ARTIFICIAL AND COMPUTATIONAL INTELLIGENCE AND KNOWLEDGE ENGINEERING

K Nearest Neighbor Classifiers and their Variations

Adrian Horzyk

horzyk@agh.edu.pl

AGH University of
Science and Technology
Krakow, Poland

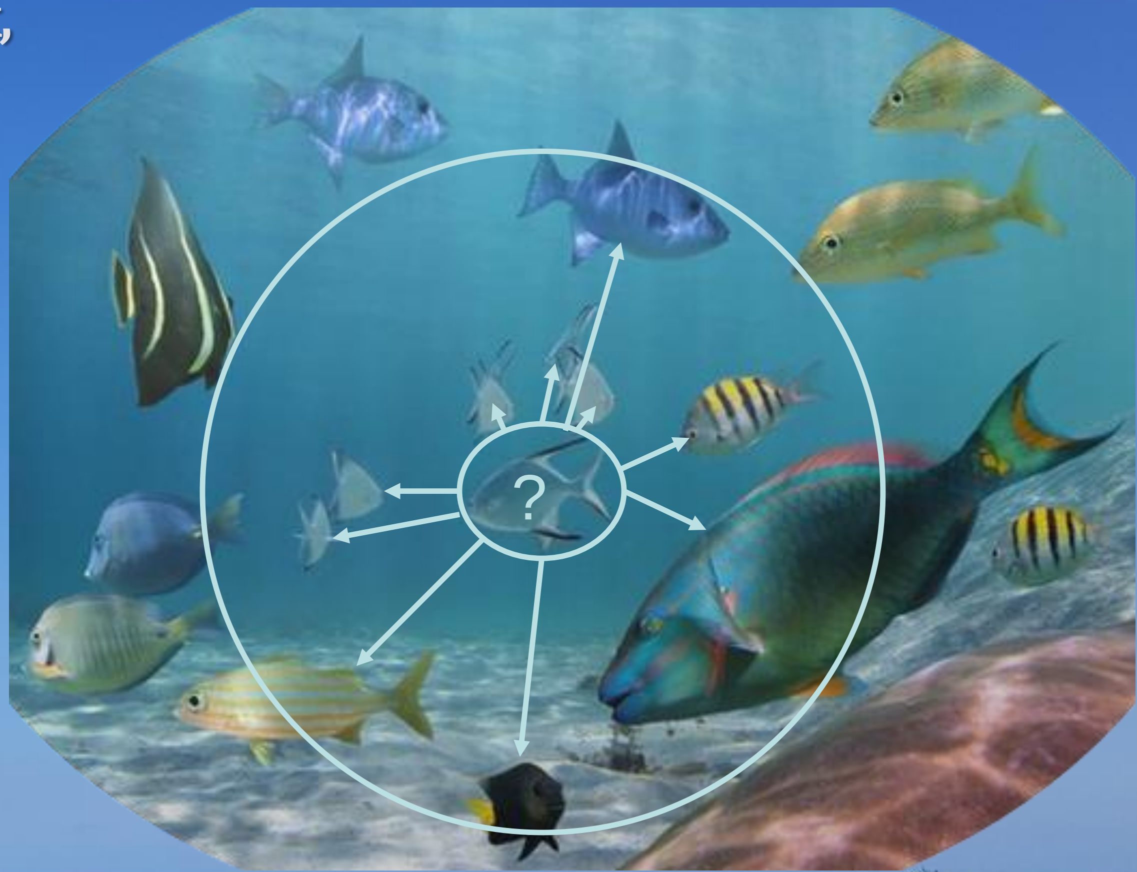
K Nearest Neighbor Classifiers

K-Nearest Neighbors (KNN) is a classifier that belongs to the group of lazy algorithms, i.e. those that do not create an internal representation of knowledge about the problem based on the data, but search for a solution after the sample presentation.
The method require to store all training patterns for which it determines the distance to the test pattern.

There is also another group learning algorithms that is called eager which develop a model first, and next use it for a give task, e.g. classification.

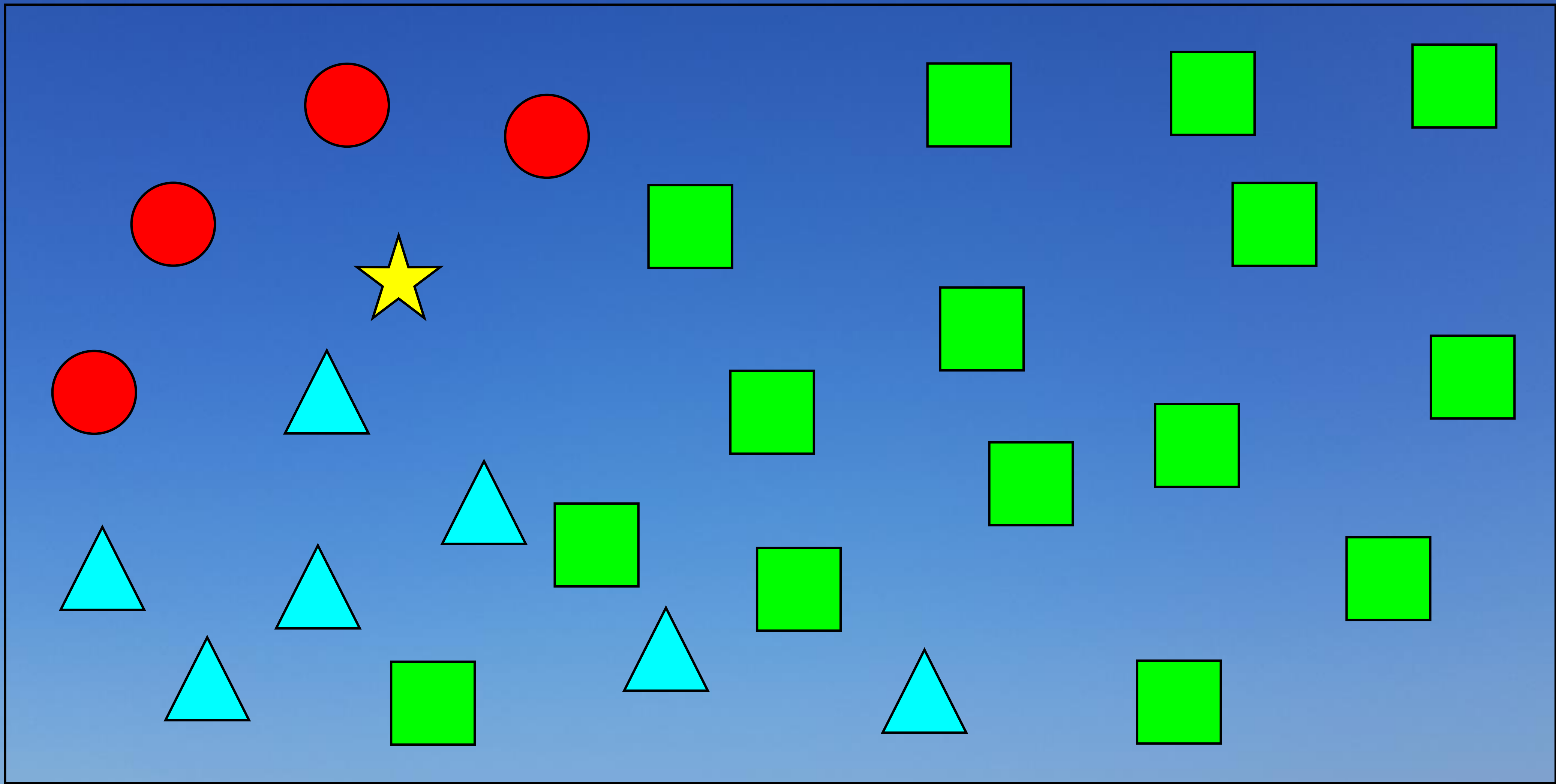
This group of methods includes all types of neural networks, fuzzy systems, decision trees, SVM, and many more.

After the training (adaptation) of such models, the training data can be deleted/removed because the classification process uses only the created model.



K Nearest Neighbor Classifiers

To which class belongs to the star: circles, triangles or squares?



K Nearest Neighbor Classifiers

The set of learning patterns (training patterns) consists of a set of pairs $\langle x^i, y^i \rangle$, where:

x^i is an input vector $x^i = [x^i_1, \ldots x^i_n]$ defining objects (usually in the form of a vector or matrix),
y^i is the predicted / related value, e.g. an index or name of the class to which the x^i belongs.

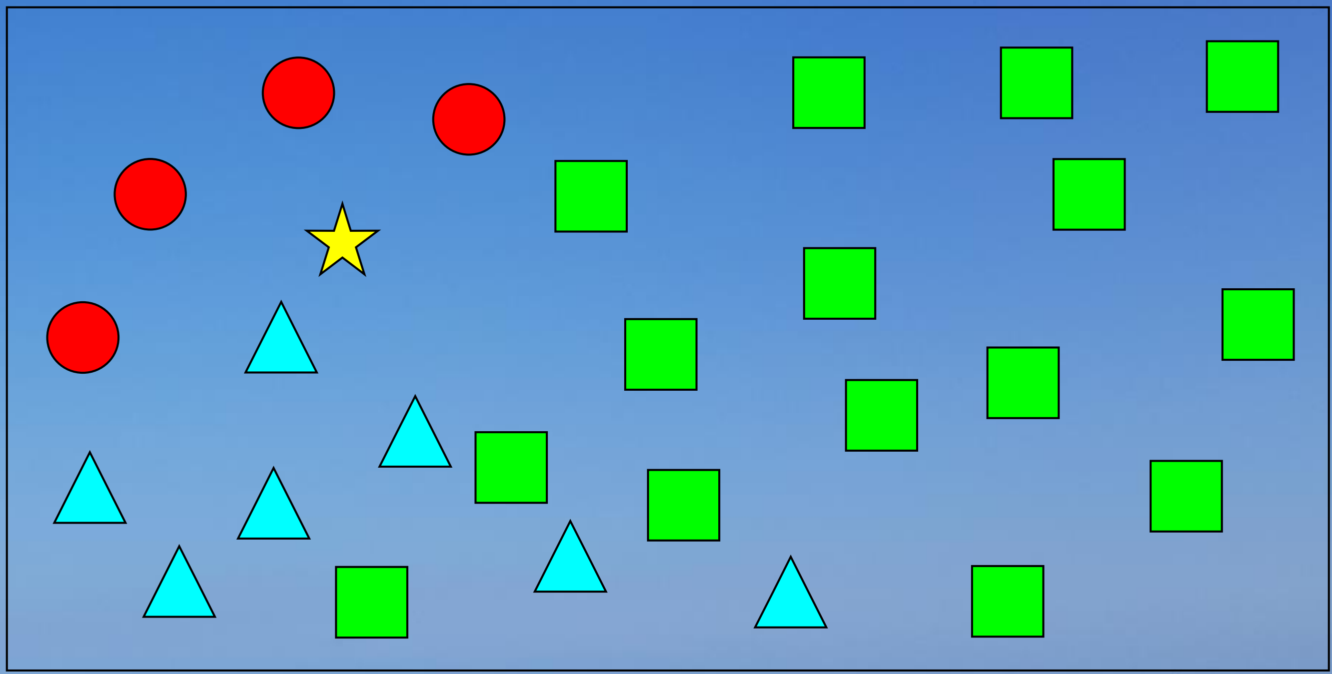
In the figure, we have objects belonging to three classes: circles, triangles, and squares.

A star is a new object that we want to classify, i.e. assign it to one of the existing classes.

To which class belongs to the star: circles, triangles or squares? What intuition tells us?

For example, you can examine the distances of the star from other objects for which the class is known, using one of the known metrics, e.g. an Euclidian distance:

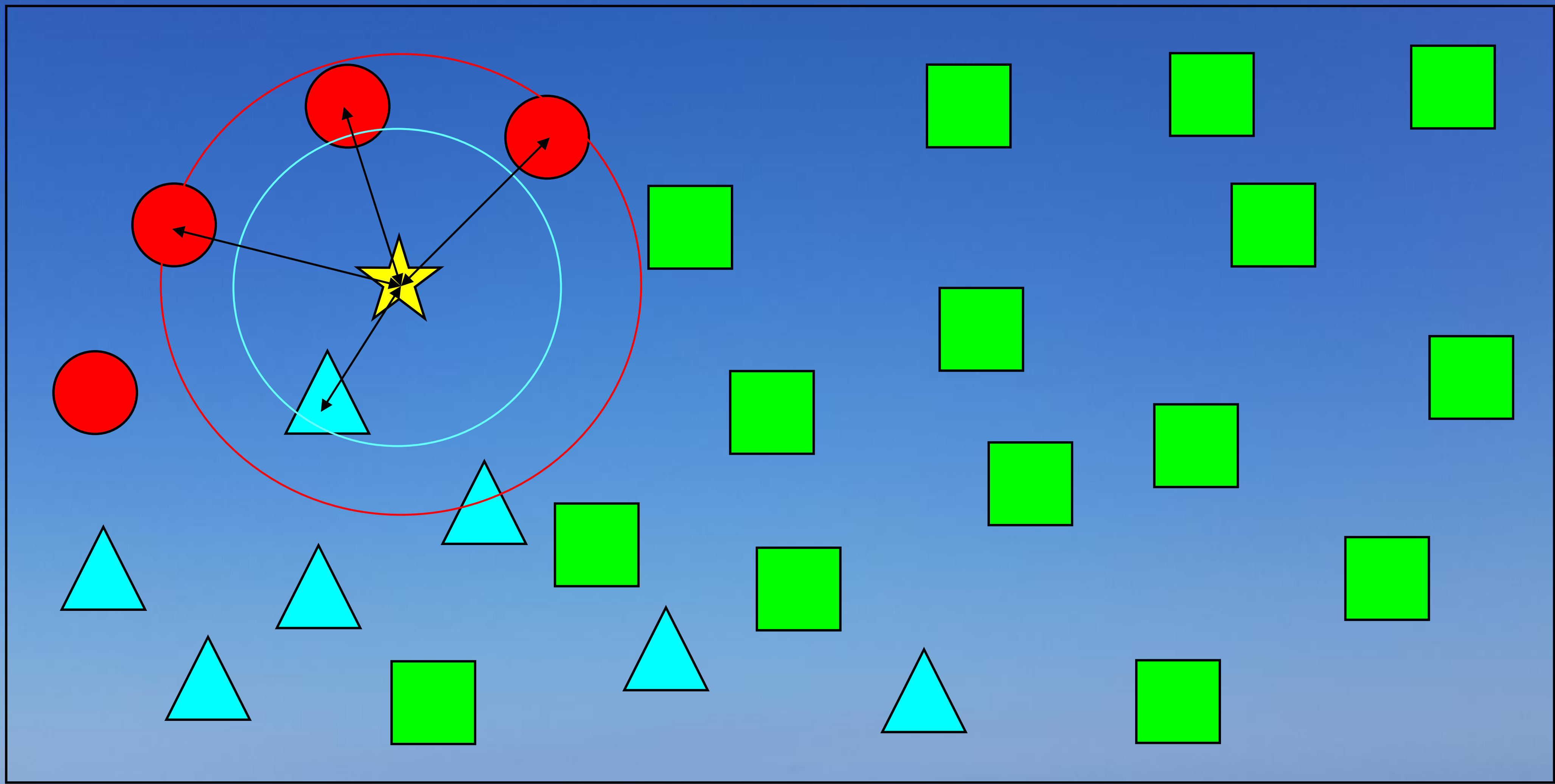
$$\left\| x - x^k \right\|_2 = \sqrt{\sum_{j=0}^{J} \left(x_j - x_j^k \right)^2}$$



and on this basis specify a class of the star.

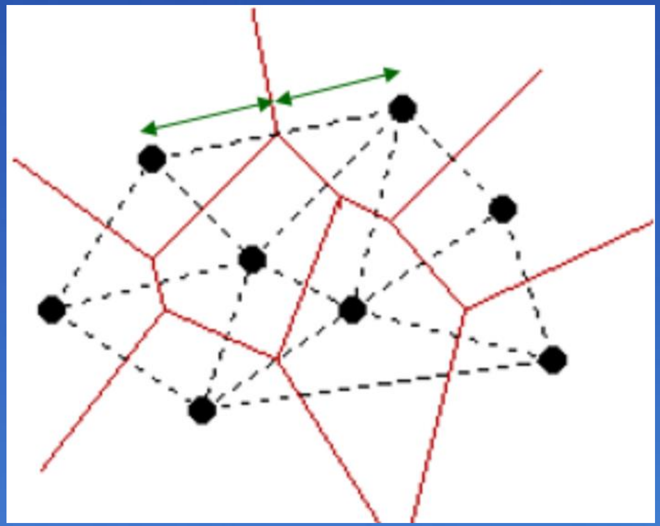
K Nearest Neighbor Classifiers

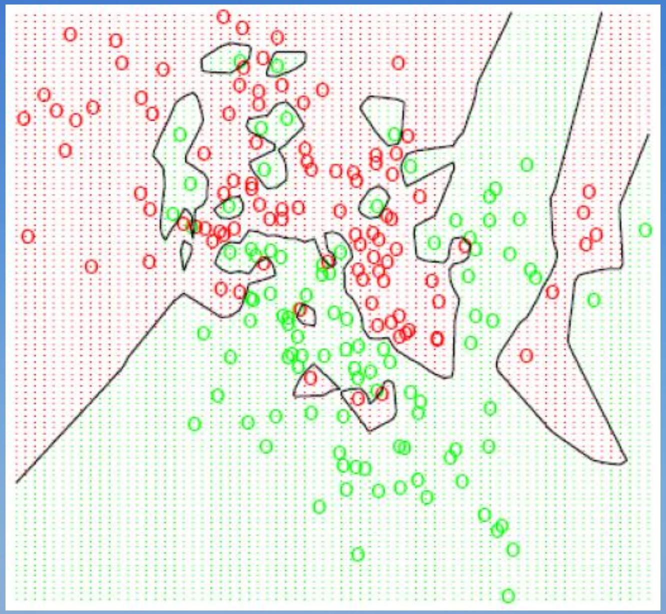
Classic K Nearest Neighbor algorithm determines k neighbors to which the classified objects is the closest in the selected metric (e.g. Euclidean), and then determines the classification result on the basis of the majority of votes of these k nearest neighbors taking into account which class is represented the largest number of times in the group of k nearest neighbor objects.

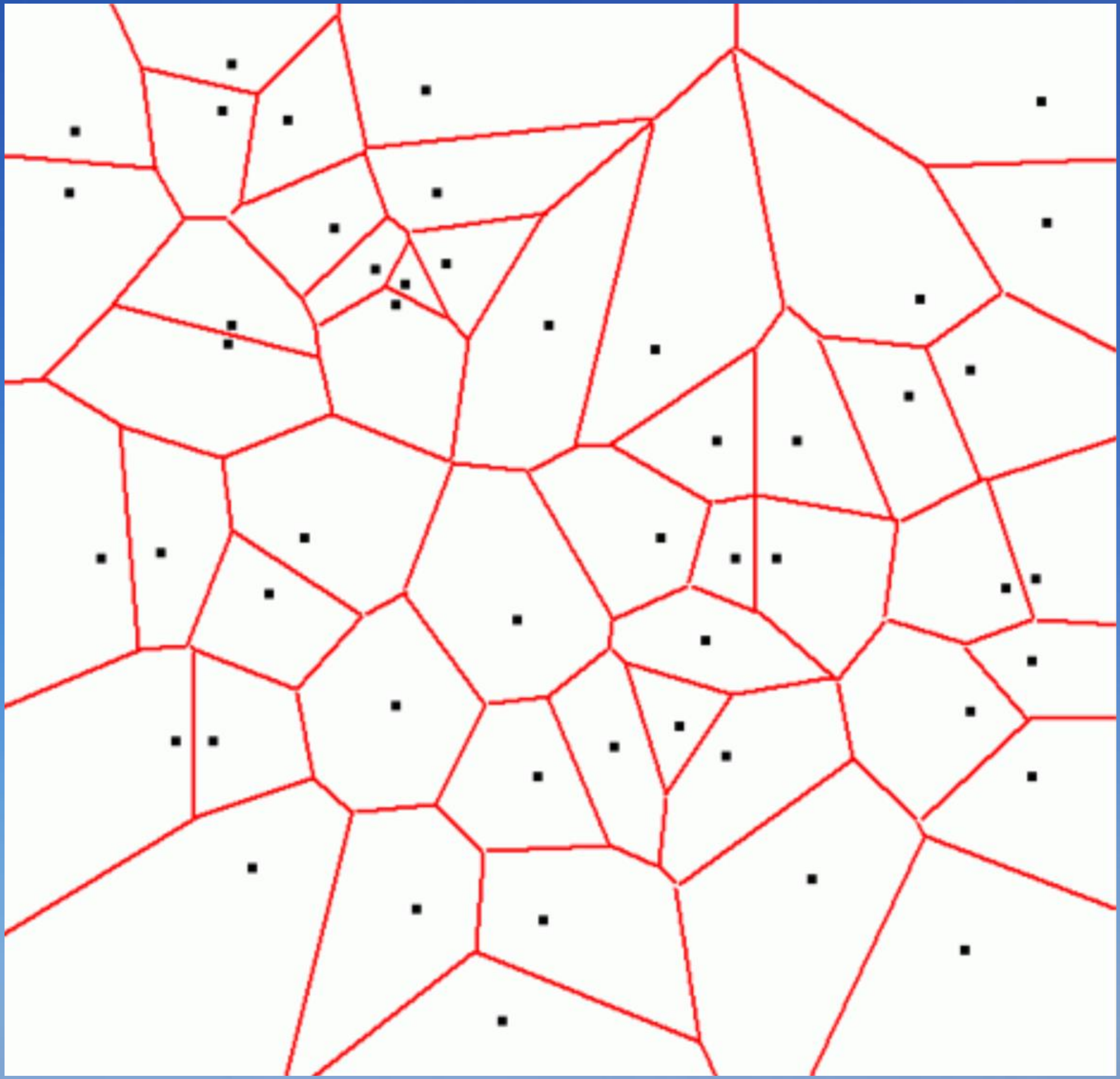


K Nearest Neighbor Classifying Regions

Voronoi diagrams are used to illustrate the areas of attraction to the nearest objects in space.







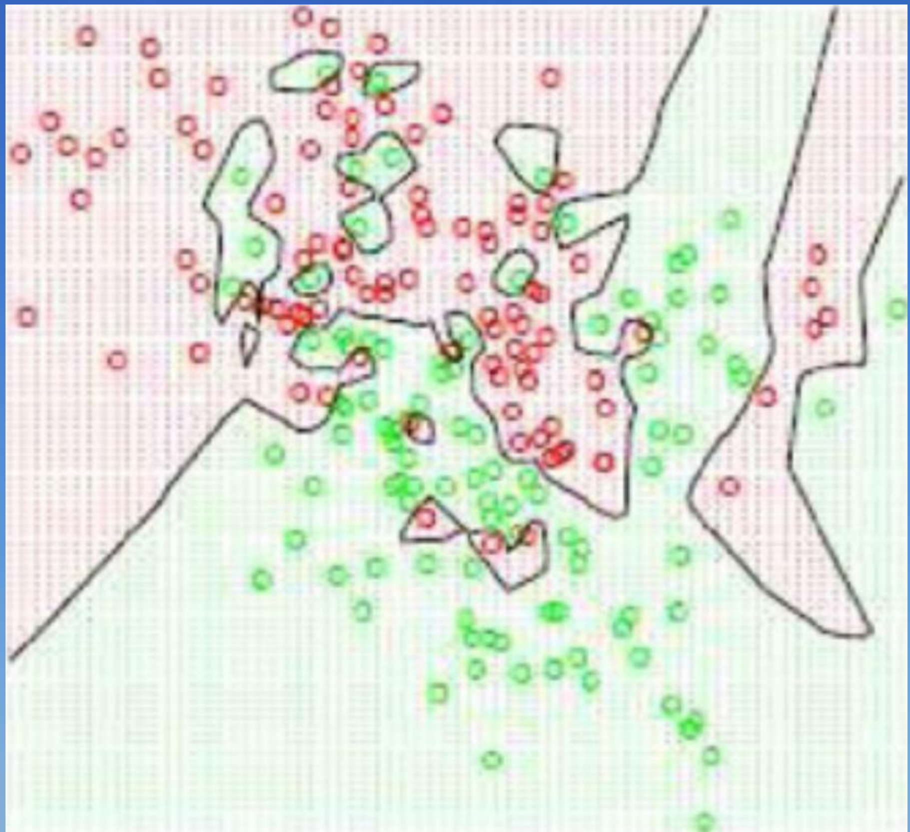
K Nearest Neighbor Classifying Regions

K Nearest Neighbor algorithm gives different results for different K.

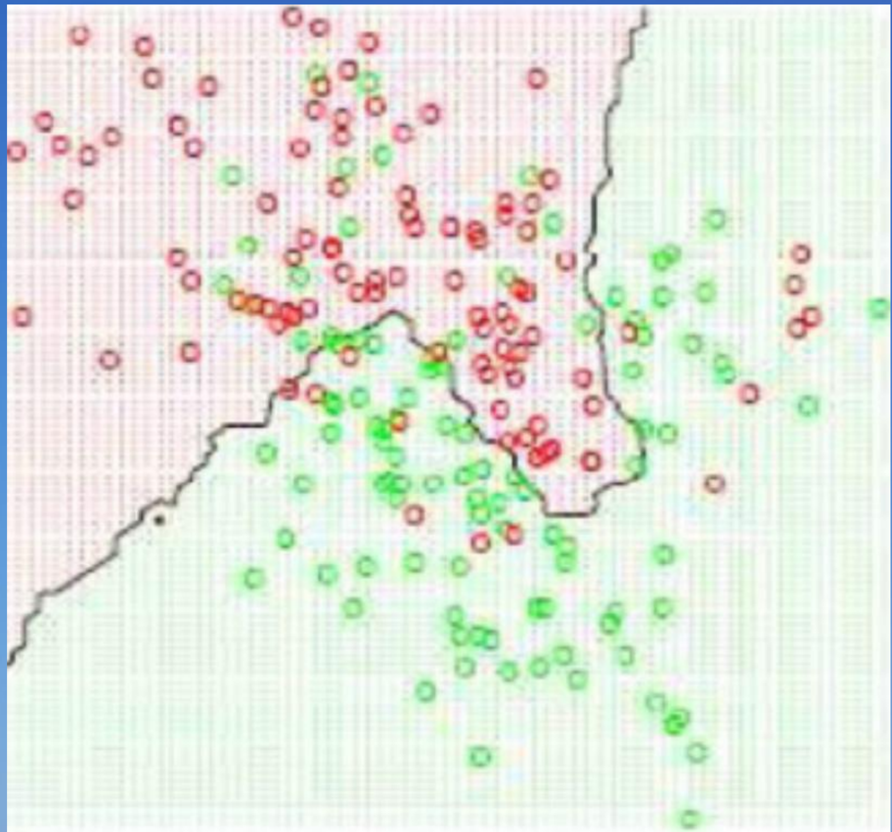
The areas of attraction determine the result of the classification.

Larger K values allow smoothing of the dividing areas, removing noise and artifacts (can better generalize), but also lead to errors in the classification of thinner patterns.

K = 1



K = 15

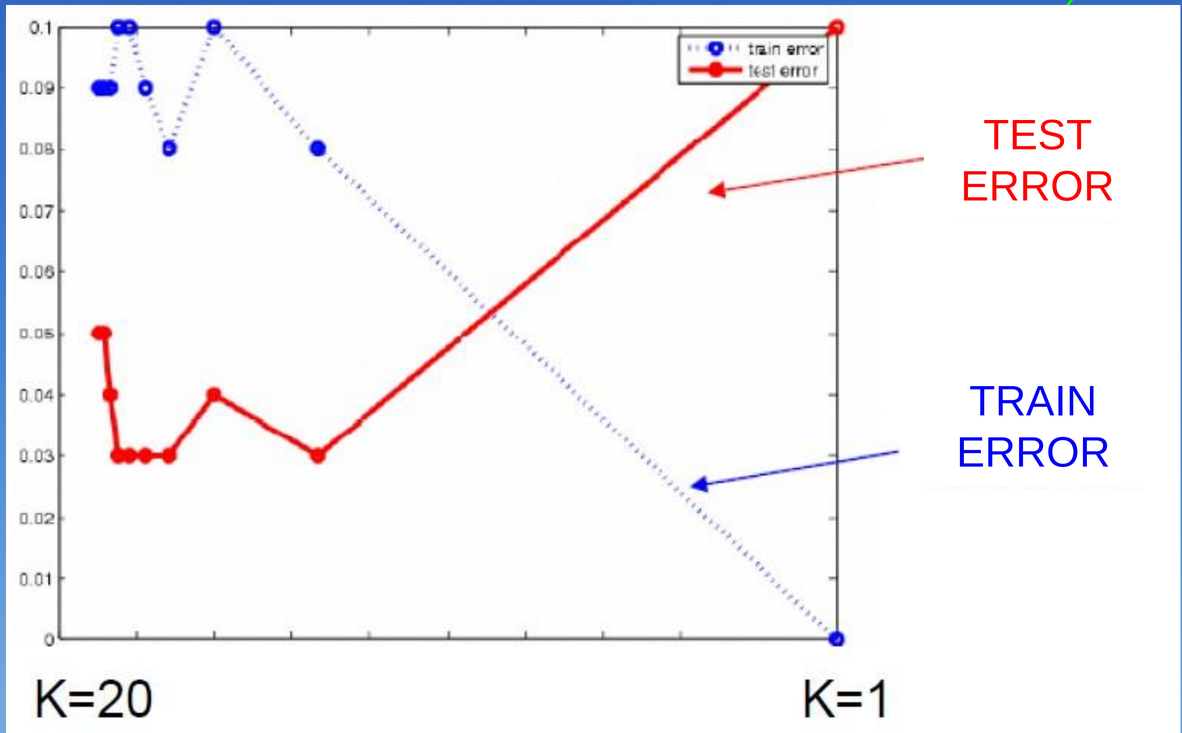


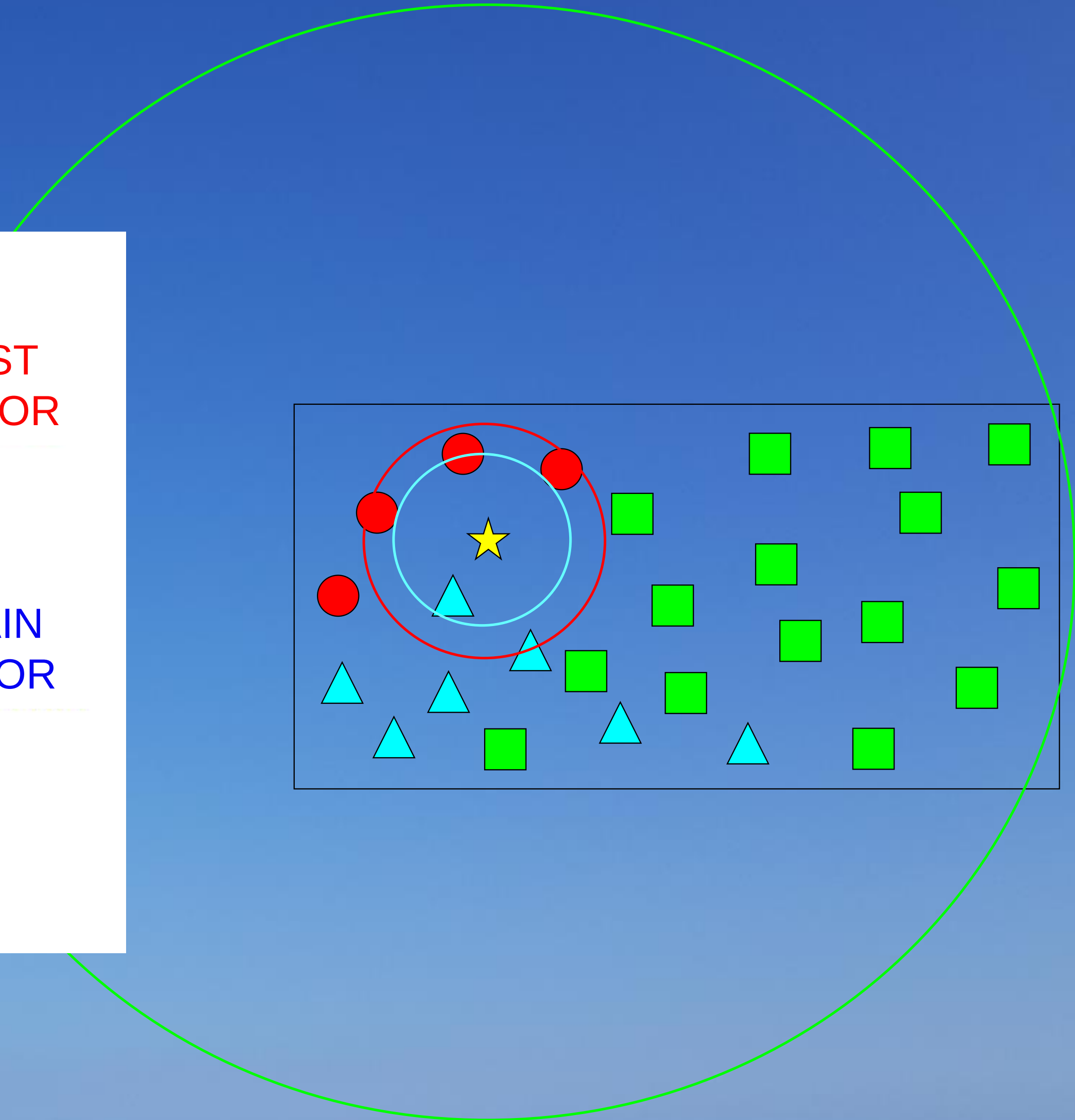
(from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

How to choose K?

If we select K = N, where N is the number of all training objects (patterns),
then the result of the classification will always be determined by the most frequent class:

The train error is always 0 for K = 1.

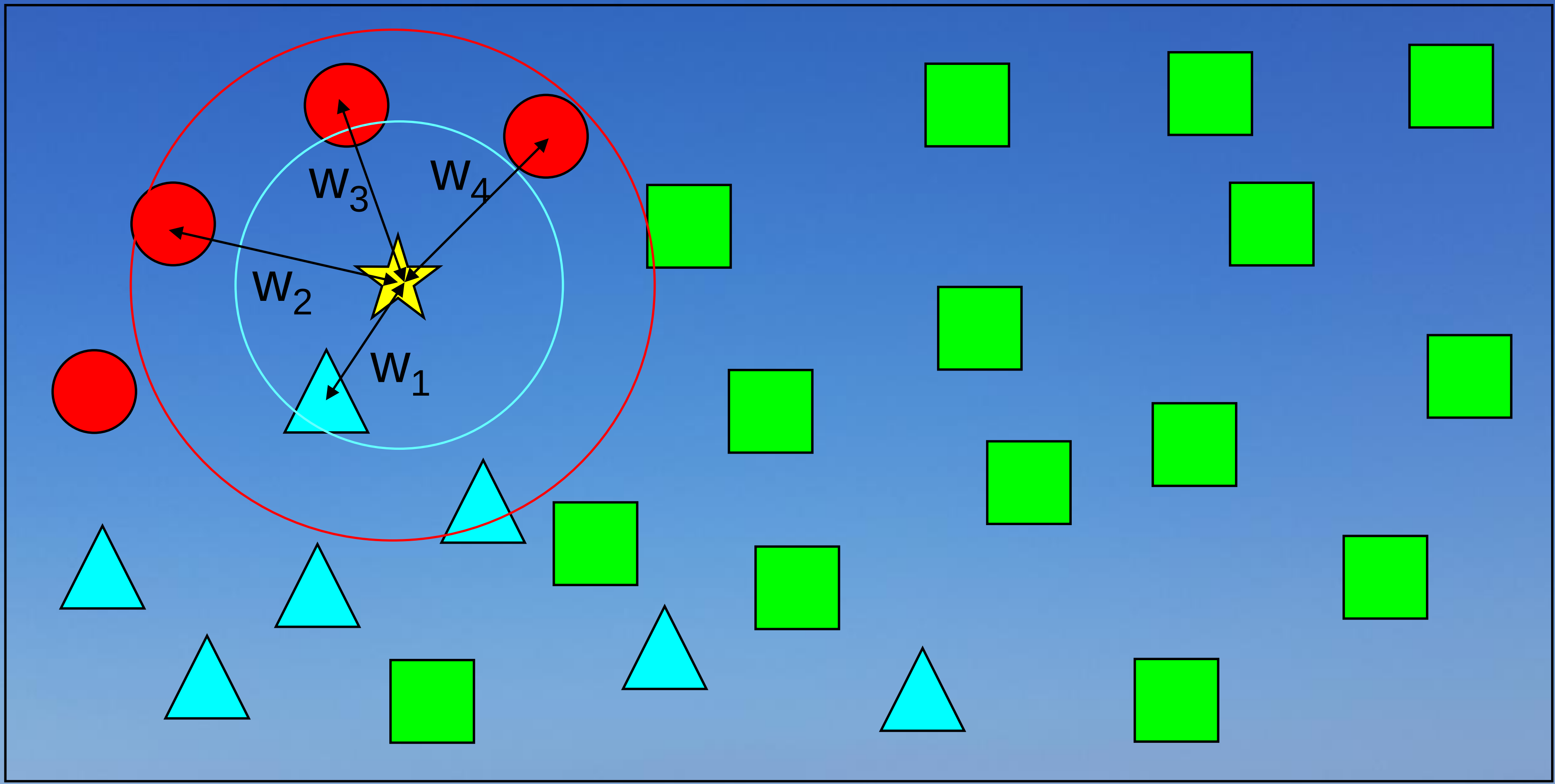




Variations of KNN

The Distance Weighted Nearest Neighbors method leads to voting on the star classification taking into account weighted distances to k nearest neighbors for the selected metric.

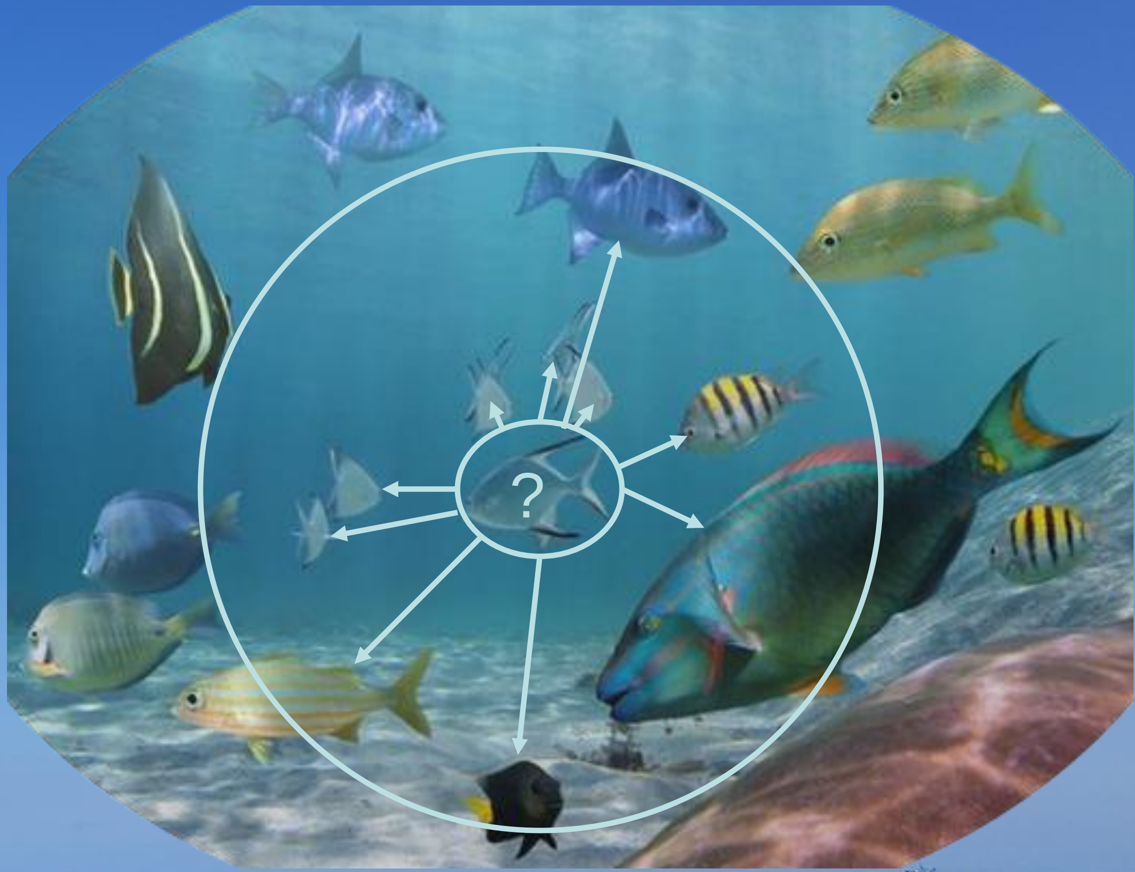
Therefore, the closest objects (patterns) will have the greatest impact on the classification result.



Drawbacks of KNN Classifiers

KNN classifiers are robust to noisy training data and very easy to implement, but they are:

- » **Lazy** because they do not create a computational model,
- » **High computational cost because they require to compute <u>the distance of each classified sample to all training data</u> (linear computational complexity for each classified sample)** while other classifiers usually have **constant computational complexity when classifying samples.**
- » The method is **sensitive to the variability** of patterns representing different classes.



Therefore, KNN cannot be efficiently used to Big Data!

BIG DATA

Why Storing Data in the Tables?

We mostly use tables to store, organize and manage data in computer science:

	ATTRIBUTES				
SAMPLE OBJECTS	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	CLASS LABEL
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica

However, common relationships like minima, maxima, identity, similarity, neighborhood, number of duplicates must be found in loops that search for them and evaluate various conditions.

The more data we have, the longer time requirements we face!

What can be done to achieve better efficiency?


Associate!
Big Data…
Big Problem?

Objectives of the Presented Research

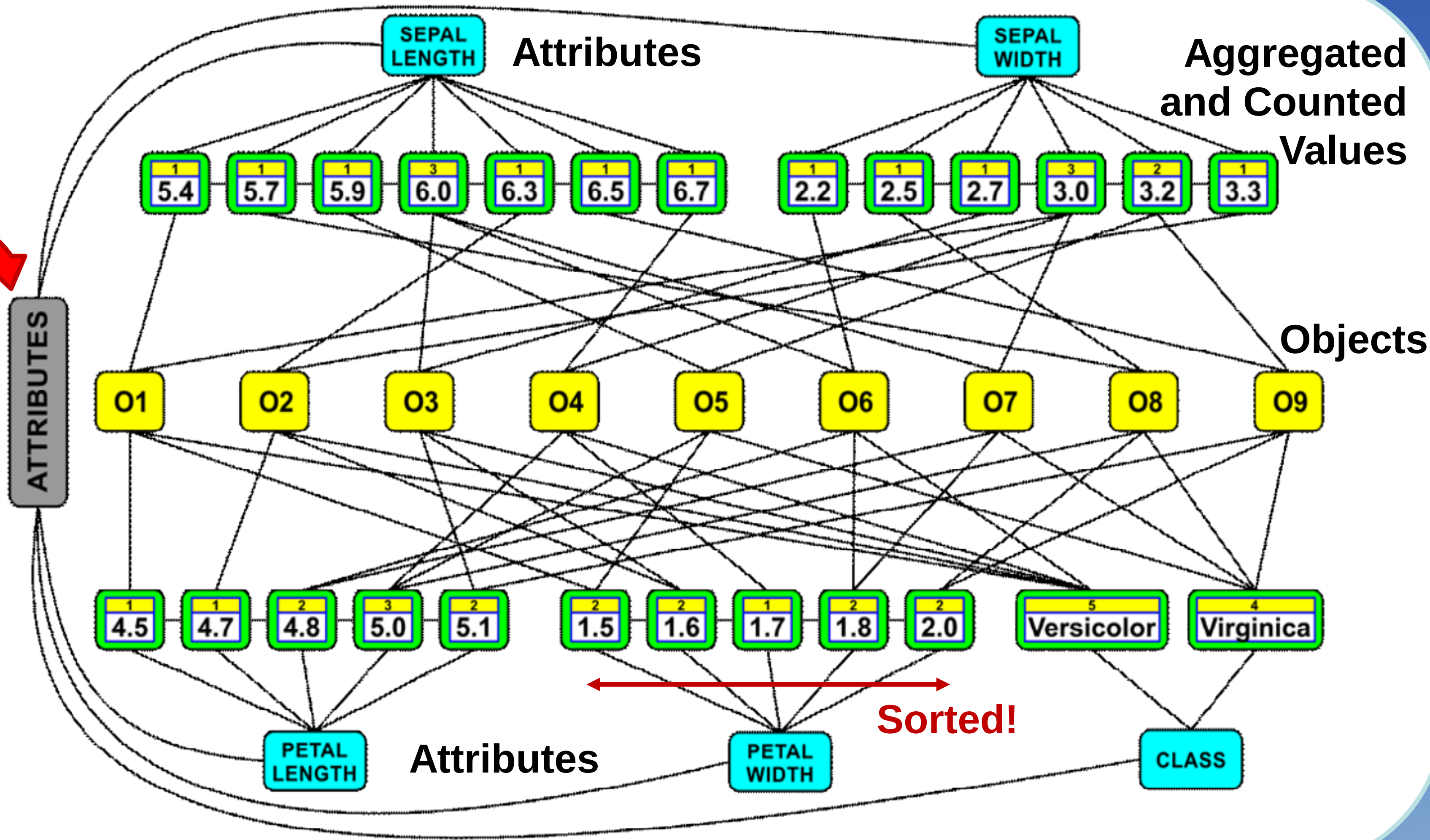
Associative Graph Data Structures (AGDS) can be easily and quickly created for any data and allow for:

- » **Rising the computational efficiency of kNN classification** typically <u>tens or hundreds of times</u> in comparison to the classic kNN approaches.
- » **Transforming lazy KNN classifiers** to **eager KNN+AGDS classifiers.**
- » **Defining an efficient computational model for KNNs.**
- » **Aggregating duplicates of values defining** training patterns and their defining attribute values smartly, saving time and memory.
- » **Avoiding looking through all training data during the classification.**
- » **Finding k nearest neighbors always in constant time** because neighbors are **searched locally only in the nearest neighborhood.**
- » Making KNN suitable and efficient for the classification of Big Data!

Associative Graph Data Structure (AGDS)

AGDS links related data of various kinds horizontally and vertically:

	ATTRIBUTES				
SAMPLE OBJECTS	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	CLASS LABEL
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica



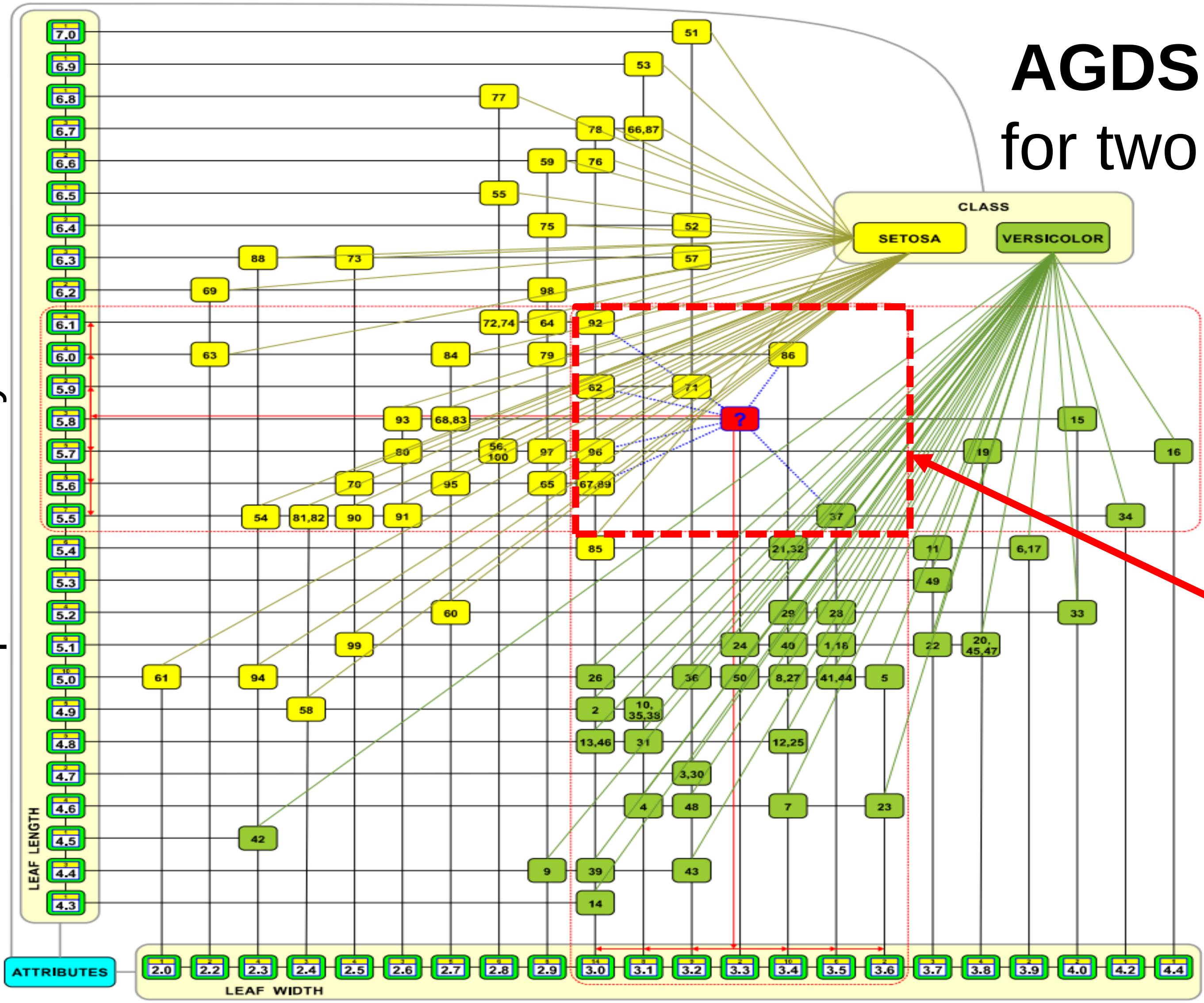
Brain inspired associative AGDS

The connections represent various relations between AGDS elements like similarity, proximity, neighborhood, order, definition etc.

K Nearest Neighbors using AGDS Structures

The search is limited to a small region where neighbors are found:

100 values represented by 28 value nodes!



100 values represented by 22 value nodes!

AGDS structure created for two selected attributes and 100 training samples of Iris data

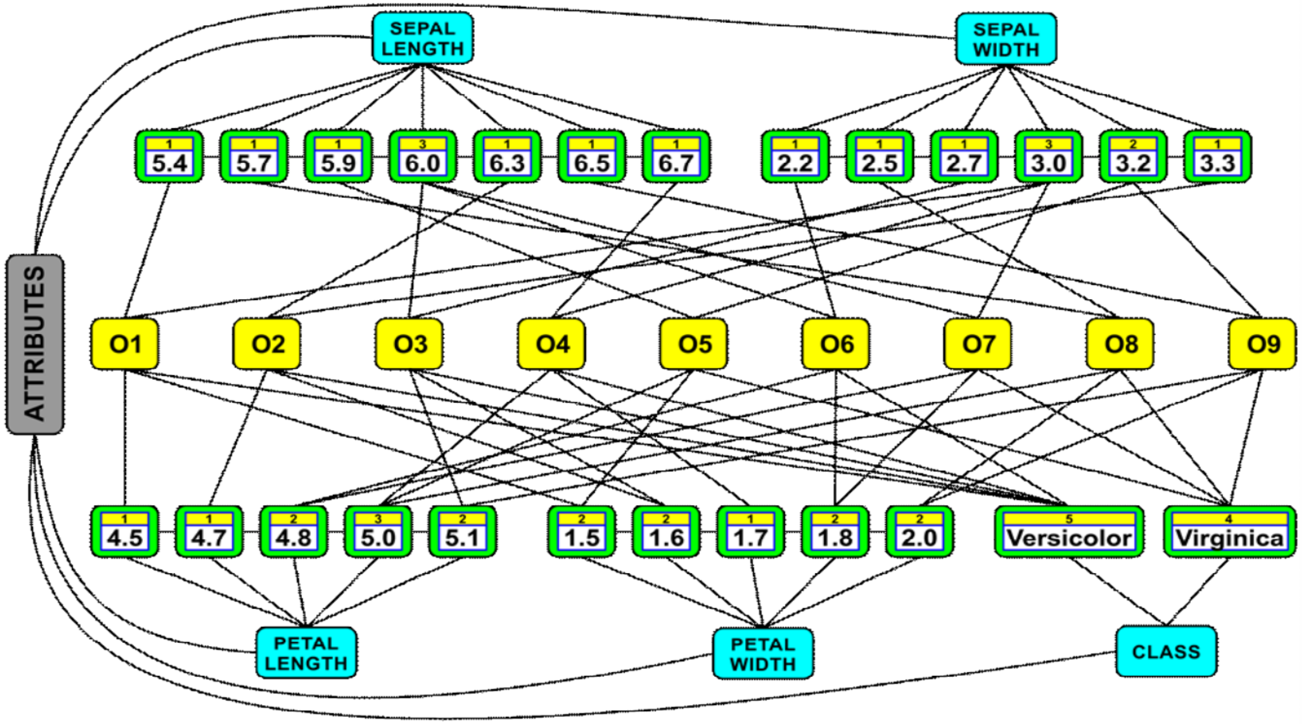
K Nearest Neighbors

are searched locally in the neighborhood of the classified sample

We can save a lot of computational time using created associations in the AGDS!

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.		
2.		
3.		

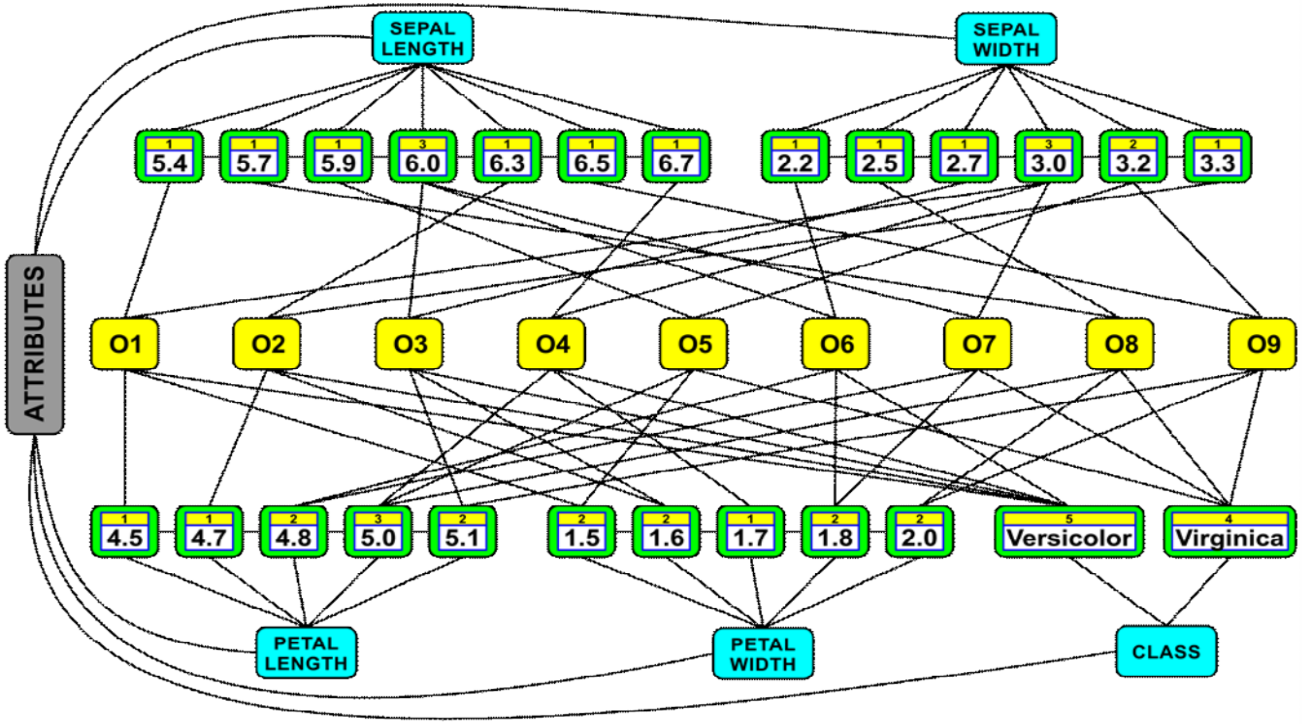


1. Create an empty k-row rank table that will consist of the pointers to the k nearest neighbors and their distances to the classified object.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.		
2.		
3.		

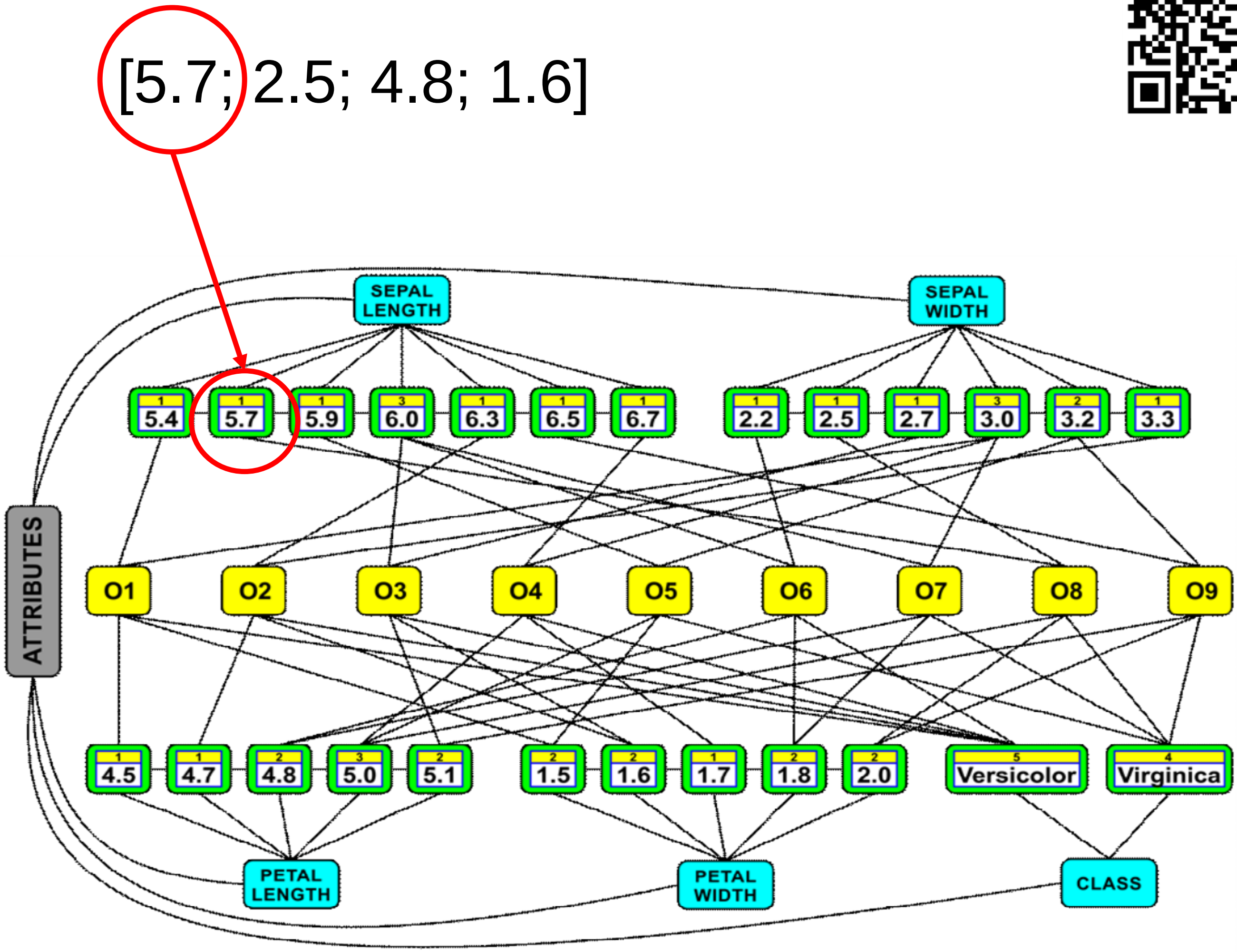
Classify [5.7; 2.5; 4.8; 1.6]



2. For the first attribute value of the classified object, find the closest attribute value in the constructed AGDS structure.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

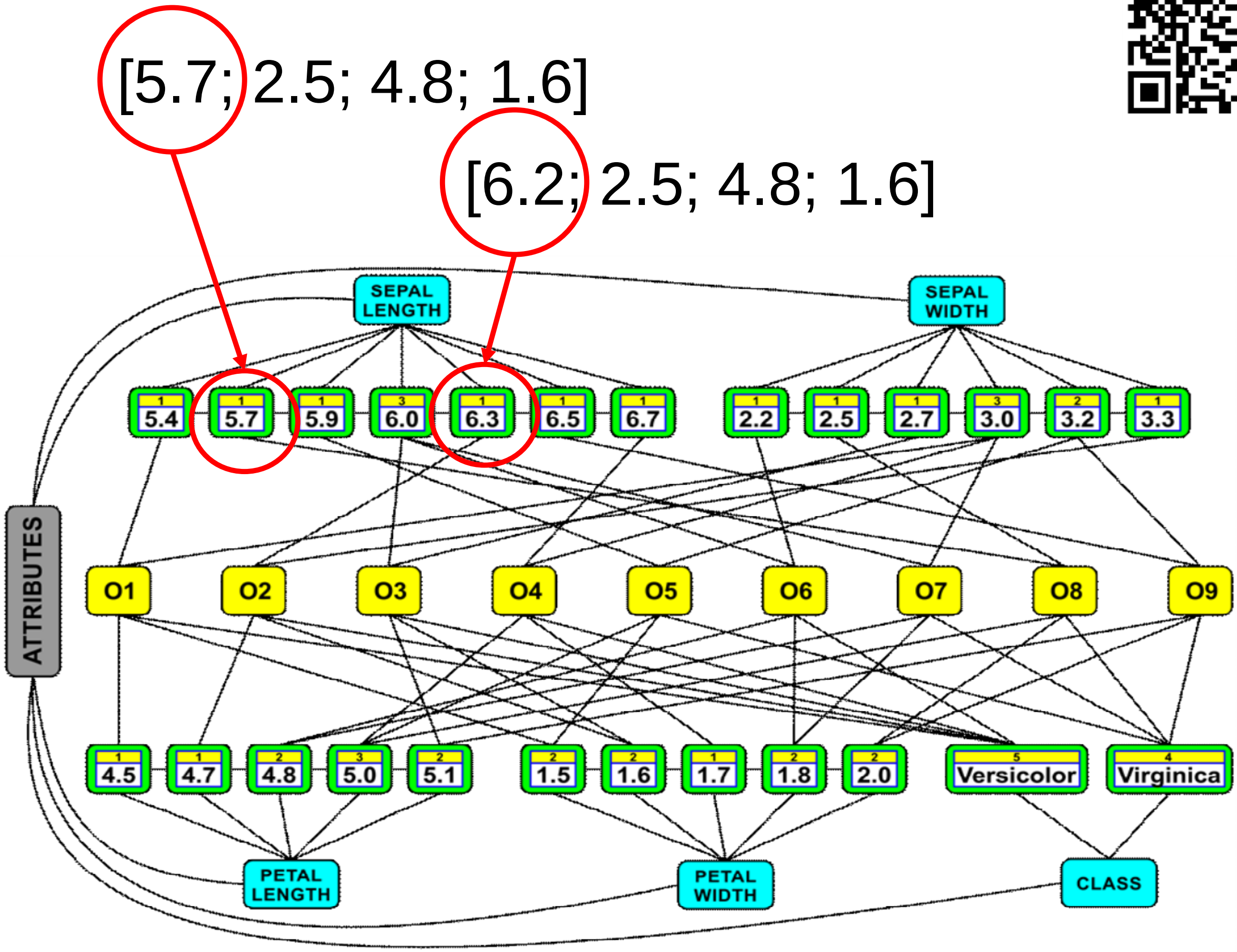
Rank table		
1.		
2.		
3.		



3. When the first attribute value of the classified object **is represented** by an existing value node of the AGDS structure, go to step 5, else go to step 4.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

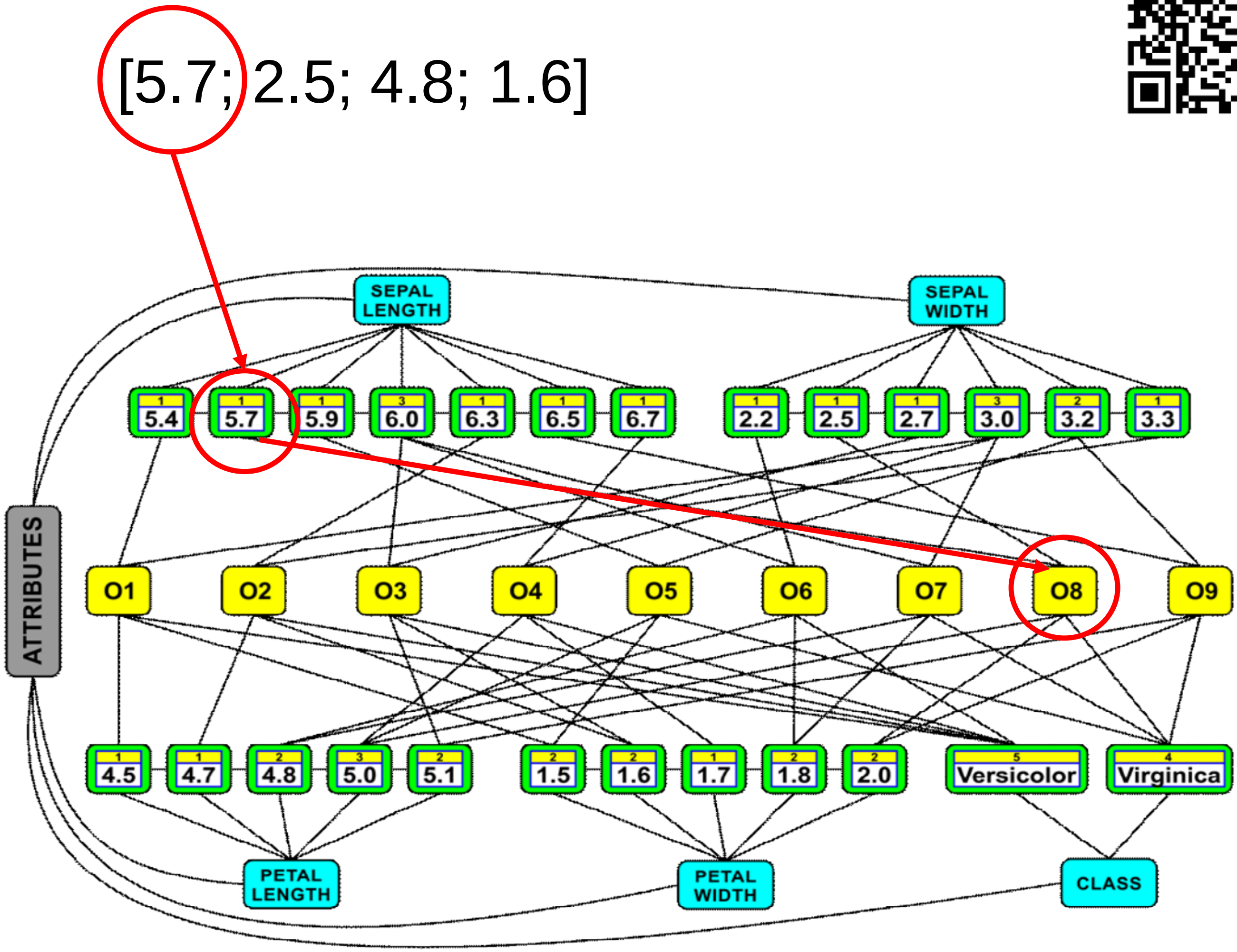
Rank table		
1.		
2.		
3.		



4. When the first attribute value of the classified object **is not represented** by any value node of this first attribute, then the closest value is represented by the value node representing the nearest lower or the nearest bigger value or both. Choose the nearest value or one of the nearest values and go to step 5.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.		
2.		
3.		



5. **Go along all edges** of the selected value node to all connected object nodes and perform step 6 for all these object nodes.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

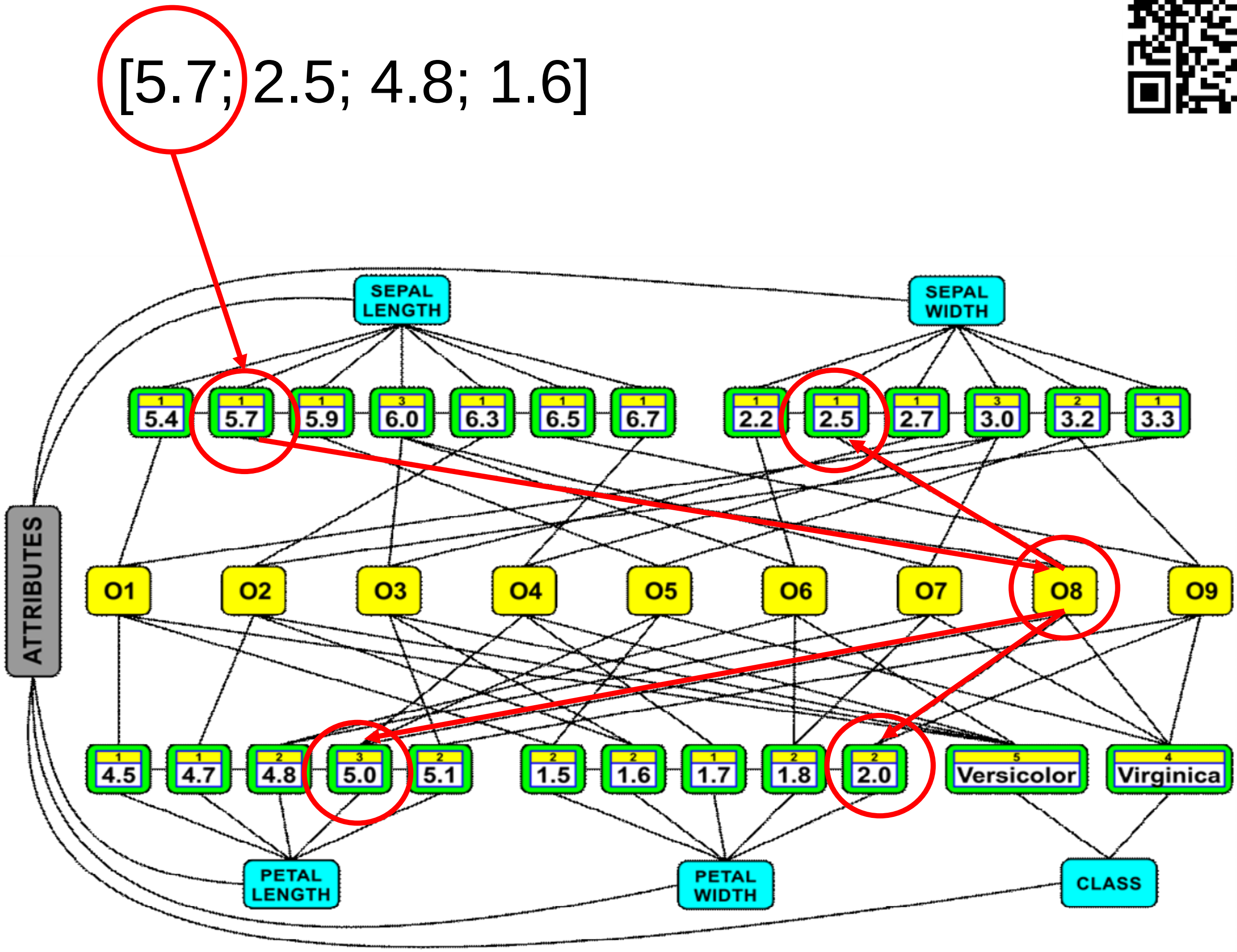
Rank table		
1.		
2.		
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$

$$d_e(x,y) = 0.45$$

$$d_m(x,y) = 0.60$$



6. For the reached object node, go to all connected value nodes, except the value node from which this object node was reached, and **compute the distance** according to (1) or (2). Next, try to insert this object node to the rank table in step 7.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

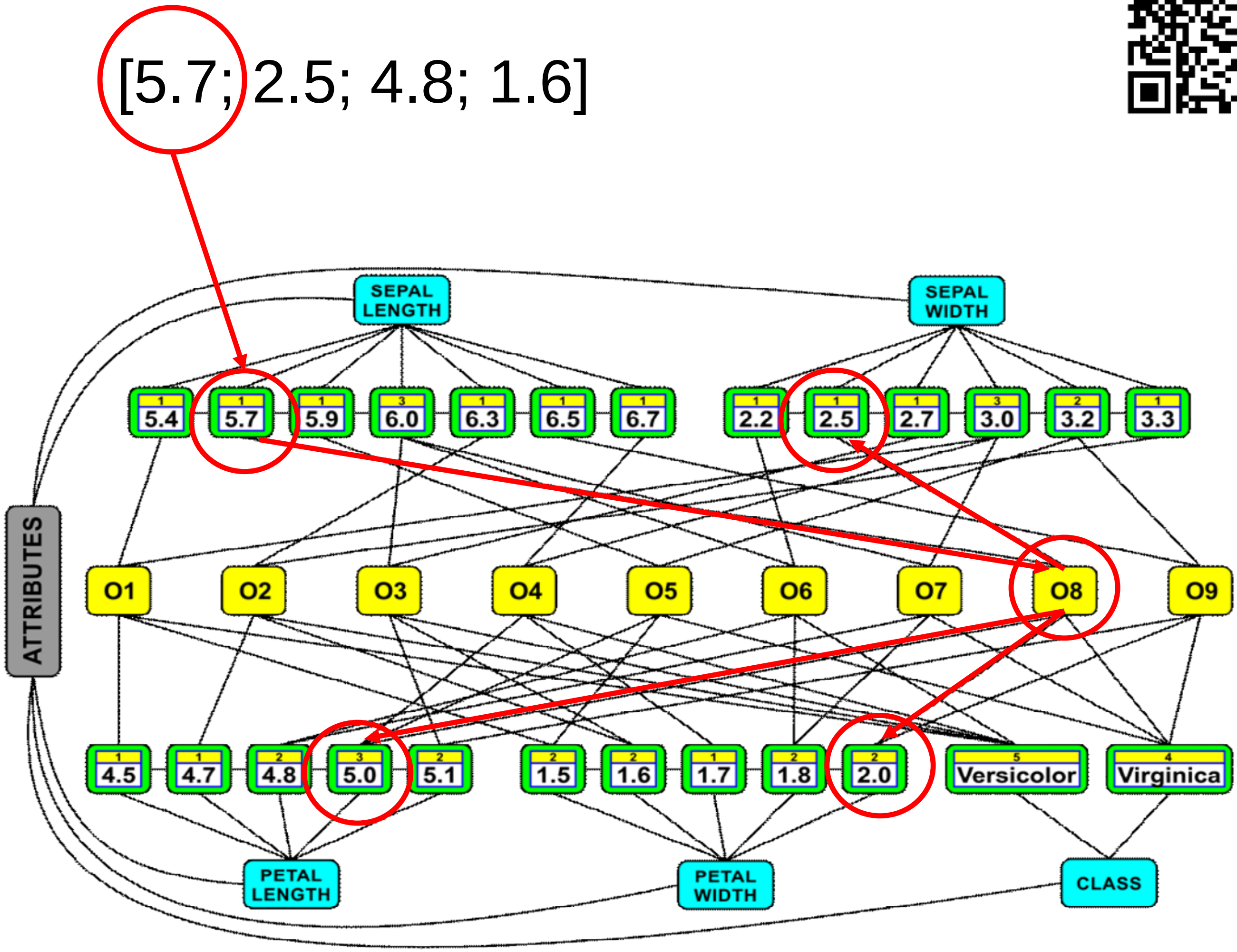
Rank table		
1.		
2.		
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n} |x_i - y_i| \quad (2)$$

$$d_e(x,y) = 0.45$$

$$d_m(x,y) = 0.60$$



7. If the k-th row of the rank table is empty or the computed distance is shorter than the distance to the object node stored in the last (k-th) row of the rank table, go to step 8, else go to step 9.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.		
3.		

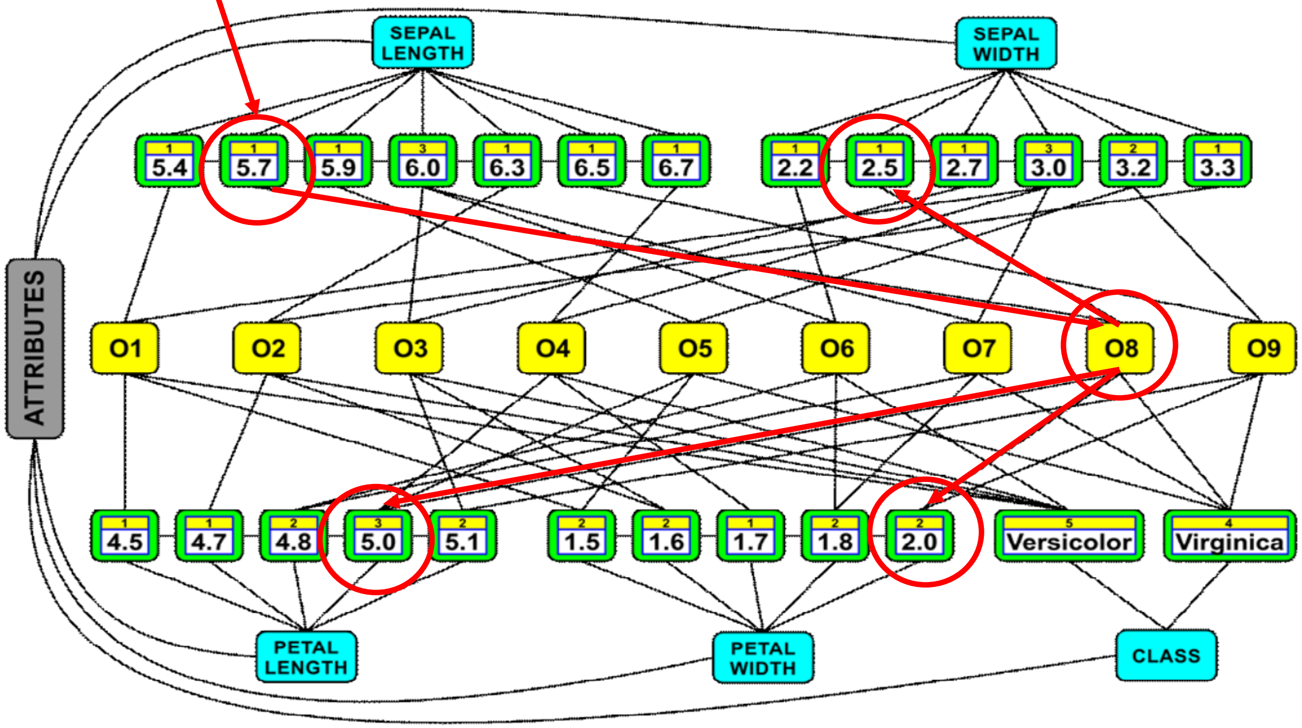
[5.7; 2.5; 4.8; 1.6]

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$

$d_e(x,y) =$ 0.45

$d_m(x,y) =$ 0.60



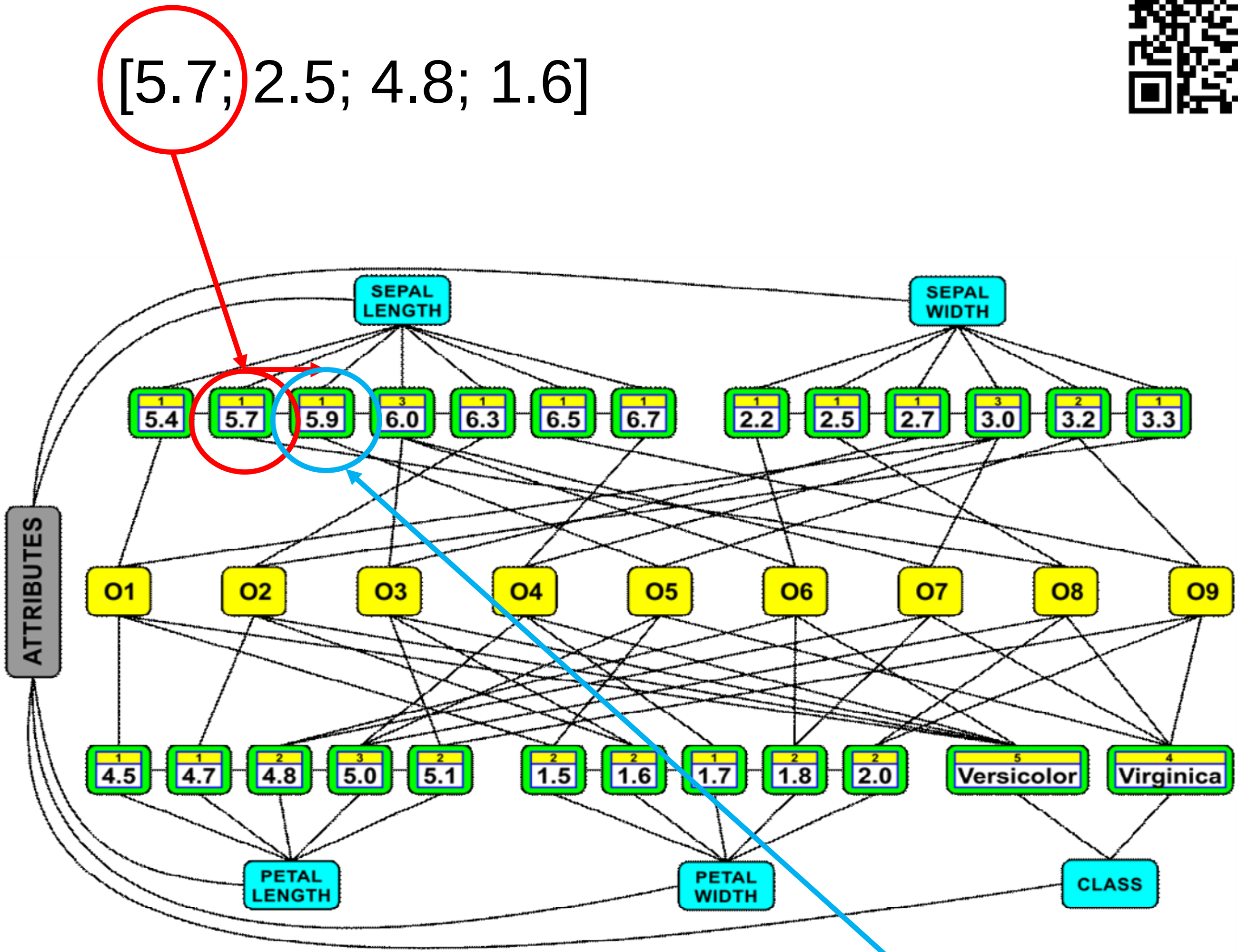
8. **Insert this node and its distance to the rank table in the ascendant order** (using (half) insertion sort algorithm), and if necessary (if the table is overfilled) remove the last (i.e. the most distant) object node together with its distance from this table.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.		
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n} |x_i - y_i| \qquad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

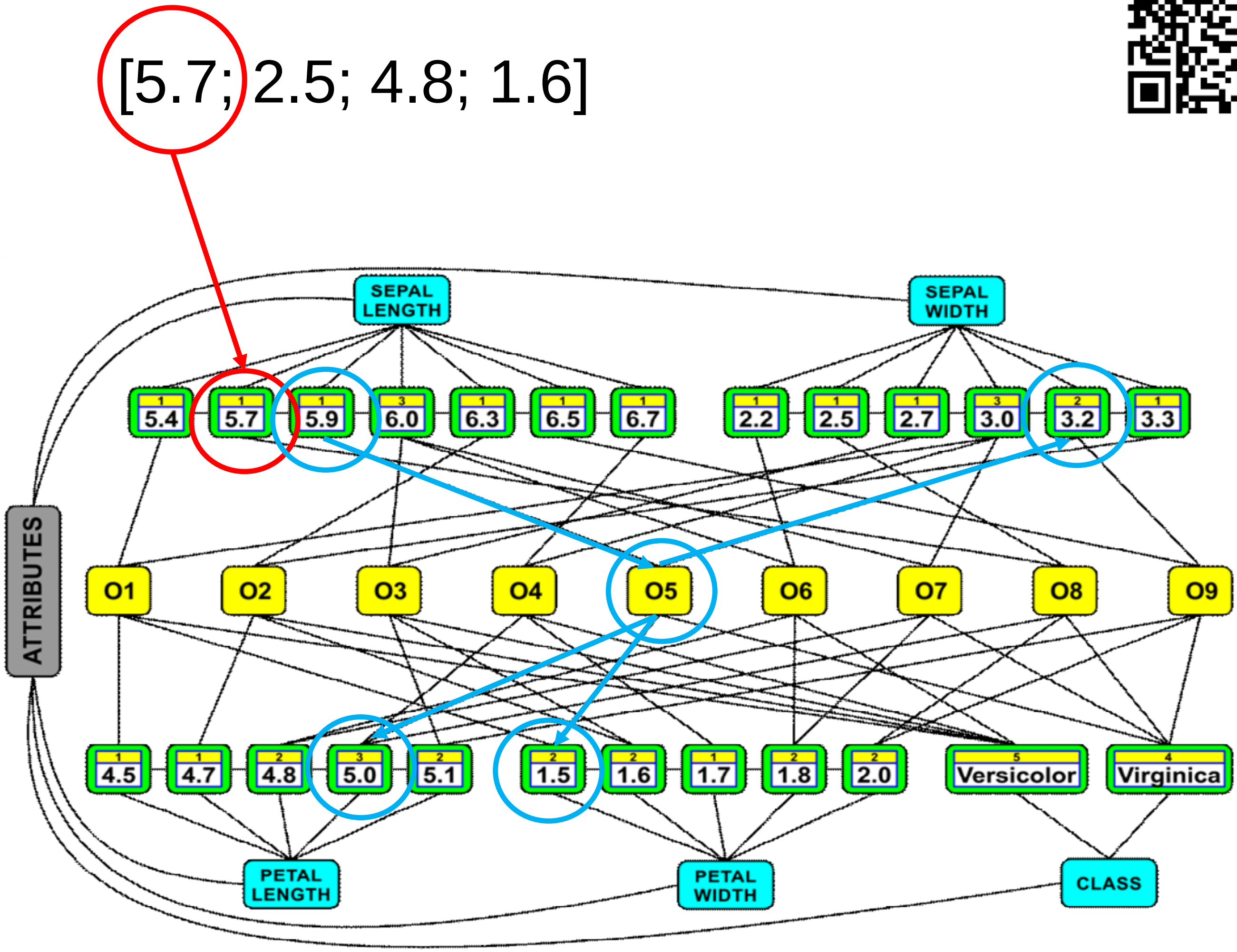
Rank table		
1.	O8	0.45
2.		
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n} |x_i - y_i| \quad (2)$$

$$d_e(x,y) = 0.48$$

$$d_m(x,y) = 0.90$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

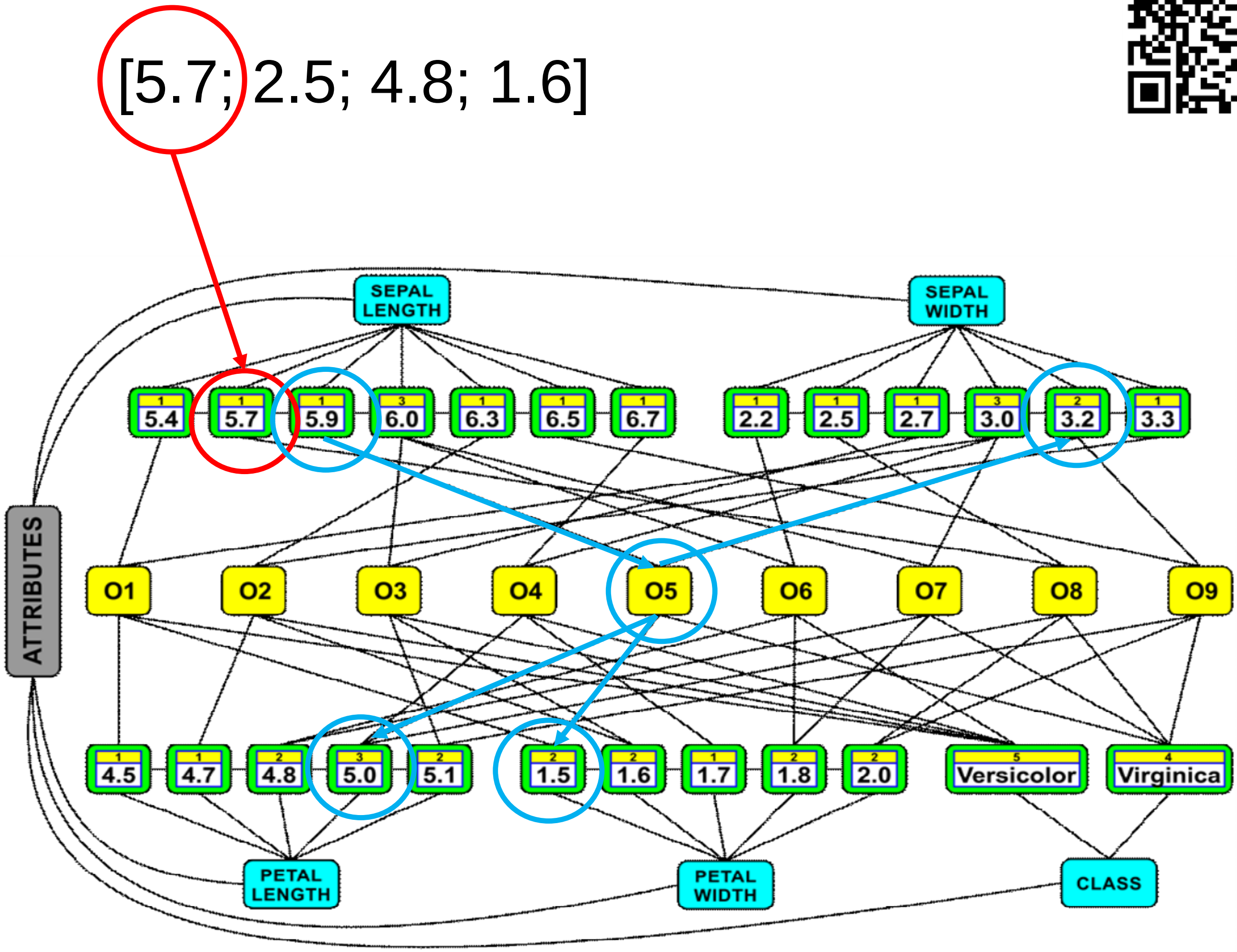
Rank table		
1.	O8	0.45
2.	O5	0.48
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$

$$d_e(x,y) = 0.48$$

$$d_m(x,y) = 0.90$$



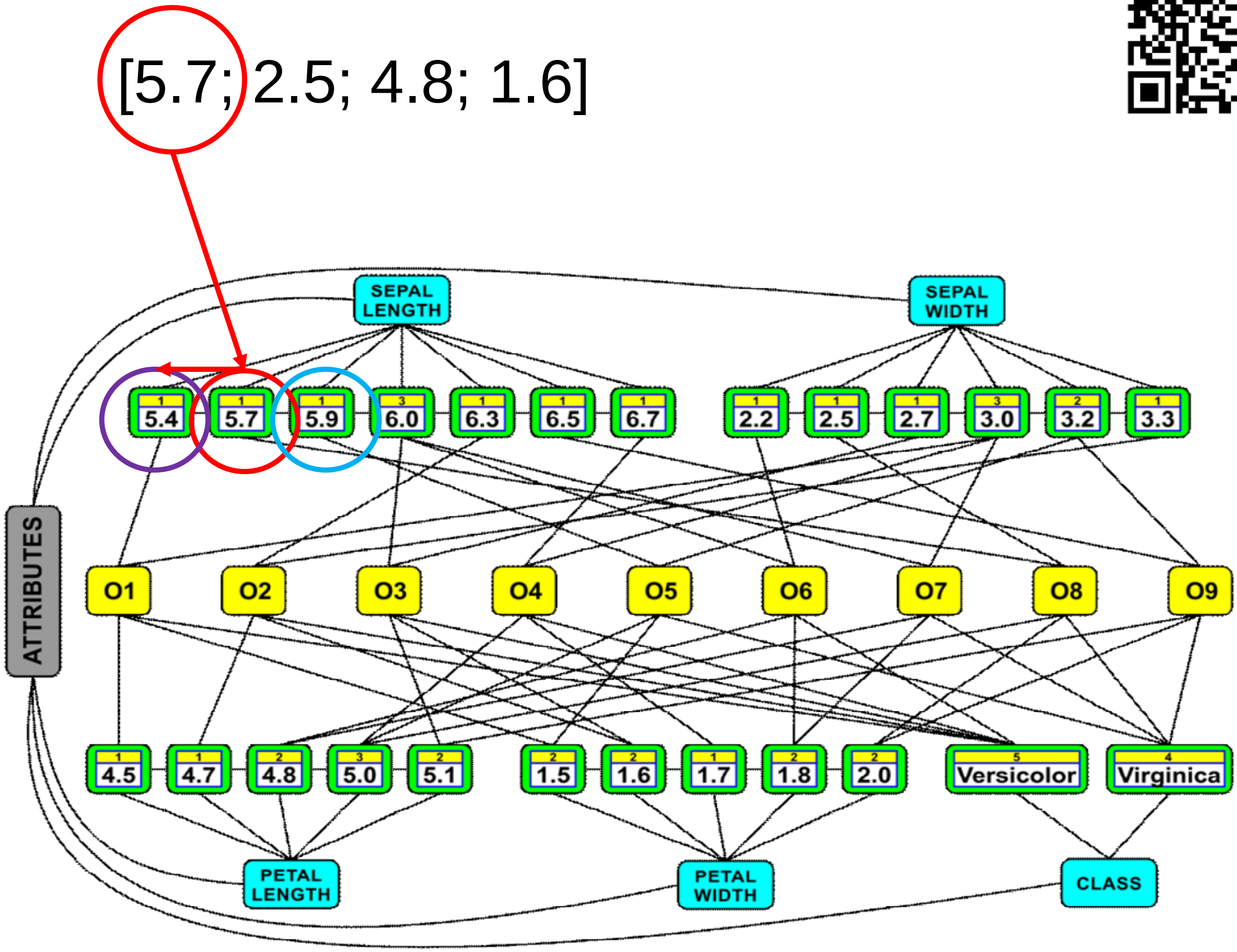
9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.	O5	0.48
3.		

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \qquad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

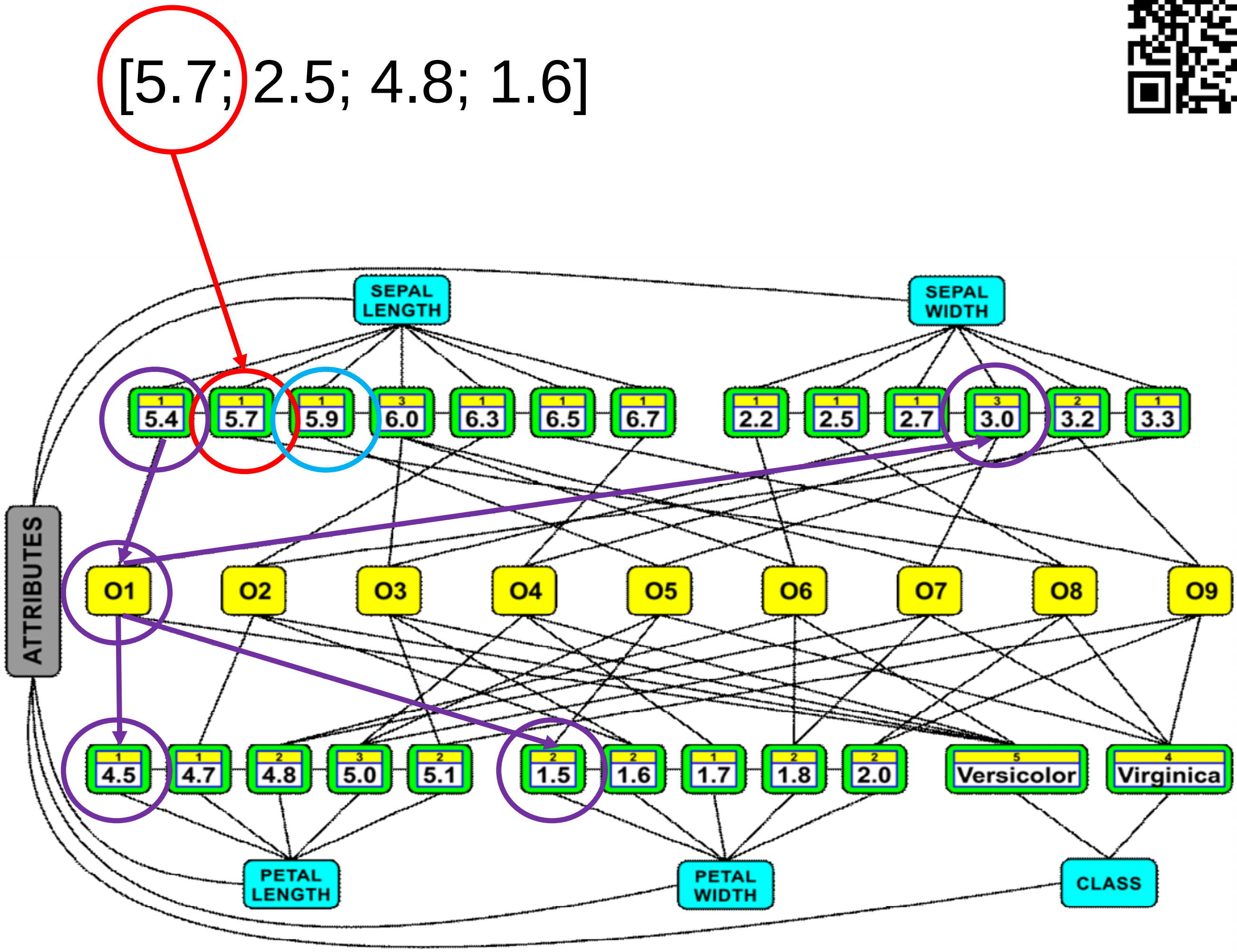
Rank table		
1.	O8	0.45
2.	O5	0.48
3.	O1	0.66

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$

$$d_e(x,y) = 0.66$$

$$d_m(x,y) = 1.2$$



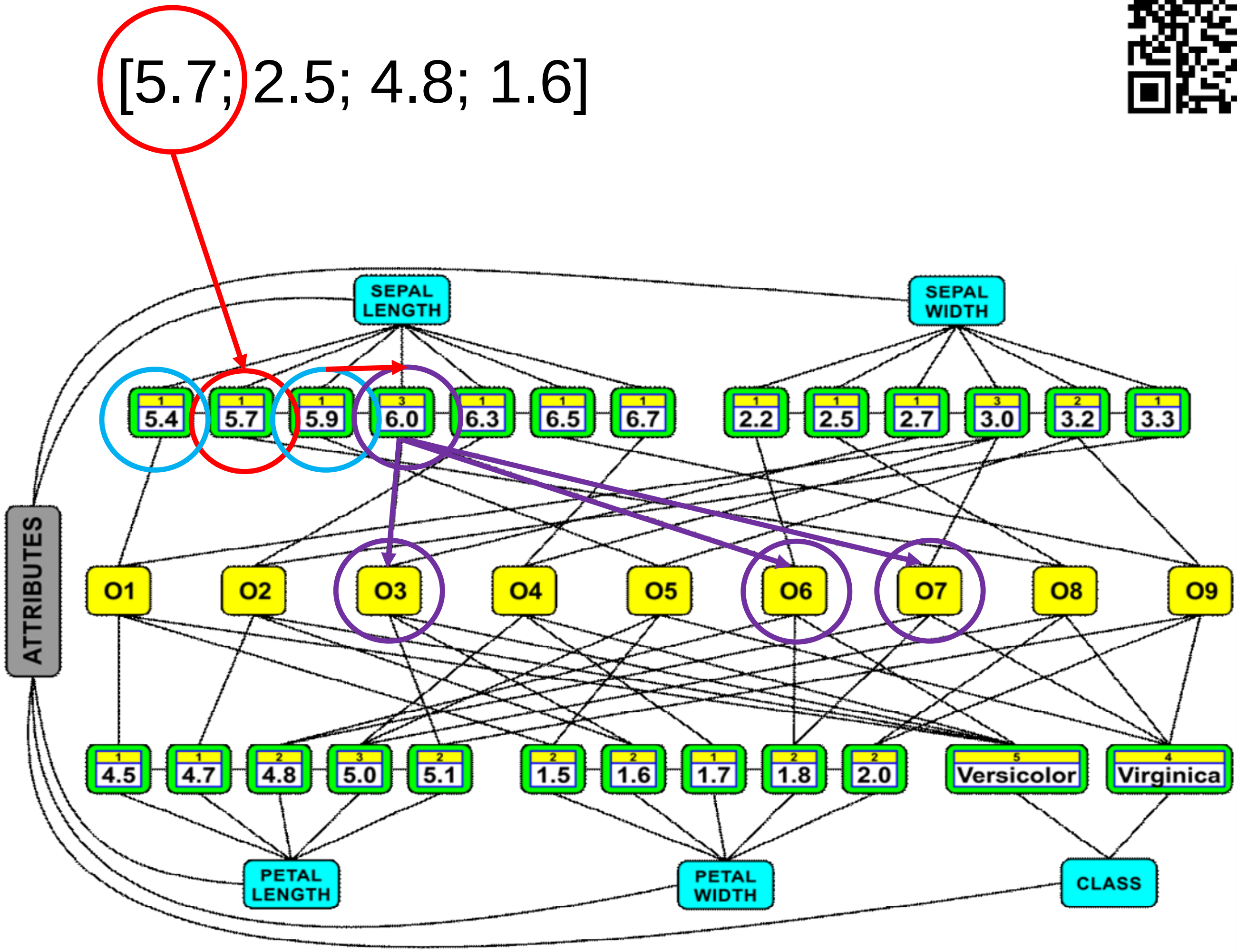
9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.	O5	0.48
3.	O1	0.66

$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \qquad (2)$$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.	O3	0.47
3.	O5	0.48
	~~O1~~	~~0.66~~

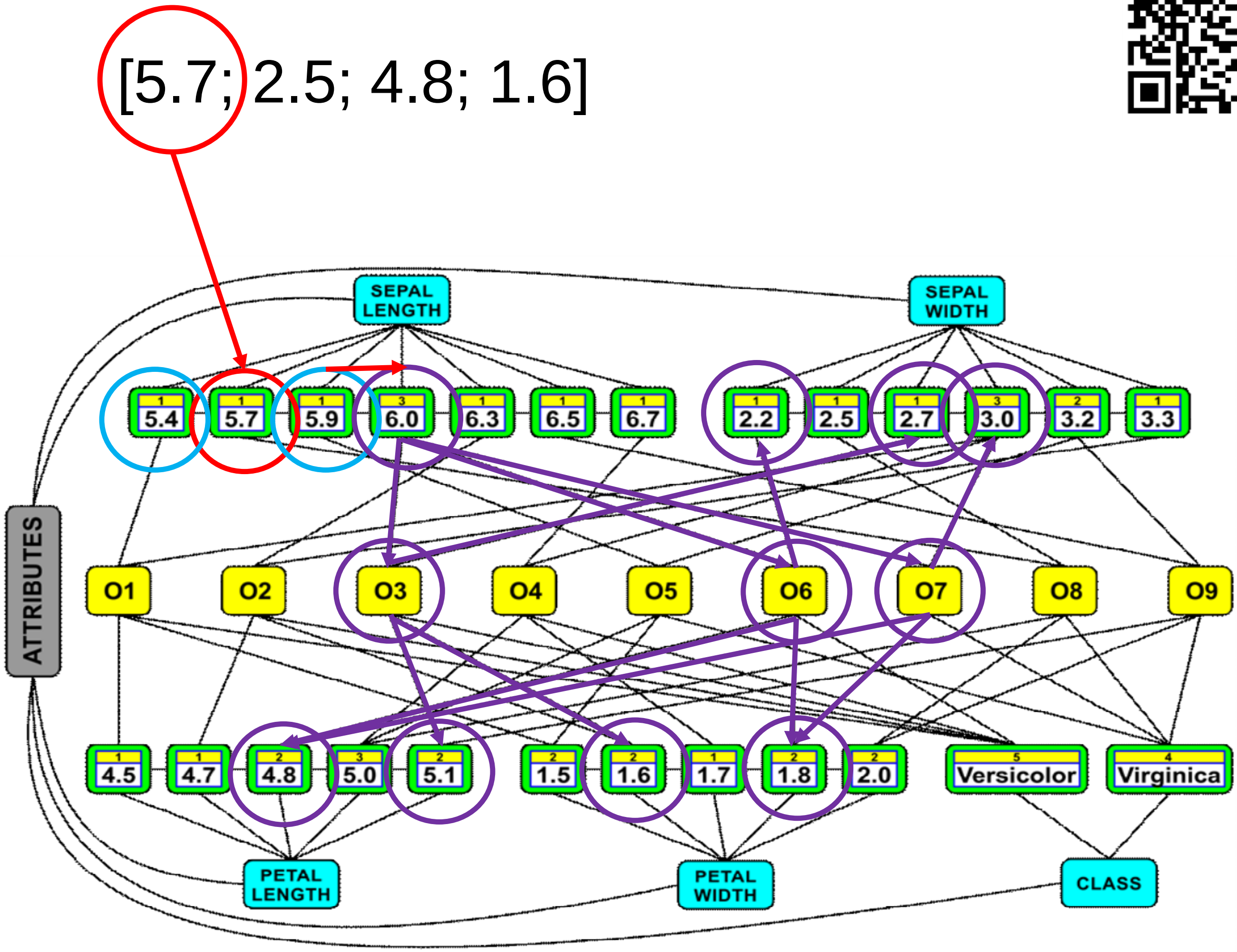
$$d_e(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x,y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$

O3
$d_e(x,y) = 0.47$
$d_m(x,y) = 0.8$

O6
$d_e(x,y) = 0.75$
$d_m(x,y) = 1.1$

O7
$d_e(x,y) = 0.62$
$d_m(x,y) = 1.0$



9. After checking all object nodes connected to the currently selected value node depicted as the closest to the first attribute value of the classified object from already not processed value nodes, go to **the next closest value node** (representing the lower or the bigger value to the first attribute value of the classified object).

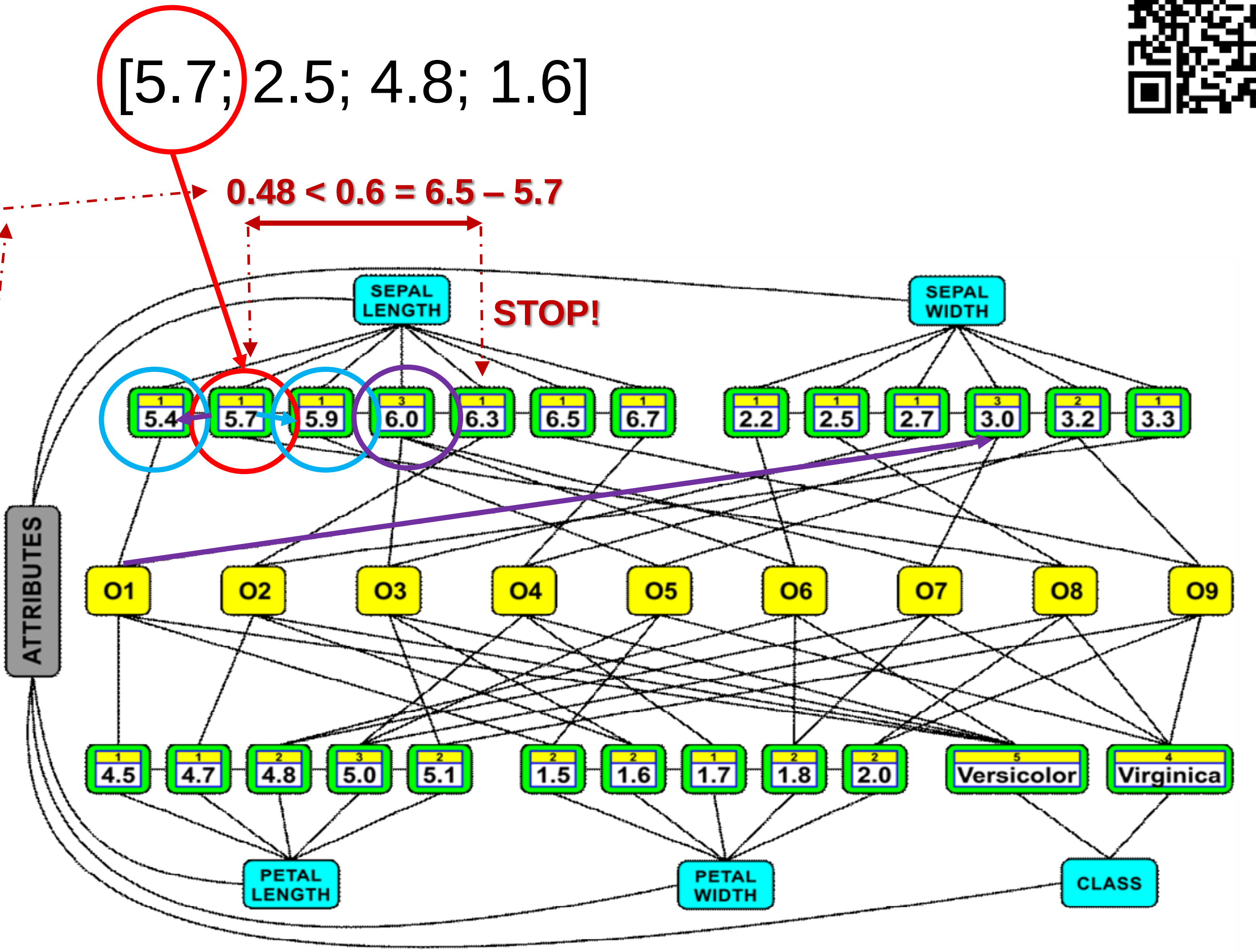
Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table		
1.	O8	0.45
2.	O3	0.47
3.	O5	0.48

K=3 Nearest Neighbors

$$d_e(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

$$d_m(x, y) = \sum_{i=1}^{n}|x_i - y_i| \quad (2)$$



STOP CONDITON

10. Next, go to step 5 **if the difference between these two values is less than the distance of the last object node stored in the rank table**; else finish the algorithm because the rank table already contains k nearest neighbors together with their distances to the classified object.

Acceleration Associative Algorithm for KNN+ AGDS classifiers

Rank table

1.	O8	0.45	Virginica
2.	O3	0.47	Versicolor
3.	O5	0.48	Virginica

K Nearest Neighbors

[5.7; 2.5; 4.8; 1.6]

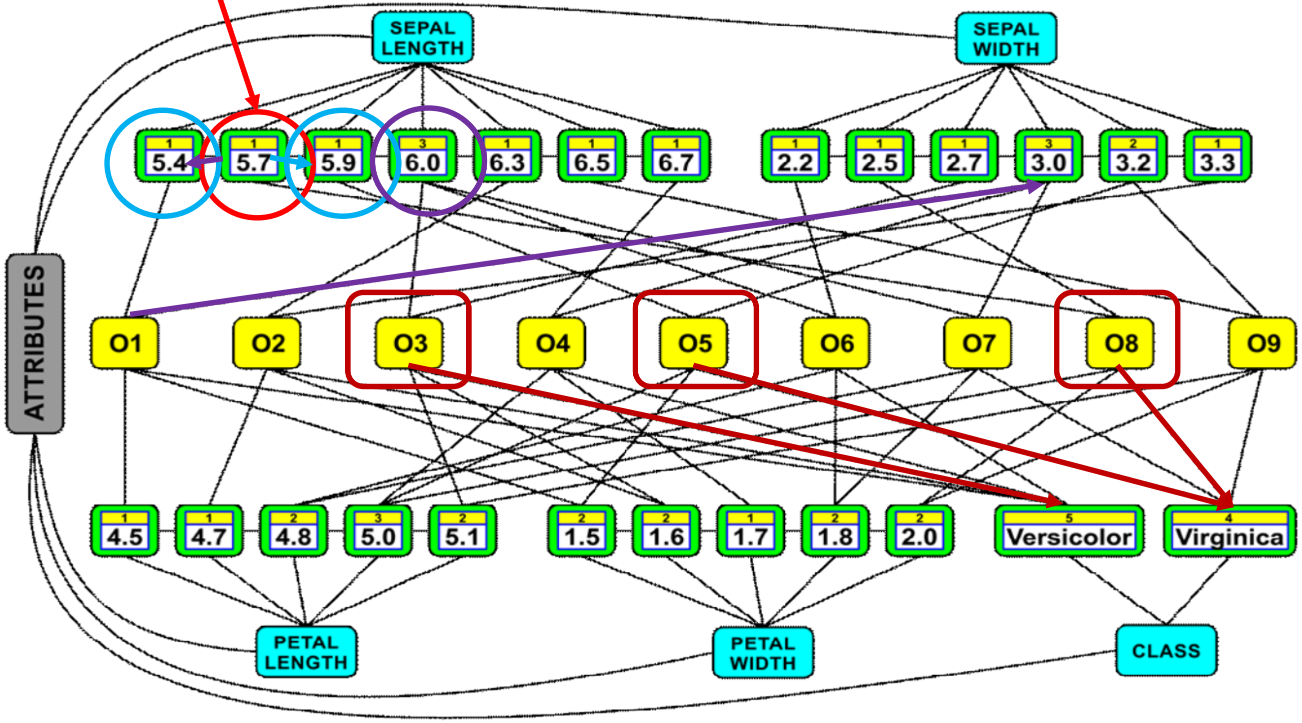
Votes

2 x Virginica

1 x Versicolor

WINNING CLASS

	ATTRIBUTES				
SAMPLE OBJECTS	SEPAL LENGTH	SEPAL WIDTH	PETAL LENGTH	PETAL WIDTH	CLASS LABEL
O1	5.4	3.0	4.5	1.5	Versicolor
O2	6.3	3.3	4.7	1.6	Versicolor
O3	6.0	2.7	5.1	1.6	Versicolor
O4	6.7	3.0	5.0	1.7	Versicolor
O5	6.0	2.2	5.0	1.5	Virginica
O6	5.9	3.2	4.8	1.8	Versicolor
O7	6.0	3.0	4.8	1.8	Virginica
O8	5.7	2.5	5.0	2.0	Virginica
O9	6.5	3.2	5.1	2.0	Virginica



11. Count votes and check which class has the most votes to establish the winning class to get the KNN classification.

Comparison of Results and Efficiencies

Time Efficiency:

Table 2. Comparison of classification time using kNN and kNN+AGDS.

Dataset	Number of instances	Number of attributes	kNN classification time [ms]	kNN+AGDS classification time [ms]	kNN+AGDS construction time [ms]
Iris	150	4	0.10	0.08	1
Banknote	1372	4	0.29	0.09	5
HTRU2	17898	8	3.14	0.09	134
Shuttle	43500	9	7.67	1.06	278
Credit Card	30000	23	8.69	1.07	499
Skin	245057	3	26.87	1.10	683
Drive	58509	48	46.15	1.24	2224
HEPMASS	1048576	28	362.32	1.41	31214

Classification time for KNN+AGDS is almost constant regardless of the size of the used training data sets.

Comparison of Results and Efficiencies

Time Efficiency as a function of the number of instances and attributes:

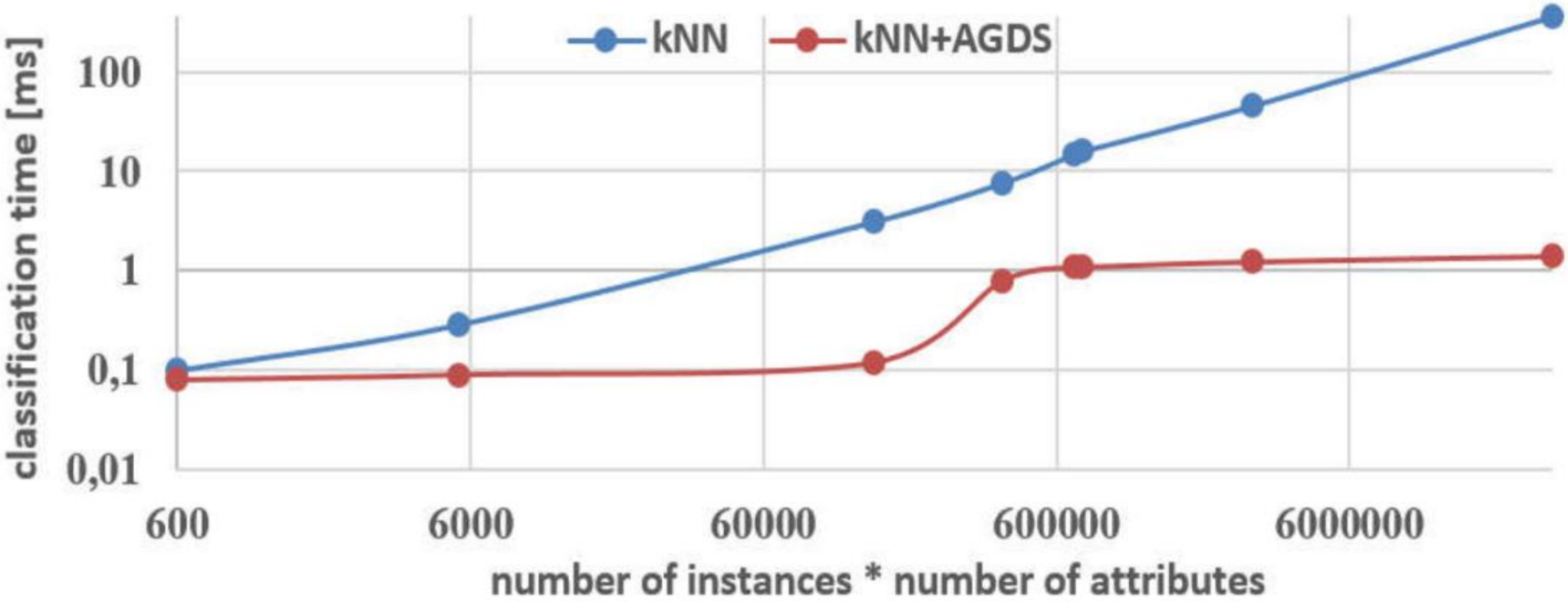


Fig. 2. Classification time as a function of the number of the instances multiplied by the number of attributes, i.e. the number of data stored in the training data tables.

The size of training data and the number of attributes do not influence KNN+AGDS efficiency as is in the classic KNN classifiers.

Comparison of Results and Efficiencies

Memory Efficiency:

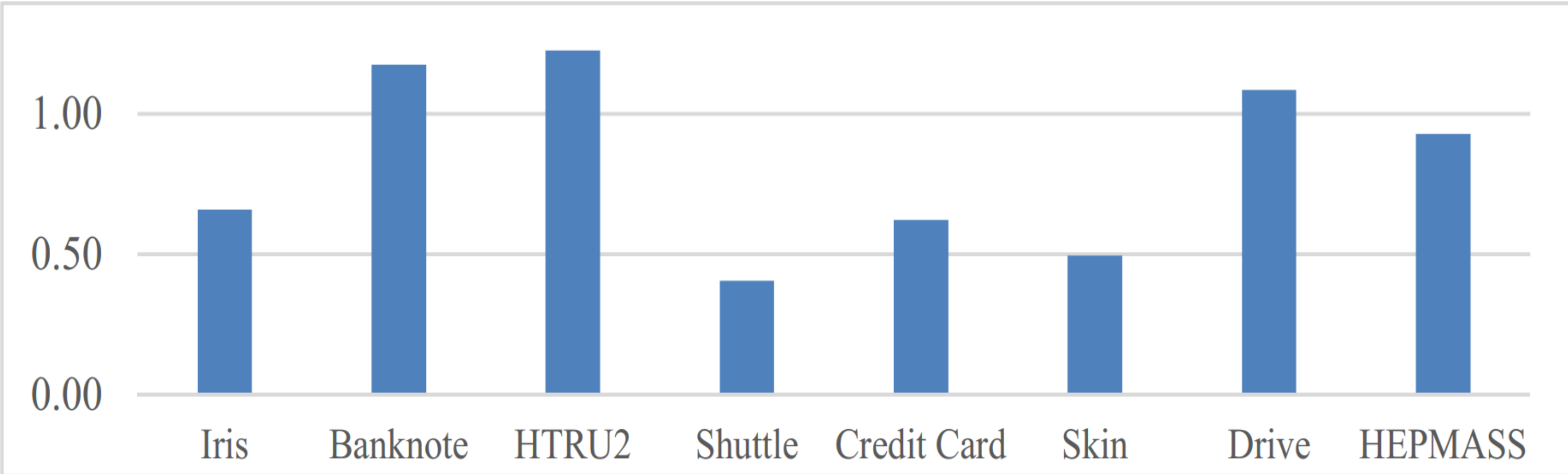


Fig. 3. Memory usage ratio of using AGDS structures to arrays for various training data.

KNN+AGDS classifiers also usually **use less memory** than KNN classifiers due to the number of duplicated values in training data.

Conclusions and Important Remarks

- ✓ AGDS structures provide high-speed access to neighbor values and similar objects because of the aggregations and ordering of all values simultaneously for all attributes.
- ✓ AGDS stores data together with the most common vertical and horizontal relations, so there is no need to loop and search for these relations wasting resources.
- ✓ Typical operations on the AGDS structures take logarithmic time of the number of unique attribute values that are operated, but the expected complexity on real data containing many duplicates is usually constant.
- ✓ The efficiency of the presented classification algorithm using AGDS structures grows with the amount of the training data.

Algorithm, pseudocode … can be found in the paper

Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers

Adrian Horzyk(✉) and Krzysztof Gołdon

AGH University of Science and Technology, Krakow, Poland
horzyk@agh.edu.pl, krzysztofgoldon@gmail.com

Abstract. This paper introduces a new associative approach for significant acceleration of k Nearest Neighbor classifiers (kNN). The kNN classifier is a lazy method, i.e. it does not create a computational model, so it is inefficient during classification using big training data sets because it requires going through all training patterns when classifying each sample. In this paper, we propose to use Associative Graph Data Structures (AGDS) as an efficient model for storing training patterns and their relations, allowing for fast access to nearest neighbors during classification made by kNNs. Hence, the AGDS significantly accelerates the classification made by kNNs, especially for large and huge training datasets. In this paper, we introduce an Associative Acceleration Algorithm and demonstrate how it works on this associative structure substantially reducing the number of checked patterns and quickly selecting k nearest neighbors for kNNs. The presented approach was compared to classic kNN approaches successfully.

```
FindNextClosest(val)
if (valueNodeLessClosest == null) and (valueNodeGreaterClosest == null)
then
   valueNodeLessClosest = BinSearchEqualOrLess(val)
   if (valueNodeLessClosest == null)
   then valueNodeGreaterClosest = First
        return valueNodeGreaterClosest
   else if (valueNodeLessClosest.Val == val)
      then valueNodeGreaterClosest = valueNodeLessClosest
           return valueNodeGreaterClosest
      else if (valueNodeLessClosest.IsNotMax)
            then valueNodeGreaterClosest = valueNodeLessClosest.Next
                 if   (val-valueNodeLessClosest.Val   <   valueNodeGreaterClosest.Val-
            val)
                 then return valueNodeLessClosest
                 else return valueNodeGreaterClosest
            else valueNodeGreaterClosest = null
                  return valueNodeLessClosest
else if (val - valueNodeLessClosest.Val < valueNodeGreaterClosest.Val - val)
   then if (valueNodeLessClosest.IsNotMin)
         then valueNodeLessClosest = valueNodeLessClosest.Prev
         else if (valueNodeGreaterClosest.IsNotMax)
               then valueNodeGreaterClosest = valueNodeGreaterClosest.Next
               else return null
   else if (valueNodeGreaterClosest.IsNotMax)
         then valueNodeGreaterClosest = valueNodeGreaterClosest.Next
         else if (valueNodeLessClosest.IsNotMin)
               then valueNodeLessClosest = valueNodeLessClosest.Prev
               else return null
   if (val - valueNodeLessClosest.Val < valueNodeGreaterClosest.Val - val)
   then return valueNodeLessClosest
   else return valueNodeGreaterClosest
```

A. Horzyk and K. Gołdon, Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers, In: 27th International Conference on Artificial Neural Networks (ICANN 2018), Springer-Verlag, LNCS 11139, pp. 648-658, 2018.

Bibliography and Literature

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016, ISBN 978-1-59327-741-3 or PWN 2018.
2. Holk Cruse, Neural Networks as Cybernetic Systems, 2nd and revised edition
3. R. Rojas, Neural Networks, Springer-Verlag, Berlin, 1996.
4. A. Horzyk, Associative Graph Data Structures with an Efficient Access via AVB+trees, In: 2018 11th International Conference on Human System Interaction (HSI), 2018, IEEE Xplore, pp. 169 - 175.
5. A. Horzyk and K. Gołdon, Associative Graph Data Structures Used for Acceleration of K Nearest Neighbor Classifiers, In: 27th International Conference on Artificial Neural Networks (ICANN 2018), Springer-Verlag, LNCS 11139, pp. 648-658, 2018.

AGH

AGH University of

Science and Technology

Krakow, Poland

Adrian Horzyk
horzyk@agh.edu.pl