

COMP 755 Final Project: Adapting Target-Decoy Competition For Real Time Experiments

Trent Stohrer

November 30, 2018

1 Abstract

One of the key computational tasks of proteomics is matching data from mass spectrometers (mass spectra) to a database of theoretical mass spectra of peptides based on known protein amino acid (AA) sequences. My main project in the lab has been based on doing these database matches in real-time to more efficiently decide when the mass spectrometer (MS) should perform various types of scans during our experiments. Currently, these decisions are based on a static threshold; only if a candidate database match is above a pre-defined cross correlation threshold will the MS perform a scan dedicated to quantifying the peptide. However, using a static threshold leads to many quantification scans for peptides whose AA sequences are not reliably identified, and these false positives waste the MS's limited resources. Many of the advancements in improving database search identification rates have been based on machine learning methods that incorporate multiple features. The general method for training these models is a per-experiment semi-supervised binary classification model that is trained after all the data has been acquired. In this model, "target" peptides from a database are considered to be "possibly" positive examples, and decoy peptides, created by reversing or shuffling real peptide sequences, are used as negative examples on which to train. In this project, I evaluate the effectiveness of various classification schemes that do not use the full data set, including training on previous experiments, training on the first n scans, and classification using

online learning, the goal of all of these being to better identify peptides in real-time, with an emphasis on reducing false positives, and therefore improve decision making with regards to which scans the MS should perform next.

2 Introduction

Semi-supervised methods for predicting peptide-spectrum matches (PSMs) are based on the concept of target-decoy competition, which uses the aforementioned shuffled peptide sequences as definitive negative examples for training, and controls the false discovery rate using the number of decoys that are misclassified as correct PSMs. On the other hand, PSMs that map to target peptides are an unknown mixture of positive and negative matches. The simplifying assumption of the target-decoy competition is that incorrect matches are equally likely to be assigned to either a decoy or target peptide. Therefore, for every decoy sequence which is misclassified as a correct match, we expect there is also a target peptide that is misidentified as a correct match. I will further expound on the repercussions of semi-supervision in the methods section.

Without going too far in depth, the features we use are all meant to be indicative of a good peptide-spectrum-match or to otherwise inform how the other features should be interpreted. For an example of the first, IonFrac is a fraction of how many of the ion peaks we expect based on the theoretical spectrum are actually present in the experimental spectrum. The general idea here is that in a good match we will see most, or almost all of the ion peaks we expect to see. For the other type of feature, charge1-5 is a good example. These are five “one hot” features that represent the charge state of the ion in the experimental spectra. It is common to require higher cross correlation scores for peptides with higher charge, so by including these features, the hope is the classifier will learn this aspect of the boundary.¹

¹For a full description of each feature used see: <https://media.nature.com/original/nature-assets/nmeth/journal/v4/n11/extref/nmeth1113-S1.pdf>. Of those features the ones we actually used were XCorr, δC_n , δC_n^L , Sp, ln(rSp), $\delta M \text{ abs}(\delta M)$, ionFrac, ln(NumSp), enzInt, pepLen, charge1-3 (expanded to 1-5). We had access to Mass, enzN, and enzC as well, but due to some processing errors not noticed until very late, these were not included in the data import.

3 Method

The model used by the algorithm we wanted to approximate (Percolator) is SVM, so we decided to go with the sklearn python packages SGDClassifier using hinge loss and L2 regularization to build a similar model that would allow for online learning, though this did limit us to linear SVM. We also standardized our data, based on the understanding that SGD classifiers can be sensitive to data scaling issues. For all the schemes we used, we used the means and standard deviations of the training set to scale all of the data, as in practice we would not have access to these overall values for a live experiment. To generate the input data we ran our own experimental data (mzML mass spec file) through the tide database search as part of the crux analysis toolkit, stripped out the non-feature columns, and sorted by scan number to simulate the arrival time of an actual experiment.

As a note for terminology, from here I will refer to PSMs which appear to be correct matches to target peptides as target matches. Semi-supervision means that the labels for potential target peptides will change as the classifier is better able to differentiate decoy and incorrect targets from the correct target matches. This leads to a programmatic structure where we go through multiple rounds of training, finding a new false-discovery-rate-controlled threshold for each round, re-classifying based on that threshold and the current classifier, randomly selecting new false examples, and re-training in each round.

Semi-supervision also means that we have to bias our initial positive examples or we will be learning the boundary between targets and decoys rather than the boundary between target matches and all non-matches (both decoys and targets). We actually end up using the same feature as the static threshold, cross correlation, controlled for false discovery in the training set, as the initial direction for classification before subsequent rounds classified based on the SGDClassifier for that round. All of the methods we tried use the same general training sequence: pick an FDR-controlled cross correlation threshold, do nine additional rounds of training with their own FDR-controlled SVM score thresholds then test on the test set. Since we have access to all the classifier scores as we go, we can write these out to a file and select a false discovery after the fact. In an actual experiment we would need to select our false discovery rate before hand, but in this setting we can build a ROC curve to evaluate the selectivity/sensitivity tradeoff for a variety of FDRs. The difference between the methods is how much additional training we do,

how we do the additional training, what we use as the training set, and to a smaller degree how we use the threshold. I will go into more detail on the difference between methods in the results section. I did not find the time to tweak the parameters of the SGDClassifier, but I did tweak and test some other parameters important to the semi-supervised method. For all of the methods I changed the initial direction false discovery rate as well as a separate false discovery rate for the learning thresholds and the final threshold. For the “first N” methods I changed the FDR’s as well as the N. There was not sufficient time to fully test different values for these parameters, and I did not get to change batch size for the online learning models at all, but my results have included a few different results where relevant. The results are based on two distinct experiments, a whole cell lysate experiment with no labeling and no MS3 scans, MA1923, and a MIBMS experiment with labeling and interleaved MS3 scans, MA1750.

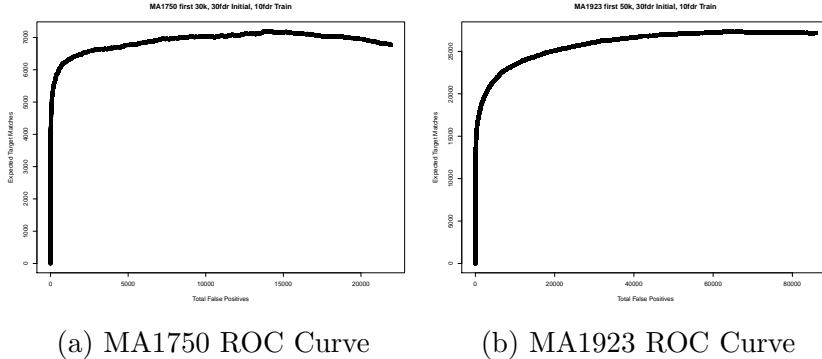
4 Results

The first method we tried was to use a previous, similar experiment as the training set and to test on a subsequent experiment. This limited us to only testing on MA1750, as getting data for an experiment similar to MA1923 was not time-feasible. Before I present the results, I need to explain the columns of the chart, which will apply to subsequent charts. I have included a row for static in each chart, though this will be the same regardless of the method were looking at. All columns between Static or ML and TM Found are the parameters we tweaked. TM Found is the number of True Matches found in the test set, here we are using the results of percolator run on the test set using a false discovery rate cutoff of 1% to define which matches are True Matches. TM Missed is the number of True Matches found by percolator that we did not find with the given parameters and classifier and False TM is the number of targets that the classifier mistakenly classifies as True Matches (mistakenly as in percolator does not identify them as such). Finally, Test Set FDR is the false discovery rate of the classifier over the test set, here calculated as $(2 * \text{decoys as targets}) / (\text{decoys as targets} + \text{targets as targets})$, where we double the numerator from the assumption that each matched decoy has an equivalent mistakenly matched target.

Next we tried using the first N scans of the experiment, each having around 5 targets and 5 decoys, as the training set, and the remaining E-N

| Data Set | Static or ML | Initial FDR | Threshold FDR | TM Found | TM Missed | False TM | Test Set FDR |
|----------|--------------|-------------|---------------|----------|-----------|----------|--------------|
| MA1750 | Static | N/A | N/A | 6836 | 107 | 4760 | 41.4% |
| MA1750 | ML | 10% | 10% | 4178 | 2765 | 26 | 0.332% |
| MA1750 | ML | 20% | 10% | 5567 | 1367 | 104 | 1.44% |
| MA1750 | ML | 30% | 10% | 6191 | 752 | 184 | 2.23% |
| MA1750 | ML | 20% | 20% | 6512 | 431 | 267 | 2.88% |
| MA1750 | ML | 30% | 20% | 6752 | 191 | 907 | 9.31% |

Table 1: Previous Scan Learning



scans as the test set. Here we also made ROC curves for our various attempts, for reasons I will explain when we get to our first attempt at online learning. These are a modified version of the typical ROC curve. Here both axes are raw counts rather than percentages, with the x-axis being decoys identified as targets for the given threshold, and the y axis being targets identified as targets minus the number of decoys identified as targets, using the target-decoy competition assumption. Note that in these ROC curves we do not use the results of percolator as our ground truth.

In our next model we do the same initial training as the first N training, but do subsequent additional training using SGDClassifiers partial_fit method on additional batches of scans to further refine the classifier. One of the issues with this method is that each additional fit changes what FDR rate any given score threshold will represent. You can see this in the charts below as a 10% FDR cutoff learned right before testing can lead to drastically different results. Since finding the FDR constantly in small batches takes too much time, we chose to use the results from the ROC curves of the first

| Data Set | Static or ML | N | Initial FDR | Threshold FDR | TM Found | TM Missed | False TM | Test Set FDR |
|----------|--------------|--------|-------------|---------------|----------|-----------|----------|--------------|
| MA1750 | Static | 30,000 | N/A | N/A | 5349 | 83 | 3475 | 41.0% |
| MA1750 | ML | 30,000 | 1% | 1% | 4544 | 888 | 149 | 4.34% |
| MA1750 | ML | 30,000 | 10% | 10% | 5322 | 110 | 869 | 8.98% |
| MA1750 | ML | 30,000 | 20% | 10% | 5312 | 120 | 921 | 13.1% |
| MA1750 | ML | 30,000 | 20% | 20% | 5349 | 83 | 1533 | 18.9% |
| MA1750 | ML | 30,000 | 30% | 10% | 5352 | 80 | 944 | 9.56% |
| MA1750 | Static | 20,000 | N/A | N/A | 6236 | 91 | 4169 | 40.6% |
| MA1750 | ML | 20,000 | 30% | 10% | 5974 | 353 | 332 | 2.66% |
| MA1923 | Static | 50,000 | N/A | N/A | 18363 | 349 | 7464 | 27.6% |
| MA1923 | ML | 50,000 | 30% | 10% | 18286 | 426 | 2144 | 8.86% |
| MA1923 | Static | 40,000 | N/A | N/A | 19471 | 367 | 8158 | 27.7% |
| MA1923 | ML | 40,000 | 30% | 10% | 18965 | 873 | 1916 | 8.14% |
| MA1923 | Static | 30,000 | N/A | N/A | 20501 | 400 | 8896 | 28.1% |
| MA1923 | ML | 30,000 | 30% | 10% | 20255 | 646 | 2281 | 8.33% |

Table 2: First N Scan Learning

N method and simply statically use that threshold to test the online learning method, doing multiple runs at each threshold to test the stability. Each of these experiments uses batch size 5.

The online learning results were not where we wanted them to be, and from here we could have gone two different ways: we could have seen how adjusting the threshold on each partial fit works, or we could do a full retrain using the first $N+x \cdot \text{batch_size}$ scans as the train set. We decided to go with this second method.

| Data Set | Iter -ation | N | Initial FDR | Threshold FDR | Classifier Threshold | TM Found | TM Missed | False TM | Test Set FDR |
|----------|-------------|--------|-------------|---------------|----------------------|----------|-----------|----------|--------------|
| MA1750 | 1 | 30,000 | 30% | 10% | -7.57 | 5357 | 75 | 3899 | 43.9% |
| MA1750 | 2 | 30,000 | 30% | 10% | -7.57 | 5358 | 74 | 3326 | 39.4% |
| MA1750 | 1 | 30,000 | 30% | 10% | -4.19 | 5345 | 87 | 2537 | 32.7% |
| MA1750 | 2 | 30,000 | 30% | 10% | -4.19 | 5339 | 93 | 1652 | 21.4% |
| MA1750 | 1 | 30,000 | 30% | 10% | -0.73 | 5294 | 138 | 1120 | 17.7% |
| MA1750 | 2 | 30,000 | 30% | 10% | -0.73 | 5306 | 126 | 938 | 14.6% |
| MA1750 | 1 | 30,000 | 30% | 10% | Runtime 10% | 5168 | 264 | 504 | 5.19% |
| MA1750 | 2 | 30,000 | 30% | 10% | Runtime 10% | 4648 | 138 | 784 | 1.82% |
| MA1923 | 1 | 50,000 | 30% | 10% | -1.63 | 18561 | 151 | 8712 | 27.6% |
| MA1923 | 2 | 50,000 | 30% | 10% | -1.63 | 18557 | 155 | 8952 | 27.3% |
| MA1923 | 1 | 50,000 | 30% | 10% | Runtime 10% | 9442 | 9270 | 164 | 1.06% |
| MA1923 | 2 | 50,000 | 30% | 10% | Runtime 10% | 11046 | 7666 | 180 | 0.799% |

Table 3: Simple Online Learning

| Data Set | N | Initial FDR | Threshold FDR | TM Found | TM Missed | False TM | Test Set FDR |
|----------|--------|-------------|---------------|----------|-----------|----------|--------------|
| MA1750 | 30,000 | 30% | 10% | 5349 | 83 | 832 | 9.97% |
| MA1923 | 50,000 | 30% | 10% | 17170 | 1542 | 1074 | 10.0% |

Table 4: Full Retrain Online Learning