Fast Mining of Massive Tabular Data via Approximate Distance Computations

Graham Cormode
Dept of Computer Science
University of Warwick, UK
grahamc@dcs.warwick.ac.uk

Piotr Indyk
Computer Science Lab
MIT, Cambridge MA, USA
indyk@theory.lcs.mit.edu

Nick Koudas S. Muthukrishnan
AT&T Research
Florham Park NJ, USA
koudas,muthu@research.att.com

Abstract

Tabular data abound in many data stores: traditional relational databases store tables, and new applications also generate massive tabular datasets. For example, consider the geographic distribution of cell phone traffic at different base stations across the country or the evolution of traffic at Internet routers over time . Detecting similarity patterns in such data sets (e.g., which geographic regions have similar cell phone usage distribution, which IP subnet traffic distributions over time intervals are similar, etc) is of great importance. Identification of such patterns poses many conceptual challenges (what is a suitable similarity distance function for two "regions") as well as technical challenges (how to perform similarity computations efficiently as massive tables get accumulated over time) that we address.

We present methods for determining similar regions in massive tabular data. Our methods are for computing the "distance" between any two subregions of a tabular data: they are approximate, but highly accurate as we prove mathematically, and they are fast, running in time nearly linear in the table size. Our methods are general since these distance computations can be applied to any mining or similarity algorithms that use L_p norms. A novelty of our distance computation procedures is that they work for any L_p norms — not only the traditional $p = 2$ or $p = 1$, but for all $p \leq 2$; the choice of p, say fractional *p, provides an interesting alternative similarity behavior!*

We use our algorithms in a detailed experimental study of the clustering patterns in real tabular data obtained from one of AT&T's data stores and show that our methods are substantially faster than straightforward methods while remaining highly accurate, and able to detect interesting patterns by varying the value of p.

1. Introduction

Tabular data abound in data stores. Traditional relations in database systems are tables. New applications also generate massive tabular datasets — consider the application of cellular telephone networks. These networks have base stations (aka cell towers) geographically distributed over the country, with each base station being responsible for handling the calls to and from a specific geographic region. This data may be represented by a table indexed by the latitude and longitude of the base station and storing the call volume for a period of time such as an hour.

As another application, consider the representation of the Internet traffic between IP hosts over time. For example, one may visualize a table indexed by destination IP host and discretized time representing the number of bytes of data forwarded at a router to the particular destination for each time period (since the current generation of IP routers route traffic based on destination IP address only, this is valuable information about network congestion and performance that IP routers routinely store and dump).

In these network data management scenarios and others, massive tables are generated routinely (see [5, 3, 9] for other examples). While some of the data may be warehoused in traditional relational databases, this is seldom true in case of emerging applications. Here, tabular data is stored and processed in proprietary formats such as compressed flat files [6]. Thus tabular data is emerging as a data format of independent interest and support within large scale applications [4, 5, 9, 10].

Of great interest in tabular data management is the task of *mining* them for interesting patterns. For example, in the examples above, the "knowledge" from tabular data analysis drives key network-management tasks such as traffic engineering and flexible capacity planning [8]. In the cellular calls case, one may be interested in finding geographic regions where the call distribution is similar, for example, can one analyze the call volume patterns to separate dense urban areas, suburban areas, commuter regions, and so on?

Thus many creative mining questions arise with tabular data. Our informal goal of mining tabular data can be instantiated in one of many standard ways: as clustering, as association rule mining or in other ways. These tasks involve comparing large (possibly arbitrary) portions of the table with each other (possibly many times). Two fundamental challenges emerge in addressing these tasks.

First, *what is a suitable notion of similarity between two regions?* This is a conceptual question. Two natural distance functions are commonly used to measure the distance between two vectors or matrices (table portions). The L_1, or Manhattan, metric is the sum of the absolute differences between corresponding entries; the L_2, or Euclidean, metric is (the square root of) the sum of squares of differences. These distances are traditionally used in image recognition, information retrieval and data mining. Depending on applications, one may consider dilation, scaling and other operations on vectors before computing the L_1 or L_2 norms.

Second, *how to compute these similarity distances efficiently?* Given two portions of the table, L_1 or L_2 distances can be computed in time linear in their size and one can not do better. So this question has to be refined further. In mining tasks such as clustering, table portions will be "compared" against many others, possibly multiple times. If we consider the problem of computing all such distances, one needs to compute a large number of them (up to $O(n^4)$ table portions for a $n \times n$ table) and computing each is expensive ($O(n^2)$ time each in the worst case). A clustering algorithm may need any subset of these computations done and computing all such distances either on demand or in one shot for all possible subtables is expensive in time and/or space.

The problem of efficient similarity computation between table portions is a particularly irksome challenge since tabular data is massive, generated at the rate of several terabytes a month in most applications [5, 4, 3, 9]. Tabular data becomes very large very quickly, since it grows with the product of its defining characteristics: an extra base station will take thousands of readings a day, and an extra day's data adds hundreds of thousands of readings. As these databases grow larger, not only does the size of data stored increase, but also the size of the interesting table portions increases. With data storage capacities easily in the terabyte range, any subregions of interest to be "compared" (eg cell call data for the Los Angeles versus San Francisco areas) can themselves be megabytes or even gigabytes in size. This means that previous assumptions that is commonly made in mining — that comparing two objects is a basic unit of computation — no longer hold. Instead, the metric by which algorithms are judged is no longer just the number of comparisons used, but rather the number of comparisons multiplied by the cost of a comparison.

In this paper, we address both the conceptual and technical challenges above in mining massive tabular data. Our contributions are as follows.

1. We study L_p norms for p which differ from the classical $p = 1$ or $p = 2$; in particular, we study non-integral values of p, $0 < p \leq 2$. Similarity of vectors or matrices based on such non-integral p reveal novelties in mining as our experimental studies show.

2. We present time and space efficient methods for quickly estimating the L_p distance between any two table portions. These are very fast to precompute and to process: they are constant in size irrespective of the size of the subtables, and provide guaranteed very accurate approximations. For p values of our interest, this involves developing "sketch" functions for computing the L_p distances using the notion of stable distributions from probability theory.

3. We experimentally study an interesting tabular data set from an AT&T data store. It is a time series evolution of the call volumes from collection stations sorted geographically in some linear fashion. We focus on a basic mining task, namely, clustering the portions of the tabular data. We apply known clustering algorithms but employ our distance computation methods and show empirically that L_p for non-integral values of p between 0 and 2 yields interesting similarity metric and that the resulting clustering algorithms are substantially faster than the straightforward methods.

Clustering is a good example of a mining task affected by growing table sizes. A typical clustering algorithm "compares" each "object" with others many times over, where each comparison involves examining the distance of one object to each of a number of others. Many good algorithms have been described in the literature already such as k-means [14], CLARANS [16], BIRCH [21, 10], DBSCAN [7] and CURE [11]. These focus on limiting the number of comparisons. As we described earlier, when objects are large, the cost of comparisons (not just the number of comparisons) affects performance significantly. Orthogonal to the efforts mentioned above, we have focused on reducing the cost of each comparison. Since what is important in clustering is often not the exact distances between objects, but rather which of a set of objects another object is closest to, and known clustering algorithms are in their nature approximate and use randomness, we are able to show that using randomness and approximation in distance estimation does not sacrifice clustering quality by any appreciable amount.

2. Related Work

Tabular data has previously been the focus for telecommunications data management problems — see, for example, [4]. Mining tables has been studied previously in the database community, chiefly as association rule mining on multiple numerical attributes [19, 15, 18]. While time series mining has been studied extensively for both single and multiple time series data, we are not aware of prior work on the tabular time series data mining we study here, nor the application of approximate distance computations to clustering that we study here. Vectors and matrices are routinely compared based on their L_p distances for $p = 1, 2$ or ∞, and more unusually other integral L_p distances [20]. Only recently has the similarity behaviour of L_p distances with non-integral p been examined for $0 < p \leq 2$, as we do here and independently in [1].

Our work involves dimensionality reduction techniques, which has been well explored in databases [2]. Previous dimensionality reduction techniques have used the first few components of the Discrete Fourier Transform, or other linear transformations (Discrete Cosine or Wavelet Transforms). This is because the L_2 distance between a pair of DFTs is the same as the L_2 distance between the original data. However, the step of approximating the distance by using only the first components is a heuristic, based on the observation that for many sequences most of the energy of the signal is concentrated in these components. Although these techniques often work well for the Euclidean (L_2) distance, they do not work for other L_p distances, including the important L_1 distance. This is because there is no equivalent result relating the L_1 distance of transformed sequences to that of the original sequences.

Furthermore, this approach does not provide mechanisms for combining sketches the way we can do for efficient distance computation. As a result, in a previous paper [13], we adopted dimensionality reduction using projections when dealing with time series data. In this paper, we extend those results to tabular data. More importantly, we extend the generally applicable dimensionality reduction techniques to the interesting direction of arbitrary L_p norms (from just L_2 in that paper) for non-integral p which will be of interest in many other applications of approximate distance computations.

3. Sketches: Definition and Computation

As mentioned earlier, our methods depend on determining "sketches" of objects. Sketches will be very small in size; two objects can be compared based on their sketches accurately and quickly. For different distances, there will be other sketch functions; they are defined by their very strong guarantees of quality. In this section we define the sketching functions for L_p norms.

3.1. Preliminaries

We shall make use of one dimensional vectors and two dimensional matrices. A one dimensional vector is written as $\vec{x}$, and has some number n of entries. The tabular data that we shall be manipulating can be thought of as a vector of vectors of the same length, or equivalently, as a two dimensional matrix. To distinguish these matrices from one dimensional vectors, we shall write these in upper case as $\vec{X}$. The ith component of a vector, $\vec{x_i}$ is a scalar; the ith component of a matrix, $\vec{X_i}$ is a vector; and $\vec{X_{i,j}}$ is the scalar that is the jth component of the ith component of $\vec{X}$.

The L_p distance between two vectors, $\vec{x}$ and $\vec{y}$, both of length n is $(\sum_{i=1}^{n} |\vec{x_i} - \vec{y_i}|^p)^{1/p}$. The L_p distance between $\vec{x}$ and $\vec{y}$ is written $||\vec{x} - \vec{y}||_p$. We extend this notation to tabular data, and so

$$||\vec{X} - \vec{Y}||_p = (\sum_i \sum_j |\vec{X}_{i,j} - \vec{Y}_{i,j}|^p)^{1/p}$$

The dot product of two vectors, $\vec{x}$ and $\vec{y}$ is $\sum_i \vec{x_i}\vec{y_i}$, and is written $\vec{x} \cdot \vec{y}$. Similarly, between two equal size matrices the dot product, $\vec{X} \cdot \vec{Y}$ is $\sum_{i,j} \vec{X}_{i,j}\vec{Y}_{i,j}$. For any d-length vector $\vec{x}$ let $\mathrm{median}(\vec{x})$ denote the median of the sequence $\vec{x_1}, \ldots, \vec{x_d}$.

3.2. Sketch Definition

A *stable distribution* is a statistical distribution, X with a parameter α in the range $(0, 2]$. It has the property that if $X_1, X_2, \ldots X_n$ are distributed identically to X then $a_1X_1 + a_2X_2 + \ldots + a_nX_n$ is distributed as $||(a_1, a_2, \ldots, a_n)||_\alpha\ X$. Several well-known distributions are known to be stable. The Gaussian distribution is stable with $\alpha = 2$; the Cauchy distribution is stable with $\alpha = 1$; and the Lévy distribution is stable with $\alpha = \frac{1}{2}$. For all other permitted values of α, stable distributions can be simulated by using appropriate transformations from uniform distributions. Further details on stable distributions and computing values taken from stable distributions can be found in [17].

Sketch for L_1 distance. The key property of sketches is the strong accuracy guarantee that they provide: if the exact distance is ℓ, then our approximation $\hat{\ell}$ is within an ϵ fraction of ℓ with probability $1 - \delta$. That is, $\Pr[|\ell - \hat{\ell}| > \epsilon\ell] \le \delta$. Let us first focus on L_1. We shall implement techniques outlined in [12]. Given a vector $\vec{x}$, we define a sketch vector of $\vec{x}$, $\vec{s}^{\,1}(\vec{x})$. This vector is of size $k = c\frac{\log 1/\delta}{\epsilon^2}$, for some constant c, to be established later. We pick k random vectors $\vec{r}[1] \ldots \vec{r}[k]$ where each $\vec{r}[i] = (\vec{r}[i]_1, \ldots, \vec{r}[i]_n)$ is made by drawing each component from the Cauchy distribution. The L_1 sketch is defined by $\vec{s}_i^{\,1}(\vec{x}) = \vec{x} \cdot \vec{r}[i]$ — that is, as the dot product of $\vec{x}$ with each of the k random vectors $\vec{r}[i]$.

Theorem 1 *With probability* $1 - \delta$, $(1 - \epsilon)||\vec{x} - \vec{y}||_1 \le \mathrm{median}(|\vec{s}^{\,1}(x) - \vec{s}^{\,1}(y)|) \le (1 + \epsilon)||\vec{x} - \vec{y}||_1$

Proof: The proof of this hinges on the fact that each element $r[i]$ of $\vec{s}^{\,1}(x)$ are drawn from a stable distribution, and so $\vec{r}[i] \cdot \vec{x} - \vec{r}[i] \cdot \vec{y} = \vec{r}[i] \cdot (\vec{x} - \vec{y})$ has same distribution as $||\vec{x} - \vec{y}||_1 X$, where X has Cauchy distribution. Let $L = |\vec{r}[i](\vec{x} - \vec{y})|$. It was shown in [12] that the median of L is equal to $||\vec{x} - \vec{y}||_1$ (recall that if M is the median of a random variable L then it must satisfy $\Pr[L \ge M] = \frac{1}{2}$.) We next apply standard results from statistical sampling: if we find the median of $O(1/\epsilon^2)$ independent repetitions of this process, then we guarantee that the approximation is within a factor of $1 \pm \epsilon$ of the true answer with some constant probability. We can then extend this by carrying out a further $O(\log 1/\delta)$ repetitions, and taking the median of these: this process amplifies the constant probability of success to $1 - \delta$. Thus, the repetition of the sampling ensures that the claimed probability bounds are met. $\square$

Sketch for L_p for any $0 \le p < 2$. Next let us focus on non-integral values of p in L_p. If we replace the variables from the Cauchy distribution with a stable variable for which $\alpha = p$ then a similar result follows for the L_p distance between vectors. In particular, we have the new theorem

Theorem 2 *For any $p \in (0, 2]$ there exists a scaling factor $B(p)$ so for all vectors $\vec{x}, \vec{y}$, with probability $1 - \delta$ we have* $(1-\epsilon)||\vec{x}-\vec{y}||_p \leq B(p)\,\mathrm{median}(|\vec{s^p}(x)-\vec{s^p}(y)|) \leq (1+\epsilon)||\vec{x}-\vec{y}||_p$

The need for a scaling factor follows from the fact that the median of p-stable distributions is only 1 when $p = 1$ or $p = 2$. We do not need to find $B(p)$, since for clustering purposes, it is only the relative distance from a given point that is needed (which point is closest). The theory has so far spoken in terms of vectors. However, it is conceptually simple to shift this theory from one-dimensional vectors to two-dimensional matrices, thanks to the nature of the L_p norms: we can think of any matrix as being represented by a vector that is linearized in some consistent way.

3.3. Computing Sketches

For a given vector or matrix, its sketch is a short real-valued vector as defined previously. According to the theory outlined above, each entry in the sketch is the dot-product of the object (vector or matrix) with a number of randomly created objects (as specified above, using values from stable distributions), necessarily of the same size. We are interested in computing sketches for *all* subtables of some large set of tabular data. We describe this in two steps:

- We construct the sketch for all subtables of a fixed dimension (size) very fast using Fast Fourier Transform.
- We choose a small, canonical set of dimensions and construct sketches for all subtables of that dimension. The pool of all such sketches is used to quickly compute the sketch of a subtable of any arbitrary dimension.

Computing sketches for all subtables of a fixed size. We first focus on computing sketches with a *fixed* subtable size, and computing the sketch for all subtables of that size. The definitions above immediately translate into algorithms. A pre-processing phase can compute the necessary k different $\vec{R}[i]$ matrices from an appropriate stable distribution. Where we have tabular data, any subtable of fixed size defines a matrix that we would like to make a sketch of. Each sketch can then be computed by finding the dot product of each of the random matrices with all sub-rectangles of the tabular data. In our scenario, when we are dealing with tabular data, we could consider each sub-rectangle of fixed size in turn, and compute the sketches individually. Suppose that the size of the input data is N and the subrectangle has size M, then the total cost of this approach is $O(kMN)$ computations. This is because for each of the N locations in the data table we must multiply k different random matrices with a subtable of size M. This can be improved, since the basic computation is simply the convolution of two matrices, which can be made more quickly using two-dimensional Fast Fourier Transforms (FFT).

Theorem 3 *All sketches of fixed size sub-rectangles of the data can be made efficiently using the Fast Fourier Transformation taking time $O(kN \log M)$.*

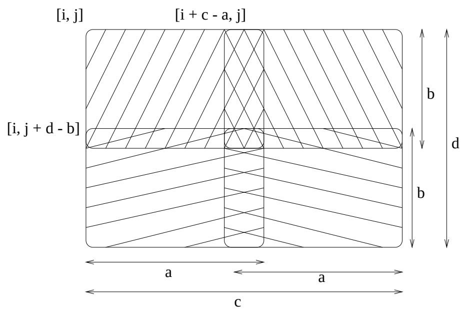


Figure 1. Compound sketches can be formed from four sketches that represent the area required.

This follows because we have to compute the dot product of each of the random matrices with every sub-rectangle of the data. This can be done more efficiently in the Fourier Doman using convolutions.

Canonical sizes and combining sketches. Next we consider computing sketches for many different sizes. Once we have all sketches for sub-rectangles of a particular size, we can combine these to make a sketch for a rectangle up to twice the size in either dimension. Suppose that four independent sets of sketches, $\vec{s}^{\,p}, \vec{t}^{\,p}, \vec{u}^{\,p}, \vec{v}^{\,p}$, have been computed for sub-rectangles of dimensions $a \times b$. Provided $a \leq c \leq 2a$ and $b \leq d \leq 2b$, a sketch can be made for the sub-rectangle of size $c \times d$. Let the data be a large table, Z, so the sketch, $\vec{s^p}(Z[i, j])$ will cover the sub-rectangle from $Z[i, j]$ to $Z[i + a, j + b]$.

Definition 4 *A* compound sketch, $\vec{s'}^p$ *can be formed as follows:* $\vec{s'}^p_i(Z[i, j]) = \vec{s}^{\,p}(Z[i, j]) + \vec{t}^{\,p}(Z[i + c - a, j] + \vec{u}^{\,p}(Z[i, j + d - b]) + \vec{v}^{\,p}(Z[i + c - a, j + d - b])$

In effect, the new sketch is formed by tiling the required sub-rectangle with sketches of overlapping rectangles. This is shown in Figure 1. From this definition, we state the theorem that shows that this compound sketch still has the desirable properties of a sketch.

Theorem 5 *For two compound sketches created as above, with probability $1 - \delta'$, we have* $||\vec{X} - \vec{Y}||_p(1 - \epsilon) \leq B(p)\ \mathrm{median}(|\vec{s'}^p(\vec{X}) - \vec{s'}^p(\vec{Y})|) \leq 4(1 + \epsilon)||\vec{X} - \vec{Y}||_p$

Because of this theorem, then by summing four sketches component-wise we can make reasonable sketches from those for smaller overlapping subtables. Therefore we choose a *canonical* collection of sizes: we fix dyadic sizes $2^i \times 2^j$ for various i and j and compute sketches for all matrices of size $2^i \times 2^j$ using Theorem 3. Following that we can compute the sketches for any $c \times d$ sized subtable using these sketches and using Theorem 5. Therefore, we conclude

Theorem 6 *Given any tabular data of size $N = n \times n$, in time $O(kN \log^3 N)$ we can compute all the $O(\log^2 N)$ sets* *of sketches corresponding to the canonical sizes. Following that, the sketch for any subtable can be computed in time $O(k)$. For L_p distances of our interest, $k = O(\frac{\log(1/\delta)}{\epsilon^2})$, and each sketch is an $4 + \epsilon$ approximation with probability at least $1 - \delta$.*

4. Experimental Results

We prototyped our algorithms and in this section we report the results of a detailed experimental evaluation.

4.1. Accuracy Measures

Let the sketched L_p distance between two matrices $\vec{X}$ and $\vec{Y}$ be denoted by $\widetilde{||\vec{X} - \vec{Y}||_p}$. We use the following measures to assess sketching accuracy:

Definition 7 *The* cumulative correctness *of a set of k separate experiments between a set of matrices $\vec{X}_i$ and $\vec{Y}_i$ is*

$$\frac{\sum_{i=1}^{k} \widetilde{||\vec{X}_i - \vec{Y}_i||_p}}{\sum_{i=1}^{k} ||\vec{X}_i - \vec{Y}_i||_p}$$

This measure gives an idea of, in the long run, how accurate the sketches are.

Definition 8 *The* average correctness *of a set of k experiments is* $1 - \frac{1}{k}\sum_{i=1}^{k}\left| 1 - \frac{\widetilde{||\vec{X}_i - \vec{Y}_i||_p}}{||\vec{X}_i - \vec{Y}_i||_p} \right|$

The above two measures are good for when we are testing the sketches as estimators of the actual distance. In our clustering application however, what is more important is the correctness of pairwise comparisons: testing whether some object $\vec{x}$ is closer to $\vec{y}$ or to $\vec{z}$.

Definition 9 *The* pairwise comparison correctness *of k experiments between three sets of matrices, X_i, Y_i, Z_i is* [1]

$$\frac{\sum_{i=1}^{k} \mathrm{xor}(||\vec{X}_i - \vec{Y}_i||_p < ||\vec{X}_i - \vec{Z}_i||_p, \widetilde{||\vec{X}_i - \vec{Y}_i||_p} > \widetilde{||\vec{X}_i - \vec{Z}_i||_p})}{k}$$

To assess the quality of a clustering obtained using exact methods against one obtained using sketches, we want to find ways of comparing the two k-clusterings. A commonly used construct in comparing clusterings is the *confusion matrix*. Each object will be associated with two clusters: one for the exact comparison case, and one for the approximate case. A $k \times k$ matrix records how many objects are placed in each possible classification.

Definition 10 *The* confusion matrix agreement *between two k-clusterings requires the use of a confusion matrix on the clusterings that records the number of tiles in each clustering that are allocated to the same cluster. If* $\mathrm{confusion}(i, j)$ *is a function that reports how many times an item is classified as being in cluster i in one clustering and in cluster j in another clustering, then the agreement is simply:*

$$\frac{\sum_{i=1}^{k} \mathrm{confusion}(i,i)}{\sum_{i=1}^{k}\sum_{j=1}^{k} \mathrm{confusion}(i,j)}$$

However, it is quite possible that two clusterings with very different allocation of tiles to clusters could still be a good quality clustering. To remedy this, for any clustering, we can compute the total distance of each element in the clustering from the center of its cluster. Any clustering algorithm should attempt to minimize this amount. We can then evaluate this distance for the clustering obtained using sketches as a percentage of the clustering obtained by exact distance computations. Let $spread_{exact}(i)$ be the spread of the ith cluster following a clustering with exact comparisons, and $spread_{sketch}(i)$ be similarly defined when approximate comparisons are used.

Definition 11 *The* quality of sketched clustering *with k clusters is defined as* $\frac{\sum_{i=1}^{k} spread_{sketch}(i)}{\sum_{i=1}^{k} spread_{exact}(i)}$

4.2. Datasets

We used real AT&T datasets in our experiments. From these we projected a tabular array of values reflecting the call volume in the AT&T network. The data sets gives the number of calls collected in intervals of 10 minutes over the day (x-axis) from approximately 20,000 collection stations allocated over the United States spatially ordered based on a mapping of zip code (y-axis). This effectively creates a tabular dataset of approximately 34MB for each day. We stitched consecutive days to obtain data sets of various sizes.

For a separate set of experiments, we constructed synthetic tabular data (128MB) to test the value of varying the distance parameter p in searching for a known clustering. We divided this dataset into six areas representing $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{16}$ of the data respectively. Each of these pieces was then filled in to mimic six distinct patterns: the values were chosen from random uniform distributions with distinct means in the range 10,000 – 30,000. We then changed about 1% of these values at random to be relatively large or small values that were still plausible (so should not be removed by a pre-filtering stage). Hence, under any sensible clustering scheme, all tiles in areas created from the same distribution should be grouped together in the clustering.

4.3. Assessing Quality and Efficiency of Sketching

These experiments were run on an UltraSparc 400MHz processor with 1.5Gbytes of memory. To assess the performance of sketch construction, we conducted the following experiment. For a tabular call volume data set corresponding to a single day (approximately 34MB), we measured the

[1]Here, $\mathrm{xor}(T, T) = \mathrm{xor}(F, F) = 0$ and $\mathrm{xor}(T, F) = \mathrm{xor}(F, T) = 1$. So the xor function as used here will only count the cases where sketching got the correct answer.

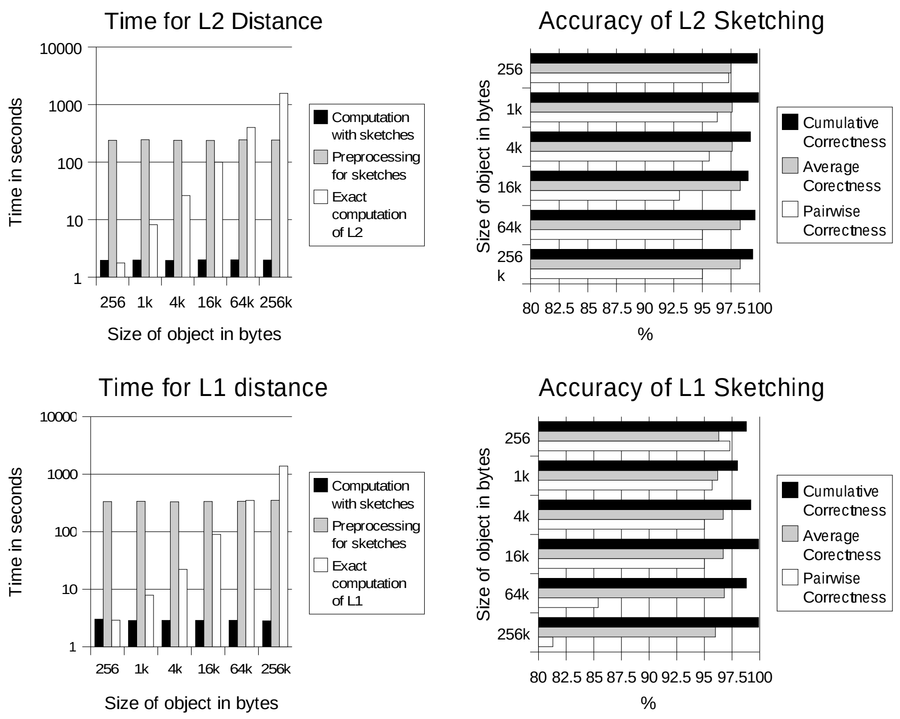


Figure 2. Assessing the distance between $20,000$ randomly chosen pairs

time to create sketches of various sizes, for both L_1 and L_2 norms. We considered objects (tiles) from size only 256 bytes up to 256k. The tests evaluated the time to assess the distance between 20,000 random pairs of tiles in the data space. Figure 2 presents the results.

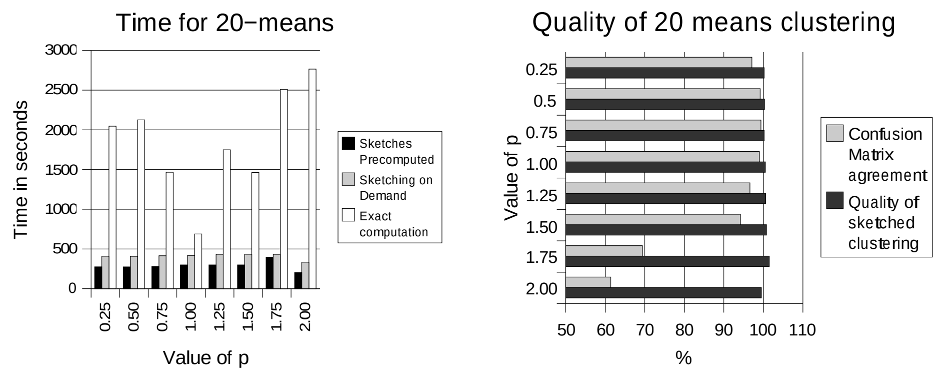
The exact method requires the whole tile to be examined, so the cost of this grows linearly with the size of the tile. The sketch size is independent of the tile size. When we create the sketches in advance, we consider all possible subtables of the data in square tiles (of size 8×8, 16×16 and so on up to 256×256). The processing cost is largely independent of the tile size, depending mainly on the data size. Since we use the same sized data set each time, the pre-processing time varies little. The time to assess the L_p distance using sketches is much faster than the exact method when sketches are precomputed in almost every case.

We computed Average Correctness and Cumulative Correctness (Definitions 8 and 7); in most cases these was within a few percent of the actual value, for relatively small sized sketches (recall that the accuracy of sketching can be improved by using larger sized sketches). Figure 2 presents the results. Note that the times are shown on a logarithmic scale, since the cost for exact evaluation becomes much higher for larger tile sizes. On the average, the cumulative distance found by sketching agrees with very high accuracy with the actual answer. Observe that the quality of the pairwise comparisons decrease slightly under L_1 distance for large tile sizes. We claim that this is explained by our data set: for large enough tiles, the L_1 distance between any pair is quite similar. Hence, with the variation introduced by our approximation, it becomes harder to get these comparisons correct. However, in this situation, we claim that such errors will not affect the quality of any derived clustering detrimentally, because following such a comparison, it does not make much difference which cluster the tile is allocated to, since the tile is approximately the same distance from both. We shall see that this claim is vindicated when we examine the quality of the clusterings derived using approximate distance comparisons.

Finally, since in clustering it is not the absolute value that matters, but the results of comparisons, we ran tests by picking a pair of random points in data space and a third point. We determined which of the pair was closest to the test point using both sketching and exact methods, and recorded how frequently the result of the comparison was erroneous. This is the pairwise comparison correctness of Definition 9. Under all measures of accuracy, the results of Figure 2 show that sketching preserves distance computations very effectively.

4.4. Clustering Using Sketches

The second set of experiments aims at evaluating the utility of sketching when applied to clustering with k-means, a popular data mining algorithm. In this experiment, we stitch 18 days of data together, constructing a data set of over 600MB of raw data. With our experiments we wish to evaluate the performance of sketching under the following possible scenarios: (1) Sketches have been precomputed, so no time is devoted for sketch computation, just for running the clustering algorithm on sketches (2) Sketches are not available and so they have to be computed "on demand"



(a) timing results for different L_p distances (b) quality results for different L_p distances

Figure 3. Results for clustering the data using k-means (k=20), with data divided into tiles of size 9K.

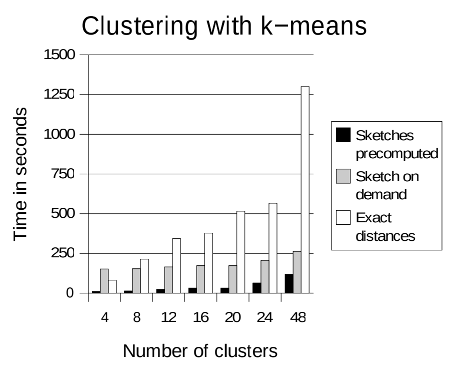
(3) Sketching is not used; instead the exact distance is computed. In each case, we divided the data up into tiles of a meaningful size, such as a day, or a few hours, and ran the k-means clustering algorithm on these tiles. To ensure that the methods were comparable, the only difference between the three types of experiments was the routines to calculate the distance between tiles: everything else was held constant.

Varying the value of p. Figure 3 presents the results for a tile of size 9K. This tile represents a day's data for groups of 16 neighboring stations. We experimented with a variety of settings for p: the traditional 2.0 and 1.0, and fractional values in between. The first set of results show that sketching is dramatically faster than using exact distance computations, while giving a clustering that is as good as that found using exact computation. To assess clustering quality we used two approaches. First, by creating a confusion matrix between the clustering using sketches and the clustering with exact computations. The percentage of tiles that are classified as being in the same cluster by both methods indicates how close sketching gets to the benchmark method, i.e., the confusion matrix agreement of Definition 10. An alternative measure of the quality of two clusterings comes by comparing the spread of each cluster; the better the clustering, the smaller this spread will be. The spread is the sum of the divergence of each cluster from the centroid of that cluster. This gives the objective way to test the quality of the clusterings described in Definition 11.

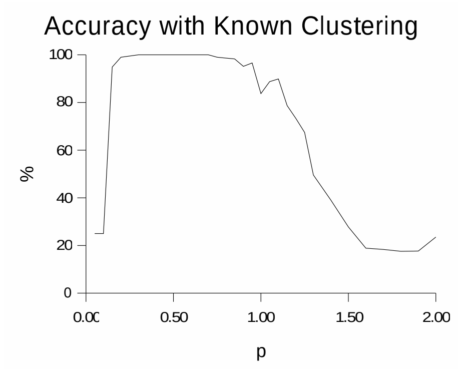
When sketches are precomputed the time to perform the clustering is many times faster, in some cases, an order of magnitude faster, than using exact distance computations. This is because the tiles being compared in this test are 9K in size, whereas the sketches are less than 1K, and so can be processed faster. By Theorem 2 the size of sketches is independent of the data size, so we know that this difference will grow more pronounced as the size of the objects being compared increases.

Perhaps less expected is the result that even when sketches are not available in advance, computing a sketch when it is first needed, and storing this for future comparisons is worthwhile. In fact we obtain major time savings: the speed-up is of the order of 3 to 5 times. Although creating a sketch can be costly (it requires the data tile to be convolved repeatedly with randomized matrices), in the application of clustering, one data tile will be compared with others many times. So the cost of making the sketch is recouped over the course of the clustering since each subsequent comparison has a much lower cost than the corresponding exact computation. Observing Figure 3, it is evident that there exists little variation in the cost of the algorithms using sketches, whereas the timings for the exact computations are much more variable, but consistently significantly more costly than with approximate comparisons. In each case, creating sketches adds the same cost (about 130s of compute time) since the number of sketches that are required, and the cost of creating each sketch is independent of the data values and the value of p. Note that since a slightly different method is used for $p = 2$ compared to $p < 2$ (see Section 3) — this means that L_2 distance is faster to estimate with sketches in this case, since the approximate distance is found by computing the L_2 distance between the sketches, rather than by running a median algorithm, which is slower.

Another positive result regards the accuracy of the sketching approach. By analyzing the confusion matrix between computations using sketches and computations using exact distances, we see that for several of our experiments there is a high degree of correlation, indicating that tiles were allocated into the same cluster. We observe that the agreement is less good for higher values of p — for L_2 distance, it reduces to around 60%. But although the clustering we get is quite different from that obtained with the exact distance, the quality of the clustering in all cases is as good as the one found by exact methods. In fact, in many cases using sketches produces a clustering that is better than that with exact comparisons (where the quality rating is greater than 100%). This at first surprising result is explained as follows: even when we compute exact distances, the k-means algorithm does not guarantee finding the best clustering. Evaluating distances using sketches introduces small distortions in the distance computation that cause a different clustering to be reached (some objects will be placed in



(a)



(b)

Figure 4. (a) Clustering with different numbers of means (b) Varying p to find a known clustering

a different cluster). This clustering can be a small improvement over the one obtained by exact computations.

Varying the number of clusters. Figure 4 (a) shows a series of experiments on the same set of data with k-means, as k is increased. The difference between having pre-computed sketches, and sketching on demand, remains mostly constant, at around 140s. The cost of using exact computations is significantly more expensive in all but one case[2], and without sketches the time cost appears to rise linearly with k. This is achieved using relatively large sketches with 256 entries. This time benefit could be made even more pronounced by reducing the size of the sketches at the expense of a loss in accuracy.

In summary, the use of approximate comparisons does not significantly affect the quality of the clustering found. Indeed, we have shown cases where the quality of the clustering is improved. The k-means algorithm is already inexact: it depends on a heuristic to generate the clustering, and uses randomness to generate the initial k-means that are refined over the course of the program. We have shown that adding approximate distance comparisons to this clustering algorithm makes it run significantly faster, without noticeably affecting the quality of the output. This provides evidence that clustering algorithms or other procedures making use of L_p distance computations can be significantly sped up by using approximate computations with sketches, and that this will yield results that are just as good as those made with exact computations.

4.5. Clustering Using Various L_p Norms

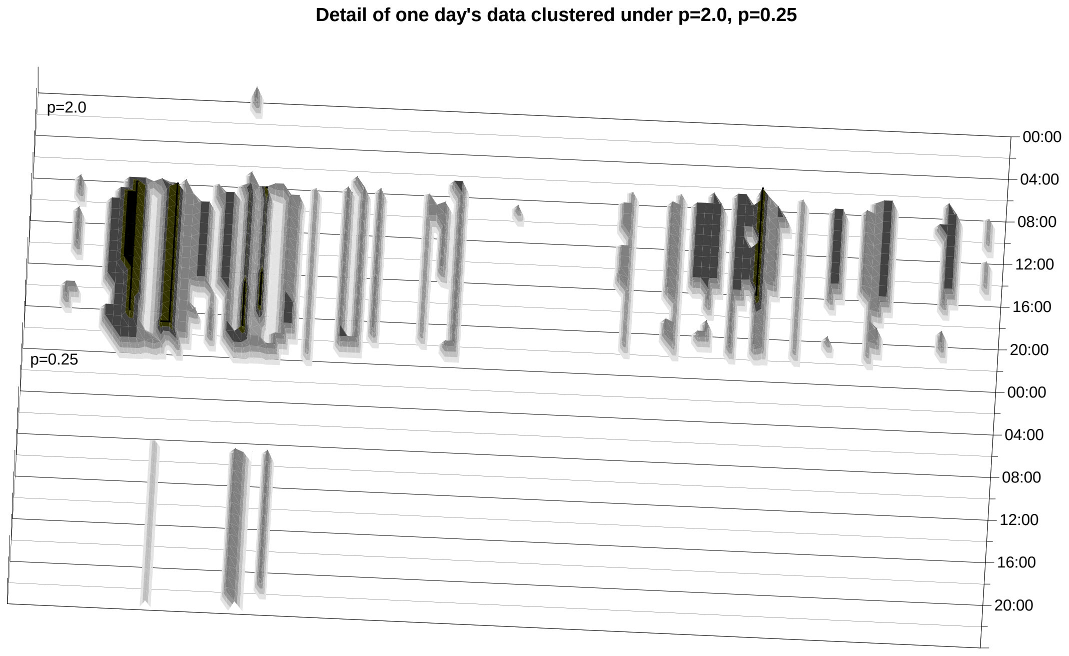
With our last experiments we wish to evaluate the utility of using various L_p norms in clustering for data mining tasks. We examine the output of the clustering procedure to determine the effects of varying the parameter p in the distance computations. We approach this in two ways: first, by examining how varying p can find a known clustering in synthetic data; and second, by presenting a case study analyzing a single day's worth of data.

[2]This occurs when the number of comparisons made in the course of clustering is not enough to "buy back" the cost of making the sketch.

Synthetic Data. Recall that our synthetic data set is made by dividing the data into six pieces, and filling each with a distinct pattern, then adding "errors" of high and low readings. We divided this data set into square tiles of size 64k, and ran clustering using sketching on them for many values of p in the range $[0, 2]$. Since we know which cluster each tile should belong to, we can accurately measure how well the clustering does in the presence of the artificial errors and the approximation caused by using sketches. We measure the percentage of the 2000 tiles allocated to their correct cluster (as per Definition 10). Figure 4 (b) shows the results as we vary p.

We first observe that traditional measures, L_1 and L_2 especially perform very badly on this dataset. But if we set p between 0.25 and 0.8, then we get the answer with 100% accuracy. The explanation for this is that the larger p is, the more emphasis it puts on "outlier" values: so if a single value is unusually high, then this will contribute a huge amount to the L_2 distance — it adds the square of the difference. With distances this high, it is impossible to decide which of the clusters a tile best belongs to, and the clustering obtained is very poor (if we allocated every tile to the same cluster, then this data set under this measure would score 25%). On the other hand, as p gets smaller, then the L_p distance gives a lower penalty to outliers, as we would wish in this case. If we keep decreasing p closer to zero, then the measure approaches the Hamming distance, that is, counting how many values are different. Since here almost all values are different, the quality of the clustering is also poor when p is too small. However, we suggest that $p = 0.5$ gives a good compromise here: the results are not overly affected by the presence of outliers, and even with the approximation of sketching, we manage to find the intended clustering with 100% accuracy.

Real Data. For the case study, the geographic data was linearized, and grouped into sets of 75 neighboring stations to faciliate visual presentation. A subsection of the results of the clustering is shown in Figure 5 . Each point represents a tile of the data, an hour in height. Each of the clusters is represented by a different shade of grey. The largest cluster is represented with a blank space, since this effectively



The top figure shows when $p = 2.0$; the lower when $p = 0.25$. Each shade of grey denotes a different cluster (the largest cluster is denoted by a blank space to aid visibility). For higher p, more detail can be seen; for lower p, the most important sections are brought to the fore.

Figure 5. Detail of a clustering for a single day

represents a low volume of calls, and it is only the higher call volumes that show interesting patterns.

Visual analysis of this clustering immediately yields information about the calling patterns. Firstly, it is generally true that access patterns in any area are almost identical from 9am till 9pm. We can see very pronounced similarities throughout this time — long, vertical lines of the same color indicating that an area retains the same attributes throughout the day. Call volume is negligible before 9am, but drops off gradually towards midnight. It is also clear how different values of p bring out different features of the data in this data set. For $p = 2$, perhaps the most common distance metric, a large fraction is allocated to non-trivial clusters. These clearly correspond to centers of population (New York, Los Angeles etc.) represented by clusters of darker colors. On either side of the center of these areas (ie, the greater metropolitan areas), patterns are less strong, and access is only emphasized during peak hours. We see how the clusters represented by darker colors are often flanked by those of lighter colors, corresponding to this phenomenon. For $p = 1$, there is less detail, but equally, the results are less cluttered. Most of the clusters are indistinguishable from the background, leaving only a few places which can be seen to differ under the L_1 measure. It is significant that these also show strong vertical similarity of these clusters, suggesting that these areas show particular points of interest. When studying the full data set, we noticed that while some areas figure throughout the day, some sections on one side of the data are only active from 9am till 6pm; at the other extreme of the data diagram there are regions that are active from 6am to 3pm. This is explained by business hours and the three hour time difference between East and West coasts . This difference is not so clearly visible on the more populated clustering with $p = 2$. As we further decrease p to 0.25, only a few regions are distinct from the default cluster. These are clearly areas worthy of closer inspection to determine what special features they exhibit. It therefore seems that p can be used as a useful parameter of the clustering algorithm: set p higher to show full details of the data set, or reduce p to bring out unusual clusters in the data.

This example demonstrates how clusterings of the tabular data set can highlight important features. It has been confirmed by our knowledge of this data sets; in novel data sets, we would expect to be able to find the important features of the data, without the foreknowledge of what these might be. It also demonstrates the importance of different values of p, in contrast with previous attention focusing on the traditional distance measures L_1 and L_2. The whole continuum of L_p distances can give different and useful information about the data.

5. Conclusion

Mining tabular data calls for computing distances between different subtables. This presents two challenges: First, what is a suitable distance function that captures the similarity between two subtables? We propose L_p norms not only for the classical $p = 1, 2$, but also for non-integral p, $0 < p \leq 2$, which presents a novel, fully tunable, similarity notion between matrices and between vectors. By setting p lower than 1, we are able to reduce the effect that a few outlier values can have on the distance comparisons. Second, how to compute distances between multiple pairs of subtables most efficiently? This is crucial since tabular data are often massive in our motivating applications and computing each distance would be time consuming. We present sketching methods for computing L_p distances; while there are well known dimensionality reduction techniques for $p = 2$, we present sketching functions based on stable probability distributions for other p. While other transform based methods (like Discrete Cosine Transforms or Haar Wavelet Coefficients) may be used to approximate the distances between subtables, they fail in two ways: first, they do not estimate L_p distances for $p \neq 2$ adequately, and second, they are not amenable to the composition of sketches we perform here, which is crucial for the efficiency of our algorithms. As we have shown, our sketch computations are provably accurate in estimating the L_p distances.

We use these algorithms to estimate distances between subtables in the k-means clustering algorithm. We considered more than half a gigabyte of real data. We were able to show experimentally that our algorithms are substantially faster than straightforward methods, and they are accurate in the clustering context. Furthermore, we get visual and experimental confirmation that L_p distance presents an interesting notion of similarity for non-integral values of p.

Our work initiates the study of tabular data mining in which similarity between subtables can be approximated rapidly using our sketching methods. Many mining issues in tabular data remain to be explored. We also find it intriguing that using L_p distances with non-integral values of p presents interesting clustering behavior, both visually and experimentally. It appears that p is a fully adjustable parameter that can be used as a slider to vary the nature of the clustering.

References

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *8th International Conference on Database Theory*, pages 420–434, 2001.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms*, volume 730, pages 69–84. Lecture Notes in Computer Science, Springer, 1993.

[3] S. Babu, M. Garofalakis, and R. Rastogi. Spartan: A model-based semantic compression system for massive data tables. In *Proceedings of ACM SIGMOD*, pages 283–294, 2001.

[4] D. Belanger, K. Church, and A. Hume. Virtual data warehousing, data publishing, and call details. In *Proceedings of Databases in Telecommunications*, volume 1819, pages 106–117. Lecture Notes in Computer Science, Springer, 1999.

[5] A. L. Buchsbaum, D. F. Caldwell, K. W. Church, G. S. Fowler, and S. Muthukrishnan. Engineering the compression of massive tables: an experimental approach. In *Proceedings of the 11th Annual Symposium on Discrete Algorithms*, pages 175–184, 2000.

[6] Daytona. AT&T research, Daytona Database Management System. Details at `http://www.research.att.com/projects/daytona/`.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226, 1996.

[8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, 2000.

[9] A. Feldmann, A. G. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proceedings of SIGCOMM*, pages 257–270, 2000.

[10] V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: Mining and monitoring evolving data. In *Proceedings of ICDE*, pages 439–448, 2000.

[11] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD*, pages 73–84, 1998.

[12] P. Indyk. Stable distributions, psuedorandom generators, embeddings and data stream computation. In *40th Symposium on Foundations of Computer Science*, 2000.

[13] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *26th International Conference on Very Large Databases*, pages 363–372, 2000.

[14] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[15] M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, page 207, 1997.

[16] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *20th International Conference on Very Large Databases*, pages 144–155. Morgan Kaufmann, 1994.

[17] J. Nolan. Stable distributions. Available from `http://academic2.american.edu/∼jpnolan/stable/chap1.ps`.

[18] R. Rastogi and K. Shim. Mining optimized support rules for numeric attributes. In *Proceedings of the International Conference on Data Engineering*, pages 126–135, 1999.

[19] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of ACM SIGMOD*, pages 1–12, 1996.

[20] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. In *26th International Conference on Very Large Databases*, 2000.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD*, pages 103–114, 1996.