March 2010

Gene Expression Classification

Gregory Morgan Barrett
Worcester Polytechnic Institute

Gene Expression Classification

A Major Qualifying Project Report

submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

in

Computer Science

By

Gregory M. Barrett

Date: March 16, 2010

Approved:

Professor Carolina Ruiz, Major Advisor

Professor Elizabeth Ryder, Co-Advisor

ABSTRACT

In this project, we constructed classification models for gene expression based on association rules. Gene expression patterns from cell types in the nematode *C. elegans* were used. The promoter regions associated with these genes were gathered. Motifs were mined from this data set. Multiple methods were used to select subsets of these motifs. Classification models were built from these subsets. The performance of these models was analyzed. A novel method of selecting motifs was shown to produce the best models.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This project aims to expand upon past work in building classifiers for gene expression. There are two primary goals, to gather a new set of *C. elegans* genetic data, and to use it to construct an accurate classifier.

The genetic data was gathered using a tool called WormMart (Wormbase, 2010). Ten cell types of interest were chosen. The first 1,000 base pairs from the promoter regions of 462 unique genes contained in these cells types were collected, along with their expression information.

Motifs were mined from the promoter regions of these genes using the Weeder tool (Tompa, 2005). Mining was performed on each cell type separately. Forty motifs were mined for each cell type. In total, 322 unique motifs were found. A variety of methods were used to select a subset of these motifs to use when building a classifier.

After annotating each gene with the selected motifs, the Weka data mining tool (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) was used to build a classifier. The Apriori Sets and Sequences algorithm (Pray, 2004) was used to mine classification association rules. From these rules classifiers were constructed and their performance analyzed.

2. BACKGROUND

BIOLOGICAL BACKGROUND

DNA

DNA is the 'code of life'. Every organism's DNA contains the information it needs to perform its biological functions. DNA is organized in an anti-parallel helix. DNA consists of four nucleotides, Adenine, Thymine, Guanine, and Cytosine (Figure 2-1). The arrangement of these nucleotides acts as a language which the cellular machinery can read. When examining an organism's DNA, these nucleotides are often referred to by the first letter of their names: A, T, C, and G.

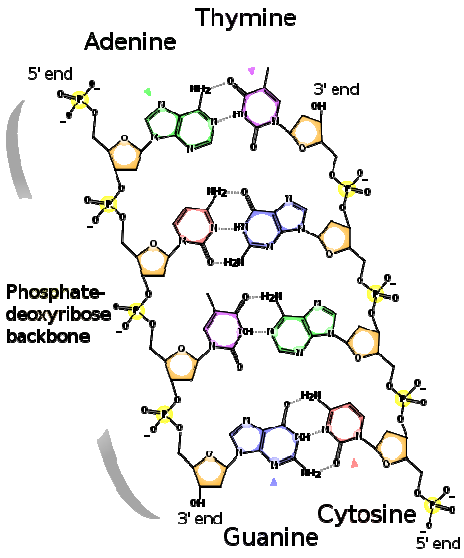

FIGURE 2-1: THE CHEMICAL STRUCTURE OF DNA - (BALL, 2007)

The central dogma of molecular biology states that DNA is transcribed into RNA, and then translated into a protein (Figure 2-2). This process occurs in every cell of every living organism (Griffiths, Wessler, Lewontin, & Carroll, 2008). Proteins are biologically active molecules, and play a part in almost all cellular functions. For example, proteins are an active part of the immune system, and can regulate chemical reactions within cells. Not all proteins

are needed in every cell type. Your stomach cells may require certain proteins for digestion, but these same proteins may not be needed in brain cells.

GENES

A gene is a section of DNA which codes for a protein. The gene contains instructions for the cellular machinery, telling the cell where to begin and end transcription, and how to create the protein. Upstream of the gene's coding region is an area called the promoter region. This promoter region plays a significant role in controlling when the gene is 'turned on'. A gene is 'turned on' when it is being transcribed into RNA and translated into a protein.

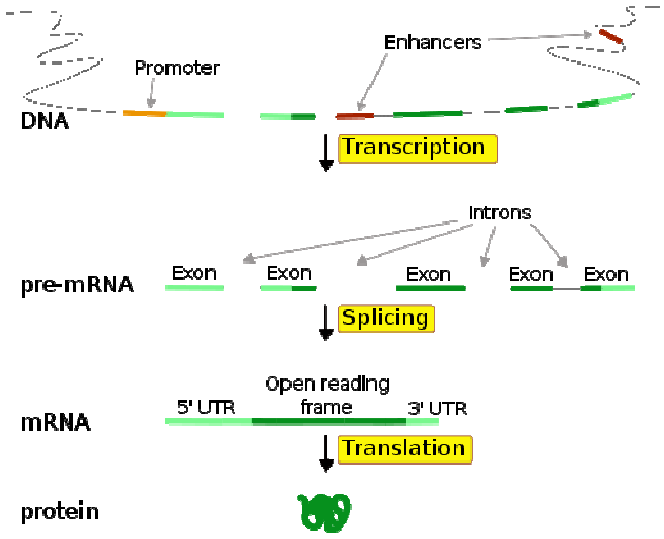

FIGURE 2-2: THE CENTRAL DOGMA OF MOLECULAR BIOLOGY (FORLUVOFT, 2007)

Within the promoter region there are short sequences of DNA called 'transcription factor binding sites' (TFBS). These sites range in length, but are usually short. For this project, lengths of between six and twelve bases were used. These sites recruit transcription factors, which are proteins that control whether or not a gene is 'expressed'; that is, whether it is turned on. There are three categories of transcriptions factors. 'Basal' transcription factors are part of the minimum set of transcription factors required for transcription to occur. 'Master' transcription factors control a set of genes, for example all neural cells. 'Specific' transcription factors control a small set of genes, such as a subtype of neural cells.

In this project, data mining techniques are used to identify potential TFBS's. Potential TFBS identified this way are referred to as motifs. The term motif refers to a pattern of DNA which is conjectured to have biological significance.

FIGURE 2-3: PROMOTOR REGION OF GENE 'X'. M1, M2, AND M3 ARE MOTIFS. TF1, TF2, AND TF3 ARE TRANSCRIPTION FACTORS. DISTANCES BETWEEN MOTIFS ARE INDICATED AS CIRCLED NUMBERS. (ICEV, RUIZ, & RYDER, 2003)

Certain combinations of motifs, as well as their relative positions, can play a role in determining a gene's expression. Figure 2-3 shows three motifs and their corresponding transcription factors. In this example, it may be the case that for the gene to be expressed, the motifs must be present in the order shown. Hence, the occurrence of the same motifs in a different order, or at different relative positions, may not allow the gene to be expressed. Thus, genes that are expressed in a particular cell type would be expected to share common motifs, possibly in a common order or spacing pattern.

CAENORHABDITIS ELEGANS(C. ELEGANS)

Often in biology model organisms are used to study biological processes in situations where use of human subjects would not be feasible. The roundworm *C. elegans* is one such organism (Figure 2-4). It is useful to study for a number of reasons. It is approximately one millimeter in length, so many can be bred. It exhibits a generation time of four days, making it easy to study across generations. The entire *C. elegans* genome has been sequenced. This information is easily accessible with a variety of tools on the internet. Up to 60% of human genes are thought to have *C. elegans* analogues.

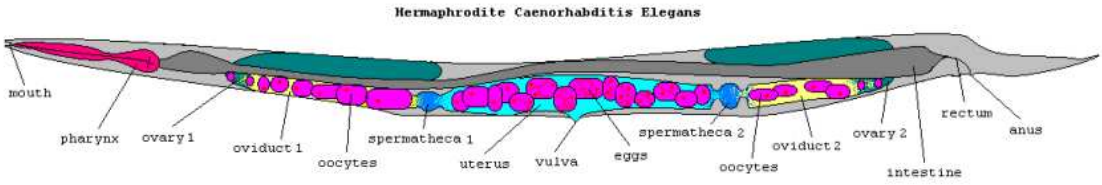

FIGURE 2-4: C. ELEGANS ANATOMY (GYLL, 2008)

THE PROCESS OF BUILDING OUR CLASSIFIER

There are four steps involved in building our classifier for gene expression based on promoter regions. First, the genes and cell types of interest must be selected and their promoter regions collected. Second, motifs are mined from the promoter regions of the genes in each cell type. Third, classification rules are generated from the motif and expression pattern data. Lastly, a classifier is built from these classification rules.

COLLECTING RAW GENE SEQUENCES

In order to mine for motifs, a collection of promoter regions must first be assembled. In order to do this, a list of cell types of interest, and the genes expressed in each of these cell types must be compiled. The WormBase web site is used to accomplish this (WormBase, 2010). Once this list of genes is chosen, a WormBase tool called WormMart is used to collect their promoter regions. These promoter regions consist of a string of text, with each character corresponding to a nucleotide base (Figure 2-5).

```
>WBGene00001075|dpy-14
gattagttgaaaaactgcccgacgagtatgtttcaaaatcgttaaagtttaatt
gaatatatcaattatttcagaccaaatcgtttaatgttggttcatggattgatgg
atgagaatgttcattttgcacatttaacacatcttgttgatgagtgtattaaaaa
aggaaaatggcatgagttggtggtaagtgtcgaaa
```

FIGURE 2-5: FIRST 200 BASE PAIRS OF THE PROMOTOR REGION OF GENE 'DPY-14'

It is possible for TFBSs to occur as far as 5000 base pairs into the promoter region, however the farther away from the gene a TFBS occurs, the less likely it is to play a role in gene expression. For this project, a promoter region consisting of the first 1,000 base pairs upstream

of the gene was used. Figure 2-5 shows the first 200 base pairs of the 1,000 base pair promoter region gathered for the gene 'dpy-14'.

MINING FOR MOTIFS

There are a number of tools which can be used to mine for motifs. All of them perform the same basic function, looking for repeated sequences within the promoter regions that are provided. For this project, the motif mining tool Weeder (Pavesi, 2009) was used. This decision was made primarily based upon an article which found that Weeder consistently performed better than the other tools available (Tompa, 2005). Weeder exhaustively finds all possible motifs within the sequences provided, and compares the frequency of their occurrences to determine the most conserved sequences that occur in many of the promoter regions. It looks for sequences which are found in many of the promoter regions provided. It can be used first to identify motifs, and subsequently locate each occurrence of that motif in all the promoter regions.

MINING FOR CLASSIFICATION ASSOCIATION RULES

Classification association rules (CARs) are a special type of association rule. An association rule is a rule of the form:

A → B, where A and B are both sets of attributes.

This rule says that if the attributes in set A are found in a given instance, those in set B are likely to be found as well. The left-hand side is referred to as the antecedent, and the right-hand side is the consequent. The following is an example of an association rule:

Customer buying shoes → Customer buying socks

This rule would indicate that if a customer bought shoes, he/she is likely to also have bought socks. CARs are association rules where the consequent is a class value. For example:

Two tires, Motor powered → Motor Cycle

This classification rule classifies a type of vehicle. In this case if a vehicle has two tires and a motor, it is predicted that it is a motorcycle. For this project, CARs are used where the antecedent is a set of motifs, and the consequent is a cell type. For example:

Motif 1, Motif 3 → Neural Cell

In order to find CARs, the 'Apriori Algorithm' (Agrawal & Srikant, 1994) is used. It processes data which consist of sets of motifs paired with sets of expression types, and returns possible classification rules. The algorithm also takes into account the relative positions of the motifs within each genes promoter region, but not the actual distances between them. A past student (Pray, 2004) implemented this algorithm within a data mining software package called Weka (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009).

The following is an example of data that could be mined for CARs:

Gene #	Motifs found in promoter region	Cell types gene is expressed in
Gene 1	M1, M3, M6	EXC, ASI
Gene 2	M2, M3, M6	EXC, ASI
Gene 3	M3, M4, M5	PHA
Gene 4	M2, M3	EXC

FIGURE 2-6: EXAMPLE OF DATA. NOTE THAT ONLY MOTIF ORDERING IS INCLUDED IN THIS EXAMPLE, NOT SPACING

There are numerous classification rules which could be produced for this data. The following are examples of rules which could be generated for the example data:

Rule 1: M3, M6 → EXC, ASI

Rule 2: M3, M4 → PHA

Rule 3: M3 → PHA

FIGURE 2-7: EXAMPLE OF CLASSIFICATION ASSOCIATION RULES

Rules 1 and 2 are both good rules, since they make an accurate prediction each time their antecedents are found. Rule 1 only applies for Genes 1 and 2, since only those genes contain both M3 and M6. Rule 3 on the other hand is a poor rule. All four genes contain M3, but it only classifies one of them correctly.

Two metrics were used when evaluating CARs, support and confidence. Support is the portion of the genes which contain the antecedent of a rule. Confidence is the percentage of these instances which are correctly classified by the rule. In the model above, the support of Rule 1 is .5 since M3 and M6 are found in two genes out of four. The confidence of this rule is 1.0 since it classifies both of these instances correctly. The support for Rule 3 is 1.0, since M3 is found in all four genes. Its confidence is .25, because it only classifies one of these four instances correctly.

Building a Classifier

The process of building a classifier entails selecting the classification rules which will make the 'best' model. There are three methods we used in constructing a model. The simpler of them is the all rules model. As its name implies, this model finds all the rules which apply to a specific gene. The antecedents from these rules are combined to make a prediction. In the example, rules 1 and 3 both apply, so the prediction would be EXC, ASI, and PHA. It is apparent why this is a problem, since this combination of rules incorrectly predicts PHA, when rule 1 alone would have made the correct prediction.

To address this problem, a model can be created in a second way, known as CBA (Palanisamy, 2006). This method first incorporates the rule which correctly classifies the most genes. It then removes these genes from consideration. A second rule is selected which classifies the greatest quantity of the remaining genes correctly. This process is repeated until adding further rules reduces the quality of the classifier. At this point, a default class is determined based on which cell type occurs most in the remaining genes. When using a model generated this way to classify a gene, the first rule which applies to the gene is used. If no rule applies, the default is used. If CBA were used with the example above, rule 1 would be used first since it predicts the most cases correctly. When using this model on the first gene, it would make the correct prediction of EXC and ASI.

In order to ensure that the set of rules chosen includes representatives from each cell type, a third method can be used, known as Modified CBA. Within each cell type, rules are

added until either they no longer improve the quality of the model, or the maximum number of rules is reached for that cell type. When making a prediction, this model uses all the rules whose antecedents are met. The union of their consequents is used as the prediction.

When building a classifier, different sets of data are used to train and test it. The training data is used to produce the rules and models, and then the performance is gauged on the test data. A method called 'n-fold cross validation' can be used to accomplish this. It splits the data in to n subsets. It then trains on n-1 of these subsets, and tests on the remaining one. This process is repeated n times, with each subset being used as a test set once. For this project, 4-fold cross validation was used.

EVALUATION OF CLASSIFIERS

When evaluating a model, we used four metrics: accuracy, precision, recall, and E-measure. Accuracy indicates how close a classifier is to making a correct prediction in all cases. Precision measures the correctness of a rule or model when it makes a prediction. Recall measures the portion of the occurrences of a class a rule or model successfully predicts. The formulas for these measures are provided below. Note that these metrics range from 0 to 1, with 1 being the best value.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{(True\ Positives + True\ Negatives + False\ Positives + False\ Negatives)}$$

$$Precision\ = \frac{True\ Positive}{True\ Positives + False\ Positives}$$

$$Recall\ = \frac{True\ Positive}{True\ Positives + False\ Negatives}$$

FIGURE 2-8: FORUMAS FOR PERFORMANCE METRICS

E-measure combines precision and recall, giving each of them a certain weight based on a value, Beta. For the purposes of this research Beta is always set to one, weighting them

equally. E-measure can range from 0 to 1, with 0 being the best value. E-measure is useful because of the relationship between precision and recall.

$$Emeasure = 1 - \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * (precision + recall)}$$

FIGURE 2-9: FORMULA FOR E-MEASURE

Generally speaking, if a model is tuned to make more predictions, its recall will improve but its precision will worsen. Consider the extreme case: a model which predicts every cell type for each gene, regardless of the motifs present. Such a model would have perfect recall, but very low precision.

The inverse of this is also true. When a model is tuned to make fewer predictions, its precision will improve but its recall will worsen. Consider a model which contained only a few very specific rules. When any of these rules is met, the model may make the correct prediction, however the rules will be met infrequently. Such a model would have very high (perhaps perfect) precision, but very low recall.

By balancing these metrics, E-measure gives a gauge of the quality of a model that takes into account this relationship between precision and recall. Both of the models described would have very poor E-measures, since they have poor precision and recall, respectively. Consider models with the following values:

Model	Precision	Recall	E-Measure
A	1	.1	.82
B	0.1	1	.82
C	.25	.25	.75

FIGURE 2-10: EXAMPLE OF E-MEASURE

Models A has perfect precision and model B has perfect recall, however both models have poor E-measure. Even though model C has only one fourth as much precision as model A, it has a better E-measure because it has a balance of precision and recall.

3. Data Collection

Past Work

A previous project (Thakkar, 2007) selected nine 'high density' cell types from *C. elegans*. 'High Density' refers to there being enough genes known to be expressed in these cells to produce statistically viable results. Approximately 30 genes of interest expressed in each of the cell types were selected. The size of this data set was limited by the available expression information at the time. Fortunately, new *C. elegans* data are being collected all the time, and a significant amount of additional information is now available.

Cell Types Selected

In order to obtain results which could be compared with past work, the nine cell types used in (Thakkar, 2007) were used in this project. An additional cell type was selected as well, excretory cells. This cell type was chosen on the basis that a great deal of gene expression information was known for it. A list of the cell types chosen is below (Figure 3-1).

Cell Type	Number of Genes	Number of Unique Genes
ADL	72	9
ALM	61	25
ASE	106	45
ASH	77	5
ASI	90	21
ASK	70	7
CAN	58	19
HSN	77	32
PHA	76	15
EXC	312	284
Totals:		
10	999	462

FIGURE 3-1: GENES EXPRESSED PER CELL TYPE

A complete list of the genes expressed in each cell type can be found in the appendix.

GENES SELECTED

In total there are 606 unique genes expressed in these cell types. This is a significantly larger data set than the 77 genes used in past studies. These genes were chosen using WormBase's Worm Mart tool. For this project, all genes that were expressed in a given cell type at any point in the organism's development were used. The decision to use genes from the entire life cycle as opposed to only from the adult organism was made based upon the available data. Not all genes in Worm Mart are annotated with life cycle data, so selecting only those genes annotated with the adult life stage would ignore many valid genes. Including all genes regardless of life cycle stage provides a larger data set to experiment on, at the cost of that data set being less accurate, since it is possible that different motifs would be required to allow gene expression at different life cycle stages.

PROMOTER REGIONS

Once the genes of interest have been chosen, their promoter regions must be gathered. Two decisions were made in deciding what portion of the promoter region would be most useful.

The number of base pairs used from each promoter region has a significant impact on the results. Using too small of a selection causes potentially useful motifs to be ignored. Using too large of a promoter region both increases the computation time, and can cause motifs to be found which are too far from the gene to be of biological significance. For this project, it was decided that examining the first 1,000 base pairs would be the best balance between these extremes. Initial research in motif elicitation used 5,000 base pairs or more, but since then experimentation performed by (Stolzar, 2006) found that Weeder did not produce significantly different motifs when 5,000 base pairs were used instead of 1,000. After running experiments with data sets of both sizes during this project these results were confirmed. It was also found that the run-time of Weeder increased at a greater than linear rate with increased sequence length. Searches were significantly more than 5 times longer when using the 5,000 base pair version as opposed to the 1,000 base pair version. For these reasons 1,000 base pairs was used.

When gathering the promoter region, the section of the gene used needed to be decided upon. The promoter region of a gene is the section at the 5' end of the gene, beginning immediately upstream of the start of transcription. Unfortunately, the precise location of the start of transcription is not known for all the genes used. However, the beginning of the gene coding region is known for all of these genes. The distance between the start of transcription and the beginning of the coding region is generally not too large in *C. elegans*. Due to these factors, it was decided that the flank of the start of the gene coding region would be used. This choice allows for a consistent flanking region to be gathered.

4. Mining For Motifs

The process of mining for motifs with Weeder takes place in two steps. First, the promoter regions are examined in order to find all the possible motifs. Second, motifs which are expected to be of use in determining the expression type of a gene are selected. This second step is necessary since some of the motifs found will be equally distributed among genes expressed in all of the cell types. Such motifs may in fact be binding sites, but they are not useful for building a classifier since they do not aid in differentiating among genes.

Initial Mining

The weederTFBS tool (Pavesi, 2009) was used to elicit motifs from the promoter regions. Each cell type was processed separately. It is more likely that a motif useful in differentiating cell types will be found when looking at an individual cell type. A motif may occur in over half of the genes of one cell type, yet may not be found at all in the other types. Such a motif would be very useful in building a classifier. If all the cell types were looked at together, the motifs found would be those that promote transcription in all cell types, and would not be useful in differentiating among them.

Within each cell type, ten motifs of length 6, 8, 10, and 12 were mined, for a total of 40 motifs for each cell type. Transcription factor binding sites do not always need to be exact matches to perform their function, so some amount of error is permitted when mining for motifs. In this case, 0, 1, 2, and 3 errors were allowed for the aforementioned motif lengths, respectively. These values were arrived at based upon recommendations in Weeder's documentation.

Naïve Selection of Motifs

The initial mining provided a total of 400 motifs. Out of this list of 400, 322 were unique. This result indicates that there were 78 instances of a motif occurring frequently in more than one cell type. This value only accounts for exact matches; it is likely that there are other occurrences where a motif occurred in more than one cell type, with a slight mutation. These

motifs may be binding sites for master or basal transcription factors. Basal transcription factors are unlikely to be of use in building a classifier since they will occur in all the cell types. Master transcription factors may be useful, since they could help differentiate between neural and excretory cell types. It is likely that master transcription factors will be found, since 144 of the 606 genes occur in more than one cell type.

Specific transcription factors binding sites are most useful in building a classifier. These transcription factors play a role in regulating which cell types a gene is expressed in, and will likely occur in only a subset of the cell types.

It was expected that not all of the motifs found would be useful in building a classifier. Numerous methodologies were implemented and tested to select an optimal subset from the original list of 400. These methods and their strengths and shortcomings are described below.

SELECT TOP FROM EACH CELL TYPE

The first method tried was to select the 'best' motifs of each length for each cell type. Each cell type would contribute one motif of each length, for a total of 40 motifs. Each motif was chosen over the others in its group based upon a value assigned to it by Weeder, which was reflective of the frequency of its occurrence. This frequency value includes multiple occurrences within one promoter region. There are numerous flaws with this technique.

- Over Occurring - Motifs selected this way may to occur too frequently to be useful. For example, a motif of length six which occurs 10 times in every gene is unlikely to be useful. An example of this is the repeated sequence 'AAAAAA'. Weeder does not eliminate such motifs from consideration itself.
- Distribution between cell types - Selecting motifs this way does not take into consideration whether or not a motif occurs with equal frequency in all cell types. If a motif occurs in this pattern, it cannot be used to differentiate between different expression patterns.
- Eliminates potentially useful motifs – If two motifs of the same length occurring in the same cell type would both be very useful in building a model, only one of them can be used. The top two motifs of length 12 in a given cell type may be better than the best motifs of length 6, 8, and 10, but in this model one will still never be used.

Selecting the most expressed motifs

This method chooses motifs which are expressed the most based on Weeder's output. This method disregards the motif length and cell type a motif is expressed in, and simply uses the frequency of its occurrence. This approach partially solves the problem of eliminating motifs because another one of the same length occurs more frequently than it, but introduces another problem. Motifs of length six are bound to occur much more frequently than those of length twelve. A variation on this method selects the top motifs of each length, solving this problem. This method still does not solve the problem of over-occurring motifs, nor does it address the distribution of a motifs representation between cell types.

Selecting the motifs shared by the most cell types

As mentioned before, out of 400 motifs mined, only 322 were unique. This method examined the motifs which were repeated between cell types, since they are more likely to be true binding sites. This method still shared the flaws of its predecessors; specifically it increases the likelihood that an over-occurring motif is selected. It also exacerbates the problem of choosing motifs which may be true binding sites, but do not aid in differentiating between cell types. Even if a motif is a true binding site, if it is found in all the promoters, it is likely to be a basal factor binding site, rather than a master or specific binding site.

Selection of Useful Motifs

The three techniques discussed so far shared common flaws, and due to these flaws resulted in models which were not very predictive. Two approaches were combined and implemented to address these problems.

Compensating for over occurrence

When Weeder is used to find occurrences of a motif, it allows the user to provide a threshold for how closely a sequence must match a motif to be considered an instance of that motif. Reasonable values for this threshold range between 85-99%. Biologically, it is the case that some binding sites must be matched exactly, whereas others can allow for a certain degree

of mutation. The previous techniques used one threshold for all the motifs of a given length; however this resulted in some motifs rarely occurring while others occurred too frequently. The new approach examines the occurrences of each motif at values of 85, 90, 95, and 99. This allows for the threshold to be selected for each motif individually based on the number of occurrences at each of these thresholds. This threshold is different from the number of errors allowed when mining for motifs in the 'Initial Mining' section.

SELECTING MOTIFS BASED ON DISTRIBUTION AMONG CELL TYPES

In order to find motifs which would aid in differentiating between cell types, the number of times a given motif occurred in each cell type was examined. A chi squared test can be used to compare distributions among cell types. This test compares the distribution found for a given motif with the expected distribution if the motif were distributed at random among all the promoters. It produces a P-value, which represents the likelihood that the distribution would occur by chance. A value of less than 5% would indicate that a distribution is very unlikely to occur by chance. In the context of building a classifier, a motif with such a distribution is very useful, since this value means that it is occurring in some cell types at higher rates. In this case, the expected distribution would be the number of genes in each cell type. Consider the following example (Figure 4-1).

Cell types	A	B	C	D	E	
Number of genes	10	10	2	2	5	Chi-Squared Value
Motif 1: Number of occurrences	5	6	1	2	2	0.904
Motif 2: Number of occurrences	10	1	0	0	1	0.012

FIGURE 4-1: EXAMPLE OF MOTIF DISTRIBUTION (MULTIPLE OCCURANCES IN ONE PROMOTOR ONLY COUNTED ONCE)

The expected distribution in this example would match the distribution of genes. If a motif occurred with equal frequency in each gene, for example with a frequency of 0.5 per gene, we would expect it to be found in each cell type at a rate of 0.5*(number of genes in that cell type). In the example above, it is evident that motif 1 exhibits this behavior. This indicates that motif 1 is more likely to be a basal transcription factor binding site, and is not useful in

building a classifier. This motif occurs at the expected frequency within each cell type, and the P-value produced by a chi squared test is quite high: 0.904. This indicates that such a distribution is likely to occur by chance.

By contrast, the distribution of motif 2 is significantly different than the expected distribution. It occurs in every gene of cell type A, and in the other cell types at a much lower frequency. This makes motif 2 very useful in building a classifier. The rule 'Motif 2 → Cell type A' would be quite accurate. The chi squared test yielded a very low P value for this Motif: 0.012. This indicates that this distribution is very unlikely to occur by chance.

By using a chi squared test, a meaningful value can be assigned to the usefulness of each motif. Using this knowledge, a two step process can be applied to select the most useful motifs. First, the different thresholds for each motif were compared. As mentioned above, when finding occurrences of a motif within the promoter, a threshold for how much error is allowed is set. In this study, 85, 90, 95 and 99 percent accuracy were used. In order to determine which threshold would be most useful on a motif by motif basis, chi squared tests were conducted on each threshold level for each motif. Then, within each motif, the p-value for each threshold level was compared. The threshold with the lowest p-value for that motif was selected, and only the occurrence data of that motif for that threshold value was kept.

Next, the motifs were compared with each other. The motifs with the best chi squared values were kept. The threshold for maximum chi squared test p-value allowable can be adjusted in order to obtain the desired number of motifs. The value 0.05 is often used in statistics as the threshold for considering a distribution to be significantly different from the expected value. Once the choices for motif have been made, their occurrences were annotated (using data from the particular accuracy threshold chosen for each motif) and used as the input for mining classification association rules and building a model.

5. RESULTS

In order to determine which methods of motif selection produced the best model, the different techniques above were used to select motifs and annotate their occurrences in each gene's promoter region. The resulting data were processed by the Associative Classification Classifier implemented by past students. (Palanisamy, 2006) (Pray, 2004) (Rudolph, 2009)This classifier was used to mine classification association rules and build a model out of them. The model was evaluated on three metrics, E-measure, precision, and recall, which are explained in 'Section 2 - Background'. For specific setting information, refer to 'Appendix E - Mining Classification Association rules and building a model'.

NEGATIVE CONTROL FILE

In order to be able to evaluate the quality of a classifier, a basis for comparison must first be established. This is referred to as a negative control. To accomplish this, multiple files were created, each made up of randomly assigned motifs and expression types. For each file, three hundred fake genes were generated. Each of these genes was given a random number of motifs, ranging from five to twenty. These motifs came from a list of forty fake motifs, (M1-M40). Each gene was also given a random number of cell types to be expressed in, ranging from one to three. These cell types were chosen from a list of 10 fake cell types. These values were chosen in order to be similar to the real data generated by some of the motif selection methods. Since any rules found in these randomly generated file would not have any biological significance, the values achieved for the metrics being used to analyze performance are considered the baseline against which actual models are compared.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.06	14	.26	.20	.79

FIGURE 5-1: VALUES OBTAINED BUILDING A MODEL FROM FAKE.ARFF

These tests indicated that even with random data, an E-Measure of .79 can be achieved. This value varied insignificantly among the different random files. Keep in mind that a lower E-measure is better.

NAÏVE MOTIF SELECTION

SELECTING THE TOP MOTIFS OF EACH LENGTH

The first motif selection tested was composed of the top motif of each length for each cell type. The classifiers built from this selection were not biologically significant.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.1	.1	24	.59	.14	.78
.3	.05	3	.39	.43	.60

FIGURE 5-2: VALUES OBTAINED BUILDING MODELS FROM 36MOTIFSALLEXC.ARFF

The best E-Measure achieved by these models was 0.6. This value is 25% better than the 0.79 value obtained from the random data; however in this case the value is meaningless because the model only predicts excretory cell expression.

motifs=M7 motifs=M6 ==> expr=EXC

motifs=M7 motifs=M40 ==> expr=EXC

motifs=M40 ==> expr=EXC

Default Class Label: expr=EXC

FIGURE 5-3: A MODEL GENEREATED FROM 36MOTIFSALLEXC.ARFF

This model only used three rules. The higher than expected values are the result of using every gene to generate the rules and model, including all the genes found only in excretory cells. Since there are so many more excretory cell genes than there are genes of the other types, even a model which predicts exclusively excretory cells, like this one, actually obtains seemingly good values. When a lower minimum confidence is used, such as 0.1, rules for the other cell types are included, but the quality of the model remains poor. Based upon this result, future tests used only a subset of excretory cell-only genes.

SELECTING THE TOP MOTIFS OF EACH LENGTH, USING ONLY A SUBSET OF EXCRETORY CELLS

The same set of motifs was used for this test as in the previous one, only with all but thirty excretory-only genes removed. The value thirty was arrived at based on the number of

other genes which were expressed in only one cell type. Not counting excretory only genes, there were 178 genes expressed in only one cell type. This means that on average each non-excretory cell type had twenty genes which were expressed uniquely in it. The choice of thirty excretory-only genes was made to be reasonably close to this value. This left 58 genes expressed in excretory cells in total, since there were an additional 28 genes expressed in excretory cells that are found in other cell types as well. Prior experimentation indicated that which excretory cell-specific genes were used did not have a substantial effect on the models produced; thus the thirty excretory cell-specific genes chosen were selected arbitrarily. By selecting only thirty, models which are not biased by an over abundance of excretory cells could be constructed.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.05	16	.22	.25	.81
.1	.15	30	1.0	.21	.68

FIGURE 5-4: VALUES OBTAINED BUILDING A MODEL FROM 36MOTIFS30EXC.ARFF

These models provided insight into the impact of minimum confidence and minimum support on what rules are created. When a minimum confidence of 0.3 was used, only 16 rules could be found, even when minimum support was reduced to .05. This produced poor results. When minimum confidence was lowered to .1, the desired 30 rules were found without having to lower the minimum support below .15. Also, a recall of 1.0 was achieved, meaning that the model predicted almost every cell type for every gene. Examination of the rules reveals why.

motifs=M7 ==> expr=ASE	motifs=M7 ==> expr=PHA
motifs=M7 motifs=M1 ==> expr=ASE	motifs=M7 motifs=M1 ==> expr=PHA
motifs=M1 ==> expr=ASE	motifs=M1 ==> expr=PHA
motifs=M7 ==> expr=ASI	motifs=M7 motifs=M1 ==> expr=ASK
motifs=M7 motifs=M1 ==> expr=ASI	motifs=M7 ==> expr=ASK
motifs=M1 ==> expr=ASI	motifs=M1 ==> expr=ASK

FIGURE 5-5: PART OF A MODEL GENERATED FROM 36MOTIFS30EXC.ARFF

Motifs 1 and 7 were found in almost every gene, so all the rules consist of one or both of them. This is a particularly meaningless result, indicating that the motif selection was poor.

SELECTING THE MOST FREQUENTLY OCCURRING MOTIFS

Instead of selecting only the top motif of each length from each cell type, the most frequently expressed motifs were chosen, regardless of which cells they were expressed in or their length. This resulted in a selection containing the top 42 motifs. In order to avoid over representation of excretory cells, only 30 excretory only genes were kept as in the last test.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.05	12	.31	.27	.76
.1	.05	27	.52	.25	.72

FIGURE 5-6 VALUES OBTAINED BUILDING A MODEL FROM MOTIFS30EXC.ARFF

Despite altering the heuristic for motif selection, these results are not substantially different from the other naïve methods. It was determined that without a better method of selecting motifs, a meaningful model could not be constructed.

USING CHI SQUARED TESTS TO SELECT MOTIFS

In order to select more meaningful motifs, chi-squared tests were used to find motifs which would be more useful in building a classifier. This methodology is described in the 'Selection of Useful Motifs' portion of Section 0. The program written to select motifs allows a desired number of motifs to be provided. This functionality was used to build models from selections of motifs of various sizes.

USING 25 MOTIFS

The first models were constructed using a selection of 25 motifs. These motifs all had chi-squared values below 0.0085. This value is very low, and indicates that their distribution between cell types is very different than the expected distribution assuming occurrence based simply on the number of genes expressed in a given cell type.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.07	5	.25	.24	.79
.2	.05	19	.61	.21	.73
.1	.05	31	.77	.20	.72

FIGURE 5-7: VALUES OBTAINED BUILDING A MODEL FROM CHI025.ARFF

The best model constructed from this selection of motifs was not any better than the models constructed with the naïve methods. After this selection had been tested it was decided that a larger selection of motifs should be used. The chi-squared values of the motifs used in this selection were extremely low. It is likely that other motifs with chi squared values above .0085, but still low, will also be useful.

USING 50 MOTIFS

In order to select the top 50 motifs, Motif Analysis used a threshold for the chi-squared value of 0.0475. This value is slightly below the .05 threshold often used as the cutoff point for statistical significance.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.05	19	.45	.28	.71
.2	.05	36	.65	.25	.69
.1	.05	38	.67	.24	.70

FIGURE 5-8: VALUES OBTAINED BUILDING A MODEL FROM CHI050.ARFF

The models produced by this selection of motifs had E-measures slightly better than both those selected using the naïve methods and those produced with the 25 motif selection. The E-measure at all three confidence levels was approximately the same, ranging only from 0.69 to 0.71. Although the E-measure did not vary greatly, the recall and precision both did.

USING 150 MOTIFS

Based upon the success of the 50 motif selection, a larger selection was tried. The motifs in this selection had chi-squared values below 0.239. The 100 additional motifs included in this selection had higher chi-squared values than those in the previous model, so by themselves they are less likely to have predictive power, so it is reasonable to expect they will not be of use in creating a more accurate model.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.3	.05	31	.52	.27	.70
.2	.05	36	.55	.25	.70
.1	.05	40	.62	.24	.70

FIGURE 5-9: VALUES OBTAINED BUILDING A MODEL FROM CHI150.ARFF

The addition of these 100 motifs did not improve the quality of the classifiers relative to the 50 motif selection.

USING 310 MOTIFS

In order to exhaustively try all possibilities, a model was built using almost all of the motifs. The motifs selected all had chi-squared values below 0.95. The remaining twelve motifs that were discarded had chi squared values above 0.95, and were short motifs unlikely to have biological significance, such as 'AAAAAA' and 'TTTTTT'. This motif selection still used chi-squared tests to determine the best threshold for accuracy within each motif.

Minimum Confidence	Minimum Support	Number of rules	Recall	Precision	E-Measure
.2	.12	23	.59	.24	.71
.1	.1	35	.77	.23	.68

FIGURE 5-10: VALUES OBTAINED BUILDING A MODEL FROM CHI312.ARFF

Models produced from this set of motifs produced the best results of any approach tested. The best E-measure found was 0.68, which is 0.02 lower than the best value obtained with naïve methods, and 0.11 lower than with the randomly generated motifs. This model is included in 'Appendix F – Models'.

An interesting aspect of this model is that many of the rules use the same motif to make different predictions. For example, motif 6 is used to predict the ADL, ASH, and PHA cell types. This pattern of individual motifs predicting multiple cell types makes sense biologically. Cell types are likely to contain some of the same transcription factors in order for genes to be expressed in common between them.

PERFORMANCE OF INDIVIDUAL RULES

Another area explored was how well a model performed at predicting each cell type. The model generated using 310 motifs was analyzed to determine the performance of the rules

for each cell type. When the Associative Classifier builds makes a prediction, it examines all of the rules. Each rule whose antecedents are met is counted toward the prediction. This means that if one of the ASE rules and 3 of the ASI rules are met, the classifiers prediction will be ASE and ASI.

Based on this behavior, all of the rules for each cell type were looked at simultaneously. Each gene was compared against all these rules, and if any of the rules antecedents were found in the promoter region, it was considered a positive for that gene. If the gene was expressed in that cell type, the prediction was a true positive. If not, it was a false positive. If none of the rules were matched, but the gene was expressed in that cell type, the prediction was a false negative. If none of the rules were matched and the gene was not expressed in that cell type, the prediction was a true negative.

Using the values obtained from this method, precision, recall, accuracy, and e-measure were computed.

Cell Type	True Positives	True Negatives	False Positives	False Negatives	Precision	Recall	Accuracy	E-Measure
ASE	93	66	170	13	0.35	0.88	0.46	0.50
ASI	78	72	180	12	0.30	0.87	0.44	0.55
HSN	70	74	191	7	0.27	0.91	0.42	0.59
PHA	73	41	225	3	0.24	0.96	0.33	0.61
ASH	69	40	225	8	0.23	0.90	0.32	0.63
ALM	56	59	222	5	0.20	0.92	0.34	0.67
ADL	71	13	257	1	0.22	0.99	0.25	0.65
ASK	69	30	242	1	0.22	0.99	0.29	0.64
EXC	3	123	171	14	0.02	0.18	0.41	0.97
CAN	51	48	236	7	0.18	0.88	0.29	0.70

FIGURE 5-11: PERFORMANCE BY CELL TYPE OF BEST E-MEASURE MODEL

These metrics reveal that most of the rules making up the model have very high recall. This means that the motifs in these rules are found in most of the genes in the data set. The accuracy for the cell types ranged from 0.25 to 0.46, indicating that the model was significantly

better at predicting some cell types than others. The value of 0.46 for ASE confidence indicates that almost half of the time that ASE was predicted for a gene, that gene was actually expressed in ASE. This combination of high recall and low precision indicates that the model made too many guesses. Less than one quarter of the positive predictions made were true positives.

6. Conclusions and future work

Model Building

Models were constructed using multiple strategies for selecting the best motifs, including choosing those that occurred most frequently, and choosing those with the best chi-squared values. The best model was constructed using 310 motifs, many of which had poor chi-squared values. It was unexpected that including such motifs would improve the classifiers performance. This result can be attributed to Apriori Sets and Sequences (Pray, 2004) effectiveness at choosing the best motifs. When the set of motifs was pruned using the chi squared method, motifs which were actually useful in building a model were eliminated. These motifs' distributions did not all differ significantly from the expected distribution; however, they still exhibited some predictive power.

Although the chi-squared method for selecting the accuracy threshold was not useful for eliminating motifs from consideration, it was still important in building models. It solved the problem of over represented motifs by selecting higher accuracy thresholds for them. This ensured that the classifier was given the best version of each motif.

Future work could explore building models which classify only one cell type at a time. Currently, the consequent of the classification association rules can only be a cell type or set of cell types. It would be useful to be able to use a Boolean value instead. The Boolean would indicate whether a gene was expressed in the cell type of interest or not. Rules mined this way may be more effective at making predictions within a given cell type than the current methodology.

The methods used in this project were not able to demonstrate a statistically significant improvement in performance relative to models generated from random data. In order to make a claim of statistical significance, the entire process could be performed on randomly generated promoter regions. Results generate this way would be directly comparable to the models generated from actual genetic data. For this project this process was approximated by

generating fake genes contain random motifs and expression patterns. While this method is a reasonable approximation, it is not sufficient to make claims of statistical significance.

DATA COLLECTION

This project was very successful in collecting new data. Many additional genes were added to the data set, and improvements to WormMart have made gathering this data easier than it was in the past. Future work could be performed to include more different cell types. If genes from entirely different cell types were used, it would be possible to mine for master transcription factors. The current data set uses 9 subsets of neural cell types as well as excretory cells. Data from different organisms, such as *C. briggsae* could also be used.

MOTIF MINING TOOLS

Future work could be done to explore how different motif mining tools would perform with this new data set. Other tools were used with past data sets, and applying them to this data set could yield different results than were obtained with Weeder.

7. APPENDICES

APPENDIX A – GATHERING RAW GENETIC DATA

DETERMINING ANATOMY TERMS

In order to gather a list of genes expressed in each cell type, WormMart was used. WormMart uses 'anatomy terms' to refer to the location a gene is expressed. As such, the anatomy terms corresponding to each cell type must first be gathered.

A List of cell/tissue types and their corresponding anatomy terms can be found in WormBase's 'Expression Search' tool. Each cell type has two anatomy terms associated with it, one for each the left and right half of the organism. While there is a great deal of overlap between these two, both were used for completeness.

FINDING GENES FOR EACH EXPRESSION PATTERN

Using this list of anatomy terms, the new set of genes was collected using WormMart. The process to use this tool is explained below, along with notes about the reasoning behind choices that were made.

- Navigate to WormMart
- Under 'Choose Database', select 'Release 195'

Upon entering WormMart, the database release must be selected. Release 195 was the most recent at the time of this project, so it was used in order to produce the greatest number of genes.

- Under 'Choose Dataset', select 'Expression Pattern'

This dataset allows searches of what genes are expressed in what parts of the organism.

- Select 'Filters' | 'Expressed in'
- Check 'Specified identifiers of type'

In order to gather a list of genes expressed in each cell type, the data must be filtered by Anatomy Term.

- Fill in the first Anatomy Terms in the space provided

A separate query must be performed for each cell type. A list of expression patterns used can be found in Appendix A. Each cell type has two Anatomy Terms, one for each the left and right. Both should be inserted on separate lines.

- Select 'Attributes'
- Uncheck 'Expression Pattern ID' and 'Expression Pattern'
- Under 'Expression of Gene' select 'Gene (WB Gene ID)'
- Click 'Results'

This action will produce a list of Gene IDs expressed in this cell type. Before clicking 'Results', 'Count' can be used to see how many results will be generated.

- On the results page, select 'Unique Results only'
- For a file format select 'TSV'
- Click 'Go' to download a file of the results

The TSV file format will produce a plaintext line break delimited list of Gene IDs. Selecting 'Unique Results Only' will remove duplicates caused by redundancies between the left and right half of each cell. The resulting TSV file will be used as an input in future steps. Name this file according to the cell type it is concerned with, and remove the first line from it which reads 'Gene (WB Gene ID)'.

Repeat the above steps for each cell type. Perform the procedure one additional time, entering all of the Anatomy terms to generate a complete list of WB Gene IDs. This complete list will be used when annotating the data at a later step.

COLLECTING PROMOTER REGIONS

Now that a list of genes has been assembled, their promoter regions can be collected. Using WormMart, perform the following steps for each gene type:

- Choose ‘WormBase Release 195’ as before
- Select the ‘Gene’ Dataset

This will allow searches for information about genes, as opposed to expression patterns as in the last step.

- Select Filters | Identification.
- Select ‘choose file’ and upload the first of the TSV files created in the previous step

These steps will select from WormMart each gene for a given cell type.

- Choose Attributes | Gene Sequences
- Under Sequence Type, select ‘Flank (Gene Coding Region)’

This specifies which part of the genetic sequence we are interested in. Gene Coding Region was chosen since the precise flank of the gene is not known for all genes, but the beginning of the Gene Coding Region is.

- Select ‘Flanking Regions’
- Check ‘Upstream Flank’
- Fill in a value of 1000
- Select the ‘Header Attributes’ category
- Ensure that only ‘Gene WB ID’ is checked in this section

This decision was made somewhat arbitrarily. For the purposes of parsing the sequences, it is simpler to use only one identifier per gene. Since the Gene ID is more useful when dealing with WormBase, it was chosen. If it becomes desirable to obtain Gene Public Names at any point in the future, WormBase can be used to ascertain them at that time.

- Click ‘Results’ followed by ‘Go’ to download the sequence file in FASTA format

FASTA files can be read directly by Weeder. Save this file with an appropriate name. [Cell Type].1000.FASTA was used for this project. Repeat this process for each cell type, and with the complete list.

APPENDIX B - MINING FOR MOTIFS

The next stage in the process uses Weeder to mine for motifs. For this project, Weeder 1.4.2 for *nix was used. For this project, two components of Weeder were used, along with Java classes created to work with Weeder's output.

WEEDERTFBS

This Weeder tool is used to mine for motifs within each cell type. It can be run from the *nix shell. It creates three files for each FASTA file it processes. This project is interested in the '.mix' file, which contains a list of frequently occurring motifs.

PROCEDURE

WeederTFBS.out finds potential Transcription Factor Binding Sites. It must be executed on each FASTA file separately. Additionally, it only produces motifs of a specific length on each run, so it must be run for motifs of length 6, 8, 10 and 12 independently. In order to do this efficiently a script (runWeederTFBS.sh) was used (see appendix B). The parameters used in this script, such as how many errors to allow for each motif, were determined through experimentation. It was found that motifs of length six occurred too frequently to be useable if any errors were allowed, whereas motifs of length ten barely occurred at all unless two errors were allowed. After running this script, three output files are generated for each FASTA file. The .mix files contain a list of motifs. To concatenate all these motifs another script was written (catMixFiles.sh), which is also in appendix B. After running both of these scripts, a file 'allmotifs.mix' will be produced.

APPENDIX C – NAÏVE MOTIF SELECTION

LOCATOR

This Weeder tool was used to find occurrences of motifs mined with weederTFBS from within the sequences. It processes a motif and a FASTA file and returns a file containing a list of the motifs occurrences.

GMB.MIXPARSER

This Java Class is used to process the concatenated output from the ten runs of weederTFBS. This class examines the motifs found, and returns those with the most useful ones. It removes motifs that are unlikely to be binding sites, such as a string containing only one base pair (example: AAAAAA). Motifs are compared against a threshold value, which will reject any motifs which do not meet it. Motifs which occur in more than one cell type are weighted more heavily. The user can supply a desired number of Motifs, and the threshold value will be iteratively reduced until at least this many motifs are chosen. To use this class, specify the input file and threshold in the beginning of the class. It will return a string of commands calls to Weeder's Locator tool that can be run in the root directory of Weeder. This class uses naïve motif selection methods.

GMB.WEETOARFF

This Java Class takes in the output from 'locator' and produces an ARFF file which can be used by Weka.

PROCEDURE

Use Java Class MixParser to process the mix file generated in the mining for motifs section, as described above. The output from MixParser can be run at the command line to search for occurrences of each motif in the sequence data. In order to do this, use the FASTA file containing the sequences from every gene that was generated earlier.

After executing locator on each motif using the script, .wee files will have been generated for each motif. These files contain a list of each place a motif occurred in each gene. Concatenate all of these .wee files.

The concatenated .wee file must be processed in order to be used by Weka. To do this, use the Java Class gmb.WeeToARFF. This class requires as input the .wee file and a list of which cell types and what genes are expressed in them (See appendix A). The .wee file is consumed to produce an array of motifs and their locations for each gene. This array is then converted into the ordered string format used by 'Apriori Sets and Sequences'. The gene expression list is used to create a similar string of cell types each gene is expressed in. The output of the class can be saved as an ARFF file and used by Weka.

APPENDIX D – MOTIF SELECTION IMPLEMENTING CHI SQUARED TESTS

Instead of using the methodology outlined in Appendix C, the Java Class MotifAnalysis can be used. It requires as input a file containing list of motifs, a file containing expression data, and a set of .wee files. There must be four .wee file for every motif, one for each threshold level (85, 90, 85, 99). Paths to these files must be specified at the beginning of the Java Class. When all of these files have been generated, a desired motif count can be set at the beginning of the class as well. Running the class will result in output formatted as an .arff file, complete with sequence data. This file can be used by Apriori Sets and Sequences.

APPENDIX E - MINING CLASSIFICATION ASSOCIATION RULES AND BUILDING A MODEL

Previous projects have produced a WPI fork of the Weka project which can be used to build classifiers. For this project, the branch 'weka344_project' was used. In order to use it, check this branch out from the bioinformatics project's sourceforge repository at 'sourceforge.wpi.edu:/cvsroot/wekacode '. Run weka.gui.GUIChooser, and select 'Explorer'. Select the .ARFF file you generated in the previous step for the file. Go the to the 'Classify' tab, and choose 'wpi.classifiers.AssociativeClassification'. This will allow you to build classification models. In order to produce reasonable results, use the following configuration as a starting point. These values were arrived at through experimentation.

Mining Delta: .01

Lower Bound – Minimum Support: .01

Number of Rules Mined: 40

Model Selection: Modified CBA Model

Adaptive Minimum Support: true

Adaptive Minimum Support – Range for Number of Rules: 30-50

MCBA Rules per Class: 4

Set Valued Prediction Mode: all consequents

Test Options: Cross Validation, Folds: 4

APPENDIX F –MODELS

HIGHEST SCORING MODEL: 310 MOTIFS, CONFIDENCE 0.1

M97[rp0-rp1] ==> expr=ASE [Conf: 0.4231, Sup: 0.1294, Lift: 1.3700, p-value: 2.47E-3]
M233[rp0-rp1] ==> expr=ASE [Conf: 0.4071, Sup: 0.1353, Lift: 1.3182, p-value: 5.66E-3]
M58[rp0-rp1] ==> expr=ASE [Conf: 0.4048, Sup: 0.1000, Lift: 1.3107, p-value: 2.83E-2]
M152[rp0-rp1] ==> expr=ASE [Conf: 0.3962, Sup: 0.1235, Lift: 1.2830, p-value: 1.89E-2]
M192[rp0-rp1] ==> expr=ASI [Conf: 0.3636, Sup: 0.1059, Lift: 1.3737, p-value: 8.05E-3]
M295[rp0-rp1] ==> expr=ASI [Conf: 0.3617, Sup: 0.1000, Lift: 1.3664, p-value: 1.22E-2]
M152[rp0-rp1] ==> expr=ASI [Conf: 0.3491, Sup: 0.1088, Lift: 1.3187, p-value: 1.77E-2]
M80[rp0-rp1] ==> expr=ASI [Conf: 0.3333, Sup: 0.1382, Lift: 1.2593, p-value: 1.58E-2]
M92[rp0-rp1] ==> expr=HSN [Conf: 0.3386, Sup: 0.1265, Lift: 1.4950, p-value: 1.37E-4]
M217[rp0-rp1] ==> expr=HSN [Conf: 0.3077, Sup: 0.1176, Lift: 1.3586, p-value: 4.87E-3]
M40[rp0-rp1] ==> expr=HSN [Conf: 0.3043, Sup: 0.1029, Lift: 1.3439, p-value: 1.42E-2]
M307[rp0-rp1] ==> expr=HSN [Conf: 0.2881, Sup: 0.1000, Lift: 1.2723, p-value: 4.76E-2]
M196[rp0-rp1] ==> expr=PHA [Conf: 0.2857, Sup: 0.1176, Lift: 1.2782, p-value: 2.13E-2]
M4[rp0-rp1] ==> expr=PHA [Conf: 0.2727, Sup: 0.1147, Lift: 1.2201, p-value: 6.36E-2]
M16[rp0-rp1] ==> expr=PHA [Conf: 0.2556, Sup: 0.1000, Lift: 1.1436, p-value: 2.55E-1]
M241[rp0-rp1] ==> expr=PHA [Conf: 0.2483, Sup: 0.1059, Lift: 1.1107, p-value: 3.45E-1]
M241[rp0-rp1] ==> expr=ASH [Conf: 0.2828, Sup: 0.1206, Lift: 1.2650, p-value: 2.38E-2]
M4[rp0-rp1] ==> expr=ASH [Conf: 0.2727, Sup: 0.1147, Lift: 1.2201, p-value: 6.36E-2]
M80[rp0-rp1] ==> expr=ASH [Conf: 0.2695, Sup: 0.1118, Lift: 1.2057, p-value: 8.67E-2]
M16[rp0-rp1] ==> expr=ASH [Conf: 0.2632, Sup: 0.1029, Lift: 1.1773, p-value: 1.6E-1]
M18[rp0-rp1] ==> expr=ALM [Conf: 0.2677, Sup: 0.1000, Lift: 1.4922, p-value: 1.05E-3]
M241[rp0-rp1] ==> expr=ALM [Conf: 0.2345, Sup: 0.1000, Lift: 1.3070, p-value: 2.25E-2]
M67[rp0-rp1] ==> expr=ALM [Conf: 0.2250, Sup: 0.1059, Lift: 1.2541, p-value: 3.89E-2]
M220[rp0-rp1] ==> expr=ALM [Conf: 0.2246, Sup: 0.1235, Lift: 1.2519, p-value: 1.64E-2]
M16[rp0-rp1] ==> expr=ADL [Conf: 0.2556, Sup: 0.1000, Lift: 1.2072, p-value: 1.12E-1]
M164[rp0-rp1] ==> expr=ADL [Conf: 0.2160, Sup: 0.1353, Lift: 1.0198, p-value: 8.06E-1]
M67[rp0-rp1] ==> expr=ADL [Conf: 0.2125, Sup: 0.1000, Lift: 1.0035, p-value: 9.75E-1]
M220[rp0-rp1] ==> expr=ADL [Conf: 0.2086, Sup: 0.1147, Lift: 0.9848, p-value: 8.73E-1]
M4[rp0-rp1] ==> expr=ASK [Conf: 0.2448, Sup: 0.1029, Lift: 1.1888, p-value: 1.31E-1]
M171[rp0-rp1] ==> expr=ASK [Conf: 0.2379, Sup: 0.1441, Lift: 1.1553, p-value: 7.06E-2]
M164[rp0-rp1] ==> expr=ASK [Conf: 0.2300, Sup: 0.1441, Lift: 1.1174, p-value: 1.54E-1]
M270[rp0-rp1] ==> expr=ASK [Conf: 0.2298, Sup: 0.1088, Lift: 1.1162, p-value: 3.01E-1]
M220[rp0-rp1] ==> expr=CAN [Conf: 0.1925, Sup: 0.1059, Lift: 1.1285, p-value: 2.35E-1]
M164[rp0-rp1] ==> expr=CAN [Conf: 0.1596, Sup: 0.1000, Lift: 0.9357, p-value: 4.86E-1]
M171[rp0-rp1] ==> expr=EXC [Conf: 0.1699, Sup: 0.1029, Lift: 1.2035, p-value: 5.93E-2]
Default Class Label: null

Classifier contains 35 rules.

25 MOTIF MODEL, CONFIDENCE 0.1

M5[rp0-rp1] ==> expr=HSN [Conf: 0.3623, Sup: 0.0853, Lift: 1.4346, p-value: 1.64E-2]
M8[rp0-rp1] ==> expr=HSN [Conf: 0.3600, Sup: 0.0922, Lift: 1.4254, p-value: 1.3E-2]
M2[rp0-rp1] ==> expr=HSN [Conf: 0.2619, Sup: 0.1126, Lift: 1.0370, p-value: 7.49E-1]
M16[rp0-rp1] ==> expr=HSN [Conf: 0.2588, Sup: 0.0751, Lift: 1.0248, p-value: 8.75E-1]
M2[rp0-rp1] ==> expr=ASE [Conf: 0.3095, Sup: 0.1331, Lift: 1.0190, p-value: 8.52E-1]
M16[rp0-rp1] ==> expr=ASE [Conf: 0.3059, Sup: 0.0887, Lift: 1.0070, p-value: 9.6E-1]
M4[rp0-rp1] ==> expr=ASE [Conf: 0.2410, Sup: 0.0683, Lift: 0.7933, p-value: 1.42E-1]
M8[rp0-rp1] ==> expr=ASE [Conf: 0.2400, Sup: 0.0614, Lift: 0.7901, p-value: 1.64E-1]
M5[rp0-rp1] ==> expr=CAN [Conf: 0.3043, Sup: 0.0717, Lift: 1.5924, p-value: 6.23E-3]
M16[rp0-rp1] ==> expr=CAN [Conf: 0.2941, Sup: 0.0853, Lift: 1.5389, p-value: 4.15E-3]
M8[rp0-rp1] ==> expr=CAN [Conf: 0.2400, Sup: 0.0614, Lift: 1.2557, p-value: 2.12E-1]
M2[rp0-rp1] ==> expr=CAN [Conf: 0.2143, Sup: 0.0922, Lift: 1.1212, p-value: 3.81E-1]
M2[rp0-rp1] ==> expr=ALM [Conf: 0.2698, Sup: 0.1160, Lift: 1.3401, p-value: 1.11E-2]
M16[rp0-rp1] ==> expr=ALM [Conf: 0.2471, Sup: 0.0717, Lift: 1.2269, p-value: 2.12E-1]
M5[rp0-rp1] ==> expr=ALM [Conf: 0.2319, Sup: 0.0546, Lift: 1.1516, p-value: 4.7E-1]
M4[rp0-rp1] ==> expr=ALM [Conf: 0.2169, Sup: 0.0614, Lift: 1.0770, p-value: 6.77E-1]
M5[rp0-rp1] ==> expr=ASI [Conf: 0.2464, Sup: 0.0580, Lift: 1.0462, p-value: 8.08E-1]
M16[rp0-rp1] ==> expr=ASI [Conf: 0.2000, Sup: 0.0580, Lift: 0.8493, p-value: 3.6E-1]
M2[rp0-rp1] ==> expr=ASI [Conf: 0.1825, Sup: 0.0785, Lift: 0.7751, p-value: 6.35E-2]
M4[rp0-rp1] ==> expr=ASI [Conf: 0.1807, Sup: 0.0512, Lift: 0.7674, p-value: 1.65E-1]
M8[rp0-rp1] ==> expr=ASH [Conf: 0.2267, Sup: 0.0580, Lift: 1.0063, p-value: 9.73E-1]
M16[rp0-rp1] ==> expr=ASH [Conf: 0.2000, Sup: 0.0580, Lift: 0.8879, p-value: 5.08E-1]
M2[rp0-rp1] ==> expr=ASH [Conf: 0.1825, Sup: 0.0785, Lift: 0.8104, p-value: 1.28E-1]
M2[rp0-rp1] ==> expr=ASH expr=PHA [Conf: 0.1190, Sup: 0.0512, Lift: 0.8944, p-value: 5.38E-1]
M16[rp0-rp1] ==> expr=ADL [Conf: 0.2118, Sup: 0.0614, Lift: 1.0516, p-value: 7.77E-1]
M2[rp0-rp1] ==> expr=ADL [Conf: 0.1270, Sup: 0.0546, Lift: 0.6306, p-value: 5.82E-3]
M2[rp0-rp1] ==> expr=PHA [Conf: 0.1905, Sup: 0.0819, Lift: 0.8859, p-value: 3.74E-1]
M2[rp0-rp1] ==> expr=ASH expr=PHA [Conf: 0.1190, Sup: 0.0512, Lift: 0.8944, p-value: 5.38E-1]
M4[rp0-rp1] ==> expr=EXC [Conf: 0.1807, Sup: 0.0512, Lift: 1.2314, p-value: 3.02E-1]
M2[rp0-rp1] ==> expr=ASK [Conf: 0.1508, Sup: 0.0648, Lift: 0.7751, p-value: 1E-1]
M2[rp0-rp1] ==> expr=ASI expr=ASK [Conf: 0.1190, Sup: 0.0512, Lift: 0.8720, p-value: 4.49E-1]
Default Class Label: null

Classifier contains 31 rules.

50 MOTIF MODEL, CONFIDENCE 0.2

M2[rp0-rp1] ==> expr=ASI [Conf: 0.5128, Sup: 0.0590, Lift: 1.9316, p-value: 2.01E-4]
M16[rp0-rp1] ==> expr=ASI [Conf: 0.4400, Sup: 0.0649, Lift: 1.6573, p-value: 2.47E-3]
M25[rp0-rp1] ==> expr=ASI [Conf: 0.3778, Sup: 0.0501, Lift: 1.4230, p-value: 6.7E-2]
M24[rp0-rp1] ==> expr=ASI [Conf: 0.3659, Sup: 0.0885, Lift: 1.3780, p-value: 1.81E-2]
M2[rp0-rp1] ==> expr=ASE [Conf: 0.4615, Sup: 0.0531, Lift: 1.4901, p-value: 2.93E-2]
M16[rp0-rp1] ==> expr=ASE [Conf: 0.4400, Sup: 0.0649, Lift: 1.4206, p-value: 3.1E-2]
M24[rp0-rp1] ==> expr=ASE [Conf: 0.3537, Sup: 0.0855, Lift: 1.1418, p-value: 3.23E-1]
M49[rp0-rp1] ==> expr=ASE [Conf: 0.3462, Sup: 0.0796, Lift: 1.1176, p-value: 4.28E-1]
M16[rp0-rp1] ==> expr=ASK [Conf: 0.3800, Sup: 0.0560, Lift: 1.8403, p-value: 1.03E-3]
M24[rp0-rp1] ==> expr=ASK [Conf: 0.2561, Sup: 0.0619, Lift: 1.2402, p-value: 2.02E-1]
M31[rp0-rp1] ==> expr=ASK [Conf: 0.2396, Sup: 0.0678, Lift: 1.1603, p-value: 3.44E-1]
M22[rp0-rp1] ==> expr=ASK [Conf: 0.2353, Sup: 0.0590, Lift: 1.1395, p-value: 4.48E-1]
M16[rp0-rp1] ==> expr=PHA [Conf: 0.3800, Sup: 0.0560, Lift: 1.6950, p-value: 4.22E-3]
M22[rp0-rp1] ==> expr=PHA [Conf: 0.2941, Sup: 0.0737, Lift: 1.3119, p-value: 7.41E-2]
M49[rp0-rp1] ==> expr=PHA [Conf: 0.2821, Sup: 0.0649, Lift: 1.2581, p-value: 1.63E-1]
M31[rp0-rp1] ==> expr=PHA [Conf: 0.2604, Sup: 0.0737, Lift: 1.1616, p-value: 3.15E-1]
M11[rp0-rp1] ==> expr=HSN [Conf: 0.3788, Sup: 0.0737, Lift: 1.6677, p-value: 1.05E-3]
M19[rp0-rp1] ==> expr=HSN [Conf: 0.3418, Sup: 0.0796, Lift: 1.5047, p-value: 5.49E-3]
M12[rp0-rp1] ==> expr=HSN [Conf: 0.3413, Sup: 0.1268, Lift: 1.5025, p-value: 1.15E-4]
M42[rp0-rp1] ==> expr=HSN [Conf: 0.3258, Sup: 0.0855, Lift: 1.4346, p-value: 9.65E-3]
M16[rp0-rp1] ==> expr=ASH [Conf: 0.3400, Sup: 0.0501, Lift: 1.5368, p-value: 2.84E-2]
M49[rp0-rp1] ==> expr=ASH [Conf: 0.3077, Sup: 0.0708, Lift: 1.3908, p-value: 3.6E-2]
M24[rp0-rp1] ==> expr=ASH [Conf: 0.2561, Sup: 0.0619, Lift: 1.1576, p-value: 3.82E-1]
M22[rp0-rp1] ==> expr=ASH [Conf: 0.2471, Sup: 0.0619, Lift: 1.1167, p-value: 5.08E-1]
M11[rp0-rp1] ==> expr=CAN [Conf: 0.3182, Sup: 0.0619, Lift: 1.8597, p-value: 4.06E-4]
M31[rp0-rp1] ==> expr=CAN [Conf: 0.3021, Sup: 0.0855, Lift: 1.7656, p-value: 5.69E-5]
M36[rp0-rp1] ==> expr=CAN [Conf: 0.2976, Sup: 0.0737, Lift: 1.7395, p-value: 3.85E-4]
M23[rp0-rp1] ==> expr=CAN [Conf: 0.2500, Sup: 0.0619, Lift: 1.4612, p-value: 2.68E-2]
M33[rp0-rp1] ==> expr=ALM [Conf: 0.3034, Sup: 0.0796, Lift: 1.6859, p-value: 4.16E-4]
M18[rp0-rp1] ==> expr=ALM [Conf: 0.2791, Sup: 0.0708, Lift: 1.5509, p-value: 5.6E-3]
M31[rp0-rp1] ==> expr=ALM [Conf: 0.2708, Sup: 0.0767, Lift: 1.5051, p-value: 6.18E-3]
M3[rp0-rp1] ==> expr=ALM [Conf: 0.2677, Sup: 0.1003, Lift: 1.4878, p-value: 1.13E-3]
M24[rp0-rp1] ==> expr=ADL [Conf: 0.2683, Sup: 0.0649, Lift: 1.2810, p-value: 1.33E-1]
M49[rp0-rp1] ==> expr=ADL [Conf: 0.2564, Sup: 0.0590, Lift: 1.2243, p-value: 2.45E-1]
M18[rp0-rp1] ==> expr=ADL [Conf: 0.2326, Sup: 0.0590, Lift: 1.1104, p-value: 5.42E-1]
M31[rp0-rp1] ==> expr=ADL [Conf: 0.2188, Sup: 0.0619, Lift: 1.0445, p-value: 7.91E-1]
Default Class Label: null

Classifier contains 36 rules.

150 MOTIF MODEL, CONFIDENCE 0.3

M6[rp0-rp1] ==> expr=ASI [Conf: 0.4762, Sup: 0.0588, Lift: 1.7989, p-value: 9.06E-4]
M100[rp0-rp1] ==> expr=ASE [Conf: 0.4667, Sup: 0.0824, Lift: 1.5111, p-value: 3.54E-3]
M38[rp0-rp1] ==> expr=HSN [Conf: 0.4595, Sup: 0.0500, Lift: 2.0288, p-value: 3.35E-4]
M37[rp0-rp1] ==> expr=ASI [Conf: 0.4231, Sup: 0.0647, Lift: 1.5983, p-value: 4.91E-3]
M61[rp0-rp1] ==> expr=ASI [Conf: 0.4231, Sup: 0.0647, Lift: 1.5983, p-value: 4.91E-3]
M80[rp0-rp1] ==> expr=ASI [Conf: 0.3864, Sup: 0.0500, Lift: 1.4596, p-value: 4.99E-2]
M4[rp0-rp1] ==> expr=ASE [Conf: 0.4286, Sup: 0.0529, Lift: 1.3878, p-value: 7.28E-2]
M6[rp0-rp1] ==> expr=ASE [Conf: 0.4286, Sup: 0.0529, Lift: 1.3878, p-value: 7.28E-2]
M37[rp0-rp1] ==> expr=ASE [Conf: 0.4231, Sup: 0.0647, Lift: 1.3700, p-value: 5.27E-2]
M113[rp0-rp1] ==> expr=HSN [Conf: 0.4146, Sup: 0.0500, Lift: 1.8309, p-value: 2.14E-3]
M42[rp0-rp1] ==> expr=HSN [Conf: 0.3750, Sup: 0.0882, Lift: 1.6558, p-value: 2.84E-4]
M43[rp0-rp1] ==> expr=HSN [Conf: 0.3623, Sup: 0.0735, Lift: 1.5998, p-value: 2.53E-3]
M4[rp0-rp1] ==> expr=ASH [Conf: 0.4048, Sup: 0.0500, Lift: 1.8108, p-value: 2.6E-3]
M15[rp0-rp1] ==> expr=ASH [Conf: 0.3429, Sup: 0.0706, Lift: 1.5338, p-value: 7.16E-3]
M61[rp0-rp1] ==> expr=ASH [Conf: 0.3269, Sup: 0.0500, Lift: 1.4626, p-value: 5.18E-2]
M50[rp0-rp1] ==> expr=ASH [Conf: 0.3158, Sup: 0.0882, Lift: 1.4127, p-value: 1.1E-2]
M111[rp0-rp1] ==> expr=CAN [Conf: 0.3774, Sup: 0.0588, Lift: 2.2121, p-value: 1.33E-5]
M43[rp0-rp1] ==> expr=CAN [Conf: 0.3043, Sup: 0.0618, Lift: 1.7841, p-value: 9.38E-4]
M61[rp0-rp1] ==> expr=ASK [Conf: 0.3654, Sup: 0.0559, Lift: 1.7747, p-value: 2E-3]
M100[rp0-rp1] ==> expr=ASK [Conf: 0.3167, Sup: 0.0559, Lift: 1.5381, p-value: 1.94E-2]
M78[rp0-rp1] ==> expr=ASK [Conf: 0.3108, Sup: 0.0676, Lift: 1.5097, p-value: 1.16E-2]
M19[rp0-rp1] ==> expr=ASK [Conf: 0.3065, Sup: 0.0559, Lift: 1.4885, p-value: 3.03E-2]
M61[rp0-rp1] ==> expr=PHA [Conf: 0.3654, Sup: 0.0559, Lift: 1.6346, p-value: 7.63E-3]
M15[rp0-rp1] ==> expr=PHA [Conf: 0.3143, Sup: 0.0647, Lift: 1.4060, p-value: 4.08E-2]
M78[rp0-rp1] ==> expr=PHA [Conf: 0.3108, Sup: 0.0676, Lift: 1.3905, p-value: 4.16E-2]
M33[rp0-rp1] ==> expr=PHA [Conf: 0.3088, Sup: 0.0618, Lift: 1.3816, p-value: 5.91E-2]
M50[rp0-rp1] ==> expr=ADL [Conf: 0.3263, Sup: 0.0912, Lift: 1.5409, p-value: 1.28E-3]
M15[rp0-rp1] ==> expr=ADL [Conf: 0.3143, Sup: 0.0647, Lift: 1.4841, p-value: 1.85E-2]
M36[rp2-rp3] M112[rp0-rp1] ==> expr=ALM [Conf: 0.3208, Sup: 0.0500, Lift: 1.7878, p-value: 3.51E-3]
M46[rp0-rp1] ==> expr=ALM [Conf: 0.3077, Sup: 0.0588, Lift: 1.7150, p-value: 2.73E-3]
M101[rp0-rp1] ==> expr=ALM [Conf: 0.3034, Sup: 0.0794, Lift: 1.6909, p-value: 3.89E-4]
Default Class Label: null

Classifier contains 31 rules.

APPENDIX G – LIST OF GENES EXPRESSED IN EACH CELL TYPE

ADL

WBGene00005430 WBGene00017774 WBGene00000446 WBGene00003000 WBGene00003889 WBGene00005071 WBGene00005149
WBGene00005641 WBGene00003886 WBGene00001665 WBGene00000485 WBGene00000482 WBGene00003669 WBGene00000301
WBGene00003020 WBGene00003036 WBGene00001663 WBGene00001673 WBGene00001677 WBGene00006654 WBGene00001949
WBGene00003884 WBGene00000453 WBGene00004265 WBGene00003745 WBGene00003746 WBGene00003748 WBGene00004782
WBGene00006527 WBGene00001681 WBGene00004215 WBGene00003838 WBGene00003839 WBGene00002081 WBGene00018746
WBGene00006895 WBGene00000289 WBGene00000584 WBGene00002242 WBGene00000424 WBGene00000390 WBGene00006830
WBGene00001464 WBGene00004789 WBGene00001447 WBGene00001667 WBGene00001668 WBGene00011289 WBGene00007883
WBGene00016397 WBGene00001127 WBGene00019482 WBGene00014159 WBGene00021915 WBGene00018727 WBGene00010898
WBGene00010742 WBGene00019641 WBGene00011261 WBGene00005563 WBGene00005349 WBGene00002235 WBGene00001119
WBGene00006655 WBGene00001118 WBGene00008765 WBGene00004346 WBGene00001546 WBGene00014025 WBGene00002141
WBGene00000567 WBGene00012352

ALM

WBGene00003167 WBGene00000951 WBGene00001135 WBGene00003171 WBGene00002178 WBGene00002177 WBGene00001130
WBGene00003909 WBGene00006528 WBGene00003003 WBGene00006818 WBGene00006826 WBGene00001619 WBGene00006768
WBGene00001210 WBGene00000897 WBGene00003738 WBGene00000482 WBGene00000289 WBGene00001171 WBGene00000301
WBGene00003020 WBGene00003036 WBGene00006593 WBGene00006805 WBGene00003239 WBGene00003174 WBGene00004032
WBGene00001648 WBGene00003166 WBGene00003168 WBGene00004212 WBGene00004782 WBGene00004732 WBGene00001172
WBGene00003931 WBGene00002081 WBGene00003170 WBGene00003471 WBGene00003172 WBGene00000424 WBGene00000390
WBGene00001189 WBGene00006462 WBGene00003403 WBGene00001052 WBGene00000901 WBGene00001463 WBGene00000443
WBGene00002233 WBGene00002235 WBGene00017157 WBGene00044330 WBGene00016158 WBGene00003243 WBGene00003144
WBGene00006802 WBGene00006772 WBGene00017644 WBGene00015735 WBGene00004457

ASE

WBGene00017774 WBGene00000446 WBGene00001171 WBGene00001532 WBGene00003563 WBGene00003889 WBGene00006526
WBGene00006745 WBGene00003886 WBGene00001665 WBGene00000485 WBGene00000482 WBGene00000301 WBGene00003020
WBGene00003036 WBGene00003969 WBGene00003884 WBGene00003741 WBGene00003745 WBGene00003752 WBGene00004782
WBGene00006527 WBGene00006525 WBGene00001841 WBGene00002081 WBGene00018746 WBGene00003807 WBGene00000584
WBGene00000812 WBGene00005077 WBGene00002242 WBGene00003403 WBGene00000424 WBGene00000390 WBGene00000483
WBGene00001173 WBGene00006773 WBGene00001464 WBGene00000457 WBGene00001448 WBGene00001449 WBGene00001456
WBGene00001463 WBGene00000895 WBGene00007883 WBGene00019482 WBGene00014159 WBGene00021915 WBGene00018727
WBGene00010898 WBGene00010742 WBGene00019641 WBGene00011261 WBGene00003561 WBGene00004032 WBGene00000563
WBGene00017157 WBGene00000995 WBGene00001119 WBGene00001096 WBGene00010453 WBGene00002084 WBGene00008765
WBGene00001547 WBGene00001528 WBGene00001530 WBGene00001531 WBGene00001544 WBGene00007314 WBGene00006772
WBGene00016905 WBGene00001954 WBGene00000459 WBGene00000095 WBGene00001440 WBGene00022639 WBGene00044068
WBGene00003640 WBGene00005187 WBGene00005967 WBGene00019999 WBGene00005111 WBGene00000953 WBGene00009144
WBGene00012116 WBGene00000846 WBGene00002105 WBGene00010575 WBGene00004831 WBGene00020506 WBGene00011382
WBGene00006978 WBGene00044020 WBGene00010357 WBGene00011289 WBGene00010236 WBGene00021152 WBGene00001533
WBGene00001534 WBGene00002988 WBGene00003088 WBGene00001447 WBGene00001075 WBGene00001540 WBGene00001545
WBGene00020838

ASH

WBGene00017774 WBGene00000446 WBGene00003890 WBGene00003889 WBGene00005032 WBGene00005071 WBGene00006748
WBGene00003886 WBGene00001665 WBGene00000485 WBGene00000482 WBGene00003669 WBGene00001135 WBGene00000301
WBGene00003020 WBGene00003036 WBGene00001663 WBGene00001673 WBGene00001675 WBGene00001676 WBGene00001677
WBGene00006778 WBGene00003850 WBGene00001949 WBGene00003884 WBGene00004265 WBGene00003741 WBGene00003753
WBGene00004782 WBGene00002210 WBGene00001172 WBGene00006527 WBGene00001681 WBGene00004215 WBGene00003839
WBGene00002081 WBGene00018746 WBGene00003807 WBGene00000289 WBGene00002242 WBGene00003403 WBGene00000424
WBGene00000390 WBGene00000483 WBGene00001173 WBGene00001464 WBGene00003000 WBGene00001709 WBGene00001668
WBGene00011289 WBGene00007883 WBGene00016397 WBGene00001127 WBGene00019482 WBGene00014159 WBGene00021915
WBGene00018727 WBGene00010898 WBGene00019641 WBGene00011261 WBGene00002235 WBGene00004032 WBGene00017157
WBGene00000995 WBGene00001119 WBGene00006655 WBGene00006830 WBGene00001118 WBGene00003182 WBGene00001837
WBGene00008765 WBGene00003366 WBGene00001841 WBGene00004346 WBGene00007801 WBGene00002141 WBGene00001075

ASI

WBGene00000446 WBGene00000903 WBGene00003848 WBGene00003890 WBGene00003889 WBGene00005032 WBGene00005079
WBGene00006526 WBGene00006743 WBGene00003886 WBGene00000907 WBGene00001665 WBGene00002048 WBGene00000485
WBGene00000482 WBGene00006070 WBGene00000301 WBGene00003020 WBGene00003036 WBGene00001663 WBGene00001667
WBGene00001666 WBGene00001668 WBGene00001672 WBGene00001676 WBGene00006654 WBGene00003884 WBGene00000262
WBGene00003739 WBGene00003765 WBGene00003743 WBGene00003744 WBGene00003745 WBGene00003747 WBGene00003752
WBGene00003756 WBGene00003762 WBGene00006071 WBGene00004782 WBGene00006980 WBGene00006981 WBGene00006942
WBGene00002210 WBGene00006527 WBGene00001681 WBGene00006525 WBGene00002181 WBGene00002081 WBGene00000289
WBGene00000424 WBGene00000390 WBGene00000920 WBGene00001527 WBGene00004804 WBGene00004789 WBGene00001445
WBGene00001453 WBGene00001464 WBGene00011289 WBGene00007883 WBGene00016397 WBGene00001127 WBGene00019482
WBGene00014159 WBGene00021915 WBGene00018727 WBGene00010898 WBGene00010742 WBGene00019641 WBGene00011261
WBGene00002235 WBGene00000563 WBGene00017157 WBGene00001119 WBGene00006655 WBGene00000767 WBGene00001118
WBGene00002084 WBGene00008765 WBGene00004346 WBGene00001528 WBGene00001529 WBGene00001530 WBGene00001550
WBGene00014025 WBGene00007801 WBGene00001968 WBGene00002141 WBGene00000567 WBGene00020838

ASK

WBGene00017774 WBGene00003848 WBGene00003889 WBGene00005033 WBGene00005035 WBGene00005160 WBGene00005166
WBGene00006526 WBGene00003886 WBGene00000907 WBGene00001665 WBGene00002048 WBGene00000485 WBGene00000482
WBGene00001135 WBGene00000301 WBGene00003020 WBGene00003036 WBGene00001676 WBGene00001677 WBGene00003884
WBGene00000262 WBGene00003746 WBGene00003748 WBGene00003752 WBGene00004782 WBGene00006981 WBGene00006982
WBGene00002210 WBGene00006527 WBGene00006525 WBGene00002081 WBGene00018746 WBGene00000289 WBGene00002242
WBGene00000424 WBGene00000390 WBGene00001173 WBGene00006830 WBGene00004789 WBGene00000096 WBGene00001456
WBGene00000895 WBGene00011289 WBGene00007883 WBGene00016397 WBGene00001127 WBGene00019482 WBGene00014159
WBGene00021915 WBGene00018727 WBGene00010898 WBGene00010742 WBGene00019641 WBGene00011261 WBGene00001119
WBGene00006655 WBGene00001118 WBGene00008765 WBGene00004346 WBGene00001550 WBGene00014025 WBGene00007801
WBGene00001968 WBGene00002141 WBGene00000567 WBGene00006772 WBGene00003377 WBGene00020838 WBGene00001075

CAN

WBGene00010315 WBGene00017774 WBGene00000295 WBGene00000446 WBGene00001587 WBGene00002178 WBGene00002177
WBGene00001824 WBGene00000069 WBGene00000068 WBGene00000936 WBGene00003003 WBGene00001745 WBGene00003911
WBGene00002048 WBGene00000527 WBGene00000482 WBGene00000289 WBGene00000301 WBGene00003020 WBGene00003036
WBGene00001672 WBGene00001676 WBGene00006852 WBGene00006874 WBGene00006805 WBGene00003239 WBGene00001648
WBGene00000435 WBGene00006808 WBGene00000463 WBGene00003748 WBGene00003753 WBGene00000424 WBGene00004782
WBGene00004732 WBGene00002181 WBGene00004215 WBGene00002081 WBGene00018746 WBGene00000390 WBGene00006462
WBGene00000037 WBGene00004777 WBGene00001324 WBGene00001325 WBGene00006775 WBGene00006498 WBGene00017157
WBGene00016872 WBGene00016158 WBGene00003243 WBGene00001121 WBGene00006802 WBGene00001365 WBGene00001371
WBGene00010700 WBGene00020838

HSN

WBGene00000295 WBGene00000395 WBGene00000481 WBGene00001587 WBGene00001821 WBGene00003238 WBGene00002178
WBGene00002177 WBGene00006756 WBGene00006776 WBGene00006748 WBGene00006818 WBGene00001824 WBGene00001695
WBGene00001174 WBGene00001207 WBGene00001745 WBGene00004729 WBGene00001616 WBGene00002048 WBGene00001208
WBGene00001210 WBGene00004773 WBGene00006600 WBGene00003738 WBGene00000482 WBGene00004345 WBGene00001518
WBGene00000530 WBGene00001470 WBGene00000289 WBGene00000301 WBGene00003020 WBGene00003036 WBGene00001145
WBGene00006805 WBGene00003239 WBGene00006753 WBGene00006786 WBGene00001648 WBGene00006808 WBGene00003741
WBGene00003753 WBGene00004782 WBGene00001172 WBGene00002181 WBGene00006788 WBGene00002126 WBGene00002081
WBGene00003114 WBGene00003733 WBGene00003170 WBGene00000424 WBGene00000390 WBGene00001189 WBGene00006365
WBGene00006830 WBGene00006742 WBGene00001462 WBGene00003395 WBGene00001325 WBGene00000239 WBGene00001211
WBGene00002203 WBGene00000443 WBGene00006498 WBGene00004254 WBGene00002062 WBGene00016158 WBGene00003404
WBGene00004767 WBGene00000053 WBGene00006802 WBGene00003557 WBGene00003243 WBGene00020142 WBGene00001950

PHA

WBGene00017774 WBGene00000438 WBGene00000446 WBGene00001538 WBGene00003000 WBGene00003890 WBGene00003889
WBGene00005071 WBGene00005170 WBGene00006776 WBGene00003886 WBGene00001663 WBGene00001664 WBGene00001665
WBGene00001207 WBGene00002048 WBGene00003563 WBGene00000900 WBGene00000485 WBGene00003745 WBGene00003752
WBGene00000482 WBGene00000484 WBGene00000301 WBGene00003020 WBGene00003036 WBGene00001675 WBGene00001676
WBGene00001677 WBGene00004032 WBGene00001648 WBGene00003884 WBGene00000262 WBGene00004782 WBGene00006527
WBGene00004215 WBGene00003839 WBGene00002081 WBGene00018746 WBGene00003807 WBGene00000424 WBGene00000390
WBGene00006830 WBGene00004789 WBGene00005643 WBGene00001447 WBGene00001450 WBGene00001458 WBGene00000193
WBGene00021349 WBGene00003850 WBGene00001668 WBGene00020607 WBGene00005065 WBGene00011289 WBGene00007883

WBGene00016397 WBGene00001127 WBGene00019482 WBGene00021915 WBGene00018727 WBGene00010898 WBGene00011261 WBGene00003561 WBGene00002235 WBGene00001119 WBGene00006655 WBGene00000767 WBGene00004346 WBGene00001542 WBGene00007801 WBGene00001968 WBGene00000567 WBGene00001505 WBGene00012352 WBGene00015735

EXC

WBGene00000002 WBGene00000013 WBGene00000156 WBGene00000170 WBGene00000171 WBGene00000176 WBGene00000184 WBGene00000229 WBGene00000255 WBGene00000301 WBGene00000390 WBGene00000395 WBGene00000396 WBGene00000424 WBGene00000431 WBGene00000448 WBGene00000458 WBGene00000459 WBGene00000477 WBGene00000482 WBGene00000530 WBGene00000531 WBGene00000533 WBGene00000567 WBGene00000802 WBGene00000833 WBGene00000874 WBGene00000905 WBGene00000911 WBGene00000914 WBGene00000958 WBGene00000961 WBGene00001001 WBGene00001027 WBGene00001030 WBGene00001124 WBGene00001198 WBGene00001228 WBGene00001231 WBGene00001234 WBGene00001331 WBGene00001333 WBGene00001366 WBGene00001368 WBGene00001394 WBGene00001431 WBGene00001468 WBGene00001621 WBGene00001624 WBGene00001651 WBGene00001674 WBGene00001745 WBGene00001747 WBGene00001792 WBGene00001796 WBGene00001834 WBGene00001874 WBGene00001976 WBGene00002005 WBGene00002038 WBGene00002050 WBGene00002053 WBGene00002062 WBGene00002065 WBGene00002081 WBGene00002134 WBGene00002135 WBGene00002148 WBGene00002173 WBGene00002181 WBGene00002189 WBGene00002201 WBGene00002202 WBGene00002220 WBGene00002295 WBGene00002855 WBGene00003002 WBGene00003011 WBGene00003020 WBGene00003035 WBGene00003036 WBGene00003047 WBGene00003154 WBGene00003164 WBGene00003170 WBGene00003218 WBGene00003408 WBGene00003549 WBGene00003561 WBGene00003616 WBGene00003633 WBGene00003647 WBGene00003648 WBGene00003655 WBGene00003681 WBGene00003722 WBGene00003723 WBGene00003733 WBGene00003736 WBGene00003807 WBGene00003901 WBGene00003905 WBGene00003997 WBGene00003998 WBGene00004005 WBGene00004006 WBGene00004025 WBGene00004048 WBGene00004088 WBGene00004117 WBGene00004134 WBGene00004215 WBGene00004270 WBGene00004308 WBGene00004344 WBGene00004358 WBGene00004368 WBGene00004410 WBGene00004419 WBGene00004420 WBGene00004421 WBGene00004424 WBGene00004431 WBGene00004435 WBGene00004436 WBGene00004446 WBGene00004448 WBGene00004461 WBGene00004463 WBGene00004464 WBGene00004486 WBGene00004490 WBGene00004492 WBGene00004496 WBGene00004498 WBGene00004510 WBGene00004766 WBGene00004782 WBGene00004788 WBGene00004801 WBGene00004859 WBGene00004862 WBGene00004902 WBGene00004918 WBGene00005657 WBGene00005712 WBGene00006351 WBGene00006352 WBGene00006410 WBGene00006433 WBGene00006438 WBGene00006462 WBGene00006483 WBGene00006490 WBGene00006524 WBGene00006537 WBGene00006610 WBGene00006626 WBGene00006688 WBGene00006698 WBGene00006788 WBGene00006804 WBGene00006805 WBGene00006839 WBGene00006887 WBGene00006910 WBGene00006911 WBGene00006912 WBGene00006913 WBGene00006914 WBGene00006917 WBGene00006920 WBGene00006921 WBGene00006924 WBGene00006941 WBGene00007059 WBGene00007138 WBGene00007144 WBGene00007287 WBGene00007429 WBGene00007443 WBGene00007549 WBGene00007561 WBGene00007880 WBGene00008195 WBGene00008729 WBGene00009119 WBGene00009174 WBGene00009323 WBGene00009337 WBGene00009553 WBGene00009587 WBGene00009617 WBGene00009632 WBGene00009783 WBGene00009800 WBGene00009876 WBGene00009882 WBGene00009929 WBGene00010043 WBGene00010260 WBGene00010405 WBGene00010671 WBGene00010681 WBGene00010757 WBGene00010778 WBGene00010788 WBGene00010789 WBGene00010809 WBGene00010911 WBGene00011043 WBGene00011099 WBGene00011112 WBGene00011115 WBGene00011128 WBGene00011228 WBGene00011263 WBGene00011408 WBGene00011520 WBGene00011580 WBGene00011587 WBGene00011679 WBGene00011687 WBGene00011872 WBGene00011887 WBGene00012005 WBGene00012097 WBGene00012121 WBGene00012184 WBGene00012205 WBGene00012348 WBGene00012351 WBGene00012487 WBGene00012896 WBGene00012969 WBGene00013025 WBGene00013039 WBGene00013143 WBGene00013355 WBGene00013460 WBGene00013499 WBGene00013595 WBGene00013963 WBGene00013996 WBGene00014023 WBGene00014025 WBGene00014083 WBGene00015061 WBGene00015160 WBGene00015181 WBGene00015258 WBGene00015286 WBGene00015344 WBGene00015391 WBGene00015684 WBGene00015702 WBGene00015897 WBGene00016258 WBGene00016496 WBGene00016676 WBGene00016789 WBGene00016945 WBGene00016990 WBGene00017004 WBGene00017774 WBGene00017891 WBGene00018206 WBGene00018285 WBGene00018411 WBGene00018604 WBGene00018610 WBGene00018735 WBGene00018740 WBGene00018746 WBGene00018758 WBGene00018866 WBGene00018976 WBGene00019234 WBGene00019521 WBGene00019607 WBGene00019747 WBGene00019824 WBGene00019999 WBGene00020026 WBGene00020052 WBGene00020066 WBGene00020097 WBGene00020142 WBGene00020146 WBGene00020207 WBGene00020485 WBGene00020504 WBGene00020596 WBGene00020647 WBGene00020906 WBGene00021004 WBGene00021292 WBGene00021369 WBGene00021444 WBGene00021637 WBGene00021800 WBGene00021811 WBGene00021883 WBGene00021929 WBGene00022127 WBGene00022423 WBGene00022448 WBGene00022554 WBGene00022603 WBGene00022673 WBGene00044072 WBGene00044077

APPENDIX H – LIST OF MOTIFS

The following list contains the 400 motifs that were mined using Weeder. In each line, the set of letters is the DNA sequence mined. The number is a value assigned by Weeder indicating the frequency of occurrence for that motif.

ATTTTT 1.73
AAAAAA 0.60
AAAATT 0.55
TAATAC 0.50
TAGTAA 0.46
ACATCA 0.46
TTCCTA 0.45
ATGATG 0.45
TGTCAT 0.44
TTGATC 0.44
TGCATATC 0.66
CCATGACA 0.65
CCTAACTC 0.64
TACCTGCA 0.62
CCTAAATG 0.62
TTCACCTG 0.60
CATCTTGC 0.59
CTCATTGC 0.58
TGGGTGAA 0.57
ATTTAGGT 0.56
CCAAGGCAAC 0.81
CCATGACAAC 0.74
CATAGACTCC 0.69
GTAGTCTATG 0.67
TCCCTACTGA 0.67
TTACACCTGC 0.66
TGCAGGTAAC 0.64
TCCTCTATCC 0.61
TACCCTCATT 0.59
CAGGCGAAAC 0.59
GAGAATTAGAGA 1.71
AGAGAATTAGAG 1.69
GAGAGAAGGAGA 1.56
GAAGGTGAAGAA 1.51
AAATTAAAGAAA 1.49
AAATCAGGGAAA 1.49
GAGAGAGGAGAC 1.49
AAAATGAGTGGA 1.48
AGAGAAAAAGGA 1.47
AAAATTAAAGAA 1.46

ATTTTT 1.53
GATGAG 0.64
GAAGAC 0.64
AGAAGG 0.60
GAGACA 0.59
CCCAAT 0.56
AAAAAA 0.54
CTTCCT 0.52
CATCTT 0.51
CTCTTC 0.51
TCCATCCT 0.78
ATCCATCC 0.73
CATCCATC 0.72
GAAGACGT 0.72
AGGGAAGC 0.70
GAGGGAAG 0.69
TCATGTCC 0.68
CCTCCATC 0.68
CTCTCCAC 0.67
GTCGTCTT 0.67
TCACCCATCC 0.79
TGTCACACCC 0.79
CCCTCTCCAC 0.77
CCTCATATCC 0.77
GTCATCCATC 0.75
TCATCCATCC 0.75
GCGCTTCCTC 0.75
GTGTGTACGT 0.75
CCATCTCGTC 0.74
TCTATCCCTC 0.73
GAATAAAGGGAA 1.89
AAGAGAGAAAGA 1.85
AGAATAAAGGGA 1.82
AAAGAGAAGAAG 1.81
AGAGAGAAAGAG 1.80
GAAGAGAGAAAG 1.75
AAGAGAAGAAGC 1.74
GAAAGAAAGAAG 1.72
GAAAGAGAAGAA 1.72
AGTAGAAGAAGA 1.72

AAAAAT 1.60
AGAAGC 0.57
TTTTTT 0.48
GAGAAG 0.46
TGATGT 0.46
CTTCAT 0.44
ATCTTC 0.44
CATCTT 0.43
TAGTAA 0.43
CATCAA 0.42
CCCTTCAT 0.63
CGGCTTCT 0.63
ATGGCTTC 0.62
GAGAAGCC 0.61
AAGAAGCC 0.60
ATCTTCCC 0.57
TCCCATCA 0.57
CATGAGTT 0.57
GAGTTCCA 0.55
CCTCCTAA 0.55
GAGGGTGGAG 0.73
GGAGCTCAAG 0.66
GAGAAGCCGT 0.64
TCCCTTCATC 0.61
AGAAGCCGTA 0.60
CTTCAGCTCC 0.59
TCTGACGCAT 0.59
GGAACTCATG 0.59
AGAGGGTGGA 0.59
ATCTAGAGCT 0.57
AAGGAAAAGGAG 1.54
AGAAGGATAGAA 1.47
GAAATGAAGAGA 1.47
ATGAAGGAGTAA 1.46
GAGAAAGAGAAG 1.44
AAGAATGAAAAC 1.44
GAGAGAAGGAGA 1.43
GAGAAGGAGATG 1.43
GAGAAAAAGGAG 1.43
GAAAAGAAGACG 1.43

AAAAAT 1.64
CATCTT 0.54
GCTTCT 0.51
TAGTAA 0.50
TTTTTT 0.50
TAGGAA 0.45
AATTTT 0.45
ATGTGA 0.44
CCTATT 0.43
GTAATA 0.42
CTGCTCTC 0.65
TGCTCTCC 0.61
GTCTTCAC 0.60
GAAGTCTG 0.59
AGACTTCC 0.58
GTCGTCTT 0.58
TCTCAGTC 0.58
ATCTGGAT 0.57
CCATCTTC 0.57
ATGTCATC 0.56
ACTGCTCTCC 0.65
CTTACCCTCT 0.65
CTGCTCTCCA 0.62
GTAGTAAATG 0.62
GTAAGTGTGG 0.62
TCGCTCACAT 0.60
CTGCTCTCCC 0.60
TCTTAAGATC 0.59
CTTCCCAGCT 0.58
CATCTTACCT 0.58
AGAGAATTAGAG 1.73
GAGAATTAGAGA 1.72
GAGAGAAGGAGA 1.72
AGAGAAGGAGAG 1.71
GAGAAGAAGAGA 1.68
AGAATAAGAAGA 1.67
AGAGAAGAAGAG 1.63
GAGAGAAGAAGA 1.61
GAAAAGAAGGGG 1.59
GAGAAGAGAGGA 1.58
ATTTTT 1.64
ACTCAT 0.54
CTCTCA 0.51
TCACTC 0.50
CTTGAT 0.49
AAAAAA 0.48
GCTTCT 0.46
CTCATT 0.44
ATCTTC 0.43
CATCTT 0.42
AGCTCAAC 0.76
GTCTGACT 0.67
GAGTCAGA 0.61
CCTCATTA 0.61
CTCATGGT 0.60
CGAGTTCC 0.60
TCTCATGG 0.59
ATGAGTGG 0.59
GAGGTCTT 0.58
ATAAGCAC 0.57
TCGTCTGACT 0.70
GCTAATGAGC 0.67
GTCTGACTCA 0.66
CACGAGTTCC 0.65
CGTCTGACTC 0.65
GCTCATTAAC 0.63
GCTCATTCAC 0.62
GTTGAGCTCC 0.60
CTTAACTCCT 0.60
CATTCAAGGC 0.60
GAGAGGAAGAGA 1.67
GGAGAGAATGAG 1.64
GAGAGAAGGAGA 1.61
GAGAGTAAGAGA 1.60
GAGAGAGGAAGA 1.60
GAGACAGAGGAA 1.55
GAGAGAAGAAGA 1.53
GAGATGTAGAGA 1.52
AGAGAAGGAGAG 1.50
AGAGAGGAAGAG 1.50
ATTTTT 1.68
AAAAAA 0.56
ATGTCT 0.50
TATTAC 0.48
AGAAGC 0.47
ATCAAG 0.44
TTTAGC 0.43
ATGATG 0.43
CATTCA 0.43
TCATTC 0.42
CTCGTATA 0.76
CTCAAGCA 0.64
GGGTGTAT 0.64
TGGATTAG 0.63
ACCTGCAA 0.62
CCTAAATG 0.60
CATGAGGA 0.60
GTCAACCA 0.59
CCAAAGGA 0.59
TATCCGTA 0.58
GTGACTGAGA 0.70
CTTGATGACC 0.69
GCTCAAGCAC 0.69
CTTACCCTCT 0.69
TGAGTCACAC 0.68
CATGACAACG 0.68
TGTGCATCTG 0.65
CATGAAACCC 0.65
TACCCTCATT 0.65
AGGAGTTACG 0.65
GAGAGTAAGAGA 1.68
GAGAGAAGGAGA 1.68
AGAGAAGGAGAG 1.59
GAGAGAGGAAGA 1.58
GAGAGGAAGAGA 1.57
AGAGTGAGAGAA 1.57
TGAGAGAAGGAG 1.55
AGAGAGGAAGAG 1.55
GAGAGAAGAAGA 1.52
GAAGGGAACGGG 1.52
ATTTTT 1.50
CTCATC 0.66
GTCTTC 0.63
GCTCTT 0.59
CCTTCT 0.56
AAAAAA 0.53
CTTCTT 0.52
TCCTTC 0.51
GAGAGA 0.51
AATCGT 0.48
ACGTCTTC 0.83
CACACCCA 0.79
CGTTCCCT 0.75
TCGTCCTT 0.74
AGACGTTA 0.74
TCGTTACC 0.73
GCTTGAGT 0.69
CCTCATCT 0.68
TCGTAGTC 0.68
TCACAGAC 0.68
CTCACCCCCA 0.86

CGCTTGTCTC 0.84
TGCGTATGTG 0.82
TCACTCACTC 0.81
TCGTCCTTTC 0.80
GACGAAGACG 0.79
GAGGGTGAGG 0.79
TCACACCCAC 0.79
CTCCTGCATC 0.79
CTTCGCTTCT 0.77
AGAGAGGAAGAG 2.05
GAAGAAGATGAA 2.02
GAGAGTAAGAGA 2.00
AGAGAGAAAGAG 1.97
AGAAGAAGATGA 1.97
AGAAAGAGAAGA 1.96
AAAGAGAAGAAG 1.95
GAAAGAGAAGAA 1.93
GAGAGGAAGAGA 1.92
GAGAGAAAAAGA 1.91
ATTTTT 1.73
TTTTTT 0.56
CTCTTC 0.40
AATCAC 0.38
ATTGCA 0.36
GATTAA 0.36
TTCATC 0.35
TCAATG 0.34
TCGATT 0.34
TTCTCG 0.34
TCGTTGCA 0.45
TCGATTAC 0.43
CCGATTAA 0.42
TCACGTTG 0.42
TGCGTTAT 0.42
CTTTCCGC 0.41
GCAAGAAC 0.41
CAAGGAGA 0.41
CTCCATTC 0.40
TCGATGTG 0.40
CGTAGTCTTC 0.49
GCAAACGAGC 0.47
AGAGACCGAG 0.46
TAATGCATTA 0.45
AACGATCGGA 0.45
CATGCGTCTT 0.44
GCAAACACGC 0.43
TTCGACGGTT 0.42
AACTGCGTCA 0.42
GAGAGACCGA 0.42
GAGAGGAAGAGA 1.55
AAAATAGAGAAA 1.52
AAATAGAGAAAA 1.51
AAGAGAAACAGA 1.49
AAAAAGGAGAAA 1.46
AAATAGAGAGAA 1.46
GAGAGAAAGAGG 1.45
GAAAAATGGAGA 1.43
AGAGAAACAGAC 1.42
AGAGACGGAGAG 1.41
AAAAAT 1.49
GTCTTC 0.59
GTGTTC 0.55
CATCTT 0.53
CTTCTT 0.50
CTCTTC 0.48
TTTTTT 0.46
TCTCTA 0.45
GTGAGA 0.45
GTTCTT 0.45
AAGGATGG 0.75
TTCCATCC 0.68
TGAGGGGA 0.67
TCATCCGT 0.67
TCCATCCT 0.67
GTCTTCCT 0.65
CATCCATC 0.63
TGTCTTCC 0.63
TGATGGAG 0.63
GTGTCTTC 0.63
CTCAGTCACT 0.72
GATGAGGGGA 0.71
CCTTACCATC 0.71
CCTTCCATCC 0.71
GAGTGGAAGA 0.71
CCTCTGCTTC 0.69
GTCATCCATC 0.69
CTCTTCCACT 0.68
TGTGGGCGTG 0.68
GAGTGAGTGA 0.68
AGAGAGGAAGAG 2.14
GAGAGGAAGAGA 2.05
AAAGAAAGAAGA 1.95
GAGAACGAGAGA 1.94
AGAGAGAAAGAG 1.94
GAGAGTAAGAGA 1.94
GAAGGAAAGAAG 1.93
GAGAGAGGAAGA 1.92
AAGAGAGAAAGA 1.92
GAGAACTAGAGA 1.89
ATTTTT 1.81
TTTTTT 0.63
AATTTT 0.57
AGAAGC 0.52
TGGAGA 0.49
ACATCT 0.49
CATCTT 0.48
CTCTCA 0.44
ATTCAC 0.43
TATTAG 0.43
CTGCTCTC 0.66
CCTCATCT 0.63
TCTCCATG 0.63
CCCTTCAT 0.60
GCCAACCA 0.60
ATATCAGG 0.60
TAGATGTG 0.59
GTCTTTAG 0.59
GCTCTCCA 0.59
TGCTCTCC 0.57
CTCGTATACT 0.76
CACTCGTATA 0.71
ACTCGTATAC 0.70
ACGCCTTCAT 0.69
GGTTGTCATG 0.66
GTTGTCATGG 0.65
CCTTAACTAC 0.64
CTGCTCTCCA 0.63
CCTGCTCTCC 0.63
ATCTCCATGA 0.62
AGAGAGAATGAA 1.88
AAAAAGAGAGAA 1.72
GAGAGAAGGAGA 1.68
CAGAGAGAATGA 1.66
AGAGAAGGAGAG 1.65
GAGAGTAAGAGA 1.64
GGAGAGAATGAG 1.59
AAAAGAGAGAAT 1.56
AGAGAAAGAAGA 1.54
GAGAGGAAGAGA
1.53GAGAGGAAGAGA 1.53 3

8. REFERENCES

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Very Large Data Base Conference* (pp. 487-499). Santiago, Chile: Very Large Data Base Endowment.

Ball, M. P. (2007, March 28). *File:DNA chemical structure.svg*. Retrieved March 1, 2010, from Wikimedia.org: http://commons.wikimedia.org/wiki/File:DNA_chemical_structure.svg

Forluvoft. (2007, March 16). *File:Gene2-plain.svg*. Retrieved March 1, 2010, from Wikimedia Commons: http://commons.wikimedia.org/wiki/File:Gene2-plain.svg

Griffiths, A., Wessler, S., Lewontin, R., & Carroll, S. (2008). *Introduction to Genetic Analysis.* New York, NY: W. H. Freeman and Company.

Gyll. (2008, January 1). *File:C elegans anatomy.png*. Retrieved March 1, 2010, from Wikimedia Commons: http://commons.wikimedia.org/wiki/File:C_elegans_anatomy.png

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations , 11* (1).

Icev, A., Ruiz, C., & Ryder, E. (2003). *Distance-Enhanced Association Rules for Gene Expression.*

Palanisamy, S. K. (2006). *Association Rule Based Classi.* Worcester, MA: Worcester Polytechnic Institute.

Pavesi, G. (2009, October 20). *Weeder 1.4.2*. Retrieved January 15, 2010, from Tools for MOtif Discovery in nucleotide sequences: http://159.149.109.9/modtools/downloads/weeder.html

Pray, K. (2004). *AprioriSetsAndSequences: Mining Association Rules from Time Sequence Attributes.* Worcester, MA: Worcester Polytechnic Institute.

Rudolph, J. E. (2009). *Gene Expression Rule Modeling.* Worcester, MA: Worcester Polytechnic Institute.

Stolzar, L. (2006). *Bioinformatics ISP.* Worcester, MA: Worcester Polytechnic Institute.

Thakkar, D. (2007). *Hypothesis-Driven Specialization-based Analysis of Gene Expression Association Rules.* Worcester, MA: Worcester Polytechnic Institute.

Tompa, M. (2005). Assessing computational tools for the discovery of transcription factor binding sites. *Nature Publishing Group* , 137-144.

WormBase web site, http://www.wormbase.org, Release WS210, Date January 15, 2010