Finding Discriminatory Genes: a methodology for validating microarray studies

***Abstract*—This paper explores the challenge of efficiently collecting data to find which genes (from a given set of candidates) are differentially expressed . We consider several algorithms for this task, including some that assume there are only two types of genes: those that are not differentially expressed, and those that are differentially expressed to the same level. We provide a framework for evaluating such algorithms and also present an algorithm that has nice theoretical properties and performs very well on both real and simulated data.**

***Keywords*—Biomarker Discovery, Microarray Analysis, Sequential Design.**

I. Introduction

Many researchers in bioinformatics and related fields are interested in the biomarker discovery problem; the search for genes, proteins and other small molecules that are associated with a specific disease or phenotype. As gene expression microarrays can measure the expressions of tens of thousands of genes simultaneously from a given sample, they have become one of the dominant tools for biomarker discovery over the last 10 years. There are currently 40 thousand publicly available datasets from such studies available on the NCBI-GEO database [1].

The typical result from a microarray association study is a list of 'differentially expressed' genes [2]. Unfortunately, the biomarkers found in one study are often very different from the ones found in another [3]. The discrepancy has been attributed to statistical issues that originate from the extremely high dimensionality of the data [4], and also to the sensitivity of the results to the definition of differential expression used [5]. As a result of these deficiencies, there is a growing interest from members within the community to mandate a validation study [6], [7].

In a validation study, researchers will use a more accurate and reliable technology, such as PCR [6], [7], to confirm that their reported genes actually have different distributions of expression levels in the two classes of interest. We consider an experimentation setup wherein the researcher collects data in a series of 'probes', where each probe is a measurement of the expression level of a specific gene for a specific tissue sample. After observing the values of a certain number of probes the experiment ends and the results are reported.

We focus on the challenge of designing such probe sequences to efficiently identifying which genes are truly differentially expressed. Here, we are given a set of N genes, and have the opportunity to probe the expression value of any specific gene on an instance of a specified class, *e.g.*, you can ask for the expression value of gene#3 from a cancer patient, or for the gene#22 of a control patient, etc. After observing a pre-specified number of probes, our algorithm then returns a subset, possibly all or none, of the genes that it believes are differentially expressed. The algorithm is evaluated by the accuracy of its predictions: rewarded for correctly identifying differentially expressed genes, and penalized both for both the non-differentially expressed genes it incorrectly identifies and the differentially expressed genes that it fails to identify.

In this paper we present three algorithms. The first algorithm, Round Robin (RR), is a naive experimental design, which is easy to analyze under basic modelling assumptions. We then show a more intelligent sequential method, Greedy, that is difficult to analyze and so its performance cannot be predicted. Our final algorithm, Not As Naive (NAN), acts as a stepping stone between Greedy and RR. We prove under the modelling assumptions that Greedy must do no worse than NAN, which means NAN's performance provides a lower bound on Greedy's. Furthermore, we show that on real gene expression data where the modelling assumptions do not hold, NAN has the best performance.

The algorithms are presented and analyzed in Section II. (Appendix A provides references for the distributions necessary in the analyses.) Section III compares the algorithms on both synthetic and real data to provide evidence supporting our claims.

We close this section by providing a formal description of our problem and then presenting similar problems in the literature.

A. Formal Model

In keeping with the rest of the gene expression analysis literature, we assume that expression levels follow a normal distribution [8]. For each gene, there are then two normal distributions $\mathcal{N}(\mu_+, \sigma^2)$ and $\mathcal{N}(\mu_-, \sigma^2)$ for the $+/-$ classes respectively[1]. If $\mu_+ \neq \mu_-$, the gene is said to be differentially expressed. In microarray studies $\mu_+ - \mu_-$ tends to follow a heavy tail distribution in accordance with Zipf's law [9], but for this paper we consider a simple binary model: each gene $g^{(i)}$ is either discriminatory[2], in that $(\mu_+^{(i)} - \mu_-^{(i)})/\sigma^{(i)} = \Delta$ (for a fixed Δ, over all i), or non-discriminatory.

$$\frac{\mu_+^{(i)} - \mu_-^{(i)}}{\sigma^{(i)}} = \begin{cases} \Delta & \rightarrow \quad \text{discriminatory} \\ 0 & \rightarrow \quad \text{non-discriminatory} \end{cases} \qquad (1)$$

We begin with the further simplifying assumption that we know the distribution for the $-$ classes of all genes is a standard $\mathcal{N}(0, 1)$ distribution. The assumption does not cause a loss of generality as we can standardize data from any normal

[1] Our simplified model assumes $\sigma = \sigma_+ = \sigma_-$; this is not standard.

[2] We use the term "discriminatory" to avoid the ambiguities associated with the term "differential expression".

distribution given that we know its parameters. A probe is defined to be an observation of a single expression value drawn from the $+$ class distribution for a specific gene. Later, for our real data experiments, we will define a probe as a pair of observations drawn from both the $+$ and $-$ distributions of a specific gene.

We consider the task:

Definition [discriminatory genes problem] Given a set of N genes and a budget $B \in \mathbb{N}$ of probes, return the subset of size N_d genes that are discriminatory w.r.t. Δ (1). Equivalently the goal may be to find the $N_n = N - N_d$ subset of genes at are not discriminatory[3].

In this paper we consider algorithms that know a priori N_d, N_n, Δ and B. Algorithms will be evaluated based on their true and false positives in the returned set of genes, as well as the true and false negatives left unreturned, respectively denoted as TP, FP, TN and FN. The evaluation function,

$$eval(\text{TP}, \text{FP}, \text{TN}, \text{FN}), \tag{2}$$

$$\frac{d}{d\text{TP}} eval(\text{TP}, \text{FP}, \text{TN}, \text{FN}) \geq 0 \tag{3}$$

$$\frac{d}{d\text{FP}} eval(\text{TP}, \text{FP}, \text{TN}, \text{FN}) \leq 0 \tag{4}$$

may be any generic function subject to two constraints (3) and (4). Thus, the algorithm is rewarded for each correct decision it makes and penalized for every wrong one, but the rewards and penalties are not necessarily equal nor constant.

As there are typically disproportionately fewer discriminatory genes than non-discriminatory genes in a study, we use the F_1 measure as our evaluation function.

$$F_1 = eval(\text{TP}, \text{FP}, \cdot, \text{FN}) = \frac{2 \times \text{TP}}{\text{FN} + \text{FP} + 2 \times \text{TP}} \tag{5}$$

We define $F_1(0,0,0) = 1$ in the event that no genes are discriminatory. The F_1 measure provides a balance between the precision and recall on the returned set. Note that although we favour F_1, the algorithms and analyses presented here hold for any function satisfying (2).

B. Related Works

The problem of testing if a *single* gene is discriminatory exactly fits the framework of the sequential probability ratio tests (SPRT) [10], that provide a methodology for sequentially collecting data to test whether data is drawn from one of two hypotheses, $H0$ versus $H1$. The methodology is optimal in that no other sequential method can perform the test to the same degree of accuracy with fewer expected probes. Schuurmans and Greiner [11] consider the case where a finite budget is imposed on a SPRT, which makes it exactly the same as the discriminatory genes problem if only one gene is considered. While we could try to cast the general problem of finding discriminatory genes as N parallel SPRTs operating under a common probe budget, it is unclear how to set the SPRT parameters nor how to allocate the probes across the SPRTs to maximize the expected evaluation.

The m-best arm identification is also a very similar but different problem [12], [13], [14]. Given a set of N distributions (arms), the goal of the best arm identification problem is to construct an algorithm to pay for probes to learn about the arms and return a set of m good arms, where good means that the true mean of the returned arms is at most ϵ less than the mean of the m'th best arm, *e.g.*, if we have arms with means $\{0.2, 0.4, 0.6, 0.7, 0.8, 0.9\}$ and $m = 2$ then a valid solution is to return two arms with mean greater than $0.8 - \epsilon$. If $\epsilon > 0.1$ then the arms $\{0.7, 0.9\}$ are a valid solution but if $\epsilon < 0.1$ then the only solution is $\{0.8, 0.9\}$. The discriminatory gene problem can be described in this framework by setting $m = N_d$ and $\epsilon = 0$. Unfortunately, algorithms for the m-best arm identification are constructed in a PAC setting; given m and ϵ find a good set of arms with probability of error bounded by δ. The algorithms are typically constructed such that their asymptotic probe complexity[4] can be analyzed by a series of Hoeffding and union bounds. Our problem is is different as we start with a hard constraint on the total number of probes.

A second very strong difference is that the evaluation for m-best arm identification is very harsh; they say that an algorithm fails if it does not return exactly all the discriminatory genes and nothing else. Our F_1 based evaluation measure is more appropriate for the community doing biomarker discovery.

Finding discriminatory genes can also be viewed as structure learning problem in graphical models. In a minimal Naive Bayes model there will only be an arc from the class node to a gene IFF their distributions are dependent [15]. If their distributions are dependent then the gene must then be differentially expressed w.r.t. to the class, so finding the true minimal Naive Bayes model solves the discriminatory genes problem. This relates to work, by Tong and Koller [16] and extended by Li *et. al.* [17], that sought an efficient way to gather the relevant generative information by requesting a sequence of [feature,instance] probes. However, those papers focused only on finding the parameters for a fixed structure, but did not consider finding the minimal (sub)structure.

II. Algorithms

A. Round Robin (RR)

Round Robin distributes the probe budget across all the genes and then returns all those with sample mean above a threshold τ. It is also possible to return the top k genes based on sample means but we will show in Section II-C that that behaviour is better described by the NAN algorithm.

Algorithm 1 RR($N \in \mathbb{N}, B \in \mathbb{N}, \tau \in \Re$)

```
1: $p \leftarrow \lfloor B/N \rfloor$
2: probe each gene $p$ times
3: estimate the mean for each gene
4: **return** all genes with sample mean $\geq \tau$
```

RR is naive in that it does not use information obtained from the probes to guide its future probes. For this reason RR is important as it sets a baseline that other algorithms must

[3]This alternative view is useful for the task of identifying "house keeping genes".

[4]The probe complexity of these algorithms is the number of probes spent before termination.

beat. Since the probe data is normally distributed, the sample mean will also be normally distributed, which makes it easy to evaluate the probability of labeling a gene as discriminatory. This leads to two binomial distributions, one for the number of TP, p_{tp}, returned and one for the FP, p_{fp}. Using $\Phi(\cdot)$ as the standard normal CDF the expected evaluation is obtained by,

$$p_{tp} = 1 - \Phi\left(\frac{\tau - \Delta}{\sqrt{M/N}}\right) \tag{6}$$

$$p_{fp} = 1 - \Phi\left(\frac{\tau}{\sqrt{M/N}}\right) \tag{7}$$

$$P(\text{TP} = a) = \binom{N_d}{a} p_{tp}^a (1 - p_{tp})^{N_d - a} \tag{8}$$

$$P(\text{FP} = b) = \binom{N_n}{a} p_{fp}^b (1 - p_{fp})^{N_n - b} \tag{9}$$

$$\begin{aligned} &E[eval(\cdot)] \\ &= \sum_{a=0}^{N_d} \sum_{b=0}^{N_n} P(\text{TP} = a) P(\text{FP} = b) eval(\text{TP}, \text{FP}, \text{TN}, \text{FN}). \end{aligned} \tag{10}$$

Given the problem description N_d, N_n, Δ and B, we can use gradient ascent to easily tune τ to maximize the expected evaluation.

B. Greedy

A more intelligent approach than RR is to use the information gained by the probes to guide a sequential method. An obvious strategy is exploit the knowledge that we know the true distributions for each type of gene (although not which gene is which type) and so compute the log-likelihood ratio (LLR) of the two models for each gene. Using $x_j^{(i)}$ to denote the j'th probe for gene i, the LLR $\Lambda^{(i)}$ for gene $g^{(i)}$ after seeing p probes is,

$$\Lambda^{(i)} = \frac{\Delta}{2} \sum_{j=1}^{p} (2x_j^{(i)} - \Delta). \tag{11}$$

A greedy strategy would then be to decide *a priori* a number of genes to return k and then probe the gene with the k'th largest LLR at each step. Such a strategy maximizes the minimum uncertainty of the genes returned.

Algorithm 2 Greedy($N \in \mathbb{N}, B \in \mathbb{N}, \Delta \in \Re, k \in \mathbb{N}$)

```
1: probe each gene once
2: for probe = $N+1$ to $M$ do
3:     compute $\Lambda^{(i)}$ for each gene
4:     $i^* \leftarrow$ index of gene $k$'th largest $\Lambda^{(i)}$
5:     probe $g^{i^*}$
6: return top $k$ genes based on $\Lambda^{(i)}$
```

Given the problem description N_d, N_n, Δ and B, it is difficult to predict the performance of Greedy as a function of k. One may argue that if the budget B is small then we should use a small value of k as that will lead to high precision, but of course having a larger k would mean having a better recall. What we can say about Greedy is that, depending on how we choose k, it will eventually lock on to different sets of the genes. If $k = N_d$ then Greedy will eventually find all the discriminating genes and then proceed to keep probing them. If $k < N_d$ then Greedy will then eventually lock onto a subset of discriminatory genes and spend its remaining budget on them. If $k > N_d$ then Greedy will eventually find all the discriminatory genes and then proceed to spend the remaining probes on the non-discriminatory genes. For our experiments in Section III we set $k = N_d$.

C. Not As Naive (NAN)

The main idea of the NAN algorithm is to break the overall sequence of probes into a series of rounds, during each of which all (remaining) genes are probed equally. At the end of a round, the algorithm then either accepts the gene with the largest sample mean as discriminatory, or rejects the gene with smallest sample mean as non-discriminatory. Once a gene is either accepted or rejected, NAN commits to the decision and stops probing it. The motivation for this behaviour is that it is highly likely that genes with extreme sample means will mostly be correctly identified, and by ceasing to probe them, more effort can be spent on the genes that are harder to identify. To help analyze NAN, we require a preset pattern of accept/reject decisions $\mathbf{d}$ and a fixed budget to spend probing the genes in each round $\mathbf{m} = [\mathbf{m}_1, \ldots, \mathbf{m}_N]$.

Algorithm 3 NAN($N \in \mathbb{N}, B \in \mathbb{N}, \mathbf{d} \in \{a, r\}^N, \mathbf{m} \in \mathbb{N}^N$)

```
Require: $\sum_{p=1}^{N-1} (N + 1 - p) m_p \leq B$
Require: $\mathbf{m}_N = 0$
 1: $active \leftarrow \{1, \ldots, N\}$
 2: $accept \leftarrow \{\}$
 3: $reject \leftarrow \{\}$
 4: for $round = 1$ to $N$ do
 5:     probe each gene in $active$, $\mathbf{m}_{round}$ times
 6:     if $\mathbf{d}_{round} = a$ then
 7:         $i \leftarrow \arg\max_{i \in active} \Lambda^{(i)}$
 8:         $accept \leftarrow accept \cup \{i\}$
 9:     else
10:         $i \leftarrow \arg\min_{i \in active} \Lambda^{(i)}$
11:         $reject \leftarrow reject \cup \{i\}$
12:     $active \leftarrow active \setminus \{i\}$
13: return $accept$
```

Interestingly NAN can be viewed as a generalization of RR by forcing the algorithm to spend all the probe budget on the first round; here $\mathbf{m}_1 = \lfloor B/N \rfloor$ and $\mathbf{m}_2 = \mathbf{m}_3 = \cdots = \mathbf{m}_N = 0$. A slight difference is that NAN will return a fixed number of genes, $k|$, as discriminatory while RR will return a variable amount based on τ. If k and τ are tuned to to optimize the evaluation for both algorithms then, RR will on average return $\approx k$ genes and thus both algorithms will receive the same score[5].

[5]For the optimal τ RR might return some fraction of of the genes in order to get a slightly higher expected evaluation but this difference will be negligible given the variance of the distributions.

The median elimination algorithm [12] also is a special case of NAN. Median elimination is an algorithm for the 1-best arm identification that works by doing rounds of probes and eliminating half the genes every round. For a small problem with $N = 8$ genes and $B = 14$ probes, median elimination would probe each gene once, and eliminate the 4 worst, then probe the remaining 4 genes once and eliminate the worst 2, and finally probe the remaining 2 and keep the best. This behaviour is easily encoded in the NAN parameters $\mathbf{d} = [r, r, r, r, r, r, r, a]$ and $\mathbf{m} = [1, 0, 0, 0, 1, 0, 1, 0]$. The SAR algorithm [14] is a more general m-best arm algorithm that is also very similar to NAN. We can set $\mathbf{m}$ such that both algorithms allocate their probes in the same fashion but SAR decides to accept/reject genes based on a statistical test whereas NAN requires the preset sequence $\mathbf{d}$.

D. Analyzing the NAN algorithm

Here we show how to compute a very good approximation of the probability that NAN correctly accepts a discriminatory gene at one of its decision points. The probability of correctly rejecting a non-discriminatory gene can be computed in an analogous manner. Using this method we can then construct a simple dynamic programming algorithm that computes the expected evaluation score by obtaining a distribution over all possible outcomes of NAN.

The key insight is that the decision to accept a gene is correct if the maximum sample mean of the active discriminatory genes is larger than the maximum sample mean of the non-discriminatory genes. If we have a reasonable number of genes of both types, we can use the Fisher-Tippett-Gnedenko theorem that states that, given $z = \max_{i=1..N} x_i$ where x_i are i.i.d. normal r.v.'s from $\mathcal{N}(\mu, \sigma^2)$ then z is a Gumbel r.v.:

$$z \sim gumbel(\alpha, \beta) \tag{12}$$

$$\alpha = \sigma\Phi^{-1}\left(1 - \frac{1}{N}\right) + \mu \tag{13}$$

$$\beta = \sigma\left[\Phi^{-1}\left(1 - \frac{1}{eN}\right) - \Phi^{-1}\left(1 - \frac{1}{N}\right)\right] \tag{14}$$

At the time of accepting a gene, we have two Gumbel r.v.'s d and n, corresponding to the maximum sample mean of the active discriminatory and non-discriminatory genes. The probability that NAN accepts a discriminatory gene is then $P(d - n > 0)$. Unfortunately, the difference of Gumbel r.v.'s can only be computed analytically if they are i.i.d., which is generally not true here[6]. However, if the scale parameter of the Gumbel distribution is sufficiently small then the distribution can be well approximated by moment matching to a normal distribution, to produce a distribution that agrees on median, mode, and entropy[7].

During execution there will either be a large number of genes that have been probed very little, or a very small number genes that have been probed heavily. The combined effect is that the scale parameter for the Gumbel distribution will be reasonably small.

After moment matching, the probability is easily calculated because the difference of normal r.v.'s is normal and we can get the probability via normal CDF. Given $d^{(r)}$ and $n^{(r)}$ discriminatory and non-discriminatory genes at round r, each of which have been probed p times, Algorithm 4 provides a method for computing the probability of correctly accepting a gene.

To get the desired expected score, we first use the dynamic programming method presented in Algorithm 5 to compute the probabilities of all possible decision outcomes. Then since there is always one gene left in the active set at the end of NAN, Algorithm 6 accounts for this to get a final count of the TP and FP for each outcome and then computes the expectation.

In total the analysis requires $O(N^3)$ time. However, this can be reduced to $O(N)$ for the special case where the evaluation is the $0/1$ loss[8] because then we need only compute the probability that no errors are made. The bottle neck of the computation is the **for** loop at line 19 of Algorithm 5. The computation can be sped up by only looping over the most probable states instead of all states. Such a trick was employed in [18] to speed up a similar algorithm.

Algorithm 4 approx($d^{(r)} \in \mathbb{N}, n^{(r)} \in \mathbb{N}, p \in \mathbb{N}, \Delta \in \Re$)

```
1:  if $d^{(r)} = 0$ return 0
2:  if $n^{(r)} = 0$ return 1
3:  if $p = 0$ return $d^{(r)}/(d^{(r)} + n^{(r)})$
4:  $\sigma \leftarrow p^{-1/2}$
5:  if $N_d > 1$ then
6:      $\alpha_d \leftarrow \sigma\Phi^{-1}\left(1 - (d^{(r)})^{-1}\right) + \Delta$
7:      $\beta_d \leftarrow \sigma\left[\Phi^{-1}\left(1 - (e \times d^{(r)})^{-1}\right) - \Phi^{-1}\left(1 - (d^{(r)})^{-1}\right)\right]$
8:      $m_d \leftarrow \alpha_d + \gamma\beta_d$ {$\gamma$ is the euler-mascheroni constant}
9:      $v_d \leftarrow \frac{1}{6}(\pi\beta_d)^2$
10: else
11:     $m_d \leftarrow \Delta$
12:     $v_d \leftarrow \sigma^2$
13: if $N_n > 1$ then
14:     $\alpha_n \leftarrow \sigma\Phi^{-1}\left(1 - (n^{(r)})^{-1}\right)$
15:     $\beta_n \leftarrow \sigma\left[\Phi^{-1}\left(1 - (e \times n^{(r)})^{-1}\right) - \Phi^{-1}\left(1 - (n^{(r)})^{-1}\right)\right]$
16:     $m_n \leftarrow \alpha_n + \gamma\beta_n$
17:     $v_n \leftarrow \frac{1}{6}(\pi\beta_n)^2$
18: else
19:     $m_n \leftarrow \Delta$
20:     $v_n \leftarrow \sigma^2$
21: if $(d^{(r)} = n^{(r)})$ and $(d^{(r)} > 1)$ then
22:     return 1-$\left(1 + e^{\Delta/\beta_d}\right)^{-1}$
23: else
24:     return $\Phi\left(\frac{m_d - m_n}{\sqrt{v_d + v_n}}\right)$
```

[6]Iff $N_d = N_n$ then we can consider d and n to be i.i.d. Gumbel r.v.s of generated from the max of N_d $\mathcal{N}(0, \sigma^2)$ normal variables and then compute the probability $P(d + \Delta - n > 0)$ exactly, see the Logistic distribution in Appendix A.

[7]This result is easy to see from the formulae in Appendix A. As $\beta \to 0$ the mean, median, and mode converge to α and the entropy converges to $\log(\beta)$, which is the same as those for a normal distribution with equal mean and variance.

[8]This means the algorithm scores 1 iff it returns exactly all the discriminatory genes, else-wise it scores 0, *i.e.*, $F_1 = 1$.

Algorithm 5 NANStateProbabilities($N \in \mathbb{N}, \Delta \in \Re, \mathbf{d} \in \{a,r\}^N, \mathbf{m} \in \mathbb{N}^N$)

```
1: $p_{state}(\cdot,\cdot,\cdot,\cdot) \leftarrow 0$
2: $p \leftarrow 0$
3: $A \leftarrow 0$
4: $R \leftarrow 0$
5: for $i = 1$ to $N-1$ do
6:     $p+ = \mathbf{m}_i$
7:     if $\mathbf{d}_i = a$ then
8:         $A+ = 1$
9:     else
10:        $R+ = 1$
11:    for $\forall$TP, FP, TN, FN :
       $\text{TP}+\text{FP}+\text{TN}+\text{FN} = i-1$,
       $\text{TP}+\text{FN} \leq A$,
       $\text{FP}+\text{TN} \leq R$ do
12:        $N'_d \leftarrow N_d - \text{TP} - \text{FN}$
13:        $N'_n \leftarrow N_n - \text{TN} - \text{FP}$
14:        if $\mathbf{d}_i = a$ then
15:            $\theta \leftarrow approx(N'_d, N'_n, p, \Delta)$
16:            $p_{state}(\text{TP}+1,\text{FP},\text{TN},\text{FN})+ =$
                   $\theta \times p_{state}(\text{TP},\text{FP},\text{TN},\text{FN})$
17:            $p_{state}(\text{TP},\text{FP}+1,\text{TN},\text{FN})+ =$
                   $(1-\theta) \times p_{state}(\text{TP},\text{FP},\text{TN},\text{FN})$
18:        else
19:            $\theta \leftarrow approx(N'_n, N'_d, p, \Delta)$
20:            $p_{state}(\text{TP},\text{FP},\text{TN}+1,\text{FN})+ =$
                   $\theta \times p_{state}(\text{TP},\text{FP},\text{TN},\text{FN})$
21:            $p_{state}(\text{TP},\text{FP},\text{TN},\text{FN}+1)+ =$
                   $(1-\theta) \times p_{state}(\text{TP},\text{FP},\text{TN},\text{FN})$
22: return $p_{state}$
```

Algorithm 6 NANscore($N \in \mathbb{N}, \mathbf{d} \in \{a,r\}^N$)

```
1: $p_{state} \leftarrow NANStateProbabilities(N, \Delta, \mathbf{d}, \mathbf{m})$
2: $A = \sum_i^{N-1} \mathrm{I}(\mathbf{d}_i = a)$
3: $R = N - 1 - A$
4: $E_x \leftarrow 0$
5: for $\forall$TP, FP, TN, FN :
   $\text{TP}+\text{FP}+\text{TN}+\text{FN} = N-1$,
   $\text{TP}+\text{FN} \leq A$,
   $\text{FP}+\text{TN} \leq R$ do
6:     $N'_d \leftarrow N_d - \text{TP} - \text{FN}$
7:     $N'_n \leftarrow N_n - \text{TN} - \text{FP}$
8:     if $\mathbf{d}_N = a$ then
9:         if $N'_d = 1$ then
10:            $x \leftarrow eval(\text{TP}+1,\text{FP},\text{TN},\text{FN})$
11:        else
12:            $x \leftarrow eval(\text{TP},\text{FP}+1,\text{TN},\text{FN})$
13:    else
14:        if $N'_n = 1$ then
15:            $x \leftarrow eval(\text{TP},\text{FP},\text{TN}+1,\text{FN})$
16:        else
17:            $x \leftarrow eval(\text{TP},\text{FP}+1,\text{TN},\text{FN}+1)$
18:    $E_x+ = x \times p_{state}(\text{TP},\text{FP},\text{TN},\text{FN})$
19: return $E_x$
```

E. NAN provides a lower bound for Greedy

Here we show that the expected score of Greedy is lower bounded by NAN. For this to hold true, we assume that both algorithms have been tuned to return the same number of genes. Thus, Greedy can behave exactly as NAN and get the same score or it can deviate in its behaviour; bellow we will show such deviations will increase its expected evaluation. To show this, we enumerate below all the ways in which Greedy can deviate from NAN.

Some of the arguments hinge on the notion of "exploration" done by greedy, *i.e.*, the uniformity of the Greedy's probe allocation. RR would be an example of 100% exploration. NAN has some forced exploration pattern dictated by its $\mathbf{m}$ parameter. The exploration of an an algorithm can be thought of as a a measure of how close its probe distribution is to the uniform distribution.

Greedy and NAN make the same probe allocation:
If both algorithms make the same probe allocation, *i.e.*, both algorithms have probed each gene $g^{(i)}$ $p^{(i)}$ times, but differ in their returned gene sets, the set returned by Greedy must have a higher expected evaluation. This is obvious as Greedy returns the set of genes that is most likely to be discriminatory, so if there is a discrepancy NAN must have selected a gene that has a lower probability of being discriminatory than any of those selected by Greedy. Thus Greedy has a higher expected precision and by the monotonicity of the evaluation function w.r.t. TP, it will therefore have a higher expected score.

Greedy has done less exploration than NAN:
The Greedy algorithm is designed to keep probing what it believes to be the discriminatory genes, and in doing so it increases its confidence that they are discriminatory. If Greedy has locked on to a set of genes and is spending notably more probes on them compared to NAN, it will again have higher confidence on the genes in its returned set.

Greedy has done more exploration than NAN:
If Greedy has done more exploration than NAN this means the problem is particularly hard and it is unlikely that either algorithm will do well. Their expected scores will be within error bars of each other.

III. Experiments

We run two types of experiments. The first experiment is done on synthetic data to show that indeed analysis of the NAN algorithm does lower bound of the performance of the Greedy algorithm. The remaining experiments we run on more realistic synthetic data and real gene expression data where NAN can outperform Greedy and RR.

A. Synthetic Data Experiment 1

We consider a simple problem with $N_d = 3$ discriminatory genes and $N_n = 7$ non-discriminatory genes. The expression levels for the discriminatory genes follow a $\mathcal{N}(1,1)$ distribution and the non-discriminatory genes follow a $\mathcal{N}(0,1)$ distribution. Recall we are just consider ing the $+$ class, as here we know that the $-$ class is drawn from $\mathcal{N}(0,1)$. For evaluation we use the F_1 measure (5).

For fair comparison we allow RR and NAN to tune their parameters knowing the problem setup to maximize their scores. RR tunes its decision threshold τ by gradient ascent. NAN tunes $\mathbf{d}$ and $\mathbf{m}$ by brute force searching all combination and selecting the setting with maximum expected score[9]. We then we set Greedy to return the same number of genes as NAN; for these experiments this was always 3 genes.

Figure 1[A] shows a comparison of the various probe budgets, $M = 2N \dots 7N$. The results shown for RR and NAN are computed analytically, using the previously presented methods[10], while results for Greedy are obtained from Monte Carlo simulation. We see that NAN provides a lower bound for Greedy on these problems as expected.

B. *Synthetic Data Experiment 2*

We repeat Synthetic Data Experiment 1 but begin to make the problem more realistic by dropping the assumption that we know the expression level for the $-$ class is $\mathcal{N}(0,1)$ distributed. Now a probe is defined as observing a draw from both the $+$ and $-$ classes and we update our definition of discriminatory to:

$$\text{discriminatory} \rightarrow X_+ \sim \mathcal{N}(1,1),\ X_- \sim \mathcal{N}(0,1) \quad (15)$$

$$\text{non-discriminatory} \rightarrow X_+, X_- \sim \mathcal{N}(0,1) \quad (16)$$

After collecting the probes, the algorithms must estimate $\hat{\mu}_-$ and the pooled variance such that they can standardize the $-$ class, and then applying the same transformation to the data drawn from the $+$ class they can proceed as previously.

We now repeat the same experimental conditions as in the first experiment, *i.e.*, we use the same parameterizations of the algorithms for each of the conditions. Results are shown in Figure 1[B]. Note that NAN has become a clear winner amongst the algorithms. Greedy is doing terrible. And most surprisingly, RR is getting worse as the probe budget increases.

To explain these behaviours, note that the algorithms are now receiving data from a non-central t-distribution, instead of a normal distribution. For small sample sizes, the difference between these is quite notable; in particular the non-central t-distribution will have a much higher variance and a slightly larger mean. Thus, for the different sample sizes RR keeps setting very poor threshold choices. Note that if the budget is large this will no longer be an issue because the t-distribution approaches the normal distribution as its degrees of freedom increase.

The behaviour of Greedy is most interesting here because it is seemingly invariant to the probe budget, it gets $E[F_1] \approx 0.3$. This is because the increased variance of the non-central t-distribution means its rankings are extremely noisy. In fact a simple calculation shows that we would receive a similar score if we just returned genes at random, without even looking at the data.

$$\begin{aligned} & E[F_1 | \text{return 3 genes at random}] \\ = & \sum_{i=0}^{3} F_1(i, 3-i, 3-i) \frac{\binom{N_d}{i}\binom{N_n}{3-i}}{\binom{N}{3}} \quad (17) \\ = & \ 0.3 \end{aligned}$$

[9]Given the small size of this problem, brute force search is easily executed.

[10]Our empirical results confirm that RR and NAN algorithms perform as predicted.

NAN does surprisingly well because it relies very little on its prior model of the underlying distribution, but instead, only uses the fact that it is highly likely that genes with extreme sample means can be correctly accepted or rejected.

C. *Real Data Experiments*

To show performance on real data we consider three breast cancer datasets downloaded from NCBI-GEO, picked because of they have a common phenotype (ER status $+/-$), relatively large sample sizes and are on Affymetrix Human Genome U133 Plus 2.0 arrays.[11]; see Table I.

TABLE I. DATASETS FOR REAL DATA EXPERIMENTS. AVAILABLE AT HTTP://WWW.NCBI.NLM.NIH.GOV/GEO/

GEO ID	# ER +	# ER −
GSE2034	209	77
GSE3494	213	34
GSE6532	262	45

To set a ground truth for each dataset, we first examine all the data and for each gene we estimate the mean for each class, μ_+ and μ_-, and the pooled variance s. We label the gene as discriminatory or non-discriminatory accordingly.

$$\sqrt{(\mu_+ - \mu_- - 1)^2 + (s-1)^2} \leq 0.25 \rightarrow \text{discriminatory} \quad (18)$$

$$\sqrt{(\mu_+ - \mu_-)^2 + (s-1)^2} \leq 0.25 \rightarrow \text{non-discriminatory} \quad (19)$$

For each dataset we have $10,000$ trials for each algorithm for each probe budget. Each trial begins by randomly drawing a set of $N_d = 3$ and $N_n = 7$ genes from their respective piles. This makes the experiments comparable to the synthetic data, and avoids the issue of picking a specific set of $N = 10$ genes. The algorithms are called with the same parameter settings tuned based on the assumption of normally distributed data. To draw probes for the real data we sample from the patients uniformly at random, without replacement. If an algorithm wishes to probe more samples from a gene than we have patients in the dataset we begin to sample with replacement.

The results in Figure 1[C,D,E] show that the algorithms perform much the same as in synthetic experiment 2. NAN is clearly the best algorithm for finding the discriminatory genes.

IV. CONCLUSION

Future Work: For this work we have tuned the parameters for the NAN algorithm, $\mathbf{d}$ and $\mathbf{m}$, by brute force searching all possible combinations. While this was easily done for the experiments done here, it will not scale up well for larger problems as the search space is combinatorial. There are 2^N possible $\mathbf{d}$ sequences and approximately $\binom{N+B-1}{B}$ unique $\mathbf{m}$ vectors. However, many of these will have similar evaluation scores and a smart search algorithm could tune parameters avoiding uninteresting areas of the parameter space. For example, consider a simple problem with where $m_r > 0$ and $m_{r+1} = 0$, since no data is collected on the $r+1$'th round, accepting on round p and rejecting on $p+1$ is equivalent to rejecting on p and accepting on $p+1$. We are currently

[11]This is currently the favoured brand on the market.

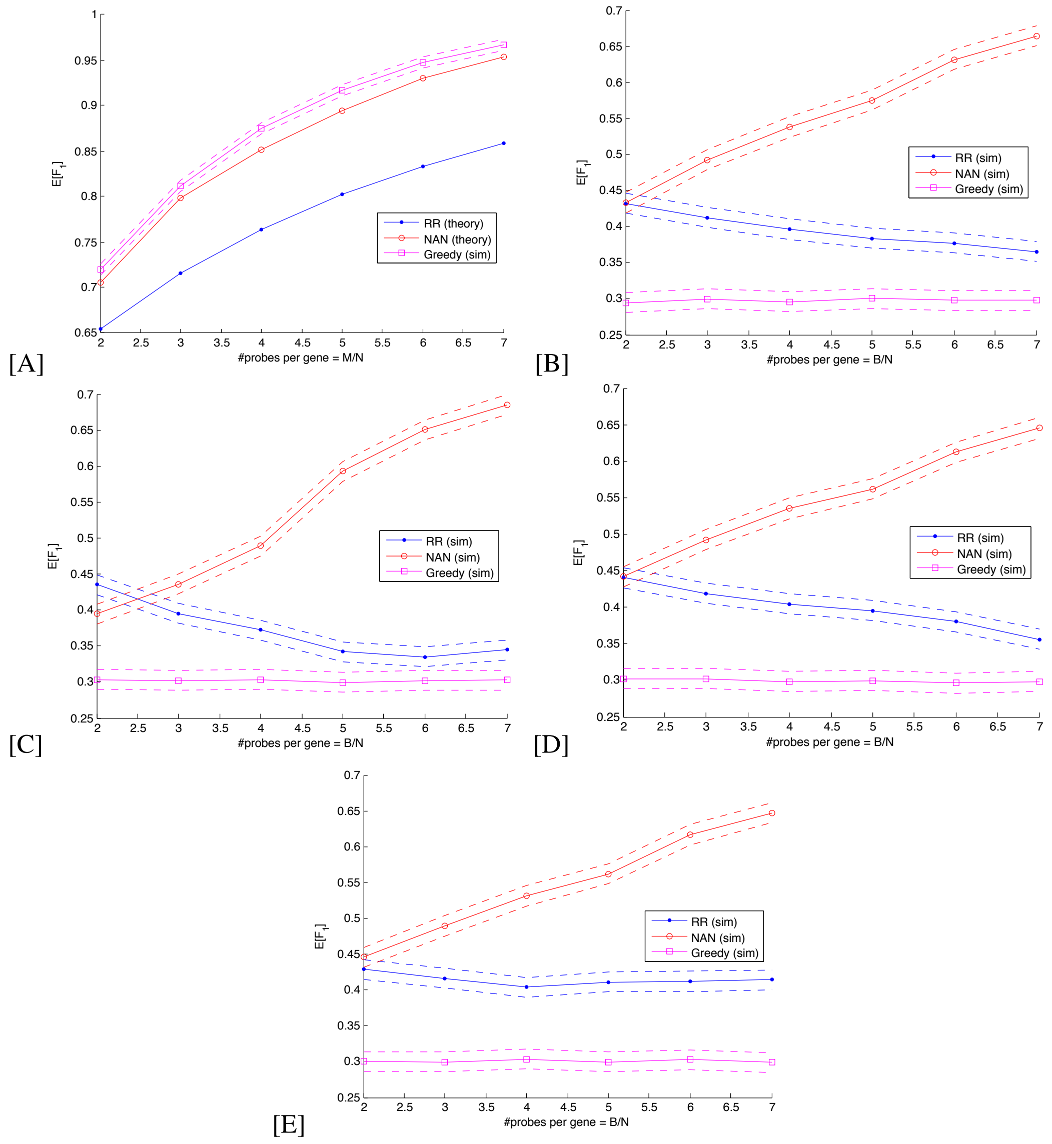

Fig. 1. Experimental results comparing $E[F_1]$ vs. budget per gene B/N for our algorithms: [A] synthetic data experiment 1, [B] synthetic data experiment 2, [C] real data GSE2034, [D] real data GSE3494, [E] real data GSE6532. Dashed lines provide 95% confidence intervals.

investigating this and other effective ways to tune parameters for large problem instances.

While this paper only considered gene expression studies, this framework could apply to other association studies as well such as genome wide association studies (for SNPs or CNVs, etc.) or metabolomic associations for peptides. Also, outside of bioinformatics, the problem of finding discriminatory genes could have analogs in crowd-sourcing where it is desired to quickly identify which workers are sufficiently proficient at the tasks that need to be assigned.

Contributions: We presented the problem of finding discriminatory genes as a potential framework for validating biomarkers from gene expression microarray studies. This is a problem in experimental design where the goal is to spend our resources collecting data to allow us to *efficiently* find a good set of biomarkers. We presented 3 algorithms for this problem: RR, Greedy, and NAN where RR is a naive strategy that ignored the sequential nature of the problem, while NAN will sequentially remove one gene on each iteration, and probe the remaining genes, and Greedy will probe the gene it considers more relevant – the one right on the cusp of being included. If the data collected is normally distributed, we argued analytically that the Greedy will have the best performance but we cannot predict what that performance is. We then showed that NAN provides a good lower bound for Greedy. Furthermore, we also showed that on real data, NAN has superior performance to the other algorithms, as it is least reliant on the modelling assumptions.

Typically, expression studies aim to identify as many genes as possible for a fixed false discovery rate, which does not explicitly consider how many truly discriminatory genes were missed. Competing algorithms from outside of bioinformatics adopt a much harsher objective of finding exactly all of the differentially expressed genes. A nice property of our framework is that it is generic enough to accommodate different evaluation functions for the genes returned. Given this criteria our NAN algorithm can be tuned to maximize this score.

Appendix

The Gumbel Distribution:
The Gumbel distribution is defined by its location and scale parameters α and β. If $x \sim gumbel(\alpha, \beta)$ then:

$$\begin{aligned}
f_{\alpha,\beta}(x) &= \frac{1}{\beta}\exp\left(-\frac{x-\alpha}{\beta} - \exp\left(-\frac{x-\alpha}{\beta}\right)\right) && (20)\\
F_{\alpha,\beta}(x) &= \exp\left(-\exp\left(-\frac{x-\alpha}{\beta}\right)\right) && (21)\\
E[x] &= \alpha + \beta\gamma && (22)\\
mode(x) &= \alpha && (23)\\
median(x) &= \alpha - \beta\log(\log(2)) && (24)\\
var(x) &= \frac{1}{6}(\pi\beta)^2 && (25)\\
entropy(x) &= \log(\beta) + \gamma + 1 && (26)
\end{aligned}$$

$\gamma = 0.577$ is the Euler-Mascheroni constant.

The Logistic Distribution:
If x and y are i.i.d. random variables from a $gumbel(\alpha, \beta)$ distribution then their difference follows a logistic distribution.

$$\begin{aligned}
z &= x - y && (27)\\
f_{\beta}(z) &= \beta^{-1}\exp\left(-\frac{z}{\beta}\right)\left(1+\exp\left(-\frac{z}{\beta}\right)\right)^{-2} && (28)\\
F_{\beta}(z) &= \left(1+\exp\left(-\frac{z}{\beta}\right)\right)^{-1} && (29)
\end{aligned}$$

References

[1] R. Edgar, M. Domrachev, and A. E. Lash, "Gene expression omnibus: NCBI gene expression and hybridization array data repository," *Nucleic Acids Research*, vol. 30, no. 1, pp. 207–210, 2002.

[2] D. Murphy, "Gene expression studies using microarrays: principles, problems, and prospects." *Adv Physiol Educ*, vol. 226, no. 1-2, pp. 256–60, 2002.

[3] J. P. A. Ioannidis, D. B. Allison, C. A. Ball *et al.*, "Repeatability of published microarray gene expression analyses." *Nature Genetics*, vol. 41, no. 2, pp. 149–55, Feb. 2009.

[4] L. Ein-Dor, O. Zuk, and E. Domany, "Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer." *PNAS*, vol. 103, no. 15, pp. 5923–8, Apr. 2006.

[5] A.-L. Boulesteix and M. Slawski, "Stability and aggregation of ranked gene lists." *Briefings in Bioinformatics*, vol. 10, no. 5, pp. 556–68, Sep. 2009.

[6] J. C. Rockett and G. M. Hellmann, "Confirming microarray data–is it really necessary?" *Genomics*, vol. 83, no. 4, pp. 541–9, May 2004.

[7] D. B. Allison, X. Cui, G. P. Page *et al.*, "Microarray data analysis: from disarray to consolidation and consensus." *Nature Reviews. Genetics*, vol. 7, no. 1, pp. 55–65, Jan. 2006.

[8] B. M. Bolstad, F. Collin, K. M. Simpson *et al.*, "Experimental design and low-level analysis of microarray data." *International review of neurobiology*, vol. 60, pp. 25–58, Jan. 2004.

[9] D. C. Hoyle, M. Rattray, R. Jupp *et al.*, "Making sense of microarray data distributions." *Bioinformatics*, vol. 18, no. 4, pp. 576–84, May 2002.

[10] A. Wald, "Sequential Tests of Statistical Hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[11] D. Schuurmans and R. Greiner, "Sequential PAC Learning," in *COLT*, 1995.

[12] E. Even-Dar, S. Mannor, and Y. Mansour, "Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems," *Machine Learning*, 2006.

[13] S. Kalyanakrishnan, A. Tewari, P. Auer *et al.*, "PAC Subset Selection in Stochastic Multi-armed Bandits," *ICML*, 2012.

[14] S. Bubek, T. Wang, and N. Viswanathan, "Multiple Identifications in Multi-Armed Bandits," *ICML*, 2013.

[15] D. Koller and N. Freidman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[16] S. Tong and D. Koller, "Active learning for parameter estimation in Bayesian networks," *NIPS*, 2001.

[17] L. Li, B. Póczos, C. Szepesvári *et al.*, "Budgeted Distribution Learning of Belief Net Parameters," *ICML*, 2010.

[18] M. Jansche, "A Maximum Expected Utility Framework for Binary Sequence Labeling," *ACL*, 2007.