BIOINFORMATICS

Vol. 18 Suppl. 1 2002
Pages S164–S171

Evaluating machine learning approaches for aiding probe selection for gene-expression arrays

J. B. Tobler[1], M. N. Molla[1, 3,*], E. F. Nuwaysir[3], R. D. Green[3] and J. W. Shavlik[2]

[1]Department of Computer Science, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706,, [2]Departments of Computer Science, and Biostatistics and Medical Informatics, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, and [3]NimbleGen Systems, Inc., One Science Ct., Madison, WI 53711,

Received on January 24, 2002; revised and accepted on March 27, 2002

ABSTRACT

Motivation: Microarrays are a fast and cost-effective method of performing thousands of DNA hybridization experiments simultaneously. DNA probes are typically used to measure the expression level of specific genes. Because probes greatly vary in the quality of their hybridizations, choosing good probes is a difficult task. If one could accurately choose probes that are likely to hybridize well, then fewer probes would be needed to represent each gene in a gene-expression microarray, and, hence, more genes could be placed on an array of a given physical size. Our goal is to empirically evaluate how successfully three standard machine-learning algorithms—naïve Bayes, decision trees, and artificial neural networks—can be applied to the task of predicting good probes. Fortunately it is relatively easy to get training examples for such a learning task: place various probes on a gene chip, add a sample where the corresponding genes are highly expressed, and then record how well each probe measures the presence of its corresponding gene. With such training examples, it is possible that an accurate predictor of probe quality can be learned.

Results: Two of the learning algorithms we investigate—naïve Bayes and neural networks—learn to predict probe quality surprisingly well. For example, in the top ten predicted probes for a given gene not used for training, on average about five rank in the top 2.5% of that gene's hundreds of possible probes. Decision-tree induction and the simple approach of using predicted melting temperature to rank probes perform significantly worse than these two algorithms. The features we use to represent probes are very easily computed and the time taken to score each candidate probe after training is minor. Training the naïve Bayes algorithm takes very little time, and while it takes over 10 times as long to train a neural network, that time is still not very substantial (on the order of a few hours on a desktop workstation). We also report the information contained in the features we use to describe the probes. We find the fraction of cytosine in the probe to be the most informative feature. We also find, not surprisingly, that the nucleotides in the middle of the probes sequence are more informative than those at the ends of the sequence.

Contact: molla@cs.wisc.edu

Keywords: microarrays; probe selection; artificial neural networks; decision trees; naïve Bayes.

INTRODUCTION

Overview

Oligonucleotide microarrays, commonly known as *gene chips*, are a fast and cost-effective method of performing thousands of DNA hybridization experiments simultaneously. The general procedure is quite simple. Short (typically around 24 base pairs, or *24-bp* for short) strands of known DNA sequence, called *probes*, are affixed to specific positions on a chip's surface. A fluorescently labeled RNA sample is then washed over this surface. Some of this RNA will hybridize to complementary strands of DNA. The amount of RNA hybridized to each position on the microarray, and therefore in the sample, can be inferred from fluorescence measurements measured by a laser that scans the surface of the chip returning a *fluorescence intensity value* (Snustad and Simmons, 1999).

In order to measure the expression level of a specific gene in a sample, one must design a microarray containing DNA strands complimentary to the gene of interest. Because a typical probe is much shorter than a typical gene, a probe cannot contain the entire complement of a gene. In fact, due to issues of dilution and economy, the total length of probes used to hybridize a gene is typically only a small fraction of the length of the gene.

Choosing good probes is a difficult task. Different sequences have different hybridization characteristics

*To whom correspondence should be addressed.

(Breslauer *et al.*, 1986). Expression levels can only be accurately measured if a good set of probes is chosen. Otherwise, inadequate or nonspecific hybridization can occur, confounding the results of the experiment. Effective computational aids for probe selection can substantially improve the design of gene chips, and computationally aiding the probe-selection task is the focus of this article.

Task definition

We apply three well-established, machine-learning techniques to the problem of probe selection, in order to judge how well probe quality can be predicted and which learning algorithm performs best on this important task. Specifically, we compare the performance of artificial neural network (ANN) training, the naïve Bayes algorithm, and a decision-tree learner (Mitchell, 1997). In order to do this, we frame the problem of probe selection as a *category-prediction* task. Each 24-bp probe is called an *example*. An example's *features* are derived (as described below) from the sequence of 24 bases that make up the probe. The goal is to predict the quality of the probe strictly from these features. Our set of probes is the set of all possible 24-bp probes from each of the eight genes in the *E. coli* and *B. subtilis* genomes (four genes from each genome), also known as a *tiling* of the genes. As a measure of probe quality, we use the measured fluorescence levels of our probes after they have been exposed to a sample in which all eight of these genes are highly expressed. To summarize, our task is to algorithmically learn how to perform the following:

Given: a 24-bp probe from a gene *not used during training*

Do: predict if this probe's fluorescence level will rank in the top third of all the possible probes for this gene when exposed to a sample where this gene is highly expressed

If accurately predicted, this information (when combined with other constraints) can prove highly beneficial when deciding which probes to use to represent a given gene when designing a microarray. If we can increase the probability that a chosen probe will be bound strongly when the probe's gene is expressed in the current sample, then fewer probes will be needed per gene that we wish to detect by the microarray and, hence, more genes can be measured by a gene chip of a given physical size.

Related work

Heuristics have been developed to discard probes based on based on knowledge about hybridization characteristics (Lockhart *et al.*, 1996) such as self-hybridization and degenerate repeats. Others have attempted to use melting point (T_m) equations derived from genetic material in solution (Kurata and Suyama, 1999; Li and Stormo, 2001). Kurata and Suyama (1999) also investigate the use of predictions of stable secondary structures and probe uniqueness to create criteria for selecting good probes. Our contribution is a successful empirical evaluation of three standard machine-learning methods applied to the task of learning to predict good probes.

SYSTEMS AND METHODS

Data

Our experimental data consists of tilings of four genes from each of *E. coli* and *B. subtilis,* measured on 'maskless' microarrays (Singh-Gasson *et al.*, 1999) produced by NimbleGen, Inc. In order to standardize the data across the eight genes, the measured fluorescence intensity for each probe is normalized such that the bottom and top 2.5% of the measured intensities are labeled 0 and 1 respectively, and all other values are linearly interpolated between 0 and 1. Mapping the bottom and top 2.5% to 0 and 1 reduces the impact of outliers.

Datasets

Supervised machine-learning algorithms like naïve Bayes, decision trees, and ANNs typically require training on a set of data with labeled (i.e., categorized) examples; this set of data is called the *training set*. The classifier produced by the learning algorithm after training is then tested using another dataset, the *test set*. The predicted classifications of the trained machine-learning classifier are compared to the correct outputs in the test set in order to estimate the accuracy of the classifier. We used the microarray data for our eight genes to generate eight training-set and test-set pairs using the commonly applied *leave-one-out* method. In this method, each training set consists of the probes from the tilings of seven genes and the corresponding test set contains the remaining gene's probes.

We use only simple features to describe the 24-bp probes (also called *24mers*) in each data set. The features we chose are described in Table 1. Certainly much more expressive data descriptions are possible, but using these features allow us to determine how much useful information is contained in the basic 24mer structure.

Using the normalized intensity, we chose to label each probe example with a discrete output of *low*, *medium*, or *high*. We sought to roughly evenly distribute the *low*'s, *medium*'s, and *high*'s. The normalized intensity to discrete-output mappings we chose are: the interval [0.00–0.05] maps to *low* (45% of the examples), (0.05–0.15] (23% of the data) to *medium*, and (0.15–1.00] (32%) to *high*.

In all of our experiments, during *training* we discard all of the probes labeled *medium*, since those are arguably ambiguous. Of course it would not be fair to ignore the *medium* probes during the *testing* phase (and we do not do so), since the objective during testing is to *predict* the output value for probes not seen during training.

Table 1. Features used to describe probes

Feature Name	Description
$fracA$, $fracC$, $fracG$, $fracT$	The fraction of A, C, G, or T in the 24mer
fracAA, fracAC, fracAG, fracAT, fracCA, fracCC, fracCG, fracCT, fracGA, fracGC, fracGG, fracGT, fracTA, fracTC, fracTG, fracTT	The fraction of each of these dimers in the 24mer
$n1, n2, \ldots, n24$	The particular nucleotide (A, C, G, or T) at the specified position in the 24mer (n1 is the $5'$ end)
$d1, d2, \ldots, d23$	The particular dimer (AA, AC, … TT) at the specified position in the 24mer (d1 is the $5'$ end)

However, we did remove some probes from the test sets, namely those that had less than three mismatches with any other 24-bp region in the full genome containing the probe's gene. Such probes are not useful, since that are insufficiently indicative of a single gene. We did leave such 'insufficiently unique' probes in the *training* sets since that may still provide useful information to the learning algorithms. In any case, the number of discarded (from the test sets) genes is very small, only a very small fraction of the probes had at least one close match in their genome.

Methods

As mentioned above, we evaluate the well-established machine learning algorithms of naïve Bayes, decision trees, and artificial neural networks (ANNs) on our task.

Naïve Bayes is a practical, successful machine-learning algorithm that assumes independence among the features for a given example. Using this independence assumption, a simple ratio can be used to compute the relative likelihood that a test example with feature values $v_1, v_2, \ldots, v_N$ should be labeled *high* or *low*:

$$NBratio = \frac{P(high)\prod_i P(f_i = v_i|high)}{P(low)\prod_i P(f_i = v_i|low)} \tag{1}$$

where *P(high)* and *P(low)* are the number of training examples labeled high and low, respectively. $P(f_i = v_i|$ *high)* is estimated by simply counting the number of examples in the training dataset with output labeled *high* (or *low* for the terms in the denominator) and feature f_i equal to value v_i.

To avoid bias toward underestimating the probability of a given output when an occurrence of the feature value is not seen in the training data, we used the m-estimate of probability. We use the following to actually estimate probabilities:

$$P(f_i = v_i|c = \{high, low\}) = \frac{n_c + mp}{n + m} \tag{2}$$

where c is either *low* or *high*, n_c is the number of occurrence that have feature value equal to v_i and have output c, n is the number of examples with output c, m is chosen by us to equal n (we did not experiment with other settings), and $p = 1/k$ for features with k discrete feature values (Mitchell, 1997).

We discretized the non-discrete features (*fracA, fracAT*, etc.) by binning them into five equally distributed bins. The naïve Bayes algorithm is a fast algorithm for training and classification. Training and classification of a given train/test fold take on average less than 10 minutes on a standard desktop PC.

The second classifier we evaluate for use in probe selection is a decision tree. The algorithm most often used to generate decision trees is ID3 (Quinlan, 1986) or it successor C4.5 (Quinlan, 1996a). This algorithm selects the next node to place in the tree by computing the *information gain* for all candidate features and then choosing that feature that gains the most information about the output category of the current example. Information gain is a measure of how well the given feature separates the remaining training examples, and is based on Shannon (1948) information theory. Information gain is calculated as described in Equations 3 and 4.

$$\begin{aligned} Entropy(S) \equiv & -P(low)\log_2 P(low) \\ & -P(high)\log_2 P(high) \end{aligned} \tag{3}$$

where S is a set of examples, and *P(low)* and *P(high)* are estimated by computing the fractions of *low* and *high* labeled examples in S.

$$InfoGain(S, F) \equiv$$
$$Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{4}$$

where *Values(F)* is the set of all possible values for feature F and S_v is the subset of S for which feature F has value v (Mitchell, 1997).

We use the University of Waikato's *Weka 3 Machine Learning Algorithms in Java* package (http://www.cs.waikato.ac.nz/ml/weka/index.html) to run the decision-tree experiments. The Weka algorithm for decision-tree learning is named J48, but it uses the same algorithm as Quinlan's C4.5. We used Quinlan's *reduced-error-pruning* algorithm to avoid overfitting of the training data (Quinlan, 1996a); this procedure removes those portions of the initially induced decision tree that seem to be overfitting the data. At each leaf in the pruned tree, the fraction of the training set reaching that node that was *high* and *low* is recorded. We slightly modified the Weka code to report these fractions when classifying each test example; this provides a crude estimate of the probability that the current test-set example should be called *high*. Using the WEKA software package, training and classification of a decision tree for a given train-test pair takes under 5 minutes on the standard PC used in all our experiments.

The third approach we evaluate is to use a multi-layered ANN trained using backpropagation, which is the standard algorithm used for training neural networks. This algorithm attempts to minimize the squared-error[†] between the network output values and the target value for these outputs. The algorithm searches a *weight space* (defined by all possible weight values for each arc in the network) for an error minimum. Because a non-linear, multi-layered network is used, the algorithm is not guaranteed to find the *global* minimum error, but rather it may find a local minimum (Mitchell, 1997).

The networks we train consist of 485 input units, produced by using one input unit for each real-valued feature (e.g., *fractA)* and a *1-of-N* encoding for each discrete-valued feature (e.g., $n1$). A *1-of-N* encoding requires one Boolean-valued input unit for each possible feature value. When an example is input into the network, only the input unit representing that value for a given discrete-valued feature is set to 1 with the other inputs set to 0 (e.g., $n1 = A$ would be represented by 1, 0, 0, and 0 as the values for the four input units associated with $n1$). We also use a single layer of *hidden units*. In general, too many hidden units leads to overfitting the training data, while using too few hidden units can lead to underfitting the data. Hidden units free an ANN from the constraints of the feature set, and they allow for the network to discover intermediate, non-linear representations of the data (Mitchell, 1997). Sarle (1995) and others assert that using large numbers of hidden units is necessary to avoid finding bad local minima. Thus we decided to use 161 hidden units (1/3rd the number of input units), each employing the standard sigmoidal activation function. Finally, we use two *output units* with sigmoid activation functions. The two output units represent the estimated probability that the output should be *high* and *low* respectively. The network is fully connected with each input unit connected to each of the hidden units, and each hidden unit connected to each of the output units. Each arc in the network is initialized with a random weight between −0.3 and 0.3, following standard practice.

Training consists of 100 cycles through the training set. We use *early stopping* to avoid overfitting the training data by training too long. To decide when to 'stop' training, we first create a *tuning* set of data by randomly removing 10% of the training data, and after each cycle we measure the current accuracy of the ANN's predictions on the tuning set. The ANN's weight settings for the cycle that performs the best on the tuning set are the weight settings that we use to classify the test set. Each training and classification run takes over an order of magnitude longer than those for naïve Bayes and decision trees.

RESULTS

After each algorithm is run on training set j, we sort the predicted scores for each probe in test set j. For the naïve Bayes and ANN classifiers, the sorting is done from the highest to lowest according to the ratio:

$$\frac{\textit{prob(label} = \textit{high for testset example x)}}{\textit{prob(label} = \textit{low for testset example x)}}$$

The decision-tree sorting is solely based on *prob(label = high for test-set example x)*, which, as mentioned above, is estimated from the distribution of *high* and *low* examples in the leaf of the pruned decision tree that is reached by test-set example x.

By sorting in this manner, we produce an ordering of the *best* test-set examples as predicted by the machine-learning classifier. The question we would like to answer is the following:

> *Assume we want to get at least* N *'good' probes for a gene, how far down our sorted list do we need to go?*

This question is similar to that asked of information-retrieval systems (e.g., search engines): in order to get N relevant articles, how many of the highest-scoring articles should be returned?

We consider various definitions of a 'good' probe in Figure 1. In Panel (a), we define 'good' as measuring at or higher than 0.5 on our normalized [0–1] scale; about 13.5% of our probes have normalized measured intensities at or above 0.5. For example, when we look at the *test-set* probes with the 10 highest predicted scores, it turns

[†] For categorization tasks, Rumelhart *et al.* (1995) argue that the cross-entropy error function is more appropriate. However, in other work we found little difference in predictive accuracy when comparing the squared-error and cross-entropy error functions and so for this project we selected the more commonly used squared error.

out that for ANN and naïve Bayes nearly all of them had normalized measured intensities at or above 0.5. The 'ideal' curve is a 45-degree line: if *all* of the probes in the top *N* are considered good, then the results would fall on the ideal curve. Panels (c) through (d) report the same information for increasingly strict definitions of 'good' (6.3% of the probes have normalized intensities at or above 0.75, 3.7% at or above 0.9, and, by construction, 2.5% normalize to 1.0). In all cases, the decision-tree learner performs very poorly. Neural networks and naïve Bayes perform surprisingly well, with the neural networks doing slightly better.

Included in Figure 1 is the curve that presents results from ordering probes simply by their predicted melting point. We calculated the melting points using the formula presented by Aboul-ela *et al.* (1985). As can clearly be seen, predictors based on neural networks and naïve Bayes are substantially more accurate than simply using predicted melting point, at least according to our metric.

Figure 2 presents a visualization of the predicted intensities by the learning algorithms for a typical gene region. We generated these curves by manipulating the ratios used to determine the best probes reported in Figure 1. *It is important to note that this figure is for visualization purposes only.* The functions used to generate Figure 2s curves are partially fitted to the testing data, which invalidates their use in quantitative evaluation. These curves are generated as described below.

For naïve Bayes and ANNs, we use the log of the ratio of *prob(label = high) / prob(label = low)* to create a *predicted output* for each probe for a given gene. Similarly, decision trees simply use *prob(label = high).* We then compute the squared error between these predicted values and the normalized measured intensities, across all the probes in a given gene. We next consider raising the predicted values to increasing powers from 1 to 20, and the power with the minimum squared error is chosen for use in these visualization graphs; the selected power used to generate the curves is shown in the figure legend. In our quantitative experiments we are only interested in *relative* predicted values for the various probes (e.g. Figure 1), and raising the scores to increasing exponents has no impact on their sorted order. However, to better visualize the predictions we have found it useful to manipulate the predictions.

The naïve Bayes and ANN predicted curves closely fit the measured probe intensities over high and low intensity values. The decision-tree curve does not fit the actual probe curve nearly as well as the other two algorithms over all intensity ranges. While Figure 2 only shows a short, 50-bp region of probes, the results are typical of what is produced over all of our eight genes.

Figure 3 offers insight into what features are providing the most information for classification, where information gain is computed using Equation 4. This curve is generated by computing the information gain of each individual feature over the eight training datasets; values are normalized to the information content of the highest-scoring feature. *FracC* is the most informative feature - it provides nearly double the information of the next most informative feature. The bell-shaped curves for the $n1 - n24$ and $d1 - d23$ features show that the nucleotides in the middle of the probe are substantially more informative than those at either end of the 24mer, as one might expect.

DISCUSSION

The threshold normalized intensity experiments show that the ANN and naïve Bayes can competently pick efficiently hybridizing probes. When the normalized-intensity threshold is 0.50, both algorithms choose probes whose normalized intensity is above this threshold more than 90% of the time. On average, at even the strictest normalized intensity threshold (1.00), four of the first six probes chosen meet the standard. Though more experimentation is needed to discern what an appropriate normalized intensity threshold is and how many probes are really needed in order to accurately sense a gene, these results are highly encouraging.

Also encouraging are the results of the ANN and naïve Bayes on the task of predicting the normalized intensity across a particular gene. Obviously, the task is very similar to choosing probes above a threshold. The subtle difference is that consistency in prediction of the relative intensities of the weaker probes, not just the top twenty or so, is required in order to make the landscapes line up.

The decision tree performs consistently worse than the other two algorithms on these tasks. This suggests that a more probabilistic representation is preferable to the all-or-nothing nature of a decision-tree model. However, it is possible that other variants of decision-tree induction (e.g. boosting and bagging; Quinlan, 1996b, or regression trees that learn to predict real values directly; Quinlan, 1996a) would perform well on this task.

The relative information gain of different features is also quite interesting. Bases near the ends of the probe are clearly less important than those in the middle. Intuitively considering the kinetics of the situation, this makes sense. Imagine a 24 bp or so strand of 'target' RNA bound to a complimentary 24mer DNA probe by just the four bases at the very end of the strand. This leaves a 20-base-long RNA arm free to float, twist, and generate torques that might break the duplex apart. Now imagine, instead, that it is the middle four bases of the two complementary strands that are bound together. In this case, the swinging ends are only ten bases long and the torques will be much smaller and less likely to free the RNA. So, in order for the RNA strand to stay connected to the probe long enough to hybridize fully, it is most critical that the middle of the probe can make a good bond to the RNA.

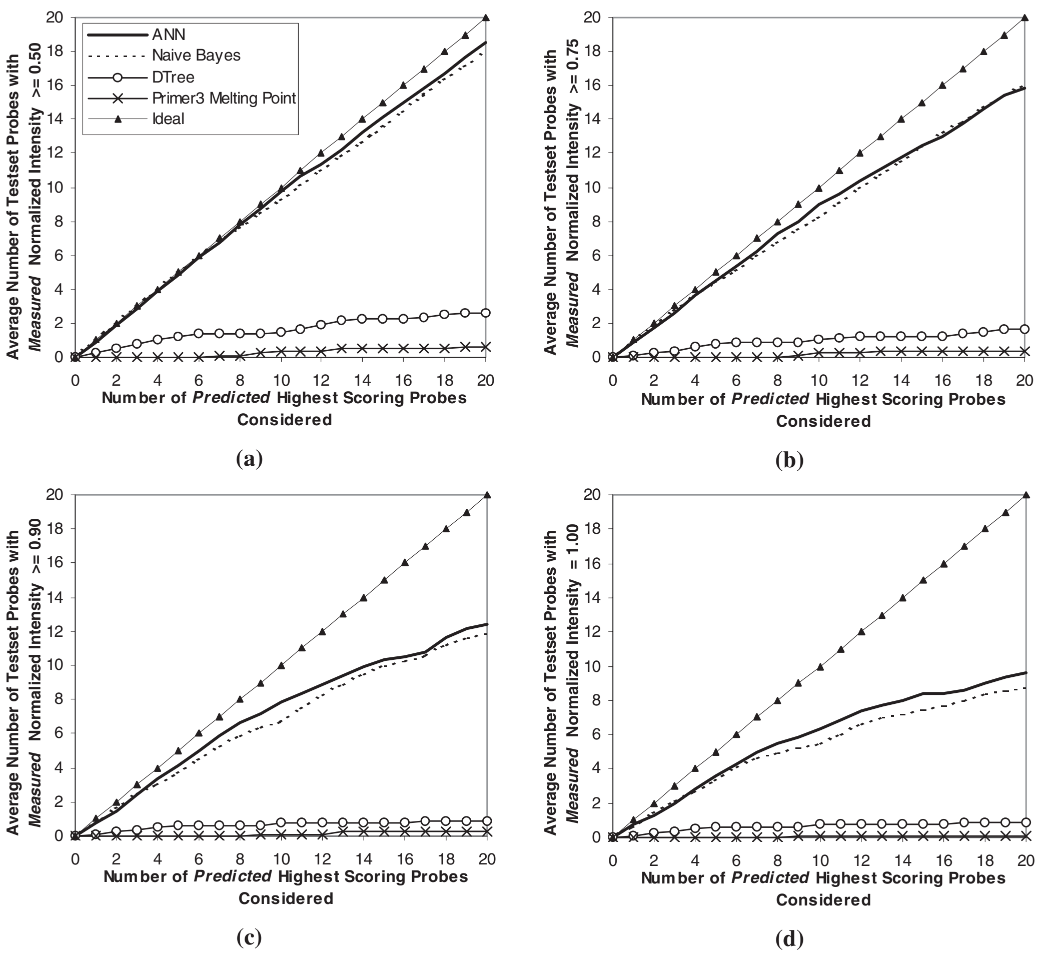

Fig. 1. Number of *Test-set* probes in the *N* highest-scoring predictions that exceed a given threshold for their normalized measured intensity (per learning algorithm and averaged over the eight test sets

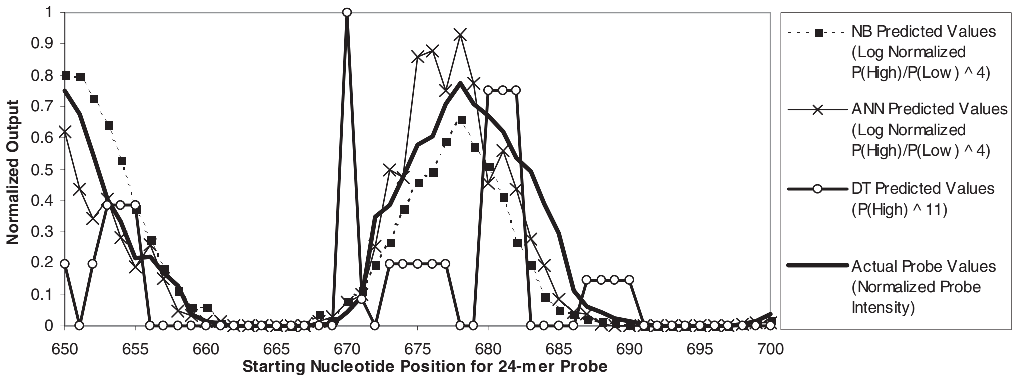

Fig. 2. Probe intensity and classifier output versus starting nucleotide position for 24mer probe for gene 8

The relative information gain of overall base content is also informative. 'C' content is, by far, the most informative feature. Along with 'G' content, they are the two most important single bases. This is not surprising given the relative stability of the G-C bond over the A-T bond.

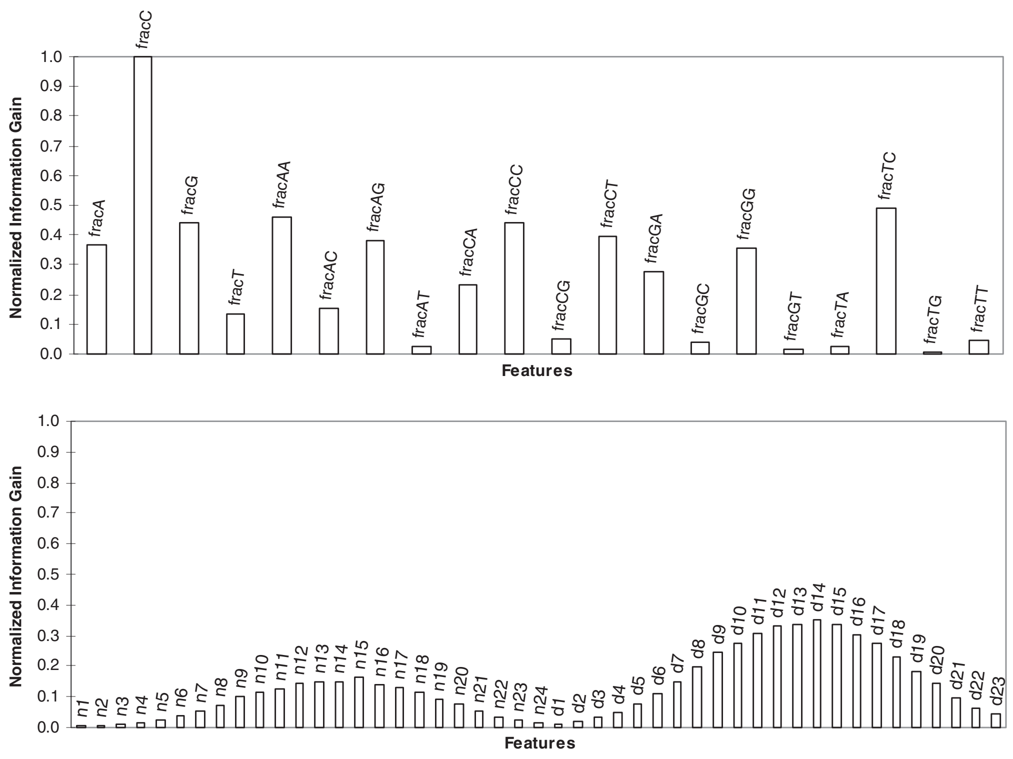

Fig. 3. Normalized average information gain per probe feature over eight training sets (Table 1 contains definitions for these features).

The dimer frequencies are also interesting, but more difficult to explain. The approximate symmetry (the importance of the frequency of XY is similar to the importance of the frequency of YX) probably reflects the approximate symmetry in stacking energies between a duplex and its exact reverse (switching 3′ and 5′ ends). These stacking energies have a great influence on the energetic of characteristics of the hybridization. The information content of these dimers roughly follows the relative stacking energies of the dimers with the exception of the CG and GC dimers that are known to have high stacking energies, but show low information content as features. This is a subject that needs further study.

FUTURE WORK

Though we have developed strategies that successfully find probes that bind with RNA when the RNA is present in a sample, one limitation of our approach is that it may not filter out probes that bind non-specifically. The use of *mismatch* probes (probes that are the nearly same as the target probe with only a small number of exceptions) alongside the target probe is a common method for identifying this behaviour. If the binding is nonspecific, the mismatch probes will usually also have high intensity values.

We currently are extending our experiments to address non-specific probes. We are also investing the task of predicting the amount of fluorescence measured when the sample applied to the microarray does *not* contain the RNA for a given probe's gene. More specifically, we are now investigating *four* learning tasks (and are also converting from categorical to real-valued prediction). For a given probe, we wish to predict the measured fluorescence under four conditions:

1. the probe's fluorescence level when the associated gene's RNA is (substantially) *present* in the sample,
2 the probe's fluorescence level when the associated gene's RNA is *absent* from the sample,
3. the maximum fluorescence level for the probe's *mismatch probes* when the associated gene's RNA is (substantially) *present* in the sample, and
4. the maximum fluorescence level for the probe's *mismatch probes* when the associated gene's RNA is *absent* from the sample.

The predicted quality of a potential probe will be a combination of these four predicted values.

We also plan to evaluate the use of machine learning in conjunction with other methods. For example, we are considering using a heuristic method to filter out some probes that the learner did not; if self-complementarity proves to be a difficult characteristic for the learner to infer, a simple computational test for self-complementarity could be used to eliminate some probes.

Another topic we are addressing is evaluating machine learners on a much larger set of probes, including those for eukaroyotic genes.

Finally, we are currently considering using enriched sets of features to represent probes. For instance, we are using the Zuker *et al.* (1999) *mfold* algorithm to predict secondary structures in the probes (and corresponding targets), and are devising features based on *mfold*'s predictions. ANNs and naïve Bayes appear to be highly accurate even when only using the very simple (and easy to compute) features of Table 1, so we may not need richer features in order to obtain an acceptable level of predictive accuracy. However, we are also interested in judging which features (and combinations of features) prove helpful in making accurate computational models of probe quality, in order to obtain more insight into improving the probe-design process.

CONCLUSION

We address the important task of predicting the quality of the various short probes that can be used to represent a given gene on a microarray. Our approach is to apply three well-established, machine-learning algorithms to this task in order to see how well machine learning works on this problem and also to determine which of the evaluated learning approaches is most accurate on this task. We obtained training data for our problem by putting all possible 24mers from eight prokaryotic genes on a gene chip, then measuring the fluorescence of each probe when a sample where all eight genes were highly expressed was applied to the microarray. Ideally, all the probes would hybridize strongly with the sample, but 45% of the probes hybridized very weakly (basically, at the background-noise level). We trained our learning algorithms to predict which probes would hybridize strongly and which would not.

Two learning algorithms performed remarkably well on ‘held aside’ test data: naïve Bayes and artificial neural networks. Both worked much better than our usage of a decision-tree-induction algorithm. They also performed much better than using calculated melting point to predict the quality of probes. Following training, both naïve Bayes and neural networks can quickly rate candidate probes; the time needed to judge probe (quasi-)uniqueness will be much more than that needed to rate the probe's hybridization quality. Our results strongly suggest that ‘off the shelf’ machine-learning methods can greatly aid the important task of probe selection for gene-expression arrays.

ACKNOWLEDGEMENTS

We wish to thank Darryl Roy for helping in creating the training data. This research was funded by grants NIH Grant 2 R44 HG02193-02, NLM 1 R01 LM07050-01, NSF IRI-9502990, NIH 2 P30 CA14520-29, and NIH 5 T32 GM08349.

REFERENCES

Aboul-ela,F., Koh,D. and Tinoco,H. (1985) Base-base mismatches: thermodynamics of double helix formation for $dCA_3XA_3G + dCT_3YT_3G(X, Y = A, C, G, T)$. *Nucleic Acids Res.*, **13**, 4811–4824.

Breslauer,K.J., Frank,R., Blocker,H. and Marky,L.A. (1986) Predicting DNA duplex stability from the base sequence. *Proc. Natl Acad. Sci. USA*, **83**, 3746–3750.

Li,F. and Stormo,G. (2001) Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics*, **17**, 1067–1076.

Lockhart,D.J., Dong,H., Byrne,M.C., Follettie,M.T., Gallo,M.V., Chee,M.S., Mittmann,M., Wang,C., Kobayashi,M., Horton,H. and Borwn,E.L. (1996) Expression monitoring by hybridization to high density oligonucleotide arrays. *Nat. Biotechnol.*, **14**, 1675–1680.

Kurata,K. and Suyama,A. (1999) Probe design for DNA chips. *J. Japanese Soc. Bioinformatics*.

Mitchell,T. (1997) *Machine Learning*. McGraw-Hill, Boston, MA.

Quinlan,J.R. (1986) Induction of decision trees. *Machine Learning*, **1**, 81–106.

Quinlan,J.R. (1996a) *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA.

Quinlan,J.R. (1996b) Boosting, Bagging, and C4.5. *Proceeding of the National Conference on Artificial Intelligence (AAAI-96)*. MIT/AAAI Press.

Rumelhart,D., Durbin,R., Golden,R. and Chauvin,Y. (1995) *Back-propagation: The Basic Theory*, In Back-propagation: Theory, Architecture, and Applications, Chauvin,Y. and Rumulhart,D.E. (eds), Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 1–34.

Sarle,W.S. (1995) Stopped training and other remedies for overfitting. *Proceedings of the 27th Symposium of the Interface of Computing Science and Statistics*. pp. 352–360.

Shannon,C.E. (1948) A mathematical theory of communication. *Bell System Technical Journal*, **27**, 379–423 and 623–656, July and October.

Singh-Gasson,S., Green,R.D., Yue,Y., Nelson,C., Blattner,F.R., Sussman,M.R. and Cerrina,F. (1999) Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nat. Biotechnol.*, **17**, 974–978.

Snustad,D.P. and Simmons,M.J. (1999) *Principles of Genetics*, 2nd Edition, Wiley, New York, NY.

Zuker,M, Mathews,D.H. and Turner,D.H. (1999) Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide. In Barciszewski,J. and Clark,B.F.C. (eds), *RNA Biochemistry and Biotechnology*, NATO ASI Series, Kluwer Academic Publishers.