Supplementary Materials of “GOLabeler: Improving Sequence-based Large-scale Protein Function Prediction by Learning to Rank”

Ronghui You[1,2], Zihan Zhang[1,2], Yi Xiong[3], Fengzhu Sun[2,4], Hiroshi Mamitsuka[5,6], and Shanfeng Zhu[1,2*]

[1] School of Computer Science and Shanghai Key Lab of Intelligent Information Processing and

[2] Centre for Computational System Biology, Fudan University, Shanghai, China,

[3] Department of Bioinformatics and Biostatistics, Shanghai Jiaotong University, Shanghai, China,

[4] Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA,

[5] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan,

[6] Department of Computer Science, Aalto University, Finland

January 29, 2018

1 Learning to Rank

Learning to rank (LTR) [1, 2] is a powerful machine learning paradigm for ranking objects. It has been found wide applications in document retrieval, collaborative filtering, web searching and bioinformatics. For instance, with respect to users queries, LTR model has been used to rank more relevant web pages higher. AFP can be deemed as a large-scale multi-label learning problem, since each protein can be assigned many GO terms. Here protein is treated as instance, and GO term is tread as label, where we have around 40000 labels. AFP is similar to web searching in the sense that we can rank the labels (GO or web pages) according to the relevance scores (association scores). Thus we can make use of LTR to solve the AFP problem. In the testing stage of AFP, after we rank labels (GO) according to the prediction scores, we can choose top ranked labels as the true labels.

The detailed methods used in LTR can be divided into 3 groups: point-wise approaches, pair-wise approaches, and list-wise approaches. In point-wise approaches, the ranking problem is approximately by a classical classification or ordinal regression problem [3, 4]; For example, the LTR model tries to predict the relevance score of a query-document pair. In pair-wise approaches, such as LambdaMART [5] and RankingSVM [6], the ranking problem is transformed to a pairwise classification problem. Given a pairs of documents with respect to a specific query, the LTR model tries to tell which document is more relevant. List-wise approaches take ranked lists as instances in both training and prediction, such as ListNet [7] and ListMLE [8]. In this case, although the group structure of ranking is fully utilized in learning, the optimization of evaluation measure is difficult and approximations or bounds have to be used. In this study, we make use of LambdaMART to solve the challenge of AFP, which has demonstrated its good performance in multiple international machine learning competitions, such as BioASQ challenge [9] and Yahoo Learning to Rank competition [10].

2 Experimental setting

2.1 Experiment set-up

The main purpose of this work is to design an effective algorithm for the large-scale function prediction of new proteins (no-knowledge proteins), which is also the motivation of CAFA challenge. Although many protein function prediction methods have been proposed, the performance of existing computational methods over CAFA1 and CAFA2 is far from satisfactory, where homology based methods were very competitive. The test data in GOLabeler exactly follows the setting of CAFA on no-knowledge protein, which has gained experimental annotations after T0 (Jan 2016) and before T1 (Oct 2016). Please note that these protein might have been added into UniProt before 2016, but only received experimental annotations between Jan 2016 and Oct 2016. The experimental results of GOLabeler demonstrate that, the good performance of GOLabeler is largely attributed to two component methods, BLAST-KNN and LR-Interpro. In contrast to BLAST-KNN utilizing protein homology information, LR-Interpro makes use of domain, family and motif information of protein for function prediction. This means that only sequence similarity is insufficient for the good performance of GOLabeler. Since we focus on the function prediction of no-knowledge proteins, it is expected that the number of test proteins will not be large. In spite of relatively large number of training data, the moderate performance of computational methods in CAFA highlights the challenge of this problem, which GOLabeler tries to address.

2.2 The setting of k

For training GOLabeler, we combined top k GO terms by each component method as the candidate GO terms. For choosing a suitable k, we did 5-fold cross validation over LTR training set with k=10, 30, 50, 70. Table 1 showed the performance. We found that k=30 reached the best F_{max}. With a large value of k, the performance will decrease slightly and it will take much more time to train the ranking model. Considering all of these, we use k=30 in GOLabeler.

Table 1: The performance of GOLabeler over LTR training by 5-fold cross validation with different setting of k.

k	F_{max}			AUPR		
	MFO	BPO	CCO	MFO	BPO	CCO
10	0.612	0.312	0.689	0.566	0.170	0.692
30	**0.615**	**0.314**	**0.691**	**0.574**	0.172	**0.705**
50	0.608	**0.314**	**0.691**	**0.574**	**0.174**	**0.705**
70	0.608	0.312	0.690	**0.574**	**0.174**	**0.705**

3 Experimental result

3.1 The hierarchical consistency of preidictions

Given a target protein, the candidate GO terms are ranked by ranking model in GOLabeler. Although it is possible that a parent GO term receives a lower score than its children terms, this kind of inconsistence is rare. We have tried a direct refinement strategy to ensure the consistency, if the original prediction score of a GO term is lower than the score of any its children term, we replace its score with the maximum score of its children term. We find that the performance of GOLabeler only changes very slightly after this refinement (around 0.001 in terms of F_{max}). To reduce confusion, we make use of this refinement strategy on the output of GOLabeler in the online webserver.

3.2 The performance of different methods in terms of average precision and recall

Tables 2 and 3 show the summary results in terms of average precision and recall.

Table 2: Performance comparison of GOLabeler with component methods. B-K: BLAST-KNN. GOLabeler (B+I): only BLAST-KNN and LR-InterPro were used for components in GOLabeler. GOLabeler (All): all five components are used.

	F_{max}			Avg-Precision			Avg-Recall		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.242	0.299	0.653	0.353	0.303	**0.766**	0.184	0.296	0.570
B-K	0.573	0.339	0.620	0.574	0.350	0.686	0.571	0.329	0.566
LR-3mer	0.255	0.301	0.664	0.342	0.306	0.736	0.203	0.296	0.604
LR-InterPro	0.556	0.351	0.654	0.577	0.340	0.705	0.537	0.363	0.611
LR-ProFET	0.330	0.265	0.633	0.324	0.246	0.666	0.336	0.286	0.603
GOLabeler (B+I)	0.578	0.352	0.665	0.554	**0.378**	0.677	**0.604**	0.330	0.654
GOLabeler (All)	**0.580**	**0.370**	**0.687**	**0.580**	0.331	0.696	0.580	**0.419**	**0.676**

Table 3: Performance comparison of GOLabeler with ensemble techniques and also improvement by adding LTR2 to LTR1. WV: Weighted voting.

	F_{max}			Avg-Precision			Avg-Recall		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
One vote	0.543	0.335	0.682	0.556	0.302	0.658	0.530	0.376	**0.707**
WV (LTR1)	0.571	0.368	**0.692**	0.570	0.330	0.702	0.571	0.416	0.682
WV (LTR1+2)	0.571	0.370	**0.692**	0.563	**0.361**	0.701	0.579	0.380	0.683
Consensus	0.543	0.360	**0.692**	0.539	0.337	**0.711**	0.548	0.386	0.673
GOLabeler (LTR1)	0.580	0.370	0.687	**0.580**	0.331	0.696	0.580	**0.419**	0.676
GOLabeler (LTR1+2)	**0.586**	**0.372**	0.691	0.571	0.356	0.704	**0.602**	0.390	0.678

3.3 p-values for main comparison experiments

Tables 4 and 5 show the summary results with p-values.

References

[1] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[2] H. Li. A short introduction to learning to rank. *IEICE Transactions*, 94-D(10):1854–1862, 2011.

[3] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2007.

[4] Koby Crammer, Yoram Singer, et al. Pranking with ranking. In *Nips*, volume 1, pages 641–647, 2001.

[5] Christopher Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010.

Table 4: Performance comparison of GOLabeler with component methods using 100 bootstrapped datasets with replacement. We use paired *t*-test to statistically evaluate the performance difference between the best performance method (in boldface in tables) and all other methods. GOLabeler (B+I): only BLAST-KNN and LR-InterPro were used for components in GOLabeler. GOLabeler (All): all five components are used.

methods	AUPR			F_{max}			S_{min}		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.140 4.535×10^{-123}	0.151 6.958×10^{-113}	0.591 2.332×10^{-129}	0.243 2.107×10^{-134}	0.299 6.755×10^{-115}	0.653 3.085×10^{-79}	7.654 9.995×10^{-16}	15.931 1.495×10^{-77}	6.555 5.547×10^{-57}
BLAST-KNN	0.452 4.848×10^{-96}	0.190 1.247×10^{-74}	0.559 1.889×10^{-114}	*0.572** 1.921×10^{-13}	0.339 1.926×10^{-89}	0.623 1.819×10^{-95}	*5.119** 4.732×10^{-16}	15.683 1.495×10^{-77}	*5.640** 5.547×10^{-57}
LR-3mer	0.143 5.897×10^{-123}	0.152 8.670×10^{-113}	0.600 1.775×10^{-128}	0.255 1.562×10^{134}	0.301 1.959×10^{-115}	*0.664** 9.499×10^{-71}	7.557 1.920×10^{-121}	15.902 6.513×10^{-92}	6.437 5.096×10^{-97}
LR-InterPro	*0.537** 0.659×10^{-18}	*0.197** 1.322×10^{-77}	*0.636** 4.976×10^{-97}	0.557 8.512×10^{-62}	*0.351** 3.616×10^{-78}	0.655 1.776×10^{-82}	5.216 2.441×10^{-40}	*15.622** 1.256×10^{-87}	5.789 1.858×10^{-84}
LR-ProFET	0.175 1.802×10^{-122}	0.095 1.448×10^{-137}	0.551 4.157×10^{-94}	0.331 8.533×10^{-129}	0.264 1.684×10^{-136}	0.634 8.743×10^{-96}	7.780 2.113×10^{-103}	16.990 5.921×10^{-124}	6.388 2.901×10^{-87}
GOLabeler (B+I)	0.538 5.815×10^{-40}	0.172 3.742×10^{-107}	0.659 2.990×10^{-89}	0.578 1.332×10^{-6}	0.351 8.891×10^{-76}	0.667 4.656×10^{-64}	5.096 3.301×10^{-25}	15.193 5.644×10^{-18}	**5.442**
GOLabeler (ALL)	**0.546**	**0.224**	**0.701**	**0.579**	**0.369**	**0.686**	**5.051**	**15.147**	5.524 5.781×10^{-36}

[6] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, 1999.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[8] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[9] Ke Liu, Shengwen Peng, Junqiu Wu, ChengXiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. Meshlabeler: improving the accuracy of large-scale mesh indexing by integrating diverse evidence. *Bioinformatics*, 31(12):339–347, 2015.

[10] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, pages 1–24, 2011.

Table 5: Performance comparison of GOLabeler with ensemble techniques and also improvement by adding LTR2 to LTR1.

methods	AUPR			F_{max}			S_{min}		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
Consensus	0.454 3.178×10^{-83}	0.222 1.169×10^{-44}	0.669 1.191×10^{-68}	0.543 8.384×10^{-81}	0.359 1.858×10^{-81}	**0.693**	5.654 4.705×10^{-60}	15.486 1.762×10^{-83}	5.675 8.816×10^{-41}
One Vote	0.423 3.596×10^{-87}	0.193 3.900×10^{-94}	0.651 3.105×10^{-78}	0.543 2.404×10^{-74}	0.334 3.168×10^{-108}	0.683 1.895×10^{-60}	6.127 2.940×10^{-64}	16.199 6.787×10^{-112}	5.908 2.576×10^{-62}
Weighted Voting (LTR1)	0.531 2.234×10^{-53}	0.229 3.935×10^{-49}	0.694 4.461×10^{-61}	0.571 1.245×10^{-54}	0.367 1.522×10^{-45}	**0.693** 0.045	5.088 1.294×10^{-29}	15.088 6.981×10^{-37}	5.523 3.704×10^{-13}
Weighted Voting (LTR1+2)	0.531 7.285×10^{-53}	0.231 1.924×10^{-36}	0.695 1.042×10^{-56}	0.572 6.714×10^{-53}	0.369 3.426×10^{-26}	**0.693** 0.475	5.088 7.760×10^{-29}	15.068 1.525×10^{-28}	5.513 1.437×10^{-7}
GOLabeler (LTR1)	0.546 5.394×10^{-25}	0.224 4.311×10^{-68}	<u>**0.701**</u>	0.579 1.838×10^{-36}	0.369 1.798×10^{-29}	0.686 1.833×10^{-35}	5.051 2.039×10^{-27}	15.147 6.095×10^{-64}	5.524 2.484×10^{-17}
GOLabler (LTR1+2)	<u>**0.549**</u>	<u>**0.235**</u>	0.697 2.020×10^{-56}	<u>**0.586**</u>	<u>**0.372**</u>	0.692 3.110×10^{-5}	<u>**5.000**</u>	<u>**15.022**</u>	<u>**5.491**</u>