Bioinformatics, 31(5), 2015, 745–752
doi: 10.1093/bioinformatics/btu715
Advance Access Publication Date: 29 October 2014
Original Paper

Data and text mining

Measuring the wisdom of the crowds in network-based gene function inference

W. Verleyen, S. Ballouz and J. Gillis*

Stanley Institute for Cognitive Genomics, Cold Spring Harbor Laboratory, Woodbury, NY 11797, USA

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on 1 May 2014; revised on 13 October 2014; accepted on 23 October 2014

ABSTRACT

Motivation: Network-based gene function inference methods have proliferated in recent years, but measurable progress remains elusive. We wished to better explore performance trends by controlling data and algorithm implementation, with a particular focus on the performance of aggregate predictions.

Results: Hypothesizing that popular methods would perform well without hand-tuning, we used well-characterized algorithms to produce verifiably 'untweaked' results. We find that most state-of-the-art machine learning methods obtain 'gold standard' performance as measured in critical assessments in defined tasks. Across a broad range of tests, we see close alignment in algorithm performances after controlling for the underlying data being used. We find that algorithm aggregation provides only modest benefits, with a 17% increase in area under the ROC (AUROC) above the mean AUROC. In contrast, data aggregation gains are enormous with an 88% improvement in mean AUROC. Altogether, we find substantial evidence to support the view that additional algorithm development has little to offer for gene function prediction.

Availability and implementation: The supplementary information contains a description of the algorithms, the network data parsed from different biological data resources and a guide to the source code (available at: http://gillislab.cshl.edu/supplements/).

Contact: jgillis@cshl.edu

1 Introduction

High-throughput genomic data often relies on computational methods for functional inference and interpretation. Improving our knowledge of gene function in otherwise uncharacterized genes is one major task to which these computational methods are put (Eisenberg *et al.*, 2000). While this is often called gene function prediction when being treated as a machine learning problem, essentially the same methods underlie a variety of important biomedical applications, such as candidate disease gene prioritization (Tranchevent *et al.*, 2011). Like many methods in machine learning, a major concern for function inference methods is the degree to which their performance is robust and generalizes from benchmark tasks to novel data (Bousquet and Elisseeff, 2002; Domingos, 2012). To some extent, systematic problems with generalizing past gene function prediction to future performance have been recognized by the field. These have led to critical assessments of function prediction, e.g. the critical assessment of *Mus musculus* gene function prediction (MouseFunc) (Pena-Castillo *et al.*, 2008) and the critical assessment of function annotation (CAFA) (Radivojac *et al.*, 2013), where groups compete to predict gene function. These assessments are intended to provide field-wide benchmarks for comparative performance and thereby help determine which research directions may be more fruitful. Generally speaking, gene function prediction methods rely on (i) prior function assignments for genes, (ii) data to characterize genes and (iii) an algorithm which learns the data features associated with the previous function assignments. Novel gene-function mappings are then predicted based on the learned data features; this can be done either in a gene-centric or function-centric way. Because individual laboratories approach this problem in quite different ways, characterizing the field overall in critical assessment has been difficult; it is hard to know what factors in one laboratory's implementation drives results. Indeed, even in critical assessments, the same laboratory may submit slightly altered versions of the same software and find their performance changes dramatically (Hamp *et al.*, 2013).

We solve the over-training problem addressed by critical assessments in an alternate way; by using verifiable implementations it can be checked that there are no adjustments or tweaks. Normally, 'fine-tuning' algorithms and data to work appropriately are necessary to obtain reasonable performance, but this is precisely what may contribute to poor generalization. Based on performance trends in previous critical assessments, we hypothesized that well-developed machine learning approaches would perform at a high level if using gold-standard data resources. That the performance assessment is not tuned to obtain artificially high performance can be ensured by using pre-existing and verifiable tests, i.e. the critical assessments. By having an in-house representative of the field as a whole, we are able to conduct additional experiments with greater control.

One of our principal interests is in exploring how aggregation improves performance. That it generally does so has been a finding common to previous assessments of function prediction and network inference methods and is a frequent expectation for machine learning in general (Gillis and Pavlidis, 2013a,b; Marbach *et al.*, 2012; Opitz and Maclin, 1999). However, the factors central to this effect have not been well characterized, likely because most laboratories will have an individualized approach which makes well-controlled comparison difficult. To overcome this, we will first benchmark a set of machine learning algorithms based upon data resources available from MouseFunc to establish they are representative samples of high-performing methods. Then, we will systematically apply these algorithms on various types of *Saccharomyces cerevisiae* data as a sample of 'field-wide' properties and investigate how their combination improves performance and on what factors this depends.

2 Methods

2.1 Network and gene annotation data

For the gene network data, each gene's interaction profile or set of gene interactions was treated as the feature data associated with that gene. We had access to the MouseFunc networks and data resources (Pena-Castillo *et al.*, 2008). These included protein annotation from Pfam (Finn *et al.*, 2006) and InterPro (Mulder *et al.*, 2005), protein–protein interaction (Brown and Jurisica, 2005), phenotype ontology from the mouse genome informatics database (Smith *et al.*, 2005), co-expression (Siddiqui *et al.*, 2005; Su *et al.*, 2004; Zhang *et al.*, 2004), phylogenetic profile (O'Brien *et al.*, 2005), disease data for *M.musculus* (Hamosh *et al.*, 2005). For the *S.cerevisiae* data, we generated networks from four types of biological data: (i) protein–protein interaction, (ii) sequence similarity, (iii) co-expression and (iv) semantic similarity or shared pathways data (e.g. Mistry and Pavlidis, 2008), each parsed in a specific manner. The protein–protein interaction network was constructed from the BioGRID database (version 3.2.103) (Chatr-Aryamontri *et al.*, 2013). We generated a binary network of only the physical interactions between the yeast species, with no further filtering on experimental type. For the sequence similarity, we generated a weighted network using the protein sequences from the *Saccharomyces* Genome Database (Cherry *et al.*, 2012) (orf_trans_all.fasta/31-5-2013). The sequence hits were calculated with psiBLAST (Altschul *et al.*, 1997) and the negative log10 *E*-value of the psiBLAST score (and thresholding to zero all values below 1 after transformation), previously shown to be a simple and reliable measure for sequence similarity (Enright *et al.*, 2002; Hawkins *et al.*, 2009), was used to weight the interactions in the network. We used an aggregate co-expression network, generated from 30 yeast microarray experiments on the Affymetrix Yeast Genome S98 Array (GPL90), characterizing 5457 genes with unique probes across 966 samples. Briefly, we downloaded each expression dataset from the Gene Expression Omnibus then used the Spearman correlation coefficient as weight edges between gene pairs in the co-expression network, and the aggregate was generated by summing the individual experiments (Gillis and Pavlidis, 2011a,b). The final network was then rank standardized. The semantic similarity network was generated based upon the biological pathways of *S.cerevisiae* in the kyoto encyclopedia of genes and genomes (KEGG) database (Ogata *et al.*, 1999). In short, the weight of each interaction in the network was taken as the Jaccard index of two genes sharing biological pathways, according to their appearance in the corresponding KEGG pathways. We used an overlap/intersect of 5455 genes for *S.cerevisiae*. As a gene annotation set, we used the gene ontology (GO) annotations from April 13, 2013. We propagated over the GO structure, and then filtered for GO groups on size such that each remaining group had between 20 and 1000 associated genes while excluding inferred from electronic annotation, a range that generally gives stable performance (Gillis and Pavlidis, 2011a,b). We were left with 1313 GO groups to be used in the remaining analyses. Note that while we use the term 'gene function prediction' throughout, there is generally no difference from the methods, data and analysis present in protein function prediction.

2.2 Algorithms and parameter settings

We benchmarked six machine learning algorithms on the MouseFunc data: (i) logistic regression (Fan *et al.*, 2008), (ii) random forest (Breiman, 2001), (iii) support vector machine with a stochastic gradient descent solver (Su *et al.*, 2004), (iv) passive aggressive (Crammer *et al.*, 2006), (v) GeneMANIA (Mostafavi *et al.*, 2008) and (vi) neighbor voting (Gillis and Pavlidis, 2011a,b). The first four algorithms are implementations from the scikit-learn machine learning framework (Pedregosa *et al.*, 2011) with GeneMANIA and neighbor voting having matlab implementations which we used. The first five algorithms are comparatively sophisticated methods, while neighbor voting serves as a simple but useful baseline. We then applied these algorithms to the four yeast data networks described previously. These methods were chosen based on their relative prominence within the machine learning literature, their similarity to methods used in Mousefunc and their continued popularity in gene function prediction research.

For the MouseFunc benchmarking, we used the default parameters for each of the algorithms. Logistic regression was used with dual formulation regularized with L2-norm penalization for the LIBLINEAR library, classes are weighted inversely proportional to their frequency; the regularization strength ($C = 1.0$) and tolerance ($\text{tol} = 0.001$) were kept to their default settings. The random forest classifier was used with 512 decision trees; the Gini impurity was used to split the decision tree whereby the maximum number of features for each individual decision trees was the square root of the total number of features. The stochastic gradient descent solver for the support vector machine uses the hinge loss function with L2-norm penalization; we used 10 iterations over the training data, classes were weighted inversely proportional to their frequency, and the regularization strength was kept to its default value ($C = 1.0$). Passive aggressive uses the hinge loss function; we used 10 iterations over the training data, and the regularization strength was kept to its default value ($C = 1.0$). GeneMANIA and neighbor voting were used with their default parameters. While in general we avoided altering any parameters to prevent over-training, one parameter of these algorithms was changed when switching from the MouseFunc data to the yeast data: the amount of trees in the random forest algorithm was changed from 512 to 300, roughly in proportion to the change in feature density. For our own algorithm implementation, the balanced ensemble logistic regression, for each training set of genes, an equally sized set of genes randomly drawn from the remainder was treated as the negative set; this was repeated 25 times and the aggregate prediction used as the algorithm's output.

All aggregation occurred at the level of the output from our predictors (i.e. gene list scores), allowing straightforward comparison between the effects of aggregation using different algorithms or different data. The aggregation score was computed based upon the average probabilistic scores of our predictors and the correlation-based feature selection (CFS) (Hall, 1998). For plain aggregation, predictor output is simply averaged. When using feature selection, we are attempting to select the predictors which worked better and did so independently from other predictors. These more useful predictors can then be weighted more heavily. This is accomplished by picking the predictors which gave results correlated with the output class and but not correlated with each other. In other words, those predictors which divided the data up correctly, but are not too similar to one another. CFS was applied to each fold independently, yielding a fold-specific aggregate prediction which was used to predict the held-out data in the conventional way. Consistent with our general methodology, we used the pre-existing WEKA implementation. Further details including the source code are available at the supplementary material at http://gillislab.cshl.edu/supplements/.

2.3 Performance metrics

Through-out, our evaluation is function-centric. To assess the benchmarking results and the performance of the algorithms on the data in our analyses, we relied upon four metrics: (i) area under the ROC curve (AUROC), (ii) area under the precision recall curve (AUPRC), (iii) maximum *F*-measure ($F_{\max}$) and (iv) precision by 20% recall (P20R). We ran the algorithms on each dataset and used a 3-fold cross validation whereby each fold has genes that were randomly left out from the functional annotation and the average over all folds yielding the reported performance.

3 Results

In characterizing and exploiting ensemble properties of gene function prediction algorithms, we are first concerned with assembling a robust algorithm and data resource set (Sections 3.1 and 3.2), before considering two related concepts: whether algorithms produce function predictions of similar quality (Section 3.3), and in what way combining predictions allows for improving the quality of function prediction (Sections 3.4 and 3.5).

3.1 Algorithms show broadly similar high performances on the MouseFunc data

The network label propagation algorithm we used, GeneMANIA, was one of the best performing algorithms in the original MouseFunc competition. Thus, in addition to the absolute performance we can measure methods exhibiting, the GeneMANIA results provide a good internal benchmark for our other methods. We performed our benchmarking of the individual algorithms using the MouseFunc data 'tasks'.

Each task used a set of GO groups from the Biological Process category with either 11–30, 31–100 or 101–300 genes. For each 'task' set, we used the provided MouseFunc network and a machine learning algorithm to obtain a set of predictions. This was repeated for each method, and the performance metrics were calculated using performance at predicting future (at the time of Mousefunc) gene-function assignments from GO (as in the last Mousefunc assessment task). In each case, the output is a list of ranked scores for the genes. The task set provides the true positives, which allows us to generate an ROC for each GO term in the task set. We then use the AUROC as the score for that GO term, and the average AUROC of those terms as the measure of performance of the algorithm (Fig. 1). Our algorithms show broadly similar performance in predicting gene functions with mean ROC performance of 0.81, 0.85 and 0.84 in the three biological process categories compared to that of 0.77, 0.82 and 0.8 among the original participants. The slight decrease in performance among the participants is due to occasional low scores among particular participants rather than superlative performance among our benchmarked set. While our benchmarked set do not outperform the winning algorithm, GeneMANIA, the AUROC distribution of GeneMANIA is similar in mean to the AUROC distribution of the neighbor voting, logistic regression, random forest and stochastic gradient descent for the gene set sizes >30 [$P > 0.1$, Kolmogorov–Smirnov (KS) test], with only modest differences in the smaller sized gene sets. Only the AUROC distribution of passive aggressive is significantly different from that observed with GeneMANIA (KS test $P \sim 0.0002$ for large GO groups in the Biological Process categories with an ROC of 0.81 compared to 0.88 for GeneMANIA). As we have abundant data to characterize

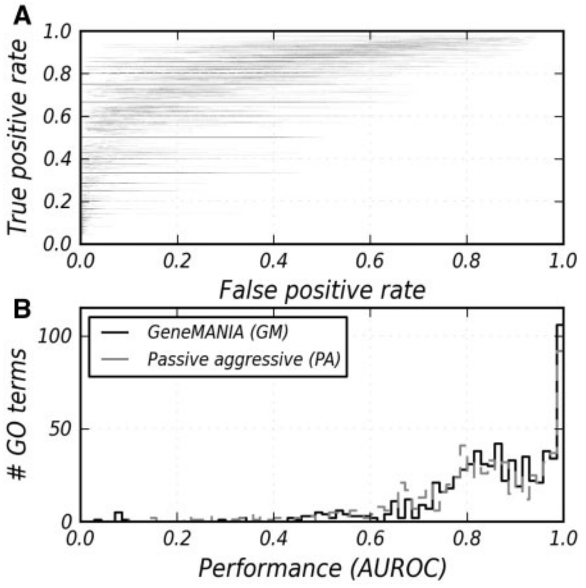

Fig. 1. ROC performance in Mousefunc data. (**A**) The average of ROCs using neighbor voting on aggregated MouseFunc data (mean AUROC = 0.82 and std AUROC = 0.16). The shaded lines represent the overlay of all 774 ROC curves, mostly passing through a similar range of values. (**B**) The distribution of the AUROCs for aggregated MouseFunc data of the best performing algorithm (GeneMANIA, black line, mean AUROC = 0.84 and std AUROC = 0.15) compared with the worst performing algorithm (passive aggressive, dashed line, mean AUROC = 0.81 and std AUROC = 0.16)

the distribution of performances (774 GO groups in this exercise), these results suggest that methods are performing well in conventional terms. At least some of this performance (AUROC $\sim$ 1) is due to mapping between InterPro and Pfam (used as data resources in Mousefunc) and GO, but our similar performance trends holds not just in these potentially information-retrieval cases, but excluding them as well. Note that similar performance does not imply that the performances are in any way attributable to the same data features (or that the algorithm outputs are similar); it is these properties we ultimately explore.

One interesting deviation from MouseFunc results is that our algorithm performance set exhibits a somewhat diminished variance in mean performance across all tasks. While algorithms had quite similar performance in MouseFunc (mean AUROC standard deviation across algorithms <0.06 in each task) ours was somewhat lower (<0.02). While this does not quite rise to the level of generating a significant change in the distribution (KS test $P > 0.15$, comparing within each task), we suggest that the effect observed is still potentially real and likely because we used the network data as parsed and aggregated by the GeneMANIA team as input to our algorithms. To investigate this further, we performed a series of follow-up experiments related to the parsing of our data, focusing on logistic regression (as one of the most popular and general methods, and the highest performing outside of GeneMANIA in most of our analysis).

We first repeated our benchmarking using the raw data as provided to Mousefunc competitors rather than that parsed by the GeneMANIA team. In most data types, we obtained extremely similar performances regardless of representation, with average AUROCs differing by $\sim$0.01 in each data type (but in no fixed direction). Another possibility we wished to consider is that class imbalance might be driving algorithms toward more similar findings as a form of consistent bias. To examine this, we altered our implementation of logistic regression to one which sampled equally balanced training data to construct an ensemble classifier for its genome-wide prediction, as described in the methods. This algorithm did not perform significantly differently from the previous implementations (KS test $P > 0.1$, mean AUROC $\sim$ 0.80), suggesting this factor is also not critical. A more prominent source of performance variation is simply the weighting of the underlying data resources, which may have varied from team to team. Using the pre-existing feature data independent networks gave substantial variation, with average ROCs for each data type varying by 0.07 (mean AUROC standard deviation) using GeneMANIA alone. Knowing how to weight these networks when aggregating is a potentially critical step which may drive results, as has been previously discussed (Pavlidis *et al.*, 2002). To account for this, in the remainder of our analyses, we consider individual network performances separately, as well as explicitly comparing some standard feature selection and aggregation strategies.

3.2 High average performance generalizes to *S.cerevisiae* data

Having implemented and benchmarked our set of algorithms, we move to benchmarking our data resources.

We use data resources for *S.cerevisiae* because gold-standard resources are best represented there. The GO exhibits annotations over a much greater fraction of the yeast genome (Gillis and Pavlidis, 2013a,b), protein–protein interaction data are much more evenly and highly represented, and so forth, diminishing the degree to which artifacts can generate unrealistically high performance.

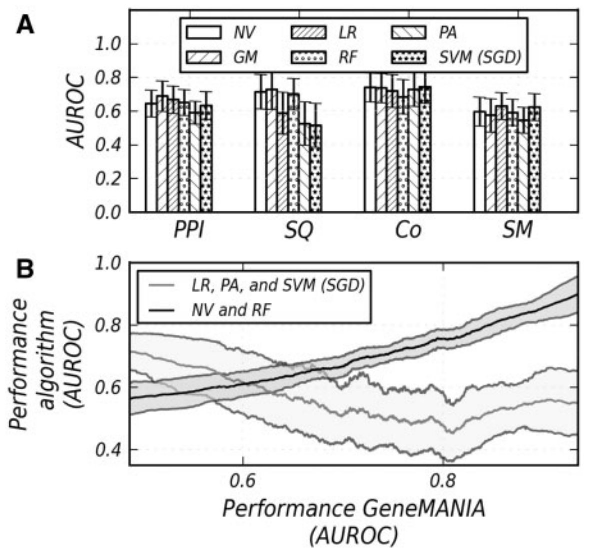

Fig. 2. Systematic performance analysis of gene function prediction on data from *S.cerevisiae.* (**A**) Comparison of algorithm performance on four different data resources: protein–protein interaction (PPI), sequence similarity (SQ), co-expression (Co) and semantic similarity (SM). The six algorithms used are: neighbor voting (NV), GeneMANIA (GM), logistic regression (LR), random forests (RF), passive aggressive (PA) and support vector machine with stochastic gradient descent solver [SVM (SGD)]. The algorithms all perform similarly except on the sequence similarity data, which shows the highest variance. (**B**) The average performance AUROCs of the top 2 performing algorithms (NV, RF—black line) and the bottom 3 performing algorithms [LR, PA and SVM (SGD)—gray line] against the performance of GeneMANIA on the sequence similarity data

Most algorithms performed quite well in each data resource (Fig. 2A), with co-expression showing the highest average algorithm performance (mean AUROC = 0.72) with substantial variability across GO groups for all algorithms (std AUROC = 0.097 in co-expression data). The lowest performing data were the semantic similarity data which represents shared pathways as assessed by the Jaccard index. This occurs principally because pathway data have comparatively low coverage. Across all GO groups, the semantic similarity data had an average AUROC of 0.59, but this is the only dataset in which many GO groups had genes which were not represented in the data. Indeed, in every other dataset, at least half of genes in every GO group were represented in the network data, whereas 965 GO groups (about 2/3) in the semantic data failed this criterion. Excluding those GO groups, the semantic data had a performance of 0.68. In other words, where there was data about what pathways a gene belongs to, genes could be classified into the pathway comparably to other resources. Sequence similarity showed the most variable performance across algorithms (Fig. 2B), with feature based algorithms showing quite negative correlations ($r \sim -0.41$) with the best performing algorithm (GeneMANIA), compared to network based and ensemble algorithms high positive average correlation among ($r \sim 0.91$). To measure the effect of a possible representation bias where we do make predictions, we assessed the node degree of networks for their predictive power across all GO groups. If it is merely the presence of features (rather than their exact nature) which drives performance, then node degree will be a powerful predictor. This is an example of the 'rich get richer' principle, meaning that genes which have many (non-zero) features are likely better than average candidates to have many functions due to overlapping selection biases. As our co-expression network exhibits pairwise interactions between every gene, this effect should not be substantial there. Indeed, co-expression node degree, calculated for each gene as the summation of its pairwise co-expression values, does not substantially predict function across all algorithms and GO groups (mean AUROC 0.50). Sequence similarity exhibits a mild bias (mean AUROC 0.53), as does semantic similarity (mean AUROC 0.54), while protein interaction data exhibits the largest bias (mean AUROC 0.57). In other words, simply knowing that a gene has been popular in PPI data provides a modest amount of predictive performance. This is unsurprising as the PPI data are potentially subject to selection biases in aggregation (reflecting the summated preferences of many individual experimentalists, particularly in bait-prey proteomics studies). The lack of coverage in semantic data hurts its performance, but does not appear to be a major contributor to its good performance when that does occur (as using coverage as a predictor of function is not too effective).

3.3 GO group learnability varies by data, not by algorithm

Overall performance of the algorithms generalized quite well from MouseFunc to the yeast data, with essentially all algorithms performing in the range of the gold standard, GeneMANIA. Indeed, if a GO group using one data type was well predicted by one method, it was well predicted by other methods in that same data type. For instance, correlations in GO group performance were very high between algorithms in co-expression $r \sim 0.86$, yet more modest in other data types ($r \sim 0.65$ in PPI data and $r \sim 0.56$ in semantic data), and with only sequence data being unusually low at $r \sim 0.18$. This decrease in correlation with respect to sequence similarity data was due to an elevated variance in algorithm performance, relative to other data types ($P = 0.012$, Wilcoxon test), with only algorithms specifically designed for network data or using ensembles performing well. Indeed, the top and bottom performing methods exhibit GO group performances (ROC scores) which were the most strongly negative correlated, $r \sim -0.51$, in pairwise comparisons. The methods comparisons exhibiting negative correlations (six pairs with $r < 0$) were precisely the pairs of algorithms in which mean performance differed most. In other words, algorithms differed as to which GO groups were easier to predict in a given dataset only when the algorithms had very different overall performances in that dataset; doing well overall means doing well on the same tasks.

This trend was at its most extreme in high performing GO groups. For each dataset, we calculated the correlation between the union of the top performing (top 5%) GO groups across all algorithms. In the subsets of GO groups, performance correlations between algorithms were very close to 1 (>0.99 in all cases). This does not mean that the actual algorithm outputs were identical. Indeed, calculating the ranks of positives for each GO group among the held-out fraction of data had an average correlation of 0.67 in PPI data, 0.51 in semantic similarity data, 0.37 in sequence similar data and 0.24 in co-expression data. By this, we mean that the actual ranks of predicted positives for a given GO group were highly correlated, but far from those seen in the similarities in performances. This variance in output as highly as to what each algorithm is 'getting right' provides a strong expectation that aggregation will serve to increase performance somewhat despite the minimal variance in actual performance among algorithms.

A similar comparison between data resources exhibits much lower overall agreement for a given algorithm across GO groups. That is, working well in one GO group for one data type does not ensure that method will work well in a new data type, with correlations across all comparisons average ($r \sim 0.20$) with standard deviation ($r \sim 0.14$).

In contrast with comparison across algorithms within a dataset, there was no trend that top and bottom performing resources showed unusually low correlations as to which GO groups they predicted; indeed, the opposite was true (albeit less strongly). The only two significantly negative correlations ($P < 0.01$ after multiple test correction) involved comparisons between sequence and semantic similarity performance and in both cases it was the two *top* performing algorithms exhibiting the negative correlation across GO groups; that is, doing well overall meant doing well on different tasks in this case. Despite this variability across data resources, there were general trends overall, with cellular compartment terms being more highly/better ranked by AUROC on average (Mann–Whitney test, $P < 1\text{E-}12$), biological process terms being worse ranked ($P < 1\text{E-}7$) and no relative trend for molecular function.

As might be expected from the greater variability across data resources as compared to algorithms, there was very little concurrence as to how different data yielded high performance (i.e. which positive guesses were right). When calculated for each GO group, the correlation between the ranks of held out true positives for a given algorithm rose to a maximum of only 0.15 for logistic regression. This suggests profound differences in the information available in different data resources, a property we seek to exploit in aggregation.

3.4 Aggregating methods allows for modest performance improvements

One of the most robust findings in machine learning is that aggregation improves performance. While, in some of the critical assessment analyses, a common data resource is being used, but in most the distinction between data and algorithm aggregation is lost. In our analysis, performing algorithm aggregation by averaging output

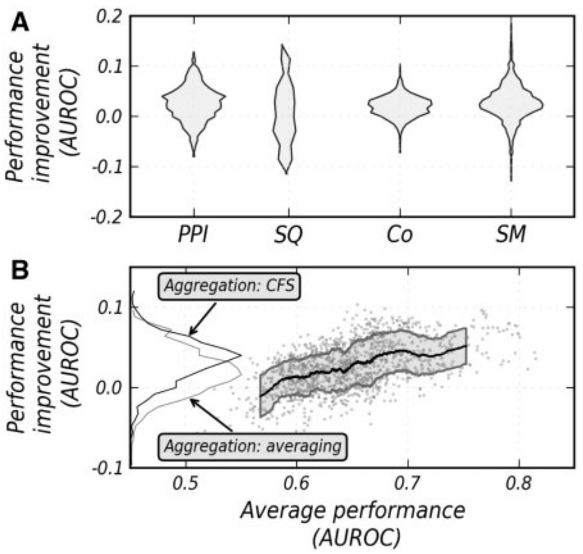

Fig. 3. Algorithm aggregation improves gene function prediction. (**A**) Violin plots of the distribution of the average performance of the six algorithms on the four data types: protein–protein interactions (PPI), sequence similarity (SQ), co-expression (Co) and semantic similarity (SM). (**B**) The average performance AUROCs plotted against the increment in performance (*y*-axis) averaged across the four data types. The average performance improvement is shown in the light gray marginal distribution along the *y*-axis. The same result using CFS is shown in dark gray, yielding a modest improvement in performance

probability scores increases prediction performance modestly in all data resources, shifting AUROCs upward 0.03 on average from an overall average of 0.65, averaging an 18% increase in performance (Fig. 3A). This is a very significant increase ($P < 1\text{E-}131$, Mann–Whitney test) as we are measuring across all GO groups, but it does suggest this strategy may be of limited value. Interestingly, the mean algorithm aggregate performance is slightly lower (5%) than the top performing single algorithm.

To investigate whether there were further potential gains through more careful aggregation, we perform CFS in our aggregation; in this case the improvement is slightly larger (AUROC 0.04). Interestingly, the degree to which aggregation improves performance is correlated ($r \sim 0.53$, $P < 1\text{E-}91$) with the original performances (Fig. 3B). This is potentially counterintuitive, as one might imagine that the higher the performance the harder it is to further increase it, and additionally, the more likely the individual algorithms are extracting the complete signal. Possibly higher performances would be more affected by these potential confounds, but in our analysis the tendency to have a signal in each algorithm is also evidence in favor of their being even more signal to extract through aggregation.

3.5 Aggregating data allows for large performance improvements

The pattern when aggregating by data is strikingly different from that based on algorithms (Fig. 4). All algorithms exhibit substantially greater and more consistent improvements from aggregating by data with a mean AUROC performance improvement of 0.128 across all algorithms, equivalent to an 88% per function improvement in performance above the null (AUROC = 0.5). In contrast to algorithm aggregation, this is substantially above (23%) even the maximum performing data source—and this understates gains somewhat, as we approach a performance ceiling in at least some functions. As can be seen in Figure 4A, even the algorithms benefiting least from data aggregation, obtain strong gains from doing so (AUROC improvement 0.075), while the largest improvement are tremendous (AUROC improvement of 0.17). Interestingly, CFS-based aggregation has a negative effect, decreasing average performance improvements to 0.109. This suggests determining how to weight datasets is potentially challenging, reinforcing our earlier observations both that this appeared to contribute to GeneMANIA's success within Mousefunc and that algorithms exhibit more similarity in their 'good' performance than datasets.

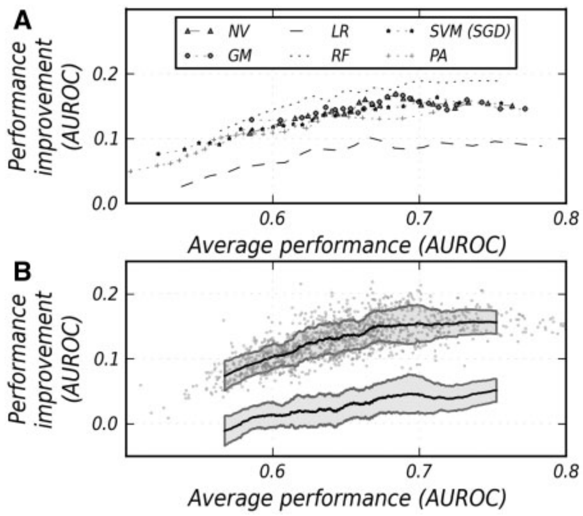

Fig. 4. Gene function prediction performance benefits strongly from data aggregation. (**A**) The average performance AUROCs plotted against the increment in performance (*y*-axis) for the six machine learning algorithms (individual curves) averaged across all data. (**B**) The data aggregation (black curve on top) compared to the algorithm aggregation (black curve on bottom). Data aggregation improves performance substantially, as shown by non-overlap of the two curves. The six algorithm abbreviations: neighbor voting (NV), GeneMANIA (GM), logistic regression (LR), random forests (RF), passive aggressive (PA) and support vector machine with a stochastic gradient descent solver [SVM (SGD)]. Each point represents the performance score for a GO term. All curves are smoothed with bins of size 100 with standard deviation shown

One factor we looked to consider in explaining the success of aggregation is node degree. Because node degree is a good prior for predicting a gene as possessing function in many datasets (as noted in Section 3.2), algorithms tend to exploit it (Gillis and Pavlidis, 2011a,b). However, we would not expect predicting the prior in this way to benefit from algorithm aggregation at all. While this was not a strong factor in our predictive performance, we do see a modest aggregate trend in which GO groups of high node degree benefit less from aggregation ($r \sim -0.36$, $P = 5.1\text{E-}41$). In some ways, this should be regarded as surprising, as we observed higher performing GO groups have, if anything, higher node degree and higher performing GO groups have better aggregation performance. But not all good performance is the same and when it is due to high node degree, aggregation does not add much in a dataset. This argument would not apply to aggregation across data, and indeed, little trend with node degree is observed there ($r \sim -0.13$). When node degree is the only contributor to performance, algorithms must be doing the same thing, and therefore cannot benefit from aggregation.

3.6 Metrics matter

We wished to briefly consider other factors which might affect the degree to which our own results might generalize. One such factor is our use of the AUROC metric. While this is probably the most popular metric, others exist, among them area under the precision–recall curve, max *F*-measure and precision at 20% recall (the latter two having been used in CAFA and Mousefunc, respectively). For each of these metrics we recalculated our results (available in the supplementary material). Because the null for precision–recall varies with GO group size, correlations of performances become more consistent (our AUROCs trends were not GO group size dependent). Interestingly, the performance trends for aggregation remain qualitatively the same regardless of metric, although the exact strength of the dependencies changed. Finally, if the major purpose of the metric is to assess which GO groups are 'easy' to predict, they reveal substantial variability. Using the GO groups listed by performance averaged across all metrics but one as the left-out metrics predicting the top half of performing GO groups gave only moderate AUROCs (mean = 0.86). This is not surprising—the different metrics measure different things—but does suggest that care must be taken in choosing metrics appropriately, as results may be sensitive to them. In specific, even sensibly conceived novel metrics may not generalize to conventional ones while incurring a penalty by providing developers with an additional degree of freedom in reporting results.

Because of the data intensive nature of gene function analyses, computation is often a limiting factor to comparative assessment (or in algorithm development), so it is natural to ask whether we could have assessed the algorithms or data based on only a subset of tasks. To determine this, we compared GO-wide performance with GO-slim performance for all algorithms and network data types. The degree to which GO-slim effectively estimated general trends was very high. Mean performances for any algorithm varied only slightly if averaged over all of GO or just GO slim, with the change in mean performance never exceeding an AUROC of 0.01. Thus, while the hierarchy as a whole may be useful at various other stages in algorithm implementation, using GO-slim is probably a reasonable approximation for performance assessment between algorithms.

4 Discussion

There is a simple and consistent trend in our results: even algorithms of wholly distinct conception and design give quite similar results (in all ways) on the same data. This may seem an obvious finding—one cannot get blood from a stone—but it is, in fact, contrary to much of the published literature.

Summarizing our points of departure from the previous literature, the most similar previous analysis to ours concluded that 'it's the machine that matters', even where similarly using default implementations (Wang and Marcotte, 2010). Similarly, it is the consensus that algorithm aggregation alone can provide substantial benefits on the closely related task of network inference (Marbach *et al.*, 2012), which would normally suggest that algorithms vary enough to benefit from aggregation. Supported by these general findings, more focused assessments have also concluded that simply aggregating algorithms on the same data offers enormous benefits, again, on tasks closely related to but not identical to function prediction, e.g. mutation prioritization (ManChon *et al.*, 2014).

While we draw superficially contrary conclusion, our results are actually surprisingly consistent with these claims. Within the range of performance of our algorithms on a given dataset, aggregation has a very large influence. Thus, in any competition in which data is held constant, we too, would suggest aggregation as a potentially useful strategy. It is only by examining variation across data resources that we see how comparatively modest these benefits are.

In support of this finding, in critical assessments where developers may alter their data choice, high-performing methods have tended to use large and diverse datasets, as in the top-performing CAFA method, where data integration was one of the chief areas of novelty (Cozzetto *et al.*, 2013). This suggests that comparative assessment of data (or even its integration) may be more fruitful than the current algorithmic focus. In fact, it would then be important to hold methods somewhat constant. Fortunately, our analysis suggests this is rather easy and that even where methods differ in basic design and conception, they are quite similar in performance (relative to variation caused by data). Because of this, downstream results can be assumed to be the property of the data being studied, rather than the inference method being used to study it. While it is true that good experimental design can ensure over-training does not occur, if a custom method is necessary to get a dataset to work or the custom method works unusually well, we suggest there is a heavy onus to show it does not reflect accidental over-training in some way.

Of course, another reason for our performance trends is that all our algorithms performed well. If some performed badly, then the machine could be said to matter more. We believe there may be a subtle selection bias at play here which is difficult to evaluate formally. Our function inference task was more computationally challenging than what might be considered a typical assessment for a targeted function or multiple functions in a targeted dataset. While our algorithms are not customized, many methods we initially tried simply did not operate readily in the large data and prediction space we wished to assess. This may well have selected for general-purpose methods or even just higher-quality ones. Likewise, this may explain why our performance is close to carefully tailored methods in critical assessment; the task is so broad that feature selection provides few benefits. Our observation was that when algorithms worked at all, they worked well, but some failed catastrophically. It is possible the divide in data interpretation is between methods that basically work and those which basically fail—and we simply seen combinations of this. Certainly, this is supported by our observation in the sequence similarity data that what worked, worked the same, and what failed, failed the same.

An encouraging finding from our work is the generally positive relationship between the learnability of a function and the degree to which aggregation is of value. Aggregation is of no likely value for combinations of random data/methods (AUROC 0.5) and also of no value for perfect output (AUROC 1.0). Despite seeing good overall performance, apparently we are still not in the realm where performance ceilings are a limiting factor; the overall trend is still dominated by the addition of signal yielding better aggregate performance (rather than the exhaustion of additional signal diminishing aggregation's value). This, combined with the high level of improvement from data aggregation overall, suggests that we have not come close to maximizing performance gains attainable from additional data.

Acknowledgements

Sara Mostafavi and Quaid Morris generously provided the Matlab implementation of GeneMANIA.

Funding

J.G., W.V. and S.B. were supported by a grant from T. and V. Stanley. No funding source played any role in the design, in the collection, analysis and interpretation of data; in the writing of the manuscript; and in the decision to submit the manuscript for publication.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bousquet,O. and Elisseeff,A. (2002) Stability and generalization. *J. Mach. Learn. Res.*, **2**, 499–526.

Breiman,L. (2001) Random forests. *Mach. Learn.*, **45**, 5–32.

Brown,K.R. and Jurisica,I. (2005) Online predicted human interaction database. *Bioinformatics*, **21**, 2076–2082.

Chatr-Aryamontri,A. *et al.* (2013) The BioGRID interaction database: 2013 update. *Nucleic Acids Res.*, **41**, D816–D823.

Cherry,J.M. *et al.* (2012) Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic Acids Res.*, **40**, D700–D705.

Cozzetto,D. *et al.* (2013) Protein function prediction by massive integration of evolutionary analyses and multiple data sources. *BMC Bioinformatics*, **14** (Suppl. 3), S1.

Crammer,K. *et al.* (2006) Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, **7**, 551–585.

Domingos,P. (2012) A few useful things to know about machine learning. *Commun. ACM*, **55**, 78–87.

Eisenberg,D. *et al.* (2000) Protein function in the post-genomic era. *Nature*, **405**, 823–826.

Enright,A.J. *et al.* (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, **30**, 1575–1584.

Fan,R.E. *et al.* (2008) LIBLINEAR: a library for large linear classification. *J. Mach. Learn. Res.*, **9**, 1871–1874.

Finn,R.D. *et al.* (2006) Pfam: clans, web tools and services. *Nucleic Acids Res.*, **34**, D247–D251.

Gillis,J. and Pavlidis,P. (2011a) The impact of multifunctional genes on "guilt by association" analysis. *PLoS One*, **6**, e17258.

Gillis,J. and Pavlidis,P. (2011b) The role of indirect connections in gene networks in predicting function. *Bioinformatics*, **27**, 1860–1866.

Gillis,J. and Pavlidis,P. (2013a) Assessing identity, redundancy and confounds in Gene Ontology annotations over time. *Bioinformatics*, **29**, 476–482.

Gillis,J. and Pavlidis,P. (2013b) Characterizing the state of the art in the computational assignment of gene function: lessons from the first critical assessment of functional annotation (CAFA). *BMC Bioinformatics*, **14**(Suppl. 3), S15.

Hall,M.A. (1998) *Correlation-based Feature Selection for Machine Learning*. The University of Waikato, New Zealand.

Hamosh,A. *et al.* (2005) Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Res.*, **33**, D514–D517.

Hamp,T. *et al.* (2013) Homology-based inference sets the bar high for protein function prediction. *BMC Bioinformatics*, **14** (Suppl. 3), S7.

Hawkins,T. *et al.* (2009) PFP: automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. *Proteins*, **74**, 566–582.

ManChon,U. *et al.* (2014) Prediction and prioritization of rare oncogenic mutations in the cancer kinome using novel features and multiple classifiers. *PLoS Comput. Biol.*, **10**, e1003545.

Marbach,D. *et al.* (2012) Wisdom of crowds for robust gene network inference. *Nat. Methods*, **9**, 796–804.

Mistry,M. and Pavlidis,P. (2008) Gene Ontology term overlap as a measure of gene functional similarity. *BMC Bioinformatics*, **9**, 327.

Mostafavi,S. *et al.* (2008) GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biol.*, **9** (Suppl. 1), S4.

Mulder,N.J. *et al.* (2005) InterPro, progress and status in 2005. *Nucleic Acids Res.*, **33**, D201–D205.

O'Brien,K.P. *et al.* (2005) Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Res.*, **33**, D476–D480.

Ogata,H. *et al.* (1999) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **27**, 29–34.

Opitz,D. and Maclin,R. (1999) Popular ensemble methods: an empirical study. *J. Artif. Intell. Res.*, **11**, 169–198.

Pavlidis,P. *et al.* (2002) Learning gene functional classifications from multiple data types. *J. Comput. Biol.*, **9**, 401–411.

Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Pena-Castillo,L. *et al.* (2008) A critical assessment of *Mus musculus* gene function prediction using integrated genomic evidence. *Genome Biol.*, **9**(Suppl. 1), S2.

Radivojac,P. *et al.* (2013) A large-scale evaluation of computational protein function prediction. *Nat. Methods*, **10**, 221–227.

Siddiqui,A.S. *et al.* (2005) A mouse atlas of gene expression: large-scale digital gene-expression profiles from precisely defined developing C57BL/6J mouse tissues and cells. *Proc. Natl Acad. Sci. USA*, **102**, 18485–18490.

Smith,C.L. *et al.* (2005) The Mammalian Phenotype Ontology as a tool for annotating, analyzing and comparing phenotypic information. *Genome Biol.*, **6**, R7.

Su,A.I. *et al.* (2004) A gene atlas of the mouse and human protein-encoding transcriptomes. *Proc. Natl Acad. Sci. USA*, **101**, 6062–6067.

Tranchevent,L.C. *et al.* (2011) A guide to web tools to prioritize candidate genes. *Brief. Bioinform.*, **12**, 22–32.

Wang,P.I. and Marcotte,E.M. (2010) It's the machine that matters: predicting gene function and phenotype from protein networks. *J. Proteomics*, **73**, 2277–2289.

Zhang,W. *et al.* (2004) The functional landscape of mouse gene expression. *J. Biol.*, **3**, 21.