ProteinLasso: A Lasso regression approach to protein inference problem in shotgun proteomics

Ting Huang, Haipeng Gong, Can Yang, Zengyou He

School of Software, Dalian University of Technology, China
Department of Epidemiology and Public Health, Yale University School of Medicine, New Haven, CT 06520, USA

1. Introduction

Proteomics is a relatively new but rapidly developing research area whose main objective is to conduct large-scale study of proteins expressed in an organism (Blackstock and Weir, 1999). Mass spectrometry (MS) is a widely used tool for proteomics research due to its sensitivity and to the versatility of the instrumentation (Vitek, 2009). A primary goal of MS-based proteomics is to identify all the proteins expressed in a cell or tissue. In MS-based proteomics, shotgun proteomics has become popular due to its high throughput. Shotgun proteomics identifies complex protein mixtures using a combination of high performance liquid chromatography (LC) and mass spectrometry (MS). A general view for protein identification in shotgun proteomics is illustrated in Fig. 1. Proteins in the sample are first digested into mixtures of peptides with an enzyme such as trypsin and then optionally eluted from their biological source. This digestion procedure is necessary since the sensitivity of the mass spectrometry is much higher for peptides than that for proteins (Vitek, 2009). The eluted peptides are subsequently ionized, fragmented and scanned by tandem mass spectrometry (MS/MS) to produce a set of MS/MS spectra. Finally, peptides and proteins are identified by computational subsequent analysis of the acquired MS/MS spectra (Fenyo, 2000; Chakravarti et al., 2002).

For computer analysis, two major steps are required: firstly, each MS/MS spectrum is analyzed to determine the underlying amino acid sequence of a peptide; secondly, the identified peptides are grouped together to identify the proteins. For the peptide identification step, typical approaches involve database search approach (Eng et al., 1994) and de novo sequencing (Dancik et al., 1999). Moreover, the database-based approach which compares each observed spectrum against entries in a database to find a best Peptide-Spectrum Match (PSM) is the most popular peptide identification method (Fig. 1). After peptide identification, it is still not straightforward to generate a reliable list of proteins. Fig. 2 shows that one can model the relationship between the identified peptides and their candidate proteins in the database as a bipartite graph. Each vertex $T_i$ represents an identified peptide. The vertex $R_j$ is a candidate protein in the database that matches at least one identified peptide $T_i$. This is the standard input setting for most protein inference algorithms. The general problem of inferring correct proteins from this bipartite graph is difficult to solve owing to the existence of degenerate peptides and “one-hit wonders”.

Degenerate peptides, also called shared peptides, denote the identified peptides shared by multiple proteins in the database. Such cases often result from the existence of homologous proteins in the database and make it difficult to distinguish between two possibilities: (1) all of the related proteins are present in the
sample; (2) only some proteins are truly present. For example, \( R_1 \) and \( R_2 \) in Fig. 2 have the same set of identified peptides: \( T_1 \) and \( T_2 \), which are degenerate peptides. If there is no other supporting information like the peptide \( T_3 \), we cannot determine which protein is indeed present in the sample.

“One-hit wonders” refer to those proteins which have only one single identified peptide. Theoretically, it is difficult to infer proteins based on one single peptide identification owing to the possibility of false positive identification. For example, the protein \( R_3 \) in Fig. 2 is a one-hit wonder that has only one matching peptide \( T_4 \). Compared with \( R_4 \) which has two identified peptides, it is not easy to determine the existence of \( R_3 \).

During the past decade, researchers have developed many protein inference algorithms to tackle above challenges (Tabb et al., 2002; Yang et al., 2004; Zhang et al., 2007; Ma et al., 2009; Slotta et al., 2010; Alves et al., 2007; Nesvizhskii et al., 2003; Price et al., 2007; Feng et al., 2007; Weatherly et al., 2005; Li et al., 2009b, 2010, 2009a; Serang et al., 2010; Moore et al., 2002; Sadygov et al., 2004; Bern and Goldberg, 2008; Shen et al., 2008; Spivak et al., 2012; Searle, 2010; He et al., 2011; Lu et al., 2008; Kearney et al., 2008; Ramakrishnan et al., 2009a,b; Grobei et al., 2009; Qeli and Ahrens, 2010; Gerster et al., 2010). The detailed description and a summary on these methods can be found in some recent review papers (Huang et al., 2012; Li and Radivojac, 2012). Briefly, all these efforts aim at solving the peptide degeneracy problem and one-hit wonder problem so as to achieve better identification accuracy. Though they have the same objective, different algorithms model the protein inference problem in quite different ways. On one hand, those methods such as ProteinProphet (Nesvizhskii et al., 2003) and IDPicker (Zhang et al., 2007; Ma et al., 2009) use standard bipartite graph as input with emphasis on developing rigorous models and algorithms under different assumptions. However, it is still very difficult to distinguish proteins that share exactly the same set of peptides if there is no other supporting information. On the other hand, researchers begin to realize that the employment of some supplementary information such as gene model (Gerster et al., 2010), protein–protein interaction network (Li et al., 2009a) and peptide detectability (Alves et al., 2007) is more promising for resolving the peptide degeneracy problem and one-hit wonder issue completely. Among different types of supplementary information, peptide detectability is an intrinsic property of the peptide that can well explain the assignment of degenerate peptides in principle. Therefore, the objective of this paper is to develop effective protein inference algorithms founded on the concept of peptide detectability.

Peptide detectability is defined as the probability of detecting a peptide in a standard sample by a standard proteomics routine if its parent protein is expressed (Tang et al., 2006). Peptide detectability is an intrinsic property of the peptide that is mainly decided by its primary sequence and its parent protein sequence. Peptide detectability indicates that not only identified peptides but also those that are missed (not identified) are important for protein inference.

Based on the concept of peptide detectability, Li et al. (2009b) proposed the MSBayesPro algorithm that places particular emphasis on the peptide degeneracy problem. Since MSBayesPro models the protein inference problem as a combinatorial optimization problem, it is almost infeasible to obtain the optimal protein

---

**Fig. 1.** A general pipeline for protein identification in shotgun proteomics.

**Fig. 2.** The bipartite graph represents the input of a standard protein inference problem. We take only the best spectrum match for each peptide.
probabilities within a reasonable time slot. That means there is no guarantee on finding the best protein inference results under its optimization model. More importantly, MSBayesPro is time-consuming when the datasets are relatively large.

In this paper, we formulate the protein inference problem as a constrained Lasso (Tibshirani, 1996) regression problem and then solve it with a fast pathwise coordinate descent algorithm (Friedman et al., 2007, 2010). This new protein inference algorithm is named as ProteinLasso. Lasso is a shrinkage least squares method in statistical learning, which introduces an additional $L_1$ norm penalty term to the least squares objective function so as to achieve sparsity.

More precisely, we first show that the probability of each identified peptide can be expressed as the linear combination of protein probabilities, where the coefficients are the conditional peptide probabilities given proteins (i.e., peptide detectabilities). If we take the protein probabilities as unknown variables and assume that peptide probabilities and peptide detectabilities are known in advance, one very natural idea is to formulate the protein inference problem as a constrained least squares regression problem. Here we pose some additional constraints since the probability of each protein should fall into $[0, 1]$.

Next, we further introduce an $L_1$ norm penalty into the model to enforce sparsity since we are interested in finding a subset of proteins that are truly present in the sample. Such modification leads to a constrained Lasso regression problem, which can be solved very efficiently through a coordinate descent algorithm in a pathwise manner.

Although the pathwise coordinate descent algorithm is able to quickly solve the constrained Lasso problem for a set of different sparsity penalty parameters, we still need to choose a proper value from this candidate set for model selection. Instead of using Bayesian information criterion (BIC) or cross validation to choose a single parameter, we adopt an ensemble learning strategy in which one protein’s probability is calculated as the average of its probabilities from all models under different sparsity penalty parameters. Such ensemble learning idea can address the parameter estimation problem. Experimental results on three datasets show that our ProteinLasso algorithm outperforms those state-of-the-art protein inference algorithms in terms of both identification accuracy and running efficiency.

The rest of this paper is organized as follows. In Section 2, we describe our method in detail. Section 3 presents the experimental results and Section 4 concludes the paper.

2. Model and algorithm

This section formulates the protein inference problem as a constrained Lasso regression problem and then shows that it can be solved efficiently via the coordinate descent algorithm.

2.1. Model

Suppose the bipartite graph contains $n$ identified peptides ($T_1, T_2, \ldots, T_n$) and $m$ candidate proteins ($R_1, R_2, \ldots, R_m$) with at least one peptide hit. Let $Pr(T_i = 1)$ denote the probability that the peptide $T_i$ is present and $V = j$ denote the event that only the protein $R_j$ is present in the sample, then $Pr(V = j) = Pr(R_1 = 0, \ldots, R_j = 1, \ldots, R_m = 0)$. According to the sum rule of probability theory, we have:

$$Pr(T_i = 1) = \sum_{j=1}^{m} Pr(T_i = 1, V = j) = \sum_{j=1}^{m} Pr(T_i = 1 | V = j) Pr(V = j),$$

(1)

where $Pr(T_i = 1 | V = j)$ denotes the conditional probability that the peptide $T_i$ is present in the sample if given only the protein $R_j$.

The assumption behind Eq. (1) is that only one protein is present in the sample, i.e. $\sum_{j=1}^{m} Pr(V = j) = 1$. This assumption is not so strong as it seems since the input bipartite graph for protein inference can be decomposed into many smaller unconnected bipartite graphs and the majority of them contain only one protein. As shown in the experimental section, nearly 90% of these unconnected bipartite graphs have only one protein across all three datasets used in this paper. Based on this fact, we can approximate the protein probability $Pr(R_j = 1)$ with $Pr(V = j)$. Then, Eq. (1) becomes:

$$Pr(T_i = 1) = \sum_{j=1}^{m} Pr(T_i = 1 | V = j) Pr(R_j = 1).$$

(2)

In practice, we do not know the true probability values in Eq. (2) and have to obtain their estimations from the data. The estimated value $y_i$ for the peptide probability $Pr(T_i = 1)$ is provided by database search algorithms such as Mascot (Perkins et al., 1999) or post-processing tools such as PeptideProphet (Keller et al., 2002). The conditional probability $Pr(T_i = 1 | V = j)$ can be approximated with the peptide detectability value $d_{ij}$, which is an intrinsic property of the peptide and can be predicted using models built on training data (Tang et al., 2006). Then our task in protein inference is to find a good estimation $x_j$ for the protein probability $Pr(R_j = 1)$. A natural idea is to formulate such inference problem as a least square regression problem with some additional constraints:

$$\min_X f(X) = \min_X \sum_{i=1}^{n} (y_i - \sum_{j=1}^{m} d_{ij} x_j)^2$$

subject to $0 \leq x_j \leq 1$ for all $1 \leq j \leq m$,

where the protein probability vector $X = (x_1, x_2, \ldots, x_m)$.

The least square formulation assumes that all proteins contribute to the peptide probability, i.e. the probabilities of most proteins should be non-zeros. However, only a small subset of candidate proteins is truly present in the sample. That means most protein probability values should be zeros. Therefore, we add an $L_1$ norm penalty to the least squares objective function so as to shrink some protein probabilities to 0. Such modification leads to a constrained Lasso (Tibshirani, 1996) regression problem:

$$\min_X \sum_{i=1}^{n} (y_i - \sum_{j=1}^{m} d_{ij} x_j)^2 + \lambda \sum_{j=1}^{m} x_j$$

subject to $0 \leq x_j \leq 1$ for all $1 \leq j \leq m$,

where $\lambda \geq 0$. Here we can use $\lambda$ to control the number of non-zero protein probabilities.

2.2. Algorithm

The constrained Lasso regression problem is a convex optimization problem, whose optimal solution can be found with standard solvers for convex optimization. In addition, there are also some specific algorithms that can solve the Lasso-type problems more efficiently such as LARS (Efron et al., 2004) and the coordinate descent algorithm (Friedman et al., 2007). In this paper, we utilize the pathwise coordinate descent algorithm proposed by Friedman et al. (2007) for its efficiency and simplicity. Note that our formulation is different from the standard Lasso regression model since we have some additional constraints on the range of variables. In this section, we show that we can still use the coordinate descent algorithm to solve such a constrained Lasso regression problem with only minor modifications.
Suppose currently we have estimates on $x_k$ for $k \neq j$ and we wish to partially optimize with respect to $x_j$, then the objective function in Eq. (4) can be re-written as:

$$f(X) = \sum_{i=1}^{n} \left( y_i - \sum_{k \neq j} d_{ik} \hat{x}_k - d_{ij} x_j \right)^2 + \lambda \sum_{k \neq j} |\hat{x}_k| + \lambda |x_j|,$$

(5)

where all the variables of $x_k$ for $k \neq j$ are held fixed at values $\hat{x}_k$. We can differentiate Eq. (5) to get

$$\frac{df}{dx_j} = \sum_{i=1}^{n} (-2)d_{ij} (y_i - \sum_{k \neq j} d_{ik} \hat{x}_k - d_{ij} x_j) + \lambda = 0.$$ 

(6)

Equivalently, the coordinate-wise update has the form

$$x_j = \left\{ \begin{array}{ll}
0 & x_j < 0 \\
1 & x_j > 1 \\
\hat{x}_j & \text{otherwise}.
\end{array} \right.$$ 

(7)

where $\hat{y}_j = \sum_{k \neq j} d_{kj} \hat{x}_k$. According to the constraint $0 < x_j \leq 1$, the updated value should be calculated as:

$$x_j = \frac{\sum_{i=1}^{n} d_{ij} r_i + \sum_{i=1}^{n} d_{ij}^2 \hat{x}_j}{K},$$

(8)

The update Eq. (8) is repeated for $j = 1, 2, \ldots, m$, $1, 2, \ldots$ until convergence. From Eqs. (7) and (8), we can easily understand why the Lasso regression approach can control the size of reported proteins with non-zero probabilities. Making $\lambda$ sufficiently large will cause some protein probabilities to be exactly zero.

Looking more closely at Eq. (7), we can see that

$$y_i - \hat{y}_j = y_i - \hat{y}_i + d_{ij} \hat{x}_j = r_i + d_{ij} \hat{x}_j,$$

(9)

where $\hat{y}_i$ is the calculated probability of the $i$th peptide based on the current estimated protein probabilities, and hence $r_i$ is the current residual between the observed and calculated peptide probability. Thus

$$\sum_{i=1}^{n} d_{ij} r_i = \sum_{i=1}^{n} d_{ij} (y_i - \hat{y}_j) - \sum_{i=1}^{n} d_{ij}^2 \hat{x}_j.$$ 

(10)

As shown in Friedman et al. (2010), further efficiencies can be achieved in computing the updates in Eq. (10) by writing the first term on the right as

$$\sum_{i=1}^{n} d_{ij} r_i = (d_{j}, y) - \sum_{k:|x_k|>0} (d_{j}, d_{k}) \hat{x}_k,$$

(11)

where $(d_{j}, y) = \sum_{i=1}^{n} d_{ij} y_i$. Substituting Eqs. (10) and (11) into Eq. (7) leads to

$$x_j = \frac{(d_{j}, y) - \sum_{k:|x_k|>0} (d_{j}, d_{k}) \hat{x}_k + \sum_{i=1}^{n} d_{ij}^2 \hat{x}_j - \frac{1}{2} \lambda}{\sum_{i=1}^{n} d_{ij}^2}.$$ 

(12)

Since it is impossible to know which $\lambda$ value generates the best solution in advance, the coordinate descent algorithm computes the solution at a grid of values of $\lambda$ (Friedman et al., 2010). It is initialized with the smallest value $\lambda_{\text{max}}$ for which the protein probability vector $X=0$, and then decreased gradually. Each time $\lambda$ is reduced a little and under this $\lambda$ value the coordinate descent algorithm cycles through the variables until convergence. The process is repeated using the previous solution as a “warm start”. Since the minimizers for many of the protein probabilities do not change through the update of $\lambda$ value, this pathwise coordinate descent procedure is very fast. We know from Eq. (12) that $x_j$ will be zero if $(d_{j}, y) \leq 0.5 \lambda$. Hence $\lambda_{\text{max}} = 2 \max_j (d_{j}, y)$. The algorithm then selects a minimum value $\lambda_{\text{min}} = \epsilon \lambda_{\text{max}}$, and constructs a sequence of $K$ values of $\lambda$ decreasing from $\lambda_{\text{max}}$ to $\lambda_{\text{min}}$ on the log scale. We choose $\epsilon = 0.001$ and $K = 100$ in our experiment.

To address the model selection problem, we use an ensemble learning strategy to avoid selecting a single $\lambda$ for protein inference. That is, we run the coordinate descent algorithm over the grid of values of $\lambda$ and combine their results to generate a final protein probability:

$$x_j = \frac{\sum_{k=1}^{K} x_{jk}}{K},$$

(13)

where $x_{jk}$ is the estimated probability of protein $j$ at value $\lambda_k$.

In fact, we have also tried to use BIC for selecting a single $\lambda$ for protein inference. However, our experimental results show that the use of BIC cannot provide very good identification results.

3. Experimental results

In order to test the performance of our algorithm, we have compared our method with ProteinProphet (Nesvizhskii et al., 2003) and MSBayesPro (Li et al., 2009b) on three datasets.

3.1. Datasets and experiment set-up

We use three datasets in our experiments. Each of these datasets has a reference set that contains the ground-truth proteins.

3.1.1. Mixture of 18 purified proteins

The first dataset is a mixture of 18 highly purified proteins from ISB Standard Protein Mix Database (Klimek et al., 2008). Among the 27 datasets provided in the website, we choose one dataset from mix2 acquired on the LTQ instrument. This is because it has more unique peptides and higher protein coverage. We use a database containing 1819 protein sequences for peptide identification. Both the data and database are available at http://regis-web.systemsbiology.net/PublicDatasets/.

3.1.2. Sigma49 dataset

Sigma49 is a mixture of 49 human proteins. The database used for peptide identification is composed of 15,682 protein sequences. The data and database are available at https://proteomecommons.org/dataset.jsp?i=71610.

3.1.3. Yeast dataset

This dataset has been used by Ramakrishnan et al. (2009a). The raw data is available at http://www.marcottelab.org/users/MSData/Data_02/. The reference set is generated by an intersection of 4 MS-based proteomics datasets and 3 non-MS-based datasets. It contains 4265 proteins observed in either two or more MS datasets or any of non-MS datasets and is available at http://www.marcottelab.org/MSData/gold_yeast.html. The database used in the experiment contains 6714 protein sequences, which is available at the same website as the raw data.
All tandem mass spectra are searched against the respective protein sequence database with X!Tandem (v2010.10.01.1) search engine. We use default search parameters wherever possible, assuming these parameters have already been optimized. Some important parameter specifications are listed in the following:

- Fragment monoisotopic mass error = 0.4 Da;
- Parent monoisotopic mass error = 100 ppm;
- Minimum peaks = 15;
- Minimum fragment m/z = 150.

The peptide probability is calculated with PeptideProphet embedded in Trans-Proteomic Pipeline (TPP) v4.5. The peptides with PeptideProphet probability >0.05 are considered as candidate peptides, and the proteins with at least one peptide hit are used as candidate proteins. We compare ProteinLasso with ProteinProphet and MSBayesPro. ProteinProphet is the most popular method for protein inference so far. We include MSBayesPro in the performance comparison since it also uses peptide detectability as input.

3.1.4. ProteinProphet

ProteinProphet puts proteins that cannot be distinguished with respect to identified peptides into the same group. In performance evaluation, we consider all proteins in each group and use the group probability as the probability of each protein in that group.

3.1.5. MSBayesPro

We run MSBayesPro according to the procedure below:

1. Obtain the peptide identifications and their probabilities from PeptideProphet files. If there are more than one spectrum match for a peptide, we choose the highest value as the peptide probability.
2. Obtain the predicted peptide detectabilities from http://darwin.informatics.indiana.edu/applications/PeptideDetectability Predictor/. This software currently only predicts scores for tryptic peptides. However, some identified peptides are nontryptic peptides and hence we have to assign detectability scores to these peptides by ourselves. The detectability value used for these peptides is: median (predicted detectability scores from the same parent protein)/3.
3. Run MSBayesPro for the first time to estimate the protein priors.
4. Run MSBayesPro for the second time to obtain the final protein probabilities with priors from the first run as input.

3.1.6. ProteinLasso

Similar to MSBayesPro, ProteinLasso also needs peptide detectability values as input. We adopt the same peptide detectability generation procedure used in MSBayesPro. We set $\epsilon = 0.001$ and $K = 100$ in our experiment. According to the peptide identification and peptide detectability files, we can calculate the $\lambda_{\text{max}}$ value. Then $\lambda_{\text{min}} = \epsilon \lambda_{\text{max}}$. We choose K values of $\lambda$ decreasing from $\lambda_{\text{max}}$ to $\lambda_{\text{min}}$ on the log scale. Based on each $\lambda$ value, ProteinLasso outputs a list of protein probabilities and then combines them to generate the final result.

3.2. Results

Since all the datasets in our experiment have their reference sets, we label a protein as a true positive if it is present in the corresponding reference set, and as a false positive otherwise. We use receiver operator characteristic (ROC) curve to compare different algorithms, which plots the true positive rate (TPR) as a function of the false positive rate (FPR). The FPR at a probability threshold $t$ is calculated as $\text{FPR}_t = F_1 / F_{\text{total}}$, where $F_1$ represents the number of false positives with probability $\geq t$ and $F_{\text{total}}$ is the total number of false positives. Likewise, the TPR at a threshold $t$ is the fraction of true positives that have probability $\geq t$. We also calculate the area under the ROC curve (AUC) to measure the overall performance of each algorithm. Fig. 3 describes the identification performance of different methods evaluated through ROC curve. The result shows that ProteinLasso outperforms ProteinProphet and MSBayesPro consistently on all three datasets. In particular, the performance of ProteinLasso is much better than that of ProteinProphet and MSBayesPro on the 18 mixtures dataset. One may argue that ProteinLasso only achieves minor performance improvement over ProteinProphet and MSBayesPro in terms of AUC value on another two datasets. Indeed, the AUC value measures the performance over a broad range of probability threshold. As a result, it is a good indicator for the overall performance but may not fully describes the true performance in a high-precision regime. In the evaluation of peptide/protein identification algorithms, researchers are more interested in reporting more true positives at a low false discovery rate (FDR) or q-value. Therefore, we also plot the number of true positives as a function of q-value in Fig. 4 to check the performance gap of different algorithms.\(^1\)

As shown in Fig. 4, ProteinLasso always outperforms ProteinProphet and MSBayesPro when the q-value is small. This indicates that our method can also achieve very good performance in a high precision regime. In particular, ProteinLasso allows for protein distinction on a fine level, whereas ProteinProphet and MSBayesPro often assign the maximal score of one to many proteins. As a result, ProteinLasso exhibits zero false positives among its H top-scoring proteins, where H is 14, 6 and 401 on 18 mixtures, Sigma49 and yeast datasets, respectively. In contrast, ProteinProphet and MSBayesPro cannot produce zero false positives since the probabilities of their top-scoring proteins are all equal to 1. The good performance of ProteinLasso should be attributed to its fine-grained ranking of the identified proteins.

As discussed in Section 2, the assumption behind our model is that only one protein is present in the sample. This assumption is not so strong as it seems since the input bipartite graph can be decomposed into many smaller connected components. For all three datasets used in our experiment, most of these components have only one protein, as shown in Fig. 5. It indicates that our assumption fits the real data well.

Degenerate peptide is one of the major difficulties for protein inference problem. A connected component would not contain any degenerate peptide if it has only one protein. According to this fact, we divide the proteins into two classes: "degenerate proteins" which belong to the component having more than one protein and "simple proteins" which belong to the component having only one protein. To compare the capability of different methods in tackling the peptide degeneracy issue, we present the identification results of three methods when inferring "degenerate proteins" in Table 1. For each dataset, we count the number of true positive and false positive proteins identified by ProteinProphet, MSBayesPro and ProteinLasso with the same number of reported proteins. Then, we count the number of "degenerate proteins" and "simple proteins" among these true positives and false positives, respectively. From Table 1, we have the following observations. On one hand, MSBayesPro can always report the least number of false positive degenerate proteins across all three datasets at the cost of identifying less true positive degenerate proteins. This means that

\[^1\text{Given a certain probability threshold } t, \text{ suppose there are } T_t \text{ true positives and } F_t \text{ false positives, the FDR is estimated as } \text{FDR}_t = F_t / (F_t + T_t). \text{ The corresponding q-value is the minimal FDR that a protein is reported: } q_t = \min_{t' \geq t} \text{FDR}_{t'}.\]
MSBayesPro is more powerful in controlling the false discovery rate with respect to degenerate proteins. On the other hand, ProteinProphet is able to identify more true positive degenerate proteins than MSBayesPro and ProteinLasso but it also reports more false positive degenerate proteins. In contrast to ProteinProphet and MSBayesPro, our method offers a reasonable trade-off between true positive and false positive rates.

ProteinLasso requires two parameters: $K$ and $\varepsilon$. We empirically choose the parameters as $\varepsilon = 0.001$ and $K = 100$. In fact, a different choice of parameters $K$ and $\varepsilon$ has very limited effect on the performance of ProteinLasso. We show this by performing ProteinLasso over two rough grids of $K$ and $\varepsilon$: $K$ ranges from 10 to 100 when $\varepsilon$ is fixed at 0.001 and $\varepsilon$ ranges from 0.001 to 0.01 when $K$ is fixed at 100. For each pair of parameters, we compute the corresponding AUC value on each dataset. Fig. 6 demonstrates that AUC values fluctuate slightly and thus ProteinLasso is robust to changes in parameters.

Table 2 presents the running time of three protein inference methods on three datasets. ProteinLasso runs much faster than the

Table 1
Accuracy on proteins containing degenerate peptides. For the three datasets, we count the number of true positives and false positives identified by ProteinProphet, MSBayesPro and ProteinLasso among their top-$k$ ranked proteins, where $k$ is 31, 43 and 538 for 18 mixtures, Sigma49 and yeast datasets, respectively. The value of $k$ is decided by the number of proteins with probability of 1.0 reported by ProteinProphet. We divide the identified proteins into two classes: “degenerate proteins” are proteins that belong to the components which have more than one proteins and “simple proteins” do not share a peptide with any other protein so that they belong to the components containing only one protein. Methods are abbreviated as PP = ProteinProphet, MSB = MSBayesPro, and PL = ProteinLasso.

<table>
<thead>
<tr>
<th></th>
<th>18 mixtures</th>
<th>Sigma49</th>
<th>Yeast</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PP</td>
<td>MSB</td>
<td>PL</td>
</tr>
<tr>
<td>Simple proteins</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td>17</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>FP</td>
<td>8</td>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>Degenerate proteins</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>TP</td>
<td>5</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>FP</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. 4. Identification performance comparison among ProteinProphet, MSBayesPro and ProteinLasso in a high-precision regime. In general, researchers are more interested in the performance of different algorithms when the q-value or FDR is very small, e.g., $\leq 0.05$. However, the probabilities of top-scoring proteins in ProteinProphet and MSBayesPro are all equal to one, we have to ship these proteins with same probabilities and then calculate the q-value at the first appearing protein with a different probability. This process causes that there are no points in the range $[0, 0.05]$ for the curves of ProteinProphet and MSBayesPro on 18 mixtures and Sigma49 datasets and hence we cannot set the q-value range to $[0, 0.05]$ for these two datasets.

Fig. 5. The distribution of components with respect to the number of proteins in each component. The y-axis stands for the percentage of components that contain a certain number of proteins. Most of the connected components have only one protein. In particular, there are no components containing more than three proteins in 18 mixtures and Sigma49 datasets.
it is robust to parameter specification so that we can use it with ease.

Acknowledgements

We thank Dr. Haixu Tang and Dr. Yong Fuga Li for their help on peptide detectability. We thank Dr. Smriti R. Ramakrishnan for providing us the protein reference set of yeast dataset used in the experiments. This work was partially supported by the Natural Science Foundation of China under Grant no. 61003176.

References


