Fast Regular and Interval-based Classification, using parSITs

Balázs Tusor[1], Annamária R. Várkonyi-Kóczy[2]

[1]Doctoral School of Applied Informatics and Applied Mathematics
Óbuda University, Budapest, Hungary; E-mail: tusor.balazs@kvk.uni-obuda.hu
[2]Institute of Automation, Óbuda University, Budapest, Hungary
E-mail: varkonyi-koczy@uni-obuda.hu

Abstract: Parallelized Sequential Indexing Tables (parSITs), are classifiers that have been developed for the processing of large volumes of data rapidly. Their base idea is implementing a sequential indexing table structure with parallelization techniques, using a sequence of Lookup Tables in order to build a chain of value combinations. Although the inference (evaluation) method, that it was originally developed for, is very fast, its performance significantly depends on the arbitrary order of the attributes, in multi-class cases, thus, reducing its classification performance. In this work, we introduce a new inference method, that increases the classification performance of the classifier, at the cost of a small increase in computational complexity.

Keywords; Big Data classification; interval-based classification; parallel computing; sequential indexing tables; lookup tables; machine learning

1 Introduction

Machine learning has been one of the most important areas of computer science in the past couple of decades. Numerous systems have been developed to address many different kinds of machine learning problems in a wide range of scientific fields. To mention a few recent developments, machine learning has been successfully used in biomedical engineering [1] [2], spam email detection [3], movement detection [4], the recognition of electromyographic hand gesture signals for prosthetic hand control [5], Big Data systems [6] [7] [8], etc.

Parallelized Sequential Indexing Tables (parSITs) are classifiers that have been developed for processing large volumes of data rapidly. Their base idea is implementing a sequential indexing table structure [9] [10] with parallelization techniques, using a sequence of Lookup Tables [11] in order to build a chain of value combinations, describing the data extracted from the training data in a compact way, organized into a layered structure where each layer takes care of a

given dimension of the problem space (i.e. a given attribute from the training data).

In previous work, we developed a training algorithm for the parSIT classifier [12] [13], along with a simple inference (evaluation) method that focuses on finding the index that is the closest to the given input value for each attribute. However, although this leads to a very fast inference, its performance also significantly depends on the arbitrary order of the attributes (which directly influences the structure itself). This has the disadvantage that for input samples that are very similar to a learned sample in all but one attribute, if that attribute is situated high in the structure, the classifier has a high chance to misclassify the sample, or not recognize it at all.

In order to solve this problem, in this paper a new inference method is presented for the parSIT that uses a different approach: instead of choosing the closest values, it evaluates all values within a given range. Although this results in a higher computational complexity inference and a higher implementation complexity, but it is shown that it also boosts the classification performance significantly.

The rest of this paper is structured as follows. In Section 2 the parSIT classifier is described alongside the proposed new method: In Subsection 2.1 the general architecture is presented, while Subsections 2.2 and 2.3 briefly summarize the training procedure and the proximity-based inference algorithm, respectively; and finally, in Subsection 2.4 the new interval-based inference algorithm is proposed in detail. Section 3 illustrates the classification performance of the new method in Subsection 3.1 compared to the proximity-based inference, then complexity analysis is given in Subsection 3.2. Finally, Section 4 concludes the paper and presents some future work possibilities.

2 Parallelized Sequential Indexing Tables

2.1 General Architecture

The parSIT classifier builds and maintains a layered structure that models the problem space based on the data used for its training. The structure built from the data of an N-dimension classification problem (thus, the data having N data attributes and 1 class attribute) consists of N+1 layers, where the first N layers handle the attributes of the data, each one regarding the values of corresponding attribute of the input data. Fig. 1 (a) shows an example for a trained network that has been built using training dataset X (shown in (b)).

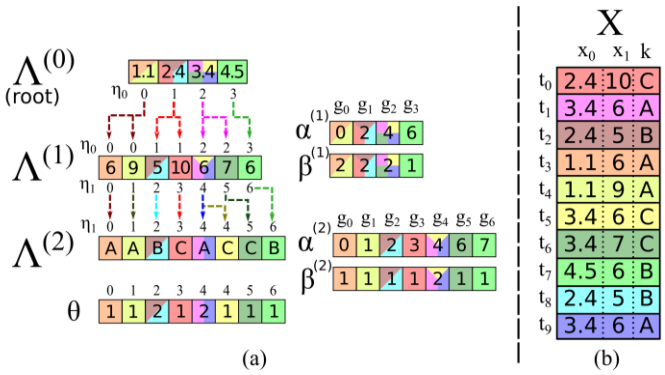

Figure 1

The general architecture of ParSITs (a) trained on a given training dataset (b)

The array elements are colored accordingly to the training sample ($t_0..t_9$) that it represents. The first "*root*" layer contains a 1D index array $\Lambda^{(0)}$, which stores all different input values gained from the first attribute (x_0), sorted and free of duplicates. The elements in the index arrays are called markers in the following. The layers are connected so for each trained data tuple, an implicit "route" can be followed in order to gain the class label in the last layer that is associated with the given tuple.

In Fig. 1, it can be seen that x_0 has 4 different values in the training dataset, thus, the size of the index array $\Lambda^{(0)}$ is 4. The first element ($\Lambda_0^{(0)} = 1.1$) belongs to training tuples t_3 and t_4. Since these two tuples have different values in their second attribute (x_1), in the second layer they take up two different array elements (given by their values: '6' and '9'). However, since they share the same first attribute value, they make up a *group* in the second layer, which is addressed by the position of the marker of their first value in the previous layer (η_0). Each group in a given layer is accounted for by storing their starting locations in a given 1D array $\alpha^{(i)}$ (where i denotes which layer it belongs to) and the number of markers in each group in 1D array $\beta^{(i)}$. For example, the starting location of the group (g_0) of the mentioned 2 markers in the second layer is $\alpha^{(1)}$=0, while it contains $\beta^{(1)}$=2 markers. This way, the evaluation is faster, since not all of the markers are need to be regarded, only the groups marked by the significant markers in the layer before it. In the root layer, there are no additional arrays, since all markers are observed.

The last (class) layer contains one index array $\Lambda^{(N)}$ that contains the class labels, and an occurrence array Θ that accounts for how many times each given value sequence has been seen during the training. Occurrence values that are higher than one ($\Theta_j > 1$, for all j elements) indicate redundancy in the training data, while groups in $\Lambda^{(N)}$ that have more than one markers in them indicate inconsistency in the training data (i.e. there are two or more tuples that share the same attribute values, but differ in class label).

2.1 Training Algorithm

The training of the parSIT structure builds the Λ, α and β arrays for each layer, one layer at a time. It uses parallel computing to sort the values of the given attribute, then build a temporary array *H*, and remove the redundant elements to gain the compact representation of the given attribute. Let *P* denote the number of training tuples.

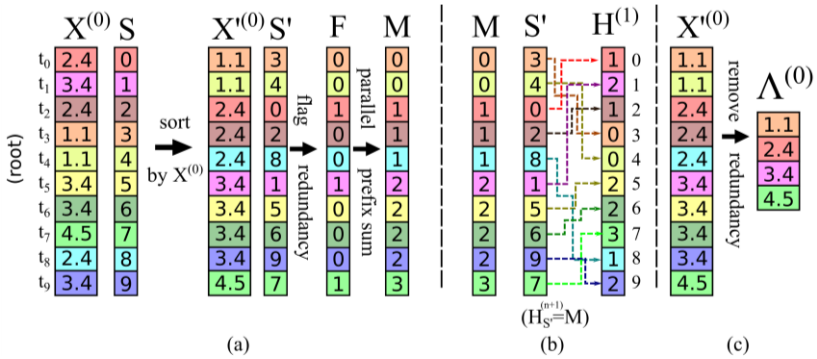

Figure 2

The training of the first layer of the structure shown in Fig. 1

Fig. 2 shows the training procedure for the root layer, processing attribute 0. The attribute values of $X^{(0)}$ (a column array taken from x_0) is sorted alongside a simple in an increasing sequence array *S* (containing values from 0 to *P*-1), using $X^{(0)}$ as key. The sorting results in $X'^{(0)}$ and *S'*. After that, the redundant elements in $X'^{(0)}$ are flagged in flag array *F* for $\forall p \in [0, P-1]$:

$$F_p = \begin{cases} 1, & if\ X^{(0)}_{p-1} = X^{(0)}_p\ and\ p \neq 0 \\ 0, & otherwise \end{cases} \quad (1)$$

Then, a parallel computing technique called parallel prefix sum (PPS, [14]) is used on array *F* to gain array *M*, which indicates the new place of each attribute value in the reduced array Λ:

$$\Lambda^{(0)}_{M_p} = \begin{cases} x^{(0)}_0 & if\ p = 0 \\ x^{(0)}_p & if\ p \neq 0\ and\ F_p = 1 \end{cases} \quad (2)$$

The size of Λ is determined from the last value of *M*:

$$m = M_{P-1} + 1 \quad (3)$$

Furthermore, by using *S'* to rearrange *M*, we gain the temporary array *H* that is used to distinguish the groups in the next ((i+1)th) layer:

$$H^{(i+1)}_{S_p} = M_p, \forall p \epsilon [0, P-1] \quad (4)$$

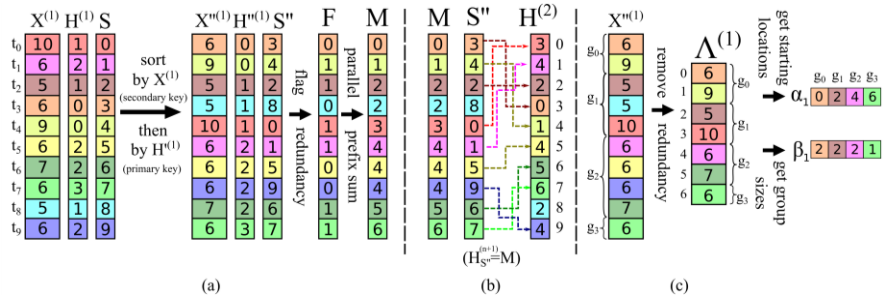

Figure 3

The training of the second layer of the structure shown in Fig. 1

The training of the rest of the layers ($i > 0$) is similar (shown in Fig. 3), with the difference that H is also used to sort the data: $X^{(i)}$, $H^{(i)}$ and S are all sorted by H first, then X second, thus, gaining an ordering ($X''^{(i)}$, $H''^{(i)}$ and S'') where each *group* is represented in the order of their "parent" markers in the previous layer (i.e. the value that they share in their previous (i - 1) attribute), and the markers within each group are sorted as well. Thus, H is indicating their *group number*.

Similarly, to the root layer, in the next steps the flag array F is determined:

$$F_p = \begin{cases} 1, & if\ x''^{(i)}_{p-1} = x''^{(i)}_{p}\ and\ H''^{(i)}_{p-1} = H''^{(i)}_{p} \\ 0, & otherwise \end{cases} \tag{5}$$

which is used to create marker placement array M by applying parallel prefix sum on array F. Array $H^{(i+1)}$ is calculated using Eq. (4) and the compact index array is determined:

$$\Lambda^{(i)}_{M_p} = \begin{cases} x^{(i)}_0 & if\ p = 0 \\ x^{(i)}_p & if\ p \neq 0\ and\ F_p = 1 \end{cases} \tag{6}$$

After that, all non-layers ($i > 0$) calculate the starting locations ($\alpha^{(i)}$) and the group sizes ($\beta^{(i)}$) for each group j:

$$\alpha^{(i)}_j = min\{k \mid \eta_k = j\} \tag{7}$$

$$\beta^{(i)}_j = \begin{cases} \alpha^{(n)}_{j+1} - \alpha^{(n)}_j, & if\ i < g-1 \\ m - \alpha^{(n)}_j, & if\ i = g-1 \end{cases} \tag{8}$$

In the last layer, the algorithm ends after the occurrence array Θ is calculated.

2.2 Proximity-based Inference

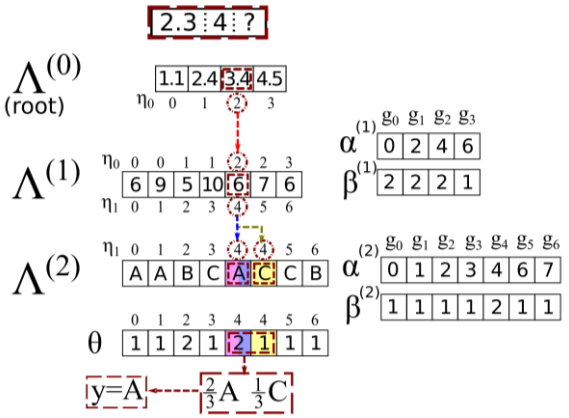

Figure 4

An illustration for the proximity-based inference algorithm for a single input sample

Fig. 4 depicts the proximity-based inference algorithm. In the root layer, it simply polls each marker $\Lambda_i^{(0)}$ for each input sample value $\mathrm{x}_p^{(0)}$, i.e. calculates their distances, and chooses the index number of the marker that is the closest to the given input value:

$$H_p = \mathrm{argmin}_{\forall i}\left(d\left(\Lambda_i^{(0)} - \mathrm{x}_p^{(0)}\right)\right) \tag{9}$$

where $d(\ )$ is a suitable distance measure. In this research, we use the Euclidian distance, but various other distances can be used, such as a Gaussian function. marker is chosen for each input tuple p ($p \in [0, P-1]$), and only the corresponding *groups* are polled in the next layer:

$$H_p = \mathrm{argmin}_{\forall i}\left(d\left(\Lambda_{\upsilon_p+i}^{(n)} - \mathrm{x}_p^{(n)}\right)\right) \tag{10}$$

where υ_p is the starting position of the group that belongs to the chosen marker H_p in the previous layer for each tuple p:

$$\upsilon_p = \alpha_{H_p} \tag{11}$$

Finally, the output of the proximity-based inference is an array y given by the indices in the last layer:

$$y = \Lambda_z^{(N-1)} \tag{12}$$

where the index is the most regularly occurring class that belongs to the marker chosen in the previous layer:

$$z = \mathrm{argmax}_{\forall i}\left(\Theta_{\upsilon_p+i}\right) \tag{13}$$

2.2 Interval-based Inference

As it was mentioned in the introduction, the proximity-based inference method is very fast, but has a disadvantage that it focuses on only one marker in each layer. For example, in the structure shown in Fig. 1, if the input tuple is t_{10}=[1.5 10], then, the closest learned tuple should be [2.4 10] (as their Euclidian distance is $d(t_9,t_{10})$=0.9), but the inference will choose [1.1 9] ($d(t_3,t_{10})$ = 1.07), because 1.5 is closer to 1.1 than to 2.4. This often leads to misclassification, decreasing the classification accuracy of the system.

While the proximity-based inference described in the previous subsection focuses on quickly finding the value that is the closest to the currently examined input value for each input sample, the interval-based inference *investigates the area around the known values as well*, thus, can regard multiple values for each input sample.

Fig. 5 illustrates the inference algorithm for a single input [2 4], where the ρ ranges of each attribute are defined at [1.5 3] (i.e. for the first attribute, the interval [0.5, 3.5] is investigated, while for the second attribute the interval [1, 7] is regarded). In the figure, each "route" is color-coded to make the paths of the inference more easily discernable.

In the root layer (L_0), the first 3 markers are polled as positive (i.e. being part of the sought interval), so in the next layer (L_1), the groups linked to them (markers #0 to #5) are regarded, comparing their values to the interval of the second attribute. In the given example, only markers (η_1) #0, #2, #4 and #5 are polled as positive, so in the last (class) layer the class markers ($\Lambda^{(2)}$) and their occurrences (θ) are counted.

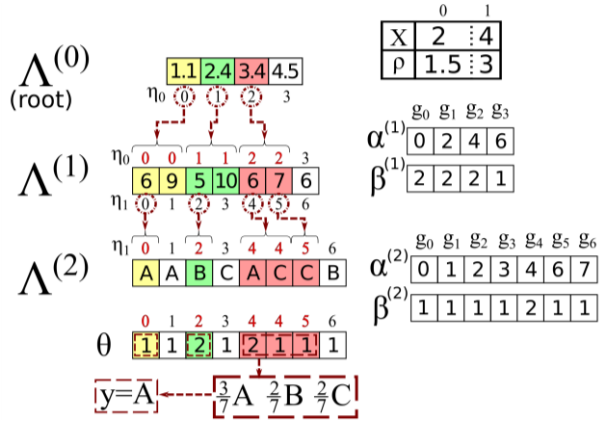

Figure 5

An illustration for the interval-based inference algorithm considering one input sample

The output of the system can be either the class with the highest occurrence rate, or the whole array of classes with their measured occurrence rates (as statistical information to enhance classification performance). If the inference stops before reaching the class layer (i.e. does not find any markers that are close enough to the input sequence), then the default class is returned (which has the highest overall occurrence rate).

Remark: An easy way to determine the range value for each attribute is determining the size of their domain (i.e. the largest value their attribute can take) and set the range to an arbitrary percentage of it.

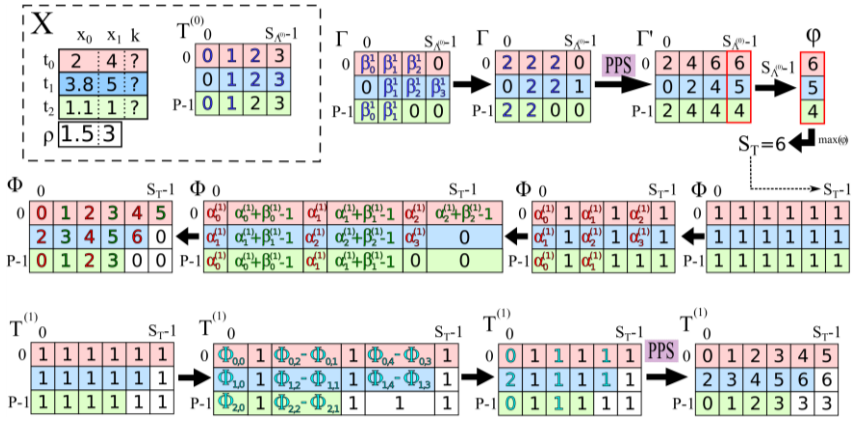

Figure 6

An illustrative example for the detailed steps of the interval-based inference algorithm processing the root layer of the structure in Fig. 5, considering 3 input samples (set *X*)

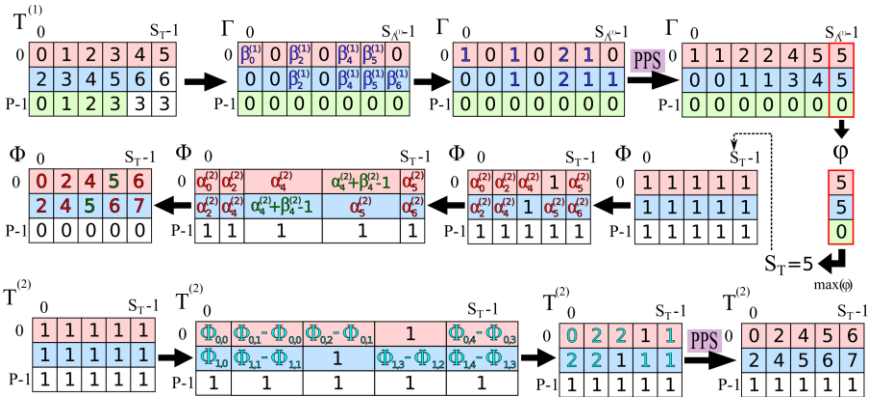

Figure 7

The continuation of the interval-based inference algorithm, processing the second and third layers of the structure in Fig. 5

A more detailed illustration for the interval-based inference algorithm is shows in Figures 6 and 7, where the input data set X (with 3 samples: t_0=[2 4], t_1=[3.8 5] and
t_2=[1.1 1]) is evaluated for the structure in Fig. 1, using range parameters ρ=[1.5 3]. For easier readability, each row is colored in accordance to their corresponding sample. Since the amount of positively polled markers is varied among the input samples, the coloring also indicates how many elements are used in each row. Fig. 6 shows the evaluation of the root layer, while Fig. 7 depicts its continuation for the rest of the structure.

The algorithm is implemented through matrices, where each row processes a given input sample. The goal in each layer is to determine which groups of markers have to be regarded in the next layer, and at the last layer, the determination of the class distribution among the selected markers, for each input sample.

Let us consider $T: P \times S_{\Lambda^{(0)}}$ 2D array in the beginning, where P is the number of the input samples and $S_{\Lambda^{(0)}}$ is the size of the 1D index array in the root layer.

T is used to store the list of the markers that is needed to be evaluated in the next layer. For the root layer, $T^{(i=0)}$ simply contains an increasing sequence of numbers from 0 to $S_{\Lambda^{(0)}} - 1$:

$$T_{p,j}^{(i=0)} = j : \forall j \in [0, S_{\Lambda^{(0)}} - 1], \forall p \in [0, P-1] \tag{14}$$

In order to determine the size of the array $T^{(i+1)}$ ($i > 0$), temporary 2D array Γ with size $P \times S_{\Lambda^{(i)}}$ is created in each given layer. It is initialized with zeros, then in each row p, a given element $\Gamma_{p,j}$ is set to $\beta_j^{(i)}$, if the corresponding $\Lambda_{T_{p,j}}^{(j)}$ marker value is within the ρ range of the given input attribute value $X_{p,i}$:

$$\Gamma_{p,T_{p,j}} = \begin{cases} \beta_{T_{p,j}}^{(i+1)} & \text{if } X_{p,i} - \rho \le \Lambda_{T_{p,j}}^{(j)} \text{ and } \Lambda_{T_{p,j}}^{(j)} \le X_{p,i} + \rho \\ 0 & otherwise \end{cases} \tag{15}$$

for all $j \in [0, S_{\Lambda^{(0)}} - 1]$ and all $p \in [0, P-1]$. In Figures 6 and 7, the modified cell values are marked with blue font color.

After that, parallel prefix sum (PPS) is done on Γ in order to collect the number of all elements (which are needed to be regarded in the next layer) for each row, gaining Γ'. As a result of the PPS step, the last column contains these values, which are collected in array φ (with size $P \times 1$):

$$\varphi_p = \Gamma'_{p,S_{\Lambda^{(i)}}-1} \tag{16}$$

for all $p \in [0, P-1]$, and the largest such number in S_T (which is the size of $T^{(i+1)}$):

$$S_T = \max_{\forall p} \varphi_p \tag{17}$$

In order to construct $T^{(i+1)}$, another temporary 2D array: Φ is created (with size $P \times S_T$). In Φ the index numbers of the markers (to be regarded in the next layer) are needed to be collected. However, this is not done by directly addressing the elements of Φ: instead, it is done indirectly, by using the elements in Γ that polled positively ($v(p,j) = \Gamma_{p,j}$) to set only the relevant starting positions: $\Phi_{p,v(p,j)}$. These are marked with red font color in Figures 6 and 7:

- If the first element in a row in Γ is polled positive (its value is bigger than 0), then the first element in the same row of Φ is set to 0, since that is the leftmost marker:

$$\Phi_{p,0} = \alpha_0^{(i+1)} \tag{18}$$

- If its value in Γ is 0 (i.e. polled negative), then the sought sequence does not start at 0. However, the starting position can still simply be determined from Γ' (using $v(p,j) = \Gamma'_{p,j}$):

$$\Phi_{p,v(p,j-1)} = \alpha_j^{(i+1)}, \text{ if } j > 0 \text{ and } \Gamma_{p,j} > 0 \tag{19}$$

Remark: it is important to note that at this point Γ only shows if an element polled positive or not, while the position information is taken from Γ'.

For example, in Fig. 6 the first row (p=0) of Γ is [2 2 2 0], while Γ' is [2 4 6 6]. This shows that only the first 3 elements polled positive, so the first element of the same row in Φ is set to 0, then the 2nd element (j=1) sets $\Phi_{0,v(0,j-1)} = \Phi_{0,v(0,0)} = \Phi_{0,2}$ to $\alpha_{j=1}^{(i=1)} = 2$; while the 3rd element (j=2) sets $\Phi_{0,v(0,j-1)} = \Phi_{0,v(0,1)} = \Phi_{0,4}$ to $\alpha_j^{(i+1)} = \alpha_2^{(1)} = 4$.

After that, the corresponding *ending locations* are set for each row p in Φ, for each positively polling element j in Γ:

$$\Phi_{p,\ v(p,j-1)+\beta_j^{(i+1)}-1} = \alpha_j^{(i+1)} + \beta_j^{(i+1)} - 1 \tag{20}$$

These elements are marked with green font color in Φ in Figures 6 and 7. If there are no groups that have more than 2 elements ($\beta_j^{(i+1)} > 2$), then Φ can be already used as $T^{(i+1)}$ in the next layer. However, if there are more than 2 elements in at least one group, then there will be "holes" (zeros) in the rows, which is why Φ is only a temporary array that is used to construct $T^{(i+1)}$.

With Φ set up, $T^{(i+1)}$ is initialized (with the same size as Φ), however, with ones. The elements of $T^{(i+1)}$ are calculated differently for the first element in each row p:

$$T_{p,0}^{(i+1)} = \Phi_{p,0} \tag{21}$$

and subsequent elements:

$$T_{p,v(p,j-1)}^{(i+1)} = \Phi_{p,v(p,j-1)} - \Phi_{p,v(p,j-1)-1} \tag{22}$$

This can be seen in Figures 6 and 7, where the affected array elements are marked with cyan font color. After that, parallel prefix sum is done on $T^{(i+1)}$, and the evaluation moves onto the next layer.

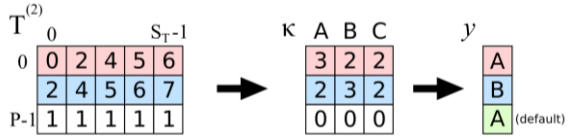

Figure 8
The last step of the interval-based inference algorithm

Fig. 8 shows the last step of the evaluation, when the last layer is reached. Class collector array κ is constructed (with size $P \times k$) and using $T^{(N-1)}$, the occurrences are counted in κ for each $j \in [0, \varphi_p - 1]$, $\forall p \in [0, P-1]$:

$$\kappa_{p,\,\Theta_{T_{p,j}^{(N-1)}}} = \sum_{j=0}^{\varphi_p - 1} \Theta_{T_{p,j}^{(N-1)}} \tag{23}$$

Finally, the output array y is calculated for all p:

$$y_p = \begin{cases} \text{argmax}_{\forall j}\, \kappa_{p,\text{j}} & \text{if } \varphi_p > 0 \\ \text{default} & \text{otherwise} \end{cases} \tag{24}$$

In Fig. 8, for the first sample (p=0) the class distribution of A, B and C is 3/7, 2/7 and 2/7, respectively, thus the output is A. For the third sample (p=2) the search has reached a dead end at layer #1, thus, the default class (A) is returned.

3 Performance Evaluation

3.1 Experimental Results

The proposed new inference method had been tested on two real-life benchmark problems from the UCI data repository [15] that are very commonly used to test the classification performance of machine learning methods. The implementation has been done on an average PC (Intel® Core™ i5-4590 CPU @ 3.30 GHz, 16 GB RAM), using CUDA v9.2 and Thrust v1.9 [16].

In the first set of experiments, the *Wisconsin Breast Cancer* (WBC) [17] dataset is used to compare the classification performances of the *proximity-based inference* (PBI) method and the new *interval-based inference* (IBI) method. The dataset consists of N=9 attributes and P=500 training samples, which have been separated into a training and a testing dataset, in various *training to testing ratios* (TTRs) from 5:95% (i.e. 5% of the 500 samples are used for training and 95% for testing), to 95:5% (vice versa). The parSIT is trained for each TTR and the same trained structure is used for the PBI and IBI phases.

Table 1
The calculation of classification measures: recall, precision and balanced accuracy

	Recall	**Precision**	**Balanced Accuracy**
Formula	$\frac{TP}{TP+FN}$	$\frac{TP}{TP+FP}$	$\frac{\frac{TP}{TP+FN}+\frac{TN}{TN+FP}}{K}$

In order to measure the performance, the *recall*, *precision* [18] and *balanced accuracy* rates [19] have been measured. Table 1 summarizes the formulas with which these values are derived, where K is the number of classes (K=2 in case of the WBC dataset). The data is calculated from the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) results from the inference.

For each class, the *recall* ratio shows how many instances of a given class are positively identified, while the *precision* ratio shows how many of all positive claims are actually true. The *balanced accuracy* ratio shows how well the classifier can identify both the positive and negative samples.

The inference results are compared in Figs. 9-11. For the interval-based inference, ρ=15% range value is used. As it can be seen in Fig. 9, the recall ratio of the proximity-based inference peaks around 96%, then slowly declines to 77% as the amount of training data ratio decreases (relative to the testing data size). The reason for this is the "narrow" search approach the PBI method uses in the problem space, only considering one value for each given attribute. However, the proposed interval-based method is more stable, providing a 94-97.4% recall rate even with fewer training data, since it expands the search area in the problem space to a wider region, thus, can better utilize the same trained structure.

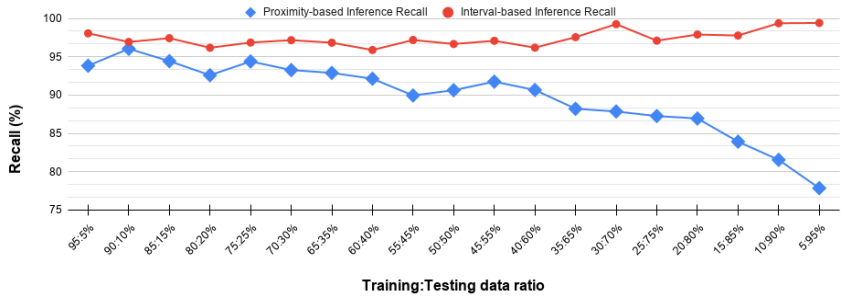

Figure 9
Comparison of the *recall* rates of the two inference methods using the WBC dataset

As Fig. 10 shows, their precision ratio is roughly the same (90-96%), although in case of lower TTRs (lower than 15:85%) the original PBI method preforms better, implying that the higher recall rate is at the cost of a lower precision rate.

In terms of balanced accuracy (Fig. 11), we can also see that the new IBI method provides a more stable (>95%) performance for the considered TTRs, while the PBI shows a slow decline for decreasing training set cardinalities.

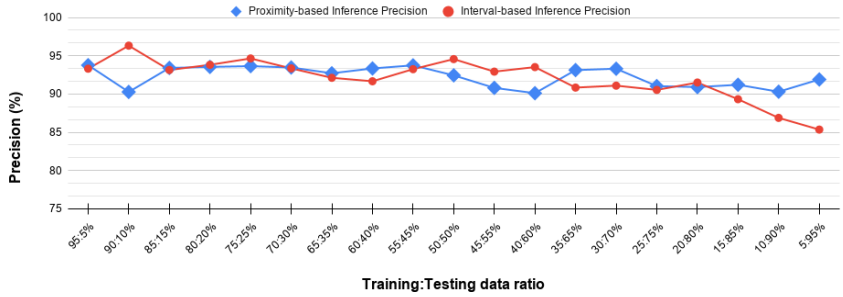

Figure 10

Comparison of the *precision* rates of the two inference methods using the WBC dataset

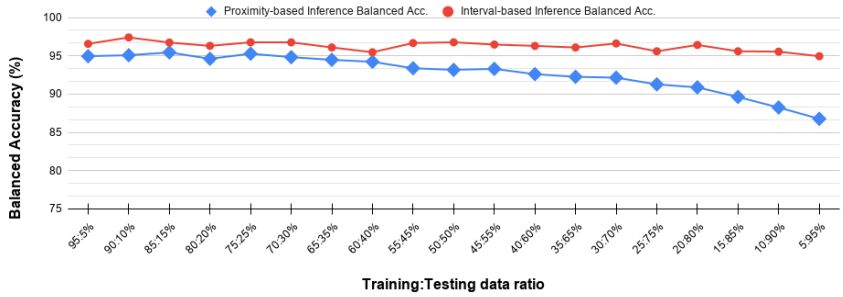

Figure 11

Comparison of the *balanced accuracy* rates of the two inference methods using the WBC dataset

Table 2 summarizes the results, breaking down the considered TTR spectrum into 5 intervals (from left to right in the figures): *very high* (the cases where the training to testing set ratios are more than 4:1 (80:20), i.e., there are more than 4 times as many training samples as testing samples), *high* (4:1 – 1.5:1), *moderate* (1.5:1 – 1:1.5), *low* (1:1.5 – 1:4) and *very low* (less than 1:4). The recall, precision and balanced accuracy rates have been averaged in these intervals, which can be seen in the table. The difference (Δ) in percentage between the IBI and PBI measures is also shown (marked with *bold text*), which indicates how the new IBI algorithm really performs compared to the PBI method.

Table 2

Comparison of the recall, precision and balanced accuracy rates using the WBC dataset

TTR (%)	**Recall (%)**			**Precision (%)**			**Balanced accuracy(%)**		
	PBI	*IBI*	Δ	*PBI*	*IBI*	Δ	*PBI*	*IBI*	Δ
Very High	94.2	97.1	**2.9**	92.7	94.1	**1.4**	95.0	96.8	**1.7**
High	93.2	96.7	**3.5**	93.3	92.9	**- 0.3**	92.3	93.6	**1.2**

Moderate	90.8	97.0	**6.2**	92.3	93.6	**1.2**	93.3	96.6	**3.4**
Low	88.5	97.5	**9.0**	91.9	91.5	**- 0.4**	92.1	96.2	**4.1**
Very Low	82.6	98.6	**16.0**	91.1	88.3	**- 2.8**	88.9	95.6	**6.7**

As it can be seen, while the precision is roughly the same (with very little difference) for very high to moderate TTRs, while for lower TTRs the precision of the IBI method is worse than that of the PBI method. On the other hand, a steady increase can be seen in the recall and balanced accuracy rates for the difference in favor of the IBI method as the TTRs decrease.

The experiments have been done on a multi-class problem as well: The Iris dataset, which consists of P=150 samples, N=4 attributes and K=3 classes. The results are counted as follows: for each class j, if the classifier marks a given input as part of the class, and if correct, it is counted as true positive, and false positive otherwise. Similarly, if the inference marks the sample as *not* being part of the class, then it counts to true negative if it is correct, and false negative if it is not correct. The recall and precision rates are averaged among all classes.

Figs. 12-14 show the results of the classification using the same performance measures. As it can be seen, the multiclass problem was much harder to the proximity-based inference method, exactly due to the reason that has been outlined in the introduction. The new inference method, however, provides not only a more stable, but also much higher rate for all three performance ratios.

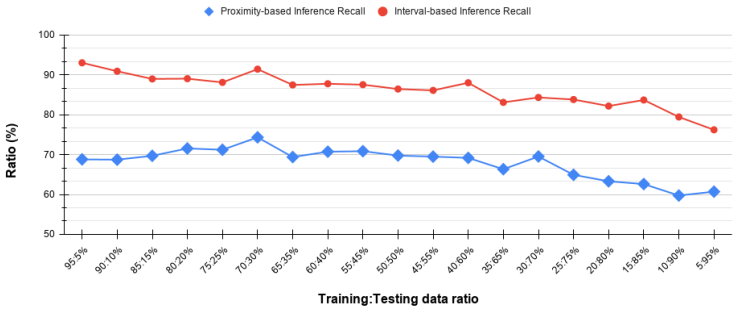

Figure 12

Comparison of the *recall* rates of the two inference methods using the Iris dataset

Although compared to the 2-class case of the previous experiment, the IBI method shows a decrease in recall rate for lower TTRs, but it still generally provides a better recall rate over the PBI method by at least 15 percentage points, as Fig. 12 indicates.

The precision rate of the IBI, on the other hand, is more stable in comparison to that of the PBI, providing a rate of ~88% for all the considered TTRs while the precision rate of the PBI is gradually decreasing with the TTR. This implies that the IBI method is much better suited for multiclass problems.

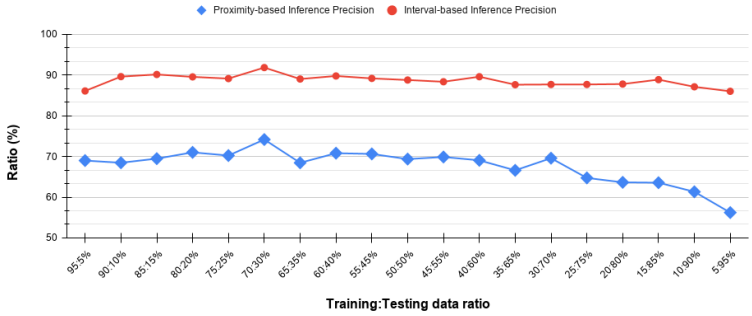

Figure 13

Comparison of the *precision* rates of the two inference methods using the Iris dataset

The balanced accuracy rate (Fig. 14) shows a very slow decline with the TTRs for the IBI method, but still provides an at least 80% rate for the lowest TTRs, while the PBI method only provides a ~70% balanced accuracy rate for the same. In general, the IBI outperforms the PBI method by 10-15%.

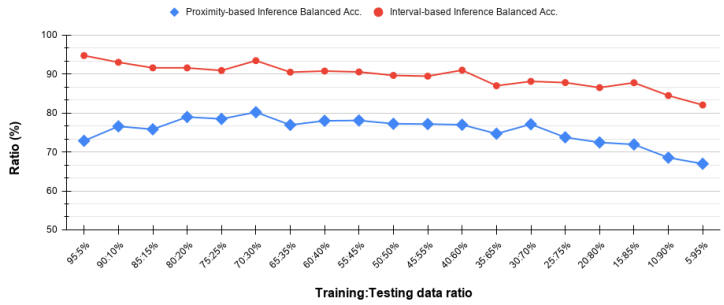

Figure 14

Comparison of the *balanced accuracy* rates of the two inference methods using the Iris dataset

Table 3 summarizes the results (averaged for TTR intervals) for the Iris dataset, the same way as Table 1 of the previous experiment. According to the results, the recall and precision rates of the PBI method is roughly the same, ~70% on average for very high to moderate TTRs, while decreasing to ~60% for lower TTRs. The recall rate of the IBI method slowly decreases from ~90% to ~80% throughout the TTR spectrum, while its precision rate stays around 88%, meaning that the classes it marks as a positive hit are correct in the majority of time. The balanced accuracy of the PBI is relatively stable at ~70-77.5% throughout the TTR spectrum (with a very slow decrease for lower intervals), while that of the IBI method slowly decreases from ~92.8% to ~85.2%.

Table 3
Comparison of the recall, precision and balanced accuracy rates using the Iris dataset

TTR (%)	Recall (%)			Precision (%)			Balanced accuracy(%)		
	PBI	*IBI*	Δ	*PBI*	*IBI*	Δ	*PBI*	*IBI*	Δ
Very High	69.7	90.5	**20.8**	69.5	88.9	**19.4**	76.1	92.8	**16.7**
High	71.4	88.7	**17.3**	70.9	90.0	**19.0**	70.0	88.8	**18.8**
Moderate	70.1	86.7	**16.7**	70.0	88.8	**18.8**	77.5	89.9	**12.4**
Low	67.5	84.8	**17.3**	67.5	88.2	**20.7**	75.7	88.5	**12.9**
Very Low	61.6	80.4	**18.8**	61.2	87.5	**26.3**	70.0	85.2	**15.2**

The effects of range parameter ρ to the classification performance has also been examined. Fig. 15 shows the recall, precision and balanced accuracy rates of the interval-based inference, on the Iris dataset (using 70% of the samples for training and 30% for testing). As it can be seen, at ρ = 5%, all the performance measures are at their maximum and maintain a high value until around 15%, where a steady decline begins. The recall decreases to 33%, which is expected for a 3-class problem, as the covered interval is large enough to cover the whole domain, thus, only returning the default class for any given inputs. The balanced accuracy falls to 50%, while the precision rises back to ~75% for higher ρ values.

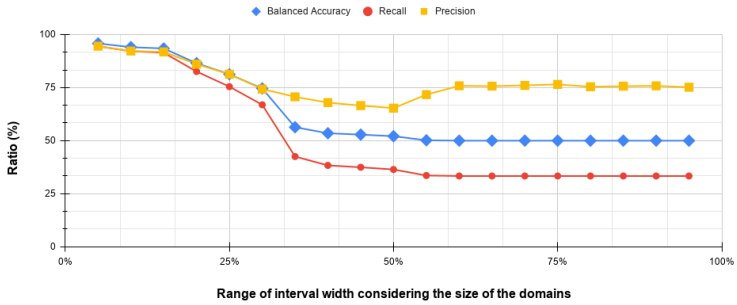

Figure 15
Performance measure analysis of using different range sizes on the Iris dataset

Remark: For large range values, the ~75% precision rate is caused by the way the measure is calculated, i.e. taking the ratio between the true positive findings and all positive claims (TP+FP). If there are no positive claims for a given class at all, then the recall for that class is 0%, while the precision is 100% (since none of the positive claims are wrong). In this case, the classifier only returns the default class, which means 100% precision for the two other classes, while only 25-33% precision for the default class, which makes the average, approximately 75%.

Interestingly, for the WBC dataset, ρ = 25% provided the best results, even though the performance for Iris dataset peaked at ρ = 5%, which shows that the optimal ρ value is primarily dependent on the given data. Thus, it is recommended that for

any given problem, the inference should be tried for different values between 5% and 30%, to find the value that is most suitable.

3.2 Complexity Analysis

The computational or time complexity of the proximity-based inference is $O(N \cdot m)$, where m is the average number of markers per layer. Since the new interval-based inference method uses parallel prefix sum twice in each layer, it will be inherently slower than the proximity-based counterpart, at $O(N \cdot m \cdot \log_2 m)$. The new method also requires more parallel processing units to compute, which can limit its usability for large index arrays, if the range parameter is also large.

However, both methods still only marginally dependent on the amount of input data (as the number of processes contributes to the time complexity, given that they are needed to be managed by the parallel computing framework).

Remark: It is recommended to set the order of the attributes to such that, the one with the least value variety, is the root layer, since both inferences have to poll all elements of the index array in the root layer.

Conclusions

In this paper, a new inference method is presented for Parallelized Sequential Indexing Table classifiers. While the original inference method uses a proximity-based algorithm, where the inference is only considering a one route (a single series of attribute values closest to the input data values) through the problem space, the newly proposed algorithm, does a more thorough search through, by regarding intervals of values for each attribute and thus, provides a more accurate classification.

The new interval-based inference method has been tested on two real-life benchmark problems that are very commonly used to test the classification performance of machine learning methods. Overall, the original proximity-based inference method has a lower computational complexity, thus it has a faster operation, requires fewer processing units, and according to the test results, it performs reasonably well on 2-class problems (with a balanced accuracy rate of ~88.9-95%), though less so on multiclass problems (~70-77.5%), due to the higher complexity of the problem. The proposed new inference method has a slightly higher computational complexity and thus, have a slower operation compared to that of the proximity-based inference method, it is more intensive regarding the processing units, but in return it performs slightly better on 2-class problems (~93.6-96.8% balanced accuracy rate) and much better on multiclass problems (~85.2-92.8%) compared to the proximity-based systems, even though they both use the same trained classifier.

The experiments have shown that the proposed inference method can provide good classification metrics, even for multiclass problems (~80% recall, ~87.5%

precision and ~85.2% balanced accuracy rates), for cases where the testing data cardinality significantly outweighs that of the training data, meaning that with it the parSIT classifier is a reasonable choice for a low-complexity, fast training and fast performing classifier for such problems.

In future work, we will further improve the proposed inference method, in order to increase its speed, and based on the classifier, we will develop new methods, where the processing order of the inputs are not bounded by a single ordering scheme.

Acknowledgement

Supported by the ÚNKP-19-3-IV-OE-56 New National Excellence Program of the Ministry for Innovation and Technology.

References

[1] S. Hussein, P. Kandel, C. W. Bolan, M. B. Wallace, U. Bagci, "Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches," *in IEEE Transactions on Medical Imaging*, Vol. 38, No. 8, pp. 1777-1787

[2] S. Roy et al., "Deep Learning for Classification and Localization of COVID-19 Markers in Point-of-Care Lung Ultrasound," *in IEEE Transactions on Medical Imaging*, Vol. 39, No. 8, pp. 2676-2687

[3] Gibson, B. Issac, L. Zhang, S. M. Jacob, "Detecting Spam Email With Machine Learning Optimized With Bio-Inspired Metaheuristic Algorithms," *in IEEE Access*, Vol. 8, pp. 187914-187932

[4] J. Yun, J. Woo, "A Comparative Analysis of Deep Learning and Machine Learning on Detecting Movement Directions Using PIR Sensors," *in IEEE Internet of Things Journal*, Vol. 7, No. 4, pp. 2855-2868

[5] G. Jia, H. -K. Lam, S. Ma, Z. Yang, Y. Xu, B. Xiao, "Classification of Electromyographic Hand Gesture Signals Using Modified Fuzzy C-Means Clustering and Two-Step Machine Learning Approach," *in IEEE Trans. on Neural Syst. and Rehabilitation Engineering*, Vol. 28, No. 6, pp. 1428-1435

[6] M. Jocic, E. Pap, A. Szakál, D. Obradovic, Z. Konjovic, "Managing Big Data Using Fuzzy Sets by Directed Graph Node Similarity," *Acta Polytechnica Hungarica*, Vol. 14, No. 2. 2017, pp. 183-200

[7] R. Spir, K. Mikula, N. Peyrieras "Parallelization and validation of algorithms for Zebrafish cell lineage tree reconstruction from big 4D image data," *Acta Polytechnica Hungarica*, Vol. 14, No. 5. 2017, pp. 65-84

[8] A. Vukmirović, Z. Rajnai, M. Radojičić, J. Vukmirović, M. J. Milenković, "Infrastructural Model for the Healthcare System based on Emerging Technologies," *Acta Polytechnica Hun.*, Vol. 15, No. 2, 2018, pp. 33-48

[9] A. R. Várkonyi-Kóczy, B. Tusor, J. T. Tóth, "A Multi-Attribute Classification Method to Solve the Problem of Dimensionality," *in Proc. of the 15th Int. Conf. on Global Research and Education in Int. Sys. (Interacademia'2016), Warsaw, Poland*, 2016, pp. PS39-1–PS39-6

[10] B. Tusor, A. R. Várkonyi-Kóczy, J. T. Tóth, "Active Problem Workspace Reduction with a Fast Fuzzy Classifier for Real-Time Applications," *IEEE International Conference on Systems, Man, and Cybernetics, Budapest, Hungary*, October 9-12, 2016, pp. 4423-4428, ISBN: 978-1-5090-1819-2

[11] B. D. Zarit, B. J. Super, F. K. H. Quek, "Comparison of five color models in skin pixel classification," *in Proc. of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, Corfu, Greece*, Sep. 26-27, 1999, pp. 58-63

[12] B. Tusor, J. T. Tóth, A. R. Várkonyi-Kóczy, "Parallelized Sequential Indexing Tables for Fast High-Volume Data Processing," *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Dubrovnik, Croatia*, 2020, pp. 1-6

[13] B. Tusor, A. R. Várkonyi-Kóczy, "Memory Efficient Exact and Approximate Functional Dependency Extraction with ParSIT," *2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES), Reykjavík, Iceland*, 2020, pp. 133-138

[14] M. Safari, W. Oortwijn, S. Joosten, M. Huisman, "Formal Verification of Parallel Prefix Sum," In: Lee R., Jha S., Mavridou A. (eds) *NASA Formal Methods. NFM 2020. Lecture Notes in Computer Science*, Vol. 12229, Springer, Cham, 2020

[15] D. Dua, C. Graff, *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019

[16] A. V. George, S. Manoj, S. R. Gupte, S. Mitra, S. Sarkar, "Thrust++: Extending Thrust Framework for Better Abstraction and Performance," 2017 *IEEE 24th International Conference on High Performance Computing (HiPC), Jaipur*, 2017, pp. 368-377, doi: 10.1109/HiPC.2017.00049

[17] O. L. Mangasarian, W. H. Wolberg: "Cancer diagnosis via linear programming", *SIAM News*, Vol. 23, No. 5, September 1990, pp. 1&18

[18] M. Buckland, F. Gey: "The relationship between recall and precision," *Journal of the American Soc. for Inf. Science*, Vol. 45, No. 5, 1994, pp. 12-19

[19] V. García, R. A. Mollineda, J. S. Sánchez, "TI - Index of Balanced Accuracy: A Performance Measure for Skewed Class Distributions", in: *Pattern Recognition and Image Analysis*, Springer Berlin Heidelberg, pp. 441-448, 2009