The N-tuple Subspace Classifier: Extensions and Survey

Robert M. Haralick (Life Fellow IEEE) and Ahmet Cem Yuksel
Graduate Center, City University of New York

This paper is written in recognition of W. W. Bledsoe, who with Browning, introduced the N-tuple subspace classifier in 1959. This 1959 paper was the first paper to introduce subspace classifiers and the sum rule to combine the outputs of the classifiers.

A mathematical notation is given to easily express in a precise and unambiguous way everything going on in the N-tuple subspace classifier. Extensions of the N-tuple method are discussed using a generalized product expression and we relate the generalization to graphical models. We discuss the sum rule, the product rule and the plurality voting rule for combining the scores of the subspace classifiers.

We selected a representative sample of papers that the 1959 N-tuple subspace classifier inspired. Some of the papers introduced specialized improvements. Many of the papers showed the value of the N-tuple subspace classifier in all kinds of applications and compared the results of one or more varieties of the N-tuple subspace classifier with other state of the art classifiers. Their experiments showed that the N-tuple subspace classifier was competitive with the state of the art classifiers and often had a higher accuracy. Finally we highlight some papers that describe experiments of the N-tuple subspace classifier executing in a quantum computer.

I. Introduction

This paper is a tribute to Woodrow Wilson Bledsoe (1921-1995). He worked in a variety of areas including automatic theorem proving, character recognition, face recognition, artificial intelligence and much more. He was very active in the Artificial Intelligence community and served on the International Joint Conference on Artificial Intelligence (IJCAI) board of trustees from 1978-1983. In 1983 he was elected president of the American Association For Artificial Intelligence. A memorial to him was written by Michael Ballantyne, Robert Boyer, and Larry Hines [1]. This memorial was made into a resolution passed by the faculty at the University of Texas, Austin.

Here we concentrate only on the N-tuple subspace classifier introduced by Bledsoe and Browning in 1959 [2] and many of the papers that this 1959 paper inspired in the following 61 years. Their paper stands out for two reasons: it is the first and main paper, as far as we know, to introduce the idea of subspace classifiers and the way of combining subspace classifiers using the sum rule.

The N-tuple subspace classifier was originally designed for the recognition of hand-written characters. It is now just over 60 years from 1959 and we might ask, why is this relevant today? We have a variety of subspace classifiers, random forests, neural networks, deep learning and many more methods available, too many more to survey or even mention in a paper devoted to what now may be termed an ancient method. The N-tuple method and its extensions is appropriate for problems even with 1000 dimensions and 100 classes for which it or its extensions may be the most economical on-line classification method that trades off computational complexity with fast memory, the cost of which has become cheaper and cheaper over the last decades.

In 1960 the cost of a megabyte of DIMM memory was around 5×10^6 dollars per megabyte. This motivated researchers to find N-tuple memory reduction strategies such as Ullmann [3]. In 2020, the cost of a megabyte of DIMM memory is about $\$.0035$. It is this fact that makes the N-tuple classifier, which can be memory intensive, appropriate for training sets having well over 10^7 tuples and for classifying streaming big data at rates of over 50MB per second with electronics packaged in a relatively small volume.

There are multiple names by which the N-tuple classifier and its variants have been called. For example, there are RAM based neural networks [4] [5], single layer lookup perceptions [6] and weightless neural networks [7].

We discusss how the N-tuple subspace classifier works, motivate why it works and describe some extensions to make it more powerful. The Bledsoe and Browning subspace classifier easily fits into the decomposition mode of how a complex problem can be solved exactly or approximately in Computer Science. Most commonly there are the recursive decompositions, data decompositions, functional decompositions or search space decompositions. The Bledsoe and Browning subspace classifier is mostly in the mode of functional decomposition and the classification problem can often be solved nearly optimally.

The idea is that the problem is broken into pieces. The pieces are designed so that the dependencies are maximized within each of the pieces and the different pieces are nearly independent of each other. Each piece is solved optimally or nearly optimally. Then they are threaded together to form a solution.

II. Notation

In this section we will detail our notational conventions that will facilitate our discussion of the inner workings of the N-tuple method. Our notation will be the standard set builder notation. $\{x \in \mathcal{M} \mid P(x) > \alpha\}$ specifies a subset of $\mathcal{M}$ consisting of those elements having probability $P(x)$ greater than α. $\langle x_1, \ldots, x_Z \mid x_z \in \mathcal{M}\rangle$ means a sequence of Z tuples each one being a member of $\mathcal{M}$. Since it is a sequence, some elements from $\mathcal{M}$ may not be present in the sequence and some tuples of the sequence may occur more than once.

Each feature variable is associated with a unique index. If there are N feature variables, then the index set I associated with the N feature variables is given by $I = \{1, \ldots, N\}$. In general, an index set is an ordered set of natural numbers. For example an index set can be $J = \{5, 7, 8, 10\}$. Each index in an index set serves as an index to the features associated with them.

The feature whose index is j takes its values from the range set L_j. Measurement space, designated as $\mathcal{M}$, is the set of all possible measurements; $\mathcal{M} = \times_{i \in I} L_i$ All of the sets we use and define are finite sets. For any set A, $|A|$ is the number of elements in the set A. We define an Indexed List Relation as a list of tuples all from the same subspace of measurement space. The index set keeps track of which components of a tuple correspond to which features.

Definition 1: Let feature j take its value from range set L_j. Let I be the index set for all features. Then $\mathcal{U}$ is a Subspace of Measurement space $\mathcal{M}$ if and only if for some $J \subset I$, $\mathcal{U} = \times_{j \in J} L_j$.
For the N-tuple world, subspaces are axis parallel subspaces and the projections to the subspaces are orthogonal projections.

Definition 2: An Indexed List Relation (J, R) is a pair whose first component, J, is an index set and whose second component $R = \langle x_1, x_2, \dots x_Z \rangle$ is a sequence of tuples where x_z is a tuple having $|J|$ components and $x_z \in \times_{j \in J} L_j$.

For example if $J = \{1, 4, 5, 8, 9\}$, then $x_z = (x_{z1}, x_{z2}, \dots, x_{z5})$ is a tuple having 5 components. The first component is associated with index 1 and takes a value from range set L_1, the second component is associated with index 4 and takes a value from range set L_4 and so on.

The last notation we need is for projection. An indexed list relation (I,R) may be projected to a subspace defined by the set J, $J \subset I$. We denote the projection operator onto the subspace indexed by J by π_J.

Definition 3: Let (I, R) be an indexed list relation. and $J \subset I$. Then the Projection from the subspace indexed by I to the subspace indexed by $J = \{j_1, j_2, \dots, j_{|J|}\}$ is defined by.

$$\pi_J(I, R) = (J, S)$$

where $(x_1, \dots x_{|J|}) \in S$ if and only for some tuple $(y_1, \dots, y_{|I|}) \in R$ $x_1 = y_{j_1}$, $x_2 = y_{j_2}, \dots, x_{|J|} = y_{j_{|J|}}$.

For example if $I = \{1, 4, 5, 8, 9\}$ and $J = \{4, 9\}$ and a tuple $(x_1, x_2, x_3, x_4, x_5) \in R$ then the tuple $(y_1, y_2) \in S$, where $y_1 = x_4$ and $y_2 = x_9$. We use the standard notation that $[1, M] = \{1, \dots, M\}$.

III. Bledsoe and Browning's N-tuple Subspace Classifier

First we explain what we mean by a subspace classifier.

Definition 4: A Subspace Classifier is one that projects the measurement tuple to multiple subspaces where each projected tuple is processed. Then the processed projected tuples are combined in a way to form an assigned classification for the measurement space tuple.

Bledsoe and Browning worked with characters. Their characters were the digits 0 through 9 and the letters A through Z, a total of 36 classes. Each character was positioned in a 15 row and 10 column binary image, thereby making a 150-dimensional measurement tuple. They normalized their characters by doing small translations and rotations of the original character to put it in the upper left hand corner of the retina.

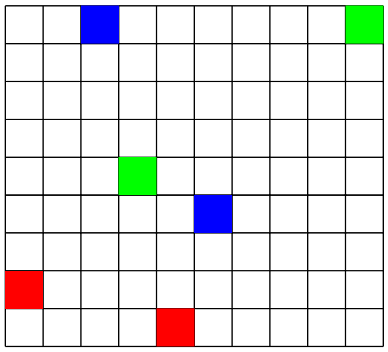
Fig. 1. Shows an example 9 row by 10 column retina in which the character is located. Each pair of cells shown in the same color are selected by one of the mutually exclusive sized 2 index sets.

Bledsoe and Browning did not use elaborate features calculated from the observed measurement tuples. In their initial experiments, their measurement tuples were 150 dimensions. They selected at random 75 subspaces, each 2-dimensional. Since they worked with a binary image they picked out 75 mutually exclusive pairs of pixel positions, each pair of positions having a pair of binary values based on what the pattern on the retina was. Each pair of binary numbers addresses a memory of 4 addresses. There were 75 such 4 address memories for each class. The word size of the memory they used was 36 bits. So they used one bit for each of the 36 character classes. Each character class had its own 75 4 address memories. The total memory they had was $75 \times 4 \times 36 = 10,800$ bits for their initial experiments. The value of each two feature pair was used to form an address. Note that whether the feature values are numeric or symbolic the N-tuple method would work and it works identically the same regardless of the $1-1$ function from the numeric or symbolic tuples to the addresses.

Their first results on typewriter fonts was near perfect accuracy so they went on to do more difficult classification problems, like reading block print characters or handwritten cursive characters. As they went through the training set, the value of each 2 feature pair would be converted an address Its class tag would select one of the 36 memories. Each memory would have 75 associated memories of 4 addresses each. They would then work with the 75 memories associated with the class tag of the tuple to be classified. Each of the 75 two feature pairs would form an address in its associated memory. The addressed bit associated with the class tag and the address of the feature pair would be set to 1 for each of the 4 address 75 memories. When a new character was scanned and thresholded, they accessed the 75 4 address memories, associated with each class and counted how many ones each class had over its projected measurement tuples. The sum was the score for that class. Then they classified the measurement tuple with the character class that had the largest score.

A. The Mathematical Description

Now we will mathematically describe Bledsoe and Browning's original N-tuple subspace method. The notation used here was first given by Haralick [8].

Let the tuple size be N. This was 150 in the case of Bledsoe and Browning, see Figure 1. The full index set I is $I = \{1, \ldots, N\}$. We designate the range set for the i^{th} component of a measurement tuple to be L_i, $i \in I$. Each measurement tuple $x \in \times_{i \in I} L_i = \mathcal{M}$, the measurement space. We designate the set C of K classes by $C = [1, K]$ and designate the training set by $\{\langle x_1, \ldots x_Z \mid x_z \in \mathcal{M}\rangle, \langle c_1, \ldots, c_Z \mid c_z \in [1, K]\rangle\}$. Let the M index sets specifying the subspaces be $\mathcal{J} = \{J_1, \ldots, J_M\}$. Bledsoe and Browning first worked with $\mathcal{J}$ being a partition of I, with each cell of the partition being approximately the same size. To construct a random partition, they determined a random permutation $i_1, \ldots, i_N$ of $1, \ldots, N$. For simplicity assume that $V = N/M$, V an integer. In this case the size $|J_m|$ of each J_m index set is V. Then they took the first V elements of the permutation to form the set J_1, the next V elements of the permutation to form the second set J_2 and so on. the Bledsoe and Browning called each index set J_m an N-tuple since it was indexing components of the measurement tuples and the components they indexed constituted a pattern.

Let the training set be $\{\langle x_1, \ldots, x_Z\rangle, \langle c_1, \ldots, c_Z\rangle\}$ Based on the training set, tables are formed, one binary valued table for each $m \in [1, M]$ and each class $k \in C$: $T_m : (\times_{j \in J_m} L_j) \times C \to \{0, 1\}$ is defined by

$$T_m(u \mid k) = \begin{cases} 1 \text{ if for some } z \in [1, Z], \pi_{J_m}(x_z) = u \\ \qquad\qquad\qquad\qquad \text{and } c_z = k \\ 0 \text{ otherwise} \end{cases}$$

As the notation indicates, $T_m(\pi_{J_m}(x) \mid k)$ is given the value 1 if the estimate of the class conditional probability $\hat{P}_m(\pi_{J_m}(x) \mid k) > 0$. Otherwise, it takes the value 0. This could be generalized by assigning the value 1 if $\hat{P}_m(\pi_{J_m}(x) \mid k) > \alpha$, for a pre-specified α. That was not an option that Bledsoe and Browning tried. It was tried when the bleaching threshold was introduced. But that is a topic we will cover later.

To assign a new measurement tuple to a class, they used the T_{mk} tables to define a score S_k for each class k.

$$S_k : \mathcal{M} \to \mathbb{R}$$

The score function is defined by

$$S_k(x) = \sum_{m=1}^{M} T_m(\pi_{J_m}(x) \mid k) \qquad (1)$$

The measurement tuple $x \in \mathcal{M}$ is assigned to class $k \in C$ if $S_k(x) > S_j(x), j \in C - \{k\}$. Saying this another way,

$$k = argmax_{j \in C} S_j(x)$$

If the maximizing score is not unique, the assignment was chosen with equal probability among the classes that maximize the score.

Haralick [9], knowing about the Bledsoe and Browning paper, described a table look-up classifier used for crop or land use identification in Satellite imagery, that made a more restrictive suggestion: Assign x to class c if $S_c(x) = M$, where S_c is calculated using the sum rule. If c is not unique assign x to reserve decision.

It is interesting that Bledsoe and Browning chose a way of combining the class k conditional scores, $T_m(\pi_{J_m}(x) \mid k), m \in [1, M]$ by a sum. It was not until the mid 1990's that exploration of ways of combining different classifiers was undertaken. [10], [11]. Interesting enough Kittler [12], more than three decades later, did a study that experimentally showed that the Bledsoe and Browning sum method as given in Equation (1) outperformed other combining methods. Bledsoe and Browning anticipated the combination of an ensemble of classifiers by the sum rule.

Many of the N-tuple papers use an engineering block diagram to show the structure of the N-tuple subspace classifier. One such a block diagram is shown in Figure 2.

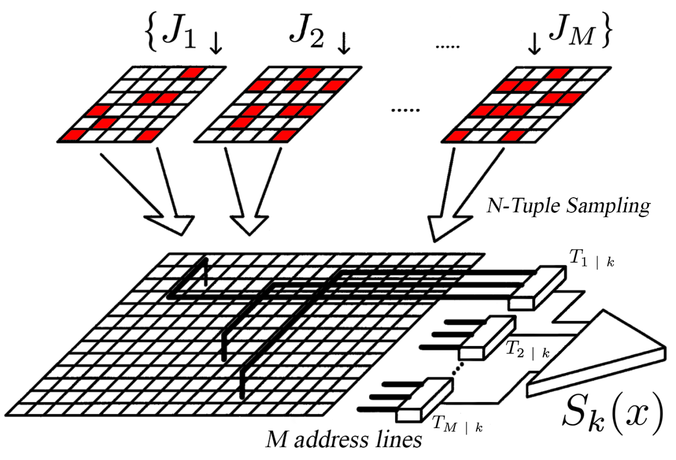

Fig. 2. Shows one of the block diagrams that papers have used to represent the N-tuple neural networks. This figure is adapted from [13] [14].

Figure 3 shows another block diagram often shown with the commercial implementation of the N-tuple method called WiSARD (**Wi**lkie, **S**tonham and **A**leksander's **R**ecognition **D**evice) [15]. For each given class k, they use the term discriminator to refer to the $T_{1k}, \ldots, T_{Mk}$ tables and its score function. The dashed line from the discriminator to the top and bottom of the grid, represent the values in the selected pixel locations corresponding to the M index sets $J_1 \ldots, J_M$. Axis d represents the 10 comparisons done among the 10 discriminators and the S represents the class score for each class. The digit discrimination is done by selecting the discriminator output that has the largest score [16].

This is the argument than Bledsoe and Browning made as to why the method has the possibility of working. Consider the letter I.

> The very shape of a character, such as the letter I, forbids certain states $(\pi_{J_m}(x))$ for certain pairs (for certain J_m). The existence of these forbidden states lies at the heart of our method, for without them the logic would saturate (all the M projections from

measurements x of one character class would be the same for a different character class) [2] page 227.

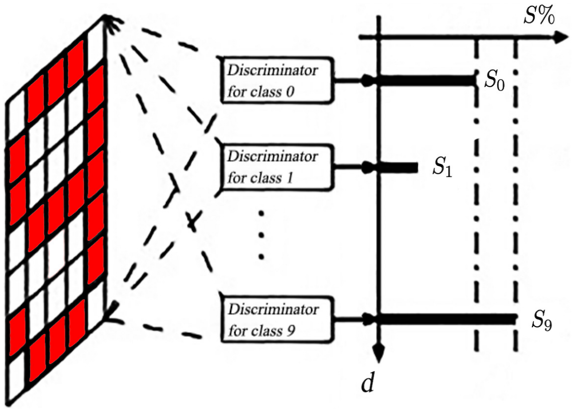
S%
Discriminator for class 0
Discriminator for class 1
Discriminator for class 9
S_0
S_1
S_9
d

Fig. 3. Shows the block diagram often used by the commercial implementation of the N-tuple method called WiSARD (Wilkie, Stonham and Aleksander's Recognition Device) [15]. The recognition is for the digits 0 to 9 . Notice that 9 had the largest score and that 0 had the next largest score. The normalized difference $\frac{S_9 - S_0}{S_9}$ may be used to determine if the class having the next to highest score is too close to the highest score.

Because the translation and rotation of the character on the photocell grid could vary. They normalized by translating and rotating the character to the top left corner of the photocell grid. It was not clear from the paper how they decided to locate the center of the rotation.

During the course of their experiments, they worked with typewriter fonts, handwritten block printed characters and handwritten cursive script characters. With typewriter fonts the identification accuracy was perfect. To make it challenging, they purposely worked with hand written block printed and handwritten cursive script.

The identification accuracy was not high with the handwritten cursive script characters, principally because the resolution of their retina, which was constructed by an array of 15 rows by 10 columns of photocells, was too coarse and the centering, positioning and rotation of the written script was too varied. However with the machine printed digits they did achieve some success although it was not near perfect. Table I shows two of Bledsoe and Browning's results. Although the paper does not explain the difference between the two results shown, it is probably the case that they corresponded to two different collections of index sets.

The Bledsoe and Browning results are shown in Table I, using the original method having entries that were 0 or 1 in the T_{mk} tables. Column 1 shows the accuracy without positioning and rotation, column 2 for the case of positioning and no rotating and column 3 for the case of positioning and rotating. The results indicate that when both positioning and rotating are done, the identification accuracy is highest: 89% to 90%.

TABLE I
SHOWS THREE RESULTS FOR HANDPRINTED BLOCK CHARACTERS FOR THE DIFFERENT COMBINATIONS OF POSITIONING AND ROTATING .

Initial	Positioned	Positioned and Rotated
80	84	89
72	88	90

As well, they worked with different sizes of the $J_1, \ldots, J_M$ index sets: $|J_m| \in \{1,2,3,5,8\}$; different numbers of index sets $M \in \{30, 50, 75, 150, 128, 256, 512, 1024\}$ and different number of classes: $|C| \in \{10, 36\}$.

Bledsoe and Browning tried a different related N-tuple scheme that they called averaging, but it is actually more like a correlation scheme. In this method for each class, all the training sequence measurements are averaged producing for class k a class mean tuple μ_k. Let the training sequence for class k be $< x_1^k, x_2^k, \ldots, x_{Z_k}^k >, k = 1, \ldots, K$. Then,

$$\mu_k = \sum_{z=1}^{Z_k} x_z^k$$

When a new x comes to be assigned a class, the score

$$S_k = \frac{1}{N} \sum_{n=1}^{N} T_n(\pi_{\{n\}}(x) \mid k) \pi_{\{n\}}(\mu_k)$$

is computed and the class associated with the highest score is assigned to x.

They indicated that experiments which used non-exclusive tupling had some small improvements, but that they thought that the additional memory and longer computing time negated their use. Remember that in 1959, memory was expensive. Table II gives the results of some of their experiments which varied the index set size and whether the index sets were exclusive or nonexclusive, and whether the protocol was like their initial method or had averaging or rotating.

TABLE II
SHOWS THE PERCENT IDENTIFICATION ACCURACY WITH HANDWRITTEN BLOCK CHARACTERS FOR DIFFERENT SIZED INDEX SETS.

Index Set Size	Exclusiveness	Condition	Accuracy
3	Exclusive	Initial	78
2,3,5	Exclusive	Averaging	77-84
2,3,5	Non-exclusive	Initial	80-85
3	Non-exclusive	Rotating	88-92

They also tried experiments to see how much a change occurred when different mutually exclusive random index sets were tried. The results, shown for one decimal place accuracy, are for the handwritten block characters in Table III. The mean correct identification accuracy was 78.4 % and the standard deviation was 1.0 %, indicating that for their data set, the collection of random mutually exclusive index sets were used made little difference.

TABLE III
SHOWS THE PERCENTAGE OF CORRECT IDENTIFICATION FOR FIVE DIFFERENT CHOICES OF RANDOM N-TUPLING WHERE THE N-TUPLE SIZE WAS 2.

Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5	Mean	St. Dev.
78.5	78.2	77.2	77.8	80.1	78.4	1.0

In one section of the paper, where they discuss other ways of setting up the subspace classification method, they write:

> It might well prove useful to employ a correlation technique in which a sum is taken of the products of corresponding scores [2] page 230.

It is interesting that because that sentence foresaw a more principled way of designing a subspace classifier as we shall read in the next section.

The Bledsoe and Browning paper quickly stirred comments. The first to comment was Highleyman and Kamentsky [17]. They had a handprinted data set produced by 50 different people at Bell Labs with 36 character classes and they had a machine printed character database of the 10 digits produced by an IBM 407 line printer. As mentioned in Bledsoe and Browning, they shifted each character to the upper left hand corner of the box containing the character. They used index sets of size 2, like the initial experiment in the Bledsoe and Browning paper. They used a specially built generalized scanner for pattern and character recognition studies that produced a thresholded image of 12 rows by 12 columns. In addition to the replication of the first experiment of the Bledsoe and Browning paper, they tried the averaging characters technique, meaning for each class they averaged the class measurements over the 50 training instances and recognized by doing a cross correlation of the measurement to the average of the measurements in each class. Likewise for the 10 digits of the machined printed data. Their results are shown in Table IV. Highleyman's conclusion was

> It is evident that this study does not verify the results of Bledsoe and Browning.

TABLE IV
SHOWS PERCENT CORRECT RECOGNITION IN HIGHLEYMAN'S REPLICATION OF ONE OF THE BLEDSOE AND BROWNING EXPERIMENTS.

Data Set	Results Obtained By Bledsoe Browning	Replication of Bledsoe Browning	Character Averaging
Hand Printed	78.4	19.6	77.2
Machine Printed	100.0	86.7	99.6

Highleyman and Kamentsky [17] also used a different method to normalize the entries in the table where for each of the MK tables they divided each class k table entry by the square root of the sum of squares of all table entries for that particular class k table.

They tried the maximum likelihood method. The normalization was made by replacing each table entry with its logarithm and the table entries that are 0 are replaced with some predefined negative number. As it can be observed from the Figure IV, among the various table models, normalized tables created by the maximum likelihood method gives the best performance.

They also tried a cross-correlation method where for each character class, a pixel position contained the probability of that position being occupied by one of the training tuples of the class. They cross-correlated the measurement tuple with the average probability matrix for each class and assigned the class that whose cross-correlation peak was highest [18]. For the hand printed data set, the test set was the same as the training set. For the machine printed data, the test set was independent of the training set.

Bledsoe [19] seeing that the Highleyman and Kamentsky replication of his own work produced an accuracy of only 19.6% when his own work produced an accuracy of 78.4% soon published a comment on the Highleyman and Kamentsky article. He argued that the cause of their small accuracy was due to the size of the index tuple, which was 2 in their initial experiments, should be chosen depending on the data sets. To support his argument, he makes reference to the fact that Highleyman had shared his data set with the Sandia Corporation and they used index sets of size 6 and 8, and obtained the results shown in Table V.

TABLE V
SANDIA PERCENT CORRECT RECOGNITION [19]

Index Set Size	Results Obtained by Sandia	Accuracy
6	Hand Printed	80
8	Hand Printed	86
6	Machine Printed	100
8	Machine Printed	100

IV. EXTENSIONS

We begin by discussing the first extension done by Bledsoe [19]. The Bledsoe and Browning paper was published in 1959 at the Eastern Joint Computer Conference. Marvin Minsky was one of the people present and had listened to the Bledsoe presentation and read their paper and even commented on it in a discussion of problems in pattern recognition [20] at the Eastern Joint Computer Conference. Minsky [21] wrote a paper in 1961 that referenced the Bledsoe and Browning paper. In the page before he mentions Bledsoe and Browning, Minsky speaks of combining properties by using the assumption of class conditional independence.

Bledsoe read the Minsky paper and jumped on the idea of using products as he had mentioned in his 1959 paper. Minsky gave the equation for the class conditional independence assumption. Let the feature tuple be $(f_1, \ldots, f_N)$. The conditional probability of the feature tuple given a class c is written as $P(f_1, \ldots, f_N \mid c)$. The class conditional independence assumption states

$$P(f_1, \ldots, f_N \mid c) = \prod_{n=1}^{N} P(f_n \mid c)$$

The maximum likelihood method finds that class c which maximizes $P(f_1, \ldots, f_N \mid c)$.

What Bledsoe and Bisson [22] did is to think of their features as being generated by their projection of the measurement tuple $x^{N\times 1}$ to the subspaces defined by the mutually exclusive index sets $J_1, \ldots, J_M$. Under the class conditional independence assumption

$$P(x \mid c) = \prod_{m=1}^{M} P(\pi_{J_m}(x) \mid c)$$

In order to use class conditional probabilities, the definition for the T_{mk} tables must be changed to make the entry mean an estimate of the class conditional probability.

$$T_m(u \mid k) = \frac{|\{z \in [1, Z] \mid u = \pi_{J_m}(x_z) \text{ and } c_z = k\}|}{|\{z \in [1, Z] \mid c_z = k|\}}$$

Hence, $T_m(\pi_{J_m}(x) \mid c)$ is an estimate of $P(\pi_{J_m}(x) \mid c)$ and by the class conditional independence assumption, for any x, the maximum likelihood method would compute $\prod_{m=1}^{M} T_m(\pi_{J_m}(x) \mid c)$ for each class c and then find the maximizing class c to assign to x.

Table VI shows a selection of their results using the handwritten block printed characters of Highleyman [17]. The images were 12×12. In Table VI, the 40 sets training and 50 sets testing means that the 40 sets training is included in the 50 set testing. The 40 sets training and 10 sets testing means that the training and testing sets were independent.

TABLE VI
PERCENT CORRECT RECOGNITION
SIZE 2 AND SIZE 6 REFER TO THE SIZE OF THE INDEX SETS [22].
HK IS THE ABBREVIATION FOR HIGHLEYMAN AND KAMENTSKY.

Method	40 Sets Training 50 Sets Testing		40 Sets Training 10 Sets Testing	
	Size 2	Size 6	Size 2	Size 6
Initial	13.0	62.0	2	24
Averaging	48.5		25	
HK	78.5	81.5	58	53
HK Zero-State Suppressed	77.2	80.7	62	51
Max. Likelihood	83.2	91.7	63	61

It is clear that the technique is memorizing rather than generalizing and the training set needs to be much larger. Indeed, Kirsch [20] made the statement that in general, researchers were using training and test sets that were too small.

The class conditional independence assumption has an extension. Let $\{J_1, \ldots, J_M\}$ be a partition of $I = \{1, \ldots, N\}$ with $|J_m| = s,\ m \in [1, M]$ and $\{J_{M+1}, \ldots, J_{2M}\}$ be another partition of I with $|J_m| = s,\ m = M+1, \ldots, 2M$.

Then with the class conditional independence assumptions both

$$P_1(x \mid c) = \prod_{m=1}^{M} P_m(\pi_{J_m}(x) \mid c)$$
$$P_2(x \mid c) = \prod_{m=M+1}^{2M} P_m(\pi_{J_m}(x) \mid c)$$

hold. Different conditional independence assumptions yield different joint probabilities, but each joint probability is an extension of the marginals forming them. In that case, it would be reasonable to assign class c to x when

$$c = argmax \prod_{m=1}^{2M} T_m(\pi_{J_m}(x) \mid c)$$

Or using the Bledsoe and Browning sum rule for scores,

$$c = argmax \sum_{m=1}^{2M} T_m(\pi_{J_m}(x) \mid c)$$

Those who adopt the Bayesian position would write

$$c = argmax \sum_{m=1}^{2M} T_m(\pi_{J_m}(x) \mid c)P(c)$$

Obviously this idea works for any number of partitions. From the N-tuple point of view, using multiple partitions allows for the index sets of one partition to overlap with the index sets of another partition.

Bledsoe and Bisson [22] specifically discuss optimization. They state,

> Optimization, therefore, must at least include: finding the *best* n (index set size) and finding the *best* corresponding memory matrix (values to store in the T_{mk} tables). [22] page 415.

They did try to find a better way to form the N-tuple index sets. In this effort they worked with the Highleyman data set [17]. They determined the probability, taken over all classes, that each position took the value 1. Let $p_1, \ldots, p_N$ be those probabilities. They then sorted the probabilities in descending order. Let the resulting order be $p_{i_1}, \ldots, p_{i_N}$. They formed 12 index sets by taking 6 sequential indexes at a time from this ordering. The unused bottom 72 positions not covered by these index sets were pixels that were close to the border of the grid and whose probability of taking the value 0 was high.

They compared taking the index sets to be random size 6 index sets and index sets obtained from the ordering of the probabilities that a pixel takes the value 1. They compared two methods: the first using their initial approach and the second using the maximum likelihood approach. Their results are shown in Table VII. 40 sets means the same 40 complete alphabet sets were used for training and testing. 10 sets means 40 complete alphabet sets were used for training and 10 complete alphabet sets were used for testing. The 0-1 approach means the initial approach they took using table entries which were 0 or 1. The maximum likelihood approach means the table entries were the estimated class conditional probabilities.

TABLE VII
PERCENTAGE OF CORRECTION RECOGNITION

Method	40 sets: Testing = Training	10 sets: Testing
0-1 approach; random	62	24
0-1 approach; improved	98	50
maximum likelihood; random	91.7	61
maximum likelihood; improved	99.75	67

This suggests a generalization where the values a feature takes is more than 2. Determine the entropy for each of the N

order 1 marginal distributions. Low entropy means that a few values have a high probability and most of the values have low probability. The distribution is informative about the values the distribution generates. High entropy means that the probability of any value is similar to any other value. The distribution is not informative about the values the distribution generates. Therefore there is more information in the sample of values that the distribution generates. [23]. Discrimination means that the sample values are informative. With this idea we can we can order the first order marginals by entropy in descending order and create size s index sets, taking each successive index sets from the ordering.

The question naturally arises what are the best collection of index sets to use. This is equivalent to ask what are the best subspaces? Lewis [23] argues from an information and entropy approach, that the best subspaces to use in a product approximation of an unknown joint distribution where the only thing known are its marginal probabilities for the subspaces, defined by the index sets $J_1, \dots, J_M$, are those that maximize the entropy of the distributions on the subspaces.

His argument is from an information point of view. If we are given the marginal distributions whose product forms a joint probability distribution that is an extension of the marginal distributions, then in an information sense, the closest product approximation to the unknown joint distribution is that formed by a product having the largest entropy and therefore minimum information. Since the approximation is a product form, the entropy of the approximation is the sum of the entropies of the marginals forming the product. Interestingly enough Kullback [24] also used the same information point of view and it has come to be known as the Kullback-Liebler Divergence. Shore and Johnson [25] discuss the issue of the maximum entropy criterion.

Definition 5: The Kullback-Liebler Divergence from a probability distribution Q to a probability distribution P, both defined on the set X is given by

$$D_{KL}(Q||P) = \sum_{x \in X} log_2 P(x) \left(\frac{P(x)}{Q(x)} \right)$$

Historically, Lewis was not aware of the 1951 Kullback and Liebler [24] article although his paper appeared later in 1959, which was same year as the Kullback book [26].

In our context, we have estimates of the class conditional marginals defined by the subspace indexed by each of the index sets for each class. The implication from Lewis suggests that the collection of index sets of one class should be independent of the collection of index sets for another class. If the index sets are the same size, from the computation point of view, there is no more computation and memory used when the same collection of index sets is used to define the subspaces, as it is in the original formulation of the N-tuple method, for each of the class conditional compared to when the collection of index sets are different for each of the classes.

However, in the more generalized case, we do not have to have the restriction that each of the index sets have to be same size. But we do have to have the restriction that the total memory to store the class conditional distributions is no more than the size of memory we are restricted to use by the computer hardware on which the N-tuple method is executed. Since the range sets are fixed, then the restriction amount to designing the index sets $J_{mk}, m = 1, \dots M; k = 1, \dots, K$ to satisfy

$$\sum_{m=1}^{M} \sum_{k=1}^{K} | \times_{j \in J_{mk}} L_j | < \text{allowable maximum memory size}$$

When would we want the collection of index sets for a particular class to have one or more index sets larger than the common size for the others? We would make this happen for a class c that had lower identification accuracy or a class c that had a larger false identification rate for a different class c' than desired; i.e. $P(assign\ c' \mid true\ c) > \beta, c' \neq c$. Of course if we make one class have an index set larger than the common index size, we would have to make other index sets smaller.

A. Generalization of the Class Conditional Independence Assumption

It is also possible to generalize the class conditional independence assumption in a principled way and allow for overlapping index sets. We will illustrate with a small concrete example and then describe the general case. Consider the following expression where each of the probabilities P_{134}, P_{352}, is greater than 0 regardless of their values of their arguments.

$$\frac{P_{134}(x_1, x_3, x_4) P_{352}(x_3, x_5, x_2)}{P_3(x_3)}$$

Notice that this form does define a probability distribution. Since each of the terms are positive, the fraction is positive. And the sum over all values for x_1, x_2, x_3, x_4, x_5 equals 1. To see how this works, sum on x_1, x_4 first and discover that the total is 1.

$$\sum_{x_3, x_5, x_2} \sum_{x_1, x_4} \frac{P_{134}(x_1, x_3, x_4) P_{352}(x_3, x_5, x_2)}{P_3(x_3)} = 1$$

This shows that the product forms a probability distribution. The next question is whether the product distribution is an extension of the third order marginals used in the product. If the sum over all values of (x_1, x_4) is done first we are left with the marginal $P_{352}(x_3, x_5, x_2)$. And its sum over x_3, x_5, x_2 is 1. So the product distribution is an extension of P_{352}. If we do the sum over all the values of (x_2, x_5) we are left with the marginal P_{134}. So the product distribution is an extension of P_{134}. Hence the product distribution is an extension of the given marginals.

Now we describe the general pattern where the successive marginal distributions overlap with one common variable. Let $J_1, \dots, J_M$ be the index sets defining the subspaces. Consider the following constraints

$$J_a \cap J_b = \varnothing \text{ if } b > a + 1 \tag{2}$$
$$J_a \cap J_{a+1} = \{j_a\}, a = 1, \dots, M - 1 \tag{3}$$

Constraint (2) requires that non-successive index sets in the ordering $\langle 1, 2, \ldots, M\rangle$ have no elements in common.

Constraint (3) requires that successive index sets have only one element in common. Constraint (2) implies that the one element in common of successive index sets are unique.

If constraints (2) and (3) are satisfied, then

$$\frac{\prod_{m=1}^{M} P_{J_m}}{\prod_{m=1}^{M-1} P_{j_m}}$$

is the largest entropy probability distribution that is an extension of the given marginals $P_{J_1}, \ldots, P_{J_M}$.

Everything that we did for the case of an overlap of one, is true for the overlap of 2 or more and the overlaps do not all have to be the same size. For example,

$$\frac{P_{134}(x_1, x_3, x_4) P_{x_3,x_4,x_5}(x_3, x_4, x_5)}{P_{34}(x_3, x_4)}$$

is also a probability distribution that is an extension of its third order marginals.

What we have used here is a specialization of the theory of graphical models [27], [28], [29]. Form a graph whose nodes are the variables. Make each index set form a complete subgraph. Make sure that cliques are the index sets. Using the ordering $\langle J_1, \ldots, J_M\rangle$ and have the index sets satisfying constraints (2) and (4)

$$J_a \cap J_a + 1 \neq \emptyset, \; a = 1, \ldots, M-1 \tag{4}$$

The resulting graph will be triangulated (every loop of 4 or more will have a chord). Define $S_m = J_m \cap J_{m+1}, m \in [1, M-1]$. In graphical models $S_1, \ldots, S_{M-1}$ are called the separators. Then the joint probability of all the variables can be expressed by the product form

$$P_I(x) = \frac{\prod_{m=1}^{M} P_{J_m}(\pi_{J_m}(x))}{\prod_{m=1}^{M-1} P_{S_m}(\pi_{S_m}(x))}$$

Now we bring in the argument of Lewis [23] in a more general sense. Suppose we could know all the

$$C_i^N = \left(\frac{N!}{i!(N-i)!}\right)$$

i^{th} order marginals and we want to choose a subset of them that are mutually exclusive and which cover I. Which ones should be selected so that the entropy of the product extension has largest entropy? This is a combinatorial problem which Chow and Liu solved [30] for the case $i = 2$.

Suppose that there are N variables and the marginal probability of each of the $N(N-1)/2$ second order probabilities $P_{i,j}$ are known. Chow and Liu form a complete weighted graph with one node for each variable. The edge between node i and node j is given the weight defined by the mutual information of (i, j). He proves that the largest entropy product is obtained by finding spanning the tree with the largest total weight. Variations of the Chow and Liu procedure can be found in [31].

Definition 6: The Mutual Information of variables indexed by i and j is defined by

$$\mathcal{I}(i, j) = \sum_{x_i, x_j} P_{ij}(x_i, x_j) log_2 \frac{P_{ij}(x_i, x_j)}{P_i(x_i), P_j(x_j)}$$

Chow and Liu used Kruskal's algorithm [32] to compute the maximal mutual information sum spanning tree. They called the spanning tree a dependence tree. Although they only did the proof for the case where order 2 marginals are used, the idea of a dependence tree itself holds in the more general case. We consider the case of 4^{th} order marginals.

We will construct a graph. Suppose that $I = \{1, \ldots, N\}$. For the sake of simplicity of explaining, we assume that N is dividable by 2. There are $Q = N(N-1)/2$ size 2 subsets of I. Let 1,...,Q be the designations for the nodes, each associated with a subset of I of size 2. We call those subsets $W_1, \ldots, W_Q$. Form a graph having Q nodes. Connect node a with node b with an edge if and only if $W_a \cap W_b = \emptyset$. Put a weight

$$\mathcal{I}(W_a, W_b) = \sum_x P_{W_a \cup W_b}(\pi_{W_a \cup W_b}(x)) \\ log \frac{P_{W_a \cup W_b}(\pi_{W_a \cup W_b}(x))}{P_{W_a}(\pi_{W_a}(x)) P_{W_b}(\pi_{W_b}(x))}$$

on the edge between node a and node b. The difference now is that we are not interested in a spanning tree on the Q nodes. Rather we are interested in a subtree of $N/2$ nodes with each node designated by having a pair $\{i, j\} \subset I$ satisfying that that the $N/2$ node subsets are mutually exclusive and cover I. To find the tree we are interested in, we cannot use Kruskal's spanning tree algorithm because we are not going to construct a spanning tree on the Q possible nodes. However, like the Chow and Liu idea, we will construct a dependence tree T on a subset of $N/2$ nodes and $N/2 - 1$ edges which is a tree in the big graph of Q nodes satisfying that $\{W_{i_1}, \ldots, W_{i_{N/2}}\}$ is a partition of I.

We would like that dependence tree having the maximal sum of mutual information. To do this we suggest a greedy suboptimal algorithm. Let $S_n = W_{i_n}, \; n = 1, \ldots, N/2$. Begin by choosing the pair of nodes whose edge has the largest mutual information. Then successively connect a node to the existing tree having no overlap with the nodes in the graph so far, and not forming a loop, and having the largest mutual information. We continue until the graph is a tree of $N/2$ nodes whose associated subsets $S_1, \ldots, S_{N/2}$ form a partition of I. Figure (4) illustrates this. The edge set E of the tree in Figure (4) is

$$E = \{\{1, 2\}, \{2, 4\}, \{2, 5\}, \{1, 3\}\}$$

In our clique-separator product form, the index sets $S_1 \cup S_2, \ldots, S_{N/2-1} \cup S_{N/2}$ serve as the cliques and the index sets $S_1, \ldots, S_{N/2-1}$ serve as the separators.

The joint probability P_I of the product decomposition is suboptimally the largest entropy extension of the marginals $P_{S_i \cup S_j}$ for $\{i, j\} \in E$. Since we are computing the class conditional probabilities, let c be the class we are working with. Then

$$P_I(x^{N \times 1} \mid c) = P_{S_1}(\pi_{S_1}(x)| \; c) \prod_{\{i,j\} \in E} \frac{P_{S_i \cup S_j}(\pi_{S_i \cup S_j}(x)| \; c)}{P_{S_i}(\pi_{S_i}(x)| \; c)}$$

There are also other combinatorial approaches to constructing a tree to cover all the variables, but we have no space here to discuss it.

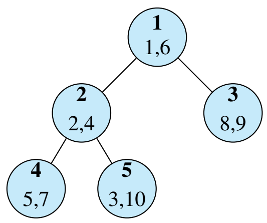

Fig. 4. Shows a dependence tree example for a measurement tuple with 10 components. Since each node has 2 indexes, the tree has five nodes. Each edge is associated with the pair of indexes in the upper node combined with the pair of indexes in the lower node thus forming a size 4 index set, indicating an explicit dependence among the index sets. Because of this dependence, the marginal probabilities are fourth order. The top of nodes are labeled 1 through 6 with bold font. The bottom of the nodes are the corresponding size 2 sets of indexes.

Further information can be found about product extensions in [33]. and there is a statistical methodology to go from the training set for a given class to the conditional independence graph from which the cliques (our index sets) and separators can be determined. [34], [35] [36]. But there is not enough space here to even open up the topic.

Having explored some of the different kinds of product forms, we will now explore a different kind of sum form. Recall that for any measurement tuple x, $T_m(\pi_{J_m}(x) \mid c)$ is an estimate for $P_m(\pi_{J_m}(x) \mid c)$. What happens if we consider each of the probabilities $P_m(c \mid \pi_{J_m}(x))$, the probability of class c being the true class given the evidence $\pi_{J_m}(x)$ as the basis for a classifier in its own right.

This is a problem that began to be examined in the 1990's with the classifier ensembles. The most popular method is having each classifier vote its most likely class. Tabulate the votes and the class having the plurality of votes is the class that gets assigned.

For the case of two classes, the rational for counting the votes for each class and then assigning the class having the majority comes from Nicolas Caritat de Condorect (1743-1794), a mathematician who advocated that it was better to have a group of people who are partial experts than one person who is an expert to make a decision. This has come to be known as the Condorcet Jury Theorem. It states that if there are two classes and an odd number of n jurists, ($n = 2m + 1$ for some m), who act independently, each with a probability $p > 1/2$, of making a correct decision, then the probability of a jury using the majority vote rule will make a correct decision monotonically increases to 1 as the odd n increases to infinity [37].

More precisely, if S is the sum of the majority n voters for an action, where $n = 2m + 1$, then the probability that $S \geqslant m + 1$ is given by

$$P(S \geqslant m+1) = (2m+1) \begin{pmatrix} 2m \\ m \end{pmatrix} \int_0^p x^m (1-x)^m dx$$

This is the conceptual basis for the ensemble classifiers such as [11]. Those interested in reading more about the use of multiple classifiers and combining their results can consult [38], [39], [10], [40], [41], [42].

To make use of the voting, if there are W partitions then for each w, the subspaces specified by the mutually exclusive $J_1^w, \ldots, J_M^w$ index sets each having its own classifier rule based on the estimated class conditional probability $T_{J_m}^w(\pi_{J_m}(x) \mid c)$.

For the voting scenario, when

$$T_{J_m}^w(\pi_{J_m}(x \mid c) P(c) > T_{J_m}^w(\pi_{J_m}(x \mid k) P(k), k \neq c$$

a vote for class c is counted. So taken over the ensemble consisting the M index sets in each of W partitions the vote count for class c is given by

$$V_c = \sum_{w=1}^{W} \sum_{m=1}^{M} \begin{cases} 1 & \text{if } P_{J_m}^w(\pi_{J_m}(x,c) > P_{J_m}^w(\pi_{J_m}(x,k), \\ & \qquad k \neq c \\ 0 & \qquad \text{otherwise} \end{cases}$$

Kittler and Alkoot [43] showed that when the error of the estimates for the class conditional probability for the various subspaces has fat tails, then the plurality vote method does better.

In closing this section, we mention the possibility of using deep learning to perform automatic feature extraction from images, sequence of images, multiband images and other forms of spatial data. The automatically extracted features can be input to the N-tuple subspace classifier. This is advantageous because the N-tuple classifier can use class prior probabilities and is able to control misidentification errors by reserve decisions.

V. Survey

In this section we select a group of representative papers that the original Bledsoe and Browning 1959 paper inspired and give a short description of their contribution, in the way of innovations, applications and experimental comparisons of the accuracy of the N-tuple subspace classifier with other state of the art classifiers.

Ullmann [44] did experiments to observe how the tuple size behaves differently with different sizes of training sets. The task is to recognize hand-printed characters, more specifically the numerals from 0 to 9, that were written by 650 different people. Then the papers that the subjects wrote on, were digitized, centered, size-normalized producing 30×22 sized images for each character. Figure 5 shows the true probability of correct identification as a function of index set size.

As it can be observed from the results of the experiment, as the index size increases, depending on the size of the training set, the accuracy increases, stabilizes and then decreases. The larger the training set size the higher the accuracy is obtained, particularly for the larger sized index sets. From Figure 6, the number of index sets of size 14 to 20 gives the best accuracy is 264 and above.

In order to use the N-tuple method, real-valued measurements must be transformed to take integer values, the smallest being zero, so that the integers in the different components can be used to form an address into memory. In general an L level quantizer is a monotonically increasing function that takes in a real number and assigns a non-negative number from 0 to a maximum integer $L - 1$. The function that takes

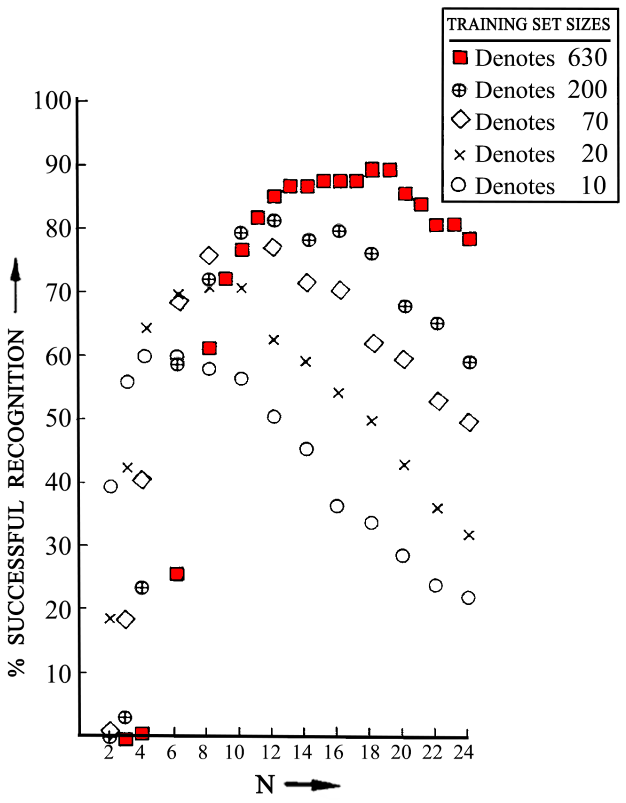

Fig. 5. Taken from 5, the figure shows that as the training set size becomes larger up to 630 measurements, index set sizes between 14 and 20 give the best accuracy. When the index set sizes go beyond 20, the accuracy decreases. This experiment was done on a plain, non-weighted version of the N-tuple subspace classifier and it did not have any flexibility other than allowing the selection of the size of the index sets for different sizes of the training.

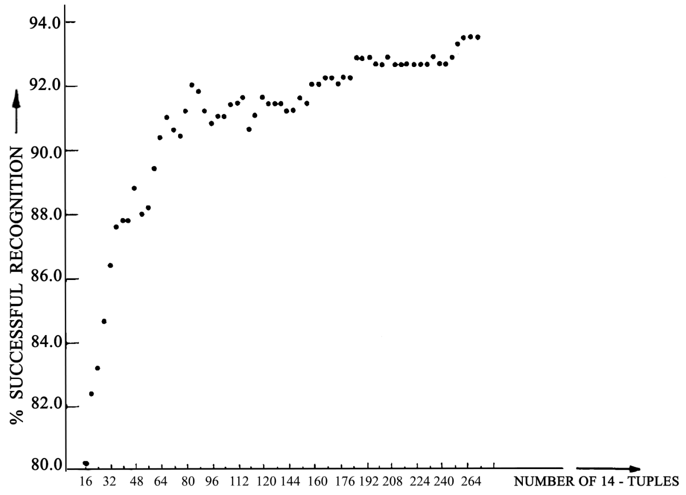

Fig. 6. Shows the experimental results for an N-tuple model with different numbers of 14-tuples along with the identification for the training and testing sets of 600 and 50, respectively per class for the same experimental data shown in Figure 5 .

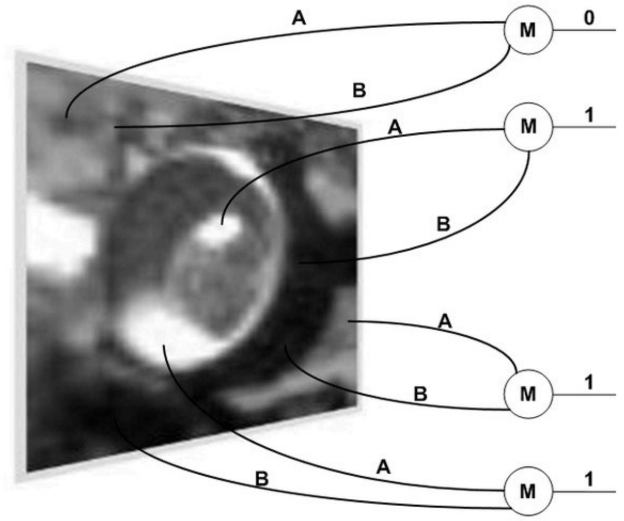

Fig. 7. Shows Minchinton Cells. As seen from the example preprocessing elements, M cells sample two indexes from the data, and produce either binary 1 or 0 based on the comparison of point A and B: adapted from [48]

in a quantized measurement tuple and converts it to a memory address is the kind of function that determines an address in a multi-dimensional array.

A preprocessing quantization technique that has been used with the N-tuple technique is called Minchinton Cells. In this technique, before sending the input vector into the network there are preprocessing elements called the Minchinton cells [45]. Minchinton cells act as simplifiers of the data. Each cell selects 2 random indexes, adjacent or close to each other, from the input vector and produces a binary 1 if the first value is greater than the second or a binary 0 otherwise. The new set of binary 0 or 1 values form the feature set. Also just like the N-tuple subspace method, the Minchinton cells can be implemented in hardware as discussed in Mitchell and Minchinton's 1993 article [46] see Figure 7. In 1998, Mitchell and Bishop conducted experiments [47] that provided evidence that Minchinton cells produce a good quantization.

In the previous section, we mentioned the commercial WiSARD that implemented the N-tuple method. A good discussion of the WiSARD by Alexsander can be found in [49]. We just note here that there are also modified versions of the WiSARD model, where they merge the WiSARD model with a virtual neuron technique [50], [51]. There is another extension DRASIW with the capability of producing the frequencies of seen patterns, prototypes during the training in an internal data structure called "mental images" [52] [53] [54], WiSC; extends the learning and the classification capabilities of WiSARD with the data transformations for numeric/symbolic data processing discussed in detail in [55], CDNet; a special version designed by De Gregorio for change detection [56] and tWiSARD (threshold WiSARD) created by [57] for open set recognition.

The index sets $J_1, \ldots, J_M$ can be optimized for highest classification accuracy. Jung et. al. [58] proved that this problem is NP complete. However, they indicate that one can

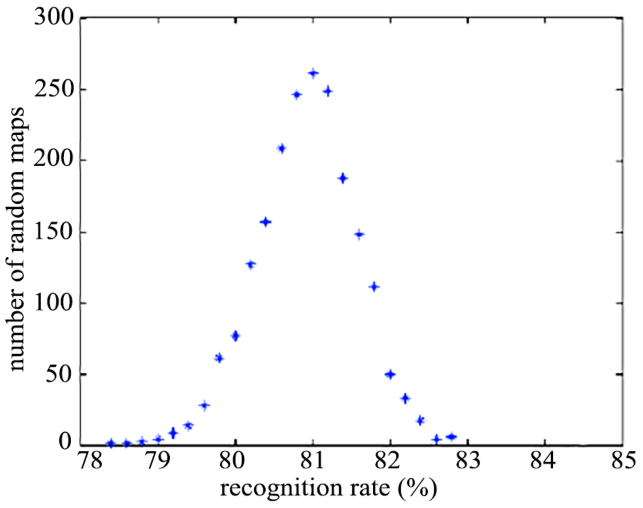

Fig. 8. Shows the probability of a random collection of index sets producing a given probability of correct identification for the NIST handwritten character database. This figure is taken from [63].

design generators that do not take exponential time and get close to the best answer [59].

One of the many strategies for selecting the index sets uses a genetic algorithm. Although the idea dates back to 1970, Cavicchio [60] suggested that genetic algorithms might be used for the selection of optimum sets for pattern recognition tasks, this was first proposed by Bishop [61] for the N-tuple method. In his method, the index set selection technique uses an evolutionary learning strategy. By the definition of genetic algorithms, the logic is simple, generate children better than their parents, or the same logic as survival of the fittest in the hopes that better tuples will result in generating even better tuples. If a randomly selected index set gives an undesirable result compared to the other randomly selected index sets, then it is discarded, and a new index sets is generated based on the mutation and crossing over between the index sets.One of the early experiments using this optimization technique is done by Badii and Mumford [62] in their speech recognition task.

Each class $k \in [1, K]$ has its own collection $\mathcal{J}_k$ of index sets. M is the number of index sets for each class.

```
1) For $k \in [1, K]\{$
2)  Set $\mathcal{J}_k = \varnothing$,
3)  Do While $|\mathcal{J}_k| < M\{$
4)    Generate random index sets $J_1, ..., J_M$ for class $k$
5)    Using the training set, compute the class $k$ scores
        for each of these $M$ index sets.
6)    Based on the class $k$ score, sort the $M$ index sets
        in descending order
7)    If Score$(J_{(1)})$ is high enough, add $J_{(1)}$ to $\mathcal{J}_k$
8)    $\}$
9) $\}$
```

Azhar et. al. [63] have an interesting figure in their paper. They did an experiment with the NIST character database. They randomly chose 2000 collections each of 140 sized 8 index sets. For each collection, they determined the probability of correct identification for the N-tuple subspace classifier. From this data they are able to graph the probability of a randomly selected collection of 140 sized 8 index sets producing any given percentage of correct identification.

This is shown in Figure 8. The correct identification mode is 81%. The left tail is larger with a minimum correct identification of about 78%. The smaller right tail shows a maximum correct identification near 83%. It was undoubtedly the information of this figure that prompted Azhar et. al. to design a stochastic search algorithm to find a good performing collection of index sets. Noteworthy is that they find a different collection of index sets for each class.

We show here a slightly modified, more effective version, of their algorithm. Each class $k \in [1, K]$ has its own collection $\mathcal{J}_k$ of index sets. M is the number of index sets for each class.

```
1) For $k \in [1, K]\{$
2)  Set $\mathcal{J}_k = \varnothing$,
3)  Do While $|\mathcal{J}_k| < M\{$
4)    Generate random index sets $J_1, ..., J_M$ each of sized
      $s_k$ for class $k$
5)    Using the training set, compute the class $k$ scores
        for each of these $M$ index sets.
6)    Based on the class $k$ score, sort the $M$ index sets
        in descending order
7)    If Score$(J_{(1)})$ is high enough, add $J_{(1)}$ to $\mathcal{J}_k$
8)    $\}$
9) $\}$
```

An alternate approach of generating index sets is the domain-aware selection of indexes for index set J_m. In this approach, the index set generation is not totally random but modified such that some indexes will be picked more and some certain indexes will be omitted. A domain aware selection of J_m, [56] [57] pays attention to the dataset properties e.g. centered, normalized aiming to select indexes that will yield meaningful tuples as well as aiming to avoid indexes that create fewer different tuples. For instance, if the input vector has all same values in some certain indexes for every class, then it will be useless to include those indexes in the neural network model. Distinctive indexes are preferred.

Giordano and De Gregorio [64] designed a way for optimizing the index sets. They use a full-evolutionary algorithm to optimize the selection of the index sets. First, they create $N \times M$ matrices to represent the number M of index sets and the index set size t as a 2-dimensional array. Then they apply the steps of evolutionary algorithms. Create a matrix S, Swapping two indexes from the same index set or between two index sets. They measure the fitness of the index sets with the given tuples by calculating the 10-fold cross-validation accuracy for each combination. After generating initial matrices and measuring their fitness, they apply crossing-over and mutation. And they check the new fitness of the newly generated index sets with the given tuples. After calculating the 10-fold cross-validation accuracy for them, they take the fittest to the generation of index sets, or they discard it depending on the accuracy they get.

Roy and Sherman [65] show a correspondence between the N-tuple subspace classifier and the Φ machine. The Φ machine consists of two parts: the Φ preprocessor followed by a simple neural network. The ϕ preprocessor inputs the pattern tuple

$(x_1, \ldots, x_N$ and outputs the R tuple $(f_1(x), \ldots f_R(x))$. In the case of the ϕ quadratic preprocessor, the output consists of the original tuple components plus all pairs of products of the tuple's N components. If the N-tuple subspace classifier has M mutually exclusive sized 2 index sets then a full quadratic preprocessor will have more terms to sum. But the terms of the N-tuple classifier are more effective because they are class conditional probabilities. When the index sets are sized $s > 2$, the class conditional probabilities of a selected set of s-tuples could very well be competitive with a s-term Φ preprocessor. The Φ machine can do statistical approximation which was adapted by Kamentsky-Liu [66] and Steck [67]. Interestingly, Jürgen Schürmann [68] used a quadratic Φ preprocessor for the recognition of letter postal addresses for the German Post Office. His reader was used for many years in the German Post Office.

In 1994, Aleksander proposed a hybrid method, that uses a weighted neural network, and weightless neural network concepts combined to perform the machine learning tasks. He aimed to use the advantages of each approach; synaptic sensitivity of weighted models and the pattern sensitivity of weightless ones [69]. If a probabilistic model is used, in case that there are skewed class priors, there is a work by Linneberg and Joergensen [70] who describe a design algorithm to implement the cross-validation technique for the N-tuple subspace method. They proposed a non-complex design strategy that allows the N-tuple subspace method to perform satisfactorily even in cases with skewed class priors. In 1999, Jørgensen and Linneberg [70] discussed the relationship between the Hamming Distance and the class scores for the N-tuple subspace method. They stated that the class scores can be also calculated using a Hamming Distance function.

The original $0-1$ N-tuple subspace classifier, can have a problem called saturation, when the training set is very large or when the patterns for each class are widespread. In this case nearly all the table entries for each class will be binary 1 after some point [71]. For this reason, adding a 1 each time a table entry is accessed by the training data can be useful.

Grieco et. al. [54] first introduced the bleaching threshold for the table entries. But instead of using the threshold on the class conditional probabilities, the use it on the un-normalized frequency counts.

$$T_m(u \mid k) = |\{z \in [1, Z] \mid u = \pi_{J_m}(x_z) \text{ and } c_z = k\}|$$

Any table entries that are greater than or equal to this bleaching threshold b is set to binary 1. If they are less than this bleaching variable then they are set to binary 0. By doing this the researchers solve the problem of saturation that occurs with over-training. The technique finds an optimal bleaching threshold, since larger thresholds will yield to only the most frequent sub-patterns present which would create poor learning.

The method was used and tested by a number of researchers beginning with Carvalho et. al. [72]. Coutinho et. al. [73] and França [74] designed DRASiW to improve WiSARD. Then it was used by Kappaun et. al. [75], and proven effective by Carneiro [76]. His work indicated that bleaching is a powerful enhancement that does not negatively impact the classification ability of the model.

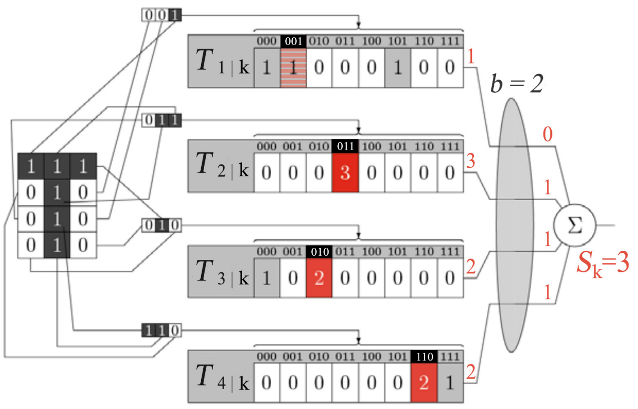

Fig. 9. A demonstration of bleaching where the class is k, Index set size is 3, bleaching threshold $b = 2$ and $S_k T$ the class score for class T. The grey addresses are from other training set measurement tuples. The red addresses are from the pattern on the retina having counts at least as large as the bleaching threshold. The hashed red address is from the pattern on the retina but does not pass the bleaching threshold. This figure is adapted from [72].

The question then becomes how to find the optimal bleaching threshold since larger thresholds will yield to only the most frequent sub-patterns present which would create poor learning. Smaller thresholds will yield to saturation [72].

The optimization procedure starts with setting the bleaching threshold b to 0 and observe the outcomes for all the tables that belong to the same class. After this process, some of the tables will have a 1 while some others have 0 as their output. Then among those tables whose output is 1, by increasing b by 1 and observing the outcome of those tables whose output was 1 with the increased bleaching threshold. At the end of these iterations, only the most relevant tables will stay. And we can assign the class by choosing the highest score only among these remaining tables. We think that it would have been better to define the bleaching threshold for the class conditional probabilities rather than the frequency counts because depending on the prior class probabilities, one class could have high frequency counts and another low frequency counts.

As it can be seen in this example [72], shown in Figure 9, the bleaching threshold b sets the score of $Table_1$ to 0, since its table entries are less than the bleaching threshold which is 2, and the scores of $Table_2$, $Table_3$, $Table_4$ to 1 since their entries are greater than or equal to the bleaching value, 2. In this example, we can observe that the sub-pattern of $Table_1$ is not useful, and the remaining 3 tables are relatively more useful(with some higher score than $Table_1$).

Let $f(T_{mk}(\pi_{J_m}(I, q)))$ is the bleaching function for class k discriminator that consists of M tables and b the bleaching threshold. Then the function $f(T_{mk}(\pi_{J_m}(I, q)))$ for each table m can be defined as follows

$$f(T_{mk}(\pi_{J_m}(I,q))) = \begin{cases} 1, & if\ T_{mk}((\pi_{J_m}(I,q))) \geqslant b \\ 0, & \text{otherwise} \end{cases}$$

Later in 2016, Kappaun et. al. [75] compared different encoding techniques such as threshold, thermometer encoding, local threshold, Marr-Hildreth filter, and Laplacian filters [77] using bleaching technique on the mnist data set of the handwritten numerals [78]. In their work they concluded that using 32-bit WiSARD model with 1-threshold, is ideal for the mnist data set, resulting in 94 % accuracy.

Bloom [79] shows the similarity between the data structures that are likely to be used in the implementation of the weightless neural networks and the Bloom filters. Bloom filters are space-efficient data structures for an Approximate Membership Query (AMQ) which tests if an element belongs to a given set or not, with a certain false positive probability. A Bloom filter consists of an m-bit array and k independent hash functions that map an element into k-bit array positions [80]. In this work, the researchers stated that the data structures for the N-tuple subspace method, are often dictionaries, and hash tables. And an alternate approach is to use Bloom filters along with such data structures, hash tables. They created a unified model called Bloom WiSARD, where they use the advantage of the Bloom filters in the WiSARD model. They observed a significant increase in the model's accuracy compared to plain WiSARD, and Dictionary WiSARD. Also in 2020, Nazuno, Nava, Medina used a set of multiple weightless neural networks working together to perform the tridimensional pattern reconstruction [81].

VI. Applications and Experiments

In this section we discuss the variety of experiments using the N-tuple method, on real world data sets. Also from these experiments we can observe the performance of the N-tuple subspace classifier compared to the performance of other competitors. Throughout the years, researchers used the N-tuple method not only for the classification of hand printed numerals but also for a wide area of machine learning tasks, including face/image recognition [15] [82] [83], fingerprint recognition [84] speech recognition [62] [85], change detection [86], 3D video acceleration [87], automated text categorization and clustering [88] [89] [90], fault detection [91], background modeling [16] [92] [93], financial credit analysis [94], forecasting stock market prices [95] [96] [97], automatic disease diagnosis using wearable technology [98], P300-Based Brain-Computer Interface [99], real-time music tracking [100] robotics [101] [102] [103], observing artificial consciousness [104] [105] [106], monitoring zoological behaviors [107], nonintrusive load monitoring [108], predicting multi-modal empathy [109], solving an NP-Complete 3-SAT problem [110], the detection of elementary particles [111], process mining classification [112], deforestation surveillance and visual navigation [113], and open set recognition [57] [114].

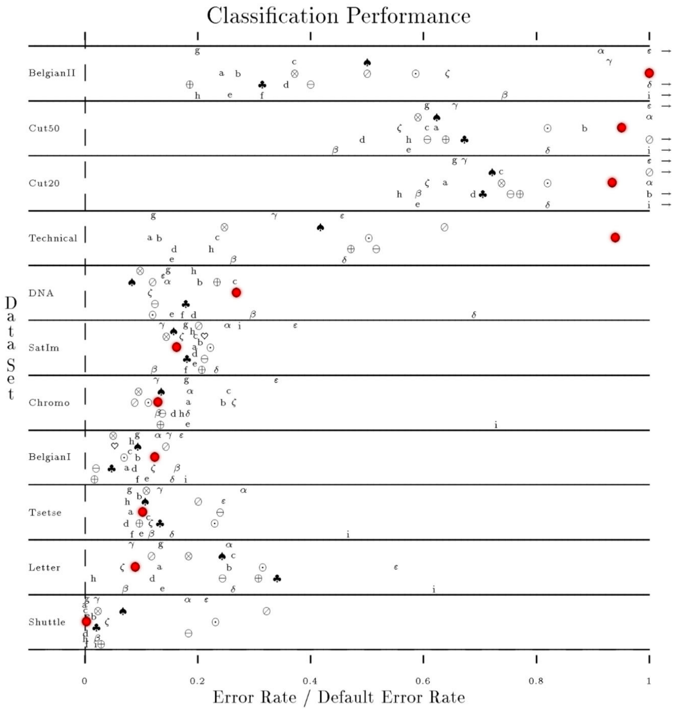

Fig. 10. Shows Rohwer and Morciniec's large set of experimental results with different classifiers on different data sets. The x-axis is the relative error rate. The y-axes are designating different data sets. Each of the 24 symbols, shown in the figure, designate a different classifier type. The red circle is the N-tuple subspace classifier. A table of the kinds of classifiers and the kinds of data sets can be found in [115] . The x axis is the relative error rate which is defined as the error rate divided by the default error rate. The default error rate is the error rate obtained by random guessing.

Rohwer and Morciniec [115] performed a large series of experiments on standard pattern recognition databases that contain real world data sets (European StatLog project). Their experiments for most real data sets showed that the best results were achieved with N-tuples of size 8 and the total number of index sets to be approximately 1000. The experiments use various classifiers as the competitors of the N-tuple method, such as k-nearest neighbors, various kinds of decision trees, radial basis functions, and multi-layer perceptrons. They thought of the competitors in different groups: 8 different kinds of discriminators, 6 different methods related to density estimation and 9 different kinds of decision trees. Figure 10 shows the results of their experiments on 11 different data sets. The N-tuple subspace classifier was competitive with 7 out of the 11 experiments.

Rohwer and Morciniec write about these four abnormal cases. For one of them the training set was way too small in size. For the others the prior probability of the classes were way skewed. It is understandable why the skewed priors would mess up their implementation of the N-tuple subspace classifier. Their implementation did not use the prior class probabilities, as a Bayesian would. Later this project used cross-validation techniques for the N-tuple method [70]

York and Duggan [116] did an experiment on the processing of tomographic data using weightless neural networks. They used the plain WiSARD [15] method in their experiments related to image reconstruction, component fraction estimation, and flow regime. Since they used a plain version of the WiSARD model, the optimization that they implemented was to test and decide the index set sizes (5,6,10) for each of the training sets. They also observed how the tuple size selection shows different results for different training set sizes. Their average identification accuracy was 89% accuracy.

In 1998, Mitchell and Keating [117], created a combined neural network model using Aleksander's WiSARD [15] and Hopfield Networks, to perform the control task of a simple mobile robot. In their work, they used an array of Minchinton Cells for the quantization. They used an algorithm to combine the neural networks to successfully allow the robot to navigate back to a position it recognizes within its environment.

In 2002, Azhar and Dimond [118] implemented Aleksander's WiSARD model [15] in a Field Programable Gate Array (FPGA) for a robotic navigation task. Their results show that the weightless neural network is perfectly capable of performing the task of robotics navigation using limited processor capability and memory size.

In 2003, Yao et. al. [101] adapted the WiSARD model to another robotics task: the collision avoidance problem. They used a WiSARD model with a threshold. Their WiSARD neural network was implemented in a robot with a very modest microprocessor system as well as a small amount of data memory (512 bytes). This clearly means that with a low computational complexity and space complexity, the N-tuple method has an advantage over the other methods that are capable of this robotics task. As a result of their experiment they observed that their neural network was easy to train and the robot demonstrated the ability to detect obstacles and navigate within its environment in real time.

In 2004, a group of biologists used the N-tuple subspace classifier for a protein classification task [119]. They combined the Hidden Markov Model to the N-tuple subspace method. The learning time for protein classification was shown to be substantially faster than the traditional neural network models, since they were dealing with larger numbers of the input units. As a result, 7 out of 9 sampled protein families were classified correctly with an accuracy as high as 100%.

In 2005, De Souto, Ludermir and De Oliveira [120] developed a weightless neural network by adding hidden neurons (state neurons) and a single output neuron (a feedforward node) which they called General Single-layer Sequential Weightless Neural Networks (GSSWNNs). They proved the weightless neural network model they developed behaves exactly like Probabilistic Automata.

In 2006, Mpofu used the weightless neural network having a RAM node for forecasting the stock prices in the market. prices [95]. The model was created based on the interpretation of Aleksander's [15] version of the N-tuple method, and compared results with the Single Exponential Smoothing (SES) forecasting model. The results showed that on stock data, the weightless neural network forecasting model gave a smaller mean squared error of 0.7 whereas the SES forecasting model gave a mean squared error of 15.4.

A year later in 2007, Oliveira et. al. used a Virtual Generalizing RAM (VG-RAM) network which is a variation of the N-Tuple subspace classifier for the classification of economic activities from free text descriptions [121]. For the free text, they used the Vector Space Model (VSM) and Ludermir's version of the weightless neural network [7]. They compared these two models, and their results for correct classification were 63.36% and 67.03% for VS and VG-RAM WNN respectively.

In 2008, Arguelles-Cruz et.al. used a modified version of a weightless neural network called the Alpha-Beta Weightless Neural Networks [122]. They developed their own learning algorithm to work with the data set. Although their results were approximately as high as other state-of-art classifiers such as RAMP, SVM, kNN, MLP, RM, C4.5, and C4.5+m, their model could not do better in terms of accuracy. However, they stated this trade-off in accuracy was worth it because the difference in accuracy was small and the training was quicker. Finally the on-line execution, whether by computer or special hardware was quick.

In 2008, DeSouza [123] conducted face recognition experiments with the AR Face Database and the Extended Yale Face Database B. They used the Minchinton cells for data quantization. They compared the N-tuple subspace classifier, which they called a VG-Ram weightless neural network, to other classifiers such as PCA, NMF, LNMF, LEM, WER, and ARG. In some of their experiments they used local features and in other experiments they used more global features. Their results showed the strength of the N-tuple subspace classifier in an image classification, producing the highest accuracy compared to all their other classifiers both with the local features and the global features.

In 2010 Carneiro et. al. [124] introduced the WiSARD form of the N-tuple classifier to tag the Portuguese language. Then in 2015, Carneiro et. al. [125] extended the tagger for multiple languages including Mandarin Chinese, Portuguese, Japanese, Italian, English, German, Russian, and Turkish. They called their tagger The Multilingual Weightless Artificial Neural Network tagger.

For optimization, they used the b-bleaching technique [54] [72]. Their accuracy was compared to other state of art speech taggers such as the CRF and TNT-Tagger. The tagger using the N-tuple subspace classifier outperformed or was just as good as CRF or TNT. The standard deviation on its accuracy was .25%. In comparison, the standard deviation accuracy of the TNT tagger was 2.25% and standard deviation of the accuracy of the CRF ranged as low as 1.0% in English to as high as 3.6% in Italian. They also compared the speed of the learning and the tagging with the other taggers. The long cross-validation process of CRF was beaten by the speed of the N-tuple classifier. Although the CRF tagging was slightly faster, the N-tuple classifier's learning was much faster.

De Oliveira and Wilson [126] is the first paper mentioning a quantum computing approach for the weightless neural network nodes. In polynomial time, they solved the 3-SAT problem, which is an NP-Complete problem, using the N-tuple subspace classifier with a quantum computer with qRAM nodes. The result and future work section of their paper gave some promising hope for other NP-Complete problems to be solved in polynomial time.

Cardoso et. al. [114] [57] extended the application area of the N-tuple subspace classifier by applying it to the open set recognition task and data stream clustering [127]. In their work of data stream clustering they used a special algorithm of time stamps presenting extra information about table entries such as every time an address is recorded, its reference is updated as the most recently used. They compared their interpretation of WiSARD; WCDS to several conventional clustering techniques such as k-means algorithm and HDB-SCAN. Their results show that both in synthetic dataset, and Complex8 dataset, their WiSARD model gave better results than most of the competitors. Open set recognition is a term given when the training set does not include all classes that might be encountered and the classifier must recognize that a measurement tuple that it is to assign a class does not belong to any of the classes it has been trained on. Open set recognition is a term given when the training set does not include all classes that might be encountered and the classifier must recognize that a measurement tuple that it is to assign a class does not belong to any of the classes it has been trained on. Cardoso et. al. write

> Any observation is labeled as the class it best fits if the uncertainty of this prediction is considered acceptable; otherwise, the observation is labeled as belonging to an unknown class. [114]

The classical pattern recognition world would call the unknown class option a reserved decision or a reject possibility.

They used a modified version of the WiSARD classifier, and called it tWiSARD (threshold WiSARD). Cardoso et. al. [57] compared their model with other state-of-art classifiers such as Support Vector Machines, Gaussian Naive Bayes, as well as the regular WiSARD model. For most of the cases they examined, tWiSARD had the better performance.

In 2020, Gryak et. al. [128] worked on a cryptography problem. The competing classifiers were decision trees and random forest classifiers [11]. They set up the N-tuple classifier to solve the conjugacy decision problem for a pair of elements from some six different finitely presented non free infinite groups. This problem is known to be undecidable. They used a greedy approach to select the M index sets of the same size and they evaluated the quality of index sets that are chosen with an accuracy threshold, that was initially 60%. Above 97% was considered as acceptable. They generated three data sets for each group, three non-virtually nilpotent groups, two non-polycyclic metabelian groups, and one a non-solvable linear group. Each of the data sets consists of $20,000$ geodesic word pairs, $10,000$ of which represented a pair of conjugate elements, while the other 10,000 represented pairs of non- conjugate elements. The N-tuple classifier performed better on five of the six groups. The highest accuracy were 98.49%, for the decision trees with entropy and depth limit; 99.41% for the random forest with 200 trees; and 99.81% for the N-tuple subspace classifier using 50 index sets each of size 4.

Ferreira [129], performed experiments to improve the time and the space complexity of the WiSARD model. Using hash tables, instead of indexed lookup tables thus reducing the number of discriminators for each of the classes, they Their experiments were conducted on the mnnist data set [78]. The results did not show a radical difference in accuracy from the traditional WiSARD model. The accuracy for the traditional WiSARD model on the same data set is approximately 93% and the WiSARD using hash tables on the same data set is 92% for the largest sized hash table they used in their experiments. Their approach presented an insight about reducing the time and the memory requirement. Their experimental results showed a great decrease in memory usage with a small identification accuracy trade-off.

In 1991, Specht [130] introduced the neural network possibility for performing a kernel regression. In 1996, Kolcz and Allinson [14] adapted the N-tuple subspace classifier structure for the regression task. They experimentally showed the N-tuple neural network (NTNN) general regression model produced less error than a Gaussian kernel regression. In 2019 Filho et. al. [131] used the N-tuple Regression network to predict palm oil production. In 2020, Filho et. al. [132] created a software library including ClusWiSARD which allows the same class to have more than one discriminator, Regression WiSARD for prediction tasks, and ClusRegression which separates measurement tuples that are not sufficiently similar. Both Regression WiSARD and ClusRegression WiSARD models were adapted from Kolcz and Allinson's N-tuple Regression weightless neural model. Their results showed that the two modified versions of the WiSARD model they used are very competitive with the state-of-the-art methods, while being four orders of magnitude faster during the training phase [133]. In addition, they tested their regression models on three classic regression datasets, also presenting competitive performance with respect to other models often used in this type of regression tasks [134] [109].

In 2020, another quantum computing approach model [135] of the weightless neural networks was used and tested on different public benchmark data sets. The model used a customized version of a probabilistic quantum memory nodes PQN to perform pattern classification.

VII. Conclusion and Future Work

We have explained the mechanism of the N-tuple subspace classifier in a notation which is unambiguous and complete. We have suggested a number of extensions that should be tested and have potential for making the N-tuple classifier more accurate, including more generalized class conditional probability assumptions and related these to the graphical models. We discussed the sum rule of combining and product rule of combining the outputs of the subspace classifiers. We surveyed a selection of the papers that were inspired by the Bledsoe Browning 1959 paper and had some innovation. Then we surveyed a selection of application papers using the N-tuple subspace classifier. Our conclusion is that the N-tuple subspace classifier should be considered as being in the group of state of the art classifiers. It trades off more expensive computation to cheaper large memory lookup. It can easily be adapted to a parallel computing environment where there would be relatively little communication required between processors.

Our planned future work for the N-tuple subspace classifier will be to use simultaneous perturbation stochastic approximation (SPSA) for optimizing the T_{mk} tables [136]. The SPSA algorithm has been successfully applied to an optimization in various situations having large numbers of variables and an objective function that does not have an analytic form, but which can be evaluated for any value of its arguments. The method is faster than gradient search and does not need to compute the Hessian matrix.

With the work of de Oliveira and Wilson Rosa [126] weightless neural networks obtained a popularity for another reason: the possibility of being implemented in quantum computers as described in papers such as [74] [137] [138] [139].

There will undoubtedly be quantum extensions to the N-tuple subspace classifier using q-bits [140] [141], instead of classical Boolean neurons. The recent advances in quantum computing are very likely to affect the interpretation of the N-tuple subspace method.

References

[1] M. Ballantyne, R. Boyer, and L. Hines, "Woody Bledsoe His Life and Legacy," *AI Magazine*, vol. 17, no. 1, pp. 7–20, 1996. [Online]. Available: https://www.cs.utexas.edu/users/boyer/bledsoe-memorial-resolution.pdf

[2] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Eastern Joint Comptuer Conference*. ACM and IEEE, December 1959, pp. 225–232.

[3] J. Ullmann, "Reduction of the storage requirements of bledsoe and browning's n-tuple method of pattern recognition," *Pattern Recognition*, vol. 3, pp. 297–306, 1971.

[4] J. Austin, "A review of ram based neural networks," in *Fourth International Conference on Micro-electronis for Neural Networks and Fuzzy Systems*. IEEE, 1994, pp. 58–66.

[5] ——, *RAM-Based Neural Networks*. USA: World Scientific Publishing Co., Inc., 1998, vol. 9.

[6] G. Tattersall, S. Foster, and P. Linford, "Single layer look-up perceptrons," in *1989 First IEE International Conference on Artificial Neural Networks*, vol. 313. IET, 1989, pp. 148–152.

[7] T. Ludermir, A. de Carvalho, A. Braga, and M. Souto, "Weightless neural models: A review of current and past works," *Neural Computing Surveys*, vol. 2, pp. 41–61, 01 1999.

[8] R. M. Haralick, "Dependence," *Pattern Recognition*, vol. 124, pp. 2–20, june 2019.

[9] ——, "The table look-up rule," *Communications in Statistics-Theory and Methods*, vol. 5, no. 12, pp. 1163–1191, 1976.

[10] T. K. Ho, J. Hull, and S.N.Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 66–75, 1994.

[11] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[12] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[13] N. Allinson and A. Kolcz, "A principled approach to n-tuple recognition systems," in *IEE Colloquium Pattern Recognition*. IET, 1997, pp. 2/1–2/10.

[14] A. Kolcz and N. M. Allinson, "N-tuple regression network," *Neural networks : the official journal of the International Neural Network Society*, vol. 9 5, pp. 855–869, 1996.

[15] I. Aleksander, W. Thomas, and P. Bowden, "Wisard a radical step forward in image recognition," *Sensor review*, vol. 4, no. 3, pp. 120–124, 1984.

[16] M. De Gregorio and M. Giordano, "Background modeling by weightless neural networks," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 493–501.

[17] W. Highleyman and L. Kamentsky, "Comments on a character recognitions method of bledsoe and browning," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 2, p. 263, 1960.

[18] W. H. Highleyman, "An analog method for character recognition," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 502–512, 1961.

[19] W. Bledsoe, "Further results on the n-tuple pattern recognition method," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 1, pp. 96–96, 1961.

[20] W. Bledsoe, J. Bomba, I. Browning, R.J.Evey, R. Kirsch, R. Mattson, M. Minsky, U.Neisser, and O. Selfridge, "Discussion of problems in pattern recognition," in *Eastern Joint Computer Conference*. ACM and IEEE, December 1959, pp. 233–237.

[21] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[22] W. Bledsoe and C. Bisson, "Improved memory matrices for the n-tuple pattern recognition method," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 3, pp. 414–415, 1962.

[23] P. Lewis, "Approximating probability distributions to reduce storage requirements," *Information and Control*, vol. 2, no. 3, pp. 214 – 225, 1959. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995859902074

[24] S. Kullback and R. Leibler, "On information adn sufficiency," *Annals Of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[25] J. Shore and R. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Transactions on Information Theory*, vol. IT-26, pp. 26–37, 1980.

[26] S. Kullback, *Information Theory and Statistics*. John Wilesy and Sons, 1959.

[27] S. Lauritzen, *Graphical Models*. Clarendon Press, 1997.

[28] D. Koller and N. Friedman, *Probabilistic Graphical Models*. MIT Press, 2009.

[29] J. Whitaker, *Graphical Models in Applied Multivariate Statistics*. John Wiley, 1990.

[30] C. Chow and C. Liu, "Approximating discrete probbility distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[31] S. Wong and F. Poon, "Comments on approximating discrete probability distributions with dependence trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 3, pp. 333–335, 1989.

[32] J. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[33] F. Malvestuto, "Existence of extnesions and product extensions for discrete probability distributions," *Discrete Mathematics*, vol. 69, pp. 61–77, 1988.

[34] P. Spirtes, C. Blymour, and R. Sheines, *Causation, Prediction, and Search*. Springer-Verlag, 1993, vol. 81.

[35] T. B. Berrett, Y. Wang, R. F. Barber, and R. Samworth, "The conditional permutation test for independence while controlling for confounders," *Journal Royal Statisical Society B*, vol. 82, pp. 175–197, 2020.

[36] T.-M. Huang, "Testing conditional independence using maximal nonlinear conditional correlation," *The Annals of Statistics*, vol. 38, no. 4, pp. 2047–2091, 2020.

[37] P. Boland, "Majority systems and the condorcet jury theorem," *Statistician*, vol. 38, pp. 181–189, 1989.

[38] H. Peng, C. Lin, L. Luo, and Q. Zhou, "Accuracy of classifier combining based on majority voting," in *IEEE International Conference on Control and Automation*, 2007, pp. 2654–2658.

[39] L. Xu, A. Kryzak, and C. Suen, "Methods of combining multiple classifiers and their application to handrwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, pp. 418–435, 1992.

[40] A. Narasimhamurthy, "Theoretical bounds of majority voting performance for a binary a classification problem," *IEEE Transactions on Pattern Analysis and machine Intelligence*, vol. 27, no. 12, pp. 188–1995, 2005.

[41] L.I.Kuncheva, C.J.Whitaker, C. Shipp, and R. Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Analysis and Applications*, vol. 6, no. 1, pp. 22–31, 2003.

[42] L. Lam and C. Suen, "Application of majority voting to pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, no. 5, pp. 553–568, 1997.

[43] J. Kittler and F. Alkoot, "Sum versus vote fusion in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 110–115, 2003.

[44] J. R. Ullmann, "Experiments with the n-tuple method of pattern recognition," *IEEE Transactions on computers*, vol. 100, no. 12, pp. 1135–1137, 1969.

[45] P. Minchinton, J. Bishop, and R. Mitchell, "The minchinton cell-analogue input to the n-tuple net," *The Department of Cybernetics, The University of Reading, UK*, 1989.

[46] R. Mitchell, P. Minchinton, J. Bishop, J. Day, J. Hawker, and S. Box, "Modular hardware weightless neural networks," in *Proc. Weightless Neural Networks Workshop*, vol. 93, 1993, pp. 123–128.

[47] R. Mitchell, J. Bishop, S. Box, and J. Hawker, "Comparison of methods for processing grey level data in weightless networks," in *Proc. of Weightless Neural Network Workshop WNNW95, Kent at Canterbury, UK*, 1995, pp. 76–81.

[48] K. De Meyer, "Foundations of stochastic diffusion search," Ph.D. dissertation, University of Reading, 2004.

[49] I. Alexsander, *An Introduction to Neural Computing*. Chapman and Hall, 1990.

[50] S. Vitabile, V. Conti, F. Gennaro, and F. Sorbello, "Efficient mlp digital implementation on fpga," in *8th Euromicro Conference on Digital System Design (DSD'05)*. IEEE, 2005, pp. 218–222.

[51] I. Aleksander and W.-K. Kan, *A Probabilistic Logic Neuron Network for Associative Learning*. Cambridge, MA, USA: MIT Press, 1989, p. 156–171.

[52] E. Burattini, M. De Gregorio, and G. Tamburrini, "Mental imagery in explanations of visual object classification," in *Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks*, 2000, pp. 137–143.

[53] B. P. Grieco, P. M. Lima, M. De Gregorio, and F. M. França, "Extracting fuzzy rules from "mental" images generated by modified wisard perceptrons," in *Proc. E*, vol. 26, 2008, pp. 101–773.

[54] ——, "Producing pattern examples from "mental" images," *Neurocomputing*, vol. 73, no. 7-9, pp. 1057–1064, 2010.

[55] M. De Gregorio and M. Giordano, "The wisard classifier," *ESANN 2016 - 24th European Symposium on Artificial Neural Networks*, pp. 447–452, 2016.

[56] M. D. Gregorio and M. Giordano, "A wisard-based approach to cdnet," in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, 2013, pp. 172–177.

[57] D. O. Cardoso, J. Gama, and F. M. França, "Weightless neural networks for open set recognition," *Machine Learning*, vol. 106, no. 9-10, pp. 1547–1567, 2017.

[58] D. Jung, M. Krishnamoorthy, G. Nagy, and A. Shapira, "N-tuple features of ocr revisited," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 734–745, 1996.

[59] A. Shapira, "Experiments on the generation of distinguishing n-tuples for selected character dichotomies," Rensselaer Polytchnic Institute, Tech. Rep. ECSE-OCR-18DEC95, 1995.

[60] D. Cavicchio, *Adaptive search using simulated evolution*. Computer Science, University of Michigan, 1970. [Online]. Available: https://books.google.com/books?id=DFXoHAAACAAJ

[61] J. Bishop, A. Crowe, P. Michinton, and R. Mitchell, "Evolutionary learning to optimise mapping in n-tuple networks," in *IEE Colloquium on Machine Learning*. IET, 1990, pp. 3–1.

[62] A. Badii, M. Binstead, A. J. Jones, T. Stonham, and C. L. Valenzuela, "10 applications of n-tuple sampling and genetic algorithms to speech recognition," *Neural Computing Architectures: The design of brain-like machines*, p. 172, 1989.

[63] H. B. Azhar and K. Dimond, "A stochastic search algorithm to optimize an n-tuple classifier by selecting its inputs," in *International Conference Image Analysis and Recognition*. Springer, 2004, pp. 556–563.

[64] M. Giordano and M. De Gregorio, "An evolutionary approach for optimizing weightless neural networks." in *ESANN*, 2019.

[65] R. J. Roy and J. Sherman, "Two viewpoints of k-tuple pattern recognition," *IEEE Transactions on Systems Science and Cybernetics*, vol. 3, no. 2, pp. 117–120, 1967.

[66] L. Kamentsky and C. Liu, "A theoretical and experimental study of a model for pattern recognition," in *Computer and Information Sciences*. Spartan, 1964, pp. 194–218.

[67] G. P. Steck, "Stochastic model for the browning-bledsoe pattern recognition scheme," *IRE Transactions on Electronic Computers*, no. 2, pp. 274–282, 1962.

[68] J. Schürmann, "A multifont word recognition system for postal address reading," *IEEE Transactions on Computers*, vol. C-27, no. 5, pp. 721–732, 1978.

[69] I. Aleksander, T. Clarke, and A. Braga, "Binary neural systems: combining weighted and weightless properties," *Intelligent Systems Engineering*, vol. 3, no. 4, pp. 211–221, 1994.

[70] C. Linneberg and T. M. Joergensen, "Cross-validation techniques for n-tuple-based neural networks," in *Ninth Workshop on Virtual Intelligence/Dynamic Neural Networks*, vol. SPIE 3728. International Society for Optics and Photonics, 1999, pp. 266–277.

[71] E. C. D. B. C. Filho, M. C. Fairhurst, and D. L. Bisset, "Analysis of saturation problem in ram-based neural networks," *Electronics Letters*, vol. 28, no. 4, pp. 345–347, 1992.

[72] D. S. Carvalho, H. C. Carneiro, F. M. França, and P. M. Lima, "B-bleaching: Agile overtraining avoidance in the wisard weightless neural classifier." in *ESANN*, 2013.

[73] P. Coutinho, H. C. Carneiro, D. S. Carvalho, and F. M. França, "Extracting rules from drasiw's" mental images"." in *ESANN*, 2014.

[74] F. França, M. De Gregorio, P. Lima, and W. De Oliveira, "Advances in weightless neural systems," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2014, pp. 497–504.

[75] A. Kappaun, K. Camargo, F. Rangel, F. Firmino, P. M. V. Lima, and J. Oliveira, "Evaluating binary encoding techniques for wisard," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2016, pp. 103–108.

[76] H. C. C. Carneiro, C. E. Pedreira, F. M. G. França, and P. M. V. Lima, "The exact vc dimension of the wisard n-tuple classifier," *Neural Computation*, vol. 31, no. 1, pp. 176–207, 2019, pMID: 30462587. [Online]. Available: https://doi.org/10.1162/neco_a_01149

[77] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[78] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[79] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[80] L. Santiago, L. Verona, F. Rangel, F. Firmino, D. S. Menasché, W. Caarls, M. Breternitz Jr, S. Kundu, P. M. Lima, and F. M. França, "Weightless neural networks as memory segmented bloom filters," *Neurocomputing*, 2020.

[81] J. F. Nazuno, A. V. Nava, and E. V. Medina, "Tridimensional pattern reconstruction by using weightless artificial neural networks," in *International Workshop on Neural Networks Applied to Control and Image Processing*, November 7-10 1994.

[82] T. J. Stonham, *Practical Face Recognition and Verification with Wisard*. Dordrecht: Martinas Nijhoff Netherlands, 1986, pp. 426–441. [Online]. Available: https://doi.org/10.1007/978-94-009-4420-6_44

[83] K. M. Khaki, "Weightless neural networks for face recognition," Ph.D. dissertation, Brunel University School of Engineering and Design PhD Theses, 2013.

[84] V. Conti, L. Rundo, C. Militello, G. Mauri, and S. Vitabile, "Resource-efficient hardware implementation of a neural-based node for automatic fingerprint classification." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 8, no. 4, pp. 19–36, 2017.

[85] M. Sixsmith, "Speech recognition using n-tuple techniques." *British Telecom technology journal*, vol. 8, pp. 50–60, 1990.

[86] M. De Gregorio and M. Giordano, "Change detection with weightless neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 403–407.

[87] C. B. Do Prado, F. M. Franca, E. Costa, and L. Vasconcelos, "A new intelligent systems approach to 3d animation in television," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, 2007, pp. 117–119.

[88] C. B. Do Prado, F. M. França, R. Diacovo, and P. M. Lima, "The influence of order on a large bag of words," in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, vol. 1. IEEE, 2008, pp. 432–436.

[89] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue, "Automated multi-label text categorization with vg-ram weightless neural networks," *Neurocomputing*, vol. 72, no. 10-12, pp. 2209–2217, 2009.

[90] C. S. Gonçalves, F. T. Pedroni, and A. F. De Souza, "Multi-label text classification using vg-ram weightless neural networks," in *2008 10th Brazilian Symposium on Neural Networks*, 2008, pp. 105–110.

[91] J. C. M. Oliveira, K. V. Pontes, I. Sartori, and M. Embiruçu, "Fault detection and diagnosis in dynamic systems using weightless neural networks," *Expert Systems with Applications*, vol. 84, pp. 200–219, 2017.

[92] M. De Gregorio and M. Giordano, "Cwisardh $^{+}$: Background detection in rgbd videos by learning of weightless neural networks," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 242–253.

[93] ——, "Background estimation by weightless neural networks," *Pattern Recognition Letters*, vol. 96, pp. 55–65, 2017.

[94] D. O. Cardoso, D. S. Carvalho, D. S. Alves, D. F. Souza, H. C. Carneiro, C. E. Pedreira, P. M. Lima, and F. M. França, "Financial credit analysis via a clustering weightless neural classifier," *Neurocomputing*, vol. 183, pp. 70–78, 2016.

[95] N. Mpofu, "Forecasting stock prices using a weightless neural network," *Journal of Sustainable Development in Africa*, vol. 8, no. 1, pp. 115–119, 2006.

[96] J. Alhassan and S. Misra, "Using a weightless neural network to forecast stock prices: A case study of nigerian stock exchange," *Scientific Research and Essays*, vol. 6, no. 14, pp. 2934–2940, 2011.

[97] A. F. De Souza, F. D. Freitas, and A. G. C. de Almeida, "High performance prediction of stock returns with vg-ram weightless neural networks," in *2010 IEEE Workshop on High Performance Computational Finance*. IEEE, 2010, pp. 1–8.

[98] R. Cheruku, D. R. Edla, V. Kuppili, R. Dharavath, and N. R. Beechu, "Automatic disease diagnosis using optimised weightless neural networks for low-power wearable devices," *Healthcare technology letters*, vol. 4, no. 4, pp. 122–128, 2017.

[99] M. Simões, C. Amaral, F. França, P. Carvalho, and M. Castelo-Branco, "Applying weightless neural networks to a p300-based brain-computer interface," in *World Congress on Medical Physics and Biomedical Engineering 2018*, L. Lhotska, L. Sukupova, I. Lacković, and G. S. Ibbott, Eds. Singapore: Springer Singapore, 2019, pp. 113–117.

[100] D. F. de Souza, F. M. França, and P. M. Lima, "Real-time music tracking based on a weightless neural network," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*. IEEE, 2015, pp. 64–69.

[101] Q. Yao, D. Beetner, D. C. Wunsch, and B. Osterloh, "A ram-based neural network for collision avoidance in a mobile robot," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 4. IEEE, 2003, pp. 3157–3160.

[102] P. Coraggio and M. De Gregorio, "Wisard and nsp for robot global localization," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 2007, pp. 449–458.

[103] M. Staffa, M. Giordano, and F. Ficuciello, "A wisard network approach for a bci-based robotic prosthetic control," *International Journal of Social Robotics*, vol. 12, no. 3, pp. 749–764, 2020.

[104] I. Aleksander, "Capturing consciousness in neural systems," *Artificial Neural Networks, 2. Proc. ICANN*, vol. 92, pp. 17–22, 1992.

[105] ——, *The world in my mind, my mind in the world*. Andrews UK Limited, 2013.

[106] I. Aleksander, M. De Gregorio, F. M. G. França, P. M. V. Lima, and H. Morton, "A brief introduction to weightless neural systems." in *ESANN*. Citeseer, 2009, pp. 299–305.

[107] D. Chan, S. Hockaday, R. D. Tillett, and L. G. Ross, "Factors affecting the training of a wisard classifier for monitoring fish underwater." in *BMVC*. Citeseer, 1999, pp. 1–12.

[108] G. C. De Lello, J. F. Caldeira, M. Aredes, F. M. França, and P. M. Lima, "Weightless neural networks applied to nonintrusive load monitoring," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2020, pp. 844–851.

[109] L. A. Lusquino Filho, L. F. Oliveira, H. C. Carneiro, G. P. Guarisa, A. Lima Filho, F. M. França, and P. M. Lima, "A weightless regression system for predicting multi-modal empathy," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)(FG)*, 2020, pp. 554–558.

[110] F. M. de Paula Neto, T. B. Ludermir, W. R. de Oliveira, and A. J. da Silva, "Solving np-complete problems using quantum weightless neuron nodes," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2015, pp. 258–263.

[111] P. M. Xavier, M. De Gregorio, F. M. França, and P. M. Lima, "Detection of elementary particles with the wisard n-tuple classifier," in *ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligenceand Machine Learning*. i6doc.com, 2020, pp. 643–648.

[112] R. G. Barbastefano, M. C. Lippi, and D. Carvalho, "Process mining classification with a weightless neural network," *arXiv preprint arXiv:2009.12416*, 2020.

[113] V. A. Torres, B. R. Jaimes, E. S. Ribeiro, M. T. Braga, E. H. Shiguemori, H. F. Velho, L. C. Torres, and A. P. Braga, "Combined weightless neural network fpga architecture for deforestation surveillance and visual navigation of uavs," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103227, 2020.

[114] D. Cardoso, F. França, and J. Gama, "A bounded neural network for open set recognition," in *International Joint Conference on Neural Networks*, 07 2015, pp. 1–7.

[115] R. Rohwer and M. Morciniec, "The theoretical and experimental status of the n-tuple classifier," *Neural Networks*, vol. 11, no. 1, pp. 1–14, 1998.

[116] T. York and P. Duggan, "Processing of tomographic data using weightless neural networks," *Journal of Intelligent Systems*, vol. 7, no. 3-4, pp. 349–366, 1997.

[117] R. J. Mitchell and D. A. Keating, "Neural network control of a simple mobile robot," in *Concepts for Neural Networks*. Springer, 1998, pp. 95–107.

[118] M. H. B. Azhar and K. R. Dimond, "Design of an fpga based adaptive neural controller for intelligent robot navigation," in *Proceedings Euromicro Symposium on Digital System Design. Architectures, Methods and Tools*. IEEE, 2002, pp. 283–290.

[119] M. C. W. Keat, R. Abdullah, R. A. Salam, and A. A. Latif, "Weightless neural network array for protein classification," in *International Conference on Parallel and Distributed Computing: Applications and Technologies*. Springer, 2004, pp. 168–171.

[120] M. C. P. de Souto, T. B. Ludermir, and W. R. de Oliveira, "Equivalence between ram-based neural networks and probabilistic automata," *IEEE transactions on neural networks*, vol. 16, no. 4, pp. 996–999, 2005.

[121] E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, F. Pedroni, and A. Souza, "Intelligent classification of economic activities from free text descriptions," in *V Workshop em Tecnologia da Informaçao e da Linguagem Humana-TIL*, 2007.

[122] A. J. Arguelles-Cruz, I. López-Yáñez, M. Aldape-Pérez, and N. Conde-Gaxiola, "Alpha-beta weightless neural networks," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2008, pp. 496–503.

[123] A. De Souza, C. Badue, F. Pedroni, E. Oliveira, S. Dias, H. Oliveira, and S. F. de Souza, "Face recognition with vg-ram weightless neural networks," in *ICANN Lecture Notes in Computer Science*, vol. Springer Lecture Notes in Computer Science 5163, 09 2008, pp. 951–960.

[124] H. C. Carneiro, F. M. França, and P. M. Lima, "Wann-tagger-a weightless artificial neural network tagger for the portuguese language," in *International Conference on Neural Computation*, vol. 2. SciTePress, 2010, pp. 330–335.

[125] ——, "Multilingual part-of-speech tagging with weightless neural networks," *Neural Networks*, vol. 66, pp. 11–21, 2015.

[126] W. R. de Oliveira, "Quantum ram based neural netoworks." in *ESANN*, vol. 9, 2009, pp. 331–336.

[127] D. O. Cardoso, F. M. França, and J. Gama, "Wcds: A two-phase weightless neural system for data stream clustering," *New Generation Computing*, vol. 35, no. 4, pp. 391–416, 2017.

[128] J. Gryak, R. M. Haralick, and D. Kahrobaei, "Solving the conjugacy decision problem via machine learning," pp. 2–20, 2020.

[129] V. C. Ferreira, A. S. Nery, L. A. Marzulo, L. Santiago, D. Souza, B. F. Goldstein, F. M. França, and V. Alves, "A feasible fpga weightless neural accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[130] D. F. Specht, "A genral regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[131] L. L. Filho, L. F. R. Oliveira, A. L. Filho, G. P. Guarisa, P. Lima, and F. M. G. França, "Prediction of palm oil production with an enhanced n-tuple regression network," in *ESANN*, 2019.

[132] A. L. Filho, G. P. Guarisa, L. F. Oliveira, F. M. Franca, P. Lima *et al.*, "wisardpkg–a library for wisard-based models," *arXiv preprint arXiv:2005.00887*, 2020.

[133] L. A. Lusquino Filho, L. F. Oliveira, A. Lima Filho, G. P. Guarisa, L. M. Felix, P. M. Lima, and F. M. França, "Extending the weightless wisard classifier for regression," *Neurocomputing*, 2020.

[134] R. N. Rocha, L. Leopoldo Filho, M. Aredes, F. M. França, and P. M. Lima, "Regression wisard application of controller on dc statcom converter under fault conditions," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2020, pp. 860–867.

[135] R. S. Sousa, P. G. dos Santos, T. M. Veras, W. R. de Oliveira, and A. J. da Silva, "Parametric probabilistic quantum memory," *Neurocomputing*, 2020.

[136] J. C. Spall, "Simultaneous perturbation stochastic approximation," *Introduction to stochastic search and optimization: Estimation, simulation, and control*, pp. 176–207, 2003.

[137] A. J. da Silva, W. R. de Oliveira, and T. B. Ludermir, "Training a classical weightless neural network in a quantum computer," *Quantum*, vol. 1, no. 1, p. 1, 2014.

[138] A. Silva, W. de Oliveira, and T. Ludermir, "A weightless neural node based on a probabilistic quantum memory," in *2010 Eleventh Brazilian Symposium on Neural Networks*. IEEE, 2010, pp. 259–264.

[139] A. Silva, T. Ludermir, and W. Oliveira, "Single-shot learning algorithm for quantum weightless neural networks," *ChemBioChem*, pp. 1–6, 2016.

[140] F. M. de Paula Neto, W. R. de Oliveira, T. B. Ludermir, and A. J. da Silva, “Chaos in a quantum neuron: An open system approach,” *Neurocomputing*, vol. 246, pp. 3 – 11, 2017, brazilian Conference on Intelligent Systems 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217302394

[141] S. B. Ramezani, A. Sommers, H. K. Manchukonda, S. Rahimi, and A. Amirlatifi, “Machine learning algorithms in quantum computing: A survey,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.