Distributed Top-K Outlier Detection from Astronomy Catalogs using the DEMAC System

Haimonti Dutta* Chris Giannella† Kirk Borne‡ Hillol Kargupta§

Abstract

The design, implementation and archiving of large sky surveys is an important part of astronomy research. The Sloan Digital Sky Survey (SDSS), The Two Micron All Sky Survey (2MASS) are some such surveys producing tera bytes of geographically distributed data which need to be stored, analyzed and queried to enable scientific discoveries. In this paper, we describe the architecture of a system for Distributed Exploration of Massive Astronomy Catalogs (DEMAC) which is built on top of the existing National Virtual Observatory environment. We describe distributed algorithms for doing Principal Component Analysis (PCA) using random projection and sampling based techniques. Using the approximate principal components, we develop a distributed outlier detection algorithm which enables identification of data points that deviate sharply from the "correlation structure" of the data. We provide simulation results with data obtained from sky-surveys SDSS and 2MASS.

1 Introduction

The design, implementation, and archiving of very large sky surveys is playing an increasingly important role in today's astronomy research. Many projects (such as GALEX[1], ROSAT[2]) are producing enormous geographically distributed catalogs (tables) of astronomical objects. These virtual catalogs are expected to increase science return and enable scientific discoveries by cross correlation of data across multiple sky surveys. The development and deployment of a U.S. National Virtual Observatory (NVO) [3] is a step in this direction. However, processing, mining, and analyzing these distributed and vast data collections are fundamentally challenging tasks since most off-the-shelf data mining systems require the data to be down-loaded to a single location before further analysis. This imposes serious scalability constraints on the data mining system and fundamentally hinders the scientific discovery process. This motivates the need to develop communication-efficient distributed data mining (DDM) techniques.

In this paper, we extend the functionality of the system for Distributed Exploration of Massive Astronomy Catalogs (*DEMAC*) [4]. Built on top of the National Virtual Observatory, DEMAC facilitates distributed Principal Component Analysis (PCA), decision tree induction and outlier detection for astronomy catalogs. The work in [4] proposes an architecture for DDM on astronomy catalogs and addresses the problem of distributed covariance matrix computation using a random projection based approach. The main contributions of this paper are: (1) Distributed covariance matrix computation using random sampling and comparison with random projection based approach [4]. (2) A PCA based algorithm for distributed outlier detection. (3) Two astronomical case studies, fundamental plane computation and outlier detection.

2 DEMAC - A System for Distributed Exploration of Massive Astronomical Catalogs

This section describes the high level design of the proposed DEMAC system. DEMAC is designed as an additional web-service which seamlessly integrates into the NVO. It consists of two basic services – (1) *WS-DDM*: which provides DDM capabilities for vertically distributed sky surveys and (2) *WS-CM*: which provides cross-matching capabilities for vertically distributed sky surveys and is intensively used by WS-DDM. We outline here the architecture for the two web-services we will develop.

2.1 WS-DDM – DDM for Heterogeneously Distributed Sky-Surveys This web-service will allow running DDM algorithms on a selection of sky-surveys. The user would use existing NVO services to locate sky-

*Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, USA. hdutta1@csee.umbc.edu

†Department of Mathematics and Computer Science, Goucher College, USA. cgiannel@acm.org

‡Department of Computational and Data Sciences, George Mason University, USA. kborne@gmu.edu

§Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, USA. Also affiliated with AGNIK LLC, Columbia, MD USA. hillol@cs.umbc.edu.

[1] http://www.galex.caltech.edu/

[2] http://www.xray.mpe.mpg.de/cgi-bin/rosat/rosat-survey

surveys and define the portion of the sky to be data mined. The user would then use WS-CM to select a cross-matching scheme for those sky-surveys. Following these two preliminary phases the user would submit the data mining task.

2.2 WS-CM – Cross-Matching for Heterogeneously Distributed Sky-Surveys We illustrate the problem with two archives: the Sloan Digital Sky Survey (SDSS) [1] (containing upward of 100 million records) and the 2-Micron All-Sky Survey (2MASS) [2] (containing 470 million records). Each record contains sky coordinates (ra,dec) identifying the astronomical point sources' position in the celestial sphere as well as many other attributes (460+ for SDSS; 420+ for 2MASS). While each of these catalogs individually provides valuable data for scientific exploration, efficient analysis of the virtual catalog formed by joining these catalogs would enhance their scientific value significantly.

To form the virtual catalog, records must be matched based on their position in the celestial sphere. Consider record t from SDSS and s from 2MASS with sky coordinates $t[ra, dec]$ and $s[ra, dec]$. Each record represents a set of observations about an astronomical object *e.g. a galaxy*. The sky coordinates are used to determine if t and s match, *i.e.* are close enough that t and s represents the same astronomical object. For each match (t, s), the result is a record $t \bowtie s$ in the virtual catalog with all of the attributes of t and s.

To achieve this, *match indices* are created and co-located with each sky survey. Specifically, for each pair of surveys (tables) T and S, a distinct pair of match indices must be kept, one at each survey.The indices do not need to be created each time a data mining task is run. Instead, each pair of indices only need be created *once* and then any data mining task can use them. The net result is the ability to mine virtual tables at low communication cost.

2.3 DDM Algorithms: Definitions and Notation In the next two sections we describe DDM algorithms[3] to be used as part of the WS-DDM web service. Let M denote an $n \times m$ matrix with real-valued entries. This matrix represents a dataset of n tuples from $\mathbb{R}^m$. Let M^j denote the j^{th} column and $M^j(i)$ denote the i^{th} entry of this column. Let $\mu(M^j)$ and $Var(M^j)$ denote the *sample mean and variance* respectively of this column. Let $Cov(M^j, M^k)$ denote the *sample covariance* of the j^{th} and k^{th} columns *i.e.* $\frac{\sum_{i=1}^{n}[\mu(M^j)-M^j(i)][\mu(M^k)-M^k(i)]}{n-1}$. Note, $Var(M^j) = Cov(M^j, M^j)$). Finally, let $Cov(M)$ denote the *covariance matrix* of M *i.e.* the $m \times m$ matrix whose $(j,k)^{th}$ entry is $Cov(M^j, M^k)$.

Assume this dataset has been vertically distributed over two sites S_A and S_B. Since we are assuming that the data at the sites is perfectly matched, then S_A has the first p attributes and S_B has the last q attributes $(p+q = m)$. Let A denote the $n \times p$ matrix representing the dataset held by S_A, and B denote the $n \times q$ matrix representing the dataset held by S_B. Let $A : B$ denote the concatenation of the datasets *i.e.* $M = A : B$. The j^{th} column of $A : B$ is denoted $[A : B]^j$.

Following standard practice in applied statistics, we pre-process M by normalizing so that $\mu(M^j) = 0$ and $Var(M^j) = 1$. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$ denote the eigenvalues of $Cov(M)$ and $v_1, v_2, \ldots, v_m$ the associated eigenvectors (we assume the eigenvectors are column vectors *i.e.* $m \times 1$ matrices.) (pairwise orthonormal). The j^{th} *principal direction* of M is v_j. The j^{th} *principal component* is denoted z_j and equals Mv_j (the projection of M along the j^{th} direction).

3 Virtual Catalog Principal Component Analysis

PCA is a well-established data analysis technique used in a large number of disciplines: astronomy, computer science, biology, *etc.* To our knowledge, the problem of vertically distributed PCA computation has been addressed in [5], [7] and [4]. We briefly review the algorithm for distributed PCA using random projection in this section.

Recall that $A : B$ is normalized to have zero column sample mean and unit sample variance. As a result, $Cov([A : B]^j, [A : B]^k) = \frac{\sum_{i=1}^{n}[A:B]^j(i)[A:B]^k(i)}{n-1}$ which is the *inner product* between $[A : B]^j$ and $[A : B]^k$. Clearly this inner product can be computed without communication when $[A : B]^j$ and $[A : B]^k$ are at the same site (*i.e.* $1 \leq j, k \leq p$ or $p+1 \leq j, k \leq p+q$). It suffices to show how the inner product can be approximated across different sites, in effect, how $A^T B$ can be approximated. The key idea is based on the fact that high-dimensional random vectors are nearly orthogonal [11].

FACT 1. *Let R be an $\ell \times n$ matrix each of whose entries is drawn independently from a distribution with variance one and mean zero. It follows that $E[R^T R] = \ell I_n$ where I_n is the $n \times n$ identity matrix.*

We will use the following steps for computing $A^T B$ – (1)S_A sends S_B a random number generator seed. (2) S_A and S_B generate a $\ell \times n$ random matrix R

[3] All of these algorithms assume that the participating sites have the appropriate match indices.

where ℓ. Each entry is generated independently and identically from any distribution with mean zero and variance one.(3) S_A sends RA to S_B; S_B sends RB to S_A. (4) S_A and S_B compute $D = \frac{(RA)^T(RB)}{\ell}$. Note that, $E[D] = E[\frac{A^T(R^TR)B}{\ell}] = \frac{A^TE[R^TR]B}{\ell}$, which, by Fact 1, equals $\frac{A^T(\ell I_n)B}{\ell} = A^TB$. Hence, on expectation, the algorithm is correct. However, its communication cost (bytes) divided by the cost of the centralization-based algorithm, $\frac{4m\ell+4}{4nm} = \frac{\ell}{n} + \frac{1}{nm}$, is small if $\ell << n$. Indeed ℓ provides a "knob" for tuning the trade-off between communication-efficiency and accuracy.

Another technique to approximate principal components is to use uniform sampling. The key steps involved are as follows – (1) The central co-ordinator site sends S_A and S_B the percentage p of data to be sampled. (2) S_A and S_B send to the co-ordinator site p% of their uniformly sampled local data. (3) The co-ordinator site estimates the covariance matrix and hence the Principal Components (PCs) on the sampled data. (4)The approximated PCs are sent to S_A and S_B.

4 PCA Based Methods for Outlier Detection

In this section, we describe a PCA-based technique for outlier detection (on *centralized* data) using the last principal components([6]). These techniques look to identify data points which deviate sharply from the "correlation structure" of the data.

Using the notation introduced in section 2.3, the j^{th} component is a linear combination of the columns of M (the k^{th} column has coefficient $v_j(k)$) with sample variance λ_j *i.e.*the variance over the entries of z_j is λ_j. Thus, if λ_j is very small and there were no outlier data points, one would expect the entries of z_j to be nearly constant. In this case, v_j expresses a nearly linear relationship between the columns of M. A data point which deviates sharply from the correlation structure of the data will likely have its z_j entry deviate sharply from the rest of the entries (assuming no other outliers). Since the last components have the smallest $\lambda's$, then an outlier's entries in these components will likely stand out. This motivates examination of the following statistic for the i^{th} data point (the i^{th} row in M) and some $1 \leq a \leq m$: $d^2_{1,i}(a) = \sum_{j=a}^{m} z_j(i)^2$ where $z_j(i)$ denotes the i^{th} entry of z_j. a is a user defined parameter. A possible criticism of this approach as pointed out in [6] is that $d^2_{1,i}$ gives insufficient weight to the last few PCs. This effect will be more pronounced if some of the PCs have very small variances and these are the ones that are of most interest in outlier detection.

To address this criticism, the components are normalized to give equal weight. Let w_j denote the normalized j^{th} principal direction: the $m \times 1$ vector whose i^{th} entry is $\frac{v_j(i)}{\sqrt{\lambda_j}}$. The normalized j^{th} principal component is $\hat{z_j} = Mw_j$. The sample variance of $\hat{z_j}$ equals one, so, the weights of the normalized components are equal. This statistic we use for the i^{th} data point is $d^2_{2,i}(a) = \sum_{j=a}^{m} \hat{z_j}(i)^2$. Next, we discuss the development of a distributed algorithm for finding the top k ranked data points in $[A : B]$ with respect to $d^2_{2,i}(a)$. For this, we only consider the case involving the last component, $a = m$:

$$d^2_{2,i} = d^2_{2,i}(m) = \hat{z_m}(i)^2. \tag{4.1}$$

4.1 Vertically Distributed Top k Outlier Detection First, the algorithm in Section 3 is applied to approximate $Cov(A : B)$ – at the end, each site has the approximation. The eigenvalues and vectors of this matrix are computed $\tilde{\lambda_1}, \ldots, \tilde{\lambda_m}$ and $\tilde{v_1}, \ldots, \tilde{v_m}$, along with the normalized eigenvectors $\tilde{w_1}$, ..., $\tilde{w_m}$ (no communication required). However, the approximations to the last component cannot be computed without communication. The approximation to the last normalized principal component is defined to be $\tilde{z_m} = [A : B]\tilde{w_m}$.

Let $\tilde{w_m}(A)$ denote the $p \times 1$ vector consisting of the first p entries of $\tilde{w_m}$. This is the part of $\tilde{w_m}$ corresponding to the attributes at site S_A. Let $\tilde{w_m}(B)$ denote the $q \times 1$ vector consisting of the last q entries of $\tilde{w_m}$. This is the part of $\tilde{w_m}$ corresponding to the attributes at site S_B. We have $\tilde{z_m} = [A : B]\tilde{w_m} = A\tilde{w_m}(A) + B\tilde{w_m}(B) = \tilde{z_m}(A) + \tilde{z_m}(B)$. Given $1 \leq i \leq n$, let $\tilde{z_m}(A, i)$ denote the i^{th} entry of $A\tilde{w_m}(A)$ and $\tilde{z_m}(B, i)$ denote the i^{th} entry of $B\tilde{w_m}(B)$. Since, $d^2_{2,i} = \hat{z_m}(i)^2$, we define the approximation to $d^2_{2,i}$ as $\tilde{d_{2,i}}^2 = (\tilde{z_m}(A, i) + \tilde{z_m}(B, i))^2$. Our goal is to develop a distributed algorithm for computing the top k among $\{\tilde{d_{2,i}}^2 : 1 \leq i \leq n\}$. Since $\tilde{z_m}(A, i)$ and $\tilde{z_m}(B, i)$ can be computed without communication, it suffices to solve the following problem.

PROBLEM 1. (DISTRIBUTED SUM-SQUARE TOP K) *Site S_A has $a_1, \ldots, a_n \in \mathbb{R}$ and S_B has $b_1, \ldots, b_n \in \mathbb{R}$. Both sites have integer $n > k > 0$. The sites must compute the top k among $c_1 = (a_1 + b_1)^2$, ..., $c_n = (a_n + b_n)^2$.*

A communication-efficient algorithm for solving this problem, Distributed Sum-Square Top K (DSSTK) is described next.

4.2 An Algorithm for the DSSTK Problem In this section, we describe an algorithm for solving the problem when $k = 1$. At the end, both sites have indices $i^* = argmax\{c_i : 1 \leq i \leq n\}$ and c_{i^*}. Hence, the problem when $k > 1$ is solved by removing a_{i^*} and b_{i^*} then repeating, *etc.*

The algorithm for finding i^* carries out two stages. First, S_A and S_B solve the following two distributed sub-problems: find $i^{*M} = argmax\{a_i + b_i\}$ (Distributed Max Algorithm) and $i^{*m} = argmin\{a_i + b_i\}$ (Distributed Min Algorithm). Second, each site returns $argmax\{(a_{i^{*M}} + b_{i^{*M}})^2, (a_{i^{*m}} + b_{i^{*m}})^2\}$. Clearly this equals i^*, so, all that remains is to explicate the Distributed Max algorithm. This is described in Algorithm 4.2.1. The Min algorithm is completely analogous and obtained by replacing all occurrences of "argmax" with "argmin".

Algorithm 4.2.1 Distributed Max Algorithm

1. S_A selects $i^A = argmax\{a_i : 1 \leq i \leq n\}$ and sends $\langle a_{i^A}, i^A \rangle$ to S_B. In parallel, S_B selects $i^B = argmax\{b_i : 1 \leq i \leq n\}$ and sends $\langle b_{i^B}, i^B \rangle$ to S_A. [**16 bytes**]
2. S_A receives the message from S_B and replies with $\langle a_{i^B} \rangle$. In parallel, S_B receives the message from S_A and replies with $\langle b_{i^A} \rangle$. [**8 bytes**]
3. Both sites now have data points a_{i^A}, a_{i^B}, b_{i^A}, b_{i^B}, and indices i^A, i^B. If $i^A = i^B$, then both sites terminate.
4. Otherwise, S_A replaces a_{i^A} and a_{i^B} with $\frac{a_{i^A} + b_{i^A}}{2}$ and $\frac{a_{i^B} + b_{i^B}}{2}$. Site S_B replaces b_{i^A} and b_{i^B} in exactly the same way. Goto step 1.

Next we prove that the Distributed Max Algorithm is correct *i.e.* the algorithm terminates and both sites return the correct index. In fact, we prove a more general result. Namely, that both sites return index $i^{*M} = argmax\{f(a_i + b_i)\}$ where $f : \mathbb{R} \to \mathbb{R}$ is any strictly increasing function.

Correctness of the algorithm follows from the result that shows that for each i, a_i and b_i change values at most once during the course of the algorithm. Moreover, when they change, they change to $\frac{a_i + b_i}{2}$. Let $a_i(0), b_i(0)$ denote a_i, b_i and $a_i(j)$, $b_i(j)$ denote the value of a_i and b_i held at the end of round $j \geq 1$.

LEMMA 4.1. *Suppose the algorithm just completed round $t \geq 1$. For each $1 \leq i \leq n$, let $j(i)$ denote the round at which index i was first selected by either site (if i was not selected during the t rounds, then let $j(i) = t + 1$). It is the case that*

1. *for all $1 \leq j < j(i)$, $a_i(j) = a_i$ and $b_i(j) = b_i$;*
2. *for all $j(i) \leq j \leq t$, $a_i(j) = b_i(j) = \frac{a_i + b_i}{2}$.*

The proof (omitted due to space constraints) is based on the following idea: Suppose the algorithm did not terminate after n rounds. Let $(i_1^A, i_1^B), \ldots, (i_n^A, i_n^B)$ denote the indices selected during each of n rounds. It follows from Lemma 4.1 that a pair of indices (i_j^A, i_j^B), will not be selected if i_j^A was selected at a previous round, $j_1 < j$ and i_j^B was selected at a previous round, $j_2 < j$ (possibly different than j_1). Otherwise, $i_{j_1}^A = i_{j_1}^B$ and, hence, the algorithm would have terminated at round j_1. But, at most $n - 1$ pairs of indices can satisfy this condition. A contradiction is reached.

Let $1 \leq t \leq n$ denote the termination round and let i^M denote the index agreed upon by both sites upon termination. To complete the correctness proof, we need show that i^M is the correct index, $i^{*M} = argmax\{f(a_i + b_i)\}$. Consider any index $1 \leq \ell \leq n$. From Lemma 4.1, one of the following holds: (1)$a_\ell(t) = a_\ell$ and $b_\ell(t) = b_\ell$; (2)$a_\ell(t) = b_\ell(t) = \frac{a_\ell + b_\ell}{2}$. In either case it follows that $a_\ell(t) + b_\ell(t) = a_\ell + b_\ell$. Hence, $i^{*M} = argmax\{f(a_i(t) + b_i(t))\}$.

By definition of the algorithm, $i^M = argmax\{a_i(t)\} = argmax\{b_i(t)\}$. Thus, $a_{i^M}(t) + b_{i^M}(t) \geq a_\ell(t) + b_\ell(t)$. Therefore, since f is strictly increasing, $f(a_{i^M}(t) + b_{i^M}(t)) \geq f(a_\ell(t) + b_\ell(t))$. We conclude that $i^M = argmax\{f(a_i(t) + b_i(t))\} = i^{*M}$ as desired.

Comment: Each round of the Distributed Max algorithm requires 24 bytes of communication. Thus, the total communication is 24ρ where ρ is the number of rounds until termination. This is quite a small communication requirement unless ρ is large(In the worst case ρ is O(n), equivalent to centralizing the data. On average case it is O(lg n)). However, this communication efficiency is won at a price – the algorithm requires ρ rounds of synchronization. Even for relatively modest ρ, the cost of these synchronizations could be unacceptably high. A simple way to address this problem is to have each site, at each round, select its top $\nu > 1$ tuples instead of just its top tuple. The communication cost per round increases to 24ν bytes, but the total number of rounds executed will decrease to ρ/ν.

5 Experimental Results

The identification of correlations among astrophysical parameters has lead to important discoveries in astronomy. For e.g., the class of elliptical and spiral galaxies have been found to occupy a 2D space (called the *Fundamental Plane*[9]) inside a 3D space of observed parameters – radius, mean surface brightness and velocity dispersion. Identification of galaxies that do not lie on the fundamental plane is interesting since outliers can lead to the discovery of unusual or rare types of astronomical sources or phenomena ([10]). This section presents a case study involving the detection of outliers in the fundamental plane of galaxy parameters distributed across two catalogs: 2MASS and SDSS.

In our study we measure the accuracy of our distributed versus the centralized approach. We measure the amount of variance captured by the first two

PCs in the distributed and centralized setting using – (1) random sampling and (2) random projection. Next, we use the approximate PCs to perform the outlier detection. For brevity, we simply use the approximate PCs obtained from the sampling scheme[4]. The metric used for comparison of outliers is *Normalized Ranking Error* (NRE) which is defined as follows: $\frac{|rank_{centralized} - rank_{distributed}|}{\text{Number of tuples}}$. Note that if NRE is 0, the two algorithms chose the same outliers.

5.1 Experiment Set One We prepared our test data as follows. Using the web interfaces of 2MASS[5] and SDSS[6], we obtain a dataset of galaxies lying in the sky region between right ascension (RA) 150 and 200 degrees, declination (dec) 0 and +15. It had the following attributes from SDSS: Petrosian I band angular effective radius (*Iaer*), redshift (*rs*), and velocity dispersion (*vd*);[7] and K band mean surface brightness (*Kmsb*)[8] from 2MASS. After removing tuples with missing attributes, we had a 1307 tuple dataset with four attributes. The pre-processing step involved producing two new attributes, logarithm Petrosian I band effective radius (*log(Ier)*), as log(Iaer*rs) and logarithm velocity dispersion (*log(vd)*). The final dataset had three attributes log(Ier), log(vd), Kmsb and was normalized.

We applied PCA directly to this dataset to obtain the centralization based results. Then we treated this dataset as if it were distributed with attributes log(Ier) and log(vd) located at one site and attribute Kmsb at another. We applied both our sampling and random projection based algorithms to approximate the PCs.

Figure 1 shows the average percentage of variance captured as a function of communication percentage. Error bars indicate standard deviation over 100 trials. The percentage captured by the centralized approach, 90.5%, replicates the known result that a fundamental plane exists among these parameters. Also observe that the percentage of variance captured by the distributed algorithm using as little as 10% communication never strays more than 5% from 90.5% for both techniques. This is a reasonably accurate result indicating that the distributed algorithms identify the existence of a plane using 90% less communication.

Figure 2 illustrates the mean NRE as a function of the communication percentage. At 10% communication, the average NRE is found to 0.08 which is a reasonably good approximation at 90% less communication cost.

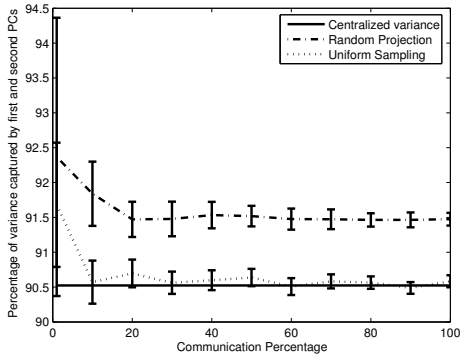

Figure 1: Communication percentage vs. Percentage of variance captured by Centralization, Random Projection and Uniform Sampling on the (log(Ier), log(vd), Kmsb) dataset.

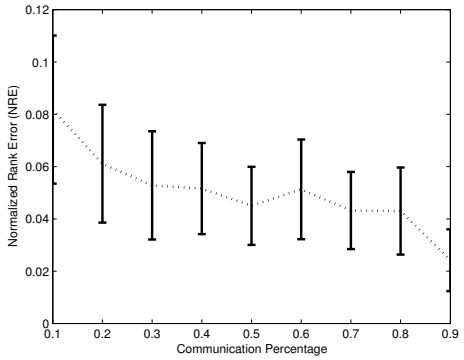

Figure 2: Communication percentage vs. Normalized Rank Error on the, (log(Ier), log(vd), Kmsb) dataset.

5.2 Experiment Set Two Using the web interfaces of 2MASS and SDSS as before, we obtained a dataset having the following attributes from SDSS: Petrosian Flux in the R band (*petroR*) and Petrosian Flux in the I band (*petroI*)[9]; and from 2MASS: K band mean surface brightness (*Kmsb*) and K concentration index (*KconInd*).[10] We had a 29638 tuple dataset with seven attributes. The pre-processing step involved production of new attributes,*petroR-I* = petroR - petroI

[4]Note that Random Sampling is shown to have a better approximation compared to Random Projection for both the datasets used.

[5]http://irsa.ipac.caltech.edu/applications/Gator/

[6]http://cas.sdss.org/astro/en/tools/crossid/upload.asp

[7]*petroRad_i* (galaxy view), z (SpecObj view) and velDisp (SpecObj view) in SDSS DR4

[8]*k_mnsurfb_eff* in the extended source catalog in the All Sky Data Release, http://www.ipac.caltech.edu/2mass/releases/allsky/index.html

[9]*petroMag_r*, *petroMag_i* from (PhotoObjAll)

[10]*k_mnsurfb_eff* and *k_con_indx* in the extended source catalog in the All Sky Data Release

and logarithm K concentration index (*log(KconInd)*). The final dataset was normalized and consisted of attributes petroR-I, log(KconInd), and Kmsb.

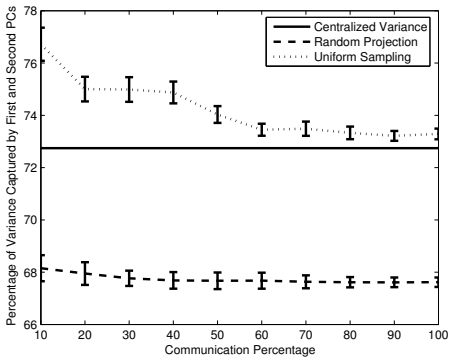

Figure 3: Communication percentage vs. Percentage of variance captured in Centralization, Random Projection and Uniform Sampling on the (petroR-I, log(KconInd),Kmsb) dataset.

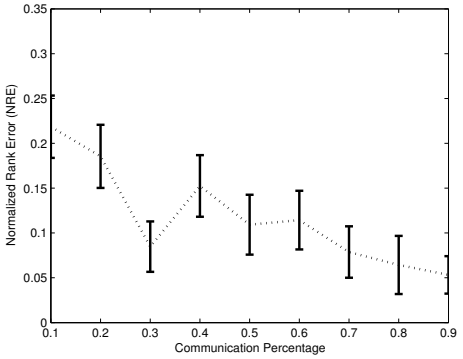

Figure 4: Communication percentage vs. Normalized Rank Error on the (petroR-I, log(KconInd),Kmsb) dataset.

As in Experiment One, we compared the percentage of variance captured by the principal components in the distributed (Random Projection and Uniform Sampling) versus the centralized case. The Figure 3 shows the percentage of variance captured on the petroR-I dataset as a function of communication percentage. Figure 4 reports the mean NRE as a function of the communication percentage for the topmost outlier produced by the centralized and distributed algorithms.

6 Conclusions and Future Work

We proposed a system for the Distributed Exploration of Massive Astronomical Catalogs (DEMAC) in an earlier work [4]. It is built on top of the existing U.S. National Virtual Observatory environment and provides tools for data mining (as web services) without requiring datasets to be down-loaded to a centralized server. In order to extend the functionality of the system, we developed an algorithm for distributed PCA using uniform sampling and compared its performance with a random projection based approach proposed earlier. Using the approximate PCs we designed a communication-efficient distributed algorithm for outlier detection. We carried out two case studies for detecting outliers in the fundamental planes of astronomical parameters. We envision DEMAC to increase the ease with which large, geographically distributed astronomy catalogs are explored and astronomers can better tap the riches of distributed virtual sky survey catalogs.

7 Acknowledgments

This work is supported in part by NSF CAREER Award, IIS-0093353.

References

[1] The Sloan Digital Sky Survey, http://www.sdss.org.

[2] The Two-Micron All Sky Survey, http://www.ipac.caltech.edu/2mass/

[3] US National Virtual Observatory, http://www.us-vo.org/

[4] Chris Giannella, Haimonti Dutta, Ran Wolff, Kirk Borne and Hillol Kargupta. Distributed Data Mining in Astronomy Databases, In the 9th Workshop on Mining Scientific and Engineering Data Sets, 2006.

[5] Kargupta H., Huang W., Sivakumar K., and Johnson E. Distributed Clustering using Collective Principal Component Analysis. Knowledge and Information Systems Journal, 3:422–448, 2001.

[6] Jolliffe I. Principal Component Analysis. Springer-Verlag, 2002.

[7] Kargupta H. and Puttagunta V. An Efficient Randomized Algorithm for Distributed Principal Component Analysis from Heterogeneous Data. In Proceedings of the Workshop on High Performance Data Mining, 2004.

[8] Hodge V. and Austin J. A Survey of Outlier Detection Methodologies. Artificial Intelligence Review, 22(2):85126, 2004.

[9] Elliptical Galaxies: Merger Simulations and the Fundamental Plane, http://irs.ub.rug.nl/ppn/244277443.

[10] Zhang X, Luo A.L. and Zhao Y.H. Outlier detection in astronomical data, Optimizing Scientific Return for Astronomy through Information Technologies. Proceedings of the SPIE, Volume 5493, pp 521-529 (2004).

[11] Arriaga R. and Vempala S. An Algorithmic Theory of Learning: Robust Concepts and Random Projection. In Proceedings of the 40th Foundations of Computer Science, 1999.