# An efficient weighted nearest neighbour classifier using vertical data representation

William Perrizo

Department of Computer Science,
North Dakota State University,
P.O. Box 5164, Fargo, ND 58105-5164, USA
E-mail: william.perrizo@ndsu.edu

Qin Ding*

Department of Computer Science,
Pennsylvania State University at Harrisburg,
Middletown, PA 17057, USA
Fax: +1-717-948-6352 E-mail: qding@psu.edu
*Corresponding author

Maleq Khan

Department of Computer Science,
Purdue University, 250 N. University St.,
West Lafayette, IN 47907, USA
E-mail: mmkhan@cs.purdue.edu

Anne Denton

Department of Computer Science and Operations Research,
North Dakota State University,
P.O. Box 5164, Fargo, ND 58105-5164, USA
E-mail: anne.denton@ndsu.edu

Qiang Ding

Jiangsu Telecom Co., Ltd.,
Huan Cheng Nan Lu #88, Nantong, Jiangsu 226001, China
E-mail: qding74@gmail.com

**Abstract:** The k-nearest neighbour (KNN) technique is a simple yet effective method for classification. In this paper, we propose an efficient weighted nearest neighbour classification algorithm, called PINE, using vertical data representation. A metric called HOBBit is used as the distance metric. The PINE algorithm applies a Gaussian podium function to set weights to different neighbours. We compare PINE with classical KNN methods using horizontal and vertical representation with different distance metrics. The experimental results show that PINE outperforms other KNN methods in terms of classification accuracy and running time.

**Keywords:** nearest neighbours; classification; data mining; vertical data; spatial data.

**Reference** to this paper should be made as follows: Perrizo, W., Ding, Q., Khan, M., Denton, A. and Ding, Q. (200x) 'An efficient weighted nearest neighbour classifier using vertical data representation', *Int. J. Business Intelligence and Data Mining*, Vol. x, No. x, pp.xxx–xxx.

**Biographical notes:** William Perrizo is a Professor of Computer Science at North Dakota State University. He holds a PhD Degree from the University of Minnesota, a Master's Degree from the University of Wisconsin and a Bachelor's Degree from St. John's University. He has been a Research Scientist at the IBM Advanced Business Systems Division and the USA Air Force Electronic Systems Division. His areas of expertise are data mining, knowledge discovery, database systems, distributed database systems, high-speed computer and communications networks, precision agriculture and bioinformatics. He is a member of ISCA, ACM, IEEE, IAAA and AAAS.

Qin Ding is an Assistant Professor in Computer Science at Pennsylvania State University at Harrisburg, USA. She received her PhD in Computer Science from North Dakota State University, USA, MS and BS in Computer Science from Nanjing University, China. Her research interests include data mining, database, bioinformatics and spatial data.

Maleq Khan received BS in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Bangladesh and MS in Computer Science from North Dakota State University, ND, USA. He is currently working toward the PhD Degree in Computer Science at Purdue University, IN, USA. His research interests include distributed algorithms, communication networks, wireless sensor networks, bioinformatics and data mining.

Anne Denton is an Assistant Professor in Computer Science at North Dakota State University, USA. Her research interests are in data mining, knowledge discovery in scientific data and bioinformatics. Specific interests include data mining of diverse data, in which objects are characterised by a variety of properties such as numerical and categorical attributes, graphs, sequences, time-dependent attributes and others. She received her PhD in Physics from the University of Mainz, Germany and her MS in Computer Science from North Dakota State University, Fargo, ND, USA.

Qiang Ding received his MS and PhD in Computer Science from North Dakota State University, USA, BS in Computer Science from Nanjing University, China. He was an Assistant Professor in Computer Science at Concordia College, MN, USA, before he joined Jiangsu Telecom Co., Ltd., China. His research interests include database and data mining.

## 1 Introduction

The nearest neighbour technique (Cover and Hart, 1967; Cover, 1968; Dasarathy, 1991; Duda and Hart, 1973; McLachlan, 1992; Safar, 2005) is a simple yet effective method for classification. Given a set of training data, a KNN classifier predicts the class value for a new sample X by searching the training set for the KNNs of X based on certain distance

metric and then assigning to X the most frequent class occurring in its KNNs. The value of *k*, i.e., the number of neighbours, is pre-specified. Nearest neighbour classifiers are lazy learners in that a classifier is not built until a new sample needs to be classified. They differ from eager classifiers, such as decision tree induction (Breiman et al., 1984; Quinlan, 1993; Jeong and Lee, 2005; James, 1985; Fu, 2005).

In classical KNN methods, data are represented in horizontal format. Each training observation is a tuple that consists of *n* features (or called attributes). Each time when a new sample needs to be classified, the entire training data set is scanned to find the KNNs of this new sample. The database scan can be costly if the training data is extremely large. In this paper, we propose a nearest neighbour algorithm, called Podium Incremental Neighbour Evaluator (PINE), using vertical data representation. Vertical data representation can usually provide high compression ratio for large datasets. Vertical representation is particularly suitable for spatial data, not only because the spatial data set is typically large, but also because the continuity of spatial neighbourhood provides potential for even higher compression ratio using vertical representation.

In most KNN methods, each of the KNNs casts an equal vote for the class of the new sample. In the PINE algorithm, we apply a Gaussian podium function to different neighbours with different weights. The podium function is chosen in such a way that the neighbours close to the sample have higher impact than others in determining the class of the sample. In addition, all the samples in the database are considered neighbours, though some samples have very low or even zero weights. By doing so, there will be no need to pre-specify the *k* value and more importantly, high classification accuracy will be achieved.

The idea of using weights in KNN is not new. It is common to assign weights to different features in the distance metrics since some features may be more relevant than others in terms of 'closeness' or distance (Fu and Wang, 2005). Weights can also be assigned to the instances. Cost and Salzberg presented a method of attaching weights to different instances for learning with symbolic features (Cost and Salzberg, 1993). Some instances are more reliable and are considered 'exemplars'. These reliable exemplars are given smaller weights to make them appear closer to a new sample. In our work, based on the distances from the new sample, we partition the entire training set into different groups (called 'rings') and then assign a weight to each ring using a podium function. The podium function establishes a riser height for each step of the podium weighting function as the distance from the sample grows. The idea of a podium function is similar to the concept of a 'radial basis function' (Neifeld and Psaltis, 1993; Ali and Smith, 2005).

In this paper, we use a vertical data representation called P-tree[1] (Perrizo et al., 2001a; Ding et al., 2002). P-trees are compact and data-mining-ready structure, which provides lossless representation of the original horizontal data. P-trees represent bit information that is obtained from the data through a separation into bit planes. The multi-level structure is used to achieve high compression. A consistent multi-level structure is maintained across all bit planes of all attributes. This is done so that a simple multi-way logical AND operation can be used to reconstruct count information for any attribute value or tuple.

Different metrics can be defined for 'closeness' of two data points. In this paper, we use a metric called High Order Basic Bit similarity (HOBBit) (Khan et al., 2002). HOBBit distance metric is particularly suitable for spatial data.

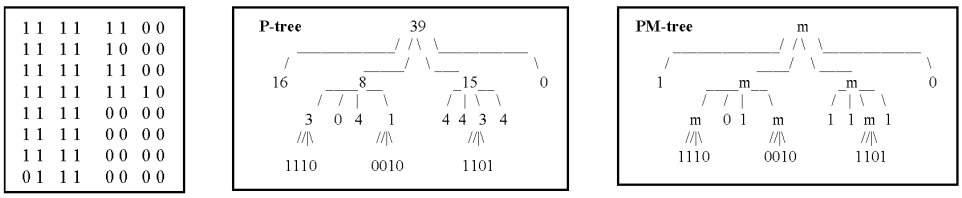
We run the PINE algorithm on very large real image datasets. Our experimental results show that it is much faster to build a KNN classifier using vertical data representation than using horizontal data representation for very large datasets. It also shows that the PINE algorithm achieves higher classification accuracy than other KNN methods.

The rest of the paper is organised as follows. In Section 2, we review the vertical P-tree structure. In Section 3, we introduce the HOBBit metric and detail our PINE algorithm. Performance analysis is given in Section 4. Section 5 concludes the paper.

## 2 The vertical P-tree structure review

We use a vertical structure, called a Peano Count Tree (P-tree), to represent the data. We first split each attribute value into bits. For example, for image data, each band is an attribute and the value is represented as a byte (8 bits). The file for each individual bit is called a bSQ file. For an attribute with m-bit values, we have m bSQ files. We organise each bSQ bit file, $B_{ij}$ (the file constructed from the $j$th bits of $i$th attribute), into a P-tree. An example of an $8 \times 8$ bSQ file and its P-tree is given in Figure 1.

**Figure 1** P-tree and PM-tree



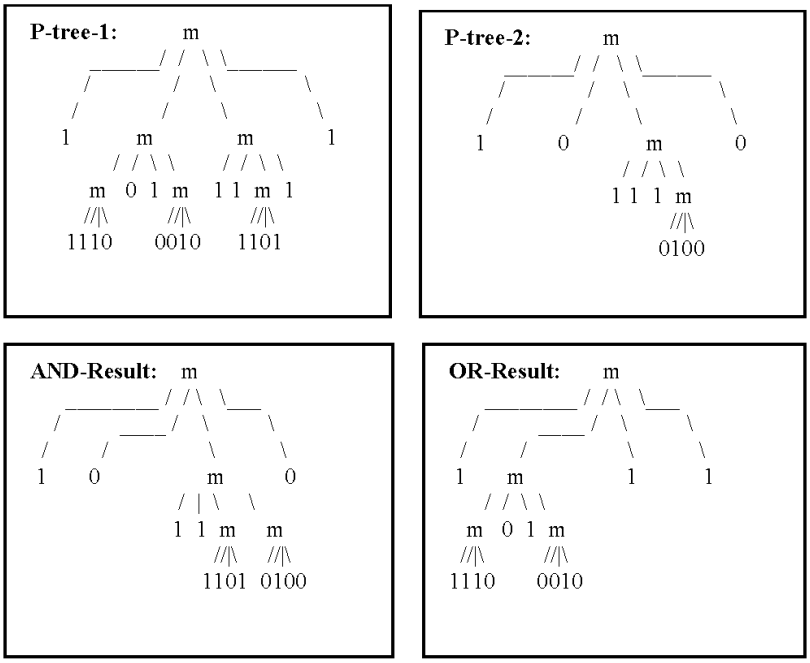

In this example, 39 is the count of 1's in the bSQ file, called root count. The numbers at the next level, 16, 8, 15 and 16, are the 1-bit counts for the four major quadrants. Since the first and last quadrant is made up of entirely 1-bits and 0-bits respectively (called pure1 and pure0 quadrant respectively), 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 P-trees. The process will eventually terminate at the 'leaf' level where each quadrant is a 1-row-1-column quadrant.

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

For efficient implementation, we use a variation of P-trees, called PM-tree (Pure Mask tree), in which mask instead of count is used. In the PM-tree, 3-value logic is used, i.e., 11 represents a pure1 quadrant, 00 represents a pure0 quadrant and 01 represents a mixed quadrant. To simplify, we use 1 for pure1, 0 for pure0 and $m$ for mixed. This is illustrated in the third part of Figure 1.

P-tree algebra contains operators AND, OR, NOT and XOR, which are the pixel-by-pixel logical operations on P-trees (Ding et al., 2002). The NOT operation is a straightforward translation of each count to its quadrant-complement. The AND and OR operations are shown in Figure 2.

**Figure 2** P-tree algebra (AND and OR)




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 P-tree for attribute, $b$ and value, $v$, where $v$ can be expressed in any bit precision. Using the 8-bit precision for values, $P_{b,11010011}$ can be constructed from the basic P-trees as:

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

Where ′ indicates the 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 attribute-$i$, the quadrant-wise count of occurrences of this combination of values is given by:

$$P_{(v_1,v_2,\ldots,v_n)} = P_{1,v_1} \text{ AND } P_{2,v_2} \text{ AND} \ldots \text{AND } P_{n,v_n}$$

P-trees are implemented in an efficient way, which facilitates fast P-tree operations and considerable saving of space. More details can be found in Ding et al. (2002) and Perrizo et al. (2001b).

## 3 Distance-weighted nearest neighbour classification using vertical data representation

In classical KNN classification techniques, there is a limit placed on the number of neighbours. In our distance-weighted neighbour classification approach, the podium or distance weighting function establishes a riser height for each step of the podium weighting function as the distance from the sample grows. This approach gives users maximum flexibility in choosing just the right level of influence for each training sample in the entire training set. The real question is, can this level of flexibility be offered without imposing a severe penalty with respect to the speed of the classifier. Traditionally, sub-sampling, neighbour-limiting and other restrictions are introduced precisely to ensure that the algorithm will finish its classification in reasonable time. The use of the compressed, data-mining-ready data structure, the P-tree, in fact, makes PINE even faster than traditional methods. This is critically important in classification since data are typically never discarded and therefore the training set will grow without bound. The classification technique must scale well or it will quickly become unusable in this setting. PINE scales well since its accuracy increases as the training set grows while its speed remains very reasonable (see the performance study below). Furthermore, since PINE is lazy (does not require a training phase in which a closed form classifier is pre-built), it does not incur the expensive delays required for rebuilding a classifier when new training data arrives. Thus, PINE gives us a faster and more accurate classifier.

Before explaining how distance-weighted neighbour classifier PINE using P-trees works, we give an overview of the technique of KNN classification and illustrate how to calculate KNN using P-trees without considering weights. In KNN the basic idea is that the tuples that most likely belong to the same class are those that are similar in the other attributes. This continuity assumption is consistent with the properties of a spatial neighbourhood.

Based on some pre-selected distance metric or similarity measure, such as Euclidean distance, classical KNN finds the $k$ most similar or nearest training samples to an unclassified sample and assigns the plurality class of those $k$ samples to the new sample (Dasarathy, 1991; Neifeld and Psaltis, 1993). The value for $k$ is pre-selected by the user based on the accuracy required (usually the larger the value of $k$, the more accurate the classifier) and the delay time required for classifying with that $k$-value (usually the larger the value of $k$ the slower the classifier). The steps of the classification process are:

- determine a suitable distance metric
- find the KNNs using the selected distance metric
- find the plurality class of the KNNs (voting on the class labels of the KNNs)
- assign that class to the sample to be classified.

We use a distance metric called HOBBit, which provides an efficient method of computation based on P-trees. Instead of examining individual training samples to find the nearest neighbours, we start our initial neighbourhood of the target sample within a specified distance in the feature space based on this metric, and then successively expand the neighbourhood area until there are at least $k$ tuples in the neighbourhood set.

Of course, there may be more boundary neighbours equidistant from the sample than are necessary to complete the KNN set, in which case, one can either use the larger set or arbitrarily ignore some of them. Traditional methods find the exact KNN set, since that is easiest using traditional techniques although it is clear that allowing some samples at that distance to vote and not others, will skew the result. Instead, the P-tree-based KNN approach builds a *closed* nearest neighbour set (closed-KNN) (Khan et al., 2002), that is, we include all of the boundary neighbours. The inductive definition of the closed-KNN set is given below.

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 tuple which cannot be produced by steps (a) and (b).

Experimental results show closed-KNN yields higher classification accuracy than KNN does. If there are many tuples on the boundary, inclusion of some but not all of them skews the voting mechanism. The P-tree implementation requires no extra computation to find the closed-KNN. Our neighbourhood expansion mechanism automatically includes the entire boundary of the neighbourhood. P-tree algorithms avoid the examination of individual data points, which improves the classification efficiency.

### *3.1 Higher Order Basic bit (HOBBit) distance*

Using a suitable distance or similarity metric is very important in KNN methods because it can strongly affect performance. There is no metric that works best for all the cases. Research has been done to find the right metric for the right problem (Short and Fukanaga, 1980, 1981; Friedman, 1994; Myles and Hand, 1990; Hastie and Tibshirani, 1996; Domeniconi et al., 2002).

The most commonly used distance metric is Euclidean metric. For two data points, $X = \langle x_1, x_2, x_3, \ldots, x_{n-1} \rangle$ and $Y = \langle y_1, y_2, y_3, \ldots, y_{n-1} \rangle$, the *Euclidean* similarity function is defined as

$$d_2(X,Y) = \sqrt{\sum_{i=1}^{n-1} (x_i - y_i)^2}$$

It can be generalised to the *Minkowski* similarity function,

$$d_q(X,Y) = \sqrt[q]{\sum_{i=1}^{n-1} w_i |x_i - y_i|^q}$$

If $q = 2$, this gives the Euclidean function. If $q = 1$, it gives the *Manhattan* distance, which is

$$d_1(X,Y) = \sum_{i=1}^{n-1} |x_i - y_i|$$

If $q = \infty$, it gives the *max* function

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

In this paper we use the HOBBit distance metric (Khan et al., 2002), which is a natural choice for spatial data. The HOBBit metric measures distance based on the most significant consecutive bit positions starting from the left (the highest order bit). Similarity or closeness is of interest. When comparing two values bitwise from left to right, once a difference is found, any further comparisons are not needed.

The HOBBit similarity between two integers $A$ and $B$ is defined by

$$S_H(A, B) = \max\{s | 0 \le i \le s \Rightarrow a_i = b_i\} \quad (1)$$

where $a_i$ and $b_i$ are the $i$th bits of $A$ and $B$ respectively.

The HOBBit distance between two tuples $X$ and $Y$ is defined by

$$d_H(X, Y) = \max_{i=1}^{n-1} \{m - S_H(x_i, y_i)\} \quad (2)$$

where $m$ is the number of bits in binary representations of the values; $n - 1$ is the number of attributes used for measuring distance (the nth being the class attribute); and $x_i$ and $y_i$ are the $i$th attributes of tuples $X$ and $Y$. The HOBBit distance between two tuples is a function of the least similar pairing of attribute values in them.

From the experiments, we found that HOBBit distance is more suitable for spatial data than other distance metrics.

### *3.2 Closed-KNN*

To find the closed KNN set, first we look for the tuples which are identical to the target tuple in all 8 bits of all bands, i.e., the tuples, $X$, having distance from the target $T$, $d_H(X, T) = 0$. If, for instance, $t_1 = 105$ ($01101001_b = 105_d$) is a target attribute value, the initial interval of interest is [105, 105] ([01101001, 01101001]). If over all tuples the number of matches is less than $k$, we compare attributes on the basis of the seven most significant bits, not caring about the 8th bit. The expanded interval of interest would be [104,105] ([01101000, 01101001] or [0110100-, 0110100-]). If $k$ matches still have not been found, removing one more bit from the right gives the interval [104, 107] ([011010--, 011010--]). Continuing to remove bits from the right we get intervals, [104, 111], then [96, 111] and so on.

This process is implemented using P-trees as follows. $P_{i,j}$ is the basic P-tree for bit $j$ of band $i$ and $P'_{i,j}$ is the complement of $P_{i,j}$. Let $b_{i,j}$ be the $j$th bit of the $i$th band of the target tuple, and for implementation purposes let the representation of the P-tree depend on the value of $b_{ij}$. Define:

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

Then the root count of $Pt_{i,j}$ is the number of tuples in the training dataset having the same value as the $j$th bit of the $i$th band of the target tuple. Define:

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

where & is the P-tree AND operator and $n$ is the number of bands. $Pv_{i,1-j}$ counts the tuples having the same bit values as the target tuple in the higher order $j$ bits of $i$th band. Then a neighbourhood P-tree can be formed as follows:

$$Pnn(j) = Pv_{1,1-j} \,\&\, Pv_{2,1-j} \,\&\, Pv_{3,1-j} \,\&\, \ldots \,\&\, Pv_{n-1,1-j} \tag{4}$$

We calculate the initial neighbourhood P-tree, $Pnn(8)$, matching exactly in all bands, considering 8-bit values. Then we calculate $Pnn(7)$, matching in seven higher order bits; then $Pnn(6)$ and so on. We continue as long as the root count of $Pnn(j)$ is less than $k$. Let us denote the final $Pnn(j)$ by $Pcnn$. $Pcnn$ represents the closed-KNN set and the root count of $Pcnn$ is the number of the nearest tuples. A bit of one in $Pcnn$ for a tuple means that the tuple is in the closed-KNN set. For the purpose of classification, we do not 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 $Pcnn \,\&\, P_n(i)$ represents the tuples having a class value $i$ that are in the closed-KNN set, $Pcnn$. An $i$, which yields the maximum root count of $Pcnn \,\&\, P_n(i)$ is the plurality class; that is

$$\text{predicted class} = \arg\max_i \{\text{RC}(Pcnn \,\&\, P_n(i))\} \tag{5}$$

where, RC($P$) is the root count of $P$.

### *3.3 PINE: weighted nearest neighbour classifier using vertical data representation*

The continuity assumption of KNN tells us that tuples that are more similar to a given tuple have more influence on classification than tuples that are less similar. Therefore giving more voting weight to closer tuples than distant tuples increases the classification accuracy. Instead of considering the KNNs, we include all of the points, using the largest weight, 1, for those matching exactly, and the smallest weight, 0, for those furthest away. Figure 3 shows the neighbourhood rings using HOBBit metric. Many weighting functions which decrease with distance can be used (e.g., Gaussian, Kriging, etc.). A simple example of such function is a linear podium function (shown in Figure 4) which decreases step-by-step with distance.

Note that the HOBBit distance metric is ideally suited to the definition of neighbourhood rings, because the range of points that are considered equidistant grows exponentially with distance from the centre. Adjusting weights is particularly important for small to intermediate distances where the podiums are small. At larger distances where fine-tuning is less important the HOBBit distance remains unchanged over a large range, i.e., podiums are wider. Ideally, the 0-weighted ring should include all training samples that are judged to be too far away (by a domain expert) to influence class.

**Figure 3** Neighbourhood rings using HOBBit metric

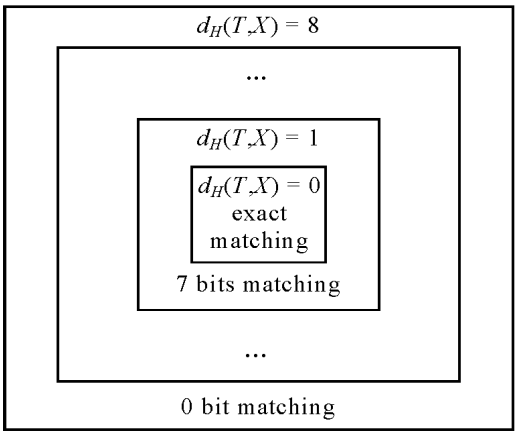
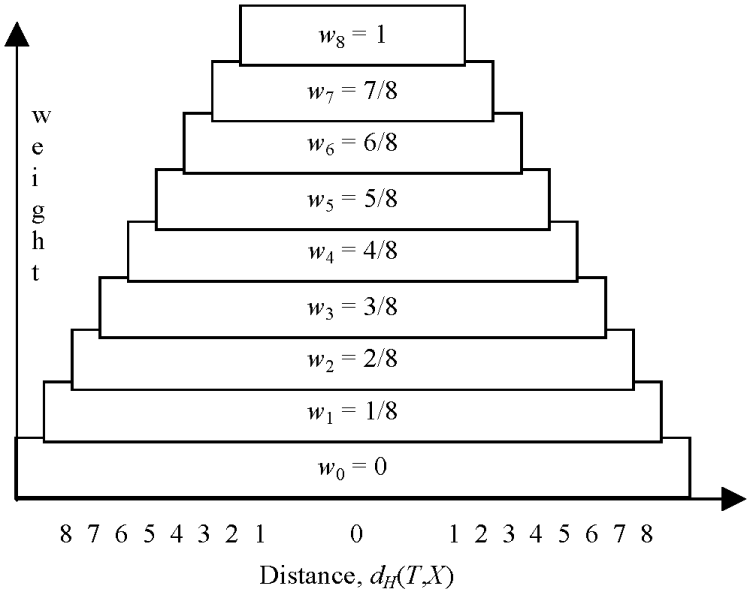



**Figure 4** An example of a linear podium function




We number the rings from 0 (outermost) to $m$ (innermost). Let $w_j$ be the weight associated with the ring $j$. Let $c_{ij}$ be the number of neighbour tuples in the ring $j$ belonging to the class $i$. Then the total weight vote by the class $i$ is given by:

$$V(i) = \sum_{j=0}^{m} w_j c_{ij} \tag{6}$$

This can easily be transformed to:

$$V(i) = w_0 \sum_{k=0}^{m} c_{il} + \sum_{j=1}^{m} \left\{ (w_j - w_{j-1}) \sum_{k=j}^{m} c_{ik} \right\} \tag{7}$$

Let circle $j$ be the circle formed by the rings $j, j + 1, \ldots, m$, that is, the ring $j$ including all of its inner rings. Referring to equation (4), the P-tree, $Pnn(j)$, represents all of the tuples in the circle $j$. Therefore, $\{Pnn(j) \,\&\, P_n(i)\}$ represents the tuples in the circle $j$ and class $i$; $P_n(i)$ is the P-tree for class $i$. Hence:

$$\sum_{k=j}^{m} c_{ik} = \mathrm{RC}\{Pnn(j) \,\&\, P_n(i)\} \tag{8}$$

$$V(i) = w_0 \mathrm{RC}\{Pnn(0) \,\&\, P_n(i)\} + \sum_{j=1}^{m} [(w_j - w_{j-1}) \mathrm{RC}\{Pnn(j) \,\&\, P_n(i)\}] \tag{9}$$

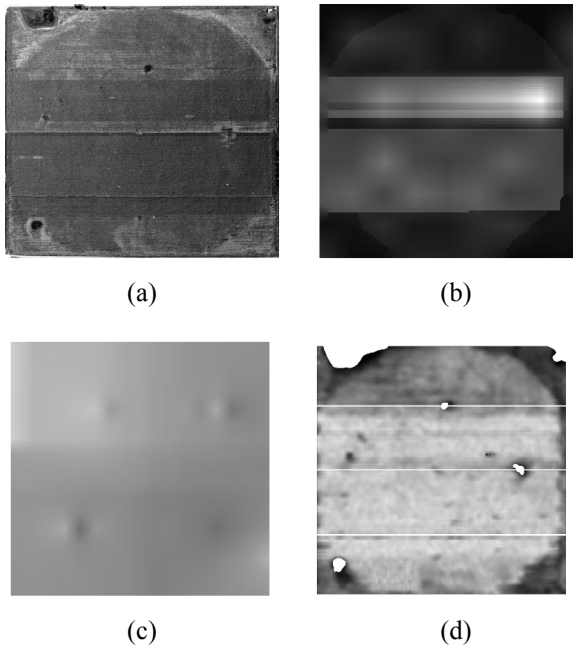
An $i$ which yields the maximum weighted vote, $V(i)$, is the plurality class or the predicted class; that is:

$$\text{predicted class} = \arg\max_i \{V(i)\} \tag{10}$$

## 4 Performance analysis

We have performed experiments to evaluate PINE on real data sets. An example data set is given in Figure 5. This data set includes an aerial TIFF image (with Black grey, Light grey and White grey bands) and associated ground data, such as nitrate, moisture and yield, of the Oaks area in North Dakota. In this data set, yield is the class label attribute, i.e., the goal is to predict the yield based on the values of other attributes. This data set and some other data sets are available at http://www.cs.ndsu.nodak.edu/~datasurg/.

**Figure 5** An image dataset: (a) TIFF image; (b) nitrate map; (c) moisture map and (d) yield map



(a) (b)

(c) (d)

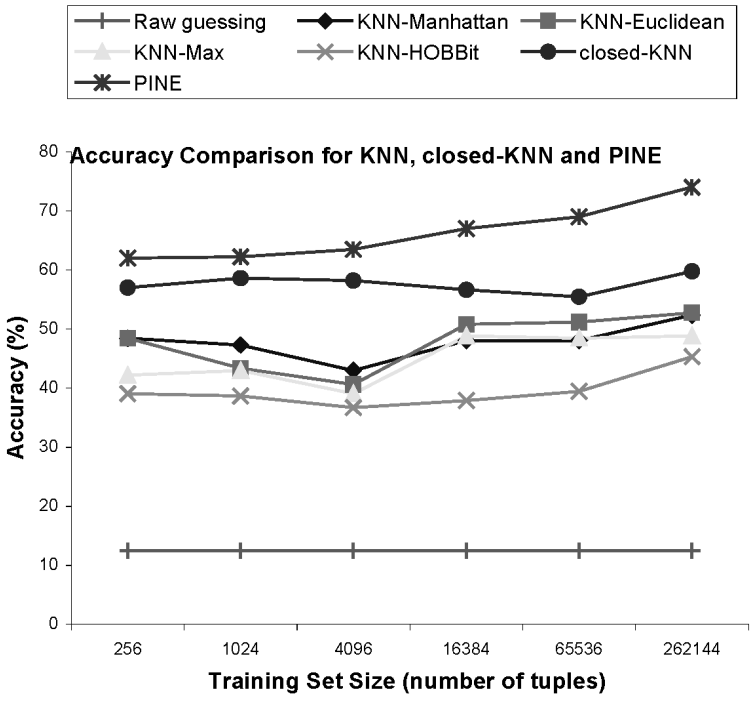
We tested KNN with Manhattan, Euclidean, Max and HOBBit distance metrics; closed-KNN with the HOBBit metric and PINE. In PINE, HOBBit was used as the distance function and the Gaussian function was used as the podium function. The function is $\exp(-(2^{2\times d})/(2\times\sigma^2))$, where $d$ is the HOBBit distance and the variance $\sigma$ is $2^4$. The mapping of the function is given in Table 1.

**Table 1** Gaussian weighs as the function of HoBBit distance

| HOBBit distance | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Gaussian weigh | 1.00 | 1.00 | 0.97 | 0.88 | 0.61 | 0.14 | 0 | 0 |

The comparison of the accuracy is given in Figure 6. We observed that PINE achieves higher classification accuracy than closed-KNN does. Especially when the training set size increases, the improvement of PINE over closed-KNN is more apparent. In additional, PINE performs much better than classical KNN methods using various metrics. All these classifiers work better than raw guessing, which is 12.5% in the data set with eight classes.

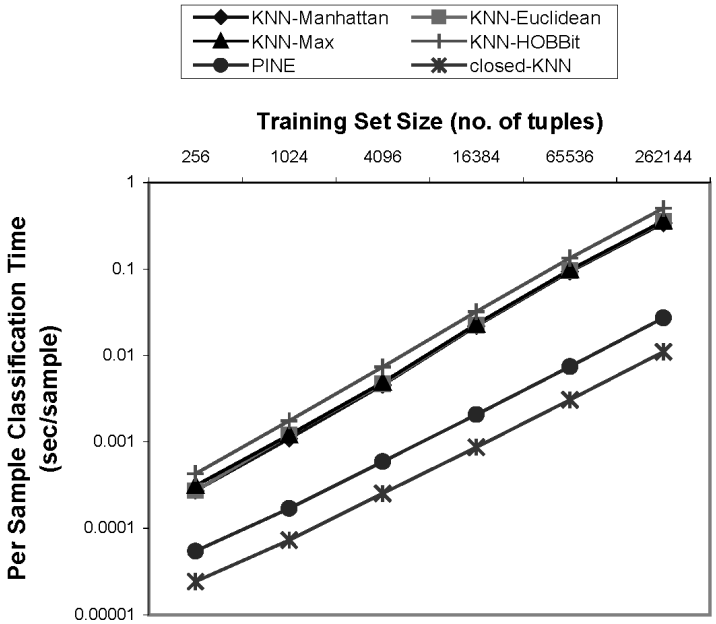
**Figure 6** Comparison of classification accuracy for KNN using different metrics, closed-KNN and PINE




In terms of running time, from Figure 7, we observed that both closed-KNN and PINE are much faster than classical KNN methods using various metrics (notice that both the size and classification time are plotted in logarithmic scale). The reason behind this is because both PINE and closed-KNN use vertical data representation, which provides fast calculation of nearest neighbours. We can also see that PINE is slightly slower than

closed-KNN. This is expected because in PINE all the training samples are included as neighbours while in closed-KNN only *k* samples (with possibly extra points on the boundary) are selected as neighbours to be involved in the calculation. However, this additional time cost is relatively small. On the average, PINE is eight times faster than classical KNN, and closed-KNN is ten times faster. Both PINE and closed-KNN increase at a lower rate than classical KNN methods do when the training set size increases.

**Figure 7** Comparison of classification time per sample (size and classification time are plotted in logarithmic scale)




## 5 Conclusions

In this paper, we propose a weighted nearest neighbour classifier, called PINE. PINE uses a vertical data structure P-tree and a distance metric called HOBBit. A Gaussian podium function is used to weight the different neighbours based on the distance from the sample data. PINE does not require providing the number of neighbours in advance. PINE is particular useful for classification on large data sets, such as spatial data sets, which have high compression ratio using vertical data representation. Performance analysis shows that PINE outperforms classical KNN methods in terms of accuracy and speed of classification on spatial data.

In addition, PINE has potential for efficient classification on data streams. In data streams, new data keep arriving, so the classification efficiency will be an important issue. PINE provides high efficiency as well as accuracy for classification on data streams. As the areas of data mining applications grow (Chen and Liu, 2005), our approach has potential to be extended to some of these areas, such as DNA microarray data analysis and medical image analysis.

## Acknowledgement

This work is partially supported by GSA Grant K96130308.

## References

Ali, S. and Smith, K.A. (2005) 'Kernel width selection for SVM classification: a meta-learning approach', *International Journal of Data Warehousing and Mining*, Idea Group Inc., Vol. 1, No. 4, pp.78–97.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classfication and Regression Trees*, Wadsworth, Belmont.

Chen, S.Y. and Liu, X. (2005) 'Data mining from 1994 to 2004: an application-orientated review', *International Journal of Business Intelligence and Data Mining*, Inderscience Publishers, Vol. 1, No. 1, pp.4–21.

Cost, S. and Salzberg, S. (1993) 'A weighted nearest neighbour algorithm for learning with symbolic features', *Machine Learning*, Vol. 10, No. 1, pp.57–78.

Cover, T.M. (1968) 'Rates of convergence for nearest neighbour procedures', *Proceedings of Hawaii International Conference on Systems Sciences*, Honolulu, Hawaii, USA, pp.413–415.

Cover, T.M. and Hart, P. (1967) 'Nearest neighbour pattern classification', *IEEE Trans. on Information Theory*, pp.21–27.

Dasarathy, B.V. (1991) *Nearest-Neighbour Classification Techniques*, IEEE Computer Society Press, Los Alomitos, CA.

Ding, Q., Khan, M., Roy, A. and Perrizo, W. (2002) 'The P-tree algebra', *Proceedings of ACM Symposium on Applied Computing*, pp.426–431.

Domeniconi, C., Peng, J. and Gunopulos, D. (2002) 'Locally adaptive metric nearest neighbour classification', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 9, pp.1, 281–1, 285.

Duda, R.O. and Hart, P.E. (1973) *Pattern Classification and Scene Analysis*, Wiley, New York.

Friedman, J. (1994) *Flexible Metric Nearest Neighbour Classification*, Technical report, Stanford University, Stanford, CA, USA.

Fu, L. (2005) 'Novel efficient classifiers based on data cube', *International Journal of Data Warehousing and Mining*, Idea Group Inc., Vol. 1, No. 3, pp.15–27.

Fu, X. and Wang, L. (2005) 'Data dimensionality reduction with application to improving classification performance and explaining concepts of data sets', *International Journal of Business Intelligence and Data Mining*, Inderscience Publishers, Vol. 1, No. 1, pp.65–87.

Hastie, T. and Tibshirani, R. (1996) 'Discriminant adaptive nearest neighbour classification', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 6, pp.607–615.

James, M. (1985) *Classification Algorithms*, John Wiley & Sons, New York.

Jeong, M. and Lee, D. (2005) 'Improving classification accuracy of decision trees for different abstraction levels of data', *International Journal of Data Warehousing and Mining*, Idea Group Inc., Vol. 1, No. 3, pp.1–14.

Khan, M., Ding, Q. and Perrizo, W. (2002) '*k*-Nearest Neighbour classification on spatial data streams using P-trees', *Proceedings of Pacific and Asia Knowledge Discovery and Data Mining (PAKDD), Lecture Notes in Artificial Intelligence*, Vol. 2336, pp.517–528.

McLachlan, G.J. (1992) *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York.

Myles, J.P. and Hand, D.J. (1990) 'The multi-class metric problem in nearest neighbour discrimination rules', *Pattern Recognition*, Vol. 23, pp.1, 291–1, 297.

Neifeld, M.A. and Psaltis, D. (1993) 'Optical implementations of radial basis classifiers', *Applied Optics*, Vol. 32, No. 8, pp.1370–1379.

Perrizo, W., Ding, Q., Ding, Q. and Roy, A. (2001a) 'On mining satellite and other remotely sensed images', *Proceedings of ACM Workshop on Research Issues on Data Mining and Knowledge Discovery*, Santa Barbara, CA, USA, pp.33–44.

Perrizo, W., Ding, Q., Ding, Q. and Roy, A. (2001b) 'Deriving high confidence rules from spatial data using Peano count trees', *Lecture Notes in Computer Science*, Vol. 2118, pp.91–102.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Safar, M. (2005) 'K nearest neighbour search in navigation systems', *Mobile Information Systems*, IOS Press, Vol. 1, No. 3, pp.207–224.

Short, R. and Fukanaga, K. (1980) 'A new nearest neighbour distance measure', *Proceedings of International Conference on Pattern Recognition*, Miami Beach, FL, USA, pp.81–86.

Short, R. and Fukanaga, K. (1981) 'The optimal distance measure for nearest neighbour classification', *IEEE Trans. on Information Theory*, Vol. 27, pp.622–627.

## Note

[1]P-tree technology is patented by North Dakota State University (William Perrizo, primary inventor of record); patent number 6,941,303 issued September 6, 2005.

## Website

TIFF image data sets, available at http://www.cs.ndsu.nodak.edu/~datasurg/.