without noise similar to that shown in Fig. 2.1(**B**). For each of 16 factors, the number of initial unitary scores was the same as in the dataset but they were chosen randomly, i.e., independently of the presence of the relating factors in the images. LM found only 8 factors and consequently the gain amounts to only $G = 0.43$ that is almost twice smaller than "theoretical" gain $G = 0.82$ for the not noisy data that is calculated for the precise generative model parameters. The gain achieved by LANNIA is equal to "theoretical" one, that is, it provides the perfect bars problem solution (it is illustrated in Fig. 6.1(a)).

4.4 The version of the method without likelihood maximization – BANNIA

In order to illustrate the role of LM procedure in LANNIA, we also tested the version of the method in which LM procedure is replaced by simple adaptive Bayesian Classifier (BC). This version is referred to as BANNIA (BC + LANNIA) and was introduced in [33]. BANNIA is very similar to LANNIA with two distinctions. The first distinction concerns the estimation of factor scores. In BANNIA it is performed by the adaptive Bayesian Classifier after all factors have been found by ANNIA. On the contrary, in LANNIA the score estimation is performed by alternations of LM and ANNIA. BC implemented in BANNIA takes account of probabilities $p_{ij}^{\alpha} = P(x_{mj} = 1|s_{mi} = \alpha)$ that j-th component of observation $\mathbf{x}_m$ takes 1 when observation contains ($\alpha = 1$) or does not contain ($\alpha = 0$) i-th factor. According to the Bayesian classification approach, the value s_{mi} providing the greater value

$$\Lambda_{mi}|_{s_{mi}} = \sum_{j=1}^{N}[x_{mj}\log p_{ij}^{\alpha} + (1 - x_{mj})\log(1 - p_{ij}^{\alpha})] + \alpha\log\pi_i + (1-\alpha)\log(1-\pi_i),$$

where $\alpha = s_{mi}$ is chosen to decide whether observation $\mathbf{x}_m$ contains or does not contain i-th factor. Then, the next observation $\mathbf{x}_{m+1}$ is processed. When all observations of the dataset are classified into two groups, the first one containing and the second one, not containing the i-th factor; probabilities p^1_{ij} and p^0_{ij} are estimated as frequencies of the j-th attribute appearance in patterns containing and not containing i-th factor, respectively. Then, the procedure is repeated with corrected values of p^1_{ij} and p^0_{ij} until convergence. After that the procedure goes to process the next factor. Thus, in fact, BC in BANNIA, in contrast to LANNIA, provides the search of factor scores for a given factor ignoring the specific combination of other factors in each particular observation of the dataset.

The second distinction concerns the suppression of the dominant attractors. In BANNIA it is performed by subtraction $\Delta\mathbf{J}^i$ from the matrix of synaptic connections $\mathbf{J}$ for each i-th found factor. Matrix $\Delta\mathbf{J}^i$ is defined as

$$\Delta J^i_{jk} = 0.5\lambda^i[(x^i_j(t)-a^i_f)(x^i_k(t+1)-a^i_f) + (x^i_j(t+1)-a^i_f)(x^i_k(t)-a^i_f)],\ \Delta J^i_{jj} = 0, \qquad (4.14)$$

where $\mathbf{x}^i(t)$ and $\mathbf{x}^i(t+1)$ are successive patterns of the network activity in the attractor corresponding to i-th factor, a^i_f is the portion of active neurons in the factor and λ^i is the value of the Lyapunov function in the attractor. In contrast to LANNIA (compare (4.14) and (4.13)), BANNIA ignores the fact that different factor's components can have different probabilities p_{ij} to appear in patterns of a dataset containing the factor. Therefore, the factor can be excluded from the network dynamics only partially, and so it can be found again in the form of its fragment or duplicate. BC used in BANNIA does not provide the exclusion of such fragments or duplicates from the list of found factors while LANNIA does. The presence of false factors of this kind in the list of found factors is the main disadvantage of BANNIA, compared with LANNIA.

Since BANNIA does not provide an estimate of generative models parameters required for gain computing, they are estimated by the M-step of the LM procedure applied to factor scores provided by BANNIA.

Chapter 5

Properties of attractor neural network with increasing activity

The proposed attractor neural network with increasing activity differs from common Hopfield network by sparse coding and original two-run network dynamics. The properties of this network not investigated yet. It is well known for common Hopfield network that network dynamics can converge to true or spurious attractors. True attractors in BFA notation correspond to one of the factors, spurious attractors are far from all factors. The ability to distinguish between them and the probability of the convergence to true attractors are key features of the network. In this chapter, we investigate the influence of the parameters of the neural network on the ability of revealing all true attractors. A more detailed description of the investigation presented in this chapter can be found in [28, 29, 43, 44].

In this chapter, each pattern of the training set is supposed to be generated in the form $\mathbf{x}_m = \bigvee_{i=1}^{L} s_{mi} \wedge \mathbf{f}_i$, where $\mathbf{x}_m$, $m = 1,\dots,M$, is a binary row vector of dimension N, $\mathbf{f}_i \in B_n^N$[1], $i = 1,\dots,L$, is a binary row vector of factor loadings of dimension N (hereafter, it is also referred to as factor), and $s_{mi} \in B_C^L$ is a binary factor score of i-th factor in m-th pattern. It means that in the generative model (2.5) noise is absent ($p_{ij} = f_{ij}$ and $q_j = 0$), the number of factors mixed in one pattern is fixed and is equal to parameter $C = \sum_{i=1}^{L} s_{mi}$ called as a signal complexity, the number of attributes constituting the factor (i.e., for attributes with $f_{ij} = 1$) is fixed and is equal to $n = pN = \sum_{j=1}^{N} f_{ij}$ where parameter p specifies the level of sparseness of the factors encoding (for other attributes not constituting the factor $f_{ij} = 0$).

[1] $B_n^N = \{\mathbf{x} | x_i \in \{0,1\}, \sum_{i=1}^{N} x_i = n\}$

The properties of the network are investigated in dependence on the five parameters: N, M, L, C, p. The size of the training set M should be large enough so that each factor could be presented several times in combinations with different other factors, i.e., $MC/L \gg 1$, however we put $C/L \ll 1$. Additionally we put $L \gg 1$ and $N \gg 1$, $p \ll 1$.

5.1 Lyapunov function of true attractors

By definition, the Lyapunov function of each attractor can be estimated as a mean synaptic excitation produced in the neurons of attractor by their proper activity. When $r < p$ true attractors are created by fragments of factors. Thus to evaluate the Lyapunov function for true attractors we have to evaluate a synaptic excitation produced in the neurons of factors by activation of their fragments. Along the trajectories of network dynamics those fragments create attractors which neurons are the most tightly connected among the neurons of factors. However, we ignore the specificity of activated fragments and calculate synaptic excitation produced by the fragment as if its neurons were chosen from factor's neurons randomly with equal probabilities. Therefore we obtain slightly underestimated values of the Lyapunov function of true attractors.

Without any loss of generality, we may assume that a fragment of factor $\mathbf{f}_1$ is activated. In conformity with [16], we call n and $N - n$ neurons of $\mathbf{f}_1$ with states One and Zero as "high" and "low" neurons, respectively. According to (4.1), (4.3), (4.4) and (4.6), when $\mathbf{x}^{\text{in}}$ is activated in the network, the neurons synaptic excitations can be presented in the form $h_i = \Sigma_i^1 + \Sigma_i^0 - \Sigma_i'$ where

$$\Sigma_i^\mu = \sum_{m \in \{m : s_{m1} = \mu\}} (x_{mi} - a_m) \sum_{j \neq i} (x_{mj} - a_m) x_j^{\text{in}}.$$

Two first sums $\Sigma_i^1 + \Sigma_i^0$ is a contribution of principal network neurons into synaptic excitations and the third sum

$$\Sigma_i' = M(b_i - \bar{b}) \sum_j (b_j - \bar{b}) x_j^{\text{in}}$$

is the inhibition produced by the additional neuron. The sum Σ_i^1 contains M^1 patterns of the learning set which include the recalled factor $\mathbf{f}_1$ and Σ_i^0 contains M^0 patterns which do not include it. Since the variance of a_m is of the order $1/N$ (see Appendix A), then standard deviation of a_m is small compared to its mean and one can put

$$a_m \simeq \langle a_m \rangle = \bar{a} = 1 - (1 - p)^C \simeq 1 - \exp(-pC) \tag{5.1}$$

where $\langle \cdots \rangle$ means averaging over all factor scores and all factors except $\mathbf{f}_1$.

The activated fragment of $\mathbf{f}_1$ contains $n_1 = rN$ high neurons and no low neurons of $\mathbf{f}_1$. However, we consider now more general cases when the pattern $\mathbf{x}^{\text{in}}$ contains additionally n_0 low neurons of $\mathbf{f}_1$. The formula for this case will be used later in Section 5.3.1. For high neurons of $\mathbf{f}_1$

$$\langle \Sigma_i^1 \rangle = \langle M^1 \rangle [(1-\bar{a})^2 n_1 + (1-\bar{a})(\bar{a}'-\bar{a})n_0]$$

and for low neurons

$$\langle \Sigma_i^1 \rangle = \langle M^1 \rangle [(a'-\bar{a})(1-\bar{a})n_1 + (a'-\bar{a})^2 n_0]$$

where

$$a' = 1-(1-p)^{C-1} = 1-(1-\bar{a})/(1-p)$$

is the probability of the neuron to be active in the pattern of the learning set due to the presence of other factors except $\mathbf{f}_1$. Therefore,

$$\langle \Sigma_i^1 \rangle = \langle M^1 \rangle (1-\bar{a})^2 (f_i^1 - p)[n_1 - p(n_1+n_0)]/(1-p)^2$$

for both high and low neurons.

Since $\mathbf{x}^{\text{in}}$ is independent of all factors except $\mathbf{f}_1$, then

$$\langle \Sigma_i^0 \rangle = \langle M^0 \rangle (n_1+n_0) \langle (x_{mi}-a_m)(x_{mj}-a_m) \rangle .$$

By the definition of a_m, for each pattern of the learning set $\sum_{i=1}^{N}(x_{mi}-a_m) = 0$. Hence

$$\langle (x_{mi}-a_m)(x_{mj}-a_m) \rangle = -\langle (x_{mi}-a_m)^2 \rangle/(N-1) \simeq -\bar{a}(1-\bar{a})/N \tag{5.2}$$

and $\langle \Sigma_i^0 \rangle = -\langle M^0 \rangle \bar{a}(1-\bar{a})[n_1+n_0]/N$.

Since $\langle (b_i - \bar{b}) \rangle$ is evidently equal to 0, then $\langle \Sigma_i' \rangle = 0$ and the mean synaptic excitation amounts to

$$\begin{aligned} \langle h_i \rangle &= \langle M^1 \rangle (1-\bar{a})^2 (f_{1i}-p)[n_1 - p(n_1+n_0)]/(1-p)^2 \\ &- \langle M^0 \rangle \bar{a}(1-\bar{a})[n_1+n_0]/N. \end{aligned} \tag{5.3}$$

For activated fragment of $\mathbf{f}_1$ $n_1 = rN$ and $n_0 = 0$ then its mean Lyapunov function is equal to

$$\lambda_{\text{true}} = [\langle M^1 \rangle (1-\bar{a})^2 - \langle M^0 \rangle \bar{a}(1-\bar{a})/N] rN.$$

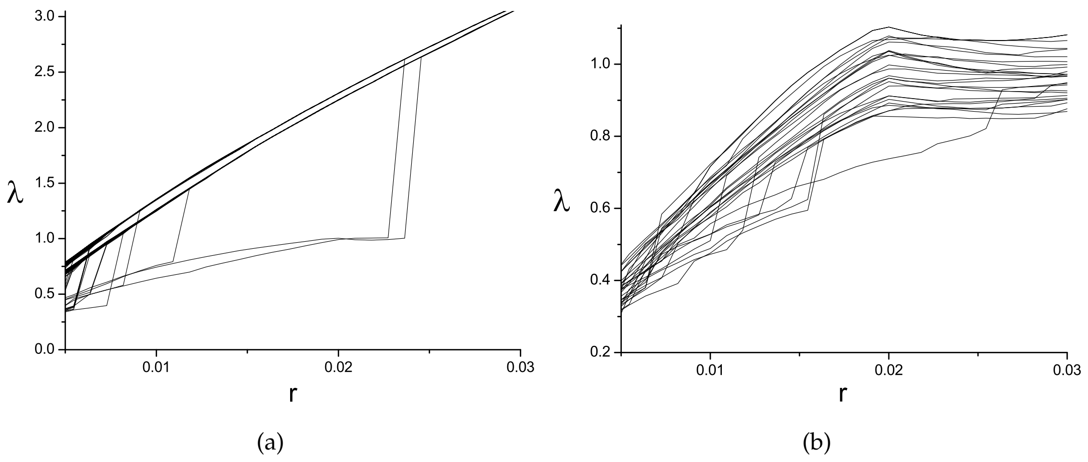


FIGURE 5.1: Lyapunov function λ dependent on the relative network activity $r = k/N$, when special inhibitory neuron is excluded from (a) and included in (b) the network. Lyapunov function is normalized by its mean value for factors, i.e., at the point $r = p = 0.02$.

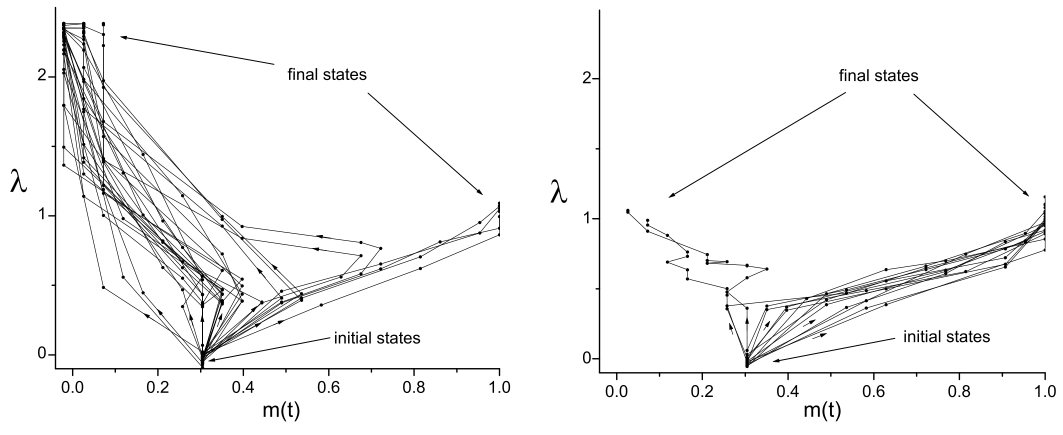
The probability of factor to be mixed in the learning pattern is C/L. Thus $\langle M^1 \rangle = MC/L$ and $\langle M^0 \rangle = M(1 - C/L) \simeq M$ and the first term in the expression for λ_{true} is of the order MC/L and the second - M/N. Usually $CN/L \gg 1$ and the second term can be neglected. Then in the range $r \leq p$ the mean Lyapunov function for true attractor can be estimated as

$$\lambda_{\text{true}} = MCrN(1 - \bar{a})^2/L \tag{5.4}$$

5.2 Global spurious attractors

In this section we investigate the nature of the global spurious attractors and the performance of the of the additional inhibitory neuron that ensures the eliminations of the global spurious attractors from the network dynamics. After that we investigate why the global spurious attractors were not observed previously for ordinary Hopfield network used, for example, as associative memory. For this purpose we estimate the Lyapunov function of the global spurious attractors.

In order to clarify the performance of the additional inhibitory neuron, computer simulations were made. Figure 5.1 demonstrates trajectories of the network dynamics obtained without (a) and with (b) the use of additional inhibitory neuron. For the presented example, $L = 800$, $N = 1100$, $M = 40000, C = 20, p = 0.02$. Fifty trajectories are shown in each figure. Trajectories started from random states with $k_{\text{in}} = 5$ active neurons and continued to $k_{\text{fin}} = 33$. If the inhibitory neuron is not included (Fig. 5.1(a)), only two trajectories with a high Lyapunov function dominate and attract all other trajectories. As demonstrated in Fig. 5.2(a) the trajectories with a high Lyapunov function are spurious. The trajectories in Fig. 5.2 are obtained for



(a) (b)

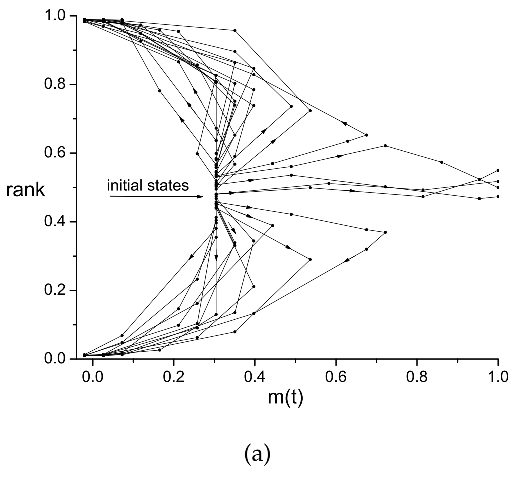
FIGURE 5.2: Trajectories of neurodynamics when special inhibitory neuron is excluded from (a) and included in (b) the network. Abscissa is an overlap $m(t)$ between the current network activity and the recalled factor, ordinate is Lyapunov function normalized as in Fig. 5.1.

the case when the number of active neurons is fixed and equal to the number ones in factors, i.e., $k = n = 22$. The trajectories are shown in the plane constituted by axes $[m(t), \lambda(t)]$ where t is the time step, $\lambda(t)$ is the Lyapunov function, $m(t)$ is the overlap of the current network state $\mathbf{x}(t)$ with a recalled factor $\mathbf{f}$ (nearest factor) defined as

$$m(t) = m(\mathbf{x}(t), \mathbf{f}) = \frac{1}{Np(1-p)} \sum_{i=1}^{N} (x_i(t) - p)(f_i - p).$$

According to this formula, the overlap is equal to 1 if $\mathbf{x}(t)$ is identical to $\mathbf{f}$ and is equal to 0 in average if $\mathbf{x}(t)$ is independent of $\mathbf{f}$. The trajectories are started from random states with $m(0) = m_{\text{in}} = 0.3$. When additional inhibitory neuron does not perform (Fig. 5.2(a)), only small portion of trajectories converge to factors, i.e., to the final states with $m = 1$. The most trajectories converge to spurious states with high values of the Lyapunov function and m is close to 0. Attractors corresponding to these spurious states are called as global spurious attractors. When additional inhibitory neuron performs (Fig. 5.2(b)), the most trajectories converge to factors while spurious attractors are characterized by relatively small values of the Lyapunov function.

The global spurious attractors are two attractors created by neurons contained in the highest and in the lowest numbers of factors, respectively. To demonstrate this fact we redrew in Fig. 5.3 the trajectories in the axes $[m(t), \text{rank}(t)]$ where $\text{rank}(t)$ indicates whether the neurons contained in the most or least numbers of factors contribute to the current network activity. To calculate $\text{rank}(t)$, all neurons were ordered by the number of factors that contained them, so the neurons contained in the smallest number of factors had the lowest rank, and those contained in the largest number of factors had the highest rank. The rank of the current activity



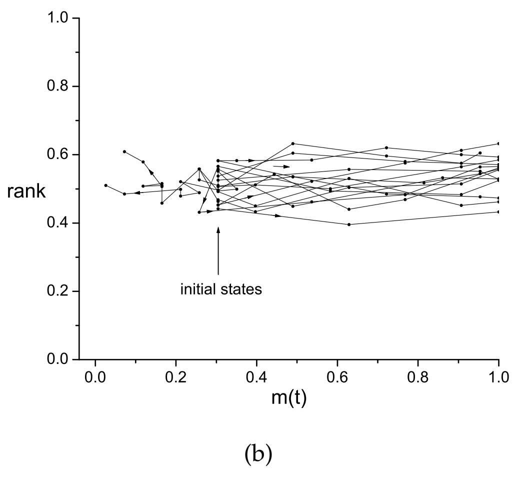


FIGURE 5.3: The same trajectories as in Fig. 5.2 drawn in the plane $(m(t), \mathrm{rank}(t))$, where rank is an index showing contribution of neurons most and least often contained in factors set. (a)– inhibitory neuron is excluded, (b)– included.

was calculated as the sum of the ranks of active neurons. The obtained rank was normalized so that the patterns created by the neurons contained in the smallest number of factors have the rank close to zero, whereas those created by the neurons contained in the largest number of factors have the rank close to one, and those created by random neurons have the rank close to 0.5. Figure 5.3(a) demonstrates that the patterns created by the global spurious attractors have ranks close to 0 and 1, while true attractors have ranks around 0.5. When two global spurious attractors completely suppressed (Fig. 5.3(b)), trajectories that converge to states far from the factors are attracted by local spurious attractors. These attractors have ranks close to 0.5 and their Lyapunov function value is smaller than that for global spurious attractors (Fig. 5.2(b)).

Two global spurious attractors dominate because their Lyapunov function exceeds that of true attractors. To estimate the value of Lyapunov function for global spurious attractors, we take into account that they are created by neurons most and least often contained in the whole set of factors. We also take into account that $p \ll 1$ and $r \ll 1$.

Let k be the number of factors containing a given neuron. The mean and variance of k are both equal to pL. In Gaussian approximation the number of neurons which belong to $k > k_1$ factors can be estimated as $N\Phi(u_1)$ where $u_1 = (k_1 - pL)/\sqrt{pL}$. To choose rN neurons with the largest k from totally N neurons, one must put $k_1 = pL + u_1\sqrt{pL}$ where u_1 satisfies equation $\Phi(u_1) = r$. On average, each of the chosen neurons belongs to $k_2 = pL + u_2\sqrt{pL}$ patterns where

$$u_2 = \frac{1}{\Phi(u_1)\sqrt{2\pi}} \int_{u_1}^{\infty} u \exp(-u^2/2) \mathrm{d}u.$$

The probability of one of these neurons to be active during the presentation of a learning pattern is $a'' \simeq 1 - \exp(-k_2 C/L) \simeq \bar{a} + (1-\bar{a})Cu_2\sqrt{p/L}$ where $\bar{a} = 1-(1-p)^C$. Then a mean augmentation of synaptic connection between two of these neurons during the presentation of input pattern is $\Delta J = (a'' - \bar{a})^2 = (1-\bar{a})^2 C^2 u_2^2 p/L$ and a mean strength of connection after presentation of the whole learning set is $J = M\Delta J = M(1-\bar{a})^2 C^2 u_2^2 p/L$. Hence the Lyapunov function for this attractor can be estimated as

$$\lambda_{\text{spur}}^{\text{gl}} \simeq JrN = MrN(1-\bar{a})^2 C^2 u_2^2 p/L.$$

Since $r \ll 1$ then $u_2 \simeq u_1 \simeq [2\ln(1/r)]^{\frac{1}{2}}$. Consequently

$$\lambda_{\text{spur}}^{\text{gl}} \simeq 2Mr\ln(1/r)N(1-\bar{a})^2 C^2 p/L \tag{5.5}$$

Similarly, it is easy to estimate the Lyapunov function for the attractor created by neurons belonging to the smallest number of factors. To do this, it is enough to replace k_2 in the formula for $\bar{a}''$ by $k_3 = pL - u_2\sqrt{pL}$ keeping all other equations. This results in the same expression for the Lyapunov function as (5.5).

According to (5.4) and (5.5) the Lyapunov function of true attractors increases proportionally to C while of spurious attractors proportionally to C^2. Their ratio amounts to

$$\lambda_{\text{spur}}^{\text{gl}}/\lambda_{\text{true}} \simeq 2Cp\ln(1/r). \tag{5.6}$$

Note that formula (5.5) is valid for the whole range of r while (5.4) is valid only for $r \leq p$. Thus comparison of values of the Lyapunov function for global spurious and true attractors by (5.6) is valid only for $r \leq p$.

Figure 5.4 demonstrates the ratio of Lyapunov functions for global spurious and true attractors in dependence on C obtained by computer simulation and by (5.6) for $r = p = 0.02$. It is shown that (5.6) gives a rather accurate estimation of this ratio. Thus the global spurious attractors become to dominate if $2Cp\ln(1/p) > 1$. For example, for $p = 0.02$ the critical complexity of their dominance amounts to $C \simeq 10$. According to (5.6), the critical complexity increases when sparseness increases (i.e. p decreases). Thus the global spurious attractors were not observed previously for ordinary Hopfield network used because for this network $C = 1$, and $\lambda_{\text{spur}}^{\text{gl}}/\lambda_{\text{true}} \ll 1$.

5.3 The size of factors attraction basins

The network dynamics converge to one of the factors only when the initial state falls inside its attraction basin. In other cases it converges to one of the spurious attractors. Since generally

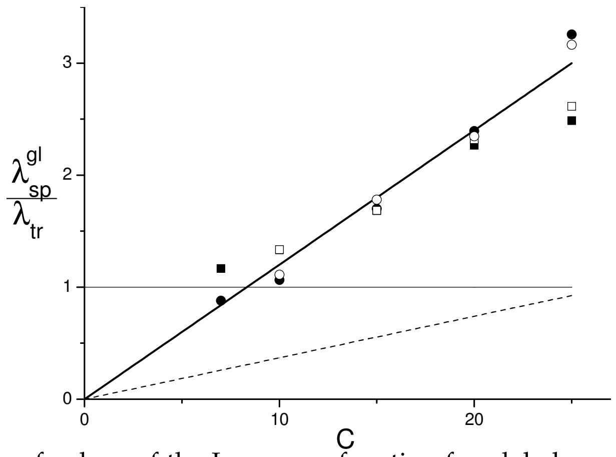


FIGURE 5.4: The ratio of values of the Lyapunov function for global spurious and true attractors. Estimations by (5.6) are shown by a thick solid line for $p = 0.02$, and a dashed line for $p = 0.004$. Points are experimental data averaged over 50 realizations of L random factors for $p = 0.02$, $M = 40000$, $N = 1100$: squares – $L = 800$, circles – $L = 1600$, open and full points – attractors created by neurons least and most often contained in factors, respectively. The thin horizontal line indicates the equality of values of the Lyapunov function for spurious and true attractors.

there is no a priori information on factors the initial network state can only fall into a factor attraction basin by chance. Therefore the ability of a Hopfield-like network to perform factor searches is determined by the probability that network activity converges to one of the factors starting from a random state. And the estimation of parameters significant for the size of attraction basins is required.

In this section, the size of attraction basins around factors is estimated for the case of fixed level of activity of the neural network during network dynamics: the number of active neurons is supposed to be constant and equal to the number of ones in factor, i.e., $k = n = \sum_{j=1}^{N} f_{ij}$. The size of attraction basins in the case of variable level of the neural network activity is estimated in Section 5.4.

5.3.1 Attraction basins around factors in single-step approximation

Single-step (SS) approximation has been proposed by [67] for the densely encoded Hopfield network. It has been shown by other theoretical approaches [5, 6] and by Monte-Carlo simulations [6, 49, 68] that SS approximation is very inaccurate for dense coding. However, it becomes quite accurate when sparseness increases [26]. The principal peculiarity of this approach is that at each time step one ignores the statistical dependence between the network activity and the connection matrix and takes into account only two macroparameters of neurodynamics: the overlap $m(t)$ between the current and recalled patterns and the total network activity. If the analysis is restricted to the case of fixed level of the relative network activity

$r = p$, then the recall process is described by the evolution of only one parameter $m(t)$. Omission of the statistical dependence between the network activity and the connection matrix is possible only for the first step, when the initial activity is actually stated independently of the connection matrix. This is why this approximation is called the "single-step" or "first-step" approximation.

The mean synaptic excitation to high and low neurons of any factor (for example for factor $\mathbf{f}_1$) is given by (5.3) where for the case under consideration $n_1 = N[m(t)p(1-p)+p^2]$, $n_0 = Np - n_1$. Thus

$$\langle h_i \rangle = \frac{MNCp(1-q)^2}{L(1-p)m(t)}(f_i^1 - p) - Mpq(1-q). \tag{5.7}$$

The mean synaptic excitation for high neurons (for them $f_{1j} = 1$) is larger than that for low neurons (for them $f_{1j} = 0$). Thus the high neurons have higher probability to be activated at the next step of the recall procedure.

To estimate the probabilities of high and low neurons to be active, let us now estimate the variance of synaptic excitations. Since $M^0 \gg M^1$, for the network without special inhibition neuron, it is determined by the variance of Σ_i^0. Therefore,

$$\mathrm{D}\{h_i\} = Np\mathrm{D}\{J'_{ij}\} + N^2p^2\mathrm{Cov}\{J'_{ij}, J'_{ik}\}, k \neq j \neq i \tag{5.8}$$

where $\mathbf{J}'$ is given by (4.1). As shown in Appendix B

$$\mathrm{D}\{J'_{ij}\} = \frac{M^2C^2p^2(1-q)^4G_1(\mu)}{L(1-p)^2} \tag{5.9}$$

where $\mu = C^2/L$ is a relative signal complexity,

$$G_1(\mu) = [\exp(\mu(\frac{1}{(1-p)^2} - 1)) - 2\exp(\mu(\frac{1}{1-p} - 1)) + 1](1-p)^2/(\mu p^2). \tag{5.10}$$

As shown in Fig. 5.5 in the range $0 < \mu < 1$ the graph of this function is close to the straight line and tends to the line $G = 1 + \mu$ when sparseness increases.

When the special inhibitory neuron is included to the network, the variance of synaptic excitation is determined by the variance of $\Sigma_i^0 - \Sigma_i'$. Therefore,

$$\mathrm{D}\{h_i\} = Np\mathrm{D}\{J_{ij}\} + N^2p^2\mathrm{Cov}\{J_{ij}, J_{ik}\},\ k \neq j \neq i$$

where $\mathbf{J}$ is given by (4.4). As shown in Appendix C

$$\mathrm{D}\{J_{ij}\} = \frac{M^2C^2p^2(1-q)^4G_2(\mu)}{L(1-p)^2} \tag{5.11}$$

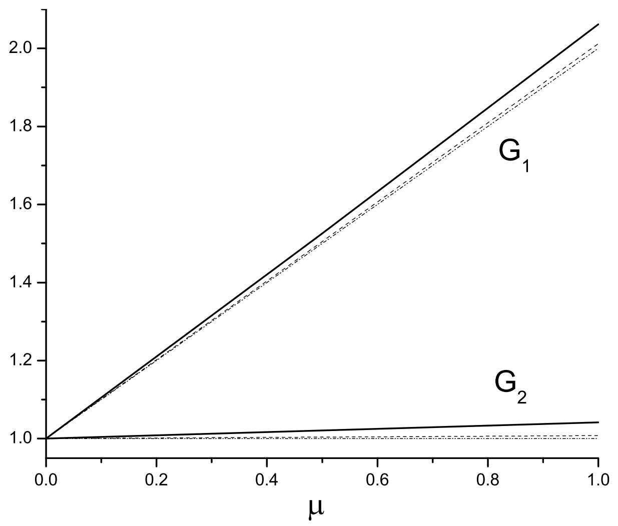


FIGURE 5.5: The Functions $G_1(\mu)$ and $G_2(\mu)$ for the case without and with additional inhibition, respectively. $\mu = C^2/L$ is a relative index of signal complexity. $p = 0.02$ – solid line, $p = 0.004$ – dashed line, $p \to 0$ – dashed-dotted line.

where

$$G_2(\mu) = [\exp(\mu(\frac{1}{(1-p)^2} - 1)) - \exp(2\mu(\frac{1}{1-p} - 1))](1-p)^2/(\mu p^2). \tag{5.12}$$

Function $G_2(\mu)$ is compared with $G_1(\mu)$ in Fig. 5.5. It is is shown that due to the inhibition by the special inhibitory neuron, the variance of synaptic connections becomes only slightly dependent on μ and G_2 tends to the line $G_2 = 1$ when sparseness increases.

To estimate $\mathrm{Cov}\{J'_{ij}, J'_{ik}\}$ one can notice that according to the learning rule (4.1)

$$\sum_{j=1,j\neq i}^{N} J'_{ij} = -\sum_{m=1}^{M} (x_{mi} - a_m)^2.$$

Thus

$$(N-1)\mathrm{D}\{J'_{ij}\} + (N-1)(N-2)\mathrm{Cov}\{J'_{ij}, J'_{ik}\} =$$
$$-M\mathrm{D}\{(x_{mi} - a_m)^2\} - M(M-1)\mathrm{Cov}\{(x_{mi} - a_m)^2, (x_{li} - a_l)^2\}.$$

Both terms on the right side of this equation, which are respectively of order M and M^2, can be neglected when compared with the first term on the left side of this equation that is of order NM^2. Hence $\mathrm{Cov}\{J'_{ij}, J'_{ik}\} = -\mathrm{D}\{J'_{ij}\}/N$. The same is valid for $\mathrm{Cov}\{J_{ij}, J_{ik}\}$. Then according to (5.8), (5.9) and (5.11)

$$\mathrm{D}\{h_i\} = \sigma^2 = Np(1-p)\frac{M^2C^2p^2(1-q)^4G_\varepsilon(\mu)}{L(1-p)^2} \tag{5.13}$$

where ε is 1 for the network without inhibition and 2 with inhibition.

In the limit case $N \to \infty$, the distributions of synaptic excitations can be approximated by normal ones. Then at the first step of the recall process

$$\mathrm{Prob}\{x_i(1) = 1\} = \Phi(\theta_i)$$

where

$$\theta_i = (T - \langle h_i \rangle)/\sigma,$$

$$\Phi(x) = 1/(2\pi)^{1/2} \int_x^\infty \exp(-u^2/2)\mathrm{d}u \tag{5.14}$$

and T is an activation threshold. According to (5.3) and (5.13) for high and low neurons of $\mathbf{f}_1$

$$\theta^1 = \theta - \frac{m(t)(1-p)}{\sqrt{Lp(1-p)G_\varepsilon(\mu)/N}} \qquad \theta^0 \quad = \quad \theta + \frac{m(t)(1-p)}{\sqrt{Lp(1-p)G_\varepsilon(\mu)/N}}$$

where $\theta = (T + Mpq(1-q))/\sigma$ is a scaled activation threshold. In the model the threshold is chosen in such a way that a total level of the network activity is the same as in factors, i.e. is chosen to satisfy condition

$$pp_1 + (1-p)p_0 = p$$

where $p_1 = \Phi(\theta_1)$ and $p_0 = \Phi(\theta_0)$ are probabilities for high and low neurons to be active. As a result, the overlap changes to

$$m(t+1) = p_1 - p_0.$$

These equations together with initial condition $m(0) = m_{\mathrm{in}}$ completely determine the evolution of network activity which depends on only parameters p and parameter $\gamma = \alpha G_\varepsilon$ that is a combined index of the relative signal complexity μ and of the relative informational loading $\alpha = Lh(p)/N$ where $h(x)$ is the Shannon function.

The curves which characterize the behavior of the network activity are presented in Fig. 5.6. The curves are drawn in the plane $(m_{\mathrm{in}}, \gamma)$. Each curve corresponds to some fixed p. Let the initial state of the network activity for a given α be characterized by the point $(m_{\mathrm{in}}, \gamma)$. If this point is under the curve, the overlap between the current pattern and the recalled pattern moves during the recall process to the right, that is to the final overlap m_{fin} given by the right branch of the curve. The overlap moves to the left for each point above the curve. Maximum at the curve $\gamma_{\max}$ defines the critical information loading $\alpha_{\mathrm{cr}} = \gamma_{\max}/G(\mu)$, when factors cease to be attractors of the network dynamics. If $\gamma > \gamma_{\max}$, then network dynamics moves to some

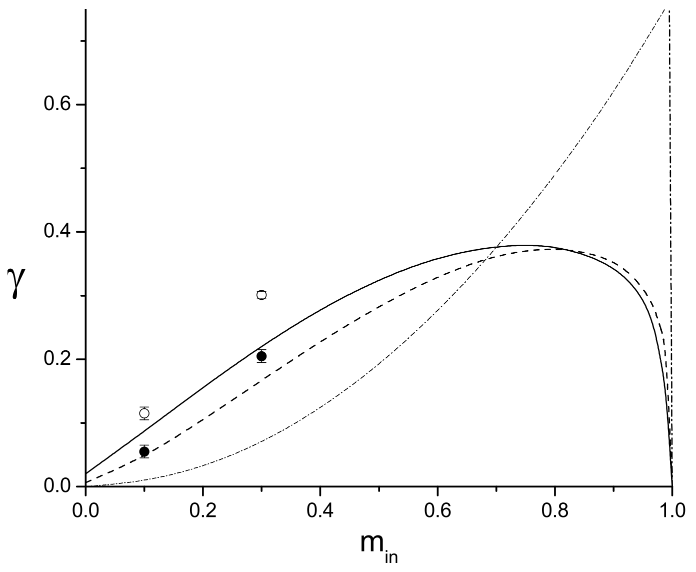


FIGURE 5.6: Sizes of attraction basins m_{in} in dependence on relative informational loading α and signals complexity μ ($\gamma = \alpha G_\varepsilon(\mu)$). Lines - Single-step approximation: $p = 0.02$ - solid line, $p = 0.004$ - dashed line, $p \to 0$ - dashed-dotted line. Points - results of multi-step recall obtained by computer simulation in the case when special inhibitory neuron is included: $\circ$ – $p = 0.02$, $\bullet$ – $p = 0.004$.

spurious state ($m_{\text{fin}} = 0$), even starting from factor ($m_{\text{in}} = 1$). If $\gamma < \gamma_{\max}$ and $m_{\text{in}} = 1$, then network dynamics converges to some attractor in the vicinity of the factor with m_{fin} defined by right branch of the curve. Thus, the left branch of the curve corresponds to the border of an attraction basin and the right branch defines the distance between factor and attractor in its vicinity. In the SS approximation the border of the attraction basin corresponds to the condition $m(1) = m_{\text{in}}$.

Figure 5.6 demonstrates that the size of attraction basin decreases, when γ increases. The increase of γ can be provided by the increase of the relative informational loading α and by the increase of the function G_ε. Since $G_2 < G_1$, the inclusion of the special inhibitory neuron, in addition to suppression of global spurious attractors, provides an essential increase of attraction basins around factors due to decreasing of γ. On the other hand, the size of the attraction basins monotonically decreases when the encoding sparseness increases under the fixed γ. As shown by [26], the dynamics of the ordinary Hopfield network is determined by the same equations as described above if $G_\varepsilon = 1$. Since $G_2 \simeq 1$ the network dynamics with special inhibition almost coincides with that of an ordinary Hopfield network under the Single-Step approximation.

5.3.2 Attraction basins around factors in multi-step recall

Figures 5.2 and 5.3 demonstrate that as for the ordinary Hopfield network, the borders of the attraction basins around factors are fuzzy: starting from the states with the same m_{in}, the trajectories may converge to the recalled factor or to some spurious state far from all factors. Consequently, the distribution of final overlaps has two distinct modes: $m_{\text{fin}} \approx 1$ ("true") and $m_{\text{fin}} \ll 1$ ("spurious"). It is well known for the ordinary Hopfield network that for small informational loading "true" mode prevails, and as informational loading increases, the distribution maximum shifts to "false" mode, demonstrating a sharp transition from a retrieval to a not-retrieval network dynamics at a certain $\alpha = \alpha_{\text{cr}}$. The transition becomes more sharp when network size increases. Let us consider probability P that starting from some initial state with a given initial overlap m_{in} with a factor, the network activity converges to it. For the ordinary Hopfield network (i.e., when $C = 1$) P depends on m_{in}, α and N. For the network performing Boolean factor analysis (i.e., when $C > 1$) P additionally depends on C and on the size of the learning set M.

For each set of parameters the probability P was estimated as a portion of true trajectories at the histogram of m_{fin} distribution. In order to separate true and spurious modes at the histogram, the border $m_{\text{fin}} = 0.7$ was used. Since the true and spurious modes were always well separated, an exact choice of the border was not important. The computed values of P were transformed by the following logistic mapping to variable F:

$$P = \frac{1}{1 + e^{-F}} \tag{5.15}$$

The advantage of this transformation is that F is unlimited, whereas $0 \leq P \leq 1$. Thus, to approximate F by some regression model it is not required to use any constrains to F as it would be required for direct approximation of P.

As an example, the dependence of F on M is shown in Fig. 5.7 for $N = 3000$, $C = 20$ $m_{\text{in}} = 0.3$ and different α. Data were obtained with the use of the special inhibitory neuron. As shown in Fig. 5.7, F can be well fitted by linear dependence on L/CM. Intercepts of the regression lines that approximate the dependence of F on L/CM were used to estimate the asymptotic values of F for $M \to \infty$.

Figure 5.8 presents the dependence of F on N and α for $m_{\text{in}} = 0.3$ and $C = 20$ and $C = 1$. The results for $C = 1$ and $C = 20$ are close, although the ability to recall stored factors happened to be even higher for $C = 20$ than for the ordinary Hopfield network (especially when informational loading α increases). One of the possible explanations is that storing complex patterns produces noise in the connection matrix, and this noise suppresses some local spurious attractors of the ordinary Hopfield network.

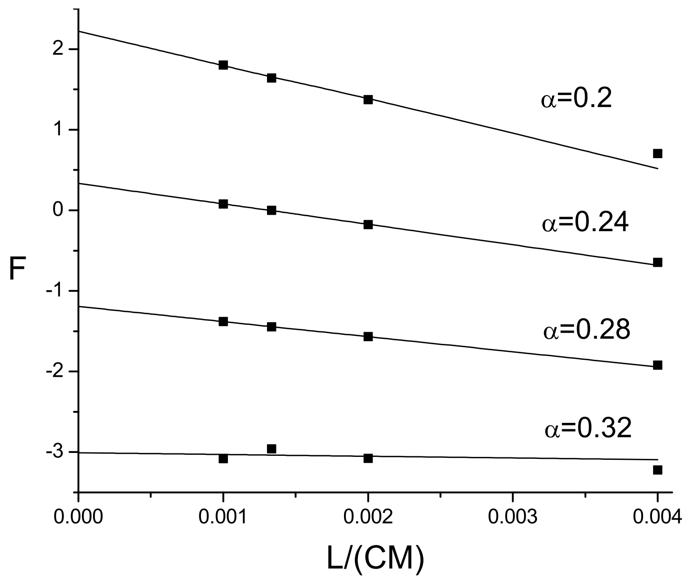


FIGURE 5.7: Transformed probability F of trajectories convergence to factors in dependence on $L/(CM)$ for $N = 3000$, $C = 20$ and $m_{\text{in}} = 0.3$. Intercepts with ordinate axes were used as experimental estimations of transformed probability for $M \to \infty$.

Since the data happened to be close for $C = 20$ and $C = 1$, they were combined in one family of data for approximation by the regression model

$$F = a_0 + a_1\alpha + a_2 N + a_3 \ln N + a_4 \alpha N. \tag{5.16}$$

The fitted curves that approximate the data for fixed α are shown in Fig. 5.8 by thin lines. According to this approximation, the lines constitute two groups. The upper lines are concave and tend to $+\infty$ when $N \to \infty$. The lower line is convex and tends to $-\infty$ when $N \to \infty$. Transition from one to another group occurs due to the change of α. The value of α which corresponds to the thick dashed line separating these groups is chosen as critical α_{cr}. For each $\alpha < \alpha_{\text{cr}}$ the probability of trajectories to converge to factors tends to 1, when N increases. And conversely for $\alpha > \alpha_{\text{cr}}$ it tends to zero. Thus α_{cr} corresponds to sharp transition from retrieval to nonretrieval conditions for $N \to \infty$. As mentioned above for Single-step approximation α_{cr} is found from the condition $m(1) = m_{\text{in}}$.

From the regression model, α_{cr} can be found as $\alpha_{\text{cr}} = -a_2/a_4$. For data combined in a joint family $\alpha_{\text{cr}} = 0.307 \pm 0.006$. This value is shown in Fig. 5.8 by the thick dashed line. For $p = 0.02$ it corresponds to $L = 2.17N$. The regression model, applied to each of three sets of data separately, gives $\alpha_{\text{cr}} = 0.303 \pm 0.005$ for $C = 20$, $\alpha_{\text{cr}} = 0.307 \pm 0.008$ for the ordinary Hopfield network. All these values differ insignificantly. However, they significantly exceed the value $\alpha_{\text{cr}} = 0.22$, predicted by SS (see Fig. 5.6). Thus for sparse encoding ($p = 0.02$) the SS approximation underestimates the size of attraction basins. This is confirmed by the computer simulation performed for $p = 0.02$ and $m_{\text{in}} = 0.1$ and for $p = 0.004$, $m_{\text{in}} = 0.1$ and 0.3 for the ordinary

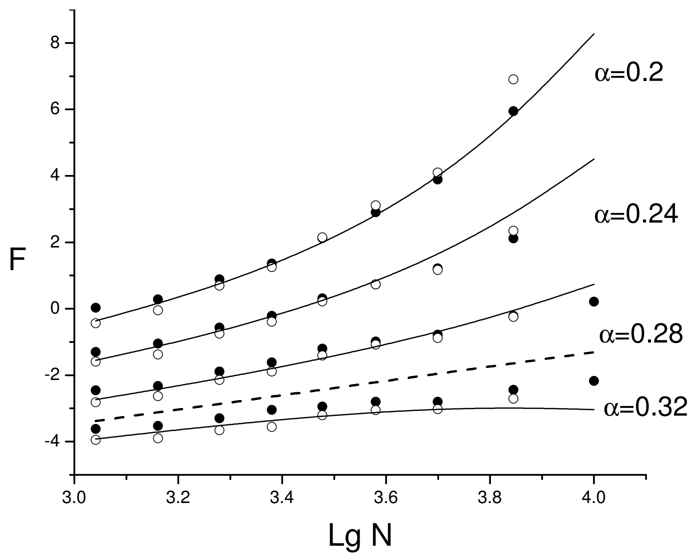


FIGURE 5.8: Transformed probabilities F of trajectories convergence to factors in dependence on the network size N and informational loading α for $C = 1$ (○) and $C = 20$ (●), $m_{\text{in}} = 0.3$. Thin lines correspond to regression model (5.16). The thick dash line corresponds to the critical value of informational loading $\alpha_{\text{cr}} = 0.303$.

Hopfield network. The obtained estimations are also shown in Fig. 5.6. When α is smaller than α_{cr} predicted by SS, then $m(t)$ increases monotonically according to SS assumption. But when α is larger than α_{cr} predicted by SS but smaller than α_{cr} obtained experimentally, then $m(t)$ changes non-monotonically: trajectories move away from the recalled factors at the first step but then return and terminate in their vicinities.

5.4 Probability of true trials during random searches

The dominance of spurious attractors that prevents activation of true attractors could disturb the ability of the method to reveal factors. There are two reasons why spurious attractors come to dominate when L increases: firstly, due to the increase of their Lyapunov function, and secondly, due to the increase of their number. It is well known (see, for example, [4, 6]) that the Lyapunov function of spurious attractors increases when relative loading L/N increases, and the number of spurious attractors increases exponentially when network size N increases. Thus one can expect the existence of two limits that restrict the network's ability to search for factors. One relates to the critical relative loading L/N and the other to the critical network size N under the fixed relative loading.

To find the first limit the values of the Lyapunov function for spurious attractors λ_{sp} depending on L/N when $N \to \infty$ were estimated. The values of λ_{sp} obtained by averaging across 10,000 spurious trials and normalized by mean values of the Lyapunov function over true

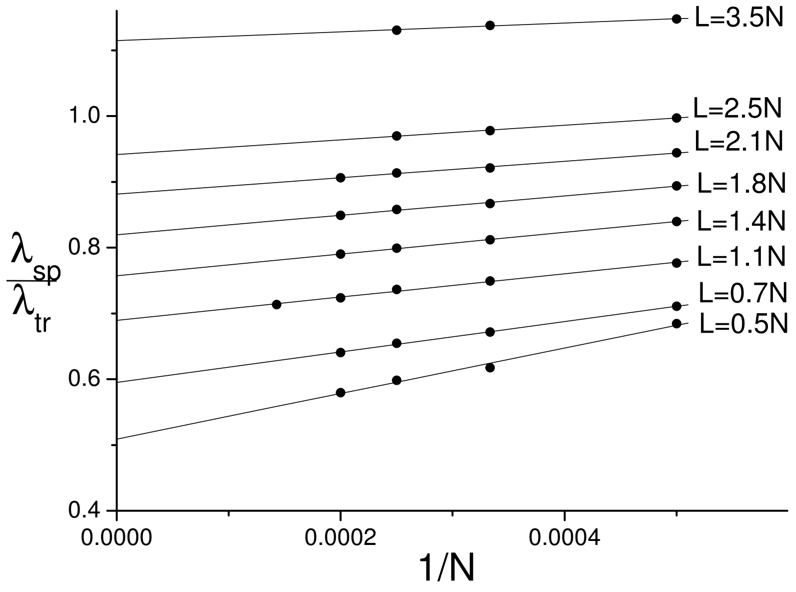


FIGURE 5.9: Lyapunov functions of spurious attractors normalized by mean values of this function over true attractors at the point $r = p$ in dependence on L/N and $1/N$ for $C = 1$. Each point was obtained as the average from over 10,000 trials. Experimental points are approximated by straight lines.

attractors λ_{tr} are shown in Fig. 5.9 for $r = p$ and $C = 1$. For each L/N ratio, experimental points were approximated by a linear regression function depending on $1/N$. The regression intercept was treated as estimations of the ratio $\lambda_{\text{sp}}/\lambda_{\text{tr}}$ for $N \to \infty$ under fixed L/N. The estimates obtained for $C = 1$, $C = 10$ and $C = 20$ are presented in Fig. 5.10. The Lyapunov function of spurious attractors reaches the Lyapunov function of true attractors at $L \simeq 2.8N$. Thus, the critical loading, L_1, when spurious attractors become dominant due to their large Lyapunov function, is approximately $L_1 = 2.8N$. This value is slightly higher than the critical number of factors $\alpha_{\text{cr}}N/h(p) = 2.17N$ obtained in Section 5.3.2.

To estimate the second limit the probability of transitions from spurious to true trajectories along the recall process were analyzed. As shown in Fig. 4.1(a), initially, when $r = r_{\text{in}}$ most trajectories start as spurious but many of them transform into true ones. Transitions from spurious to true trajectories may occur at any point during the recall process and the probability P_{spur} that the trajectory is spurious monotonically decreases when r increases. As an example, probability P_{spur} dependent on r and N is shown in Fig. 5.11 for $L = 0.7N$. Each point in Fig. 5.11 was obtained from 10,000 trials. It quickly drops when network size N is relatively small and remains high for large N. Thus, when network size is relatively small most trajectories become true during the recall process and can be used for factor recognition. However, when network size increases considerably almost all trajectories are spurious and the recall procedure becomes incapable of factor searches.

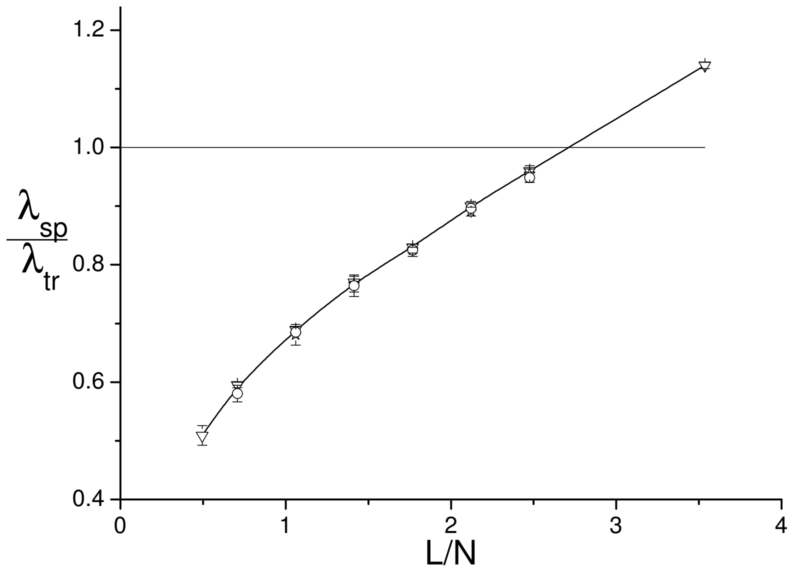


FIGURE 5.10: Lyapunov functions of spurious attractors for $r = p$ normalized by mean values of this function over true attractors at the point $r = p$. ▽ - an ordinary Hopfield-like network ($C = 1$), ⋆ - $C = 10$, ∘ - $C = 20$. Thin solid line - B-Spline approximation of experimental points. The horizontal line gives the mean Lyapunov function value for true attractors.

Figure 5.12 demonstrates the probability of transition from spurious to true attractor P_{trans} dependent on r and N. Probability P_{trans} was calculated as:

$$P_{\text{trans}} = (P_{\text{spur}}(k) - P_{\text{spur}}(k+1))/P_{\text{spur}}(k)$$

It has the maximum around $r = 0.01$. As shown in Fig. 4.1(a), the apex point corresponds approximately to the point where the values of Lyapunov functions for true trajectories become markedly higher than those for spurious trajectories and, thus, true attractors become more attractive. For $r > 0.01$, $\ln P_{\text{trans}}$ linearly decreases when r increases. The slope of this decrease appeared to be independent of N. However overall, P_{trans} decreases when N increases. This dependence of P_{trans} on r and N can be represented by the following regression model:

$$\ln P_{\text{trans}} = -aN - br. \tag{5.17}$$

For the example shown in Fig. 5.12 the parameters of the regression model were estimated as $a = (6.2 \pm 0.1) \cdot 10^{-4}$ and $b = 82 \pm 5$.

For sufficiently large N when r can be considered as a continuous variable, probability P_{spur} has the form:

$$P_{\text{spur}}(r) = \exp(-N \int_{r_{\text{in}}}^{r} P_{\text{trans}}(x)\mathrm{d}x).$$

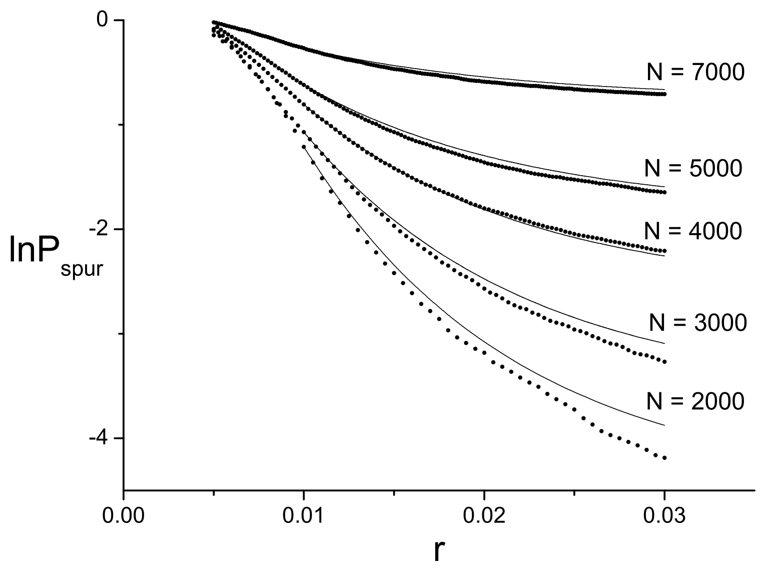


FIGURE 5.11: Probability P_{spur} dependent on r and N for $C = 1$, $L = 0.7N$. $P_{\text{spur}}(r)$ is the probability that a trajectory remains spurious until given r. Each experimental point was obtained from over 10,000 trials of computer simulation. Solid lines are approximations of experimental data by the formula (5.18).

Then according to (5.17) P_{spur} can be estimated as

$$P_{\text{spur}}(r) = \exp[-\frac{N}{b}\exp(-aN)(\exp(-br_{\text{in}}) - \exp(-br))]. \tag{5.18}$$

Figure 5.11 demonstrates the accuracy of the used approximation. It is less accurate for smaller N due to the effect of the discontinuity of r during the trials.

Coefficient a in regression model (5.17) happened to be proportional to L/N and was presented as: $a = c_1 L/N$, while coefficient b could be presented as $b = c_2 + c_3 L/N$. Thus as a whole, the dependence of P_{trans} on L, N and r could be presented as

$$\ln P_{\text{trans}} = -c_1 L - c_2 r - c_3 r L/N. \tag{5.19}$$

Coefficients c_i were found as the best fit over the whole set of tested network parameters: $L/N = 0.5, 0.7, 1, 1.4$ and $N = 2{,}000, 3{,}000, 4{,}000, 5{,}000, 7{,}000$. Each value P_{trans} was calculated over 10,000 trials. Coefficients c_i for $C = 1$ were estimated as $c_1 = (8.8 \pm 0.1) \cdot 10^{-4}$, $c_2 = 25.5 \pm 3$ and $c_3 = 80 \pm 2$ while for $C = 20$ they are $c_1 = (9.6 \pm 0.6) \cdot 10^{-4}$, $c_2 = 690 \pm 30$ and $c_3 = 254 \pm 18$ and for $C = 10$ they are $c_1 = (9.0 \pm 0.4) \cdot 10^{-4}$, $c_2 = 680 \pm 20$ and $c_3 = 367 \pm 14$. According to (5.18) the probability that a trial finally happened to be true, i.e. $1 - P_{\text{spur}}(r_{\text{fin}})$, mainly depends on the term $aN - \ln(N/b) = c_1 L - \ln(N/(c_2 + c_3 L/N))$. The probability is relatively high when this term is small. Thus the probability of true trials is relatively high when

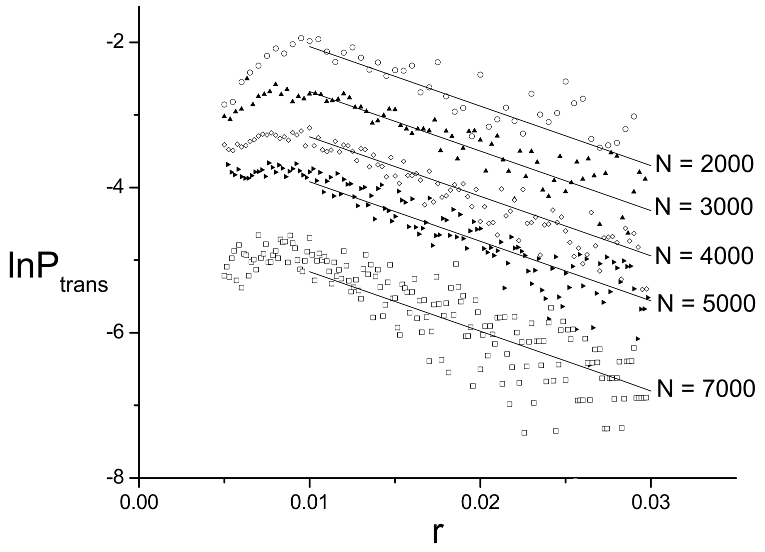
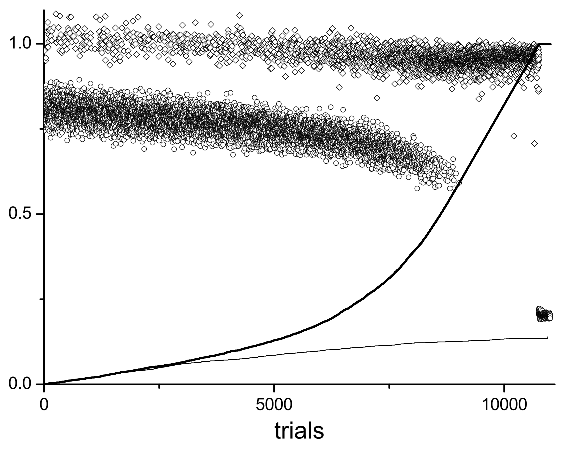


FIGURE 5.12: Probability of transition P_{trans} from spurious to true attractor dependent on r and N for $C = 1$, $L = 0.7N$. Each experimental point was obtained from over 10,000 trials of computer simulation. Solid lines are approximations of experimental data by the formula 5.17.

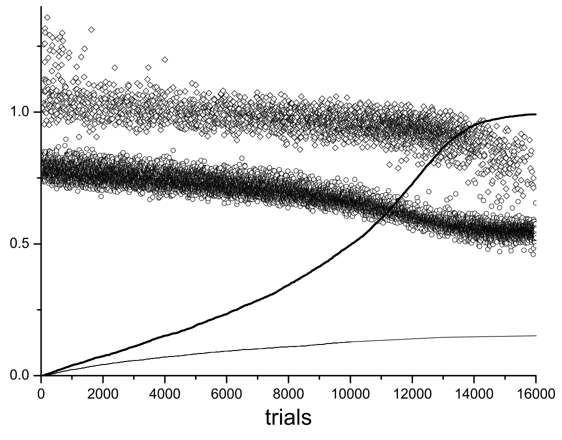
$L < \ln(N/(c_2 + c_3L/N))/c_1$ and it drops to zero when this condition is no longer fulfilled. For large N this condition is satisfied for small loading L/N. In this case one can ignore the term c_3L/N comparing with c_2 and rewrite this condition as $L < L_2 \simeq \ln(N/c_2)/c_1$. Thus for large N the probability of true trials is relatively high when $L < L_2 \simeq \ln(N/c_2)/c_1$. For $C = 1$ $L_2 \simeq 10^3 \ln(0.04\,N)$, for $C = 10$ and $C = 20$ $L_2 \simeq 10^3 \ln(0.0014\,N)$. The random search of factors is possible when L satisfies both conditions $L < L_1$ and $L < L_2$.

5.5 Hebbian unlearning of found factors

Even in the case when the probability of true trials is rather high, the search of all factors could be impossible due to the dominance of some of them. Figure 5.13(a) illustrates this statement: the thin line shows the number of found different factors dependent on the number of trials for $N = 3,000$, $L = 1.4N$, $C = 1$. According to (5.18) for these network parameters the probability of a true trial amounts to around 0.1. Initially the number of new found factors increases proportionally to the number of trials with a proportionality coefficient equal to the probability of true trials (i.e., 0.1). However later the search of new factors becomes slower because all trajectories are attracted by the fraction 0.14 of all factors. The same slowdown of the search of new factors was observed for $C = 20$ (Fig. 5.13(b)).



(a)



(b)

FIGURE 5.13: Normalized Lyapunov function (points) and the portion of found factors (lines) dependent on the number of recall trials for $N = 3,000$, $L = 1.4N$ and $C = 1$ (a) and $C = 20$ (b). $\diamond$ - true attractors, $\circ$ - spurious attractors. The thick and thin solid lines are rates of found factors obtained with and without unlearning, respectively.

This difficulty can be easily overcome by Hebbian unlearning when attractors which appeared during the recall process are deleted from the network memory. The deletion was performed according to the Hebbian unlearning rule by subtracting ΔJ_{ij} from synaptic connections J_{ij} where

$$\Delta J_{ij} = J[(x_i(t) - r)(x_j(t+1) - r) + (x_i(t+1) - r)(x_j(t) - r)], j \neq i \tag{5.20}$$

$\mathbf{x}(t)$ and $\mathbf{x}(t+1)$ are successive patterns of network activity in the attractors, and $J = \lambda/rN$ is a mean weight of synaptic connections between neurons of the factor. The unlearning ensures that both the point attractors when $\mathbf{x}(t+1) = \mathbf{x}(t)$ and also the cyclic attractors are suppressed. Figure 5.13 shows the effect of unlearning (thick line). The points are values of Lyapunov functions obtained for $r = p$ normalized by mean values of this function over true attractors found without unlearning. Points with high and low values of normalized Lyapunov function correspond to true and spurious attractors. Values of normalized Lyapunov functions are shown only for the recall with unlearning. In the recall without unlearning, the distribution of experimental points does not depend on the number of trials and corresponds to the initial stage of the recall with unlearning. Initially the rate of true attractors in unlearning cases is the same as that in cases without unlearning. However soon the rate of the search of new factors speeds up. This process is accompanied by the decrease of Lyapunov functions for both true and spurious attractors. For true attractors it decreases because of the elimination of factors with high Lyapunov function values. For spurious attractors it decreases because the elimination of factors is equivalent to the network unloading (i.e., the reduction of the number of factors), and as shown in Fig. 5.10, the Lyapunov function values for spurious attractors monotonically decreases when the number of factors L reduces. Since Lyapunov function values decrease faster for spurious attractors, the rate of true attractors increases and at the final stage of the recall process the probability of spurious attractors falls to zero and each trial results in the retrieval of a new factor. Only about 11,000 trials for $C = 1$ and about 16,000 trials for $C = 20$ are required to reveal all 4,200 factors. When almost all factors are found the probability of spurious trials increases again, but with very small Lyapunov function values.

Chapter 6

Application of hybrid ANNIA and likelihood maximization method

We assume that expression (2.4) defining the BFA generative model provides a rather general form of binary signal representation. For example, for textual data, a factor is some topic characterized by keywords related to factor loadings, and each factor score is defined by whether a given document is dedicated to the topic. Though each topic is represented by a set of keywords, there are no or few documents containing the whole set. Factor distortion means the absence of some keywords from a topic keyword list in a given document dedicated to the topic. Each specific factor relates to each individual word. It is characterized by the probability of the related word to be present in the document independently of topics. Another example of a real-world BFA application is the role mining problem [22]. This task requires to infer a user-role assignment matrix as a matrix of factor loadings **F** and a role-permission assignment matrix as a matrix of factor scores **S** from a Boolean user-permission assignment matrix **X** defining an access-control system.

In this chapter we present several examples of application of LANNIA to real world binary datasets that are supposed to comply with BFA generative model. For all presented examples this assumption seems to be justified because revealed factors can be interpreted and information gain for LANNIA results is rather high. In order to compare LANNIA with state-of-art methods, in some examples, we applied BFA related methods described in Chapter 3 to the same datasets.

6.1 Comparison of LANNIA with other methods in solving the bars problem

The Bars Problem (BP) introduced by Foldiak [20] and described in Section 2.4 is a common benchmark to reveal strengths and weaknesses of BFA methods. In this section, the efficiency of LANNIA is compared with four other BFA related methods in solving BP. Some of them were supposed [71, 87] to be the most efficient for BFA, at least for solving the bars problem. The first method, described in Section 3.2.3, is the Dendritic Inhibition (DI) neural network. The second one, described in Section 3.3.2, is the Maximal Causes Analysis restricted to the case when each pattern of the dataset contains not more than three factors (MCA_3). The third method is Expectation-Maximization Binary Factor Analysis (EMBFA), described in Section 3.3.1. The fourth method is fast Boolean Matrix Factorization based on Formal Concept Analysis (BMFCA), described in Section 3.1.2. The last one is the special version of LANNIA in which LM procedure is replaced by simple bayesian classifier, described in Section 4.4. The efficiency of LANNIA was also compared with Boolean matrix factorization for noisy datasets [70] and with other methods in [23, 36, 58–60, 76, 84, 85].

The results of BFA methods are also compared with the right bars problem solution when all bars are revealed in all tested images. The comparison is performed in terms of information gain. In experiments presented below each value of information gain was obtained by averaging results of analyzes over 50 datasets, each containing M bars problem images.

For EMBFA, DI and MCA_3, the number of desired factors has to be set in advance. In the experiments presented below, as in [41, 71, 87], the predefined number of desired factors was taken twice higher than the actual number of factors. In the computer experiments involving DI and MCA_3 we used the parameters recommended in the original papers [71, 87]. LANNIA, BANNIA, BMFCA and EMBFA are free of any parameters.

In addition to testing the methods on the bars problem in its original formulation in Section 6.1.1, the following versions of the problem were used for comparison of the methods: strongly overlapping bars (Section 6.1.2), in the presence of noise that was assumed to be distributed uniformly over signal components and factors (Section 6.1.2), and with the increased mean number of bars mixed in images (Section 6.1.4).

As it follows from the experiments described below, only LANNIA provides almost the right bars problem solution for all analyzed bars problem tasks. It is least sensitive to noise in data, to the number of factors mixed in each observation and to the insufficient number of observations (decrease of M) in the dataset. BMFCA is perfect only in the absence of noise in factors, then it is insensitive to decrease of dataset size and to specific noise. However, it loses to other methods in the presence of noise in the form of factor distortion. On the contrary, MCA_3 is insensitive to factor distortion but very sensitive to specific noise. MCA_3 is also unable to reveal factors when the number of factors mixed in the pattern of the dataset $C > 3$. DI is moderately sensitive to noise of both kinds but very sensitive to signal complexity C. When C increases, the method becomes unstable in the sense that its operation strongly depends on the realization of the initial synaptic weights of the basic network. For one set of initial weights, DI may converge to true factors, while for another, it converges to a random solution. The EMBFA method exhibits a low sensitivity to noise of both kinds but is moderately sensitive to signal complexity C. BANNIA is only sensitive to a ratio of number of observations M in the dataset to the number of attributes N. If $M < \alpha N$, observations of the dataset create individual attractors of ANNIA dynamics which can dominate over attractors created by factors preventing the search of factors. Coefficient α is estimated in Section 5.4. LANNIA is able to overcome this problem by means of highly-accurate unlearning of factors revealed first, but BANNIA does not.

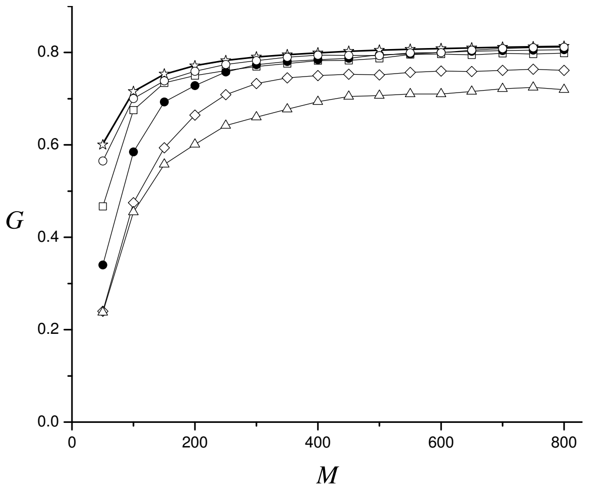
6.1.1 Standard bars problem

The bars problem in its original formulation is described in Section 2.4. The dependence of the information gain G on the number of observations M in a dataset for the standard bars problem is shown in Fig. 6.1(a). LANNIA and BMFCA provide an exact solution of the bars problem. When the number of observations of the dataset becomes larger ($M \geq 300$), BANNIA also provides an exact solution. For sufficiently large M, DI and MCA_3 precisely reveal all true factors.In spite of the fact that all true factors were found, the information gains obtained by both methods are less than the information gain of right solution. The reason for the decrease in G is the omission of some factor scores. For DI, the fraction of missing scores was 2.3%, and for MCA_3 and EMBFA it was about 10%. Recall that both EMBFA and MCA_3 are restricted to the case that not more than three factors are mixed in an observation. In the case when each observation contains exactly two randomly chosen bars both methods provide the right solution [35].

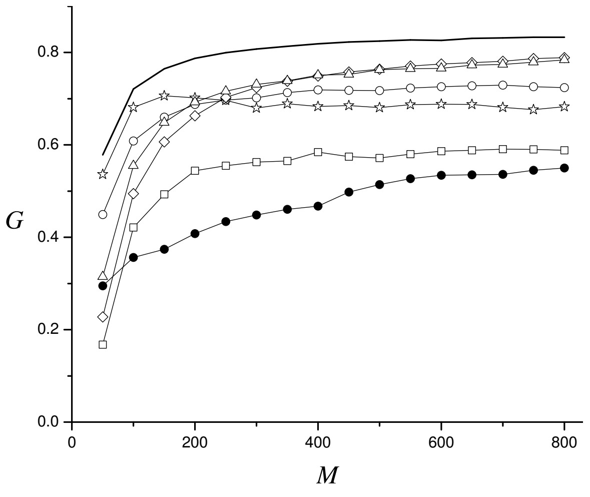
6.1.2 Sensitivity to factor overlapping

The sensitivity of the methods to strong factor overlapping is tested on the version of bars problem when each of 16 factors is a double bar overlapping with two other factors by half of its pixels as shown in Fig. 6.2(A). As for the case of single pixel bars, the probability of a factor's appearance in the dataset is $\pi_i = 1/8$ for all factors (Fig. 6.2(B)).

As shown in Fig. 6.1(b), BANNIA is the most sensitive to factor overlapping. The reason is that after excluding a found factor from the network, ANNIA is unable to find factors strongly



(a)



(b)

FIGURE 6.1: Information gain G for the six BFA methods in dependence on number of observations M in dataset. Noise is absent ($q_j = 0$, $p_{ij} = f_{ij}$), (a) – standard BP problem, (b) – factors are double bars. ○ – LANNIA, ● – BANNIA, ☆ – BMFCA,△ – EMBFA, □ – DI, ◇ – MCA$_3$. Thick line – right solution.

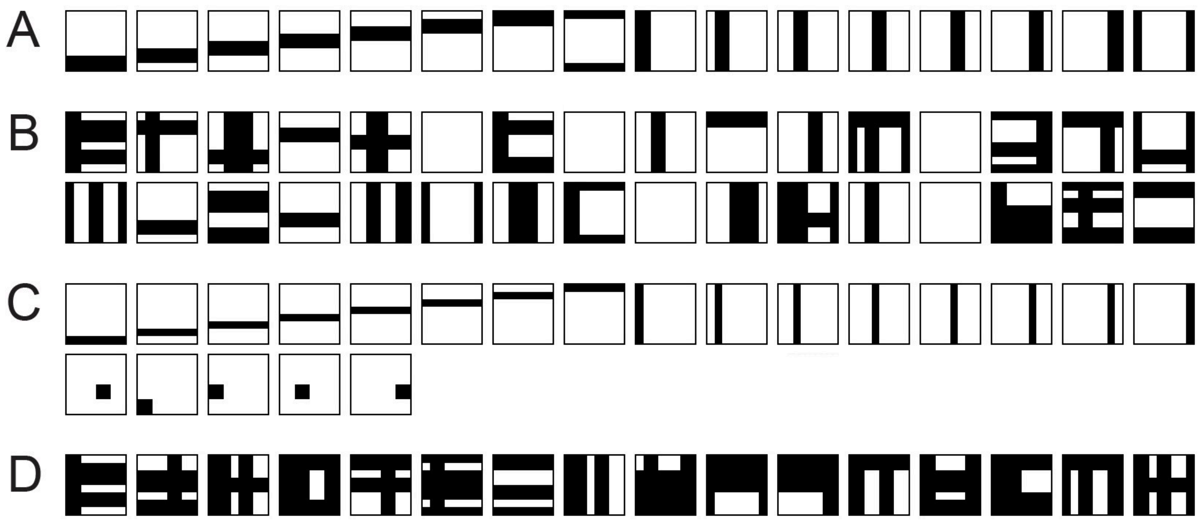


FIGURE 6.2: **A** Sixteen vertical and horizontal double bars. **B** Examples of images in double bars problem. Each image contains two bars on average. **C** Factors found by BMFCA in double bars problem. **D** Examples of patterns in double bars problem where DI is unable to reveal any factors.

overlapping with it. LANNIA partially overcomes this problem because of the modified unlearning rule. BMFCA loses its high performance because the greedy algorithm identifies as factors single pixel bars that are actually just fragments of factors (Fig. 6.2(C)). DI loses to other methods because it misses some factor score. In contrast to the case of single pixel bars, DI found only 62% of the true scores. Examples of dataset patterns where DI is unable to reveal any factors are shown in Fig. 6.2(D). All such patterns contain a mixture of three or more double bars. In this case the high activity at the input layer of the DI network suppresses activity at the output layer because of the strong competition between its neurons. MCA_3 and EMBFA happened to be not sensitive to factor overlapping.

6.1.3 Sensitivity to noise

Figure 6.3 demonstrates the sensitivity of BFA methods to noise in solving the standard BP. The noise was assumed to be distributed uniformly over observation components and factors so that $q_j = q$ for any j, and $p_{ij} = pf_{ij}$ for any i and j. This means that pixels constituting a bar can take 0 with the equal probabilities $1 - p$ and any pixel can take 1 with the probability q due to related specific factor. The methods were tested using datasets containing $M = 800$ observations. As shown in Fig. 6.1, in the absence of noise the information gain G reach saturation at that particular M. The saturation is reached for $M = 800$ in the presence of noise also.

As shown in Fig. 6.3(a), LANNIA, BANNIA and BMFCA provide almost the right BP solution independent of the noise produced by specific factors. The information gains G obtained by EMBFA is less than the information gain of right solution due to omission of some factor scores (see Sec. 6.1.1), but EMBFA is insensitive to specific noise too. MCA_3 and DI demonstrate strong sensitivity to specific factors. For DI, an increase in q results in a decrease in G, because the number of found true factors decreases. For MCA_3, G drops near to zero when q increases to 0.2. In this case, the solution of the bars problem by MCA_3 becomes unstable, that is, it drastically depends on the peculiarities of the dataset or on the choice of initial parameters for the EM procedure. With one random realization of the dataset, MCA_3 may provide a perfect solution (in our experiments after approximately 300 steps of the EM procedure), but with another random realization chosen from the same distribution, the procedure converges to some random images as factors (in just 3–5 steps). For $q = 0.2$, a successful search for bars by MCA_3 was observed only in two out of 50 trials. Thus, the standard deviation for this method amounted to $5 \cdot 10^{-2}$.

As shown in Fig. 6.3(b), BMFCA happened to be the most sensitive to noise in the form of factor distortion. In this case, BMFCA is able to find only fragments of factors. The number of all factors extracted by BMFCA as a multitude of bar fragments rapidly increases with decreasing p. This results in a gain drop. DI exhibits similar sensitivity to factor distortion as to specific noise, while MCA_3 is only slightly sensitive to this kind of noise. The reason is that factor distortion is a part of the MCA_3 generative model. The information gain is smaller than that for right solution only due to missing scores in observations containing mixture of more than three bars. LANNIA, BANNIA and EMBFA provide almost the right BP solution.

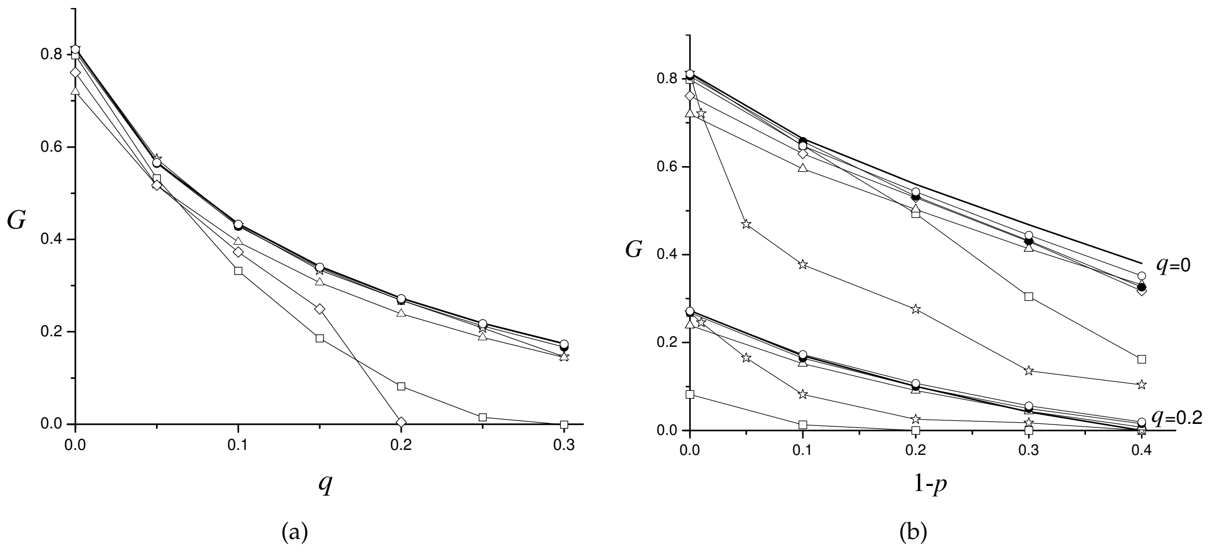


FIGURE 6.3: Information gain G for the six BFA methods in dependence on q for $p = 1$ (a) and on p for $q = 0$ and $q = 0.2$. ○ – LANNIA, ● – BANNIA, ☆ – BMFCA,△ – EMBFA, □ – DI, ◇ – MCA3. Thick line – right solution.

Fig. 6.3(b) also demonstrates the sensitivity of the BFA methods to both kinds of noise applied simultaneously ($q = 0.2$, $p < 1$). For such noise, MCA3 is not able to reveal any proper factor (thus, its gain is not depicted in Fig. 6.3(b)). DI provides considerably smaller G than that provided by the right solution. LANNIA, BANNIA and EMBFA provide much better solutions and the information gain G obtained by these methods is again close to that for the right solution.

Some patterns of the dataset for $q = 0.2$ and $p = 0.7$ are shown in Fig. 6.4(**A**) as examples. As shown in Fig. 6.3(b), for this level of noise information gain provided by the BFA solutions is close to zero and the factor structure of data is actually invisible. Nevertheless, factors extracted by LANNIA from this dataset almost coincide with bars. For the found solution information gain is even higher than for the right solution. This is a frequent case for the solutions found by LM for dataset, containing a relatively small number of very noisy observations.

LANNIA is also almost insensitive to not homogeneous noise. For example, when 4 pixels of each bar are not distorted by noise ($p_{ij} = 1$) and 4 other pixels are flopped to zero due to noise with probability 0.7 ($p_{ij} = 0.3$), LANNIA provides the solution with $G = 0.57$, which is close to the value $G = 0.59$ provided by the right solution. Such a kind of factor distortion when different factor's attributes have different probabilities p_{ij} of appearance in observations of dataset containing the factor is most critical for BANNIA performance because the unlearning rule (4.14) used in BANNIA, in contrast to the rule (4.13) used in LANNIA, does not take this possibility into account. Actually, BANNIA provides $G = 0.49$, which is much smaller than

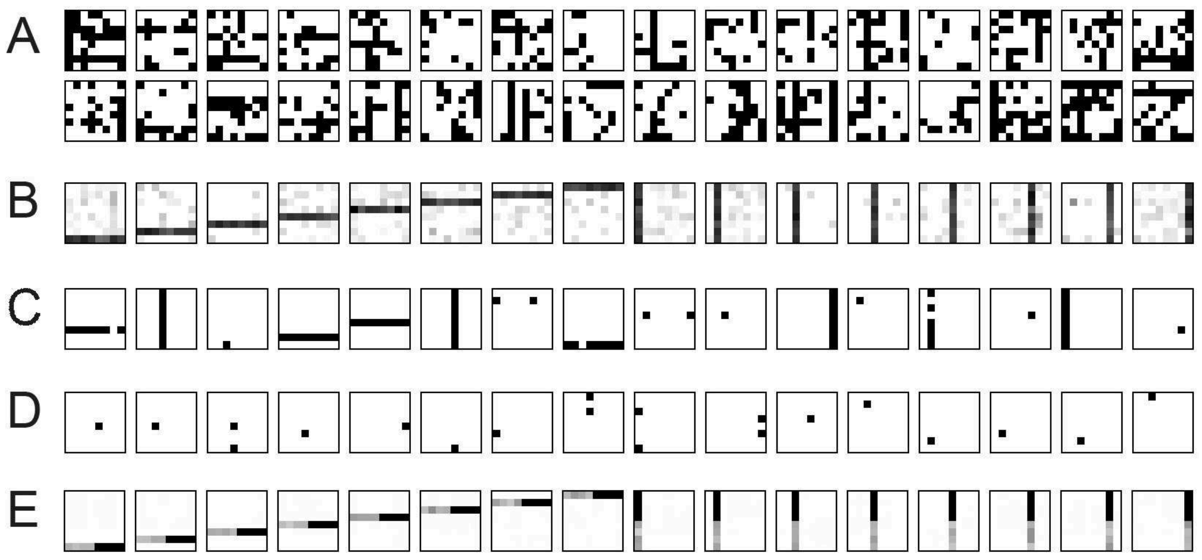


FIGURE 6.4: **A** Examples of noisy images ($p = 0.7$, $q = 0.2$). **B** Factors found by LANNIA when solving BP with the noisy images ($p = 0.7$, $q = 0.2$, $M = 800$). **C** and **D** Factors found by BMFCA when $q = 0.2$ and $p = 0.9$ and $p = 0.8$, respectively. **E** Factors found by LANNIA when $p_{ij} = 1$ for four pixels of each bar and $p_{ij} = 0.3$ for another four pixels. In **B** and **E**, the black pixels correspond to $p_{ij} = 1$, the white pixels correspond to $p_{ij} = 0$, and the gray pixels correspond to the intermediate values of p_{ij}.

that provided by LANNIA. Factors found by LANNIA in one of the trials are shown in Fig. 6.4(**E**) as an example. LANNIA found all factors with almost precise p_{ij} estimation.

Fig. 6.4(**C**) and (**D**) also demonstrate that in the case of noisy factors ($p < 1$), BMFCA is able to find only fragments of factors. When p decreases, found factors become increasingly smaller until they are reduced to individual pixels and information gain drops to zero.

6.1.4 Sensitivity to the mean number of bars mixed in images

To investigate the sensitivity of the BFA methods to the mean number of bars mixed in observations (encoded by parameter C), the size of the images should be increased in order to provide sparse coding of signals. The increased image size to a grid of 16×16 pixels allowed us to study the effects of increasing C up to $C = 10$. Only noiseless case is considered here (i.e., $p_{ij} = f_{ij}$, $q_j = 0$). The methods were tested using datasets containing $M = 800$ observations.

LANNIA, BANNIA and BMFCA appeared absolutely insensitive to increasing C and provided almost the right solution of the bars problem even for $C = 10$ (Fig. 6.5). DI performance gets worse with increasing C. The reason is the loss of solution stability similar to the case of MCA3, described above in Sec. 6.1.3. When the number of active neurons at the input of the DI network becomes relatively large due to large C, DI fails in finding the proper factors.

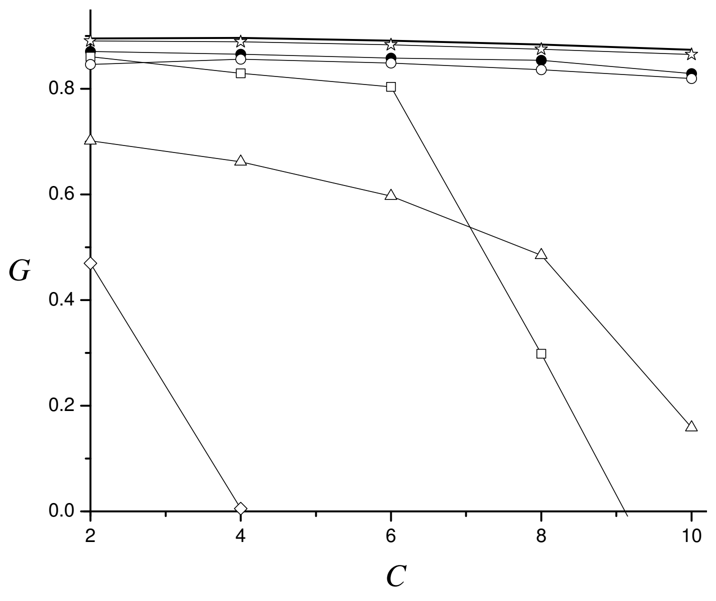


FIGURE 6.5: Information gain G vs C for 16-by-16 pixel images at $M = 800$. ○ – LANNIA, ● – BANNIA, ☆ – BMFCA,△ – EMBFA, □ – DI, ◇ – MCA3. Thick line – right solution.

Since MCA3 and EMBFA are both restricted to the case of sparse scores ($C \leq 3$), one would expect that those methods are most sensitive to an increase of C. This expectation proved to be true for MCA3: it failed in finding the proper factors for $C \geq 4$. However, EMBFA amazingly gives reasonable results even for $C = 8$, although not quite right. The reason is the EMBFA ability to treat some redundant bars as noise when the number of bars mixed in the observation exceeds three [35].

6.1.5 Computational complexity of the BFA methods

The computational complexity is estimated in the limit of large M, L, and N. In this limit, the number of operations for BMF is proportional to $\Omega_{BMF} = MLN^2\langle n_f\rangle\langle p_j\rangle$, where $\langle n_f\rangle$ is the mean number of ones in the factors and $\langle p_j\rangle$ is the mean probability of each signal component's being one in the dataset. For a PC Core2 6400, 2.13 GHz, the execution time of one operation in seconds amounts to about 10^{-11}. For all methods, this time was estimated by dividing the total execution time by Ω when M, L, and N were sufficiently large so that their doubling resulted in a 5% change in the estimated value.

The number of operations for DI in one iteration step is approximately proportional to $\Omega_{DI} = (2L)MN\langle p_j\rangle$ and the execution time for one step amounts to about $10^{-7}\Omega_{DI}$. Usually about 15–20 steps are required.

The number of operations required for one iteration step of EMBFA, according to formulas (3.6) and (3.10), is proportional to $\Omega_{EM} = MN(2L)^3\langle p_j\rangle$ and the execution time for one step amounts to $5 \cdot 10^{-9}\Omega_{EM}$. For EMBFA the mean number of iteration steps until convergence is about 100. Thus to evaluate the total execution time one must multiply the execution time for one step by a factor of 100.

The number of operations required for one iteration step of MCA_3 is the same as for EMBFA. However, the mean time of one operation is approximately twice as high as that for EMBFA, and the mean number of iteration steps until convergence is about 300. Thus, the total execution time for MCA_3 is six times higher than for EMBFA.

The execution time required for ANNIA is composed of two times. The first time T_1 is required to create connection matrix. In the limit of large M, L and N, $T_1 \simeq 10^{-9}MN^2$. The second time is required to find factors: $T_2 \simeq 10^{-8}LN\langle n_f \rangle^2$. For LANNIA additional time $T_3 \simeq 10^{-7}ML\langle n_f \rangle$ is required to perform the procedure of likelihood maximization. Thus, the execution time for one cycle of LANNIA amounts to about $\Omega_{LANNIA} = T_1 + T_2 + T_3$. Usually about 2–10 cycles are required for convergence of the whole algorithm.

For the bars problem $n_f = \sqrt{N}$, $L = 2\sqrt{N}$, and in the absence of noise $\langle p_j \rangle \simeq C/\sqrt{N}$. As a whole, to perform BFA for a dataset of 3200 images of 64 by 64 pixels, containing two undistorted bars, about 460 sec is required for BMF, 110 sec for DI, 240 hours for MCA_3, 40 hours for EMBFA, and about 300 sec for LANNIA.

6.2 Application to text datasets

Due to the proliferation of information in textual databases, and especially, on the Internet, the issue of minimization of the search space with the proper selection of a keyword list becomes more and more important. Unsupervised word clustering (providing a kind of thesaurus) is a standard approach to perform this task [48]. Based only on the words statistics, it allows the overcoming of the diversity of synonyms used by authors of different expertise and background. The challenge is to treat this problem with the Boolean factor analysis.

We supposed that each textual document of the dataset is presented as a binary vector of the dimension of the used term dictionary. Each component of the vector is 1 or 0 depending on the presence or absence of the corresponding term in the document. Like many others since the paper [91], we hypothesize that each topic is characterized by a specific set of terms (keywords) which appear in an article on the topic. Following [19] we call this set as a *concept*. The coherent appearance of the terms of the concept in the article constitutes evidence that the article is dedicated to the corresponding topic.

In the frame of Boolean factor analysis, each concept represents a factor, factor loading is 1 or 0 depending on whether the term belongs to the concept, and factor score is also 1 or 0 depending on whether the article belongs to the topic. The identification of factors is equivalent to the automatic extraction of topics keywords. It is supposed that concepts occur in documents independently of each other. Though each topic is represented by a set of keywords, there are no or few documents containing the whole set. Factor distortion means the absence of some keywords in the document dedicated to the topic. Each specific factor relates to each individual word to be present in the document independent of topics. One can expect that, first, factors are distorted inhomogeneously, that is $p_{ij} \neq pf_{ij}$, and, second, that all words are distributed in the documents with different probabilities, that is $q_j \neq q$. The examples of application of BFA to text datasets can be found in [1, 31–34, 45, 51, 53, 55, 56, 61, 81].

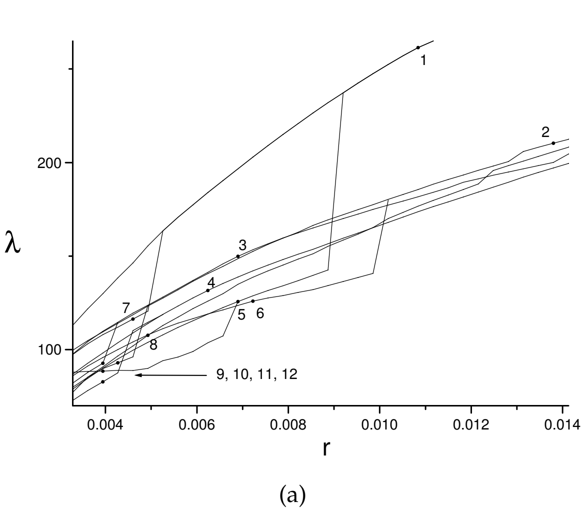
6.2.1 Analysis of the proceedings of the IJCNN and Neuroinformatics conferences

As a source for textual databases we used the papers published in the proceedings of the IJCNN conferences held in 2003 and 2004, and in the proceedings of the Russian conference on Neuroinformatics held in 2004 and 2005. The sizes of the considered databases amounted to $M = 1042$ and $M = 189$ articles, respectively. After stop-words and rare words filtering, i.e., those that appeared in less than in 3 percent of the articles, the sizes of the dictionaries were $N = 3044$ and $N = 1716$ words. The article length, i.e., the number of different words used, varied from 14 to 573 (mean 280, mean 847 before filtering) in English articles, and from 16 to 514 (mean 184, mean 312 before filtering) in Russian articles. The documents were represented as vectors, see e.g. [15].

Databases of international and Russian conferences were analyzed separately. We are interested in comparing the sets of topics of different conferences and the contents of similar concepts. In the present case, this task seems to be rather difficult a priori because all the topics of the conferences belong to one narrow domain "Neural Networks" and it is well known that keyword extraction for topics of a narrow domain is one of most difficult tasks in text analysis [3].

The changes of the Lyapunov function along 12 trajectories of the network dynamics are shown in Fig. 6.6(a) for the IJCNN database (further designated as IJCNN). Only these 12 different trajectories were obtained in 200 trials of ANNIA. The recall was performed without Hebbian unlearning but we believe that the search is exhaustive because a further run of 200 trials did not add new trajectories to those shown in Fig. 6.6(a). The values of $h_{\max}$ in (4.9) ranged from 45 to 59. Since the values of the Lyapunov function for the shown trajectories exceeded $h_{\max}$ we suppose that all these trajectories are not spurious.

The twelve factors that correspond to the peaks of R' along these trajectories are marked in Fig. 6.6(a) by points. The factor with the highest value of the Lyapunov function (i.e.,



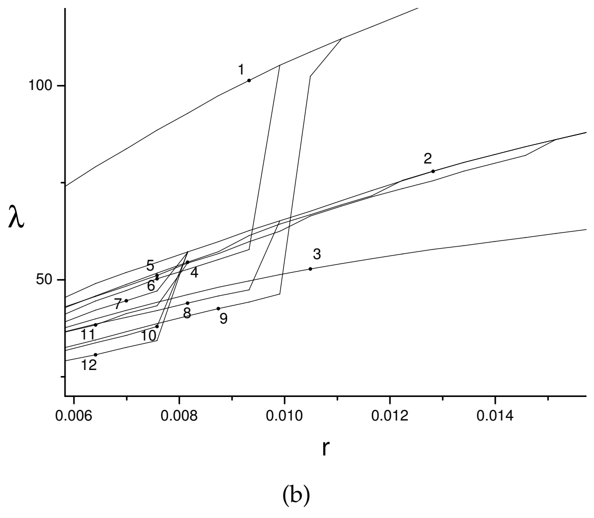


FIGURE 6.6: The Lyapunov function dependent on the relative network activity r for textual databases IJCNN (a) and Neuroinformatics (b). The points on the curves are factors found by the peaks of R'.

with the most powerful attractor) contains 33 words. Ten of them are presented in Table 6.1. By the specificity of words the factor could be easily recognized as corresponding to the topic "Neurobiology". The articles belonging to this topic were found by the procedure presented in Section 4.4. The properties of the other 11 factors are also shown in Table 6.1. The significance of a given word for a given topic was calculated by Kullback divergence

$$v = p\ln(p/q) + (1-p)\ln((1-p)/(1-q))$$

which takes into account probabilities p and q that the word appears in articles, respectively, belonging and not belonging to the topic. We related other factors to the topics: 2) Classification, 3) Optimization, 4) Probability, 5) Hardware, 6) Genetic Algorithms, 7) Image processing, 8) Multilayer networks, 9) Dynamic stability, 10) Self-organizing mapping, 11) Source separation. The twelfth factor looks strange. It contains abbreviations "Fig.N" printed without space and "IEEE Trans". We revealed that this factor was created due to the fact that articles from 2003 contained the misprint "Fig.N" without space two times more frequently than articles from 2004. The term "IEEE Trans" was bound with terms "Fig.N" due to the fact that the PDF-format for articles from 2003, in contrast to 2004, included the printing of "IEEE Trans" at the end of each page. The appearance of this factor stressed the fact that the method is based on pure statistics, however, the statistics correspond to the nature of the textual database: articles on the same topic tend to contain the same set of words. That is why all other topics are quite reasonable. The last factor was excluded from further consideration because it is meaningless.

#	Factor length	Related articles	Top words
1	33	203	cortex (3.0), excitatory (1.7), inhibitory (1.7), stimulus (1.3), spike (1.2), synapse (0.9), brain (0.9), neuronal (0.9), sensory (0.6), cell (0.5)
2	42	299	classifier (1.1), support vector machines (0.7), hyperplane (0.7), svms (0.5), validation (0.5), database (0.4), label (0.4), repository (0.4), machine learning (0.3), margin (0.3)
3	21	333	gradient (0.9), descent (0.7), convergence (0.7), approximate (0.5), guarantee (0.4), derivative (0.4), formulate (0.3), iteration (0.3), cost (0.3), satisfy (0.3)
4	19	183	estimation (0.7), observed (0.7), density (0.6), variance (0.6), gaussian (0.5), mixture (0.4), statistical (0.4), assumption (0.3), likelihood (0.3), probability (0.2)
5	21	100	vlsi (2.6), voltage (1.7), circuit (1.7), gate (1.6), transistor (1.5), cmos (1.1), hardware (1.1), block (0.9), chip (0.8), clock (0.6)
6	22	75	mutation (4.0), crossover (3.5), chromosome (3.1), fitness (2.8), genetic (2.7), population (2.2), evolutionary (1.6), generation (1.4), parent (1.1), selection (1.0)
7	14	233	pixel (2.6), image (1.9), camera (0.5), color (0.5), object (0.5), recognize (0.4), extraction (0.4), eye (0.4), vision (0.3), horizontal (0.2), vertical (0.2)
8	15	450	multilayer (1.0), hidden (1.0), perceptron (0.5), approximation (0.4), testing (0.4), backpropagation (0.3), estimation (0.3), validation (0.3), epochs (0.3), regression (0.2)
9	12	159	inequality (1.1), guarantee (1.1), proof (1.1), theorem (0.8), stability (0.7), constraint (0.7), bound (0.5), convergence (0.4), satisfy (0.4), derivative (0.4)
10	13	199	self-organizing maps (1.3), som (1.0), cluster (0.8), winner (0.7), unsupervised (0.5), neighborhood (0.5), euclidean (0.4), competitive (0.4), dimension (0.2), group (0.2)
11	12	62	independent component analysis (3.7), blind (3.4), mixing (2.3), bss (2.1), separation (2.0), source (1.7), mixture (1.6), independent (0.9), signal processing (0.6), speech (0.3)
12	12	102	fig.4 (3.3), fig.3 (3.0), fig.5 (2.8), fig.6 (2.1), fig.7 (1.9), fig.2 (1.8), fig.1 (1.4), fig.8 (1.2), fig.9 (1.1), ieee trans (0.2)

TABLE 6.1: Ten top significant terms for factors found in the IJCNN database. Significance of the terms for the factors is in brackets.

Table 6.1 demonstrates that the largest portion of the articles is related to the topic "Multilayer networks". However, the Lyapunov function of the corresponding factor is rather small. Obviously this results from the fact that the specificity of its words was relatively small. One can compare the significance of the first word of this factor with that of the factor "Neurobiology", whose attractor dominates in spite of a two times smaller number of articles on this topic.

On average one article contains 2.2 factors. The distribution of articles over factors they contained is shown in Fig. 6.7. All possible combinations of factors were not uniformly distributed over the set of articles. Most of their combinations did not appear at all. There were only 150 combinations of factors from the total set of 2048 possible ones. The factors' affinity (in the sense of their joint appearance in articles) could be revealed by the analysis of transitions between the trajectories of network dynamics in Fig. 6.6(a). For example, most transitions were observed from the trajectories "Image processing" and "Hardware" to the trajectory "Neurobiology". It results from the fact that there were many articles containing

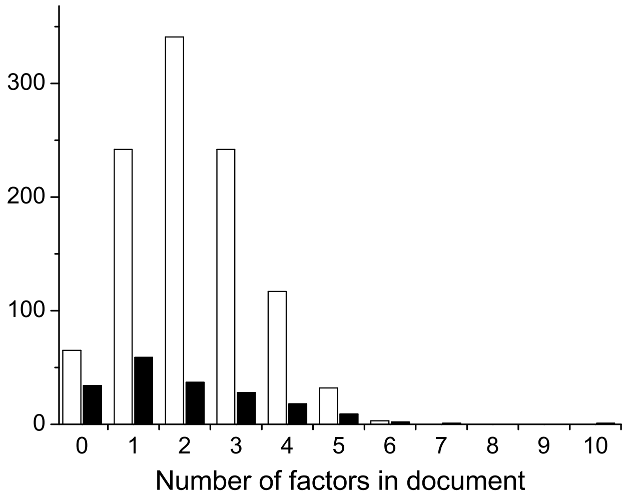


FIGURE 6.7: Distribution of articles over the number of factors being contained in them. Open bare - IJCNN, full bars - Neuroinformatics.

together the factors “Image processing” and “Neurobiology” or the factors “Hardware” and “Neurobiology”. We could see that articles containing together the factors “Neurobiology” and “Image processing”, were devoted to the problem “Image processing by the visual cortex”, and those containing together the factors “Neurobiology” and “Hardware” were devoted to the problem “Hardware implementation of natural neural networks”.

The change of the Lyapunov function along the 12 trajectories of the network dynamics is shown in Fig. 6.6(b) for the Neuroinformatics database (further designated as Neuroinformatics). As for the IJCNN, the trajectories were obtained without local unlearning by running 200 trials. This set of testing trials was exhaustive. The values of $h_{\max}$ ranged from 21 to 36. Thus we can suppose that all trajectories are not spurious.

As for the IJCNN, in the case of the Neuroinformatics, the factor with the highest value of the Lyapunov function corresponds to the topic “Neurobiology”. The properties of the other 11 factors are shown in Table 6.2. The words corresponding to the factors are translated in English. We related the other factors to the topics: 2) Multilayer networks, 3) Image processing, 4) Classification, 5) Optimization, 6) Intellectual systems, 7) Genetic Algorithms, 8) Recurrent networks, 9) Mathematics, 10) Intellectual agents, 11) Time series, 12) Clustering. On average one article contained 1.9 factors. The distribution of articles over the number of the factors being contained in them is shown in Fig. 6.7. The portion of articles containing one or no factors is larger in the Neuroinformatics collection although the number of factors is higher. It means that articles and factors in this collection are more specific.

#	Factor length	Related articles	Top words
1	16	41	physiology (2.1), nervous (1.5), excitatory (1.5), synaptical (1.5), inhibititory (1.3), activation (1.3), membrane (1.1), stimulus (1.1), brain (1.1), cortex (1.1)
2	22	39	optimization (2.6), hidden (2.4), iteration (0.8), backpropagation (0.5), layer (0.4), minimization (0.4), neural network (0.3), perceptron (0.3), sampling (0.3), weigth (0.2)
3	18	29	brightness (2.1), orientation (1.4), undertone (1.2), two-dimensional (1.2), image (1.2), radial (1.1), vision (0.9), uniform (0.9), pixel (0.9), area (0.8)
4	14	32	recognition (2.2), multilayer (1.5), classification (1.3), class (1.2), sampling (1.0), perceptron (0.8), practical (0.6), recommendation (0.5), stage (0.5), member (0.3)
5	13	24	iteration (1.5), convergence (1.1), gradient (1.0), perceptron (1.0), multilayer (0.8), stop (0.7), optimum (0.7), optimization (0.7), testing (0.5), minimization (0.4)
6	13	26	organisation (0.8), apparatus (0.8), objective laws (0.8), hierarchy (0.7), mechanism (0.6), intellectual (0.5), development (0.5), language (0.4), conception (0.4), understanding (0.3)
7	12	19	selection (1.6), independent (1.2), stop (1.1), genetic (1.0), population (1.0), sampling (0.9), mutation (0.8), optimization (0.7), criterium (0.6), efficiency (0.3)
8	14	46	vector (0.9), zero (0.8), number (0.6), equal (0.6), associative (0.6), iteration (0.4), cycle (0.3), perceptron (0.3), rule (0.3), change (0.3)
9	15	30	geometry (1.7), discret (1.7), measurment (1.1), plane (0.7), differenciation (0.7), form (0.7), physical (0.5), mathematical (0.5), boundary (0.4), integral (0.3)
10	13	24	intelligence (1.4), need (0.9), search (0.8), selection (0.8), presence (0.7), operation (0.6), mapping (0.6), quality (0.6), probability (0.5), adaptation (0.4)
11	11	38	rule (1.3), statistics (1.3), finance (1.3), vector (0.7), knowledge (0.5), prognosis (0.5), random (0.5), prediction (0.4), probability (0.3), expectation (0.2)
12	11	15	clustering (1.7), Kohonen (1.7), separation (0.8), distribution (0.3), center (0.2), noise (0.2), statistics (0.2), partitition (0.2), distance (0.1), selection (0.1)

TABLE 6.2: The same as in Table 1 but for case of Neuroinformatics database.

As for the IJCNN, the transitions between the trajectories reflect the relations between the topics. For example, the transitions from trajectories containing factors 8 and 9 to the trajectory containing factor 1 are explained by the wide presence of articles dedicated to the problems "Computer simulation of biological neural networks" and "Mathematical analysis of biological neural networks".

Seven topics in the Neuroinformatics coincide with those in the IJCNN database (namely Multilayer networks, Image processing, Classification, Optimization, Genetic Algorithms, Intellectual agents, Clustering). The topic "Mathematics" is more general than the topic "Probability" in the IJCNN. The topics "Hardware" and "Dynamic stability" are absent from Neuroinformatics, while topics "Intellectual systems", "Recurrent networks" and "Time series" are absent from IJCNN. We cannot be sure that the articles on these topics were completely absent from IJCNN. We only know that their presence was too weak to create the factors. On the other hand, the presence of the corresponding factors (as attractors of network dynamics) in Neuroinformatics could be explained by the fact that it contained a much smaller number of articles and hence factors' extraction is less reliable. The absence of the topic "Hardware" in Neuroinformatics could obviously be explained by the bad state of microelectronics in Russia.

Information gain obtained for Neuroinformatics by the method amounted to 0.15, that means that presentation of these textual data in the BFA model is quite reasonable. It is interesting that the grouping of the articles across scientific sections produced by Program Committees provided informational gain of only 0.04.

6.2.2 Analysis of the Reuters R52 dataset of news messages

Dataset R52 (from Reuters 21578) contains 9100 documents, which are labeled as belonging to 52 topics (classes). Before factor extraction stop words, rare words which appear less than in 10% of documents, and class labels were removed. As a result, the size of the dictionary amounted to $N = 3340$ words.

LANNIA revealed 39 factors within its 4 full cycles alternating ANNIA and LM, providing information gain $G = 0.12$. Three factors were found in the first cycle. Two of them had high intersections with two large classes of R52 labeled "earn" and "acq" containing 43% and 25% of the whole number of documents. The first factor was completely embedded in the first of them, with precision and recall amounted to $p = 0.94$ and $r = 0.43$. The second factor intersected with the second class with $p = 0.76$ and $r = 0.62$. During the next cycles of LANNIA these factors were replaced by smaller factors which provide the hidden structure of the dataset in more details. For example, the combination of particular six of 39 factors obtained at the last cycle provides intersection with the first class with $p = 0.91$ and $r = 0.92$ and the combination of another particular five factors provides intersection with the second class with $p = 0.78$ and $r = 0.83$.

The set of factors obtained at the first and the last LANNIA cycles intersects with all classes of R52 with micro-averaged F_1 score $F_1^{\text{micro}} = 0.55$. If to replace the 6 and 5 factors mentioned above by their combinations, then F_1^{micro} score increases to $F_1^{\text{micro}} = 0.72$. The mean number of factors mixed in one text amounts to 2. On average, one factor contains 5.8 words with $p_{ij} > 0.3$, 1.7 words with $p_{ij} > 0.5$, and only 0.5 words with $p_{ij} > 0.7$. On average q_j amounted to $4.5 \cdot 10^{-3}$. Thus the data are actually rather noisy.

It is interesting that division of R52 into classes made by experts provides smallest information gain $G = 0.09$. Note that LANNIA provides multi-assignment clustering of the dataset (for example, according to LANNIA each document is related to two topics on average) while experts prescribed each document to a single class. One may expect that multi-assignment clustering better corresponds to the data hidden structure than division of the dataset by classes without intersection.

6.3 Application to the Genome dataset analysis

One of the important problems in modern biology is to identify functions of proteins in the organisms. The extensive experimental studies are required to identify the function of even a single protein. Therefore, even for well-studied model organisms, the functions of the most proteins are yet unknown [63]. A fast growing number of organisms with fully sequenced genomes makes it possible to reveal the protein function by comparing protein phylogenetic profiles of different organisms. The protein phylogenetic profile is defined as a binary pattern that encodes by 1 and 0 the presence or the absence of proteins in a given organism with the fully sequenced genome, respectively [78]. When two proteins show the correlated events of the presence or absence over the organisms, it is assumed that these proteins are also functionally correlated. This idea is based on the observation that proteins seldom act as single isolated species to perform their functions. Usually a set of proteins is involved in each particular cellular process interacting in performing some function [92]. This leads to the concept of the modularity which assumes that the genome functionality can be partitioned into a collection of modules. Each module is a discrete entity of elementary components and performs an identifiable task, separable from the functions of the other modules [82]. Thus, revealing sets of proteins which coherently appear in different organisms may facilitate the search for functional modules in the genome structure. Recently there were many attempts to reveal the modular structure in Genome datasets by different blind statistical methods such as cluster analysis, independent component analysis and others (see [63] for review). Since the concept of the genome functional modularity is completely compatible with the BFA generative model described here it was a challenge for us to apply LANNIA to reveal the hidden factor structure in some large Genome dataset [2, 37, 40]. We consider its BFA analysis only as an example of the LANNIA application to the large real dataset and thus discuss only formal indexes of its performance such as the number of found factors, their relation to the protein phylogenetic profiles, and the information gain obtained which shows, particularly, the relevance of the BFA generative model to the genome data. The analysis of the factor contents and their relation to the known metabolic pathways is out of our competence and will not be discussed here.

For the BFA analysis the largest genome database KEGG [62] was used. It contained the fully sequenced genomes of $M = 1368$ organisms. The protein phylogenetic profile of each organism is a binary pattern $\mathbf{x}_m$ of dimension $N = 11451$, where N is a whole number of proteins taken into account (specifically, gene/protein ortholog groups). This number was obtained after excluding duplicates from the whole set of 14139 gene/protein ortholog groups of KEGG.

LANNIA revealed 38 factors after four full cycles of the combination of ANNIA and LM (steps 3–6 of the procedure presented in Section 4.3). Each cycle began by running twenty

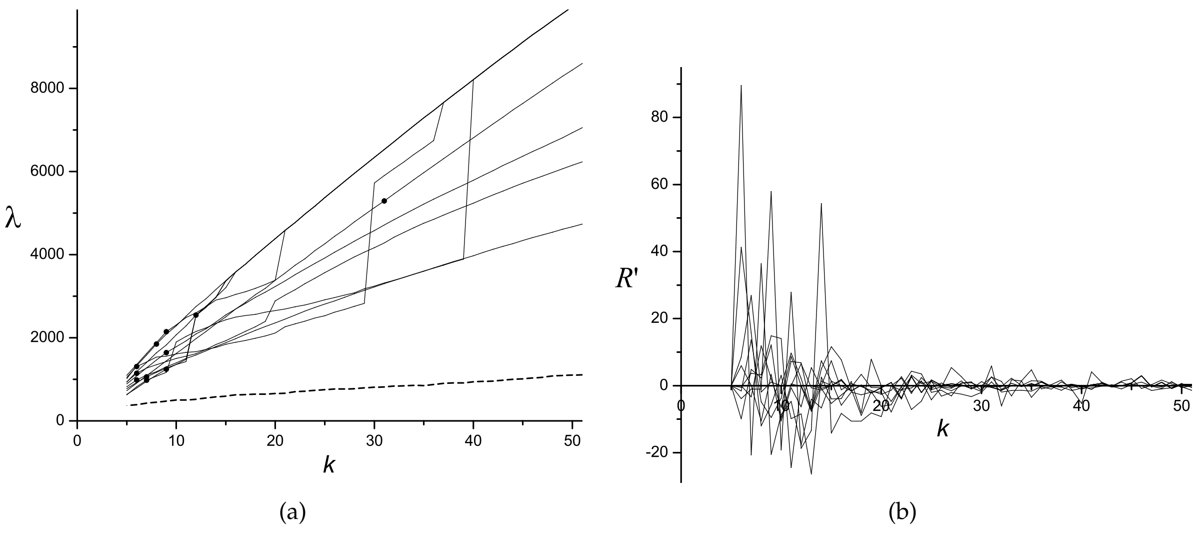


FIGURE 6.8: Lyapunov function λ (a) and function R' (b) depending on the number of active neurons k at the first cycle of LANNIA during the analysis of the KEGG dataset. Dashed line in (a) is a threshold for separating the true and spurious trajectories.

random trajectories in ANNIA. The Lyapunov functions along the eleven true trajectories at the first cycle of LANNIA are shown in Fig. 6.8(a). The peaks of R' used for the factor identification are shown in Fig. 6.8(b). During the first cycle the LM procedure converged for five steps and excluded two factors of eleven. The information gain provided after each LM step is shown in Fig. 6.9. At the fifth step it amounts to $G = 0.27$. At the second full LANNIA cycle ANNIA revealed fourteen factors. LM converged in four steps, excluding one factor and providing the gain increase up to $G = 0.31$ (Fig. 6.9). During the third and fourth full LANNIA cycles ANNIA revealed thirteen and twelve factors, LM excluded six and three, respectively. In each of the next cycle of LANNIA the growth of G was lower. During the fifth cycle the gain decreased and LANNIA was terminated. The maximal gain provided by LANNIA for the KEGG dataset amounts to 0.32. The relatively high gain obtained shows that, first, the genome data indeed correspond to the generative BFA model and, second, LANNIA is the efficient method for finding its parameters. The high information gain is in favor of the hypothesis of the modular genome structure.

Figure 6.10 demonstrates the distribution of proteins over found factors. We prescribed the protein j to the factor i if the probability p_{ij} exceeded a threshold p_{th}. The distributions are shown for three thresholds. The factors were ranged according to the decrease of the number of proteins constituting them for $p_{\text{th}} = 0.5$. On average one factor contains 235 proteins with $p_{ij} > 0.9$, 407 proteins with $p_{ij} > 0.7$ and 598 proteins with $p_{ij} > 0.5$. The number of proteins in the factors greatly exceeds the number of proteins in the metabolic pathways described in KEGG. Thus, it is unlikely that the factors correspond to metabolic pathways. Neither factor found by ANNIA contains more than 100 proteins. Thus, LM is able to enlarge the set of

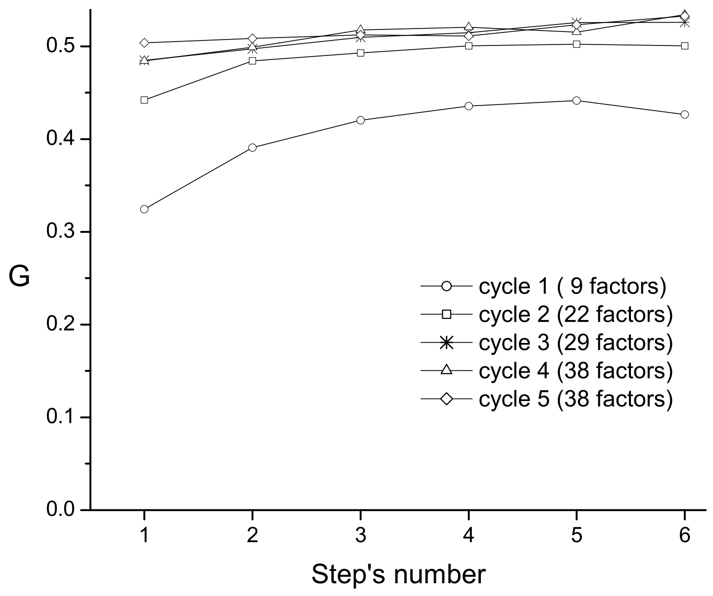


FIGURE 6.9: Increase of the information gain at each cycle of LANNIA during the analysis of the KEGG dataset.

elements constituting a factor comparing with ANNIA significantly. As mentioned above, it is also able to eliminate some factors found by ANNIA. Thus, the results obtained actually represent the synergy of both methods.

Figure 6.11 demonstrates the distribution of the factors over the organisms. All the organisms are grouped in types according to the taxonomy of KEGG from animals to bacteria. The type of the organisms is depicted by the number on the top of Fig. 6.11. The factors are ranged in descending order according to the frequencies of their appearance in the dataset. The factor number one appeared in the most organisms (in 22% of the organisms) and the factor number 38 appeared in the least of them (in 5% of the organisms). For the most factors the frequencies of their appearance in the organisms are distributed around 0.1. In Fig. 6.11 the appearance of a given factor in a given organism is marked by the point. Thus, the frequency of appearance of each factor in the dataset corresponds to the number of points in each horizontal line. Figure 6.11 demonstrates that each type of the organisms is characterized by the specific set of factors. For example, animals are characterized by factors 20 and 37, fungi by factor 20, plants by factors 2, 20 and 37 and so on. Factor 20 was identified only in eukaryotes and never in prokaryotes. Conversely, factor 1 was identified in all types of prokaryotes but never in eukaria. Thus, the distribution of factors over the types of organisms seems to reflect some peculiarities of their functioning. It is interesting that LANNIA revealed only little effect of specific factors: only 472 proteins over 11451 taken into account have q_j exceeding 0.01. Thus, almost all the organisms are completely described by common factors as predicted by the modular hypothesis.

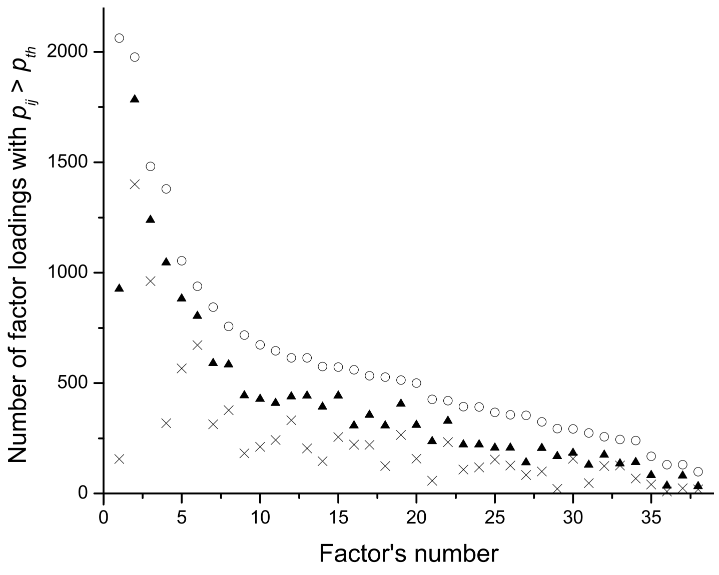


FIGURE 6.10: Distribution of number of proteins constituting factors found by LANNIA for three threshold values p_{th}, $p_{\text{th}} = 0.5$ - circles, $p_{\text{th}} = 0.7$ - triangles, $p_{\text{th}} = 0.9$ - crosses. Protein j was prescribed to factor i if $p_{ij} > p_{\text{th}}$. Factors are ranged according to decrease of the number of proteins for $p_{\text{th}} = 0.5$.

In order to compare LANNIA with other methods, we also applied BANNIA and BMFCA to KEGG. BANNIA is chosen only to demonstrate the role of LM procedure in LANNIA. BMFCA is chosen because it was perfect for noiseless data in solving BP (see Fig. 6.1), and the modular genome structure implies that the level of noise in data is small. Moreover, as shown in Fig. 6.11, the mean number C of modules mixed in genome of an organism is essentially larger than two, and BMFCA is happened to be less sensitive to increasing C (see Fig. 6.5).

BANNIA provided the maximal information gain $G = 0.35$ that was reached for 27 factors. The obtained gain is much lower than the gain provided by LANNIA. The relatively small informational gain provided by BANNIA is explained by the fact that it revealed only 7 independent factors and all other found factors happened to be similar to them. This confirms the result obtained for BP: BANNIA actually fails when the number of observations of the dataset is relatively small, whereas LANNIA is efficient also in this case.

The maximal information gain $G = 0.49$ obtained by BMFCA was reached for 84 found factors. The BMFCA efficiency in KEGG analysis is comparable with the results obtained by LANNIA.Since, as shown before, BMFCA is very sensitive to factor distortion, in contrast to LANNIA, one can expect that factors in KEGG dataset are mixed in observations with relatively small distortion. Indeed, according to LANNIA, each factor contains on average 140 proteins with $p_{ij} = 1$ and 235 proteins with $p_{ij} = 0.9$.

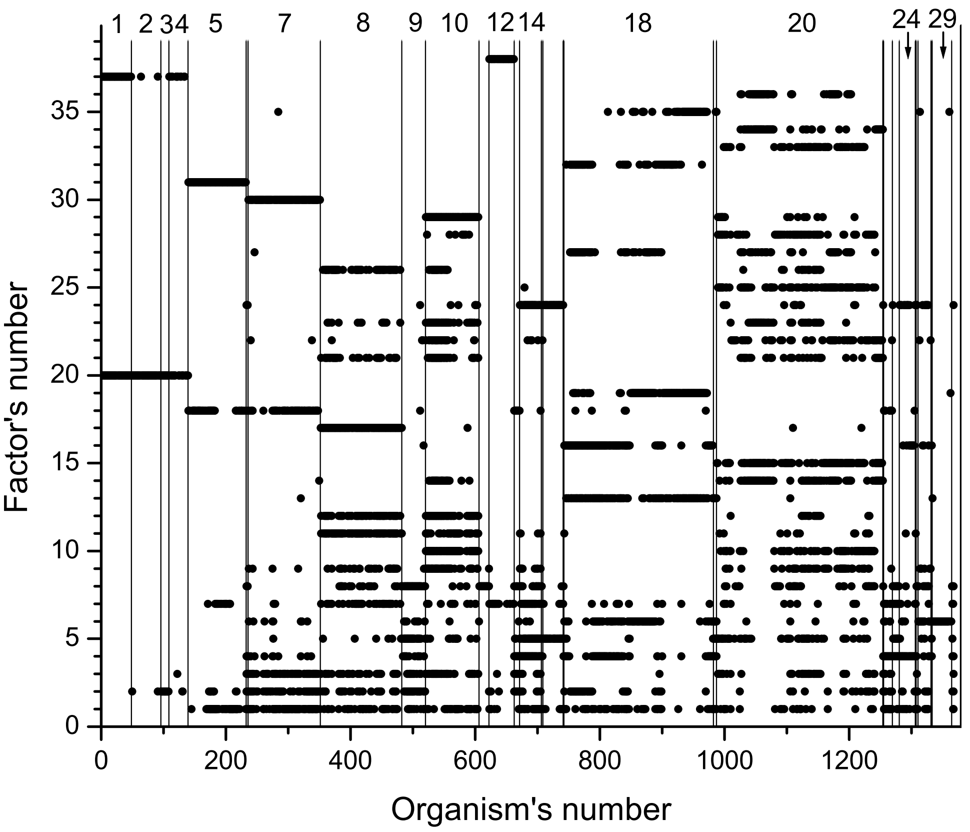


FIGURE 6.11: Distribution of factors over types of organisms. Eukaryotes: 1 – Animals, 2 – Fungi, 3 – Plants, 4 – Protists; Prokaryotes: 5 – Archaea; Bacteria: 6 – Acidobacteria, 7 – Actinobacteria, 8 – Alphaproteobacteria, 9 – Bacteroidetes, 10 – Betaproteobacteria, 11 – Chlamydiae, 12 – Cyanobacteria, 13 – Deinococcus-Thermus, 14 – Deltaproteobacteria, 15 – Elusimicrobia, 16 – Epsilonproteobacteria, 17 – Fibrobacteres, 18 – Firmicutes, 19 – Fusobacteria, 20 – Gammaproteobacteria, 21 – Gemmatimonadetes, 22 – Green nonsulfur bacteria, 23 – Green sulfur bacteria, 24 – Hyperthermophilic bacteria, 25 – Other proteobacteria, 26 – Planctomycetes, 27 – Spirochaetes, 28 – Synergistetes, 29 – Tenericutes, 30 – Verrucomicrobia.

Factors found by all methods happened to be similar. Similarity S_{ik} between factors $\mathbf{f}_i$ and $\mathbf{f}_k$ was estimated by a cosine of an angle between them:

$$S_{ik} = \sum_{j=1}^{N} p_{ij}p_{kj}/(\sum_{j=1}^{N} p_{ij}^2 \sum_{j=1}^{N} p_{kj}^2)^{1/2}.$$

Probabilities p_{ij} for factors found by BANNIA and BMFCA were estimated, using M-step of the LM procedure applied to factor scores obtained by these methods. Each factor found by BANNIA was compared with all the factors found by LANNIA. The highest cosine over all the LANNIA factors was treated as an index of similarity between this BANNIA factor and all the LANNIA factors and this index averaged over all the BANNIA factors was treated as an index of similarity between sets of BANNIA and LANNIA factors. The index amounts to 0.85 and BANNIA factors happened to be similar to the LANNIA factors found in the first its cycle. Note that if probabilities p_{ij} in each LANNIA factor are randomly permuted, then this index amounts to 0.18. The index of similarity between factors found by LANNIA and BMFCA amounts to 0.82. The coincidence of the results obtained by all methods is in favor of the conclusion that the Genome dataset actually has the latent factor structure which can be revealed by the considered methods, however LANNIA has an advantage because it provides the highest gain with the smallest number of factors.

To analyze KEGG database, about 4 hours of computational time are required for LANNIA and BANNIA, an hour for BMFCA.

6.4 Boolean factor analysis of Mushrooms dataset in comparison with classical approaches

Boolean factor analysis is useful for finding groups of similar binary variables. For example, as demonstrated in Section 6.2.1, when applied to textual data, it is capable of revealing topics as groups of coherent terms (keywords). Boolean factor analysis implies that each variable could be assigned to several factors and signals are composed by Boolean superposition of factors. In contrast, standard cluster analysis implies that all clusters are disjunctive, i.e., each variable belongs to only one cluster while linear factor analysis implies that signals are *linear* superpositions of factors. We suppose that for most binary data the first signal space model is more suitable. In this section, these three approaches are compared on a mushroom dataset that has been downloaded from the UCI web page (Repository of machine learning databases – [7]). A more detailed description of the experiment can be found in [42, 57].

The dataset includes descriptions of samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. However, identification of species is not included. Number of objects is 8124 and number of variables is 22 (all nominally valued). Each variable represents a physical characteristic (color, odor, size, shape etc.). The 23-rd variable indicates if the mushroom is edible or poisonous (this variable was used only for results interpretation). The numbers of edible and poisonous mushrooms are 4208 and 3916, respectively.

The data of the dataset were transformed into binary variables. Each categorical variable with K categories was transformed to K binary variables. The names of variables correspond to the order of attributes on the web source [7]. The individual categories are named by letters of the alphabet. The final dataset consists of 111 binary variables.

When the objects were clustered by some traditional clustering methods [69, 83], it turns out that 25 “pure” clusters (only with either edible or poisonous mushrooms) were identified. Therefore the expected number of clusters of binary variables also amounts to 25. We obtained the same results by both Jaccard and Dice coefficients. They are in Table 6.3.

Clus.	Var.	Clus.	Var.	Clus.	Var.	Clus.	Var.	Clus.	Var.
1	v01a	4	v01d	5	v22b	10	v03e	17	v15i
1	v05b	4	v02c	6	v01f	11	v03f	17	v17d
1	v21c	4	v03j	7	v02d	11	v05c	18	v09a
1	v22c	4	v04b	7	v03i	12	v03g	18	v12a
2	v01b	4	v05e	7	v07b	12	v19c	18	v13a
2	v02b	4	v09e	7	v09b	13	v03h	18	v21a
3	v01c	4	v10a	7	v14h	13	v05d	19	v09f
3	v02a	4	v12c	7	v15h	13	v14f	19	v20e
3	v03d	4	v13c	7	v20a	13	v15f	20	v09g
3	v04a	4	v14a	7	v21d	14	v09d	20	v17a
3	v05g	4	v15a	7	v22a	14	v14b	20	v20i
3	v06c	4	v19d	8	v03b	14	v15b	21	v09k
3	v07a	4	v20d	8	v09j	15	v09i	21	v18c
3	v08a	4	v22d	8	v14g	15	v14d	22	v20c
3	v09h	5	v01e	8	v15g	15	v15d	23	v20g
3	v12d	5	v03a	8	v21b	16	v06a	24	v05a
3	v13d	5	v05i	8	v22f	16	v09l	24	v13b
3	v17c	5	v08b	9	v03c	16	v14e	25	v05h
3	v18b	5	v09c	9	v05f	16	v15e	25	v22e
3	v19f	5	v10b	9	v14c	16	v17b		
3	v20b	5	v19b	9	v15c	16	v20f		
3	v21f	5	v20h	9	v18a	17	v12b		
3	v22g	5	v21e	9	v19e	17	v14i		

TABLE 6.3: Results of cluster analysis of the Mushrooms dataset.

The variables that are the linear combination of one of several other variables are not suitable for linear factor analysis. We analyzed both the same dataset as was used for cluster analysis and dataset with only $K - 1$ categories instead the variables with K categories. In both cases, we obtained the significant values (absolute values greater than 0.7) in the factor loading matrix for the same binary variables. From the reason of the comparison the results of factor and cluster analyzes, we present the results obtained for the same number of variables (see Table 6.4). In the table only variables with factor loadings exceeded 0.5 in absolute value are shown (corresponding values are highlighted).

In the first factor, we can found variables v05e, v12c, v13c, v19d and v20d with significant negative values of factor loadings. These variables are elements of the fourth cluster, see Table 6.3. For variables v04a, v12d, v13d and v19f positive values are significant. They are elements of the third cluster. In the second factor, we can found variables v06a, v09l, v14e, v15e and v17b with significant negative values of factor loadings. These variables are elements of the sixteenth cluster. In the third factor, we can found variables v05i, v08b, v09c, v19b, v20h and v21e with significant negative values of factor loadings. These variables are elements of the fifth cluster. Variables v07b, v14h, v15h and v22a are significant for fourth factor and belong to the seventh cluster. In the fifth factor, we can found variables v05f, v14c, v15c, v18a

Var.	**F1**	**F2**	**F3**	**F4**	**F5**
v04a	***0.65***	0.18	***0.52***	-0.27	-0.04
v04b	***-0.65***	-0.18	***-0.52***	0.27	0.04
v05d	-0.01	0.03	***-0.52***	-0.1	0.03
v05e	***-0.77***	0.08	0.04	-0.15	0.08
v05f	-0.02	-0.02	-0.01	-0.08	***-0.84***
v05g	***0.52***	-0.15	0.31	0.11	0.02
v05i	-0.01	0.03	***-0.52***	-0.1	0.03
v06a	0.03	***-0.96***	0.04	-0.03	-0.14
v06c	-0.03	***0.96***	-0.04	0.03	0.14
v07a	-0.09	-0.07	0	***-0.76***	-0.1
v07b	0.09	0.07	0	**0.76**	0.1
v08a	-0.05	-0.07	**0.81**	0.08	-0.07
v08b	0.05	0.07	***-0.81***	-0.08	0.07
v09c	-0.06	0.05	***-0.93***	-0.19	0.05
v09g	0.02	***-0.60***	0.02	-0.01	0.04
v09l	0	***-0.52***	0.01	-0.03	-0.3
v12c	***-0.80***	0.07	-0.17	-0.15	-0.1
v12d	**0.75**	-0.09	0.15	-0.1	0.03
v13c	***-0.80***	0.07	-0.16	-0.14	0.03
v13d	**0.72**	-0.1	0.1	-0.1	0.05
v14c	-0.02	-0.02	-0.01	-0.08	***-0.84***
v14e	0.03	***-0.99***	0.04	-0.02	0.04
v14h	0.39	0.19	-0.04	***0.59***	0.05
v15b	***-0.51***	0.03	0.19	-0.09	0.02
v15c	-0.02	-0.02	-0.01	-0.08	***-0.84***
v15e	0.03	***-0.99***	0.04	-0.02	0.04
15h	0.4	0.19	-0.03	***0.58***	0.06
v17a	0.02	***-0.69***	0.03	-0.01	0.03
v17b	0.02	***-0.69***	0.03	-0.01	0.03
v17c	-0.03	***0.98***	-0.03	0.01	-0.02
v18a	-0.02	-0.02	-0.01	-0.08	***-0.84***
v18b	-0.08	-0.03	-0.07	-0.36	***0.59***
v19b	0.02	0.08	***-0.84***	0.28	0.02
v19d	***-0.89***	0.05	0.32	-0.17	0.04
v19e	-0.02	-0.02	-0.01	-0.08	***-0.84***
v19f	***0.64***	-0.11	***0.56***	-0.13	0.06
v20d	***-0.81***	0.07	0.33	-0.1	0.07
v20h	-0.03	0.06	***-0.83***	0.05	-0.3
v21b	0.1	-0.39	0	0.09	***-0.61***
v21e	-0.09	-0.01	***-0.53***	-0.38	0.13
v22a	-0.13	0.1	0.24	***0.67***	0.08
v22b	-0.01	***-0.52***	-0.45	-0.05	0.03
v22g	0.24	0.16	0.12	***-0.61***	0.03

Table 6.4: Factor loadings matrix for the first five linear factors of the Mushrooms dataset.

$B1$	v09c v20h v08b v19b v04b v21e	~ $F3^{-}$, $C5$
$B2$	v13c v12c v05e v19d v20d v04b	~ $F1^{-}$, $C4$
$B3$	v19f v04a v12d v13d v05g v20b	~ $F1^{+}$, $C3$
$B4$	v15h v14h v22a v07b v21d v03i v02d	~ $F4^{+}$, $C7$

TABLE 6.5: Significant variables for the first four Boolean factors of the Mushrooms dataset.

and v19e with significant negative values of factor loadings. These variables are elements of the ninth cluster (only one element of this cluster – variable v03c is missing).

Four factors B_1–B_4 were identified by ANNIA. The variables composed these factors are shown in Table 6.5. In the right side of the table clusters and linear factors are given that are most overlapping with the binary factors. It is interesting to mention that variable v04b belonging to cluster C_4 is included to two Boolean factors (B_3 and B_2) and to two linear factors (F_1 and F_3). So in contrast to the clustering, both methods of factor analysis do not exclude the possibility of one element to belong to several factors.

The first factor corresponds to five variables from the fifth cluster, the exception is the variable v04b which is included also in the second factor that is close to the fourth cluster. That is the fourth cluster pulls this variable out of the fifth cluster. All variables of the second factor correspond to six variables from the fourth cluster, all variables of the third factor corresponds to six variables from the third cluster and all variables of the fourth factor correspond to seven variables from the seventh cluster.

To find the distribution of factors among mushrooms we calculate overlaps between all factors and all mushrooms. By our definition the overlap between l-th factor and m-th mushroom is the scalar production $(\mathbf{b}_l, \mathbf{a}_m)$ of vectors $\mathbf{b}_l$ and $\mathbf{a}_m$ where $\mathbf{b}_l$ is the binary vector of factor loadings and $\mathbf{a}_m$ is the binary vector presenting the mushroom features. The distributions of overlaps are shown in Fig. 6.12. All four histograms have two modes. The right mode corresponds to mushrooms which contain all or nearly all features of the given factor. The left mode corresponds to mushrooms only weakly intersected with the given factor. The vertical line is the separation border between two modes. We postulate that mushroom contains the given factor if its overlap with the factor exceeds the border.

The most contrast bimodal distribution is seen for factors B_1 and B_2. The m-th mushroom is assumed to contain one of these factors if it contains all factor's features, i.e., $(\mathbf{b}_l, \mathbf{a}_m) > 5$, $l = 1, 2$. The sets of mushrooms containing these factors are disjunctive that is each mushroom of these sets contains only one factor. It is interesting to note that these two sets contain only poisonous mushrooms. The first one consists of 1728 poisonous mushrooms, the second one consists of 1296 poisonous mushrooms (the whole collection contains 3916 poisonous mushrooms).

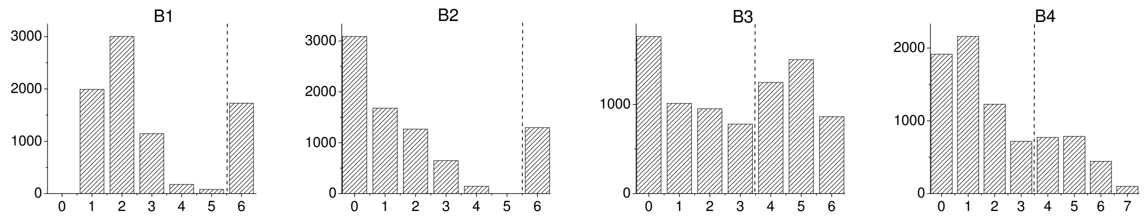


FIGURE 6.12: Distribution of mushrooms over Boolean factors.

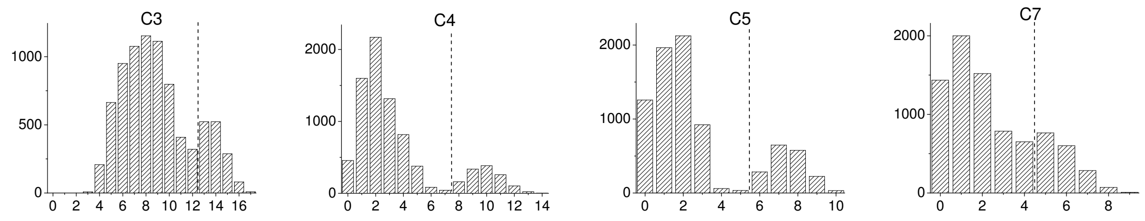


FIGURE 6.13: Distribution of mushrooms over clusters

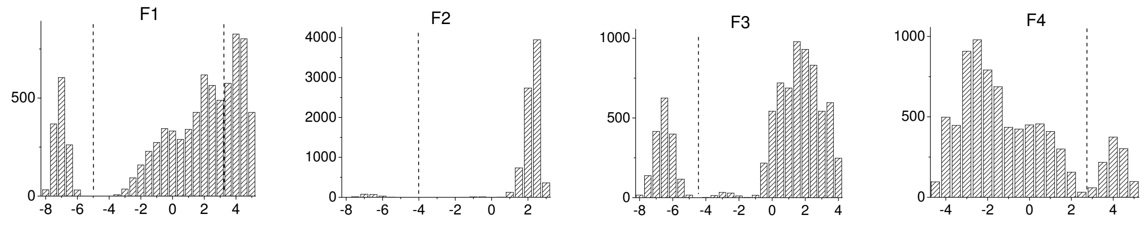


FIGURE 6.14: Distribution of mushrooms over linear factors

These same results was obtained with other methods. Fig. 6.13 demonstrates the distributions of the scalar productions $(\mathbf{c}_l, \mathbf{a}_m)$ where $\mathbf{c}_l$ is a vector with components equal to one or zero depending on whether the corresponding feature belongs or does not belong to the l-th cluster. We plot the distributions only for clusters $3, 4, 5, 7$ because other clusters contain only few variables. The distributions obtained for clusters C_4 and C_5 corresponding to factors B_2 and B_1, respectively, are most contrast. To distinguish mushrooms related and not related to C_4 we put the separation border between two modes to 7. Respectively, for C_5 the separation border was 5. Table 6.6 demonstrates the intersections between the sets of mushrooms corresponding to binary factors and clusters. The last row and the last column of the table contains the total number of mushrooms corresponding to factors and clusters, respectively. The sets of mushrooms corresponding to B_2 and C_4 (respectively to B_1 and C_5) almost coincide. As for binary factors B_1 and B_2, only poisonous mushrooms are contained in the sets for clusters C_4 and C_5.

Figure 6.14 demonstrates the distributions of the scalar productions $(\mathbf{f}_l, \mathbf{a}_m)$ where $\mathbf{f}_l$ is the

	$B1$	$B2$	$B3$	$B4$	
C5	**1728**	0	0	0	1764
C4	0	**1264**	0	0	1264
C3	0	0	**1424**	0	1424
C7	0	0	**708**	**1683**	1730
C8	0	0	174	0	174
C9	0	0	0	0	36
	1728	1296	3616	2102	8124

TABLE 6.6: Intersection between C and B mushroom sets.

	$B1$	$B2$	$B3$	$B4$	
$F3^-$	**1725**	0	0	0	1725
$F1^-$	0	**1296**	0	0	1296
$F1^+$	0	0	**2827**	**714**	2875
$F4^+$	0	0	175	**1066**	1066
$F2^-$	0	0	192	0	192
$F5^-$	0	0	0	0	36
	1728	1296	3616	2102	8124

TABLE 6.7: Intersection between Boolean and linear factors mushroom sets.

vector of factor loadings obtained by the linear factor analysis. In contrast to Boolean factor analysis, the components of $\mathbf{f}_l$ are positive and negative. Consequently, histograms of $(\mathbf{f}_l, \mathbf{a}_m)$ have positive and negative modes. The separation borders used to distinguish mushrooms related to positive and negative modes are shown in Fig. 6.14. Intersections between the sets of mushrooms corresponding to linear factors and binary factors are shown in Tab. 6.7. The sets of mushrooms corresponding to F_3^- and F_1^- almost coincide with B_1 and B_2, respectively, and these sets contain only poisonous mushrooms, too.

The sets of mushrooms related to factors B_3 and B_4 are constructed in more complicated way. For the separation borders shown in Fig. 6.14 they contain 3616 (86% edible) and 2102 (73% edible) mushrooms respectively. These sets have large intersection of 950 mushrooms which contain both B_3 and B_4. The ratios of edible mushrooms increase and intersection between two sets decreases when separation borders moves to the right. Thus as for poisonous mushrooms which split into two sets corresponding to factors B_1 and B_2, edible mushrooms separates into two sets, too. But in contrast, the sets of edible mushrooms have strong intersection: many of them contain both factors B_3 and B_4. Totally, these sets contain 4024 edible mushrooms of 4208 in the collection.

All the used methods discover four large mushrooms sets: two for poisonous and two for edible mushrooms. Two sets of poisonous mushrooms revealed by all methods are very similar.

	$F3^-$	$F1^-$	$F1^+$	$F4^+$	$F2^-$	$F5^-$	
C5	**1725**	0	0	0	0	0	1764
C4	0	**1264**	0	0	0	0	1264
C3	0	0	**1413**	0	0	0	1424
C7	0	0	**569**	**982**	0	0	1730
C8	0	0	111	0	0	0	174
C9	0	0	0	0	0	36	36
	1725	1296	2875	1066	192	36	8124

TABLE 6.8: Intersection between cluster and linear factors mushroom sets.

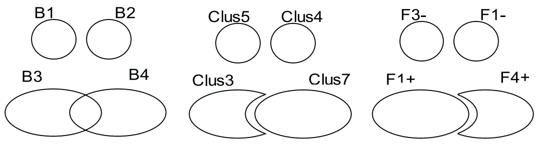


FIGURE 6.15: Shematic diagram of relations between mushrooms features sets.

This is resulted from the fact that these sets are not intersected. As for edible mushrooms, Tab. 6.6, 6.7, 6.8 show more complex interrelations between the sets of mushrooms obtained by different methods: $C_3 \subset B_3$, $F_1^+ \subset B_3$, $C_7 \subset B_4$, $F_4^+ \subset B_4$, $C_3 \subset F_1^+$, $F_4^+ \subset C_7$. If to take into account that $F_1^+ \cap F_4^+ = \varnothing$ and also $C_3 \cap C_7 = \varnothing$, the possible interpretation of these interrelations is schematically shown in Fig. 6.15. There exists two partially overlapping classes between edible mushrooms which were revealed by Boolean factor analysis. Linear factor analysis or clustering split edible mushrooms into two disjunctive sets joining the intersection to the set corresponding to B_3 in the case of linear factor analysis and to the set corresponding to B_4 in the case of clustering. These peculiarities results from foundations of these methods. In the case of clustering, the feature can belong only to one of clusters. In the case of linear factor analysis, the vectors of factor loadings should be orthogonal. Thus the advantage of BFA approach is that it provides the separation of edible mushrooms into two overlapping sets while traditional methods split them into different disjunctive groups joining intersection to one or another set.

6.5 Boolean factor analysis of parliament voting

The analysis was performed for results of roll-call votes in the Russian parliament in 2004. Each vote is considered as a binary vector with component 1 if the correspondent deputy voted affirmatively and 0 negatively. The number of votes during the year amounts 3150. The number of deputies (consequently the dimensionality of signal space and network size) amounts 430 (20 deputies voted less than 10 times were excluded from the analysis). A more detailed description of the experiment can be found in [30, 52, 54].

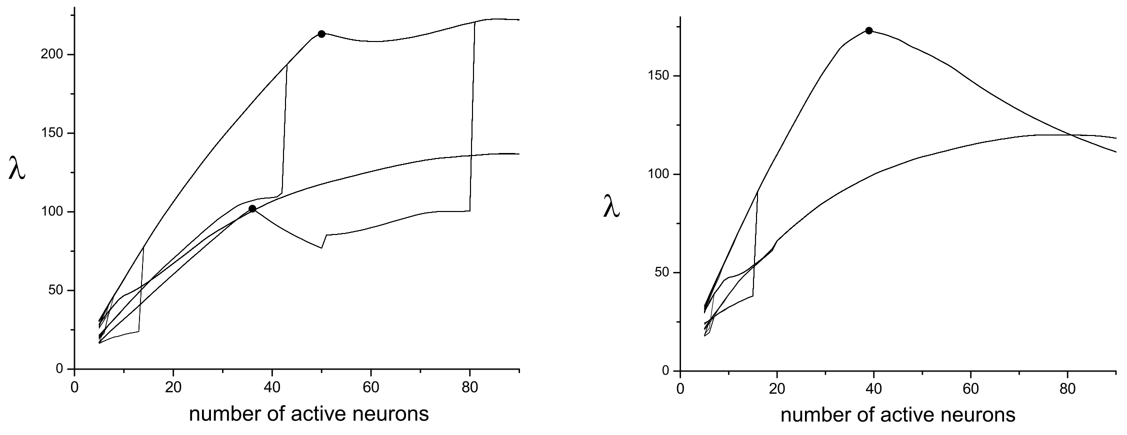


FIGURE 6.16: Lyapunov function λ for parliament data in dependence on the number of active neurons for initial trajectories (a), for trajectories after deleting two first factors (b). The points on the curves are factors found by the peaks of R'.

Figure 6.16(a) shows the Lyapunov function along trajectories of ANNIA starting from 1500 random initial states. All these states converge to four trajectories. Two of them have breaks in the points in which R' has a peak, the values of λ exceed $h_{\max}$ in (4.9), and therefore these points were identified as two factors. The factor with the highest Lyapunov function contains 51 deputies and completely coincides with the fraction of the Communist Party (CPRF). Another factor contains 36 deputies. All of them belong to the fraction of Liberal-Democratic Party (LDPR) which contains totally 37 deputies. Thus one of the members of this fraction fell out of the corresponding factor. The pointed sharp breaks at the corresponding trajectories give evidence that these fractions are most discipline and their members vote coherently.

Figure 6.16(b) demonstrates trajectories after deleting of the two factors by unlearning the rule (4.14). Starting from 1500 initial states they converge to only two trajectories. One of them has a break but it is not so sharp as for CPRF and LDPR factors. The derivative R' has a maximum in this point, $\lambda > h_{\max}$, and therefore this point was identified as third factor. The factor contains 37 deputies. All of them belong to the fraction "Motherland" (ML) which contains totally 41 deputies. Thus 4 of its members fell out of the factor. The fuzziness of break at the trajectory gives evidence that this fraction is not so homogeneous as the two first ones and actually the fraction split at two fractions in 2005.

Matching of neurons along the second trajectory in Fig. 6.16(b) with the list of deputies has shown that it corresponds to the fraction "United Russia" (UR). This fraction is the largest and contains totally 285 deputies but is less homogeneous. Therefore the Lyapunov function along the trajectory is low and it has no break at all.

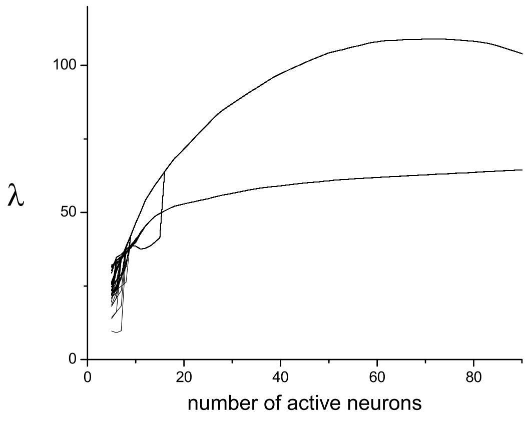


FIGURE 6.17: The same as in Figs. 6.16(a) and 6.16(b) after deleting three first factors.

fractions/factors	1	2	3	4	5
UR	283	0	0	0	2
CPRF	0	51	0	0	0
LDPR	1	0	36	0	0
ML	3	0	0	37	1
Independent	1	0	0	0	15

TABLE 6.9: Relation between parliament fractions and factors.

Fig. 6.17 shows trajectories of neurodynamics after additional deleting the third factor from the network. Two remaining trajectories contain members of UR and independent deputies (ID). The upper trajectory contains only members of UR and lower one – mainly ID but also members of UR. This is additional evidence of heterogeneity of UR. Factors UR and ID were identified by minimums of the second derivatives along the corresponding trajectories. The general relation between the parliament fractions and obtained factors is shown in Table 6.9.

The fit between the fractions and the factors was estimated by F_1-measure. Averaged over all fractions it amounted to 0.98.

Here factors do not overlap so we may interpret them as clusters and compare our results with those obtained by some traditional methods of clustering. First, we performed clustering of the parliament members with the direct use of similarity matrix. Similarity between two deputies was calculated by comparison of vectors of their voting. We used different measures of similarity: Euclidian distance, cosine, Jaccard and Dice. Both hierarchial and K-means clustering gave clusters far from parliament fractions: all fractions intersected in clusters and fraction LDPR could not be separated from ER at all.Second, we performed mapping of parliament members by the method of multidimensional scaling. The results are shown in Fig.6.18. This map was clustered. The border of clusters are shown by thin lines. Generally,

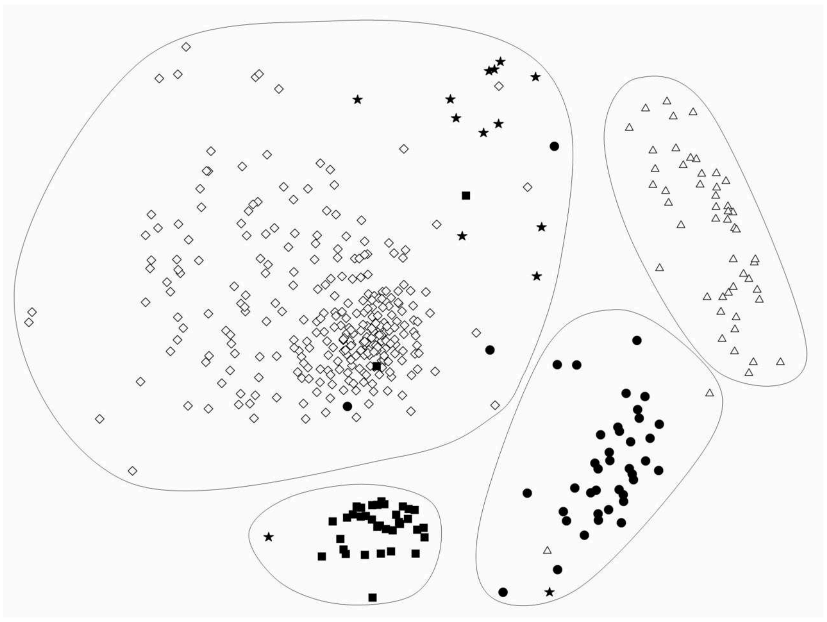


FIGURE 6.18: Two-dimensional map of parliament members vote. Thin lines - borders of clusters. $\diamond$ - UR, $\triangle$ - CPRF, $\blacksquare$ - LDPR, $\bullet$ - ML, $\star$ - ID.

fractions/clusters	1	2	3	4
UR	285	0	0	0
CPRF	0	49	0	2
LDPR	2	0	35	0
ML	3	0	0	38
Independent	14	0	1	1

TABLE 6.10: Relation between parliament fractions and clusters from Fig. 6.18.

as factors obtained before, clusters coincide with parliament fractions except for independent deputies. The results of clustering and factorization are compared in the Table 6.10. The mean F_1-measure amounted to 0.95, that is slightly smaller than that obtained for factors.

Chapter 7

Conclusion

The present work is devoted to Boolean factor analysis (BFA). The BFA problem is formulated in conjunction with definition of linear factor analysis in Chapter 2. In order to distinguish BFA from other related problems (like Boolean matrix factorization) the BFA generative model is suggested in Section 2.2. The information gain that estimates the elimination of information redundancy is suggested in Section 2.3 as criterion of optimality of BFA solution. This criterion allows to estimate BFA performance even in the case when precise solution is unknown, i.e., in the case of real-world datasets. It is shown that the information gain reaches maximum when BFA solution is right.

The main goal of this work was to develop an efficient method for BFA. The method is suggested in Chapter 4. It is based on an original Hopfield-like attractor neural network with increasing activity (ANNIA) described in Section 4.1. ANNIA is very similar to the traditional sparsely encoded Hopfield network [26] but has an original two-run recall procedure and several special techniques for factors identification. The properties of ANNIA are studied in Chapter 5 both theoretically and by computer simulations. The existence of two global spurious attractors was revealed in the experiments presented in Section 5.2. These attractors are created by neurons most and least often contained in the whole set of factors (and therefore in the dataset) and become dominant only when signal complexity is rather high. That is why they have never been observed before and pose a completely new phenomenon of attractor neural networks. When the global spurious attractors dominate in the network dynamics they prevent factors revealing. To suppress their dominance one special inhibitory neuron is added to the principal neurons of the network. It is shown that the activity of special inhibitory neuron completely eliminate the global spurious attractors.

The size of attraction basins around true attractors are estimated in Section 5.3 by computer simulations and by the single-step approximation proposed by [67] for the densely encoded Hopfield network. It turns out that for the sparsely encoded network the critical relative informational loading $\alpha_{\text{cr}} \approx 0.3$. The estimation of the attraction basins is useful only when the network dynamics starts from the corrupted version of one of the factors. The probability to find a factor when the network dynamics starts from a random initial state is estimated in Section 5.4. This probability happened to be high only when the number of factors L is less than two limits L_1 and L_2. The first limit L_1 corresponds to the number of factors L which provides equal Lyapunov function values for true and spurious attractors. If $L > L_1$ the spurious attractors dominate in network dynamics. If $L < L_1$ this value is larger for true attractors but spurious attractors can continue to dominate because of their large number. To avoid this kind of dominance, L must be smaller than L_2. For the level of sparseness of the factors encoding $p = 0.02$, these limits are estimated as $L_1 = 2.8N$ and $L_2 \simeq 10^3 \ln(0.0014\, N)$. Even in the case when the probability of true attractor is high, the search of all factors could be impossible due to the dominance of some of them. As shown in Section 5.5, this difficulty can be easily overcome by Hebbian unlearning of found factors. It is shown that ANNIA with unlearning rule is able to reveal the complete set of factors mixed in the patterns of the dataset.

Although ANNIA provides an accurate estimation of the factor loadings (as set of active neurons in true attractor), but it provides only approximate estimation of factor scores, and no estimation of the parameters of the generative model p_{ij} and q_j. This drawback is eliminated by combining ANNIA with likelihood maximization (LM) described in Section 4.2. It is shown in Section 4.3 that the LM procedure itself is able to provide complete solution of the BFA problem but requires an appropriate initial approximation: if it starts from the random initial parameters it commonly fails. In the combination of ANNIA and LM the role of ANNIA is to provide LM with the initial approximation. Another aspect of the ANNIA and LM interaction is a suppression of the dominant attractors in ANNIA using the data provided by LM. The resulting hybrid ANNIA and LM procedure (LANNIA) is presented in Section 4.3. The whole algorithm is presented in Appendix D.

The developed method LANNIA was tested on real-world binary datasets in Chapter 6. First, the performance of the method was tested on text datasets in Section 6.2. Each text document of the dataset was presented as a binary vector of the dimension of the used term dictionary. Each component of the vector was 1 or 0 depending on the presence or absence of the corresponding term in the document. The revealed factors were interpreted as topics of the dataset: the values 1 or 0 in factor loadings correspond to whether the related term belongs to the topic, the values 1 or 0 in factor scores encode whether the text document belongs to the topic. The method was applied to two types of textual data on Neural Networks in two different languages (Section 6.2.1). It is demonstrated that the obtained topics and corresponding words are at a good level of agreement despite the lexical specificity of the English and Russian languages. Each revealed topics was presented as a set of weighted words. These words turn out to be semantically close, which allows to give an interpretation of each topic.

The information gain calculated for one of the dataset turns out to be rather high ($G = 0.15$), while the grouping of the articles across scientific sections produced by Program Committees provided informational gain of only 0.04. The method was also tested on Reuters R52 dataset of news messages. As for previous dataset, the obtained information gain was rather high ($G = 0.12$), and it was higher than the information gain calculated for division of R52 into classes made by experts amounted to 0.09. The intersection between classes of R52 made by experts and topics obtained by LANNIA was evaluated using micro-averaged F_1 score. The obtained value happened to be very high $F_1^{\text{micro}} = 0.72$. Thus, for text datasets we observed high values of information gain, stability of the results when changing the dataset, strong intersect with manual classification. All these are in favor of the hypothesis that text data is appropriate to BFA generative model.

The method was applied to the problem of revealing the modular structure in genome datasets in Section 6.3. It is assumed that solving this problem would allow to identify the functions of proteins in organisms. For the BFA analysis the largest genome database KEGG containing the fully sequenced genomes of organisms was used. LANNIA revealed 38 factors with very high information gain $G = 0.32$. The high gain shows that, first, the genome data indeed correspond to the generative BFA model and, second, LANNIA is the efficient method for finding its parameters. It is demonstrated that each type of the organisms is characterized by the specific set of factors. The analysis of information loadings of the revealed factors led to the conclusion that the influence of specific factors (noise) is minimal, and almost all the organisms are completely described by common factors as predicted by the modular hypothesis.

The method was also tested on the database of roll-call votes in the Russian parliament in Section 6.5 and on the mushrooms dataset in Section 6.4. In both cases, a clear interpretation of the factors revealed by the method was found: for parliament voting, factors correspond to the fractions of deputes, for mushrooms dataset, factors correspond to groups of edible and poisonous mushrooms. The BFA was compared with classical approaches: linear factor analysis and clustering method. It was demonstrated on mushrooms dataset that in the case of partially overlapping classes, linear factor analysis and clustering split classes into disjunctive sets joining the intersection to one or another class, while BFA correctly reveals these classes as overlapping sets.

The performance of LANNIA was compared with other BFA related method in solving the bars problem in Section 6.1. It was shown that only LANNIA provides almost the right bars problem solution for all analyzed bars problem tasks. It is least sensitive to noise in data, to the number of factors mixed in each observation and to the insufficient number of observations in the dataset. Other methods exhibit good performance only in part of the tasks. The execution time of LANNIA in solving the bars problem was comparable with the fastest BFA related method. The computational complexity of the method is only quadratic in the dimension of signal space and linear in the number of observation. Thus the method could be applied for analyzing large datasets, for example, to analyze KEGG database containing 1368 observation of dimension 11451, about 4 hours of computational time are required.

7.1 Suggestions for future work

In the future, I would like to apply the LANNIA to role mining problem, market basket analysis and other actual problems. Although the computational complexity of the method is only quadratic in the dimension of signal space, the amount of operative memory required in ANNIA program increases with the square of the dimension of signal space, for example, when the number of binary variables reaches 10^5 the program requires 20 GB of RAM. On the other hand, datasets of such a large dimension are usually sparse, i.e., the frequencies of most variables are low. In this case the matrix of synaptic connection $\mathbf{J}$ containing connection weights between each pair of neurons can be replaced by sparsely encoded matrix containing for each variable connection weights with only small part of other variables. The implementation of this idea will be the object of future work.

Next, I would like to implement the method as a program or library for some statistical packages and in MATLAB. This would give the opportunity to try the new perspective statistical method to everyone.

The BFA generative model has some assumptions. The most limiting assumption is statistical independence of appearance of factors in observations stated by (2.9). I would like to extend this assumption to different types of relationships between factors: hierarchy of factors, causal relationships, and others.

Appendix A

Estimation of variance of a_m

By definition (4.1), $a_m = (1/N)\sum\limits_{i=1}^{N} x_{mi}$. Then

$$\mathrm{D}\{a_m\} = \frac{1}{N}\mathrm{D}\{x_{mi}\} + \frac{N-1}{N}\mathrm{Cov}\{x_{mi}, x_{mj}\}\,,$$

where $\mathrm{D}\{x_{mi}\} = \bar{a}(1-\bar{a})$. Covariance between x_{mi} and x_{mj} can be presented in the form $\mathrm{Cov}\{x_{mi}, x_{mj}\} = \langle(1-x_{mi})(1-x_{mj})\rangle - (1-\bar{a})^2$. Since factors are statistically independent

$$\langle(1-x_{mi})(1-x_{mj})\rangle = \langle \prod_{l\in\{l:s_{ml}=1\}} (1-f_{li})(1-f_{lj})\rangle = [(1-p)(1-p^*)]^C\,,$$

where $1-p^* = ((1-p)N-1)/(N-1)$ is a probability that the j-th neuron is not active in a given factor under the condition that the i-th neuron is not active in this factor. Thus

$$\mathrm{Cov}\{x_{mi}, x_{mj}\} = (1-q)^2[(1-\frac{p}{(1-p)(N-1)})^C - 1] \simeq -\frac{(1-q)^2 pC}{(1-p)N},$$

that means, that $\mathrm{D}\{a_m\}$ is of order $1/N$.

Appendix B

Estimation of variance of J'_{ij}

$\mathrm{D}\{J'_{ij}\}$ can be presented in the form

$$\mathrm{D}\{J'_{ij}\} = ME_1 + M(M-1)E_2 - (\langle J'_{ij}\rangle)^2 \,, \tag{B.1}$$

where

$$\begin{aligned} E_1 &= \langle (x_{mi} - a_m)^2 (x_{mj} - a_m)^2 \rangle, \\ E_2 &= \langle (x_{mi} - a_m)(x_{mj} - a_m)(x_{li} - a_l)(x_{lj} - a_l) \rangle, \qquad l \neq m \end{aligned}$$

and according to (5.2) $\langle J'_{ij}\rangle = M\langle (x_{mi} - a_m)(x_{mj} - a_m)\rangle = -M\bar{a}(1-\bar{a})/(N-1)$.

To estimate E_1 and E_2 we ignore the statistical dependence between x_{mi} and x_{mj} which results from the fact that the number of active neurons in factors is fixed and equal to n (the correlation coefficient between these variables is of the order $1/N$, see Appendix A). Then $E_1 = q^2(1-q)^2$. To estimate E_2 one must take into account the statistical dependence between the activities of the same neurons in different patterns of the learning set. This dependence results from the fact that different neurons are differently presented in a set of factors. Thus, the neurons which are contained in more factors have higher probability to be active in both learning patterns $\mathbf{x}_m$ and $\mathbf{x}_l$. In order to take into account this dependence explicitly, let us introduce probability

$$P(C_1) = \binom{C}{C_1}\binom{L-C}{C-C_1}\Big/\binom{L}{C}$$

that the given pair of signals have C_1 common factors. Then

$$E_2 = \sum_{C_1} P(C_1) E^2(C_1) \,,$$

where

$$E(C_1) = \langle (x_{mi} - a_m)(x_{li} - a_l) \rangle|_{C_1} = \langle (1 - x_{mi})(1 - x_{li}) \rangle|_{C_1} - (1 - q)^2.$$

Due to the independence of different factors

$$\langle (1 - x_{mi})(1 - x_{li}) \rangle|_{C_1} = (1 - p)^{C_1}(1 - p)^{2(C - C_1)} = (1 - \bar{a})^2/(1 - p)^{C_1},$$

i.e., $E(C_1) = (1 - q)^2[(1 - p)^{-C_1} - 1]$ and after approximation of $P(C_1)$ by Poisson distribution $P(C_1) \simeq \mu^{C_1} \exp(-\mu)/C_1!$, where $\mu = C^2/L$, and taking into account that for any a

$$\sum_{C_1} a^{C_1} \mu^{C_1} \exp(-\mu)/C_1! = \exp(\mu(a - 1))$$

one can immediately obtain

$$\begin{aligned} E_2 &= (1 - \bar{a})^4 \Big[\exp\big(\mu(\frac{1}{(1 - p)^2} - 1)\big) - 2 \exp\big(\mu(\frac{1}{1 - p} - 1)\big) + 1 \Big] \\ &= (1 - \bar{a})^4 p^2 \mu G_1(\mu)/[(1 - p)^2] \, , \end{aligned}$$

where $G_1(\mu)$ is given by (5.10). Since the first term in (B.1) is of the order M, the second one is of the order M^2 and the third one is of order $(M/N)^2$, the first and third terms can be neglected when compared with the second one. Hence $\mathrm{D}\{J'_{ij}\}$ is given by (5.9).

It must be noted that the estimation of $\mathrm{D}\{J'_{ij}\}$ by formula (5.9) is valid only when $1 - \bar{a}$ is not extremely small because the first term in (B.1) is of the order $(1 - \bar{a})^2$, the second of $(1 - \bar{a})^4$ and the third of $(1 - \bar{a})^2$. Since $1 - \bar{a} \approx \exp(-pC)$, we assume that pC is not extremely large to provide the condition that $(1 - \bar{a})^2$ and $(1 - \bar{a})^4$ are of the same order.

Appendix C

Estimation of variance of J_{ij}

The estimation is based on the same methodology as variance $\mathrm{D}\{J'_{ij}\}$ in Appendix B. According to (4.1), (4.3) and (4.4)

$$\begin{aligned} J_{ij} &= J'_{ij} - J''_{ij} = \sum_{m=1}^{M} (x_{mi} - a_m)(x_{mj} - a_m)) - M(r_i - \bar{a})(r_j - \bar{a}) \\ &\simeq \sum_{m=1}^{M} (x_{mi} - \bar{a})(x_{mj} - r_j) = \sum_{m=1}^{M} (1 - x_{mi})((1 - x_{mj}) - (1 - r_j))\,, \end{aligned}$$

where r_i is the probability that neuron i is active in a learning pattern. Then $\mathrm{D}\{J_{ij}\} \simeq M^2 E_2^{\mathrm{inh}}$, where

$$\begin{aligned} E_2^{\mathrm{inh}} &= \langle (1 - x_{mi})((1 - x_{mj}) - (1 - r_j))(1 - x_{li})((1 - x_{lj}) - (1 - r_j)) \rangle \\ &= \langle (1 - x_{mi})(1 - x_{mj})(1 - x_{li})(1 - x_{lj}) \rangle - \langle (1 - x_{mi})(1 - r_j)(1 - x_{li})(1 - x_{lj}) \rangle \\ &- \langle (1 - x_{mi})(1 - x_{mj})(1 - x_{li})(1 - r_j) \rangle + \langle (1 - x_{mi})(1 - r_j)(1 - x_{li})(1 - r_j) \rangle \\ &\simeq \langle (1 - x_{mi})(1 - x_{mj})(1 - x_{li})(1 - x_{lj}) \rangle - [\langle (1 - x_{mi})(1 - x_{li}) \rangle]^2 \\ &= \langle (x_{mi} - a_m)(x_{mj} - a_m)(x_{li} - a_l)(x_{lj} - a_l) \rangle - [\langle (x_{mi} - a_m)(x_{li} - a_l) \rangle]^2\,, \end{aligned}$$

where we took into account that $\langle (1 - x_{mi})(1 - r_i) \rangle \simeq \langle (1 - x_{mi})(1 - x_{li}) \rangle$. The terms in the last equation were obtained in Appendix B, and the substitution of them immediately gives (5.11).

Appendix D

Algorithm of LANNIA

Algorithm 1: Hybrid algorithm of ANNIA and LM for BFA

input : $\mathcal{X}$ is a collection of M row binary vectors $\mathbf{x_m}$ of dimension N with components x_{mj}, $m = 1, \dots, M$, $j = 1, \dots, N$

output: $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \dots, L, j = 1, \dots, N)$ are parameters of generative model, $\mathcal{S}$ is a collection of row binary vectors of factor scores $\mathbf{s}_m$ of dimension L with components s_{ml}, $m = 1, \dots, M$, $l = 1, \dots, L$

```
1  begin
2      for $m \leftarrow 1$ to $M$ do $a_m \leftarrow \sum_{j=1}^{N} x_{mj}/N$
3      for $i \leftarrow 1$ to $N$ do
4          for $j \leftarrow 1$ to $N$ do
5              $\tilde{J}_{ij} \leftarrow \sum_{m=1}^{M} (x_{mi} - a_m)(x_{mj} - a_m) - \frac{1}{M} \sum_{m=1}^{M} (x_{mi} - a_m) \sum_{m=1}^{M} (x_{mj} - a_m)$
6          $\tilde{J}_{ii} \leftarrow 0$
7      $G \leftarrow 0, \quad L \leftarrow 0$
8      for $j \leftarrow 1$ to $N$ do $q_j \leftarrow 1/M$
9      repeat
10         $G^{old} \leftarrow G$
11         if $L = 0$ then
12             $\mathbf{J} \leftarrow \tilde{\mathbf{J}}$
13         else
14             for $l \leftarrow 1$ to $L$ do
15                 for $i \leftarrow 1$ to $N$ do $p^0_{li} \leftarrow \sum_{m=1}^{M} x_{mi}(1 - s_{ml}) / \sum_{m=1}^{M} (1 - s_{ml})$
16                 for $i \leftarrow 1$ to $N$ do
17                     for $j \leftarrow 1$ to $N$ do
18                         $J_{ij} \leftarrow \tilde{J}_{ij} - M\pi_l(1 - \pi_l)p_{li}(1 - p^0_{li})p_{lj}(1 - p^0_{lj})$
19                     $J_{ii} \leftarrow 0$
20         $\mathcal{F} \leftarrow$ FactorsRevealing($\mathbf{J}$)
21         if $\mathcal{F} = \varnothing$ then break algorithm
22         foreach $\mathbf{f} \in \mathcal{F}$ do
23             $L \leftarrow L + 1$
24             $\mathbf{h} \leftarrow \mathbf{fJ}$
25             for $j \leftarrow 1$ to $N$ do
26                 if $h_j > 0$ then $p_{ij} \leftarrow 0.99 h_j / \max_i(h_i)$ else $p_{ij} \leftarrow 0$
27         $[\mathbf{\Theta}, \mathcal{S}] \leftarrow$ LikelihoodMaximization($\mathcal{X}, \mathbf{\Theta}$)
28         $G \leftarrow$ InformationGain($\mathcal{X}, \mathbf{\Theta}, \mathcal{S}$)
29     until $G < G^{old}$
```

Procedure FactorsRevealing($\mathbf{J}$)

input : $\mathbf{J}$ is a matrix of synaptic connections of dimensionality $N \times N$
output : $\mathcal{F}$ is a collection of found factors in the form of row vectors
parameters: n_{in} and n_{fin} are initial and final number of active neurons along each trajectory, K_{trial} is number of trials of factor search, θ_{ov} is minimal overlap between two network states subsequently appeared in neurodynamics that are considered to be in the same attractor basin

```
begin
  $\mathcal{F} \leftarrow \varnothing$
  repeat $K_{trial}$ times
    generate perm, a random permutation from 1 to $N$
    Initialize $\mathbf{x}$ and $\mathbf{f}$ as row zero vector of dimension $N$
    for $i \leftarrow 1$ to $n_{in}$ do  $x_{\texttt{perm}(i)} \leftarrow 1$
    $R'_{max} \leftarrow 0$
    for $k \leftarrow n_{in}$ to $n_{fin}$ do
      Initialize $\mathbf{x}^1$ and $\mathbf{x}^2$ as row zero vector of dimension $N$
      while $\mathbf{x} \neq \mathbf{x}^1$ and $\mathbf{x} \neq \mathbf{x}^2$ do
        $\mathbf{x}^2 \leftarrow \mathbf{x}^1$
        $\mathbf{x}^1 \leftarrow \mathbf{x}$
        $\mathbf{h} \leftarrow \mathbf{xJ}$ // row vector of synaptic excitations
        $\mathbf{x} \leftarrow \mathbf{0}$
        for $i \leftarrow 1$ to $k$ do
          $j \leftarrow \texttt{argmax}_l(h_l)$
          $x_j \leftarrow 1, \quad h_j \leftarrow -\infty$
      $\lambda \leftarrow \mathbf{x}^1\mathbf{J}\mathbf{x}^T/k, \quad T \leftarrow \max_l(h_l)$
      $R \leftarrow \lambda/(k-1) - T/k$
      if $k > k_{in}$ and $\max_{i,j=1,2}(\mathit{Overlap}(\mathbf{x}^i, \mathbf{x}^j_{prev})) > \theta_{ov}$ then
        $R' \leftarrow R - R_{prev}$
        if $R' > R'_{max}$ then
          $R'_{max} \leftarrow R', \quad \mathbf{f} \leftarrow \mathbf{x}, \quad \lambda_f \leftarrow \lambda, \quad n_f \leftarrow k$
      $R_{prev} \leftarrow R, \quad \mathbf{x}^1_{prev} \leftarrow \mathbf{x}^1, \quad \mathbf{x}^2_{prev} \leftarrow \mathbf{x}^2$
      $x_{\texttt{argmax(h)}} \leftarrow 1$ // activating non-active neuron with maximal synaptic excitation
    if $\mathbf{f} = \mathbf{0}$ or $\mathbf{f} \in \mathcal{F}$ then goto line 3
    $h_{thr} \leftarrow$ SpurTrueThreshold($n_f$, $\mathbf{J}$) // threshold separating true and spurious attractors
    if $\lambda_f > h_{thr}$ then add $\mathbf{f}$ to $\mathcal{F}$
```

Function Overlap($\mathbf{x}, \mathbf{y}$)

input : $\mathbf{x}$ and $\mathbf{y}$ are binary vectors of dimensionality N so that $|\mathbf{x}| \leq |\mathbf{y}|$ where $|\mathbf{x}|$ is the number of entries in $\mathbf{x}$ equal to one
output: Ov is an overlap between $\mathbf{x}$ and $\mathbf{y}$

```
begin
  $Ov \leftarrow \frac{1}{|\mathbf{x}|(1-|\mathbf{y}|/N)} \sum_{i=1}^{N} (x_i - |\mathbf{x}|/N)(y_i - |\mathbf{y}|/N)$
```

Function SpurTrueThreshold(n, $\mathbf{J}$)

input : n is a number of components equal to one in random vectors, $\mathbf{J}$ is a synaptic connection matrix of dimensionality $N \times N$
output : h_{thr} is a threshold separating true and spurious attractors
parameters: k_σ is standard deviation multiplier in calculation of maximal possible value of the Lyapunov function for spurious attractors

```
1 begin
2     for $k \leftarrow 1$ to 100 do
3         generate perm, a random permutation from 1 to $N$
4         for $i \leftarrow 1$ to $N$ do $x_i \leftarrow 0$
5         for $i \leftarrow 1$ to $n$ do $x_{\texttt{perm}(i)} \leftarrow 1$
6         $\mathbf{h} \leftarrow \mathbf{xJ}$
7         $h_k^{max} \leftarrow \max_i(h_i)$
8     $h_{thr} \leftarrow m + k_\sigma \sigma$ where $m$ and $\sigma$ are mean and standard deviation of $h_k^{max}$
```

Function InformationGain($\mathcal{X}, \mathbf{\Theta}, \mathcal{S}$)

input : $\mathcal{X}$ is a collection of signals with components x_{mj}, $m = 1, \dots, M$, $j = 1, \dots, N$, $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \dots, L, j = 1, \dots, N)$ are generative model parameters, $\mathcal{S}$ is collection of factor scores with components s_{ml}, $m = 1, \dots, M$, $l = 1, \dots, L$
output: G is a relative information gain for given $\mathcal{X}$, $\mathbf{\Theta}$ and $\mathcal{S}$

```
1  begin
2      define the Shannon function $\texttt{h}(x) = -x \log_2 x - (1-x)\log_2(1-x)$
3      for $j \leftarrow 1$ to $N$ do $p_j \leftarrow \sum_{m=1}^{M} x_{mj}/M$
4      $H_0 \leftarrow M \sum_{j=1}^{N} \texttt{h}(p_j)$
5      $H_1 \leftarrow M \sum_{i=1}^{L} \texttt{h}(\pi_i)$
6      for $m \leftarrow 1$ to $M$ do
7          for $j \leftarrow 1$ to $N$ do
8              $P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta}) \leftarrow x_{mj} - (2x_{mj} - 1)(1 - q_j) \prod_{i=1}^{L} (1 - p_{ij})^{s_{mi}}$
9      $H_2 \leftarrow \sum_{m=1}^{M} \sum_{j=1}^{N} \texttt{h}(P(x_{mj}|\mathbf{s}_m, \mathbf{\Theta}))$
10     $G \leftarrow \frac{H_0 - H_1 - H_2}{H_0}$
```

Procedure LikelihoodMaximization($\mathcal{X}, \boldsymbol{\Theta}$)

input : $\mathcal{X}$ is collection of signals with components x_{mj}, $m = 1, \dots, M$, $j = 1, \dots, N$, $\boldsymbol{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \dots, L, j = 1, \dots, N)$ are generative model parameters

output: $\boldsymbol{\Theta}$ are the parameters of generative model, $\mathcal{S}$ is a collection of factor scores with components s_{ml}, $m = 1, \dots, M$, $l = 1, \dots, L$ (number of factors L is less or equal to those at the input of the algorithm)

```
1  begin
2      $\mathcal{L} \leftarrow -\infty$
3      repeat
4          $\mathcal{L}^{prev} \leftarrow \mathcal{L}$
5          $\mathcal{S} \leftarrow$ProcEstep($\mathcal{X}, \boldsymbol{\Theta}$)
6          for $i \leftarrow 1$ to $L$ do
7              $\pi_i \leftarrow \sum_{m=1}^{M} s_i^m / M$
8              if $\pi_i = 0$ or $\pi_i = 1$ then
9                  discard $\pi_i$, $\{p_{ij}\}$, $\{s_{mi}\}$ from $\boldsymbol{\Theta}$ for all $j = 1, \dots, N$, $m = 1, \dots, M$
10                 $L \leftarrow L - 1$
11                 if $L = 0$ then break procedure
12         $\boldsymbol{\Theta} \leftarrow$ProcMstep($\mathcal{X}, \boldsymbol{\Theta}, \mathcal{S}$)
13         for $i \leftarrow 1$ to $L$ do
14             if $\sum_{j=1}^{N} \mathtt{sgn}(p_{ij}) = 0$ then
15                 discard $\pi_i$, $\{p_{ij}\}$, $\{s_{mi}\}$ from $\boldsymbol{\Theta}$ for all $j = 1, \dots, N$, $m = 1, \dots, M$
16                 $L \leftarrow L - 1$
17                 if $L = 0$ then break procedure
18         $\mathcal{L} \leftarrow \sum_{m=1}^{M} \Big[ \sum_{j=1}^{N} \log P(x_{mj} | \mathbf{s}_m, \boldsymbol{\Theta}) + \log P(\mathbf{s}_m | \boldsymbol{\Theta}) \Big]$
19     until $\mathcal{L} - \mathcal{L}^{prev} < 10^{-6} MN$
```

Procedure ProcEstep($\mathcal{X}, \mathbf{\Theta}$)

input : $\mathcal{X}$ is collection of row binary vectors $\mathbf{x}_m$ with components x_{mj}, $m = 1, \ldots, M$, $j = 1, \ldots, N$, $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \ldots, L, j = 1, \ldots, N)$ are generative model parameters

output: $\mathcal{S}$ is a collection of row vectors $\mathbf{s}_m$ of factor scores with components s_{ml}, $m = 1, \ldots, M$, $l = 1, \ldots, L$

```
1  begin
2      $\mathcal{S} \leftarrow \varnothing$
3      for $m \leftarrow 1$ to $M$ do
4          for $i \leftarrow 1$ to $L$ do  $s_i \leftarrow 0$
5          if $\mathbf{x}_m = \mathbf{0}$ then  goto line 15
6          repeat
7              $\mathbf{s}^{old} \leftarrow \mathbf{s}$
8              generate perm, a random permutation from 1 to $L$
9              for $l \leftarrow 1$ to $L$ do
10                 $i \leftarrow$ perm($l$)
11                 for $j \leftarrow 1$ to $N$ do  $P_j \leftarrow (1 - q_j) \prod_{k \neq i} (1 - p_{kj})^{s_k}$
12                 $\Delta I \leftarrow \sum_{j=1}^{N} \left[ x_{mj} \log \frac{1 - (1 - p_{ij}) P_j}{1 - P_j} + (1 - x_{mj}) \log(1 - p_{ij}) \right] + \log \frac{\pi_i}{1 - \pi_i}$
13                 if $\Delta I > 0$ then $s_i \leftarrow 1$ else $s_i \leftarrow 0$
14         until $\mathbf{s} = \mathbf{s}^{old}$
15         add $\mathbf{s}$ to $\mathcal{S}$
```

Procedure ProcMstep($\mathcal{X}, \mathbf{\Theta}, \mathcal{S}$)

input : $\mathcal{X}$ is a collection of row binary vectors $\mathbf{x}_m$ with components x_{mj}, $m = 1, \dots, M$, $j = 1, \dots, N$, $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \dots, L, j = 1, \dots, N)$ are generative model parameters, $\mathcal{S}$ is a collection of row vectors $\mathbf{s}_m$ of factor scores with components s_{ml}, $m = 1, \dots, M$, $l = 1, \dots, L$

output: $\mathbf{\Theta}$ are the optimized parameters of the generative model

```
1  begin
2      define $\xi$, a small positive number restricting $p_{ij}$ and $q_j$ for the sake of avoiding singularities
3      for $j \leftarrow 1$ to $N$ do
4          repeat
5              for $i \leftarrow 1$ to $L$ do $p_{ij}^{prev} \leftarrow p_{ij}$
6              for $m \leftarrow 1$ to $M$ do $P_m \leftarrow 1 - (1 - q_j) \prod_{i=1}^{L} (1 - p_{ij})^{s_{mi}}$
7              for $i \leftarrow 1$ to $L$ do
8                  $\gamma_p \leftarrow p_{ij}(1 - p_{ij})/(M\pi_i)$
9                  $p_{ij} \leftarrow p_{ij} + \gamma_p \frac{1}{1 - p_{ij}} \Big( \sum_{m=1}^{M} \frac{s_{mi} x_{mj}}{P_m} - M\pi_i \Big)$
10                 constraint $p_{ij}$ to belong to $[0; 1 - \xi]$
11                 if $p_{ij} < 1 - \prod_{l \neq i} (1 - \pi_l p_{lj})$ then $p_{ij} \leftarrow 0$
12             $\gamma_q \leftarrow q_j(1 - q_j)/M$
13             $q_j \leftarrow q_j + \gamma_q \frac{1}{1 - q_j} \Big( \sum_{m=1}^{M} \frac{x_{mj}}{P_m} - M \Big)$
14             constraint $q_j$ to belong to $[\xi; 1]$
15         until $\sum_{i=1}^{L} |p_{ij} - p_{ij}^{prev}| < 10^{-3} L$
```

Algorithm 2: Observation Generation

input : M is a number of observations, N is a dimensionality of the observations, L is a number of factors, $\mathbf{\Theta} = (p_{ij}, q_j, \pi_i, i = 1, \dots, L, j = 1, \dots, N)$ are parameters of generative model

output: $\mathcal{X}$ is a collection of M row binary vectors of dimension N, $\mathcal{S}$ is a collection of M row binary vectors of factor scores of dimension L

```
begin
    for $m \leftarrow 1$ to $M$ do
        Initialize x as row zero vector of dimension $N$
        Initialize s as row zero vector of dimension $L$
        for $i \leftarrow 1$ to $L$ do
            Generate $r$, a pseudo-random number from $[0, 1]$
            if $r < \pi_i$ then
                $s_i \leftarrow 1$
                for $j \leftarrow 1$ to $N$ do
                    Generate $r$, a pseudo-random number from $[0, 1]$
                    if $r < p_{ij}$ then $x_j \leftarrow x_j \vee 1$
        for $j \leftarrow 1$ to $N$ do
            Generate $r$, a pseudo-random number from $[0, 1]$
            if $r < q_j$ then $x_j \leftarrow x_j \vee 1$
        Add s to $\mathcal{S}$
        Add x to $\mathcal{X}$
```

Bibliography

[1] P. Yu. Polyakov A. A. Frolov, D. Gusek. Bulev factorial analysis by means of attractor neural network and its some appendices. *Nejrokomp'jutery: razrabotka, primenenie*, (1):25–46, 2011.

[2] P. Yu. Polyakov A. A. Frolov, D. Husek. Combined algorithm for boolean factor analysis based on neural network and likelihood maximization approaches. *Nejrokomp'jutery: razrabotka, primenenie*, (3):3–11, 2014.

[3] M. Alexandrov, A. Gelbukh, and P. Rosso. An approach to clustering abstracts. In *Applications of Natural Language to Data Base (NLDB'05), LNCS*, volume 3513, pages 275–285, Alicante, Spain, June 2005.

[4] S. Amari. Mathematical fondation of neurocomputing. *Proceedings of the IEEE*, 78(9):1443–1463, 1990.

[5] S. Amari and K. Maginu. Statistical neurodynamics of associative memory. *Neural Networks*, 1:63–73, 1988.

[6] D. J. Amit, H. Gutfreund, and H. Sompolinsky. Statistical mechanics of neural networks near saturation. *Annal of Physics*, 173:30–67, 1987.

[7] A. Asuncion and D. J. Newman. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, 2007.

[8] H. Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12(3):241–253, 2001.

[9] H. B. Barlow. Cerebral cortex as model builder. In D. Rose and V. G. Dodson, editors, *Models of the visual cortex*, pages 37–46. Wiley, Chichester, 1985.

[10] D. J. Bartholomew. *The Analysis and Interpretation of Multivariate Data for Social Scientists*. Chapman & Hall/CRC, 2002.

[11] R. Belohlavek and V. Vychodil. Geometry and heuristics for discovery of optimal factors in binary data. 2006. Available at http://www.researchgate.net.

[12] R. Belohlavek and V. Vychodil. On Boolean factor analysis with formal concepts as factors. In *Soft Computing and Intelligent Systems & Int. Symposium on Intelligent Systems (SCIS&ISIS 2006)*, pages 20–24, 2006.

[13] R. Belohlavek and V. Vychodil. Formal concepts as optimal factors in boolean factor analysis: implications and experiments? In *Fifth International Conference on Concept Lattices and Their Applications*, 2007.

[14] R. Belohlavek and V. Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *Journal of Computer and System Sciences*, 76(1):3–20, 2009.

[15] M. W. Berry and M. Browne. *Understanding search engines: mathematical modeling and text retrieval*. SIAM, Philadelphia, 1999.

[16] J. Bucingham and D. Willshaw. On setting unit thresholds in an incompletely connected associative net. *Network*, 4:441–459, 1993.

[17] J. de Leeuw. Principal component analysis of binary data: Applications to rollcall-analysis, 2003.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[19] J. Farkas. Documents, concepts and neural networks. *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing-Volume 2*, pages 1021–1031, 1993.

[20] P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Formal Aspects of Computing*, 64(2):165–170, 1990.

[21] P. Földiák and M. P. Young. Sparse coding in the primate cortex. *The Handbook of Brain Theory and Neural Networks*, pages 895–898, 1995.

[22] M. Frank, A. P. Streich, D. Basin, and Buchmann J. M. Multi-assignment clustering for Boolean data. *Journal of Machine Learning Research*, 13(3):459–489, 2012.

[23] A. Frolov, D. Husek, and P. Yu. Polyakov. Comparison of seven methods for Boolean factor analysis and their evaluation by information gain. *IEEE Transactions on Neural Networks*, 2015. Published online 07 April (2015).

[24] A. Frolov, P. Polyakov, and D. Husek. Boolean factor analysis by the expectation-maximization algorithm. In *International Conference on Computational Statistics (COMPSTAT'2010)*, pages 1039–1046, Paris, France, August 2010.

[25] A. A. Frolov, D. Husek, A. Abraham, P. Y. Polyakov, and H. Rezankova. Bfa and bmf: What is the difference. In *International conference on intelligent systems design and applications (ISDA 2012)*, pages 890–896, Kochi, India, November 2012.

[26] A. A. Frolov, D. Husek, and I. P. Muraviev. Informational capacity and recall quality in sparsely encoded hopfield-like neural network: Analytical approaches and computer simulation. *Neural Networks*, 10:845–855, 1997.

[27] A. A. Frolov, D. Husek, and I. P. Muraviev. Informational efficiency of sparsely encoded hopfield-like autoassociative memory. *Optical Memory Neural Networks*, 12(3):177–197, 2003.

[28] A. A. Frolov, D. Husek, I. P. Muraviev, and P. Y. Polyakov. Learning and unlearning in Hopfield-like neural network performing Boolean factor analysis. In *Advances in Machine Learning I: dedicated to the memory of professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*, pages 501–518. Springer, 2010.

[29] A. A. Frolov, D. Husek, I. P. Muraviev, and P. Yu. Polyakov. Origin and elimination of two global spurious attractors in Hopfield-like neural network performing Boolean factor analysis. *Neurocomputing*, 73(7-9):1394–1404, 2010.

[30] A. A. Frolov, D. Husek, P. Polyakov, and H. Rezankova. New neural network based approach helps to discover hidden Russian parliament voting patterns. In *IEEE International Joint Conference on Neural Networks (IJCNN 2006)*, pages 6518–6523, Vancouver, Canada, July 2006.

[31] A. A. Frolov, D. Husek, P. J. Polyakov, and H. Rezankova. Binary factorization of textual data by Hopfield-like neural network. In *Proc. Computational Statistics (Compstat'06)*, pages 1035–1041, Roma, Italy, August 2006.

[32] A. A. Frolov, D. Husek, P. J. Polyakov, H. Rezankova, and V. Snasel. Binary factorization of textual data by Hopfield-like neural network. In *Proc. Computational Statistics (Compstat'04)*, pages 1035–1041, Prague, Czech Republic, August 2004.

[33] A. A. Frolov, D. Husek, and P. Y. Polyakov. Recurrent neural network based Boolean factor analysis and its application to automatic terms and documents categorization. *IEEE Transactions on Neural Networks*, 20(7):1073–1086, 2009.

[34] A. A. Frolov, D. Husek, and P. Y. Polyakov. Atraktornaja nejronnaja set tipa hopfilda v kachestve metoda bulevskogo faktornogo analiza i nekotoryje ego socialnyje primenenija. In *Neurocomputing Paradigm and Society*, pages 62–102. Moscow University Press, 2012.

[35] A. A. Frolov, D. Husek, and P. Y. Polyakov. Two expectation-maximization algorithms for Boolean factor analysis. *Neurocomputing*, 130:83–97, 2014.

[36] A. A. Frolov, D. Husek, P. Y. Polyakov, and H. Rezankova. A comparative study of two methodologies for binary datasets analysis. *Neural Network World*, 22(6):565–582, 2012.

[37] A. A. Frolov, D. Husek, P. Y. Polyakov, and V. Snasel. New BFA method based on attractor neural network and likelihood maximization. *Neurocomputing*, 132:14–29, 2014.

[38] A. A. Frolov, D. Husek, and P. Yu. Polyakov. Expectation-maximization approach to Boolean factor analysis. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 559–566, San Jose, USA, July 2011.

[39] A. A. Frolov, D. Husek, and P. Yu. Polyakov. New measure of Boolean factor analysis quality. In *Adaptive and Natural Computing Algorhitms, part I.*, volume 6593 of *Lecture Notes in Computer Science*, pages 100–109, 2011. 10th International Conference on Adaptive and Natural Computing Algorithms, Ljubljana, SLOVENIA, APR 14-16, 2011.

[40] A. A. Frolov, D. Husek, and P. Yu. Polyakov. Attractor neural network combined with likelihood maximization algorithm for Boolean factor analysis. In *Advances in Neural Networks – ISNN 2012*, volume 7367 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2012. 9th International Conference on Advances in Neural Networks, Volume I, Shenyang, China, JUL 11-14, 2012.

[41] A. A. Frolov, D. Husek, and P. Yu Polyakov. Boolean factor analysis by expectation-maximization method. In *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, volume 179 of *Advances in Intelligent Systems and Computing*, pages 243–254. Springer Berlin Heidelberg, 2013. 3rd Intelligent Human Computer Interaction (IHCI 2011), Prague, CZECH REPUBLIC, AUG 29-31, 2011.

[42] A. A. Frolov, D. Husek, H. Rezankova, V. Snasel, and P. Polyakov. Clustering variables by classical approaches and neural network Boolean factor analysis. In *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*, pages 3742–3746, Hong Kong, China, June 2008.

[43] A. A. Frolov, A. M. Sirota, D. Husek, I. P. Muraviev, and P. J. Polyakov. Binary factorization in Hopfield-like neural networks: single-step approximation and computer simulations. *Neural Network Word*, 14:139–152, 2004.

[44] A.A. Frolov, D. Husek, I.P. Muraviev, and P.Yu. Polyakov. Boolean factor analysis by attractor neural network. *IEEE Transactions on Neural Networks*, 18(3):698–707, 2007.

[45] Alexander Frolov, Dusan Husek, and Pavel Polyakov. Estimation of Boolean factor analysis performance by informational gain. In V. Snasel, P. S. Szczepaniak, A. Abraham, and J. Kacprzyk, editors, *Advances in Intelligent Web Mastering-2*, volume 67, pages 83–94. Springer, 2010. 6th Atlantic Web Intelligence Conference, Prague, Czech Republic, Sep., 2009.

[46] X. Ge and S. Iwata. Learning the parts of objects by auto-association. *Neural Networks*, 15(2):285–295, 2002.

[47] R. L. Gorsuch. *Factor Analysis*. Lawrence Erlbaum Associates, 1983.

[48] V. J. Hodge and J. Austin. Hierarchcal word clustering - automatic thesaurus generation. *Neurocomputing*, 48:819–846, 2002.

[49] J. J. Hopfield. Neural network and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science USA*, 79:2544–2548, 1982.

[50] F. Hoppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy cluster analysis*. John Wiley & Sons, New York, 1999.

[51] D. Husek, A. Frolov, P. Polyakov, and H. Rezankova. Neural network fuzzy binary factorization. In *Abstracts of the 3rd IASC World Conference on Computational Statistics and Data Analysis (CSDA'2005)*, page 68, Limassol, Cyprus, October 2005.

[52] D. Husek, A. Frolov, P. Polyakov, and V. Snasel. Neural network Boolean factor analysis and applications. In *Proceedings of the 6th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics (CIMMACS'07)*, pages 30–35, Puerto de la Cruz, Spain, December 2007.

[53] D. Husek, A. A. Frolov, P. Polyakov, H. Rezankova, and V. Snasel. Application of neural network Boolean factor analysis procedure to automatic conference papers categorization. In *IEEE International Conference on Intelligence and Security Informatics 2007 (ISI 2007)*, pages 335–336, New Brunswick, New Jersey, USA, May 2007.

[54] D. Husek, A. A. Frolov, P. Y. Polyakov, and H. Rezankova. Neural network analysis of Russian parliament voting patterns. In *Computer Science and Information Technology (CsIT 2006)*, pages 328–334, Amman, August 2006.

[55] D. Husek, A. A. Frolov, P. Y. Polyakov, and H. Rezankova. Neural network based Boolean factor analysis: Efficient tool for automated topic search. In *Computer Science and Information Technology (CsIT 2006)*, pages 321–327, Amman, August 2006.

[56] D. Husek, A. A. Frolov, H. Rezankova, V. Snasel, M. Dufosse, and P. Polyakov. Neural network attempt to nonlinear binary factor analysis of textual data. In *Applied Stochastic Models and Data Analysis (ASMDA 2005)*, pages 1460–1467, Brest, France, May 2005.

[57] D. Husek, A. A. Frolov, H. Rezankova, V. Snasel, and P. Y. Polyakov. Some remarks on binary data grouping. In *DEXA 2008: 19th International Conference on Database and Expert Systems Application (ETID'08)*, pages 559–565, Turin, Italy, September 2008.

[58] D. Husek, A. Keprt, H. Rezankova, A. A. Frolov, P. Polyakov, and V. Snasel. Comparison of different approaches to overlapping clustering of binary variables. In *ITAT 2005, Information Technologies – Applications and Theory (Ed.: Vojtas P.)*, pages 55–64, Rackova dolina, Slovakia, September 2005.

[59] D. Husek, P. Moravec, V. Snasel, A. Frolov, H. Rezankova, and P. Polyakov. Comparison of neural network Boolean factor analysis method with some other dimension reduction methods on bars problem. In *Pattern Recognition and Machine Intelligence*, pages 235–243, Calcutta, India, December 2007.

[60] D. Husek, P. Moravec, V. Snasel, A. Frolov, H. Rezankova, and P. Polyakov. Testing of feature extracting methods on bar problem. In *Proceedings of the International conference on Statistics for Data Mining, Learning and Knowledge Extraction Models (IASC'07)*, pages 1–8, Aveiro, Portugal, August 2007.

[61] D. Husek, H. Rezankova, V. Snasel, A. Frolov, and P. Polyakov. Neural network nonlinear factor analysis of high dimensional binary signals. In *Signal & Image Technology and Internet Based Systems (SITIS 2005)*, pages 86–89, Cameroon, November 2005.

[62] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at GenomeNet. *Nucleic acids research*, 30(1):42, 2002.

[63] P. R. Kensche, V. Van Noort, B. E. Dutilh, and M. A. Huynen. Practical and theoretical advances in predicting the function of a protein by its phylogenetic distribution. *Journal of the Royal Society Interface*, 5(19):151, 2008.

[64] A. Keprt and V. Snášel. Binary Factor Analysis with Genetic Algorithms. In *Proceedings of 4th IEEE WSTST*, pages 1259–1268, 2005.

[65] Aleš Keprt. *Algorithms for Binary Factor Analysis*. PhD thesis, VŠB – Technical University of Ostrava, 2006.

[66] Aleš Keprt and Václav Snásel. Binary factor analysis with help of formal concepts. In *International Conference on Concept Lattices and Their Applications*, volume 110, pages 90–101, 2004.

[67] W. Kinzel. Learning and pattern recognition in spin glass models. *Zeitschrift für Physik B Condensed Matter*, 60(2):205–213, 1985.

[68] G. A. Kohring. A high-precision study of the hopfield model in the phase of broken replica symmetry. *Journal of Statistical Physics*, 59:1077–1086, 1990.

[69] P. Kudova, H. Rezankova, D. Husek, and V. Snasel. Categorical data clustering using statistical methods and neural networks. In *Spring Colloquium for Young Researchers in Databases and Information Sytems, SYRCoDIS'2006*, pages 19–23, 2006.

[70] C. Lucchese, S. Orlando, and R. Perego. A unifying framework for mining approximate top-k binary patterns. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2900–2913, 2014.

[71] J. Lücke and M. Sahani. Maximal causes for non-linear component extraction. *The Journal of Machine Learning Research*, 9:1227–1267, 2008.

[72] D. Marr. A Theory for Cerebral Neocortex. *Proceedings of the Royal Society of London. Series B, Biological Sciences (1934-1990)*, 176(1043):161–234, 1970.

[73] D. Marr. Simple Memory: A Theory for Archicortex. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences (1934-1990)*, 262(841):23–81, 1971.

[74] M R Mickey, P. Mundle, and L. Engelman. Boolean factor analysis. In W.J. Dixon, editor, *BMDP Statistical Software*, pages 538–545. University of California Press, Berkeley, CA, 1983.

[75] P. Miettinen, T. Mielikainen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering*, 20(10):1348–1362, 2008.

[76] P. Moravec, V. Snasel, A. Frolov, D. Husek, H. Rezankova, and P. Polyakov. Image analysis by methods of dimension reduction. In *International Conference on Computer Information Systems and Industrial Management Applications (CISIM 2007)*, pages 272–277, Elk, Poland, June 2007.

[77] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 89:355–368, 1998.

[78] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 96(8):4285–4288, 1999.

[79] P. Polyakov, A. A. Frolov, and D. Husek. Comparison of two neural networks approaches to Boolean matrix factorization. In *Networked Digital Technologies (NDT'09)*, pages 303–308, Ostrava, Czech Republic, July 2009.

[80] P. Polyakov, A. A. Frolov, and D. Husek. Expectation-maximization method for boolean factor analysis. In *Workshop of Faculty of Electrical Engineering and Computer Science (WOFEX 2011)*, pages 436–441, Ostrava, Czech Republic, September 2011.

[81] P. Y. Polyakov, A. A. Frolov, and D. Husek. Binary factor analysis by Hopfield network and its application to automatic text classification. In *Proc. Neuroinformatics'2006*, volume 3, pages 172–180, Moscow, Russia, January 2006.

[82] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, and A.L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551, 2002.

[83] H. Rezankova, D. Husek, P. Kudova, and V. Snasel. Comparison of some approaches to clustering categorical data. In *Proc. Computational Statistics (Compstat'06)*, pages 607–613, Roma, Italy, August 2006.

[84] V. Snasel, D. Husek, A. Frolov, H. Rezankova, P. Moravec, and P. Polyakov. Bars problem solving - new neural network method and comparison. In *Advances in Artificial Intelligence (MICAI 2007)*, pages 671–682, Aguascalientes, Mexico, November 2007.

[85] V. Snasel, P. Moravec, D. Husek, A. Frolov, H. Rezankova, and P. Polyakov. Pattern discovery for high-dimensional binary datasets. In *International Conference on Neural Information Processing (ICONIP 2007)*, pages 861–872, Kitakyushu, Japan, November 2007.

[86] V. Snasel, J. Platos, P. Kromer, D. Husek, and A. Frolov. On the road to genetic Boolean matrix factorization. *Neural Network World*, 17(6):675–688, 2007.

[87] M. W. Spratling. Learning Image Components for Object Recognition. *The Journal of Machine Learning Research*, 7:793–815, 2006.

[88] M. W. Spratling and M. H. Johnson. Preintegration Lateral Inhibition Enhances UnsupervisedLearning. *Neural Computation*, 14(9):2157–2179, 2002.

[89] M. W. Spratling and M. H. Johnson. Exploring the functional significance of dendritic inhibition in cortical pyramidal cells. *Neurocomputing*, 52(54):389–395, 2003.

[90] L. L. Thurstone. Multiple factor analysis. *Psychological Review*, 38:406–427, 1931.

[91] C. J. van Rijsbergen. A theoretical basis for the use of cooccurence data in informatio retrieval. *Journal of Documentation*, 33(2):106–119, 1977.

[92] C. Von Mering, R. Krause, B. Snel, M. Cornell, S.G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002.

Author's publications related to the topic of the thesis

[1] P. Yu. Polyakov A. A. Frolov, D. Gusek. Bulev factorial analysis by means of attractor neural network and its some appendices. *Nejrokomp'jutery: razrabotka, primenenie*, (1):25–46, 2011.

[2] P. Yu. Polyakov A. A. Frolov, D. Husek. Combined algorithm for boolean factor analysis based on neural network and likelihood maximization approaches. *Nejrokomp'jutery: razrabotka, primenenie*, (3):3–11, 2014.

[3] A. Frolov, D. Husek, and P. Yu. Polyakov. Comparison of seven methods for Boolean factor analysis and their evaluation by information gain. *IEEE Transactions on Neural Networks*, 2015. Published online 07 April (2015).

[4] A. Frolov, P. Polyakov, and D. Husek. Boolean factor analysis by the expectation-maximization algorithm. In *International Conference on Computational Statistics (COMPSTAT'2010)*, pages 1039–1046, Paris, France, August 2010.

[5] A. A. Frolov, D. Husek, A. Abraham, P. Y. Polyakov, and H. Rezankova. Bfa and bmf: What is the difference. In *International conference on intelligent systems design and applications (ISDA 2012)*, pages 890–896, Kochi, India, November 2012.

[6] A. A. Frolov, D. Husek, I. P. Muraviev, and P. Y. Polyakov. Learning and unlearning in Hopfield-like neural network performing Boolean factor analysis. In *Advances in Machine Learning I: dedicated to the memory of professor Ryszard S. Michalski*, volume 262 of *Studies in Computational Intelligence*, pages 501–518. Springer, 2010.

[7] A. A. Frolov, D. Husek, I. P. Muraviev, and P. Yu. Polyakov. Origin and elimination of two global spurious attractors in Hopfield-like neural network performing Boolean factor analysis. *Neurocomputing*, 73(7-9):1394–1404, 2010.

[8] A. A. Frolov, D. Husek, P. Polyakov, and H. Rezankova. New neural network based approach helps to discover hidden Russian parliament voting patterns. In *IEEE International*

Joint Conference on Neural Networks (IJCNN 2006), pages 6518–6523, Vancouver, Canada, July 2006.

[9] A. A. Frolov, D. Husek, P. J. Polyakov, and H. Rezankova. Binary factorization of textual data by Hopfield-like neural network. In *Proc. Computational Statistics (Compstat'06)*, pages 1035–1041, Roma, Italy, August 2006.

[10] A. A. Frolov, D. Husek, P. J. Polyakov, H. Rezankova, and V. Snasel. Binary factorization of textual data by Hopfield-like neural network. In *Proc. Computational Statistics (Compstat'04)*, pages 1035–1041, Prague, Czech Republic, August 2004.

[11] A. A. Frolov, D. Husek, and P. Y. Polyakov. Recurrent neural network based Boolean factor analysis and its application to automatic terms and documents categorization. *IEEE Transactions on Neural Networks*, 20(7):1073–1086, 2009.

[12] A. A. Frolov, D. Husek, and P. Y. Polyakov. Atraktornaja nejronnaja set tipa hopfilda v kachestve metoda bulevskogo faktornogo analiza i nekotoryje ego socialnyje primenenija. In *Neurocomputing Paradigm and Society*, pages 62–102. Moscow University Press, 2012.

[13] A. A. Frolov, D. Husek, and P. Y. Polyakov. Two expectation-maximization algorithms for Boolean factor analysis. *Neurocomputing*, 130:83–97, 2014.

[14] A. A. Frolov, D. Husek, P. Y. Polyakov, and H. Rezankova. A comparative study of two methodologies for binary datasets analysis. *Neural Network World*, 22(6):565–582, 2012.

[15] A. A. Frolov, D. Husek, P. Y. Polyakov, and V. Snasel. New BFA method based on attractor neural network and likelihood maximization. *Neurocomputing*, 132:14–29, 2014.

[16] A. A. Frolov, D. Husek, and P. Yu. Polyakov. Expectation-maximization approach to Boolean factor analysis. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 559–566, San Jose, USA, July 2011.

[17] A. A. Frolov, D. Husek, and P. Yu. Polyakov. New measure of Boolean factor analysis quality. In *Adaptive and Natural Computing Algorhitms, part I.*, volume 6593 of *Lecture Notes in Computer Science*, pages 100–109, 2011. 10th International Conference on Adaptive and Natural Computing Algorithms, Ljubljana, SLOVENIA, APR 14-16, 2011.

[18] A. A. Frolov, D. Husek, and P. Yu. Polyakov. Attractor neural network combined with likelihood maximization algorithm for Boolean factor analysis. In *Advances in Neural Networks – ISNN 2012*, volume 7367 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2012. 9th International Conference on Advances in Neural Networks, Volume I, Shenyang, China, JUL 11-14, 2012.

[19] A. A. Frolov, D. Husek, and P. Yu Polyakov. Boolean factor analysis by expectation-maximization method. In *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, volume 179 of *Advances in Intelligent Systems and Computing*, pages 243–254. Springer Berlin Heidelberg, 2013. 3rd Intelligent Human Computer Interaction (IHCI 2011), Prague, CZECH REPUBLIC, AUG 29-31, 2011.

[20] A. A. Frolov, D. Husek, H. Rezankova, V. Snasel, and P. Polyakov. Clustering variables by classical approaches and neural network Boolean factor analysis. In *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*, pages 3742–3746, Hong Kong, China, June 2008.

[21] A. A. Frolov, A. M. Sirota, D. Husek, I. P. Muraviev, and P. J. Polyakov. Binary factorization in Hopfield-like neural networks: single-step approximation and computer simulations. *Neural Network Word*, 14:139–152, 2004.

[22] A.A. Frolov, D. Husek, I.P. Muraviev, and P.Yu. Polyakov. Boolean factor analysis by attractor neural network. *IEEE Transactions on Neural Networks*, 18(3):698–707, 2007.

[23] Alexander Frolov, Dusan Husek, and Pavel Polyakov. Estimation of Boolean factor analysis performance by informational gain. In V. Snasel, P. S. Szczepaniak, A. Abraham, and J. Kacprzyk, editors, *Advances in Intelligent Web Mastering-2*, volume 67, pages 83–94. Springer, 2010. 6th Atlantic Web Intelligence Conference, Prague, Czech Republic, Sep., 2009.

[24] D. Husek, A. Frolov, P. Polyakov, and H. Rezankova. Neural network fuzzy binary factorization. In *Abstracts of the 3rd IASC World Conference on Computational Statistics and Data Analysis (CSDA'2005)*, page 68, Limassol, Cyprus, October 2005.

[25] D. Husek, A. Frolov, P. Polyakov, and V. Snasel. Neural network Boolean factor analysis and applications. In *Proceedings of the 6th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics (CIMMACS'07)*, pages 30–35, Puerto de la Cruz, Spain, December 2007.

[26] D. Husek, A. A. Frolov, P. Polyakov, H. Rezankova, and V. Snasel. Application of neural network Boolean factor analysis procedure to automatic conference papers categorization. In *IEEE International Conference on Intelligence and Security Informatics 2007 (ISI 2007)*, pages 335–336, New Brunswick, New Jersey, USA, May 2007.

[27] D. Husek, A. A. Frolov, P. Y. Polyakov, and H. Rezankova. Neural network analysis of Russian parliament voting patterns. In *Computer Science and Information Technology (CsIT 2006)*, pages 328–334, Amman, August 2006.

[28] D. Husek, A. A. Frolov, P. Y. Polyakov, and H. Rezankova. Neural network based Boolean factor analysis: Efficient tool for automated topic search. In *Computer Science and Information Technology (CsIT 2006)*, pages 321–327, Amman, August 2006.

[29] D. Husek, A. A. Frolov, H. Rezankova, V. Snasel, M. Dufosse, and P. Polyakov. Neural network attempt to nonlinear binary factor analysis of textual data. In *Applied Stochastic Models and Data Analysis (ASMDA 2005)*, pages 1460–1467, Brest, France, May 2005.

[30] D. Husek, A. A. Frolov, H. Rezankova, V. Snasel, and P. Y. Polyakov. Some remarks on binary data grouping. In *DEXA 2008: 19th International Conference on Database and Expert Systems Application (ETID'08)*, pages 559–565, Turin, Italy, September 2008.

[31] D. Husek, A. Keprt, H. Rezankova, A. A. Frolov, P. Polyakov, and V. Snasel. Comparison of different approaches to overlapping clustering of binary variables. In *ITAT 2005, Information Technologies – Applications and Theory (Ed.: Vojtas P.)*, pages 55–64, Rackova dolina, Slovakia, September 2005.

[32] D. Husek, P. Moravec, V. Snasel, A. Frolov, H. Rezankova, and P. Polyakov. Comparison of neural network Boolean factor analysis method with some other dimension reduction methods on bars problem. In *Pattern Recognition and Machine Intelligence*, pages 235–243, Calcutta, India, December 2007.

[33] D. Husek, P. Moravec, V. Snasel, A. Frolov, H. Rezankova, and P. Polyakov. Testing of feature extracting methods on bar problem. In *Proceedings of the International conference on Statistics for Data Mining, Learning and Knowledge Extraction Models (IASC'07)*, pages 1–8, Aveiro, Portugal, August 2007.

[34] D. Husek, H. Rezankova, V. Snasel, A. Frolov, and P. Polyakov. Neural network nonlinear factor analysis of high dimensional binary signals. In *Signal & Image Technology and Internet Based Systems (SITIS 2005)*, pages 86–89, Cameroon, November 2005.

[35] P. Moravec, V. Snasel, A. Frolov, D. Husek, H. Rezankova, and P. Polyakov. Image analysis by methods of dimension reduction. In *International Conference on Computer Information Systems and Industrial Management Applications (CISIM 2007)*, pages 272–277, Elk, Poland, June 2007.

[36] P. Polyakov, A. A. Frolov, and D. Husek. Comparison of two neural networks approaches to Boolean matrix factorization. In *Networked Digital Technologies (NDT'09)*, pages 303–308, Ostrava, Czech Republic, July 2009.

[37] P. Polyakov, A. A. Frolov, and D. Husek. Expectation-maximization method for boolean factor analysis. In *Workshop of Faculty of Electrical Engineering and Computer Science (WOFEX 2011)*, pages 436–441, Ostrava, Czech Republic, September 2011.

[38] P. Y. Polyakov, A. A. Frolov, and D. Husek. Binary factor analysis by Hopfield network and its application to automatic text classification. In *Proc. Neuroinformatics'2006*, volume 3, pages 172–180, Moscow, Russia, January 2006.

[39] V. Snasel, D. Husek, A. Frolov, H. Rezankova, P. Moravec, and P. Polyakov. Bars problem solving - new neural network method and comparison. In *Advances in Artificial Intelligence (MICAI 2007)*, pages 671–682, Aguascalientes, Mexico, November 2007.

[40] V. Snasel, P. Moravec, D. Husek, A. Frolov, H. Rezankova, and P. Polyakov. Pattern discovery for high-dimensional binary datasets. In *International Conference on Neural Information Processing (ICONIP 2007)*, pages 861–872, Kitakyushu, Japan, November 2007.

Author's publications unrelated to the topic of the thesis

[1] V. V. Pleshko, A. E. Ermakov, V. P. Golenkov, and P. Yu. Polyakov. Rco at rires 2005. In *Proc. Third Russian information retrieval evaluation seminar (ROMIP'2005)*, pages 23–39, St. Petersburg, Russia, September 2005.

[2] V. V. Pleshko and P. Yu. Polyakov. Rco at rires 2008. In *Proc. Sixth Russian information retrieval evaluation seminar (ROMIP'2008)*, pages 96–107, Dubna, Russia, October 2008.

[3] P. Yu. Polyakov, M. V. Kalinina, and V. V. Pleshko. Research on applicability of thematic classification methods to the problem of book review classification. In *Annual International Conference "Dialogue" (2012)*, volume 2, pages 51–59, Naro-Fominsk, Russia, June 2012.

[4] P. Yu. Polyakov, M. V. Kalinina, and V. V. Pleshko. Automatic object-oriented sentiment analysis by means of semantic templates and sentiment lexicon dictionaries. In *Annual International Conference "Dialogue" (2015)*, volume 2, pages 44–52, Moscow, Russia, May 2015.

[5] P. Yu. Polyakov and V. V. Pleshko. Rco at rires 2006. In *Proc. Fourth Russian information retrieval evaluation seminar (ROMIP'2006)*, pages 72–79, Suzdal, Russia, October 2006.

[6] P. Yu. Polyakov, V. V. Pleshko, and A. E. Ermakov. Rco at rires 2009. In *Proc. Seventh Russian information retrieval evaluation seminar (ROMIP'2009)*, pages 122–134, Petrozavodsk, Russia, September 2009.

	Sum	Indexed by WOS	Indexed by SCOPUS
Journal (reviewed)	10	6	7
Book			
Chapter in a book	2	1	1
Paper in proceedings	34	14	5
Patent			
Research report			
Software			

	by WOS	by Google
Citations (including selfcitations)	75	188
H-index	4	7

TABLE D.1: Publications summary