Information Theoretic Feature Selection Methods for Single Cell RNA-Sequencing

Umang Varma¹, Justin Colacino^{3,4,5}, and Anna Gilbert^{1,2}

¹Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, USA

²Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA

³Department of Environmental Health Sciences, University of Michigan School of Public Health, Ann Arbor, MI 48109, USA

⁴Department of Nutritional Sciences, University of Michigan School of Public Health, Ann Arbor, MI 48109, USA

⁵Center for Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA

Abstract

Single cell RNA-sequencing (scRNA-seq) technologies have generated an expansive amount of new biological information, revealing new cellular populations and hierarchical relationships. A number of technologies complementary to scRNA-seq rely on the selection of a smaller number of marker genes (or features) to accurately differentiate cell types within a complex mixture of cells. In this paper, we benchmark differential expression methods against information-theoretic feature selection methods to evaluate the ability of these algorithms to identify small and efficient sets of genes that are informative about cell types. Unlike differential methods, that are strictly binary and univariate, information-theoretic methods can be used as any combination of binary or multiclass and univariate or multivariate. We show for some datasets, information theoretic methods can reveal genes that are both distinct from those selected by traditional algorithms and that are as informative, if not more, of the class labels. We also present detailed and principled theoretical analyses of these algorithms. All information theoretic methods in this paper are implemented in our PICTUREDROCKS Python package that is compatible with the widely used SCANPY package.

1 Background

Single cell RNA-sequencing (scRNA-seq) technologies have generated an expansive amount of new biological information, revealing new cellular populations and hierarchical relationships. These methods quantify expression of thousands of genes across tens or hundreds of thousands of individual cells derived from a single cell suspension [1, 2]. Using unbiased methods for cell clustering, researchers define cell types based on these gene expression patterns [3]. When information derived from these transcriptionally-defined cell clusters is integrated with spatial or protein level data, additional biological insights can be gleaned. A number of technologies complementary to scRNA-seq rely on the selection of a smaller number of marker genes to accurately differentiate cell types within a complex mixture of cells. For example, fluorescence activated cell sorting (FACS) can isolate specific cell populations from a single cell suspension for further in depth biological characterization based on the expression of a distinguishing set of a small number of cell surface markers. Mass cytometry provides data on expression of up to 40 proteins for cells in a single cell suspension, adapting traditional flow cytometry methods by labeling antibodies with heavy metal isotopes rather than fluorophores and using a time-of-flight mass spectrometry readout [4]. Imaging mass cytometry further allows for in situ assessment of approximately 40 markers in tissue sections, quantifying the spatial distribution of marker expression in the tissue [5]. Finally, spatial transcriptomic profiling approaches use fluorescence in situ hybridization in tissue sections for imaging based quantification of gene expression [6, 7]. Optimal usage of each of these techniques requires selection of a small set of marker genes (or features) that can accurately assign an individual cell to a known cell type based on expression.

Feature selection is a challenging combinatorial problem; it is already known that feature selection is NP-hard [8]. In the field of scRNA-seq, the most commonly used techniques for marker gene selection data are based on highly variable gene identification [9, 10], differential expression, and logistic regression [11], the former being unsupervised and the later supervised. These methods are univariate: they consider properties of each feature individually and do not consider any interactions between features. Additionally, these methods make strong assumptions about how gene expression is distributed. They typically measure the mean and standard deviation of features and use these in their determination of variability (or differential expression). Such techniques, especially ones that output

probability values, are implicitly or explicitly making assumptions about the underlying distributions (e.g., Gaussian, or Student’s t-distribution). Even when the data suggest that these assumptions are reasonable, it is usually only the case for individual features in isolation. Supervised feature selection methods used in scRNAseq such as differential expression can natively handle binary class labels and need strategies such as one-vs-rest (OvR) or one-vs-one (OvO) to transform the problem back into a binary classification problem. This transformation causes two additional problems. First, by pooling clusters together (e.g., with OvR), we blur out valuable information by artificially suppressing genes that are informative about multiple class labels and boosting genes that are differentially expressed in one class when compared to all other classes. This puts an unnecessary importance on choosing class labels with the correct resolution. Secondly, methods to combining the lists of features identified in each pairwise comparison (either OvO or OvR) tend to be arbitrary and there is no canonical approach to reconciling the many $\binom{k}{2}$ or $k - 1$ in OvO and OvR respectively, where k is the number of class labels) lists of features into one list.

In contrast to differential expression based methods for marker selection, information-theoretic methods address a number of these issues. Rather than considering properties of each marker individually, information-theoretic algorithms consider interactions between features to various degrees. Specifically, the “degree K ” algorithm considers interactions between K features. For example, the degree 1 algorithm is a univariate algorithm, much like the differential expression methods, and the degree 2 algorithm considers interactions between pairs of features, etc. Additionally, in contrast to differential expression methods, information-theoretic methods can run as native multiclass methods as well as binary methods.

Our goal is to benchmark the efficacy of differential expression and information-theoretic methods for marker selection in scRNA-seq experiments, quantifying the strengths and weaknesses of a range of objective functions. We then focus on the degree 2 approximation, which corresponds to the conditional infomax feature extraction (CIFE) feature selection algorithm [12], and give theoretical guarantees for its performance. We then evaluate the performance of these algorithms on publicly available scRNA-seq datasets and describe optimizations for sparse data that improve the computational efficiency of these algorithms considerably.

2 Results

To assess the performance of mutual information methods in practice, we performed numerous experiments with publicly available single cell RNA-seq datasets. For each of the datasets listed below, we treat each cell as an observation and each gene as a feature.

- Paul dataset from [13] with 2730 mouse bone marrow cells and 3451 genes. Bone marrow cells form a continuous trajectory rather than distinct clusters because many of the cells are in the process of differentiating. The authors of the study assign class labels to cells based on expression of known marker genes.
- Green dataset [14] with 34633 cells from adult mouse testis and 37241 genes. The cells in dataset form a continuous trajectory, as should be expected for spermatogenesis.
- Zheng dataset from [10] with 68579 human blood cells and 32738 genes. This dataset consists of human blood cells that were clustered into distinct cell types. Zheng, et al then used these clusters to isolate more cells based on these types to validate their computational results.
- Zeisel dataset [15] with 3005 mouse neuron cells and 19972 genes. Because neuron cells lie in discrete clusters in gene space, we expect clustering and classification tasks to be easier for this dataset. Further, Zeisel, et al. have carefully assigned labels to clusters and validated their cluster labels as corresponding to known biology.

Recall that our goal is to find a small set of genes that explain the given target labels (e.g., cluster labels or biologically verified cell type labels). To this end, we performed 5-fold cross validation by splitting the datasets into 5 folds, and then, for each fold, we selected markers on the four remaining folds, projected the data onto the selected features (i.e., only considered the columns of the matrix that correspond to the selected genes), trained a classifier on the four folds, and predicted class labels on the current fold. The feature selection methods that perform well are those that select discriminative features that, together, help predict the class of an unseen observation. In other words, feature selection methods with the lowest error rates choose genes that explain the class labels best.

We tested three mutual information based feature selection algorithms:

- the CIFE algorithm (which is equivalent to the degree 2 approximation presented under **Degree K Supervised Feature Selection Algorithm** on page 20),

Figure 1: t -SNE Plot for Green Dataset using 10 features selected by CIFE (multiclass)

Figure 2: t -SNE Plot for Green Dataset using 10 features selected by JMI (multiclass)

Figure 3: t -SNE Plot for Green Dataset using 11 features selected by Logistic Regression

Figure 4: *t*-SNE Plot for Green Dataset using 11 features selected by *t*-test

Figure 5: *t*-SNE Plot for Green Dataset using 11 features selected by Wilcoxon Rank Sum

Figure 6: *t*-SNE Plot for Green Dataset using 11 features selected by CIFE (binary)

Figure 7: Classification Errors For Paul Dataset (left: multiclass MI methods, right: binary MI methods)

Figure 8: Classification Errors For Green Dataset (left: multiclass MI methods, right: binary MI methods)

- the MIM algorithm (which is equivalent to the degree 1 approximation), and
- the JMI algorithm (which dampens the redundancy penalty in CIFE by a factor of $\frac{1}{|S|}$, where S is the set of previously selected features).

For each dataset, we ran these algorithms in both a multi-class and binary fashion (see **Multiclass vs Binary Feature Selection Methods**). For all information-theoretic algorithms, we began by selecting the 5000 most relevant features (as found by the univariate MIM) before running bivariate algorithms (the run times in Figure 11 includes the MIM run time). For binary versions, this univariate filter was run for each class label separately.

We tested our algorithms with two different classifiers. Note that even for the datasets where class labels were originally found using an (unsupervised) classifier (rather than biologically), and we use supervised classifiers because we want to assess how informative the selected features are about the class labels.

- a Nearest Centroids Classifier that computes the centroid (unweighted mean of vectors) for each class label and to each unseen observation assigns the class label with the nearest centroid. We ran our experiments with this classifier to evaluate performance with a transparent classifier with minimal bells and whistles that can be easily implemented from scratch. Plots for classification errors with Nearest Centroids in the Figures section at the end of this paper.
- a Random Forests Classifier that uses an ensemble of decision trees. We use the Scikit Learn ([16]) implementation, which has numerous tweaks to improve accuracy and reduce overfitting. Although it is a more opaque

Figure 9: Classification Errors For Zeisel Dataset (left: multiclass MI methods, right: binary MI methods)

Figure 10: Classification Errors For Zheng Dataset (left: multiclass MI methods, right: binary MI methods)

Table 1: Number of Genes in Common between algorithms (Paul Dataset)

	LogReg	t-Test	Wilcoxon	RForest	JMI	MIM	CIFE	JMI(B)	MIM(B)	CIFE(B)
LogReg	100	40	36	24	18	18	16	36	31	28
t-Test	40	101	83	52	47	41	19	68	65	39
Wilcoxon	36	83	101	55	53	44	18	66	63	43
RForest	24	52	55	100	69	62	11	54	55	50
JMI	18	47	53	69	100	84	6	50	55	46
MIM	18	41	44	62	84	100	4	45	52	40
CIFE	16	19	18	11	6	4	100	9	7	12
JMI(B)	36	68	66	54	50	45	9	104	87	41
MIM(B)	31	65	63	55	55	52	7	87	104	40
CIFE(B)	28	39	43	50	46	40	12	41	40	101

Table 2: Number of Genes in Common between algorithms (Green Dataset)

	LogReg	t-Test	Wilcoxon	RForest	JMI	MIM	CIFE	JMI(B)	MIM(B)	CIFE(B)
LogReg	108	38	32	8	9	7	3	49	45	20
t-Test	38	106	89	28	24	15	6	55	47	27
Wilcoxon	32	89	103	27	28	17	11	49	41	27
RForest	8	28	27	100	45	34	3	25	19	6
JMI	9	24	28	45	100	73	8	21	16	9
MIM	7	15	17	34	73	100	3	16	12	9
CIFE	3	6	11	3	8	3	100	4	1	18
JMI(B)	49	55	49	25	21	16	4	108	79	27
MIM(B)	45	47	41	19	16	12	1	79	100	23
CIFE(B)	20	27	27	6	9	9	18	27	23	109

classifier, the Random Forests Classifier does not give each feature equal weight and will use new features only when they are helpful. This aligns more closely with our motivation of finding an efficient set of features that are collectively informative about the class labels rather than features that, individually, distinguish classes from one another.

We also tested against three differential expression algorithms (*t*-test with overestimated variation, Wilcoxon Rank Sum, and Logistic Regression) for single cell RNA sequencing (using the implementations in the widely used SCANPY package¹, see [17]). Since differential expression algorithms take a One-vs-Rest approach, we combined the lists into a single list by taking the first *k* genes (features) for each class label and removing duplicates.

Finally, since we are using a Random Forests classifier, we also evaluated Random Forests’ “feature importance” as a feature selection method. We should note that using Random Forests for feature selection and comparing against other methods using Random Forests is not a fair comparison, but we include it nonetheless for sake of comparison.

Figures 1, 2, 3, 4, 5, 6 are *t*-SNE plots (the projection of the dataset onto the top two principal components) for the Green dataset [14] generated using only a few genes. The colors of each point represents its class label (cell type).

The error rates in 5-fold cross validation with random forests are summarized in Figures 7, 8, 10, and 9 (the corresponding figures for the nearest centroids classifier are Figures 13, 14, 15, and 16 at the end of the paper).

In addition to error rates, which suggest how well the algorithms select features that explain class labels, we are also interested in knowing the extent to which these algorithm select the same features. We summarize these intersections in Tables 1, 2, 3, and 4. Because binary methods merge lists of features, we cannot ensure that all feature sets are of equal size. Therefore, not all sets are exactly of size 100 features, but they are the smallest possible sets of size ≥ 100 .

¹We used the latest developmental version (commit 80e635d2b78) available at the time as this revision fixed a bug that prevented us from using scanpy during *K*-fold cross validation.

Table 3: Number of Genes in Common between algorithms (Zeisel Dataset)

	LogReg	t-Test	Wilcoxon	RForest	JMI	MIM	CIFE	JMI(B)	MIM(B)	CIFE(B)
LogReg	100	48	43	27	29	21	4	48	50	23
t-Test	48	100	87	39	48	34	5	65	71	26
Wilcoxon	43	87	100	38	50	33	7	64	68	26
RForest	27	39	38	100	57	42	1	42	36	30
JMI	29	48	50	57	100	71	3	42	41	36
MIM	21	34	33	42	71	100	2	25	27	31
CIFE	4	5	7	1	3	2	100	4	3	2
JMI(B)	48	65	64	42	42	25	4	115	79	30
MIM(B)	50	71	68	36	41	27	3	79	100	20
CIFE(B)	23	26	26	30	36	31	2	30	20	106

Table 4: Number of Genes in Common between algorithms (Zheng Dataset)

	LogReg	t-Test	Wilcoxon	RForest	JMI	MIM	CIFE	JMI(B)	MIM(B)	CIFE(B)
LogReg	102	31	29	19	19	18	2	33	31	19
t-Test	31	110	94	52	60	59	14	61	64	34
Wilcoxon	29	94	108	53	62	62	15	57	60	36
RForest	19	52	53	100	79	77	20	53	51	51
JMI	19	60	62	79	100	94	16	58	58	50
MIM	18	59	62	77	94	100	14	55	56	49
CIFE	2	14	15	20	16	14	100	14	9	28
JMI(B)	33	61	57	53	58	55	14	101	81	45
MIM(B)	31	64	60	51	58	56	9	81	101	37
CIFE(B)	19	34	36	51	50	49	28	45	37	100

Figure 11: Runtimes for various algorithms

In order to compare the run times for the various algorithms, we timed individual runs of each algorithm on each dataset (without any cross-validation). The recorded times for bivariate methods (the multiclass and binary variants of both CIFE and JMI) include the run time for the initial univariate selection of 5000 features with MIM.

3 Discussion

On the datasets we have evaluated, the mutual information based methods seem to consistently outperform differential expression methods when run as binary methods. However, when run as multiclass methods, mutual information based methods perform better than differential expression methods only on some datasets, e.g., the Green dataset (Figure 8) and the Zheng dataset (Figure 10), while differential expression methods far outperformed multiclass methods in other datasets, e.g., the Paul dataset (Figure 7). It is worth noting that it is only on the Zheng dataset that selecting features based on Random Forests' feature importances is significantly better than other methods. We would also add that Figures 1, 2, 3, 4, and 5 demonstrate the strength of multiclass information theoretic methods on the Green dataset, but as we see by comparing Figure 8 to Figure 7, we don't expect this to be the case on all datasets.

Tables 1, 2, 4, and 3 suggest that many binary methods choose many of the same features. The notable exceptions are logistic regression and the binary CIFE method, both of which have moderate overlaps with other methods, but still select numerous features that other methods do not. The multiclass CIFE algorithm has, by far, the most distinct feature sets. This proves to be both a source of strength and weakness as the classification errors associated to these feature sets are sometimes very poor (e.g., in the Paul dataset, Figure 7, and, to a much milder extent, the Zheng dataset, Figure 10) but sometimes very competitive (e.g., in the Green dataset, Figure 8, and in the Zeisel dataset, Figure 9). This suggests that in some cases, CIFE can be used to discover novel marker genes for cell types and/or cell states in scRNA-seq.

The run times for information-theoretic algorithms, as demonstrated by Figure 11, are comparable to their differential expression counterparts. Specifically, the univariate MIM is almost always the fastest and the bivariate CIFE and JMI have similar run times to logistic regression.

Finally, we consider the assumptions we made to give our theoretical guarantees in the [Guarantees for Greedy Search](#) section and examine how close these datasets are to satisfying those assumptions. One of the most significant assumption we made (e.g., in Corollary 14) is that $I(x_i; x_j; y) \geq 0$ for all $x_i, x_j \in \mathcal{X}$. In Figure 12, we see that although the hypothesis don't hold perfectly on any dataset, they hold best on the Green dataset (where CIFE seems to perform best, see Figure 8) and worst on the Paul dataset (where CIFE seems to perform the worst, see Figure 7).

4 Conclusion

The opportunity presented by single cell RNA sequencing also poses challenges in its analysis. While traditional differential expression reveal meaningful information about the data, we believe that a multiplicity of feature selection techniques allows researchers to discover novel marker genes. Specifically, we believe that information-theoretic feature selection algorithms can detect marker genes that may otherwise be obscured by interactions with other genes or distributional assumptions of statistical tests.

We do not expect (in fact, we believe it is unrealistic to hope) that a single marker selection technique is appropriate for all purposes and datasets. The algorithms in this paper (especially CIFE) have proven to select sets of genes that are very distinct from those selected by traditional differential expression methods. In most cases the reclassification accuracy when restricting datasets to the chosen genes was comparable between information-theoretic algorithms and traditional differential expression algorithms and in some cases information-theoretic algorithms performed considerably better.

The reader will likely have noted that there are numerous possible combinations of configurations and versions of these algorithms to choose from. We attempt to summarize these options in the form of suggestions below.

- MIM is a univariate algorithm. Because it does not consider interactions between genes, it yields somewhat similar results to traditional differential expression algorithms, especially when run as a binary method rather than a multiclass method. As a multiclass method, it is particularly susceptible to selecting features with a strong signal without any consideration of how informative those features are together or what class labels they are informative about. Therefore, the output of MIM is more meaningful and easier to interpret when it is run as a binary method.

Figure 12: Histogram of $I(x_i; x_j; y)$ values for various datasets.

- CIFE is a bivariate algorithm that aggressively penalizes redundancy. When used as a multiclass method, it is particularly strong at identifying a small set of informative genes (in our examples, CIFE often far outperformed other algorithms when selecting 10 to 30 genes).

CIFE’s ability to consider interactions allows it to pick genes that may not be as informative in isolation but are informative in the context of other selected genes. However, when selecting a large number of genes, the interactions (particularly redundancies) overwhelm the algorithm and it eventually picks very uninformative features. This is especially true when individual genes have information about numerous class labels (for example, when there are many class labels in the dataset; see Example 5).

If we need a larger set of genes or want marker genes for specific class labels, we recommend using CIFE as a binary feature selection method. Running the algorithm as a binary method ensures that the number of features chosen per class label remains relatively low.

- JMI is a bivariate algorithm that acts as an intermediate between CIFE and MIM. As the algorithm selects more features, it dampens the impact of interactions resulting, eventually, in a convergence in behavior between JMI and MIM. As a consequence of this damping, it is unlikely to unearth informative features hidden behind interactions as CIFE can nor does its output reflect the strength of the signal of features in isolation. The damping in JMI is a heuristic that is not justified theoretically and should be used cautiously. However, empirical results show that its performance is consistently decent and avoids the occasional extreme poor performance of CIFE.

In all of these cases, we strongly recommend using the recursive quantile transform we describe under **Binning Technique** to discretize datasets.

Used in conjunction with traditional differential expression methods, the information-theoretic algorithms in this paper have the potential to discover novel marker genes that reveal information that otherwise be lost on differential expression algorithms that prioritize signal strength of genes over new information. Future directions include more sophisticated analysis and benchmarking of unsupervised feature selection methods for single cell RNA sequencing (see supplementary materials for examples).

5 Methods

Our benchmarking process relied heavily on SCANPY, a widely used package for single cell analysis in Python, and PICTUREDROCKS, a software package written by the first and last authors of this paper. PICTUREDROCKS is a Python package that can perform information-theoretic feature selection and also contains tools to benchmark feature selection methods via k -fold cross-validation. PICTUREDROCKS includes a interactive user interface for feature selection that can allows researchers to incorporate domain knowledge in the process of feature selection. All the code to reproduce our benchmarking study can also be found on GitHub.

There are numerous possible pitfalls when evaluating the ability of a feature selection method to select informative features. We refer the reader to the excellent section on “The Wrong and Right Way to do Cross-validation” in [18, Section 7.10.2]. In particular, we highlight the following:

- Cross-validation is effective at estimating the generalization error rather than the test error, which is susceptible to artificially being suppressed due to overfitting.
- It is extremely important to ensure that at no point does any data “leak” between folds. For example, as explained in [18], it is not fair to evaluate a feature selection method by first selecting features on the entire dataset and then report the cross-validation error. Even though the classifier used to predict labels for each fold was not trained on that fold, the feature selection algorithm had access to data from all folds.

In order to avoid such pitfalls, we performed our benchmarking in the following manner:

1. Split the dataset into 5 folds
2. For each fold $k = 1, \dots, 5$:
  - (a) use all samples not in fold k to select n features; repeat for each feature selection algorithm and store results separately

Remark 1: For binary feature selection methods (such as t -test, Wilcoxon Rank Sum, Logistic Regression, and when any of the information-theoretic methods are forcibly run as binary methods), this means

running the algorithm for each class label and combining the top $\lceil n/c \rceil$ features from each class label, where c is the number of classes. Remove duplicates. Keep track of the number of features selected, since this may not be exactly n .

Remark 2: the bivariate methods (CIFE and JMI) can slow down considerably in situations when we are selecting many features and there are many candidate features (specifically there is a $O(np)$ term in the time complexity of these methods). To avoid these prohibitively slow run times, we use a univariate method (MIM) to select 5000 features and then use the bivariate method to select among those 5000 features

- (b) train a Random Forests classifier and a Nearest Centroids classifier on all samples not in fold k with only the n (or possibly a few more) features selected; repeat for each feature selection algorithm
- (c) use both classifiers to predict labels for samples in fold k and store the predictions of each classifier separately; repeat for each feature selection algorithm

3. for each feature selection algorithm and classifier combination:

- (a) compute the error rate of the predictions as $\#(\text{wrongly classified})/\#(\text{samples})$.
- (b) for binary methods where the number of features may not be exactly n , record the number of features used as the average number of features used for each fold

For t -test, Wilcoxon Rank Sum, and Logistic Regression feature selection methods we used the standard implementation in SCANPY. As part of the feature selection subroutines, we perform basic preprocessing (as this happens on a copy of the data, it does not affect the data used for classification): **normalize per cell** followed by **log1p**. These methods scale the counts matrix so that each cell's counts sum to the same number (this is usually chosen to the median count prior to scaling) and then apply the $x \mapsto \log(1 + x)$ transformation element-wise. For the information-theoretic methods, we use the recursive quantile transformation (with 5 bins) before running the methods, see **Binning Technique** in the supplementary material of this paper.

At the classification step, we used the Random Forests classifier from the Scikit-learn package[16] with 100 estimators. As a preprocessing step, we scaled the data so that the sum of counts for each cells was 1000 (this is arbitrary but had to be chosen in order to standardize the preprocessing between training and testing) and then applied the $x \mapsto \log(1 + x)$ transformation. For the Nearest Centroids classifier, we applied the $x \mapsto \log(1 + x)$ transformation, projected onto the top 30 principal components of the training data (as euclidean distances become less meaningful in high dimensions), and the mapped each training point to the class with the nearest centroid in the training data.

When generating Figure 11 and Tables 1, 2, 3, 4, we performed the preprocessing for feature selection algorithms listed above and ran the algorithms on the entire dataset rather than use cross validation as these figures are not meant to estimate generalizability. Figures 1, 2, 3, 4, 5, and 6 were also generated with these feature sets using the t -SNE implementation in Scikit-Learn [16].

5.1 Introduction to Information Theoretic Methods

The remainder of this section introduces the basic concepts behind information theoretic feature selection methods. In addition to benchmarking these methods, we have performed theoretical analyses of these algorithms and present our findings in detail in the supplementary material.

Recall that we seek to find a small subset of genes that is most informative about the class labels. In the language of information theory, we can frame this as the problem finding a small subset S of the variables $\mathcal{X}$ (the set of all genes) such that $I(S; y)$, the mutual information between a set of features S and target labels y , is maximized. This warrants some definitions before we proceed.

Definitions, Notation, and Abuses Thereof

We begin with the definition of mutual information. The mutual information between random variables U and V can be thought of as the number of bits of information knowing U tells us about V . For example, if U is a outcome of a fair coin toss (i.e., U takes values in $\{H, T\}$ with equal probability) and V is the opposite of U (i.e., V is T if U is H and vice-versa), then their mutual information is 1 bit. However, if V is a “noisy” opposite of U so that V is the opposite of U with probability 0.99 and is equal to U with probability 0.01 then $I(U; V)$ should be just under 1 (in fact it is approximately 0.92).

Definition. If U and V are discrete random variables, then the mutual information $I(U; V)$ between U and V is

$$\begin{aligned} I(U; V) &= \sum_u \sum_v P_{U,V}(u, v) \log \left(\frac{P_{U,V}(u, v)}{P_U(u)P_V(v)} \right) \\ &= H(U) + H(V) - H(\{U, V\}), \end{aligned}$$

where

- P_X is the probability mass function of X , a random variable, in particular $P_{U,V}(u, v) = P(U = u \text{ and } V = v)$,
- $H(\cdot)$ is the Shannon entropy of a discrete random variable, and
- the above logarithms are with base 2, by convention.

We will only consider the case when U and V are discrete random variables (in practice, such algorithms are used after data has been discretized, although continuous analogues that use density estimators exist, but are computationally harder).

By abuse of notation and in line with existing literature, we will often

- treat a set of random variables as an ensemble of random variables. That is,

$$\begin{aligned} H(\{X_1, \dots, X_n\}) \\ = - \sum_{x_1, \dots, x_n} p(x_1, \dots, x_n) \log p(x_1, \dots, x_n), \end{aligned}$$

- drop the $\cup$ symbol and write $H(UV)$ to mean $H(U \cup V)$, and
- treat a feature x_i as a singleton $\{x_i\}$, so $H(Ux_i) = H(U \cup \{x_i\})$.

Information-theoretic Feature Selection

We may now return to making the statement of our problem precise. We want to find $S \subset \mathcal{X}$ such that $I(S; y)$ is maximum over subsets of size k or less of $\mathcal{X}$. Here k is a user-prescribed number of features to select. There are $\binom{P}{k}$ such subsets (for fixed k , this is $O(P^k)$). Datasets often have many thousands of features from which we may seek to select on the order of a hundred features, which makes an exhaustive search computations intractable. It is already known that feature selection is NP-hard (see [8]), as are sub-modular maximization problems for coverage-type functions (see [19]) that our problem can be reduced to by constructions similar to those discussed under **Analysis of Existing Methods** in this paper. Various greedy algorithms, usually of the form of Algorithm 1, have been studied in the literature. We give a summary of these algorithms in Table 5. These algorithms use different objective functions $J: \mathcal{X} \rightarrow \mathbb{R}$, where $J(x_i)$ estimates the (relative) value of feature x_i given that $S \subset \mathcal{X}$ has already been chosen.

Algorithm 1 General Greedy Algorithm for Feature Selection with Objective J

```

1: procedure GREEDYSEARCH( $\mathcal{X}, y, n$ )
2:    $S \leftarrow \emptyset$ 
3:   while  $|S| < n$  do
4:      $S \leftarrow S \cup \left\{ \arg \max_{x_i \in \mathcal{X}-S} J(x_i) \right\}$ 
5:   end while
6:   return  $S$ 
7: end procedure
```

If we were to naïvely maximize $I(S; y)$ via a greedy search, we would run Algorithm 1 with $J_{\text{cmi}}(x_i) = I(S \cup \{x_i\}; y)$. Computing this requires estimating the joint probability distribution of Sx_iy (which has $|S| + 2$ dimensions). This is not only computationally intractable, but also impossible to estimate reasonably in the absence of a very large number of samples. Therefore, the objective function

$$J_{\text{cmi}}(x_i) = I(S \cup \{x_i\}; y) \tag{1}$$

Objective function	Description
$J_{\text{cmi}}(x_i) = I(S \cup \{x_i\}; y)$	This is the objective we would ideally like to optimize against, but the cost of computing its empirical value is prohibitive.
$J_{\text{mim}}(x_i) = I(x_i; y)$	The Mutual Information Maximization (MIM) is a univariate feature selection method—it does not consider the interactions between variables and only considers features in isolation.
$J_{\text{mrmmr}}(x_i) = I(x_i; y) - \frac{1}{ S } \sum_{x_j \in S} I(x_i; x_j)$	The minimum Redundancy Maximum Relevance (mRMR) algorithm proposed by [21] uses a diminishing penalty on redundancy and does not distinguish between relevant and irrelevant redundancy (see Example 7).
$J_{\text{mifs}}(x_i) = I(x_i; y) - \beta \sum_{x_j \in S} I(x_i; x_j)$	The Mutual Information Feature Selection (MIFS) algorithm proposed by [21] uses a diminishing penalty on redundancy and does not distinguish between relevant and irrelevant redundancy.
$J_{\text{jmi}}(x_i) = I(x_i; y) - \frac{1}{ S } \sum_{x_j \in S} I(x_i; x_j; y)$	The Joint Mutual Information (JMI) algorithm proposed by [22], like the mRMR algorithm, uses a diminishing penalty on redundancy. However, it only penalizes relevant redundancy which, we explain under Analysis of Existing Methods , is desirable.
$J_{\text{cife}}(x_i) = I(x_i; y) - \sum_{x_j \in S} I(x_i; x_j; y)$	The (CIFE) algorithm proposed by [12] does not diminish its redundancy penalty (which it applies only to relevant redundancy, as desired). Although it can penalize redundancy somewhat aggressively, we will show that it is the degree 2 approximation to J_{cmi} and, under certain conditions, is a submodular set function. We focus on these properties of the CIFE objective to give theoretical guarantees on its performance.

Table 5: Summary of various objective functions that are used with Algorithm 1.

is difficult to estimate. We follow the terminology of Brown, et al. [20] and will refer to this as the conditional mutual information (CMI) objective because a greedy search with the conditional mutual information objective, $J(x_i) = I(x_i; y|S)$, is equivalent to using J_{cmi} from (1). This is because $I(x_i; y|S) - I(Sx_i; y)$ is constant with respect to x_i .

In this work, we present a series expansion of $I(S \cup \{x_i\}; y)$ in terms of multivariate mutual information through which we can derive degree K approximations to $I(S \cup \{x_i\}; y)$. This expansion gives us insight into the strengths and weaknesses of various objective functions in the literature. We then focus on the degree 2 approximation, which corresponds to the CIFE feature selection algorithm [12], and give theoretical guarantees for its performance. We then evaluate the performance of these algorithms on publicly available scRNA-seq datasets and describe optimizations for sparse data that improve the computational efficiency of these algorithms considerably.

5.2 Previous work

Because of the computational challenges in empirically evaluating J_{cmi} , many approximation algorithms have been proposed. Brown et al. [20] generalize a vast number algorithms in the literature as

$$J(x_i) = I(x_i; y) - \beta \sum_{j \in S} I(x_i; x_j) + \gamma \sum_{j \in S} I(x_i; x_j | y). \quad (2)$$

Specifically, the widely used mRMR algorithm in [21] has $\beta = \frac{1}{|S|}$ and $\gamma = 0$. The MIFS algorithm by [23] has $\gamma = 0$ and leaves β as a (constant) hyperparameter chosen from $[0, 1]$. The CIFE algorithm in [12] has $\beta = \gamma = 1$. The JMI algorithm ([22]) and IGFS² algorithm ([24]) use $\beta = \gamma = \frac{1}{|S|}$. See Table 5 for a summary of the various

²Brown et al. incorrectly claim IGFS is equivalent to CIFE, it is, in fact, equivalent to JMI.

algorithms described.

Brown et al. provide an interpretation of this class of objective functions in terms of three assumptions.

1. Given an unselected feature x_k ,
  - (a) the selected features are independent: $p(S|x_k) = \prod_{x_j \in S} p(x_j|x_k)$ and
  - (b) the selected features are class-conditionally independent: $p(S|yx_k) = \prod_{x_j \in S} p(x_j|yx_k)$.
2. Features are pairwise class-conditionally independent: $p(x_i x_j|y) = p(x_i|y)p(x_j|y)$.
3. Features are pairwise independent: $p(x_i x_j) = p(x_i)p(x_j)$.

5.3 Our Work

We attempt to address two sets of important questions about the information theoretic methods that we discuss in this paper:

- What are the differences between the various objective functions? Under what conditions does the CMI objective function (the ideal objective function that we would use if computing power and quantity of data were not restricted) reduce to the approximate objective functions such as MIM, mRMR, JMI, and CIFE? What explains the relative strengths and weaknesses of these objective functions?
- What guarantees can we make for the greedy search algorithm? How far from optimal are the outputs of the greedy search?

We address the first point by expanding the CMI objective function $J(x) = I(S \cup \{x\}, y)$, which considers all possible interactions, in terms of lower-order interactions under the [Series Expansion of Mutual Information](#) section. We show that truncating this series after the degree 2 terms gives us the CIFE objective. The reader may also find it helpful to read the example-based section on [Analysis of Existing Methods](#) which demonstrates where various objective functions fail.

The second set of question, on the effectiveness of the greedy search algorithm, we demonstrate in the section titled [Guarantees for Greedy Search](#) that under certain conditions, mutual information and its degree 2 approximation (i.e., CIFE) are submodular functions and existing literature on submodular functions to give theoretical guarantees for the performance of the greedy search algorithm for CIFE.

Under [Practical Implementation Issues](#), we explain various considerations that the practitioner will find useful. This includes a section our approach to preprocessing datasets for information-theoretic feature selection methods as these methods require the counts to be in discrete bins, as well as a section on trade offs between running information-theoretic methods as mutli-class and binary methods.

Finally, we note that we have implemented fast versions of these algorithms in the PICTUREDROCKS Python package and we explain some of the optimizations that we made under [Efficient Algorithms](#).

Availability of data and material

Our package PICTUREDROCKS is available on GitHub and the Python Package Index (`pip install picturedrocks`) and installs SCANPY as a dependency. The Paul dataset [13] is available on the NCBI Gene Expression Omnibus (GEO: GSE72857) and also ships with SCANPY. The Green dataset [14] is available on the NCBI Gene Expression Omnibus (GEO: GSE112393). The Zheng dataset ships with SCANPY and is available on 10x Genomics website (<https://support.10xgenomics.com/single-cell-gene-expression/datasets>). The Zeisel dataset [15] is available on the Linnarsson Lab webpage (<http://linnarssonlab.org/cortex/>). Code for all the benchmarking experiments performed in this paper is available at <https://github.com/umangv/picturedrocksbenchmarks>.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

UV developed PICTUREDROCKS and wrote the code for benchmarking and analyzed the data with assistance from JC and ACG. UV developed the theoretical analysis with assistance from ACG. All authors contributed to the writing of the manuscript and approve the final manuscript.

Funding

UV and ACG are supported in part by the Michigan Institute for Data Science, and the Chan Zuckerberg Initiative. JC is supported by grants from the National Institute of Environmental Health Sciences (R01 ES028802) and the Michigan Institute for Data Science.

Acknowledgements

The authors acknowledge the help of Jun Z. Li and Alexander Vargo for helpful discussions.

References

- [1] Gierahn TM, Wadsworth II MH, Hughes TK, Bryson BD, Butler A, Satija R, et al. Seq-Well: portable, low-cost RNA sequencing of single cells at high throughput. *Nature Methods*. 2017 2;14:395. Available from: <https://doi.org/10.1038/nmeth.4179>.
- [2] Macosko E, Basu A, Satija R, Nemesh J, Shekhar K, Goldman M, et al. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*. 2015 May;161(5):1202–1214. Available from: <https://doi.org/10.1016/j.cell.2015.05.002>.
- [3] Duò A, Robinson M, Soneson C. A systematic performance evaluation of clustering methods for single-cell RNA-seq data [version 2; peer review: 2 approved]. *F1000Research*. 2018;7(1141).
- [4] Bendall SC, Simonds EF, Qiu P, Amir EaD, Krutzik PO, Finck R, et al. Single-Cell Mass Cytometry of Differential Immune and Drug Responses Across a Human Hematopoietic Continuum. *Science*. 2011;332(6030):687–696. Available from: <https://science.sciencemag.org/content/332/6030/687>.
- [5] Giesen C, Wang HAO, Schapiro D, Zivanovic N, Jacobs A, Hattendorf B, et al. Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry. *Nature Methods*. 2014 Mar;11:417. Article. Available from: <https://doi.org/10.1038/nmeth.2869>.
- [6] Raj A, van den Bogaard P, Rifkin SA, van Oudenaarden A, Tyagi S. Imaging individual mRNA molecules using multiple singly labeled probes. *Nature Methods*. 2008 Sep;5:877. Available from: <https://doi.org/10.1038/nmeth.1253>.
- [7] Eng CHL, Lawson M, Zhu Q, Dries R, Koulina N, Takei Y, et al. Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature*. 2019;568(7751):235–239. Available from: <https://doi.org/10.1038/s41586-019-1049-y>.
- [8] Amaldi E, Kann V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*. 1998;209(1):237 – 260. Available from: <http://www.sciencedirect.com/science/article/pii/S0304397597001151>.
- [9] Satija R, Farrell JA, Gennert D, Schier AF, Regev A. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*. 2015 4;33:495. Available from: <http://dx.doi.org/10.1038/nbt.3192>.
- [10] Zheng GXY, Terry JM, Belgrader P, Ryvkin P, Bent ZW, Wilson R, et al. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*. 2017 1;8:14049. Available from: <http://dx.doi.org/10.1038/ncomms14049>.
- [11] Ntranos V, Yi L, Melsted P, Pachter L. Identification of transcriptional signatures for cell types from single-cell RNA-Seq. *bioRxiv*. 2018;.

- [12] Lin D, Tang X. Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion. In: Leonardis A, Bischof H, Pinz A, editors. Computer Vision – ECCV 2006. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006. p. 68–82.
- [13] Paul F, Arkin Y, Giladi A, Jaitin D, Kenigsberg E, Keren-Shaul H, et al. Transcriptional Heterogeneity and Lineage Commitment in Myeloid Progenitors. *Cell*. 2015;163(7):1663 – 1677. Available from: <http://www.sciencedirect.com/science/article/pii/S0092867415014932>.
- [14] Green CD, Ma Q, Manske GL, Shami AN, Zheng X, Marini S, et al. A Comprehensive Roadmap of Murine Spermatogenesis Defined by Single-Cell RNA-Seq. *Developmental Cell*. 2018;46(5):651 – 667.e10. Available from: <http://www.sciencedirect.com/science/article/pii/S1534580718306361>.
- [15] Zeisel A, Muñoz-Manchado AB, Codeluppi S, Lönnerberg P, La Manno G, Juréus A, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*. 2015;347(6226):1138–1142. Available from: <http://science.sciencemag.org/content/347/6226/1138>.
- [16] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825–2830.
- [17] Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*. 2018 2;19(1):15.
- [18] Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer New York; 2013.
- [19] Vondrák J. Submodularity in Combinatorial Optimization [Ph.D. thesis]. Charles University, Prague, Czech Republic; 2007. Available from: https://theory.stanford.edu/~jvondrak/data/KAM_thesis.pdf.
- [20] Brown G, Pocock A, Zhao MJ, Luján M. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*. 2012;13(Jan):27–66.
- [21] Peng H, Long F, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2005 8;27(8):1226–1238.
- [22] Yang HH, Moody J. Data Visualization and Feature Selection: New Algorithms for Nongaussian Data. In: Solla SA, Leen TK, Müller K, editors. Advances in Neural Information Processing Systems 12. MIT Press; 2000. p. 687–693.
- [23] Battiti R. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*. 1994 7;5(4):537–550.
- [24] El Akadi A, El Ouardighi A, Aboutajdine D. A Powerful Feature Selection approach based on Mutual Information. *International Journal of Computer Science and Network Security*. 2008 4;8(4):116–121.
- [25] Han TS. Linear dependence structure of the entropy space. *Information and Control*. 1975;29(4):337 – 368.
- [26] Graham RL, Grötschel M, Lovász L, editors. Handbook of Combinatorics: Vol. 2. Elsevier; 1995.
- [27] Nemhauser GL, Wolsey LA, Fisher ML. An analysis of approximations for maximizing submodular set functions-I. *Mathematical programming*. 1978;14(1):265–294.
- [28] Fleuret F. Fast Binary Feature Selection with Conditional Mutual Information. *J Mach Learn Res*. 2004 12;5:1531–1555. Available from: <http://dl.acm.org/citation.cfm?id=1005332.1044711>.

Figures

A Series Expansion of Mutual Information

In the **Ignoring Higher-degree Interactions Between Features** section, we give examples of *shared redundancy* and *synergy*. These interactions between features is explained by multivariate mutual information. Intuitively, we want

Figure 13: Classification Errors with Nearest Centroid Classifier For Paul Dataset (left: multiclass MI methods, right: binary MI methods)

Figure 14: Classification Errors with Nearest Centroid Classifier For Green Dataset (left: multiclass MI methods, right: binary MI methods)

to measure how much information a set of random variables share simultaneously. If the one bit of information that x_3 shares with x_1 is the same as the one bit of information that x_3 shares with x_2 , then we want $I(x_1; x_2; x_3) = 1$, and if x_3 shares different bits of information with x_1 and x_2 , then $I(x_1; x_2; x_3) = 0$.

This suggests a generalization of mutual information that was studied in [25]. We let $I_1(x) = H(x)$ to be the Shannon entropy of x (i.e., the self-information). Define the multivariate mutual information $I_n(x_1; \dots; x_n)$ recursively as $I_n(x_1; \dots; x_n) = I_{n-1}(x_1; \dots; x_{n-1}) - I_{n-1}(x_1; \dots; x_{n-1} | x_n)$, where $I_{n-1}(x_1; \dots; x_{n-1} | x_n)$ is the information shared between $x_1, \dots, x_{n-1}$ given we know the value of x_n . If $S = \{x_1, \dots, x_n\}$, we denote by $I_n(S)$ the multivariate mutual information $I_n(x_1; x_2; \dots; x_n)$. Note the subtle, but crucial, difference in notation: $I_n(S)$ is the information simultaneously shared by the n variables in S (intuitively, the intersection of information), while $I(S)$ is the self-information of the ensemble of random variables of S (intuitively, the union of the information). The following equality (à la Principle of Inclusion and Exclusion) expresses the multivariate mutual information $I_n(S)$ in terms of lower degree information terms.

Proposition 1.

$$I_n(x_1; x_2; \dots; x_n) = \sum_{X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|+1} H(X)$$

Figure 15: Classification Errors with Nearest Centroid Classifier For Zheng Dataset (left: multiclass MI methods, right: binary MI methods)

Figure 16: Classification Errors with Nearest Centroid Classifier For Zeisel Dataset (left: multiclass MI methods, right: binary MI methods)

Proof. We prove the equality above by induction. For $n = 2$, this follows from the definition of mutual information. For $n > 2$, we have

$$\begin{aligned}
 I_n(x_1; x_2; \dots; x_n) &= I_n(x_1; \dots; x_{n-1}) - I(x_1; \dots; x_{n-1} | x_n) \\
 &= \sum_{X \subseteq \{x_1, \dots, x_{n-1}\}} (-1)^{|X|+1} (H(X) - H(X | x_n)) \\
 &= \sum_{X \subseteq \{x_1, \dots, x_{n-1}\}} (-1)^{|X|+1} (H(X) - H(X x_n) + H(x_n)) \\
 &= \left(\sum_{X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|+1} H(X) \right) + \sum_{i=0}^n \binom{n}{i} H(x_n) \\
 &= \sum_{X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|+1} H(X).
 \end{aligned}$$

□

Corollary 2.

$$H(A) = \sum_{S \subseteq A} (-1)^{|S|+1} I_{|S|}(S).$$

Proof. This follows from a version of the principle of inclusion and exclusion, see [26, p. 1049 Theorem 12.1]. $\square$

We now give a series expansion of the mutual information $I(\{x_1, \dots, x_n\}, y)$ that suggests approximations to $I(\{x_1, \dots, x_n\}, y)$ using lower degree terms (i.e., without needing to estimate the joint probability distribution of a large set of variables).

Lemma 3.

$$I(\{x_1, \dots, x_n\}, y) = \sum_{k=1}^n \sum_{\substack{X \subseteq \{x_1, \dots, x_n\} \\ |X|=k}} (-1)^{k+1} I_{k+1}(X; y).$$

Proof.

$$\begin{aligned} & I(\{x_1, \dots, x_n\}, y) \\ &= H(\{x_1, \dots, x_n\}) + H(y) - H(\{x_1, \dots, x_n, y\}) \\ &= \sum_{X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|-1} I_{|X|}(X) + I(y) \\ &\quad - \sum_{X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|-1} I_{|X|}(X) \\ &= \sum_{\emptyset \subsetneq X \subseteq \{x_1, \dots, x_n\}} (-1)^{|X|+1} I_{|X|+1}(X; y) \\ &= \sum_{k=1}^n \sum_{\substack{X \subseteq \{x_1, \dots, x_n\} \\ |X|=k}} (-1)^{k+1} I_{k+1}(X; y). \end{aligned} \tag{3}$$

$\square$

A.1 Degree K Supervised Feature Selection Algorithm

It is clear that if $X \subseteq Y$, then $I_{|X|}(X) \geq I_{|Y|}(Y)$. In a practical setting it is also reasonable to believe that as k gets larger, the terms in the series in (3) converge to 0. As mentioned earlier, the computational cost of the terms in the series above grows exponentially with respect to k . This suggests that a fast “degree K ” approximation to the mutual information $I(\{x_1, \dots, x_n\}; y)$:

$$I(\{x_1, \dots, x_n\}, y) \approx \sum_{k=1}^K \sum_{\substack{X \subseteq \{x_1, \dots, x_n\} \\ |X|=k}} (-1)^{k+1} I_{k+1}(X; y).$$

At each iteration of the greedy algorithm, our goal is to compute

$$\arg \max_{x_i \notin S} I(S \cup \{x_i\}; y). \tag{4}$$

If we use the above degree K approximation, we only need to consider the terms involving x_i in the series expansion. Therefore, at each step, the degree K mRMR algorithm selects the next feature to add to S as:

$$J_{\text{deg}K}(x_i) = \arg \max_{x_i \notin S} \sum_{k=0}^{K-1} \sum_{\substack{X \subseteq S \\ |X|=k}} (-1)^k I_{k+2}(X; x_i; y).$$

Notice that for $K = 2$, this simplifies to

$$J_{\text{deg}2}(x_i) = \arg \max_{x_i \notin S} I(x_i; y) - \sum_{x_j \in S} I(x_i; x_j; y),$$

the CIFE objective in [12]. These functions $J_{\text{deg}K}$ serve as the objective function J in the greedy search algorithm (Algorithm 1).

A.2 Degree K Unsupervised Feature Selection Algorithm

In the unsupervised setting, we are trying to solve

$$\arg \max_{S \subseteq \mathcal{X}; |S| \leq n} H(S).$$

This involves approximating n -dimensional joint probability distributions, which necessitates the need for lower-degree approximations.

We use Corollary 2 to expand $H(S)$ and write

$$H(S) = \sum_{k=1}^{|S|} \sum_{U \subseteq S; |U|=k} I_{|U|}(U).$$

If we assume that there are no higher-degree interactions, we get a low degree approximation for $H(S)$.

Corollary 4. *If $I_{|U|}(U) = 0$ whenever $|U| \geq 3$, then*

$$H(S) = \widetilde{H}_2(S) := \sum_{x_i \in S} H(x_i) - \sum_{x_i, x_j \in S} I(x_i; x_j).$$

□

Using the greedy search (Algorithm 1), at each step, we identify

$$\begin{aligned} J_{\text{unsup degK}}(x_i) &= \arg \max_{x_i \notin S} \left(\widetilde{H}_2(S \cup \{x_i\}) - \widetilde{H}_2(S) \right) \\ &= \arg \max_{x_i \notin S} \left(H(x_i) - \sum_{x_j \in S} I(x_i; x_j) \right). \end{aligned}$$

B Guarantees for Greedy Search

The greedy search algorithm (Algorithm 1) is a necessary compromise to avoid the computation cost of an exhaustive search over all subsets of size at most n . However, there are some cases where such an approach is too shortsighted, even when we use the ideal objective function $J_{\text{cmi}}(x) = I(S \cup \{x\}; y)$.

Example 1. Let b_1, b_2 be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let b_3 be a random variable (independent of b_1, b_2) such that $P(b_3 = 0) = \frac{1}{2} + \varepsilon$ and $P(b_3 = 1) = \frac{1}{2} - \varepsilon$ where $0 < \varepsilon < \frac{1}{2}$. Finally, let $x_1 = b_1$, $x_2 = b_2$, $x_3 = b_1 \oplus b_2 \oplus b_3$, and $y = b_1 \oplus b_2$. It is clear that $I(x_1; y) = I(x_2; y) = 0$ and $I(x_3; y) = 1 - H(b_3)$. The optimal 2-element subset S of $\{x_1, x_2, x_3\}$ that maximizes $I(S; y)$ is obviously $\{x_1, x_2\}$, but a greedy approach will always choose x_3 at the first step.

In this section, we describe conditions under which we can give performance guarantees for the greedy search algorithm. These results rely heavily on results for the maximization of submodular functions. A function f defined on sets is submodular if it satisfies the diminishing returns property: $f(T + \{x\}) - f(T) \leq f(S + \{x\}) - f(S)$ whenever $S \subseteq T$. The seminal paper on this topic by [27] proves the following result.

Theorem 5 ([27]). *Consider the maximization problem*

$$\arg \max_{S \subseteq \mathcal{X}, |S| \leq k} f(S).$$

If f is a non-decreasing submodular function, the greedy algorithm yields a $(1 - 1/e)$ -approximation. In other words, if S_o is an optimal set and S_g is the set found via a greedy search, then

$$\frac{f(S_o) - f(S_g)}{f(S_o) - f(\emptyset)} \leq \frac{1}{e}.$$

Submodularity of Entropy and its Degree 2 Approximation

Proposition 6. *The entropy function $H : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ is submodular.*

Proof. We want to show $H(T \cup \{x\}) - H(T) \leq H(S \cup \{x\}) - H(S)$ whenever $S \subseteq T$. Let $T' = T - S$. We have $I(T'; x|S) \geq 0$ because (conditional) mutual information is always non-negative. This gives us

$$\begin{aligned} H(x|S) - H(x|ST') &\geq 0 \\ H(S \cup \{x\}) - H(S) &\geq H(S \cup T' \cup \{x\}) - H(S \cup T') \\ H(S \cup \{x\}) - H(S) &\geq H(T \cup \{x\}) - H(T) \end{aligned}$$

□

Proposition 7. *The degree 2 approximation $\widetilde{H}_2 : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ of H , given by $\widetilde{H}_2(S) = \sum_{x_i \in S} H(x_i) - \sum_{x_i, x_j \in S} I(x_i; x_j)$ is submodular.*

Proof. Note that $\widetilde{H}_2(S \cup \{x\}) - \widetilde{H}_2(S) = H(x) - \sum_{x_i \in S} I(x; x_i)$. Recall that $I(\cdot; \cdot)$ is non-negative. Clearly, $\widetilde{H}_2(T \cup \{x\}) - \widetilde{H}_2(T) \leq \widetilde{H}_2(S \cup \{x\}) - \widetilde{H}_2(S)$ whenever $S \subseteq T$. □

Submodularity of Mutual Information and its Degree 2 Approximation

In general, mutual information is not submodular. As we saw in Example 1, the “value” of a feature is not necessarily diminishing. In that particular example we had $I(x_2; y) - I(\emptyset; y) = 0$, but $I(\{x_1, x_2\}; y) - I(x_1; y) = 1$; crudely put, a greedy algorithm only appreciates the value of x_2 when it already knows x_1 . This specific phenomenon is a direct consequence of synergy and we explain this relation rigorously in this section.

Proposition 8. *If for all $x \notin T \supseteq S$, we have $I(T - S; x; y|S) \geq 0$, then the set function $f : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ given by $f(S) = I(S; y)$ is submodular.*

Proof. As before, let $T' = T - S$. We have $I(T'; x; y|S) = I(T'; x; y) - I(S; T'; x; y)$. We can now rewrite $I(T - S; x; y|S) \geq 0$ as $I(S; T'; x; y) - I(T'; x; y) \leq 0$. Adding many terms to each side of the inequality, we get

$$\begin{aligned} &I(S; y) + I(T'; y) + I(x; y) - I(S; x; y) \\ &- I(T'; x; y) - I(S; T'; y) + I(S; T'; x; y) \\ &- I(S; y) - I(T'; y) + I(S; T'; y) \\ &\leq I(S; y) + I(x; y) - I(S; x; y) - I(S; y) \end{aligned} \tag{5}$$

Therefore,

$$I(T \cup \{x\}; y) - I(T; y) \leq I(S \cup \{x\}; y) - I(S; y) \tag{6}$$

□

The analogous result for the degree approximation $\widetilde{I}_2$ of I given by $\widetilde{I}_2(S; y) = \sum_{x_i \in S} I(x_i; y) - \sum_{x_i, x_j \in S} I(x_i; x_j; y)$ (which is the function that the CIFE algorithm maximizes) is a little easier to describe, since we ignore higher order interactions. We only need to assume that there is no two-way synergy.

Proposition 9. *If for all x_i, x_j , we have $I(x_i; x_j; y) \geq 0$, then the set function $f : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ given by $f(S) = \sum_{x_i \in S} I(x_i; y) - \sum_{x_i, x_j \in S} I(x_i; x_j; y)$ is submodular.*

Proof. Note that $\widetilde{I}_2(S \cup \{x\}; y) - \widetilde{I}_2(S; y) = I(x; y) - \sum_{x_i \in S} I(x; x_i; y)$. Because we have assumed $I(x_i; x_j; y) \geq 0$ for all x_i, x_j , we have $f(T \cup \{x\}) - f(T) \leq f(S \cup \{x\}) - f(S)$ whenever $S \subseteq T$. □

Guarantees for Feature Selection

It is well known that $I(\cdot; y)$ and $H(\cdot)$ are non-decreasing set functions. We immediately have the following corollaries.

Corollary 10. *The greedy search algorithm to maximize $H(S)$ over $S \subseteq \mathcal{X}$ with $|S| \leq k$ for some user-specified k yields a $(1 - 1/e)$ -approximation (i.e., $H(S_{\text{greedy}}) \geq (1 - 1/e)H(S_{\text{optimal}})$).*

Proof. Follows by Theorem 5 and Proposition 6. □

Corollary 11. *If for all $x \notin T \supseteq S$, we have $I(T - S; x; y|S) \geq 0$, then the greedy search algorithm to maximize $I(S; y)$ over $S \subseteq \mathcal{X}$ with $|S| \leq k$ for some user-specified k yields a $(1 - 1/e)$ -approximation (i.e., $I(S_{\text{greedy}}; y) \geq (1 - 1/e)I(S_{\text{optimal}}; y)$).*

Proof. Follows by Theorem 5 and Proposition 8. $\square$

Unfortunately, our approximations $\tilde{I}_2(\cdot; y)$ and $\tilde{H}_2$ of $I(\cdot; y)$ and H functions are not non-decreasing, even though $I(\cdot; y)$ and H themselves are. This leaves us with a slightly weaker result for optimizing $\tilde{I}_2$ and $\tilde{H}_2$. The following is a generalization of Theorem 5.

Theorem 12 ([27]). *Consider the maximization problem*

$$\arg \max_{S \subseteq \mathcal{X}, |S| \leq k} f(S).$$

If f is a submodular function and $f(S \cup \{x\}) - f(S) \geq -\theta$ for all $S \subseteq \mathcal{X}$ and $x \in \mathcal{X}$, we have

$$\frac{f(S_o) - f(S_g)}{f(S_o) - f(\emptyset) + k\theta} \leq \frac{1}{e},$$

where S_o is an optimal set and S_g is the set found via a greedy search.

Corollary 13. *If $\tilde{H}_2(S \cup \{x\}) - \tilde{H}_2(S) \geq -\theta$ for all $S \subseteq \mathcal{X}$ and $x \in \mathcal{X}$, then the greedy solution $S_g \subseteq \mathcal{X}$ to $\arg \max_{S \subseteq \mathcal{X}, |S| \leq k} \tilde{H}_2(S)$ relates to the optimal solution S_o as follows:*

$$\tilde{H}_2(S_o) - \tilde{H}_2(S_g) \leq \frac{\tilde{H}_2(S_o) + k\theta}{e}$$

Corollary 14. *If $I(x_i; x_j; y) \geq 0$ for all $x_i, x_j \in \mathcal{X}$ and $\tilde{I}_2(S \cup \{x\}; y) - \tilde{I}_2(S; y) \geq -\theta$ for all $S \subseteq \mathcal{X}$ and $x \in \mathcal{X}$, then the greedy solution $S_g \subseteq \mathcal{X}$ (i.e., the output of the CIFE algorithm) to $\arg \max_{S \subseteq \mathcal{X}, |S| \leq k} \tilde{I}_2(S; y)$ relates to the optimal solution S_o as follows:*

$$\tilde{I}_2(S_o; y) - \tilde{I}_2(S_g; y) \leq \frac{\tilde{I}_2(S_o; y) + k\theta}{e}$$

In practical terms, we can interpret corollary 14 as follows: if we know that shared redundancy between any pair of features outweigh synergies and that our degree 2 approximation to mutual information is monotone, then the greedy algorithm will output an $1 - \frac{1}{e}$ (≈ 0.632) approximation (i.e., the score of the set returned by a greedy search will be at least 63.2% of the score of the optimal set). If we know that the degree 2 approximation is not monotone but can bound how much it might decrease by at any step, we can still recover a (weaker) guarantee.

C Practical Implementation Issues

There are numerous issues that arise in the implementation of the information-theoretic algorithms in this paper. These issues warrant further discussion as they have a significant impact on the output of these algorithms. We present our approach to addressing these issues in this section. Optimizations that improve runtime but do not affect the output of the algorithm are presented under **Efficient Algorithms**.

C.1 Binning Technique

All our discussion thus far has assumed that the x_i 's are discrete random variables. When working with continuous data, it is common to discretize the data prior to running information theoretic methods. Although the datasets we have used are inherently discrete (with entries in $\mathbb{Z}_{\geq 0}$), there is a need to limit the number of values that entries can take. To understand why this is so important, suppose that the x_i 's take integer values from 0 to $k - 1$. In order to empirically compute $I(x_i; x_j; y)$, we need to approximate $P(x_i, x_j, y)$. Without any additional assumptions on the joint probability distribution, we are left to approximate this distribution by counting occurrences of each possible combination, of which there are k^3 . If our dataset has n observations, we need $n \gg k^3$ to have a reasonable approximation of $I(x_i; x_j; y)$. It is also important not to collapse too many values together as that would result in

very poor resolution stifling our ability to observe significant differences. We recommend targeting $k = 5$ discrete values as a reasonable default.

The process of choosing bins has a surprisingly large impact on the performance of the algorithms in this paper. For single cell RNA-sequencing, we propose a recursive quantile transformation that adjusts for the sparsity of the data. To motivate the need for our modification to the usual quantile transform, suppose we would like $k = 5$ bins for each feature. Because scRNA-seq datasets have a large number of zero entries (in some cases $> 90\%$ of entries are zero), the $\frac{1}{5}$ -th, $\frac{2}{5}$ -th, $\frac{3}{5}$, and $\frac{4}{5}$ -th quantiles could all be 0. This leaves us with only two meaningful bins. To avoid this problem, we recommend a recursive strategy described in Algorithm 2 for selecting right endpoints for k bins. On our experiments, we perform this transformation on each feature individually to account for the different distributions of values across features.

Algorithm 2 Recursive-quantile transform

```

1: procedure QUANTBINS( $C, k$ )
2:   if  $k \leq 1$  then
3:     return ( $\max C,$ )
4:   end if
5:    $q \leftarrow \frac{1}{k}$ -th quantile of  $C$ 
6:    $C' \leftarrow \{c \in C \mid c > q\}$ 
7:   return concatenate( $(q,$ QuantBins( $C', k - 1$ ))
8: end procedure

```

C.2 Multiclass vs Binary Feature Selection Methods

The mutual information based feature selection methods are approximations of $\arg \max_{|S| \leq n} I(S; y)$. Unlike many of the differential expression methods, the mutual information based algorithms allow for y to take more than two values (i.e., they are *multiclass* rather than *binary* methods). This, in theory, allows us to select features at once, rather than having to select them for every class label (one-vs-rest) or every pair of class labels (one-vs-one).

However, as we mentioned in the **Results** section, it is also possible to run these algorithms as binary methods by letting $\bar{y}_t = \mathbb{1}_{y=t}$ and treating $\bar{y}_t$ as the class labels. This gives us the opportunity both to compare binary differential expression methods to binary mutual information based methods and to evaluate strengths and weaknesses of multiclass methods vis-a-vis binary methods.

We saw that for almost all datasets (the only exception being the Zheng dataset for a small number of markers), turning the mutual information based methods into binary methods led to a significant improvement in performance. We describe some possible ways in which these differences in performance can be explained. Note that unlike in the **Analysis of Existing Methods** section, this is not an exhaustive list of shortcomings. Instead, these examples demonstrate ways in which these approaches differ.

Sometimes features can be informative about multiple target labels. Unless the feature selection algorithm used is inherently multi-class, we cannot detect features that may have a weak signal about any particular target label but a strong signal that helps discern between sets of target labels.

Example 2. Let b_1, b_2 be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $y = b_1 b_2$ and let $x_i = \mathbb{1}_{i=(b_1 b_2)_{(b)}}$ for $i \in \{0, 1, \dots, 3\}$. Here the subscript (b) denotes that we are looking at the binary representation of the number. For example,

$$x_2 = \begin{cases} 1 & \text{if } b_1 = 1, b_2 = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Finally, let $x_4 = b_1$ and $x_5 = b_2$.

It is easy to see now that $I(x_4; y) = 1 = I(x_5; y)$. However, for $i \leq 3$ we have $I(x_i; y) = \frac{1}{4} \log(4) + \frac{3}{4} \log(\frac{4}{3}) \approx 0.811$. That is to say, multiclass methods will correctly identify x_4 and x_5 as being very informative about y .

On the other hand, if our class labels were binary, for $i \leq 3$ we have $I(x_i; \bar{y}_i) = \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3} \approx 0.811$. However, $I(x_4; \bar{y}_i) = H(x_4) + H(\bar{y}_i) - H(x_4 \bar{y}_i) = 1 + (\frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3}) - (\frac{1}{2} \log(2) + \frac{1}{2} \log(4)) \approx 0.311$. A binary method would therefore choose $x_0, \dots, x_3$ before x_4 and x_5 .

It must be noted that if we run a binary method on the example above, we will eventually choose $\{x_0, x_1, x_2, x_3\}$, which, together, are as informative as $\{x_4, x_5\}$. The following modification demonstrates a situation in which this is not true.

Example 3. Let $c_1, c_2, \dots$ be independent Bernoulli trials with probability p for some constant $p \in [0, \frac{1}{2}]$ (these will introduce noise in x_0, x_1, x_2, x_3). In other words, $\mathbb{P}(c_i = 1) = p$ and $\mathbb{P}(c_i = 0) = 1 - p$. Now follow the previous example, except let $x_i = c_i \oplus \mathbb{1}_{i=(b_1 b_2)_{(b)}}$ for $i \in \{0, 1, \dots, 3\}$. Recall $\oplus$ is the XOR operator: $a \oplus b = a$ if $b = 0$ and $a \oplus b = \neg a$ if $b = 1$.

It is clear that for small values of p , a binary method will still choose $\{x_0, x_1, x_2, x_3\}$ instead of $\{x_4, x_5\}$, even though y is a deterministic function of x_4 and x_5 but not of $x_0, \dots, x_3$. With a sufficiently large value of p (i.e., where there is enough noise in $x_0, \dots, x_3$), even a binary method will choose x_4 and x_5 .

Recall that objectives like CIFE and JMI only consider low-degree interactions between features. Therefore, it is easy to under/over-estimate redundancies. By measuring relevance ($I(x_i; y)$ term) and redundancies ($I(x_i; x_j; y)$ terms) only in the context of a specific class label, binary methods are less vulnerable to accumulating errors in approximations.

We demonstrate this idea with two example. In the first example, we consider a univariate feature selection method (e.g., MIM) that does not penalize any redundancy (i.e., it underestimates redundancy) and outline a case where the binary version outperforms the multiclass version. In the second example, we demonstrate a situation in which a bivariate feature selection method (e.g., CIFE) overpenalizes redundancy in the multiclass version and the binary version yields a better selection of features.

Example 4. Let b_1, b_2, b_3 be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $y = b_1 b_2 b_3$ and let $t_i = \mathbb{1}_{i=(b_1 b_2 b_3)_{(b)}}$ for $i \in \{0, 1, \dots, 7\}$. We will use the t_i 's to define features. Let $x_i = t_0 t_1 t_2 t_i$ for $i \in \{3, \dots, 7\}$. Let $x_8 = t_4 t_5$ and $x_9 = t_6 t_7$. A univariate multiclass method (e.g., MIM) will associate a higher score to $x_3, \dots, x_7$ because they are individually more informative about y than x_8 and x_9 . However, a binary method (when considering the class label $y = 100$ or $y = 101$) will assign the same scores to x_8, x_4 , and x_5 . Similarly, it would assign the same scores to x_9, x_6 , and x_7 when considering $y = 110$ and $y = 111$. To construct an example where the binary method will definitely pick x_8 and x_9 while the multiclass method won't, we can add noise (as in Example 3) to features $x_3, \dots, x_7$.

In the previous example, we saw how binary methods, in searching for information specific to individual class labels, are able to avoid traps of redundant information. We now demonstrate an example in which binary methods avoid overpenalizing redundancy.

Example 5. Let b_1, b_2 be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let c_2, c_3 be independent Bernoulli trials with probability $p \in [0, \frac{1}{2}]$. Let $y = b_1 b_2$ and let $t_i = \mathbb{1}_{i=(b_1 b_2)_{(b)}}$ for $i \in \{0, 1, \dots, 3\}$. We will use the t_i 's to define features. Let $x_1 = t_0 t_1 t_2$, $x_2 = t_0 t_1 t_3$, $x_3 = t_2 \oplus c_2$, and $x_4 = t_3 \oplus c_3$.

If we use a bivariate multiclass method (e.g., CIFE), without loss of generality we select x_1 at the first iteration (analogous argument if we selected x_2 first). Because $I(x_2; x_1; y)$ is significant, the penalty on x_2 (with CIFE, say) would be large enough that we would pick x_4 instead of x_2 , which is more informative. The binary version of CIFE would not have this problem.

We posit that the Paul dataset, with its high cluster resolution (numerous clusters that cannot be separated easily) resembles Example 5 and believe that this (at least partially) explains the stark contrast in the reclassification accuracy between the multiclass and binary algorithms (especially CIFE).

D Efficient Algorithms

D.1 Computing Entropy for Sparse Matrices

In this section, we outline a fast computation of entropy for sparse matrices. As a result of Proposition 1, this algorithm can be used to efficiently compute various information-theoretic objective functions for feature selection (especially low-dimension approximations, such as CIFE and JMI). In this section, we will use zero-indexing rather than one-indexing (i.e., the first element of an array A is $A[0]$).

Compressed Sparse Column (CSC) is a sparse matrix format that is optimized for column operations (e.g., slicing by columns, iterating over non-zero entries in a given column, etc.). It is defined as follows. For an $n \times m$ matrix M , the CSC format stores three arrays:

- X , the values of the non-zero entries stored in column major order.
- R , the row index of the non-zero entries in X (i.e., X and R have the same length).
- C , an array of length $m + 1$ where $C[j + 1] - C[j]$ is the number of non-zero entries in column j and $C[0] = 0$.

In other words, for any s such that $C[j] \leq s < C[j+1]$, we have $X[s]$ is the value of a non-zero entry in row $R[s]$ and column j .

Assume all values in X come from $\{0, 1, \dots, k-1\}$ (i.e., there are k distinct values in X). To efficiently compute the (empirical) entropy of an individual column j of an $n \times m$ matrix M , we follow Algorithm 3.

Algorithm 3 Entropy of a Single Column

```

1: procedure ENTROPY( $M = (X, R, C), j$ )
2:   counts  $\leftarrow$  array (length  $k$ )
3:   counts[0]  $\leftarrow n$ 
4:   for  $C[j] \leq s < C[j+1]$  do
5:     counts[0]  $\leftarrow$  counts[0] - 1
6:     counts[X[s]]  $\leftarrow$  counts[X[s]] + 1
7:   end for
8:    $H \leftarrow 0$ 
9:   for nonzero  $c$  in counts do
10:     $H \leftarrow H + (\frac{c}{n}) \log_2(\frac{c}{n})$ 
11:   end for
12:   return  $H$ 
13: end procedure

```

We often need to compute the entropy of two or more columns. A trivial way to overcome this problem is to combine multiple columns into one column. One way to combine r columns is to write each entry as an r -tuple. This combined column will retain the sparse structure as the non-zero entries will correspond to rows in which at least one entry was non-zero.³

However, the process of computing the tuple representation of multiple can sometimes be repetitive. For example, to compute $\arg \max_{x_i} I(x_i; x_{j_1}; x_{j_2}; \dots; x_{j_r})$, we need to compute $H(x_i; x_{j_1}; x_{j_2}; \dots; x_{j_r})$ for all $x_i \in \mathcal{X}$. In such cases, it make sense to precompute the tuple representation of $x_{j_1}, \dots, x_{j_r}$.

However, in most cases, the column of class labels y tends not to be a sparse column. Combining y into the tuple representation means that most entries in the column become non-zero. This voids the benefits of exploiting the sparse structure of M . We can address these concerns by precomputing some information and vectorizing the entropy computation as in Algorithm 4. The ENTROPYWRT (“entropy with respect to”) takes a list of columns $(x_{j_1}, x_{j_2}, \dots, x_{j_r}, y)$ and returns the vector $[H(x_i x_{j_1} \dots x_{j_r} y)]_{i=0}^{m-1}$. Pseudocode for ENTROPYWRT when the y column is not in the list of columns can be written with a trivial modification to Algorithm 4. Here we assume y has entries in $0, 1, \dots, n_{\text{labels}} - 1$.

The time complexity of ENTROPYWRT is $O(n + mp + q + mk^{r+1})$ where p is the number of rows with non-zero entries in columns $j_1, j_2, \dots, j_r$, and q is the number of non-zero entries in M . Typically, ENTROPYWRT will be called numerous times, in which case ycounts should be precomputed once, rather than on each call to ENTROPYWRT. In this case, the time complexity is $O(mp + q + mk^{r+1})$. Also note that when we use ENTROPYWRT, we will typically fix k beforehand (depending of discretization strategy) to be about 5. For CIFE and JMI algorithms, we have $r = 2$ because we will only ever compute the joint entropy of three columns or less. In these cases, the time complexity is $O(mp + q)$.

D.2 Efficient Optimization of CIFE and JMI

The CIFE and JMI algorithms compute $\arg \max_{x_i \notin S} J(x_i)$ at each iteration. If done naively, computing $J(x_i)$ can take longer as S grows because $J_{\text{cife}}(x_i) = I(x_i; y) - \sum_{x_j \in S} I(x_i; x_j; y)$ and $J_{\text{jmi}}(x_i) = I(x_i; y) - \frac{1}{|S|} \sum_{x_j \in S} I(x_i; x_j; y)$ have $|S| + 1$ terms. The computational complexity to select t features is $O(mt^2 f)$ where f is the time to compute $I(x_i; x_j; y)$ and is a function of n, k , and sparsity parameters of columns in X . A minor modification to vectorize this process can lower the complexity to $O(mtf)$, as shown in Algorithm 5.

Note that $I(x_s; x_j; y) = I(x_j; y) + H(x_s) - H(x_s, x_j) - H(x_s, y) + H(x_s, x_j, y)$. Since $I(x_j; y)$ was computed as **basescore**, and $H(x_s) - H(x_s, y)$ is constant with respect to j , the only two calls to ENTROPYWRT are for $H(x_s, x_j)$ and $H(x_s, x_j, y)$.

Under certain conditions, this process can be made even faster with a strategy adapted from the fast algorithm for CMIM in [28]. In the aforementioned paper, the authors propose a objective function that can be rewritten as

---

³In practice, is can be faster to represent $(a_0, a_1, \dots, a_{r-1})$ as $a_0 + a_1 k + \dots + a_{r-1} k^{r-1}$ where $k = \max_i(a_i) + 1$, but we will avoid this conversion for the sake of readability. See our source code for such an implementation.

Algorithm 4 Entropy of Multiple Columns (all but one fixed)

```

1: procedure ENTROPYWRT( $M = (X, R, C), j_1, j_2, \dots, j_r, y$ )
2:    $H \leftarrow$  zero-filled array (length  $m$ )
3:    $P \leftarrow$  sparse tuple representation of  $[M_{ij}]_{j \in \{j_1, \dots, j_r\}}$ 
4:    $ycounts \leftarrow$  array (length  $n_{labels}$ )
5:   for  $yval$  in  $y$  do
6:     increment  $ycounts[yval]$ 
7:   end for
8:   for  $0 \leq j < m$  do
9:      $counts \leftarrow$  Map:  $(r+2)$ -tuples  $\rightarrow \mathbb{Z}_{\geq 0}$ 
10:    for  $0 \leq yval < n_{labels}$  do
11:       $counts[(yval, 0, \dots, 0)] \leftarrow ycounts[yval]$ 
12:    end for
13:     $curind \leftarrow 0$ 
14:     $colindices \leftarrow$  where  $(j \neq 0)$ 
15:     $colPindices \leftarrow$  where  $(P \neq 0)$ 
16:    while exists  $x \in colPindices \cup colindices, x > curind$  do
17:       $curind \leftarrow \min \{ x : x \in colPindices \cup colindices, x > curind \}$ 
18:      decrement  $counts[(y[curind], 0, \dots, 0)]$ 
19:      increment  $counts[(y[curind], P[curind], M[curind, j])]$ 
20:    end while
21:    for  $c$  in  $counts, c > 0$  do
22:       $H[j] \leftarrow H[j] + (\frac{c}{n}) \log_2(\frac{c}{n})$ 
23:    end for
24:  end for
25:  return  $H$ 
26: end procedure

```

Algorithm 5 Vectorization of CIFE/JMI

```

1: procedure FEATURESELECTION( $X, t, method$ )
2:    $S \leftarrow \emptyset$ 
3:    $basescore \leftarrow [I(x_i; y)]_{i=0}^{m-1}$ 
4:    $penalty \leftarrow$  zero-filled vector (length  $m$ )
5:    $score \leftarrow basescore$ 
6:   for  $0 \leq \ell < t$  do
7:      $s \leftarrow \arg \max_{i \notin S} score[i]$ 
8:      $penalty \leftarrow penalty + [I(x_s; x_j; y)]_{j=0}^{m-1}$ 
9:     if  $method = \text{"CIFE"}$  then
10:       $score \leftarrow basescore - penalty$ 
11:     else if  $method = \text{"JMI"}$  then
12:       $score \leftarrow basescore - \frac{1}{|S|} penalty$ 
13:     end if
14:   end for
15: end procedure

```

$J_{\text{cmim}}(x_i) = I(x_i; y) - \max_{x_j \in S} I(x_i; x_j; y)$. They improve the efficiency of the naive algorithm by updating $J_{\text{cmim}}(x_i)$ (i.e., checking if the penalty term has increased since more features were selected into S) only $J(x_i) > J(x_j)$ for all $j < i$. This prevents some unnecessary evaluations of $I(x_i; x_j; y)$. This trick hinges on the monotonicity of $\max_{x_j \in S} I(x_i; x_j; y)$ with respect to S . In our case $\sum_{x_j \in S} I(x_i; x_j; y)$ is monotone if $I(x_i; x_j; y) \geq 0$. If we do make this assumption (as we did in Proposition 9), we could make a similar optimization. In fact, we believe such an optimization would be best implemented using a priority queue.

E Analysis of Existing Methods

Most of these algorithms are affected by three broad theoretical weaknesses. We analyse these problems in the context of existing algorithms and present simple examples where they arise. Obviously, many assumptions from [20] that we discussed under **Previous work** are violated in these simple examples. The aim of these examples, however, is to present the theoretical weaknesses of existing algorithms in information-theoretic terms. This will motivate our approach to mutual information based feature selection⁴.

E.1 Diminishing penalty on redundancy

The mRMR algorithm in [21] is a prominent example of an objective that progressively diminishes the penalty on redundancy. Specifically,

$$J_{\text{mrmr}}(x_i) = I(x_i; y) - \frac{1}{|S|} \sum_{x_j \in S} I(x_i; x_j). \quad (7)$$

The JMI algorithm in [22] also has a factor of $\frac{1}{|S|}$ on its redundancy term:

$$J_{\text{jmi}}(x_i) = I(x_i; y) - \frac{1}{|S|} \sum_{x_j \in S} (I(x_i; x_j) - I(x_i; x_j|y)). \quad (8)$$

When $|S|$ is large, the redundancy penalty is so small that these algorithms, when asked for a large set of features (but still considerably smaller than the total number of features), will yield similar outputs as MIM. We demonstrate this empirically in Tables 1, 2, 4, and 3.

In practice, the $\frac{1}{|S|}$ factor in algorithms such as mRMR and JMI serves two purposes. First, the $\frac{1}{|S|}$ factor accounts for shared redundancy between multiple variables (see examples under **Ignoring Higher-degree Interactions Between Features**). Second, it acts to mitigate the accumulating noise in empirical estimations of $I(x_i; x_j)$, because even if x_i and x_j are independent, the empirical estimation of their mutual information might be positive. In either case, the $\frac{1}{|S|}$ factor is a somewhat arbitrary approach that suppresses both signal and noise and, in doing so, impedes our ability to detect true redundancies.

Example 6. Let $b_1, b_2, \dots$ be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $x_1 = b_1 b_2 \dots b_5$ (a binary concatenation of bits), $x_2 = b_6 b_7 \dots b_{10}$, $x_3 = b_{11}$, and $x_4 = b_1 b_2 b_3$. Suppose $y = b_1 b_2 \dots b_{11}$. Algorithms such as mRMR and JMI would select x_1 and x_2 first. At this point, $J(x_3) = 1$ and $J(x_4) = 3 - \frac{1}{2} \cdot 3 = 1.5$. But clearly, x_4 contributes no new information, while x_3 does.

E.2 Penalizing Irrelevant Redundancy

Algorithms like mRMR and MIFE penalize all redundancy, even if the redundancy does provide any information about y . For example, if x_i and x_j are highly correlated, then it is possible that $I(x_i; x_j) > \max(I(x_i; y), I(x_j; y))$. This is demonstrated in the following example.

Example 7. Let $b_1, b_2, \dots$ be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $x_1 = b_1 b_2 b_3$, $x_2 = b_1 b_2 b_4$, $x_3 = b_5$, and $y = b_3 b_4$. Without loss of generality, assume the algorithm chooses x_1 first. Now $I(x_2; y) = 1$, but $I(x_1; x_2) = 2$. Such algorithms might choose x_3 instead of x_2 , because $I(x_1; x_2)$ is large, but the redundancy between x_1 and x_2 is not relevant to y and should not be penalized.

⁴It should be noted, however, that no algorithm can truly overcome all the weaknesses below for that will require being able to infer a joint probability distribution over a large number of random variables which is meaningless without an enormous number of samples and also computationally intractable.

E.3 Ignoring Higher-degree Interactions Between Features

Features can say both more and less about the target labels together than the sum of their parts. All features covered by Brown, et al.’s framework (objectives of the form (2)) are unable to detect higher-order synergies and redundancies because they only detect two-way interactions between features.

Example 8 (Synergy). Let $b_1, b_2, \dots$ be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $x_1 = b_1$ and $x_2 = b_2$. If $y = b_1 \oplus b_2$ (where $\oplus$ denotes the XOR operation on bits), then $I(x_1; y) = I(x_2; y) = 0$, but $I(x_1 x_2; y) = 1$. This can be understood in terms of the multivariate mutual information $I(x_1; x_2; y) = -1$ (which we saw in the [Series Expansion of Mutual Information](#) section)⁵

Example 9 (Shared Redundancy). Let $b_1, b_2, \dots$ be random variables chosen from $\{0, 1\}$ independently with uniform probability. Let $x_1 = b_1 b_4$, $x_2 = b_2 b_4$, $x_3 = b_3 b_4$, $x_4 = b_5$, and $y = b_1 b_2 b_3 b_4$. Without loss of generality, after two iterations, an algorithm of the form (2) will likely have $S = \{x_1, x_2\}$. Now, $I(\{x_1, x_2, x_3\}, y) = 4$ and $I(\{x_1, x_2, x_4\}, y) = 3$, but objectives of the form (2) penalize x_3 twice for its redundant bit (b_4).

F Problems with mRMR

A false theorem in [21, §2.3] states that maximizing the mRMR objective in a greedy search is “equivalent” to maximizing the CMI objective in a greedy search. The authors of this paper believe that the reasonable interpretation of “equivalent” is that given the same input, the two algorithms will return the same output. In this case, their claim (and the proof thereof) is incorrect. We have already seen in Examples 6 and 7 cases where the mRMR objective makes different choices than the CMI objective. To understand why these examples are counterexamples to the claim in [21], we walk through their proof.

For notational convenience, we let $\mathcal{X}$ be the set of all features, $S = S_{m-1}$ be the set of $m-1$ features selected after $m-1$ steps, $D(x_i) = I(x_i; y)$ be the *relevance* of feature x_i and $R(x_i) = \frac{1}{|S|} \sum_{x_j \notin S} I(x_i; x_j)$ be the *redundancy* of feature x_i . They define a quantity $J : \mathcal{P}(\mathcal{X}) \rightarrow \mathbb{R}$ and show that $I(S_{m-1} \cup \{x_i\}; y) = J(S_{m-1}, x_i, y) - J(S_{m-1}, x_i)$. Fixing S_{m-1} , they then show that the first term, $J(S_{m-1}, x_i, y)$, is maximized when $D(x_i) = I(x_i; y)$ is maximized, and the second term, $J(S_{m-1}, x_i)$ is minimized when $R(x_i) = \frac{1}{|S|} \sum_{x_j \notin S} I(x_i; x_j)$ is minimized. They then falsely conclude that

$$\arg \max_{x_i \notin S_{m-1}} I(S_{m-1} \cup \{x_i\}; y) = \arg \max_{x_i \notin S_{m-1}} D(x_i) - R(x_i). \quad (9)$$

⁵In cases of extreme synergy, such as the current example (where individual feature provide no information on their own) even the J_{cmi} objective will fail to choose an optimal set of features as a result of the shortsightedness of a greedy approach. In less extreme cases, it is still beneficial for the objective function to recognize such synergy).