Heterogeneous Data Integration with the Consensus Clustering Formalism

Vladimir Filkov[1] and Steven Skiena[2]

[1] CS Dept., UC Davis, One Shields Avenue, Davis, CA 95616
filkov@cs.ucdavis.edu
[2] CS Dept. SUNY at Stony Brook, Stony Brook, NY 11794
skiena@cs.sunysb.edu

Abstract. Meaningfully integrating massive multi-experimental genomic data sets is becoming critical for the understanding of gene function. We have recently proposed methodologies for integrating large numbers of microarray data sets based on consensus clustering. Our methods combine gene clusters into a unified representation, or a consensus, that is insensitive to mis-classifications in the individual experiments. Here we extend their utility to heterogeneous data sets and focus on their refinement and improvement. First of all we compare our best heuristic to the popular *majority rule* consensus clustering heuristic, and show that the former yields tighter consensuses. We propose a refinement to our consensus algorithm by clustering of the source-specific clusterings as a step before finding the consensus between them, thereby improving our original results and increasing their biological relevance. We demonstrate our methodology on three data sets of yeast with biologically interesting results. Finally, we show that our methodology can deal successfully with missing experimental values.

1 Introduction

High-throughput experiments in molecular biology and genetics are providing us with wealth of data about living organisms. Sequence data, and other large-scale genomic data, like gene expression, protein interaction, and phylogeny, provide a lot of useful information about the observed biological systems. Since the exact nature of the relationships between genes is not known, in addition to their individual value, combining such diverse data sets could potentially reveal different aspects of the genomic system.

Recently we have built a framework for integrating large-scale microarray data based on clustering of individual experiments [1]. Given a group of source- (or experiment-) specific clusterings we sought to identify a *consensus clustering* close to all of them. The resulting consensus was both a representative of the integrated data, and a less noisy version of the original data sets. In other words, the consensus clustering had the role of an average, or median between the given clusterings.

The formal problem was to find a clustering that had the smallest sum of distances to the given clusterings:

CONSENSUS CLUSTERING (CC): *Given k clusterings, C_1, $C_2, \ldots,$ C_k, find a consensus clustering C^* that minimizes*

$$S = \sum_{i=1}^{k} d(C_i, C^*) \ . \tag{1}$$

After simplifying the clusterings to set-partitions we developed very fast heuristics for *CC* with the *symmetric difference distance* as the distance metric between partitions. The heuristics were based on local search (with single element move between clusters) and Simulated Annealing for exploring the space of solutions [1]. Practically, we could get real-time results on instances of thousands of genes and hundreds of experiments on a desktop PC.

In *CC* it was assumed that the given data in each experiment were classifiable and it came clustered. The clustering was assumed to be hard, i.e. each gene belongs to exactly one cluster (which is what the most popular microarray data clustering software yields [2]). We did not insist on the clustering or classification method and were not concerned with the raw data directly (although as parts of the software tool we did provide a number of clustering algorithms). Although clear for microarray data it is important to motivate the case that clustering other genomic data is possible and pertinent. Data classification and/or clustering is pervasive in high-throughput experiments, especially during the discovery phase (i.e. fishing expeditions). Genomic data is often used as categorical data, where if two entities are in the same category a structural or functional connection between them is implied (i.e. guilt by association). As massive data resides in software databases, it is relatively easy to submit queries and quickly obtain answers to them. Consequently, entities are classified into those that satisfy the query and those that do not; often into more than two, more meaningful categories. Even if not much is known about the observed biological system, clustering the experimental data obtained from it can be very useful in pointing out similar behavior within smaller parts of the system, which may be easier to analyze further. Besides microarray data, other examples of clustered/classified genomic data include functional classifications of proteins [3], clusters of orthologous proteins [4], and phylogenetic data clusters (http://bioinfo.mbb.yale.edu/genome/yeast/cluster).

In this paper we refine and extend our consensus clustering methodology and show that it is useful for heterogeneous data set integration. First of all, we show that our best heuristic, based on Simulated Annealing and local search, does better than a popular Quota Rule heuristic, based on hierarchical clustering. In our previous study we developed a measure, the Average sum of distances, to assess the quality of data sets integration. Some of the data sets we tried to integrate did not show any benefit from the integration. Here we refine our consensus clustering approach by initially identifying groups of similar experiments which are likelier to benefit from the integration than a general set of experiments. This is equivalent to clustering the given clusterings. After performing this step the consensuses are much more representative of the integrated data. We demonstrate this improved data integration on three heterogeneous data sets, two of microarray data, and one of phylogenetic profiles. Lastly we address the issue of missing data. Because of different naming conventions, or due to experiment errors data may be missing from the data sets. The effect of missing data is that only the data common to all sets can be used, which might be a significant reduction of the amount available. We propose a method for computational imputation of missing data, based on our consensus clustering method, which decreases the negative consequences of missing data. We show that it compares well with a popular microarray missing value imputation method, KNNimpute [5].

This paper is organized as follows. In the rest of this section we review related work on data integration and consensus clustering. We review and compare our existing methodology and the popular quota rule in Sec. 2. In Sec. 3 we present our refined method for consensus clustering, and we illustrate it on data in Sec. 4. The missing data issue is addressed in Sec. 5. We discuss our results and give a future outlook in Sec. 6.

1.1 Related Work

The topic of biological data integration is getting increasingly important and is approached by researchers from many areas in computer science. In a recent work, Hammer and Schneider [6] identify categories of important problems in genomic data integration and propose an extensive framework for processing and querying genomic information. The consensus clustering framework can be used to addresses some of the problems identified, like for example *multitude and heterogeneity of available genomic repositories (C1)*, *incorrectness due to inconsistent and incompatible data (C8)*, and *extraction of hidden and creation of new knowledge (C11)*.

An early study on biological data integration was done by Marcotte et al. [7], who give a combined algorithm for protein function prediction based on microarray and phylogeny data, by classifying the genes of the two different data sets separately, and then combining the genes' pair-wise information into a single data set. Their approach does not scale immediately. Our methods extends theirs to a general combinatorial data integration framework based on pair-wise relationships between elements and any number of experiments.

In machine learning, Pavlidis et al. [8] use a Support Vector Machine algorithm to integrate similar data sets as we do here in order to predict gene functional classification. Their methods use a lot of hand tuning with any particular type of data both prior and during the integration for best results. Troyanskaya et al. [9] use a Bayesian framework to integrate different types of genomic data in yeast. Their probabilistic approach is a parallel to our combinatorial approaches.

A lot of work has been done on specific versions of the consensus clustering problem, based on the choice of a distance measure between the clusterings and the optimization criterion. Strehl et al. [10] use a clustering distance function derived from information theoretic concepts of shared information. Recently, Monti et al. [11] used consensus clustering as a method for better clustering and class discovery. Among other methods they use the *quota rule* to find a consensus, an approach we describe and compare to our heuristics in Sec. 2.2. Other authors have also used the quota rule in the past [12]. Finally, Cristofor and Simovici [13] have used Genetic Algorithms as a heuristic to find median partitions. They show that their approach does better than several others among which a simple element move (or transfer) algorithm, which coincidently our algorithm has also been shown to be better than recently [1].

It would be interesting in the near future to compare the machine learning methods with our combinatorial approach.

2 Set Partitions and Median Partition Heuristics

In this paper we focus on the problem of consensus clustering in the simplest case when clusterings are considered to be set-partitions. A *set-partition*, π, of a set $\{1, 2, \ldots, n\}$, is a collection of disjunct, non-empty subsets (blocks) that cover the set completely. We denote the number of blocks in π by $|\pi|$, and label them $B_1, B_2, ..., B_{|\pi|}$. If a pair of different elements belong to the same block of π then they are co-clustered, otherwise they are not.

Our consensus clustering problem is based on similarities (or distances) between set-partitions. There exist many different distance measures to compare two set-partitions (see for example [1, 14, 15]). Some are based on pair counting, others on shared information content.

For our purposes we use a distance measure based on pair counting, known as the *symmetric difference distance*. This measure is defined on the number of co-clustered and not co-clustered pairs in the partitions. Given two set-partitions π_1 and π_2, let, a equal the number of pairs of elements co-clustered in both partitions, b equal the number of pairs of elements co-clustered in π_1, but not in π_2, c equal the number of pairs co-clustered in π_2, but not in π_1, and d the number of pairs not co-clustered in both partitions. (in other words a and d count the number of agreements in both partitions, while b and c count the disagreements). Then, the symmetric difference is defined as

$$d(\pi_1, \pi_2) = b + c = \binom{n}{2} - (a + d) \ . \tag{2}$$

This distance is a metric and is related to the *Rand Index*, $R = (a + d)/\binom{n}{2}$ and other pair-counting measures of partition similarity [16]. The measure is not corrected for chance though, i.e. the distance between two independent set partitions is non-zero on the average, and is dependent on n. A version corrected for chance exists and is related to the *Adjusted Rand Index* [17]. We note that the symmetric difference metric has a nice property that it is computable in linear time [16] which is one of the reasons why we chose it (the other is the fast update as described later). The adjusted Rand index is given by a complex formula, and although it can be also computed in linear time, we are not aware of a fast update scheme for it. We will use the Adjusted Rand in Sec. 3 where the algorithm complexity is not an issue.

The *consensus clustering problem* on set-partitions is known as the *median partition problem* [18].

MEDIAN PARTITION (MP): *Given k partitions, π_1, $\pi_2, \ldots$, π_k, find a median partition π that minimizes*

$$S = \sum_{i=1}^{k} d(\pi_i, \pi) \ . \tag{3}$$

When $d(.,.)$ is the symmetric difference distance, *MP* is known to be NP-complete in general [19, 20].

In the next sections we describe and compare two heuristics for the median partition problem. In Sec. 2.1 we present briefly a heuristic we have developed recently based on a simulated annealing optimizer for the sum of distances and fast one element move updates to explore the space of set-partitions, that was demonstrated to work well on large instances of MP. In Sec. 2.2 we describe a popular median partition heuristic based on a parametric clustering of the distance matrix derived from counts of pairwise co-clusteredness of elements in all partitions. Our goal is to compare these two heuristics. We discuss their implementation and performance in Sec. 2.3.

The following matrix is of importance for the exposition in the next sections, and represents a useful summary of the given k set-partitions. The *consensus* or *agreement* matrix $A_{k\times k}$, is defined as $a_{ij} = \sum_{1\leq p\leq k} r_{ijp}$, where $r_{ijp} = 1$ if i and j are co-clustered in partition π_p, and 0 otherwise. The entries of A count the number of times i and j are co-clustered in all k set-partitions, and are between 0 and k. Note that A can be calculated in $O(n^2 k)$ time which is manageable even for large n and k.

2.1 Simulated Annealing with One Element Moves

One element moves between clusters can transform one set-partition into any other and thus can be used to explore the space of all set-partitions. A candidate median partition can be progressively refined using one element moves. We have shown [1] that the sum of distances can be updated fast after performing a one element move on the median partition, under the symmetric difference metric:

Lemma 1. *Moving an element x in the consensus partition π from cluster B_a to cluster B_b, decreases the sum of distances by*

$$\Delta S = \sum_{\substack{\text{all } i \in B_a \\ i \neq x}} (k - 2a_{xi}) - \sum_{\substack{\text{all } j \in B_b \\ j \neq x}} (k - 2a_{xj}) \ . \tag{4}$$

Thus, once A has been calculated, updating S after moving x into a different block, takes $O(|B_a| + |B_b|)$ steps. We used this result to design a *simulated annealing* algorithm for the median partition problem. We minimize the function S (the sum of distances), and explore the solution space with a transition based on random one element moves to a random block. The algorithm is summarized below, and more details are available in [1].

Algorithm 1 Simulated Annealing, One element Move (SAOM)
Given k set-partitions π_p, $p = 1 \ldots k$, of n elements,

```
1. Pre-process the input set-partitions to obtain $A_{k \times k}$
2. "Guess" an initial median set-partition $\pi$
3. Simulated Annealing
   – Transition: Random element $x$ and a random block
   – Target Function: $S$, the sum of distances
```

2.2 Quota Rule

The counts a_{ij} are useful because they measure the strength of co-clusteredness between i and j, over all partitions (or experiments). From these counts we can define distances between every pair i and j as: $m_{ij} = 1 - a_{ij}/k$, i.e. the fraction of experiments in which the two genes are not co-clustered. The distance measure m_{ij} is easily shown to be a metric (it follows from the fact that $1 + r_{acp} \geq r_{abp} + r_{bcp}$, for any $0 \leq p \leq k$). Because of that, the matrix M has some very nice properties that allow for visual exploration of the commonness of the clusterings it summarizes. Using this matrix as a tool for consensus clustering directly (including the visualization) is addressed in a recent study by Monti et al. [11].

The *quota rule* or *majority rule* says that elements i and j are co-clustered in the consensus clustering if they are co-clustered in sufficient number of clusterings, i.e. $a_{ij} \geq q$, for some q, usually $q \geq k/2$. This rule can be interpreted in many different ways, between two extremes, which correspond to single-link and complete-link hierarchical clustering on the distance matrix M. In general however any non-parametric clustering method could be utilized to cluster the n elements.

In [12] the quota rule has been used with a version of Average-Link Hierarchical Clustering known as Unweighted Pair Group Method with Arithmetic mean (UPGMA) as a heuristic for the median partition problem. Monti et al. [11] use M with various hierarchical clustering algorithms for their consensus clustering methodology.

Here we use the following algorithm for the quota rule heuristic:

Algorithm 2 Quota Rule (QR)
Given k set-partitions π_p, $p = 1 \ldots k$, of n elements,

```
1. Pre-process the input set-partitions to obtain $A_{k \times k}$
2. Calculate the clustering distance matrix $m_{ij} = 1 - a_{ij}/k$
3. Cluster the $n$ elements with UPGMA to obtain the consensus clustering
```

The complexity of this algorithm is on the order of the complexity of the clustering algorithm. This of course depends on the implementation, but the worst case UPGMA complexity is known to be $O(n^2 logn)$.

2.3 Comparison of the Heuristics

The SAOM heuristic was found to be very fast on instances of thousands of genes and hundreds of set-partitions. It also clearly outperformed, in terms of the quality of the consensus, the greedy heuristic of improving the sum by moving elements between blocks until no such improvement exists [1]. In the same study SAOM performed well in retrieving a consensus from a noisy set of initial partitions.

Here we compare the performance of SAOM with that of the Quota Rule heuristic. Their performance is tested on 8 data sets, five of artificial and three of real data. The measure of performance that we use is the average sum of distances to the given set-partitions, i.e. $Avg.SOD = S_{min}/(k\binom{n}{2})$, which is a measure of quality of the consensus clustering [1].

Since both heuristics are independent on k (the number of set-partitions given initially) we generated the same number of set partitions, $k = 50$, for the random data sets. The number of elements in the set-partition is $n = 10, 50, 100, 200, 500$ respectively. The three real data sets, *ccg*, *yst*, and *php*, are described in Section 4.1. The results are shown in Table 1, where the values are the $Avg.SOD$.

Table 1. Tests on eight data sets show SAOM yields better consensuses

ID	n	k	$QR(0.5)$	$QR(0.75)$	$SAOM$
ccg	541	173	0.141	0.145	0.139
yst	541	73	0.122	0.113	0.107
php	541	21	0.219	0.225	0.220
R_1	10	50	0.139	0.142	0.138
R_2	50	50	0.065	0.062	0.061
R_3	100	50	0.050	0.040	0.036
R_4	200	50	0.266	0.250	0.231
R_5	500	50	0.400	0.370	0.351

The SAOM was averaged over 20 runs. The QR depends on the UPGMA threshold. We used two different quotas, $a_{ij} > 0.5$, and $a_{ij} > 0.75$ which translates into $m_{ij} < 0.5$ and $m_{ij} < 0.25$, respectively.

Both heuristics ran in under a minute for all examples. In all but one of the 16 comparisons SAOM did better than QR. Our conclusion is that SAOM does a good job in bettering the consensus, and most often better than the Quota Rule.

3 Refining the Consensus Clustering

Although our algorithm always yields a candidate consensus clustering it is not realistic to expect that it will always be representative of all set-partitions. The parallel is with averaging numbers: the average is meaningful as a representative only if the numbers are close to each other.

To get the most out of it the consensus clustering should summarize close groups of clusterings, as consensus on "tighter" clusters would be much more representative than on "looser" ones. We describe next how to break down a group of clusterings into smaller, but tighter groups. Effectively this means clustering the given clusterings.

As in any clustering we begin by calculating the distance between every pair of set partitions. We use the Adjusted Rand Index (ARI) as our measure of choice, which is Rand corrected for chance [17]:

$$R_a = \frac{a+d-n_c}{\binom{n}{2} - n_c}\ , \text{ where } n_c = \frac{(a+b)(a+c)+(c+d)(b+d)}{\binom{n}{2}}\ . \tag{5}$$

Here a, b, c, and d are the pairs counts as defined in Sec. 2. The corrective factor n_c takes into account the chance similarities between two random, independent set-partitions.

Since ARI is a similarity measure we use $1 - R_a$ as the distance function. (Actually, as defined R_a is bounded above by 1 but is not bounded below by 0, and can have small negative values. In reality though it is easy to correct the distance matrix for this). The ARI has been shown to be the measure of choice among the pair counting partition agreement measures [21].

The clustering algorithm we use is average-link hierarchical clustering. We had to choose a non-parametric clustering method because we do not have a representative, or a median, between the clusters (the consensus is a possible median, but it is still expensive to compute). The distance threshold we chose was 0.5, which for R_a represents a very good match between the set-partitions [17].

Algorithm 3 **Refined consensus clustering(RCC)**

```
1. Calculate the distance matrix $d(\pi_1, \pi_2) = 1 - R_a(\pi_1, \pi_2)$
2. Cluster (UPGMA) the set-partitions into clusters $K_1, K_2, \ldots, K_m$ with a
   threshold of $\tau = 0.5$.
3. Find the consensus partition for each cluster of set-partitions, i.e. $Cons_l =$
   $SAOM(K_l), 1 \leq l \leq m$
```

Note that this is the reverse of what's usually done in clustering: first one finds medians then cluster around them.

In the next section we illustrate this method.

4 Integrating Heterogeneous Data Sets

We are interested in genomic data of yeast since its genome has been sequenced fully some time ago, and there is a wealth of knowledge about it.

The starting point is to obtain non-trivial clusterings for these data sets. In the following we do not claim that our initial clusterings are very meaningful. However, we do expect, as we have shown before that the consensus clustering will be successful in sorting through the noise and mis-clusterings even when the fraction of mis-clustered elements in individual experiments is as high as 20% [1].

4.1 Available Data and Initial Clustering

For this study we will use three different data sets, two of microarray data and one of phylogeny data.

There are many publicly available studies of the expression of all the genes in the genome of yeast, most notably the data sets by Cho et al. [22] and Spellman et al. [23], known jointly as the *cell cycling genes* (**ccg**) data. The data set is a matrix of 6177 rows (genes) by 73 columns (conditions or experiments). Another multi-condition experiment, the *yeast stress* (**yst**), is the one reported in Gasch et al. [24], where the responses of 6152 yeast genes were observed to 173 different stress conditions, like sharp temperature changes, exposure to chemicals, etc.

There are many more available microarray data that we could have used. A comprehensive resource for yeast microarray data is the Saccharomyces Genome Database at Stanford (http://genome-www.stanford.edu/Saccharomyces/). Microarray data of other organisms is available from the Stanford Microarray Database, http://genome-www5.stanford.edu/MicroArray/SMD/, with $\approx$ 6000 array experiments publicly available (as of Jan 27, 2004).

Another type of data that we will use in our studies are phylogenetic profiles of yeast ORFs. One such data set, (**php**), comes from the Gerstein lab at Yale (available from: *http://bioinfo.mbb.yale.edu/genome/yeast/cluster/profile/sc19-rank.txt*). For each yeast ORF this data set contains the number of significant hits to similar sequence regions in 21 other organisms' genomes. A significant hit is a statistically significant alignment (similarity as judged by PSI-BLAST scores) of the sequences of the yeast ORF and another organism's genome. Thus to each ORF (out of 6061) there is associated a vector of size 21.

We clustered each of the 267 total experiments into 10 clusters by using a one dimensional version of K-means.

4.2 Results

Here we report the results of two different studies. In the first one we integrated the two microarray data sets, **ccg** and **yst**. Some of the clusters are shown in Table 4.2. It is evident that similar experimental conditions are clustered together.

The consensus clusterings are also meaningful. A quick look in the consensus of cluster 4 above shows that two genes involved in transcription are clustered together, YPR178w, a pre-mRNA splicing factor, and YOL039w, a structural ribosomal protein.

In our previous study [1] we mentioned a negative result upon integrating the 73 **ccg** set-partitions together with the 21 **php** set-partitions from the phylogeny data. The results showed that we would not benefit from such an integration, because the $Avg.SOD = 0.3450$, was very close to the Avg. SOD of a set of random set-partitions of 500 elements. In our second study we show that we can integrate such data sets more meaningfully by clustering the set partitions first. After clustering the set-partitions, with varying numbers of clusters from 3 to 10 the set-partitions from the two data sets **php** and **ccg** were never clustered

Table 2. Example Consensus Clusters of **ccg** and **yst**

Cluster 4 (Avg. SOD = 0.1179)	Cluster 5 (Avg. SOD = 0.1696)
Heat Shock 05 minutes hs-1	2.5mM DTT 015 min dtt-1
Heat Shock 10 minutes hs-1	steady-state 1M sorbitol
Heat Shock 15 minutes hs-1	aa starv 4 h
Heat Shock 20 minutes hs-1	aa starv 6 h
Heat Shock 30 minutes hs-1	YPD 4 h ypd-2
Heat Shock 40 minutes hs-1	galactose vs. reference pool car-1
Heat Shock 000 minutes hs-2	glucose vs. reference pool car-1
Heat Shock 000 minutes hs-2	mannose vs. reference pool car-1
Heat Shock 015 minutes hs-2	raffinose vs. reference pool car-1
"heat shock 17 to 37, 20 minutes"	sucrose vs. reference pool car-1
"heat shock 21 to 37, 20 minutes"	YP galactose vs reference pool car-2
"heat shock 25 to 37, 20 minutes"	Heat Shock 005 minutes hs-2
"heat shock 29 to 37, 20 minutes"	elu0

together. This is an interesting result in that it indicates that the phylogenetic profiles data gives us completely independent information from the cell cycling genes data. Similarly integrating **yst** with **php** yielded clusters which never had overlap of the two data sets. The conclusion is that the negative result from our previous study is due to the independence of the **php** vs both **ccg** and **yst**, which is useful to know for further study. Our clusters are available from *http://www.cs.ucdavis.edu/~filkov/integration.*

5 Microarray Data Imputation

Microarrays are characterized by the large amounts of data they produce. That same scale also causes some data to be corrupt or missing. For example, when trying to integrate *ccg* and *yst* we found that out of the ≈ 6000 shared genes only about 540 had no missing data in both sets. Experimenters run into such errors daily and they are faced with several choices: repeat the experiment that had the missing value(s) in it, discard that whole gene from consideration, or impute the data computationally. The idea behind data imputation is that since multiple genes exhibit similar patterns across experiments it is likely that one can recover, to a certain degree, the missing values based on the present values of genes behaving similarly.

Beyond simple imputation (like substituting 0's, or the row/column averages for the missing elements) there have been several important studies that have proposed filling in missing values based on local similarities within the expression matrix, based on different models. A notable study that was first to produce useful software, (*KNNimpute*, available at *http://smi-web.stanford.edu/projects/helix/pubs/impute/*) is [5]. More recent studies on missing data imputing are [25, 26].

We use the consensus clustering methodology to impute missing values. We demonstrate our method on microarray gene expression data by comparing it to the behavior of KNNimpute. Intuitively, the value of all the genes in a cluster in the consensus have values that are close to each other. A missing value for a gene's expression in a particular experiment is similar to having an erroneous value for the expression. In all likelihood, the gene will therefore be mis-clustered in that experiment. However, in other experiments that same gene will be clustered correctly (or at least the chance of it being mis-clustered in multiple experiments is very small). Thus identifying the similar groups of experiments and the consensus clustering between them gives us a way to estimate the missing value. We do that with the following algorithm.

Algorithm 4 Consensus Clustering Impute (CCimpute)

```
1. While clustering initially (within each experiment), for all $i$ and $j$ s.t. $v_{ij}$ is missing: place gene $i$ in a singleton cluster in the $j$-th set partition
2. Cluster (UPGMA) the set-partitions under the $R_a$ measure into clusters $K_1, K_2, \dots, K_m$, with threshold $\tau = 0.5$
3. Find the consensus partition for each cluster of set-partitions, i.e. $Cons_l = SAOM(K_l), 1 \leq l \leq m$
4. Estimate a missing $v_{ij}$ as follows: Let experiment $j$ be clustered in cluster $K(j)$, and let gene $i$ be clustered in cluster $C(i)$ in $Cons_{K(j)}$. Then $\hat{v_{ij}} = \sum_{x \in C(i)} v_{xj} / |C(i)|$
```

In other words, the missing value v_{ij} is estimated as the average value of the genes' expressions of genes co-clustered with i in the consensus clustering of the experiment cluster that contains j.

We compare the performance of our algorithm to that of KNNimpute. KNNimpute was used with 14 neighbors, which is a value at which it performs best [5]. We selected randomly 100 genes with no missing values from the **ccg** experiment above. Thus, we have a complete 100×73 matrix. Next, we hid at random 5%, 10%, 15%, and 20% of the values. Then, we used the normalized RMS (root mean square) error to compare the imputed to the real matrix.

The normalized RMS error, following [5] is defined as the RMS error normalized with respect to the average value of the complete data matrix. Let n_r be the number of rows in the matrix (i.e. number of genes) and n_c be the number of columns (i.e. experiments), and v_{ij} the experimental values. Then

$$NRMS = \frac{\sqrt{\frac{1}{n_r n_c} \sum_{i=1}^{n_r} \sum_{j=1}^{n_c} (v_{ij} - \hat{v_{ij}})^2}}{\frac{1}{n_r n_c} \sum_{i=1}^{n_r} \sum_{j=1}^{n_c} v_{ij}} \quad . \tag{6}$$

The results are shown in Fig. 1. For reference a third line is shown corresponding to the popular method of imputing the row averages for the missing values. Although our algorithm is not better than KNNimpute it does do well enough for the purposes of dealing with missing values, as compared to the row-average method. Row-average on the other hand does similarly as column-average, and both do better than imputing with all zeros [5].

It is evident that the CCimpute values vary more than the results of the other methods. This may be a result of insufficient data (100 genes may be a small number). Further studies are needed to evaluate this issue.

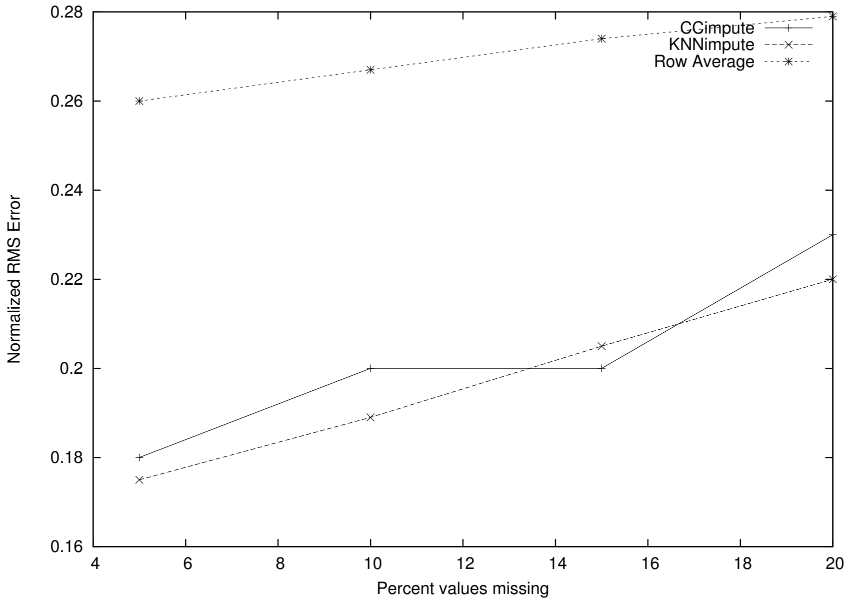

Fig. 1. Comparison of KNNimpute and CCimpute and Row Average

6 Discussion and Future

In this paper we extended our recent combinatorial approach to biological data integration through consensus clustering. We compared our best heuristic to the Quota rule with favorable results. We showed that our methods can be made more useful and biologically relevant by clustering the original clusterings into tight groups. We also demonstrated that this method can deal with missing data in the clustering.

The conclusion is that the median partition problem with the symmetric difference distance is a general method for meta-data analysis and integration, robust to noise, and missing data, and scalable with respect to the available experiments. The actual SAOM heuristic is fast (real time), and reliable, applicable to data sets of thousands of entities and thousands of experiments.

At this time we are working in two specific directions. On one hand we are trying to develop faster update methods for the Adjusted Rand measure, which is, we believe, a better way to calculate similarities between partitions. At this point our Adj. Rand methods are a thousand time slower than those on the symmetric difference metric, and hence not practical. On the other hand we are developing an integrated environment for visual analysis and integration of different large-scale data sets, which would appeal to practicing experimenters.

There is also some room for improvement of our imputation method that should make it more competitive with KNNimpute. One idea is to weigh the experimental values v_{ij}'s, which contribute to the estimated missing values, by determining how often they are co-clustered with gene i in the consensus over multiple runs of SAOM. This would effectively increase the contribution of the observed values for genes that are most often behaving as the gene whose value is missing and thus likely improve the estimate for it.

References

1. Filkov, V., Skiena, S.: Integrating microarray data by consensus clustering. In: Proceedings of Fifteenth International Conference on Tools with Artificial Intelligence, IEEE Computer Society (2003) 418–426
2. Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Science **85** (1998) 14863–8
3. Mewes, H., Hani, J., Pfeiffer, F., Frishman, D.: Mips: a database for genomes and protein sequences. Nucleic Acids Research **26** (1998) 33–37
4. Tatusov, R., Natale, D., Garkavtsev, I., Tatusova, T., Shankavaram, U., Rao, B., Kiryutin, B., Galperin, M., Fedorova, N., Koonin, E.: The cog database: new developments in phylogenetic classification of proteins from complete genomes. Nucleic Acids Res **29** (2001) 22–28
5. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.: Missing value estimation methods for dna microarrays. Bioinformatics **17** (2001) 520–525
6. Hammer, J., Schneider, M.: Genomics algebra: A new, integrating data model, language, and tool for processing and querying genomic information. In: Prooceedings of the First Biennial Conference on Innovative Data Systems Research, Morgan Kaufman Publishers (2003) 176–187
7. Marcotte, M., Pellegrine, M., Thompson, M.J., Yeates, T., Eisenberg, D.: A combined algorithm for genome wide prediction of protein function. Nature **402** (1999) 83–86
8. Pavlidis, P., Weston, J., Cai, J., Noble, W.: Learning gene functional classifications from multiple data types. Journal of Computational Biology **9** (2002) 401–411
9. Troyanskaya, O., Dolinski, K., Owen, A., Altman, R., Botstein, D.: A bayesian framework for combining heterogeneous data sources for gene function prediction (in s. cerevisiae). Proc. Natl. Acad. Sci. USA **100** (2003) 8348–8353
10. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining partitionings. In: Proceedings of AAAI, AAAI/MIT Press (2002) 93–98
11. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering. Machine Learning **52** (2003) 91–118 Functional Genomics Special Issue.
12. Gordon, A., Vichi, M.: Partitions of partitions. Journal of Classification **15** (1998) 265–285
13. Cristofor, D., Simovici, D.: Finding median partitions using information-theoretical-based genetic algorithms. Journal of Universal Computer Science **8** (2002) 153–172

14. Meilâ, M.: Comparing clusterings by the variation of information. In Schölkopf, B., Warmuth, M., eds.: Proceedings of the Sixteenth Annual Conference on Learning Theory(COLT). Volume 2777 of Lecture Notes in Artificial Intelligence., Springer Verlag (2003)
15. Mirkin, B.: The problems of approximation in spaces of relations and qualitative data analysis. Information and Remote Control **35** (1974) 1424–1431
16. Filkov, V.: Computational Inference of Gene Regulation. PhD thesis, State University of New York at Stony Brook (2002)
17. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2** (1985) 193–218
18. Barthélemy, J.P., Leclerc, B.: The median procedure for partitions. In Cox, I., Hansen, P., Julesz, B., eds.: Partitioning Data Sets. Volume 19 of DIMACS Series in Descrete Mathematics. American Mathematical Society, Providence, RI (1995) 3–34
19. Wakabayashi, Y.: The complexity of computing medians of relations. Resenhas IME-USP **3** (1998) 323–349
20. Krivanek, M., Moravek, J.: Hard problems in hierarchical-tree clustering. Acta Informatica **23** (1986) 311–323
21. Downton, M., Brennan, T.: Comparing classifications: An evaluation of several coefficients of partition agreement (1980) Paper presented at the meeting of the Classification Society, Boulder, CO.
22. Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., Davis, R.: A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular Cell **2** (1998) 65–73
23. Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell **9** (1998) 3273–3297
24. Gasch, A., Spellman, P., Kao, C., Carmen-Harel, O., Eisen, M., Storz, G., Botstein, D., Brown, P.: Genomic expression programs in the response of yeast cells to environment changes. Molecular Biology of the Cell **11** (2000) 4241–4257
25. Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T., Simon, I.: Continuous representations of time series gene expression data. Journal of Computational Biology **10** (2003) 241–256
26. Zhou, X., Wang, X., Dougherty, E.: Missing-value estimation using linear and non-linear regression with bayesian gene selection. Bioinformatics **19** (2003) 2302–2307