Probabilistic Latent Semantic Analysis for prediction of Gene Ontology annotations

Marco Masseroli
Dipartimento di Elettronica e Informazione,
Politecnico di Milano,
Piazza L. Da Vinci 32,
20133 Milan, Italy
Email: masseroli@elet.polimi.it

Davide Chicco
Dipartimento di Elettronica e Informazione,
Politecnico di Milano,
Piazza L. Da Vinci 32,
20133 Milan, Italy
Email: davide.chicco@elet.polimi.it

Pietro Pinoli
Dipartimento di Elettronica e Informazione,
Politecnico di Milano,
Piazza L. Da Vinci 32,
20133 Milan, Italy
Email: pietro.pinoli@mail.polimi.it

Abstract—Consistency and completeness of biomolecular annotations is a keypoint of correct interpretation of biological experiments. Yet, the associations between genes (or proteins) and features correctly annotated are just some of all the existing ones. As time goes by, they increase in number and become more useful, but they remain incomplete and some of them incorrect. To support and quicken their time-consuming curation procedure and to improve consistency of available annotations, computational methods that are able to supply a ranked list of predicted annotations are hence extremely useful. Starting from a previous work on the automatic prediction of Gene Ontology (GO) annotations based on the Singular Value Decomposition of the annotation matrix, where every matrix element corresponds to the association of a gene with a feature, we propose the use of a modified Probabilistic Latent Semantic Analysis (pLSA) algorithm, named pLSAnorm, to better perform such prediction. pLSA is a statistical technique from the natural language processing field, which has not been used in bioinformatics annotation prediction yet; it takes advantage of the latent information contained in the analyzed data co-occurrences. We proved the effectiveness of the pLSAnorm prediction method by performing k-fold cross-validation of the GO annotations of two organisms, Gallus gallus and Bos taurus. Obtained results demonstrate the efficacy of our approach.

I. INTRODUCTION

Novel molecular biology high-throughput technologies, which allow studying thousands of genes or gene products simultaneously, are providing unprecedented amount of valuable data. Concurrently, advancements in information technologies and biomedical informatics are providing techniques and tools to manage the amount of biomolecular data produced, as well as several methods for their analysis. Furthermore, many terminologies and ontologies not only are being developed to consistently describe biomedical-molecular features, but are also increasingly used to annotate biomolecular entities. Availability of such controlled annotations, describing structural, functional and phenotypic features of mainly genes and their protein products, is extremely valuable and allows knowledge discovery computational analyses. Currently, several controlled vocabularies are routinely used to annotate genes and proteins. Some of them have a flat structure, i.e. no explicit relationships between the terms composing the vocabulary exist (e.g. the BioCyc [1] or Reactome [2] terminologies of gene and protein pathways, respectively). Other controlled vocabularies are part of ontologies, where semantic relationships are defined between pairs of vocabulary terms. The most widely used ontology for annotating biomolecular entities is the Gene Ontology (GO) [3]. It consists of three ontologies that contain a total of more than 35,000 controlled terms describing species-independent biological processes (BP), molecular functions (MF) and cellular components (CC). Each GO ontology is designed to capture orthogonal aspects of genes and gene products, and it is structured as a directed acyclic graph (DAG) of terms hierarchically related mainly through "is a" or "part of" relationships. An edge exists from a child term a to its parent term b if a "is a" specific instance of b or it is "part of" b. Furthermore, in each GO DAG it exists a unique root, which is defined as the DAG node without parents, and each term can have multiple parents.

Controlled biomolecular annotations constitute the biological knowledge that is being gathered over the years; they are collected in annotation databases that provide access to such valuable data. Despite their relevance, there are important issues that affect available biomolecular entity annotations [4]. First, they are not exhaustive: only a subset of genes and gene products of the sequenced organisms is known and, among those, only a small fraction has been annotated until now. Furthermore, annotation profiles of individual biomolecular entities might be incomplete, since biological knowledge about the functions associated with a gene or a gene product might be yet to be discovered, or the evidence already available in the literature might be not yet described with controlled terms and entered into an annotation database. Second, available annotations might be incorrect, e.g. those inferred from electronic annotations without the supervision of a human curator.

In this scenario, computational tools able to analyze annotation data in order to assess the relevance of inferred annotations, or produce a ranked list of new predicted annotations - e.g. to speed up the curation process - are paramount. Since many bioinformatics analyses currently performed on genomic and proteomic data rely on the available annotations of genes and gene products, the improvement of such annotations both in
quantity, coverage and quality is paramount to obtain better results in these analyses. King et al. [5] proposed the use of decision trees and Bayesian networks for predicting annotations by learning patterns from available annotation profiles. More recently, Tao et al. [6] proposed to use a k-nearest neighbour (k-NN) classifier, whereby a gene inherits the annotations that are common among its nearest neighbour genes, determined according to the functional distance between genes, based on the semantic similarity of GO terms used to annotate them. More simply, by using basic linear algebra tools, Khatri et al. [7] proposed a prediction algorithm based on the singular value decomposition (SVD) of the gene-to-term annotation matrix, which is implicitly based on the count of co-occurrences between pairs of terms in the available annotation dataset. Since this last method provides the reference for comparison of the work presented in this paper, it will be subsequently summarized in Section II. In a previous work, we extended the SVD method by developing a prediction method, called SIM, that incorporates gene clustering based on gene functional similarity computed on Gene Ontology annotations [8].

Other methods based on evaluation of co-occurrences exist, in particular those related to latent semantic indexing (LSI) [9], whose main application is in natural language processing, where it has been adopted for many aims: automated document classification [10], text summarization [11], understanding software source code [12], literature-based discovery [13], and many others. Among such methods, the probabilistic latent semantic analysis (pLSA) [14] provides a well defined distribution of sets of terms as an approximation of the co-occurrence matrix. It takes advantage of the latent model of a set of terms, to get more robust annotation prediction results. In the bioinformatics context, it has been previously applied in the classification of samples in gene expression microarray experiments [15], DNA sequence analysis for identification of dependencies between transcription factor binding sites [16] and inference of protein-protein interaction from literature [17].

Missing annotations can also be inferred by taking advantage of multiple heterogeneous data sources. In [18] expression levels obtained in microarray experiments are used to train a Support Vector Machine (SVM) classifier for each GO annotation term, and consistency among predicted annotation terms is enforced by means of a Bayesian network mapped onto the GO structure. Textual information is leveraged in [19] and [20], where the literature is mined and the keywords extracted from published papers are mapped to GO concepts. By providing a list, possibly ranked, of annotations to be checked by a curator, the aforementioned techniques can drive the discovery of previously unknown annotations, as well as the detection of inconsistencies in the existing annotations. Furthermore, the updated annotation profiles can help boosting the performance of data analysis methods that rely upon them. These include, for example, search for genes based on their similarity with a target annotation profile, or clustering genes based on their annotation profile [21] and [22].

In this paper we propose the use of the pLSA method for predicting annotations of GO terms based solely on previously available GO annotations. Although there are some sophisticated techniques that predict gene functions leveraging other genomic data, computational techniques based solely on the available annotations have been demonstrated to be useful. Our contribution consists in extending the method in [14], hereafter denoted pLSA method, by normalizing annotation prediction values of each single gene (or gene product). The proposed normalized method, denoted pLSAnorm method, avoids underestimation of annotation predictions for biomolecular entities with higher number of annotations, compared to less annotated biomolecular entities. We contrast the pLSAnorm method to the method in [7], hereafter denoted SVD method, by means of k-fold cross validation on the predicted GO gene annotations of two organisms, *Gallus gallus* (chicken) and *Bos taurus* (cattle), and we demonstrate improvements in their annotation predictions.

The rest of this paper is organized as follows. Section II illustrates the SVD method, since it represents the reference of comparison for our work. Section III describes the pLSA method and its normalized extension (pLSAnorm), which we propose. In Section IV, we present the validation method, which we developed to reliably evaluate the capability of the two compared methods (pLSAnorm vs. SVD) in accurately predicting new GO annotations, and we illustrate and discuss the obtained results. Section V concludes the paper and provides some guidelines for future research on this topic.

### II. SINGULAR VALUE DECOMPOSITION (SVD)

Let \( A_d \in \{0, 1\}^{m \times n} \) define the matrix representing all direct annotations of a specific GO ontology for a given organism. The \( m \) rows of \( A_d \) correspond to genes (or gene products), while the \( n \) columns correspond to GO terms. The entries of \( A_d \) assume values from the binary alphabet \( \{0, 1\} \) according to the following rule:

\[
A_d(i, j) = \begin{cases} 
1, & \text{if gene } i \text{ is annotated to term } j \\
0, & \text{otherwise}
\end{cases} \quad (1)
\]

Annotation curators are asked to always use the most specific GO term for a given functional category. Thus, when a gene (or gene product) is annotated to a term, it is implicitly assumed to be indirectly annotated also to the more generic terms for that category, i.e. all the term ancestors in the GO DAG. As such, let \( A \) denote a modified gene-to-term matrix, where the assignment of its entries is given by:

\[
A(i, j) = \begin{cases} 
1, & \text{if gene } i \text{ is annotated to term } j \\
& \text{or to any descendant of } j \\
0, & \text{otherwise}
\end{cases} \quad (2)
\]

The \( i \)-th row of the matrix \( A \), \( a_i^T \), contains all the direct and indirect annotations of gene \( i \). Conversely, the \( j \)-th column encodes the list of genes that have been annotated (directly or indirectly) to term \( j \). This process is sometimes defined as annotation unfolding.

According to the work in [7], annotation prediction can be performed by computing the SVD of the matrix \( A \), which is
given by:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$  \hspace{1cm} (3)

where $\mathbf{U}$ is a $m \times p$ unitary matrix (i.e. $\mathbf{U}^T\mathbf{U} = \mathbf{I}$), $\Sigma$ is a non-negative diagonal matrix of size $p \times p$, and $\mathbf{V}$ is a $n \times p$ unitary matrix, where $p = \min(m,n)$. Conventionally, the entries along the diagonal of $\Sigma$ (namely singular values) are sorted in non-increasing order. The number $r \leq p$ of non-zero singular values is equal to the rank of the matrix $\mathbf{A}$. For any positive integer $k < r$, it is possible to generate a matrix $\hat{\mathbf{A}}$, with:

$$\hat{\mathbf{A}} = \hat{\mathbf{U}}\hat{\Sigma}\hat{\mathbf{V}}^T$$  \hspace{1cm} (4)

where $\hat{\mathbf{U}}$ ($\hat{\mathbf{V}}$) is a $m \times k$ ($n \times k$) matrix obtained retaining the first $k$ columns of $\mathbf{U}$ ($\mathbf{V}$) and $\hat{\Sigma}$ is a $k \times k$ diagonal matrix with the $k$ largest singular values along the diagonal. $\hat{\mathbf{A}}$ is the optimal rank-$k$ approximation of $\mathbf{A}$, i.e. the one that minimizes the norm (either the spectral norm or the Frobenius norm) $\|\mathbf{A} - \hat{\mathbf{A}}\|$ subject to the rank constraint.

In [7] it is argued that the study of the matrix $\hat{\mathbf{A}}$ reveals semantic relationships of the gene-function associations. A large value of $\hat{\mathbf{A}}(i,j)$ suggests that gene $i$ should be annotated to term $j$, whereas a value close to zero suggests the opposite. As a matter of fact, the SVD of the matrix $\mathbf{A}$ is equivalent to the method of latent semantic indexing (LSI) in information retrieval, where the input is a matrix that contains the occurrences of words in indexed documents.

In order to better understand why $\hat{\mathbf{A}}$ can be used to predict gene-to-term annotations, we point out that an alternative expression of (4) can be obtained by basic linear algebra manipulations:

$$\hat{\mathbf{A}} = \mathbf{A}\hat{\mathbf{V}}\hat{\mathbf{V}}^T$$  \hspace{1cm} (5)

Moreover, the SVD of the matrix $\mathbf{A}$ is related to the eigen-decomposition of the symmetric matrices $\mathbf{T} = \mathbf{A}^T\mathbf{A}$ and $\mathbf{G} = \mathbf{A}\mathbf{A}^T$. In fact, the columns of $\mathbf{V}$ ($\hat{\mathbf{U}}$) are a set of $k$ eigenvectors corresponding to the $k$ largest eigenvalues of the matrix $\mathbf{T}$ ($\mathbf{G}$). The matrix $\mathbf{T}$ has a simple interpretation in our context. In fact,

$$\mathbf{T}(j_1,j_2) = \sum_{i=1}^{m} \mathbf{A}(i,j_1) \cdot \mathbf{A}(i,j_2)$$  \hspace{1cm} (6)

i.e. $\mathbf{T}(j_1,j_2)$ is the number of times that terms $j_1$ and $j_2$ are used to annotate the same gene in the existing annotation profile. Therefore, $\mathbf{T}(j_1,j_2)$ expresses the (unnormalized) correlation between term pairs and it can be interpreted as a similarity score of the terms $j_1$ and $j_2$ computed solely based on the use of these terms in available annotations. The eigenvectors of $\mathbf{T}$ (i.e. the columns of $\mathbf{V}$) can be considered as a reduced set of eigen-terms. Intuitively, if two terms co-occur frequently, they are likely to be mapped to the same eigen-term. Based on (5), the $i$-th row of $\hat{\mathbf{A}}$ can be written as

$$\hat{\mathbf{a}}_i^T = \mathbf{a}_i^T\mathbf{V}\mathbf{V}^T$$  \hspace{1cm} (7)

Thus, the original annotation profile is first transformed in the eigen-term domain, while retaining only the first $k$ eigen-terms with $\mathbf{V}$, and then mapped back to the original domain by means of $\hat{\mathbf{V}}^T$. This corresponds to projecting the original vector $\mathbf{a}_i^T$ onto the $k$-dimensional subspace spanned by the columns of $\hat{\mathbf{V}}$.

The entries of the matrix $\mathbf{A}$ are real valued. In [7] it is defined a threshold $\tau$ such that, if $\hat{\mathbf{A}}(i,j) > \tau$, then gene $i$ is predicted to be annotated to term $j$. Depending on the original values assumed by the matrix $\mathbf{A}$, the following cases might occur:

- If $\mathbf{A}(i,j) = 1$ and $\hat{\mathbf{A}}(i,j) > \tau$, the annotation of gene $i$ to term $j$ is confirmed; this case is denoted as an annotation confirmed (AC), with respect to the original $\mathbf{A}(i,j)$.
- If $\mathbf{A}(i,j) = 0$ and $\hat{\mathbf{A}}(i,j) > \tau$, a new annotation is suggested; this case is denoted as an annotation predicted (AP), with respect to the original $\mathbf{A}(i,j)$.
- If $\mathbf{A}(i,j) = 1$ and $\hat{\mathbf{A}}(i,j) \leq \tau$, an existing annotation is suggested to be semantically inconsistent with the available data; this case is denoted as an annotation to be reviewed (AR), with respect to the original $\mathbf{A}(i,j)$.
- If $\mathbf{A}(i,j) = 0$ and $\hat{\mathbf{A}}(i,j) \leq \tau$, the annotation is not present in the original annotation set and it is not suggested by the analysis; this case is denoted as a non existing annotation confirmed (NAC), with respect to the original $\mathbf{A}(i,j)$.

### A. SVD truncation level

Since in SVD the $\hat{\mathbf{A}}$ matrix is a $k$-rank approximation of the $\mathbf{A}$ matrix, a keypoint of the SVD algorithm is the truncation level $k$ used. To objectively define it, we implemented the following greedy algorithm; it selects the best SVD truncation level $k$ by sampling some possible truncation values and for each of them generating a Receiver Operating Characteristic (ROC) curve, which represents the AR rate ($1 - \text{Sensitivity}$) vs. AP rate ($1 - \text{Specificity}$), and calculating its area under the curve (AUC).

If $r - 1$ is the maximum rank of $\hat{\mathbf{A}}$, i.e. the minimum between the number of rows and the number of columns of $\mathbf{A}$, in the $(1; r - 1)$ interval the algorithm selects $q = 10$ equidistant values, where $1 < q_i < q_{i+1} < \ldots < r$. Then, it considers each $q_i$ value as a truncation value for SVD, generates the corresponding $\text{ROC}_{q_i}$ curve and calculates its $\text{AUC}_{q_i}$. When all the $\text{AUC}_{q_i}$ values are computed, the algorithm takes the $q_i$ with the minimum $\text{AUC}_{q_i}$ of the related $\text{ROC}_{q_i}$ curve as the best truncation level $k$.

### III. Probabilistic Latent Semantic Analysis (PLSA)

An alternative to the SVD method, is the pLSA method, a statistical technique based on Latent Semantic Indexing (LSI) that is also known as Probabilistic Latent Semantic Analysis. LSI is an information retrieval technique whose aim is to identify latent relationships between different elements in a certain class, e.g. between documents and words within them, or between genes and their biomolecular features described by controlled annotation terms. The keypoint of LSI is to map class elements (documents and words, in the text example; genes and biomolecular feature annotation terms, in our case) to a vector space of reduced dimensionality, and then analyse it. In this
section we describe the pLSA algorithm and its implementation for biomolecular annotation prediction, i.e., to calculate the joint probability \( P(g, f) \) of the annotation (association) of a gene \( g \) to a feature term \( f \).

A. pLSA model

The core of the pLSA algorithm is the statistical model used. Suppose to have a set of genes \( G = \{g_1, ..., g_n\} \) related to a set of feature terms \( F = \{f_1, ..., f_n\} \), which, together, form a set of biomolecular annotations (i.e., associations between a gene \( g \) and a feature term \( f \)). If you consider a set of class variables \( T = \{t_1, ..., t_l\} \), called topics, every feature term \( f \in F \) could be associated with a topic \( t \in T \). The pLSA statistical model associates every unobserved class variable (topic) with each observation (feature term and gene).

In the probabilistic Latent Semantic Analysis, a multinomial distribution \( \phi(t, f) = P(f|t) \) describes the probability of a feature term \( f \) to be associated with a topic \( t \). Another multinomial distribution \( \theta(g, t) = P(t|g) \) indicates the probability of getting a topic \( t \) by selecting a gene \( g \). Furthermore, the following constraints must be always valid:

\[
\forall t \in T, \sum_{f \in F} \phi(t, f) = \sum_{f \in F} P(f|t) = 1 \tag{8}
\]

\[
\forall g \in G, \sum_{t \in T} \theta(g, t) = \sum_{t \in T} P(t|g) = 1 \tag{9}
\]

In addition, differently from the classical pLSA method, we assume that each gene has the same probability to be considered:

\[
\forall g \in G, P(g) = \frac{1}{|G|} \tag{10}
\]

Thus, a joint probability between \( g \) and \( f \), which is modeled by the latent variable \( t \), can be defined as follows:

\[
P(g, f) = \sum_{t \in T} P(t) P(g|t) P(f|t) \tag{11}
\]

which, for the Bayes’ theorem, can be written also as:

\[
P(g, f) = P(g) \sum_{t \in T} P(t|g) P(f|t) \tag{12}
\]

\( P(f|t) \) represents the topic-conditioned probability of a feature term \( f \) to be related to a certain topic \( t \), whereas \( P(t|g) \) represents the weight of the latent variable \( t \) for the gene \( g \).

B. Model training

Since the above introduced model has \( \phi(t, f) \) and \( \theta(g, t) \) as parameters, we compute its maximum likelihood estimation using the Expectation Maximization (EM) algorithm [23]. During the model training phase, we vary the topic probabilities \( P(t) \) and the topic conditioned probabilities \( P(g|t) \) and \( P(f|t) \) in order to maximise the following log-likelihood function \( L \) on a gene and feature term training set:

\[
L = \sum_{g \in G} \sum_{f \in F} a(g, f) \log P(g, f) \tag{13}
\]

where \( P(g, f) \) is as defined in (11) and \( a(g, f) \) indicates the presence of an association (annotation) between gene \( g \) and feature term \( f \):

\[
a(g, f) = \begin{cases} 
1, & \text{if gene } g \text{ is annotated to term } f \\
0, & \text{otherwise}
\end{cases} \tag{14}
\]

First, we randomly initialize \( P(t), P(g|t) \) and \( P(f|t) \) in the range \([0;1] \in \mathbb{R} \). Then, we apply the EM procedure, which iterates between the E-step and the M-step. In the E-step, the probability that a gene \( g \) annotated to a feature term \( f \) is included in the topic-class \( t \) is calculated as follows:

\[
P(t|g, f) = \frac{P(t) P(g|t) P(f|t)}{\sum_{t' \in T} P(t') P(g|t') P(f|t')} \tag{15}
\]

In the M-step, by using the values from (15), the general probability of a topic \( t \) is estimated as:

\[
P(t) = \frac{\sum_{g \in G} \sum_{f \in F} a(g, f) P(t|g, f)}{\sum_{g \in G} \sum_{f \in F} a(g, f)} \tag{16}
\]

while the probabilities of a feature term \( f \) and a gene \( g \) given the topic \( t \) are estimated as:

\[
P(g|t) = \frac{\sum_{f \in F} a(g, f) P(t|g, f)}{\sum_{g \in G} \sum_{f \in F} a(g, f, t') P(t|g, f')} \tag{17}
\]

\[
P(f|t) = \frac{\sum_{g \in G} a(g, f) P(t|g, f)}{\sum_{f' \in F} \sum_{g \in G} a(g, f') P(t|g, f')} \tag{18}
\]

The values obtained in (16), (18) and (17) are then iteratively used in the next E-step, till obtaining the \( P(t|g) \) and \( P(f|t) \) that maximise the log-likelihood in (13).

C. Model validation

The validation phase works on a gene and feature term validation set with the same feature terms, but completely different genes, respect to the ones in the training set. The aim of the validation phase is still to maximise the formula in (13), but by using the \( P(f|t) \) calculated in the training phase and varying the parameters \( P(t|g) \) related to the new genes in the validation set. Thus, in this case, \( P(g, f) \) is considered as defined in (12), taking into account the constrain in (10). Furthermore, during the EM procedure of the validation phase, in the E-step instead of the formula in (15) used during the training phase, for the Bayes’ theorem \( P(t|g, f) \) is computed as:

\[
P(t|g, f) = \frac{P(t|g) P(f|t)}{\sum_{t' \in T} P(t'|g) P(f|t')} \tag{19}
\]

In (19), \( P(f|t) \) is as estimated in (18) during the training phase and \( P(t|g) \) initially is randomly initialized in the range \([0;1] \in \mathbb{R} \). Then, the values of \( P(t|g, f) \) from the formula in (19) are used in the M-step to estimate the \( P(t|g) \) as follows:

\[
P(t|g) = \frac{\sum_{f \in F} a(g, f) P(t|g, f)}{|G|} \tag{20}
\]

The values from (20) are then iteratively used in the next E-step, till obtaining the \( P(t|g) \) that maximise the log-likelihood in
we compute its maximum $K$-fold cross validation, as suggested in [5] [6], according to (10), are used to calculate the join probabilities $P(g, f)$, which we are looking for, as in (12).

Using the obtained join probabilities $P(g, f)$, we build the output matrix $\mathbf{A}$, where $A_{ij} = P(g_i, f_j)$. This matrix, which represents the predicted annotations, is equivalent to the output matrix $\hat{\mathbf{A}}$ of the SVD method. In fact, using a parametrized matrix notation, we can write:

$$
\hat{\mathbf{U}} = [P(g_i|t_k)]_{ik},\\
\Sigma = \text{diag}[P(t_k)]_k,\\
\hat{\mathbf{V}} = [P(f_j|t_k)]_{jk},\\
\hat{\mathbf{A}} = [P(g_i, f_j)]_{ij} = \hat{\mathbf{U}} \Sigma \hat{\mathbf{V}}^T.
$$

The crucial difference between SVD and pLSA is the objective function. In SVD, it is the spectral norm induced by the L-2 norm, or the Frobenius norm, which corresponds to an implicit additive Gaussian noise assumption on the possible gene-to-term annotations. In pLSA, it relies on the log-likelihood function of the multinomial sampling and aims at an explicit maximization of the predictive power of the model.

An open question in pLSA is how to choose the best number of topics. After some tests, we heuristically fixed the best number of topics, in our cases, as 30%-40% of the number of genes involved in a pLSA computation.

D. pLSA normalization

The gene feature term conditioned probabilities $P(f|g)$ can be expressed as:

$$
P(f|g) = \sum_{t \in T} P(f|t)P(t|g) \quad (21)
$$

Since the constraints in (8) and (9), in pLSA for every gene the sum of all its feature term conditioned probabilities $P(f|g)$ results to be equal to 1:

$$
\forall g \in G, \sum_{f \in F} P(f|g) = 1 \quad (22)
$$

Therefore, genes with many annotations have average feature term probability values lower than genes with fewer annotations; this can bias the annotation predictions. To avoid such bias, we propose a normalization extension of the standard pLSA algorithm, which we called pLSAnorm. For each gene, we compute its maximum $P(f|g)$ feature term conditioned probability value ($\text{max}$); then we normalize all the feature term conditioned probabilities of the gene by multiplying each of them by $1/\text{max}$. Thus, the feature terms with the highest conditional probability for a gene always result predicted to be annotated to that gene.

IV. VALIDATION

A. Method

To assess reliability and performance of SVD and pLSA algorithms while varying the value of the threshold $\tau$, we used $K$-fold cross validation, as suggested in [5] [6], according to the following procedure.

1) First, we eliminate non-influential terms by discarding all the feature terms that are annotated to less than $M \in \mathbb{N}$ genes, and we consider only those genes that have at least $L \in \mathbb{N}$ annotations. Then, to the $\hat{m}$ rows of the obtained reduced $\hat{m} \times \tilde{n}$ matrix $\hat{\mathbf{A}}$, we apply a random permutation, in order to eliminate any form of correlation between genes mapped to adjacent rows, and divide them into $K = 10$ non-overlapping random subsets. In each of the $K$ steps of the $K$-fold cross validation process, one of these $K$ datasets is then used as validation set, while the other $K-1$ datasets are used as training set.

2) The predictive method (SVD or pLSAnorm) is applied to the training set in order to generate the predictive model.

3) For what concerns each $k$-th subset, the validation set:

   a) To every of its rows $i$, the generated predictive model is applied to predict the value of all elements $i,j$ of the row $i$ at once. Yet, of these predictions, only those for the elements $i,j$ that do not correspond to existing annotations (i.e. with $A(i, j) = 0$) are considered.

   b) For each of the elements that corresponds to existing annotations (i.e. with $A(i, j) = 1$), the prediction is build as follow:

      i) Discard the annotation in position $(i, j)$ by setting to zero the corresponding entry in the validation set.

      ii) Discard all the annotations corresponding to the descendants or ancestors of term $j$, by setting to zero the corresponding entries in the validation set.

      iii) For each annotation left of gene $i$ to a term $l \neq j$, restore part of the discarded annotations by setting to one the entries corresponding to the ancestors of $l$.

      iv) After such modifications described in the previous three steps, if row $i$ contains at least one annotation (i.e. at least one element of row $i$ is equal to 1), the predictive model is applied on the row $i$.

      v) The predicted value for element $ij$ is considered as final value for such element and replaces the element predicted value calculated at step (3.a).

   c) Operations (a) and (b) are repeated other $K-1$ times, by using every time a different validation set and the other $K-1$ sets as training set.

4) The final matrix $\hat{\mathbf{A}}$ of predictions is created as union of all the predictions made for each $K$ training set. After the final prediction matrix $\hat{\mathbf{A}}$ is build, each of its elements $\hat{\mathbf{A}}(i, j)$ is compared to the corresponding input matrix element $A(i, j)$, while varying the value of the threshold $\tau$, in the same way as described for the SVD method in Section II.

B. Results

We implemented the prediction methods and their validation procedure in a software framework developed by using Java
programming language. The mathematical core of the software was developed in C++ programming language using several multiplatform and multithreading optimized mathematical libraries, including: AMD Core Math Library (ACML) [24], Boost [25] and SvdLibC [26]. It was integrated in the Java framework by using the Java Native Interface (JNI) [27].

Evaluation of the prediction methods was performed using the Biological process (BP), Cellular component (CC) and Molecular function (MF) Gene Ontology (GO) annotations of Gallus gallus and Bos taurus genes available in July 2009 from the GO and Entrez Gene databases (http://www.geneontology.org/ and ftp://ftp.ncbi.nih.gov/entrez/respectively). Quantitative characteristics of such annotations are illustrated in Table I. Table II, Figure 1 and Figure 2 report the results obtained, by varying the threshold \( \tau \), with the SVD and pLSAnorm methods applied to the considered annotations. The ROC curves in Figure 1 and Figure 2 depict the trade-off between the \( ARrate = AR/(AC + AR) \) and \( APrate = AP/(AP + NAC) \). Notice that, in statistical terms, \( ARrate = 1 - Sensitivity \) and \( APrate = 1 - Specificity \).

The prediction was performed by using, for SVD, the best k truncation level (calculated as described in Section II-A), and, respectively). Quantitative characteristics of such annotations are illustrated in Table I. Table II, Figure 1 and Figure 2 report the results obtained, by varying the threshold \( \tau \), with the SVD and pLSAnorm methods applied to the considered annotations. The ROC curves in Figure 1 and Figure 2 depict the trade-off between the \( ARrate = AR/(AC + AR) \) and \( APrate = AP/(AP + NAC) \). Notice that, in statistical terms, \( ARrate = 1 - Sensitivity \) and \( APrate = 1 - Specificity \).

The prediction was performed by using, for SVD, the best k truncation level (calculated as described in Section II-A), and,
for pLSAnorm, the best number of topics, heuristically fixed as 40% of the number of genes involved in the prediction. Furthermore, in the matrix $A$ we heuristically retained GO terms used to annotate at least $M = 3$ genes of the considered organism and excluded annotations with evidence code IEA (inferred electronic annotations) or ND (no data available) and genes with less than $L = 1$ annotations. As an aggregated indicator of the prediction performance, we computed the area under the $ARate$ vs. $APrate$ curve in the [0,0.01] range. In fact, we are typically interested in the low range of $APrate$, since it corresponds to top ranked predictions of newly inferred annotations (AP) with the highest score. Results are shown in Table II. In every case, pLSA outperformed SVD: it always shown a lower AUC. Nevertheless, the execution times were much higher for pLSAnorm than SVD, since the complex and multi-step procedure of the former one.

We also made two quantitative evaluations of the annotations predicted (AP), obtained by using a threshold $\tau = 0.5$, by comparing them to: 1) the computationally inferred annotations (IEA) available among the same genes and ontology terms, which we excluded from the annotations given as input to the prediction algorithms; 2) the annotations among the same genes and ontology terms that became available in October 2011, i.e. 2 years and 3 months after the annotations considered as input for the prediction. Amount and differences between the former ones and the new available annotations are shown in Table I. Notably, the GO annotations of both Gallus gallus and Bos taurus genes increased, from +66.21% to +206.19%. Table III shows the results of such quantitative evaluations. Our pLSAnorm method almost always outperformed the SVD method, with higher percentages of AP confirmed by the July 2009 IEA annotations not used as input for the prediction and by the not ND annotations become available in October 2011.

V. CONCLUSIONS

In this paper we propose to use our normalized extension of the pLSA method as an alternative to the SVD method to predict gene (or gene product) annotations based on existing annotations. Experimental results on the GO annotations of the genes of Bos taurus and Gallus gallus organisms confirm the effectiveness of our proposed approach. Future work will address advantages and issues related to the annotation prediction by considering all GO ontologies jointly instead of independently, in order to take advantage of potential correlations existing between them. Furthermore, since our approach is not bounded to the GO, but can be applied to any ontological annotation, increasingly available multiple annotations of genes and gene products from different ontologies could be jointly considered to further improving prediction reliability. So far, results have been validated using objective metrics computed by means of cross-validation or by comparison with existing annotations (either computationally inferred or become available after several months) among the same genes and ontology terms. We plan to further verify the effectiveness of the proposed method by assessing the quality of the top ranked predictions by means of an expert evaluation.

ACKNOWLEDGMENT

The work of this paper is partially funded by the Entity Aware Search Engines (EASE) project. The authors would like to thank Marco Tagliasacchi for the advices given about the pLSA method, and the users’ community of StackExchange.com Q&A website for the free help in various problem situations.

REFERENCES


