Double-Layer Vector Perceptron for Binary Patterns Recognition

Vladimir Kryzhanovskiy and Irina Zhelavskaya

Abstract. A new model – Double-Layer Vector Perceptron (DLVP) – is proposed. Compared with a single-layer perceptron, its operation requires slightly more computations (by 5%) and more effective computer memory, but it excels at a much lower error rate (four orders of magnitude lower). The estimate of DLVP storage capacity is obtained.

Keywords: vector neural networks, Potts model.

1 Introduction

The Potts model [1-2] is the first and the most well known vector neural network. The model still draws much attention from researchers in such fields as physics, medicine, image segmentation and neural networks. Later, the parametric neural network [3] was offered and thoroughly studied by a small group of the Institute of Optical Neural Technologies of Russian Academy of Sciences (the Center of Optical Neural Technologies of the System Research Institute of RAS today). A similar model (CMM) was developed independently and is still investigated at York University [4]. V. Kryzhanovsky's thesis introduces a vector neural network model with a proximity measure between neuron states. This kind of neural networks generalizes all above-mentioned models. Researchers studied both fully connected and perceptron-like architectures. Various vector-net learning rules were studied [6]. The results proved the high efficiency of vector networks.

Perceptron is most suitable for associative memory-based applications (in our case it is a vector perceptron). However, it has a major drawback: even one output neuron taking a wrong state results in an input vector not being recognized.

Vladimir Kryzhanovskiy · Irina Zhelavskaya
Scientific-Research Institute for System Analysis of Russian Academy of Sciences (SRISA RAS), 36/1 Nahimovskiy Ave., Moscow, Russia, 117218
e-mail: vladimir.krizhanovsky@gmail.com

P. Koprinkova-Hristova et al. (eds.), *Artificial Neural Networks*,
Springer Series in Bio-/Neuroinformatics 4, DOI: 10.1007/978-3-319-09903-3_5

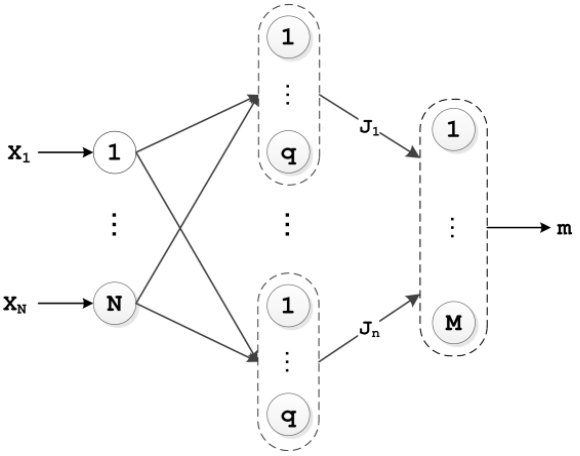

Fig. 1 The general arrangement of the double-layer vector perceptron

To overcome this, one has to raise the reliability of each neuron by increasing the net redundancy or decreasing the load of the net. In other words, the vector perceptron consists of "reliable" neurons that cannot make mistakes, which contradicts the whole philosophy of neural networks.

The alternative approach is to use weak neurons. With similar requirements for RAM, a collection of weak neurons proves to be more effective than a small number of reliable neurons. The trick is to supply a vector perceptron with an additional layer consisting of only one neuron that has a number of states equal to the number of stored patterns. Its aim is to accumulate the information from the preceding layer and to identify an input pattern. The approach is close to the idea offered in papers [7, 8].

The paper consists of three parts: formal description of the model, qualitative description with a simple example that helps to understand the point of the approach, and experimental results.

2 Setting Up the Problem

In this paper we are solving the nearest neighbor search problem, which consists in the following. Let us have a set of M N-dimensional bipolar patterns:

$$\mathbf{X}_\mu \in R^N,\ x_{\mu i} \in \{\pm 1\},\ \mu \in \overline{1,M}\ . \tag{2.1}$$

A bipolar vector $\mathbf{X}$ is applied to the inputs of the network. The goal is to find reference pattern $\mathbf{X}_m$ with the smallest Hamming distance to input pattern $\mathbf{X}$.

3 Formal Description of the Model

3.1 Model Description

Let us consider double-layer architecture (Fig. 1). The input layer has N scalar neurons, each of which takes one of two states $x_i = \pm 1$, $i = 1, 2, \ldots, N$. The first (inner) layer consists of n vector neurons. Each of these neurons has $2q$ fictive states during the training, and is described by basis vectors of q-dimensional space $\mathbf{y}_i \in \{\pm\mathbf{e}_1, \pm\mathbf{e}_2, ..., \pm\mathbf{e}_q\}$, where $\mathbf{e}_k = (0,...,0,1,0,...,0)$ is the unit vector with k-th component equal to 1. These fictive states are applicable only during the training, and can be considered as responses of the inner layer of the network. That is done since we use Hebb rule for training, so the responses of the network should be known in advance. At the recognition stage, these neurons are simple summators (thus, there is no activation function in the inner layer). This is done to simplify the description of the model. The second (output) layer has one vector neuron that can take M states, and is described by basis vectors of M-dimensional space (where M is the number of patterns in the training set) $\mathbf{O} \in \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_M\}$.

The state of the perceptron is described by three vectors:

1) Input layer is described by N-dimensional bipolar vector $\mathbf{X} = (x_1, x_2, ..., x_N)$, where $x_i = \pm 1$;
2) The first (inner) layer is described by n-dimensional $2q$-nary vector $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n)$, where $\mathbf{y}_i \in \{\pm\mathbf{e}_1, \pm\mathbf{e}_2, ..., \pm\mathbf{e}_q\}$, and $\mathbf{e}_k = (0,...,0,1,0,...,0)$ is the q-dimensional unit vector with k-th component equal to 1;
3) The second (output) layer is described by M-nary vector $\mathbf{O} \in \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_M\}$, where $\mathbf{o}_r = (0,...,0,1,0,...,0)$ is the M-dimensional unit vector holding unit in the r-th digit.

Each reference pattern $\mathbf{X}_m$ is uniquely associated with vector $\mathbf{Y}_m$. In its turn, each vector $\mathbf{Y}_m$ is uniquely associated with vector $\mathbf{o}_m$. Each component of vector $\mathbf{Y}_m$ is generated in a way that on the one hand, $\mathbf{Y}_m$ is a unique vector, and on the other hand, possible states $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_q\}$ are distributed evenly among reference vectors, i.e. $\sum_\mu \mathbf{y}_{\mu i} \equiv \frac{M}{q}(1,1,...,1)$. If the last condition is not satisfied, the error rate grows by several orders of magnitude, which was proved experimentally. So, we build a neural network that stores association:

$$\mathbf{X}_m \Leftrightarrow \mathbf{Y}_m \Leftrightarrow \mathbf{o}_m \tag{3.1}$$

3.2 Learning Procedure

The synaptic connections of the vector perceptron are computed using generalized Hebb's rule:

$$\mathbf{W}_{ji} = \sum_{m=1}^{M} \mathbf{y}_j^m x_i^m \quad \text{and} \quad \mathbf{J}_j = \sum_{m=1}^{M} \mathbf{o}_m^T \mathbf{y}_j^m , \tag{3.2}$$

where $\mathbf{W}_{ji}$ is the q-dimensional vector describing the connection between the i-th neuron of the input layer and the j-th neuron of the inner layer; $\mathbf{J}_j$ is the $M \times q$ matrix responsible for connection between j-th neuron of the inner layer and the sole output neuron, $i = \overline{1, N}$, $j = \overline{1, n}$.

3.3 Identification Process

Let us apply vector $\mathbf{X}$ to the network inputs. Let us compute the response of the net $\mathbf{O}$. For that purpose, let us first calculate local fields of the inner layer:

$$\mathbf{h}_j = \sum_{i=1}^{N} \mathbf{W}_{ji} x_i \quad . \tag{3.3}$$

Since the inner-layer neurons act as simple summators during recognition, signal $\mathbf{h}_j$ arrives to the output neuron without any changes. That is why local field of the output layer has the form:

$$\mathbf{H} = \sum_{j=1}^{n} \mathbf{J}_j \mathbf{h}_j^T . \tag{3.4}$$

The final output $\mathbf{O}$ is calculated in the following way. We identify the largest component of local field $\mathbf{H}$. Let it be component r. Then, the output of the perceptron is $\mathbf{O} = \mathbf{o}_r$. In other words, the input of the perceptron receives a distorted variant of the r-th reference pattern. And the larger the product $(\mathbf{H}, \mathbf{o}_r)$ is, the more statistically reliable the response of the network is. Moreover, if we arrange the numbers of components in increasing order the resulting list will tell us how close to corresponding vectors input vector $\mathbf{X}$ is in terms of Hamming vicinity.

4 Qualitative Description of the Model

4.1 The General Idea

Each vector neuron corresponds to a unique partition of the whole set of reference patterns into q subsets. For instance, Fig. 2 shows us two partitions of the set of M=12 patterns into q=4 subsets. For any partition we can calculate q "probabilities" (components of the vector of local fields, h_j^k) of the input pattern belonging

to one of the q subsets. Each vector neuron is basically a solver that selects a subset with the highest "probability" (in Fig. 3 it is subset No.1 in the first partition and subset No.1 in the second partition). The intersection of the subsets that were selected by all solvers determines the output of a single-layer perceptron. Calculations of the "probabilities" may contain errors due to the statistical nature of calculations. So, a solution found by selecting the "highest-probability" subsets might be wrong. Mistake in selecting a "winning" subset in at least one partition is enough to get a wrong solution (Fig. 3).

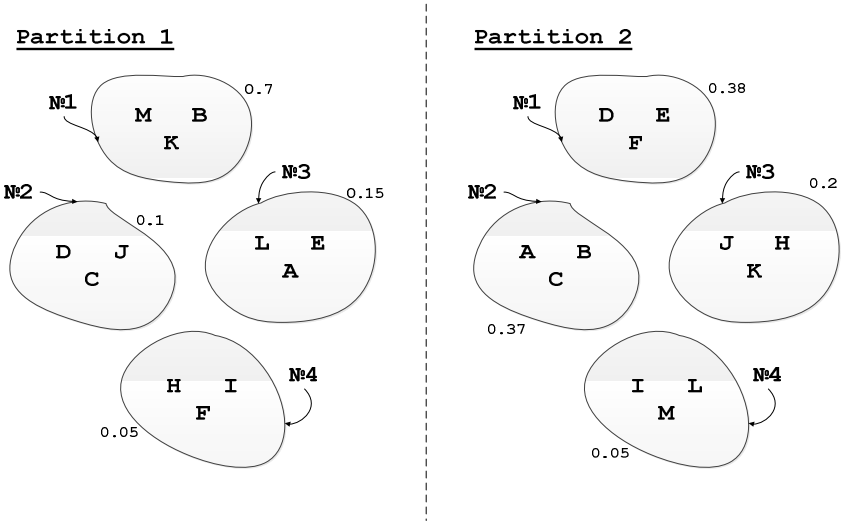

Fig. 2 Partition of a set of objects in two different ways

Fig. 3 Intersection of winning subsets from partition 1 and 2 results in a null subset

The goal of the proposed method is to overcome this drawback. The idea is to make decisions by accumulating "probabilities" over all partitions rather than using "probabilities" of partitions separately (and cutting off possible solutions by doing so). To do that, we need to interpret the probabilities $h_j^1, h_j^2, ..., h_j^q$ for j-th partition differently from what we did before. If previously we treated h_j^k as an indicator of k-th subset in j-th partition, now we will say that each element (pattern) of k-th subset in j-th partition is associated with the same indicator h_j^k. Thus, each pattern has n corresponding probabilities (where n is a number of different partitions of the total set), and their sum represents a cumulative indicator of this

pattern. Using these cumulative indicators allows us to decide which of the patterns is the winner based on the information from all subsets of all partitions.

(*It should be noted once again that the "probability" here is understood as a certain statistical quantity – a component* h_j^k *of local field, to be exact. The higher the probability of an input pattern being a pattern from a subset corresponding to this local field, the larger this component is.*)

4.2 Example

Let us exemplify the idea. Fig. 2 shows two different partitions (n = 2) of a set of 12 letter-denoted patterns into 4 subsets. Let us apply distorted pattern B to the inputs. In the figure each subset has a corresponding number, which is the calculated "probability" that an input pattern is a pattern from this particular subset.

Table 1 Probability that the input pattern belongs to a particular subset

Partition 1		
Subset number	Objects	Probability*
1	**M, K, B**	**0.70**
2	**D, J, C**	0.10
3	**L, E, A**	0.15
4	**H, I, F**	0.05

Partition 2		
Subset number	Objects	Probability*
1	**D, E, F**	**0.38**
2	**A, B, C**	0.37
3	**J, H, K**	0.20
4	**I, L, M**	0.05

*Probability - chances that the input pattern belongs to the subset.

When a single-layer perceptron is used for recognition, subset No.1 is the "winner" subset in the first partition, and it really contains the input pattern. In the second partition the "winner" is also subset No.1, yet it does not have the input pattern in it. The intersection of the two subsets gives us a null subset (Fig. 3), which means that the net cannot identify the input pattern. It is clear that the failure of one neuron causes the failure of the whole system. At the same time, we can see that the probabilities of the input pattern belonging to subset 1 or subset 2 for the second partition are almost equal – the difference is just 0.01 (1%) (Table 1). That is to say, it is almost equiprobable for the input pattern to be either in the first or the second subset. Our model takes this fact into account, and for each pattern the decision is made by using probabilities from both partitions (Table 2). The pattern that corresponds to the greatest total "probability" is selected as the response of the system. The result is a correct identification of the input pattern by the network.

Table 2 Recognition probabilities computed for two partitions and their sum for each pattern

Pattern	Probability for partition 1	Probability for partition 2	Summary probability
A	0.15	0.37	0.52
B	0.70	0.37	**1.07**
C	0.10	0.37	0.47
D	0.10	0.38	0.48
E	0.15	0.38	0.53
F	0.05	0.38	0.43

5 Details of the Algorithm

We can see from the table that the proposed model requires just 4-5% more computational resources (CPU, RAM) than the single-layer perceptron.

Table 3 Details of the algorithm

	Single layer	Two layers	Ratio*
Computational burden (number of operations)	$2Nnq$	$2Nnq+(n+1)M$	1.025
Necessary amount of RAM, bytes	$4Nnq$	$4Nnq+4nM$	1.033

* - the ratio is taken for $M = 100$; $N = 100$; $q = 300$; $n = 2$.

6 Storage Capacity

A new model of neural networks that is basically a product of adding one more layer to a single-layer perceptron was presented above. The value of this additional layer was illustrated via example in section 4.2. Now, we need to examine the properties of the model and to compare characteristics of a single- and double-layer perceptrons. This can be done in several ways:

1) We can take a range of datasets containing real data from different domains, and investigate how the proposed model and the single-layer perceptron do perform on them. As a result, we will identify types of data (types of problems), for which the networks described above work well and for which they do not. These results would be very important since through them we would be able to understand what place our model take among existing ones. The disadvantage of this approach is that a very deep analysis of the used data is required in order to understand the reasons why models work well or not, and this task is a very nontrivial task itself.

2) Another approach is to generate a number of synthetic datasets that will be considered as reference vectors (reference patterns), and test the models on them. In this case, it becomes possible to create the situations when the targeted models properties are most pronounced. The significant advantage of that is the possibility to calculate statistical characteristics such as expected mean, variance, correlations, etc., and different events probabilities analytically. Such estimates allow us to understand the endogenous processes in neural networks better.

It is obvious that for thorough investigation of the model it is necessary to go both ways. In the present work authors follow the second one: as reference patterns we use vectors, which components are generated independently with equal probabilities and take either +1 or -1.

So, with what purpose do we consider such kind of vectors? Our choice is based on several reasons. First, this case is the simplest one for analytical calculations. Second, the estimate of storage capacity would be an upper bound estimate in this case, i.e. we are estimating the maximum possible storage capacity of a neural network. So, for instance, it is well known that neural networks work worse when recognizing similar patterns, i.e. patterns that have correlations between them rather than patterns without correlations. That means that they are able to "remember" a fewer number of patterns a priori. Moreover, the probability of correct recognition highly depends on correlation values in each particular case. For that reason, we can make comparison between different models of associative memory only by using the upper bound estimate of the storage capacity for the simplest case. For example, the well-known result $0.14N$, which is the storage capacity estimate of Hopfield associative memory model, was obtained under the same assumptions.

Let us give a definition of the storage capacity. The storage capacity of associative memory is a number of reference patterns $M_{\max}$ that can be remembered by a neural network so that it can recognize all of them without error. By that it is understood that adding just one reference pattern to the training set will lead to the fact that one of the patterns is not being recognized correctly. In that event, the probability of error recognition equals to $1/(M_{\max}+1)$.

We may formulate this classical definition in a different way. The storage capacity of associative memory $M_{\max}$ is such number of reference patterns, recognizing which the probability of recognition error P is equal to $1/M_{\max}$[1]. At that, it is a common practice that neural networks are tested at reference patterns without any distortions. Authors think that it is necessary to generalize this definition, and to define $M_{\max}$ at condition that reference vectors being applied to the inputs of the network are distorted at some noise level a, and the probability of recognition error P is not greater than a predefined threshold $P_{\max}$ (value $P_{\max}$ could be any, including $1/M_{\max}$).

Authors managed to estimate storage capacity $M_{\max}$ for both models at the abovementioned conditions. Resulting estimates are in a good agreement with the

experiment differing just 1-1.3 times in magnitude from experimental results. Detailed derivations of the following estimates are presented in Appendix 1:

1) Storage capacity of double-layer vector perceptron (DLVP):

$$M < C\frac{nqN(1-2a)^2}{8\ln\left(\dfrac{nqN}{4\sqrt{2\pi}P_{\max}}\right)},\ C = 2.5\,. \tag{6.1}$$

2) Storage capacity of single-layer perceptron:

$$M = \frac{qN(1-2a)^2}{2\ln\left(\dfrac{nqN}{\sqrt{2\pi}P_{\max}}\right)} \tag{6.2}$$

Let us analyze the storage capacity of both models. We may draw the following conclusions from (6.1) and (6.2):

1) Storage capacity of both models increases linearly with N, q;
2) Storage capacity of both models decreases quadratically with a rise of the distortion level of reference patterns a;
3) Strengthening the requirements for recognition reliability, i.e. reduction of accepted probability error $P_{\max}$, leads to log decrease of storage capacity for both models;
4) And most importantly, comparing these two estimates we can see that the storage capacity of a double-layer perceptron is n times greater than the storage capacity of a single-layer perceptron!

7 Experimental Results

In this section we explore the properties of the proposed model in the following experiments:

1. First, we show that adding the second layer to the network enhances the probability of the correct recognition of input vectors. For this purpose, we experimentally compare double- and single-layer perceptrons. In these experiments we will vary external parameters N, M, a.
2. Then, we investigate the model behavior depending on internal parameters n and q. Both parameters increment enhances the probability of correct patterns recognition. However, these parameters take different effect on the model. Increment of q results in decrease of the amount of information corresponding to one synaptic connection, and increment of n allows accumulating more statistical information.

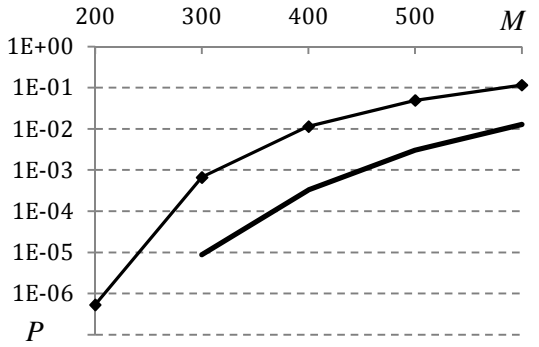

Fig. 4 Probability *P* versus the number of stored patterns *M*. Parameters *N*=100, *q*=100, *n*=2

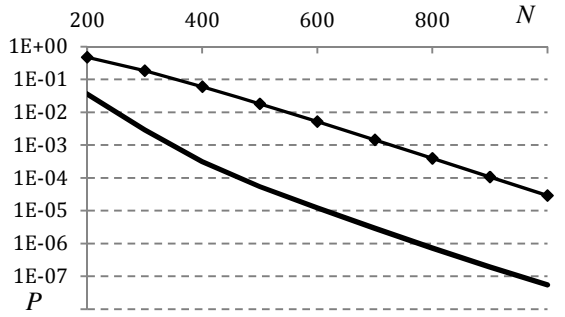

Fig. 5 Probability P versus dimensionality *N*. Parameters *M*= 1000, *q*=50, *n*=3

3. We conduct experiments on storage capacity of the proposed model, and verify the agreement between theoretical and practical results.
4. We also consider another useful option that is provided by the proposed model. That is a possibility of solving the K nearest neighbors task.

7.1 Comparison with a Single-Layer Perceptron

In this section we compare results of operation of a single- and double-layer perceptron.

In Figures 4-6 the Y-axis of the plots is the recognition error probability *P* (when the perceptron fails to recognize a distorted reference vector). In both figures the curves corresponding to the single-layer perceptron are represented by a thin line with rhombic marks (the curves are above the others). Other curves correspond to the double-layer perceptron. The plots are drawn for different *n* and *q*.

If the number of patterns *M*, their dimensionality *N,* and the noise level *a* (the probability of a component of an input binary vector being distorted) are determined by the conditions of a problem to be solved, the number of *q*-digit neurons of the inner layer and the number of their states can be varied to get satisfactory reliability.

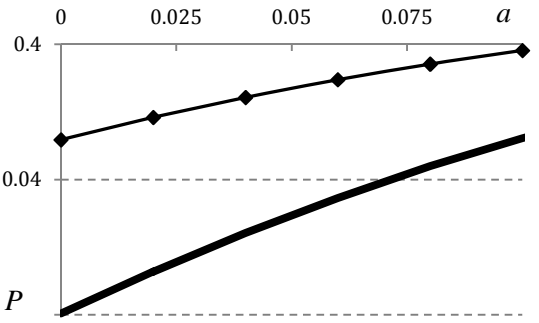

Fig. 6 Recognition failure probability P versus noise level a. M=1000, N=100, q= 200, n=2

Let us first consider how the recognition error probability varies with M and N given constant n and q (Fig. 4 and 5). As expected, the growth of dimensionality of stored patterns N or a decrease of their number M result in an exponential decrease of probability P. It is also seen that the introduction of another layer allows a more than an order of magnitude (two orders and more) decrease of P. The lower the probability P for the original single-layer net, the more significantly P decreases for the double-layer system.

The noise-resistance of the double-layer net is also higher – the rhomb-marked curve lies noticeably higher than the other curve (Fig. 6).

7.2 Model Properties Analysis

Fig. 7 shows us a few dependences of the double-layer network error probability P on the noise level a for different combinations of n and q (given $n*q$ = const). The upper dashed curve corresponds to n=40, q=10, the curve below – to n=8 and q=50. Even lower is the curve for n=4 and q=100. The combination of n=2 and q=200 (thick solid line) demonstrates the lowest P. So we see that from the reliability viewpoint it is better to use a small number of reliable (redundant) neurons for the double-layer system. However, such kind of networks cannot boast of high resistance to a failure of the net itself. The data (dashed line) shown in Fig. 7 proves that reliable and failure-resistant neural systems can be made up of unreliable elements having a considerable parameter spread.

The net with n=40 and q=10 differs from the net with n=2 and q=200 by the principles securing correct recognition. In the first case the second layer that accumulates information from a large number of unreliable elements plays a key role (for a single-layer perceptron with given parameters the recognition probability is zero). In the latter case, the second layer corrects the errors of the first layer only occasionally (thin marked line in Fig. 7).

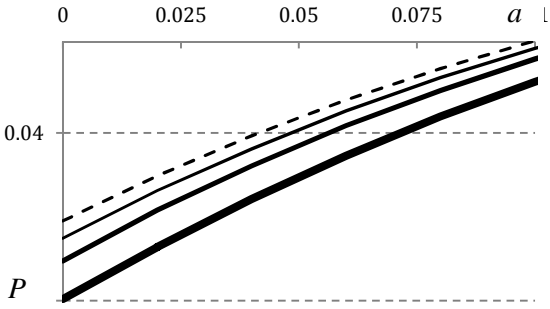

Fig. 7 Recognition failure probability P versus noise level a. M=1000, N=100

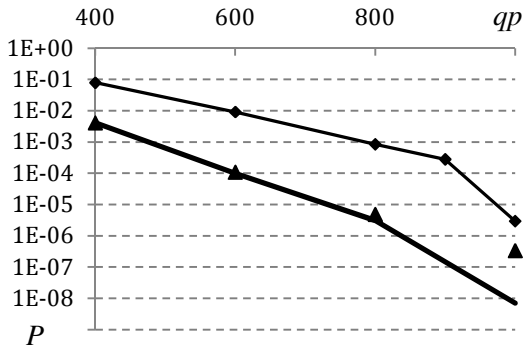

Fig. 8 Recognition failure probability P versus nq. M=1000, N=100, a=0

Fig. 8 shows how the error probability P depends on inner-layer parameters n and q. The thick line corresponds to the probability P of a double-layer network with n=2 and q=200÷500, and triangular marks correspond to n =2÷5 and q=200. Both networks have the same computational burden and requirements for RAM. The simulation shows that

1) The growth of both parameters leads to an exponential decrease of P;

2) Both nets has the same probability P for $nq < 800$ (an unexpected enough result), which once again says for the conclusion drawn above.

7.3 Storage Capacity

In this subsection we will present the experimental results of DLVP maximum storage capacity measurements and will check how well it corresponds to the theoretical estimate (6.1).

The solid line in figures 9-13 corresponds to theoretical estimate (6.1) calibrated on 2.5, markers are experimental points. The experiment was the following: we were looking for such number of reference vectors M, at which the probability of error recognition P would be equal $1/M$ at fixed parameters N, n, q and a, i.e. solving the following equation numerically:

$$P(M,N,n,q,a)=\frac{1}{M}. \tag{7.1}$$

From the plots represented in Fig. 9-13 we can see that the estimate (6.1) is consistent with the experiment quite well. Resulting curves verify the correctness of conclusions drawed at the end of section 6. It is especially worth noting that the storage capacity of DLVP increases linearly with n, while as the storage capacity of a single-layer perceptron decreases with $\ln(n)$ (see (6.2)).

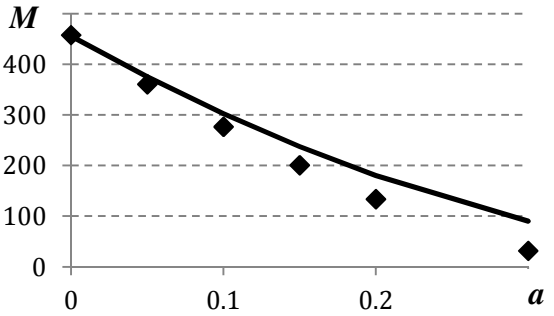

Fig. 9 DLP storage capacity M as a function of distortion level a. $N=100$, $q=50$, $n=4$, $P_{max}=1/M$

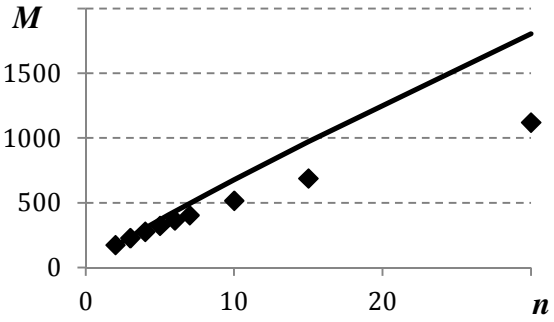

Fig. 10 DLP storage capacity M as a function of the number of vector neurons of an inner layer n. $N=100$, $q=50$, $a=0.1$, $P_{max}= 1/M$

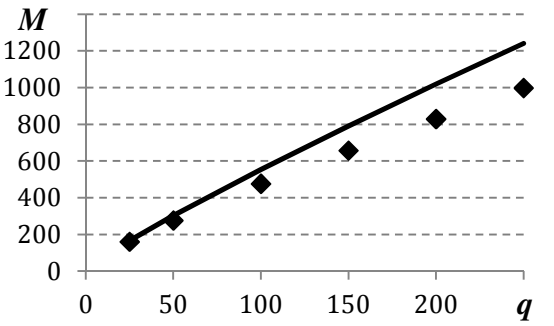

Fig. 11 DLP storage capacity M as a function of q. $N=100$, $a=0.1$, $n=4$, $P_{max}=1/M$

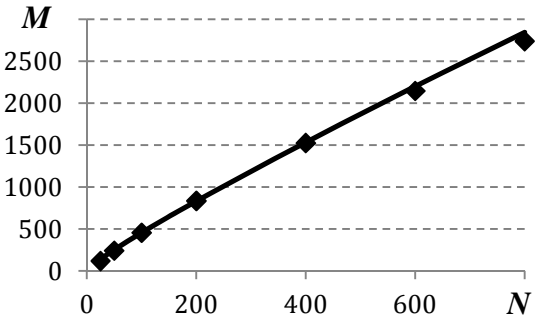

Fig. 12 DLP storage capacity M as a function of problem size N. $q=50$, $a=0.1$, $n=4$, $P_{max}=1/M$

7.4 K-Nearest Neighbors Search Task

The algorithm has yet another useful property, which a single-layer perceptron does not have. If we arrange patterns in decreasing order according to the components of their local field **H** (table 2, column "sum"), the order will tell us how close a pattern is to an input vector, while a pattern in the first place being regarded as the response of the system.

Let us demonstrate this by experiment. We will independently generate M random uncorrelated patterns, and additionaly another 5 patterns that are similar to each other to different extents (so they are correlated). The algorithm to generate these patterns is the following:

1) Generate random vector X_1;
2) Obtain vector X_2 by random distortion of 10% components of vector X_1;
3) Obtain vector X_3 by random distortion of 20% components of vector X_1;
4) Obtain vector X_4 by random distortion of 30% components of vector X_1;
5) Obtain vector X_5 by random distortion of 40% components of vector X_1;

Then, we will apply vector X_1 to the network inputs, and will monitor the values of the components of the local field **H**. Components of the local field corresponding to these 5 patterns will be greater than those corresponding to other components. At that, the maximum value of the local field will correspond to vector X_1 (since this vector was applied to the inputs). The second largest value will be the component corresponding to X_2, etc.

And indeed, the results of the experiment that are presented in Fig. 14 demonstrate it perfectly. Fig. 14 shows us distributions of the first six components of the local field **H** after applying vector X_1 to the inputs of the network. From this plot we can see that the spikes of the distributions are put in ascending order of patterns proximity to vector X_1.

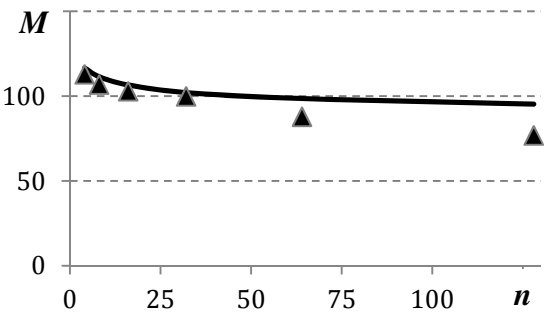

Fig. 13 Single-layer perceptron storage capacity *M* as a function of *n*. *N=100, q=50, a=0.1,* $P_{max}=1/M$. Solid line corresponds to the estimate (6.2), triangular markers corresponds to experimental points.

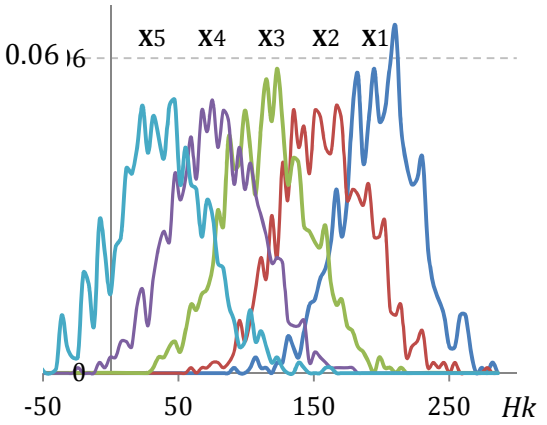

Fig. 14 Distributions of the first six components of the local field **H** after applying vector X_1 to the inputs of the network. *N=100, q=200, a=0, n=2.*

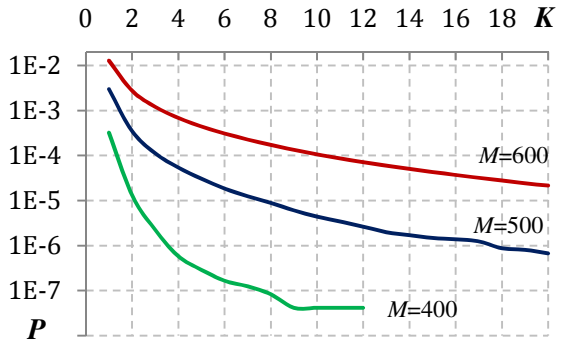

Fig. 15 Recognition error probability *P* as a function of the number of scalar products *K*. *N=100, n=2, q=100, a=0, M=400, 500, 600*

Such property allows us to solve the problem of *K* nearest neighbors search, which involves finding *K* reference patterns that are most similar to the input vector using Hamming distance. Alternatively, we can use this property to enhance the reliability of recognition for the problem of finding the first closest neighbor, i.e. for our task. To do this, we need to choose *K* reference patterns with the largest corresponding components of the local field **H**. Then, we need to

calculate scalar products of an input vector with these reference patterns, and choose the winner (it has the maximum scalar product). The result is that it becomes possible to significantly reduce the probability of recognition error at the cost of a couple of additional scalar products.

Fig. 15 shows us a very high efficiency of this improvement. We see that calculation of two additional scalar products (K=2), for example, results in decrease of recognition error P by about an order of magnitude, and at K=20 – by three orders. The gain is more, the smaller the error probabilty is in the first place.

8 Conclusion

The paper shows that it is possible to raise the efficiency of the single-layer vector perceptron by adding an extra layer. The remarkable efficiency of the algorithm is demonstrated. It is clearly shown that in contrast to a straight increase of network redundancy, purposeful construction of neural nets can give nice results.

The research is supported by projects ONIT RAN 1.8 and 2.1.

References

1. Wu, F.Y.: The Potts model. Review of Modern Physics 54, 235–268 (1982)
2. Kanter, I.: Potts-glass models of neural networks. Physical Review A 37(7), 2739–2742 (1988)
3. Kryzhanovsky, B., Mikaelyan, A.: On the Recognition Ability of a Neural Network on Neurons with Parametric Transformation of Frequencies. Doklady Mathematics 65(2), 286–288 (2002)
4. Austin, J., Turner, A., Lees, K.: Chemical Structure Matching Using Correlation Matrix Memories. In: International Conference on Artificial Neural Networks, IEE Conference Publication 470, Edinburgh, UK, September 7-10. IEE, London (1999)
5. Kryzhanovsky, V.M.: Ph.D. Thesis, Research into Binary-Synaptic-Coefficient Vector Neural Nets for Data Processing and Decision Making Problems, System Research Institute of the Russian Academy of Sci-ences (2010)
6. Kryzhanovskiy, V., Zhelavskaya, I., Fonarev, A.: Vector Perceptron Learning Algorithm Using Linear Programming. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 197–204. Springer, Heidelberg (2012)
7. Podolak, I.T., Biel, S., Bobrowski, M.: Hierarchical classifier. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 591–598. Springer, Heidelberg (2006)
8. Podolak, I.T.: Hierarchical classifier with overlapping class groups. Expert Systems with Applications 34(1), 673–682 (2008)

Appendix 1

Now, we will present a nonstrict analytical derivation of DLVP storage capacity in the engineer style, in which we will neglect different correlations and make some simplifications. Only in this case, it becomes possible to deduce the resulting expression but not another unsolvable theorem. As it was shown above (in subsection 7.3), the final estimate (6.1) is in a good agreement with experiments regardless of introduced simplifications.

If we take a particular set of reference patterns and a DLVP trained on this set, the result of the recognition of particular pattern **X** will be deterministic and nonprobabilistic (note that the process of training DLVP is also deterministic). At such approach one cannot speak of the error recognition probability and moreover, reason about storage capacity. However, let us try another approach.

Assume we have 1000 DLVPs trained on different sets of reference patterns. Let us apply first patterns $\mathbf{X}_1$ from each set to the inputs of DLVPs accordingly. Let us consider their local fields **H**. We will denote k-th component of these fields as H_k. In the case of correct recognition, first components of the local fields H_1 should be greater than other components H_i, $i = 2, 3,\ldots, M$. Then, 1000 of the first and the second components of the local fields **H** can be considered as realizations of two random variables H_1 and H_2 (as you understand, the number of DLVPs can be any, and 1000 is just an example). So, since H_k are random variables, (M-1) inequalities

$$\begin{cases} H_1 > H_2 \\ H_1 > H_3 \\ \vdots \\ H_1 > H_M \end{cases} \tag{A.1}$$

will hold with some probability 1–P, where

$$P = 1 - \Pr\left[\bigcap_{k=2}^{M} H_1 > H_k\right] \tag{A.2}$$

is a probability of error recognition. Thus, if we can analytically estimate a functional relationship between P and all the model parameters, which are reference patterns size N, number of reference patterns in the training set M, internal parameters n and q, and the noise level a, then we will be able to find model reliability at specific parameters values. This will be our first goal.

1. Error Probability

Let us neglect some aspects, which will help us to obtain an expression for P:

1. Random values H_i are dependent and correlate with each other, but we will assume that they are independent.
2. Events (H_1>H_2), (H_1>H_3), …, (H_1>H_M) are also dependent (since they all depend on H_1), but we will assume they are not.

It is obvious that each such approximation results in discrepancies between theory and experiment. But this is a price we have to pay. Eventually, we can make an approximate estimate of (A.2) in the following way:

$$P = 1 - \prod_{k=2}^{M} \Pr\left[H_1 > H_k\right]. \tag{A.3}$$

Now, we can focus on each random variable H_k separately, to evaluate its distribution function and its statistical characteristics.

2. Distribution Function of H_k

To evaluate distribution of H_k one need to substitute (3.2) and (3.3) in (3.4):

$$\mathbf{H} = \sum_{m}^{M} \mathbf{o}_m \sum_{\mu}^{M} \sum_{i}^{N} \sum_{j}^{n} (\mathbf{y}_j^m, \mathbf{y}_j^\mu) x_i^\mu \tilde{x}_i^1, \tag{A.4}$$

where index "1" and tilde in the last multiplier $\tilde{x}_i^1$ emphasizes that the first reference pattern ($\mathbf{X}_1$) was applied to DLVP inputs (see (3.3), where x_i denotes i-th component of an input vector, which is $\mathbf{X}_1$ in our case), and this pattern had aN of its components distorted, $0<a<0.5$. Recall that $\mathbf{o}_m$ is an M-dimensional unit vector containing 1 at m-th position. Subject to the last note, m-th component of vector $\mathbf{H}$ will be

$$H_{\mathrm{m}} = \sum_{\mu}^{M} \sum_{i}^{N} \sum_{j}^{n} (\mathbf{y}_j^{\mathrm{m}}, \mathbf{y}_j^\mu) x_i^\mu \tilde{x}_i^1 \tag{A.5}$$

We can see from (A.5) that H_m is basically a sum of a great number of «+1» и «-1», which can take only integer values, and therefore, its distribution is discrete. We can approximate it with the normal distribution with a reasonable accuracy. Next, we need to evaluate expected means and variances of random variables H_m.

It is worth going through the approach we conduct our analysis at one more time. We consider the multipliers in expression (A.5) as random variables. Their realizations correspond to particular sets of reference patterns and a particular input vector. So, for instance, we consider x_i^μ, which is i-th component of reference pattern $\mathbf{X}_\mu$, as a random variable, which can take either +1 or -1 equiprobably (expected mean of this value is 0, and standard deviation is 1).

3. Statistical Properties of H_k

Looking ahead, it is worth considering random variable H_1 separately from the rest of the components H_2, H_3, …, H_M, since their statistical properties are different.

To estimate the expected mean and the variance of H_1, we need to extract additives having μ=1 from (A.5):

$$H_1 = \sum_i^N \sum_j^n (\mathbf{y}^1_j, \mathbf{y}^1_j) x^1_i \tilde{x}^1_i + \sum_{\mu \neq 1}^M \sum_i^N \sum_j^n (\mathbf{y}^1_j, \mathbf{y}^\mu_j) x^\mu_i \tilde{x}^1_i \tag{A.6}$$

Taking into account the way variables $\mathbf{y}^\mu_j$ were defined in 3.1, we get the following:

$$(\mathbf{y}^1_j, \mathbf{y}^1_j) \equiv 1 . \tag{A.7}$$

Considering that input vector $\mathbf{X}$ is basically vector $\mathbf{X}_1$ with aN distorted components, we get the following as well:

$$\sum_i^N x^1_i \tilde{x}^1_i \equiv (1-2a)N \tag{A.8}$$

The first sum in (A.6) is equal strictly to $(1\text{-}2a)nN$, so (A.6) becomes:

$$H_1 = (1-2a)nN + \sum_{\mu \neq 1}^M \sum_i^N \sum_j^n (\mathbf{y}^1_j, \mathbf{y}^\mu_j) x^\mu_i \tilde{x}^1_i \tag{A.9}$$

By signal we shall call the first part of (A.9), and by noise – the second part. Let us consider multipliers of the second sum: $(\mathbf{y}^1_j, \mathbf{y}^\mu_j)$, x^μ_i and $\tilde{x}^1_i$. They are independent, so

$$E\left[(\mathbf{y}^1_j, \mathbf{y}^\mu_j) x^\mu_i \tilde{x}^1_i\right] = E\left[(\mathbf{y}^1_j, \mathbf{y}^\mu_j)\right] E\left[x^\mu_i\right] E\left[\tilde{x}^1_i\right] = 0 , \tag{A.10}$$

where $E[]$ is expectation operator. Thus, the noise term has zero mean, but has a very large variance. The noise term may turn out to be larger than the signal $(1-2a)nN$ due to statistical outliers, which will cause the error in recognition.

Ultimately, we can estimate the mean and the variance of H_1 as:

$$\begin{aligned} E[H_1] &= (1-2a)nN \\ D[H_1] &= \frac{nNM}{q} \end{aligned} \quad . \tag{A.11}$$

Random variables H_2, H_3, …, H_M variables cannot be divided into the signal and the noise terms, since they have only noise terms. These variables have same statistical characteristics:

$$\begin{aligned} E[H_k] &= 0 \\ D[H_k] &= \frac{nNM}{q}\left(1 + \frac{(n-1)q}{M} + \frac{N-1}{M}\right) \end{aligned} \cdot \tag{A.12}$$

Estimates of variances in (A.11) and (A.12) were obtained similarly to the mean estimate in (A.10). It is necessary to take into account distribution functions

of random variables $(\mathbf{y}^1_j, \mathbf{y}^\mu_j)$ and x^μ_i when deriving the estimates of mathematical expectations and variances:

$$(\mathbf{y}^1_j, \mathbf{y}^\mu_j) = \begin{cases} -1, \textit{ with a probability of } \dfrac{1}{2q} \\ +1, \textit{ with a probability of } \dfrac{1}{2q} \\ 0, \textit{ otherwise} \end{cases} \tag{A.13}$$

and

$$x^\mu_i = \begin{cases} -1, \textit{ with a probability of } \dfrac{1}{2} \\ +1, \textit{ with a probability of } \dfrac{1}{2} \end{cases} \tag{A.14}$$

4. Error Probability Estimate *P*

Let us introduce a new random variable combining both random variables H_1 and H_k:

$$\Delta_k = H_1 - H_k, \; k = \overline{2, M} \tag{A.15}$$

The assumption made above that variables H_k are independent allows us to easily calculate statistical characteristics of Δ_k:

$$\begin{aligned} E[\Delta_k] &= (1-2a)nN \\ D[\Delta_k] &\approx \frac{nNM}{q}\left(2 + \frac{nq}{M} + \frac{N}{M}\right) \end{aligned} \tag{A.16}$$

Variable Δ_k is normally distributed, so the probability of event $\Delta_k > 0$ (see multipliers in (A.3)) is defined by the following expression:

$$1 - P_k = \Pr[\Delta_k > 0] = \frac{1}{\sqrt{2\pi D[\Delta_k]}} \int_0^\infty \exp\left(-\frac{(\xi - E[\Delta_k])^2}{2D[\Delta_k]}\right) d\xi \tag{A.17}$$

We can write the expression of error recognition probability as

$$P = 1 - (1 - P_k)^{M-1}. \tag{A.18}$$

Since we are interested in the case when error probability is $P \to 0$ (so the neural network works very reliably), we can make the following estimate of (A.17):

$$P \approx MP_k . \tag{A.19}$$

Thus, we need to estimate probability P_k. There is a range of expansions for the error function

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt \tag{A.20}$$

These expansions allow us to calculate probability P_k approximately. So, let us express (A.16) in terms of error function:

$$P_k = \frac{1}{2} erfc(\gamma), \tag{A.21}$$

where

$$\gamma^2 = \frac{E^2[\Delta_k]}{2D[\Delta_k]} = \frac{nqN(1-2a)^2}{2M}\Theta, \qquad \Theta = \left(2 + \frac{nq}{M} + \frac{N}{M}\right)^{-1}. \tag{A.22}$$

The greater the value of γ^2 is, the smaller the error probability *P* is. Therefore, the approximation of (A.17) by (A.18) is made when $\gamma^2 >> 1$.

Let us take the first additive from the well-known error function expansion

$$erfc(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} \left[1 + \sum_{r=1}^{\infty} (-1)^r \frac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{\left(2x^2\right)^r} \right] \tag{A.23}$$

By doing so we get the resulting error probability expression:

$$P = \frac{M}{2\gamma\sqrt{\pi}} e^{-\gamma^2} \tag{A.24}$$

The estimate (A.23) describes the model in a qualitative manner. However, it is inconsistent with the experiment by several orders. Such significant difference is coming from the point that error probability is in exponential relationship with the parameters of the model:

$$P \sim e^{-\gamma^2} \tag{A.25}$$

Even minor errors in calculations of this exponential factor lead to significant deviations from the experiment, since $\gamma^2 >> 1$. Authors realized that introduced approximations and assumptions would lead to directly that. However, though expression (A.23) is of interest, but it is not the final goal of our derivations. Based on it, we will get a consistent estimate of storage capacity.

5. Storage Capacity

According to the definition of the storage capacity of associative memory $M_{\max}$ given in section 6, it is such number of reference patterns, recognizing which the probability of error P is not greater than a predefined threshold $P_{\max,}$ and input vectors are distorted at the noise level $a \geq 0$. Therefore, we need to solve the following equation for M:

$$P\left(M, N, n, q, a\right) \leq P_{\max}. \tag{A.26}$$

Let us take a logarithm of the right and the left-hand side of (A.25):

$$\gamma^2 \geq \ln \frac{M}{2\gamma P_{\max} \sqrt{\pi}} \tag{A.27}$$

Then, let us substitute γ^2 in this expression:

$$M \leq \frac{nqN(1-2a)^2}{\ln\left(\dfrac{M^3}{2\pi\left(nqN(1-2a)^2\Theta\right)P_{\max}^2}\right)}\Theta \tag{A.28}$$

Variable M in (A.27) is both in the right and the left-hand sides of the formula. Therefore, let us use the following trick: we will recurrently insert (A.27) into itself:

$$M \leq \frac{nqN(1-2a)^2}{2\ln\left(\dfrac{nqN(1-2a)^2\Theta}{P_{\max}\sqrt{2\pi}\ln^{3/2}(...)}\right)}\Theta \tag{A.29}$$

There still left multipliers Θ depending on M in the right-hand side of (A.28). Let us try to eliminate this dependency. If we lower the estimate of storage capacity M we will only strengthen inequality (A.25). Therefore, let us give a raw lower estimate of Θ:

$$\Theta \approx \frac{1}{4} \tag{A.30}$$

Eventually, we get the final expression for the estimate of DLVP storage capacity:

$$M < C\frac{nqN(1-2a)^2}{8\ln\left(\dfrac{nqN}{4\sqrt{2\pi}P_{\max}}\right)},\ C = 2.5\,. \tag{A.31}$$

According to multiple experiments, (A.30) gives a good qualitative description of the model, however, in order to be in a good agreement with the experiment a normalization constant C=2.5 is required.