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 has become important due to the fact that there are huge volumes of spatial data now available holding a wealth of valuable information. In this paper we consider the classification of spatial data streams, where the training dataset changes often. New training data arrive continuously and are added to the training set. For these types of data streams, 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. For that reason KNN is called a lazy classifier. KNN is extremely simple to implement and lends itself to a wide variety of variations. The traditional k-nearest neighbor classifier finds the k nearest neighbors based on some distance metric by finding the distance of the target data point from the training dataset, then finding the class from those nearest neighbors by some voting mechanism. There is a problem associated with KNN classifiers. They increase the classification time significantly relative to other non-lazy methods. To overcome this problem, in this paper we propose a new method of KNN classification for spatial data streams using a new, rich, data-mining-ready structure, the Peano-count-tree or P-tree. In our method, we merely perform some logical AND/OR operations on P-trees to find the nearest neighbor set of a new sample and assign the class label. We have fast and efficient algorithms for AND/OR operations on P-trees, which reduce the classification time significantly, compared with traditional KNN classifiers. 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.

Keywords: Data Mining, K-Nearest Neighbor Classification, P-tree, Spatial Data, Data Streams.

[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.

1. Introduction

Classification is the process of finding a set of models or functions that describes and distinguishes data classes or concepts for the purpose of predicting the class of objects whose class labels are unknown [9]. The derived model is based on the analysis of a set of training data whose class labels are known. Consider each training sample has n attributes: A_1, A_2, A_3, …, A_{n-1}, C, where C is the class attribute which defines the class or category of the sample. The model associates the class attribute, C, with the other attributes. Now consider a new tuple or data sample whose values for the attributes A_1, A_2, A_3, …, A_{n-1} are known, while for the class attribute is unknown. The model predicts the class label of the new tuple using the values of the attributes A_1, A_2, A_3, …, A_{n-1}.

There are various techniques for classification such as Decision Tree Induction, Bayesian Classification, and Neural Networks [9, 11]. Unlike other common classification methods, a k-nearest neighbor classification (**KNN classification**) 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 closeness metric [3, 7, 10]. A common similarity function is based on the Euclidian distance between two data tuples [3]. For two tuples, $X = \langle x_1, x_2, x_3, \ldots, x_{n-1}\rangle$ and $Y = \langle y_1, y_2, y_3, \ldots, y_{n-1}\rangle$ (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 $d_q(X,Y) = \sqrt[q]{\sum_{i=1}^{n-1} w_i |x_i - y_i|^q}$. 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 $d_\infty(X,Y) = \max_{i=1}^{n-1} |x_i - y_i|$. After finding the k nearest tuples based on the selected distance metric, the plurality class label of those k tuples can be assigned to the new sample as its class. If there is more than one class label in plurality, one of them can be chosen arbitrarily.

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

Nearly every other classification model trains and tests a residual "classifier" first and then uses it on new samples. KNN does not build a residual classifier, but instead, searches again for the k-nearest neighbor set for each new sample. This approach is simple and can be very accurate. It can also be slow (the search may take a long time). KNN is a good choice when simplicity and accuracy are the predominant issues. KNN can be superior when a residual, trained and tested classifier 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 [8, 12, 13]. 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 (though the dimension could just as well be 1 or 3 or higher). 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 other value bands (e.g., crop yield quantities, crop quality measures, soil attributes and radar reflectance intensities). One band such as yield band can be the class attribute. The location coordinates in raster order constitute the key

attribute of the spatial dataset and the other bands are the non-key attributes. 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 **HOBS** distance metric. **HOBS** 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. Section 5 is the conclusion.

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. This order is assumed and therefore is not explicitly indicated as a key attribute in each band (bands have just one column). In this paper, 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** [8, 12, 13]. A Landsat Thematic Mapper satellite image, for example, is in BSQ format with 7 bands, B1,…,B7, (Landsat-7 has 8) and ~40,000,000 8-bit data values. A typical TIFF image aerial digital photograph is in what is called Band Interleaved by Bit (BIP) format, in which there is one file containing ~24,000,000 bits ordered by it-position, then band and then raster-ordered-pixel-location. A simple transform can be used to convert TIFF images to BSQ and then to bSQ format.

We organize each bSQ bit file, Bij (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 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), which may or may not be at the leaf level. For example, the P-tree for a 8-row-8-column bit-band is shown in Figure 1.

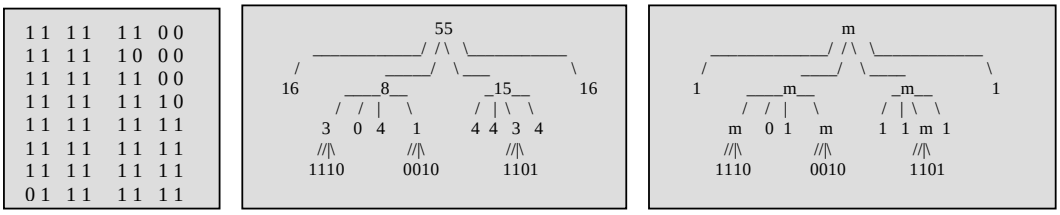


Figure 1. 8-by-8 image and its P-tree (P-tree and PM-tree)

In this example, 55 is the count of 1's in the entire image, the numbers 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. This pattern is continued recursively. Recursive raster ordering is called the Peano or Z-ordering in the literature – therefore, the name Peano Count trees. The process will definitely terminate at the "leaf" level where each quadrant is a 1-row-1-column quadrant. If we were to expand all sub-trees, including those for quadrants that are pure 1-bits, then the leaf sequence is just the Peano space-filling curve for the original raster image.

For each band (assuming 8-bit data values), we get 8 basic P-trees, one for each bit positions. For band, Bi, we will label the basic P-trees, $P_{i,1}$, $P_{i,2}$, …, $P_{i,8}$, thus, $P_{i,j}$ is a lossless representation of the j^{th} bits of the values from the i^{th} band. However, P_{ij} provides much more information and are structured to facilitate many important data mining processes.

For efficient implementation, we use variation of basic P-trees, called PM-tree (Pure Mask 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 Figure 1.

P-tree algebra contains operators, AND, OR, NOT and XOR, which are the pixel-by-pixel logical operations on P-trees. The NOT operation is a straightforward translation of each count to its quadrant-complement (e.g., a 5 count for a quadrant of 16 pixels has complement of 11). The AND operations is described in full detail below. The OR is identical to the AND except that the role of the 1-bits and the 0-bits are reversed.

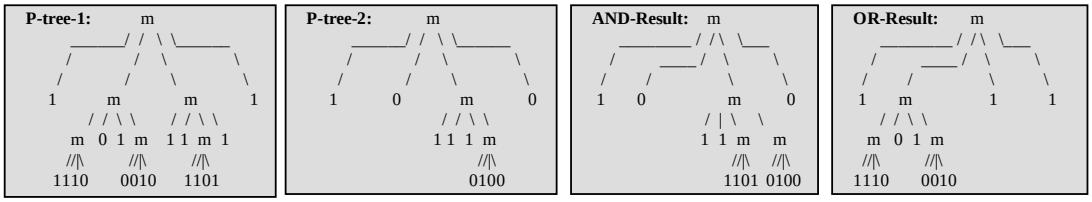


Figure 2. P-tree Algebra

The basic P-trees can be combined using simple logical operations to produce P-trees for the original values (at any level of precision, 1-bit precision, 2-bit precision, etc.). We let $P_{b,v}$ denote the Peano Count Tree for band, b, and value, v, where v can be expressed in 1-bit, 2-bit,.., or 8-bit precision. Using the full 8-bit precision (all 8 –bits) for values, $P_{b,11010011}$ can be constructed from the basic P-trees as:

$P_{b,11010011} = P_{b1}$ AND P_{b2} AND P_{b3}' AND P_{b4} AND P_{b5}' AND P_{b6}' AND P_{b7} AND P_{b8}.

Where ' indicates NOT operation. The AND operation is simply the pixel-wise AND of the bits.

Similarly, the data in the relational format can be represented as P-trees also. For any combination of values, $(v_1,v_2,\ldots,v_n)$, where v_i is from band-i, the quadrant-wise count of occurrences of this combination of values is given by:

$$P_{(v1,v2,\ldots,vn)} = P_{1,v1} \text{ AND } P_{2,v2} \text{ AND } \ldots \text{ AND } P_{n,vn}$$

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. The value for k is pre-selected. Using relatively larger k may include some pixels 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 $q = \infty$) and our newly defined **HOBS**. 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.

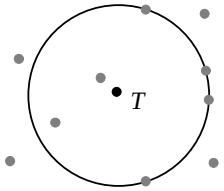


Figure 3: T, the pixel in the center is the target pixels. With k = 3, to find the third nearest neighbor, we have four pixels (on the boundary line of the neighborhood) which are equidistant from the target.

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 step 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. Unlike the traditional k-nearest neighbor classifier our classification method doesn't store and use raw training data. Instead we use the data-mining-ready P-tree structure, which can be built very quickly from the training data. 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 tremendously, since operations can be performed on the pure0 and pure1 quadrants without reference to individual bits, since all of the bits in those quadrant are the same.

3.1 Expansion of Neighborhood and Distance or Similarity Metrics

Similarity and distance can be measured by each other; more distance less similar and less distance more similar. Our similarity metric is the closeness in numerical values for corresponding bands. We begin searching for nearest neighbors by finding the exact matches i.e. the pixels having as same band-values as that of the target pixel. If the number of exact matches is less than k, we expand the neighborhood. For example, for a particular band, if the target pixel has the value *a*, we expand the neighborhood to the range [*a-b, a+c*], where *b* and *c* are positive integers and find the pixels having the band value in the range [*a-b, a+c*]. We expand the neighbor in each band (or dimension) simultaneously. We continue expanding the neighborhood 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 (Minqowski distance with q = ∞ or L_∞) and our newly defined HOBS distance, for expanding the neighborhood. The two given mechanisms have trade off between execution time and classification accuracy.

A. Higher Order Bit Similarity (HOBS): We propose a new similarity metric where we use similarity in the most significant bit positions between two band values. We consider only the most significant consecutive bit positions starting from the left most bit, which is 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
x_1:	0	1	1	0	1	0	0	1
y_1:	0	1	1	1	1	1	0	1

	1	2	3	4	5	6	7	8
x_1:	0	1	1	0	1	0	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 values A and B is defined by*

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

Or in other words, HOBS(*A, B*) = *s, where for all* $i \le s$, $a_i = b_i$ *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*

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

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*

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

where n is the total number of bands where 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 $d_p(X,T) \le 1$. 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 (since the counts are just the root counts of individual P-trees, all of which can be constructed in one operation). 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.

To improve accuracy further we propose another method called perfect centering to avoid the uneven and jump expansion. Although, in terms of accuracy, perfect centering outperforms HOBS, in terms of computational speed it is slower than HOBS.

B. 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, Minkowski distance (discussed in introduction) metric setting q = ∞.

$$d_\infty(X,Y) = \max_{i=1}^{n-1} |x_i - y_i|$$

In the initial neighborhood $d_\infty(X,T)$ is 0, the distance of any pixel *X* in the neighborhood from the target *T*. In the first expanded neighborhood [*a*-1, *a*+1], $d_\infty(X,T) \le 1$. 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 without any jumping.

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: We have the basic P-trees of all bits of all bands constructed from the training dataset and the new sample to be classified. Suppose, including the class band, there are n bands or attributes in the training dataset and each attribute is m bits long. In the target sample we have *n*-1 bands, but the class band value is unknown. Our goal is to predict the class band value for the target sample.

$P_{i,j}$ is the P-tree for bit *j* of band *i*. This P-tree stores all the j^{th} bits of the i^{th} band of all the training pixels. The root count of a P-tree is the total counts of one bits stored in it. Therefore, the root count of $P_{i,j}$ is the number of pixels in the training dataset having a 1 value in the j^{th} bit of the i^{th} band. $P'_{i,j}$ is the complement P-tree of $P_{i,j}$. $P'_{i,j}$ stores 1 for the pixels having a 0 value in the j^{th} bit of the i^{th} band and stores 0 for the pixels having a 1 value in the j^{th} bit of the i^{th} band. Therefore, the root count of $P'_{i,j}$ is the number of pixels in the training dataset having 0 value in the j^{th} bit of the i^{th} band.

Now let, $b_{i,j} = j^{th}$ bit of the i^{th} band of the target pixel.

Define $Pt_{i,j} = P_{i,j}$, *if* $b_{i,j} = 1$

$= P'_{i,j}$, 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, $Pv_{i,1\text{-}j} = Pt_{i,1}$ & $Pt_{i,2}$ & $Pt_{i,3}$ & … & $Pt_{i,j}$, here & is the P-tree AND operator.

$Pv_{i,1\text{-}j}$ counts the pixels having as same bit values as the target pixel in the higher order *j* bits of i^{th} band.

Using higher order bit similarity, first we find the P-tree $Pnn = Pv_{1,1\text{-}8}$ & $Pv_{2,1\text{-}8}$ & $Pv_{3,1\text{-}8}$ & … & $Pv_{n\text{-}1,1\text{-}8}$, where *n*-1 is the number of bands excluding the class band. *Pnn* represents the pixels that exactly match the target pixel. If the root count of *Pnn* is less than *k* we look for higher order 7 bits matching i.e. we calculate $Pnn = Pv_{1,1\text{-}7}$ & $Pv_{2,1\text{-}7}$ & $Pv_{3,1\text{-}7}$ & … & $Pv_{n\text{-}1,1\text{-}7}$. Then we look for higher order 6 bits matching and so on. We continue as long as root count of Pnn is less than *k*. *Pnn* represents closed-KNN set i.e. the training pixels having the as same bits in corresponding higher order bits as that in target pixel and the root count of *Pnn* is the number of such pixels, the nearest

pixels. A 1 bit in Pnn for a pixel means that pixel is in closed-KNN set and a 0 bit means the pixel is not in the closed-KNN set. The algorithm for finding nearest neighbors is given in figure 4.

Algorithm: **Finding the P-tree representing closed-KNN set using HOBS**

Input: $P_{i,j}$ for all i and j, basic P-trees of all the bits of all bands of the training dataset and $b_{i,j}$ for all i and j, the bits for the target pixels

Output: Pnn, the P-tree representing the nearest neighbors of the target pixel

// n is the number of bands where n^{th} band is the class band
// m is the number of bits in each band

FOR i = 1 *TO* n-1 *DO*
  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*
    $Pv_{i,j} \leftarrow Pv_{i,j-1}$ & $Pt_{i,j}$
$s \leftarrow m$ // first we check matching in all m bits
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$(Pnn) \geq k$

Figure 4: *Algorithm to find closed-KNN set based on HOBS metric.*

For Perfect Centering: Let v_i is the value of the target pixels for band i. $P_i(v_i)$ is the value P-tree for the value v_i in band i. $P_i(v_i)$ represents the pixels having value v_i in band i. For finding the initial nearest neighbors (the exact matches) using perfect centering we find $P_i(v_i)$ for all i. The ANDed result of these value P-trees i.e. $Pnn = P_1(v_1)\ \&\ P_2(v_2)\ \&\ P_3(v_3)\ \&\ \ldots\ \&\ P_{n-1}(v_{n-1})$ represents the pixels having the same values in each band as that of the target pixel. A value P-tree, $P_i(v_i)$, can be computed by finding the P-tree representing the pixels having the same bits in band i as the bits in value v_i. That is, if $Pt_{i,j} = P_{i,j}$, when $b_{i,j} = 1$ and $Pt_{i,j} = P'_{i,j}$, when $b_{i,j} = 0$ ($b_{i,j}$ is the j^{th} bit of value v_i),then $P_i(v_i) = Pt_{i,1}\ \&\ Pt_{i,2}\ \&\ Pt_{i,3}\ \&\ \ldots\ \&\ Pt_{i,m}$, m is the number of bits in a band. The algorithm for computing value P-trees is given in figure 5(b).

Algorithm: **Finding the P-tree representing closed-KNN set using max distance metric (perfect centering)**

Input: $P_{i,j}$ for all i and j, basic P-trees of all the bits of all bands of the training dataset and v_i for all i, the band values for the target pixel
Output: Pnn, the P-tree representing the closed-KNN set
// n is the number of bands where n^{th} band is the class band
// m is the number 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*
  $Pnn \leftarrow Pnn\ \&\ Pr_i$ // the initial neighborhood for exact matching
d ← 1 // distance for the first expansion
WHILE RootCount$(Pnn) < k$ *DO*
  FOR i = 1 *to* n-1 *DO*
    $Pr_i \leftarrow Pr_i \mid P_i(v_i\text{-}d) \mid P_i(v_i\text{+}d)$ // neighborhood expansion
  $Pnn \leftarrow Pr_1$ // '|' is the P-tree OR operator
  FOR i = 2 *TO* n-1 *DO*
    $Pnn \leftarrow Pnn$ AND Pr_i // updating closed-KNN set
  d ← d + 1

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

Algorithm: **Finding value P-tree**

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

If the number of exact matching i.e. root count of *Pnn* is less than 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 a value either v_i-1 or v_i or v_i+1 i.e. any value in the range [v_i-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 pixels having band values in the ranges of the corresponding bands. We continue this expansion process until root count of *Pnn* is greater than or equal to *k*. The algorithm is given in figure 5(a).

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 of considering 256 classes we partition the class band values into fewer groups by considering fewer significant bits. For example if we want to partition into 8 groups we can do it by truncating the 5 least significant bits and keeping the most significant 3 bits. The 8 classes are 0, 1, 2, 3, 4, 5, 6 and 7. Using these 3 bits we construct the value P-trees $P_n(0)$, $P_n(1)$, $P_n(2)$, $P_n(3)$, $P_n(4)$, $P_n(5)$, $P_n(6)$, and $P_n(7)$.

An 1 value in the nearest neighbor P-tree, *Pnn*, indicates that the corresponding pixel is in the nearest neighbor set. An 1 value in the value P-tree, $P_n(i)$, indicates that the corresponding pixel has the class value *i*. Therefore $Pnn \ \& \ P_n(i)$ represents the pixels having a class value *i* and are in the nearest neighbor set. An *i* which yields the maximum root count of Pnn & P_n(i) is the plurality class. The algorithm is given in the figure 6.

Algorithm: **Finding the plurality class**

Input: $P_n(i)$, the value P-trees for all class *i* and the closed-KNN P-tree, *Pnn*
Output: Pnn, the P-tree representing the nearest neighbors of the target pixel
// *c* is the number of different classes

```
class ← 0
P ← Pnn & P_n(0)
rc ← RootCount(P)
FOR i = 1 TO c - 1 DO
    P ← Pnn & P_n(i)
    IF rc < RootCount(P)
        rc ← RootCount(P)
        class ← i
```

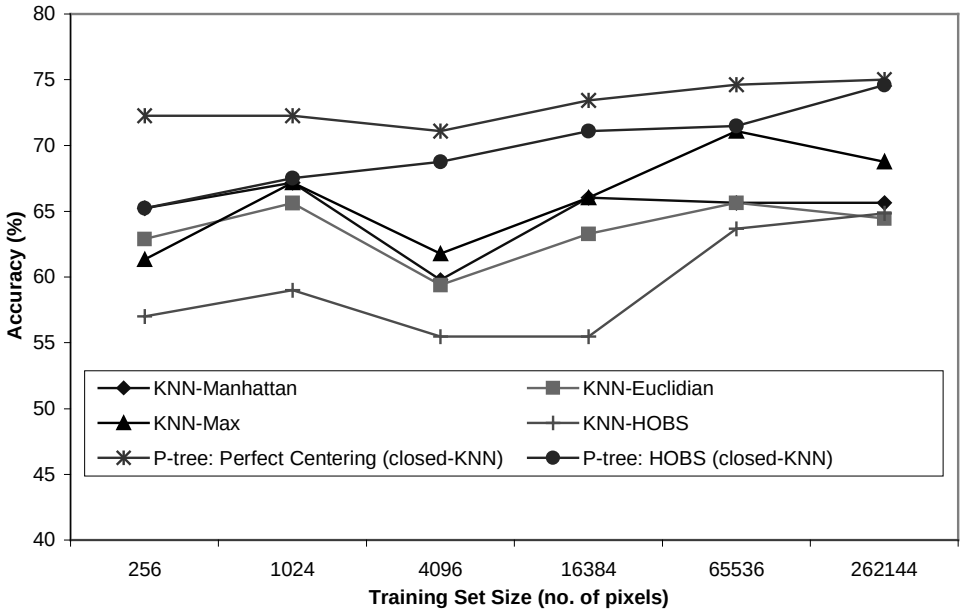
Figure 6: *Algorithm to find the plurality class*

4. Performance Analysis

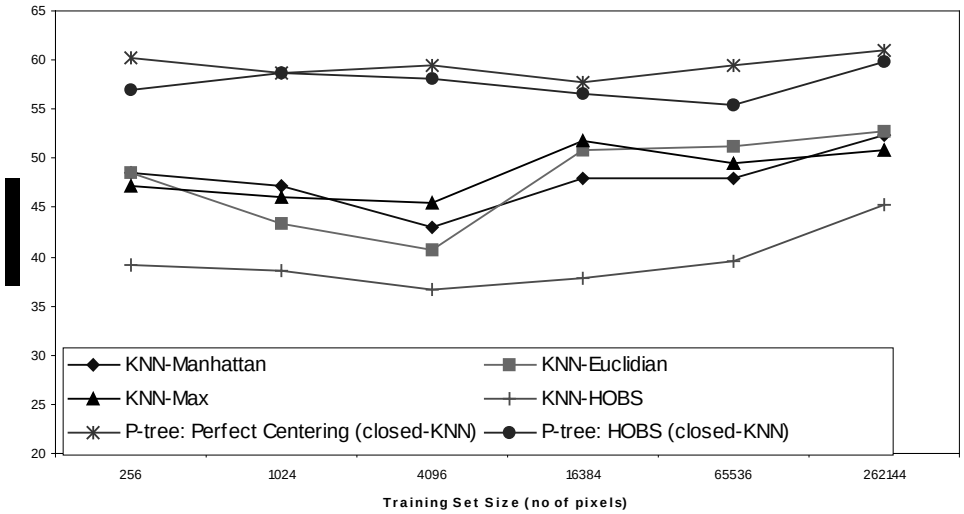
We performed experiments on two sets of Arial photographs of the Best Management Plot (BMP) of Oakes Irrigation Test Area (OITA) near Oaks, North Dakota, United States. The latitude and longitude are 45°49'15"N and 97°42'18"W respectively. The two images "29NW083097.tiff" and "29NW082598.tiff" have been 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/.

Among those 6 bands we consider the yield as 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, yield, we considered only the most significant 3 bits. Therefore we have 8 different class labels for the pixels. 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 that are uniformly distributed in the original image. In each case, we form one test set and one training set of equal size. For each of the above sizes we 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 figure 7 for both of the datasets.



(a) *29NW083097.tiff and associated other files (*1997 *dataset)*



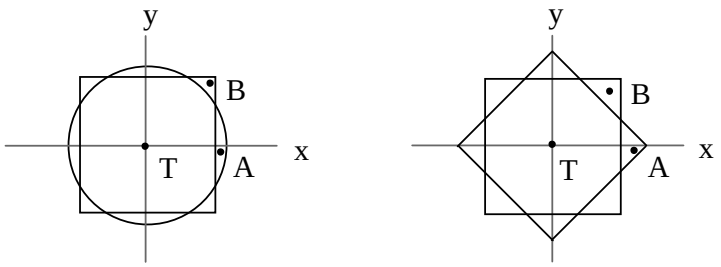
(b) *29NW082598.tiff and associated other files (*1998 *dataset)*

Figure* 7*: *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. We discussed in section 3.1, why the perfect centering methods performs better than HOBS. We also implemented the HOBS metric for KNN standard. From the result we can see that the accuracy is very poor. 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, which is explained in section 3. 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 (figure 9 in page 11).

Another observation is that for 1997 data (Figure 7(a)), in KNN implementations, the max metric performs much 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 (figure 8).



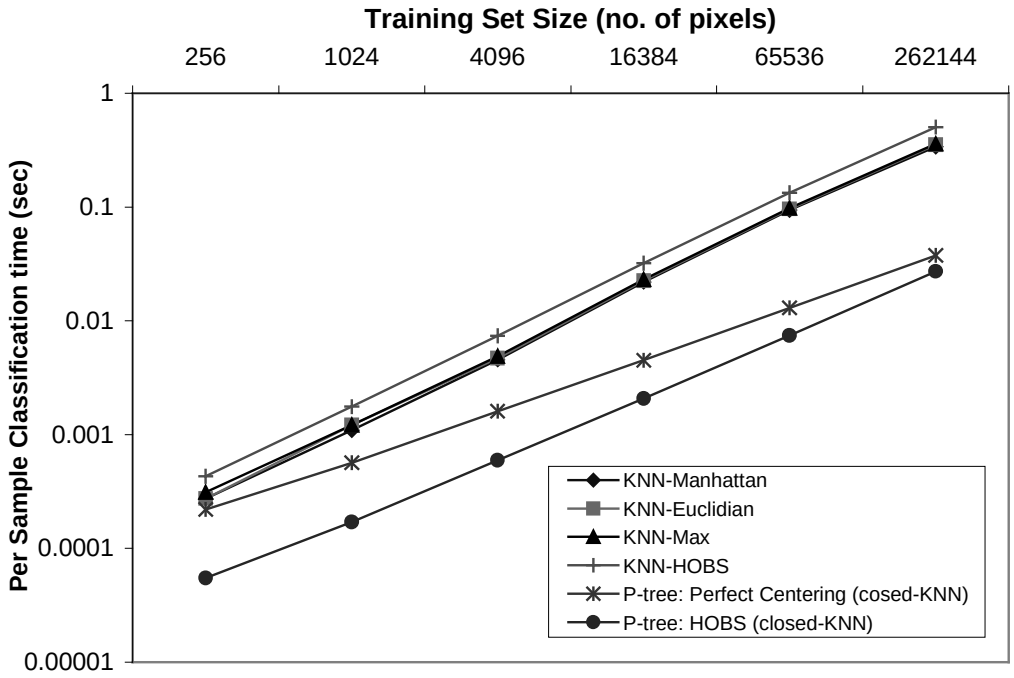
(a) *Max & Euclidian* (b) *Max & Manhattan*

***Figure* 8:** *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.*

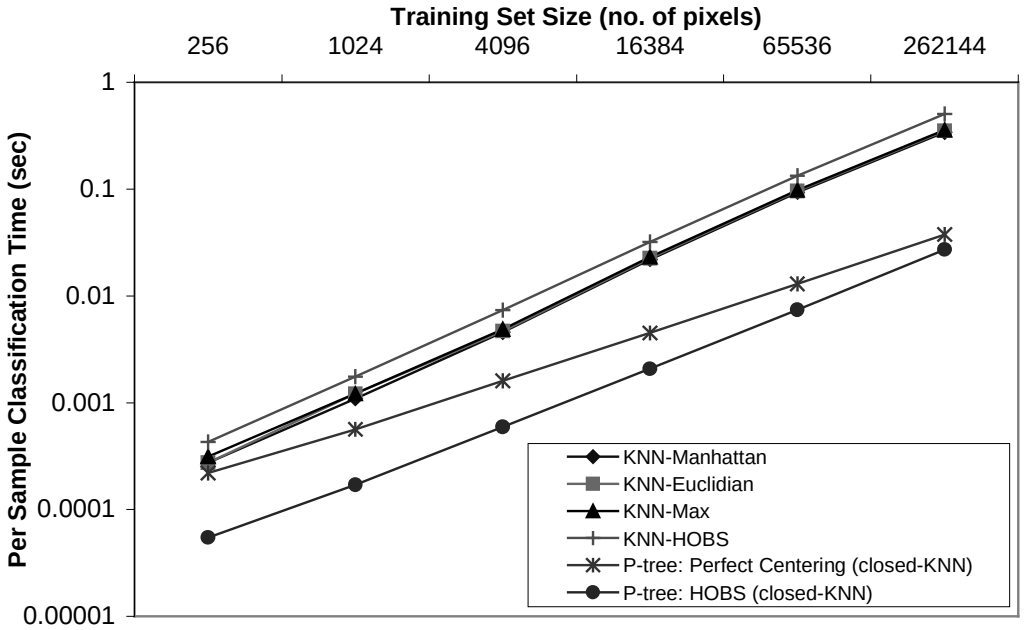
Consider the two points *A* and *B* (figure 8). Let, A be a point included in the circle, but not included in the square. Let B be a point, which is included in the square but not the circle. The point A is very similar to target T in the x-dimension but very dissimilar in the y-dimension. On the other 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 band 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.

We also observe that for almost all of the methods classification accuracy increases with the size of the training dataset. The reason is that with the inclusion of more training pixels, the chance of getting better nearest neighbors increases.

Figure 9 (in page 11) shows, on the average, the perfect centering method is five times faster than the KNN and HOBS is 10 times faster (the graphs are plotted in logarithmic scale). Classification times for all of the non-P-tree methods are almost equal. P-tree implementations are more scalable than other methods. Both perfect centering and HOBS increases the classification time with data size at a lower rate than the other methods. For the smaller dataset, the perfect centering method is about 2 times faster than the others and for the larger dataset, it is 10 times faster. This is also true for the HOBS method. The reason is that as dataset size increases, there are more and larger pure-0 and pure-1 quadrants in the P-trees, which increases the efficiency of the ANDing operations.



(a) *29NW083097.tiff and associated other files (*1997 *dataset)*



(b) *29NW082598.tiff and associated other files (*1998 *dataset)*

***Figure* 9:** *Classification time per sample for 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 closed-KNN, instead of considering exactly k nearest neighbor. 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 Similarity (HOBS) 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] Dudani, S. (1975). “The distance-weighted k-nearest neighbor rule”. IEEE Transactions on Systems, Man, and Cybernetics, 6:325--327.

[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] J. E. Macleod, A. Luk, and D. M. Titterington. “A re-examination of the distanceweighted k-nearest neighbor classification rule”. IEEE Trans. Syst. Man Cybern., SMC-17(4):689--696, 1987.

[7] Dasarathy, B.V. (1991). “Nearest-Neighbor Classification Techniques”. IEEE Computer Society Press, Los Alomitos, CA.

[8] William Perrizo, "Peano Count Tree Technology", Technical Report NDSU-CSOR-TR-01-1, 2001.

[9] Jiawei Han, Micheline Kamber, “Data Mining: Concepts and Techniques”, Morgan Kaufmann, 2001.

[10] Globig, C., & Wess, S. “Symbolic Learning and Nearest-Neighbor Classification”, 1994.

[11] M. James, “Classification Algorithms”, New York: John Wiley & Sons, 1985.

[12] William Perrizo, Qin Ding, Qiang Ding and Amalendu Roy, “On Mining Satellite and Other Remotely Sensed Images”, DMKD 2001, pp. 33-40.

[13] William Perrizo, Qin Ding, Qiang Ding and Amalendu Roy, “Deriving High Confidence Rules from Spatial Data using Peano Count Trees", Proceedings of the 2nd International Conference on Web-Age Information Management, 2001.