**Positive-Unlabeled Learning in the Face of Labelelling Bias**

## Noah Youngs

#### CY Data Science, New York Simons Center For Data Analysis, New York

Dennis Shasha Department of Computer Science, New York University, New York

Richard Bonneau Department of Computer Science, New York University, New York

## [nyoungs@cydatascience.com](mailto:nyoungs@cydatascience.com)

[shasha@nyu.edu](mailto:shasha@nyu.edu)

#### Department of Biology, Center for Genomics and Systems Biology, New York University, New York

Simons Center For Data Analysis, New York [bonneau@nyu.edu](mailto:bonneau@nyu.edu)

# ABSTRACT

Positive-Unlabeled learning scenarios, a class of semi-supervised learning where only a fraction of the data is labeled, and all available labels are positive, is becoming increasingly preva- lent as the number of large datasets grows. The human cost

of labeling these datasets can be high especially in situations where the number of unlabeled examples greatly outnumbers the labeled.

We perform several experiments, on both synthetic and real-world datasets, examining the performance of state of the art algorithms in the face of bias in the labeling process. In addition, we propose novel algorithms, and demonstrate that they outperform the current state of the art on a variety of benchmarks. Lastly, we present a methodology for remov- ing the costly parameter-tuning step in a popular positive- unlabeled algorithm, and show that the performance of this algorithm is uncompromised, and even sometimes enhanced, by this modification.

# Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous;

D.2.8 [**Software Engineering**]: Metrics—*complexity mea- sures, performance measures*

# Keywords

Labeling, Positive-Unlabeled Learning, Semi-Supervised Machine Learn- ing

# INTRODUCTION

With the number and size of uncurated datasets grow- ing exponentially, more and more learning scenarios provide few labeled examples. Labeling data is often either time- consuming, or costly in terms of domain-expertise required

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*WOODSTOCK* ’97 El Paso, Texas USA

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

to correctly perform the task. In addition, there are a num- ber of contexts in which labeling the positive class is easier than the negative. For example, when tagging documents with topics it is far easier to curate a database of topics, than to ask a user to list all topics a document is *not* about. While one could assume that the absence of a topic implies a negative example, this is only correct if the user tagging the document was provided with a *complete* set of topics before the tagging process. If an additional topic is added to the topic set, the lack of an annotation for that topic in previously labeled documents can no longer be equated with a negative example. This metaphor also extends quite eas- ily to image tagging, and several other important learning problems. This particular branch of Semi-Supervised learn- ing, where only a fraction of examples are labeled and those labels consist only of positive labels, is known as Positive- Unlabeled learning (PU learning).

Algorithms and heuristics have been applied to PU scenar- ios in a variety of fields, including text classification[], pro- tein function prediction[], gene interaction networks[], and many more. Only more recently, with the work of Elkan and Noto[], has the problem begun to be studied in a more general and theoretical framework.

We continue the general treatment of the PU problem, propose a novel algorithm, as well as examine the effect of bias in the labelling process. Additionally, we adapt an existing state of the art PU algorithm, removing the need for the computationally intensive tuning process.

# BACKGROUND AND RELATED WORK

The seminal work of Liu et al. [], formalized PU learning as a specific and noteworthy form of semi-supervised learn- ing. The work compared several existing PU algorithms in the context of text classification, as well as proposed a novel algorithm. This algorithm, Biased Support Vector Machines, demonstrated suprior performance to existing al- gorithms on several sample test datasets.

Later work by Elkan and Noto [] comped ?? Biased SVM to their own novel algorithms, based on an important the- oretical result: Elkan and Noto [] showed that the decision boundary of a classifier trained to differentiate between pos- itive and unlabeled examples will produce predictions that obey the same rank ordering as a traditional classifier that attempts to discriminate between positive and negative ex-

amples. Additionally, the decision threshold of such a clas- sifier can be transformed, such that itaˆA˘ Z´s accuracy is also in line with a traditional classifier[EN]. This result forms the basis for two algorithms: one in which a classifier is trained to discriminate between positive and unlabeled examples and then transformed, and another in which a similar methodology is used to calculate class probabilities, and then all unla- beled examples are duplicated in the training set, with each weighted according to those probabilities, and passed to a classifier that can learn on weighted data[].

Elkan and Noto, however, note that their work relies on one crucial assumption: that the set of labeled positive ex- amples is chosen uniformly at random from the set of all positive examples. While this assumption might appear in- nocuous, in many real-world PU applications, it is certainly violated. In protein function prediction, for example, anno- tations are very often propagated via homology to known se- quences, meaning that the labeled positives of a function are not selected at random from all proteins with that function,

The result of Elkan and Noto[] now becomes:

*g*(*x*) = *h*(*x*)*f* (*x*) *f* (*x*) = (*g*(*x*))*/*(*h*(*x*))

This new relation between f(x) and g(x) now relies on the ability to estimate h(x), and the ranking assumption no longer holds, unless there are specific conditions placed upon h(x). Namely:

Theorem 1.

*f* (*x*1) *≥ f* (*x*2) *≥ f* (*x*3) *↔ g*(*x*1) *≥ g*(*x*2) *≥ g*(*x*3) *iff*

*h*(*x*1) *≥ h*(*x*2) *≥ h*(*x*3)

Proof. I don’t think we need this. It follows from the multiplication rules. Can be stated as a fact.Let the following inequality hold:

*f* (*x*1) *≥ f* (*x*2) *≥ f* (*x*3) Substituting yields:

but rather according to a bias based on sequence similarity

*g*(*x*1)

*g*(*x*2)

*g*(*x*3)

to the first proteins for which that function was annotated.

It is not difficult to imagine other problem scenarios, such as text classification, where documents are given to curators based on search queries, and thus the set of labeled positive examples are again biased rather than random.

(*h*(*x*1) *≥* (*h*(*x*2) *≥* (*h*(*x*3)

Which can be split into:

*g*(*x* ) *≥ h*(*x*1) *g*(*x* )

1 *h*(*x* ) 2

2

*h*(*x*2)

# Theoretical Framework

We adopt the theoretical framework used in Elkan and Noto []. Let *x* be an example with a binary label *y ∈ {*0*,* 1*}*.

Let *s* be a second binary label for *x*, which indicates whether

*g*(*x*2) *≥*

It then follows that

*h*(*x*3)

*g*(*x*3)

the value of *y* is known. Since only positive examples are labeled in a PU scenario, we have the following axioms:

*s* = 1 *→ y* = 1

*s* = 0 *→ y* = 1 or *y* = 0

*p*(*s* = 1 *| x, y* = 0) = 0

Noah: not really clear what x, y = 0 means. Y= 0 is a logical statement so that’s fine, but x is an example, so what does “given x” mean? The assumption stated above, that the labeled examples are chosen uniformly at random from the set of positive exam- ples, can be expressed by the following: Noah:Same issue: p(y = 1| s = 1) = c is clear but not the first part.

*p*(*s* = 1 *| x, y* = 1) = *p*(*s* = 1 *| y* = 1) = *c* (1)

So the probability that any given positive example is labeled is a constant.

Elkan and Noto then propose to train a classifier *g*(*x*) =

*p*(*s* = 1 *| x*), which can be related to *f* (*x*) = *p*(*y* = 1 *| x*)

by the equation: *g*(*x*) = *cf* (*x*). The authors also present

several techniques for estimating *c* from observed data, all of which also rely on the ”selected at random” assumption stated above.

# NOVEL ALGORITHMS AND ANALYSIS

* 1. **Biased Labelling Analysis**

When the process for labeling a new positive example is at all dependent on the set of currently labeled examples the ”selected at random” assumption is no longer valid and the

probability of a positive example being labeled is now some function of that example itself: *h*(*x*) = *p*(*s* = 1*|y* = 1*, x*).

*g*(*x*1) *≥ g*(*x*2) *≥ g*(*x*3)

is gauranteed only if

*h*(*x*1) *≥* 1

*h*(*x*2)

*h*(*x*2) *≥* 1

*h*(*x*3)

which is equivalent to:

*h*(*x*1) *≥ h*(*x*2) *≥ h*(*x*3)

This result states that the ranking assumption only holds if the labeling bias is identically ranked, i.e. if example *i* is more likely to be positive than example *j* then *i* is also more likely to be labeled than *j*. This is not a prohibitive restriction, as it is not difficult to imagine a scenario in which new examples are labeled according to their similarity to existing examples (as is the normal case in bioinformatics for example) , but the restriction is worth noting.

# A Novel PU Algorithm

The goal of PU learning can be formalized in a slightly different manner as well: for each example x in the training set, we desire a function *D*, such that:

*D* : *x → Rn s.t. D*(*xp*) *∼ N* (*µp, σp*)

*D*(*xn*) *∼ N* (*µn, σn*)

The first condition Which condition are we talking about? I think it’s the condition that only positive examples are labeled, but please say. allows us to estimate *µp* and *σp*, since we have a set of labeled positive examples in the training

data, but not *µn* and *σn*, as all we have is a set of exam- ples drawn from the distribution: *N* (*µu, σu*) = *N* (*µn, σn*) + *N* (*µq, σq* ), where *q* is the set of true positives in the unla- beled set. That is to say, we can estimate the parameters of the distribution on the unlabeled training examples, but

the results will be a mixture of Gaussians, with greater bias

approximate probability of *x* coming from the positive class would be: *p*(*y* = 1*x*) = 0 = 0. Finally, if *D*(*x*) were larger than exactly half of the Monte Carlo points generated from both distributions, then the approximate probability of *x* coming from the positive class would be: *p*(*y* = 1*x*) =

*n*

0+*n*

2

*n n* = 0*.*5.

away Noah: I don’t understand. Q is whatever q is. That number could already be reflected in the N term. Maybe you just want to say with mu n reduced the greater q is. from *N* (*µn, σn*) the more true positives are included in the unlabeled set (the greater the size of *q*).

Once a function *D* is chosen, estimates are obtained for parameters of the positive and unlabeled distributions. For each example, the probability is computed that the exam- ple belongs to p, or to our proxy for n (which is trained on n + q). We attempt to compute these probabilities in three different ways: 1) We simply scale the output of *D* to [0*,* 1],

1. we approximate the probability via a monte-carlo tech- nique, and 3) we train a Guassian Mixture Model [] with two components on the output of *D*, and use the posterior prob- ability of an example belonging to each resulting component. These three techniques are described in greater detail below. We assume for simplicity that A “mu” is missing: *µp > q* , but if the choice of *D* implies the opposite, the following discussion still applies with trivial modification. By switching the direction of a few inequalities.

Once the probabilities have been approximated, they can be used as weights during the training of a classifier. In our case we use a weighted SVM classifier (see Experiments and Results), which incorporates the weights in two ways: the first being the methodology described in [EN] whereby

2 + 2

### 3.2.3 Probability Inference via Mixture Models

Assuming that we can find a function *D* that satisfies the desired properties specified in equation (@@the one in 3.2), the result of applying *D* to the underlying data will be a mixture of two guassians. Thus we can train a gaus- sian mixture model using the EM-algorithm, as presented in []. We seed the GMM algorithm with the labeled posi- tive examples as the initial members of one component and the unlabeled examples as the members of the other, with corresponding initial mixing proportions. It would also be reasonable to restrict the labeled positive examples to remaining in the positive component for the duration of the algorithm, but we leave this modification for future work.

Once we have estimates for (*µp, σp*) and (*µn, σn*), we ob- tain the probability that a particular *x* was generated from each of the two distributions. The final probability that *x* belongs to the positive class must also take into account the mixing proportions, and is given by:

*ϕ*(*x*; *µp, σp*)*πp*

all unlabeled training examples are duplicated and provided

*p*(*y* = 1*|midx*) = *ϕ*(*x*; *µ , σ* )*π*

+ *ϕ*(*x*; *µ , σ* )*π*

(3)

probability-based weights, and the second where only one

*p p p*

*n n n*

instance of each unlabeled exampled is used, but given a label based on whichever of the two classes it has a higher probability of belonging to, and also a weight corresponding to that probability. Noah: It seems that 3.2.1 to 3.2.3 are used in both new algorithms, so maybe they should be their own section.

### Probability via Scaling

We compute our scaling factor *t* as: *t* = max*x D*(*xp*), and then for all *x ∈ xu*: *p*(*y* = 1 *| x*) = *D*(*x*)

*t* , which by definition

yields results in [0*,* 1].

### Monte Carlo Probability Approximation

For our Monte Carlo approximation, we generate *n* points from the distribution *N* (*µp, σp*), and *n* points from the dis- tribution *N* (*µu, σu*). An estimate for the probability that x is in the positive class is given by:

*n ı*(*ki*; *x*)

Where *ϕ*(*x*; *µ, σ*) is the pdf of a normal distribution with

mean *µ* and *σ*, and *π* is the mixing proportion of a compo- nent.

### 3.2.4 Choice of the D Function

The ideal *D* has reduced dimensionality while maximizing the underlying variance of the data (so that it becomes separable between *p* and *n* + *u*), a natural choice for computing *D* is Principal Components Analysis[]. On very large datasets, however, PCA can become computationally inten- sive, and so we also propose a simpler *D* function, based on our previous negative algorithm selection algorithm SNOB [].

# 3.3 Biased SVM without Tuning

The Biased SVM algorithm proposed in [] demonstrates strong performance on many data sets, but relies on a tuning

*p*(*y* = 1*|x*) *∼*  *n*

*i*=1

*n*

(2)

process to calculate values for the parameters *C*

and *C* ,

*i*=1 *ı*(*ki*; *x*) +

*i*=1 **(*li*; *x*) *p u*

Where *ki* is the *i*th Monte Carlo point generated from *N* (*µp, σp*), *li* is the ith Monte Carlo point generated from *N* (*µu, σu*), and *ı*(*m*; *x*) and **(*m*; *x*) are indicator functions such that:

( 1 : *m < D*(*x*)

which control the difference in cost between misclassifying

positive and unlabeled examples in the SVM. This tuning process requires hundreds of calculations of SVMs on a sub- set of the training set in order to find the optimal pair of parameters, which is a computationally intensive process, and problematic when the training set is small and

*ı*(*m*; *x*) =

**(*m*; *x*) =

0 : *m ≥ D*(*x*)

( 1 : *m > D*(*x*)

0 : *m ≤ D*(*x*)

difficult to further subdivide. We provide an alternative to this tuning process, utilizing the probabilities calculated via any of the methods in 3.2.1-3.

The inspiration for the Biased SVM algorithm was the issue of class imbalance[], in which the two cost

Intuitively, if *D*(*x*) were so large that it was greater than

all Monte Carlo points generated from both *N* (*µp, σp*) and

*N* (*µu, σu*) the approximate probability of *x* coming from the

parameters were set such that:

+

positive class would be Needs format help: *p*(*y* = 1*x*) = *n* = 1. Conversely, Again, conditional prob vertical bar is missing

*n*+0

*C*+ = *n*

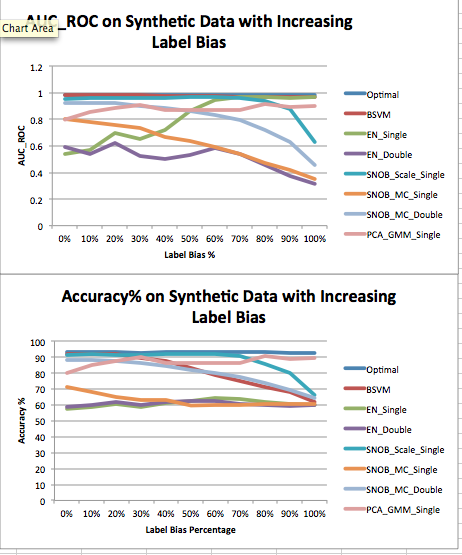
(4)

if *D*(*x*)were so small that it was less than all Monte Carlo points generated from both *N* (*µp, σp*) and *N* (*µu, σu*) the

*C− n−*

In a PU scenario, the number of true positives and nega-

tives are unknown, but we can use our class probabilities to calculate *E*(*n*+) and *E*(*n−*). This is done by simply sum- ming up the probabilities of belonging to each class, for all unlabeled examples, and then adding to *n*+ the number of labeled positive examples. Noah: Add in a table that shows how the various methods can be mixed an matched. There is a little issue of multiple testing. So I would suggest rerunning the experiments after separating training and testing data. Rerun on the training data. Find the best algorithm and then compare that best algorithms on the test data. Otherwise you could be accused of cherry-picking We then set the cost parameters:



*Cp* = 2*E*(*n−*)

*E*(*n*+) + *E*(*n−*)

+ *p −*

*Cu* = *E*(*n* )(1 *− C* ) + *E*(*n* )

*E*(*n−*)

# EXPERIMENTS AND RESULTS

We focus our experiments on learning scenarios in which

*|*P*| << |*Q*|*, as this is most common case in real-world data

where the source of the PU scenario is due to the cost of

labelling training examples. We examine the impact of bias in label choice via a synthetic example, as well as the per- formance of several algorithms on three real-world datasets. The structure of all our formalism allows for many com- binatorial variations of algorithms, using different choices of *D*, different methods of computing class probabilities, and different strategies of using those probabilities. We tested all such combinations, but for clarity only present the results from the 4 best peforming algorithms: 1) Using our gener- alization of SNOB as *D*, probability via scaling, and single copies of each unlabeled examples (SNOB Scale Single), 2) SNOB as *D*, probability via Monte Carlo approximation, single copies of each unlabled example (SNOB MC Single),

1. The same as 2 but with double copies of each unlabeled example (SNOB MC Double), 4) and using PCA as *D* (re- ducing to 10 dimensions, unless the underlying data is less than 10 dimensions, in which case the raw data is used), mixture models to calculates probabilities, and single copies of each unlabeled example (PCA GMM Single).

Alongside our algorithms, we compare the biased SVM method of [] (BSVM), the two algorithms proposed in [] (EN Single and EN Double), and an optimal upper bound calculated by revealing the true positive and negative exam- ples in the training set and training an SVM on the exposed data (Optimal).

In addition, we compare our Tuneless Biased SVM against the original Biased SVM algorithm, using three different ways to tune the parameters.

All algorithms use a SVM with linear kernel for the clas- sification task, and precision and *AUCROC* values are cal- culated using 5-fold cross-validation.

# Synthetic Data with Controlled Bias

In order to test the effect of bias on the PU learning scenario, we create 2-dimensional synthetic data from two Gaussian distributions. We generate positive and negative examples from distributions with mean *µp* = [4*,* 8], *µn* = [1*,* 5], and covariance matrices:

##### Figure 1: Results on Synthetic Data

origin are selected first, and since this direction is roughly perpendicular to the decision boundary between positive and negative values, creates a difficult learning problem. In order to explore the effect of differing amount of bias, we choose some labeled examples at random, and others according to the bias function. For example, if p = 100, and we set the bias amount to 40

# Real-World Datasets

### TCBD Database

The TCBD Database[], used as the original benchmark for the algorithms presented Elkan and Noto [], consists of text documents representing proteins. The set includes 2453 labeled positive examples, 348 unlabeled positive examples, and 4558 negative examples. We test on the original dataset, as well on the dataset with P and Q exchanged.

In addition, we explore the effect of bias on the labelling process, by selecting a random positive example, computing the correlation between the feature vector of that record and

Σ*p* =

.

( 1 0*.*5 \

0*.*5 2

Σ*n* =

( 1*.*5 0*.*5 \

0*.*5 2*.*5

the feature vectors of all other positive examples. We label 10% of all positive examples, ranking by those with highest correlation to our seed example. This process is repeated 5

We generate *p* = 100 labeled positive examples, *q* = 400

unlabeled positive examples, and *n* = 500 negative exam- ples. Additionally, when selecting the labeled positive ex- amples from the set of all true positives, we either select uni- formly at random, or according to a bias function: *bias*(*x*) =

.j(*x*2 + *x*2). Thus examples which are farther away from the

times, and the results averaged.

Lastly, we examine the effects of label bias external to the dataset. We choose positive examples as proteins from one species, or a combination of species, and let the rest of the positive examples go unlabeled. We repeat this process 5 times, with positive-label species chosen so that the number

1 2

of labeled positive examples ranges from 290 - 1142. Results are presented in Tables 1 and 2.

### Internet Advertisements Dataset

The Internet Advertisments Dataset was obtained from the UCI machine learning repository[], and first appeared in []. The dataset consists of 3 real-valued features, which have been removed, and 1555 categorical features. The data contains 458 positive examples (advertisments), 20% of which were labeled using the same seeding process de- scribed in section 4.2.1. The remaining 366 positive exam- ples were included with the 2821 negative examples (non- advertisements) as the unlabeled set. Results are presented in Figure 4.

Accuracy results for all methods were fairly similar, with the exception of our PCA GMM Single method which did not perform well. In terms of AUC ROC, however, all of our showcased algorithms outperformed the current state of the art, with an especially significant margin for SNOB MC Double.

### First-Order Theorem Proving Dataset

The First-Order Theorem Proving Dataset was also ob- tained from the UCI machine learning repository[], and first appeared in []. The dataset consists of 51 real-valued fea- tures. Positive examples were chosen as theorems in which non of the five solvers was able to find a solution, and all other examples (any of the solvers was able to find a so- lution) were considered negative. With this definition, the dataset contains 624 positive examples, 20% of which were labeled using the same seeding process described in section

4.2.1. The remaining 499 positive examples were included with the 5494 negative examples as the unlabeled set. Re- sults are presented in Figure 5.

On this dataset, most of our showcased algorithms markedly outperformed the current state of the art, in both Accu- racy and AUC ROC. SNOB MC Single in particular, shows strong performance with small variance in the results.

# 4.3 Tuneless vs. Tuned biased SVM

We compare the performance of our tunless Biased SVM algorithm against the original tuned algorithm described in[]. By removing the tuning procedure, we achieve a speed- up factor of 621 (when using the original parameter ranges for *Cp* and *Cu* specified in []) minus the cost of computing the probabilities, which is small.

Results are presented in Tables 3 and 4, where our al- gorithms (specifically GM1Bias) perform as-well of better than the original BSVM in both accuracy and AUC ROC. GM1Bias calculates *Cp* and *Cu* via the procedure specified in section 3.3, with the probabilities calculated by fitting a gaussian mixture model (section 3.2.3) to the SNOB score of the data (section 3.2.4).

# CONCLUSIONS

We have proposed several novel algorithms that perform a weighted classification, utilizing class membership probabil- ities as weights. The weights are obtained by first applying a feature transform to the data, and then turning the resulting score into a probability. Out of the different combinations of methods we examined, utilizing our SNOB feature map- ping, with a monte-carlo probability approximation, yielded the best results. These results help for both the single and

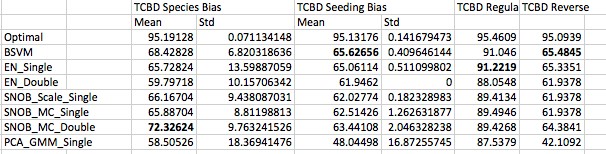
double variants, the latter in which each unlabeled train- ing example is duplicated and each replicate weighted by its respective class membership probability, while the former contains only one copy but with label and weight defined by the higher of the two class membership probabilities.

In addition to showing that our algorithms performed comparably or better than the current state of the art on several datasets, we also demonstrated that similar method- ology for calculating the class membership probabilities can be used to bypass the expensive parameter tuning step in the Biased SVM algorithm. We found that our method for calculating expected optimal parameters based on a 2- component gaussian mixture model applied to the SNOB feature mapping of the data, produced results as good or better than the tuned Biased SVM on all of our candidate datasets.

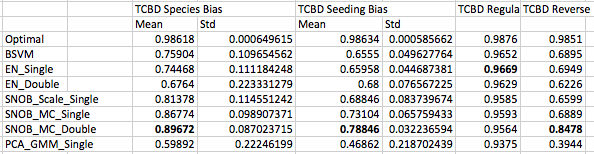
This paragraph just hurts the argument. While additional work is warranted to explore some of the underlying causes of the variance in method performance on different types and scales of data, we believe that the work presented here represents a step forward in the performance of machine learning algorithms in positive-unlabeled learn- ing scenarios.

# ACKNOWLEDGMENTS

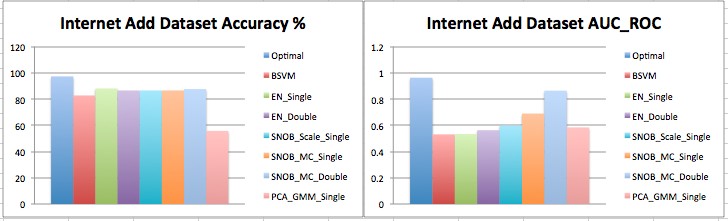
Thanks to NYU HPC Abu Dhabi.... other people



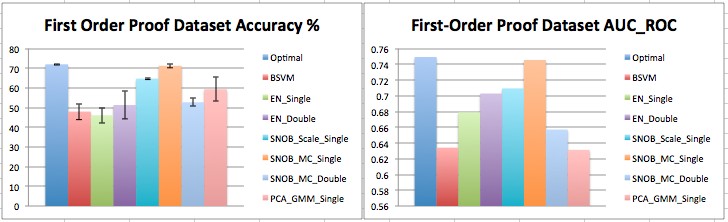
##### Figure 2: (TABLE 1) TCBD Database Accuracy Results



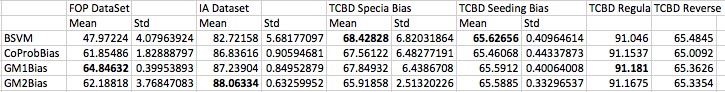
**Figure 3: (TABLE 2) TCBD Database AUC ROC Results**



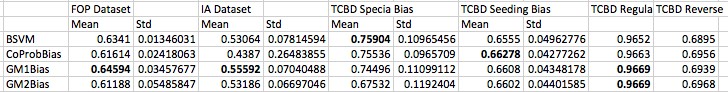
**Figure 4: IA Database Results**



**Figure 5: FOP Data Results**



**Figure 6: (TABLE 3) Tuneless BSVM Accuracy Results**



**Figure 7: (TABLE 4) Tuneless BSVM AUC ROC Results**