Sparse Binary Distributed Encoding of Scalars

Article in Journal of Automation and Information Sciences · January 2005

4 authors, including:

Dmitri A. Rachkovskij
National Academy of Sciences of Ukraine

Serge V. Slipchenko
International Research and Training Center for Information Technologies and Syste…

Ernst Kussul
Universidad Nacional Autónoma de México

Please cite as:

Dmitri A. Rachkovskij, Sergey V. Slipchenko, Ernst M. Kussul, Tatyana N. Baidyk. Sparse binary distributed encoding of scalars. Journal of Automation and Information Sciences. 2005. Vol. 37, no 6. pp. 12–23.

Abstract: Hypervectors (then known as codevectors) of scalars are considered.

Errata

Due to the translation of the original publication in Russian, which was done without consulting the authors, the translated article has several terminological errata:

–"unities" are met throughout the article, e.g.:

- M unities and $N - M$ zeros;
- the number of unities in **X**.

It means unit vector components (or just 1s).

– "units" are vector components with the value of 1, as well.

– "tenuity" means "sparsity".

– "Hemming" means "Hamming".

– On pages 16-17, and in some other places, "flocets" are met. "Flocet" should be interpreted as "float".

– "module of difference" means "absolute value of difference".

– "in the air" means "on-the-fly".

Sparse Binary Distributed Encoding of Scalars

D.A. RACHKOVSKIY, S.V. SLIPCHENKO, E.M. KUSSUL, T.N. BAIDYK

Distributed codes, their properties, similarity measures and other characteristics are described. A series of procedures of stochastic distributed encoding of scalar quantities is considered. Methods of control the slope of codes overlap are defined. The obtained codes can be used for representation of numerical vectors.

Key words: information processing, encoding, information capacity, distributed representation, thermometric encoding, flocet encoding, stochastic encoding.

Introduction

The way of representing data significantly affects efficiency of methods and algorithms of its processing. In distributed representation, data is encoded by multidimensional vectors, where individual elements usually have no single-valued meaning. In this sense they differ from symbolic and local representations, which are in unique correspondence to these or those objects. By objects here we understand data of different nature and complexity: numerical and symbolic, scalar, vector, structured etc. objects. Distributed representation of information is used both in solution of applied problems (recognition, classification, associative memory, control etc.), and in modeling cognitive processes of different levels of complexity, from sensor perception and sensor-motoric coordination up to memory, categorization and reasonings by analogy (see also [1]).

Advantages of distributed representations include [2–5]:

— efficient using resources for representation of information (high information capacity is ensured by possibility of representation of exponentially large number of objects by different codevectors of the same dimension);

— simple estimation of degree of similarity (using scalar product of codevectors of distributed representations);

— natural representation of similar objects (by correlated codevectors, i.e., by codevectors with a large magnitude of scalar product).

Other known advantages of distributed representations include possibility of their forming with learning by samples, ability to operate under conditions of noise, failures and indeterminacy, and also parallelism, neurobiological relevancy etc. (see [6, 7]). Distributed representations are closely related to the paradigm of cellular ensembles [8], which made a great influence on work of many researchers in different fields: from artificial intelligence to cognitive psychology and neurophysiology.

In the approaches to distributed representation of information, developed in the framework of paradigm of associative-projective neural networks (APNN) [1, 3, 9], information is encoded by so called binary *codevectors*, vectors with a small fraction of randomly placed units. This provides an opportunity to represent and to process structured data, to simplify implementation of algorithms and to use effective distributed associative memory.

In many technical applications, and also in simulation of cognitive activity one should be able to operate on numerical data. Recently, approaches, methods and procedures of distributed representation (or encoding) numerical data are insufficiently developed. This is in full measure related to stochastic binary sparse distributed encoding.

In this paper we study different schemes of encoding integer-valued scalar quantities using binary codevectors. The most important property of the developed encoding schemes is as follows. Proximity of numerical quantities in input (signal) space should be reflected in similarity of their codevectors in the secondary (resulting or output) space. For example, to close scalar quantities should correspond close binary codevectors, i.e., codevectors with a large number of coinciding units.

Amount of codevectors overlap after transformation is not strictly proportional to difference of numbers in the initial space, but is nonlinearly transformed instead. Derived characteristics of similarity of codevectors can be useful for solving some problems of classification, approximation, control etc., more adequately reflecting similarity of data samples in data domain. Many schemes of encoding vector data, applied in solution of these problems, are based on encoding their elements using methods, considered in this paper.

1. Distributed representations and their characteristics

In this paper we study distributed representation of numerical values of variables or features. As distributed representation, mainly codevectors with binary elements {0, 1} are used. From this point onwards, boldface letters (**X**) denote codevectors and italic letters (*X*) denote corresponding numerical values of other objects.

In binary stochastic distributed representations, unities in a codevector are placed on random positions, however to the same object corresponds the same codevector (i.e., codevectors are *pseudorandom*). Unities, randomly placed in different codevectors, can be on the same positions. The number of different N-dimensional codevectors with M unities and $N-M$ zeros is C_N^M.

Degree of similarity of two objects (for example, numerical values) can be represented by degree of correlation (or overlap) of their codevectors. For representation of dissimilar objects (for instance, symbols, independent features or distant numerical values) usually independent random codevectors [3, 9] are used. Codevectors with a larger value of overlap are used for representation of similar objects (for instance, dependent features or close numbers).

Degree of similarity of codevectors will be estimated by scalar product. For binary codevectors, scalar product is the number of coinciding unities. Sometimes we speak about codes overlap as fraction of coinciding unities, i.e., ratio of their number to N. *The relative* asymmetric overlap of codevectors **X** and **Y** (overlap, referred to the number of unities in **X**) we define as

$$V(\mathbf{X}, \mathbf{Y}) = |\mathbf{X} \wedge \mathbf{Y}| / |\mathbf{X}|, \tag{1}$$

where $\wedge$ is the elementwise conjunction, $|\mathbf{X}|$ is the number of unities in **X**.

Let **X**, **Y** be random vectors, $|\mathbf{X}| = M$, $p(\mathbf{X}) = p_\mathbf{X} = |\mathbf{X}|/N = M/N$, $p(\mathbf{Y}) = p_\mathbf{Y} = |\mathbf{Y}|/N$. Average of distribution of the number of common unities in random independent vectors **X**, **Y** is $p_\mathbf{X} p_\mathbf{Y} N$. Quantity $V(\mathbf{X}, \mathbf{Y})$ varies from $p_\mathbf{X} p_\mathbf{Y} N / M = p_\mathbf{Y}$ to 1 (in the case of the same vectors).

Scalar product of random codevectors is approximately the same for large N. Overlap for two random binary codevectors is in mean $(M/N)^2$. For dense random binary codevectors ($M = 0.5N$) overlap will be of order 0.25. For sparse random binary codevectors (for instance, $M < 0.01N$) overlap will be less, than 0.0001. In APNN, long sparse random codevectors are used. For example, often codevectors with $M \approx 1000$, $N \approx 100{,}000$ are used.

Overlap U, whose magnitude changes from 0 to 1 independently of M, N [10], is called *normalized*:

$$U(\mathbf{X}, \mathbf{Y}) = \sum_i (\mathrm{X}_i - p_\mathbf{X}) \mathrm{Y}_i / (N p_\mathbf{X} (1 - p_\mathbf{X})). \tag{2}$$

If $p_\mathbf{X} = p_\mathbf{Y} = p$, then

$$U(\mathbf{X}, \mathbf{Y}) = (V - p)/(1 - p). \tag{3}$$

For visual presentation of the character of codes similarity further we use plots of characteristics of codes overlap, i.e., dependence of magnitude of overlap of codes V and U on the difference between encoded quantities.

Consider normalized quantities $x \in [0, 1]$ Number x^* from interval $[x_{\min}, x_{\max}]$ can be transferred into [0, 1], for example, in the following way:

$$x = (x^* - x_{\min})/(x_{\max} - x_{\min}). \tag{4}$$

Transformation of scalar quantities into codevectors should be such, that

$$|x - y| > |x - z| \Rightarrow V(\mathbf{X}, \mathbf{Y}) \le V(\mathbf{X}, \mathbf{Z}), \tag{5}$$

where V is the overlap of codevectors; x, y, z are numbers (scalar quantities).

For encoding real number $x \in [0, 1]$ with an accuracy to Q gradations we can put into correspondence to it the number of its gradation q, for instance:

$$q = \text{ent}\,(xQ), \tag{6}$$

where ent is integer part.

This enables one further consider encoding of integer numbers instead of real numbers in the interval $[0, Q]$. For representation of k, $(k < Q)$ gradations of similarity of numerical values we need $M \ge k$ unities in codevector. At $M = k$ differences in the code for one gradation of similarity are minimal, just unity. Such differences are statistically unstable for stochastic encoding, that's why the condition $M \gg k$ should be, as a rule, satisfied.

1.1. Requirements to encoding

Considered codevectors are multidimensional, binary, pseudorandom (with random, but the same positioning of unities for the same encoded data) and sparse (the number of unity elements is much less, than the number of equal to zero ones). Such codevectors ensure a high information capacity, noise immunity and possibility of working with random subsets of code with preservation of properties of the whole code. Sparse binary vectors admit efficient implementation of vector and vector-matrix operations. Thanks to tenuity, a higher information capacity of associative memory, which can be used for storage and search of codevectors [11], is ensured.

The number of unity elements M in sparse codevectors should be significant for maintenance statistical stability of the number of unities and reducing its deviation about the mean value. In its turn, this is necessary for ensuring accuracy of encoding and decoding (restoring the vector of input signal space by its binary codevector). For different values of encoded quantities M should be approximately equal, in order to different density of codes would not affect amount of their overlap and functioning subsequent algorithms.

Close scalar quantities should be associated with close binary codevectors, i.e., codevectors with a large number of coinciding unities. Such character of overlap will allow us to apply to codevectors methods and algorithms, based on estimation of similarity by means of scalar product, and also can be useful for solution of some problems of classification, approximation, control etc. We should also ensure control of characteristic of overlap of the obtained codes (i.e., degree of their overlap depending on distance in the input space).

We developed methods of representation by such codevectors of different types of data, numeric scalars, considered in this paper, vectors and data with more complex structure [2, 3]. This promotes development of a unified (universal) approach to processing inhomogeneous information and creating several new intelligent information technologies. These technologies support both classification of vector information and processing complex structured information for artificial intelligence systems [2, 12].

Thus, for ensuring distributivity, uniformity and efficiency of representation and processing information the codevectors $\mathbf{X}$ of integers x should satisfy the following requirements:

— binarity (elements 0 and 1);

— multidimensionality (large number of elements, $N \gg 1$);

— pseudostochasticity (random, however unchanged for the code of the same number, location of unities);

— tenuity (the number of unity elements M is much less, than the number of zero ones, $M \ll N - M$);

— large number of unity elements ($M \gg 1$);

— approximately equal number of unities for different encoded numbers ($M \sim \text{const}$);

— regulated characteristic of codes overlap.

2. Encoding scalar quantities

In this section we consider several approaches to distributed encoding of scalar quantities, schemes of partially distributed encoding and of distributed stochastic encoding. We analyze their properties and also correspondence to requirements of Section 1.1.

2.1. Partially distributed thermometric encoding

In thermometric encoding, under increasing value of encoded quantity, more and more neighboring neurons from the encoding pool are activated, like in movement of thermometer column [13–15]. This familiar technique of binary encoding [13] is mentioned by Widrow in sixties of the past century [16] and is called by *stack*.

The number of neurons $M(q)$, activated in the course of thermometric encoding of quantity q, is

$$M(q) = \text{ent}\,(Nq/Q). \tag{7}$$

Function $X_i(q)$ of activation of the i-th neuron used in encoding quantity q has the following form:

$$X_i(q) = I(i < M(q)),\;\; i = \overline{1, N}, \tag{8}$$

where $I(i < M(q))$ is the indicator function, taking value 1 if $i < M(q)$ and 0 otherwise.

In this encoding scheme the number of neurons N regulates accuracy of representation of numbers. It should be greater, than the number of gradations Q. Examples of thermometric encoding of several values are shown in Figure 1.

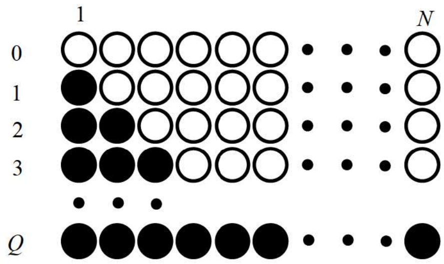

Figure 1

In thermometric encoding there exists asymmetry between codes of large and small values. Such representation obeys relationship "subset" (≤) for encoded numbers. The subset of neurons, which is encoding small values of a quantity, belongs to the subset, which is encoding its larger values. In encoding of a larger value neurons, representing the whole spectrum of its smaller values, are activated. Encoding is not sparse, since many codevectors contain more unities, than zeros. Forming codes with approximately fixed number of unities also is not ensured.

Proximity of integers in thermometric encoding can be measured in Hemming metrics. Hemming distance between two codevectors, which are encoding two numerical values, is monotonically increasing with the difference of these values. Characteristic of codes overlap at that changes nonmonotonically. It depends not only on difference of encoded values, but also on the values of these quantities. It is asymmetric with respect to increasing quantities (remains equal to the starting value) and with respect to decreasing (smoothly decreases, remaining equal to the smaller value).

Supplement of thermometric code by "inverse" one on one more neuron pool (for instance, [16]), makes number of active neurons in the code constant.

Characteristic of overlap at that becomes symmetric. Code is not sparse. The activation function has the form

$$X_i(q) = I(i < M(q) \vee i > N + M(q)). \tag{9}$$

2.2. Partially distributed flocet encoding

So-called "flocet" scale enables one forming codes with a fixed number of unities (see [17]). Different values of quantities are encoded by equal number of active neurons. Codes of values differ in location of the group of active neurons. For preserving metric, close values of the quantity are encoded by intersecting subsets of neurons, so called "flocets" (Figure 2, flocet encoding: q is the encoded value, $1 \ldots N$ is the encoding pool, $B = 3$ is the flocet length). This code is also known as the code of pointer, code of proximity, code of column diagram [13] or sliding code.

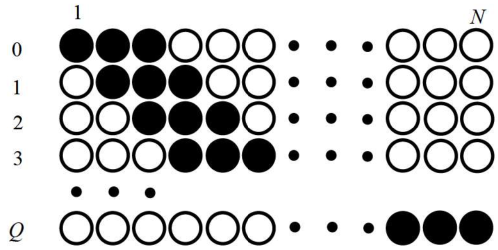

Figure 2

The function $X_i(q)$ of activation of the i-th neuron in encoding quantity q has the following form:

$$X_i(q) = I(i \in [qN/Q, qN/Q + w]), \;\; i = \overline{1, N}, \tag{10}$$

where $w = N - Q$ is the flocet width.

In this case, characteristic of codes overlap has a piecewise-linear character. Within flocet dimensions, overlap is linearly decreasing when module of difference of encoded quantities increases. Outside the flocet overlap is zero. The width of the slanted portion of overlap characteristic can be changed at the expense of changing flocet length. For the flocet length $B = N/2$, nonzero codes overlap takes place for all pairs of values of the encoded quantity (Figure 3). Unlike thermometric encoding, overlap characteristic has the same form for all values of encoded quantities.

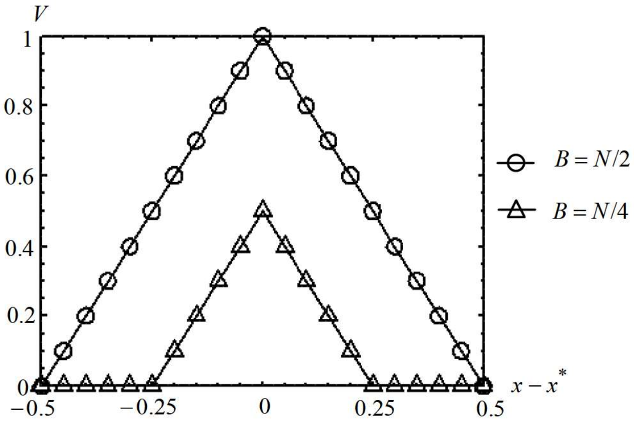

Figure 3

2.3. Partially distributed multiflocet encoding

Many methods of recognition and approximation use mapping functions φ into code space such, that scalar product of codes is nonlinearly depending on the distance between points in the original space (for example, for potential or radially-basis functions [18]).

For obtaining analogous characteristic of codes overlap one can use the multiflocet encoding method [19]. Numerical value of quantity is encoded by several flocets of different lengths within their neuron pools, which are concatenated into the common codevector. At that, the role of each flocet decreases. Characteristic of codes overlap is the sum of overlap characteristics for each flocet and piecewise-linearly approximates a bell-like shape of function (Figure 4). The activation function is

$$X_i^j(q) = I(i \in [qN/Q^j, qN/Q^j + w^j]), \; i = \overline{1, N}, \tag{11}$$

where $w^j = N - Q^j$ is the flocet width, j is the number of the pool.

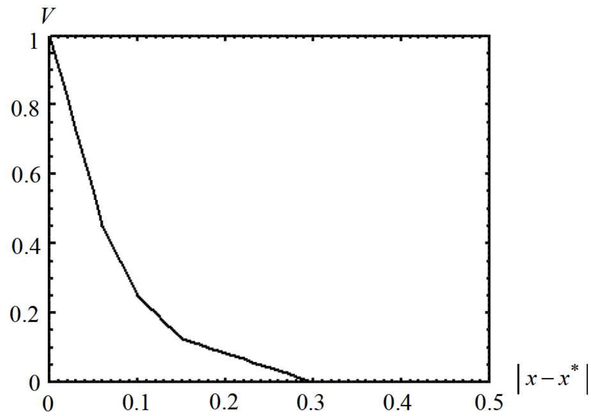

Figure 4

Analysis of schemes of thermometric and flocet encoding demonstrates, that partially distributed encoding has disadvantages, which restrict possibility of their use in applications. So, in the both schemes information capacity of code increases linearly, not (quasi) exponentially. No generalization of these schemes for the case of encoding vector quantities and more complex types of data by codevectors of the same dimension was suggested. These limitations can be overcome in the developed by the authors schemes of stochastic encoding (next section). The scheme of multiflocet encoding is an interim variant between the partially and the completely distributed encoding.

2.4. Distributed stochastic encoding

The developed here schemes of distributed binary encoding of numeric quantities give codevectors with approximately equal low density $p = M/N \ll 1$ and pseudorandom location of unity elements. To close numbers should correspond strongly intersecting codes, and to distant numbers, slightly intersecting ones. Moreover, codevectors of distant numerical values have "residual" intersection p^2N, which is equal to intersection of independent random vectors with density of unities p. By the formula of relative overlap (1), residual intersection is $V=p$, and by the formula of normalized overlap (2), residual intersection is $U=0$.

Below we consider algorithms of distributed stochastic encoding of numerical quantities using binary sparse codevectors. Similar algorithms for "diffused codes" with a high density of unities ($p=0.5$) were considered in [20].

2.5. Subtractive-additive encoding procedure

Procedure of code generation consists in modification of codevectors by deleting and adding of part of their unities, i.e., decimation and supplement. For initial value from the range of variation of encoded quantity let us generate a sparse random vector with fraction of unities p. We call it the reference codevector. For obtaining codevector, corresponding to the next gradation (value) of the value, we substitute selected in random way $k=p_kN$ units for zeroes and the same number of zeroes we substitute for units. For obtaining codevector of the next gradation of coded value we repeat this operation over the codevector of the previous gradation.

"Cancellation" of k unities can be performed by conjunction with a *decimating* vector. As the latter, we can use random vector $\mathbf{Y}_q$, containing $(1-p_k)N$ unities. Activation of k unities can be performed by disjunction with *supplemental* random vector $\mathbf{Z}_q$, containing $Np_k p/(1-p+p_k p)$ unities. Number k or p_k determines steepness (slope) of overlap characteristic.

Thus, codevector $\mathbf{X}_q$ of the q-th value of encoded quantity is constructed as follows:

$$\mathbf{X}_q = (\mathbf{X}_{q-1} \wedge \mathbf{Y}_q) \vee \mathbf{Z}_q, \tag{12}$$

where $\mathbf{X}_{q-1}$ is the codevector of the $(q-1)$-th encoded value; $\mathbf{Y}_q$ is the decimating vector containing k zero elements; $\mathbf{Z}_q$ is the supplemental vector, containing k unity elements.

For example, for $k = 0.5M$ codes of neighboring gradations will contain approximately half of identical unitary components, codes of quantities, different in two gradations, approximately quarter of unities etc. Moreover, the mean value of amount of codevectors overlap of distant numerical quantities is equal to intersection of two independently generated random vectors $M^2/N^2 = p^2 N$. Examples of experimental (V, U) and theoretical (V_T, U_T) characteristics of codes overlap are shown in Figure 5, where $V(\mathbf{X}, \mathbf{X}_0)$ is the relative overlap of the current and the initial (reference) codevectors; $V(\mathbf{X}, \mathbf{X}_1)$ — overlap of the current and terminal (supporting) codevectors; U is the normalized overlap: $N = 10000$, $M = 100$, $Q = 20$, $p_k = 0.25$.

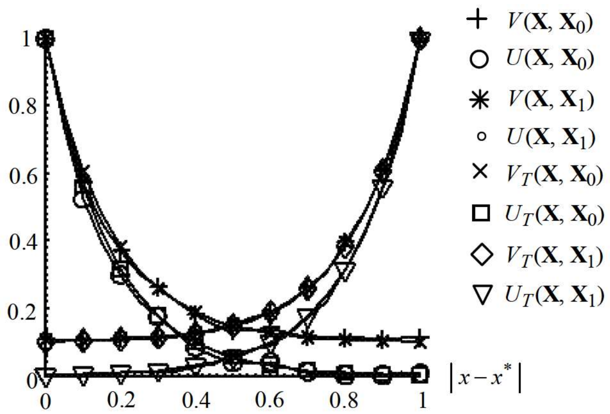

Figure 5

This procedure of code generation ensures not exact, but approximate support of the number M of unities in code and the fraction of changed unities k. Relative deviation of actual values from their averages of distribution decreases when M increases. This is one of reasons of necessity of using codevectors of large dimension $N = M/p$. This procedure can be efficiently implemented on neural computers with a large digit capacity of computer word [21].

In Figure 5 we show analytical form of characteristic of the relative code overlap (1):

$$V(\mathbf{X}, \mathbf{X}^*) = (p + (1-2p)(1-p_k)^{|x-x^*|})/(1 - p(1-p_k)^{|x-x^*|}),$$

$$V(\mathbf{X}, \mathbf{X}_0) = p + (1-p)(1-p_k)^{|x-x^*|}, \tag{13}$$

and of the normalized one (2):

$$U(\mathbf{X}, \mathbf{X}^*) = (1-p)(1-p_k)^{|x-x^*|}/(1 - p(1-p_k)^{|x-x^*|}),$$

$$U(\mathbf{X}, \mathbf{X}^*) = (1-p_k)^{|x-x^*|}. \tag{14}$$

Estimation of computational complexity of the described procedure can be performed on the basis of estimate of computational complexity of its main components:

1) generation of initial vector, $O(pN)$ operations;

2) generation of decimating vector, $O((1-p_k)N)$ operations;

3) decimation (conjunction), $O((p + (1-p_k))N)$ operations for the sparse vector;

4) generation of complementary vector, $O(p_k p/(1-p+p_k p)N)$ operations;

5) addition (disjunction) is $O((p_k p/(1-p+p_k p)+(p-p_k))N)$ operations for the sparse vector;

Total: $O((2(p+(1-p_k))+2p_k p/(1-p+p_k p)+(p-p_k))N)$ operations for the sparse vector.

2.6. Encoding by concatenation of parts of reference codevectors

For obtaining a more linear characteristic of overlap we can use the following encoding procedure. Let us take two random *reference* vectors: $\mathbf{X}_0$ and $\mathbf{X}_1$. Put one of them into correspondence to the lower boundary of the range of encoded quantities, and the other to the upper one. Codes $\mathbf{X}_q$ of interim values of numerical quantities q will be obtained by concatenation of $(1-q/Q)$ neurons from one codevector and q/Q neurons from the second codevector:

$$\mathbf{X}_q = \mathbf{X}_0\{1, N(1-q/Q)\} \vee \mathbf{X}_1\{N(1-q/Q)+1, N\}, \tag{15}$$

where $\mathbf{X}_0\{1, N(1-q/Q)\}$ is the subset of bits $\mathbf{X}_0$ with numbers from 1 to $N(1-q/Q)$; $\mathbf{X}_1\{N(1-q/Q)+1, N\}$ is the subset of bits $\mathbf{X}_1$ with numbers from $N(1-q/Q)$ to N. Characteristic of codes overlap for this procedure is shown in Figure 6.

Analytical characteristic of the relative codes overlap (1) has the form (Figure 6)

$$V(\mathbf{X}, \mathbf{X}_0) = p + (1-p)(1-x), \tag{16}$$

and of the normalized codes overlap (2) it is:

$$U(\mathbf{X}, \mathbf{X}_0) = (1-x). \tag{17}$$

If vectors $\mathbf{X}_0$ and $\mathbf{X}_1$ have no intersection, characteristic of overlap is linear.

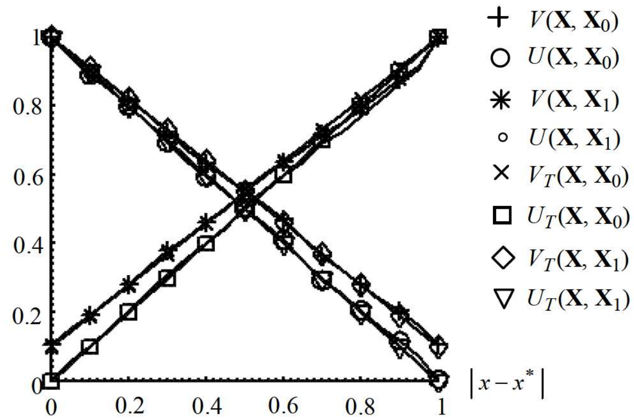

Figure 6

For obtaining a steeper overlap characteristics, let us subdivide the range of variation of the quantity into R equal subranges. Generate $R+1$ reference vectors $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_R$ corresponding to values $0, x_1, \ldots, x_r, \ldots$ $\ldots, x_R = 1$. Then the value of quantity $q = x_r + y$, $0 \le y \le x_{r+1} - x_r$ is encoded by vector $\mathbf{X}_q$, which is a concatenation of parts of neighboring codevectors $\mathbf{X}_r$ and $\mathbf{X}_{r+1}$:

$$\mathbf{X}_q = \mathbf{X}_r\{1, N - Ny/(x_{r+1}-x_r)\} \vee \mathbf{X}_{r+1}\{N - Ny/(x_{q+1}-x_q)+1, N\}, \tag{18}$$

where $\mathbf{X}_r\{1, N - Ny/(x_{r+1}-x_r)\}$ is the subset of bits $\mathbf{X}_r$ with numbers from 1 to $N - Ny/(x_{r+1}-x_r)$; $\mathbf{X}_{r+1}\{N - Ny/(x_{q+1}-x_q)+1, N\}$ is the subset of bits $\mathbf{X}_{r+1}$ with numbers from $N - Ny/(x_{r+1}-x_r)+1$ to N.

Overlap of codevectors in this scheme decreases within the subrange r. For difference of values of encoded quantities, larger, than r, the magnitude of overlap is constant and is equal to overlap of random vectors p^2N or zero by the formula of normalized overlap (2).

One can approximate bell-shaped overlap characteristic by bit-by-bit disjunction of codevectors from several "rows" of codevectors with overlap characteristics of different steepness:

$$\mathbf{X}_q = \vee \mathbf{X}_{i,q}, \tag{19}$$

where $\mathbf{X}_{i,q}$ is the vector of value q, obtained for reference vectors of the i-th row.

In this case it is evident that dimension of the final codevector does not increase. An example of experimental characteristics of relative and normalized code overlaps is represented in Figure 7.

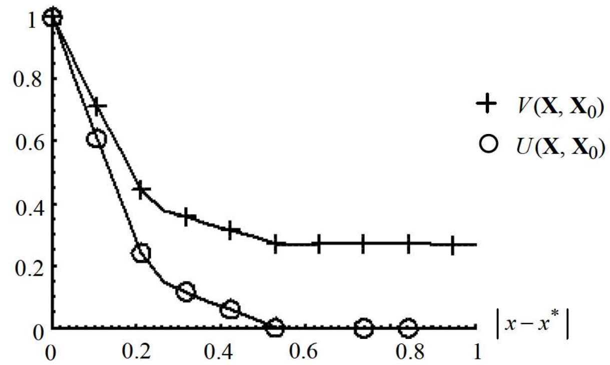

Figure 7

Estimation of computational complexity of the procedure:

1) generation of the initial and the final vector — $O(pN)$ operations;

2) concatenation of parts of reference vectors, $O((1 - q/Q)pN + q/QpN) = O(pN)$ operations for a sparse vector.

Total: $O(pN)$ operations for a sparse vector.

2.7. Procedure of stochastic encoding of cyclic quantities

Consider encoding of cyclic (periodic) quantities, whose maximal value coincides with the minimal one. Examples are degrees of an angle, time of the day etc. Overlap of codes of such quantities should reflect their proximity also in the vicinity of "discontinuity" points. For flocet encoding, the cyclic variant is evident. Let us consider also other variants of encoding procedures.

Encoding procedure "with reproduction" consists in the following. First, Q (by the number of gradations) random vectors $\mathbf{Z}_q$ with a small M/k number of unities are generated. The codevector X_q $(q = \overline{1, Q})$ is obtained by disjunction of k "preceding" codevectors

$$\mathbf{X}_q = \vee_{i=q,(q-k+Q)\mathrm{mod}Q} \mathbf{Z}_i. \tag{20}$$

Here the parameter k determines steepness of dip of code overlap characteristic, increasing k results in a more shallow curve. Characteristic of the obtained codes overlap in the decaying interval is close to linear (Figure 8).

For obtaining a bell-shaped characteristic of overlap we should generate vectors for different values of k (for instance, by the described above method, subsections 2.5, 2.6) and to unite them by disjunction (Figure 9).

Another method of obtaining cyclic codevectors is the encoding procedure "with reference codevectors". Let us consider an example of such procedure for value of degrees of an angle. Generate random sparse codevectors for reference points on the circle, corresponding, say, to 0, 90, 180, 270° (Figure 10, points A, B, C, D). Interim codevectors can be obtained by the algorithm of subsections 2.5, 2.6. Characteristic of codes overlap is shown in Figure 11, three "rows" of codevectors, 3, 5 and 6 reference vectors in each. If we need a steeper characteristic of codes overlap, we can introduce interim reference codevectors, partially correlated or not correlated at all with the original ones (see also subsections 2.5, 2.6). Examples of resulting characteristics of overlap are shown in Figure 12.

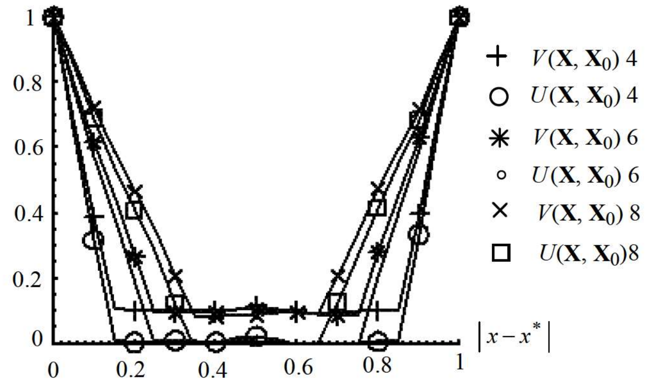

Figure 8

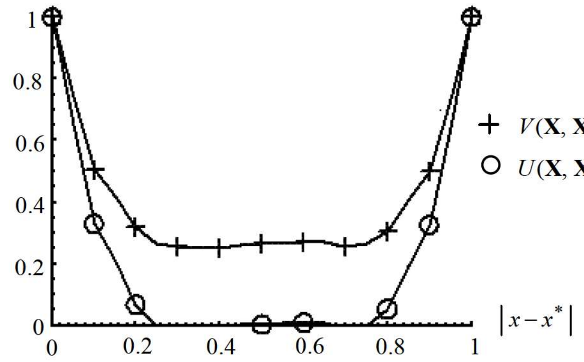

Figure 9

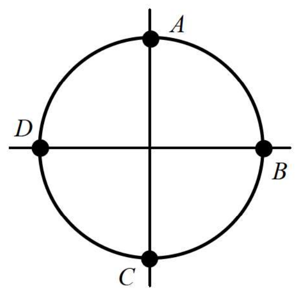

Figure 10

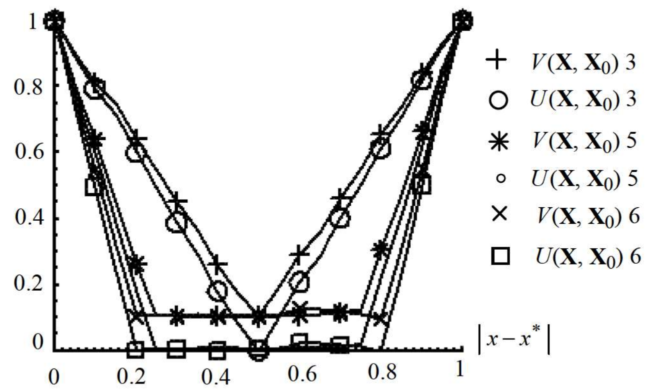

Figure 11

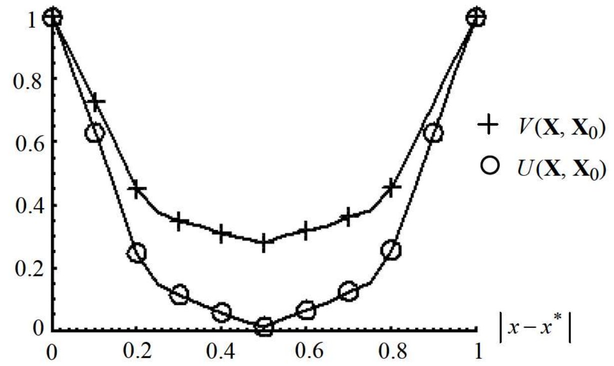

Figure 12

Conclusion

Several universal schemes of binary encoding is known. For instance, conventional binary representation of numbers and Gray codes. Their advantage consists in exponential, with respect to the bits number, code capacity. Binary code has jumps in overlap of codes of neighboring numbers. For instance, codes of neighboring numbers 3 and 4 (011 and 100) have no coinciding unities. The Gray code has no jumps in overlap, however only for neighboring numbers. For example, Gray codes of numbers 0,…, 5 are 000, 001, 011, 010, 110. Intersection of codes of neighboring numbers 1 and 2 is equal to unity, 1 and 3, to zero, 4 and 5, to two etc.

Thermometric (subsection 2.1) and flocet (subsection 2.2) encoding have no jumps of overlap characteristic. Their overlap characteristic has a piecewise-linear character. Schemes of thermometric and flocet encoding, and also their modifications, described in subsections 2.1, 2.2, are not stochastic, their capacity is linear.

"Scattered codes" [20, 22] is the only familiar to the authors scheme, which tries to overcome the mentioned problems by binding together random codevectors without increasing their dimension. Scattered codes have exponential capacity and have no jumps of overlap characteristic. However, they are not sparse ($p=0.5$) and tenuity is the most important requirement from the viewpoint of implementation efficiency and possibility of using efficient associative memory [11]. Encoding numerical vectors by scattered codes is insufficiently worked out. Just a single variant of encoding procedure by knockout-addition (an analogue of methods of the section) and binding together, by bit-by bit operation of exclusive OR (XOR) was suggested.

In this paper we considered schemes of distributed binary encoding of scalar integer quantities, suggested in the framework of papers developing APNN. Unlike schemes of rough encoding, which require performing arithmetic operations [23], the considered in this paper approaches are based mainly on bit-by-bit operations. Codevectors are either directly extracted from memory by numerical value of element, used for indexing codevectors, or are generated "in the air" from some "reference" codevectors. Operations of generation and modification of codevectors are also mainly bit-by-bit. This enables one to implement them efficiently using specialized neural computers [21].

The described schemes of stochastic encoding satisfy requirements to encoding, described in subsection 1.1, in part of encoding scalar quantities. Magnitude of codevectors overlap reflects degree of proximity (difference) of encoded quantities (numbers). We describe ways of control of steepness of overlap characteristic. It was demonstrated, that from codes of scalar quantities one can construct codes of vectors of numerical quantities. This constructing is performed on the basis of vector bit-by-bit operations over codevectors, which correspond to individual elements of the input vector.

Thus, the studied in this paper "bitwise-vector" approach to distributed encoding numerical quantities can be used for obtaining binary multidimensional codevectors with random placement of a small number of unity elements, representing points and reflecting metric of the input space in values of codes overlap.

The suggested encoding schemes, generated by them codes and their properties are perspective objects for research and application in new intelligent information technologies.

Authors are pleased in acknowledging their indebtedness to L.M. Kasatkina, A.M. Sokolov, I.S. Misuno and E.G. Revunova for fruitful discussions.

References

1. Rachkovskij D.A., Kussul E.M., Building a world model with structure-sensitive distributed representations, http://www.bbsonline.org/Preprints/Rachkovskij/Referees/Rachkovskij.pdf.
2. Rachkovskij D.A., Representation and processing of structures with binary sparse distributed codes, *IEEE Trans. on Knowledge and Data Engineering*, 2001, **2**, No. 13, 261–276.
3. Rachkovskij D.A., Kussul E.M., Binding and normalization of binary sparse distributed representations by context-dependent thinning, *Neural Comput*., 2001, **2**, No. 13, 411–452.
4. Thorpe S., Localized versus distributed representations, The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, 2003.
5. Plate T., Holographic reduced representation: distributed representation for cognitive structures, Center for the Study of Language and Information, Chicago, 2003.
6. Hinton G.E., Rumelhart D.E., McClelland J.L., Distributed representation, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 1986, 77–109.
7. Browne A., Sun R., Connectionist inference models, *Neural Networks*, 2001, **10**, No. 14, 1331–1355.
8. Hebb D.O., The organization of behavior: A neuropsychological theory, Wiley, New York, 1949.
9. Kussul E.M., Associative neuron-like structures [in Russian], Naukova dumka, Kiev, 1991.
10. Frolov A., Kartashov A., Goltsev A., Folk R., Quality and efficiency of retrieval for Willshaw-like autoassociative networks, 1. Correction, *Network*, 1995, **6**, No. 4, 513–534.
11. Frolov A.A., Rachkovskij D.A., Husek D., On information characteristics of Willshaw-like autoassociative memory, *Neural Network World*, 2002, **2**, 141–157.
12. Rachkovskij D.A., Some approaches to analogical mapping with structure sensitive distributed representations, *J. of Experiment. and Theoretic. Artificial Intelligence*, 2004, **16**, No. 3, 125–145.
13. Penz P.A., The closeness code: An integer to binary vector transformation suitable for neural network algorithms, *IEEE First Intern. Conf. on Neural Networks* (ICNN'87), IEEE, 1987, **3**, 515–522.
14. Jackel L.D., Howard R.E., Denker J.S., Hubbard W., Solla S.A., Building a hierarchy with neural networks: an example — image vector quantization, *Appl. optics*, 1987, **23**, 5081–5084.
15. Goltsev A.D., Structured neural-like network with learning, designed for texture segmentation of images, *Kibernetika i sistemnyi analiz*, 1991, No. 6, 149–161.
16. Healy M.J., Caudell T.P., Acquiring rule sets as a product of learning in a logical neural architecture, *IEEE Trans. on Neural Networks*, 1997, **3**, No. 8, 461–474.
17. Goltsev A.D., An assembly neural network for texture segmentation, *Neural Networks*, 1996, **4**, No. 9, 643–653.
18. Duda R., Hart P., Stork D., Pattern classification, John Wiley and Sons, New York, 2001.
19. Kussul E.M., Baidyk T.N., Lukovich V.V., Rachkovskij D.A., Adaptive neural network classifier with multifloat input coding, *Neuro Nimes'93*, 1993, 25–29.
20. Stanford P.H., Smith D.J., The multidimensional scatter code: A data fusion technique with exponential capacity, *Intern. Conf. on Artificial Neural Networks*, 1994, 1432–1435.
21. Gritsenko V., Misuno I., Rachkovskiy D., Revunova E., Slipchenko S., Sokolov A., Concept and architecture of program neural computer SNC, *Upravlyayushchie sistemy i mashiny*, 2004, No. 3, 3–14.
22. Kanerva P., Binary spatter-coding of ordered *K*-tuples, *Intern. Conf. on Artificial Neural Networks ICANN'96*, Springer, Bochum, Berlin, 1996, 869–873.
23. Kussul E., Rachkovskij D., Wunsch D., The random subspace coarse coding scheme for real-valued vectors, *IEEE Intern. Conf. on Neural Networks* (IJCNN'99), IEEE, 1999, **1**, 450–455.