The CRASSS plug-in for integrating annotation data with hierarchical clustering results

Eugen C. Buehler1,∗,†, Jeffrey R. Sachs1,†, Kui Shao1, Ansuman Bagchi1 and Lyle H. Ungar2

1Applied Computer Science and Mathematics, Merck Research Laboratories, P.O. Box 2000 RY84-202, Rahway, NJ 07065 0900, USA and 2Department of Computer & Information Science, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104 6389, USA

ABSTRACT

Summary: We describe an algorithm for finding the most statistically significant non-overlapping subtrees of a hierarchical clustering of gene expression data with respect to a set of secondary data labels on genes. The method is implemented as a Java plug-in for a commercial gene expression analysis program (GeneSpring).

Availability: The JAR (Java Archive) file needed to use this plug-in and instructions on its installation and use can be obtained from http://www.cis.upenn.edu/~buehler/CRASSS.html. Versions of this plug-in are available for GeneSpring 5.0.2 and GeneSpring 6.1.1, and have been tested under Windows XP.

Contact: eugen_buehler@merck.com

INTRODUCTION

Large-scale gene expression data are being generated in increasing quantities. Many analyses of these data involve the use of clustering and related methods that generate dendograms (‘trees’). Although these techniques are useful, unsupervised clustering alone does not allow researchers to test simple hypotheses about the correspondence between their expression data and secondary data sources such as prior annotations from the literature. This paper proposes an algorithm (computational recognition and analysis of statistically significant subtrees, CRASSS), which provides a statistically rigorous method for determining a minimal, complete set of such associations.

Although hierarchical clustering is an extremely popular way of visualizing gene expression data, exploration of the generated dendograms is often left to individual researchers to do by hand. Frequently, scientists will note that genes in a particular subtree have similar attributes such as their functional categorization or the presence of a particular transcription element within their promoters. For example, the seminal paper by Eisen et al. (1998) concludes that their clustering results are significant because they can identify clusters in which all or most of the characterized genes have the same functional categorization, such as ribosomal proteins or genes involved in protein degradation. The patterns observed are so striking that they are assumed to be significant. But some experiments in which categorical labels are compared to the results of clustering expression data require statistics in order to determine which results are surprising yet statistically significant. For example, some authors (Draghici et al., 2003; Tavazoie et al., 1999) have specifically considered label frequency in groups identified by gene expression versus frequency in the experimental population via a statistic based on the hypergeometric distribution. Such techniques are required to allow researchers to distinguish significant correlation from noise or simply a large multiplicity of results. It is also possible to get results that appear to be the most significant but are not, as visualization can be misleading relative to calculated statistical significance. For this reason, it is desirable to have a packaged method that can quickly evaluate the statistical significance of all subtrees in a dendogram with respect to some categorical label.

Testing the statistical significance of the distribution of a label in a subtree versus outside that subtree (in the general population) can be accomplished using Fisher’s Exact test, which tests the probability that labels conform to the hypergeometric distribution. However, if we test multiple subtrees of a hierarchical clustering for statistically significant P-values, we must apply a correction to individually calculated P-values or to our threshold of statistical significance (Levenstien et al., 2003). For this purpose, we use the Bonferroni correction, which means we multiply P-values by the total number of significance tests performed in the entire tree, in this case the number of subtrees (Shaffer, 1995; Snedecor and Cochran, 1989). This is a conservative (i.e. high) estimate of the P-value since it incorrectly
assumes independence of the subtrees. One could also use less conservative but more computationally expensive tests, such as permutation or Monte Carlo (Fisher, 1935; Goss-Tusher et al., 2001; Pitman, 1937, 1938).

Although using Fisher’s Exact test and the Bonferroni correction allows us to find statistically significant subtrees in a hierarchical clustering, we must also consider that not all such subtrees will be useful. For example, if a given subtree contains exclusively protein degradation genes, it follows that every subtree below it also contains only protein degradation genes. Rather than reporting a series of overlapping clusters to a user, which might prove difficult to interpret, it seems sensible to give the user only the most significant non-overlapping subtrees of the tree. Each subtree returned by our algorithm should have a smaller $P$-value associated with it than any of its children. Thus, our algorithm returns all subtrees which have this property but which are not contained within a subtree having this property.

**ALGORITHM**

We will now present the CRASSS algorithm for finding non-overlapping statistically significant subtrees of a hierarchical clustering. This algorithm is independent of the method used to calculate the $P$-value for the subtrees (for which we have used Fisher’s Exact test).

The input to the CRASSS algorithm is a dendrogram and a list of leaves that are ‘labeled’ (e.g. which genes have a known function). The output is a set of non-overlapping subtrees with associated $P$-values, such that each subtree is statistically significant, more significant than any of its subtrees, but has no parents with this property. This output is complete in the sense that it encompasses all of the statistically significant subtrees, and minimal in that it contains no overlaps. Though many clustering algorithms produce binary trees, this algorithm works equally well on any tree (rooted acyclic graph). It also generalizes to labels with more than two levels and to genes with multiple labels.

The CRASSS algorithm is presented below in pseudocode that proceeds as a recursive function call on subtrees, amounting to a depth first processing of the tree.

```plaintext
CRASSS(tree t)
    p ← Pvalue(t); tList ← (); smallestP ← 1; // initialize values
    foreach subtree s of t {
        child ← CRASSS(s); // call CRASSS on subtrees (depth first)
        if(child.pValue < smallestP) {smallestP ← child.pValue;}
        tList ← tList ∪ child.sssList; // add sss
        if(smallestP < p) {p ← smallestP;}
        else {tList ← {t};} // if no subtree better, only return t
        return(Pvalue = p, sssList = tList)
    }
```

Each function call on a tree, $t$, returns the smallest $P$-value of this tree and its subtrees. It also returns a list of all of the statistically significant, non-overlapping subtrees of the tree. The foreach loop makes a list of all statistically significant subtrees of $t$ by going through and finding the significant subtrees of each of $t$’s subtrees. The IF statement in this loop finds the minimal (best) $P$-value of all of the subtrees. If this minimum is less (better) than the $P$-value, $p$, of $t$, then tList already contains the list of the most significant, non-overlapping subtrees that need to be kept from $t$ and $t$’s descendants. Otherwise, $t$ is itself the most significant, non-overlapping tree from this set. It would also be possible to perform the algorithm beginning at the leaves of the tree proceeding upwards to the root, which could be done in parallel with a standard agglomerative clustering algorithm.

Figure 1 below shows values of the functions $P$value() and CRASSS() on an example tree. Marked leaves are colored white and unmarked leaves are colored black. For trees that have no subtrees, note that CRASSS() always returns a reference to the tree and its $P$-value. In the case of tree B1, its $P$-value is not more significant than A1, and so the resulting sssList value returned by CRASSS(B1) is a concatenation of the sssList values returned by CRASSS(A1) and CRASSS(A2). In the case of tree B2, its $P$-value is less than (and thus more significant) any of its subtrees, and thus CRASSS(B2) returns a reference to B2 and its associated $P$-value. The final complete list of trees returned by CRASSS(C1) is \{A1,A2,B2\}. The subtrees returned by the CRASSS program might be only \{A1\} or only \{A1,B2\}, depending on the cut-off $P$-value chosen for reporting.

**IMPLEMENTATION**

The CRASSS algorithm was implemented as a Java plugin to the commercial software GeneSpring. The CRASSS plugin is run after the user has selected a tree (hierarchical clustering) and a gene list (the ‘labeled’ genes) within GeneSpring. After the user runs CRASSS she can visualize the statistically significant subtrees and (in a separate window) the $P$-value for each statistically significant subtree.

Figure 2 is an example of the interface for CRASSS output. RNA was isolated from 15 liver samples from human organ donors and probed with an Affymetrix DNA microarray according to the manufacturer’s recommended protocol (Lockhart et al., 1996). Those 630 genes which varied by more than a factor of 2 between at least one pair of samples as measured by the SAFER algorithm (Holder et al., 2004) were clustered using the Genespring tree algorithm. In order to assess the relationship between common patterns of hepatic gene expression and genes of known classes, CRASSS was run on this tree with a number of sets of known genes.
Fig. 1. Example of values obtained from running CRASSS. White and black bars indicate marked and unmarked leaves, respectively. As illustrated here, CRASSS does not require trees to be binary.

Fig. 2. (a) The 630 genes clustered by expression level. The two subtrees found by CRASSS using a list of all 44 CYP genes in the data are colored on the bottom. Unlabeled genes are shown in gray. (b) A close up of the region containing those two trees. The Bonferroni-corrected \( P \)-values for the blue and red subtrees CYPSSS1 and CYPSSS2 are \( 2.18 \times 10^{-11} \) and 0.0128, respectively. Leaves corresponding to CYP genes are colored blue for CYPSSS1, red for CYPSSS2 and gray for CYP genes not part of any statistically significant subtree. The four ‘genes’ in CYPSSS2 are actually different probe sets for the same gene, which explains their close ‘co-regulation’. The unlabeled gene in CYPSSS1 is one of unknown function but is now seen to be potentially of interest in the drug metabolism field due to its statistically verified co-expression with the CYP genes.

CONCLUSION

Although the application we have discussed in this paper is limited to hierarchical clustering of genes, CRASSS could easily be applied in any situation in which researchers wish to incorporate secondary data labels into the analysis of a hierarchical clustering. This could include clustering compounds (Weinstein et al., 1997), species strains (Kaminski et al., 2000; Primig et al., 2000), cell-lines (Kahn et al., 1998) or tissues (Alon et al., 1999; Welsh et al., 2001). This could be particularly advantageous because several of these domains have more thorough, pre-existing labeling annotation data than genes.

ACKNOWLEDGEMENTS

Thanks to Dan Holder for many helpful discussions and SAFER processing of liver data, and Alex Elbrecht, John Thompson, David Gerhold and Jian Xu for helpful discussions about data analysis of Affymetrix data. Thanks to Karen Richards, Mona Parikh and Tom Rushmore for the liver data, Fred Guengerich (Department of Biochemistry and Center in Molecular Toxicology, Vanderbilt University School of Medicine) for the liver tissue samples, and to Jeff Yuan for bioinformatics advice. Thanks to Matthew Wiener, Arthur Fridman, Vladimir Svetnik, Reid Huntsinger, Richard Blevins, Jeff Saltzman, Robert Nachbar, Paul Liberator, and the journal’s reviewers for helpful comments and discussions.

REFERENCES

Integrating annotation and clustering results


