Deep Autoencoder Neural Networks for Gene Ontology Annotation Predictions

Davide Chicco*
Politecnico di Milano
Dipartimento di Elettronica
Informazione Bioingegneria
Milan, Italy
davide.chicco@gmail.com

Peter Sadowski
University of California, Irvine
Dept. of Computer Science
Institute for Genomics and Bioinformatics
Irvine, CA, USA
peter.j.sadowski@uci.edu

Pierre Baldi†
University of California, Irvine
Dept. of Computer Science
Institute for Genomics and Bioinformatics
Irvine, CA, USA
pfbaldi@ics.uci.edu

ABSTRACT

The annotation of genomic information is a major challenge in biology and bioinformatics. Existing databases of known gene functions are incomplete and prone to errors, and the bimolecular experiments needed to improve these databases are slow and costly. While computational methods are not a substitute for experimental verification, they can help in two ways: algorithms can aid in the curation of gene annotations by automatically suggesting inaccuracies, and they can predict previously-unidentified gene functions, accelerating the rate of gene function discovery. In this work, we develop an algorithm that achieves both goals using deep autoencoder neural networks. With experiments on gene annotation data from the Gene Ontology project, we show that deep autoencoder networks achieve better performance than other standard machine learning methods, including the popular truncated singular value decomposition.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; J.3 [**Life and Medical Sciences**]: Biology and Genetics; H.2.8 [**Database Applications**]: Data mining

Keywords

biomolecular annotations, matrix-completion, autoencoders, neural networks, Gene Ontology, truncated singular value decomposition, principal component analysis

1. INTRODUCTION

In bioinformatics, a *controlled gene function annotation* is a binary matrix associating genes or gene products with functional features from a controlled vocabulary. These annotations are important for effective communication in biomedical research, and lay the groundwork for bioinformatics software tools and data mining investigations. The *in vitro* biomolecular experiments used to validate gene functions are expensive, so the development of computational methods to identify errors and prioritize new biomolecular experiments is a worthwhile area of research [1].

The Gene Ontology project (GO) is a bioinformatics initiative to characterize all the important features of genes and gene products within a cell [2] [3]. GO is composed of three controlled vocabularies structured as mostly-separate sub-ontologies: biological processes, cellular components, and molecular functions. Each GO sub-ontology is structured as a directed acyclic graph of features (nodes) and ontological relationships (edges). In January 2014, GO contained 39,000 terms with more than 25,450 biological processes, 2,250 cellular components, and 9,650 molecular functions. However, GO annotations are constantly being revised and added as new experimental evidence is produced.

One approach to improving gene function annotation data bases like GO is to use patterns in the known annotations to predict new annotations. This can be viewed as a *matrix-completion* problem, in which one attempts to recover a matrix with some underlying structure from noisy observations. Machine learning algorithms have proved very successful in similar applications, such as the famous million-dollar Netflix prize awarded in 2009. Many machine learning algorithms have already been applied to gene function annotation ([4] [5] [6] [7] [8] [9]), but to the best of our knowledge deep autoencoder neural networks have not. *Deep* networks of multiple hidden layers have an advantage over shallow machine learning methods in that they are able to model complex data with greater efficiency. They have proven their usefulness in fields such as vision and speech recognition, and promise to yield similar performance gains in other machine learning applications that have complex underlying structure in the data.

A popular algorithm for matrix-completion is the truncated singular value decomposition method (tSVD). Khatri et al. first used this method for GO annotation prediction [10], and one of the authors of this work has extended their method with gene clustering and term-term similarity weights [11] [12]. However, the tSVD method can be viewed as a special linear case of a more general approach using autoencoders [13] [14] [15]. Deep, non-linear, autoencoder

*corresponding author
†corresponding author

BCB'14, September 20–23, 2014, Newport Beach, CA, USA.

ACM 978-1-4503-2894-4/14/09 ...$15.00.
http://dx.doi.org/10.1145/2649387.2649442.

neural networks have more expressive power, and may be better suited for discovering the underlying patterns in gene function annotation data.

In this paper, we summarize the tSVD and autoencoder methods, show how they can be used to predict annotations, and compare the performance on six separate GO datasets.

2. SYSTEM AND METHODS

In this section we describe the two annotation-prediction algorithms used in this paper: *Truncated Singular Value Decomposition* and *Autoencoder Neural Network*.

2.1 Truncated Singular Value Decomposition

Truncated Singular Value Decomposition (tSVD) [16] is a matrix factorization method that produces a low-rank approximation to a matrix. Define $A_d \in \{0,1\}^{m \times n}$ to be a matrix of annotations. The m rows of A_d correspond to genes, while the n columns correspond to GO features, such that

$$A_d(i,j) = \begin{cases} 1 & \text{if gene } i \text{ is annotated with feature } j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

When features are organized into ontologies, sometimes only the most specific feature is specified, and the more general features (ancestors) are implicit. Thus, in this work we consider a modified matrix A defined as

$$A(i,j) = \begin{cases} 1 & \text{if gene } i \text{ is annotated with feature } j \\ & \text{or with any descendant of } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The i_{th} row of the A matrix (a_i^T) contains all the direct and indirect annotations of gene i. The j_{th} column encodes the list of genes that have been annotated (directly or indirectly) to feature j. This process is sometimes defined as *annotation unfolding* [17].

Predictions are produced by computing the SVD of the matrix A and truncating the less-significant singular values. The SVD of the matrix A is given by

$$A = U\,\Sigma\,V^T \tag{3}$$

where U is a $m \times m$ unitary matrix (i.e. $U^T\,U = I$), Σ is a non-negative diagonal matrix of size $m \times n$, and V^T is a $n \times n$ unitary matrix (i.e. $V^T\,V = I$). Conventionally, the entries along the diagonal of Σ (the 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 A, where $p = \min(m,n)$. For a positive integer $k < r$, the tSVD matrix $\widetilde{A}$ is given by

$$\widetilde{A} = U_k\,\Sigma_k\,V_k^T \tag{4}$$

where U_k (V_k^T) is a $m \times k$ ($n \times k$) matrix achieved by retaining the first k columns of U (V) and Σ is a $k \times k$ diagonal matrix with the k largest singular values along the diagonal. The decomposition of the matrices and the difference between SVD and tSVD are represented in Fig. 1. The matrix $\widetilde{A}$ is the optimal rank-k approximation of A, i.e. the one that minimizes the norm (either the spectral norm or the Frobenius norm) $\|A - \widetilde{A}\|$ subject to the rank constraint.

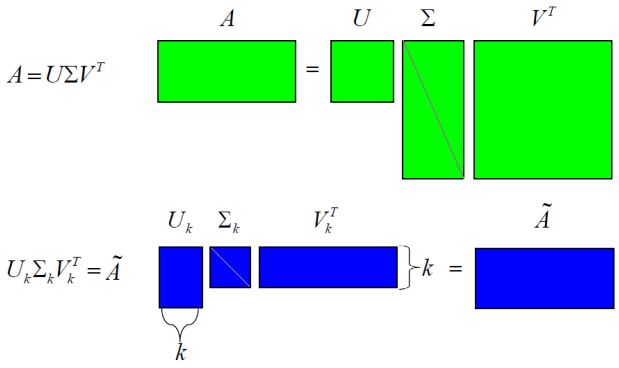


Figure 1: An illustration of the Singular Value Decomposition (upper green image) and the Truncated SVD reconstruction (lower blue image) of the A matrix. In the classical SVD decompostion, $A \in \{0,1\}^{m \times n}$, $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times n}$ $V^T \in \mathbb{R}^{n \times n}$. In the Truncated decomposition, where $k \in \mathbb{N}$ is the truncation level, $U_k \in \mathbb{R}^{m \times k}$, $\Sigma_k \in \mathbb{R}^{k \times k}$, $V_k^T \in \mathbb{R}^{k \times n}$, and the output matrix $\widetilde{A} \in \mathbb{R}^{m \times n}$

The matrix $\widetilde{A}$ is real valued and can be interpreted as a model of the noisy, incomplete observations. It can be used to predict both inaccuracies and missing gene functions — a large value of $\widetilde{a}_{ij}$ suggests that gene i should be annotated with term j, whereas a value close to zero suggests the opposite. The choice of the k truncation parameter controls the complexity of the model, and affects the predictions. Khatri et al. use a fixed value of $k = 500$ in [10] [18] [19], while one of the authors of this paper has developed a new discrete optimization algorithm to select the best truncation level on the basis of the ROC AUCs, described in [20].

In order to better comprehend why $\widetilde{A}$ can be used to predict gene-to-term annotations, we highlight that an alternative expression of Equation (4) can be obtained using basic linear algebra manipulations:

$$\widetilde{A} = A\,V_k\,V_k^T \tag{5}$$

Additionally, the SVD of the matrix A is related to the eigen-decomposition of the symmetric matrices $T = A^T A$ and $G = AA^T$. The columns of V_k (U_k) are a set of k eigenvectors corresponding to the k largest eigenvalues of the matrix T (G). The matrix T has a simple interpretation in our context. In fact,

$$T(j_1, j_2) = \sum_{i=1}^{m} A_{(i,j_1)} \cdot A_{(i,j_2)} \tag{6}$$

i.e. $T(j_1, j_2)$ is the number of genes annotated with both terms, j_1 and j_2. Consequently, $T(j_1, j_2)$ indicates the (unnormalized) correlation between term pairs and it can be interpreted as a similarity score of the terms j_1 and j_2, the computation of which is exclusively based on the use of these terms in available annotations. The eigenvectors of T (i.e. the columns of V_k) are 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 Equation (5), the i_{th} row of $\widetilde{A}$ can be written as

$$\widetilde{a}_i^T = a_i^T\,V_k\,V_k^T \tag{7}$$

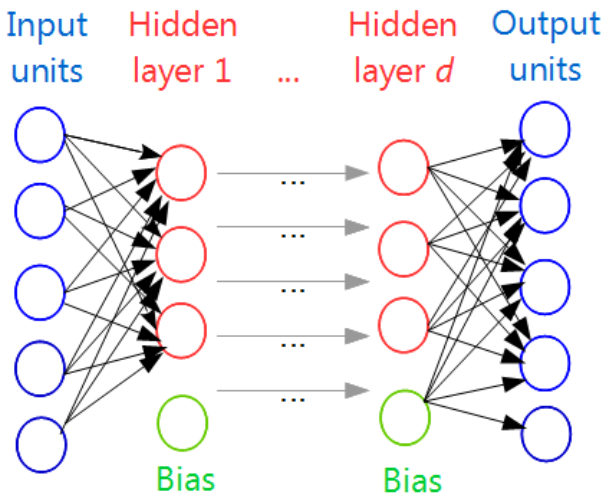


Figure 2: An autoencoder neural network with d hidden layers. The number of input units is equal to the number of output units, while there are usually fewer units in each hidden layer.

Thus, the original annotation profile is first transformed in the eigen-term domain, while retaining only the first k eigenterms by the multiplication with V_k, and then mapped back to the original domain by means of V_k^T. This corresponds to projecting the original vector a_i^T onto the k-dimensional subspace spanned by the columns of V_k.

2.2 Autoencoder Neural Network

An autoencoder is a feed-forward artificial neural network with the same input and target output. A small hidden layer in an autencoder network creates an information bottleneck, forcing the network to compress the data into a low-dimensional representation. As with the tSVD method, this modelling of the data can be used to make predictions.

For a simple autoencoder with a single hidden layer, the vector of the hidden unit activities, h, is given by

$$h = f(W_e \cdot a + bias_e) \tag{8}$$

where f is the activation function (we use the logistic sigmoid function in this work), W_e is a parameter matrix, and $bias_e$ is a vector of bias parameters. The hidden representation of the data is then mapped back into the space of a using the decoding function:

$$\hat{a} = f(W_d \cdot h, +bias_d) \tag{9}$$

where W_d is the decoding matrix and $bias_d$ a vector of bias parameters. We learn the parameters of the autoencoder by performing stochastic gradient descent to minimize the reconstruction error, the MSE between a and $\hat{a}$.

$$MSE(a, \hat{a}) = ||a - \hat{a}||_2^2 = ||a - (W_d \cdot h + bias_d)||_2^2 \tag{10}$$

When the hidden layer has fewer dimensions than a, the autoencoder learns a compressed representation of the training data. In fact, *an autoencoder with k linear hidden units will learn to project the data onto its first k principal components, and the decoded data matrix is exactly the tSVD matrix with the top k singular values* [14]. Non-linear hidden units allow an autoencoder to learn more complex encoding functions, as do additional hidden layers.

As in the tSVD approach, the matrix A is an array of m gene profiles with n possible features defined in Equation 2, such that gene profile a_i is the i^{th} row of A. An autoencoder is trained to learn these gene profiles and produces a prediction matrix $\widetilde{A}$ as described in Fig. 3.

Given the input matrix $A \in \{0,1\}^{m \times n}$, where rows and columns correspond to genes and features, respectively:

1. Fix a number h of hidden units ($h \in \mathbb{N}$, $h < m$), and a number d of hidden layers ($d \in \{1, ..., maxhl\}$)
2. Training: for each gene profile a_i of A, where $i \in [1, m]$:
   (a) for each training iteration:
      i. for each d hidden layer:
         a) compute hidden activation h_i from input a_i (Equation 8)
      ii. compute reconstructed output $\hat{a}_i$ from hidden activation h_i (Equation 9)
      iii. compute error gradient (Equation 10)
      iv. back-propagate error gradient to update weight parameters
3. Testing: for each gene profile a_i of A, where $i \in [1, m]$:
   (a) autoencode a_i and produce $\hat{a}_i$
   (b) set $\hat{a}_i$ as i^{th} row of the output matrix $\widetilde{A}$

Figure 3: Overview of the autoencoder neural network algorithm.

2.3 Predictions

The tSVD and autoencoder both provide a prediction matrix $\widetilde{A}$ of real values, with larger values indicating a higher predicted likelihood. For an ROC curve analysis, only the relative ordering of these predictions is relevant. To make binary predictions, we set a threshold τ such that $\widetilde{A}(i,j) > \tau$ is interpreted as a prediction that gene i should be annotated with feature j.

2.4 Autoencoder Training Details

Autoencoder neural networks were trained using the free GPU-accelerated software package Torch7 [21] using stochastic gradient descent with a learning rate of 0.01 for 25 iterations. L2 regularization was used on all weights, which were initialized randomly from the uniform distribution over $[0, 1]$. The hidden unit function is a Sigmoid.

2.5 Datasets

The GO database contains annotation datasets for a variety of species, and for each of the three GO sub-ontologies: *Biological Processes* (BP), *Molecular Functions* (MF), and *Cellular Components* (CC). We focused on the *Bos taurus* (cattle) and *Gallus gallus* (red junglefowl) gene sets, which are available from the Genomic and Proteomic Data Warehouse (GPDW) [22] [23]. We use the July 2009 version of the datasets for analyzing and selecting hyper-parameters, and

the March 2013 version for comparing prediction algorithms. Table 1 describes the size and number of annotations in each version. We exclude all annotations that are flagged as *IEA* (inferred from electronic annotation) or *ND* (no biological data available), and all feature terms and genes that do not appear in both dataset versions.

root term, which has the sub-ontology name (BP, CC, MF). In January 2014, GO contained about 39,000 terms describing gene and gene product features, with more than 25,450 BP, 9,650 MF and 3,350 CC terms. However, these are far from complete and new annotations are added regularly; over a third of the biological process annotations have been added within the last four years.

3. RESULTS AND DISCUSSION

We perform two separate experiments. First, we analyze the effects of hyper-parameters for both tSVD and the autoencoder algorithms on a validation set created by holding-out (removing) 10% of the annotations from the July 2009 database, then we test the prediction algorithms on *new* annotations that were added in the 2013 version. In both cases, the goal is to identify missing annotations within the large set of negative training examples. Fig. 4 visually describes the analysis procedure.

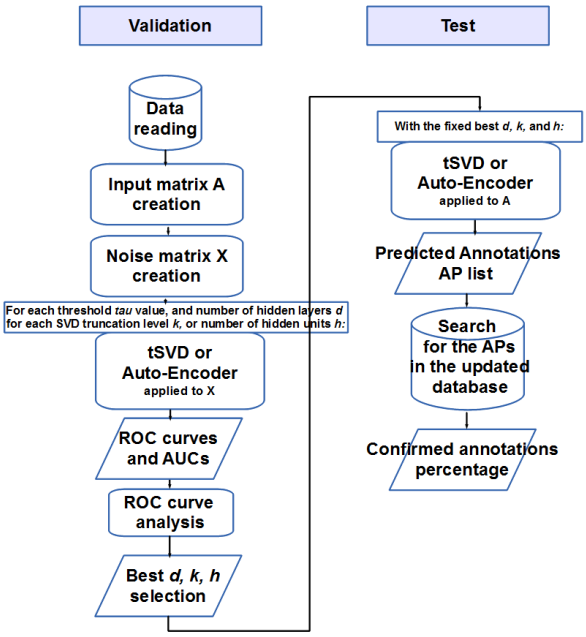


Figure 4: A flowchart of our analysis, with the hyper-parameter selection and *validation* procedure on the left, and the *test* procedure on the right. A rounded rectangle represents an operation, repeated in a cycle if attached to a sharp rectangle. A parallelogram represent an output production step, and a cylinder represents an interaction with the database.

3.1 Hyper-Parameter Analysis

For tSVD, the number of singular values is a hyper-parameter that determines the rank of the final prediction matrix, and is usually chosen through cross-validation. In an autoencoder network, the analogous hyper-parameter is the number of hidden units. These hyper-parameters control the complexity of the model; keeping a large number of singular values or using a large number of hidden units results in a very accurate reconstruction of the input data matrix, but will overfit to noise, such as missing annotations and inaccuracies. Figure 5 and Fig. 6 show how there is often an *optimal* hyper-parameter of this type. The best hyper-parameters for each data set are shown in Table 2.

The curves for each type of sub-ontology have similar behavior. For the Cellular Component annotation datasets, the autoencoder algorithm always outperform tSVD, regardless of the number of singular values. For the Molecular Function datasets, the autoencoder and tSVD have similar AUCs with singular values in the range $[20, 50]$, while autoencoder networks outperform tSVD in the other intervals. For Biological Process datasets, the autoencoders outperform tSVD only when it uses the maximum possible number of hidden units.

3.2 Predictive Accuracy

We test the tSVD and autoencoder algorithms on a set of annotations added to the database between July 2009 and March 2013. Training and testing was performed on the unfolded matrices described in Equation 2 to eliminate the possibility of trivial predictions. The performance metric is the percentage of the top 100 predictions from each method that were added to the database during this period. The results are displayed in Table 3, along with results from four other state-of-the-art algorithms from the computational gene annotation literature:

1. tSVD with gene clustering (SIM1) [24] [25]
2. tSVD with gene clustering and term-term similarity weights (SIM2) [24] [25]
3. Probabilistic Latent Semantic Analysis (pLSA) [26]
4. Latent Dirichlet Allocation (LDA) [27]

Overall, the tSVD-based techniques (tSVD, SIM1, SIM2) achieve similar performance, and LDA appears comparable to these methods. The pLSA algorithm performs slightly better than these methods on most of the datasets, and the autoencoder networks are consistently the best. The autoencoder networks improve performance by +6% to +36% with respect to the second best method.

3.3 Novel Predictions

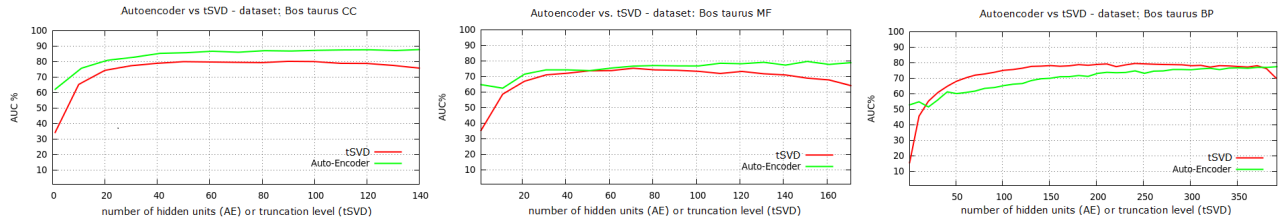
We examine the predicted annotations with highest likelihood score *that are not already annotated in the GO database*. Many of the predicted annotations are rather obvious high-level descriptive features such as *cellular process*, so we list the three *interesting* predictions with the highest likelihood in Table 4, where we define *interesting* as an annotation with distance greater than two from the root node in the ontology tree.

4. CONCLUSIONS

Gene function annotation databases are an essential tool in biomedical research, yet existing databases are incomplete and contain inaccuracies. In this work, we have shown

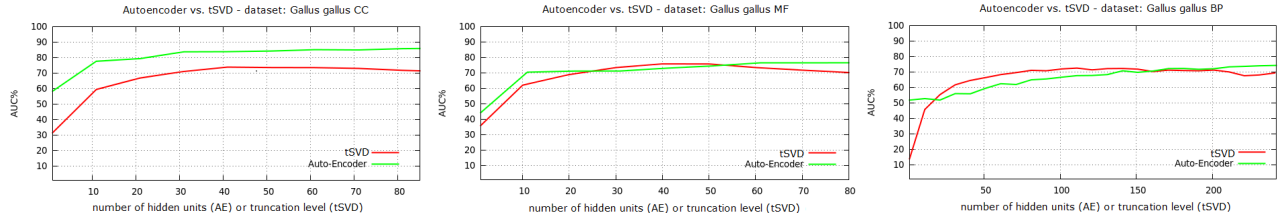
Table 1: Quantitative characteristics of the considered annotation datasets in the July 2009 database version versus the March 2013 database version used for testing. Numbers do not include annotations inferred from electronic annotations (*IEA*), those for which no biological data is available (*ND*), obsolete terms, or obsolete genes. #gs is the number of genes; #fs is the number of biological function features; #as is the number of annotations; Δ is the difference of annotation amounts of the #gs genes and the #fs features between the two database versions, and Δ% is the percentage difference.

	July 2009			March 2013	#as comparison	
Dataset	#gs	#fs	#as	#as	Δ	Δ%
Bos taurus CC	497	493	8,003	9,683	1,680	20.99%
Bos taurus MF	543	856	4,295	6,394	2,099	48,87%
Bos taurus BP	512	2,719	17,145	27,075	9,930	57.92%
Gallus gallus CC	260	344	3,717	3,798	81	2.18%
Gallus gallus MF	309	501	2,358	2,654	256	10.86%
Gallus gallus BP	275	1,824	8,350	11,984	3,634	43.52%



(a) (b) (c)

Figure 5: AUC values for the tSVD and autoencoder predictions with different hyper-parameter choices (number of singular values and number of hidden units, respectively) for Bos taurus Cellular Components (5a), Molecular Functions (5b), and Biological Process (5c). For comparison purposes, we use an autoencoder with a single hidden layer.



(a) (b) (c)

Figure 6: AUC values for the tSVD and autoencoder predictions with different hyper-parameter choices (number of singular values and number of hidden units, respectively) for Gallus gallus Cellular Components (6a), Molecular Functions (6b), and Biological Process (6c). For comparison purposes, we use an autoencoder with a single hidden layer.

Table 2: Hyper-parameters were optimized separately for each algorithm and dataset. We select the number of k singular values for tSVD, the number of clusters c for the SIM1 and SIM2 methods as described in [24]; the number of topics t in pLSA as described in [26]; the number of topics t in LDA as described in [27]; and the number of hidden units h in each of d hidden layers for the autoencoder (AE) algorithm.

	tSVD	SIM	pLSA	LDA	**AE**	
Dataset	k	c	t	t	h	d
Bos taurus CC	90	3	12		465	2
Bos taurus MF	71	3	13		302	3
Bos taurus BP	241	5	112		500	2
Gallus gallus CC	51	3	25		258	3
Gallus gallus MF	41	2	74		271	3
Gallus gallus BP	111	3	126		253	2

Table 3: Results related to the $topK = 100$ annotations predicted by several available methods applied to the considered datasets. We applied the methods to the July 2009 GPDW dataset versions, produced the top 100 most likely annotation list, and search for these annotations in the updated March 2013 GPDW database version. SIM1 is tSVD with clustering from [24]; SIM2 is tSVD with clustering and term-term similarity weights from [24]; pLSA is Probabilistic Latent Semantic Analysis from [26], and LDA is Latent Dirichlet Allocation from [27]. AE is the autoencoder. Bos taurus: cattle. Gallus gallus: red junglefowl. CC: cellular component. MF: molecular function. BP: biological process. Variable λ represents the likelihood predicting correctly 100 annotations by selecting them randomly among the non-annotations (0's) in the input matrix.

	upDB%						
Dataset	λ	tSVD	SIM1	SIM2	pLSA	LDA	**AE**
Bos taurus CC	0.69	26	23	32	33	30	**40**
Bos taurus MF	0.45	23	17	13	20	16	**34**
Bos taurus BP	0.67	14	7	14	30	19	**45**
Gallus gallus CC	0.09	10	15	14	23	20	**30**
Gallus gallus MF	0.19	6	2	2	16	8	**52**
Gallus gallus BP	0.72	13	14	13	31	10	**37**

Table 4: For each dataset, the three strongest, novel, *interesting* predictions from the autoencode algorithm are listed. These predictions are novel in that they are not included in the March 2013 version of the GO used to train the algorithm.

gene name	gene symbol	feature ID	feature name
Gallus gallus - Cellular Component			
RPS15 ribosomal protein S15.	RPS15	GO:0043231	intracellular membrane-bounded organelle
BIRC5 baculoviral IAP repeat containing 5.	BIRC5	GO:0043231	intracellular membrane-bounded organelle
CHRNA7 cholinergic receptor, nicotinic, alpha 7 (neuronal) .	CHRNA7	GO:0043231	intracellular membrane-bounded organelle
Gallus gallus - Molecular Function			
TPD52 tumor protein D52	TPD52	GO:0003676	nucleic acid binding
ATP2A1 ATPase, Ca++ transporting, cardiac muscle, fast twitch 1.	ATP2A1	GO:0003676	nucleic acid binding
DYRK2 dual-specificity tyrosine-(Y)-phosphorylation regulated kinase 2.	DYRK2	GO:0003676	nucleic acid binding
Gallus gallus - Biological Process			
NAIF1 nuclear apoptosis inducing factor 1.	C9ORF90	GO:0043170	macromolecule metabolic process
C1H2ORF49 chromosome 1 open reading frame, human C2orf49.	C2ORF49	GO:0043170	macromolecule metabolic process
ALB albumin.	ALB	GO:0043170	macromolecule metabolic process
Bos taurus - Cellular Component			
ACTA2 actin, alpha 2, smooth muscle, aorta.	ACTA2	GO:0043231	intracellular membrane-bounded organelle
KDM5D lysine (K)-specific demethylase 5D.	KDM5D	GO:0043231	intracellular membrane-bounded organelle
GDI1 GDP dissociation inhibitor 1.	GDI1	GO:0043231	intracellular membrane-bounded organelle
Bos taurus - Molecular Function			
CACNA1B calcium channel, voltage-dependent, N type, alpha 1B subunit.	CACNA1B	GO:0003824	catalytic activity
CHRNA7 cholinergic receptor, nicotinic, alpha 7.	CHRNA7	GO:0003824	catalytic activity
SGSM3 small G protein signaling modulator 3 .	SGSM3	GO:0003824	catalytic activity
Bos taurus - Biological Process			
GJD2 gap junction protein, delta 2, 36kDa.	GJD2	GO:0044237	cellular metabolic process
OPA3 optic atrophy 3 (autosomal recessive, with chorea and spastic paraplegia).	OPA3	GO:0044237	cellular metabolic process
RPGR retinitis pigmentosa GTPase regulator.	RPGR	GO:0044237	cellular metabolic process

how deep autoencoder networks can be used to help curate these annotation databases and predict novel gene functions. We have highlighted the similarities between our algorithm and tSVD, and shown that it performs better than tSVD, pLSA, and LDA on six separate datasets from the Gene Ontology consortium. The autoencoder method we have proposed is not limited to gene function annotation, and could be used for other problems such as collaborative-filtering for product-recommendation systems. The approach has numerous advantages: (1) autoencoders can be trained *online* with very large datasets, (2) they can be trained quickly using graphics processors, and (3) the number and size of the hidden layers provides an easy way of controlling the complexity of the model. In our results, autoencoders with two or three hidden layers worked better than shallow autoencoders with a single hidden layer, suggesting that deep learning methods could be useful for this application.

Future work will address advantages and issues related to the application of the same methods and rule to the prediction of multi-terminologies, not only annotations. Finally, our goal is to furnish a Web service to our implemented methods and integrate such Web application with other available services within the Search Computing framework [28], in order to provide support for answering complex life science questions [29].

5. REFERENCES

[1] G. Pandey, V. Kumar, and M. Steinbach, "Computational approaches for protein function prediction: A survey". Twin Cities: Department of Computer Science and Engineering, University of Minnesota, 2006.

[2] The Gene Ontology Consortium, "Creating the Gene Ontology Resource: Design and Implementation". *Genome Research*, vol. 11, pp. 1425-1433, 2001.

[3] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene Ontology: tool for the unification of biology". *Nature Genetics*, vol. 25.1: pp. 25-29, 2000.

[4] O. D. King, R. E. Foulger, S. S. Dwight, J. V. White, and F. P. Roth, "Predicting gene function from patterns of annotation". *Genome Research* 13.5: pp. 896-904, 2003.

[5] Y. Tao, L. Sam, J. Li, C. Friedman, and Y. A. Lussier, "Information theory applied to the sparse gene ontology annotation network to predict novel gene function". *Bioinformatics*, vol. 23.13: pp. 529-538, 2007.

[6] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function". *Bioinformatics*, vol. 22.7: pp. 830-836, 2006.

[7] S. Raychaudhuri, et al. "Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature". *Genome Research*, vol. 12.1: pp. 203-214, 2002.

[8] A. Perez, C. Perez-Iratxeta, P. Bork, G. Thode, and M. A. Andrade, "Gene annotation from scientific literature using mappings between keyword systems". *Bioinformatics*, vol. 20.13: pp. 2084-2091, 2004.

[9] G. Yu, H. Rangwala, C. Domeniconi, G. Zhang, and Z. Yu, "Protein Function Prediction with Incomplete Annotations". *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11.3: pp. 579 - 591, 2013 .

[10] P. Khatri, B. Done, A. Rao, A. Done, and S. Draghici, "A semantic analysis of the annotations of the human genome". *Bioinformatics*, vol. 21.16: pp. 3416-3421, 2005.

[11] M. Masseroli, M. Tagliasacchi, and D. Chicco, "Semantically improved genome-wide prediction of Gene Ontology annotations". *Proceedings of IEEE ISDA 2011, the 11th International Conference on Intelligent Systems Design and Applications*, pp. 1080 - 1085 , 2011.

[12] P. Pinoli, D. Chicco, and M. Masseroli. "Improved biomolecular annotation prediction through Weighting Scheme methods". *Proceedings of CIBB 2013, the Tenth International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, Nice, France, pp. 1-12, 2013.

[13] H. Bourlard, and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition." *Biological cybernetics*, vol. 59.4-5, pp. 291-294, 1988.

[14] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima." *Neural networks*, vol. 2.1, pp. 53-58, 1989.

[15] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures". *Journal of Machine Learning Research-Proceedings Track*, vol. 27, pp. 37-50, 2012.

[16] G. H. Golub, and C. Reinsch, "Singular value decomposition and least squares solutions". *Numerische Mathematik* vol. 14.5: pp. 403-420, 1970.

[17] M. Masseroli, M. Tagliasacchi, "Web resources for gene list analysis in biomedicine", In: Lazakidou, A., editor. *Web-based Applications in Health Care and Biomedicine*. Heidelberg, D: Springer, Annals of Information Systems Series, vol. 7, pp. 117-141, 2010

[18] B. Done, P. Khatri, A. Done, and S. Draghici, "Semantic analysis of genome annotations using weighting schemes". *Proceedings of CIBCB 2007 , the IEEE Symposium Computational Intelligence and Bioinformatics and Computational Biology*, pp. 212 - 218, 2007.

[19] B. Done, P. Khatri, A. Done, and S. Draghici, "Predicting novel human gene ontology annotations using semantic analysis." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7.1: pp. 91-99, 2010.

[20] D. Chicco, and M. Masseroli, "A Discrete Optimization Approach for SVD Best Truncation Choice based on ROC Curves". *Proceedings of IEEE BIBE 2013, the 13rd International Conference on Bioinformatics and Bioengineering*, pp. 1-4, 2013.

[21] R. Collobert, K. Kavukcuoglu and C. Farabet, "Torch7: A Matlab-like Environment for Machine Learning". *BigLearn*, NIPS Workshop, 2011.

[22] A. Canakoglu, G. Ghisalberti, and M. Masseroli,

"Integration of Biomolecular Interaction Data in a Genomic and Proteomic Data Warehouse to Support Biomedical Knowledge Discovery". *Computational Intelligence Methods for Bioinformatics and Biostatistics*, Springer Berlin Heidelberg, pp. 112-126, 2012.

[23] F. Pessina, M. Masseroli, and A. Canakoglu, "Visual composition of complex queries on an integrative Genomic and Proteomic Data Warehouse". *Engineering*, vol. 5:10B, pp. 1-8, 2013.

[24] D. Chicco, M. Tagliasacchi, and M. Masseroli, "Genomic annotation prediction based on integrated information". *Computational Intelligence Methods for Bioinformatics and Biostatistics*, Springer Berlin Heidelberg, pp. 238-252, 2012.

[25] D. Chicco, M. Tagliasacchi, and M. Masseroli, "Biomolecular annotation prediction through information integration". *Proceedings of CIBB 2011, the 8th Computational Intelligence Methods for Bioinformatics and Biostatistics*, pp. 1-8, 2011.

[26] M. Masseroli, D. Chicco, and P. Pinoli, "Probabilistic Latent Semantic Analysis for prediction of Gene Ontology annotations". *Proceedings of IEEE IJCNN 2012, the International Joint Conference on Neural Networks*, pp- 1-8 2012.

[27] P. Pinoli, D. Chicco, and M. Masseroli, "Latent Dirichlet Allocation based on Gibbs Sampling for Gene Function Prediction". *Proceedings of IEEE CIBCB 2014, the Conference on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 1-4, 2014.

[28] S. Ceri, D. Braga, F. Corcoglioniti, M. Grossniklaus, and S. Vadacca, "Search computing challenges and directions". *Springer*, Berlin Heidelberg, 2010.

[29] D. Chicco, "Integration of bioinformatics web services through the Search Computing technology". *Technical Report*, TR 2012/02. Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milan, Italy.