K-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees[1, 2]

Maleq Khan, Qin Ding, and William Perrizo

Computer Science Department, North Dakota State University
Fargo, ND 58105, USA
{Md.Khan, Qin.Ding, William.Perrizo}@ndsu.nodak.edu

Abstract. Classification of spatial data streams is crucial, since the training dataset changes often. Building a new classifier each time can be very costly with most techniques. In this situation, k-nearest neighbor (KNN) classification is a very good choice, since no residual classifier needs to be built ahead of time. KNN is extremely simple to implement and lends itself to a wide variety of variations. We propose a new method of KNN classification for spatial data using a new, rich, data-mining-ready structure, the *Peano-count-tree* (*P-tree*). We merely perform some AND/OR operations on P-trees to find the nearest neighbors of a new sample and assign the class label. We have fast and efficient algorithms for the AND/OR operations, which reduce the classification time significantly. Instead of taking exactly the k nearest neighbors we form a closed-KNN set. Our experimental results show closed-KNN yields higher classification accuracy as well as significantly higher speed.

1. Introduction

There are various techniques for classification such as Decision Tree Induction, Bayesian Classification, and Neural Networks [7, 8]. Unlike other common classifiers, a *k-nearest neighbor* (KNN) classifier does not build a classifier in advance. That is what makes it suitable for data streams. When a new sample arrives, KNN finds the k neighbors nearest to the new sample from the training space based on some suitable similarity or distance metric. The plurality class among the nearest neighbors is the class label of the new sample [3, 4, 5, 7]. A common similarity function is based on the Euclidian distance between two data tuples [7]. For two tuples, $X = <x_1, x_2, x_3, \ldots, x_{n-1}>$ and $Y = <y_1, y_2, y_3, \ldots, y_{n-1}>$ (excluding the class labels), the *Euclidian* similarity function is $d_2(X,Y) = \sqrt{\sum_{i=1}^{n-1}(x_i - y_i)^2}$. A generalization of the Euclidean function is the *Minkowski* similarity function is

[1] Patents are pending on the bSQ and Ptree technology.
[2] This work is partially supported by NSF Grant OSR-9553368, DARPA Grant DAAH04-96-1-0329 and GSA Grant ACT#: K96130308.

$d_q(X,Y) = \sqrt[q]{\sum_{i=1}^{n-1} w_i |x_i - y_i|^q}$. The Euclidean function results by setting q to 2 and each weight, w_i, to 1. The *Manhattan* distance, $d_1(X,Y) = \sum_{i=1}^{n-1} |x_i - y_i|$ result by setting q to 1. Setting q to ∞, results in the *max* function $d_\infty(X,Y) = \max_{i=1}^{n-1} |x_i - y_i|$.

In this paper, we introduced a new metric called *Higher Order Bit* (*HOB*) similarity metric and evaluated the effect of all of the above distance metrics in classification time and accuracy. HOB distance provides an efficient way of computing neighborhoods while keeping the classification accuracy very high.

KNN is a good choice when simplicity and accuracy are the predominant issues. KNN can be superior when a residual, trained and tested classifiers, such as ID3, has a short useful lifespan, such as in the case with data streams, where new data arrives rapidly and the training set is ever changing [1, 2]. For example, in spatial data, AVHRR images are generated in every one hour and can be viewed as spatial data streams. The purpose of this paper is to introduce a new KNN-like model, which is not only simple and accurate but is also fast – fast enough for use in spatial data stream classification.

In this paper we propose a simple and fast KNN-like classification algorithm for spatial data using P-trees. P-trees are new, compact, data-mining-ready data structures, which provide a lossless representation of the original spatial data [6, 9, 10]. In the section 2, we review the structure of P-trees and various P-tree operations.

We consider a space to be represented by a 2-dimensional array of locations. Associated with each location are various attributes, called ***bands***, such as visible reflectance intensities (blue, green and red), infrared reflectance intensities (e.g., NIR, MIR1, MIR2 and TIR) and possibly crop yield quantities, soil attributes and radar reflectance intensities. We refer to a location as a pixel in this paper.

Using P-trees, we presented two algorithms, one based on the max distance metric and the other based on our new HOB distance metric. HOB is the similarity of the most significant bit positions in each band. It differs from pure Euclidean similarity in that it can be an asymmetric function depending upon the bit arrangement of the values involved. However, it is very fast, very simple and quite accurate. Instead of using exactly k nearest neighbor (a KNN set), our algorithms build a *closed-KNN* set and perform voting on this closed-KNN set to find the predicting class. Closed-KNN, a superset of KNN, is formed by including the pixels, which have the same distance from the target pixel as some of the pixels in KNN set. Based on this similarity measure, finding nearest neighbors of new samples (pixel to be classified) can be done easily and very efficiently using P-trees and we found higher classification accuracy than traditional methods on considered datasets. Detailed definitions of the similarity and the algorithms to find nearest neighbors are given in the section 3. We provided experimental results and analyses in section 4. The conclusion is given in Section 5.

2. **P-tree Data Structures**

Most spatial data comes in a format called *BSQ* for *Band Sequential* (or can be easily converted to BSQ). BSQ data has a separate file for each band. The ordering of the data values within a band is raster ordering with respect to the spatial area represented in the dataset. We divided each BSQ band into several files, one for each bit position of the data values. We call this format *bit Sequential* or *bSQ* [6, 9, 10]. A Landsat Thematic Mapper satellite image, for example, is in BSQ format with 7 bands, $B_1,\ldots,B_7$, (Landsat-7 has 8) and ~40,000,000 8-bit data values. A typical *TIFF* image aerial digital photograph is one file containing ~24,000,000 bits ordered by it-position, then band and then raster-ordered-pixel-location.

We organize each bSQ bit file, B_{ij} (the file constructed from the j^{th} bits of i^{th} band), into a tree structure, called a *Peano Count Tree* (*P-tree*). A P-tree is a quadrant-based tree. The root of a P-tree contains the 1-bit count, called *root count*, of the entire bit-band. The next level of the tree contains the 1-bit counts of the four quadrants in raster order. At the next level, each quadrant is partitioned into sub-quadrants and their 1-bit counts in raster order constitute the children of the quadrant node. This construction is continued recursively down each tree path until the sub-quadrant is *pure* (entirely 1-bits or entirely 0-bits). Recursive raster ordering is called the Peano or Z-ordering in the literature – therefore, the name Peano Count trees. The P-tree for a 8-row-8-column bit-band is shown in Fig. 1.

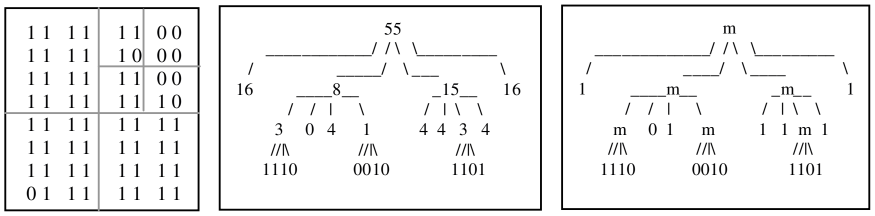


Fig. 1. 8-by-8 image and its P-tree (P-tree and PM-tree)

In this example, root count is 55, and the counts at the next level, 16, 8, 15 and 16, are the 1-bit counts for the four major quadrants. Since the first and last quadrant is made up of entirely 1-bits, we do not need sub-trees for these two quadrants.

For each band (assuming 8-bit data values), we get 8 *basic P-trees*. $P_{i,j}$ is the P-tree for the j^{th} bits of the values from the i^{th} band. For efficient implementation, we use variation of basic P-trees, called *Pure Mask tree* (*PM-tree*). In the PM-tree, we use a 3-value logic, in which 11 represents a quadrant of pure 1-bits, *pure1* quadrant, 00 represents a quadrant of pure 0-bits, *pure0* quadrant, and 01 represents a mixed quadrant. To simplify the exposition, we use 1 instead of 11 for pure1, 0 for pure0, and *m* for mixed. The PM-tree for the previous example is also given in Fig. 1.

P-tree algebra contains operators, AND, OR, NOT and XOR, which are the pixel-by-pixel logical operations on P-trees. The AND/OR operations on PM-trees are shown in Fig. 2. The AND operation between two PM-trees is performed by ANDing the corresponding nodes in the two operand PM-trees level-by-level starting from the root node. A pure0 node ANDed with any node produces a pure0 node. A pure1 node ANDed with any node, *n*, produces the node *n*; we just need to copy the node, *n*, to the resultant PM-tree in the corresponding position. ANDing of two mixed nodes produces a mixed node; the children of the resultant mixed node are obtained by ANDing children of the operands recursively. The details of P-tree operations can be found in [10, 11].

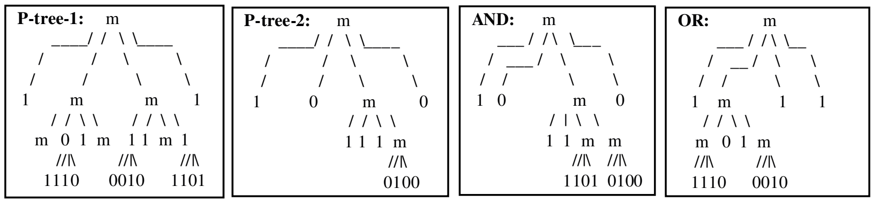


Fig. 2. P-tree Algebra

3. The Classification Algorithms

In the original k-nearest neighbor (KNN) classification method, no classifier model is built in advance. KNN refers back to the raw training data in the classification of each new sample. Therefore, one can say that the entire training set is the classifier. The basic idea is that the similar tuples most likely belongs to the same class (a continuity assumption). Based on some pre-selected distance metric (some commonly used distance metrics are discussed in introduction), it finds the *k* most similar or nearest training samples of the sample to be classified and assign the plurality class of those *k* samples to the new sample [4, 7]. The value for *k* is pre-selected. Using relatively larger *k* may include some not so similar pixels and on the other hand, using very smaller *k* may exclude some potential candidate pixels. In both cases the classification accuracy will decrease. The optimal value of k depends on the size and nature of the data. The typical value for k is 3, 5 or 7. The steps of the classification process are:

1) Determine a suitable distance metric.
2) Find the k nearest neighbors using the selected distance metric.
3) Find the plurality class of the k-nearest neighbors (voting on the class labels of the NNs).
4) Assign that class to the sample to be classified.

We provided two different algorithms using P-trees, based two different distance metrics *max* (Minkowski distance with q = ∞) and our newly defined *HOB* distance. Instead of examining individual pixels to find the nearest neighbors, we start our initial *neighborhood* (neighborhood is a set of neighbors of the target pixel within a specified distance based on some distance metric, not the spatial neighbors, neighbors with respect to values) with the target sample and then successively expand the neighborhood area until there are k pixels in the neighborhood set. The expansion is done in such a way that the neighborhood always contains the closest or most similar pixels of the target sample. The different expansion mechanisms implement different distance functions. In the next section (section 3.1) we described the distance metrics and expansion mechanisms.

Of course, there may be more boundary neighbors equidistant from the sample than are necessary to complete the k nearest neighbor set, in which case, one can either use the larger set or arbitrarily ignore some of them. To find the exact k nearest neighbors one has to arbitrarily ignore some of them.

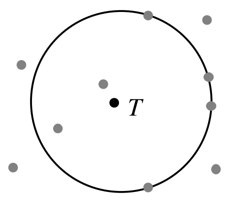


Fig. 3. *T*, the pixel in the center is the target pixels. With k = 3, to find the third nearest neighbor, we have four pixels (on the boundary line of the neighborhood) which are equidistant from the target.

Instead we propose a new approach of building nearest neighbor (NN) set, where we take the closure of the k-NN set, that is, we include all of the boundary neighbors and we call it the *closed-KNN* set. Obviously closed-KNN is a superset of KNN set. In the above example, with k = 3, KNN includes the two points inside the circle and any one point on the boundary. The closed-KNN includes the two points in side the circle and all of the four boundary points. The inductive definition of the closed-KNN set is given below.

Definition 1. a) If $x \in$ KNN, then $x \in$ closed-KNN

b) If $x \in$ closed-KNN and $d(T,y) \leq d(T,x)$, then $y \in$ closed-KNN where, $d(T,x)$ is the distance of x from target T.

c) Closed-KNN does not contain any pixel, which cannot be produced by steps a and b.

Our experimental results show closed-KNN yields higher classification accuracy than KNN does. The reason is if for some target there are many pixels on the boundary, they have more influence on the target pixel. While all of them are in the nearest neighborhood area, inclusion of one or two of them does not provide the necessary weight in the voting mechanism. One may argue that then why don't we use a higher k? For example using k = 5 instead of k = 3. The answer is if there are too few points (for example only one or two points) on the boundary to make k neighbors in the neighborhood, we have to expand neighborhood and include some not so similar points which will decrease the classification accuracy. We construct closed-

KNN only by including those pixels, which are in as same distance as some other pixels in the neighborhood without further expanding the neighborhood. To perform our experiments, we find the optimal *k* (by trial and error method) for that particular dataset and then using the optimal k, we performed both KNN and closed-KNN and found higher accuracy for P-tree-based closed-KNN method. The experimental results are given in section 4. In our P-tree implementation, no extra computation is required to find the closed-KNN. Our expansion mechanism of nearest neighborhood automatically includes the points on the boundary of the neighborhood.

Also, there may be more than one class in plurality (if there is a tie in voting), in which case one can arbitrarily chose one of the plurality classes. Without storing the raw data we create the basic P-trees and store them for future classification purpose. Avoiding the examination of individual data points and being ready for data mining these P-trees not only saves classification time but also saves storage space, since data is stored in compressed form. This compression technique also increases the speed of ANDing and other operations on P-trees, since operations can be performed on the pure0 and pure1 quadrants without reference to individual bits, since all of the bits in those quadrants are the same.

3.1 Expansion of Neighborhood and Distance or Similarity Metrics

We begin searching for nearest neighbors by finding the exact matches. If the number of exact matches is less than *k*, we expand the neighborhood. The expansion of the neighborhood in each dimension are done simultaneously, and continued until the number pixels in the neighborhood is greater than or equal to *k*. We develop the following two different mechanisms, corresponding to max distance and our newly defined HOB distance, for expanding the neighborhood. They have trade offs between execution time and classification accuracy.

Higher Order Bit Similarity (HOBS): We propose a new similarity metric where we consider similarity in the most significant consecutive bit positions starting from the left most bit, the highest order bit. Consider the following two values, x_1 and y_1, represented in binary. The 1st bit is the most significant bit and 8th bit is the least significant bit.

Bit position:	1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
x_1:	0	1	1	0	1	0	0	1	x_1:	0	1	1	0	1	0	0	1
y_1:	0	1	1	1	1	1	0	1	y_2:	0	1	1	0	0	1	0	0

These two values are similar in the three most significant bit positions, 1st, 2nd and 3rd bits (011). After they differ (4th bit), we don't consider anymore lower order bit positions though x_1 and y_1 have identical bits in the 5th, 7th and 8th positions. Since we are looking for closeness in values, after differing in some higher order bit positions, similarity in some lower order bit is meaningless with respect to our purpose. Similarly, x_1 and y_2 are identical in the 4 most significant bits (0110). Therefore, according to our definition, x_1 is closer or similar to y_2 than to y_1.

Definition 2. The similarity between two integers *A* and *B* is defined by

$$\text{HOBS}(A, B) = \max\{s \mid 0 \le i \le s \Rightarrow a_i = b_i\}$$

in other words, HOBS(A, B) = s, *where for all* $i \leq s$ *and* $0 \leq i$, $a_i = b_i$ *and* $a_{s+1} \neq b_{s+1}$. a_i and b_i are the ith bits of A and B respectively.

Definition 3. The distance between the values A and B is defined by

$$d_v(A, B) = m - \mathrm{HOBS}(A, B)$$

where m is the number of bits in binary representations of the values. All values must be represented using the same number of bits.

Definition 4. The distance between two pixels X and Y is defined by

$$d_p(X,Y) = \max_{i=1}^{n-1}\{d_v(x_i,y_i)\} = \max_{i=1}^{n-1}\{m - \mathrm{HOBS}(x_i,y_i)\}$$

where n is the total number of bands; one of them (the last band) is the class attribute that we don't use for measuring similarity.

To find the closed –KNN set, first we look for the pixels, which are identical to the target pixel in all 8 bits of all bands i.e. the pixels, X, having distance from the target T, $d_p(X,T) = 0$. If, for instance, x_1=105 ($01101001_b = 105_d$) is the target pixel, the initial neighborhood is [105, 105] ([01101001, 01101001]). If the number of matches is less than k, we look for the pixels, which are identical in the 7 most significant bits, not caring about the 8th bit, i.e. pixels having $d_p(X,T) \leq 1$. Therefore our expanded neighborhood is [104,105] ([01101000, 01101001] or [0110100-, 0110100-] - don't care about the 8th bit). Removing one more bit from the right, the neighborhood is [104, 107] ([011010--, 011010--] - don't care about the 7th or the 8th bit). Continuing to remove bits from the right we get intervals, [104, 111], then [96, 111] and so on.

Computationally this method is very cheap. However, the expansion does not occur evenly on both sides of the target value (note: the center of the neighborhood [104, 111] is (104 + 111) /2 = 107.5 but the target value is 105). Another observation is that the size of the neighborhood is expanded by powers of 2. These uneven and jump expansions include some not so similar pixels in the neighborhood keeping the classification accuracy lower. But P-tree-based closed-KNN method using this HOBS metric still outperforms KNN methods using any distance metric as well as becomes the fastest among all of these methods.

Perfect Centering: In this method we expand the neighborhood by 1 on both the left and right side of the range keeping the target value always precisely in the center of the neighborhood range. We begin with finding the exact matches as we did in HOBS method. The initial neighborhood is [a, a], where a is the target band value. If the number of matches is less than k we expand it to [a-1, a+1], next expansion to [a-2, a+2], then to [a-3, a+3] and so on.

Perfect centering expands neighborhood based on max distance metric or L_∞ metric (discussed in introduction). In the initial neighborhood $d_\infty(X,T)$ is 0. In the first expanded neighborhood [a-1, a+1], $d_\infty(X,T) \leq 1$. In each expansion $d_\infty(X,T)$ increases by 1. As distance is the direct difference of the values, increasing distance by one also increases the difference of values by 1 evenly in both side of the range.

This method is computationally a little more costly because we need to find matches for each value in the neighborhood range and then accumulate those matches but it results better nearest neighbor sets and yields better classification accuracy. We compare these two techniques later in section 4.

3.2 Computing the Nearest Neighbors

For HOBS: $P_{i,j}$ is the basic P-tree for bit j of band i and $P'_{i,j}$ is the complement of $P_{i,j}$. Let, $b_{i,j} = j^{th}$ bit of the i^{th} band of the target pixel, and define

$$Pt_{i,j} = P_{i,j}, \qquad \text{if } b_{i,j} = 1,$$
$$= P'_{i,j}, \quad \text{otherwise.}$$

We can say that the root count of $Pt_{i,j}$ is the number of pixels in the training dataset having as same value as the j^{th} bit of the i^{th} band of the target pixel. Let,

$$Pv_{i,1\text{-}j} = Pt_{i,1} \;\&\; Pt_{i,2} \;\&\; Pt_{i,3} \;\&\; \ldots \;\&\; Pt_{i,j}, \text{ and}$$

$$Pd(j) = Pv_{1,1\text{-}j} \;\&\; Pv_{2,1\text{-}j} \;\&\; Pv_{3,1\text{-}j} \;\&\; \ldots \;\&\; Pv_{n\text{-}1,1\text{-}j}$$

where $\&$ is the P-tree AND operator and n is the number of bands. $Pv_{i,1\text{-}j}$ counts the pixels having as same bit values as the target pixel in the higher order j bits of i^{th} band. We calculate the initial neighborhood P-tree, $Pnn = Pd(8)$, the exact matching, considering 8-bit values. Then we calculate $Pnn = Pd(7)$, matching in 7 higher order bits; then Then $Pnn = Pd(6)$ and so on. We continue as long as root count of Pnn is less than k. Pnn represents closed-KNN set and the root count of Pnn is the number of the nearest pixels. A 1 bit in Pnn for a pixel means that pixel is in closed-KNN set. The algorithm for finding nearest neighbors is given in Fig. 4.

```
Input: $P_{i,j}$ for all bit $i$ and band $j$, the basic P-trees
and $b_{i,j}$ for all $i$ and $j$, the bits for the target pixels
Output: $Pnn$, the P-tree representing closed-KNN
// $n$ - # of bands, $m$ - # of bits in each band
FOR $i$ = 1 TO $n$-1 D
    FOR $j$ = 1 TO $m$ DO
        IF $b_{i,j} = 1$ $Pt_{ij} \leftarrow P_{i,j}$
        ELSE $Pt_{i,j} \leftarrow P'_{i,j}$
FOR $i$ = 1 TO $n$-1 DO
    $Pv_{i,1} \leftarrow Pt_{i,1}$
    FOR $j$ = 2 TO $m$ DO
        $Pv_{i,j} \leftarrow Pv_{i,j-1}$ & $Pt_{i,j}$
$s \leftarrow m$
REPEAT
    $Pnn \leftarrow Pv_{1,s}$
    FOR $r$ = 2 TO $n$-1 DO
        $Pnn \leftarrow Pnn$ & $Pv_{r,s}$
    $s \leftarrow s - 1$
UNTIL RootCount($Pnn$) $\geq k$
```

Fig. 4. Algorithm to find closed-KNN set based on HOB metric

For Perfect Centering: Let v_i is the value of the target pixels for band i. The *value P-tree*, $P_i(v_i)$, represents the pixels having value v_i in band i. The algorithm for computing the value P-trees is given in Fig. 5(b). For finding the initial nearest neighbors (the exact matches), we calculate

$$Pnn = P_1(v_1) \;\&\; P_2(v_2) \;\&\; P_3(v_3) \;\&\; \ldots \;\&\; P_{n\text{-}1}(v_{n\text{-}1})$$

that represents the pixels having the same values in each band as that of the target pixel. If the root count of $Pnn \leq k$, we expand neighborhood along each dimension. For each band i, we calculate *range P-tree* $Pr_i = P_i(v_i\text{-}1) \mid P_i(v_i) \mid P_i(v_i+1)$. '|' is the P-tree OR operator. Pr_i represents the pixels having any value in the range $[v_i\text{-}1, v_i+1]$ of band i. The *AND*ed result of these range P-trees, Pr_i, for all i, produce the expanded neighborhood. The algorithm is given in Fig. 5(a).

```
Input: $P_{i,j}$ for all $i$ and $j$, basic P-trees and
       $v_i$ for all $i$, band values of target pixel
Output: $Pnn$, closed-KNN P-tree
// n - # of bands, m- #of bits in each band
FOR $i = 1$ TO $n$-1 DO
    $Pr_i \leftarrow P_i(v_i)$
$Pnn \leftarrow Pr_1$
FOR $i = 2$ TO $n$-1 DO
    $Pnn \leftarrow Pnn$ & $Pr_i$  //initial neighborhood
$d \leftarrow 1$     // distance for the first expansion
WHILE RootCount($Pnn$) < $k$ DO
    FOR $i = 1$ to $n$-1 DO     // expansion
        $Pr_i \leftarrow Pr_i \mid P_i(v_i-d) \mid P_i(v_i+d)$
    $Pnn \leftarrow Pr_1$          // '|' - OR operator
    FOR $i = 2$ TO $n$-1 DO
        $Pnn \leftarrow Pnn$ AND $Pr_i$
    $d \leftarrow d + 1$
```

Fig. 5(a). Algorithm to find closed-KNN set based on Max metric (Perfect Centering).

```
Input: $P_{i,j}$ for all $j$, basic P-trees of all
the bits of band $i$ and the value $v_i$ for
band $i$.
Output: $P_i(v_i)$, the value p-tree for the
value $v_i$
// $m$ is the number of bits in each band
// $b_{i,j}$ is the $j^{th}$ bit of value $v_i$

FOR $j = 1$ TO $m$ DO
    IF $b_{i,j} = 1$ $Pt_{ij} \leftarrow P_{i,j}$
    ELSE $Pt_{i,j} \leftarrow P'_{i,j}$

$P_i(v) \leftarrow Pt_{i,1}$
FOR $j = 2$ TO $m$ DO
    $P_i(v) \leftarrow P_i(v)$ & $Pt_{i,j}$
```

5(b). Algorithm to compute value P-trees

3.3 Finding the plurality class among the nearest neighbors

For the classification purpose, we don't need to consider all bits in the class band. If the class band is 8 bits long, there are 256 possible classes. Instead, we partition the class band values into fewer, say 8, groups by truncating the 5 least significant bits. The 8 classes are 0, 1, 2, …, 7. Using the leftmost 3 bits we construct the value P-trees $P_n(0)$, $P_n(1)$, …, $P_n(7)$. The P-tree Pnn & $P_n(i)$ represents the pixels having a class value i and are in the closed-KNN set, Pnn. An i which yields the maximum root count of Pnn & $P_n(i)$ is the plurality class; that is

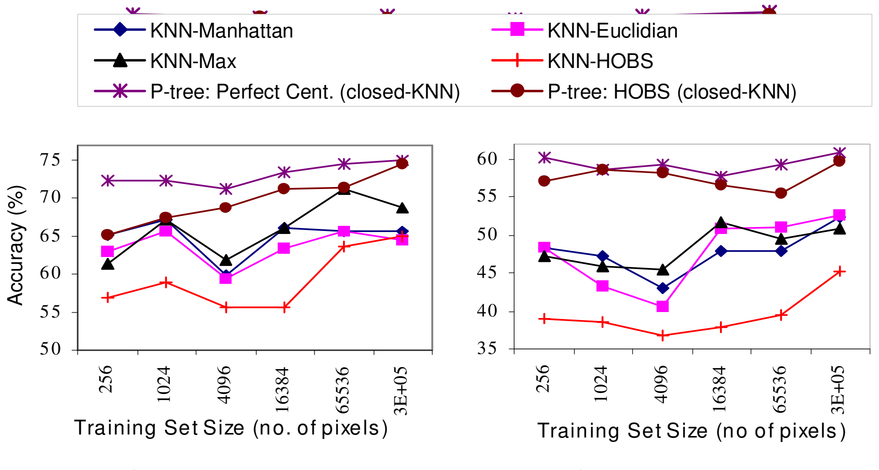
$$\text{predicted class} = \arg\max_i \{RootCount(Pnn \,\&\, P_n(i))\}.$$

4. Performance Analysis

We performed experiments on two sets of Arial photographs of the Best Management Plot (BMP) of Oakes Irrigation Test Area (OITA) near Oaks, North Dakota, United States. The latitude and longitude are 45°49'15"N and 97°42'18"W respectively. The two images "29NW083097.tiff" and "29NW082598.tiff" have been taken in 1997 and 1998 respectively. Each image contains 3 bands, red, green and blue reflectance values. Three other separate files contain synchronized soil moisture, nitrate and yield values. Soil moisture and nitrate are measured using shallow and deep well lysimeters. Yield values were collected by using a GPS yield monitor on the harvesting equipments. The datasets are available at http://datasurg.ndsu.edu/.

Yield is the class attribute. Each band is 8 bits long. So we have 8 basic P-trees for each band and 40 (for the other 5 bands except yield) in total. For the class band, we considered only the most significant 3 bits. Therefore we have 8 different class labels. We built 8 value P-trees from the yield values – one for each class label.

The original image size is 1320×1320. For experimental purpose we form 16×16, 32×32, 64×64, 128×128, 256×256 and 512×512 image by choosing pixels uniformly distributed in the original image. In each case, we formed one test set and one training set of equal size and tested KNN with Manhattan, Euclidian, Max and HOBS distance metrics and our two P-tree methods, Perfect Centering and HOBS. The accuracies of these different implementations are given in the Fig 6.



a) 1997 dataset **b)** 1998 dataset

Fig. 6. Accuracy of different implementations for the 1997 and 1998 datasets

We see that both of our P-tree based closed-KNN methods outperform the KNN methods for both of the datasets. The reasons are discussed in section 3. The perfect centering methods performs better than HOBS as expected. The HOBS metric is not suitable for a KNN approach since HOBS does not provide a neighborhood with the target pixel in the exact center. Increased accuracy of HOBS in P-tree implementation is the effect of closed-KNN. In a P-tree implementation, the ease of computability for closed-KNN using HOBS makes it a superior method. The P-tree based HOBS is the fastest method where as the KNN-HOBS is still the poorest (Fig. 8).

Another observation is that for 1997 data (Fig. 6), in KNN implementations, the max metric performs better than other three metrics. For the 1998 dataset, max is competitive with other three metrics. In many cases, as well as for image data, max metrics can be the best choice. In our P-tree implementations, we also get very high accuracy with the max distance (perfect centering method). We can understand this by examining the shape of the neighborhood for different metrics (Fig. 7).

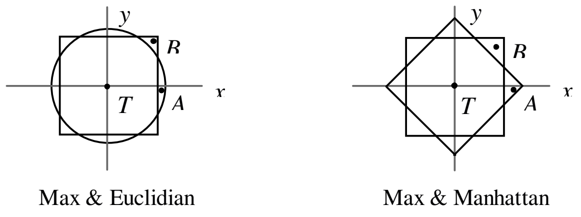


Fig. 7. Considering two dimensions the shape of the neighborhood for Euclidian distance is the circle, for max it is the square and for Manhattan it is the diamond. T is the target pixel.

Let, *A* be a point included in the circle, but not in the square and *B* be a point, which is included in the square but not in the circle. The point *A* is very similar to target *T* in the *x*-dimension but very dissimilar in the *y*-dimension. On the other hand, the point *B* is not so dissimilar in any dimension. Relying on high similarity only on one band while keeping high dissimilarity in the other bands may decrease the accuracy. Therefore in many cases, inclusion of *B* in the neighborhood instead of *A*, is a better choice. That is what we have found for our image data. For all of the methods classification accuracy increases with the size of the dataset since inclusion of more training data, the chance of getting better nearest neighbors increases.

On the average, the perfect centering method is five times faster than the KNN, and HOBS is 10 times faster (Fig. 8). P-tree implementations are more scalable. Both perfect centering and HOBS increases the classification time with data size at a lower rate than the other methods. As dataset size increases, there are more and larger pure-0 and pure-1 quadrants in the P-trees; that makes the P-tree operations faster.

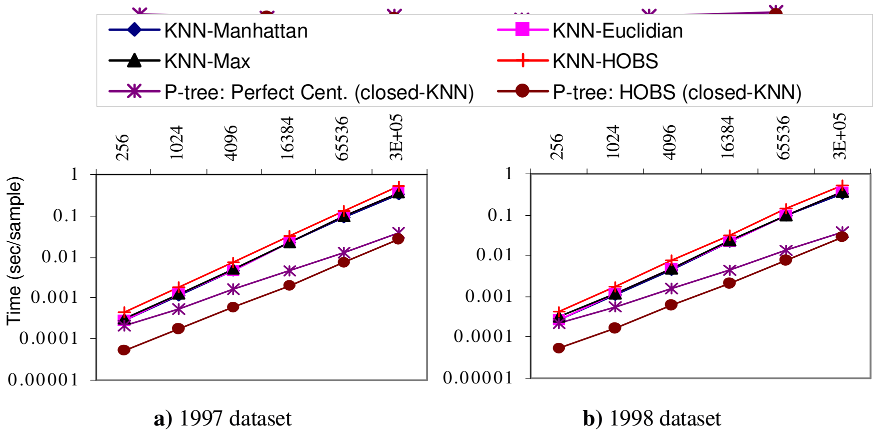


Fig. 8. Classification time per sample of the different implementations for the 1997 and 1998 datasets; both of the size and classification time are plotted in logarithmic scale.

5. Conclusion

In this paper we proposed a new approach to k-nearest neighbor classification for spatial data streams by using a new data structure called the P-tree, which is a lossless compressed and data-mining-ready representation of the original spatial data. Our new approach, called closed-KNN, finds the closure of the KNN set, we call the closed-KNN, instead of considering exactly *k* nearest neighbors. Closed-KNN includes all of the points on the boundary even if the size of the nearest neighbor set becomes larger than *k*. Instead of examining individual data points to find nearest neighbors, we rely on the expansion of the neighborhood. The P-tree structure facilitates efficient computation of the nearest neighbors. Our methods outperform the traditional implementations of KNN both in terms of accuracy and speed.

We proposed a new distance metric called Higher Order Bit (HOB) distance that provides an easy and efficient way of computing closed-KNN using P-trees while preserving the classification accuracy at a high level.

References

1. Domingos, P. and Hulten, G., “Mining high-speed data streams”, Proceedings of ACM SIGKDD 2000.
2. Domingos, P., & Hulten, G., “Catching Up with the Data: Research Issues in Mining Data Streams”, DMKD 2001.
3. T. Cover and P. Hart, “Nearest Neighbor pattern classification”, IEEE Trans. Information Theory, 13:21-27, 1967.
4. Dasarathy, B.V., “Nearest-Neighbor Classification Techniques”. IEEE Computer Society Press, Los Alomitos, CA, 1991.
5. Morin, R.L. and D.E.Raeside, “A Reappraisal of Distance-Weighted k-Nearest Neighbor Classification for Pattern Recognition with Missing Data”, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-11 (3), pp. 241-243, 1981.
6. William Perrizo, "Peano Count Tree Technology", Technical Report NDSU-CSOR-TR-01-1, 2001.
7. Jiawei Han, Micheline Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann, 2001.
8. M. James, “Classification Algorithms”, New York: John Wiley & Sons, 1985.
9. William Perrizo, Qin Ding, Qiang Ding and Amalendu Roy, “On Mining Satellite and Other Remotely Sensed Images”, DMKD 2001, pp. 33-40.
10. William Perrizo, Qin Ding, Qiang Ding and Amalendu Roy, “Deriving High Confidence Rules from Spatial Data using Peano Count Trees", Springer-Verlag, Lecturer Notes in Computer Science 2118, July 2001.
11. Qin Ding, Maleq Khan, Amalendu Roy and William Perrizo, “The P-tree Algebra”, proceedings of the ACM Symposium on Applied Computing (SAC'02), 2002.