RESEARCH

Semi-supervised learning for integrative genomic analysis using hierarchical mixture models

Daniel Dvorkin[1*], Rani K. Powers[1], Michael W. Daniels[2] and Katerina Kechris[2]

[*]Correspondence:
daniel.dvorkin@ucdenver.edu
[1]Computational Bioscience Program, University of Colorado School of Medicine, 12801 E. 17th Ave., 80045 Aurora, US
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

First part title: Text for this section.

Second part title: Text for this section.

Keywords: sample; article; author

Background

Many investigations now involve the analysis of a large and growing range of genome scale data types, including DNA-sequence-derived variables, gene expression measurements, and epigenetic information. Each of these data types can provide valuable information about the roles played by the genomic loci the data represent, but none by itself can provide a complete picture. The need for integrative approaches to realize the full potential of data growing in diversity as well as volume has been widely recognized over the last several years. Several large-scale projects exist to organize and make use of multiple data sources, and provide data and analysis tools for specific research domains [1–4].

However, there is no generally agreed-upon methodology for integrative analysis and as the available data grow in size and complexity, so do the number and variety of questions we may wish to answer using that data. Recent reviews discuss some of the methodological challenges involved: large data sets consisting of heterogeneous data types, missing and incomplete data, noisy data from which it can be difficult to ascertain statistical significance, unclear relationships between the data types, and interpretability of results [5, 6]. An important conclusion is that integrated analysis is more effective than looking at data sources in isolation. Simultaneous approaches provide more insight into the biological processes that generate the joint distribution of the data [7] and help reduce the effect of noise and increase power to detect signal [8], and this is the approach we follow here.

In many studies, genomic data integration has focused on supervised **[REF]** classification approaches, which require high-quality training sets with both positive and negative controls. When available training data sets are small, unreliable, or incomplete, unsupervised methods are more commonly used[7, 9–11]. However, even training data that are not suitable for supervised approaches can still provide valuable information. In this work, we describe extension of the unsupervised methods first described in [12] to allow the use of any available training data, no matter how small or incomplete, for semi-supervised learning.

Here we focus on a gene-centric approach, where data from genes and their cis-regulatory regions can be used to identify genes that play key roles in particular

processes and phenotypes. Although there are a variety of applications, such as predicting genes relevant to a certain disease or pathway [REF], we focus on predicting gene essentiality. Essential genes are required for the survival of an organism and are important to understand the basic principles of cellular function, to engineer microorganisms using synthetic biology, and to identify potential targets for antibiotics [13, 14]. A variety of experimental methods and many of these results are collected in databases such as DEG and OGEE [15, 16]. Although primarily focused on model organisms such as bacteria and yeast, more recent methods have studied gene essentiality in humans using CRISP-R and gene trapping methods [17]. Because of the expense and labor involved with the experimental approaches, computational methods have been used to predict essential genes, and machine learning methods which rely on training a classifier using examples of essential genes have become a common strategy [18, 19]. Several features have been used to predict essential genes including both features of the DNA sequence such as GC content, and predicted features of the translated proteins such as hydrophobicity. All of the features may be expected to be have predictive value for essentiality, but none are particularly strong predictors. By combining these measures, the hope is to predict essential genes with a much higher degree of accuracy than would be possible with any one measure alone.

Alexandridis et al. [20] describe an ad hoc method for semi-supervised mixture modeling with incomplete training data. In Section , we build on their work and the more rigorous approach of [21] to develop semi-supervised hierarchical mixture models for any type of training data, specifically designed to deal with the case when only positive examples such as known essential genes are available, often called "positive unlabeled learning" or PUL [22]. A mixture model for a single data source is described, followed by a hierarchical mixture model when there are multiple data sources. Then Sections **??** and **??** show that the inclusion of positively labeled samples in a semi-supervised learning framework improves our ability to detect genes of interest. We also show that our method outperforms other methods for PUL that are based on standard supervised learning approaches. Although our case study is on gene essentiality, our framework is general for any gene-centric problem with multiple data types and when there is a small set of positive labels.

Methods

We use a generative mixture model approach to represent classes of genes such as target vs. nontarget and essential vs. nonessential, with a hierarchical structure to represent different types of data and the relationships between them. Mixture models have a long history and a rigorous statistical framework for inference and prediction **REFERENCES**. Hierarchical mixture models have been applied in other contexts [23] and in a variety of bioinformatic applications [24, 25]. In this work, we examine models that are tree-structured directed acyclic graphs, in which the nodes represent random variables, which may be hidden or observed, and the edges represent the conditional dependence structure of the variables. This approach allows for simultaneous modeling of a wide range of data sources, with computationally efficient model fitting and easily interpretable results. We first describe the semi-supervised method for single mixture models and then extend that to hierarchical mixture models for data integration.

Single Mixture Model

A *generative* mixture model arises when we wish to model the distribution of one random variable X which depends on the value of another random variable Y, so we say that Y *generates* X. We assume Y is a univariate categorical random variable that can take on one of K categories (1,..., K), while X, which may be multivariate, can have any distribution. For notational compactness, let $f_y(x) = f(x|Y = y)$, and $f(x) = \sum_k p_k f_k(x)$, $k = 1, \ldots, K$. We also assume we observe X, but not Y—that is, Y is *hidden*. In our examples, typically $K = 2$, for example "non-essential" versus "essential." The model may also be represented graphically, as shown in **COMBINE MARGINAL AND HIERARCHICAL FIGURE?**. Our challenge is to infer the parameters θ (e.g., Gaussian mean and variance for each mixture component), which will allow us to calculate expected values for these hidden states. The joint density of X and Y is therefore $f(x, y|\theta) = p_y \ f_y(x|\theta)$.

From this, for a sample $X = (x_1, \ldots, x_N)$, we use the EM algorithm [26, 27] to estimate the parameters and find the posterior probabilities $\hat{w}_{n,y} = P(y_n = y|x_n, \hat{\theta})$. Specifically, the EM algorithm finds the maximum likelihood estimate $\hat{\theta}$ by iterative maximization of the "Q-function," or the conditional expected log-likelihood

$$Q(\theta|\theta^{(i-1)}) = E_Y \left[\sum_n \log f(x_n, y_n|\theta) \mid X, \theta^{(i-1)} \right] \tag{1}$$

where $\theta^{(i-1)}$ is the previous iteration's i estimate for the parameters. For the model,

$$Q(\theta|\theta^{(i-1)}) = \sum_{n,k} w_{n,k} \{ \log p_k + \log f_k(x_n|\theta^{(i-1)}) \} \tag{2}$$

where

$$w_{n,y} \ = \ P(y_n = y|x_n, \theta^{(i-1)}) \ = \ \frac{p_y^{(i-1)} f_y(x_n|\theta^{(i-1)})}{\sum_k p_k^{(i-1)} f_k(x_n|\theta^{(i-1)})}. \tag{3}$$

Generally, $w_{n,1}$ is the posterior probability that $y_n = 1$ given the data. In the problems at hand, $w_{n,1}$ is the posterior probability that gene n is in the more "interesting" category given the particular data source being used.

Depending on the data type and the distribution, different functional forms for f may be appropriate (e.g., discrete, continuous). In addition within a data type, different alternatives may be available. For example, the Gaussian may be compared to a longer tailed distribution like the Pearson's VII or the Poission distribution may be compared to the negative binomial [28]. For model selection among forms of f, we use the ICL-BIC (integrated complete likelihood Bayesian information criterion) [29]. See Supplementary Material for more details.

Training Data

When training labels are available for some of the samples—for example, we know a number of essential genes in *S. cerevisiae* —we may wish to incorporate this information into the estimation procedure. The authors in [20] describe a method for the standard categorical mixture models, which we refer to here as the single mixture model: briefly, at the end of each E-step, update the posterior probabilities

for the labeled samples based on their labels. For example, if the nth gene is known to be in category 1 ($y_n = 1$) then we would set $w_{n,1} = 1$ and $w_{n,k} = 0 \; \forall \; k \neq 1$, regardless of the values of $(w_{n,1}, \ldots, w_{n,K})$ calculated previously. However, this method does not extend well to models with multiple data sources. [21] present a more principled approach of incorporating training data into the model as an additional type of observed data, which they apply in a support vector machine (SVM) context. Here we apply their approach in a mixture model context to extend both the single and hierarchical mixture models, with an emphasis on the positive unlabeled learning (PUL) context in which only positive examples are known.

Single Mixture Model: Semi-Supervised Method

In the presence of partial training data, such as when only a few positive labels are known, define a new random variable, T, to represent the training labels, in addition to the observed data X and hidden states Y. T is a categorical random variable that can take on the values from $1, \ldots, K^{trn}$, where $K^{trn} = K + 1$. If the training label is known for an observation, then $T \leq K$. When the training label is unknown for an observation, then $T = K^{trn}$, which is always the case in the unsupervised model. In other words, $P(Y = T | T \leq K) = 1$ always, while $P(Y = y | T = K^{trn})$ is a free parameter to be estimated. Let $\vec{R}^{trn}$ be the $K^{trn} \times K$ matrix such that $r^{trn}_{t,y} = f(y|t) = P(Y = y | T = t)$, and observe that the top K rows of $\vec{R}^{trn}$ are fixed at $\vec{I}_K$, the $K \times K$ identity matrix, while the K^{trn}th row is free:

$$\vec{R}^{trn} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ r^{trn}_{K^{trn},1} & \cdots & r^{trn}_{K^{trn},K} \end{pmatrix} \tag{4}$$

subject to the constraint $\sum_k r^{trn}_{K^{trn},k} = 1$. In contrast to fully supervised learning methods, this formulation allows us to estimate parameters using information from both labeled and unlabeled samples simultaneously, and also to make use of label information when only positive labels are available. We show in Section **??** that this approach can significantly improve the performance of the model.

We incorporate the labels into the model as the sample $\vec{t} = (t_1, \ldots, t_N)$. For example, for our example application ($K = 2$ for "essential" or "non-essential"), we may have some genes known to be essential while others are of unknown binding status. Then for $m, n \in \{1, \ldots, N\}$, $t_m = 1$ when we know that the mth gene is essential, while $t_n = 3$ when the status of the nth gene is unknown. If we knew the mth gene to be *un*essential, we would have $t_m = 2$, but our analysis here does not consider the case of labeled negative samples. Conceptually, Y generates both T and X; those samples for which $T \leq K$ may be thought of as samples for which Y is observed rather than hidden. We assume the values of T are accurate, that is, there are no unlabeled samples. The model is illustrated graphically in Figure **??** [**NEW FIGURE**].

However, because of the fixed relationship between Y and $T \leq K$, it is more practical to perform most of the calculations on the model as though T generated

Y. To obtain $\hat{\vec{R}}^{trn}$, we need only find the MLE for the K^{trn}th row of $\vec{R}^{trn}$, that is, $\hat{\vec{r}}^{trn}_{K^{trn}} = (\hat{r}^{trn}_{K^{trn},1}, \ldots, \hat{r}^{trn}_{K^{trn},K})$. The joint density of all variables in the model is

$$f(t, \vec{x}, y|\theta) = p(T = t)p(Y = y|T = t)p(X = x|Y = y, T = t) \tag{5}$$

$$f(t, \vec{x}, y|\theta) = p^{trn}_t \; r^{trn}_{T,t} \; f_y(\vec{x}|\theta). \tag{6}$$

where $p^{trn}_t = P(T = t)$, and the Q-function is

$$\begin{aligned} Q(\theta|\theta^{(i-1)}) \;\; = \;\; & \textstyle\sum_{n,k^{trn}} t'_{n,k^{trn}} \log p^{trn}_{k^{trn}} \\ & + \textstyle\sum_{n,k^{trn},k} t'_{n,k^{trn}} w_{n,k} \log r^{trn}_{k^{trn},k} \\ & + \textstyle\sum_{n,k} w_{n,k} \log f_k(\vec{x}_n). \end{aligned} \tag{7}$$

Here $t'_{n,t} = I(t_n = t)$, and after some simplification, the central calculation for the E-step is

$$w_{n,y} \;\; = \;\; P(y_n = y|t_n, \vec{x}_n, \theta^{(i-1)}) \;\; = \;\; \frac{r^{trn(i-1)}_{t_n,y} \; f_y(\vec{x}_n|\theta^{(i-1)})}{\sum_k r^{trn(i-1)}_{t_n,k} \; f_k(\vec{x}_n|\theta^{(i-1)})}. \tag{8}$$

The modeling of the observed data is the same as in the unsupervised case. By default, labeled samples are given the same weight as unlabeled samples in the parameter estimations. However, if we have a small training sample, we may choose to assign a higher weight w^{trn} to labeled samples. Please see the supplementary material for selection of weights.

Hierarchical Mixture Models

The previous model describes the case when there is only a single data type (e.g., one normally distributed variable) or a multivariate distribution of the same type (e.g., multivariate normal). But for many integration problems, as with predicting essential genes, data sets may need to be grouped by distribution type (e.g., continuous or discrete) and measurement type (e.g., expression levels or binding information). For example, there are binary variables such as location information (e.g., is the predicted subcellular location of the corresponding protein in the nucleus) or continuous variables like expression **TABLE**. We present a hierarchical mixture model extending the framework above for any number and types of genomic level data.

At the top of the hierarchies shown in **Figure ??** is the hidden categorical random variable Y_0, which takes on integer values from 1 to K_0 for some integer $K_0 > 1$. In the problems at hand, we assume $K_0 = 2$ always and $Y_0 = 1$ corresponds to "interesting" genes. Next, let Z denote the number of data sources, and $z \in \{1, \ldots, Z\}$ denote the zth data source. The intermediate hidden categorical random variables Y_z take on integer values from 1 to K_z for some integer $K_z > 1$. The distributions of the Y_z's depend—directly or indirectly, depending on the model topology—on the value of Y_0. Also define the observed random variables $X_1, \ldots, X_Z$ (each X_z may be multivariate) where the distribution of X_z depends only on the value of Y_z. That

is, each X_z is generated by Y_z. For example, in the *Saccharomyces* data, $Z = 14$ with each value of z corresponding to a different sequence-derived measure. Each Y_0 generates $Y = (Y_1, \ldots, Y_Z)$. This model treats all observed variables as equally important to estimating the distribution of Y_0. We have explored alternative conditional relationships among the Y's in [28].

Given N genes, for $n = 1, \ldots, N$ the estimated posterior probability that the nth gene is of interest is $P(y_{0,n} = 1|\vec{x}_{\cdot,n}, \hat{\theta})$. Here $y_{0,n}$ is the nth hidden status variable, that is, a realization of Y_0. Similarly, $y_{z,n}$ is the hidden realization of Y_z for the nth gene, and $\vec{x}_{\cdot,n} = (\vec{x}_{1,n}, \ldots, \vec{x}_{Z,n})$ is the observed data for the nth gene, with $\vec{x}_{z,n}$ being a realization of X_z. Finally, $\hat{\theta}$ denotes the estimated parameters of the model. See the supplementary material for full details of the estimation procedure. Briefly, the first step in the hierarchical model fitting is to fit a single mixture model to each data source, as described in Section , to choose the number f components K_z and marginal distribution which will be used for that data. These steps follow the EM algorithm described above, and are used for initialization; the individual model fits can be updated in the full algorithm. Then after the marginal distributions are selected, this is used for initialization of the EM algorithm for the full model. The E-step consists of finding the posterior probabilities, given the data and current iteration of θ, for the hidden states Y_z for all data sources z, and for the primary hidden state Y_0. The M-step is a straightforward maximum likelihood estimation for and the parameters θ, the marginal distribution of $P(Y_0 = y_0)$, and conditional relationships among the hidden variables $P(Y_z = y_z|Y_0 = y = 0)$. More details are provided in the Supplementary Material.

Sub-sub heading for section
Text for this sub-sub-heading ...

Sub-sub-sub heading for section Text for this sub-sub-sub-heading ... In this section we examine the growth rate of the mean of Z_0, Z_1 and Z_2. In addition, we examine a common modeling assumption and note the importance of considering the tails of the extinction time T_x in studies of escape dynamics. We will first consider the expected resistant population at vT_x for some $v > 0$, (and temporarily assume $\alpha = 0$)

$$E\big[Z_1(vT_x)\big] = E\left[\mu T_x \int_0^{v\wedge 1} Z_0(uT_x)\exp\big(\lambda_1 T_x(v-u)\big)\,du\right].$$

If we assume that sensitive cells follow a deterministic decay $Z_0(t) = xe^{\lambda_0 t}$ and approximate their extinction time as $T_x \approx -\frac{1}{\lambda_0}\log x$, then we can heuristically estimate the expected value as

$$\begin{aligned} E\big[Z_1(vT_x)\big] &= \frac{\mu}{r}\log x \int_0^{v\wedge 1} x^{1-u}x^{(\lambda_1/r)(v-u)}\,du \\ &= \frac{\mu}{r}x^{1-\lambda_1/\lambda_0 v}\log x \int_0^{v\wedge 1} x^{-u(1+\lambda_1/r)}\,du \\ &= \frac{\mu}{\lambda_1-\lambda_0}x^{1+\lambda_1/rv}\left(1-\exp\left[-(v\wedge 1)\left(1+\frac{\lambda_1}{r}\right)\log x\right]\right). \quad (9) \end{aligned}$$

Thus we observe that this expected value is finite for all $v > 0$

Application to Data

Methods

Feature Description

Identifying essential genes that is, genes without which the organism cannot survive in the yeast Saccharomyces cerevisiae [30] and related species has been an area of investigation. Several features have been used to predict essential genes including both features of the DNA sequence such as GC content, and predicted features of the proteins translated from the genes such as hydrophobicity and subcellular localization. All of the features may be expected to be have predictive value for essentiality, but none are particularly strong predictors. By combining these measures, the hope is to predict essential genes with a much higher degree of accuracy than would be possible with any one measure alone.

To predict essentiality of the genes in yeast Saccharomyces cerevisiae, our analysis uses features associated with each gene from two sources: 1) fourteen sequence-derived features from the Seringhaus, et al., 2006 study, and 2) eight additional features from the Ensembl website [31]. We added new features to check how sensitive the results were to the feature set. Feature definitions are listed in Table 1. Three of 22 features (vacuole, in how many of 5 proks blast, and intron) were removed from the analysis due to low content (less than 5% of non-zero values). At lower training sizes, their low content resulted in deterministic models because the randomization employed multiple seeds until the design matrix had no columns of zeroes. Genes which had values for this low-content features would have been selected more frequently than genes without information in these features.

Cross-Validation Strategy

In the unsupervised simulation, semi-supervised was compared against the unsupervised method described in Dvorkin et. al., [12]. The unsupervised method utilizes both K-means and C-means clustering in a Bayesian Belief Network to classify essentiality of the genes in Saccharomyces cerevisiae. We trained our methods from the set of 769 essential genes (positive labels) of the total 3500 genes. Additionally, we contrasted all 22 features against a subset of 14 sequenced-derived features as predictors of essentiality. Training set sizes were based on increments of 5 with minimum set sizes greater than the number of features to prevent rank deficiency in training sets. Iterations (n=30) were used to average the AUC or other metrics used to evaluate performance.

The cross-validation strategy for the supervised case incorporates an unbalanced strategy to the test set 1 along with a contamination rate. For an unbalanced design, test sets utilize the remaining genes not used in the training sets rather than a balanced strategy which matches training and testing set sizes. The unbalanced strategy was chosen because, in practice, an investigator would typically want to test all the remaining genes for essentiality rather than just a subset of genes. Another concept interwoven into the analysis is contamination. Contamination considers the dilemma from falsely assigned genes in the training sets. The semi-supervised and unsupervised methods do not consider negative labels in their computations and,

thus, are unaffected by contamination. For supervised methods, positive labels in the training set are mixed with negative labeled genes for analysis when varying percentages of contamination are introduced.

In the supervised simulation, semi-supervised was compared against three supervised methods (LASSO, SVM, and Random Forest) at low training set sizes. AUC performance of these four methods was compared across training set sizes between 1% (n=35) and 10% (n=350) from all 3500 genes. Genes randomly chosen for the supervised training sets reflect the same ratio of positive and negative labels as seen in the full data set. Among the 3500 yeast genes, there are 769 essential genes resulting in a 21% ratio. Therefore, as an example, at 1% training size, 35 randomly chosen genes contained 7 positive labels (21% of 35) and 28 negative labels for supervised methods, while semi-supervised methods analyzed 35 positively labeled genes. In order to mimic contamination, negative labels were reassigned a positive label at rates of 0%, 20%, and 50%.

For all results, unique initial seeds were chosen based on the iteration number, training set size, and contamination (for supervised comparison only). Iterations were increased to 100 to better discriminate effects at low training sets. Once the cross-validation data was generated by a seed, the same data was used to compare each method.

Algorithms

All simulations were performed in R version 3.3.3. The semi-supervised and unsupervised analysis utilized functions from the lcmix package. The lcmix package developed and implemented in the previous paper by Dvorkin, Biehl, and Kechris **A Graphical Model Method for Integrating Multiple Sources of Genome-scale Data** [12] and can be downloaded from http://r-forge.r-project.org/projects/lcmix/. LASSO was performed using the glmnet command in the glmnet package [32]. Using cv.glmnet, k-fold cross validation optimized the minimum lambda for the LASSO function. SVM analysis used the svm command under the e1071 package [33]. Various runs using different criteria revealed a radial kernel density and C-classification optimized AUC performances. Random Forest was performed with the randomForest command under the randomForest package [34] All supervised predictions used the predict command in the stats package.

Performance

The AUC mean, variance, and CV (median absolute deviation/median) of the three supervised methods were contrasted against semi-supervised method. Because LASSO outperformed the other supervised models in AUC across all training set sizes and contamination rates, a closer evaluation of its performance was compared with semi-supervised method. In order to fairly contract LASSO performance to semi-supervised, the prediction scores were re-scaled to be between 0 and 1. Precision, recall, and f-measure further discriminated the two methods with four rescaled prediction score cutoffs including the median and prediction scores of 0.5, 0.8, and 0.95. The median cutoff is a relative measure based on the data while the other three cutoffs are absolute. The f-measure was calculated from the average precision and recall at each training set size from 1% to 5%.

Results

We describe each of the features used in the analysis along with their associated distributions based on their data types and the optimized determined **K**, univariate prediction classes, in predicting essentiality for each one in Table 2. Based on the range of univariate estimated AUCs (0.494-0.674), many of the features have low predictive value on their own but in the following comparisons we will explore their combined predictive power. We used cross-validation simulations to compare our hierarchical mixture model semi-supervised method with unsupervised and supervised methods. We hypothesize that semi-supervised method outperforms both unsupervised methods at any training set size and supervised methods at low training set sizes especially when positive labels are contaminated.

Unsupervised Comparison

The complete essentiality data for Saccharomyces cerevisiae contains n=769 positive labeled genes. To explore whether our conclusions were sensitive to the choice of features, we first used only 14 sequence-derived features from Seringhaus and then a larger set of 22 features, which included the Seringhaus features and eight additional features collected from Ensembl (see Methods). Semi-supervised performs better than unsupervised for AUC regardless of predicting with 14 sequence-derived or all 22 features or training set size (Figure 4). The AUC variance for both methods increases as training size increases when training on all essential genes. This is expected as the test set is relatively larger and more constant with the smaller training sets. The eight additional features added from Ensembl Biomart generally improves AUC performance and decreases variance for both methods.

Supervised Comparison

Next, we compared the semi-supervised method with a supervised strategy using all essential genes for the training set and all 22 features. Supervised algorithms require both positive and negative labels. Therefore, we picked a random set of the non-essential genes to be the negative labels (Figure 1), but also included some contamination (some essential genes labeled as negatives in the training set) since in practice, the complete set of negative labels will not be known in many situations.

LASSO and semi-supervised outperforms the other two supervised methods - SVM and Random Forest (Figure 8). At low training sizes (i.e., 2%, n = 70), semi-supervised method has a higher mean AUC than the three supervised methods. LASSO does not match the stability (lower variance) of semi-supervised until around 5% (n=175) training set size. However, for larger training set sizes, the AUC variance of semi-supervised increases while variance from LASSO slightly decreases. As contamination increases, all three supervised methods decrease in performance. At 50% contamination, semi-supervised method bests all methods across all training set sizes (up to 10%). The CV (median absolute deviance/median) for semi-supervised is lower than LASSO across all contamination levels and training set sizes up to 5% (Figure 12).

Semi-supervised versus LASSO Performance

To compare the best performing supervised method, LASSO, for our data to semi-supervised method, prediction scores were rescaled to be between 0 and 1. At 1%

training level, LASSO kernel densities of prediction scores exhibit an unimodal distribution while semi-supervised methods exhibit bi- or multi-modal behaviors (Figure 15). Uni-modal behavior makes it more difficult to find better separation of gene types (e.g., essential versus non-essential). As training level increases, LASSO kernel densities of prediction scores continue to exhibit unimodal distributions while semi-supervised methods maintain their multimodal behaviors.

Focusing on 0% contamination in Figure 20, the three absolute cutoffs (50%, 80%, and 95%) reveal a higher recall across all training set sizes for semi-supervised and the median cutoff shows semi-supervised outperforming LASSO up to 3% at which they become comparable. Also, up to 3%, semi-supervised outperforms LASSO in precision at the median cutoff. Precision generally increases as the absolute cutoff increases with LASSO besting semi-supervised as training set size increases. Contamination reduces all three performance measures (precision, recall, f-measure) for LASSO across training set sizes from 1% to 5% and all four cutoffs. Irrespective of contamination, f-measure for semi-supervised outperforms LASSO for all training set sizes and cutoffs.

Conclusion and Discussion for Simulation

The focus of the simulation was to evaluate real world genomic scenarios where researchers may only have a small number of known positive labels to train their data sets such as in the Drosphilia study [35] or the pre-2002 essential genes in the Seringhaus paper. In addition, simulating contamination of these known genes expounded upon the unknown possibilities that could occur. To clarify, the contamination we simulated was specific to falsely labeling positive genes as negative and not the reverse. We argue this to be the more probable scenario in the essentiality classification because the likelihood of yeast with a knock-out, truly essential gene surviving regardless of laboratory conditions to be nearly zero. Yet, poorly nutrient media, uncontrolled temperature regulation during growth phase, or an unknown, unintentional intervention may result in the misclassification from essential to non-essential. The comparison of semi-supervised to unsupervised and supervised methods under the condition of high-dimensional datasets with multiple data types and sources revealed where semi-supervised has its advantages. Semi-supervised outperformed unsupervised method across a wide range of training set sizes of the essential genes, various feature selections and data sources. Since semi-supervised utilizes the knowledge of having positive labels, it outperforms unsupervised in AUC mean but at the cost of some computational time. Contamination does not play a role in either unsupervised or semi-supervised methods. However, the misclassification of negative labels causes turbulence in the algorithms of supervised methods and can be seen in the loss of AUC mean as contamination increases (Figure 8). In general, the supervised methods such as LASSO displayed unimodal distributions of their predicted probabilities. In contrast, the multi-modality of our semi-supervised prediction scores provides a more natural cutoff than the unimodal distribution displayed by LASSO in Figure 15. Regardless of contamination rates, the prediction scores for semi-supervised outperformed LASSO in AUC and f-measure at training set sizes below 3%.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
Text for this section ...

Acknowledgements
Text for this section ...

Author details
[1]Computational Bioscience Program, University of Colorado School of Medicine, 12801 E. 17th Ave., 80045 Aurora, US. [2]Department of Biostatistics and Informatics, Colorado School of Public Health, 80045 Aurora, US.

References

1. The ENCODE Project Consortium: The ENCODE (ENCyclopedia Of DNA Elements) project. Science **306**(5696), 636–640 (2004). doi:10.1126/science.1105136
2. The ENCODE Project Consortium: An integrated encyclopedia of DNA elements in the human genome. Nature **489**(7414), 57–74 (2012). doi:10.1038/nature11247
3. Celniker, S.E., Dillon, L.A., Gerstein, M.B., Gunsalus, K.C., Henikoff, S., *et al.*: Unlocking the secrets of the genome. Nature **459**(7249), 927–930 (2009). doi:10.1038/459927a
4. National Cancer Institute: The Cancer Genome Atlas. Retrieved May 26, 2013, from http://cancergenome.nih.gov/ (2013)
5. Hamid, J.S., Hu, P., Roslin, N.M., Ling, V., Greenwood, C.M.T., Beyene, J.: Data integration in genetics and genomics: Methods and challenges. Human Genomics and Proteomics **1**(1) (2009). doi:10.4061/2009/869093
6. Hawkins, R.D., Hon, G.C., Ren, B.: Next-generation genomics: an integrative approach. Nature Reviews Genetics **11**(7), 476–486 (2010). doi:10.1038/nrg2795
7. Lemmens, K., Dhollander, T., De Bie, T., Monsieurs, P., Engelen, K., *et al.*: Inferring transcriptional modules from ChIP-chip, motif and microarray data. Genome Biology **7**(5), 37 (2006)
8. Tyekucheva, S., Marchionni, L., Karchin, R., Parmigiani, G.: Integrating diverse genomic data using gene sets. Genome Biology **12**(10), 105 (2011). doi:10.1186/gb-2011-12-10-r105
9. Xie, Y., Pan, W., Jeong, K.S., Xiao, G., Khodursky, A.B.: A Bayesian approach to joint modeling of protein-DNA binding, gene expression and sequence data. Statistics in Medicine **29**(4), 489–503 (2010). doi:10.1002/sim.3815
10. Qin, J., Li, M.J., Wang, P., Zhang, M.Q., Wang, J.: ChIP-Array: combinatory analysis of ChIP-seq/chip and microarray gene expression data to discover direct/indirect targets of a transcription factor. Nucleic Acids Research **39**(suppl 2), 430–436 (2011). doi:10.1093/nar/gkr332
11. Hoffman, M.H., Buske, O.J., Wang, J., Weng, Z., Bilmes, J.A., Noble, W.S.: Unsupervised pattern discovery in human chromatin structure through genomic segmentation. Nature Methods **9**, 473–476 (2012). doi:10.1038/nmeth.1937
12. Dvorkin, D., Biehs, B., Kechris, K.: A graphical model method for integrating multiple sources of genome-scale data. Statistical Applications in Genetics and Molecular Biology **12**(4), 469–487 (2013)
13. Zhang, Z., Ren, Q.: Why are essential genes essential? - the essentiality of saccharomyces genes. Microbial Cell **2**(8), 280–287 (2015). doi:10.15698/mic2015.08.218
14. Juhas, M., Eberl, L., Glass, J.I.: Essence of life: essential genes of minimal genomes. Trends in Cell Biology **21**(10), 562–568 (2011). doi:10.1016/j.tcb.2011.07.005
15. Luo, H., Lin, Y., Gao, F., Zhang, C.T., Zhang, R.: Deg 10, an update of the database of essential genes that includes both protein-coding genes and noncoding genomic elements. Nucleic Acids Research **42**, 574–580 (2014). doi:10.1093/nar/gkt1131
16. WH, C., P, M., MJ, L., P., B.: Ogee: an online gene essentiality database. Nucleic Acids Research **40**, 901–906 (2012). doi:10.1093/nar/gkr986
17. Wang, T., Birsoy, K., Hughes, N.W., Krupczak, K.M., Post, Y., Wei, J.J., Sabatini, D.M., *et al.*: Identification and characterization of essential genes in the human genome. Science **350**(6264), 1096–1101 (2015). doi:10.1126/science.aac7041
18. Mobegi, F.M., Zomer, A., de Jonge, M.I., van Hijum, S.A.F.T.: Advances and perspectives in computational prediction of microbial gene essentiality. Briefings in Functional Genomics **16**(2), 70–79 (2017). doi:10.1093/bfgp/elv063
19. X, Z., ML, A., N, L.: Predicting essential genes and proteins based on machine learning and network topological features: A comprehensive review. Frontiers in Physiology **7**(75) (2016). doi:10.3389/fphys.2016.00075
20. Alexandridis, R., Lin, S., Irwin, M.: Class discovery and classification of tumor samples using mixture modeling of gene expression data—a unified approach. Bioinformatics **20**(16), 2545–2552 (2004). doi:10.1093/bioinformatics/bth281
21. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining, pp. 213–220 (2008)
22. He, J., Zhang, Y., Li, X., Wang, Y.: Naive Bayes classifier for positive unlabeled learning with uncertainty. In: Proceedings of the Tenth SIAM International Conference on Data Mining, pp. 361–372 (2010)
23. Vermunt, J.K., Magidson, J.: Hierarchical mixture models for nested data structures. In: Classification–the Ubiquitous Challenge: Proceedings of the 28th Annual Conference of the Gesellschaft Für Klassifikation eV, University of Dortmund, March 9-11, 2004, vol. 28, p. 240 (2005). Springer
24. Jörnsten, R., Keleş, S.: Mixture models with multiple levels, with application to the analysis of multifactor gene expression data. Biostatistics **9**(3), 540–554 (2008). doi:10.1093/biostatistics/kxm051
25. Li, Q., MacCoss, M.J., Stephens, M.: A nested mixture model for protein identification using mass spectrometry. The Annals of Applied Statistics **4**(2), 962–987 (2010). doi:10.1214/09-AOAS316

26. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society Series B **39**(1), 1–38 (1977)
27. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions, 2nd edn. Wiley, Hoboken, New Jersey (2008)
28. Dvorkin, D.: Graphical model methods for integrating diverse sources of genome-scale data. PhD thesis, University of Colorado (2013)
29. Biernacki, C., Celeux, G., Govaert, G.: Assessing a mixture model for clustering with the integrated completed likelihood. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(7), 719–725 (2000). doi:10.1109/34.865189
30. Seringhaus, M., Paccanaro, A., Borneman, A., Snyder, M., Gerstein, M.: Predicting essential genes in fungal genomes. Genome Research **16**(9), 1126–1135 (2006). doi:10.1101/gr.5144106
31. Zerbino, D., *et al.*: Ensembl. Nucleic Acids Research **46**(D1), 754–761 (2018). doi:10.1093/nar/gkx1098
32. Jerome Friedman, R.T. Trevor Hastie: Regularization Paths for Generalized Linear Models Via Coordinate Descent. (2010). http://www.jstatsoft.org/v33/i01/
33. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.: E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. (2017). R package version 1.6-8. https://CRAN.R-project.org/package=e1071
34. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002)
35. Biehs, B., Kechris, K., Liu, S.M., Kornberg, T.B.: Hedgehog targets in the Drosophila embryo and the mechanisms that generate tissue-specific outputs of Hedgehog signaling. Development **137**(22), 3887–3898 (2010). doi:10.1242/dev.055871
36. M.B, G. http://www.gersteinlab.org/proj/predess/
37. R, J., D, G., M, G.: Relating whole-genome expression data with protein-protein interactions. Genome Research **12**(1), 37–46 (2002)
38. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics **6**(2), 461–464 (1978)
39. Ji, Y., Wu, C., Liu, P., Wang, J., Coombes, K.R.: Applications of beta-mixture models in bioinformatics. Bioinformatics **21**(9), 2118–2122 (2005). doi:10.1093/bioinformatics/bti318
40. Viroli, C.: Dimensionally reduced model-based clustering through mixtures of factor mixture analyzers. Journal of Classification **27**, 363–388 (2010)
41. Shao, J.: Linear model selection by cross-validation. Journal of the American Statistical Association **88**(422), 486–494 (1993)

Figures

Figure 1 Diagram of unbalanced design describing true label contamination in training sets for supervised methods. p_i, n_{ess}, n_{con}, and n_{non} represent training set size, total number of essential genes, number of contaminated genes, total number of non-essential genes, respectively. n_{con} is determined by contamination percentage of training set essential genes. Training and test sets are indicated above while true label of gene essentiality is indicted below figure. Simulation label describes the label assignment for analysis. Shaded area indicates the contamination of training set essential genes where simulation and true labels differ.

Tables

Additional Files

Additional file 1 — Sample additional file title

Model Selection for Marginal Distribution f: ICL-BIC

The BIC of [38] is defined as $BIC = 2\mathcal{L}_X(\hat{\theta}) - |\Theta| \log N$, with $\mathcal{L}_X(\hat{\theta})$ being the log-likelihood of the estimated parameters given the observed data and $|\Theta|$ being the size of the parameter space. $ICL - BIC$ is defined as $ICL - BIC = 2\mathcal{L}_{X,\hat{Y}}(\hat{\theta}) - |\Theta| \log N$, with $\hat{Y}$ being the maximum a posteriori (MAP) estimate of the value of the hidden data. Thus $ICL - BIC$ may be interpreted as the most probable value of BIC if all data were observed. All other things being equal, the model with the higher (often "less negative") $ICL - BIC$is preferred. See [39] for an application of this criterion to models of gene expression, and [40] for a comparison to other model selection criteria, where $ICL - BIC$ outperforms AIC (Akaike's information criterion), BIC, and other criteria in selecting the correct mixture model.

Selection of Weights w^{trn}

The modeling of the observed data is the same as in the unsupervised case. By default, labeled samples are given the same weight as unlabeled samples in the parameter estimations. However, if we have a small training sample, we may choose to assign a higher weight w^{trn} to labeled samples. For further (M-step) calculations involving the posterior probabilities calculated in Equation (8), we make the transformation $w_{n,y} \leftarrow w^{trn} w_{n,y}$ for each n such that $t_n \leq K$, while leaving $w_{n,y}$ as-is for each n such that $t_n = K^{trn}$. The effective result of this is to add "copies" of the labeled samples to the data set, thus increasing their influence on parameter estimation. For example, if we choose $w^{trn} = 2$, we are effectively doubling the size of the training data.

Although values of $w^{trn} > 1$ often lead to better parameter estimates and therefore to better model predictions, overfitting can occur if w^{trn} grows too large. We use Monte Carlo cross-validation [41] to choose the best value

Figure 2 Sequence derived features.

Figure 3 All features.

Figure 4 Boxplots of AUC at various training sizes of 769 essential genes using sequence derived (14) or all (22) features as predictors. Semi-supervised and unsupervised method are shown in red and blue, respectively.

Figure 5 0% Contamination

Figure 6 20% Contamination

Figure 7 50% Contamination

Figure 8 AUC comparison between semi-supervised and supervised methods at various training sizes with all essential genes in yeast using 19 features as predictors. 100 iterations were executed at training sets percentages (1, 1.5, 2, ... ,10) for all four methods and negative contamination levels (0% (a), 20% (b), and 50% (c)). Semi-supervised method is shown in red while the supervised methods (LASSO, SVM, and Random Forest) are shown in blue, aquamarine, and light blue, respectively.

Table 1 Feature Definitions.

	Abbreviation	Description	Type	Family	K
Sequence Derived Features	cytoplasm	Predicted subcellular location: cytoplasm	binary	bernoulli	2
	er	Predicted subcellular location: er	binary	bernoulli	2
	mitochondria	Predicted subcellular location: mitochondria	binary	bernoulli	2
	nucleus	Predicted subcellular location: nucleus	binary	bernoulli	2
	vacuole	Predicted subcellular location: vacuole	binary	bernoulli	2
	other	Predicted subcellular location: other	binary	bernoulli	2
	tm helix	Number of predicted transmembrane helices	integer	neg bin	2
	cai	Codon adaptation index	[real]	gamma	2
	l aa	Length of putative protein in amino acid	integer	neg bin	3
	nc	Effective number of codons	(real)	normal	2
	gravy	Hydrophobicity	(real)	normal	2
	gc	% GC content	[real]	gamma	2
	close ratio	% codons one-third base pairs from stop codon	[real]	gamma	2
	rare aa ratio	% of rare aa in translated ORF	[real]	gamma	2
Additional Features	intxn partners	Number of interaction proteins	integer	neg bin	3
	6 yeast blast	Number of related genes in 6 species of yeast	integer	poisson	2
	blast yeast	Number of related genes in yeast BLAST	integer	neg bin	2
	dovexpr	Dov Expression	(real)	pearson	3
	chromosome	Chr number	integer	poisson	2
	chr position	Chr position as % of chromosome length	[real]	gamma	2
	5 proks blast	Number of related genes in 5 prokaryotes BLAST	integer	poisson	2
	intron	Contains an intron in DNA/RNA sequence	binary	bernoulli	2

Table 2 Description of Features. The sequence-derived features were compiled by Seringhaus [30]. Additional features were assembled from the Gerstein labs [36]. Dov expression is the normalized difference between absolute mRNA expression levels [37]. Closed and open brackets indicate closed and open sets, respectively, under the **Type** heading. **Family** describes the distribution the marginal models utilized in the semi-supervised method. Note: chr position is labeled gamma which is a special case of the beta distribution. **K** is the calculated univariate optimized number of predicted classes for each variable.

Figure 9 AUC MAD/Median

Figure 10 AUC Median

Figure 11 AUC Median Absolute Deviation

Figure 12 Evaluation of summary statistics comparing semi-supervised and supervised methods. 100 iterations were executed at training sets (1%, 1.5%, 2%,...,5%) for all four methods and negative contamination levels (0%, 20%, and 50%). Semi-supervised method is shown in red while the supervised methods (LASSO, SVM, and Random Forest) are shown in blue, aquamarine, and light blue, respectively. The AUC mean (a), variance (b), and CV (c) contrasts the four methods at each training set size across three contamination rates. CV is calculated as the Median Absolute Deviation (MAD)/Median

Figure 13 LASSO

Figure 14 Semi-Supervised

Figure 15 Density plot of predicted scores juxtaposing semi-supervised and LASSO methods. LASSO (a) and Semi-Supervised (b) methods display kernel densities for true positive (dashed) and negative (solid) labels at the 1% training set level and 0% contamination rate.

from a list of candidate values, currently $w^{trn} \in \{1, 5, 10, 20, 50, 100\}$. For a specified number of replications, currently 30, we sample without replacement half the training data, leaving the other half to serve as testing data for the current replication. We then train the model with the first half of the data at each candidate weight, and calculate the receiver operating characteristic (ROC) area under the curve (AUC) for the trained models at each candidate weight. The weight with the highest mean ROC-AUC across all replications is chosen as the final value of w^{trn}. **DD: THIS IS CROSS-VALIDATION WITH 1/1, TRY 2/1? What is used in the package?**

EM Algorithm for Hierarchial Mixture Model

Details for the estimation of parameters and conditional probabilities for the hidden variables are provided in this section. The unconditional status probability is $p_{0,y_0} = P(Y_0 = y_0)$, where Y_0 generates the distribution for the Y_z's, and the component probability given the status is $q_{z,y_0,y_z} = P(Y_z = y_z|Y_0 = y_0)$. Given observed data $\vec{X} = (\vec{X}_1, \ldots, \vec{X}_z)$ where $\vec{X}_z = (\vec{x}_{z,1}, \ldots, \vec{x}_{z,n})$, and parameters $\theta^{(i-1)}$, denote the conditional probabilities for the hidden variables by

$$\begin{array}{rcl} u_{n,y_0} & = & P(y_{0,n} = y_0|\vec{x}_{\cdot,n}, \theta^{(i-1)}), \\ v_{z,y_0,n,y_z} & = & P(y_{0,n} = y_0, y_{z,n} = y_z|\vec{x}_{\cdot,n}, \theta^{(i-1)}) \text{ or} \\ w_{z,n,y_z} & = & P(y_{z,n} = y_z|\vec{x}_{\cdot,n}, \theta^{(i-1)}). \end{array} \tag{10}$$

Let $I(\mathcal{P})$ denote the indicator function. Then for $\vec{X}$ as above, and hidden data $\vec{Y} = (\vec{y}_1, \ldots, \vec{y}_Z)$ where $\vec{y}_z = (y_{z,1}, \ldots, y_{z,n})$ with $\vec{y}_0 = (y_{0,1}, \ldots, y_{0,n})$, the complete data log-likelihood is

$$\begin{array}{rcl} \mathcal{L}_{X,Y,y_0}(\theta) & = & \sum_{n,k_0} I(y_{0,n} = k_0) \log p_{0,k_0} \\ & & + \sum_{n,z,(k),k_z} (I) \log q_{z,(k),k_z} \\ & & + \sum_{n,z,k_z} I(y_{z,n} = k_z) \log f_{k_z}(x_{z,n}|\theta). \end{array} \tag{11}$$

where (k) denotes k_0 and (I) denotes $I(y_{0,n} = k_0, y_{z,n} = k_z)$. The Q-function is thus

$$\begin{array}{rcl} Q(\theta|\theta^{(i-1)}) & = & \sum_{n,k_0} u_{n,k_0} \log p_{0,k_0} \\ & & + \sum_{n,z,(k),k_z} v_{z,(k),n,k_z} \log q_{z,(k),k_z} \\ & & + \sum_{n,z,k_z} w_{z,n,k_z} \log f_{k_z}(\vec{x}_{z,n}|\theta^{(i-1)}). \end{array} \tag{12}$$

Figure 16 Median Cutoff

Figure 17 0.50 Predicted Score Cutoff

Figure 18 0.80 Predicted Score Cutoff

Figure 19 0.95 Predicted Score Cutoff

Figure 20 Performance of Semi-supervised and LASSO methods with predicted score cutoffs at median, 0.50, 0.80, and 0.95 at 0% contamination. Semi-supervised method is shown in red while LASSO is shown in blue. Precision, recall, and f-measures are represented by dotted, dashed, and solid lines, respectively. The median is a relative cutoff while the other cutoffs represent absolute cutoffs with re-scaled predicted probabilities.

Supplementary Figure 1 AUC SD/mean

Supplementary Figure 2 AUC Mean

Supplementary Figure 3 AUC Variance

Supplementary Figure 4 Evaluation of summary statistics comparing semi-supervised and supervised methods. 100 iterations were executed at training sets (1%, 1.5%, 2%,...,5%) for all four methods and negative contamination levels (0%, 20%, and 50%). Semi-supervised method is shown in red while the supervised methods (LASSO, SVM, and Random Forest) are shown in blue, aquamarine, and light blue, respectively. The AUC mean (a), variance (b), and CV (c) contrasts the four methods at each training set size across three contamination rates. CV is calculated as the Median Absolute Deviation (MAD) / Median

The first step in joint model fitting is to fit a single mixture model to each data source, as described in Section , to choose the number of components K_z and marginal distribution which will be used for that data. Then the initialization, execution, and output of the EM algorithm as adapted for the model topologies are as follows:

1 Initialize the parameters for the hierarchical model based on the selected individual mixture models. Note that the individual model fits are used for initialization only, and do not imply any hard categorization of the observed data before fitting the hierachical model.

2 E-step: for the ith iteration, using the previous iteration's parameter estimates $\theta^{(i-1)}$, estimate the conditional probabilities defined in Equation (10), which are

$$\begin{aligned} u_{n,y_0} &= \frac{p_{y_0}^{(i-1)} \prod_z \sum_{k_z} q_{z,y_0,k_z}^{(i-1)} g_{z,n,k_z}}{\sum_{k_0} p_{k_0}^{(i-1)} \prod_z \sum_{k_z} q_{z,k_0,k_z}^{(i-1)} g_{z,n,k_z}}, \\ v_{z,y_0,n,y_z} &= u_{n,y_0} \frac{q_{y_0,y_z}^{(i-1)} g_{z,n,y_z}}{\sum_{k_z} q_{z,y_0,k_z}^{(i-1)} g_{z,n,k_z}}, \\ w_{z,n,y_z} &= \sum_{k_0} v_{z,k_0,n,y_z} \end{aligned} \tag{13}$$

where $g_{z,n,y_z} = f_{y_z}(\vec{x}_{z,n}|\theta^{(i-1)})$.

3 M-step: estimate the current iteration's parameters, $\theta^{(i)} = \arg\max_\theta Q(\theta|\theta^{(i-1)})$. This is a straightforward maximum likelihood estimation for the p's and q's, and a weighted MLE for the parameters relating to the observed variables, using weights $\vec{w}_{z,\cdot,y_z}$ and data $\vec{X}_z$.

4 Repeat steps 2 and 3 until convergence.

5 Report the final estimated parameters $\hat{\theta}$ and posterior status probabilities $\hat{\vec{U}}$, the $N \times K_0$ matrix of which the (n, y_0)th element is $\hat{u}_{n,y_0} = P(y_{0,n} = y_0|\vec{x}_n, \hat{\theta})$. Specifically, $\hat{u}_{n,1}$ is the estimated probability, given the data and the final estimated parameters, that the nth gene is a gene of interest.

Additional file 2 — Sample additional file title
Additional file descriptions text.

Supplementary Figure 5 Median Cutoff

Supplementary Figure 6 0.50 Predicted Score Cutoff

Supplementary Figure 7 0.80 Predicted Score Cutoff

Supplementary Figure 8 0.95 Predicted Score Cutoff

Supplementary Figure 9 Performance of Semi-supervised and LASSO methods with predicted score cutoffs at median, 0.50, 0.80, and 0.95 at 20% contamination. Semi-supervised method is shown in red while LASSO is shown in blue. Precision, recall, and f-measures are represented by dotted, dashed, and solid lines, respectively. The median is a relative cutoff while the other cutoffs represent absolute cutoffs with re-scaled predicted probabilities.

Supplementary Figure 10 Median Cutoff

Supplementary Figure 11 0.50 Predicted Score Cutoff

Supplementary Figure 12 0.80 Predicted Score Cutoff

Supplementary Figure 13 0.95 Predicted Score Cutoff

Supplementary Figure 14 Performance of Semi-supervised and LASSO methods with predicted score cutoffs at median, 0.50, 0.80, and 0.95 at 50% contamination. Semi-supervised method is shown in red while LASSO is shown in blue. Precision, recall, and f-measures are represented by dotted, dashed, and solid lines, respectively. The median is a relative cutoff while the other cutoffs represent absolute cutoffs with re-scaled predicted probabilities.