BMC Bioinformatics

METHODOLOGY ARTICLE

Open Access

HARVESTMAN: a framework for hierarchical feature learning and selection from whole genome sequencing data

Trevor S. Frisby[1†], Shawn J. Baker[1†], Guillaume Marçais[1], Quang Minh Hoang[2], Carl Kingsford[1*] and Christopher J. Langmead[1*]

*Correspondence: carlk@cs.cmu.edu; cjl@cs.cmu.edu
†Trevor S. Frisby and Shawn J. Baker have contributed equally to this work
[1] Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA, USA
Full list of author information is available at the end of the article

Abstract

Background: Supervised learning from high-throughput sequencing data presents many challenges. For one, the curse of dimensionality often leads to overfitting as well as issues with scalability. This can bring about inaccurate models or those that require extensive compute time and resources. Additionally, variant calls may not be the optimal encoding for a given learning task, which also contributes to poor predictive capabilities. To address these issues, we present HARVESTMAN, a method that takes advantage of hierarchical relationships among the possible biological interpretations and representations of genomic variants to perform automatic feature learning, feature selection, and model building.

Results: We demonstrate that HARVESTMAN scales to thousands of genomes comprising more than 84 million variants by processing phase 3 data from the 1000 Genomes Project, one of the largest publicly available collection of whole genome sequences. Using breast cancer data from The Cancer Genome Atlas, we show that HARVESTMAN selects a rich combination of representations that are adapted to the learning task, and performs better than a binary representation of SNPs alone. We compare HARVESTMAN to existing feature selection methods and demonstrate that our method is more *parsimonious*—it selects smaller and less redundant feature subsets while maintaining accuracy of the resulting classifier.

Conclusion: HARVESTMAN is a hierarchical feature selection approach for supervised model building from variant call data. By building a knowledge graph over genomic variants and solving an integer linear program , HARVESTMAN automatically and optimally finds the right encoding for genomic variants. Compared to other hierarchical feature selection methods, HARVESTMAN is faster and selects features more parsimoniously.

Keywords: Feature selection, Hierarchical feature spaces, Knowledge graphs, Integer linear programming, Machine learning

Background

Introduction

Supervised learning from high-throughput sequencing data presents many challenges [1, 2]. First among these is the curse of dimensionality, which predisposes learning algorithms to overfitting and imposes barriers to scalability [3]. A second critical challenge is that raw variant calls may not be the optimal feature encoding for a given learning task [4]. The most informative, and biologically relevant encoding of a given variant may be at a higher level of organization, such as a perturbation in a particular exon, transcript, or pathway. This paper addresses both challenges by introducing Harvestman, a method that automatically identifies a non-redundant set of relevant features chosen from a hierarchy of biological encodings of the raw variants.

Strategies for finding effective representations of the data include feature engineering methods, which apply domain knowledge to define features *a priori*, and feature learning methods, which apply supervised or unsupervised learning algorithms to the task. Feature engineering is largely a manual process, but for that reason it is likely to produce encodings that are meaningful to domain experts. Feature learning methods are largely automated, but may produce features that are difficult to understand [5, 6]. Harvestman employs a hybrid approach to finding an effective encoding of the data. It first constructs a hierarchy of potential representations for each variant. We refer to that hierarchy as the *knowledge graph*. Our knowledge graph is derived from existing genomic annotations and ontologies, to ensure that each putative encoding is biologically relevant, but the Harvestman framework can also incorporate alternative, user-defined knowledge graphs.

Traditional approaches to mitigating the curse of dimensionality fall into two basic categories: feature extraction methods and feature selection methods [7]. Feature extraction methods find a new lower-dimensional feature subspace using information about the current feature space, usually in an unsupervised fashion. Hence, the extracted features are not present within the original data, but rather constructed from it. Among the most common feature extraction techniques is Principle Components Analysis. A downside with such feature extraction methods is that they may become unreliable when the vast majority of the input features are irrelevant to the prediction task, as is often the case with genomic data [8]. Moreover, the induced features are typically linear or non-linear combinations of the input covariates. Such representations can be difficult to understand.

Feature selection methods, in contrast, explicitly select informative subsets of covariates [9], and do not change the way those features are encoded. Thus, if the given features have an intuitive interpretation, the chosen subset will as well. Feature selection techniques are typically categorized as being either a filter, a wrapper, or an embedded method [10, 11]. Filter-based methods (e.g., ReliefF [12]) rank individual features according to some scalar quantity, such as the mutual information between the feature and the label. The user then selects the top k features for subsequent model-building. Wrapper methods (e.g., CFS [13]) explicitly rank various subsets of features, with the highest ranking subset then chosen. Embedded methods select feature subsets during model building (e.g., by applying $L1$-regularization).

HARVESTMAN employs supervised hierarchical feature selection under a wrapper-based regime, as it solves an optimization problem over the knowledge graph designed to select a small and non-redundant subset of maximally informative features. In this way, HARVESTMAN automatically learns the best feature encoding while performing feature selection. Critically, we will show that HARVESTMAN is more parsimonious than competing hierarchical feature selection strategies, meaning HARVESTMAN selects fewer features without sacrificing classifier accuracy.

HARVESTMAN and related work

Traditional feature selection strategies are not intended for hierarchical feature spaces, where the parent of a given feature represents an alternative encoding of the same underlying observation(s). HARVESTMAN builds on recent techniques [14–16] for solving the hierarchical feature selection problem. Let $V = \{v_1, \ldots, v_m\}$ be a set of features (nodes) and let $G = (V, E)$ be a directed acyclic graph over those features. Here, the topology of G encodes the hierarchical relationships among the features (if any) such that a directed edge from node v_i to v_j implies that v_j represents a higher level abstraction of node v_i. In the current paper, G is the knowledge graph that encodes the potential interpretations (i.e. feature encodings) for a given set of variant calls (see Fig. 5). Naturally, each vertex/feature in G will be correlated with its ancestors and descendants, and so it is important to identify and eliminate redundant features. The (supervised) hierarchical feature selection problem is: given G and a set of labeled training instances, select the most informative and least redundant subset of V.

There are many different hierarchies that might be used to construct the knowledge graph from genomic data. Perhaps the most obvious one corresponds to the overlap between genomic loci known to play a functional or regulatory role. Here, the terminal nodes of G might correspond to specific positions within the genome. Internal nodes correspond to higher levels of annotation that denote specific regions in the genome, such as transcripts or genes. Alternatively, one might use the existing Gene Ontology (GO) hierarchies [17, 18] to define the knowledge graph. The GO graphs describe the cellular components, molecular functions, and biological processes associated with each gene and its products. Unlike genome annotations that relay structural information about specific genomic regions, GO annotations provide broader information about the systems and processes that changes to these regions may affect. Combining the knowledge contained by multiple annotation types thus captures a fuller picture for genomic variation. HARVESTMAN's strategy is thus to combine any given graphs into a unified hierarchy that represents a wide range of potential interpretations of the raw variants. In the current paper, HARVESTMAN combines a graph extracted from the genome annotation with the three GO graphs.

Hierarchical feature selection is a relatively new area of research. HARVESTMAN is most closely related to a recent greedy algorithm by Ristoski and Paulheim named SHSEL [14], as both approaches seek to maximize feature relevance while reducing feature redundancy. The SHSEL algorithm has two steps. In the first step, SHSEL iteratively processes the graph from the leaves to the root. A node is removed from G if it is uncorrelated with the label (i.e. irrelevant), or highly correlated with one of its ancestors (i.e. redundant). In the second step, SHSEL computes the average relevance of the

remaining nodes along each path from the root to a leaf. A node is removed if its relevance is below-average on a given path. The SHSEL algorithm is elegant, but it is not guaranteed to output an optimal set of features (i.e., those that are both maximally relevant, and least redundant). Like HARVESTMAN, SHSEL is also a supervised approach. While we focus this work on supervised methods, we recognize that unsupervised methods have also been explored [16].

Applications of hierarchical feature selection to biology and medicine have been reported, including the HIP (select Hierarchical Information Preserving features) and MR (select Most Relevant) algorithms [15, 19, 20]. Like HARVESTMAN, these approaches use GO to define a hierarchy of binary features, although are intended for the analysis of gene expression data rather than genomic variants, and do not incorporate any other hierarchy types. Additionally, HIP and MR are intended for lazy-learning, where feature selection and model building are performed for each new instance. HARVESTMAN instead identifies features that work well across a cohort of samples.

As previously mentioned, one limitation of existing hierarchical feature selection methods is that they provide no guarantees with respect to the optimality of the chosen features. HARVESTMAN, in contrast, formulates the problem as an integer-linear program (ILP) where the user specifies an objective function and an optional set of constraints. The objective function defines the desired tradeoff between some measure of feature relevance (e.g., mutual information) and redundancy (e.g., correlation). The user may also specify suitable linear constraints, such as the maximum number of features to be selected. An ILP solver then returns a subset of maximally informative and minimally redundant subset of features, subject to the constraints, or else reports that no solution exists. Ghalwash et al. [21] propose a similar ILP-based method for (non-hierarchical) feature selection using expression data. They ultimately relax their ILP to a convex optimization problem, because integer programming is NP-complete. We will demonstrate that when using modern ILP solvers, it is possible to perform hierarchical feature selection over very large knowledge graphs. We note that while HARVESTMAN does make simplifying assumptions with the data prior to solving an ILP, the problem presented to the ILP is solved exactly. When we refer to the optimality of HARVESTMAN, we are referring to the value of the user-specified objective.

Results

We evaluated HARVESTMAN in two ways. First, we tested its scalability using the 1000 Genomes data. Second, we compared HARVESTMAN to existing methods for feature selection on a subset of The Cancer Genome Atlas (TCGA) breast cancer data.

Evaluating HARVESTMAN's scalability using 2504 whole genome sequences

To demonstrate the effectiveness of HARVESTMAN at scale, we apply our method to data obtained from the 1000 Genomes Project [22], a large and well-known publicly available DNA sequencing data set. In these experiments, we use their most recent Phase 3 data, which includes a combination of low-coverage whole genome and high-coverage exome sequencing for each of 2504 samples. These samples belong to one of five ethnic superpopulations (African, American, South Asian, East Asian, and European). In this experiment, we perform feature selection and model building with the task of predicting ethnic

super-population from DNA sequence, and evaluate Harvestman's scalability using a variety of solvers and progressively more powerful public cloud computing instances.

We run five-fold cross validated experiments using two versions of the knowledge graph, one without SNP representation totalling roughly 1.5 million nodes, and one with SNP representation totalling roughly 25 million nodes. In Fig. 1 we demonstrate both Harvestman and SHSEL as computationally intensive feature selection methods, and a simple threshold test as a baseline. Memory usage is not shown, since it remains relatively constant across experiments and never exceeds 8 GB. Since the knowledge graph is fully constructed while running the inexpensive threshold test, and training a multi-class logistic regression classifier takes no more than several seconds, we can use threshold timings as a proxy for the time spent loading nodes and edges into memory and computing mutual information.

We apply several pre-processing steps before running the feature selection methods. Since building an ILP problem with the full graph would take at least two days of processing time, we filter nodes from the smaller and larger graphs by applying a mutual information threshold of 0.2 and 0.4, respectively. This reduces total feature size to between 15,000 and 30,000 and produces no more than 120,000 correlations to consider. For ease of reproduction, we limit the selection set size to 1000 and use COIN-OR solver with a maximum runtime of 1 h.

While running Harvestman to construct an ILP with four CPU cores, graph construction takes 33% of the runtime in the small graph but as much as 79% of the runtime in the SNP-enhanced graph. Harvestman then spends its remaining time computing correlations, setting up an ILP problem, and finally running a solver instance. Once the instance is constructed, even difficult problems can be solved quickly and economically with commercial solvers such as CPLEX. To demonstrate this, we select 5000 features to predict ancestry on the SNP-inclusive 1000 Genomes graph, which corresponds to an

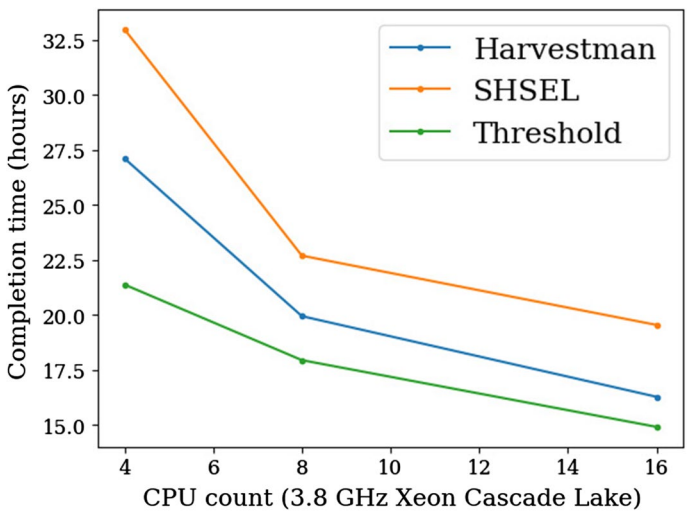

Fig. 1 Harvestman can solve large problem instances in hours. Timing for graph construction and feature selection using Harvestman, SHSEL, and an MI threshold with access to varying numbers of CPUs on the 1000 Genome data. The initial graph consisted of 23,393,068 nodes. This figure was generated using Matplotlib version 3.2.1

ILP problem with roughly 70,000 considered features and over 1.2 million correlations between them. CPLEX will find an optimal solution in under a minute and six gigabytes of RAM. The commercial Gurobi solver and open source COIN-OR solver will not come to a solution on such a problem in under 48 h, but are suitable for smaller problems.

Our method, like SHSEL, scales to multiple processors. Each fold of cross-validation can be run in separate threads that receive data from a single producer thread, which allows us to implement coarse grained parallelism with relative ease. We can achieve finer parallelization by dividing the computation of correlations and other graph processing tasks as we build the ILP. For solving the ILP, both CPLEX and Gurobi allow multiple threads and scaling across multiple machines using message passing. The most time consuming steps that must be performed serially are reading the knowledge graph and associated feature vectors from the binary files, and accessing the solver API to construct the problem from the processed graph. However, these actions can be performed concurrently with the other parts of the program, and do not pose a major hindrance with the number of cores available at the range of a high-end desktop or economically-available cloud instance.

Once the features are selected, training a logistic regression (or any other) classifier takes a trivial amount of time, and its memory consumption increases with the number of features. Models built with the selected features were able to properly classify ethnicity with over 99% accuracy. This high level of accuracy is expected, as it is known that there are strong markers for geographically separated populations scattered throughout the genome [22].

Using Harvestman to predict cancer survival outcomes

A difficult yet important problem in cancer genomics is finding markers that are predictive of patient outcomes. Adding to the difficulty is that the available training data may be small, with respect to the number of patients, and/or imbalanced with respect to the relative proportions of each outcome. We demonstrate here the effectiveness of using our hierarchical representation of DNA sequence data in these settings by building models for two binary breast cancer survival outcomes.

Using a curated subset of the TCGA BRCA cancer data, we considered two binary endpoints: predicting five-year survival and five-year disease-free survival. The five-year survival data set contains 136 samples with a 100/36 outcome ratio. The five-year disease-free survival data set contains 120 samples with an 89/31 outcome ratio. Sequencing and survival status data was obtained from [23], and all data was initially processed according to the original TCGA specifications [24]. We report results obtained from ten different permutations of the data. For each permutation, we held out 30% of the data for testing. We did five-fold cross validation on the remaining 70%, and report the cross-validated accuracy on the holdout set. Thus, each permutation has a unique hold-out set and training set. These ten permutations were created to test the robustness of the feature selection and model building steps. Note that the training and holdout sets for each permutation are not identical between endpoints.

For comparison, we also applied the previously described SHSEL hierarchical feature selection algorithm [14] and the ReliefF [12] algorithm to the same data partitions. ReliefF is a well-known, scalable, but non-hierarchical approach to feature selection.

Briefly, ReliefF ranks features according to their ability to discriminate between labels. We note that, being a filter method, relieff simply ranks the features. The user then decides how many features to include in the classifier.

The main tunable parameter for SHSEL is a similarity threshold, which is analogous to Harvestman's mutual information threshold, t (see Methods). We used SHSEL's recommended similarity threshold of 0.99 in our experiments. With relieff, we select the top c features, where c is the number of features selected by Harvestman when given the same train-test split. We did not place a limit on the number of features Harvestman should choose, rather we allow the ILP to simply choose the number of features that maximize the objective. For each experiment, an identical knowledge graph was used as a starting point for each algorithm. To further show robustness of the method, we report classification accuracy obtained with three different classifier types, logistic regression (LR) with no regularization, random forest (RF) using 100 trees, and support vector machine (SVM) with radial basis function kernel. All unspecified classifier parameters were left in their default settings.

Harvestman's knowledge graph is more informative than a binary encoding of raw SNPs

Harvestman is predicated on the idea that the knowledge graph, a hierarchical representation of prior knowledge over the human genome, may contain more suitable feature encodings than raw SNPs. By construction, the bottom layer of Harvestman's hierarchy consists of annotated SNPs, and further loci-centric annotation comprise the higher layers. In order to demonstrate the informative value of the knowledge graph with respect to that of SNPs alone, we ran experiments using three segments of the knowledge graph:

1 All node types
2 All node types except SNPs
3 Only SNPs

For both survival endpoints, we initialized knowledge graphs using starting MI thresholds of 0.05, 0.075, 0.1, and 0.125. For cases (1) and (2) above, we then applied Harvestman's ILP-based feature selection strategy. We trained classifiers on the selected feature subsets as well as with the SNPs from each graph alone, and report the model AUC as a function of feature counts in Fig. 2. Each data point in Fig. 2 corresponds to one of the four initial knowledge graphs, where the initial MI threshold decreases moving from left to right. We do not perform further feature selection on segment 3 (only SNPs). With respect to the five year survival endpoint, the number of nodes in segment 3 is less than the total number of nodes selected by Harvestman on segments 1 and 2. In contrast, the number of segment 3 nodes is more comparable to those selected on segments 1 and 2.

For both endpoints, the features selected by Harvestman from segment 1 and 2 knowledge graphs achieve a higher AUC than those for segment 3 ($p < 0.05$, paired t-tests), and this behavior generalizes across three classifier types. This suggests that features encoded within the knowledge graph are more informative for these classification tasks than are encodings of SNPs alone. Furthermore, there is no difference in AUC

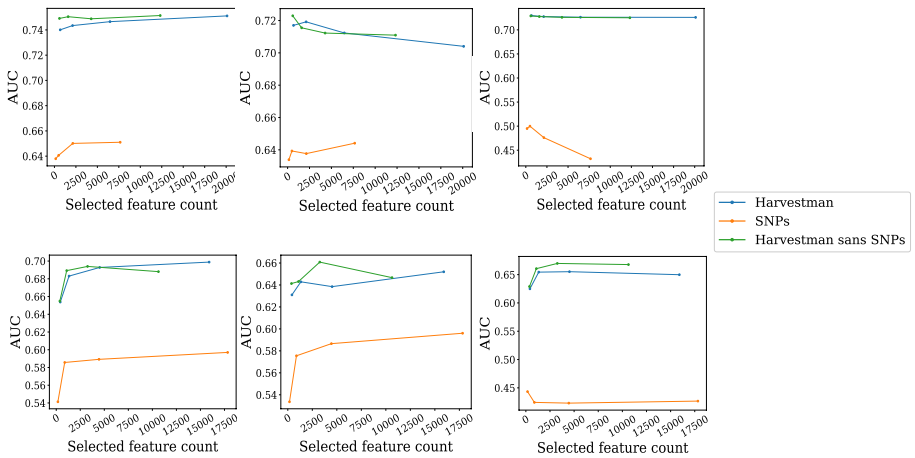

Fig. 2 The knowledge graph is more informative than raw SNPs. AUC as a function of feature counts for five year survival (top) and five year disease free survival (bottom) obtained with logistic regression (left), random forest (middle), and SVM (right). Each point in each figure corresponds to a knowledge graph that has been filtered by one of four MI thresholds (0.125, 0.1, 0.075, and 0.05.). The points are ordered from left to right in order of decreasing MI thresholds. Harvestman selects different numbers of features (x-axis), depending on the input graph. Comparisons are made using models trained on binary encodings of SNPs that passed those same thresholds. This figure was generated using Matplotlib version 3.2.1

Table 1 Harvestman applied to the five-year survival endpoint over a full knowledge graph (GO + Genomic) and both of its constituent components (GO, Genomic)

Knowledge graph	Selected count	AUC
GO + GENOMIC	23,939	0.74
GO	85	0.58
Genomic	23,836	0.74

Values shown are averaged over five-fold cross validation, and AUC was obtained from a logistic regression classifier

when comparing Harvestman over the complete knowledge graph and Harvestman over the knowledge graph sans SNPs. This further verifies that the best set of features is obtained using portions of the graph representing genomic loci at a broader scale than individual SNPs. If this were not the case, then we would expect Harvestman sans SNPs to perform more poorly than Harvestman over the complete knowledge graph. Additionally, Harvestman is still able to identify these informative, higher level features even when presented less informative SNP nodes. We conclude that the knowledge graph effectively encodes genomic features better than using raw SNPs alone.

Recall that the knowledge graph is built from two primary sources—genomic loci-based annotations and the GO hierarchy. To evaluate the relative contributions of both components to these two prediction tasks, we ran experiments where we limited the construction of the knowledge graph to each of these constituent parts. Tables 1 and 2 show results when using Harvestman on these knowledge graphs applied to the five-year survival and five-year disease-free survival endpoints, respectively. For simplicity, these results focus on the most lenient MI threshold (0.05) and use only a single classifier type (logistic regression). For these two endpoints, the differences in model AUC make it apparent that the Genomic portion of the knowledge graph is more informative than the GO portion. While the raw number of selected Genomic features is also much

Table 2 HARVESTMAN applied to the five-year disease-free survival endpoint over a full knowledge graph (GO + Genomic) and both of its constituent components (GO, Genomic)

Knowledge graph	Selected count	AUC
GO + GENOMIC	11,671	0.68
GO	107	0.55
Genomic	11,497	0.68

Values shown are averaged over five-fold cross validation, and AUC was obtained from a logistic regression classifier

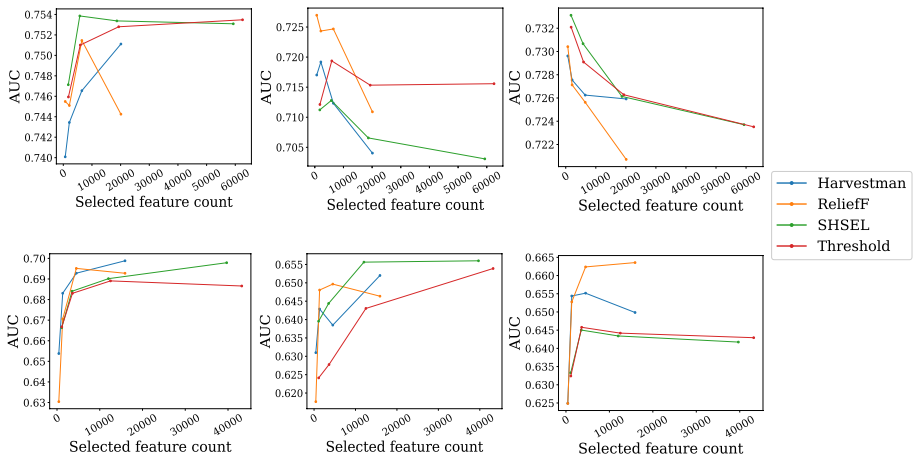

Fig. 3 HARVESTMAN is more parsimonious than SHSEL, meaning it selects fewer features than SHSEL without sacrificing model AUC. AUC as a function of feature counts for five year survival (Top) and five year disease free survival (bottom) obtained with logistic regression (left), random forest (middle), and SVM (right). In each, HARVESTMAN is applied to complete knowledge graphs with MI thresholds 0.125, 0.1, 0.075, and 0.05, moving left to right. This figure was generated using Matplotlib version 3.2.1

higher, the net reduction in features chosen from each graph relative to its initial size is similar, as the GO graph starts with approximately 43,000 nodes, whereas the Genomic graph starts with greater than 2 million.

HARVESTMAN selects fewer features than SHSEL without sacrificing model AUC

Given the success of using the knowledge graph compared to an encoding of SNPs alone, we next compare HARVESTMAN to SHSEL and RELIEFF over knowledge graphs containing each node type. Using the same initial MI filters as before, we show in Fig. 3 the AUCs of three different classifiers on both survival tasks as a function of the number of features selected by each method. We also include as a natural baseline a model trained on all features that passed each initial threshold.

Reducing the dimensionality of data is the primary goal of any feature selection strategy. In each experiment, we find that HARVESTMAN selects significantly fewer features than SHSEL or the threshold baseline ($p < 0.05$, paired two-sided t-test).

As the initial graph increases in size, this effect becomes increasingly more pronounced. In the case of the most lenient threshold used (0.05), we find that SHSEL selects nearly 60,000 features from the five year survival knowledge graph and 40,000 from the five year disease free survival, and does not improve much upon the threshold

baseline. This is compared to about 20,000 and 15,000 features respectively with Harvestman. Thus, Harvestman is more effective, in terms of reducing total feature counts for these two endpoints.

When we consider the underlying AUC of each trained model, we see several patterns that depend on classifier type and the survival endpoint. With respect to disease free survival, there is a general trend that as we decrease the initial threshold, the AUCs tend to increase or plateau, as each feature selection strategy chooses larger numbers a features that are used in training. The highest average AUC measured overall corresponds to Harvestman using LR, though SHSEL and ReliefF obtain the highest AUCs when using RF or SVM, respectively.

In general, we note that all models obtain higher AUC in the five year survival setting compared to five year disease free survival. With the LR classifier, we found no statistically significant differences between model AUCs, though note that the best overall AUC was obtained by a model using SHSEL. RF and SVM classifiers show evidence of overfitting, as AUC tends to decrease as the number of features increases. Note that the scaling of the AUC and feature count axes indicate these differences are not as stark as they may at first seem, as AUC varies by less than 0.2 across experiments using both classifiers. In any case, this trend occurs independent of feature selection strategy. With SVM, there is no statistical difference between AUCs across experiments. With RF, there are instances where a ReliefF model (initial MI threshold 0.125) and Threshold model (initial MI threshold 0.05) outperform the Harvestman model. In the Threshold case, we note that Harvestman had selected significantly fewer features in that experiment. The most common result is that model AUCs obtained with each feature selection method are statistically indistinguishable for a given MI threshold and classifier type. In general, this means that Harvestman can select fewer features than the hierarchical method SHSEL and the baseline threshold method without sacrificing model AUC. Harvestman is thus more parsimonious than SHSEL.

Comparing selected feature sets by average correlation

It is desirable for feature selection algorithms to select non-redundant features. We investigated the redundancy of features selected by each algorithm over knowledge graphs by computing pairwise correlations between subsets of selected features. For each experiment, we randomly selected 1000 features selected by each feature selection strategy, and report in Fig. 4 the absolute values of pairwise Pearson correlations between those features.

In both endpoints, we notice some similar trends. For one, ReliefF consistently selects the most redundant features. While both Harvestman and SHSEL consider pairwise similarity of hierarchically related features as a means of reducing redundancy among their selected feature subsets, ReliefF does not. It therefore makes sense that Harvestman and SHSEL should select less redundant features than ReliefF. Among features that pass the initial mutual information threshold, ReliefF selects a redundant subset, which is why ReliefF also has higher pairwise correlation than the Threshold.

With respect to the two hierarchical feature selection methods, we notice that as the initial MI threshold decreases, there is a more pronounced difference between pairwise correlations obtained by Harvestman relative to SHSEL. For both endpoints, when

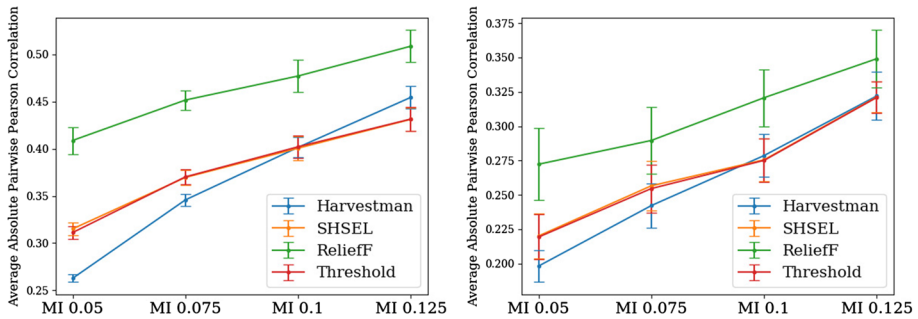

Fig. 4 HARVESTMAN selects fewer redundant features than other methods as graph size increases. The average absolute pairwise correlation of 1000 randomly sampled selected features for each selection algorithm for both the five year survival (Left) and five year disease free survival (Right). Error bars denote standard errors over ten runs with different train-test permutations. This figure was generated using Matplotlib version 3.2.1

considering knowledge graphs constructed with MI thresholds 0.05 and 0.075, features selected by HARVESTMAN have lower pairwise correlation compared to SHSEL, and the differences are statistically significant. For thresholds 0.1 and 0.125, HARVESTMAN does not select less redundant subsets. Since lower initial thresholds correspond to larger initial knowledge graphs, this suggests that HARVESTMAN is more adept at finding less redundant features in larger problem instances. Additionally, where HARVESTMAN is able to select fewer redundant features than SHSEL, so too is it able to select fewer features than the MI baseline, whereas SHSEL is unable to improve upon the baseline.

Discussion

In comparison to alternative methods, HARVESTMAN tends to make more parsimonious selections, meaning smaller or similar sized subsets of features. Smaller feature subsets are desirable for both practical and statistical reasons. In particular, smaller subsets may be easier for humans to interpret and understand, and will produce simpler models that are less likely to overfit the training data. Additionally, standard results from statistical learning theory [25] show that the number of samples required in order to successfully learn a discriminative model is linear in the VC dimension [26] of the hypothesis space. The VC dimension for most models is typically linear in the number of parameters [27] which, in turn, is at least linear in the number of input features. Like SHSEL, our method uses a knowledge graph to guide feature selection. However, as the knowledge graph increases in size, HARVESTMAN is more aggressive when it comes to eliminating redundant features.

By construction, each node (feature) in our knowledge graph corresponds to specific genomic annotation. This makes it straightforward to relate prior knowledge to the classification task at hand, and strengthen existing relationships or forge new ones. Moving forward, we look to strengthen our knowledge graph by adding more diverse sets of annotation, particularly those that identify regulatory elements across the genome. This can help make the graph more directly interpretable, and may make inspecting selected features instructive. While our experiments are performed over our knowledge graph, we note that HARVESTMAN and SHSEL are easily configured to use any suitable knowledge graph.

Harvestman provides an additional degree of customization through the ILP objective. In particular, the ILP objective function specifies the global properties of the resulting feature set. For example, users can control the size of the set and adjust the tradeoffs between feature relevance and redundancy. SHSEL, in contrast, performs feature selection via explicit graph traversals. That is, SHSEL selects features based on local properties in the graph, while Harvestman solves a global optimization problem, exactly. Harvestman's ILP-based formulation also facilitates the enumeration of multiple, distinct solutions, a capability not provided by SHSEL. This means that after the first solution is found, it is possible to force the ILP solver to find a second solution with the same objective value if one exists. In this way, it is possible to generate multiple solutions. In general, it is possible to augment a given ILP objective with arbitrary constraints, and then re-run the solver. In particular, we could add constraints that force the ILP to find a solution that differs from the previous by some minimum number of features. In future work, we envision exploring this functionality as a means to identify robust features by solving iterative ILP problems. Since integer programming is NP-complete, one drawback with these approaches is that there is no guarantee that they can be solved quickly. While the commercial CPLEX solver works well on the problems we tested, it is likely that there exists problems where the solver would not perform as efficiently. We leave effort towards relaxing Harvestman's ILP formulation to the easily solvable LP case for future work.

We note that it is straightforward to adapt Harvestman to incorporate additional data types (e.g. expression data). The knowledge graph does not need to be a connected graph. Thus, it is easy to incorporate additional hierarchies, even isolated feature nodes. This same capability means that it is possible to evaluate different ways of constructing internal nodes. In our experiments, the binary vectors associated with the internal nodes of the knowledge graph were primarily constructed using logical `or`'s. If desired, one could create and include additional graphs where the binary vectors are constructed using logical `and`'s or any other user-specified function. This capability may be useful when it is unclear which relationships among features best reflect the true relationships between features. While our experiments were performed on binary-valued features, the approach is capable of incorporating numeric features. In principle, many functions can be used to create parent feature vectors from their child nodes, including those that take and emit real values. As an example, we envision this functionality could be used to create nodes that represent gene expression.

Finally, in evaluating Harvestman's performance in selecting features and making predictions for survival endpoints in cancer, it is important to consider potential limitations of this data and prediction task. For one, survivorship is a complex issue that has many contributing factors. While genetics certainly plays a role in the likely survival and treatment options available to those with cancer, there are other environmental and lifestyle factors, such as tobacco usage [28] or age [29], that also contribute to survival. Furthermore, the small sample size and imbalanced nature of the data further contribute to making this feature selection and prediction task a difficult enterprise. Still, within this challenging setting, Harvestman was able to identify predictive feature subsets, and thus expose specific genomic markers that may play a role in cancer survival.

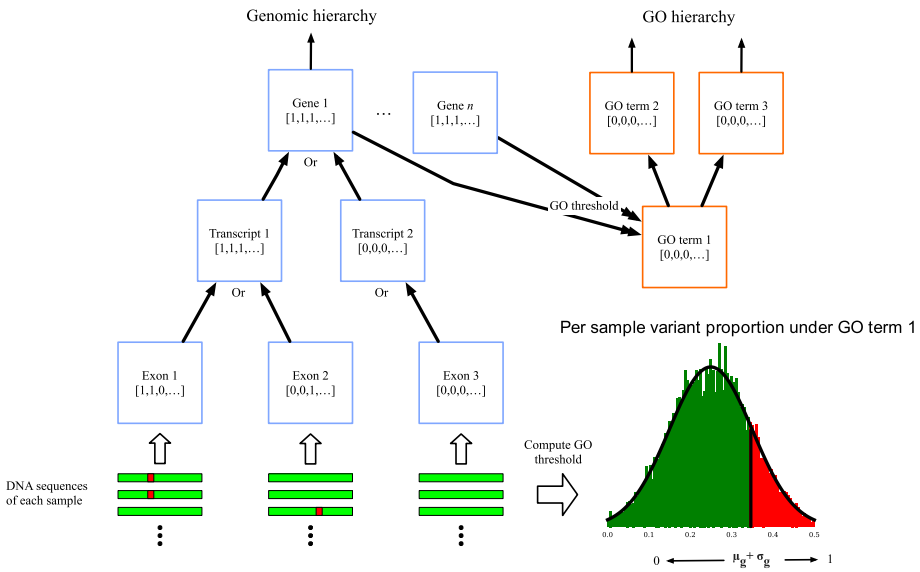

Fig. 5 HARVESTMAN's knowledge graph and variant encoding scheme. The knowledge graph is composed of the genomic hierarchy (blue boxes) and GO hierarchy (orange boxes). Binary vectors at the genomic hierarchy leaf nodes are determined directly from DNA sequences (shown by green bars, variants in sequence shown by red boxes). Binary vectors at parent nodes are computed by taking the logical OR of their child nodes or directly from the DNA sequence. A GO threshold is determined for each GO term from variant sequences related to its connected gene nodes. We use this threshold to determine a binary vector that reflects whether or not each sample is greater or less than the threshold. This figure was generated using Matplotlib version 3.2.1 and OmniGraffle

Conclusion

We have introduced HARVESTMAN, a new approach to supervised hierarchical feature selection, and demonstrated it on our knowledge graphs built from high-throughput sequence data. Using the 1000 Genomes Project, we show HARVESTMAN scales to thousands of genomes, and demonstrate that we can perform feature selection quicker with HARVESTMAN than the hierarchical feature selection method SHSEL when allocated 4, 8, or 16 CPUs. Next, using breast cancer data from The Cancer Genome Atlas, we show that HARVESTMAN selects a rich combination of representations used to predict five year patient survival and disease status, and that these representations perform better than a binary representation of SNPs alone. Finally, we compare HARVESTMAN to existing feature selection methods, hierarchical (SHSEL) and otherwise (RELIEFF), and demonstrate that our method is more *parsimonious*—it selects smaller and less redundant feature subsets while maintaining accuracy of the resulting classifier.

Methods

HARVESTMAN performs automatic feature learning, feature selection, and model building in three steps: (i) hierarchy construction; (ii) optimal hierarchical feature selection; and (iii) model building. These steps are described in the following subsections.

Table 3 The Ensembl IDs that are used by Harvestman, as well as a description of the genomic feature type that they represent

Annotation type	Description
Genes	A known protein or RNA coding gene
Transcripts	A known transcript in a coding gene
Exons	A known exon in a coding gene
Peptides	Identifies sequences associated with a known peptide
UTR	5 or 3 untranslated regions
Bio region	A catch-all annotation describing a genomic region relevant to some biological process
SNP	SNPs annotated by NCBI's RefSNP database

Constructing a hierarchy of feature representations

Harvestman leverages hierarchical relationships among potential feature encodings to facilitate feature selection. Figure 5 outlines the process of constructing a knowledge graph from one or more variant call format (VCF) files. The first step encodes annotated variants and structural elements (see Table 3) as a directed acyclic graph. We refer to this initial graph as the 'genomic hierarchy'. Each node in the genomic hierarchy corresponds to a genomic element. The topology of the graph reflects the logical relationships among these elements. For example, the children of a node representing a gene will be the known transcripts of that gene. Similarly, the children of a node representing a particular transcript will be the exons contained in that transcript.

To build the genomic hierarchy, we use Reference SNP (RefSNP) [30] and Ensembl annotations [31]. Ensembl IDs provide unique identifiers that match regions of the genome to structural and functional elements, such as exons, transcripts, genes, and 3′ or 5′ untranslated regions. Inclusion of these elements are motivated by the biological contributions each can make with respect to certain diseases and phenotypes, particularly in the presence of genomic variation.

Variation on the level of exons, transcripts, and genes perhaps have the most clear relationship to observable phenotypes or disease, as each of these elements contribute to the creation of functional proteins from genomic sequence. Alternative splicing could also be explained by variation at this level, and can contribute to cancer [32]. 5′ and 3′ untranslated regions are critical regulators of post-transcriptional gene regulation, and genomic variants in these regions are also implicated in cancer [33]. Ensembl also identifies genomic regions with predicted functional roles, including transcription start sites, enhancers, and promoters, or those that are associated with known peptides. These regions are collectively referred to as 'biological regions' or 'peptides' accordingly, and variation in such sites have also been tied to cancer among other diseases [34–36]. Associating an Ensembl ID to each node in the graph thus makes clear the biological interpretation of each feature.

Harvestman grafts the three Gene Ontology (GO) hierarchies onto the genomic hierarchy (Fig. 5, orange nodes). GO is a knowledgebase that relates genes to gene products with respect to cellular components, molecular function, and biological process. Since each GO term is associated with a specific set of genes, Harvestman adds a directed edge from each leaf node in the GO graphs to the appropriate 'gene'

nodes in the genomic hierarchy (i.e. those with annotation ID 'gene'). These edges create a combined graph that includes the initial genomic hierarchy and the GO hierarchies. We refer to this structure as HARVESTMAN's knowledge graph.

HARVESTMAN assigns an n-element binary vector to each leaf node in the knowledge graph. Here, n corresponds to the number of samples within the VCF file(s). For each leaf node, the algorithm sets bit i to 1 if the ith sample contains a variant associated with that node. Otherwise, the ith bit is set to zero. Then, in a bottom-up fashion, the algorithm assigns n-element binary vectors to internal nodes by applying a function that combines the vectors from that node's children. Various functions can be used, including logical `or`'s, `and`'s, `xor`'s, and threshold functions. The framework is general enough that the choice of function can be tailored to suit properties or defining characteristics of specific portions of the graph. If desired, multiple functions can be evaluated in the same knowledge graph through a duplication of parent nodes. For simplicity, when we refer to the knowledge graph in the following sections, we assume each node has been assigned a binary vector.

In our experiments, most vectors were combined by computing a logical `or`. However, a threshold function was used at the nodes that were originally leaves in the GO graphs. Threshold functions were used at these nodes to avoid saturation of the binary vectors (i.e. vectors of all ones). Recall that edges were created between the GO leaf nodes and their associated gene nodes in the genomic hierarchy to create the knowledge graph. Each GO leaf node may be associated with many genes, so it is easy for their binary vectors to become saturated. Saturated vectors are not informative and will therefore never be selected as a feature. If vectors are combined using logical `or`s, then any ancestor of a saturated node will also be saturated and so a single saturated vector may effectively eliminate an entire subgraph from being selected. We avoid this problem by computing a threshold for each GO leaf node. This threshold identifies samples that have an accumulation of genomic variants in a given region of the genome, which is a hallmark trait of malignancies [37]. A visual example of this procedure is shown in Fig. 5. Let v be an arbitrary GO leaf node. The threshold associated with v is based on the statistics of the number of variants that are observed among the genes connected to v. Briefly, let μ (resp. σ) be the average (resp. standard deviation) of the number of variants from each sample that map to the genes associated with v. This is shown as a distribution in Fig. 5. We say that sample i is enriched in v if the number of variants in the ith sample that map to any of the genes associated with v is $\geq \mu + \sigma$. The ith bit of the binary vector associated with v is set to one if sample i is enriched for v (these samples are highlighted in red in Fig. 5). Otherwise, it is set to zero (shown in green). This threshold function is only used to compute the binary vectors for the GO leaf nodes. The remaining nodes from the GO hierarchy are assigned binary vectors by computing logical `or`'s.

It is possible for the binary vectors associated with two nodes connected via an edge to be identical. This will occur, for example, if the out-degree of a node is one. When this happens, the two nodes are redundant, and there is no need to consider both of them during feature selection. We handle this situation by "collapsing" redundant nodes (and pathways of redundant nodes) into a single node. This has the benefit of reducing the total number of features that need to be considered during feature selection.

Optimal hierarchical feature selection via integer linear programming

We introduce here an ILP-based approach that identifies relevant and non-redundant features based on the mutual information (MI) between the features and the given label, and the pairwise correlation between features in the knowledge graph.

Let $B_i \in \{0,1\}^n$ be the binary vector associated with node i in G, and let $L \in \{0,1\}^n$ be a binary label vector encoding, for example, the presence or absence of a particular phenotype. We denote the mutual information (MI) between B_i and L as $I(B_i; L)$. This is a measure of feature relevance, as it indicates how much information we gain about the label after observing feature vector B_i. More precisely, the MI between two random variables is defined as:

$$I(B_i; L) = H(L) - H(L|B_i) \tag{1}$$

Here, $H(L)$ is the entropy of the labels, L, and $H(L|B_i)$ the conditional entropy of labels, after observing binary feature vector B_i. The entropies quantify the uncertainty of the corresponding random variables representing the labels and features. Let $p(X)$ denote the probability that random variable $X = 1$. Then, for $b \in B_i$ and $\ell \in L$ we have the following:

$$\begin{aligned} H(L) &= -\sum_{j=1}^{n} p(\ell_j) \log p(\ell_j) \\ H(L|B_i) &= -\sum_{k=1}^{n} p(b_k) \sum_{j=1}^{n} p(\ell_j|b_k) \log p(\ell_j|b_k) \end{aligned} \tag{2}$$

Thus, high MI between feature vector and the label corresponds to lower uncertainty, making such features relevant to classification tasks. MI has commonly and effectively been used as a measure of feature relevance in biological settings [38–40].

In practice, we find it useful to pre-filter features using an MI threshold designed to eliminate features that are clearly irrelevant. We consider this an information theoretic equivalent to variant filtration pre-processing steps common to DNA sequence analyses. Harvestman lets the user specify a threshold, $t \geq 0$, and it pre-filters feature i if $I(B_i; L) < t$.

The knowledge graph contains many highly correlated features by construction. Highly correlated features provide little or no additional information and increase model complexity, and so should be eliminated. We denote the correlation between features i and j as $\text{Corr}(B_i, B_j)$. We used the Pearson correlation coefficient in our experiments, although many other measures of correlation between binary features can be used [41]. Harvestman pre-computes the correlations between all pairs of features that pass the MI filter outlined above.

We include correlations of only a subset $\mathcal{P}$ of all feature pairs when solving the ILP. We consider correlations between all pairs of features that share a common 'gene' node as an ancestor. Additionally, because two different genes may overlap, we also consider correlations between pairs of features that have overlapping gene nodes as ancestors. Next, to account for relationships between nodes across the genomic and GO hierarchies, we include all pairs of features that fall along a directed path within the knowledge graph.

Any pair of features fitting the above requirements that are at least moderately correlated (Pearson correlation ≥ 0.3) are included in $\mathcal{P}$.

The elements of $\mathcal{P}$ are included as terms in the ILP objective, which is defined as follows:

$$\begin{array}{ll} \text{maximize}_{w,z} & \sum_{i=1}^{n} w_i I(B_i; L) - \lambda \sum_{(i,j)\in\mathcal{P}} z_{ij} |\text{Corr}(B_i, B_j)| \\ \text{subject to:} & z_{ij} \geq w_i + w_j - 1, \forall i, \forall j \\ & \sum_{i=1}^{n} w_i \leq c \text{ for some } c \in \mathbb{N} \\ & w_i, z_{ij} \in \{0, 1\} \end{array} \tag{3}$$

Each feature i is associated with a binary decision variable, w_i, and each feature pair, (i, j), is associated with decision variable z_{ij}. If $w_i = 1$, then feature i is selected. By considering the absolute value of the correlations, we ensure that anti-correlated unselected pairs will not artificially boost the objective value. Parameter λ adjusts the relative importance between mutual information and pairwise correlation. Parameter c imposes a constraint on the maximum number of features to select. If c is set equal to the number of input features, the ILP will naturally find the optimal number of features to select. In our experiments, we set $\lambda = 1$ and varied c.

We emphasize that Harvestman's ILP-based approach to feature selection does not involve the construction and evaluation of classifiers (or any predictive model). Whether a given feature is selected is determined entirely based on the optimal solution to problem (3).

Finally, Harvestman uses standard Machine Learning libraries to train classifiers and regression models, after the feature selection step. In our experiments, we used Microsoft's ML.NET machine learning library and scikit-learn [42] for model building. We emphasize that model building and evaluation occur after setting aside a hold-out test set. This is done to prevent data leakage from the procedure used to select features to the procedure used to evaluate model performance. To assess model accuracy, we report Area Under the receiver operating Curve (AUC).

Abbreviations

ILP: Integer linear program; MI: Mutual information; VCF: Variant Call Format; TCGA: The Cancer Genome Atlas; GO: Gene Ontology; SNP: Single nucleotide polymorphism; UTR: Untranslated region; AUC: Area under the curve; CPU: Central processing unit.

Acknowledgements

Not applicable.

Authors' contributions

TF conducted experiments using Harvestman and led in writing the manuscript. SJB implemented Harvestman and assisted in writing the manuscript and conducting experiments. GM and QMH assisted with computing resource issues and helped with experimental design. CK and CJL designed and supervised the research. All the authors read and approved the manuscript.

Funding

This work was partially funded by the Center for Machine Learning and Health at Carnegie Mellon University and is supported by the CURE Grant 4100070287 from the Pennsylvania Department of Health (PA DOH). The content of this paper is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies. The Pennsylvania Department of Health specifically disclaims responsibility for any analyses, interpretations or conclusions. This work used the Extreme Science and Engineering Discovery Environment [43], which is supported by National Science Foundation Grant Number ACI-1548562 through allocation TG-DMS160012. This project used the Hillman Cancer Bioinformatics Services, which is supported in part by the National Cancer Institute award P30CA047904. This research is supported by an NIH T32 training Grant T32 EB009403 as part of the HHMI-NIBIB Interfaces Initiative. This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to C.K., and by the US National Institutes of Health (R01GM122935). None of the funding bodies played a role

in the design, analysis, or interpretation of data, or in writing the manuscript. The CURE award provided access to the TCGA data.

Availability of data and materials
The data used is available through the 1000 Genomes Project and The Cancer Genome Atlas. Access to TCGA data requires the completion of a Data Access Request through the Database of Genotypes and Phenotypes (dbGaP). Binary releases of Harvestman for multiple systems are available for download at https://github.com/cmlh-gp/Harvestman-public/releases.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
C.K. is co-founder of Ocean Genomics, Inc. G.M. is VP of software engineering at Ocean Genomics, Inc.

Author details
[1] Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA, USA. [2] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA.

Received: 2 December 2020 Accepted: 22 March 2021
Published online: 01 April 2021

References

1. Leung MKK, Delong A, Alipanahi B, Frey BJ. Machine learning in genomic medicine: a review of computational problems and data sets. Proc IEEE. 2016;104(1):176–97.
2. D'Argenio V. The high-throughput analyses era: Are we ready for the data struggle? High Throughput. 2018;7(1):8.
3. Clarke R, Ressom HW, Wang A, Xuan J, Liu MC, Gehan EA, Wang Y. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. Nat Rev Cancer. 2008;8:37–49.
4. Libbrecht MW, Noble WS. Machine learning applications in genetics and genomics. Nat Rev Genet. 2015;16:321–32.
5. Domingos P. A few useful things to know about machine learning. Commun ACM. 2012;55(10):78–87.
6. Bengio Y, Courville AC, Vincent P. Unsupervised feature learning and deep learning: a review and new perspectives. CoRR arXiv:abs/1206.5538 2012.
7. Li J, Cheng K, Wang S, Morstatter F, Trevino RP, Tang J, Liu H. Feature selection: a data perspective. ACM Comput Surv. 2017;50(6):94–19445.
8. Xing EP, Jordan MI, Karp RM. Feature selection for high-dimensional genomic microarray data. In: Proceedings of the eighteenth international conference on machine learning; 2001, pp. 601–608.
9. Blum AL, Langley P. Selection of relevant features and examples in machine learning. Artif Intell. 1997;97(1–2):245–71.
10. Hira ZM, Gillies DF. A review of feature selection and feature extraction methods applied on microarray data. Adv Bioinform. 2015;2015:198363–198363.
11. Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics. Bioinformatics. 2007;23(19):2507–17.
12. Kononenko I, Šimec E, Robnik-Šikonja M. Overcoming the myopia of inductive learning algorithms with relieff. Appl Intell. 1997;7(1):39–55.
13. Hall MA. Correlation-based feature selection for machine learning. Technical report, The University of Waikato; 1999.
14. Ristoski P, Paulheim H. Feature selection in hierarchical feature spaces. In: International conference on discovery science. Springer; 2014, pp. 288–300.
15. Wan C, Freitas AA. Two methods for constructing a gene ontology-based feature network for a Bayesian network classifier and applications to datasets of aging-related genes. In: Proceedings of the 6th ACM conference on bioinformatics, computational biology and health informatics—BCB'15. ACM Press, Atlanta, Georgia; 2015, pp. 27–36.
16. Wang S, Wang Y, Tang J, Aggarwal C, Ranganath S, Liu H. Exploiting hierarchical structures for unsupervised feature selection; 2017, pp. 507–515.
17. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G. Gene ontology: tool for the unification of biology. Nat Genet. 2000;25:25–9.
18. The Gene Ontology Consortium. Expansion of the gene ontology knowledgebase and resources. Nucl Acids Res. 2017;45(D1):331–8.
19. Wan C, Freitas A. Prediction of the pro-longevity or anti-longevity effect of caenorhabditis elegans genes based on bayesian classification methods. In: 2013 IEEE international conference on bioinformatics and biomedicine; 2013, pp. 373–380.
20. Wan C, Freitas AA. An empirical evaluation of hierarchical feature selection methods for classification in bioinformatics datasets with gene ontology-based features. Artif Intell Rev. 2018;50(2):201–40.
21. Ghalwash MF, Cao XH, Stojkovic I, Obradovic Z. Structured feature selection using coordinate descent optimization. BMC Bioinform. 2016;17(1):158.

22. The 1000 Genomes Project Consortium. A global reference for human genetic variation. Nature. 2015;526:68–74.
23. Cooper GF. CURE project. unpublished, in prep. 2019.
24. Network CGA. Comprehensive molecular portraits of human breast tumours. Nature. 2012;490(7418):61–70.
25. Vapnik VN. Statistical learning theory. Adaptive and learning systems for signal processing, communications, and control. New York: Wiley; 1998.
26. Vapnik VN, Chervonenkis AY. On the uniform convergence of relative frequencies of events to their probabilities. Theory Prob Appl. 1971;16(2):264–80.
27. Vapnik VN. Estimation of dependences based on empirical data. Springer series in statistics. New York: Springer; 1982.
28. ...Ng AK, DeMichele A, Alter BP, Rabkin CS, Pui C-H, Ambrosone CB, Begg CB, Malkin D, Hall EJ, Allan JM, Little JB, Offit K, Robison LL, Brown LM, Travis LB, Strong L, Tucker MA, Greene MH, Gospodarowicz MK, Hisada M, Rothman N, Caporaso N, Inskip P, Shields PG, Kleinerman R, Chanock S, Taniguchi T, Figg WD. Cancer survivorship genetic susceptibility and second primary cancers: research strategies and recommendations. JNCI J Natl Cancer Inst. 2006;98(1):15–25.
29. Nordenskjöld AE, Fohlin H, Arnesson LG, Einbeigi Z, Holmberg E, Albertsson P, Karlsson P. Breast cancer survival trends in different stages and age groups a population based study 1989 through 2013. Acta Oncol. 2019;58(1):45–51.
30. ...Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Geer LY, Kapustin Y, Khovayko O, Landsman D, Lipman DJ, Madden TL, Maglott DR, Ostell J, Miller V, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Tatusov RL, Tatusova TA, Wagner L, Yaschenko E. Database resources of the national center for biotechnology information. Nucl Acids Res. 2007;35(Database issue):5–12.
31. The Ensembl Consortium. Ensembl 2018. Nucl Acids Res. 2017;46(D1):754–61.
32. El Marabti E, Younis I. The cancer spliceome: reprograming of alternative splicing in cancer. Front Mol Biosci. 2018;5:80–80.
33. Schuster SL, Hsieh AC. The untranslated regions of mRNAs in cancer. Trends Cancer. 2019;5(4):245–62.
34. Hua JT, Ahmed M, Guo H, Zhang Y, Chen S, Soares F, Lu J, Zhou S, Wang M, Li H, Larson NB, McDonnell SK, Patel PS, Liang Y, Yao CQ, van der Kwast T, Lupien M, Feng FY, Zoubeidi A, Tsao M-S, Thibodeau SN, Boutros PC, He HH. Risk SNP-mediated promoter-enhancer switching drives prostate cancer through lncRNA PCAT19. Cell. 2018;174(3):564–57518.
35. Farman FU, Iqbal M, Azam M, Saeed M. Nucleosomes positioning around transcriptional start site of tumor suppressor (rbl2/p130) gene in breast cancer. Mol Biol Rep. 2018;45(2):185–94.
36. Rhie SK, Yao L, Luo Z, Witt H, Schreiner S, Guo Y, Perez AA, Farnham PJ. Zfx acts as a transcriptional activator in multiple types of human tumors by binding downstream of transcription start sites at the majority of cpg island promoters. Genome Res. 2018;28(3):310–20.
37. Talseth-Palmer BA, Scott RJ. Genetic variation and its role in malignancy. Int J Biomed Sci IJBS. 2011;7(3):158–71.
38. Jansi Rani M, Devaraj D. Two-stage hybrid gene selection using mutual information and genetic algorithm for cancer data classification. J Med Syst. 2019;43(8):235.
39. Sun Z, Zhang J, Luo Z, Cao D, Li S. A fast feature selection method based on mutual information in multi-label learning. In: Sun Y, Lu T, Xie X, Gao L, Fan H, editors. Computer supported cooperative work and social computing. Singapore: Springer; 2019. p. 424–37.
40. Zhu Q, Fan Y, He Y, Xu Y. Effective cancer classification based on gene expression data using multidimensional mutual information and elm. In: 2018 IEEE 7th data driven control and learning systems conference (DDCLS); 2018, pp. 954–958.
41. Choi S, Cha S-H, Tappert C. A survey of binary similarity and distance measures. J Syst Cybern Inf. 2009;8.
42. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
43. Towns J, Cockerill T, Dahan M, Foster I, Gaither K, Grimshaw A, Hazlewood V, Lathrop S, Lifka D, Peterson GD, Roskies R, Scott JR, Wilkins-Diehr N. Xsede: accelerating scientific discovery. Comput Sci Eng. 2014;16(5):62–74.

Publisher's Note