



ELSEVIER

Information Sciences xxx (2002) xxx–xxx

INFORMATION
SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Interpreting microarray expression data using text annotating the genes

Michael Molla ^{a,*}, Peter Andreae ^{a,1}, Jeremy Glasner ^b,
Frederick Blattner ^b, Jude Shavlik ^{a,c}

^a Department of Computer Sciences, University of Wisconsin—Madison, Madison, WI, USA

^b Department of Genetics, University of Wisconsin—Madison, Madison, WI, USA

^c Department of Biostatistics and Medical Informatics, University of Wisconsin—Madison,
Madison, WI, USA

Received 1 June 2001; received in revised form 2 November 2001; accepted 1 February 2002

Abstract

12 Microarray expression data is being generated by the gigabyte all over the world with
13 undoubted exponential increases to come. Annotated genomic data is also rapidly
14 pouring into public databases. Our goal is to develop automated ways of combining
15 these two sources of information to produce insight into the operation of cells under
16 various conditions. Our approach is to use machine-learning techniques to identify
17 characteristics of genes that are up-regulated or down-regulated in a particular micro-
18 array experiment. We seek models that are (a) accurate, (b) easy to interpret, and (c)
19 stable to small variations in the training data. This paper explores the effectiveness of
20 two standard machine-learning algorithms for this task: Naïve Bayes (based on prob-
21 ability) and PFOIL (based on building rules). Although we do not anticipate using our
22 learned models to predict expression levels of genes, we cast the task in a predictive
23 framework, and evaluate the quality of the models in terms of their predictive power on
24 genes held out from the training. The paper reports on experiments using actual *E. coli*
25 microarray data, discussing the strengths and weaknesses of the two algorithms and
26 demonstrating the trade-offs between accuracy, comprehensibility, and stability.

27 © 2002 Published by Elsevier Science Inc.

28

* Corresponding author.

E-mail addresses: molla@cs.wisc.edu (M. Molla), pondy@cs.wisc.edu (P. Andreae), jeremy@genome.wisc.edu (J. Glasner), fred@genome.wisc.edu (F. Blattner), shavlik@cs.wisc.edu (J. Shavlik).

¹ On leave from Victoria University of Wellington, NZ.

29 1. Introduction

30 RNA is the medium by which an organism's genes produce specific proteins,
31 which are the building blocks of life. An understanding of how an organism
32 regulates the production of specific RNA sequences is crucial to an under-
33 standing of the mechanism by which that organism functions. The *expression*
34 *level* of a gene is a measure of the amount of RNA being produced by that gene
35 at a particular time. *Microarrays* are a way to quickly and inexpensively
36 measure the expression levels of thousands of genes simultaneously. Micro-
37 arrays employ fluorescently labeled fragments of RNA that bind to known
38 locations on the microarray's surface. A scanning laser measures the intensity
39 of fluorescence at each point on that surface. The levels of expression of specific
40 RNAs can be inferred from the intensity values measured by the laser. Mi-
41 croarrays are often used to measure expression levels before and after a specific
42 physical event to see which genes changed their expression levels in response to
43 that event. Genes whose expression levels increased are said to be *up-regulated*,
44 while those whose levels decreased are said to be *down-regulated*.

45 The development of microarrays and their associated large collections of
46 experimental data have led to the need for automated methods that assist in the
47 interpretation of microarray-based biomedical experiments. We present a
48 method for creating interpretations of microarray experiments that combines
49 the expression-level data with textual information about individual genes.
50 These interpretations consist of models that characterize the genes whose ex-
51 pression levels were up- (or down-) regulated. The goal of the models is to
52 assist a human scientist in understanding the results of an experiment. Our
53 approach is to use machine learning to create models that are (a) accurate, (b)
54 comprehensible, and (c) stable to small changes in the microarray experiment.

55 In order to make them comprehensible, our models are expressed in terms of
56 English words from text descriptions of individual genes. We currently get
57 these descriptions from the curated SwissProt protein database [1]. It contains
58 annotations of proteins; we use the text associated with the protein generated
59 by a gene as the description of that gene. Our models consist of sets of words
60 from these descriptions that characterize the up-regulated or down-regulated
61 genes. Note that we can use the same text descriptions of the genes to generate
62 interpretations of many different microarray experiments—in each experiment,
63 different genes will be up-regulated or down-regulated, even though the text
64 description associated with each gene is the same across all experiments.

65 Our work is related to several recent attempts to use machine learning to
66 predict gene-regulation levels (e.g., [2,4,12]), but our focus is different in that
67 our goal is not to *predict* gene-regulation levels, but to automatically generate
68 human-readable characterizations of the up- or down-regulated genes to help a
69 human scientist generate hypotheses explaining an experiment.

70 A key aspect of our approach for interpreting microarray experiments is
71 that we evaluate the accuracy of potential models of the experimental data by
72 using a statistical technique called *cross-validation* (described later). Although
73 it is not our goal to make predictions about new genes, one of our criteria for
74 what makes a good explanatory model is that if some of the genes' expression
75 levels had not been measured in the experiment, then our model would have
76 accurately predicted these expression levels.

77 We investigate two standard and successful algorithms from the machine-
78 learning literature, evaluating how well they satisfy our three desiderata of
79 accuracy, comprehensibility, and stability. We also present and evaluate some
80 variants and combinations of these two approaches. One standard approach
81 we investigate is Naïve Bayes [9], which is based on probability; the other is
82 PFOIL (Mooney, 1995), a rule learner based on propositional logic. Naïve
83 Bayes has proven successful on a wide range of text problems [7] but produces
84 models that are not particularly human-readable, while a major strength of rule
85 learners is that they do tend to produce human-readable models.

86 Before describing the learning algorithms we evaluate, we first describe the
87 basic task that we assign to the learners. The input to each learner consists of
88 two parts: (a) the (numeric) RNA-expression levels of each gene on a gene
89 array under two conditions, before and after a particular event (e.g., antibiotic
90 treatment), and (b) the SwissProt text describing the protein produced by each
91 gene on the microarray. The output of each learner is a model that accurately
92 characterizes the genes that were up-regulated (or down-regulated) in response
93 to the event.

94 In our current set of experiments, we consider a gene up-regulated if its ratio
95 of RNA_{after} to RNA_{before} (the gene's expression ratio) is greater than 2; if this
96 ratio is less than $1/2$ we consider it down-regulated. As is commonly done, we
97 currently discard as ambiguous all genes whose expression ratio is between $1/2$
98 and 2.

99 In cross-validation, some (e.g., 80%) of the gene-regulation examples are
100 used as the training data for a learning algorithm, while the remaining ("held
101 aside") examples are used to estimate the accuracy of the learned model. In N -
102 fold cross-validation, the examples are divided into N subsets, and then each
103 subset is successively used as the held-aside test set while the other $(N - 1)$
104 subsets are pooled to create the training set.

105 Our second desired property is human *comprehensibility*. This is very diffi-
106 cult to measure and we use a crude approximation by counting the number of
107 distinct SwissProt words appearing in a given model.

108 Our third important property is *stability*. Ideally, the models learned would
109 not change much if the gene-array experiment were repeated under slightly
110 different conditions. We do not currently have actual replicates of a given
111 microarray experiment, so we instead measure stability by repeatedly creating
112 models from random samples of the genes and then comparing these models to

113 each other. For example, if a microarray contains 1000 genes, we would several
114 times randomly select 900 of these genes, and then learn a model for each
115 random sample. We then apply our stability measure (defined later) to this set
116 of learned models.

117 The next section presents the machine-learning algorithms we investigate in
118 our experiments. Section 3 further explains our experimental methodology and
119 Section 4 presents and discusses experimental results obtained using data from
120 the Blattner *E. coli* Laboratory at the University of Wisconsin. Section 6 de-
121 scribes some related work. The final section describes some of our planned
122 follow-up research and summarizes the lessons learned so far in our work to
123 create a tool that uses English-text protein annotations to assist in the inter-
124 pretation of microarray experiments.

125 2. Algorithm descriptions

126 This section describes the two algorithms—Naïve Bayes [9] and PFOIL
127 (Mooney, 1995)—and the variants that we are exploring. Both algorithms take
128 as input a collection of training instances (in our case, genes), labeled as be-
129 longing to one of two classes (which we will call up and down), and described
130 by a vector of Boolean-valued features, W . Each feature corresponds to a word
131 being present or absent from the text description of the gene. Both algorithms
132 produce a model that can be used to categorize an unlabeled gene on the basis
133 of its feature values (i.e., the words describing it).

134 2.1. Naïve Bayes

135 The Naïve Bayes algorithm works by computing estimates of conditional
136 probabilities from the input data, and using those probabilities in Bayes' rule to
137 determine the most probable class of an unlabeled instance. The algorithm
138 counts the occurrences of words in all the instances of the positive and negative
139 classes to compute four probabilities for each word. These probabilities are
140 combined into two likelihood ratios for each word that represent the signifi-
141 cance of the presence or the absence of that word:

$$\text{lr}_i(w_i \text{ present}) = p(w_i \text{ present}|\text{up})/p(w_i \text{ present}|\text{down})$$

$$\text{lr}_i(w_i \text{ absent}) = p(w_i \text{ present}|\text{up})/p(w_i \text{ absent}|\text{down})$$

It uses a standard Laplace estimator to deal with infrequent words; that is, 1 is added to each numerator and denominator used to estimate probabilities, thereby preventing estimating probabilities of zero.

146 The algorithm will classify an unlabeled instance (gene) as an up instance if

$$\prod_i \text{lr}_i(w_i \text{ is } X) \frac{p(\text{up})}{p(\text{down})} > 1$$

where w_i is the i th word in the vocabulary and X is either *present* or *absent* depending on whether the word is present or absent in the instance. The algorithm is called “naïve” because we ignore any dependencies among the words when calculating these probabilities. This assumption of independence (of the features given the class) has proven effective in many practical problems, especially those involving English words as features [9].

154 A major problem with the Naïve Bayes classifier is the lack of comprehen-
155 sibility of its characterization of the positive class: for a vocabulary of 5000
156 words, the characterization involves 10,000 likelihood ratios. The variant of
157 Naïve Bayes that we explore in our experiments improves the comprehensibility
158 by pruning the characterization to just the most significant words and their
159 likelihood ratios. Note that this pruning could reduce the accuracy; the ex-
160 periments described in Section 4 explore the trade-off between accuracy and
161 comprehensibility.

162 Our Pruned Naïve Bayes algorithm has two phases:

- 163 1. Construct a subset of the vocabulary consisting of the n words with the
164 highest score according to an *information-gain* [9] measure.
- 165 2. Prune this subset further by incrementally removing words one at a time.

The result is a characterization of the class in terms of a much smaller set of words. A pseudocode version of the algorithm is given in Table 1.

168 The information-gain measure used in the first phase is the standard mea-
169 sure of the effectiveness of a word at separating the up and down classes, as
170 used in many decision-tree algorithms [9]. All the words are scored according
171 to their information-gain, and the n words with the highest scores are selected.

Table 1
Pruned Naïve Bayes algorithm

Disjointly split data set into *training* and *testing* sets.
Disjointly subdivide *training set* into *train' set* and *tuning set*.
For each word w_i in the *train' set*: Compute InfoGain_i
Set W to the set of n words with the largest InfoGain_i
While W is not empty
 For each $w_i \in W$
 Remove w_i from W
 Construct Naïve Bayes classifier on *train'*, using only words in W
 Evaluate classifier accuracy on *tuning set*
 Return w_i to W
 Remove from W the w_i that resulted in the best tuning-set accuracy
 Report current W and tuning and testing set accuracies of classifier

172 For the second phase of the Pruned Naïve Bayes algorithm, the training set is
173 separated into a train' set and a tuning set (we use 25% of the data for the
174 tuning set). To choose a word to prune, the algorithm constructs each subset
175 that can be obtained by removing just one word from the current subset of
176 words and generates a Naïve Bayes classifier on the train' set with the instance
177 descriptions restricted to remaining words, and determines the average accu-
178 racy of the resulting classifier on the tuning set. On each iteration, it chooses
179 the word that results in the most accurate classifier and prunes it from the
180 current subset. Since we are interested in the trade-off between accuracy and
181 comprehensibility, the algorithm reports the current subset and its accuracy on
182 the test set at each iteration, until the subset is empty.

183 2.2. PFOIL

184 PFOIL (Mooney, 1995) is a propositional version of FOIL [11], a rule-
185 building algorithm that incrementally builds rules that characterize the in-
186 stances of a class in a data set. FOIL builds rules for a first-order logic lan-
187 guage, so that the rules are conjunctions of literals that may contain logical
188 variables (and may even be recursive) and must be interpreted by a first-order
189 reasoning engine such as Prolog. PFOIL uses a simpler propositional language,
190 and builds rules that are conjunctions of features. PFOIL rules can be inter-
191 preted straightforwardly—a rule covers an instance if each feature in the rule is
192 true of the instance. In our domain, a rule specifies words that must or must
193 not be present in a gene's annotation.

194 PFOIL builds a rule set by constructing one rule at a time. It constructs each
195 rule by adding one feature at a time to the current rule. At each step, it chooses
196 the feature that maximizes the performance of the rule according to the *Foil-*
197 *Gain* measure. It stops adding to a rule when either the rule covers only positive
198 instances, or none of the remaining features have a positive *FoilGain*. When a
199 rule is complete, the algorithm removes all the positive instances covered by the
200 rule from the data set, and starts to build a new rule.

201 *FoilGain* is a measure of the improvement that would be obtained by adding
202 a new feature to a rule. It is a trade-off between the coverage of the new ru-
203 le—the number of positive instances of the class that are covered by the ru-
204 le—and the increase in precision of the rule—the fraction of the instances
205 covered by the rule that are positive:

206 $FoilGain(rule, f) = p[\log(p/(p+n)) - \log(P/(P+N))]$ where P and N are
207 the number of positive and negative instances covered by rule, and p and n are
208 the number of positive and negative instances that are covered when feature f is
209 added to rule.

210 As described by Mooney (1995), PFOIL does not prune its rule set. Because
211 PFOIL keeps constructing rules until it has covered all the positive instances, a
212 data set with noise is likely to result in a large set of rules, many of which may

Table 2
PFOIL with pruning

Disjointly split data set into *training* and *test* sets.
Disjointly subdivide *training set* into *train'* and *tuning sets*.
Construct PFOIL rule set that completely fits *train'*.
While the rule set is not empty
 For each feature in each rule in current rule set
 Temporarily remove feature from rule.
 Evaluate rule set accuracy on *tuning set*.
 Return the feature to the current rule set.
 Remove from rule set the feature that resulted in the best accuracy.
 Remove any empty rule from the rule set.
 Report current rule set and its accuracy on *train'*, *tuning*, and *test sets*.

213 be very specific to particular instances. Even though individual PFOIL rules are
214 more comprehensible than the vectors of likelihood ratios produced by Naïve
215 Bayes, a large rule set is not particularly comprehensible.

216 To address this problem, we have extended PFOIL to include a rule pruning
217 stage, along the lines of the pruning in FOIL (Table 2).

218 In the pruning stage, the algorithm repeatedly removes a single feature from
219 one of the rules, choosing the feature whose removal results in the highest
220 accuracy of the remaining rule set. When all the features are removed from a
221 rule, the rule is removed from the rule set. As in the pruning version of Naïve
222 Bayes, we separate the training data set into a *train'* set and a tuning set, using
223 the tuning set to evaluate the predictive accuracy of the rules learned on the
224 *train'* set. Rather than halting the pruning when the tuning-set accuracy peaks,
225 in our experiments we continue the pruning until the rule set is empty in order
226 to explore the trade-off between comprehensibility and accuracy.

227 2.3. Improving stability of PFOIL

228 The Naïve Bayes algorithm is fairly stable to small variations in the data
229 since it is based on estimated probabilities, but PFOIL—being a greedy, rule-
230 based algorithm—is not so stable. The set of rules it produces can differ con-
231 siderably with just a small change in the data. We have constructed a variant of
232 PFOIL that restricts the set of features from which it will build its rules in order
233 to encourage greater stability. The algorithm first runs PFOIL on a number of
234 random subsets of the training set and produces a set of rules for each subset. It
235 collects those words that are used in at least m of these rule sets. It then reruns
236 PFOIL (with rule-set pruning) on the whole training set, but restricted to
237 building rules containing only words found in the first stage. The pseudocode is
238 given in Table 3.

Table 3
Pseudocode of stabilized PFOIL

Five times disjointly split data set into *training* and *test* sets.
 Generate Set of Words for Training Set *i*:
 Initialize counts on each word to zero.
 Repeat *n* times
 Choose a random 80% of training set for the *train'* set.
 Run PFOIL on *train'* set (with no pruning).
 Increment *count* on each word used in any rule in rule set.
 Set *W* to the words with *count* > *m*
 Generate Rule sets Using Only the Words in *W*:
 Disjointly split *training* set into *train'* and *tuning* sets.
 Run PFOIL (with pruning) on *train'* set, restricted to words in *W*.
 Record rule set and accuracy on *test* set.
 Compute stability and average accuracy of rule sets across all five folds.

239 To evaluate this algorithm, we do a fivefold cross-validation and compute a
 240 measure of the similarity of the rule sets from each of the five folds. We use a
 241 coarse measure that only considers the words used in the rule sets and ignores
 242 the structure of the rules. The measure is a root-mean-square (RMS) average of
 243 the number of times a word is duplicated across the five rule sets:

$$\text{stability} = \sqrt{\frac{\sum_{w_i \in U} (\text{count}(w_i) - 1)^2}{|U|} \frac{1}{N - 1}}$$

where *U* is the set containing those words that appear in any of the rule sets, *N* is the number of rule sets, and $\text{count}(w_i)$ is the number of rule sets in which the word w_i appears. Using an RMS average weights words that occur in more data sets more heavily.

249 For example, if all *N* sets of words were identical, $\text{count}(w_i)$ would equal *N*
 250 for each $w_i \in U$. So, stability would be 1. If the *N* sets were completely disjoint,
 251 $\text{count}(w_i)$ would equal 1 for each $w_i \in U$. In this case, stability would be 0.

252 3. Experimental methodology

253 The data we are using are from microarray experiments performed by the
 254 Blattner *E. coli* Sequencing Laboratory at the University of Wisconsin. In our
 255 current computational experiments, we use the biological event of antibiotic
 256 treatment; the “after” gene expressions are measured 20 min after treatment.
 257 We use fivefold cross-validation, five times training on 80% of the data and
 258 then evaluating our models on the “held aside” 20%. We use all of the text
 259 fields in the SwissProt database. These include the comment (CC) fields (with
 260 the exception of the database (CDB) and mass spectrometry (CMS) topics), the

261 description (DE) field, the Organism Classification (OC) field, the keyword (KW)
262 field, and the reference title (RT) fields (with the exception of titles containing:
263 The Complete *Sequence of the E. coli K12 Genome*).

264 As is common when using text, we convert all words in the annotations to
265 their root form by using the Porter [10] stemming algorithm. Brackets (“[]”)
266 are used in this paper to signify that a word has been stemmed. For example:
267 residu[al|e] refers to both residual and residue. Other than the Porter stemmer,
268 we implemented the algorithms ourselves in C.

269 4. Experimental results and discussion

270 The first set of experiments explores the trade-off between accuracy and
271 comprehensibility of the two algorithms. Fig. 1 shows the result of a fivefold
272 cross-validation experiment running PFOIL and Naïve Bayes, both with
273 pruning. The average error rate is plotted against the number of words left in
274 the model. The baseline error rate for this data set (predicting the most fre-
275 quent class) is 38%. Both algorithms perform reasonably well. Naïve Bayes is
276 consistently better than PFOIL and both are consistently better than the
277 baseline.

278 After a brief initial decline the PFOIL error rate increases as words are
279 pruned from the rule set, but the increase is surprisingly slight until only four
280 words remain. Unexpectedly, over most of the experiment, pruning words from
281 the Naïve Bayes model decreases the error rate to a minimum at 23 words. The
282 error rate then increases again. The error rate of unpruned Naïve Bayes, i.e.,
283 using all of the SwissProt words, is only 12%—the higher error rate (26%) at
284 $x = 100$ in the graph is the result of the initial pruning phase, where only the
285 100 most informative features are used. For both PFOIL and Naïve Bayes, the
286 accuracy is still high for models that have been pruned to a very comprehen-
287 sible size.

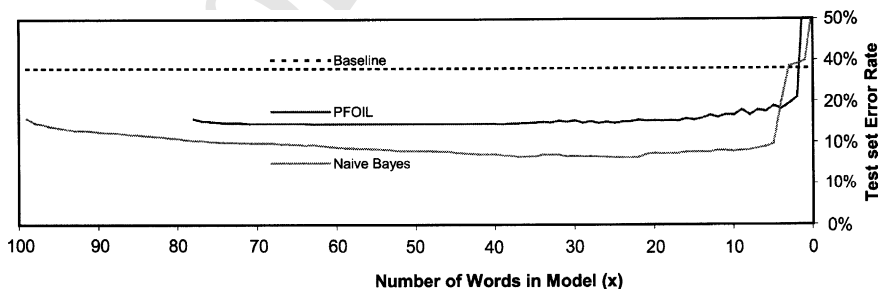


Fig. 1. Test set error as a function of model size.

Table 4
Sample small models of the experimental data

<i>PFOIL (disjunctive rules for up)</i>		
Rule 1	NOT map	AND
	NOT biosynthesis	AND
	NOT proteins	AND
	NOT gene	
Rule 2	NOT biosynthesis	AND
	NOT map	AND
	NOT with	AND
	NOT encoded	
Rule 3	Hypothetical	
Rule 4	570-kb	
<i>Naïve Bayes</i>		
Word (<i>w</i>)	lr (<i>w</i> present)	lr (<i>w</i> absent)
Flagellar	0.07	1.09
Deriv[ative ation]	0.09	1.03
Ubiquinone	0.09	1.03
Nuo	0.09	1.03
Mitochondri[a al]	0.12	1.04
Aspart[ate yl]	0.14	1.04
Nadh	0.15	1.03
Complex[[es ed]	0.19	1.15
Biosynthesis	0.29	1.05
Permeas[e]	0.31	1.05

288 Table 4 shows sample models from both PFOIL and Naïve Bayes. The PFOIL
 289 section contains the disjunctive rules. The Naïve Bayes section shows the
 290 likelihood ratios. A high likelihood ratio for present or absent suggests that the
 291 word's presence or absence, respectively, is associated with a gene being up-
 292 regulated. Low likelihood ratios are associated with a gene being down-regu-
 293 lated. The models shown were chosen when the pruning had reduced the
 294 models to 10 words. They share only one word, which might seem surprising
 295 given that they are characterizing the same experiment. That word, "biosyn-
 296 thesis", likely reflects an underrepresentation of basic biosynthetic genes
 297 among those up-regulated. Note also that the PFOIL rules are dominated by
 298 words being absent, whereas the Naïve Bayes model contains no words with a
 299 high likelihood ratio for the word being absent. It appears that the two algo-
 300 rithms find quite different kinds of characterizations of the data.

301 The second set of experiments explored the trade-off between accuracy and
 302 stability for the PFOIL algorithm. Fig. 2 shows the averaged fivefold result of
 303 running Stabilized PFOIL on various values of *m* between 1 and 50 on the same
 304 data set as Fig. 1; *m* = 1 represents the least constrained—all words that oc-

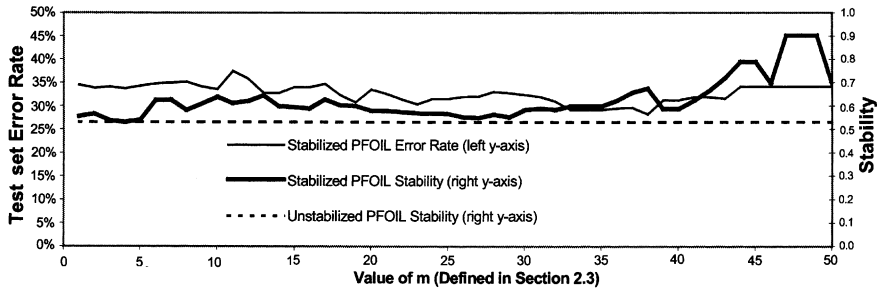


Fig. 2. Trade-off between rule set stability and test set error rate for PFOIL.

305 curred in a rule set from any of the 50 runs (an average of 396 words) were
306 available for the second phase; $m = 50$ represents the most constrained—only
307 words that occurred in all 50 rule sets were available for the second phase. The
308 stability measure represents the most constraint—only words that occurred in
309 all 50 rule sets were available for the second phase. The stability measure
310 represents the similarity of the five rule sets. The maximum observed stability
311 (0.9) could result from 55% of the words having appeared in all five rule sets
312 and the rest appearing in four of the five rule sets. The minimum stability (0.4)
313 could result from half of the words appearing in three of the five rule sets and
314 the other half appearing in only two.

315 The stability of the rule sets increased significantly as the number of words
316 available was reduced. Stabilized PFOIL is also clearly more stable than the
317 original version. The error rate does not change very much as a result of the
318 stabilizing algorithm. However, the error rate is considerably worse than for
319 the unstabilized version, which suggests that forcing stability has a significant
320 accuracy cost.

321 5. Discussion

322 The pruning experiments suggest that, using our greedy algorithm, the
323 trade-off between accuracy and comprehensibility is not very severe. In fact
324 decreasing the rule set size can be beneficial and does not significantly hamper
325 accuracy until well into the range of comprehensible rules (<10 words). The
326 information-gain-based pruning, however, that we use to prune the Naïve
327 Bayes model from thousands of words to one hundred words does seem to have
328 an adverse affect: The error rate increases from 12% to nearly 26% over the
329 course of that pruning but drops back down to 16% with greedy pruning.

330 The fact that PFOIL's error rate is higher than that of Naïve Bayes suggests
331 that PFOIL is not effective at discovering the real regularities in the gene data.
332 We speculate that this is because PFOIL's hill climbing search strategy depends

333 on evaluating the benefit of adding features one at a time. Dependencies be-
334 tween features cause valleys in the search space that mislead PFOIL. A direc-
335 tion for future work is to explore ways of smoothing the search space.

336 Naïve Bayes does not even attempt to identify dependencies between the
337 features, treating each feature as completely independent. Its model, even when
338 pruned to a manageable number of words, is not helpful in identifying de-
339 pendencies. In the model in Table 4, both Permeas[e] and Flagellar are strongly
340 correlated with up-regulated genes, but the model does not say, for example,
341 whether the genes characterized by Permeas[e] are the same or different from
342 the genes characterized by Flagellar. In spite of this, the Naïve Bayes model
343 consistently gets a higher accuracy than the PFOIL models. We speculate that
344 one reason is that the PFOIL rules are “crisp” in the sense that an instance
345 must have every feature specified in the rule to satisfy the rule, whereas the
346 Naïve Bayes model is “softer” and allows a range of alternative features to
347 contribute to the probability of being up-regulated. A direction for future work
348 is to explore the use of rule-building algorithms that allow softer rules, for
349 example, allowing M of N rules [3].

350 As noted above, the two algorithms produce very different characterizations
351 of the data. This is partly due to the different “default rules” in the algorithms.
352 When attempting to characterize the *up* genes, PFOIL’s default is to classify any
353 gene not covered by rule as a *down*. One possible explanation for the abun-
354 dance of NOT’s in the rules is that PFOIL attempts to find rules that exclude
355 down genes, which is most easily achieved by excluding words that are com-
356 mon in down genes. Pruned Naïve Bayes, on the other hand, has a bias towards
357 the larger class so its default in this data set is the *up* genes. When pruning,
358 there is little advantage in keeping words that pick out *up* genes, since it tends
359 to classify them correctly. Like PFOIL, it retains words that will pick out the
360 down genes, but only when they are very infrequent in the *up* genes. That
361 means that $lr_i(\text{present})$ will be much smaller than 1.

362 We note also that although Naïve Bayes does not take dependencies between
363 words into account, the pruning will tend to retain only one of a group of
364 dependent words, since there will generally be little accuracy advantage in
365 keeping the other words.

366 Both of the algorithms initially produce incomprehensible models contain-
367 ing many more features than a human can make sense of. However, our ex-
368 periments show that heavy pruning can result in much more comprehensible
369 models, with only moderate loss of accuracy. The results for Naïve Bayes are
370 particularly striking, where a model containing only 10 words has only a
371 moderate loss of accuracy, but a 500-fold increase in comprehensibility. Since
372 comprehensibility is essential for this task, these results are encouraging.

373 PFOIL produces unstable models that can vary significantly with particular
374 genes in the training data. The experiments have shown that our algorithm for
375 improving the stability of PFOIL is effective at producing much more stable

376 rule sets, though at a cost of reduced accuracy. This is a promising result, and
377 we intend to use this technique with more sophisticated rule-building algo-
378 rithms.

379 6. Related work

380 A great deal of research has been done in text mining, much of which in-
381 volves biomedical tasks. Hearst's LINDI system [5] searches medical literature
382 for text relating to a particular subject or problem and tries to make logical
383 connections to form a hypothesis. One of the benefits of our approach is that
384 researchers do not need to know what they are looking for in advance. Given
385 expression data and textual descriptions of the genes represented in the ex-
386 pression data, this system makes an automated "first pass" at discovering what
387 is interesting. The PubGene tool [6] also interprets gene-expression data based
388 on textual data. One big difference is that PubGene compiles clusters of text
389 ahead of time and tries to match the expression data to an already-made
390 cluster. Our tool is designed to allow the expression data itself to define its
391 models. Masys et al. [8] also use text associated with genes to explain experi-
392 ments. However, they cluster expression values across experiments and then
393 use the text to explain the clusters, whereas we use the text directly during the
394 learning process and can explain single experiments.

395 7. Future directions and conclusions

396 We have presented an approach for aiding in the interpretation of micro-
397 array experiments that is based on machine learning and uses SwissProt text as
398 its representation of the microarray's genes. We argue that there are at least
399 three important properties for such a computational tool: it should produce
400 models that are (1) accurate, (2) readable, and (3) stable to small changes in the
401 microarray data. We empirically studied two widely successful algo-
402 rithms—Naïve Bayes and PFOIL—on an *E. coli* microarray experiment, eval-
403 uating these two approaches and some variants with respect to our desiderata.
404 We have shown that both algorithms are able to find characterizations of the
405 data with reasonable accuracy and that by adding pruning to the algorithms,
406 comprehensibility can be achieved with only a minor reduction in accuracy. We
407 also presented a modification to PFOIL that has increased stability, but at a
408 cost of decreased accuracy. We noted that the algorithms construct different
409 kinds of characterizations. We intend to explore ways of combining the de-
410 sirable properties of Naïve Bayes and PFOIL, and to develop new algorithms
411 for the task.

412 Another current limitation to this approach is that it relies only on text,
413 though the sequence data are almost always also available. We plan to explore
414 methods that make use of both the sequence data and the text annotating the
415 genes. Another enhancement would be to increase the textual data we use.
416 Abstracts from appropriate articles would be a logical next step. Finally, we
417 plan to empirically evaluate our approach on additional microarray experi-
418 ments.

419 Acknowledgements

420 This research was funded by grants NLM 1 R01 LM07050-01, NSF IRI-
421 9502990, NIH 2 P30 CA14520-29, and NIH 5 T32 GM08349.

422 References

- 423 [1] A. Bairoch, R. Apweiler, The SWISS-PROT protein sequence database and its supplement
424 TrEMBL in 2000, *Nucl. Acids Res.* 28 (2000) 45–48.
- 425 [2] W. Brown, D. Lin, D. Cristianini, C. Sugnet, T. Furey, M. Ares, D. Haussler, Knowledge-
426 based analysis of microarray gene expression data using support vector machines, *Proc. Natl.*
427 *Acad. Sci.* 97 (1) (2000) 262–267.
- 428 [3] M. Craven, J. Shavlik, Learning symbolic rules using artificial neural networks, in: *Proc. Intl.*
429 *Conf. on Machine Learning*, Morgan Kaufmann, 1993, pp. 73–80.
- 430 [4] S. Dudoit, Y. Yang, M. Callow, T. Speed, *Statistical Methods for Identifying Differentially*
431 *Expressed Genes in Replicated cDNA Microarray Experiments*, UC-Berkeley Statistics Dept.
432 TR #578, 2000.
- 433 [5] M. Hearst, Untangling text data mining, in: *Proc. 37th Annual Meeting of Assoc. for*
434 *Computational Linguistics*, 1999.
- 435 [6] T.-K. Jenssen, A. Lægreid, J. Komorowski, E. Hovig, A literature network of human genes for
436 high-throughput gene-expression analysis, *Nat. Genet.* 28 (2001) 21–28.
- 437 [7] C.D. Manning, H. Schütze, *Foundations of Statistical Natural Language Processing*, The MIT
438 Press, 1999.
- 439 [8] D. Masys et al., Use of keyword hierarchies to interpret gene expression patterns,
440 *Bioinformatics* 17 (2001) 319–326.
- 441 [9] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- 442 [10] M. Porter, An algorithm for suffix stripping, *Program* 14 (1980) 130–137.
- 443 [11] J.R. Quinlan, Learning logical descriptions from relations, *Mach. Learn.* 5 (1990) 239–266.
- 444 [12] E. Xing, M. Jordan, R. Karp, Feature selection for high-dimensional genomic microarray
445 data, in: *Proc. Intl. Conf. on Machine Learning*, Morgan Kaufmann, 2001, pp. 601–608.