A Measure for Vector Approximation

Benjamin J. Briedis
The School of Computer Science & Engineering
The University of New South Wales, UNSW Sydney 2052, Australia.
E-mail: bbriedis@cse.unsw.edu.au

UNSW-CSE-TR-0001
February 2000

Abstract

The creation of approximations for vectors for use in similarity searching (also known the retrieval of the k-nearest neighbours) is examined. A measure is derived that is suitable for judging the quality of a set of vector approximations. This measure is used in the modification of a technique used in similarity searching known as the VA-file. The modified VA-file is evaluated, and a clear improvement in performance is demonstrated.

1 Introduction

The potential use of *approximate vectors* for similarity searching is investigated in this report. An approximate vector is a vector that is constrained in the values it can take and which is used to represent another vector. Approximate vectors are of interest as they are effectively compressed forms of the vectors they represent. The major task in creating approximations of vectors is to preserve as much of the information content present in the original vectors as possible while compressing the representations. A measure is presented that is useful for judging the quality of vector approximations. This measure is used to modify a method for quickly searching vectors, known as the VA-file [8], and the performance of this modified technique is then evaluated experimentally.

It is assumed that there is a large collection of objects and that searches are frequently conducted on this collection in order to find some number of objects that are similar to the given query objects. It is further assumed that the objects and the queries are represented using vectors, and that these vectors are compared using a similarity measure. This form of similarity searching is often called *k-nearest neighbour searching.* In this case, however, objects are thought of as being represented as points, and a distance measure is used to compare points. The terminology adopted here will be one of vectors and similarity, rather than one of points and distances. The vectors that are being investigated here will typically be long (from about 10 dimensions to a few hundred dimensions), and the searches that are performed are likely to be incomplete, in that they will find most rather than all of the vectors closest to a query vector. This type of search has a wide range of applications [2, 1, 4, 7, 3, 6, 5].

2 Measuring the Quality of Approximations

Given that a number of vectors are to be approximated in preparation for later indexing, it will generally be desirable to minimise the error that results from the process of approximation. In order to determine how best to perform the approximations it is therefore necessary to choose some measure of error that compares the true similarity of a pair of vectors to the similarity achieved when using approximate forms of the vector pairs. Let $\vec{X}$ be a vector random variable that represents an item in the collection and let $\vec{Y}$ be a vector random variable that represents a query. Let $S = s(\vec{X}, \vec{Y})$ be the similarity between $\vec{X}$ and $\vec{Y}$. Let $\vec{X}'$ and $\vec{Y}'$ be approximations of $\vec{X}$ and $\vec{Y}$ and let $T = t(\vec{X}', \vec{Y}')$ be the *approximate similarity* of $\vec{X}$ and $\vec{Y}$. s and t are functions that return a similarity and they are probably the same function. Note that each value of S has an associated approximate similarity T. If a sample of readings is taken, a set is obtained of the form $\{\langle s_i, t_i \rangle \mid 1 \leq i \leq n\}$ where n is the sample size. The measure proposed here to determine the quality of the approximations operates on this set of pairs. Measures that compare the similarity / approximate similarity pairs have the virtue of not needing to explicitly refer to the similarity measure used or to the distribution of the data.

Various measures could be used for comparing sets of pairs of elements, two examples being the total sum of squares (TSS) and the correlation. The measure that appears to be the most appropriate to use, however, is $\mathrm{Var}(S-T)$. Justification for use of $\mathrm{Var}(S-T)$ as a measure is largely provided by the Theorem 1, presented below.

First define the *completeness* (C) of a search to be:

$$\theta = \frac{|\mathcal{C} \cap \mathcal{R}|}{|\mathcal{C}|} \tag{1}$$

where $\mathcal{C}$ and $\mathcal{R}$ are the sets of *close* items and *retrieved* items respectively. Also let $\bar{\mathcal{C}}$ and $\bar{\mathcal{R}}$ be their complements. It is assumed that $|\mathcal{R}| = |\mathcal{C}|$.

Theorem 1 *Let S, T and D be random variables with S representing the similarities between items in the collection and all possible queries, and T representing the approximations of these similarities. Let $T = S + D$, where D is independent of S. Minimising* $\mathrm{Var}(S-T)$ *maximises the expected completeness of retrieval.*

Proof: Let $D = \mu + \sigma N$, where N is a standardised random variable. Note that $\mathrm{Var}(S-T) = \sigma^2$, and that minimising σ is equivalent to minimising $\mathrm{Var}(S-T)$.

Various values of σ will considered. Let the unprimed forms of the variables (e.g. $\mathcal{R}$) relate to σ, the primed forms (e.g. $\mathcal{R}'$) relate to σ', and the double primed forms (e.g. $\mathcal{R}''$) relate to σ''. Let variables for which $\sigma = 0$ be denoted using a * (e.g. $\mathcal{R}^*$).

For a particular query consider two values taken by T, t_a and t_b. All approximate similarities may be calculated as:

$$t_i = s_i + \sigma n_i + \mu \tag{2}$$

where i is an item to be retrieved and s_i and n_i are values taken by S and N respectively. From this it is evident that t_a and t_b are both linear with respect to σ. Consequently the lines given by (σ, t_a) and (σ, t_b) cross at most once.

In the case where $t_a > t_b$ and $t'_b > t'_a$ given $\sigma' > \sigma \geq 0$ the lines must cross in the interval $(\sigma..\sigma')$ as there is at most one crossover, $t''_b > t''_a$ for all $\sigma'' \geq \sigma'$.

Consider two points $a \in \mathcal{C} \cap \bar{\mathcal{R}}'$ and $b \in \bar{\mathcal{C}} \cap \mathcal{R}'$. $a \in \mathcal{R}^*$ as $a \in \mathcal{C}$ and $b \in \bar{\mathcal{R}}^*$ as $b \in \bar{\mathcal{C}}$. All close items have similarities greater than those of non-close items so $s_a > s_b$ or, in an alternative notation, $t_a > t_b$ where $\sigma = 0$. As $a \in \bar{\mathcal{R}}'$ and $b \in \mathcal{R}'$, and as all items that are retrieved have approximate similarities greater than those that are not retrieved, it can be seen that $t'_b > t'_a$. The lines given by (σ, t_a) and (σ, t_b) therefore must cross in the interval $(0..\sigma')$. Therefore

$$t''_b > t''_a \quad \text{for all} \quad \sigma'' > \sigma' \quad \text{when} \quad a \in \mathcal{C} \cap \bar{\mathcal{R}}',\, b \in \bar{\mathcal{C}} \cap \mathcal{R}' \tag{3}$$

Consider Equation 1. As $|\mathcal{C}|$ is fixed, an increase in θ implies an increase in $|\mathcal{C} \cap \mathcal{R}|$. An increase in $|\mathcal{C} \cap \mathcal{R}|$ will only occur if an element of in $\mathcal{C} \cap \bar{\mathcal{R}}$ replaces an element in $\bar{\mathcal{C}} \cap \mathcal{R}$. Increasing σ cannot result in such a replacement by Equation 3. Therefore θ is non-increasing with respect to σ. Note that θ is a function of $\mathcal{R}$ (Equation 1), which is a function of the values of T, which are functions of σ (Equation 2), so a non-increasing functional relationship $(\theta = f(\sigma))$ exists between θ and σ.

The *expected completeness* is the expected value of θ ($\mathrm{E}\theta$) over all searches. Now $\mathrm{E}\theta = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} f_q(\sigma)$, where $\mathcal{Q}$ is the set of queries. As each f_q is non-increasing with respect to σ, $\mathrm{E}\theta$ is non-increasing with respect to σ. Thus minimising σ maximises $\mathrm{E}\theta$, and as minimising σ is equivalent to minimising $\mathrm{Var}(S-T)$, minimising $\mathrm{Var}(S-T)$ maximises the expected completeness. Note, however, that not every reduction in $\mathrm{Var}(S-T)$ need lead to an increase in the expected completeness. □

This theorem makes one assumption: that the true score and the error that is introduced by creating an approximation of it are independent of one another. This assumption is unlikely to ever hold entirely. For a 1-dimensional vector the dependency will be high. It is believed, however, that as dimensions are added the effects due to any one dimension are likely to be obscured. This belief is tested below. Furthermore is should be noted that if the distribution of the error only shifts gradually, particularly for scores around the score used as a cutoff for retrieval, then the measure should remain fairly effective.

In order to test the extent of the dependency of the error D on the similarity score S, four simple experiments of the following form were conducted. A set of tuples whose elements are vectors were generated. The Euclidean distance (S) between each pair of vectors was calculated. The domain of each of the dimensions of the vectors was partitioned, and each partition assigned an approximate value. The distance between the approximate vectors was calculated (T), so the error between each pair of vectors was $D = T - S$. The probability density function (PDF) of D was calculated for those vector pairs whose values of S lay within the largest 10%, middle 10% and smallest 10% of values. The PDFs for vectors of dimensionality 1 and 10 are shown in Figure 1. This procedure was followed for four data sets: for 1-dimensional uniformly distributed vectors, for 10-dimensional uniformly distributed vectors, for 1-dimensional normally distributed vectors, and for 10-dimensional normally distributed vectors. The standard normal distribution was used to generate the elements in each of the dimensions in the normal data sets. In the case of the uniformly distributed data the partition boundary points were placed at 0.125, 0.25, 0.375, 0.5, 0.625, 0.75 and 0.875, with the approximation values of each of the partitions being placed at the midway points between the partition boundary points (where 0 and 1 are treated as the endpoints of the end-most partitions). In the case of the normally distributed data the partition boundary points were placed at by partitioning the domain of each element at the points -3, -2, -1, 0, 1, 2 and 3, with the approximation values being -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5 and 3.5. The partitioning was somewhat arbitrary in both the uniform and normal cases, but it was designed to cover the data fairly well.

These graphs demonstrate that adding dimensions does reduce the dependency of the error on the similarity. When there is only one dimension there is a very strong dependency whereas when there are 10 the PDFs more closely resemble one another. Nonetheless some dependency is clearly present and this will effect the performance of $\mathrm{Var}(S-T)$ as a measure. The persistence of the dependency is probably due to the strength of the dependency in the single dimension case. The similarity measure used, which emphasises differences in individual dimensions would also have an effect. The degree of dependency that is evident for 10 dimensions does indicate that some caution is needed when using $\mathrm{Var}(S-T)$ as a measure. It is possible that even in a data set which has a very large number of dimensions,

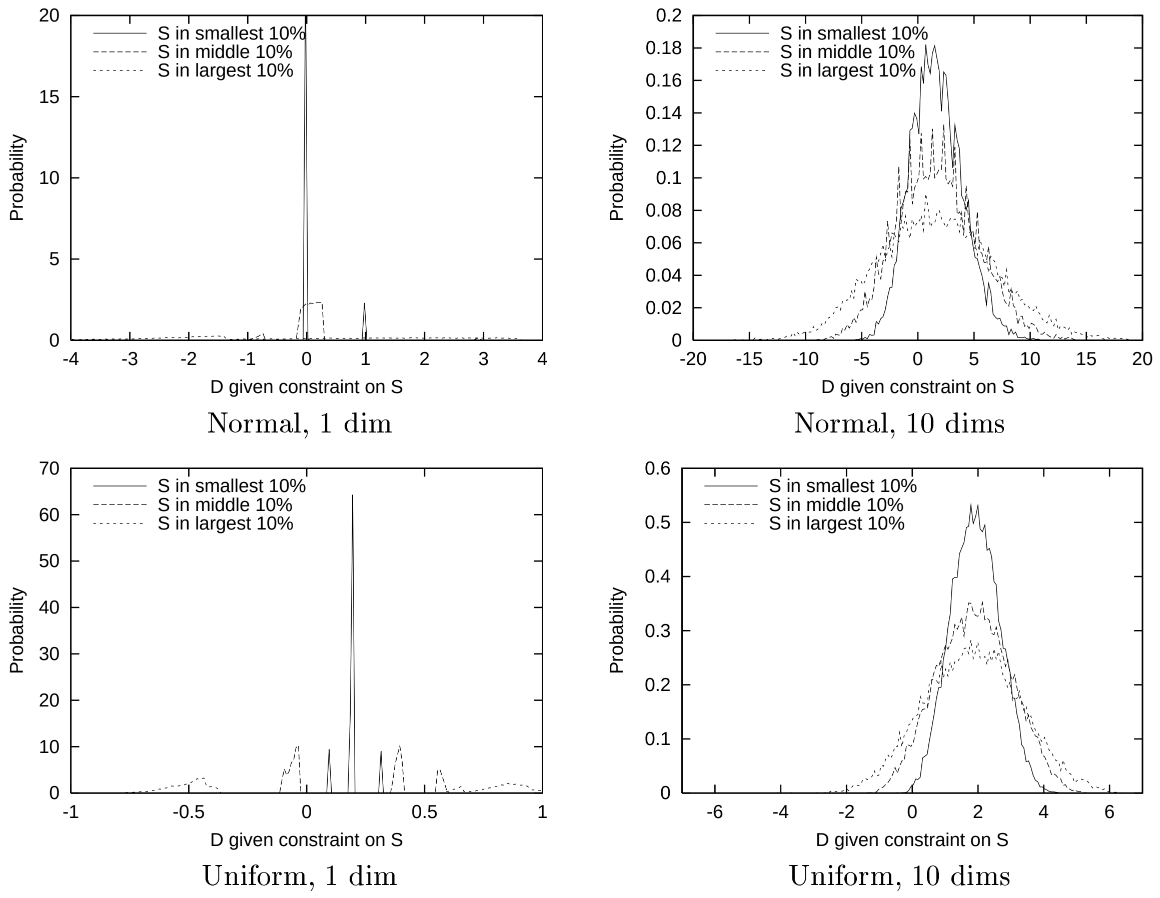

Figure 1: The PDFs of the error D given that the similarity S is in the largest, middle or smallest 10% of S values for 1- and 10-dimensional vectors respectively.

a few may dominate in their contribution to the overall similarity scores, and to the error.

3 Minimising Error in One Dimension

3.1 Simplifying Assumptions

It is desirable to be able to deal with each dimension independently of the others when deciding where to place the partition boundary points as this simplifies calculations and reduces computation time. To aid this, two assumptions are adopted.

3.1.1 Independence of Elements

It is assumed that the distributions of elements in each dimension are independent of the other dimensions for both the item vectors and the query vectors.

3.1.2 The Similarity Measure as a Summation

In order to work with the dimensions separately, it is useful to break the similarity measure down into smaller components. Several measures can be represented as, or approximated by, a sum of *part-similarities*, each part being attributable to one dimension. Thus the similarity may be represented as:

$$s(\vec{x},\vec{y}) = \sum_{i=1}^{d} s_i(x_i, y_i)\,, \tag{4}$$

where $\vec{x} = (x_1, x_2, \ldots, x_d)$ and $\vec{y} = (y_1, y_2, \ldots, y_d)$ are two vectors. In many cases $s_i = s_j$, for all $1 \leq i, j \leq d$.

One of the simplest cases is the dot product. It can easily be seen that the dot product can be expressed in the form of Equation 4:

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{d} x_i y_i \tag{5}$$

The form in Equation 4 can normally also be used in conjunction with the l_p metrics. In retrieving nearest neighbours only the order of the distances of the items is generally significant, not the actual value of the distances. Thus the same results are produced using the measure $[l_p(\vec{x},\vec{y})]^p$ as are produced when using $l_p(\vec{x},\vec{y})$. The first of these expressions is simply a summation of terms each dependent on only one dimension each:

$$[l_p(\vec{x},\vec{y})]^p = \sum_{i=1}^{d} |x_i - y_i|^p \tag{6}$$

and this is in the form of Equation 4. In this case each term is a *part-distance*, rather than a part-similarity.

The cosine measure *cannot* be represented exactly in this form. A possible approximation, however, is:

$$\sum_{i=1}^{d} \frac{x_i y_i}{\sqrt{(x_i - y_i)^2 + \mathrm{E}\sum_{j \neq i}(x_j - y_j)^2}} \tag{7}$$

where $\mathrm{E}\sum_{j \neq i}(x_j - y_j)^2$ is treated as a constant. It is, however, simpler to normalise all of the vectors (including the query vectors) and to use the dot product or the Euclidean distance instead. The dependency between the elements of the dimensions which is introduced by using the cosine measure is typically small and can probably be disregarded in most cases.

3.2 The Calculation of Error

In order to approximate the elements of one dimension it is desirable to have a separate measure of error for each dimension. As has been seen, the similarity score may be expressed

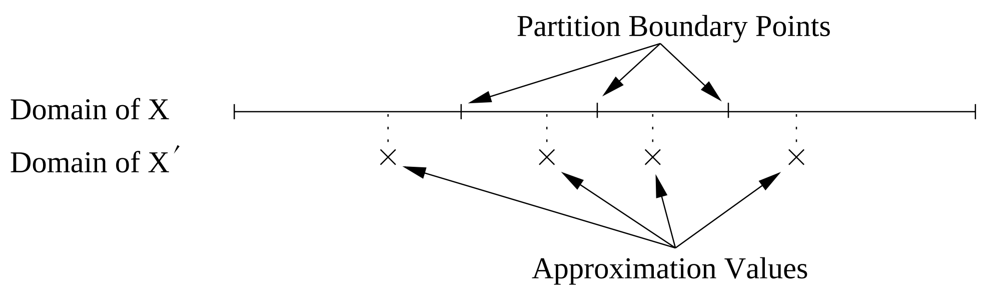

Figure 2: Approximating the elements in a dimension

as a summation of part-similarities. T may similarly be considered as a summation of *part-approximate similarities*:

$$T = \sum_{i=1}^{d} T_i \tag{8}$$

So choosing $\mathrm{Var}(S - T)$ as the measure of error:

$$\begin{aligned} \mathrm{Var}(S-T) &= \mathrm{Var}\left(\sum_{i=1}^{d} S_i - \sum_{i=1}^{d} T_i\right) \qquad \text{(by Equations 4 and 8)} & (9)\\ &= \sum_{i=1}^{d} \mathrm{Var}(S_i - T_i) & (10)\end{aligned}$$

Equation 10 uses the assumption that the dimensions are independent of one another. This independence implies that the terms $S_i - T_i$ are independent of one another. Note that they are random variables and that the variances of independent random variables sum. Thus minimising $\mathrm{Var}(S_i - T_i)$ for each dimension i individually minimises $\mathrm{Var}(S - T)$ overall.

Let X be a random variable representing the elements from one dimension in a set of vectors. In order to approximate X, let the domain of X be partitioned and associate with each partition an *approximation value*. Let X' be the approximation of X (see Figure 2). Although the values of X' are depicted as being drawn from the corresponding partitions, this need not necessarily be the case.

3.3 Optimising the Parameters

It is possible to optimise the approximations for a dimension i by adjusting the partition boundary points and the values taken by X' so as to minimise $\mathrm{Var}(S_i - T_i)$. If possible it would be best to set these parameters so that $\mathrm{Var}(S_i - T_i)$ achieves its global minimum. No technique for finding this minimum is known, so instead a heuristic optimisation algorithm has been used. A number of algorithms were designed, implemented and tested. The algorithm which performed the best, both in terms of minimising $\mathrm{Var}(S_i - T_i)$ and in the speed of its operation, is presented in the appendix. This algorithm was used in the experiments whose results are presented in Section 5.

Collection Name	Num. of candidates	Num. seeks	Index scan time	Seek time	Total time
Uniform (l_2)	22815.3	13.48	1.84 secs	2.12 secs	3.95 secs
Uniform (cos)	53087.7	17.70	2.91 secs	10.41 secs	13.33 secs
Normal(l_2)	99863.7	30.12	1.74 secs	8.85 secs	10.59 secs
Normal (cos)	100000.0	31.45	1.55 secs	8.78 secs	10.33 secs
Mixed (l_2)	71747.2	26.74	1.75 secs	5.48 secs	7.23 secs
Mixed (cos)	99361.7	39.56	1.82 secs	8.95 secs	10.77 secs
AustLII (cos)	98571.3	30.91	2.93 secs	8.72 secs	11.65 secs
ColourHistR1 (l_2)	47477.7	23.4	4.48 secs	8.01 secs	12.49 secs
ColourHistR5 (l_2)	59825.8	22.93	11.54 secs	6.41 secs	17.95 secs

Table 1: Statistics for exact VA-file retrieval. All statistics are mean averages. Each collection consists of 100 000 vectors, each element was approximated using 4 bits, and 100 queries were run for each collection.

4 The Modified VA-file

The VA-file as described by Weber [8] performs exact searches. Two separate files of vectors are maintained, a set of full-length vectors and a set of vector approximations. In order to create the file of vector approximations, the domain of each of the dimensions are first partitioned. Weber forms partitions that contain equal numbers of elements. When a search is conducted, the file of vector approximations is scanned and a minimum and maximum distance is calculated between each of the points in the collection and the query point. The kth smallest of the maximum distances is taken as a threshold. The closest k items must have distances less than or equal to the threshold. The vectors which have minimum distances that exceed this threshold are rejected. The remaining items will be referred to here as the *candidate* vectors. The candidates now need to be checked by accessing the full vectors from disk. In order to minimise the number of disk accesses required, the candidates are first sorted according to their maximum distances. The candidate vectors are then retrieved from the disk in this order, with the threshold being updated after each access. Any vectors whose minimum distances exceed the current threshold can be discarded. In practice it is unnecessary to retrieve the full vectors of most of the candidates. Checking the candidates is time-consuming as they are spread throughout the file of full-length vectors and each one requires a separate file seek. The seek times for some collections are given in Table 1. Note that very few of the items identified as candidates result in a seek. Nonetheless, the proportion of the search time devoted to these seeks is high. It is also interesting that the proportion of time spent on seeks drops for very long vectors. In the case of ColHistR5, with 320 dimensions, the proportion of total search time spent on seeks was 35.7% whereas for ColHistR1, which is a very similar data set but with 64 dimensions, the proportion of the total search time was 64.1%. The actual times taken are dependent on the hardware used (in this case a DEC Alpha with an unknown hard drive) and the operating system (DEC's OSF/1 UNIX). UNIX makes it difficult to give precise times due to automatic caching and time slicing. The most important parameters are of the disk drive used are the average seek time and the speed of sequential search. The ratio of these parameters largely determines the ratio of the seek time to the index scan time. It is assumed that the same hard drive is used for both the file of complete vectors and for the file of approximate vectors, although this admittedly need not always be the case.

Despite the limitations of the results in Table 1 it is clear that searching may be sped up considerably by removing the requirement that a search be exact. Instead of being used to identify candidates, the file of vector approximations may be used to calculate approximate similarity scores. The final check of candidates can then be omitted, thus saving a large amount of time. Other advantages to allowing incomplete searches are that the file of complete vectors is not required, saving disk memory, and a number of searches may be conducted simultaneously if the CPU time sufficiently exceeds the disk scanning speed. Similarity searches are generally inherently heuristic, so exact solutions are rarely required.

Another possible modification is to set the partitions and approximation value so as to minimise $\mathrm{Var}(S_i - T_i)$ for each dimension i. The algorithm presented in the appendix may be used for this purpose. It is desirable also to allow the dimensions to have different numbers of partitions, as this allows for the more efficient use of memory. If the overall number of bytes for a vector approximation is fixed then the number of partitions may be assigned so as to minimise $\mathrm{Var}(S - T)$ overall. For simplicity it is assumed that the number of partitions for each dimension is a power of 2. This allows each element to be approximated by a few bits that are kept independent of the other dimensions and thus simplifies the encoding and decoding routines considerably. Algorithm 1 is used to allocate the available bits between the dimensions. The allocation process involves adding and removing bits from each dimension to determine the relative effects of each on the measure $\mathrm{Var}(S - T)$. It is necessary to call the program implementing Algorithm 2 many times during this process, so consequently the process is quite slow. It was, for instance, necessary to limit each dimension to a maximum of 8 bits as it took too long to set the parameters if more bits were allocated. It would be desirable to be able to predict the number of bits required for each dimension by analysing the data distribution ahead of time. No adequate method, however, has yet been found.

5 Results

In all of the tests presented here 100 queries were run on collections consisting of 100 000 vectors. In each case the completeness is the average percentage of the 10 items closest to the query that were actually retrieved. Six different collections were used. The *uniform* data set has 50 dimensions, with the data being uniformly distributed in a hypercube. The *normal* data set has 50 dimensions, with the elements in each dimension having a standard normal distribution. The *mixed* data set has 50 dimensions, with each dimension being distributed according to a different type of random distribution, and with the queries having different distributions to the indexed vectors. The *AustLII* data set is a 100-dimensional data set derived from performing dimension reduction on a collection of legal documents. Finally *ColourHistR1* and *ColourHistR5* are two data sets derived using colour histograms

Algorithm 1 Allocate Bits

```
Assign bits evenly amongst elements
while finding improvements do
  for $i = 1$ to $d$ do
    $b$ = current number of bits assigned to dimension $i$
    $increase_i = \mathrm{Var}_b(S_i - T_i) - \mathrm{Var}_{b+1}(S_i - T_i)$
    {$\mathrm{Var}_b$ indicates the variance given $b$ bits. $increase$ and $decrease$ are lists.}
    if $b > 0$ then
      $decrease_i = \mathrm{Var}_{b-1}(S_i - T_i) - \mathrm{Var}_b(S_i - T_i)$
    end if
  end for
  Sort $increase$ into descending order
  Sort $decrease$ into descending order
  for = 1 to $d$ do
    if $increase_i > decrease_i$ then
      decrement the number of bits assigned to the dimension of the decrease
      increment the number of bits assigned to the dimension of the increase
    end if
  end for
end while
```

from the same collection of images taken from the Internet. The *ColourHistR1* data set contains histograms for one region and has 64 dimensions, whereas the *ColourHistR5* data set contains histograms for 5 regions and has 320 dimensions. These two data sets were created by Roger Weber. The similarity measures used in testing were the cosine measure (cos) and the Euclidean distance (l_2).

Table 2 lists the average number of seeks required in searching through the candidates in order to locate all of the 10 vectors most similar to the query vector. The first column contains the figures for when each element is partitioned so that each partition contains an equal number of elements (*data partitioning*) and the second column contains the figures for when the Algorithm 2 is used to optimise the partition boundary points with respect to the measure $\mathrm{Var}(S_i - c_iT_i)$ (*error minimisation*). In most cases optimising $\mathrm{Var}(S_i - c_iT_i)$ produced better results, with considerable better results for the *Normal* and *Mixed* data sets. In the two cases which run against the trend (*Uniform* (l_2) and *AustLII* (cos)), the difference is small. Sampling error is the most likely cause, although another possible reason is that the assumption in Theorem 1 may not be met sufficiently.

It is to be noted that in all cases in which both l_2 and cos were tested, performance was superior for l_2. The reason for this is not clear. One possibility, however, is that when l_2 is used items close to the centre of the query distribution are likely to be retrieved regardless of the query. Using cos is effectively the same as normalising the data and normalising the data reduces the importance of the positioning of items relative to the centre of the query distribution. Thus when cos is used as a measure there are a larger number of vectors that are close to the query and the task of distinguishing between them is more difficult.

Collection	Data Partitioning	Error Minimisation
Uniform (l_2)	13.5	13.8
Uniform (cos)	17.7	15.0
Normal (l_2)	30.1	17.6
Normal (cos)	31.5	19.8
Mixed (l_2)	26.7	18.9
Mixed (cos)	39.6	23.5
AustLII (cos)	30.9	31.2
ColourHistR1 (l_2)	23.4	21.2
ColourHistR5 (l_2)	22.9	22.2

Table 2: The average number of seeks required when using data partitioning and when using error minimisation. Each element has 16 partitions.

Collection	Data Partitioning	Error Minimisation
Uniform (l_2)	82.4%	82.2%
Uniform (cos)	70.6%	82.6%
Normal (l_2)	32.1%	74.1%
Normal (cos)	29.1%	70.0%
Mixed (l_2)	41.5%	74.7%
Mixed (cos)	24.0%	51.2%
AustLII (cos)	31.1%	32.5%
ColourHistR1 (l_2)	33.0%	57.0%
ColourHistR5 (l_2)	42.6%	69.3%

Table 3: The average completeness when using data partitioning and when using error minimisation. Each element has 16 partitions.

Table 3 again compares the partitioning of dimensions into equal parts with the minimisation of $\mathrm{Var}(S_i - c_i T_i)$. In this case an approximate searches were performed, and the figures given are completeness values. In the case of error minimisation the approximate values used to the elements in each partition are real values that lie within the partition that have been determined in accordance with the equations in the appendix. In the case of data partitioning the values midway between the boundaries of each partition was used. The outer boundary of an end-most partition is taken to be the minimum or maximum of the values in the partition. In most cases error minimisation can be seen to provide considerably better performance than data partitioning.

Using different numbers of partitions for each dimension may improve the quality of retrieval. For the sake of simplicity the number of partitions each dimension is divided into is a power of two. This allows each bit in the vector approximation to be dedicated to only one dimension. Table 4 contains the results of tests that allow the number of partitions for each dimension to vary, and these results are compared to the situation in

Collection	Same number of partitions for each dimension	Varying number of partitions
Mixed (l_2)	74.7%	82.7%
Mixed (cos)	51.2%	58.7%
AustLII (cos)	32.5%	35.0%
ColourHistR1 (l_2)	57.0%	82.2%
ColourHistR5 (l_2)	69.3%	91.8%

Table 4: The average completeness when fixing the number of partitions of each dimension to 16 and when allowing the bits to be redistributed.

which the number of partitions is the same for each dimension. Algorithm 1 is used to allocate the number of partitions to for each dimension. $\lceil d/2 \rceil$ bytes are allocated for each vector approximation (where d is the number of dimensions), with a maximum of 256 partitions allowed for any one dimension. The number of partitions is limited due to speed limitations in optimisation algorithm that sets the boundary partition points. In order to save more time during optimisation, when an element was allocated 256 partitions, the optimisation routine was only run once in order to set the partition boundary points. When 128 partitions were allocated only two runs were performed. In all other cases three runs were performed, with the best solution being used in testing. Only those test sets that do not have identically distributed dimensions were tested. As can be seen, allowing different numbers of bits to be allocated to different dimensions results in improved performance in all cases. Allocating the bits to each dimension was a slow process, and a faster technique would be desirable.

The completeness figures presented in the previous tests indicated the average of items amongst the 10 closest to the query that were actually retrieved, where 10 items were retrieved. In the results presented in Table 5 the number of items amongst the closest 10 that appear in the top 10, 20, 30, 40 and 50 are presented. It is evident that if more than 10 items are retrieved then the completeness of the search rapidly increases. In most cases nearly all of the 10 items closest to the query are retrieved within the top 50 items. These results agree with Table 1 in which it can be seen that the average number of seeks required to find all of the closest 10 items is typically between about 15 and 40.

The effect of using different lengths of approximations vectors was tested, and the results are presented in Table 6. The number of partitions for each dimension was allowed to vary. Approximation vectors that had an average of 1, 2, 3 and 4 bits per dimension were tested. As would be expected, using longer vectors increases the completeness in all cases. It appears that in general 4 or more bits per dimension are required. It is to be noted that for *ColourHistR5*, with 320 dimensions, fewer bits are required than for *ColourHistR1*, with 64 dimensions, which is a very similar data set.

Table 7 presents an overview of exhaustive search, the VA-file and the modified VA-file. Note that the number of seeks includes the initial seek necessary to locate the index file.

Collection	10/10	10/20	10/30	10/40	10/50
Uniform (l_2)	82.8%	98.4%	100.0%	100.0%	100.0%
Uniform (cos)	82.8%	98.1%	99.7%	100.0%	100.0%
Normal (l_2)	72.7%	91.6%	97.4%	99.2%	99.7%
Normal (cos)	71.9%	92.6%	97.9%	99.3%	99.7%
Mixed	82.7%	98%	99.9%	100.0%	100.0%
Mixed (cos)	58.7%	89.1%	98.0%	99.2%	99.9%
AustLII (cos)	35.0%	48.7%	55.1%	60.0%	63.9%
ColourHistR1 (l_2)	82.2%	97.5%	99.6%	99.9%	100.0%
ColourHistR5 (l_2)	91.8%	99.5%	99.8%	100.0%	100.0%

Table 5: The percentage of the top 10 items retrieved in the first 10, 20, 30, 40 and 50 items. Each approximation vector was allocated $\lceil d/2 \rceil$ bytes, with dimensions having variable numbers of partitions.

Collection	Average bits per element			
	1	2	3	4
Uniform (l_2)	3.5%	34.8%	62.3%	82.8%
Uniform (cos)	5.2%	32.9%	60.2%	82.8%
Normal (l_2)	1.6%	20.0%	46.8%	72.7%
Normal (cos)	2.7%	18.7%	47.1%	71.9%
Mixed (l_2)	8.7%	44.9%	67.2%	82.7%
Mixed (cos)	16.4%	30.6%	42.2%	58.7%
AustLII (cos)	2.5%	15.0%	27.9%	35.0%
ColourHistR1 (l_2)	39.8%	62.3%	75.8%	82.2%
ColourHistR5 (l_2)	51.3%	72.5%	82.6%	91.8%

Table 6: The average completeness of search given different lengths of approximation vector.

Collection	Exhaustive Search			VA-file			Modified VA-file		
	Index size (10^6 bytes)	Seeks	Comp.	Index size (10^6 bytes)	Seeks	Comp.	Index size (10^6 bytes)	Seeks	Comp.
Uniform (l_2)	20	1	100%	2.8	14.5	100%	2.8	1	82.8%
Uniform (cos)	20	1	100%	2.8	18.7	100%	2.8	1	82.8%
Normal (l_2)	20	1	100%	2.8	31.1	100%	2.8	1	72.7%
Normal (cos)	20	1	100%	2.8	31.5	100%	2.8	1	71.9%
Mixed (l_2)	20	1	100%	2.8	27.7	100%	2.8	1	82.7%
Mixed (cos)	20	1	100%	2.8	40.6	100%	2.8	1	58.7%
ColourHistR1 (l_2)	25.6	1	100%	3.2	24.4	100%	3.6	1	82.2%
ColourHistR5 (l_2)	128	1	100%	16	23.9	100%	16	1	91.8%

Table 7: Exhaustive search, VA-file search (as described by Weber [8]) and the modified VA-file are compared. For the VA-file and modified VA-file an average of around 4 bits per element was used.

It is apparent that the modified VA-file will be appreciably faster than the original VA-file as only one seek is required. On the other hand there is a cost in terms of completeness. If the average completeness of searching is too low then possible solutions include using longer approximation vectors, or retrieving more than the required k items. CPU costs are ignored here as it is assumed that disk speeds are the limiting factor. Whether or not this is so depends upon the architecture of the machine in question.

6 Conclusion

The measure $\mathrm{Var}(S-T)$ should prove to be useful for judging the quality of the approximations of vectors for use in similarity searching for a wide range of data sets. This measure was proven to be optimal under one assumption. It was shown that the assumption comes closer to being met as the dimensionality of the vectors increases, so long as no dimensions dominate in their contribution to the similarity scores. Given two further assumptions regarding the independence of elements in different dimensions, the measure may be used in constructing vector approximations. The VA-file was selected as an example of one application that can take advantage of vector approximations, and it was seen that when the measure was used to construct the vector approximations, improved performance resulted for a number of data sets.

7 Acknowledgements

Thanks to Tom Gedeon, Arthur Ramer and Graham Mann for their supervision of the research presented here, and to Roger Weber who supplied the data sets *ColourHistR1* and *ColourHistR5*.

A The Optimisation Algorithm

A.1 Overview

In order to perform the optimisation the variance between the true similarities (S) and approximate similarities (T) is used. This variance is estimated by calculating the variance for a random sample of n pairs of values of S and T. A value of 100000 was used for n during testing. Let s_i and t_i be the ith readings of S and T, so the variance is:

$$\left[\frac{1}{n}\sum_{i=1}^{n}(s_i - t_i)^2\right] - \left[\frac{1}{n}\sum_{i=1}^{n}(s_i - t_i)\right]^2 \tag{11}$$

Clearly this is slow calculation to make. It is even slower when there are many dimensions as s_i and t_i are dependent on the number of dimensions. Samples smaller than 100 000 could be used, but doing so will of course result in less accurate optimisation.

One way of increasing the speed of optimisation is to make only slight changes to the parameters (these being the boundary points of the partitions and the approximation values), and to update the score by only considering those readings of T that are affected (the values of S do not change). This avoids the need to fully recalculate the variance each time. The increased speed of the calculations also allow many more optimisation steps to be conducted, resulting in improved optimisation.

In order to simplify calculations only one partition boundary point is moved at a time and by a small distance. Moves are determined using a random descent strategy. While in theory moving only one point at a time can result in missing minima in the error surface, this is unlikely to occur in high-dimensional space, where there are many possible directions to take to and from any given point. Changing a number of points by a small amount in turn is almost equivalent to changing all the points simultaneously. In order to remove some potential bias the order of update is chosen randomly. To further simplify calculations all moves are made relative to the rank order of elements, not to their values.

The approximation values used for each partition also need to be optimised. This, however, is not done using random descent. Instead it is possible to set an approximation value optimal in constant time if it is assumed that all of the partition boundary points and other approximations are fixed. Note that setting one approximation value affects the optimal approximations of all of the other partitions and of the placement of the boundary partition points. As boundary partition points are only moved small amounts the approximations generally only vary by small amounts. In order to give the system extra stability the approximation values may also be restricted to be within the partitions they represent. As it is very quick to set an approximation value, all of the approximation values may set one or more times each time a boundary partition point is moved.

The Algorithm 2 is used to update the parameters. To start with the algorithm tries to move the partition boundary points by moderately-sized steps, and when no improvements can be found the size of the steps is reduced. The approximation values are updated frequently to take advantage of the fact that this update is a quick operation.

Algorithm 2 Optimisation algorithm for setting partition boundary points and approximations using incremental update of statistics

```
Set all partition boundary points using data partitioning
Set all approximations to be mid-way values between boundary points
Derive all statistics
for all Partition in random order (repeat 3 times) do
   Set approximation value
   Update statistics (1)
end for
$commonRadius = 1000$ {For a sample size of 100 000}
while $commonRadius > 0$ do
   $radius = commonRadius$
   for all boundary points and both directions in random order do
      while no improvement in $\mathrm{Var}(S - T)$ and $radius > 0$ do
         Calculate prospective $\mathrm{Var}(S - T)$ for a change in the boundary point by $radius$
         in the given direction
         if improvement then
            Set the new partition boundary point
            Update statistics (2)
            Set approximation value of partition to the left of boundary point
            Update statistics (1)
            Set approximation value of partition to the right of boundary point
            Update statistics (1)
            for all partitions in random order do
               Set approximation value
               Update statistics (1)
            end for
         end if
         $radius = radius/2$
      end while
   end for
   if Number of moved boundary partition points $< 2$ (or 1 for 4 partitions or fewer)
   then
      $commonRadius = commonRadius/2$
   end if
end while
```

A.2 Finding the Best Approximations

If it is assumed that all the partition boundary points and all the approximation values bar one are constant then it is possible to calculate the best possible value for the free approximation value in constant time. Let $A = S - T$ where S is the part-similarity and T is its approximation. Let x be the approximation value that is being set and let x be in interval i. Let I represent the fact that X is in interval i, and $\bar{I}$ represent the fact that it is not in the interval. Let an I subscript to a random variable (e.g. T_I) indicate that the X value the random variable is dependent upon is in partition I. Then:

$$\begin{aligned}
\mathrm{Var}A &= \mathrm{E}(A^2) - (\mathrm{E}A)^2 \\
&= \Pr I\ \mathrm{E}A_I^2 + \Pr \bar{I}\ \mathrm{E}A_{\bar{I}}^2 - [\Pr I\ \mathrm{E}A_I + \Pr \bar{I}\ \mathrm{E}A_{\bar{I}}]^2 \\
&= \Pr I\ \mathrm{E}A_I^2 + \Pr \bar{I}\ \mathrm{E}A_{\bar{I}}^2 - (\Pr I)^2(\mathrm{E}A_I)^2 \\
&\quad -2\Pr I\ \Pr \bar{I}\ \mathrm{E}A_I\ \mathrm{E}A_{\bar{I}} - (\Pr \bar{I})^2(\mathrm{E}A_{\bar{I}})^2 \\
&= \Pr I[\mathrm{E}A_I^2 - \Pr I(\mathrm{E}A_I)^2 - 2\Pr \bar{I}\ \mathrm{E}A_I\ \mathrm{E}A_{\bar{I}}] \\
&\quad + \Pr \bar{I}[\mathrm{E}A_{\bar{I}}^2 - \Pr \bar{I}(\mathrm{E}A_{\bar{I}})^2] \qquad (12)
\end{aligned}$$

Note that this expression is true regardless of the similarity measure.

In order to find the optimal value of x the expression $\frac{\partial \mathrm{Var}A}{\partial x}$ will be considered. In Equation 12 only the terms $\mathrm{E}(A_I)$, $\mathrm{E}(A_I^2)$ and $\mathrm{E}(A_I)^2$ are dependent upon x. The partial derivates of these terms are:

$$\begin{aligned}
\frac{\partial \mathrm{E}A_I}{\partial x} &= \frac{\partial \mathrm{E}(S_I - T_I)}{\partial x} \\
&= -\mathrm{E}\frac{\partial T_I}{\partial x} \qquad (13)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathrm{E}A_I^2}{\partial x} &= \frac{\partial \mathrm{E}(S_I^2 - 2S_IT_I + T_I^2)}{\partial x} \\
&= 2\left[\mathrm{E}\left(T_I\frac{\partial T_I}{\partial x}\right) - \mathrm{E}\left(S_I\frac{\partial T_I}{\partial x}\right)\right] \qquad (14)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial (\mathrm{E}A_I)^2}{\partial x} &= 2\mathrm{E}A_I\frac{\partial \mathrm{E}A_I}{\partial x} \\
&= 2[\mathrm{E}T_I - \mathrm{E}S_I]\mathrm{E}\frac{\partial T_I}{\partial x} \qquad (15)
\end{aligned}$$

If the Euclidean distance is used to measure dissimilarity then the part-similarity for one dimension is $S = (X - Y)^2$ and the part-approximate similarity is $T = (x - Y)^2$. Note that x is in interval I. The above partial derivatives are thus:

$$\begin{aligned}
\frac{\partial \mathrm{E}A_I}{\partial x} &= -\mathrm{E}\frac{\partial T_I}{\partial x} \\
&= 2(\mathrm{E}Y - x) \qquad (16)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathrm{E}A_I^2}{\partial x} &= 2\left[\mathrm{E}\left(T_I\frac{\partial T_I}{\partial x}\right) - \mathrm{E}\left(S_I\frac{\partial T_I}{\partial x}\right)\right] \\
&= 4[\mathrm{E}(x-Y)^3 - \mathrm{E}(x-Y)S_I] \\
&= 4(x^3 - 3x^2\mathrm{E}Y_I + 3x\mathrm{E}Y_I^2 - \mathrm{E}Y_I^3 - x\mathrm{E}S_I + \mathrm{E}S_IY_I) \qquad (17)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial (\mathrm{E}A_I)^2}{\partial x} &= 2[\mathrm{E}T_I - \mathrm{E}S_I]\mathrm{E}\frac{\partial T_I}{\partial x} \\
&= 4(x^2 - 2x\mathrm{E}Y_I + \mathrm{E}Y_I^2 - \mathrm{E}S_I)(x - \mathrm{E}Y_I) \\
&= 4\{x^3 - 3x^2\mathrm{E}Y_I + [2(\mathrm{E}Y_I)^2 - \mathrm{E}S_I + \mathrm{E}Y_I^2]x + \\
&\quad \mathrm{E}Y_I(\mathrm{E}S_I - \mathrm{E}Y_I^2)\} \qquad (18)
\end{aligned}$$

If Equations 16, 17 and 18 are substituted into Equation 12 then the partial differential $\frac{\partial \mathrm{Var}A}{\partial x}$ is:

$$\begin{aligned}
\frac{\partial \mathrm{Var}A}{\partial x} &= 4\Pr I\{x^3 - 3x^2\mathrm{E}Y_I + 3x\mathrm{E}Y_I^2 - \mathrm{E}Y_I^3 - x\mathrm{E}S_I + \mathrm{E}S_IY_I - \Pr I\{x^3 - \\
&\quad 3x^2\mathrm{E}Y_I + [2(\mathrm{E}Y_I)^2 - \mathrm{E}S_I + \mathrm{E}Y_I^2]x + \mathrm{E}Y_I(\mathrm{E}S_I - \mathrm{E}Y_I^2)\} - \\
&\quad \Pr \bar{I}\ \mathrm{E}(A_I \mid \bar{I})(\mathrm{E}Y_I - x)\} \qquad (19)
\end{aligned}$$

Rearranging this expression and fixing it equal to 0 gives:

$$\begin{aligned}
&\Pr \bar{I}x^3 - 3\mathrm{E}Y_I \Pr \bar{I}x^2 + [-\mathrm{E}S_I \Pr \bar{I} + 3\mathrm{E}Y_I^2 - 2\Pr I(\mathrm{E}Y_I)^2 - \\
&\Pr I\mathrm{E}Y_I^2 + \Pr \bar{I}\mathrm{E}A_{\bar{I}}]x + \Big[-\mathrm{E}{Y_I}^3 + \mathrm{E}S_IY_I - \Pr I\ \mathrm{E}Y_I\ \mathrm{E}S_I + \\
&\qquad \Pr I\ \mathrm{E}Y_I\ \mathrm{E}{Y_I}^2 - \Pr \bar{I}\ \mathrm{E}A_{\bar{I}}\mathrm{E}Y_I\Big] = 0 \qquad (20)
\end{aligned}$$

This is a cubic polynomial with x as the variable. It may be solved using standard techniques to find the turning points. The optimal value of x must lie at one of the turning points. If in addition the approximation value x is constrained to lie within the domain of partition I, then the boundaries of I must also be tested. In order to solve this equation it is necessary to store and maintain each of the statistics used (i.e. $\mathrm{E}Y_I$,$\mathrm{E}Y_I^2$,$\mathrm{E}Y_I^3$,$\Pr \bar{I}$,$\mathrm{E}S_I$ and $\mathrm{E}S_IY_I$). These statistics need to be updated each time a partition boundary point is moved or an approximation value that belongs to a partition is changed. This involves many update operations, most of which are fairly straightforward. The more complex calculations are described in the next section.

A.3 Updating Statistics after Changing an Approximation Value

Let x be an approximation value that is changed to x'. x is in partition I. Only those elements whose X value is in I is affected. Consider updating Var$A$ after changing one approximation value. From Equation 12 it may be seen that

$$\begin{aligned}
\mathrm{Var}A' - \mathrm{Var}A &= \Pr I\{(\mathrm{E}{A_I'}^2 - \mathrm{E}A_I^2) - \Pr I[(\mathrm{E}A_I')^2 - (\mathrm{E}A_I)^2] \\
&\quad -2\Pr \bar{I}\ \mathrm{E}A_{\bar{I}}(\mathrm{E}A_I' - \mathrm{E}A_I)\} \\
&= \Pr I\{(\mathrm{E}{A_I'}^2 - \mathrm{E}A_I^2) - \Pr I[(\mathrm{E}A_I')^2 - (\mathrm{E}A_I)^2] \\
&\quad +2\Pr \bar{I}\ \mathrm{E}A_{\bar{I}}(\mathrm{E}T_I' - \mathrm{E}T_I)\} \qquad (21)
\end{aligned}$$

Furthermore:

$$\begin{aligned}\mathrm{E}{A'_I}^2 - \mathrm{E}A_I^2 &= [\mathrm{E}(S_I - T'_I)]^2 - [\mathrm{E}(S_I - T_I)]^2 \\ &= \mathrm{E}{T'_I}^2 - \mathrm{E}T_I^2 - 2(\mathrm{E}S_I T'_I - \mathrm{E}S_I T_I) \qquad (22)\end{aligned}$$

$$\begin{aligned}(\mathrm{E}A'_I)^2 - (\mathrm{E}A_I)^2 &= (\mathrm{E}S_I - T'_I)^2 - (\mathrm{E}S_I - T_I)^2 \\ &= (\mathrm{E}T'_I)^2 - (\mathrm{E}T_I)^2 - 2\mathrm{E}S_I(\mathrm{E}T'_I - \mathrm{E}T_I) \qquad (23)\end{aligned}$$

At this point it is necessary to specify the similarity measure that is used. If the square of the Euclidean distance is used then:

$$\mathrm{E}T'_I - \mathrm{E}T_I = {x'}^2 - x^2 - 2(x' - x)\mathrm{E}Y_I \qquad (24)$$

$$\mathrm{E}S_I T'_I - \mathrm{E}S_I T_I = ({x'}^2 - x^2)\mathrm{E}S_I - 2(x' - x)\mathrm{E}S_I Y_I \qquad (25)$$

$$\begin{aligned}(\mathrm{E}T'_I)^2 - (\mathrm{E}T_I)^2 &= [\mathrm{E}(x' - Y_I)^2]^2 - [\mathrm{E}(x - Y_I)^2]^2 \\ &= ({x'}^4 - x^4) - 4({x'}^3 - x^3)\mathrm{E}Y_I + 2({x'}^2 - x^2)\mathrm{E}Y_I^2 \\ &\quad +4({x'}^2 - x^2)(\mathrm{E}Y_I)^2 - 4(x' - x)\mathrm{E}Y_I\mathrm{E}Y_I^2 \qquad (26)\end{aligned}$$

$$\begin{aligned}\mathrm{E}{T'_I}^2 - \mathrm{E}T_I^2 &= \mathrm{E}(x' - Y_I)^4 - \mathrm{E}(x - Y_I)^4 \\ &= ({x'}^4 - x^4) - 4\mathrm{E}Y_I({x'}^3 - x^3) + 6\mathrm{E}Y_I^2({x'}^2 - x^2) \\ &\quad -4\mathrm{E}Y_I^3(x' - x) \qquad (27)\end{aligned}$$

The simpler working has been omitted. Several of the statistics used above also need to be updated when an approximation value is changed or when a partition boundary point is moved. These updates are less complicated than the ones presented and so have been omitted. If a measure other than the Euclidean distance is used then similar calculations to the ones above need to be made if the full calculation of $\mathrm{Var}(S_i - T_i)$ is to be avoided after each change of a parameter. Note that for the cosine measure it is possible to simply normalise the original vectors and to use the Euclidean distance as the measure of dissimilarity.

References

[1] S. Berchtold, , and H.-P. Kriegel. S3: Similarity search in CAD database systems. In J. Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, volume 26(2) of *SIGMOD Record*, pages 564–567. ACM Press, 1997.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, Sept. 1990.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, volume 23(2) of *SIGMOD Record*, pages 419–429. ACM Press, 1994.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and query content: The QBIC system. *Computer*, 28(9):23–32, 1995.

[5] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, 1992.

[6] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman & Hall / CRC, Boca Raton, Florida, 1992.

[7] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Winter 1991.

[8] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th VLDB Conference*, New York, USA, 1998.