Capturing Best Practice for Microarray Gene Expression Data Analysis

Gregory Piatetsky-Shapiro
KDnuggets
gps@kdnuggets.com

Tom Khabaza
SPSS
tkhabaza@spss.com

Sridhar Ramaswamy
MIT / Whitehead Institute
sridhar@genome.wi.mit.edu

ABSTRACT

Analyzing gene expression data from microarray devices has many important applications in medicine and biology, but presents significant challenges to data mining. Microarray data typically has many attributes (genes) and few examples (samples), making the process of correctly analyzing such data difficult to formulate and prone to common mistakes. For this reason it is unusually important to capture and record good practices for this form of data mining. This paper presents a process for analyzing microarray data, including pre-processing, gene selection, randomization testing, classification and clustering; this process is captured with "Clementine Application Templates". The paper describes the process in detail and includes three case studies, showing how the process is applied to 2-class classification, multi-class classification and clustering analyses for publicly available microarray datasets.

Keywords

microarrays, gene expression, data mining process,
application template, Clementine.

Categories

H.2.8 Database Applications - Data Mining, I.5.2 Design Methodology - classifier design and evaluation.

1. MICROARRAYS: AN OVERVIEW

Cells in the same organism normally have the same genes, but these genes can be expressed differently, i.e. manufacture different messenger RNA or mRNA, which in turn manufacture different proteins, allowing creation of a huge variety of different cells. Virtually all differences in cell state or type are correlated with changes in the mRNA levels of many genes. Detection and cure of many diseases can be assisted by understanding gene expression in human and animal tissues and cells.

Microarray chips measure the expression levels of many genes simultaneously. There are several different types of microarrays, including

- Short oligonucleotide arrays (made by Affymetrix);
- cDNA or spotted arrays (originated by Pat Brown lab at Stanford);
- Long oligonucleotide arrays (Agilent Inkjet);
- Fiber-optic arrays.

Different types of microarray use different technologies for measuring RNA expression levels; detailed description of these technologies is beyond the scope of this paper. Here we will focus on the analysis of data from Affymetrix arrays, which are currently the most popular commercial arrays. However, the methodology for analysis of data from other arrays would be similar, but would use different technology-specific data preparation and cleaning steps.

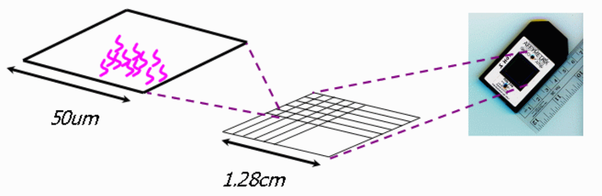


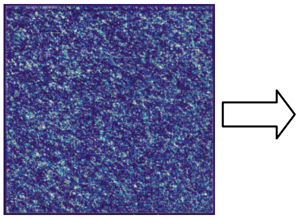
Figure 1: Affymetrix GeneChip® (right), its grid (center) and a cell in a grid (left).

This type of microarray is a silicon chip that can measure the expression levels of thousands of genes simultaneously. This is done by hybridizing a complex mixture of mRNAs (derived from tissue or cells) to microarrays that display probes for different genes tiled in a grid-like fashion. Hybridization events are detected using a fluorescent dye and a scanner that can detect fluorescence intensities. The scanners and associated software perform various forms of image analysis to measure and report raw gene expression values. This allows for a quantitative readout of gene expression on a gene-by-gene basis. As of 2002, microarrays such as the Affymetrix U133 2-chip set, can measure expression of over 30,000 genes, likely a majority of the expressed human genome.

Microarrays have many potential applications, including:

- More accurate disease diagnosis from gene expression levels;
- Predicting treatment outcome;
- Tailoring drug therapy based on gene expression levels (pharmacogenomics);
- Drug discovery and toxicology studies;
- Assisting fundamental biological discovery.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.



Gene	*Value*
D26018_at	193
D26067_at	-70
D26068_at	144
D26129_at	33

Figure 2: An example raw microarray image for a single sample (image courtesy of Affymetrix). Brightness of dots represents the intensity of genes expressed. The image on the left is translated by microarray software into numbers like the ones on the right

Microarray data analysis is fast becoming an essential tool in biomedical research.

The main types of data analysis needed to support the above applications include:

- Gene Selection – in data mining terms this is a process of attribute selection, which finds the genes most strongly related to a particular class.
- Classification – here we classify diseases based on gene expression patterns, predict outcomes and/or select the best treatment.
- Clustering – using clustering we can find new biological classes and refine existing ones.

2. DESCRIBING A KNOWLEDGE DISCOVERY PROCESS WITH CLEMENTINE APPLICATION TEMPLATES

To get the best results from data mining in any field it is advisable to use a well-defined process and to adopt best practices for the task at hand. In this section we outline the CRISP-DM standard data mining process model, and show how best practices for a given application can be captured in a CRISP-DM framework using a visual programming environment.

2.1 CRISP-DM – a CRoss-Industry Standard Process model for Data Mining

Now widely adopted as a standard process model for data mining, CRISP-DM [3,16] offers many benefits to data mining practitioners; one of these is to provide a standard high-level structure and terminology for the data mining process. If we wish to capture best practice for a particular data mining application, we must describe the process in detail, and CRISP-DM gives us a framework for doing so.

Figure 3 shows the "phases" of the CRISP-DM process and the main information flows between them; these phases provide a high-level classification of data mining activities, such as "data understanding", "data preparation" and "modeling", to help us describe data mining.

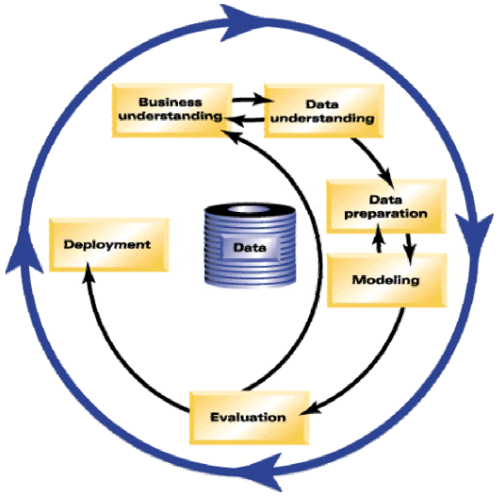


Figure 3: Phases and Information flow in CRISP-DM

2.2 Describing a Data Mining Process

For a given data mining application, best practice will specify a number of steps, for example certain data preparation and modeling steps, which are combined to solve the problem at hand. Figure 4 shows a diagram depicting how a collection of data preparation and modeling steps fit together for the task of genome classification in a multi-class problem. (In figure 4, ellipses represent operations applied to the data and rectangles represent raw data or intermediate data which is created by one step and used in another. Codes starting with P represent data preparation steps and codes starting with M represent modeling steps.)

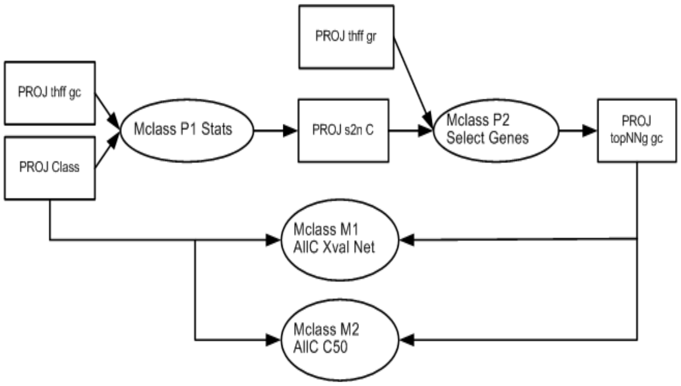


Figure 4: Data Mining Process Diagram for a Multi-Class Classification Task

A diagram such as that in figure 4 describes a "module"; a collection of modules forms an "application template", a complete description of the data mining process for one application. The steps in the process are realized using a data mining toolkit. The diagram in figure 4 uses only an informal notation, to help an analyst understand how a collection of steps and data files fit together.

2.3 Using a Data Mining Toolkit to Realize the Steps of a Data Mining Process

To capture a data mining process in sufficient detail that it can be re-used, each step must be realized in an executable form. This can be achieved using a data mining toolkit based on visual programming, such as Clementine [6,12], depicted in figure 5.

In figure 5, Clementine's main drawing area contains a *stream diagram*; an executable visual program which performs one of the modeling steps in figure 4. Each icon in the stream diagram depicts a low level operation in the data mining process; data flows from data sources (circular icons) through data manipulation operations (hexagonal icons) to a C5.0 modeling operation (a pentagon) and through the model (diamond-shaped) to various reports and visualizations.

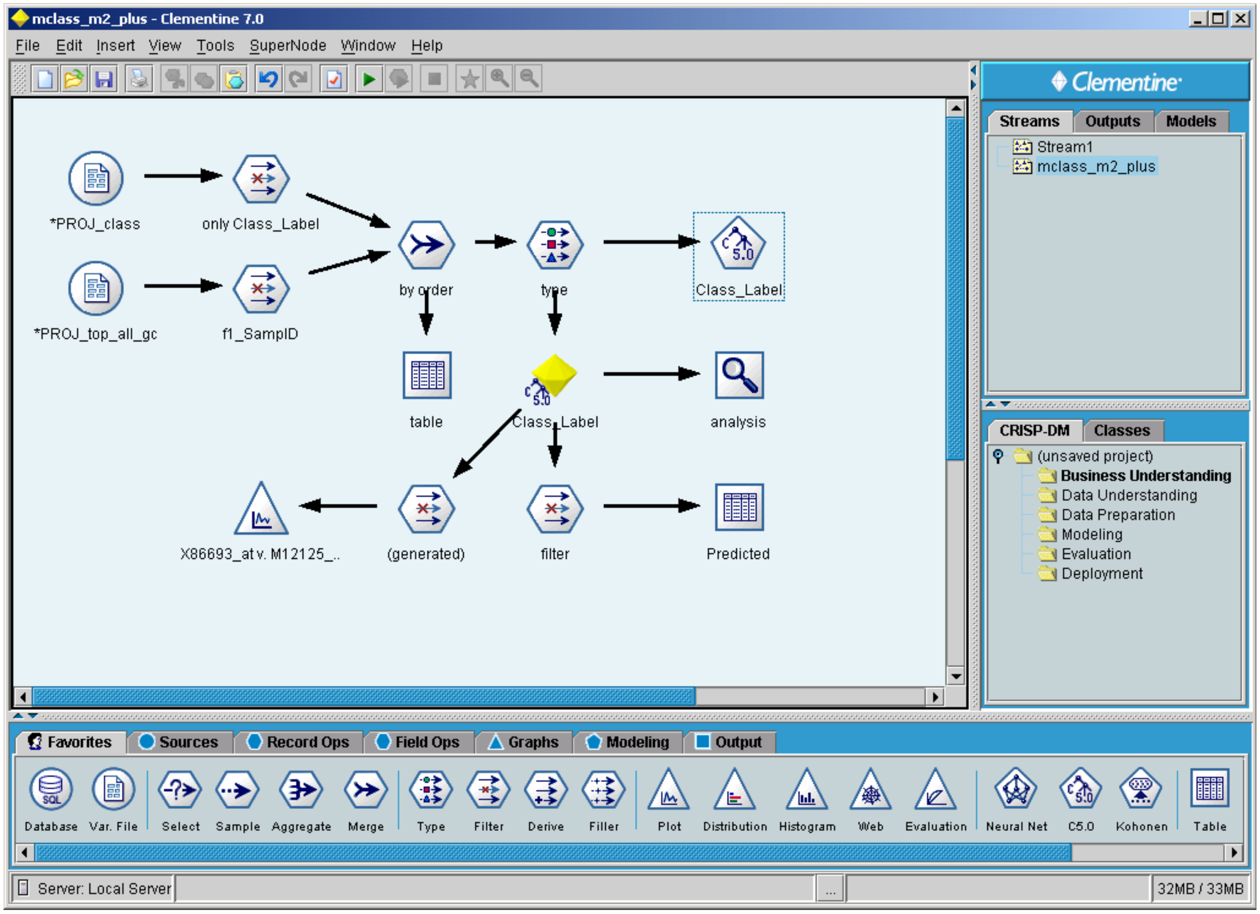
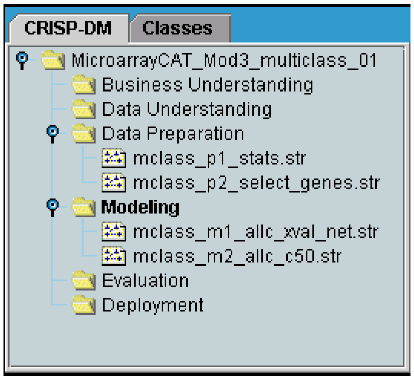


Figure 5: Clementine Visual Programming Interface & Microarray Analysis Stream Diagram

Each step in a data mining process can be realized using a stream diagram of this kind. A collection of stream diagrams and data files designed to achieve a particular goal is called a Clementine Application Template, or "CAT" [7].

The CRISP-DM process model is used to classify the streams of a CAT according to the role they play in this standard view of the data mining process. Clementine also supports the use of CRISP-DM via its CRISP-DM Project facility; figure 5 shows the streams from one module of a Microarray analysis CAT organized using this tool.

A Clementine Application Template (CAT) can be used to capture best practice for a specific data mining application. Clementine streams form the detailed realization of a process. The higher-level structure of the process is a collection of modules, each of which is depicted informally as a data mining process diagram, and the CRISP-DM framework is used to classify the steps of the process. The remainder of this paper specifies in detail the capture of best practice for microarray gene expression data analysis.


CRISP-DM Classes

MicroarrayCAT_Mod3_multiclass_01
Business Understanding
Data Understanding
Data Preparation
mclass_p1_stats.str
mclass_p2_select_genes.str
Modeling
mclass_m1_allc_xval_net.str
mclass_m2_allc_c50.str
Evaluation
Deployment

Figure 6: Microarray CAT Module 2 Streams organized as a CRISP-DM Project

3. MICROARRAY DATA ANALYSIS ISSUES

Comparing microarray data analysis with the more widespread applications of data mining, such as analytical CRM using customer data, we can see that it presents quite different challenges, for two reasons. First, the typical number of records (or samples) for microarray analysis is small – usually less than 100. This is likely to remain so for many areas of investigation, especially for human data, because of the difficulty of collecting and processing samples.

Second, the number of columns (or genes) in microarray data is normally large – typically many thousands. When building predictive models, having so many columns relative to the number of rows is likely to lead to "false positives" – gene combinations what correlate with a target variable purely by chance. This is particularly true for learning algorithms such as decision trees and neural networks, which find complex non-linear combinations of features, and therefore have a large model space in which to find spurious models.

One useful technique in this situation is to assess the likelihood of getting chance correlations by using randomization. This technique randomly permutes the class column many times, and compares the strength of correlation obtained with a randomized class column with that from the actual class column.

Another useful technique is to select a smaller number of the most promising genes, for example 100-200, and build models using only these genes. Genes can be ranked by comparing the mean expression value for each class with that of the rest, and computing measures like T-values or signal-to-noise (S2N) ratios. Models produced in this way are more accurate, and generalize better, than models produced using the complete set of available genes.

These techniques for reducing the number of columns in the, otherwise very wide, microarray data, are collectively referred to as "feature reduction".

4. MICROARRAY DATA ANALYSIS PROCESS

The process of analyzing microarray gene expression data is summarized in figure 7, which shows the major flows of data and the iterative relationship between model building and feature and parameter selection.

The process in figure 7 is that used for modeling the classification of genomes, where both gene data and class data are used – the modeling is "supervised learning". For data-sets which are too small to separate training and test data (a common situation with Microarray data), cross-validation can be used to evaluate the likely quality of a model. Where the modeling is unsupervised, no separation of training and test data is required, and feature selection based on class is explicitly excluded because the purpose of the exercise is to discover new classes.

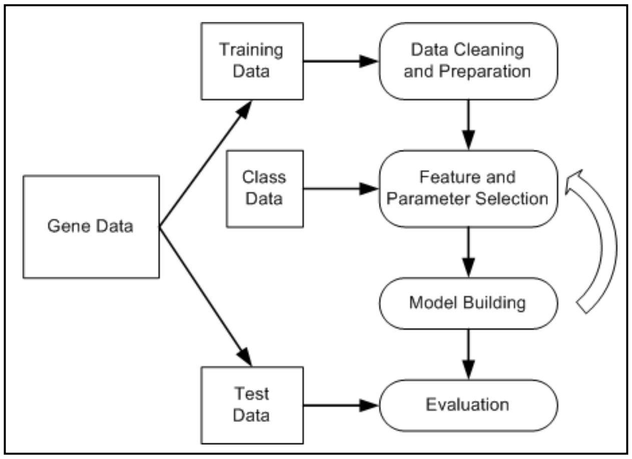


Figure 7: Microarray Classification Process Overview

4. 1 Microarray Data Cleaning and Preparation

At a very high level, the goal of data preparation in microarray data analysis is the same as for all data mining, that is to transform the data to make it suitable for analysis and to aid in producing the best possible models. However the unusual properties of microarray data give a special character to this phase of the data mining process; in this context data preparation takes place in two stages.

The first stage could be characterized as "pure" data preparation. This stage contains those aspects of data preparation which are independent of any class data; these are thresholding, normalization and filtering. The thresholding and filtering operations achieve a substantial amount of feature reduction, typically by about 50%.

Thresholding and filtering are "low-level" forms of data cleaning; techniques of this kind are broadly applicable, but the details will vary with the microarray device used to produce the data. The thresholding and filtering details given in this paper therefore have the status of examples only, and are specific to the Affymetrix device.

The second stage contains those aspects which make specific use of class data, and are broadly termed "feature selection". Here we are performing data reduction by narrowing the set of features to those relevant to the specific set of classes to be analyzed. These operations are therefore specialized to the character and number of these classes, and are also not suitable for "pure" discovery, for example uses of clustering to discover new classes which may be independent of known classes.

4.1.1 Thresholding

The goal of all microarray devices is to measure the expression level of mRNA, which in Affymetrix devices is measured indirectly by comparing PM (perfect match), and MM (mismatch) probes, using 20 probes in the HuFL6800 chip, and 11 probes in the latest chip. The MAS-4 software combines the PM and MM values by subtracting MM from PM values, so it is possible for gene expression to be negative, which means that MM probes have stronger signals than PM. Since we do not know what is matching the MM probes, these signals are not useful, and for this reason, data processed with MAS-4 software is heuristically set to a minimum of 20. Affymetrix MAS-5 software does not generate negative expression values and does its own thresholding.

Studies of measurement error, which process the same sample several times [17], have shown that the measurements were reproducible above values of 100, and much less reproducible below 100. For data believed to be more noisy, a lower threshold of 100 would be appropriate.

The upper threshold is usually 16,000, because image intensities above this level tend to have a non-linear correlation with the actual expression levels.

4.1.2 Normalization

While classification algorithms can use the actual expression levels, data normalization is required for clustering.

The best results are obtained by normalizing the data to mean zero, standard deviation 1 across genes.

4.1.3 Filtering

Since many genes are not expressed at all or do not vary sufficiently to be useful, a filtering operation is usually applied before adding the class information. Typical filtering excludes genes with low variation across samples, for example

MaxValue(G) / MinValue(G) < 5 and

MaxValue(G) – MinValue(G) < 500

where MaxValue(G) and MinValue(G) are the maximum and minimum values of gene G across all samples.

4.2 Feature Selection

After data preparation and cleaning, we apply feature selection by adding class information and looking for genes that can distinguish between classes.

Most learning algorithms look for non-linear combinations of features and can easily find many spurious combinations given small number of records and large number of genes. To deal with this problem, we next reduce number of genes by a linear method, such as T-values for mean difference between classes:

$$\text{T-test for Mean Difference} = \frac{(Avg_1 - Avg_2)}{\sqrt{(s_1^2 / N_1 + s_2^2 / N_2)}}$$

or a simplified version,

$$\text{called Signal to Noise ratio (S2N)} = \frac{(Avg_1 - Avg_2)}{(s_1 + s_2)}$$

where N_K is the number of examples in class K, Avg_K is the sample mean gene expression for class K, and s_K is the sample standard deviation of gene expression for class K.

These formulas assume 2 classes. In a multi-class case, we compute these formulas for each class vs. all the other classes.

In a typical multi-class data analysis (such as Ramaswamy [11] or Pomeroy [10] which analyze different tumor types), we frequently see that some of the classes are much more clearly characterized than others. Thus, if we select genes only with the highest values of the statistic, we risk having only genes representative of only one class and not get enough genes from all classes.

To avoid this problem, we apply an additional heuristic: select an equal number of genes from each class. This is implemented by computing for each gene a separate measure rank for each class, and then ordering genes by the rank.

4.3 Dealing with False Positives

Because of the large number of columns compared to a relatively small number of samples, we are likely to get false positives, i.e. genes that appear to be correlated with the target classes but are not.

We can measure the likelihood of false positives in several ways. First, if we use T-value, we can compute the significance value for the T-value. If we can assume equal variances, number of degrees of freedom is computed as ($N_1 + N_2$ -2). However, the variances are rarely equal. For example, applying F-test to the top 200 genes with the highest T-value for the ALL/AML experiment, we find that about 80% of the genes have significantly different (at p=0.01) variance. We should then use the version of T-test for different variances. In that case, the number of degrees of freedom is estimated as [9]

$$df = \frac{(s_1^2 / N_1 - s_2^2 / N_2)}{(s_1^2 / N_1)^2 / (N_1 - 1) + (s_2^2 / N_2)^2 / (N_2 - 1)}$$

Using this formula, we can compute the significance value for the top genes. For example, among the top 200 genes for ALL/AML case, the p-value for the most significant gene is around 10^{-9} , while the p-value for the least significant gene is around 0.001.

Since we evaluate several thousand genes, we can expect that by chance at least one from a thousand will have a value at p=0.001 level.

We can test for this randomness effect by randomization, that is randomly permuting the class column a number of times. We then compute the T-value value for the mean difference for each randomization and for each gene, and compute for each gene the maximum and minimum T-value obtained.

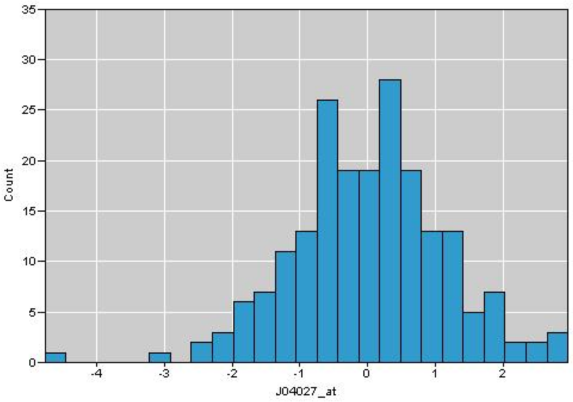


Figure 8: Randomization T values for ALL/AML data, for gene J04027 after 500 randomizations. Note that the lowest value is less than -4.

After performing 500 randomizations on the top 100 genes for each class, only about 11 genes have maximum (or minimum) T-values that exceed the actual values.

A faster, but more conservative way to compensate for multiple tests is to use Bonferroni adjustment. i.e. to divide the desired significance by the number of tests. To get genes with expression difference significant at 0.01 value, if we test 1000 genes, we would establish a threshold of 0.01/1000 = 0.00001 for each gene.

However, removing genes whose T-value falls below some arbitrary significance threshold is not the best way to get a good classifier, since we risk eliminating valuable genes.

Instead, we suggest using the wrapper approach, as described in the following sections, to select the best subset.

We note the many biologists use S2N measure for its simplicity. However, unlike the T-test, there is no obvious analytical solution for measuring the significance of S2N, so the randomization approach can be used to estimate the significance level.

More detailed analysis of randomization and assessing the significance level and false detection rate is provided by Tusher [14].

5. BUILDING CLASSIFICATION MODELS

Many different classification methods have been applied to microarray data. From biological considerations, we know that a typical diagnostic is more likely to have weak inputs from multiple genes working in parallel, rather than strong inputs from one gene. For this reason, sequential classifiers like Decision Trees, though they have been applied to microarray data, do not work well (see e.g. Dubitzky[4]), because they try to find the smallest gene sets. This also makes them less robust against false positives.

Other methods that have been applied to microarray data classification include:

- K-nearest neighbors (Pomeroy [10]) – robust for small numbers of genes;
- Support Vector Machines [1] - these seem to produce the best classification accuracy, but are hard to explain to biologists in terms of genes.

Neural Nets are more noise tolerant and are designed to handle many parallel inputs, thus well suited to genetic classification. Although they usually cannot handle thousands of genes in the full dataset, they seem to work well for reduced number of genes. We have experimented with a number of approaches available within Clementine (e.g. C5.0 and C&RT) , but obtained the best results with neural nets.

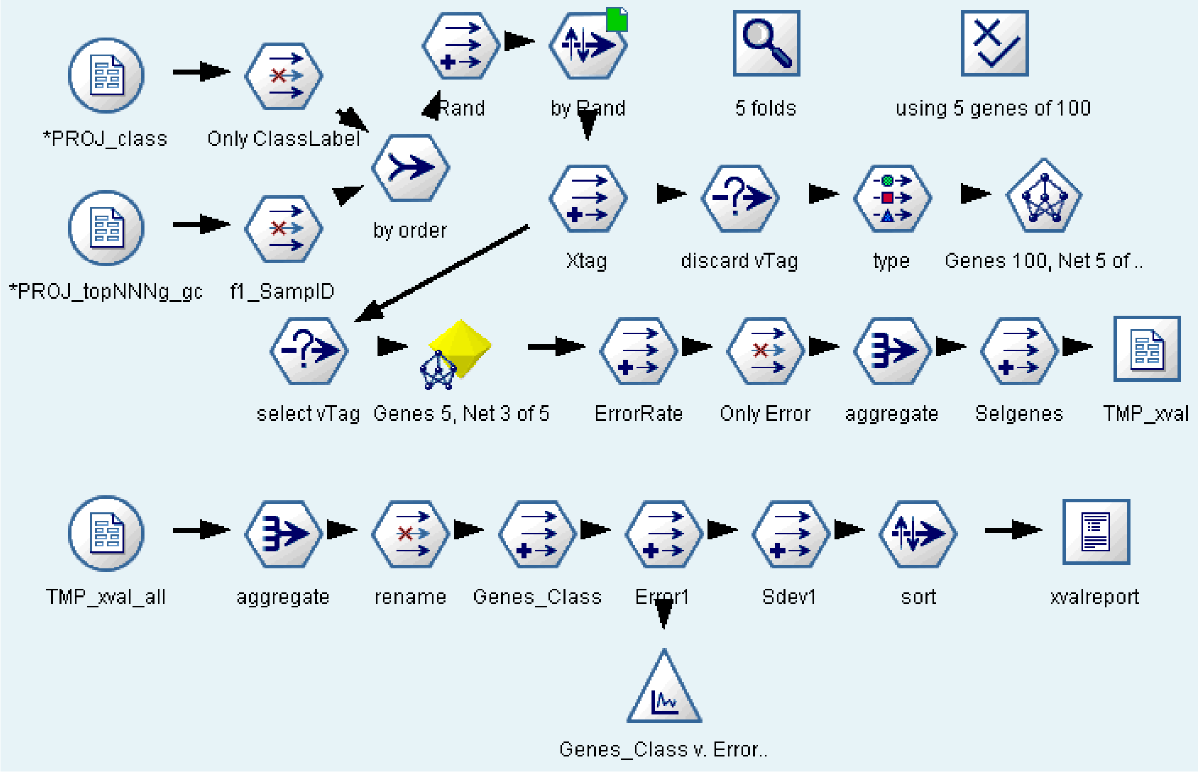


Figure 9: Stream for Subset selection by Cross-Validation

5.1 Finding a Good Gene Set Using the Wrapper Approach

Reducing the number of genes to those with sufficiently low significance level is only the beginning. Usually, we still get several hundred genes that satisfy those conditions which is still too many for most classification algorithms. We can further determine a good subset of genes by using a wrapper approach.

This approach was suggested by Kohavi & John [8] to determine the optimal set of parameters for a classification algorithm. Here we propose to use it to determine a "good" subset of genes.

For this step, we suggest selecting the top 100-200 genes per class ordered by their rank within each class. The exact number to select is not that critical, but we usually select fewer genes if the number of samples is small, and we also can select more genes if the significance testing indicates that there are many genes with significant differences.

For example, if we have 5 classes, then the first 5 genes are the top genes for each class, then we have the second best genes for each class, etc. If there are duplicate genes, we leave them in.

We next test models using 1, 2, 3, 4, 5, 6. 8, 10, 12, 16, 20, 30, 40, ... genes from each class. For each model, we perform a V-fold cross-validation. Five fold is sufficient for an initial run, while leave-one-out cross-validation is considered to give most accurate results. If we are using a randomized algorithm, such as Clementine neural nets, the results may differ from run to run, so we suggest repeating each cross-validation run at least 10 times.

6. CASE STUDY 1: ANALYZING LEUKEMIA

As an example of a 2-class classification, we used Leukemia data with 2 classes: ALL/AML (Golub, [5]), available from the MIT Whitehead Institute. This dataset has a training set with 38 samples and a separate test set of 34 samples. Both have raw Affymetrix expressions for about 7,000 genes.

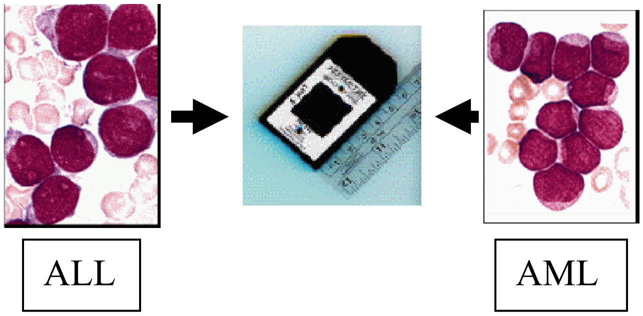


Figure 10: ALL and AML Leukemia – Visually similar, but genetically very different

First we applied a preprocessing stream to threshold gene expression values to at least 20 and at most 16,000.

Then we used a filtering stream to remove the genes where (Maximum value – minimum value) < 500 across samples and (Maximum value / minimum value) < 5 across samples. This filter has a biological motivation and helps to reduce the number of genes. We next computed the mean expression values of genes for each class, and computed T-value for the difference, assuming unequal variances. We then select the top 100 genes from each class with the highest T-value. We used randomization stream to verify that the selected genes had T-values significantly higher than those obtained by randomization.

We then selected the best gene set by building neural net models using potential subsets with 1, 2, 3, 4, 5, 10, etc genes from each class, and for each gene subset evaluated the error rate by using 10-fold cross-validation.

A good cross-validation curve will show a high error in the beginning (when number of genes is too small), with a gradual decrease as the number of genes approaches the optimum plateau, and then a gradual increase as the number of genes becomes too large and we begin to overfit the data. The best gene subset to select is the one in the middle of the plateau with the lowest error.

For ALL/AML data, the cross-validation curve shows the desired behavior, with 10 genes being at the center of the optimum plateau. We selected this gene subset to build a new neural net model on the full training data (using 70 percent as the training data) and applied it to the test data (34 samples), which up to this point has not been used in any way in model development.

Evaluation on the test data gave us 33 correct predictions out of 34 (97% accuracy), which compares well with many previously reported results on the same data.

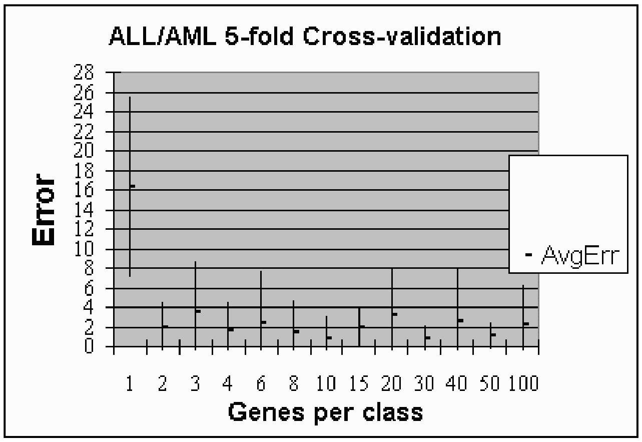


Figure 11: Cross-validation errors for different gene subsets, for ALL/AML data, each cross-validation repeated 10 times. Central point is the average error for each cross-validation, bars indicate one St, Dev up and down.

Note: the single misclassification was on sample 66 which has been consistently misclassified by other methods and is believed to be incorrectly labeled by the pathologist [13], which suggests that the classifier was actually 100% correct. These results are as good as any reported on this dataset, which was the subject of CAMDA-2000 conference and competition [2].

7. CASE STUDY 2: ANALYZING BRAIN DATA

As an example of multi-class data, we used Brain dataset A, from [10], available from the MIT Whitehead institute [15]. This dataset has 42 examples, about 7,000 genes, and 5 classes: Medulloblastoma, Malignant glioblastoma, AT/RT (rhabdoid), Normal cerebellum, and PNET.

The preprocessing and filtering was the same as for Leukemia dataset.

We selected the top genes most strongly related to each class by analyzing each class against all others and computing a signal to noise statistic (a simplified version of the T-test), which is simply the difference between average for this class and the rest, divided by sum of standard deviations for this class and the rest.

For multi-class data, one class might be expressed much more strongly than the others, so if we choose genes solely by decreasing T-values we might choose the most representative genes for one class only. To avoid this problem we use the heuristic of choosing an equal number of genes from each class

We evaluated subsets with 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, and 20 genes per class, and for each subset built neural net models using 10-fold cross-validation. We aggregated average errors per class.

The lowest *average* error, of about 15%, was obtained using 12 genes per class (60 genes in total), which is equal to the best reported results on this data [10]. Due to the small size of this dataset, we did not have a separate test set.

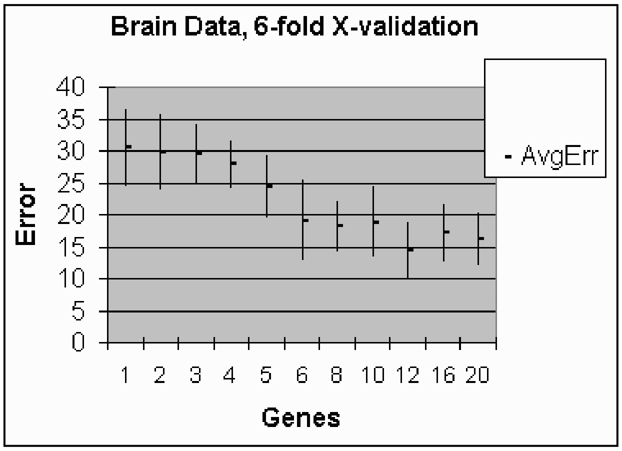


Figure 12: Cross-validation errors for different gene subsets, for multi-class brain data, each cross-validation repeated 10 times.

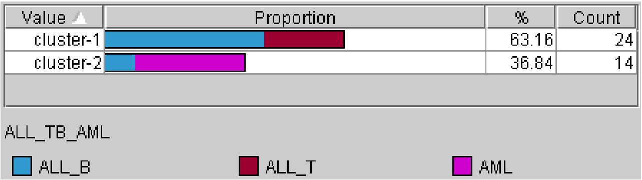
8. CASE STUDY 3: CLUSTER ANALYSIS

The goals of cluster analysis for microarray data include finding natural classes in the data, identifying new classes and gene correlations, refine existing taxonomies and supporting biological analysis and discovery.

A number of different methods have been used for clustering. We applied two clustering algorithms included in Clementine (Kohonen SOM and TwoStep) to the Leukemia data. We subdivided ALL into 2 classes: ALL-T and ALL-B. This gave us 3 natural classes: ALL-T, ALL-B, and AML, and we were interested in seeing whether the clustering could rediscover the natural classes.

Preprocessing included the same thresholding and filtering as for classification; we also normalized each sample separately to Mean = 0, Std. Deviation = 1 by subtracting from each gene expression the mean value for the sample and dividing it by the standard deviation. (It is also possible to normalize across genes, but we found that these results are harder to interpret).

We then applied both Kohonen clustering and TwoStep to the normalized data and compared the discovered classes to natural classes. When run without specifying the number of clusters, TwoStep quickly found 2 clusters, which matched well with AML/ALL division (see Figure 13).



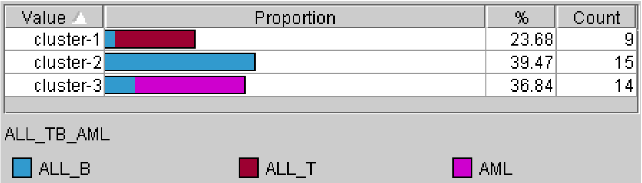
Value	Proportion	%	Count
cluster-1		63.16	24
cluster-2		36.84	14

ALL_TB_AML

🟦 ALL_B 🟥 ALL_T 🟪 AML

Figure 13: Proportion of natural classes in 2 clusters found by TwoStep algorithm

When asked to find 3 clusters, TwoStep produced clusters that matched closely the separation between AML, ALL-T, and ALL-B (see Figure 14).



Value	Proportion	%	Count
cluster-1		23.68	9
cluster-2		39.47	15
cluster-3		36.84	14

ALL_TB_AML

🟦 ALL_B 🟥 ALL_T 🟪 AML

Figure 14: Proportion of natural classes in 3 clusters found by TwoStep algorithm

Similar results were obtained with Kohonen clustering.

9. FUTURE DIRECTIONS AND CONCLUSIONS

Other types of analysis possible with Clementine include combining clinical and genetic data and performing outcome / treatment success prediction.

While this paper was focused on analysis of microarray data, many ideas presented here are applicable to other domains, such as cheminformatics or drug design, where the number of attributes is very large and much larger than the number of samples. For example, a version of these techniques was used to analyze chemical structure data with 100,000 Boolean attributes and 20,000 examples.

One promising idea for future extension is to perform cost-sensitive classification. In many domains, classification errors are not equal - for example, missing a treatable cancer is a much more serious error than missing an untreatable cancer. The C5.0 algorithm allows cost-sensitive learning that can be used to develop models that minimize cost of errors, according to a custom error cost matrix; this could be used to develop classification models which would be more useful in practice.

Clementine is also integrated with text-mining technology, which can be used to access the huge medical textual resources.

In conclusion, we emphasize that using a principled methodology can lead to good results. In this paper we have shown how to use Clementine Application Templates to obtain very good results in analyzing microarray datasets, based on best practice and organized using CRISP-DM.

10. ACKNOWLEDGMENTS

Our thanks to Pablo Tamayo for helpful suggestions and ideas.

11. REFERENCES

[1] Brown et al., Knowledge-based analysis of microarray gene expression data by using support vector machines, *PNAS* 97(1):262–267, 2000.

[2] CAMDA 2000, *Proceedings of Critical Assessment of Microarrays Conference*, Duke University, 2000.

[3] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. *CRISP-DM 1.0 Step-by-step data mining guide,* CRISP-DM Consortium, 2000, available at http://www.crisp-dm.org

[4] Dubitzky et al., Symbolic and Subsymbolic Machine Learning Approaches for Molecular Classification of Cancer and Ranking of Genes**,** in *Proceedings of CAMDA 2000*, Duke University, 2000.

[5] Golub et al., Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring, Science, vol. 286, October 1999.

[6] Khabaza, T. and Shearer, C., Data Mining with Clementine, *IEE Colloquium on Knowledge Discovery in Databases*, IEE Digest No 1995/021(B), London, February 1995.

[7] Khabaza, T. & Sigerson, D., WebCAT: the Clementine Application Template for Web-Mining and Analytical eCRM, web-mining workshop paper, *First SIAM International Conference on Data Mining*, Chicago, April 2001.

[8] Kohavi, R, John, G., Wrappers for Feature Subset Selection, *Artificial Intelligence*, 1997.

[9] *NIST Engineering and Statistics Handbook*, http:// www.itl.nist.gov/div898/handbook/eda/section3/eda353.htm

[10] Pomeroy et al., Prediction of central nervous system embryonal tumour outcome based on gene expression, Nature, vol. 415, January 2002.

[11] Ramaswamy, S. et al, Multiclass cancer diagnosis using tumor gene expression signatures, *PNAS* 98(26):15149-15154, 2001.

[12] Shearer, C. and Khabaza, T., Data Mining by Data Owners, *Intelligent Data Analysis*, Baden-Baden, Germany, August 1995.

[13] Tamayo, P., personal communication, 2002.

[14] Tusher, Tibshirani, and Chu, Significance analysis of microarrays applied to the ionizing radiation response. *PNAS* 2001 98: 5116-5121.

[15] Whitehead (MIT) Institute Cancer Genomics Publications Data Sets, http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi

[16] Wirth, R. & Hipp, J., CRISP-DM: Towards a Standard Process Model for Data Mining, in *Proc. of the 4th Int. Conf. on The Practical Applications of Knowledge Discovery and Data Mining*, Manchester UK, April 2000, The Practical Application Company.

[17] Saccone, R. A., Rauniyar, R. K. and Patti M.-E., Sources of Experimental Variability In Expression Data Derived From High-Density Oligonucleotide Microarrays: Practical Experience From An Academic Core Laboratory, *2nd Annual UMass Bioinformatics Conference*, UMass Lowell, 2002.