International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:6, No:10, 2012

Inverse Sets-based Recognition of Video Clips

Alexei M. Mikhailov

***Abstract*—**The paper discusses the mathematics of pattern indexing and its applications to recognition of visual patterns that are found in video clips. It is shown that (a) pattern indexes can be represented by collections of inverted patterns, (b) solutions to pattern classification problems can be found as intersections and histograms of inverted patterns and, thus, matching of original patterns avoided.

***Keywords*—**Artificial neural cortex, computational biology, data mining, pattern recognition.

I. Introduction

THE theory of pattern recognition is featured by two paradigms that originated in late 1950s. The first paradigm can be described as probabilistic or discriminant functions-based approach and the second paradigm - as addressing-based approach that led to development of weightless neural networks. The first paradigm also relies on matching-based techniques that attempt to classify patterns either by looking for maximal values of pre-designed decision functions (or outputs of neural networks) or by directly comparing unknown patterns to a number of pre-stored prototype patterns. The problem with matching-based approaches is that computational complexity at least linearly increases with the number of learned patterns. Hence, as the pattern database size grows, so does the price of hardware, or the recognition process unacceptably slows down.

This paper discusses an indexing-based approach to pattern recognition that complies with the second paradigm. For this approach, the *N*-dimensional feature space is no longer used. Instead, it is replaced by a pattern index or a neural cortex that was modeled after the biological grey matter. A proper design of the pattern index implies that the amount of computations remains flat even if the pattern database size keeps on growing. Often a hierarchy of pattern indexes is used, where a common fundamental algorithm repeats itself on all levels.

The indexing approach to pattern recognition uses a definition of inverse sets, which the definition of a pattern index is based on. The input to the pattern index is an unknown pattern and the output is the class that the unknown pattern belongs to. If there is no output then the input is a new, previously unseen pattern. In this case a new class name is generated, which is assigned to the input pattern. Next, this pattern is registered in the pattern index.

The paper discusses the mathematics of indexing structures that emerged as a mathematical assessment of data delivered for the past century by neuroscience. We have speculated ([1], [2]) that the brain's cortex is a hierarchical biological index of real-world patterns. This viewpoint may explain the speed the brain almost instantaneously recognizes the visual, auditory, olfactory, tactile and taste patterns, despite the fact that maximum firing rate of neurons does not exceed 400 Hz, which is by 7 orders of magnitude slower than GHz frequencies of the computers. Indeed, if the brain's cortex is a pattern index then an efficient addressing system creates in a few steps a pointer to the result without resorting to lengthy calculations and massive amount of matches.

As an application example the paper discusses a real-time recognition of arbitrary gestures performed by one or two hands that are recorded by web-cameras.

A. M. Mikhailov was with Ngee Ann Polytechnic, Singapore. He is now with the Institute for Control Problems, Russian Academy of Sciences, Moscow, Russia (phone: 7903-994-4208; fax: 7495-334-90-00; e-mail: alx_mikh@yahoo.com).

II. Definition of Inverse Sets

For a collection of finite sets $\{f\}_n$, $n \in N$, a collection of inverse sets $\{n\}_f$, $f \in F$, was defined in [3] as

$$\{n\}_f = \{n : f \in \{f\}_n, n \in N\}. \tag{1}$$

This inverse collection is called a pattern index ([3]). Here N is the set of pattern names and the pattern index domain F is the union of elements of the original finite sets.

$$F = \bigcup_n \{f\}_n, n \in N$$

We are referring to elements f as pattern features and to subscripts $n \in N$ as pattern names.

A. Example 1

For a collection of sets
$z = \{a, c\}$, $y = \{b, c, d\}$, $u = \{a, b\}$, $x = \{a, d\}$, $v = \{d\}$
the collection of inverse sets, in accordance with (1), is as follows.
$a = \{z, u, x\}$, $b = \{y, u\}$, $c = \{y, z\}$, $d = \{y, x, v\}$.
These two collections can be depicted as the two following "fractions".

$$\begin{array}{ccccc} & d & & & \\ c & c & b & d & \\ a & b & a & a & d \\ \hline z & y & u & x & v \end{array} \qquad \begin{array}{cccc} x & & & v \\ u & u & y & x \\ z & y & z & y \\ \hline a & b & c & d \end{array}$$

The columns of the left-hand "fraction" contain the elements of the original sets, whereas the columns of the right-hand "fraction" contain the names of the original sets. The denominator of the left-hand "fraction" contains the names of the original sets, whereas the denominator of the right-hand "fraction" contains the names of the inverse sets. The most left column of the left-hand "fraction" comprises the first original set. The next adjacent column comprises the elements of the second set, and so on. Each column of the right-hand "fraction" comprises the elements of the corresponding inverse set. Clearly, the right-hand "fraction" represents the index of the original sets. Indeed, the first column of this fraction shows that the element *a* belongs to the sets *z*, *u*, and *x*. The second column shows that the element *b* belongs to the sets *y* and *u* and so on.

B. Example 2

Fig. 1 shows a collection of "red" circles that are denoted by an "*R*", "green" circles denoted by an "*G*", and "yellow circles" denoted by an "*Y*". The horizontal coordinates of each circle center were normalized as 1, 2, 3, 4, and 5. The circles under the dividing horizontal line show inverse sets, that is, the colors associated with each center. The number of inverse sets equals to 5 since the circles above the dividing line are spread across 5 horizontal coordinates. The inverse sets are as follows: $\{R\}_1$, $\{R, G\}_2$, $\{R, G, B\}_3$, $\{G, B\}_4$ и $\{B\}_5$. Here the subscripts represent the names (horizontal coordinates) of inverse sets, whose colors under the dividing line are denoted as "*R*" for red, "*Y*" for yellow, "*W*" for white, "*C*" for cyan and, "*B*" for blue according to the superposition of colored circles above the divider.

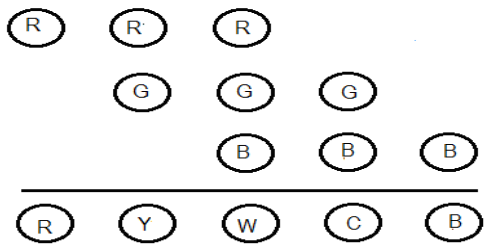

Fig. 1 Superposition of colors

A halftone color index can be defined as well. For instance, an image pixel can be associated with the colors $\{5R, 10G, 35B\}$, where the factors 5, 10, and 35 represent the brightness of the corresponding colors. In this respect the computer screen is an example of an indexing device.

III. Set Transform

The expression (1) defines a transform $\boldsymbol{I}$ that converts a collection of sets into inverse sets. Let F be a finite set, whereas $\boldsymbol{X} = \{\{f\}_n, n \in N\}$ and $\boldsymbol{Y} = \{\{g\}_m, m \in M\}$ are the two different collections of subsets from F. We denote by $\boldsymbol{X}^{1}$ and $\boldsymbol{Y}^{1}$ the two collections of inverse sets, which are built in accordance with the definition of inverse sets (1).

A. Statement 1

Any two different collections $\boldsymbol{X}$ and $\boldsymbol{Y}$ of subsets from a finite set F have two different inverse collections $\boldsymbol{X}^{1}$ и $\boldsymbol{Y}^{1}$.

This theorem implies that the transform (1) is a one-to-one mapping, that is, for each collection of sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ there exists one and only one collection of inverse sets $\boldsymbol{X}^{1}$ и $\boldsymbol{Y}^{1}$, such that if $\boldsymbol{X} \neq \boldsymbol{Y}$ then $\boldsymbol{X}^{1} \neq \boldsymbol{Y}^{1}$, or $\boldsymbol{I(X)} \neq \boldsymbol{I(Y)}$.

B. Proof

The following proof is based on the representation of a pattern index with a binary matrix, whose columns' names are the names of the features f, and whose rows' names are the pattern names n. The matrix cell (*f, n*) contains 1 if and only if the feature *f* belongs to the pattern *n*. Otherwise this cell contains 0. A general case of such binary matrix is shown in Fig. 2.

$$\frac{\begin{matrix} e_{11}, e_{12}, \dots, e_{1N} \\ e_{21}, e_{22}, \dots, e_{2N} \\ \dots\dots\dots\dots \\ e_{M1}, e_{M2}, \dots, e_{MN} \end{matrix}}{f_1, f_2, \dots, f_N} \qquad (2)$$

Fig. 2 Binary matrix representing a pattern index

Here, the element e_{mn} is equal to 1 if and only if the feature f_n belongs to the set m. Otherwise, e_{mn} is set to 0.

While the matrix (2) represent original sets $\{f\}_n$, $n \in N$, the columns of matrix (2) represent the inverse sets $\{n\}_f$, $f \in F$.

C. Example 3

Let us consider the collection of two following sets.

$$A = \{a, b\} \quad \text{и} \quad B = \{b, c\}. \qquad (3)$$

Here the lower-case characters represent the elements of the sets and upper-case characters represent these sets' names. We can represent these sets with the following binary matrix.

$$M_1 = \begin{array}{c|c} \left\| \begin{matrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{matrix} \right\| & \begin{matrix} A \\ B \end{matrix} \\ \begin{matrix} a & b & c \end{matrix} & \end{array}$$

Here, the 1s in the first row of the matrix M_1 correspond to the elements *a* and *b* of the set *A*, whereas the 0 in the row 1 indicates that the element c does not belong to the set *A*. Analogously, the 0 in the second row indicates that the element a does not belong to the set *B*. The width of the matrix equals 3 since the union $\{a, b, c\}$ of the sets *A* and *B* contains only 3 elements. Applying the definition (1) to the collection (3) we get the following inverse sets: $a = \{A\}$, $b=\{A, B\}$, and $c=\{B\}$, which we represent with the binary matrix M_2, that is,

$$M_2 = \begin{array}{cc|c} 1 & 0 & a \\ 1 & 1 & b \\ 0 & 1 & c \\ A & B & \end{array}$$

Clearly, the matrix M_2 is a transpose of the matrix M_1, that is, $M_2 = M_1^T$. While the columns of M_2 represent the inverse sets $\{n\}_f$, $f \in F$, the rows of M_1 represent the original sets $\{f\}_n$, $n \in N$. Since this is true in a general case of N-dimensional matrix, the transform ***I***, which is defined by expression (1), amounts to interchanging the rows and columns of the matrix (2). Obviously, it would be impossible to arrive at the same matrix by subjecting two different matrices to a transpose operation. Hence, the transform ***I*** is a one-to-one mapping.

Note 1. The above considered binary matrices (index matrices) constitute a special case of matrices that are treated in the Formal Concept Analysis (FCA). This is because the index matrices, unlike FCA-matrices, feature unique rows. Indeed, if we suppose that an index matrix can have two or more identical rows then the implication would be that the collection of original sets comprises two or more identical elements (sets). But this contradicts to fundamental definition of a set.

Note 2. Representation of collections of sets by index matrices is not efficient from a computational point of view as the 1s of index matrices are sparsely distributed. However, representation of sets by index matrices can prove to be useful when it comes to set comparison operations, which are needed in case a distance between sets needs to be calculated. This issue will be considered in Section VI.

IV. Nearest Neighbor Approach to Pattern Recognition

A popular in practice, nonlinear classifier makes use of a nearest neighbor (NN) rule. According to this rule, at the reception of an unknown feature vector, its distance from all training vectors (of the various classes) is computed. Next, the unknown vector is assigned to the class of its nearest neighbor.

If the number of training vectors is N and the vector length is M then the amount of computations is on the order of $O(NM)$ and it keeps on increasing linearly with N. This paper discusses a way of keeping the amount of computations flat even if the number of training vectors grows. Since we treat the inputs as dimension-less sets, we have to rephrase the NN-problem in terms of sets as the following search problem.

A. Set Search Problem

Let F_n, $n \in N$, be a finite collection of finite sets. For a given finite set U find the set F_m, such that

$$D(F_m, U) = \min_{n \in N} D(F_n, U) \qquad (4)$$

Here $D(F_m, U)$ is a distance between sets F_m and U.

If the unknown input pattern U is identical to one of prototype patterns F_m, $m \in N$, then the right-hand side of the equation (4) is equal to 0

$$D(F_m, U) = 0 \qquad (5)$$

B. Statement 2

A necessary condition for a pattern, whose name is m, to be a solution to (5) is that the name m belongs to the intersection of inverse sets

$$m \in \bigcap_{f \in \{u\}} \{n\}_f \qquad (6)$$

Here $\{n\}_f$, $f \in F$, are the inverse prototype sets (patterns) and $f = \bigcup_{n \in N} \{f_n\}$.

C. Proof

Let inverse sets $\{n\}_f$, $f \in \{u\}$, and only these sets contain the name m that is a solution to (5). Then the name m belongs to the intersection (6). If the name m does not belong to the intersection (6) then m does not belong to all inverse sets $\{n\}_f$, $f \in U$, and $F_m \neq U$. Hence, the necessary condition for m to be a solution to (12) is that m belongs to the intersection (6). However, (6) is not a sufficient condition. Indeed, should there exist a superset W such that $U \subset W$ then the name w of the set W also belongs to the intersection (6).

D. Pattern Histogram

If the intersection on the right-hand side of (6) is empty then we are back to the problem of minimizing (4). However, even if (6) is empty, a full search can be avoided by using a pattern histogram. The pattern histogram $H_U(n)$ is a histogram of names that are contained in inverse sets $\{n\}_f$, $f \in U$. The following statement is true.

E. Statement 3

The height of the nth sample of the pattern histogram is equal to the number of times the name n occurs in inverse sets $\{n\}_f$, $f \in U$.

Indeed. Since every inverse set contains no more that one copy of each name, the total number of copies of the name n in inverse sets $\{n\}_f$, $f \in U$, does not exceed the size of the set U: $H_U(n) \leq |U|$. If the name n is found only in a strict subset $V \subset U$ then $H_U(n) < |U|$.

F. Computational Aspects

Statement 3 constitutes a basis of an efficient search algorithm that delivers a solution to the set search problem (4) by calculating $H_U(n)$-histogram. The histogram is calculated by accessing only the inverse sets $\{n\}_f$, $f \in U$. The rest of the inverse sets, that is, the sets with the addresses $F \setminus U$ are ignored. As a result, only a small subset of histogram samples is calculated. The dependence of the histogram on the input pattern U is marked by way of using a histogram's subscript U: $H_U(n)$. The solution m to (4) is obtained as

$$m : H_U(m) = \max_{n \in N} H_U(n) \qquad (7)$$

$H_U(n)$-based search algorithm is computationally efficient if the following two conditions are satisfied.

(a) $|U| << N$ (size $|U|$ of the pattern U is much less than the total number N of prototype patterns).

(b) $|\{n\}_f| \approx |\{f\}_n|$ (average size of inverse patterns (sets) is roughly equal to the average size of prototype patterns (sets).

If condition (a) and (b) hold then the search amounts to about $|U| * |\{n\}_f|$ arithmetical operations that are needed to calculate the histogram plus $|U| * |\{n\}_f|$ operation that are needed to find the histogram's maximum. This amount of computations is much less than $|N| * |\{f\}_n|$, which is the number of operations that are needed in case a full search is employed, that is, an unknown pattern is matched to all prototype patterns.

Note that the condition (a) is always met in applications with large numbers of prototypes and the condition (b) can be satisfied by feature aggregation. For instance, it can be shown that by aggregating x and y features as $x + y* L$, where L is the dynamic range of original features, the value of $|\{n\}_f|$ decreases, on average, up to $|\{n\}_f| / L$. If the features x, y и z are aggregated as $x + y* L + z* L^2$ then the value of $|\{n\}_f|$ will be, on average, $|\{n\}_f| / L^2$ etc.

By using a normalized histogram $H_U(n) / |U|$ the distance $D(F_m,U)$ between unknown and prototype patterns can be reduced to [0, 1] interval, where 0 stands for pattern identity and 1 – for the absence of common elements between F_m- and U- patterns.

In practical applications a recognition threshold $T < 1$ is used, such that if $D(F_n, U) < T$ then the pattern F_n is considered to be a candidate to quality as a nearest neighbor. The final decision is made after substituting the candidates into (4) and selecting a minimum distance candidate.

V. Recognition of Video Clips

This section deals with recognition of video clips scenarios. For instance, the videos can show flying birds, dancing pairs, hand gestures, etc. Here we focus on arbitrary hand gestures. For example, hands can orbit an imaginary ball, change their shape and show a different number of fingers. The system comprises a preprocessing module that analyses frame sequences delivered by a 2D-web-camera and extracts the contour of moving objects. For contour extraction two successive frames are superimposed and a difference image is calculated, which is represented by an unordered list of pixel coordinates. An (x, y)-pixel qualifies for the list if its red component brightness frame difference exceeds a threshold. Although a 3D-camera would provide a better performance, the chosen 2D-inverse sets-based approach still proved to be applicable even in case of fuzzy and broken contours.

The system uses three pattern indexes. First level pattern index recognizes local features of hand contours. The local features are obtained from the unordered difference pixel list in the following way. The center of an S x S-window (for instance, $S = 16$) is placed in turn at a number of selected contour pixel, such that the contour is covered at a chosen step.

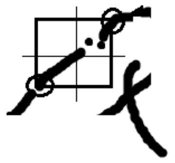

Fig. 3 Window frame against the contour

The positions of pixels along the window frame are sequentially numbered from 0 to $4*S$-1. In Fig. 3 the intersections of the window frame and the contour are marked with two circles. The 1s in the binary sequence show the positions of intersections in terms of frame pixel numbers. For instance, starting the count from the upper-left corner we may get the following sequence 0000001100000000000101000000, which represents one local feature of the given contour that comprises two parts shown with fat curves.

The first level index is described with the index pair $\{p\}_f$, $\{f\}_p$, where f is the local feature name and p is the position of 1-valued bit in this feature. The positions p of 1-valued bits are determined by the intersections. An example of two inverse sets, that is, the pattern index, which contains the copies of two names, is shown below.

```
 2           2          2
 11  2      1 1       1 2 1
___________________________
0110000000101000000000001      (1st feature)
0100100000001000000001001      (2nd feature)
012345…                        (coordinates of columns)
```

The numbers above the dividing line represent the features' names in the corresponding columns $\{f\}_p$, whose addresses are determined by the positions of 1s in binary representations of local features. When an unknown feature uf arrives, the columns $\{f\}_u$ are accessed, so that the number of feature name occurrences is calculated. As a result a histogram $H_{uf}(f)$ is created. The values of the histogram subscripts f, where $H_{uf}(f)$ exceeds a recognition threshold T_f, are used as names of short-listed candidate features, which we denote as CF. Finally, candidate features CF are in turn compared to the unknown input uf with the Soft Hamming distance $D_{SH}(f, uf)$ (distance measure (9), Section VI) in order to minimize the expression

$$D_{SH}(f, uf) \rightarrow \min_{f \in CF}$$

The unknown local feature uf is assigned the name of the winning prototype feature. However, if the candidate set is empty then the unknown feature is considered to be a new one and is assigned a name by incrementing the value of the last issued name by 1 because all names are integers. For the above discussed two features example the value of the name would be 3. Next, the copies of this name are written in the columns of the local feature index $\{f\}_p$. The number of copies equals the number of 1s in the binary representation of the new local feature. The addresses of the columns that are to be updated are determined by the positions of 1s in the binary representation of this feature.

The number of feature classes depends on the feature recognition threshold T_f. If $T_f = 1$ then there exist only one feature class. In the other extreme case at $T_f = 0$, the number of classes is equal to the number of different features, that is, any tiny variation of the feature causes the registration of a new feature class. For the given hand gestures recognition application it was experimentally shown that at $T_f = 0.5$ the number of registered feature classes amounts to 29.

The same indexing algorithm repeats itself on the levels 2 and 3. On the 2nd level the index pair $\{f\}_c$, $\{c\}_f$ is created, where c is the name of a hand(s) contour. Coordinates of the contours' index $\{c\}_f$ are the names of local features that represent the contour c. If the local feature f is found more than one time in a contour c then the number of repeats is ignored. Next, the histogram $H_{uc}(c)$ is created, where uc stands for the unknown contour, and the contour recognition threshold T_c is set. Using T_c candidate contours CC are selected. Finally, the set distance (distance measure (11), Section VI)

$$D_T(c, uc) \to \min_{c \in CC}$$

is minimized and the winning contour obtained. For this application it was experimentally shown that the number of contour classes is 99 at $T_c = 0.4$.

On the 3rd level each gesture g is described by a sequence of contours $\{c\}$ and approximately represented by an unordered set $\{s\}$ (2nd order alphabet, Section VI). Hence, the index pair $\{s\}_g$, $\{g\}_s$ is created, where g is the name of a gesture. Coordinates of the gesture index $\{g\}_s$ are the names of the characters of the 2nd order alphabet. When an unknown gesture ug arrives, the columns $\{g\}_s$ and the number of gesture name occurrences are calculated. Once the gesture histogram $H_{ug}(g)$ was created, where ug stands for unknown gesture, and candidate gestures CG short-listed the weighted set distance (distance measure (12), Section VI).

$$D_T(g, ug) \to \min_{g \in C}$$

is minimized and the winning gesture obtained.

External (given) gesture classes are fixed before the learning/recognition process starts. The number of external classes is that of the number of gestures to be learned. Normally, the system automatically creates on all levels sets of internal classes, whose number may be greater then the number of external classes. As a result, a correspondence function *COR* needs to be introduced: external class = *COR* (internal class).

Three video clips (3 different gestures) were recorded that are shown on Fig. 1-6. Each clip contains from 400 to 600 frames. At 20 frames per second rate the duration of each clip is about 25 seconds. Because of the limited page size each picture shows only a few fragments of each gesture. Each clip comprises 6 repeats of the same gesture. The repeats are separates by 1 to 2 seconds intervals. Since a human can not exactly reproduce his/her gestures, the first 3 gestures were used for training and the remaining once – for testing the quality of the recognition process. It was found that the system was capable to correctly identify the 3 remaining gesture samples. The following recognition thresholds were used: $T_f = 0.5$, $T_c = 0.4$, $T_g = 0.5$. As a result, 29, 99, and 17 internal classes were automatically created on the levels 1, 2, and 3, correspondingly.

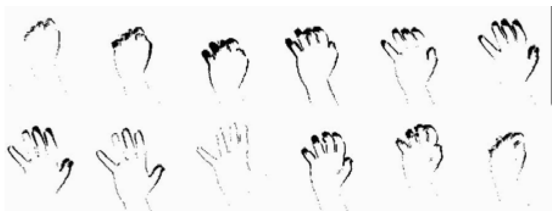
Fig. 4 Gesture 1: two rows of selected samples of the changing hand shape (from left to right)

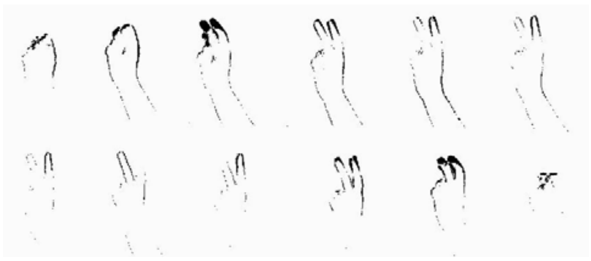
Fig. 5 Gesture 2: two rows of selected samples of the changing hand shape (from left to right)

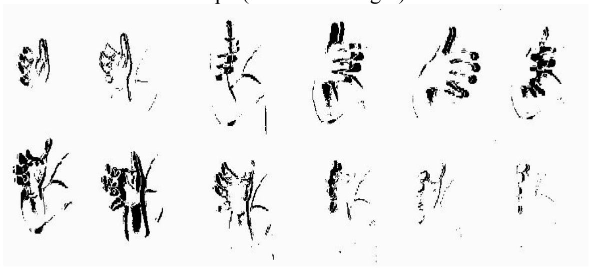
Fig. 6 Gesture 3: two rows of selected samples of the changing hand shape (from left to right)

VI. Distance Measures

A. Representation of Sets with Binary Sequences

A collection of sets can be represented with binary sequences. For instance, the sets $\{a, b, z\}$ and $\{b, r\}$ can be represented with the sequences 1101 and 0110.

B. Soft Hamming Distance

This distance measure can be applied, for instance, to collections of discrete dots on an axis. Below are given 3 binary sequences, whose 1-valued bits represent 3 collections of dots' positions on an integer axis.

$A = 1000110000100100001$
$B = 0100001001000010010$
$C = 1000010000100100001$,

If we make use of Hamming distance

$$D_H(A,B) = \sum_n (a_n XOR b_n) \tag{8}$$

then D_H $(A, B) = 11$ and D_H $(A, C) = 1$. However, the distribution patterns of 1s in B and C are quite similar. This similarity can be accounted for with the following measure, which we refer to as a Soft Hamming distance.

$$D_P(A,B) = \sum_n (\underset{k}{\&}(a_n XOR b_{n-k})) \tag{9}$$

where $|k| \leq R$, $n = 1, 2, \ldots, N$ and R is a given surrounding region. For the above example, if R= 1 then D_P $(A, B) = 0$ и $D_P(A, C) = 0$. It means that a shift of dots within the surrounding region does not affect the positional distance.

C. Distance between Sets

In accordance with the remark ((*A*), Section VI) the well-known Tanimoto distance between sets

$$D_T(A,B) = |A \Delta B| / |A \bigcup B| \tag{10}$$

where $A \Delta B = (A - B) \cup (B - A)$ is the symmetrical difference between sets A and B, can be calculated as

$$D_T(A,B) = \sum_n (a_n XOR b_n) / \sum_n (a_n OR b_n) \tag{11}$$

Note that the order of set characters is irrelevant since sets are unordered objects.

In case of weighted sets, where weights are represented with non-negative integers a_n, b_n, …, the distance (11) can be rewritten as

$$D_{TW}(A,B) = \sum_n \min(a_n, b_n) / \sum_n \max(a_n, b_n) \tag{12}$$

D. Distance between Sequences

The distance measures (8)-(12) cannot be applied to sequences, that is, to ordered sets. For instance, if the measure (10) is used to estimate the distance between the sequences *abc* and *bac* then we get a 0-distance as the symmetrical difference between sets {*a, b, c*} and {*b, a, c*} is an empty set. Normally, an Edit distance is used for matching of sequences.

The Edit distance takes into account the number of deletions, insertions, and substitutions required to change one sequence into another. For the sake of consistency, we use an extended alphabet-based approach, which allows to approximating sequences with sets. The original alphabet *P* is replaced with an alphabet that comprises character pairs, character triplets, etc. For instance, for an original three character alphabet (*a*, *b*, *c*) the second order alphabet *PP* comprises the characters *A = aa, B = ab, C = ac, D = ba, E = bb, F = bc, G = ca, H = cb*, and *I = cc*. In this case the sequences ***A*** = *abcb* and ***B*** = *bcab* are represented as *BFH* and *FGB*, correspondingly. The 3rd order alphabet *PPP* comprises $|P|^3$ characters and so on. The discussion as to how an *n*th order alphabet approximates the Edit distance is beyond the scope of this paper. Our experiments show that the 3rd order alphabet allows distinguishing all words in English language by approximating the words with sets (in general case with weighted sets). For example, if we apply the 2nd order approximation to the sequences ***A*** and ***B*** then we find that these sequences are different even though the original sets {*a, b, c, b* } and {*b, c, a, b*} are identical. Indeed, the distance (10) amounts to $D_T(\boldsymbol{A}, \boldsymbol{B}) = |\{B, F, H\} \Delta \{F,G, B\}| / |\{B, F, H\} \cup \{F,G, B\}| = 2/4$.

References

[1] A. Mikhailov, and Y. M. Pok, "Artificial neural cortex," in *Proc. of Artificial Neural Networks in Engineering Conf.*, vol. 11, St. Louis, 2001, pp. 113–120.
[2] A. Mikhailov, and Y. M. Pok, "How does the Grey Matter Work?" in *Proc. of International Congress on Biological and Me4dical Engineering, December 2002, Singapore.*
[3] A. Mikhailov, "Biologically Inspired Artificial Neural Cortex and its Formalism", World Academy of Engineering, Science and Technology, vol.56 (2009), pp. 121-125.