5-2014

Predicting Gene Ontology Annotations Based on Literature Co-Occurrence

Rosemary Steup
Trinity University, rsteup@trinity.edu

Predicting Gene Ontology Annotations Based on Literature Co-Occurrence

Rosemary Steup

Abstract

In recent years, the amount of digital data that we produce has increased exponentially. This flood of information, often referred to as “big data,” is creating both opportunities and challenges in all areas of life. In the domain of biology, technology has enabled us to sequence the genomes of humans and many other organisms, but we are far from understanding the biological roles played by all of these genes. The Gene Ontology seeks to address this problem by annotating genes to terms describing biological processes, molecular functions, and cellular components. However, the ontology’s manual curators cannot keep up with the rate at which information is being discovered and published. Hence, there is a need for computational methods that can rapidly process the biomedical literature and suggest new annotations for verification.

This study uses support vector machines to predict Gene Ontology annotations for *Saccharomyces cerevisiae* (yeast). I tested the usefulness of two types of literature features: co-occurrence of gene names in articles, and co-occurrence in abstracts of gene names with keywords taken from GO term definitions. My results demonstrate that support vector machines using literature co-occurrence data as features can predict GO annotations with high accuracy. In many cases where simple gene-gene co-occurrence does not work well, better results can be obtained using gene-keyword co-occurrence. I found that a very simple text mining strategy — identifying words that occur in only one GO term definition — was an effective way of choosing keywords. Although predictions based on gene-gene co-occurrence and those based on gene-keyword co-occurrence were highly correlated, there are terms for which one set of predictions was significantly more accurate than the other. I was able to combine the two sets of predictions effectively using a voting scheme in which gene-gene predictions were weighted at 70% and gene-keyword predictions at 30%.

Acknowledgments

I owe a big "thank you" to my thesis advisor, Dr. Hibbs, for generously sharing his expertise in computational biology and guiding me through the research process. I would also like to thank my parents, who never stop believing in me, as well as the patient friends who are probably almost as glad as I am that this thesis is done.

Table of Contents

1. Introduction

1.1. Greater Context

1.1.1. The Global Impact of “Big Data”

In recent years, as computers have become an increasingly indispensable part of daily life and more and more processes have been automated, the amount of data we produce has increased exponentially. In 2013, the amount of stored data in the world was estimated at 1200 exabytes, up from 300 exabytes in 2007 (Mayer-Schönberger 2013). This is a staggering number given that a single exabyte is equivalent to 10^{18} bytes, or 4000 times the amount of information contained in the US Library of Congress (Manyika 2011). Almost all of this information is digital; analog information, which made up around three quarters of stored global data in the year 2000, now accounts for less than two percent (Mayer-Schönberger 2013).

The term “big data” was coined to describe this flood of digital information. Big data represents tremendous potential for generating new insights, but it is too unwieldy to be analyzed with traditional methods like relational databases. The sheer volume of data being produced is one problem; another is its unstructured and heterogeneous nature. This has prompted a shift in the way we use data. The focus now is on knowledge discovery – finding patterns in the data. Vasant Dhar, describing the field of data science, characterizes it as an inversion of the database-query model: instead of searching a database for items that satisfy a particular pattern (the query), we are looking for patterns that can explain our data (Dhar 2013). This process of extracting insights from masses of data often involves machine learning.

Big data is producing changes in all areas of life, and its importance is widely recognized. More and more businesses are jumping on the big data bandwagon; in a 2012 survey of 600 global business leaders, three quarters of participants described their organizations as data-driven. In 2013, the US government allocated $200 million to improve America's infrastructure for managing and using data (Gobble 2013). Data are changing the face of science as well. In a 2007 talk to the Computer Science and Telecommunications Board of the National Research Council, Turing Award winner Jim Gray spoke about the advent of “data-intensive science”. New technologies are enabling scientists to collect data faster and more cheaply than ever, but it is increasingly difficult to analyze and share all these data. Gray called for new tools to support the systematic curation and publication of scientific data. He envisioned a future in which all scientific articles with their associated data are freely available online, and the text and data interoperate with each other (Hey 2009).

1.1.2. Data Mining in Biology

Nearly every natural and social science discipline now has a computational branch focused on simulating processes, and an informatics branch focused on collecting and analyzing experimental data. These new computational pursuits have greatly accelerated rates of knowledge discovery and increased the accuracy and sophistication of models and predictions. Biology was a latecomer to the field of computational science, lagging behind other disciplines such as chemistry, astronomy, economics, and political science. It is only in the past 10-15 years that biology has begun to generate enough data to make computational methods and research necessary. Thanks to recent technological advances, however, the rate of biological data generation is now far outpacing Moore's law – that is, the amount of data needing to be processed is increasing faster than the ability of computers to process it. This has created a need for the development and implementation efficient computational methods to solve biological problems.

1.1.3. Gene Ontology

The Human Genome Project and other large sequencing efforts have revealed the genetic DNA sequences that encode life, but much work remains to be done before we can take full advantage of this information. These DNA sequences, or genes, are similar to a "parts list" or "manifest" of a highly complex machine; unfortunately, we still lack knowledge of the purpose and function of the majority of these genes/parts. In order to develop new genetic-based diagnostic and therapeutic tools, we need to understand the functional roles and interactions of tens of thousands of genes.

The Gene Ontology (Ashburner 2000), or GO, represents an effort to consolidate such biological data so that researchers can find information quickly and efficiently. It provides a controlled vocabulary for talking about gene products and their biological roles in a species-independent way, and it has succeeded in uniting numerous databases dedicated to the genomes of individual organisms. The ontology is divided into three branches: biological processes, molecular functions, and cellular components. Within each branch are terms organized in a hierarchy, with the most general terms at the top and specific terms near the bottom. Where a relationship is known to exist between a gene product and an ontology term, the gene is annotated to that term. Annotations to a term implicitly annotate a gene to all parent terms as well.

Although the Gene Ontology is now accepting some computationally generated annotations (tagged with the evidence code IEA, inferred from electronic annotation), the majority of new annotations are assigned by human curators who manually sift through the biomedical literature to find evidence of gene-term relationships. This process of manual annotation, while still the most reliable way to assign annotations, is laborious, expensive, and too slow to accommodate the volume of new information being produced. Consequently, there is a need for computational methods that can assist curators by processing the literature to predict where genes should be annotated.

1.2. Objective

Reliable predictions reduce the curator's task to one of verification, speeding up the annotation process. This study harnesses the biomedical literature freely available online to make annotation predictions which can then be verified either experimentally (by biologists) or through a directed search of the literature (by curators). I tested the usefulness of two types of literature features for this task: co-occurrence of gene names in articles, and co-occurrence in abstracts of gene names with keywords taken from GO term definitions. Predictions were made using support vector machines (SVMs).

1.3. Related Work

1.3.1. Gene and Protein Function Prediction

Many studies have used machine learning for gene function prediction or the related task of protein function prediction. These studies typically use features like gene expression microarrays or protein-protein interaction data, often in combination. (For examples, Troyanskaya, Huttenhower). In Huttenhower (2009), experimental validation was used to prove the viability of machine learning predictions. Vinayagam (2004) and Lewis (2006) specifically used SVMs to predict GO annotations.

1.3.2. Use of Literature Co-Occurrence

Various studies have explored the usefulness of literature co-occurrence — e.g. Jensenn (2001), who created a co-citation network for human genes and confirmed that co-occurrence reflects meaningful biological relationships. Stapley (2000) used co-occurrence data to create a weighted graph that can help scientists visualize the relationships between genes. Gabow (2008) successfully used co-occurrence in conjunction with protein-protein interaction data in a graph-theoretic prediction method.

1.4. Novel Aspects of this Project

Few studies have focused on literature co-occurrence as a source of features for gene function prediction. Co-occurrence data is more often used as a basis for making graphs or networks. If it is used for prediction, it is usually supplementing another type of data. In this paper, I demonstrate the value of literature co-occurrence when used independently of other data sources.

My method also differs from existing approaches to computationally assisted ontology curation, which focus on finding relationships that are explicitly present in articles. (For an overview of assisted curation efforts, see Winnenburg 2008.) They use text mining methods to identify genes and terms in text and characterize described relationships; in effect, they are trying to create algorithms that can “read” the text and pick out salient information. My goal is different: I aim to infer relationships based on global patterns. My approach therefore has the potential to discover new relationships not present in the text.

2. Methods and Tools

2.1. Python

I used the Python programming language throughout this project. To access NCBI's Entrez system (see 2.2.2. below) I used the Biopython suite of tools (Cock 2009) available at www.biopython.org. For graphing, I used the matplotlib library.

2.2. Data Sources

2.2.1. Saccharomyces Genome Database

In order to demonstrate the effectiveness of my approach, I chose to focus on yeast (*Saccharomyces cerevisiae*) because it is a model organism with a relatively thorough set of GO annotations. I obtained a list of all annotated yeast genes from a copy of the Saccharomyces Genome Database (SGD) (Cherry 1997) gene association file downloaded from geneontology.org on March 9, 2013. From the same source, I downloaded the Yeast GOSlim, a file containing a subset of 155 Gene Ontology terms chosen to be broadly representative and relevant (covering the entire breadth of the hierarchy structure) and largely non-overlapping biologically (no term in the set is a parent of any other term in the set). Focusing on these 155 slim terms allowed me to make predictions about individual terms without the additional concern of possibly creating inconsistencies in the hierarchy (e.g. by annotating a gene to a term but not to its parent), while broadly covering all areas of yeast biology. From the slim file, I produced a “term file” for each slim term, containing the systematic names of all the yeast genes annotated to that term. (Systematic names for yeast genes conform to a standard established by the SGD.)

2.2.2. Entrez

The National Center for Biotechnology Information (NCBI) hosts a system called Entrez (Maglott 2010) which facilitates information retrieval from 38 different databases, including PubMed, a repository of citations for biomedical literature. I accessed this system using the Entrez module of Biopython. I first searched for each gene name in NCBI’s Gene database. Since some names can refer to different genes in different organisms, I specified *Saccharomyces cerevisiae* as the organism for my search and used the genes’ systematic names when possible. In case a search returned multiple results, the first result was used. I then used the eLink function of Entrez to retrieve a list of PubMed article IDs associated with each gene. In this way, articles were found for 6317 of 6381 yeast genes. The remaining 64 genes have not been specifically mentioned in the literature, and thus were omitted from the study.

2.3. Classification

Classification is one of the most studied problems in machine learning. The goal is to produce a classifier, or a function whose input is a vector of feature values and whose output is a single value representing a class. In order to get this classifier, there must first be a learner which is fed a set of training examples — that is, a set of observed inputs and their associated outputs. The learner attempts to find the function that best fits the training examples, and this function becomes the classifier. A wide variety of algorithms have been developed to accomplish this task, and which one is best depends on the application.

2.3.1. Support Vector Machines

The predictions in this study were made using support vector machines. (See Cortes 1995, Vapnik 1999.) The SVM is a classification technique that has been proven to do well with noisy data, making it ideal for biological applications (Lewis 2006). Using an SVM involves a training phase and a test phase. In the training phase, the SVM is given examples which are labeled as belonging to one of two classes. It projects these examples into a many-dimensional space and attempts to find a plane that separates the two classes of examples. The ideal plane is the one that separates the classes while being as far as possible from both. In the test phase, the SVM is given unlabeled examples and classifies them based on which side of the plane they fall on.

2.3.2. SVM Light

SVM Light (Joachims 1998) is an implementation of support vector machines written in C by Thorsten Joachims. I downloaded version 6.02 of this software from Joachims' website svmlight.joachims.org. For this project, SVM Light was used in classification mode with default settings.

SVM Light requires that its input files have a specific format, which is the same for both training and test files. Each line in the input file represents one example. The line consists of a target value (+1 for a positive example, -1 for a negative example) followed by pairs of numbers denoting features and their values. The feature-value pairs must be arranged in ascending order by feature number. Features with a value of 0 can be skipped. For this project, I created one input file per term, each containing one example per gene. The target value was 1 if the gene was annotated to the term in question (i.e. if it was found in the relevant term file), and -1 if it was not. The features varied; I conducted three trials with three different sets of features. (See 2.8.)

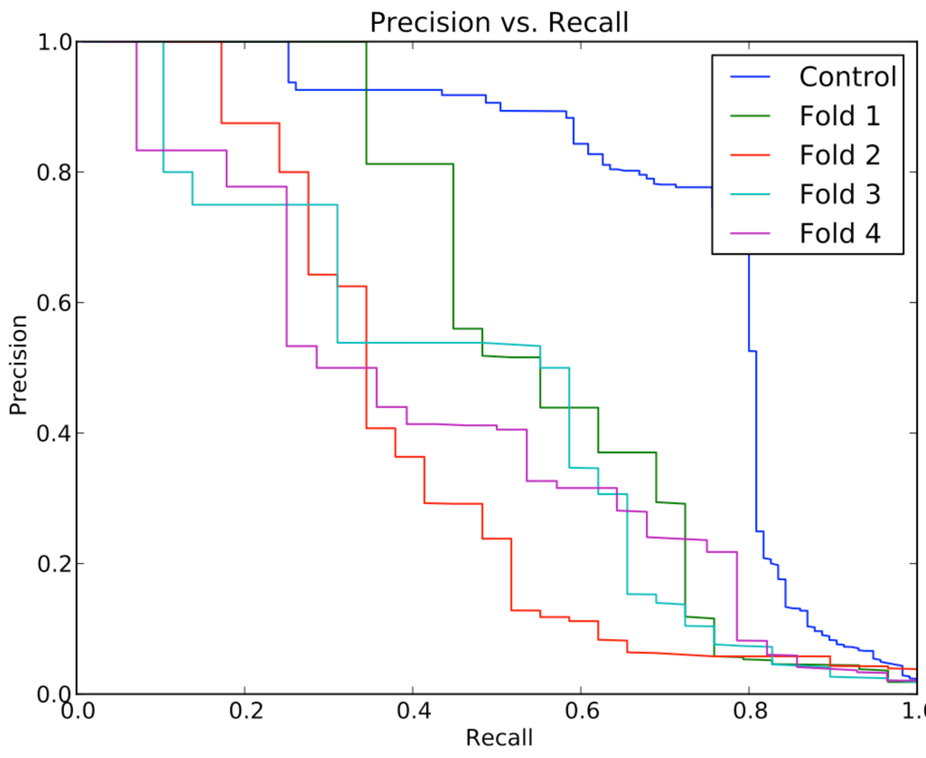
SVM Light comes with the executables svm_learn – the module responsible for training SVMs – and svm_classify. svm_learn takes an input file containing training examples and produces a model file. The model file, together with an input file containing test examples, is input into svm_classify, which outputs a prediction file. (For a diagram of this process, see Appendix, Figure 1.) The prediction files produced by SVM_classify mirror the example files that produced them. Each line consists simply of a floating point number representing the SVM's prediction for that example. A higher number represents a greater likelihood that the example is a positive one (in this case, that the gene is or should be annotated to the term in question); a lower number means the example more likely belongs to the "negative" class (the gene is not related to the term).

2.4. Bootstrap Aggregation

2.4.1. Preliminary Tests With Cross-Validation

To ensure generalizability, and not become "circular," a classifier must be tested on examples that were not part of its training set. This often means that to assess reliability of classifiers, some examples need to be held out during training. However, holding out examples reduces the information available to the learner; it is not seeing the whole picture, and therefore cannot produce the optimal classifier. This problem can be mitigated using cross-validation, in which multiple classifiers are each trained on part of the data and tested on the rest. Performance is then estimated by averaging the results from all of the classifiers. (For an illustration, see Appendix, Figure 2.)

In my earliest experiments, I used four-fold cross-validation. The example file was divided into four segments, each containing ¼ of the positive examples and ¼ of the negative examples. Each of these segments became an SVM test file, and a corresponding training file was created from the other three segments. (So each example appeared in exactly one test file.) An SVM was trained on each of the training files and then used to classify the examples in the corresponding test file. This cross-validation scheme prevents "circularity," in that predictions are only created for genes not used as training examples. However, as a control, I also trained and tested a "circular" SVM on all examples in the original file. When summary statistics were computed (see 2.6), they revealed significant discrepancies between folds and a large performance gap between the folds and the control. These results indicated that overfitting was a problem in my tests: the SVM was tailoring its models too closely to the training data, leading to highly accurate classification of the training examples but comparatively poor classification of new examples. To combat overfitting, I decided to move from four-fold cross-validation to a more sophisticated cross-validation technique: bootstrapping.



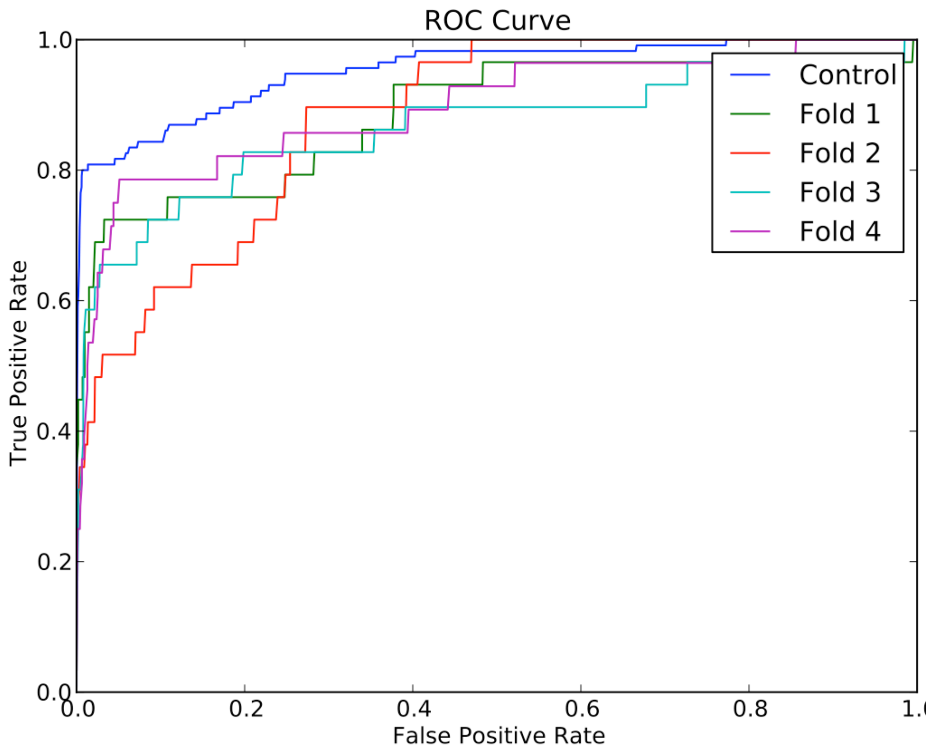


Figure 1: Precision-recall and ROC curves for GO:0008033. The excellent performance of the control as compared to the four test folds reveals a high degree of overfitting.

2.4.2. Bootstrapping

Unlike the fold-based process described above, bootstrapping involves sampling with replacement; each example can, and ideally does, appear in multiple test files. To create the training and test files, I randomly selected 70% of the examples to write to a training file and wrote the rest to a corresponding test file. This process was repeated fifty times for a total of 100 files per term. As before, the ratio of positive to negative examples in each training and test file was held constant. The results of each bootstrap were combined using “out of bag” aggregation, meaning that predicted classification values were taken only for test examples and were averaged when examples occurred in multiple test files. Similar to four-fold cross-validation, this process also prevents “circularity,” but more thoroughly samples the training space to reduce variation and overfitting.

2.4.3. Selecting Number of Bootstraps

To decide the appropriate number of bootstraps, I ran initial tests on several terms and produced graphs by plotting average AUC and AUPRC (described in 2.6.1) against the number of bootstraps. I looked for the number of bootstraps at which the lines leveled out, showing that the cumulative averages had become resistant to further change. Based on these tests, I made the conservative choice of fifty.

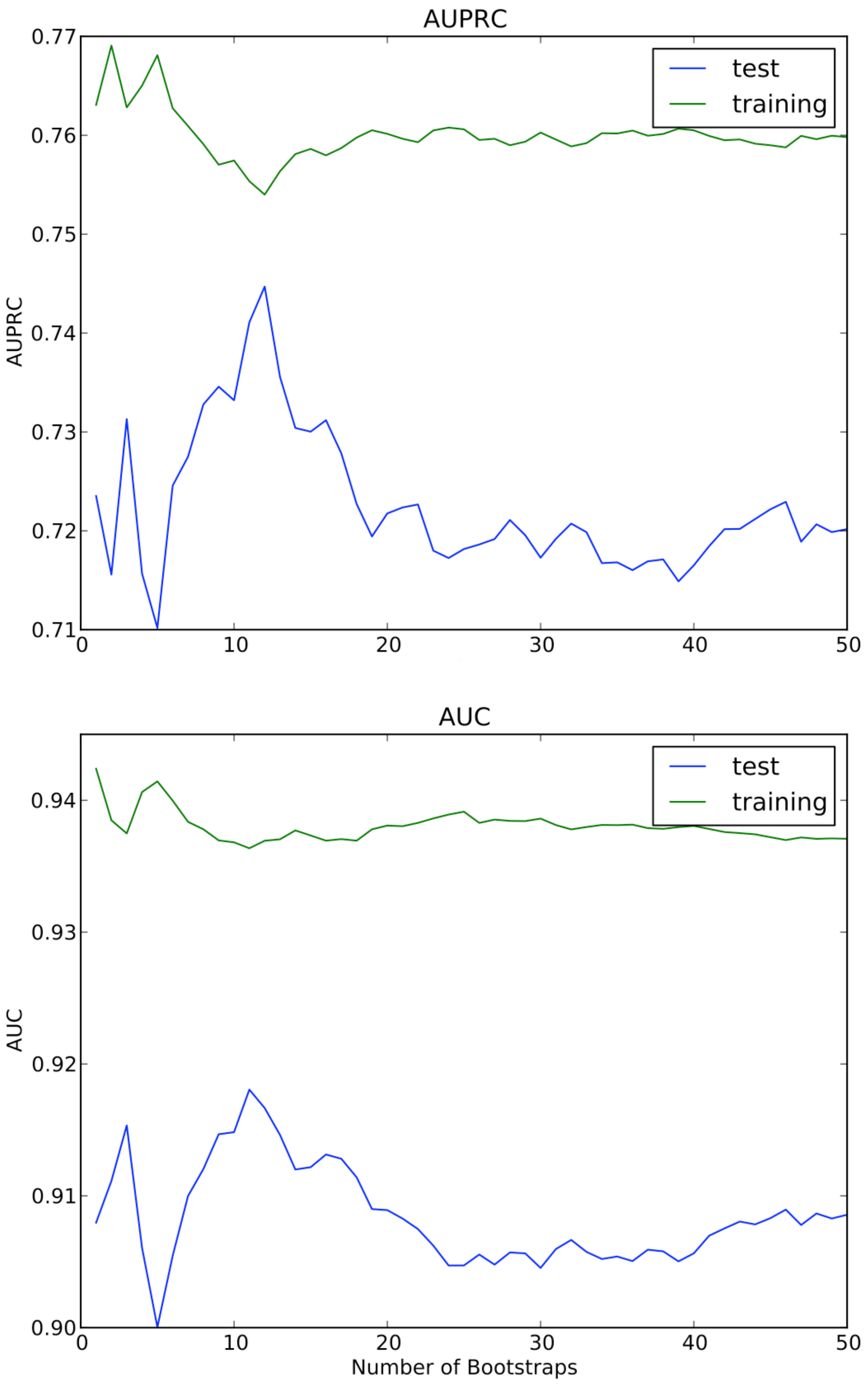


Figure 2: The cumulative average AUPRC and AUC for GO:0008380 level out as the number of bootstraps approaches 50.

2.5. Basic Procedure

For each term, one example file was created. That file was then used to create fifty training and fifty test files according to the procedure described above. The training files were input into the svm_learn module of SVM Light, each yielding a model file. These model files, with their corresponding test files, were then fed into the svm_classify module of SVM Light to produce fifty prediction files. Training files were also run through svm_classify, to provide a benchmark for measuring overfitting. My final predictions were obtained by averaging predictions from all the test files. To measure the accuracy of these predictions, six summary statistics were computed: area under the ROC curve (AUC), area under the precision-recall curve (AUPRC), percent improvement over baseline precision, precision at 1% recall, precision at 10% recall, and precision at 50% recall. Precision-recall and receiver operating characteristics (ROC) curves were also plotted for each term.

2.6. Summary Statistics

2.6.1. Measures Used

These statistics are best understood in terms of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). True positives are examples that were correctly classified as positive; false positives are examples that were incorrectly classified as positive; and so on. (For an illustration, see Appendix, Figure 5.) Precision is the fraction of all positive predictions that are really positive: TP/(TP + FP). Recall is the fraction of all positive examples that were correctly classified: TP/(TP + FN). A precision-recall curve plots recall on the x-axis and precision on the y-axis, and illustrates the trade-off between accuracy and inclusiveness. A ROC curve plots recall against the false positive rate (FPR), or the fraction of all negative examples that were incorrectly classified as positive: FP/(FP + TN). AUPRC is the area under the precision-recall curve; AUC is the area under the ROC curve. AUPRC and AUC are discussed in more detail in section 3.1.

2.6.2. Computing Summary Statistics

To obtain these statistics from the SVM prediction files, I sorted the predicted values in descending order and chose the highest value as the cutoff separating positive predictions from negative ones: any value greater than or equal to the cutoff was considered to be a positive prediction, while values less than the cutoff were considered to be negative predictions. I calculated precision, recall, and false positive rate at this cutoff and stored the resulting values. I then set the next-highest value as the cutoff, recalculated the statistics and stored the new values; and so on, until nothing was left below the cutoff (i.e. TN and FN were both 0).

2.7. Negative Control

To ensure that the SVM was identifying real, useful relationships in the data and not just finding patterns in noise, I created a negative control set, consisting of six fake term files containing random combinations of the gene names found in the actual yeast slim term files. Of these fake terms, two are small (fewer than 80 genes), 2 are medium (80-150 genes), and 2 are large (151-500 genes). These terms were tested along with the real ones. SVM performance for the randomly generated terms was as poor as expected: AUC and AUPRC were close to the values that would have been obtained by guessing at random. The failure of the SVM to produce accurate predictions for these fake terms indicates that the results achieved for the real terms are genuinely meaningful.

2.8. Features

2.8.1. Trial 1 - Gene-Gene Co-Occurrence Features

In my first trial, the SVM features were based on co-occurrence of genes in articles. To help with constructing the example files for the SVM, I first made a co-occurrence matrix in which each column represented a gene and each row represented a gene. At the intersection of Gene A and Gene B was the number of PubMed article IDs associated with both of those genes. In making the SVM Light example file, each row of the co-occurrence matrix was associated with an example and each column with a feature. For an illustrated example of this process, see Appendix, Figure 3.

2.8.2. Failure of Feature Selection

In this trial, the number of features (6317) was equal to the number of examples. In general, it is better to have more examples than features, because a high feature-to-example ratio can contribute to overfitting. With this in mind, I attempted to reduce the number of features in my input files through a process called feature selection. Feature selection posits that where there is a large number of features, some of them may be redundant or meaningless. By eliminating these features, which provide no useful information to the SVM, one can reduce overfitting without seriously lowering the quality of the SVM's predictions.

I chose to eliminate as features all genes which were not annotated to a given term. (This naturally results in a different number of features for each term, but the reduction is drastic in all cases; the largest term in the yeast slim has 478 genes annotated to it, and the smallest 22.) I tested this strategy with four terms of varying sizes -- two large, one medium, and one small -- which had had different levels of success in my initial tests. New example files were created and run through SVM Light, and summary statistics were calculated. The results of these tests showed that feature selection was not helpful; prediction quality suffered greatly and overfitting was not reduced. The features that I eliminated clearly were not redundant, indicating that some of the information utilized by the SVMs involves "indirect" literature connections through genes not annotated to the GO term in question.

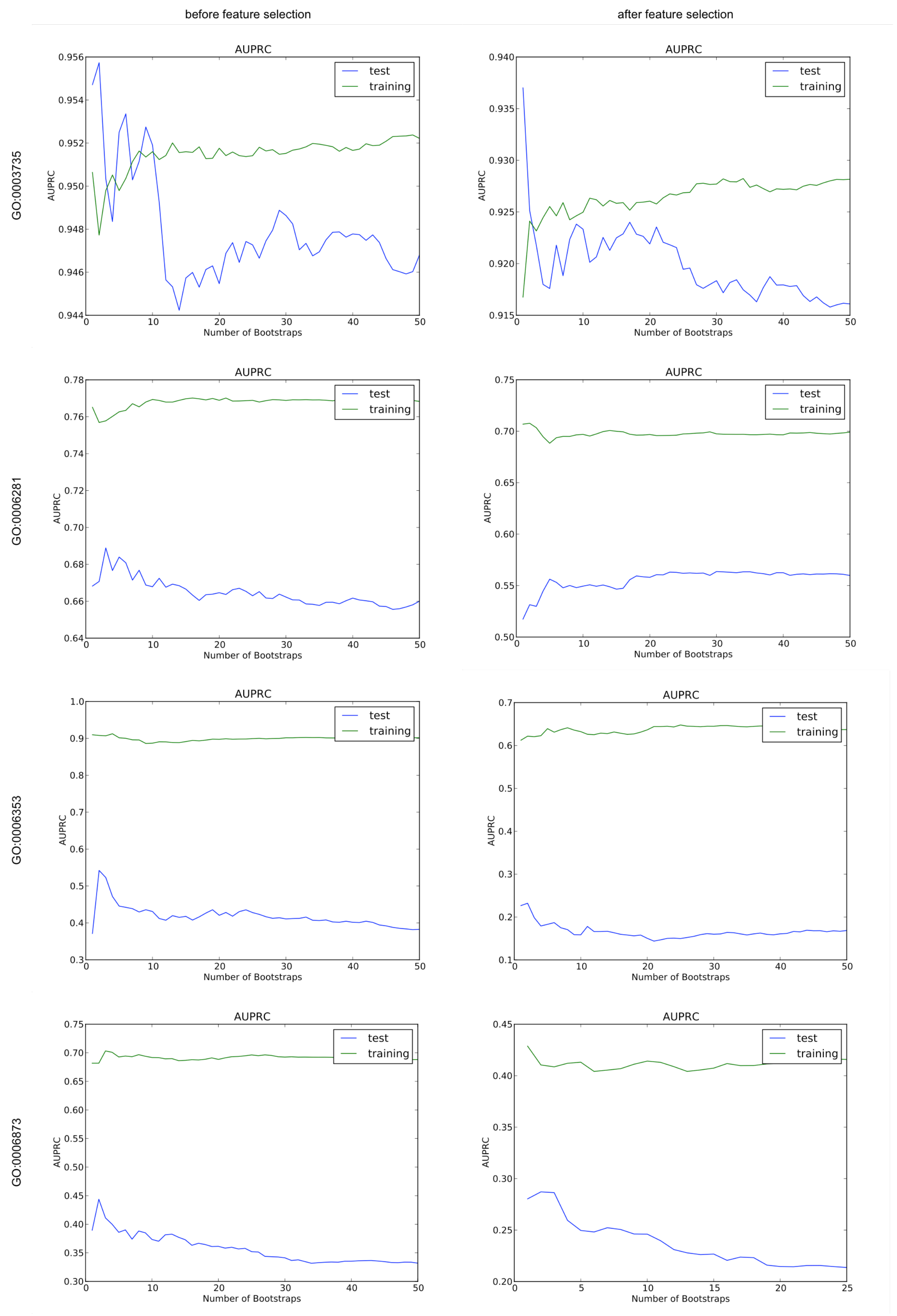


Figure 3: Cumulative average AUPRCs for four terms before and after feature selection. A drastic drop in performance is reflected in the values on the Y-axis.

2.8.3. Trial 2 - Gene-Keyword Co-Occurrence Features

For this second trial, SVM features were based on co-occurrence of genes and 629 keywords found in the definitions of yeast slim terms. I established a list of keywords by identifying words that occur in only one term definition. I did not refine this list by eliminating variants of words (such as different tenses of a verb) or unhelpful words such as “there” and “hence”. I then used Biopython.Entrez to download an abstract for each article ID retrieved in Section 2.2.2. In the SVM example files, each keyword served as a feature. The value of that feature, for each gene, was the number of abstracts associated with that gene that contained the keyword. For more on the creation of example files in Trial 2, see Appendix, Figure 4.

2.8.4. Trial 3 - Combined Features

Having observed that some terms performed better in Trial 1 while others did better in Trial 2 (details below), I combined both sets of features (genes and keywords) in the hope that they would complement each other and produce better overall results. This resulted in a total of 6946 features.

3. Results

3.1. Measuring Performance

3.1.1. AUPRC

A precision-recall curve, as previously mentioned, plots recall on the x axis and precision on the y axis. In making the precision-recall graphs, I actually graphed the convex hull of the data points. The area under this curve (AUPRC) is therefore equivalent to the average precision of all points in the curve. (After convex hulling, precision only changes when recall does – each time a positive example is correctly classified – and recall always changes by the same amount: 1/P where P is the number of positive examples.) (See Appendix, Figure 6, for an example of a precision-recall curve.)

3.1.2. Improvement Over Baseline Precision

In addition to comparing the AUPRC for different terms, I also compared the AUPRC of each term to the baseline precision of that term. Baseline precision is the average precision that we would expect to obtain if we were to guess the class of each example at random. It is equal to the number of genes annotated to the term divided by the total number of genes being classified. The percent difference between AUPRC and baseline precision is a useful indicator of how successful the SVM was at making predictions for individual terms. It can also serve as a basis for comparing terms, since it is essentially a normalized version of AUPRC.

3.1.3. Precision at n% Recall

AUPRC can sometimes be misleading because it does not tell us the shape of the precision-recall curve. A curve that has high precision at low recall, but then drops off steeply, may have the same AUPRC as a curve that starts at a lower precision and slopes downward more gradually. However, the first curve is more desirable from our perspective because it is the low-recall, high-precision area from which we hope to take our predictions. For this reason, I supplemented the AUPRC for each term by calculating the precision at 1%, 10%, and 50% recall

3.1.4. Receiver Operating Characteristic (ROC) Curve and AUC

The area under the ROC curve is referred to as AUC. As previously mentioned, a ROC curve plots the false positive rate – the fraction of negative examples which are wrongly classified as positive – on the x axis, and true positive rate, or recall – the fraction of positive examples which are correctly classified as positive – on the y axis. Both of these quantities increase as the cutoff for positive predictions is lowered, eventually reaching a value of 1. Ideally, recall increases much faster than the false positive rate, resulting in a curve that hugs the upper left edge of the graph. Random guessing would result in a diagonal line reaching straight from (0,0) to (1,1) and an AUC of .5. (See Appendix, Figure 7, for an example of a ROC curve.)

3.1.5. Additional Considerations for Predicting GO Annotations

In a classic classification problem, the SVM (or other supervised learning algorithm) is trained and tested on examples whose class is definitively known. The model it produces, once it has been vetted for accuracy, is then used to classify brandnew examples. For instance, a well-known application of machine learning is the detection of spam emails. In this case, the training set is a collection of emails that have been manually labeled as spam or not spam. A classifier is trained and tested on these examples and then put to use sorting new emails that are sent to email users' inboxes. New emails are constantly being generated, and the number of possible examples is unbounded.

Predicting GO annotations is different in two important ways. First, since each example is a gene, the set of examples is more or less fixed. So the examples we train and test on are the same examples whose class we want to predict. This leads into the second difference: Our training set is not entirely accurate or complete. If it were, there would be no point to projects like this one; we would already have a perfect set of annotations. Instead, we know that the Gene Ontology is missing some annotations; e.g. Gene x is related to Term A but is not annotated to it because the relationship has not been discovered, or because it has yet to be codified by the ontology curators. When classifying genes with respect to Term A, Gene x is considered a negative example and, if classified as positive by our model, will be counted as a false positive. However, this "false" prediction is not a failure on the part of the model. Instead, it means the model has done exactly what it was supposed to do: inferred a new relationship based on patterns it found in the training data. It is important to keep this in mind when considering measures like precision and FPR, which penalize models for generating false positives.

3.2. Initial Results

3.2.1. Trial 1 – Gene-Gene Co-Occurrence Features

The results of my first trial showed that SVMs trained on gene-gene co-occurrence data can predict GO annotations with a high level of accuracy. For all terms, AUPRC was significantly higher than the baseline precision, with the smallest improvement being 263.8%. (For comparison, the biggest improvement over baseline among the six fake terms was 57.5%.) The median improvement was 1976%. AUCs were also well above baseline, with a median of .87. The AUC for term GO:0000902, at .53, was comparable to the AUCs of the six randomly generated terms; but aside from this one abnormally low result, the lowest AUC was .65. The median precision at 1% recall was 1 -- indicating perfect accuracy -- and the median precision at 10% recall was .9.

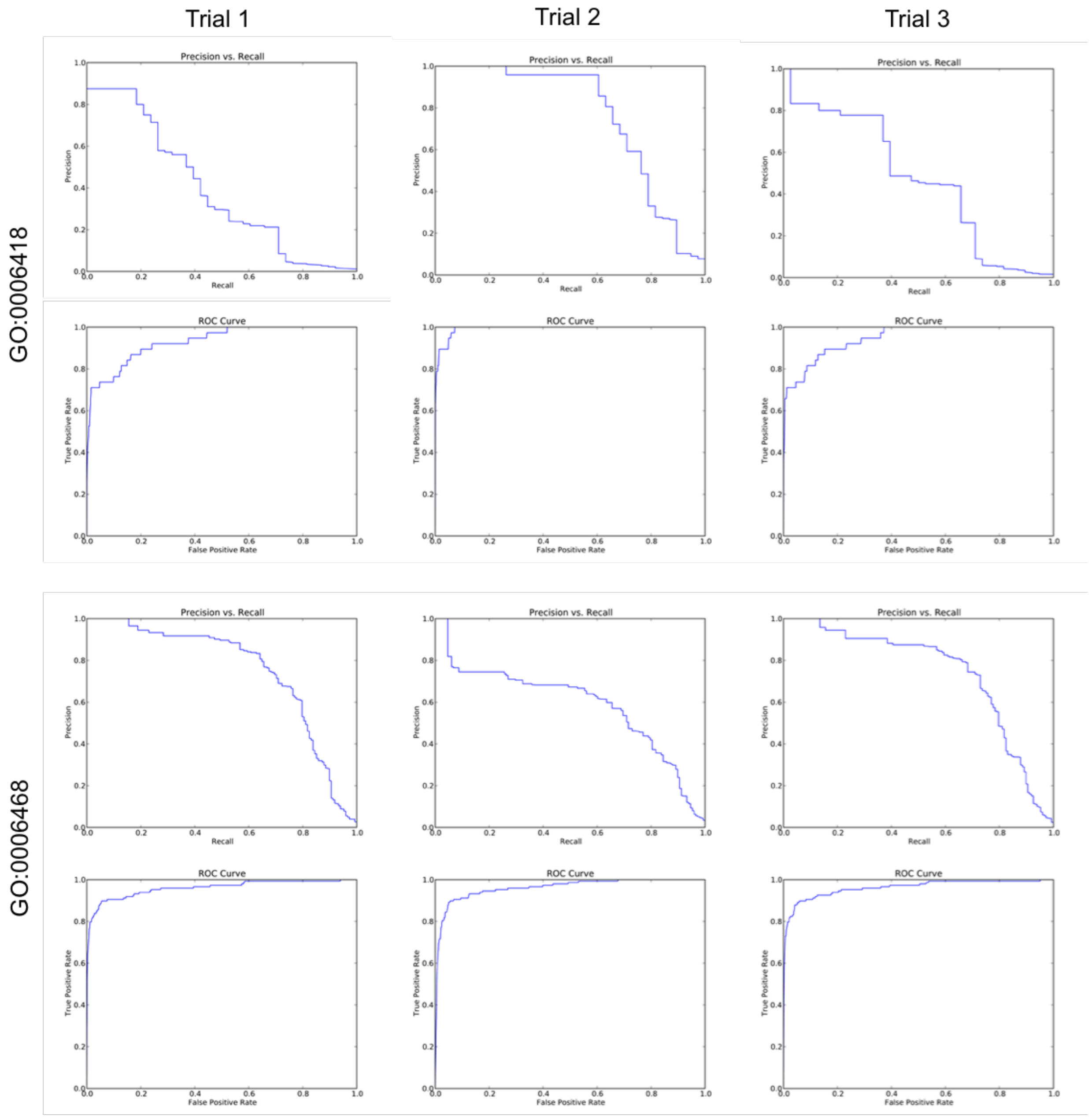


Figure 4: Precision-Recall and ROC Curves for two terms, GO:0006418 and GO:0006468, across all three trials

3.2.2. Trial 2 – Gene-Keyword Co-Occurrence Features

SVMs trained on gene-keyword co-occurrence data also performed well. In fact, gene-keyword co-occurrence narrowly beat out gene-gene co-occurrence, yielding a median AUC of .89. Median improvement over baseline precision was 2006%. More importantly, however, improvement was not consistent across all terms. Some terms for which SVM performance was mediocre or even very poor in the first trial fared much better in Trial 2. One such term is GO:0006418 (pictured above) whose Trial 2 AUPRC of .76 was an 88% improvement over its Trial 1 result of .40. Comparing the two precision-recall curves, we can see that precision starts higher in the second trial and drops off later, remaining at .9 up through 60% recall. Altogether, the Trial 2 AUPRC was at least 5% higher than the Trial 1 AUPRC for 79 out of 155 terms.

Conversely, there were some terms who fared significantly worse in Trial 2. The AUPRC for term GO:0006468 (also pictured above) dropped from .75 in the first trial to .59 in the second. This difference is clearly reflected in the shape of its precision-recall curve: in the Trial 1 curve, precision stays high (above .89) through 50% recall; meanwhile, the Trial 2 curve drops off abruptly at about 5% recall, and by 50% recall, precision is down to .67. Overall, there were 43 terms for which AUPRC decreased by 5% or more from Trial 1 to Trial 2.The graphs below show the distribution of Trial 1 and 2 AUCs and AUPRCs for all terms. Dots close to the diagonal represent terms that performed about the same in both trials. Dots above the line are terms that did better in Trial 2; dots below the line are terms that did better in Trial 1. The points are strongly correlated, suggesting that similar information is represented in both Trials; however, there are also points on both sides of the line, indicating that no one set of SVM features was better for all terms. In some cases, gene-keyword co-occurrence produces more accurate predictions; in others, gene-gene co-occurrence is superior.

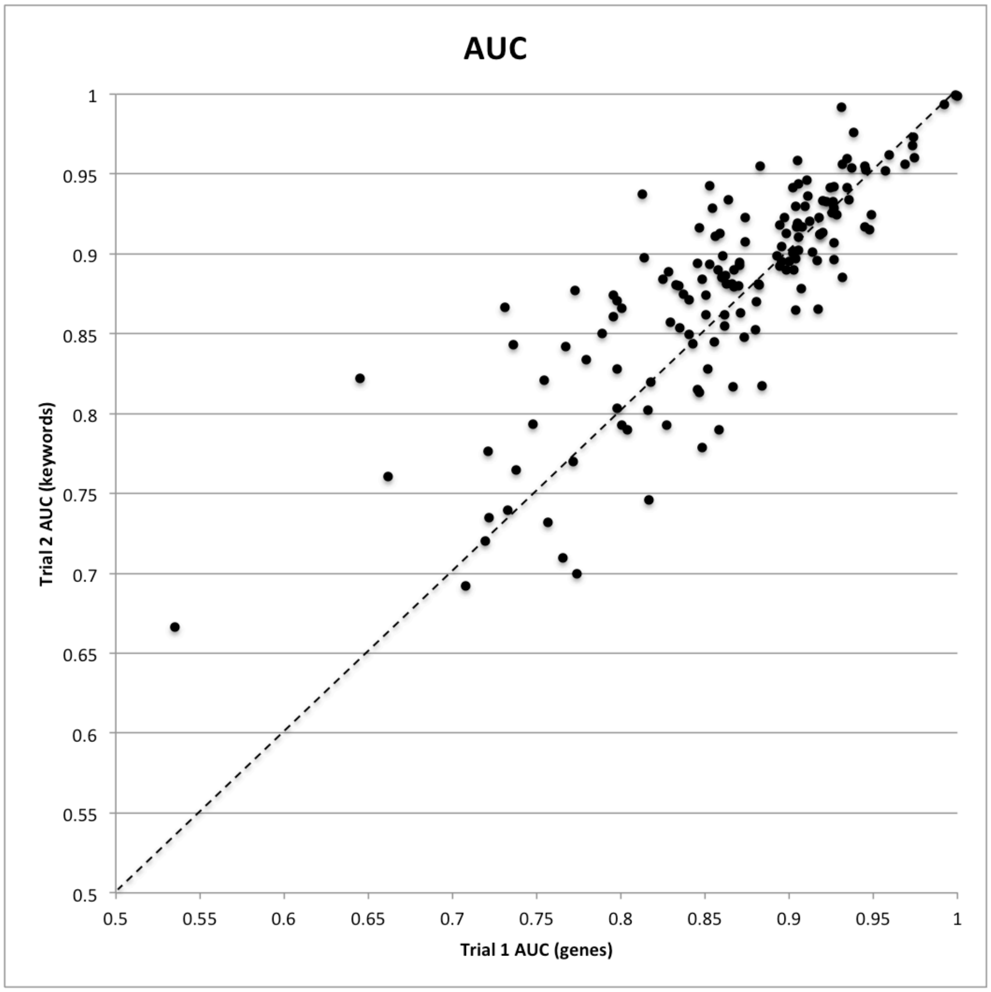


Figure 5: Distribution of AUCs in Trials 1 and 2

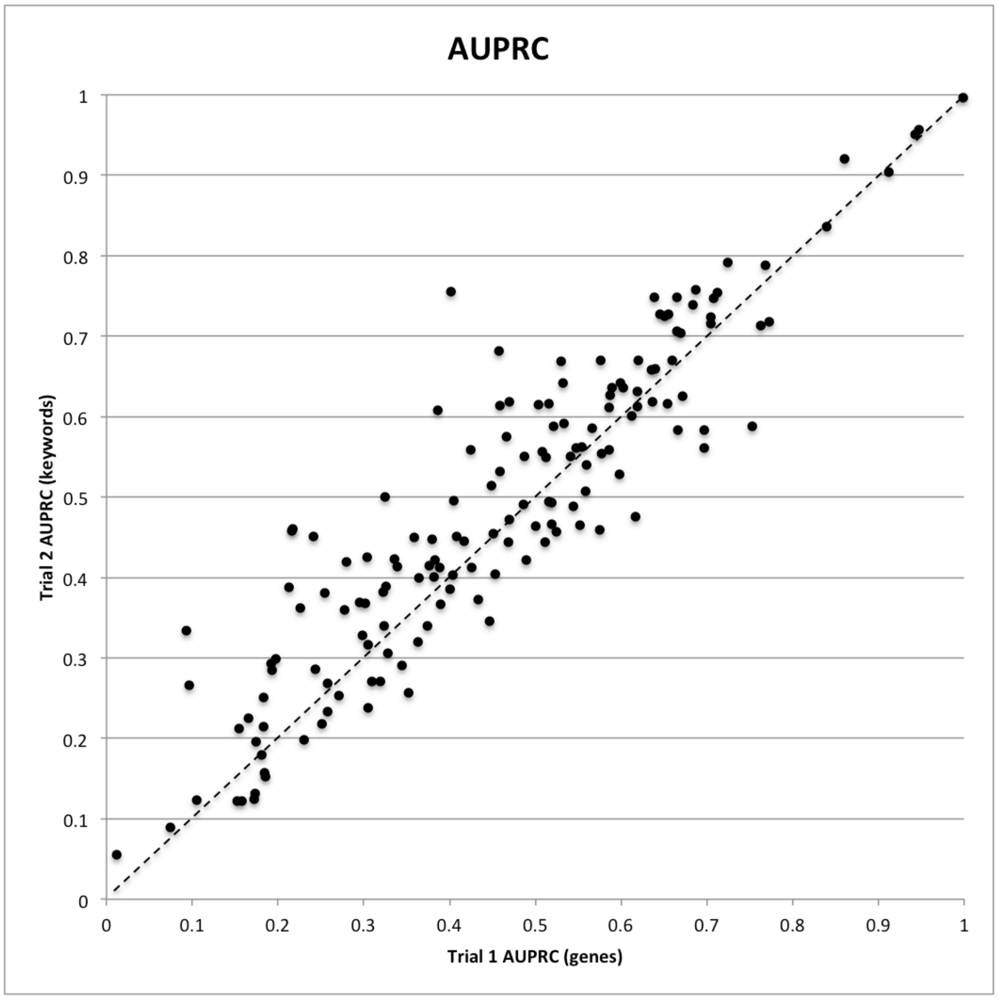


Figure 6: Distribution of AUPRCs in Trials 1 and 2

3.2.3. Trial 3 – Combined Features

I hoped that by combining the feature sets from the first two trials, I could achieve better results than I got using either set on its own. This was not the case, however. On average, the SVMs trained on the combined feature set were about as successful as those trained on gene-keyword co-occurrence. Median AUC was .89, and median improvement over baseline precision was 2065%. But on a term-by-term basis, the Trial 3 results were unimpressive.

For the vast majority of terms (145 out of 155), the Trial 3 AUPRC was intermediate, in that there was an improvement over the worse of the Trial 1 and 2 AUPRCs; however, it represented a decrease from the better of the Trial 1 and 2 AUPRCs for 119 terms. There were only 5 terms for which the Trial 3 AUPRC improved on the better of the first two results by 5% or more. The same pattern holds for AUC.

3.3. Voting Scheme

Ultimately, I found that the predictions from the first two trials could be effectively combined using a voting scheme in which most of the weight is given to the Trial 1 (gene-gene co-occurrence) predictions. The predictions produced by this voting scheme were superior overall to those of the three individual trials.

3.3.1. Determining Optimal Weights

To determine the optimal weight, I took weighted averages of the two sets of predictions according to the equation: $p_v = \alpha p_w + (1 - \alpha)p_g$, where p_w is the Trial 2 prediction, p_g is the Trial 1 prediction, and p_v is the result of voting. I calculated AUPRCs and AUCs for all terms at eleven different values of α: 0, 0.1, 0.2 … 1.0. With the resulting values, I created one graph per term illustrating the effect of weighting on AUPRC and AUC. I also plotted all 155 terms in two composite graphs (one for AUPRC and one for AUC) to see whether weighting affected all terms in a consistent way. It was difficult to spot trends in these graphs because the variation among terms was so great (with AUPRCs ranging from less than .1 to nearly 1.0), so I created two normalized versions. (See below.) The first of these shows how much AUPRC/AUC differs from the mean at each weight. The second shows difference from the mean divided by the standard deviation; this makes the shape of the curve clearer, at the expense of exaggerating small changes.

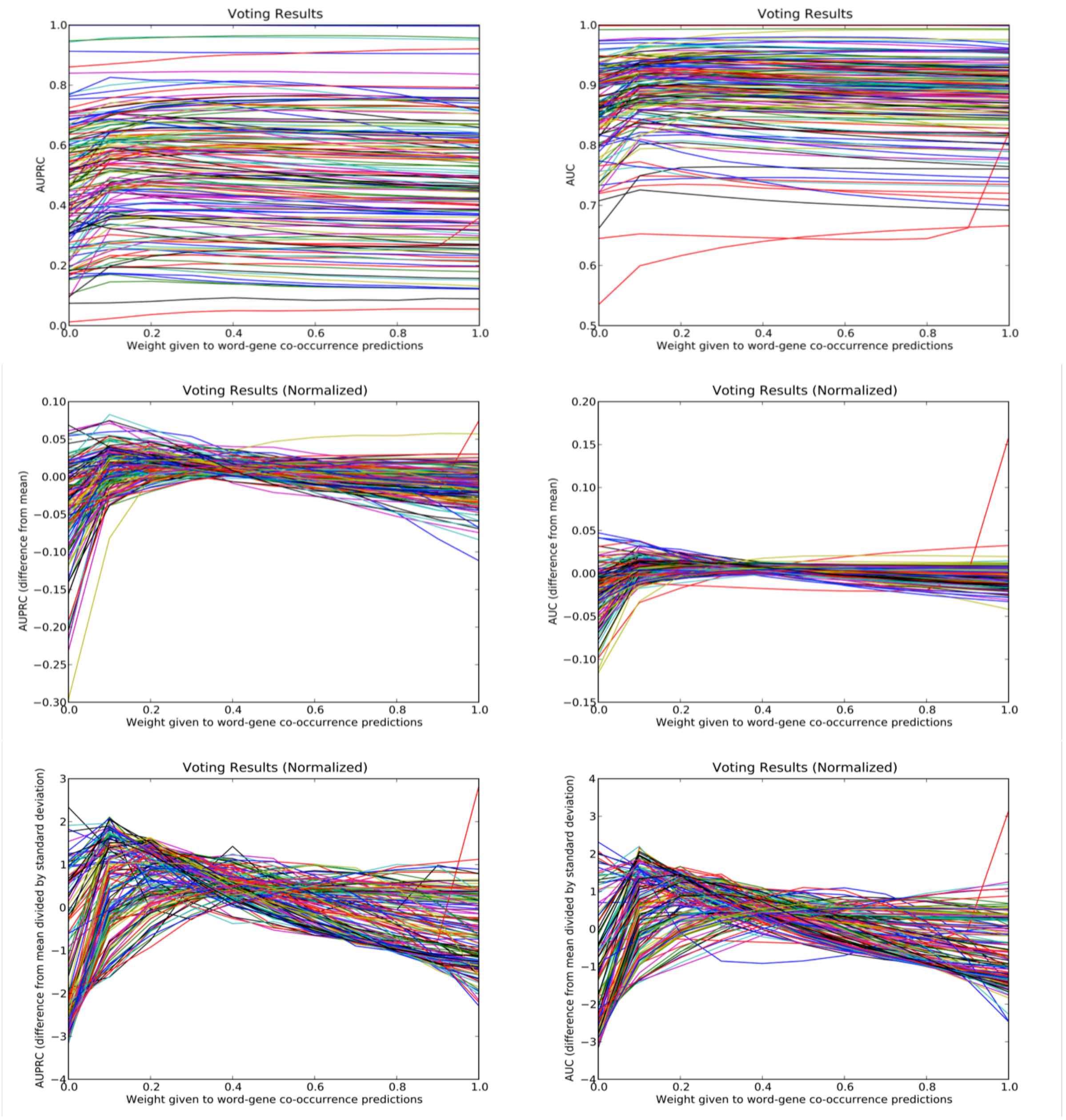


Figure 7: Effect of different weighting schemes on AUPRC and AUC

Even in the non-normalized graph, it is clear that there is a sharp uptick in AUPRC and AUC for many terms at .1, the point at which the Trial 2 predictions were first included. The curves for individual terms take a variety of shapes, but I distinguished two main groups, one of which climbs steeply and gradually levels off (presumably the terms which did best in Trial 2), while the other decreases steadily (the terms which did best in Trial 1). The two groups appear to converge when the weight of the Trial 2 predictions is .4. I compared AUPRCs and AUCs for weights of .1, .2, .3, and .4, and found that .3 maximized both measures for the greatest number of terms. Thus my final predictions were generated using a voting scheme with Trial 1 predictions weighted at 70% and Trial 2 predictions weighted at 30%.

3.3.2. Voting Outperforms Trials 1-3

The voting scheme with words weighted at .3 produced the best AUPRC for 102 of the 155 terms (65.8%) and the best AUC for 83 terms. In the cases where voting did not produce the best AUPRC, the difference from the best was small; the discrepancy was less than 5% for all but 6 terms. Median AUC was .90, a slight improvement over the other trials. Looking at the box-and-whisker plots in Figures 8 and 9 below, we can see that the predictions resulting from voting give the highest values for all three quartiles.

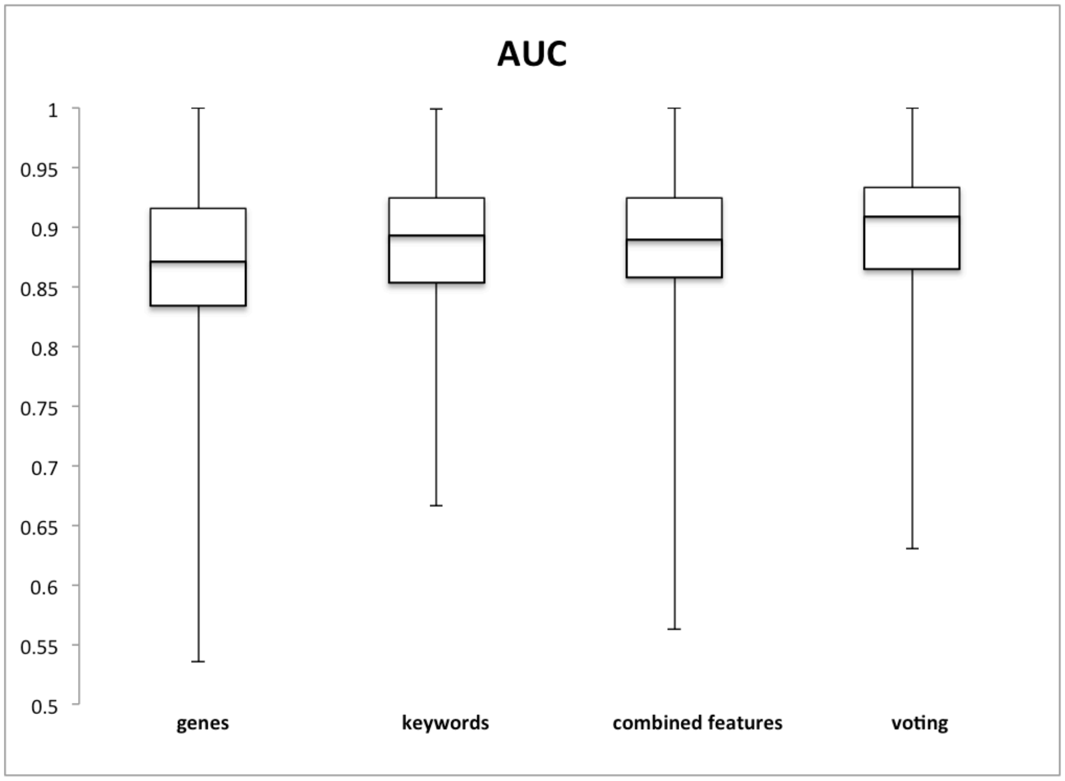


Figure 8: Box-and-whisker plot comparing AUCs from Trials 1-3 and from voting

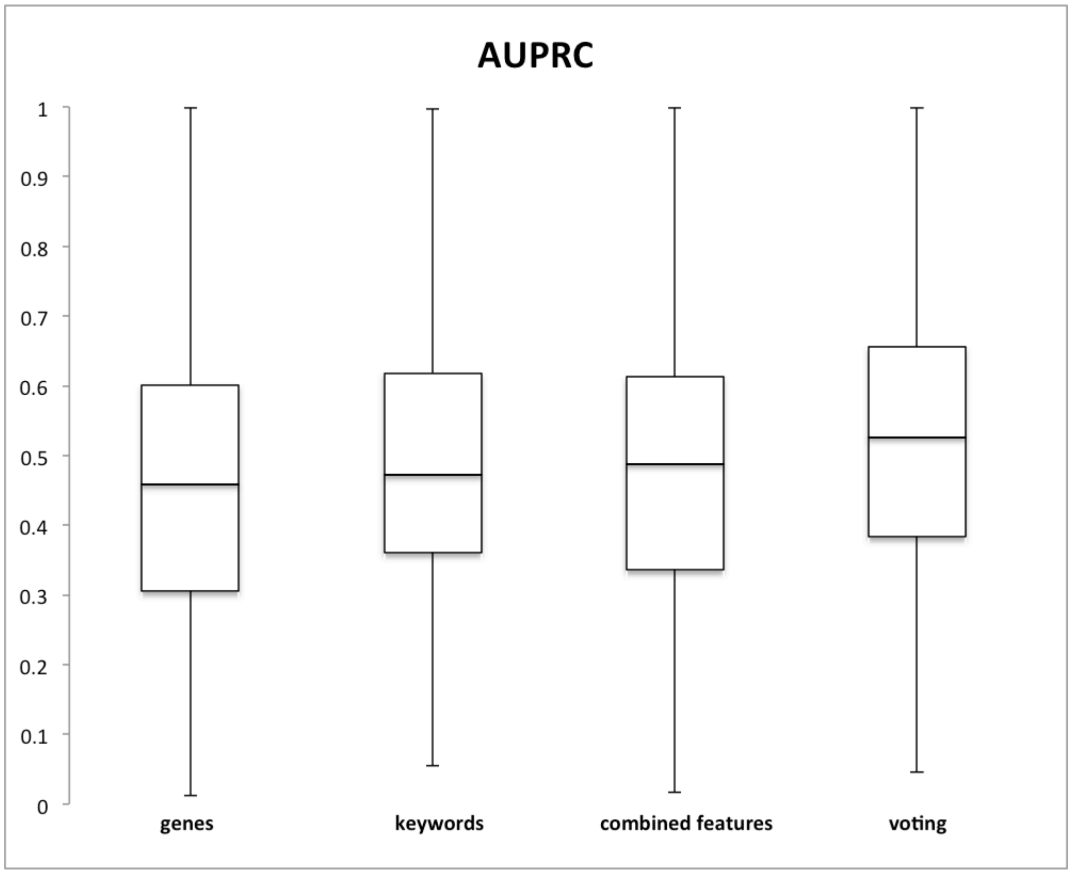


Figure 9: Box-and-whisker plot comparing AUPRCs from Trials 1-3 and from voting

4. Conclusion

This study demonstrates that support vector machines using literature co-occurrence data as features can predict GO annotations with high accuracy. In many cases where simple gene-gene co-occurrence does not work well, better results can be obtained using gene-keyword co-occurrence. I found that a very simple text mining strategy — identifying words that occur in only one GO term definition — was an effective way of choosing keywords. Although predictions based on gene-gene co-occurrence and those based on gene-keyword co-occurrence were highly correlated, there are terms for which one set of predictions was significantly more accurate than the other. I was able to combine the two sets of predictions effectively using a voting scheme in which gene-gene predictions were weighted at 70% and gene-keyword predictions at 30%.

My method has some notable limitations, one being that it does not take advantage of the hierarchical structure of the Gene Ontology. If it were used to make predictions for all ontology terms, instead of the small, non-overlapping subset included in the yeast slim, it would encounter the type of problems described by Barutcuoglu (2006) – namely, some predictions would be hierarchically inconsistent. Additionally, I suspect that my method for selecting keywords, while very successful in my tests, would not work as well if applied to a larger number of terms. With more terms, it would presumably be harder to find words that occurred in only one term definition. In this case, a more sophisticated text mining strategy might be necessary.

Limitations notwithstanding, the methods used in this study produced classifiers that are highly accurate by biological standards and have the potential to add to our knowledge of the genome by successfully predicting new GO annotations. In addition, my results indicate that literature co-occurrence is a rich source for inferring biological relationships, and that it does not need to be combined with other types of data to be useful. This result has implications for anyone involved in GO annotation prediction or related prediction tasks (gene function prediction, protein function prediction, etc.)

4.1. Future Work

A logical follow-up to this project would be to take my most confident set of predictions and identify all the false positives that occur below a certain recall — i.e. cases where the SVM models predicted a high likelihood that a gene was annotated to a given term, but no such annotation existed. These false positives could then be experimentally validated and potentially lead to new GO annotations, improving biologists' understanding of the yeast genome as well as analogous genes in more complex organisms. Additional future work could include applying my method to other model organisms, such as mouse, to test its generalizability, and ultimately applying the approach in humans.

5. Bibliography

Ashburner, Michael, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, et. al. 2000. “Gene Ontology: Tool for the unification of biology.” *Nature Genetics* 25 (1): 25-29.

Barutcuoglu, Zafer, Robert E. Schapire, and Olga G. Troyanskaya. 2006. “Hierarchical multi-label prediction of gene function.” *Bioinformatics* 22 (7): 830-836.

Cherry, J. Michael, Caroline Adler, Catherine Ball, Stephen A. Chervitz, Selina S. Dwight, Erich T. Hester, Yankai Jia, et. al. 1997. “SGD: Saccharomyces Genome Database.” *Nucleic Acids Research* 26 (1): 73-79.

Cock, Peter J. A., Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg. 2009. "Biopython: freely available Python tools for computational molecular biology and bioinformatics." Bioinformatics 25(11): 1422-1423.

Cortes, Corinna and Vladimir Vapnik. 1995. “Support-Vector Networks.” *Machine Learning* 20: 273-297.

Dhar, Vhasant. 2013. “Data Science and Prediction.” *Communications of the ACM* 56 (12): 64-73.

Gabow, Aaron P., Sonia M. Leach, William A. Baumgartner, Lawrence E. Hunter, and Debra S. Goldberg. 2008. “Improving protein function prediction methods with integrated literature data.” *BMC Bioinformatics* 9:198. http://www.biomedcentral.com/1471-2105/9/198

Gobble, MaryAnne M. 2013. “Big Data: The next big thing in innovation.” *Research Technology Management* 56 (1): 64-66.

Hey, Tony, Stewart Tansley, and Kristin Tolle. 2009. *The Fourth Paradigm: Data-intensive scientific discovery*. Redmond, WA: Microsoft Research.

Huttenhower, Curtis, Matthew A. Hibbs, Chad L. Myers, Amy A. Caudy, David C. Hess, and Olga G. Troyanskaya. 2009. “The impact of incomplete knowledge on evaluation: an experimental benchmark for protein function prediction.” *Bioinformatics* 25 (18): 2404-2410.

Jensenn, Tor-Kristian, Astrid Lægreid, Jan Komorowski, and Eivind Hovig. 2001. “A literature network of human genes for high-throughput analysis of gene expression.” *Nature Genetics* 28 (1): 21-28.

Joachims, Thorsten. 1998. “Making large-Scale SVM Learning Practical.” In *Advances in Kernel Methods: Support Vector Learning*, edited by Bernhard Schölkopf, Christopher Burges, and Alexander J. Smola, 169-184. Cambridge: MIT Press, 1999.

Lewis, Darrin P., Tony Jebara, and William Stafford Noble. 2006. “Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure.” *Bioinformatics* 22 (22): 2753-2760.

Maglott, Donna, Jim Ostell, Kim D. Pruitt and Tatiana Tatsuova. 2010. “Entrez Gene: Gene-centered information at NCBI.” *Nucleic Acids Research* 39 (Database Issue): 52-57.

Manyika, James, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. 2011. “Big Data: The next frontier for innovation, competition, and productivity.” *McKinsey Global Institute*, May. http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation

Mayer-Schönberger, Viktor, and Kenneth Cukier. 2013. *Big Data: A revolution that will transform how we live, work, and think*. Boston: Houghton Mifflin Harcourt.

Stapley, B.J., and G. Benoit. 2000. “Bibliometrics: Information retrieval and visualization from co-occurrence in Medline abstracts.” *Pacific Symposium on Biocomputing* 5: 526-537.

Troyanskaya, Olga G., Kara Dolinski, Art B. Owen, Russ B. Altman, and David Botstein. 2003. “A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*).” *Proceedings of the National Academy of Sciences* 100 (14): 8348-8353.

Vapnik, Vladimir N. 1999. “An Overview of Statistical Learning Theory.” *IEEE Transactions on Neural Networks* 10 (5): 988-999.

Vinayagam, Arunachalam, Rainer König, Jutta Moormann, Falk Schubert, Roland Eils, Karl-Heinz Glatting, and Sandor Suhai. 2004. “Applying Support Vector Machines for Gene ontology based gene function prediction.” *BMC Bioinformatics* 5:116. http://www.biomedcentral.com/1471-2105/5/116

Winnenburg, Rainer, Thomas Wächter, Conrad Plake, Andreas Doms, and Michael Schroeder. 2008. “Facts from text: can text mining help to scale-up high-quality manual curation of gene products with ontologies?” *Briefings in Bioinformatics* 9 (6): 466-478.

6. Appendix: Explanatory Figures

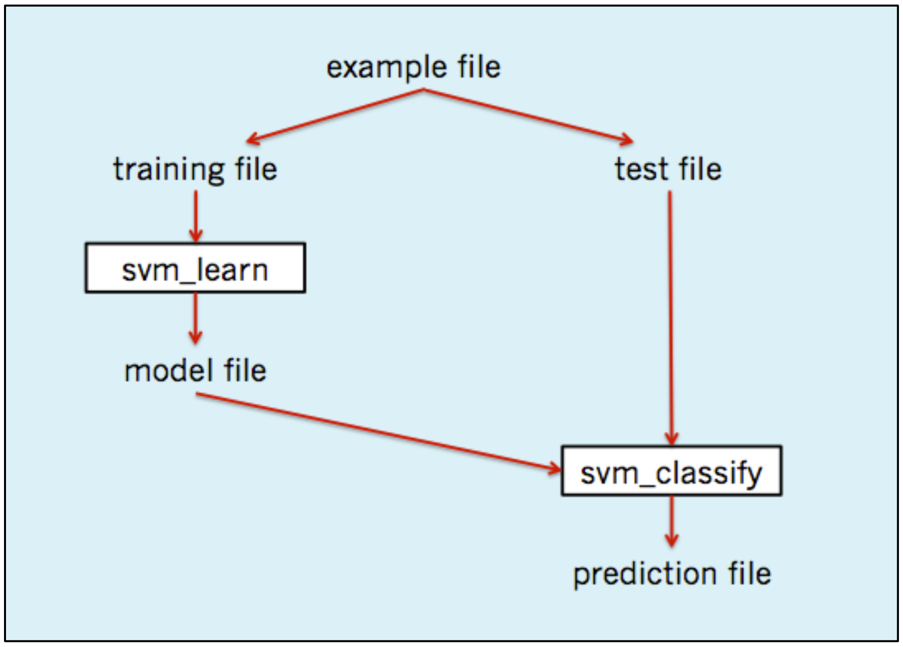


Figure 1: Flowchart illustrating my use of SVM Light

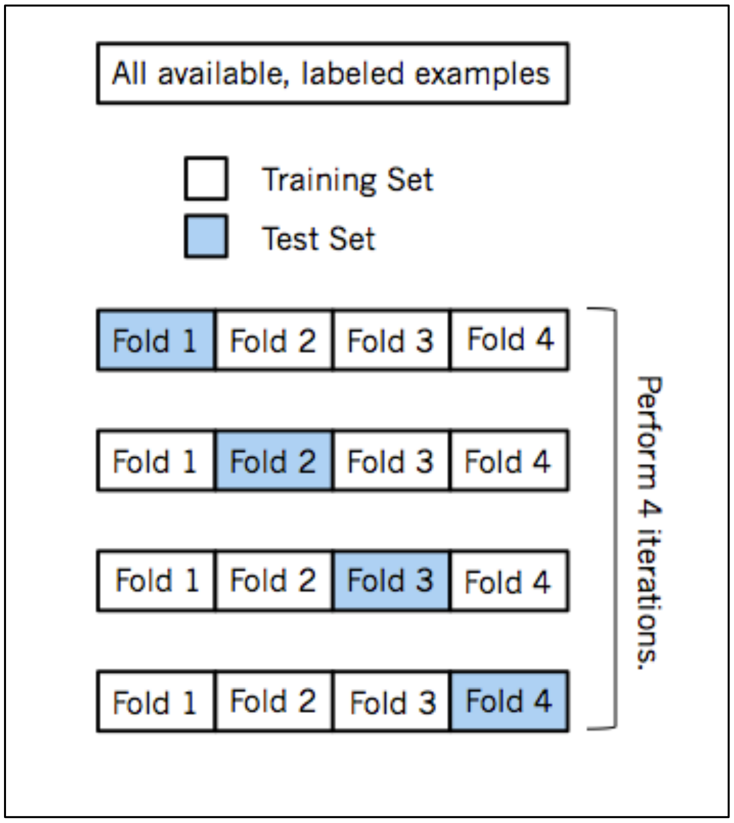
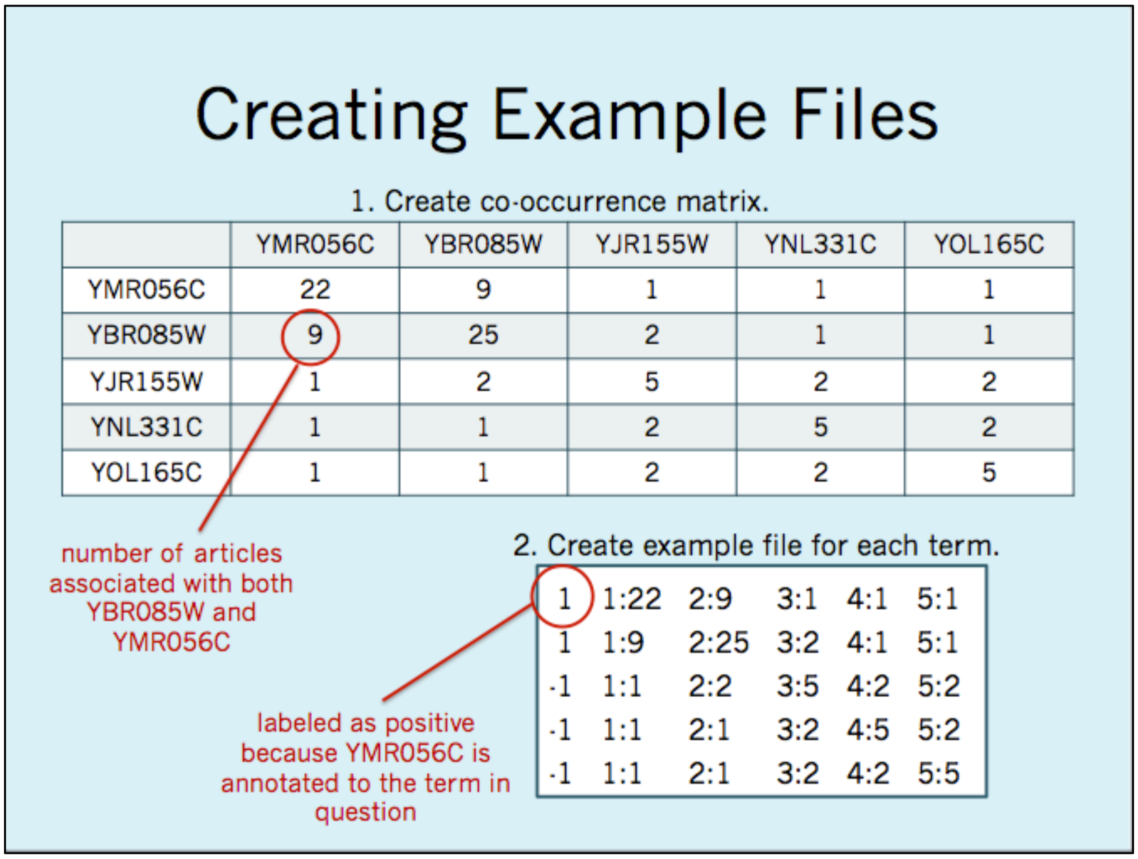


Figure 2: Illustration of fold-based cross-validation



Creating Example Files

1. Create co-occurrence matrix.

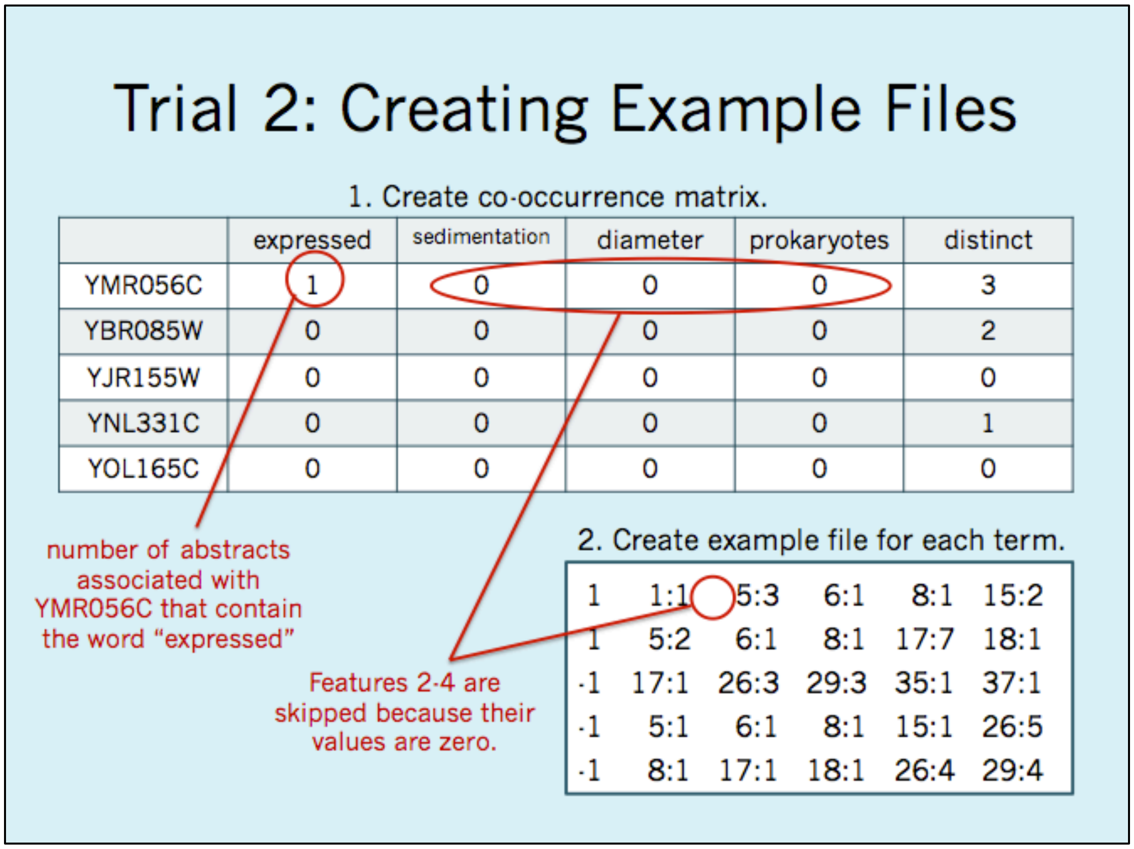
	YMR056C	YBR085W	YJR155W	YNL331C	YOL165C
YMR056C	22	9	1	1	1
YBR085W	9	25	2	1	1
YJR155W	1	2	5	2	2
YNL331C	1	1	2	5	2
YOL165C	1	1	2	2	5

2. Create example file for each term.

```
1   1:22  2:9   3:1  4:1  5:1
1   1:9   2:25  3:2  4:1  5:1
-1  1:1   2:2   3:5  4:2  5:2
-1  1:1   2:1   3:2  4:5  5:2
-1  1:1   2:1   3:2  4:2  5:5
```

number of articles associated with both YBR085W and YMR056C
labeled as positive because YMR056C is annotated to the term in question

Figure 3: This figure illustrates the process by which a co-occurrence matrix was used in Trial 1 to create the example files for SVM Light. In making the SVM Light example file, each row of the co-occurrence matrix was associated with an example and each column with a feature. For instance, observe that the gene YJR155W is the third column in the co-occurrence matrix and thus becomes feature number 3 in the SVM Light example file. The values for feature 3 – 1, 2, 5, 2, and 2 – correspond to the values in the third column of the matrix. The same gene, YJR155W, is the third row in the co-occurrence matrix and so also the third example in the example file.



Trial 2: Creating Example Files

1. Create co-occurrence matrix.

	expressed	sedimentation	diameter	prokaryotes	distinct
YMR056C	1	0	0	0	3
YBR085W	0	0	0	0	2
YJR155W	0	0	0	0	0
YNL331C	0	0	0	0	1
YOL165C	0	0	0	0	0

number of abstracts associated with YMR056C that contain the word “expressed”

2. Create example file for each term.

```
1    1:1   5:3   6:1   8:1   15:2
1    5:2   6:1   8:1   17:7  18:1
-1   17:1  26:3  29:3  35:1  37:1
-1   5:1   6:1   8:1   15:1  26:5
-1   8:1   17:1  18:1  26:4  29:4
```

Features 2-4 are skipped because their values are zero.

Figure 4: This figure illustrates the process used to create SVM example files for Trial 2. As in Trial 1, feature values in the example file correspond to the numbers in the co-occurrence matrix. (Rows in the matrix are associated with examples, columns with features.) In this case, however, the columns in the co-occurrence matrix represented words rather than genes. Many features were omitted from the example file because their value was zero.

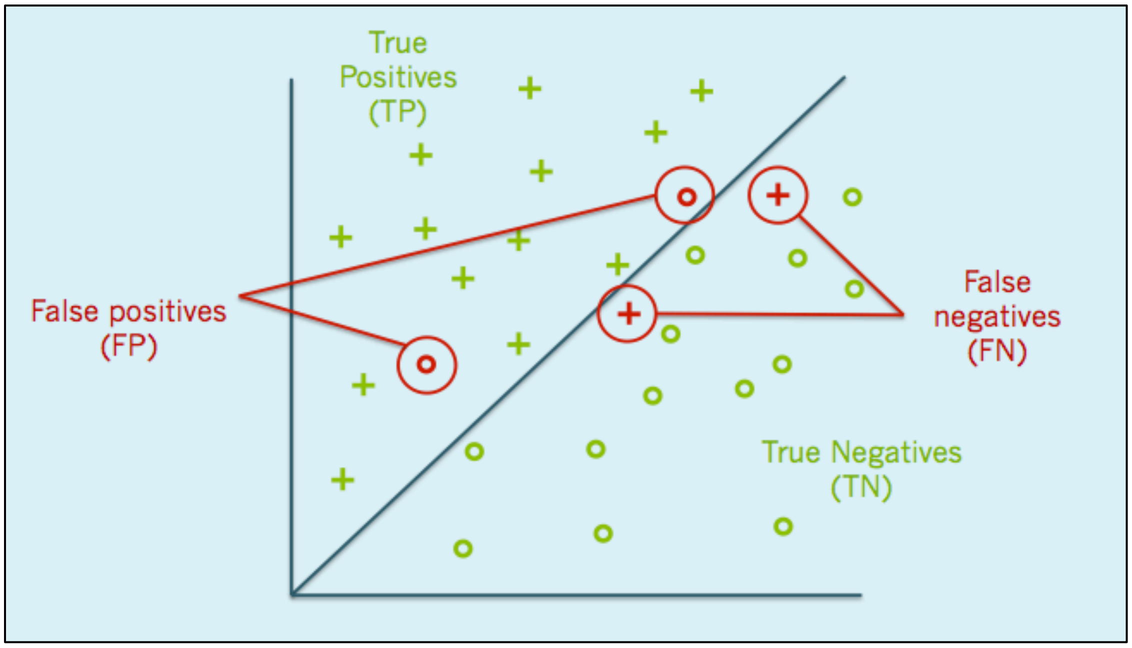
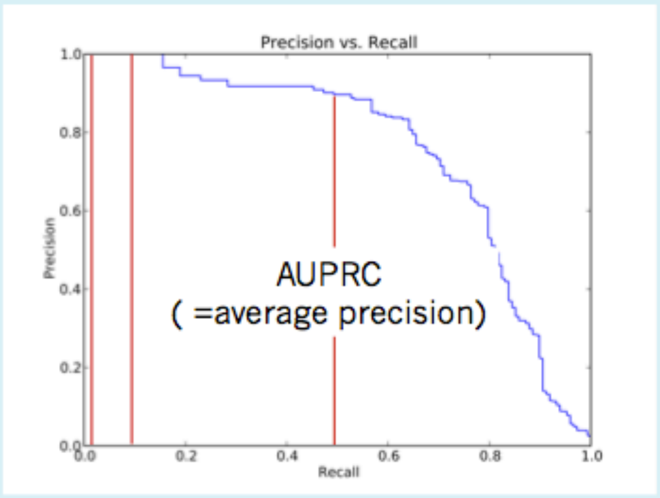


Figure 5: This figure illustrates the definitions of true positives, false positives, true negatives, and false negatives given in section 2.6.1. Plus signs represent positive examples, circles negative examples. The diagonal blue line represents the classifier chosen by a hypothetical SVM to divide the two classes of examples. All examples above this line have been classified as positive, while examples below the line have been classified as negative. The incorrectly classified examples – negative examples above the line and positive examples below the line – are shown in red.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

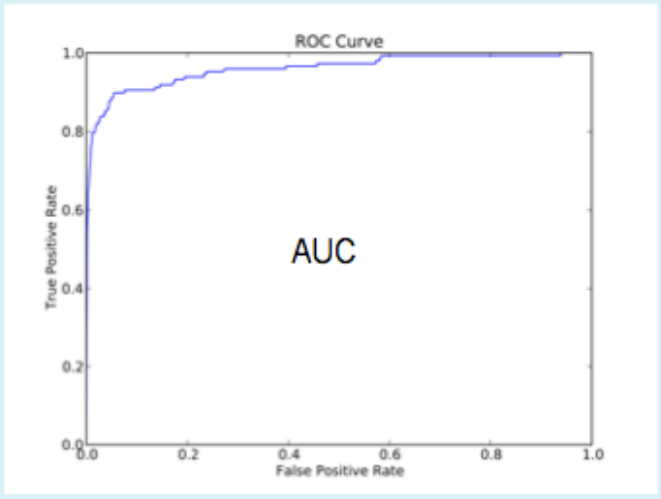


$$\text{Baseline AUPRC} = \frac{\text{\# of genes annotated to term}}{\text{\# of genes being classified}}$$

Precision at 1, 10, and 50% recall

Figure 6: Formulas for precision, recall, and baseline AUPRC (area under the precision-recall curve), along with an example of a precision-recall curve. 1%, 10%, and 50% recall are marked with red lines.

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$



Baseline AUC = 0.5

Figure 7: Formula for false positive rate, the measure used along with recall to plot ROC curves. A sample ROC curve is also shown. This particular graph has a high AUC (area under the curve) – well above the baseline of 0.5.