Bioinformatics
doi.10.1093/bioinformatics/xxxxxx

GENVISAGE: Rapid Identification of Discriminative Feature Pairs for Genomic Analysis

Abstract

A common but critical task in biological data analysis is the *separability* task: finding features that explain the difference between two different classes of objects with high dimensional feature representations, such as genes described by their network connectivities, or gene signatures described by their transcriptomic profiles. We develop an interactive data exploration tool called GENVISAGE for this task that rapidly identifies discriminative feature pairs and outputs the corresponding visualizations. Since quickly finding top-k feature pairs is computationally challenging, especially when the numbers of objects and features are large, we propose a suite of optimizations to make GENVISAGE more responsive, and demonstrate that our optimizations lead to a 400X speedup over competitive baselines for multiple biological data sets. With this speedup, GENVISAGE enables the exploration of more datasets and alternative hypotheses in an interactive fashion. Finally, we apply GENVISAGE to uncover pairs of genes whose transcriptomic responses more significantly discriminate treatments of a given drug (e.g., GLRX and NMEZ expression discriminate vorinostat treatment in MCF7 cell lines).

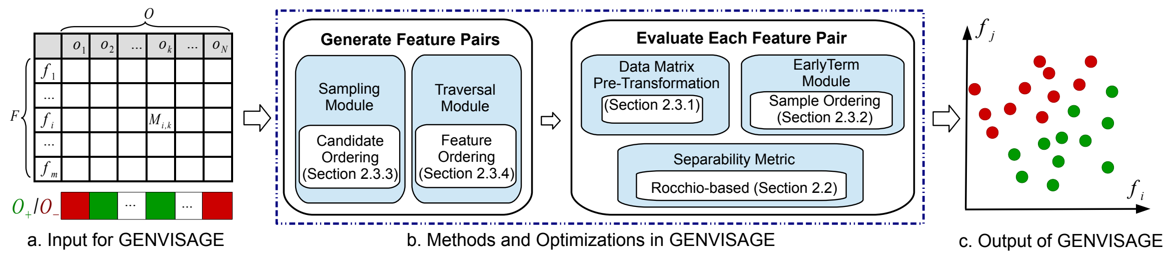
1 Introduction

A common approach to discovery in biology is to construct experiments or analyses that directly contrast two specific classes of biological objects. Examples of this include examining patient samples contrasting tumor versus normal tissue [CITE] , studying the differences in molecular effects of two competing drug treatments [CITE] , or characterizing differentially expressed genes compared to genes with unaltered gene expression in a carefully designed experiment [CITE] . As is often the case with genomics, these biological objects (e.g., tissue samples or drug experiments) are frequently represented by high dimensional numeric feature vectors (e.g., tens of thousands of transcript abundance measurements). To understand the mechanisms that lead to or explain these two object classes, researchers often employ statistical and machine learning tools to identify a manageable subset of features that accentuate or explain the differences between them. We term this the *separability problem*. Often, researchers address this problem by ranking each individual feature using statistical tests [CITE] and univariate classifiers [CITE] . These approaches are fast and scale linearly in the number of features, but do not offer insight into the interplay between important features that can provide a better explanation or characterization of the factors or features that separate the two object classes. Alternatively, larger subsets of separating features can be identified by more complex machine learning approaches, such as multivariate regression with LASSO regularization [CITE] or pattern mining from random forest models [CITE] **Han paper**. However, running these machine learning methods on datasets with tens of thousands of features is extremely time-intensive. Motivated by these observations, we sought to build a tool that strikes a balance between the two above-mentioned “extreme” strategies, enabling us to retrieve more than just the best *singular* features, and doing so in an *interactive* manner.

We present GENVISAGE, an interactive data exploration tool to address the separability problem. Given a matrix of values for object features and two labeled object sets, GENVISAGE rapidly identifies the top ranking pairs of features that most clearly separate the objects of the different classes. We demonstrate that the feature pairs identified by GENVISAGE more significantly discriminate between the object classes than the corresponding best ranking individual features. Thus, by focusing on separating feature pairs, GENVISAGE offers researchers the ability to gain additional insight beyond singular features, without the prolonged waiting time needed to train a complex machine learning model. Moreover, the identified feature pairs can be effortlessly visualized in a two-dimensional interface. However, quickly identifying top feature pairs in typical biological datasets with many features is a challenging task due to some fundamental limitations: first, the number of feature pairs can be large; second, the feature-object matrix must be examined multiple times to evaluate the separability of each feature pair; and third, any precomputation to reduce latency would be useless, given that object sets may be provided on-the-fly. For example, if objects corresponded to experimental conditions, and features were gene expression measurements, having about $20K$ gene features means that we need to evaluate nearly $200M$ feature pairs, and for each feature pair we need to at least scan all of the objects, say $100K$ objects, if using a naive approach. Therefore, sophisticated methods are required to make the task of ranking feature pairs interactive.

In developing GENVISAGE, we identified a measure to score the separability of a given feature pair for two object classes that is conducive to optimization techniques for enabling rapid computation of feature pair rankings, while at the same time providing an intuitive way to interpret the resulting visualizations. Satisfying both of these requirements enables GENVISAGE to offer separability analysis in a responsive exploration environment that promotes the generation and prioritization of hypotheses for further investigation. Specifically, we develop various

© The Author xxx. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com



O
o_1 o_2 … o_k … o_N
F
f_1 … f_i … f_m
$M_{i,k}$
O_+/O_-
a. Input for GENVISAGE
Generate Feature Pairs
Sampling Module
Candidate Ordering (Section 2.3.3)
Traversal Module
Feature Ordering (Section 2.3.4)
Evaluate Each Feature Pair
Data Matrix Pre-Transformation (Section 2.3.1)
EarlyTerm Module
Sample Ordering (Section 2.3.2)
Separability Metric
Rocchio-based (Section 2.2)
b. Methods and Optimizations in GENVISAGE
f_j
f_i
c. Output of GENVISAGE

Fig. 1. Overall GENVISAGE Workflow. Given (a) a feature-object matrix and positive and negative class labels on the objects, where red refers to the negative objects and green refers to the positive objects, GENVISAGE (b) evaluates all pairs of features using several optimizations to identify (c) the top feature pair(s) and its corresponding visualization that best separates the object classes.

novel optimizations to make GENVISAGE more interactive, targeting: *(a)* how to eliminate repeated computations; *(b)* how to prune feature pairs during early execution; *(c)* how to use sampling techniques to further reduce running time but with a statistical guarantee; and *(d)* how to traverse the search space of all feature pairs for improved efficiency. While similar optimizations have been used in other settings, e.g., sampling for approximate computation[CITE] , the techniques need to be adapted for the separability problem.

We applied our GENVISAGE tool to two genomic datasets. In one, called LINCS, we find pairs of genes whose expression discriminates between perturbagen experiments of different drugs and in the other, called MSigDB, we find pairs of annotations (such as pathway membership) that separates differentially expressed cancer genes from all others. Both of these applications are on objects with high-dimensional feature vectors, so it is computationally expensive to score the separability for all possible feature pairs. With the GENVISAGE tool, its carefully designed linear separability metric, and its suite of sophisticated optimizations that alter how to and which feature pairs to evaluate, we are able to ***accurately return the highest ranking separating feature pairs for both datasets within two minutes***. This reflects a ***180X*** and ***400X*** speedup over a competitive baseline for the MSigDB and LINCS data sets respectively. With this speedup, GENVISAGE enables researchers to quickly explore their data, identify the strongest, most compelling features, and form hypotheses about the interplay between them and with the object classes. It also allows researchers to investigate different definitions of the object classes on the fly, interactively, and investigate alternative hypotheses, as well as set up more in-depth, longer machine learning based analysis that builds upon the GENVISAGE results.

We further performed an in-depth analysis for nine drugs in the LINCS dataset and found 1070 feature pairs that more significantly separated the perturbagen experiments than their corresponding single features and were enriched in literature support for known relationships between the genes and the drug, as well as between the pair of genes themselves.

GENVISAGE is available as a web-based tool at `knowcluster03.knoweng.org:9581` [Charles: change] and is open-sourced at `https://github.com/KnowEnG/Genvisage`. [Silu: add screenshot]

2 Methods

We begin by formally defining the *separability* problem, introduce our separability metric, and finally detail several optimizations that enable the rapid identification of the best separating feature pairs.

2.1 Problem Definition

Let $\mathcal{M}$ be a feature-object matrix of size $m \times N$, where each row is a feature and each column is an object as shown in Figure 1(a). An example feature-object matrix is one where each object corresponds to a tissue sample from a cancer patient and each feature correspond to a gene, such that the $(i,j)^{th}$ entry represents the expression level of the i^{th} gene in the j^{th} tissue sample. We denote the m features as $\mathcal{F} = \{f_1, f_2, \cdots, f_m\}$ and N objects as $\mathcal{O} = \{o_1, o_2, \cdots, o_N\}$. Each entry $\mathcal{M}_{i,j}$ in $\mathcal{M}$ corresponds to the value of feature f_i for object o_j as illustrated in Figure 1(a).

In addition, we are given two non-overlapping sets of objects, one with a positive label and the other with a negative label, denoted as $\mathcal{O}_+$ and $\mathcal{O}_-$ respectively. Both sets are disjoint subsets of all objects, i.e., $\mathcal{O}_+, \mathcal{O}_- \subset \mathcal{O}$, and $\mathcal{O}_+ \cap \mathcal{O}_- = \emptyset$. In our example, we may have the tumor samples, $\mathcal{O}_+$, be assigned the positive label, and the healthy tissue samples, $\mathcal{O}_-$, be assigned the negative label. Let $\widehat{\mathcal{O}}$ be the union of positive and negative objects and n be the total number of labeled objects, i.e., $\widehat{\mathcal{O}} = \mathcal{O}_+ \cup \mathcal{O}_-$, $n = |\widehat{\mathcal{O}}|$; $n \leq N$. Also, let l_k be the label of object $o_k \in \widehat{\mathcal{O}}$, i.e., $l_k = 1$ if o_k is positive and $l_k = -1$ if o_k is negative.

The goal of GENVISAGE is as follows: given a matrix $\mathcal{M}$ and two object sets $\mathcal{O}_+$ and $\mathcal{O}_-$ as depicted in Figure 1(a), GENVISAGE aims to find feature pairs that offer the best separation of objects in $\mathcal{O}_+$ from those in $\mathcal{O}_-$ represented by only those two features, and output a visualization that demonstrates the separability as shown in Figure 1(c). (We will define the metric for separability subsequently.) A feature pair that leads to a good “visual” separation between the positive and the negative sets may be able to explain or characterize the differences between the two via a interesting, complex relationship of the features. The overall workflow is depicted in Figure 1. We formally define the separability problem as follows.

Problem 1 (Separability). *Given a feature-object matrix $\mathcal{M}$ and two labeled object sets $(\mathcal{O}_+, \mathcal{O}_-)$, rapidly identify the top-k feature pairs (f_i, f_j) separating $\mathcal{O}_+$ from $\mathcal{O}_-$ based on a given separability metric, and produce their corresponding visualizations.*

We will describe our separability metric in Section 2.2, and then discuss optimization techniques in Section 2.3. Table 1 summarizes our notation.

2.2 Separability Metric

Given a feature pair (f_i, f_j) as axes, we can visualize the object sets $\mathcal{O}_+$ and $\mathcal{O}_-$ in a 2-D space, where each object corresponds to a point with x-value and y-value as the object’s value on feature f_i and f_j respectively. A desirable (i.e., both interesting and interpretable) visualization would be one in which the objects are *linearly separated*, defined as follows. Two sets of objects, i.e., $\mathcal{O}_+$ and $\mathcal{O}_-$, are said to be *linearly separable* if there exists at least one straight line in the 2-D space such that all points from $\mathcal{O}_+$ are on one side of the line while all points from $\mathcal{O}_-$ are on the other side. We focus on metrics that capture this linear separation, since it corresponds to an intuitive 2-D visualization. Given a feature pair (f_i, f_j), we can represent a line ℓ as follows, where x and y represent an object’s value on feature f_i and f_j respectively, and w_0, w_i and w_j are coefficients, where $w_j > 0$.

$$\ell : \quad w_i \cdot x + w_j \cdot y + w_0 = 0 \tag{1}$$

Symb.	Description	Symb.	Description
$\mathcal{M}$	feature-object matrix	$\mathcal{F}$	feature set in $\mathcal{M}$
f_i	feature i in $\mathcal{F}$	m	number of features in $\mathcal{F}$
$\mathcal{O}$	object set in $\mathcal{M}$	N	number of objects in $\mathcal{O}$
$\mathcal{O}_+$	positive object set	$\mathcal{O}_-$	negative object set
$\widehat{\mathcal{O}}$	labelled object set	n	number of labelled objects in $\widehat{\mathcal{O}}$
o_k	object k in $\widehat{\mathcal{O}}$	l_k	label of object o_k
ℓ	separating line in 2-D	L	representative line in 2-D
$\eta_{i,j}^{\ell,k}$	predicted label of o_k	$\theta_{i,j}^{\ell,k}$	o_k is correctly separated?
$\theta_{i,j}^{\ell}$	# correctly separated o_k	$\theta_{i,j}$	separability score
$\widetilde{\mathcal{M}}$	$\mathcal{M}$ after transformation	$\tilde{\theta}_{i,j}$	estimated $\theta_{i,j}$

Table 1. GENVISAGE Method Notation

Given a feature pair (f_i, f_j) and a line ℓ, we can predict the label of an object o_k based on the separating line in Equation 1. Let $\eta_{i,j}^{\ell,k}$ be the predicted label of o_k, calculated using Equation 2 below:

$$\text{Predicted Label}: \quad \eta_{i,j}^{\ell,k} = sign(w_i \cdot \mathcal{M}_{i,k} + w_j \cdot \mathcal{M}_{j,k} + w_0) \qquad (2)$$

If o_k lies above the line ℓ, i.e., o_k has higher value on y-axis than the point on line ℓ with the same value on x-axis as o_k, then $\eta_{i,j}^{\ell,k} = 1$; otherwise, $\eta_{i,j}^{\ell,k} = -1$. Let $\theta_{i,j}^{\ell,k}$ be the indicator variable denoting whether an object o_k is correctly separated (i.e., the sign of the predicted label is the same as the real label l_k):

$$\theta_{i,j}^{\ell,k} = \begin{cases} 1 & if \quad \eta_{i,j}^{\ell,k} \cdot l_k = 1 \\ 0 & otherwise \end{cases} \qquad (3)$$

If there is a line ℓ such that for every object $o_k \in \widehat{\mathcal{O}}$, the object is correctly separated (i.e., $\theta_{i,j}^{\ell,k} = 1$), then we say $\mathcal{O}_+$ and $\mathcal{O}_-$ are linearly separable.

For GENVISAGE, we rank feature pairs by a separability metric based on *how well the objects in the feature pair's 2-D visualization can be linearly separated*, formally defined next. Given a feature pair (f_i, f_j) and a line ℓ, the separability score of the line (denoted $\theta_{i,j}^{\ell}$) is defined as the sum of the indicators ($\theta_{i,j}^{\ell,k}$) for all objects, as shown in Equation 4:

$$\theta_{i,j}^{\ell} = \sum_k \theta_{i,j}^{\ell,k} \qquad (4)$$

Figure 2(a) shows separability scores $\theta_{i,j}^{\ell}$ for different separating lines. For example, the separating line with $\theta_{i,j}^{\ell} = 12$ correctly separates six green points and six red points. The final separability score for a feature pair (f_i, f_j) (denoted as $\theta_{i,j}$) is defined as the best separability score $\theta_{i,j}^{\ell}$ among all possible lines ℓ. Accordingly, we define the overall separability error of the feature pair as $err_{i,j} = n - \theta_{i,j}$.

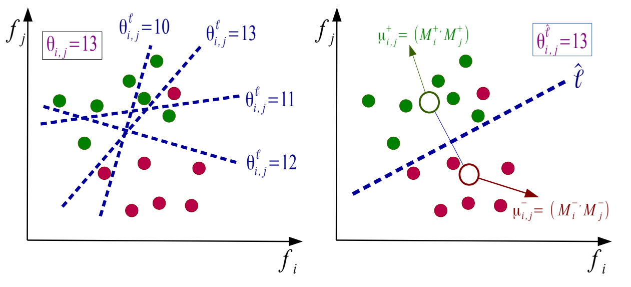
Brute Force Calculation of $\theta_{i,j}$. As suggested in Figure 2(a), the simplest way to calculate $\theta_{i,j}$ is to first enumerate all possible separating lines ℓ and calculate $\theta_{i,j}^{\ell}$ for each of them. This is infeasible as there are an infinite number of possible lines. However, we can easily trim down the search space to $O(n^2)$ lines by linking the points corresponding to every two objects in the 2-D plane. This is because the results of all other possible lines can be covered by these $O(n^2)$ lines [3]. Nevertheless, it is still very time-consuming to consider $O(n^2)$ lines for each feature pair (f_i, f_j).

Rocchio-based Measure. Rather than enumerating through the $O(n^2)$ possible lines to find the one with maximum $\theta_{i,j}^{\ell}$, we propose to speed up the process by intelligently selecting a single *representative line* L that will provide us with an estimated separability score, $\theta_{i,j}^{L}$ that approximates the true linear separability score $\theta_{i,j}$. In order to achieve a fast and reliable estimate, we select our representative line based on Rocchio's algorithm [2]. More specifically, let us denote the centroids of positive objects $\mathcal{O}_+$ and negative objects $\mathcal{O}_-$ for a given (f_i, f_j) as $\mu_{i,j}^{+} = (\mathcal{M}_i^{+}, \mathcal{M}_j^{+})$ and $\mu_{i,j}^{-} = (\mathcal{M}_i^{-}, \mathcal{M}_j^{-})$ respectively, where $\mathcal{M}_i^{+}$ and $\mathcal{M}_j^{+}$ are the values of the centroids of the positive objects on feature f_i and f_j, and $\mathcal{M}_i^{-}$ and $\mathcal{M}_j^{-}$ are the values of the centroids of the negative objects on feature f_i and f_j respectively. The perpendicular bisector of the two centroids is selected as the representative separating line L, following the template of Equation 1 with coefficients as in Equation 5:

$$\begin{aligned} w_i &= \mathcal{M}_i^{+} - \mathcal{M}_i^{-} \\ w_j &= \mathcal{M}_j^{+} - \mathcal{M}_j^{-} \\ w_0 &= -\left(\frac{(\mathcal{M}_i^{+})^2 - (\mathcal{M}_i^{-})^2}{2} + \frac{(\mathcal{M}_j^{+})^2 - (\mathcal{M}_j^{-})^2}{2}\right) \end{aligned} \qquad (5)$$

In Figure 2(b), the "Rocchio-based" measure with the representative separating line $\theta_{i,j}^{L}$ equals 13 with one negative object (red point) mis-predicted as positive, while the true $\theta_{i,j}$ also equals 13.

Brute-force vs. Rocchio-based. Compared to the brute force calcluation, the Rocchio-based measure is much more light-weight in terms of running time, but at the cost of accuracy in calculating $\theta_{i,j}$. Intuitively, the representative line is a reasonable proxy to the best separating line since the Rocchio-based measure computes the centroids of the two classes as their representatives and assigns each object to its nearest centroid. We will further empirically demonstrate that $\theta_{i,j}^{L}$ is a good proxy for $\theta_{i,j}$ in Section 3.2. Thus, we will focus on the Rocchio-based measure in the remainder of the manuscript, using $\theta_{i,j}$ and $\theta_{i,j}^{L}$ interchangeably.



(a) Brute Force (b) Rocchio-Based

Fig. 2. Calculating Separability Score $\theta_{i,j}$. The scored separating line can be defined using (a) brute force or (b) the representative line from a Rocchio-based measure based on the object class centroids (white circles).

2.3 Proposed Suite of Optimizations

In this section, we first analyze the time complexity of identifying the top-k feature pairs using the Rocchio-based measure, and then propose several optimization techniques to reduce the complexity.

Time Complexity Analysis. In order to calculate the separability of a given feature pair (f_i, f_j), if we have already calculated the class centroids for each feature, the separating line L can be calculated in $O(1)$ time using Equation 5. We must then calculate the number of correctly separated objects $\theta_{i,j}^{L}$ by evaluating all objects with respect to the line, i.e., $O(n)$ evaluations. Since there are $O(m^2)$ feature pair candidates, the total time complexity is $O(m^2 n)$, which can be very large, since m and n are typically large.

Optimizations: Overview. To reduce the time complexity, we introduce two categories of optimizations: First, those that reduce the amount of time required for fully evaluating a given feature pair and calculating its Rocchio-based measure (Section 2.3.1, 2.3.2). Second, those that reduce the number of feature pairs that require full evaluation without dropping many feature pairs that are in the top-k (Section 2.3.3, 2.3.4).

The first optimization module within the first category, TRANSFORMATION (Section 2.3.1), reduces redundant calculations across feature pairs by mapping the original feature-object matrix $\mathcal{M}$ into a new space that enables faster evaluation of object labeling, and thus faster calculation of separability score. The second optimization module, EARLYSTOP (Section 2.3.2), takes advantage of the fact that evaluation of a poorly separating feature pair can be terminated early without having to evaluate the separability of all n objects.

The first optimization module within the second category, SAMPLING module (Section 2.3.3) first identifies likely top-k feature pair candidates by evaluating their separability on a sampled subset of all objects, and then conducts full evaluations only on these feature pair candidates—thereby reducing the number of feature pairs that are fully evaluated. Finally, the TRAVERSAL module (Section 2.3.4) reduces the number of feature pairs checked by greedily choosing feature pairs based on the separability of the corresponding single features. These optimization modules can be used on their own or combined with each other.

In Section 3, we will show how these optimizations modules greatly reduce the running time of finding the top-k separating feature pairs without significantly affecting the accuracy.

2.3.1 Pre-Transformation for Faster Feature Pair Evaluation

Let us review the process of computing the separability $\theta^L_{i,j}$. Given a feature pair (f_i, f_j) and the corresponding positive and negative centroids, *(i)* we first compute w_0, w_i and w_j for L based on Equation 5. Next, for each object o_k, *(ii)* we obtain the predicted label $\eta^{L,k}_{i,j}$ according to Equation 2. This step requires two multiplications and three additions. Finally, *(iii)* we calculate $\theta^{L,k}_{i,j}$ and the separability $\theta^L_{i,j}$ based on Equation 3 and 4 respectively. This whole process is repeated for every feature pair candidate. However, there is massive redundancy across the processing of different feature pairs. For instance, when calculating the separability for two different feature pairs (f_i, f_j) and $(f_i, f_{j'})$ with a common f_i, w_i is in fact shared, and calculation of $w_i \cdot \mathcal{M}_{i,k}$ in Equation 2 is repeated for each object o_k. Given this, we propose to first transform $\mathcal{M}_{i,k}$ into another space to reduce this computational redundancy. [Saurabh: is there another work we can cite that has done this?]

High Level Idea. The high level idea for TRANSFORMATION is to pre-calculate some common computational components across different feature pairs, and reuse these components when computing the separability for each feature pair, thereby eliminating repeated computation. In the following, we identify the common computations across different feature pairs, transform $\mathcal{M}_{i,k}$ into another space using those components, and update the separability score equation accordingly.

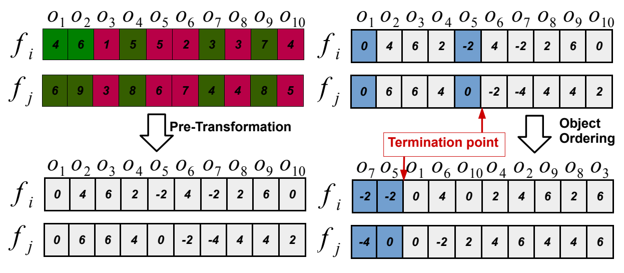
Details. For each feature f_i, we find the average values of the positive and negative objects for that feature, $\mathcal{M}^+_i$ and $\mathcal{M}^-_i$ respectively, and then we pre-transform $\mathcal{M}_{i,k}$, i.e., the value of object o_k on the feature i, to $\widehat{\mathcal{M}}_{i,k}$:

$$\widehat{\mathcal{M}}_{i,k} = \left((\mathcal{M}^+_i - \mathcal{M}^-_i) \cdot \mathcal{M}_{i,k} - \frac{(\mathcal{M}^+_i)^2 - (\mathcal{M}^-_i)^2}{2} \right) \cdot l_k \qquad (6)$$

The basic idea is to decompose Equation 2 into two components, with each one only related to a single, individual feature. This pre-transformation incorporates the class centroids into the matrix values, obviating their integration later for every feature pair that involves the given feature. Equation 6 also multiplies in the class label of the object, l_k, rather than repeating this multiplication every time the object is evaluated. With this transformation of the feature-object matrix, evaluating whether an object was correctly separated is simplified from Equation 3 into Equation 7: [Aditya: confusing because L is not there in Equation 3]

$$\theta^{L,k}_{i,j} = \begin{cases} 1 & \textit{if} \quad sign(\widehat{\mathcal{M}}_{i,k} + \widehat{\mathcal{M}}_{j,k}) = 1 \\ 0 & \textit{otherwise} \end{cases} \qquad (7)$$

After pre-transformation, we are now ready to compute the separability score $\theta^L_{i,j}$. Given a feature pair (f_i, f_j), we can compute $\theta^{L,k}_{i,j}$ for each object o_k based on Equation 7. Note that this step only involves one addition and one comparison. Next, we can calculate overall separability score $\theta^L_{i,j} = \sum_k \theta^{L,k}_{i,j}$ similarly to Equation 4 without the pre-transformation. In all, compared to evaluations without the pre-transformation, we not only eliminate the steps of computing w_0, w_i and w_j for every feature pair, but also reduce the cost of calculating $\eta^{L,k}_{i,j}$ in Equation 2. In the following sections, we will consider $\widehat{\mathcal{M}}$ instead of $\mathcal{M}$, and Equation 7 instead of Equation 2 and 3.


o_1 o_2 o_3 o_4 o_5 o_6 o_7 o_8 o_9 o_{10}
f_i 4 6 1 5 5 2 3 3 7 4
f_j 6 9 3 8 6 7 4 4 8 5
Pre-Transformation
f_i 0 4 6 2 -2 4 -2 2 6 0
f_j 0 6 6 4 0 -2 -4 4 4 2
Termination point
Object Ordering
o_7 o_5 o_1 o_6 o_{10} o_4 o_2 o_9 o_8 o_3
f_i -2 -2 0 4 0 2 4 6 2 6
f_j -4 0 0 -2 2 4 6 4 4 6

(a) Pre-Transformation (b) Early Termination ($\tau = 1$)

Fig. 3. Optimizations for Feature Pair Evaluation. (a) The TRANSFORMATION module is applied once to the original values in the feature-object matrix $\mathcal{M}$ (above) using Equation 6 to produce $\widehat{\mathcal{M}}$ (below), which will reduce later calculations. (b) With the EARLYSTOP module, a feature pair can be discarded from consideration for the top-k before all objects are evaluated (above) with even greater speedup potential after re-ordering the objects (below).

Example 1 (Transformation). *Figure 3(a) illustrates the transformation for two features, f_i and f_j. The top half depicts $\mathcal{M}_{i,k}$ and $\mathcal{M}_{j,k}$ before transformation, where green color represents a positive label and red color represents a negative label. In this example, the centroids of the positive and negative objects are $\mu^+_{i,j} = (5, 7)$ and $\mu^-_{i,j} = (3, 5)$ respectively. Hence, we can rewrite Equation 6 as $\widehat{\mathcal{M}}_{i,k} = (2\mathcal{M}_{i,k} - 8) \cdot l_k$ and $\widehat{\mathcal{M}}_{j,k} = (2\mathcal{M}_{i,k} - 12) \cdot l_k$ for features f_i and f_j respectively. After calculation, we can obtain the values for $\widehat{\mathcal{M}}_{i,k}$ and $\widehat{\mathcal{M}}_{j,k}$ shown in the bottom half of Figure 3(a).*

[Saurabh: at this point, write a summary of the time complexity of calculating $\theta^L_{i,j}$ for a feature pair i,j using transformation (include preprocessing separately)]

[Aditya: suggest delete next para; one line replacement: note that this approach can also be extended to the single feature case.]

Extension to Single Features. For a feature pair (f_i, f_j) where $i = j$, we can treat the corresponding $\theta_{i,i}$ as the separability score for a single feature f_i. Equation 7 can be further reduced to Equation 8.

$$\theta^{L,k}_{i,i} = \begin{cases} 1 & \textit{if} \quad sign(\widehat{\mathcal{M}}_{i,k}) = 1 \\ 0 & \textit{otherwise} \end{cases} \qquad (8)$$

Hence, we can calculate the separability score $\theta_{i,i}$ very efficiently for each single feature, based on the transformed matrix $\widehat{\mathcal{M}}$.

2.3.2 Early Termination

Given a feature pair (f_i, f_j) and line L, we need to make a full scan of all objects to compute the separability score $\theta^L_{i,j}$ according to Equation 4. However, our goal is to not compute the separability score for all feature pairs, instead we only need to identify the top-k feature pairs. Thus, we propose to reduce the running time using early termination when scanning the objects, denoted as the EARLYSTOP module as shown in Figure 1.

High Level Idea. The basic idea is to maintain a upper bound for the separability error $err_{i,j}$ of the top-k feature pairs, denoted as τ. Correspondingly, the lower bound of the separability score can be denoted as $(n - \tau)$, where n is the total number of objects $\hat{\mathcal{O}}$. Given a feature pair (f_i, f_j), we start to scan the object list until the number of incorrectly classified objects exceeds τ. If so, we can terminate early and prune this feature pair since it cannot be among the top-k feature pairs. Otherwise, if the number of incorrectly classified objects never exceeds τ during this scan, then (f_i, f_j) is added to the top-k feature pair set and we update τ accordingly. Although EARLYSTOP has the potential to always reduce the running time, its benefits are sensitive to the ordering of the objects for evaluation. Since we terminate as soon as we find τ incorrectly classified objects, we can improve our running time if we examine "problematic" objects that are unlikely to be correctly classified relatively early.

Enhancement by Object Ordering. To evaluate "problematic" objects earlier, we order the objects in descending order of the number of single features f_i that incorrectly classify the object o_k, i.e., $\widehat{\mathcal{M}}_{i,k} \leq 0$. Thus, the first object evaluated is the one that is incorrectly classified by the most single features. We study the impact of this strategy in Section 3.

Example 2 (EARLYSTOP & Object Ordering). *Figure 3(b) illustrates how early termination and object ordering help reduce the running time. The top half depicts the $\widehat{\mathcal{M}}$ after transformation. Say the current error upper bound is $\tau = 1$. We start scanning the object list from left to right and determine whether each object is correctly separated based on Equation 7. Incorrectly classified objects are marked in blue, such as o_1, since $\widehat{\mathcal{M}}_{i,1} + \widehat{\mathcal{M}}_{j,1} \leq 0$. We can terminate after evaluating o_5 since the number of incorrectly classified objects exceeds $\tau = 1$.*

The bottom half of Figure 3(b) shows $\widehat{\mathcal{M}}$ after object reordering where "problematic" objects are placed on the left. In this way, after checking only the first two objects, we can terminate and eliminate this feature pair from further consideration.

[Charles: note, 0's are treated as mis-labels and ties are broken randomly.]

2.3.3 Sampling-based Estimation

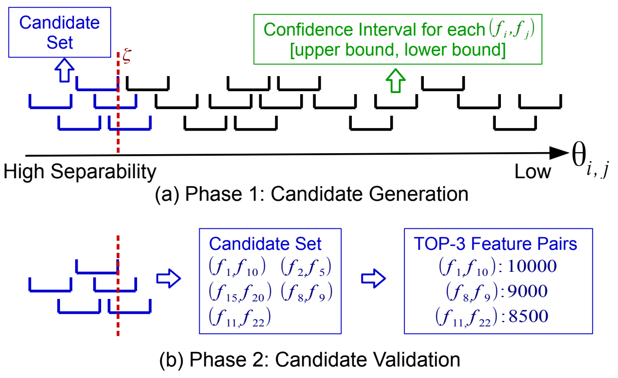
One downside of the EARLYSTOP module, which reduces the number of examined objects, is that the improvement in the running time is highly data-dependent. Here, we propose a stochastic method, called SAMPLING, that further reduces the number of examined objects. Instead of calculating $\theta_{i,j}$ over the whole object set $\hat{\mathcal{O}}$, SAMPLING works on a sample set drawn from $\hat{\mathcal{O}}$. [Saurabh: use $\theta^L_{i,j}$ instead of $\theta_{i,j}$] [Charles: we introduced this simplification at the end of Section 2.2][Aditya: yes, but you used it in 2.3.1.]

High Level Idea. SAMPLING primarily consists of two phases, called *candidate generation* and *candidate validation* as shown in Figure 4. In phase one (Figure 4(a)), we estimate the confidence interval of $\theta_{i,j}$ for each feature pair using a sampled set of objects and generate the candidate feature pairs for full evaluation based on where their confidence intervals lie. If the confidence interval overlaps with the score range of the current top-k , then it is selected for evaluation. In phase two (Figure 4(b)), we evaluate only the feature pairs in the candidate set, calculating $\theta_{i,j}$ over the whole object set, $\hat{\mathcal{O}}$, to obtain the final top-k feature pairs. Unlike our previous optimizations, SAMPLING returns an approximation of the top-k ranking feature pairs. In Section 3.2, we study how the greatly reduced running time of SAMPLING affects the accuracy of the top-k pairs.

Candidate Generation. Let $\mathcal{S}$ be a sample set drawn uniformly from the object set $\hat{\mathcal{O}}$. Given a feature pair (f_i, f_j), let $\theta_{i,j}(\mathcal{S})$ be the number of correctly separated objects in $\mathcal{S}$. Let $\tilde{\theta}_{i,j}$ be the estimated separability score of all objects based on the sample set $\mathcal{S}$. First, we can estimate $\tilde{\theta}_{i,j}$ from $\theta_{i,j}(\mathcal{S})$ using Equation 9 by assuming the ratio of correctly separated samples in $\mathcal{S}$ is the same as that in $\hat{\mathcal{O}}$.

$$\tilde{\theta}_{i,j} = \frac{\theta_{i,j}(\mathcal{S})}{|\mathcal{S}|} \cdot n \tag{9}$$

Next, we show that with a constant sample set size $|\mathcal{S}|$, $|\theta_{i,j} - \tilde{\theta}_{i,j}|$ is bounded by a small value with high probability, using Hoeffding's' inequality [1]. We formally quantize the sample set size in Theorem 1.



Candidate Set
ζ
Confidence Interval for each (f_i,f_j) [upper bound, lower bound]
High Separability
Low
$\theta_{i,j}$
(a) Phase 1: Candidate Generation
Candidate Set
(f_1,f_{10}) (f_2,f_5)
(f_{15},f_{20}) (f_8,f_9)
(f_{11},f_{22})
TOP-3 Feature Pairs
(f_1,f_{10}): 10000
(f_8,f_9): 9000
(f_{11},f_{22}): 8500
(b) Phase 2: Candidate Validation

Fig. 4. Phases of SAMPLING. (a) To calculate the top-3 feature pairs, the confidence interval of $\theta_{i,j}$ is calculated for all feature pairs evaluated on the sample set $\mathcal{S}$ and a subset of candidates (blue intervals) are selected for validation. (b) The five selected candidates (center box) are evaluated on the whole object set $\hat{\mathcal{O}}$ to compute the exact $\theta_{i,j}$ and pick the top-3 (rightmost box).

Theorem 1 (Estimation Accuracy). *By considering $\Omega(\frac{1}{\epsilon^2} \cdot \log(\frac{1}{\delta}))$ samples, with probability at least $1 - \delta$, we have $|\frac{\theta_{i,j}(\mathcal{S})}{|\mathcal{S}|} - \frac{\theta_{i,j}}{n}| \leq \epsilon$, i.e., $|\tilde{\theta}_{i,j} - \theta_{i,j}| \leq \epsilon n$.*

Proof. Based on Hoeffding's inequality:

$$Pr(|\frac{\theta_{i,j}(\mathcal{S})}{|\mathcal{S}|} - \frac{\theta_{i,j}}{n}| \geq \epsilon) \leq 2e^{-2|\mathcal{S}|\epsilon^2}$$

Hence, when $|\mathcal{S}| = \frac{1}{2\epsilon^2} \cdot \log(\frac{2}{\delta}) = \Omega(\frac{1}{\epsilon^2} \cdot \log(\frac{1}{\delta}))$, we have:

$$Pr(|\frac{\theta_{i,j}(\mathcal{S})}{|\mathcal{S}|} - \frac{\theta_{i,j}}{n}| \geq \epsilon) \leq \delta \qquad \square$$

We can treat $\log(1/\delta)$ as a constant, e.g., by setting $\delta = 0.05$. Thus, Theorem 1 essentially states that with only $\Omega(\frac{1}{\epsilon^2})$ samples, with probability 95%, the confidence interval for $\theta_{i,j}$ is $[\tilde{\theta}_{i,j} - \epsilon n, \tilde{\theta}_{i,j} + \epsilon n]$. Note that the sample size $|\mathcal{S}|$ only depends on ϵ and is independent of the number of objects, n. Hence, the SAMPLING module helps GENVISAGE scale to datasets with large n.

With Theorem 1, we can first calculate the confidence interval of $\theta_{i,j}$ for each feature pair (f_i, f_j), as shown in Figure 4(a). Next, we compute the lower bound of $\theta_{i,j}$ for the top-k feature pairs, denoted as ζ as shown by the red dotted line in Figure 4(a). Finally, we can prune feature pairs away whose upper bound is smaller than ζ. Accordingly, the candidate set $\mathcal{C}$ corresponds to the feature pairs depicted by the blue intervals in Figure 4(a). These feature pairs $\mathcal{C}$ will be further validated in phase two, i.e., candidate validation. Typically, $|\mathcal{C}|$ will be orders of magnitude smaller than m^2, the original search space for all feature pairs.

Example 3 (Candidate Generation). *We illustrate phase one using Figure 4(a). Set $\epsilon = 0.05$ and $K = 3$. We sample a subset of all objects and compute the confidence interval of $\theta_{i,j}$ for each (f_i, f_j). Next, we obtain the 3^{rd} lower bound ζ (red dotted line) in descending order among all the confidence intervals. Finally, we designate all feature pairs whose upper bound is larger than the red dotted line as blue feature pair candidates $\mathcal{C}$.*

Candidate Validation. Since we only estimated the separability scores $\theta_{i,j}$ for our feature pair candidates in phase one, we re-evaluate all of the candidates generated from phase one to produce our final feature pair ranking. This phase two evaluation is performed using the whole object set $\hat{\mathcal{O}}$. [Saurabh: are we using early termination and object ordering in phase two?] [Silu: no. We tried both and found that early termination is not helping most of time since 1) the feature pair candidates, which perform well in phase 1, are indeed the ones that are hard to prune and 2)

EARLYSTOP incurs some overhead itself.] For each feature pair $(f_i, f_j) \in \mathcal{C}$, we calculate the separability score $\theta_{i,j}$ and find the top-k feature pairs (Figure 4(b)).

Enhancement by Candidate Ordering. Recall that in Section 2.3.2 we proposed an enhancement that allows us to terminate computation by cleverly manipulating the order of the objects; here we similarly found a way to reduce the running time by changing the order in which feature pair candidates are validated in phase two. Instead of directly validating each feature pair candidate, we first order the candidates in descending order according to the upper bound of each candidate's confidence interval. Then, we sequentially calculate the full separability score $\theta_{i,j}$ for each feature pair, and update ζ correspondingly. Recall that ζ is the current estimate of the lower bound of $\theta_{i,j}$ for the top-k feature pairs. Finally, we terminate our feature pair validation when the next feature pair's upper bound smaller

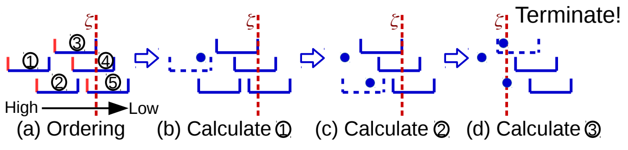


Fig. 5. Enhancement by Candidate Ordering. (a) Feature pair candidates are sorted by the upper bounds of their confidence intervals (solid red boundary), and the lower bound of the top-3 feature pairs, i.e., ζ, is set (red dotted line). (b,c,d) For each feature pair, we calculate $\theta_{i,j}$ (filled blue circle) using all objects and update ζ if necessary. Note that ζ is increased in (d) after evaluating the third feature pair and since ζ is larger than the upper bound of the fourth feature pair, candidate validation phase can terminate and return the top ranking feature pairs.

[X SS: example descriptions moved to figure legends] [Saurabh: some readers will be confused by higher values being on the left]

2.3.4 Search Space Traversal

The optimizations discussed so far check fewer than n objects for each feature pair and reduce the number of feature pairs for full evaluation in order to reduce the overall running time. Our TRAVERSAL module aims to reduce the number of feature pairs considered from m^2 to a smaller number.

High Level Idea. Instead of examining each feature pair, we only examine a limited number of feature pairs, but in an optimized traversal order. The number of examined feature pairs determines a trade-off between efficiency and accuracy. In general, fewer feature pairs checked will result in faster running times, though at the cost of accuracy in terms of the output. The full search space for the feature pairs is the upper triangle in Figure 6(a), where each row represents f_i and each column represents f_j. Let χ be the limit on the number of feature pairs to be examined. We propose two different traversal orders over the search space: horizontal and vertical, as follows:

•*Horizontal traversal (Figure 6(a)):* for each feature f_i, pair it with each other feature f_j, where $j \geq i$, to obtain (f_i, f_j). Repeat for each f_i, where $1 \leq i \leq m$.

•*Vertical traversal (Figure 6(b)):* for each feature f_j, pair it with each other feature f_i, where $i \leq j$, to obtain (f_i, f_j). Repeat for each f_j, where $1 \leq j \leq m$.

However, the traversal order does not convey any intuition if the feature set $\mathcal{F}$ is in random order. Hence, we further propose to enhance TRAVERSAL by feature ordering in $\mathcal{F}$.

Enhancement by Feature Ordering. We propose to first sort features based on single features' separability scores $\theta_{i,i}$. Single features with good separability individually are ranked higher. Let $\{f'_1 f'_2, \cdots, f'_m\}$ be the sorted single-feature set. Next, we discuss the intuition behind the two different traversal order schemes. Consider partitioning the sorted single-feature list into three categories based on the single-feature separability score $\theta_{i,i}$: *good, average,* and *poor.*

•*Horizontal traversal:* the intuition is that there is at least one *good* or *average* single feature in each top-k feature pair (f_i, f_j).

•*Vertical traversal:* the intuition is that both features in each top-k feature pair (f_i, f_j) must be categorized as *good* or *average* individually.

We examine the impact of these traversal schemes in our evaluations.

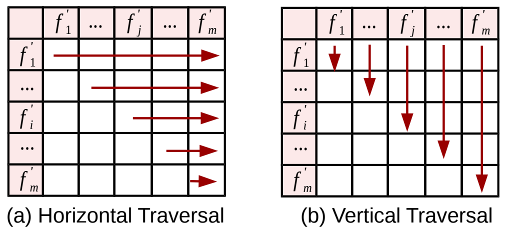


Fig. 6. TRAVERSAL Orderings. GENVISAGE is able to examine the feature pair search space (a) horizontally or (b) vertically.

Example 4 (TRAVERSAL). *Assume $m = 2 \times 10^4$. Initially, the number of possible feature pairs is roughly $\frac{m^2}{2} = 2 \times 10^8$. However, if we limit the number of considered feature pairs to $\chi = 10^7$, we reduce our search space to $\frac{1}{20}$ of the total number of feature pairs. We order the single features by their individual separability scores. In horizontal traversal, only feature pairs with at least one individual feature ranked in the top 500 will be considered; while vertical traversal will consider only feature pairs with both individual features ranked better than 2000.*

3 Results

In this section, we will illustrate that GENVISAGE can rapidly identify meaningful feature pairs for the separability problem in real biological datasets to provide significant and interesting insights. First, we describe the datasets and algorithms used in our evaluations. Each algorithm that we evaluate represents a combination of optimization modules for ranking top-k feature pairs using our Rocchio-based separability measure. We report the running time and accuracy of the selected algorithms. Finally, we enumerate and visualize the top-k feature pairs returned by GENVISAGE, compare them to the corresponding top-k single features, and examine their significance and support in existing publications.

3.1 Evaluation Setup

	$\lvert\mathcal{F}\rvert$=m	$\lvert\mathcal{O}\rvert$=N	$\lvert\mathcal{S}\rvert$	χ	# of $\widehat{\mathcal{O}}$	avg($\lvert\mathcal{O}_+\rvert$)	avg($\lvert\mathcal{O}_-\rvert$)
MSigDB	19,912	22,209	400	10^7	10	295	21,914
LINCS	22,268	98,061	400	10^7	40	165	97,897

Table 2. Dataset Statistics. For each dataset, the number of features m, objects N, sample size $|\mathcal{S}|$ used by SAMPLING module, feature pairs χ examined by TRAVERSAL module, number of object sets: # of $\widehat{\mathcal{O}}$, average positive set size: avg($|\mathcal{O}_+|$), and average negative set size: avg($|\mathcal{O}_-|$).

Datasets. We conducted our tests on two different biological applications. In the first application, which we call MSigDB, we find pathways and other gene annotations that separate the differentially expressed genes from the undisturbed genes in specific cancer studies. In the other application, called LINCS, we find genes whose expression levels can distinguish experiments in which specific drug treatments were administered from others.

More specifically, in the MSigDB application, we are given a feature-object matrix with genes as the objects and different gene properties as the features. A natural way to construct this matrix is to define a collection of properties such as Gene Ontology terms, PFam domains, Pathways [CITE] each of these sources, etc., and use a binary representation to indicate whether each gene is known to have a property. This is equivalent to constructing a network with gene and property nodes, with edges

	Transformation	EARLYSTOP	Object Ordering	SAMPLING	Candidate Ordering	TRAVERSAL	Feature Ordering	Complexity
Baseline	yes	no	no	no	no	Any	no	$O(m^2n)$
EarlyOrdering	yes	yes	yes	no	no	Any	no	$O(m^2n)$
SampOnly	yes	no	no	yes	no	Any	no	$O(mn + m^2\|\mathcal{S}\| + \|\mathcal{C}\|n)$
SampOpt	yes	no	no	yes	yes	Any	no	$O(mn + m^2\|\mathcal{S}\| + \|\mathcal{C}\|n)$
HorizSampOpt	yes	no	no	yes	yes	Horizontal	yes	$O(mn + \chi\|\mathcal{S}\| + \|\mathcal{C}\|n)$
VertSampOpt	yes	no	no	yes	yes	Vertical	yes	$O(mn + \chi\|\mathcal{S}\| + \|\mathcal{C}\|n)$

Table 3. Selected Algorithms Using Different Optimization Modules. For each algorithm (row), shows which optimization modules are employed and the running time complexity for that combination where m and n are the number of features and objects, $\mathcal{S}$ is the sampled set size, χ is the limit on the number of feature pairs considered, and $\mathcal{C}$ is the number of generated feature pair candidates.

representing known annotations, and recording adjacency information on each gene as its feature values. We generalized this notion of gene-property relationships by adding homology edges (connecting homologous gene pairs based on BLAST scores) to the above network, and used the DRaWR algorithm **[CITE] DRAWR** to perform a random walk with restarts on the heterogeneous network, thereby scoring the connectivity of each property node to each gene node. This resulted in the assignment of a numeric value for each gene-property pair (g, r), representing not only whether the gene is annotated with that property but also whether other genes closely related to gene g are annotated with property r. We thus obtained a feature-object matrix describing each gene (object) as a vector of its strength of association with each property (feature) in light of prior biological knowledge. [Charles: still need to be certain on the steps required to create this matrix, also may want to mention restart prob, etc] The positive genes for each dataset in the MSigDB collection are the set of differentially expressed genes (DEGs) in a specific cancer study, as per Molecular Signatures Database **[CITE] MSIGDB**. Each of our tests is an application of GENVISAGE to such a dataset, reporting pairs of properties that separate DEGs of that cancer study (the “positive” set) from all other genes (the “negative” set).

In the LINCS application, the feature-object matrix contains expression values for different genes (features) across many drug treatment experiments (objects) conducted on the MCF7 cell line by the LINCS L1000 project **[CITE] lincs**. The values of the matrix are gene expression quantities as reported by the “level-4’ imputed z-scores measured in the L1000 project. In each dataset, the positive object set includes multiple experiments that used the same drug, at varying dosages and for varying durations. We applied GENVISAGE on each dataset so as to find the top pairs of genes (feature pairs) whose expression values separates the LINCS experiments relating to a single drug from all other LINCS experiments.

See Table 2 for further details of datasets used in the two applications just introduced. Note that the average number of positive objects in a dataset is far fewer than the average number of negative objects. In order to address this imbalance, we adjust Equation 4 to a weighted sum form as shown in Equation 10.
[Saurabh: why did we chose the number of object sets, 10 and 40]

$$\theta^{\ell}_{i,j} = \sum_{o_k \in \mathcal{O}_-} \theta^{\ell,k}_{i,j} + \frac{|\mathcal{O}_-|}{|\mathcal{O}_+|} \cdot \sum_{o_k \in \mathcal{O}_+} \theta^{\ell,k}_{i,j} \qquad (10)$$

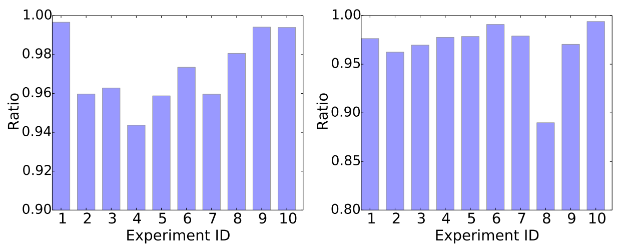
[Saurabh: maybe naming algorithms A0, A1-A5.] [Saurabh: describe table as tree?] [Aditya: add approximate column]

Algorithms. In our evaluation, we experimented with six different selected combinations of the optimization modules introduced in Section 2.3, as listed in Table 3. Other combinations that were not selected were always inferior to the ones shown, e.g., the unlisted algorithm with only TRANSFORMATION and EARLYSTOP modules always has a worse running time and outputs the top-k feature pairs with the same separability score as the selected EarlyOrdering combination. For our baseline, [Aditya: if we are using this for our baseline than I propose we discard its description if we are lacking in space...let’s discuss] we use the algorithm with only the feature-object matrix pre-transformation optimization module (TRANSFORMATION). The last column in Table 3 shows the running time complexity of different algorithms. Take HorizSampOpt as an example on how the running times are calculated. First, the TRANSFORMATION module takes $O(mn)$ time. [Charles: note, scoring single features takes O(mn) time and uses only transformation optimization.] Then, the TRAVERSAL module requires a sorting over the feature set, taking $O(m \log m)$[1] time. Finally, using the SAMPLING module over χ feature pairs, the running time is reduced from $O(m^2n)$ time to $O(\chi|\mathcal{S}| + |\mathcal{C}|n)$ time, where the first and second term represent the time for candidate generation and candidate validation respectively. Note that $|\mathcal{C}|$ is typically orders of magnitude smaller than χ in HorizSampOpt, as discussed in Section 2.3.3.

Evaluation Setup. We implemented the algorithms in C++, and conduct the evaluations on a machine with 16 CPUs and 61.9 GB of RAM.

3.2 Comparison of Different Algorithms

In this section, we first justify that Rocchio-based measure is a good proxy for the best possible separating score, i.e., $\theta_{i,j}$, computed by a brute force method Then we compare the performance of the various selected algorithms in terms of the running time and the separability scores of their top-1000 reported feature pairs.



(a) Best Feature Pair Comparison (b) Measure Comparison

Fig. 7. Comparison of Brute Force-based and Rocchio-based separability score. (a) For each of 10 datasets, we display the ratio of the true separability score between the best feature pair chosen by brute force and by the Rocchio-based method. (b) For each dataset, we display the ratio of the true separability score and the Rocchio-based separability score for the best feature pair selected using Rocchio-based method.

[1] Since $m \log m < nm$, we omit term $m \log m$ from the overall complexity.

Brute Force vs. Rocchio-based. As we have discussed in Section 2.2, when using brute force, we need to consider $O(n^2)$ lines in order to find the best separating line $\ell^* \leftarrow \arg_\ell \max\{\theta^\ell_{i,j}\}$, leading to a total running time complexity $O(n^2m^2)$ when considering all feature pairs. An alternative is to use Rocchio-based representative separating line L, dramatically reducing $O(n^2)$ lines considered to $O(1)$. Since the brute force method becomes computationally infeasible for datasets with large n, we compared the Rocchio-based measure to the brute force-based measure using the 10 datasets in MSigDB, for which the average number of objects, i.e., $|\widehat{\mathcal{O}}|$, is 295. [Saurabh: Is the notation correct? Isnt 295 the average number of *positive* objects?] We call the brute force-based separability score the *true* separability score, since it examines all possible separating lines. We first find the best feature pair using Rocchio-based measure and the brute force based measure separately, denoted as FP_{roc} and FP_{bf} respectively, then calculate the ratio of the true separability score of FP_{roc} vs. the true separability score of FP_{bf} (Figure 7(a)). [Saurabh: Are these notations used below? They're not used in this paragraph.] We observe that the Rocchio-based method picks a best feature pair that has true separability score similar to the best pair picked by brute force, with the ratio of the two scores being better than 0.94 in all ten datasets. Second, for the best feature pair identified by Rocchio-based method, we calculate the ratio of the brute force-based separability score and Rocchio-based separability score (Figure 7(b)), and find the difference to be small. [Aditya: odd to have this here when this section is mostly about algo comparison. maybe bring it up later?]

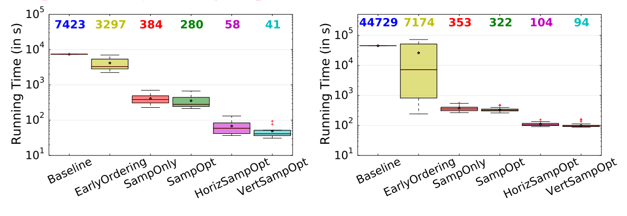


Fig. 8. Running Time Comparison. A boxplot for each selected algorithm is shown with the median value appearing in matching color above. For each boxplot, whiskers are set to be $1.5\times$ the interquartile range, the outliers are shows as red dots, and the average is marked with as a black star. The number on the top shows the median running time for each algorithm.

Running Time. Figure 8 depicts the running times of our different selected algorithms, as stated in evaluation setup. [X SS: what are the specs of the machine for the tests]. Each plotted box corresponds to one algorithm, representing the distribution of running times for finding the top-k feature pairs (by Rocchio score) for all datasets.

First, let us compare the median running times among different algorithms. For MSigDB, the Baseline takes more than 2 hours, EarlyOrdering takes less than 1 hour, SampOnly and SampOpt take around 6 and 5 minutes respectively, while HorizSampOpt and VertSampOpt both take only 1 minute on average. Overall, the optimizations result in a reduction of the running time by over $180\times$. We next examine the effect of different modules on the running time. *(a)* EARLYSTOP*:* we observe that the EARLYSTOP module helps achieve a $2\times$ speed up, with the average number of checked objects (genes) reduced from $20K$ to $5K$ (Supplementary Table 5); *(b)* SAMPLING*:* the SAMPLING module helps reduce the running time dramatically, with $20\times$ reduction from Baseline to SampOpt, since on average only $2M$ candidates are generated from all possible $200M$ feature pairs (Supplementary Table 5); *(c)* TRAVERSAL*:* by considering approximately $\frac{1}{20}$ of all possible feature pairs, [Saurabh: why 1/20? was it explained before? if so, no change needed here.] modules HorizSampOpt and VertSampOpt achieve an additional $6\times$ speed-up compared to SampOpt. This speedup of HorizSampOpt and VertSampOpt is limited by the feature ordering overhead (around $6s$) and the time for the TRANSFORMATION module (around $8s$). Also, during the candidate generation phase, SampOpt is able to prune a greater percentage of feature pairs than HorizSampOpt and VertSampOpt, since on average the feature pairs considered in HorizSampOpt and VertSampOpt have higher separability scores than those in SampOpt. [Saurabh: previous sentence is not clear. Even on second reading.]

Next, let us examine the log-scale interquartile range (IQR) of the running times for the different selected algorithms (Figure 8). We observe that EarlyOrdering has the largest interquartile range, indicating that the EARLYSTOP module used by this algorithm, which tries to reduce the number of objects evaluated for each feature pair, is very dependent on the object set and feature values. As we discussed in Section 2.3.2, EARLYSTOP has no guarantee on improving the running time. In fact, the algorithm can occasionally be worse than the Baseline as shown in Figure 8(b) because EARLYSTOP incurs additional overhead for checking the criteria for pruning and early termination when scanning the object list for each feature pair. [Charles: we need more clarity (maybe in the discussion) why the early termination is included if it is not performing well, i.e. cases where we would recommend using it]

Similar results can be found in Figure 8(b) for the LINCS application. Here, we observe over $400\times$ average decrease in the running time of finding the top-k feature pairs that separate the LINCS experiments of a single perturbagen from others. The greatest speedup comes with adding the SAMPLING module. In this setting, only $100K$ feature pair candidates, i.e., $|\mathcal{C}|$, are checked out of all $250M$ feature pairs (Supplementary Table 5). For the selected algorithms with best running times, HorizSampOpt and VertSampOpt, the pre-transformation and feature ordering overhead account for an average of $45 + 35 = 80s$ of the overall 104 and 94 median seconds respectively.

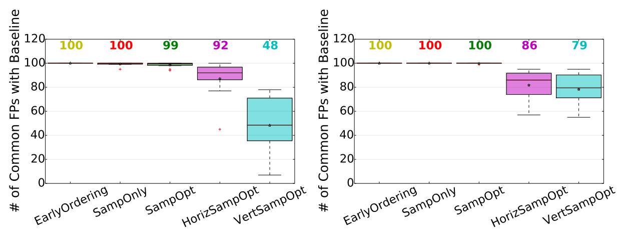


Fig. 9. Separability Quality Comparison. Boxplots in the style of Figure 8 comparing the number of feature pairs each method returned from the 100 best feature pairs of the Baseline method.

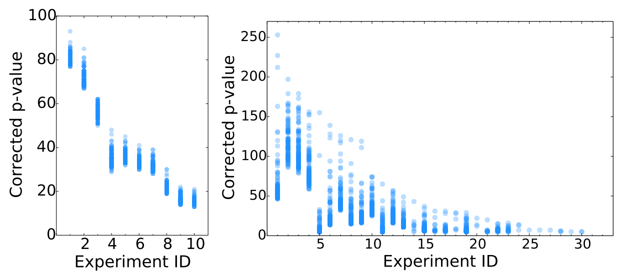
[Saurabh: alignment of algorithm names not ok in figure]

Separability Quality. Recall that the EARLYSTOP module is a deterministic optimization and will produce the same top-k feature pairs as the baseline method. The SAMPLING module, on the other hand, is stochastic and places a probabilistic guarantee on the separability score of the feature pairs and therefore can only provide an approximate estimate of the top-k feature pair ranking. Finally, the TRAVERSAL module is heuristic and may output top-k feature pairs that are very different from the ranking produced by the Baseline algorithm. Since Baseline returns the true Rocchio-based separability score of each feature pair, we measured the quality of each selected algorithm by counting the number of common feature pairs returned in the top-100 between the Baseline and the given algorithm. Figure 9 shows this separability quality comparison. Let us first focus on the MSigDB application. EarlyOrdering, as expected, has exactly the same separability quality as the Baseline. We also observe that the SampOnly and SampOpt rankings are nearly identical to the top-100 feature pairs of the Baseline, owing to the probabilistic guarantee as stated in Theorem 1. The HorizSampOpt and VertSampOpt algorithms output a median of 92 and 48 feature pairs in common with Baseline, respectively, because of the heuristic TRAVERSAL module. In the MSigDB results, HorizSampOpt performs much better than VertSampOpt, with the median much higher and the interquartile range much narrower, as shown in Figure 9(a). This suggests, as we hypothesized, that interesting separating feature pairs exist outside of only the combinations of the top single features as in VertSampOpt. We repeated this quality analysis for the LINCS application and found that the SAMPLING based algorithms returned identical top-100 feature pairs for all 40 datasets. The quality of the TRAVERSAL based algorithms was again lower, though there the separation in the performance of the HorizSampOpt and the VertSampOpt algorithms was not as large as for the MSigDB application.

Takeaways. If the accuracy is paramount to the user, SampOpt is the suggested algorithm; while if the running time is paramount to the user, HorizSampOpt is the desired algorithm.

3.3 Feature Pair vs. Single Feature.

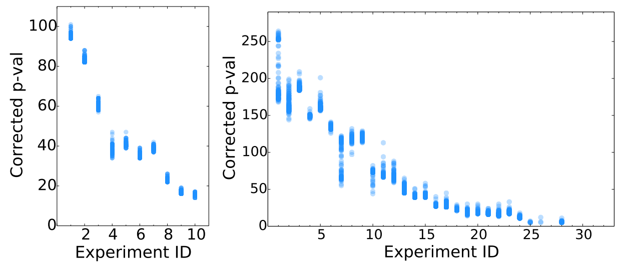
In this section, we quantify how statistically significant the top ranking results of the selected algorithms are. We show that we are often able to find separating feature pairs that are more significant than the best single separating feature. In order to assess the significance of a separating feature or feature pair, we first calculate the p-value using the one-sided Fisher's exact test on a 2×2 contingency table. This contingency table is constructed with the rows being the true positive and negative labels, the columns being the predicted positive and negative labels, and the values being the number of objects that belong to each table cell. Using the Fisher's exact test p-value, we assert that feature pairs can provide better separability performance or new insights compared to single features, i.e., (a) feature pairs have improved p-values compared to the corresponding individual features even after the appropriate multiple hypothesis correction and (b) there exist high-ranked pairs of features that are poorly-ranked on their own as single features.



(a) MSigDB (b) LINCS

Fig. 10. Single Feature Corrected P-value Distribution [Saurabh: Experiment ID -> Dataset ID]. For each experiment (x-axis), shows the significance ($-\log_{10}(corrected\_pval)$) of the top-100 best single features (blue dots) for the (a) MSigDB and (b) LINCS datasets. We order the datasets by their best corrected single feature p-value, and discard the datasets where no single feature has corrected p-value better than 10^{-5}.

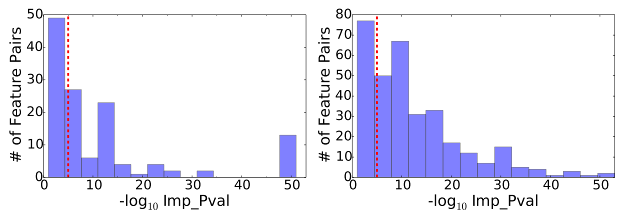
Single Feature. Finding top-k single features is a special case of finding feature pairs by setting $i = j$, as noted in Section 2.3.1 and Equation 8. For each single feature obtained, we compute the p-value with Fisher's exact test, denoted as $pval$. Next, we define the Bonferroni corrected p-value as $corrected\_pval = pval \times m \times n$, since there are $m \times n$ possible hypotheses, one for each possible single feature and separating line. We say a selected feature is significant if the corrected p-value is smaller than the threshold 10^{-5}, i.e., $-\log_{10}(corrected\_pval) \geq 5$. In Figure 10, we plot the distribution of the corrected p-value of the top-100 features reported for each dataset in MSigDB and LINCS. We observe that 10 out of 10 datasets in MSigDB and 32 out of 40 datasets in LINCS have at least one significant single feature. We will focus on these 10 runs in MSigDB and 32 datasets in LINCS for the remainder of the paper. We observe very small p-values, $\leq 10^{-50}$, in the left part of Figure 10(a) and 10(b), indicating a very strong relationship between the predicted and true object labels. This shows the effectiveness of linear separability and the Rocchio-based separability metric in identifying features that predict the different classes of objects.



(a) MSigDB (b) LINCS

Fig. 11. Feature Pairs' Corrected P-value Distribution. Plotted significance for the top-100 best feature pairs (blue dots) for each experiment in the (a) MSigDB and (b) LINCS datasets.

Feature Pair. We next build the contingency tables and calculate the p-value for the top-k feature pairs. To correct for m^2 possible feature pairs and the n^2 possible ways to choose the separating lines for each feature pair, we apply a Bonferroni p-value correction to produce the $corrected\_pval = pval \times m^2 \times n^2$. We plot the distribution of these corrected p-value for the top-k feature pairs in Figure 11. Once again, the threshold for defining a significant feature pair is set to 10^{-5}. We find that 10 out of 10 datasets in MSigDB and 27 out of 40 datasets in LINCS have at least one significant feature pair by this metric. Visual comparison of the top-100 single features (Figure 10) to the top-100 feature pairs (Figure 11) per dataset reveals several datasets where the corrected p-values of the feature pairs are more significant than those of the best single features, even after accounting for the larger search space. Admittedly, this is not always the case, e.g., for five LINCS datasets no feature pair was found to be significant at $corrected\_pval \leq 10^{-5}$ while at least one single feature did meet this threshold. Overall, this analysis suggests that rapid discovery of top feature pairs may identify more significant patterns in the given dataset than a traditional single-feature analysis does. In the following, we further illustrate that feature pairs can also provide better and new insights compared to single features.



(a) MSigDB (b) LINCS

Fig. 12. Histogram of $improv\_pval$. For the top-20 feature pairs from all runs from the (a) MSigDB and (b) LINCS datasets, distribution of the improvement of the feature pair significance over the corresponding single feature significance. The red line shows the significance threshold of 5.

Improvement from Single Feature to Feature Pair. We have computed the corrected p-value for each single feature and feature pair. Now let us examine the improvement of each feature pair from its two corresponding single features in terms of p-value. For each feature pair (f_i, f_j), we define the improved p-value [Saurabh: can we call this something else? It is grammatically incorrect. Not sure what though.] as the ratio between the corrected p-value of (f_i, f_j) and the better of the corrected p-value of f_i or f_j, i.e., $improv = \frac{corrected\_pval(f_i, f_j)}{\min(corrected\_pval(f_i), corrected\_pval(f_j))}$. In this set of results, we will focus on top-20 feature pairs for each dataset to improve the clarity of the presentation. In Figure 12, we plot the histogram of $improv\_pval$ for 10 runs in MSigDB and 32 runs in LINCS. In both datasets, there are many feature pairs (to the right of the red dotted line) in the top 20 that are more significant than their corresponding single features, e.g., on average 8 among top-20 feature

Drug	NumExprs	Avg Signif	Top1000 Signif	Top1000 Improved
vorinostat	904	235.5	1000	287
trichostatin	689	277.1	1000	0
estradiol	325	166.8	1000	203
tamoxifen	122	105.8	1000	9
doxorubicin	104	28.0	1000	369
gemcitabine	97	52.5	1000	116
daunorubicin	91	40.9	1000	28
idarubicin	78	30.1	1000	58
pravastatin	61	-7.5	0	0
Grand Total		43.1	9068	1070

Table 4. For each chosen drug from LINCS, the number of experiments in MCF7 cell line that were performed with that drug (NumExprs), and statistics for the top1000 feature pairs for that drug inclding the average $-\log_{10} corrected\_pval$ (Avg Signif), number of feature pairs with $-\log_{10} corrected\_pval > 5$ (Top1000 Signif), and number with $-\log_{10} improv\_pval > 0$ (Top1000 Improved).

pairs with $-\log_{10} improv\_pval > 5$ for LINCS. This shows that there is a great improvement from single features to some feature pairs in terms of the separability significance, at least for several datasets in our evaluations. [Saurabh: note here about possibility of redundant feature pairs contributing to the histogram.]

New Insight from Feature Pairs. In order to assess the quality of the top ranking feature pairs, we focused on the LINCS data set where the objects were experimental treatments on the MCF7 breast cancer cell line with the same *drug* and the features were expression values for different *genes*. For the evaluations above, we used object sets for the 40 drugs with the largest number of LINCS experiments. For the following analysis, we refine our list to selected drugs that are commonly administered [Saurabh: how did you arrive at this designation Charles?] and have at least 60 LINCS experiments on the MCF7 cell line. These nine drugs are vorinostat, trichostatin, estradiol, tamoxifen, doxorubicin, gemcitabine, daunorubicin, idarubicin, and pravastatin. For each chosen drug, we ran our SampOpt algorithm to rank the top-1000 feature (gene) pairs for separating the LINCs experiments of the drug from all other MCF7 experiments.

For all drugs, except pravastatin, all of the top-1000 ranked feature pairs were found to be significant ($-\log_{10} corrected\_pval > 5$). The average log significance of the top-1000 feature pairs correlated with the number of experiments ($r = 0.91$) in the drug's positive object set (see Table 4), suggesting the SampOpt algorithm requires a minimum number of example objects to learn a meaningful separator between the two classes. [Saurabh: I dont see this as the suggestion/inference. This is expected because p-values for data from the alternative hypothesis tend to be smaller for greater sample sizes.] As described in the Section 3.3, we are especially interested in feature pairs where its corrected p-value is better than the corrected p-values of its corresponding single features ($-\log_{10} improv\_pval > 0$). We found 1070 feature pairs with improved separability over their single feature among the top1000 of these evaluation drugs sets. One drug, trichostatin, had especially strong performing single features and showed no feature pairs that significantly improved on them. However, for the remaining seven drugs, there were as few as nine significant, improved feature pairs for tamoxifen and many as 369 pairs from doxorubicin (Table 4). [Saurabh: revisit this sentence. Doesnt make sense.]

Many of the above-mentioned 1070 significantly improved feature pairs are partially redundant, in the sense that they comprise a common best-ranked single feature (gene). An example of this is with the object set for the drug (compound) estradiol. We found the gene PRSS23 as the single feature with the highest separability and many feature pairs containing PRSS23 and a second gene as having an improved corrected p-value, for example (PRSS23, RAP1GAP), (PRSS23, TSC22D3), and (PRSS23, BAMBI). We looked for evidence of the relationship between the drug estradiol and these feature pair genes in the Comparative Toxicogenomics Database (CTD) **[CITE] CTD** and with our own literature survey. From this search, we found evidence for the pronounced effect of estradiol in increasing expression levels of PRSS23, RAP1GAP, and BAMBI, and decreasing expression of TSC22D3 **[CITE] CTD pubmeds**. So although the top single feature (gene PRSS23) reoccurred in multiple top feature pairs, each secondary feature gene was also meaningfully related to the administered drug in this case.

We next examined these [Saurabh: 'these top' is unclear. You mean 'the above 1070'?] top improved feature pairs to determine their consistency with existing biological knowledge sets. For our biological knowledge sets, we downloaded gene interaction datasets that had been derived from the databases of STRING, Reactome, Pathway Commons, HumanNet, BioGRID, Intact, DIP and BLAST **[CITE]** [Charles: may want supplementary methods/tables?]. These downloaded interaction networks covered 23,167 genes and had at least one known interaction between 2.17% of all possible gene pairs. Of the 996 unique feature pairs with significant $improv\_pval$ where both genes mapped onto the genes covered by the interaction networks, 133 gene pairs (13.4%) were found to have at least one known interaction. This six-fold enrichment demonstrates that GENVISAGE [Saurabh: beginning to use GENVISAGE as a name here, used to be SampleOpt above] more often finds feature pairs of genes that have a known relationship than is expected by chance. One example of such a feature pair is (GLRX, NME7), which is especially good for separating vorinostat experiments from all others. Not only are both of these genes known to have increased mRNA expression in response to vorinostat **[CITE] CTD pubmeds**, but the two genes are annotated by STRING to both be in database pathways of nucleotide biosynthesis, co-express with each other in other model organisms, and mentioned together often in literature abstracts. [Charles: link to STRING interface: https://string-db.org/cgi/network.pl?taskId=oggKTi2cKPAc] Furthermore, in Section 3.4 we will demonstrate that the positive objects and negative objects are visually separated under this feature pair, as shown in Figure 14.

In Supplementary Table [Charles: in google spreadsheet https://goo.gl/Z5jJ6Q for now], we examine several top improved feature gene pairs reported by SampOpt [Saurabh: keep to GENVISAGE or SampOpt consistently]. Of twenty-nine feature pairs in this table, six of them have three types of accompanying evidence: 1) literature-based relationship between the drug and the first gene, 2) literature-based relationship between the drug and the second gene, and 3) interaction network relationship between the pair of genes. Thirteen have just two of the three types of evidence and there are only eight with no evidence at all [Charles: this may improve when I finish additional manual search]. Particularly interesting are the top improved feature pairs in which neither of the single gene features ranked well alone. An example is the gene pair CDKN1A and CEBPB for separating doxorubicin experiments. Either gene feature alone is not within the top 600 genes for separating doxorubicin experiments from others. However, the combination of the pair is significant at a corrected p-value of [Saurabh: insert here] and is the second most improved feature pair for doxorubicin. This feature pair also has all three types of accompanying evidence; doxorubicin is known to increase expression of CDKN1A **[CITE] CTD pubmeds** and CEBPB **[CITE] CTD pubmeds**, and the pair of genes are annotated in STRING to have evidence for co-expression and text mining relationships. [Charles: STRING link https://string-db.org/cgi/network.pl?taskId=6L40K8DVTkDJ] This feature pair can be used to form an interesting hypothesis for further analysis or experiment. The potential for finding more significant and previously unidentified features is the reason we designed GENVISAGE to recover top ranking feature pairs instead of just single features.

[Charles: I did not insert a substitute for this figure 13 (no longer referred to in the text and to be deleted). It may not be as necessary with some of the examples in the supplementary table.]

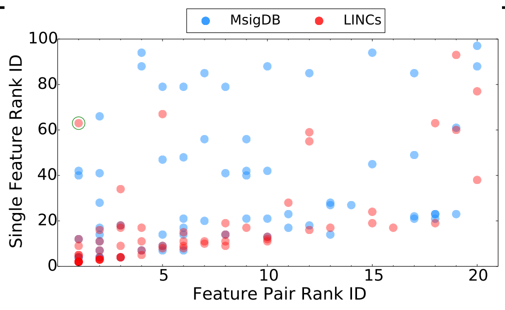
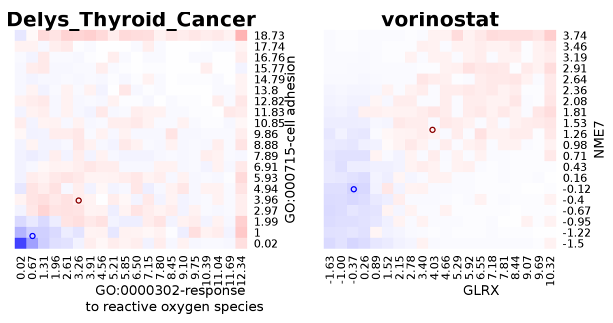


Fig. 13. Comparison of Feature Pair and Single Feature Rankings. The top-20 feature pairs from MSigDB (blue) and LINCS (red) datasets are depicted, with each circle representing a feature pair. The x-axis is the rank of the feature pair among all feature pair candidates, and y-axis is the better rank of its two individual single feature among all single features. The feature pair with the green circle is highlighted in the text.

[Saurabh: why overlap the MSigDB and LINCS? suggests a simultaneous examination, which is not what we want here] [Aditya: yes]



(a) MSigDB (b) LINCS

Fig. 14. Visualization Output of GENVISAGE. Heatmap visualization with the pair of top features providing the x and y axes and the name of the run providing the plot title. The relative density of objects determines the color of the heatmap cells with blue indicating a greater proportion of positive objects and red indicating a greater proportion of negative objects. The class centroids are represented by blue and red circles, positive and negative classes respectively. The two examples shown have the most $improv\_pval$ for the (a) MSigDB and (b) LINCS datasets.

3.4 Output Visualization

As discussed in Section 1, the output of GENVISAGE is not simply a ranking of the top feature pairs with their separability scores, but also a visualization that helps users to interpret the separability of the object classes. In Figure 14, we depict sample output visualizations from the MSigDB and LINCS runs. We pick the feature pair with the highest improved p-value, i.e., $improv\_pval$, using SampOpt in MSigDB, and the feature pair (GLRX, NME7) in LINCS as described in the “new insight” section [Saurabh: section number] above. We use a heatmap to illustrate the distribution of positive objects (in blue) and negative objects (in red) under the selected feature pair (f_i, f_j), where f_i is x-axis, f_j is y-axis. The color represents the difference between the weighted number of positive objects and negative objects. If the number of positive objects is larger than negative objects in the cell, the cell is colored as blue, otherwise, it is red. For the ease of interpretation, we also add the centroids of positive and negative objects using red circle and blue circle respectively. For the MSigDB example (Figure XX [Saurabh: missing number]), we observe that the stationary probability [Saurabh: this term has been forgotten by now, something simpler perhaps? feature values?] for negative objects is clustered around zero, while positive objects have higher stationary probability overall. For the LINCS example (Figure XX [Saurabh: missing number]), positive objects all have one gene probe up and one probe down, much different from the negative objects with zero-mean Gaussian distribution. [Charles: panel a relates to (DELYS_THYROID_CANCER_DN, response to reactive oxygen species, cell adhesion) which I cannot really appreciate other than the fact that cancer gene sets often relate to cell adhesion. panel b relates to (Vorinostat, TRAK2, KEAP1) and there is a paper that shows that vorinostat with other drugs induces upregulation of KEAP1, however I found nothing related to TRAK2.] These two examples illustrate how visualization of significant feature pairs can be a useful way to explain the separability of object sets and understand the data.

[Charles: SS would like us to figure out how to make a simple interface for this. so we need a design of configure page, design of results page, and examples how to call backend code]

[Charles: for the discussion: non-linear features?]

[Charles: Low priority, but I am interested in knowing for all the feature pairs we return that are better/new. what type of relationship between the pair are we finding. maybe captured by the slope of the perp bisector?]

[Charles: again low priority, but how robust are these results. If you run a sampling method twice, do you get the same feature pairs. if you change the positive set slightly, are your results comparable?]

1 Appendix

References

[1]W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[2]J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[3]V. N. Vapnik and V. Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

		Baseline	EarlyOrdering	SampOnly		SampOpt		HorizSampOpt		VertSampOpt	
				Phase 1	Phase 2	Phase 1	Phase 2	Phase 1	Phase 2	Phase 1	Phase 2
MSigDB	FPs Checked	2×10^8	2×10^8	2×10^8	4.1×10^6	1.9×10^8	2.3×10^6	10^7	1.2×10^6	10^7	5.5×10^5
	Objects Checked	22209	5427	694	22209	694	22209	694	22209	694	22209
LINCS	FPs Checked	2.5×10^8	2.5×10^8	2.5×10^8	3.3×10^5	2.5×10^8	9.1×10^4	10^7	8.6×10^4	10^7	5.0×10^4
	Objects Checked	98061	15631	564	98061	564	98061	564	98061	564	98061

Table 5. Supplement Table for Running Time Comparison in Figure 8. FPs represents feature pairs.