2021

ENSEMBLE PROTEIN INFERENCE EVALUATION

Kyle Lee Lucke
University of Montana, Missoula

ENSEMBLE PROTEIN INFERENCE EVALUATION

By

Kyle L. Lucke

Bachelor of Science, University of Montana, Missoula, MT, 2019

Thesis

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

Fall 2021

Approved by:

Ashby Kinch Ph.D., Dean
Graduate School

Dr. Oliver Serang Ph.D., Chair
Computer Science

Dr. Douglas Brinkerhoff Ph.D.
Comuter Science

Dr. Eric Chesebro Ph.D.
Mathematics

Lucke, Kyle L., M.S., October 2021 Computer Science

Ensemble Protein inference evaluation

Chairperson: Dr. Oliver Serang

The Protein inference problem is becoming an increasingly important tool that aids in the characterization of complex proteomes and analysis of complex protein samples. In bottom-up shotgun proteomics experiments the metrics for evaluation (like AUC and calibration error) are based on an often imperfect target-decoy database. These metrics make the inherent assumption that all of the proteins in the target set are present in the sample being analyzed. In general, this is not the case, they are typically a mix of present and absent proteins. To objectively evaluate inference methods, protein standard datasets are used. These datasets are special in that they have been carefully prepared to contain only the proteins specified in the target set. Though this helps, it is still unclear which metrics most adequately capture all the important aspects of a good protein inference method. In this manuscript, a novel protein standard dataset, an ensemble protein inference engine that utilizes several metrics and protein standard datasets to evaluate the performance of inference methods, and several novel protein inference methods are presented.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Oliver Serang for always being there to help me see the light at the end of the tunnel. One could not ask for a better advisor, mentor, or boss. Thank you for always being there to pick me up when I fell. I would also like to thank my coworkers, Patrick Kreitzberg and Jake Pennington. Thank you for always having my back and helping me through the day. Finally, I would like to thank my family, my friends, and most of all, my significant other. To my parents, thank you for instilling in me a love of learning and what it means to be a hard worker. To my friends, thank you for always being there for me. Finally, to my significant other, thank you for all your love and support through it all.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 State of the field

The genome gives us an incredible amount of information; however, it does not give us as much information as one may think. For instance, the nucleus of each cell in the human body contains the same genetic information, and yet, clearly, a white blood cell performs different tasks than a muscle cell. The reason for this is the different proteins expressed in individual cells. Since before the first human genome was successfully sequenced, scientists have investigated proteomes, entire collections of proteins that can be expressed by a tissue, cell, or organism.

1.2 Bottom up Proteomics

Bottom-up shotgun proteomics is a popular technique for the characterization of complex protein mixtures [1]. In a typical bottom-up shotgun proteomics experiment, proteins are first digested by an enzyme (typically trypsin) into their constituent set of proteolytic peptides. These peptides are then separated by their hydrophobicity via liquid chromatography and then analyzed by a mass spectrometer, producing an MS1 spectrum containing the mass to charge (m/z) ratios of intact peptides. Then, if data-dependent acquisition is being used,the most abundant peak in the m/z window is selected and fragmented, often by collision-induced disassociation. This is in contrast to data-independent acquisition [2], where a set of many smaller, predefined windows are taken and everything in each of these smaller windows are fragmented simultaneously. The resulting fragments are analyzed by a second round of mass spectrometry, producing a tandem mass spectrum. This process is repeated for every peptide that elutes out of the sample, producing many tandem mass spectra. To determine which peptides emitted which spectra, a peptide search must be performed. First, an *in silico* digestion is performed on the protein sequences in the target-

decoy database according to the enzyme used in the experiment, producing a set of candidate peptides. Then, a theoretical spectrum for each candidate peptide is generated. Each observed spectrum is scored against each theoretical spectrum using a cross-correlation function, producing a set of peptide spectrum matches (PSMs). In the case of multiple peptides matching a spectrum, the PSM with the maximum score is typically retained.

These PSMs can then be post-processed with software, such as `Percolator` [3], to produce peptide level probabilities. There are a few different ways to calculate these probabilities, each with their own interpretation. One version is that this probability represents the probability that this PSM is due to random chance, sometimes referred to as the posterior error probability [4] (PEP). Another way to calculate this probability is as a likelihood: $\Pr(D_k|Y_j = y_j)$, $y_j \in \{0, 1\}$, which represents the chances of observing the data, D_k, given that it was emitted by peptide Y_j. Note that the latter can be recovered from the former using Bayes' theorem [5]. The latter is more desirable when computing peptide-level probabilities to be used in protein inference, as it does not include a peptide-level prior (since it is a likelihood). This is important when computing protein-level posteriors because the protein-level priors may conflict with the peptide-level priors. Further, it is impossible to properly compute $\Pr(Y_j)$, the prior probability of peptide Y_j being present in the sample, without some knowledge of the relations between the proteins and peptides under consideration, as this prior is informed by the detectability of the peptide, which is an intrinsic property of both the peptide itself and the parent protein which the peptide resides in [6]. The detectability of a peptide is defined as the probability that a peptide has of being identified in an experiment [7]. In contrast to bottom-up experiments, top-down assays do not digest the protein but instead send the intact protein through the mass spectrometry machine and then fragment the protein [8].

A common way to visualize the relationship between proteins the peptides they may emit is a bipartite graph. The node partitions of the graph represent proteins and peptides, with an edge between a protein and a peptide if that protein may have emitted that peptide. This representation is typically referred to as a protein-peptide graph (PPG). It should be noted that there is sometimes a distinction made between the theoretical PPG and the observed PPG. A theoretical PPG, pictured in Figure 1.1a, relates proteins to their theoretical peptide sets (and sometimes PSMs, in which case it is then a tripartite graph), while an observed PPG, pictured in Figure 1.1b, relates proteins to their observed peptide sets, that is, the set of peptides in the PSMs.

An inference method then uses the observed PPG, along with the set of scored peptides to produce protein-level confidence scores indicating how strongly the inference method believes each protein is present in the sample. These scores can then be thresholded in some manner to produce a set of proteins the inference method believes to be present and absent. For example, a naive thresholding method would be to identify all proteins with a protein-level confidence score above 0.9 as present; however, for such a thresholding method to work well, it is required that the scores produced to be $\in [0.0, 1.0]$ and are well calibrated (*i.e.* true probabilities). If the scores are discriminative but not well calibrated, a global false discovery rate [9] (FDR) or q-value [10] threshold can be utilized. This way, it is not necessarily required the method produce true probabilities for the thresholding to work well.

1.3 Definitions and Notation

Several definitions that will be used throughout this manuscript are defined in Table 1.1. Although it may not be immediately obvious, the terms present and

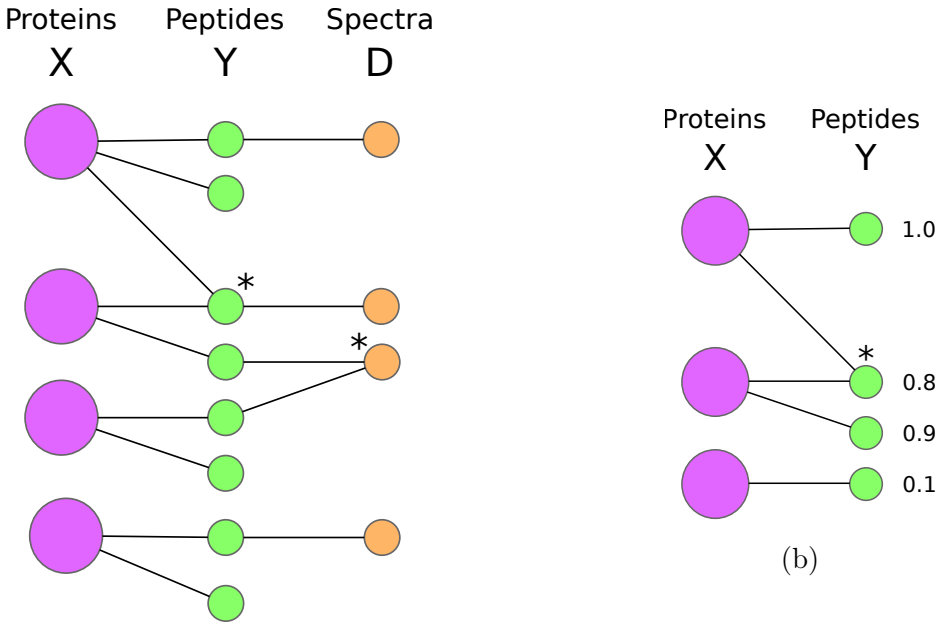


Figure 1.1: **Two different ways to visualize the relation between proteins and peptides.** Both panels represent the same underlying data. The * denotes a degenerate peptide or spectra, respectively. Degenerate peptides are peptides which may have been emitted by multiple proteins, while degenerate spectra are spectra which may have been emitted by multiple peptides. **(a)** Graphical representation of a theoretical PPG. **(b)** Graphical representation of an observed PPG. Notice that the nodes in **(a)** which did not have a PSM are no longer included. The values next to the peptides indicate peptide-level confidence scores.

absent are orthogonal to the notions target and decoy, respectively. Further, decoys must be certainly be absent, they are used as false positives (FPs).

1.4 Existing methods and evaluation challenges

In this section, some existing protein inference methods are presented. Additionally, current protein inference evaluation metrics and some existing challenges are discussed.

Term or Notation	Definition
Present protein	A protein which is truly in the sample.
Absent protein	A protein which is certainly not in the sample.
Target proteins	Superset of proteins which are expected to be in the sample.
Decoy proteins	Set of proteins known to be absent from the sample.
Target-decoy-contaminant database	Database relating protein accessions for the target, decoy, and contaminant sets to their respective label.
Gold standard target-decoy-contaminant database	Target-decoy-contaminant database containing ground truth labels.
Obfuscated target-decoy-contaminant database	Modified Gold standard target-decoy-contaminant database where some of the decoy proteins have been intentionally relabeled as targets.
Entrapment protein	Ground truth decoy protein which has been relabeled as a target.
Degenerate peptide	A peptide which may be emitted by multiple proteins.
Present peptide	A peptide that is truly in the sample.
Absent peptide	A peptide that is not found in the sample.
X	Set of indicator variables for presence of proteins in the sample, indexed by i.
Y	Set of indicator variables for presence of peptides in the sample, indexed by j.
Protein-peptide graph	Bipartite graph relating proteins to the peptides they may have emitted.
Observed peptide	A peptide which has been matched to a spectrum.
Adjacent	A peptide, Y_j is said to be adjacent to a protein X_i (and vice-versa) if Y_j is found in the sequence of X_i.

Table 1.1: **Table of definitions which are used throughout this paper.**

1.4.1 Protein Inference

The protein inference problem is canonically posed as a set cover problem. Let U be the universe set, let S be a collection of subsets of U, and let C be any collection of subsets of S whose union is equal to U be called a covering. In the set cover problem, the objective is to find the smallest covering C. From a protein inference perspective, U typically consists of the set of observed peptides; however, to help reduce the number of FPs, these peptides are typically thresholded according to their

peptide-level confidence scores in some way first. The subsets in S consist of the set of observed peptides for the proteins under consideration. Hence, the goal is to find a minimal set of proteins which explain all of the (potentially filtered) observed peptides. Note that set-cover, when posed as an optimization problem, is NP-hard [11]. Hence, since protein inference can be solved with a set-cover routine, the protein inference problem is itself also NP-hard.

As with any inverse problem, properly handling degeneracy is a difficult task. Here, peptides are considered degenerate when they are found in the sequence of more than one protein in the target-decoy-contaminant database. How a method handles these degenerate peptides has a significant effect on how well an inference method performs in general. The notion of one-hit wonders can also complicate things. One-hit wonders are proteins with a single observed, typically high scoring, unique peptide. Such proteins can result from a few different scenarios. One scenario is that the single observed peptide was erroneously observed due to noise. In this case, the protein which may have emitted this peptide is likely absent. The other possibility is that the protein being investigated is small (*i.e.* does not consist of many peptides). It could also be that although the protein itself is truly present in the sample, other peptides for that protein are simply not observed. Unobserved peptides can result from several things: proteins do not always cleave properly when being digested, peptides with similar hydrophobicity may co-elute with one another in the liquid chromatography phase thus producing a chimeric spectrum [12], or the peptide has poor detectability. Peptides may have poor detectability due to the protein which emitted that peptide as well as the peptides found next to it in the protein sequences. These one-hit wonders pose a problem as they could easily be a FP (*e.g.* a protein inference method may erroneously identify a truly absent target protein as present).

FUTURE WORK: incorporate the idea of chimeric spectra into semi-supervised

features as well (e.g. we observed this peptide with this confidence, but there's x percent chance it's actually a chimeric spectra, etc.)

1.4.1.1 Existing inference methods

The following section briefly details models which are currently in use in the field. They are presented in no particular order. Some of the models described in this section have free parameters, and thus, require some form of parameter optimization. One of the heavily utilized metrics in this paper is the result `quality`, a convex combination of AUC and CE. Specifically, `quality` is computed as:

$$\texttt{quality} = (1 - \lambda) \cdot \text{CE} - \lambda \cdot \text{AUC}.$$

Where $\lambda = 0.15$, which was determined empirically [5]. It should be noted that pushing λ towards 0.0 will result in a model that favors calibration, while pushing λ towards 1.0 will result in a model that favors discrimination.

***N*-peptide model** The canonical 1-peptide model, sometimes called the 1-peptide rule, identifies all proteins which are adjacent to at least one observed unique peptide as present; however, this can lead to many FPs being identified as present. Since the mass spectrometry process is not perfect, the single peptide could easily be erroneously identified due to things such as noise in the mass spectrometry process or contamination. To help mitigate this problem, the 2-peptide rule was invented. This model works in the exact same way as the 1-peptide rule except that a protein must now be adjacent to at least two observed unique peptides to be identified as present. The proteins which are identified as present are then assigned the score of the lowest ranking of the top N unique peptides. Proteins which are identified as absent are given a score of 0. Of course, this idea can easily be extended to N-peptides; however,

there is a trade off: the larger N gets, the less FPs the model will identify as present, but it will become increasingly unlikely the model will identify a large number of proteins, simply because they will not be adjacent to have enough peptides.

ProteinProphet ProteinProphet [13] is an iterative model which computes protein-level confidence scores and then uses these protein scores to determine how degenerate peptide scores should be partitioned among the proteins which may have emitted them. While there are a few sensible ways to partition this information, ProteinProphet assigns weights proportional to the proteins peptide score, with the additional constraint that all the weights for a peptide must sum to unity. When the protein scores are first computed, the peptide scores are partitioned uniformly among proteins. The protein score for a particular protein, X_i is computed according to: $1 - \prod_{j \in Y_j:(i,j) \in E}(1 - s_{j,k}) \cdot w_{i,j}$, where $w_{i,j}$ is the weight assigned to peptide j for protein i. This iterative process continues until the protein-level confidence scores produced iteration i are within a user specified tolerance of the protein-level confidence scores produced at iteration $i - 1$. There are also additional pieces of information about a protein, such as the number of sibling proteins, that are taken into account to further improve the performance of the model. Sibling proteins are defined as the set of proteins which share peptides with this protein. One shortcoming of this model is that, due to the way protein-level confidence scores are computed, a single protein with many pieces of low scoring peptide evidence can receive a good score. Results in this paper were obtained with the ProteinProphet version bundled with `Trans-proteomic pipeline` [14] v5.2.0 Flammagenitus, Build 202011101623-exported (Linux-x86_64).

Fido Fido [5] is a probabilistic model that produces posteriors on proteins by computing the joint distribution over all proteins under consideration and then marginalizing this joint distribution to obtain protein posteriors. It has three free parameters: α, β, and γ, which are all the same for each protein and are $\in [0.0, 1.0]$. α represents the chance a protein emits a particular peptide, given that the protein is truly present. β represents the chance a truly absent peptide is observed due to noise. γ represents an independent and identically distributed (IID) prior on a protein being present in the sample. These parameters are optimized via a golden search [15] over `quality`. Results in this paper were obtained with an in-house implementation which utilizes the `EvergreenForest` inference engine [16].

Epifany Epifany [17], like Fido, is a probabilistic model that produces true posteriors on proteins. It also computes protein posteriors in a similar manner; however, the generative model for Epifany differs from that of Fido in that there is an optional regularizing prior on the number of proteins which may produce a peptide, and an optional greedy post-processor. Epifany has a similar parameter set; however, Epifany uses a grid search to optimize parameters, rather than a golden search. One advantage of using a grid search over a golden search is that many instances of the model with different parameter sets can be ran in parallel, as opposed to the 2^V (where V is the number of variables we are optimizing over) instances than can be ran at a time when using golden search. While it is possible to parallelize a golden search routine beyond this using branch and bound like techniques, it is still not as amenable to parallelization since the results of the previous iteration affect the results of the next iteration. Results in this manuscript were obtain with the Epifany executable packaged with version 2.6.0 Sep 30 2020, 11:01:01, revision: c26f752 of the OpenMS [18] software platform.

ProteinLP ProteinLP [19] is a protein inference model which uses a transformation of the joint probability distribution to express the protein and peptide probabilities in terms of linear combinations of one another. In this way, the authors present the protein inference problem as an optimization problem. The objective of this optimization problem is to produce a minimal set of proteins under the constraint that the peptide probabilities calculated by the model are within some tolerance of the peptide-level confidence score generated by the peptide search algorithm. The resulting linear program was solved with CPLEX. Results reported in this manuscript were obtained using an in-house `python` implementation.

1.4.2 Protein Inference Evaluation

Just as the protein inference problem is not yet solved, evaluation of protein inference methods is also still quite tricky.

1.4.2.1 Target-decoy

A necessary part of any protein inference experiment is the target-decoy-contaminant database. A target-decoy-contaminant database relates protein accessions to their respective labels. There are three disjoint sets in the database: targets, decoys, and contaminants. The target set consists a superset of proteins we expect to find in the sample. It is a superset because, in general, it is impossible to know *a priori* what proteins are truly present in the sample. If it was already known what was in the sample, there would be no need to perform the assay. Hence, it is typically the case that some of the targets are absent from the sample. The decoy set consists of proteins known to be absent from the sample. The contaminant set consists of proteins that may or may not be in the sample and are the result of proteins foreign to the proteome like keratin finding their way in to the sample. Though they are typically

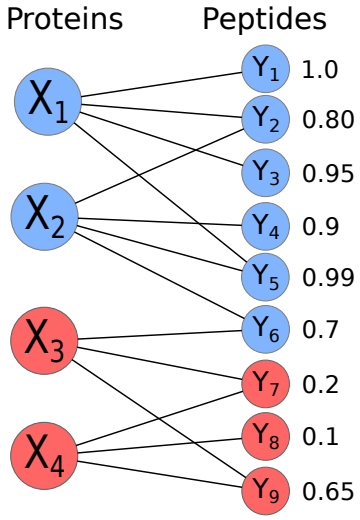


Figure 1.2: **Simple illustration of the PPG representation of a typical target-decoy database.** Target proteins X_1, X_2, share peptide Y_2, which was identified by peptide search with an 80% peptide-level confidence level. Decoy proteins X_3, X_4 share peptide Y_7, which was identified by peptide search with a 20% peptide-level confidence score. Target protein X_2 and decoy protein X_4 share peptide Y_6, which was identified by peptide search with a 70% peptide-level confidence score. Since Y_6 is adjacent to a target protein, we expect it to receive a higher peptide-level confidence score than peptides like Y_8, which are adjacent to only decoy proteins. Blue nodes indicate target proteins and target peptides. Red nodes indicate decoy proteins and decoy peptides.

of little biological interest, they can be thought of as another target in the sense that they may more accurately "explain" some set of observed peptides better than any of the proteins in the target set (likely because they are actually present in the sample). Hence, if we disregard this contaminant altogether, we may erroneously identify an absent target as present. A PPG representation of a typical target-decoy database is shown in Figure 1.2.

While this paper focuses on database oriented techniques for performing a peptide search, alternative techniques like *de novo* [20] are extremely powerful. Programs such as MS-GF [21] can be used to recover the most probable peptide sequence which would have resulted from the given spectra. *De novo* is particularly useful

when prior information about the contents of the sample is limited and so a target-decoy-contaminant database would need to be prohibitively large to avoid excluding potential present proteins [22]; however, if we are certain only a small subset of a small set of a couple hundred proteins can be targets, or when strong prior information about the sample contents is available, experiments can benefit from analysis via database [23]. It is essentially a trade-off between speed and accuracy: this is the "no free lunch" theorem from statistics [24].

1.4.2.2 Protein Inference Evaluation Challenges

To evaluate the performance of a protein inference method, the list of protein scores are evaluated under statistical metrics like area under the receiver operator characteristic curve [25] (AUC) and calibration error [26] (CE). AUC is a statistical test that measures how well an inference method is able to discriminate between true positives (TPs) (targets) and FPs (decoys) and is typically computed at a predetermined FDR or q-value threshold, while CE measures how closely the protein scores produced by an inference method resemble true probabilities, measured at an FDR threshold. More specifically, CE measures the squared error between the computed FDR and the empirical FDR, up to a threshold.

One problem with current metrics is that they inherently assume all target proteins are present in the sample; however, target-decoy-contaminant databases, in general, are not perfect. The target set, in general, is almost certainly a mix of present and absent proteins. A perfect target set would require prior knowledge on which proteins are truly present, but then why perform an assay if you have a perfect target set. Attempts have been made to help circumvent this problem altogether, like examining the abundance of messenger RNA to determine what is in the sample; however, this does not, in general, correlate well with whether or not a particular protein is found

in the sample [27]. Thus, since the labels in target-decoy-contaminant databases are not perfect, under current evaluation metrics, an inference method can erroneously identify an absent target protein as present and this will be seen as favorable. An inference method can identify an absent target protein as present for many reasons, one reason is that the method is utilizing the labels in the target-decoy-contaminant database too heavily. For example, consider an "Idealist" method that simply assigns the maximum possible protein score to any protein with a target label, while assigning the minimum possible protein score to any protein with a decoy label. This would cause the model to achieve maximum possible CE and AUC, but a method which gives scores only based on labels is inherently flawed by the fact the target set is flawed and thus, AUC and CE by themselves are not necessarily enough to determine a good method.

Due to this fact, it is quite difficult to objectively evaluate an inference method. Moreover, current metrics do not fully capture all the important qualities a good inference method should have. For instance, AUC is merely one way of measuring discrimination. Another way to measure discrimination is to separate the protein scores by target and decoy label, and then measure the difference in median value of these collections.

There has been significant research done to help compensate for the fact that targets are truly a mix of present and absent proteins. One technique, initially pioneered by Storey [28] involves estimating π_0, which represents the portion of all hypothesis under consideration that are truly null hypothesis. This method uses the assumption that null hypotheses are uniformly distributed and that most observations above a certain p-value threshold correspond to null hypotheses. The ratio of null hypothesis in this interval can then be used to estimate the overall number of null hypotheses. In a protein inference experiment, estimating π_0 is akin to estimating the percentage of

absent targets in the sample. For example, if we estimate π_0 to be 0.5, we expect roughly half of our targets to be absent. The π_0 estimate can then be used to estimate the PEP associated with each protein. This metric can then be used to perform more accurate inference. This idea is currently used in programs such as `qvality` [29] to asses the probability that a particular PSM is correct.

The choice of decoy set also greatly affects how well an inference method performs. In the best world, decoys and absent targets resemble one another; however, if the model is optimized for discrimination between targets and decoys and absent targets can clearly be distinguished from decoys, all target proteins, present and absent, can be given scores $\geq$ decoy proteins. This results in the method identifying *all* targets as present in the sample; a naive target-decoy approach would reward such a conclusion as many target proteins would be identified at a FDR of 0.0 instead of at a correct, much higher (*e.g.* 0.1) FDR. A Typical choice for generating the decoy database is to use the reversed target protein sequences; however, shuffling the protein sequences in the target set can also be used [30]. Another option is to use an entirely different organism as the decoy database.

1.4.2.3 The Hitchhiking Problem

The fact that targets are a mixture of present and absent proteins leads to a problem known as hitchhiking. Hitchhiking can occur when there is an absent target protein that shares evidence with a present target protein; hence, the absent target protein will be adjacent to a few pieces of high-scoring, degenerate peptide evidence. The major challenge with hitchhiking is that it is easier to construct an inference method which treats all shared peptide evidence as belonging to all proteins or does not take in to account sharing of peptides at all (*e.g.* parsimony) instead of solving a generalization of set-cover, which would allow different elements to be covered multiple

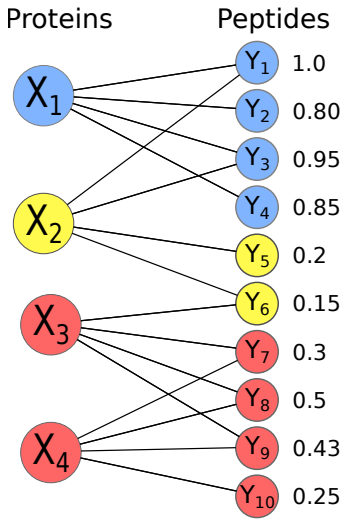


Figure 1.3: **An illustration showing a situation in which hitchhiking may occur.** The blue protein, X_1, is a present target and thus, is adjacent to many peptides which were identified by peptide search with high peptide-level confidence. Peptides Y_1 and Y_3 are identified by peptide search with a fairly high (95% or above) peptide-level confidence score, despite being adjacent to an absent target, the yellow protein X_2. This is due to the fact that these peptides are also adjacent to X_1, a present target. Since peptide Y_5 is adjacent only to an absent target, it is identified by peptide-search with a fairly low (20%) peptide-level confidence score. These shared high scoring peptides may cause an inference method to erroneously identify X_2 as present. Since X_2 and X_3 are absent, we expect this peptide to receive a fairly low peptide-level confidence score. Blue nodes indicate present target proteins and present target peptides. Yellow nodes indicate absent target proteins and absent target peptides. Red nodes represent absent decoy proteins and absent decoy peptides.

times (*e.g.* degenerate peptides could be handled on a case by case basis).

Eukaryote lysates, especially those from higher order organisms, are notorious for hitchhiking due to homology. In cell lysis, a whole cell is split open and examined. Hence, because in general it is not guaranteed the entire proteome is expressed in a cell at a time, there are many target proteins which will be absent. Contaminants may also cause hitchhiking; however, these are typically much less of an issue than absent target proteins. It is also possible to have a situation where a target shares evidence with a decoy, though this is less common, as is shown by Figure 1 in Elias

and Gygi's paper "Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry" [31] where a plot of frequency of shared tryptic peptides between the human proteome and the reversed protein sequences is displayed. In this table, it is shown that there is little overlap between the tryptic peptides found in the target and decoy protein sequence.

1.5 Gold Standard data

One way to evaluate an inference method under an accurate target-decoy-contaminant database is to use a protein standard dataset for evaluation. The samples analyzed for a protein standard dataset have been carefully prepared to contain only the proteins in the standard. Thus, the labels in the target-decoy-contaminant database can be considered ground truth and hence, any target proteins an inference method identifies as present are correct. In this way, protein standards allow an inference method to be objectively evaluated. Note that since all current evaluation metrics presented do not consider contaminant proteins when computing evaluation metrics, the target-decoy-contaminant database will be referred to as the target-decoy database for the remainder of the manuscript.

Hence, datasets like the ISB 18 mix protein standard [32] and the Sigma-Aldrich 49 standard [33] (commonly referred to as the UPS1 protein standard) provide us with a way to solve the circularity: we already know what is in the sample, thus, we can objectively evaluate how well we do. A strong challenge of these protein standards remains limited sequence similarity among target proteins. The protein standard used in the iPRG 2016 [34] study is intended to help solve this problem. In this standard, the present target proteins were engineered to deliberately share peptide evidence with absent target proteins. The absent target proteins are denoted as entrapment

proteins.

One challenge that remains for protein standards is that most standards lack sufficient complexity in terms of peptide degeneracy and difficulty. Having a sufficiently complex protein standard is especially important for investigating how well an inference method solves the hitchhiking problem.

CHAPTER 2 METHODS AND WORK

In this chapter, novel work is presented. This work consists of a new protein standard, an ensemble protein inference evaluation engine, and several novel protein inference methods.

2.1 ProteomeTools Hitchhiking Peptide Standard

In this section, a novel peptide standard is presented: the ProteomeTools Hitchhiking Peptide Standard (PHPP). This standard is intended to compliment currently existing standards by being deliberately complex in terms of shared peptides.

Properly solving the hitchhiking problem is something that any reliable protein inference method will need to overcome. How to best solve this problem is still an open question; however, it is impossible to answer this question without some way to check the answer. This standard aims to do just this; although, in general, engineering such a situation can be quite difficult. One must create a target set that contains present and absent targets with the added criteria that the absent targets should share some evidence with the present targets.

The target protein sequences in this dataset were deliberately designed to mix the benefits and hazards of shared peptides. In this way, it is impossible for either methods which disregard shared peptides altogether or methods which make parsimonious assumptions about shared peptides to perform extremely well on this standard. The

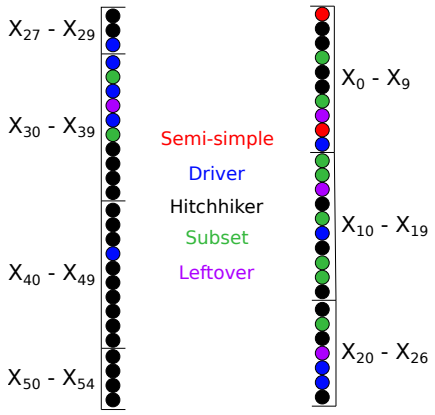


Figure 2.1: Graph relating proteins to one of the 5 protein classification types: Semi-Simple, Subset, Driver, Hitchhiker, or Leftover. While the majority of proteins are hitchhikers, the protein sequences for this standard were deliberately designed to have several proteins belonging to each different class, as depicted in the figure. The intent of this is to mix the benefit and hazard of shared peptides. The numbers next to the brackets indicate what proteins are contained in this bracket, *e.g.* the second protein in the X_0 - X_9 bracket is X_1. Node colors indicate the protein's classification. Purple nodes indicate proteins which are classified as Leftover, green nodes indicate proteins which are classified as Subset, blue nodes indicate proteins which are classified as Driver, red nodes indicate proteins which are classified as Semi-Simple, black nodes indicate proteins which are classified as Hitchhiker.

motivation behind how sequences were chosen is described in greater detail below.

2.1.1 Protein Sequence Design

There are three types of present peptides. "Unique" present peptides are found only in one protein. "Share" present peptides are found in two or more present proteins and no absent proteins. These peptides will prefer a less parsimonious model, so that both proteins can claim the evidence and be identified. "Choose" present peptides are found in at least one present protein and also in at least one absent protein. These peptides will prefer a more parsimonious model, so that only the present protein will be identified (hopefully).

Based on what proportion of peptide types a protein has, we can use this to classify

the proteins. A protein is classified as "Simple" if it has only Unique peptide evidence. A protein is classified as "Semi-Simple" if the majority of the peptide evidence is Unique peptides. A protein is classified as "Subset" if it is a present protein which does not have a majority of Unique peptide evidence and of the non-Unique peptide evidence, the majority are Share peptides. It is likely that more parsimonious models will tend to "explain away" this protein, as it is a subset of another protein. A protein is classified as "Driver" if it is a present protein which does not have majority Unique peptide evidence. Of the non-Unique peptide evidence, the majority are Choose peptides. A protein is classified as "Hitchhiker" if it is an absent protein which does not have majority Unique peptide evidence. Of the non-Unique peptide evidence, the majority are Choose peptides. A protein is classified as "Leftover" if it is anything not in the classes above. The proteins in this standard were deliberately designed to have representatives from each class, with the exception of Simple proteins as this standard is intended to complement other protein standards where these types of proteins are sufficiently covered. A graph relation of proteins to their respective protein classes is shown in Figure 2.1. The full theoretical PPG relations for the entire dataset can be seen in Figure 2.2a, Figure 2.2b, Figure 2.3a, Figure 2.3b, and Figure 2.4. Note that, because the dataset itself is one large connected component, the graphs were broken into sepset like structures for illustrative purposes. That is, the graphs were broken into clusters that contained as many nodes as possible while having as few edges between other such clusters as possible.

2.1.2 Sample Preparation

2.1.2.1 Synthetic Peptides

Under the umbrella of the ProteomeTools project [35], 153 peptides were individually synthesized on cellulose membrane following the Fmoc-based solid phase synthesis strategy using a purpose-built peptide synthesizer [36]. The crude peptides were cleaved off the membrane in 21 predefined pools of peptides and dried. Dried peptide pools were initially solubilized in 100% dimethyl sulfoxide (DMSO) to a concentration of 10 pmol μl^{-1} by vortexing for 30 min at room temperature. The pools were then diluted to 10% DMSO using 1% formic acid in liquid chromatography (HPLC)-grade water to a stock solution concentration of 1 pmol μl^{-1} and stored at $-20\,^{\circ}C$ until use.

2.1.2.2 Data Acquisition

The stock solution was transferred to a 96-well plate, diluted 10-fold with 0.1% formic acid in in liquid chromatography (HPLC)-grade water, and an estimated amount of 100 fmol of every peptide in a pool was subjected to liquid chromatography using a Dionex 3000 HPLC system (Thermo Fisher Scientific) using in-house-packed C18 columns. The setup consisted of a 75 $\mu m \times 2$ cm trap column packed with 5-μm particles of Reprosil Pur ODS-3 (Dr. Maisch) and a 75 $\mu m \times$ 40 cm analytical column packed with 3-μm particles of C18 Reprosil Gold 120 (Dr. Maisch). Peptides were loaded onto the trap column using 0.1% formic acid in water. We separated the peptides by using a linear gradient from 4% to 35% acetonitrile with 5% DMSO [37], 0.1% formic acid in water over 50 min followed by a washing step (60 min total method length) at a flow rate of 300 nl min^{-1} and a column temperature of $50\,^{\circ}C$. The HPLC system was coupled online to an Orbitrap Fusion Lumos mass spectrometer (Thermo Fisher Scientific). Each peptide pool was measured using a method triggering both an

HCD (NCE 28; Fourier transform mass spectrometry (FTMS)) and collision-induced dissociation (CID;NCE 35, ion trap mass spectrometry (ITMS)) fragmentation event on every detected precursor.

2.1.3 Data Processing

The Thermo RAW files were converted to mzML [38] using the `ThermoRawFileParser` [39] (version 1.2.0) software. Decoy sequences were generated by reversing the target protein sequences. The target-decoy database was created by concatenating the target fasta [40] file with the decoy fasta file. The mzML files were then searched against this target-decoy database using `Comet` [41] (version 2018.01, rev. 0) as part of the `Crux` [42] software package (version 3.2-0d57cff) to produce a pepXML [14] file containing a set of scored PSMs. These PSMs were then post-processed with `Percolator` (version 3.02.0, build Date May 30 2018 17:04:51), also a part of the `Crux` software package, to produce a pepXML file containing peptide-level confidence scores.

2.2 Best In Show: ensemble evaluation of protein inference engines

The `Best In Show` protein inference evaluation engine accepts several different protein inference methods and runs each method on a handful of protein standard datasets. The results from each method are then evaluated under several different metrics and ranked with respect to one another. Additionally, due to the fact that gold standard datasets are not representative of a real life dataset, we perturb the datasets in a special manner. Finally, the engine produces a ranking on all methods.

Clearly, discrimination and calibration are important metrics to measure: a good

model should accurately rank TPs above FPs and produce true probabilities; however, it is not clear what single statistical test adequately captures how well an inference method performs these tasks. An alternative to this is to use and ensemble of tests, all of which measure discrimination or calibration in a slightly different manner. With enough tests and enough datasets, it becomes impossible for a poorly performing inference method, like the Idealist model, to do well on all datasets across all metrics. The results of these metrics can be thought of as features of a particular method. The question then becomes how to best aggregate these features into a single, easily interpretable numeric value.

To get a better idea of how a method would perform on a general sample (*e.g.* a sample which is not from a protein standard) the engine also measures the degree to which a method utilizes the labels in the target-decoy database to make decisions about the model. This is achieved by randomly selecting a percentage of decoys in the gold standard target-decoy database, relabeling them as targets, and then investigating how the results of the model change with respect to the gold standard target-decoy database. The more drastically the results of a model change under the same observed data (*e.g.* identical sets of scored peptides) when utilizing an obfuscated target-decoy database, the more heavily a model utilized the target-decoy labels to fit parameters of the model. This can be thought of as something akin to a derivative: if we perturb the labels a small amount, how much does performance suffer. The gold standard target-decoy database consists of ground truth target and decoy sets, while this relabeled target-decoy database is referred to as the obfuscated target-decoy database. The notion of an obfuscated target-decoy database is somewhat reminiscent of the notion of entrapment-proteins used in the 2016 iPRG study [34], as such the decoys which have been relabeled as targets in the obfuscated target-decoy database will be referred to as entrapment proteins for the remainder of the

manuscript.

To ensure the end user has access to as little information about how the obfuscated target-decoy database is created as possible (with the intention of preventing them from using this information to somehow easily recover the gold standard target-decoy database), the percentage of decoys to be relabeled as targets is sampled from a uniform distribution over some interval $[\min, \max]$, $\min \in [0.0, 1.0]$, $\max \in [0.0, 1.0]$, where min is the user specified minimum percentage and max is the user specified maximum percentage (*e.g.* if the user specifies 0.1 as the minimum percentage and 0.2 as the maximum percentage, some random percentage $\in [10\%, 20\%]$ will be selected). Additionally, to prevent the inference method from examining the relabeled protein accessions in the obfuscated target-decoy database and using those to recover the gold standard labels, all the accessions in the dataset are renamed with randomly generated unique identifiers.

2.2.1 Ranking

The ranking scheme used in this engine is slightly different than typical ranking schemes like those found in packages such as NumPy. The ranks start at zero and ties are given equal weight for a rank. The next rank is then incremented by the number of items that tied for the previous rank. Say we have five students who took a test. Their scores are as follows: Student A, 95; student B, 60; student C 80; student D 80, student E 80. The rankings would be as follows: Student A, 0.0; student C 1.33; student D 1.33; student E 1.33, student B 4.0. In this way, the ranks can essentially be interpreted as the number of students that performed better than the student in question. Ties are not penalized as aggressively as they would be under other ranking schemes. For example, under SciPy's `rankdata` function, using the dense method the rankings are as follows: A: 0.0, B: 4.0 , C: 2.0 , D: 2.0, E: 2.0. So, we see that

the ranks for B, C, and D are higher under the ordinal scheme than they would be under our scheme, specifically due to the way ties are handled.

The result of some metrics are not a single numeric value, but instead are a tuple of numeric values. This is the case for metrics like the Kolmogorov-Smirnov[43] (K-S) test which reports a p-value as well as a test statistic. In this case, the metrics are first ranked by p-value and any ties are broken according to the value of the statistic. Additionally, to simplify ranking and interpretation of raw metric results, we ensure that for all metrics higher values are better by negating the result of any metric where lower valued indicate better performance (for instance, the lower CE a model has, the better, so it is negated when reported by the engine).

2.2.2 Evaluation Engine

The engine accepts a list of protein inference methods to be ran in the form of commands. By accepting the command to run the method, the engine can be agnostic of what language the method is written in. This also allows users to easily write wrappers for existing inference methods if they do not adhere to the interface specified by the engine. The only assumption made about these inference methods is that they accept the following arguments in this order: pepXML file [14], target-decoy database, and fasta file. Note that this does not imply the method need use all this information (some methods, like the 1-peptide rule, do not look at target-decoy labels or the fasta file), but is merely a way to establish an easy interface for the engine. The pepXML file contains the post-processed output of the peptide-search. The target-decoy file is a simple XML file that relates each protein accession under consideration in this experiment to exactly one of the labels definitely present (targets), maybe present (contaminants), or definitely absent (decoys). These labels were deliberately chosen to be representative of the fact that the engine is intended to utilize only protein

standard datasets. The fasta file is a plain text file that relates protein accessions to their amino-acid sequences.

The engine then runs each of the provided methods on each of the benchmark datasets, which are discussed in greater detail in 2.2.4. Once all the methods have been ran, they are then evaluated on several different metrics, discussed in greater detail in the following section. The results of these metrics are then all ranked against one another on a per dataset basis. These individual ranks are then accumulated on a per category basis for each dataset into a rank sum. Each method is then re-ranked by their rank sum on a per category basis, according to discrimination and calibration. These category ranks are then accumulated across all benchmark datasets for each method into a final cumulative rank sum. These cumulative rank sums are then ranked to produce the final table of rank based on cumulative rank sums. Additionally, the cumulative rank sum for each method is normalized by dividing the rank sum by the product of the number of models being evaluated and the number of categories being utilized (*e.g.* if there are 5 methods and 2 categories we would divide the cumulative rank sums by 10). The model with the lowest cumulative rank sum is deemed the winner. The lower this value is, the more performant the model is. This normalized cumulative rank sum can be thought of as the expected value that this model would perform worse than another randomly selected model, on a randomly selected metric, on a randomly selected protein standard dataset.

Note that although contaminants are reported in the inference results, even on the gold standard datasets, they are not assumed to be present nor absent. So any metrics which compute results based upon whether a protein is truly present or absent will ignore contaminants. This is because, although contaminants can be thought of as another type of target, and are important for ensuring higher quality peptide-search results, they are typically of little biological interest for most applications and

experiments.

2.2.3 Evaluation Metrics

In this section, the novel evaluation metrics utilized in the engine are described in greater detail. Note that there are a few metrics which are qualitatively similar in nature. When this is the case, the average of the ranks of these metrics are used in the results. Where applicable, version 1.5.4 of `SciPy` [44], version 1.19.4 of NumPy [45], and lmxl [46] version 4.6.1 were used.

2.2.3.1 Data perturbance based stats

Ideally, `Best In Show` would evaluate real-life performance on a real dataset (*e.g.* a sample of pond water); however, this is impossible as we do not have any way to truly know what is in the sample. Hence, we would like a way to examine how imperfect data affects the results of a method while still having some way to access the ground truth data. This is achieved by slightly perturbing the ground truth data in some way, and then examining how drastically the ranks of the results change. One way to do this would be to simply use the raw rank differential between the gold standard results and the obfuscated results; however, this is not terribly robust. A more robust alternative to this would be to use the rank differentials in some sort of statistical test. Since the metrics can be thought of as trials, a natural choice is to use a binomial test [47]. The binomial test has three parameters k, n, and p, which represent the probability that we would observe k successes out of n trails, where each trial has a hypothesized probability p of succeeding. For this test, n is the number of methods. This is because we interpret this method beating each other method as a trail (*e.g.* you have n chances to "win" against the other models). The parameter k is calculated as the number of wins this method had when using the obfuscated

target-decoy database. The parameter p is computed as $\frac{g}{n+1}$ where g is the number of wins this method has when using the gold standard target-decoy. The value $n+1$ is used to ensure $p \neq 1$.

These metrics, like others, are not perfect. Since they are inherently rank based, the result of the metric is heavily dependent on the performance of all of the other models being evaluated; however, there are enough published models which are known to perform well under current evaluation metrics on most inference standards that finding other well performing models to evaluate against should not be difficult.

Label Perturbation Response To investigate how heavily a method utilizes the labels in the target-decoy database, we measure the change in ranks of the evaluation metrics when using different target-decoy databases to fit the model. Specifically, the change in ranks of the evaluation metrics under the obfuscated target-decoy database with respect to the gold standard target-decoy database is measured. The larger the change in ranks, the more the model is penalized for it. For example, recall our Idealist model. Such a model would perform perfectly when utilizing the gold standard target-decoy database; however, when utilizing the obfuscated target-decoy database, all of the entrapment proteins receive the same score as all of the ground truth target proteins, and hence, would have poor discrimination and calibration, thereby leading to a large p-value.

Peptide score perturbation We also measure the degree to which slightly modifying the peptide scores affects how well the method performs. This represents a situation in which the results of the peptide-search are slightly corrupted by noise or imperfections in the mass spectrometry process. This peptide score perturbation is done in addition to the label obfuscation. Hence, for this metric, the result is

calculated between the model using the obfuscated target-decoy database and the model which utilized the perturbed peptide scores. The peptide scores are perturbed via rejection sampling with samples drawn from a Gaussian distribution with mean 0.0 and standard-deviation 0.05. If the given sample would perturb the score such that $s_j > 1.0$ or $s_j < 0.0$, the sample is rejected. Further, the Gaussian is parameterized in such a way that 95% of the samples are within ± 0.1.

Graph perturbation Another complexity seen in datasets created from general samples that are not typical of protein standard datasets is an imperfect target-decoy database. To simulate this situation (and subsequently evaluate how well a method would do when faced with this additional complexity) the adjacencies in the observed protein-peptide graph representing the target-decoy database are randomly added and removed with user specified probabilities. As with the peptide score perturbation metric, these graph perturbations are performed on the dataset with the obfuscated target-decoy database and compared to the results of the model under the obfuscated target-decoy database.

Shared Only Metrics Since proteins which have degenerate peptides are of particular interest, we also evaluate models under all metrics by solely examining how well the model scored the proteins with only shared peptides. This is to say that when evaluating results we disregard proteins which have any observed unique peptides. These metrics are denoted with a parenthetical "shared only" in the results tables. Since most of these tests are statistical in nature, care must be taken to avoid a small sample size. If a dataset has less than 10 target proteins or less than 10 decoy proteins which do not meet the aforementioned criteria, evaluation of these metrics is skipped for this dataset.

Entrapment Gain (Average and Maximum) This metric measures the change in ranks of the entrapment proteins when using the obfuscated target-decoy database as opposed to the gold standard target-decoy database. For example, say a method gives an entrapment protein a rank of 100 when utilizing the gold standard target-decoy database and a rank of 50 when utilizing the obfuscated target-decoy database. Then, it is clear that the model is heavily utilizing the target-decoy labels and therefore should be penalized for it since this is an indicator that the model relies too heavily on the often imperfect target-decoy database. The rank differential is calculated for each of the relabeled decoy proteins and the average difference in ranks as well as the maximum difference in ranks is recorded. Note that each both the average and maximum entrapment gain are recorded and ranked. These ranks are then averaged together to produce one final rank.

Two-Sample Kolmogorov-Smirnov test The two-sample K-S test is a non-parametric statistical test which measures discrimination. It determines whether or not two samples are drawn from the same distribution. A one-sided, two-sample K-S test is performed between the empirical cumulative density functions (CDFs) of the target protein-level confidence scores and decoy protein-level confidence scores. This test reports a p-value and a statistic, D, which represents the supremum of the set of distances between the CDFs.

Mann-Whitney U The Mann-Whitney U [48] (MWU) is a non-parametric statistical test which measures discrimination. Specifically, this test determines whether randomly selected samples from two different distributions are greater than one another. In this case, the empirical posterior distribution on target proteins is compared to the empirical posterior distribution on decoy proteins. That is, whether a randomly

selected target is expected to receive a higher score than a randomly selected decoy. Hence, empirical distributions representing the protein scores of target proteins and decoy proteins are built and used as input to the test.

L1 Calibration Error Measures the maximum squared deviation from the the $y = x$ axis in the calibration curve. The calibration curve is created by computing the empirical and computed FDR at various thresholds.

Non-parametric Cutout Index Th Non-parametric Cutout Index (npCI) is a non-parametric test that computes the likelihood that the given set of identified proteins are correctly identified. This is done by creating two distributions, referred to as "absent only" and the "leftover" distribution. The "absent only" distribution is an empirical probability density function (PDF) consisting of peptide-level confidence scores of peptides adjacent to only decoys (and hence, is static). The "leftover" distribution consists of the set of peptides adjacent to the set of proteins which the inference method identifies as absent. The two distributions are then smoothed using a Gaussian Kernel Density Estimator [49] (GKDE) and normalized. Finally, the similarity between the distributions is computed at various protein score thresholds and the metrics for the threshold at which these distributions are most similar is reported. There are a few choices for the similarity metric; however, in this paper, we use symmetric Kullback-Leibler (KL) divergence [50] and a two-sided, two-sample K-S test. We use a symmetric KL divergence and a two-sided K-S test because we are concerned only with the overall similarity between distributions, as opposed to how similar one distribution is to the other. The ranks of these test are then averaged together to produce one final rank for this test.

Sensitivity at Global FDR This metric measures the number of targets identified at a specified Global FDR threshold, a measure of discrimination. In this implementation of the engine, the default threshold is 0.05.

Local FDR Local FDR [51] measures the probability that next protein we accept as present is a decoy. Local FDR is a measure of discrimination. This is calculated by creating empirical PDFs for the target and decoy protein posteriors, smoothing them using a GKDE, and then taking the ratio between the densities at the given FDR threshold.

Incorrect Protein Differential The incorrect protein differential measures the difference in magnitude between the lowest scoring target and the highest scoring decoy. This is a measure of discrimination. A model which discriminates well should not score a decoy above a target, the larger this differential the worse the method is at discriminating.

Area Under Receiver Operating Curve The AUC measures how well a model is discriminating at a threshold. Note that while there is no good way to choose the evaluation threshold for this test, for most experiments, a threshold of 0.05 or less is typically sufficient (which is the default value used in this implementation of the engine). The numeric result of this test represents the probability that the classifier being evaluated will rank a randomly chosen TP instance higher than a randomly chosen FP one.

Compare Medians This metric measures the difference in median values of the empirical target distribution on posteriors and the empirical decoy distribution on posteriors. This is a measure of discrimination. A model which discriminates

well will score TPs and FPs such that the median of the distributions of their scores should be far from one another. By comparing the medians, this test is less sensitive to the tails of the distributions, as opposed to a metric which measures the difference in the average of the distributions.

Picked Metrics (AUC, CE) In addition to the typical FDR and q-value based metrics, there is an alternative metric, the Picked FDR [52]. The main difference between the Picked FDR and a typical FDR calculation is that under a picked FDR, where applicable, targets and their reversed decoy counterpart are treated as a single entity. Hence, when computing the picked FDR, if the target protein scores higher than it's decoy counterpart, it is counted as a target hit, otherwise it is counted as a decoy hit. This is in contrast to a typical FDR calculation which treats targets and decoys as their own independent entities. This idea is easily applied to the AUC and CE metrics, as they both depend on an FDR calculation. Hence, we can simply replace the typical FDR calculation with a Picked FDR calculation.

2.2.4 Benchmark Datasets

All datasets were reproduced from their respective RAW files. The Thermo RAW files were converted to mzML [38] files using version 1.2.0 of the `ThermoRawFileParser` [39]. The peptide search was then ran with version 2018.01, rev. 0 of `Comet` as part of version 3.2-0d57cff of the `Crux` [42] software package to produce a pepXML file which was then post-processed with version 3.02.0 of `Percolator`, also a part of `Crux`. This resulted in a final pepXML file which serves as the input to all inference models. For the peptide search performed on each dataset, a list comprising contaminants commonly found in other proteomics experiments was included in the target set [53].

18 Mix The ISB 18 mix dataset [32] is a protein standard containing 18 target proteins from various organisms such as bovines, rabbits, and horses. The data was searched against a target-decoy database containing the 18 proteins pipetted into the sample as well as the 15 contaminants proteins that were manually identified with high confidence as targets while the proteome of *H. influenzae* was used as decoys.

iPRG 2016 The protein standard used in the iPRG 2016 [34] study was designed to be intentionally complex in terms of degenerate peptides. This was accomplished by selecting and expressing a set of partially overlapping oligopeptides in the sample. The data was searched against a target-decoy database containing the fasta files in the PRIDE [54] repository as the target set, with the reversed target sequences serving as the decoy set. Additionally, the proteome of *E. coli* was included in the set of contaminants.

Yeast Part of a so called gold standard of protein expression in Yeast [55]. The data was searched against a target-decoy database containing the *S. cerevisiae* proteome as the target set, with the reversed target sequences utilized as the decoy set.

ProteomeTools Hitchhiking Peptide Standard The ProteomeTools Hitchhiking Peptide Standard is a peptide standard designed to deliberately mix the hazard and benefit of shared peptides. In this way, it is intended to compliment existing protein standards, most of which lack sufficient complexity in terms of shared peptides. It is deemed a peptide standard rather than a protein standard because the proteins represented in the standard are never actually synthesized, just the peptides. The data was searched against a target-decoy database containing the target proteins specified in the paper as targets, with the reversed target set serving as the decoy set.

Peptide-Shaker The peptide-shaker protein standard [56] was created to contain many proteins, be intentionally complex in terms of degenerate peptides, and be as biologically representative of a typical human sample as possible. The data was searched against a target-decoy database containing the proteome of *P. furiosus* as the target set, with the reversed target sequences serving as the decoy set.

UPS1 The target-decoy database consists of the 48 proteins specified by the Sigma-Aldrich 49 standard [33] as the target set, with the reversed target set utilized as decoy proteins.

2.3 New Methods

In this section, several novel protein inference methods are described. They are in no particular order and have been split into the following subcategories: N-peptide like models, set-cover like models, iterative models, probabilistic models, and linear Program (LP) models. Note that, as before, some models described in this section have free parameters, and thus, require some form of parameter optimization.

2.3.1 N-peptide model variations

As described in the previous chapter, the canonical 1-peptide model identifies proteins with at least one unique peptide as present. Present proteins are assigned the peptide-level confidence score of their lowest scoring unique N peptide.

N-peptide model (with shared peptides) This model assigns protein-level confidence scores in the same manner as the canonical 1-peptide model, except the requirement that the peptides be unique is removed. Hence, this model considers shared peptides when identifying proteins and assigning protein scores. This leads to

results that are quite liberal in comparison to canonical 1-peptide model. Proteins are assigned scores in the same manner as the canonical 1-peptide model.

2.3.1.1 N-peptide (Expectation) and variants

The canonical 1-peptide model identifies all proteins with at least one unique peptide as present; however, the manner in which the protein-level confidence scores are assigned in this case are quite optimistic and somewhat more pessimistic in the 2-peptide case. An alternative to this is to assign present proteins the average of the top N peptide-level confidence scores. If the protein in question has less than the specified N unique peptides, it is considered absent and assigned a score of 0.

With sharing This model removes the requirement that the protein must be adjacent to unique peptides. Hence, this model considers shared peptides when identifying proteins and assigning protein scores.

2.3.2 Iterative Models

These models iteratively update their protein scores in a manner similar to expectation-maximization [57] (EM). That is, they typically start with random parameter values, use these to produce protein-level confidence scores, and then from these scores, re-estimate parameters.

ProteinProphet (simplified) This simplified ProteinProphet model only performs the iterative partitioning of peptide scores among proteins until the resulting protein-level confidence scores are within a user specified tolerance of the previous scores. This is to say that this model does not utilize information about things such as

the number of sibling proteins. Results in this paper were obtained with an in-house `python` implementation.

***p*-norm** This model calculates the probability of protein presence based on the peptide score vector under various p-norms. The model can also optionally punish a protein for having a peptide below a given score threshold, τ. If the model does punish the protein, then the score of this peptide will be ignored when calculating the p-norm of this protein. The optimal τ value is found by doing a line search over `quality`.

***p*-norm (Dynamic Punishment)** This model is similar to the previous model in that it also uses the peptide score vector under various p-norms to calculate the probability of protein presence. In contrast to the previous model, the model punishes *shared* peptides below a threshold, τ. The optimal τ value is found by doing a line search over `quality`.

***p*-norm (Iterative)** This model operates in a manner very similar to Protein-Prophet, except p-norms are used to compute protein scores rather than the product of peptide scores. This model can also optionally punish proteins based on peptide with scores below some threshold τ. The optimal τ is found using a line search over `quality`.

2.3.3 Probabilistic Models

Probabilistic models are ones which use Bayes rule to produce true posterior probabilities on proteins. Since they produce true probabilities, these models tend to be quite well calibrated.

Fido-EM An alternative to using golden search for parameter optimization is to use an EM like optimization routine. The Fido-EM model does just this: it uses an EM like routine to optimize the free parameters for the Fido model. This method begins by randomly initializing α, β, and γ according to samples drawn from a uniform distribution. These parameters are then used to produce posteriors according to the Fido model, which are then used to re-estimate new parameters. These re-estimated parameters are then used to produce new posteriors. This proceeds in the same manner for a user specified number of iterations or until the difference in the approximated log-likelihoods of the joint distributions between iterations is within a user specified tolerance.

The parameter α represents the probability that a protein emits a peptide, given that protein was present. So, for each Y_j, the sum of $\Pr(X_i = 1)$ is computed and product of this and the peptide-level confidence score is taken. Then, we sum this value over all Y_j. This value is finally divided by the sum of the protein scores (allowing each X_i to be counted multiple times). Let $G[N]$ be the set of nodes adjacent to node N in G. The parameters are re-estimated as follows:

$$\alpha_{\mathrm{r}} = \frac{\sum_{\forall j} \sum_{i:G[Y_j]} \Pr(X_i = 1) \cdot s_{j,k}}{\sum_{\forall j} \sum_{i:G[Y_j]} \Pr(X_i = 1)}$$
$$\beta_{\mathrm{r}} = \frac{\sum_{\forall j} \sum_{i:G[Y_j]} \Pr(X_i = 0) \cdot s_{j,k}}{\sum_{\forall j} \sum_{i:G[Y_j]} \Pr(X_i = 0)}$$
$$\gamma_{\mathrm{r}} = \frac{\sum_{\forall i} \Pr(X_i = 1)}{|X_i|}$$

Generalized Cardinal Model The Generalized Cardinal Model is built on a set of beliefs any sane inference model should obey. These beliefs are primarily concerned with the cardinality of proteins and peptides. $Pr(X_i|M_i)$ represents the

probability of protein X_i being present given M peptides. It is required to be a monotonic increasing function because the more pieces of supporting evidence we accept, the more strongly we should believe our hypothesis. $\Pr(Y_j|N_j)$ represents the probability of peptide Y_j being present given N proteins. It is required to be a monotonic increasing function because the greater the number of present proteins which may have emitted this peptide, the more likely it should be that this peptide is observed. $\Pr(N_j)$ represents the prior on peptide sharing. It should be a monotonic decreasing function because as more hypotheses are supported by the same amount of evidence, we become more skeptical. $\Pr(Y_j)$ represents the prior on a peptide. $\Pr(X_i)$ represents a prior on proteins. Note that $\Pr(X_i)$ and $\Pr(Y_j)$ do not have restrictions on functions, due to the fact that these variables are binary.

The implementation in this paper is realized in a manner similar to Fido. There are six free parameters: α, β, γ, ζ, ι, and λ. α, β, and γ all have the same interpretation as Fido. A noisy-OR [58] function is used to parameterize $\Pr(Y_j|N_j)$ and $\Pr(X_i|M_i)$, while an exponential is used to parameterize $\Pr(N_j = n)$, and the peptide-level confidence score, s_j is used as the prior probability of peptide Y_j being present. Specifically:

$$\Pr(Y_j = 1|N_j = n) = 1 - ((1 - \beta) \cdot (1 - \alpha^n))$$
$$\Pr(X_i = 1|M_i = m) = 1 - ((1 - \zeta) \cdot (1 - \iota^m))$$
$$\Pr(N_j = n) = \lambda \mathrm{e}^{-\lambda n}$$
$$\Pr(Y_j = 1) = s_j$$

The six free parameters are then optimized via an EM like optimization procedure, similar to the one used to optimize the Fido-EM model.

α, β, γ are all re-estimated in the exact same manner as the EM routine for Fido, described above. The closed form equations for re-estimating ι, ζ, and λ are:

$$\iota_{\mathrm{r}} = \frac{\sum_{\forall i}(\sum_{j:G[X_i]} \Pr(Y_j = 1)) \cdot \Pr(X_i = 1)}{\sum_{\forall i} \sum_{j:G[X_i]} \Pr(Y_j = 1)}$$
$$\zeta_{\mathrm{r}} = \frac{\sum_{\forall i}(\sum_{j:G[X_i]} \Pr(Y_j = 0)) \cdot \Pr(X_i = 1)}{\sum_{\forall i} \sum_{j:G[X_i]} \Pr(Y_j = 0)}$$
$$\lambda_{\mathrm{r}} = \frac{\sum_{\forall j} \Pr(N_j = 1)}{|N_j|}$$

Measure Model The Measure Model is loosely based upon the idea of a protein algebra. With this algebra, it is possible to create a partial ordering on proteins. It should be noted that the following does not readily take into account other factors such as detectability of a peptide, prior probability of observing a particular protein in the sample, etc; however, even without this information, it should be possible to put a partial ordering on proteins. Further, for the purposes of creating this ordering, we are primarily concerned with immediate protein information, that is, for a particular protein, we only investigate: the protein in question, it's adjacent peptides, and any proteins adjacent to these peptides (which is necessary to determine shared peptides).

We define three classes of protein evidence: "Unique", "Shared", and "Unique and Shared". Proteins which have only unique peptides are typically trivial, the more peptide evidence there is, the more likely a protein is to be present. Hence, a protein with three pieces of unique peptide evidence would be ranked above a protein which only has two pieces of peptide evidence. Proteins which have only shared peptides are a bit more complicated. Resolution of a partial ordering in this case essentially comes down to the partition function used for peptide ownership: how much does the protein in question own each of it's shared peptides. This then reduces to the "only

unique peptides" case with fractional peptide counts. The final case is that a protein has unique and shared peptides. This case then reduces to a question of which protein has more peptide evidence.

While it may be questionable, from a biological perspective, to allow proteins to own part of a peptide, what we are trying to accomplish with this partial ordering is essentially the prior likelihood of the protein being present. This is to say that just because we allow the peptide *evidence* to be partitioned among the proteins does not in any way mean our model is explicitly entertaining the idea that a peptide was actually emitted from multiple proteins, though this is certainly possible.

Note that, in general, it will not be possible to create a total ordering on proteins based on this algebra. For instance, in Figure 2.5 since $s_1 < s_3$ and $s_2 > s_4$ it is unclear whether we should score $X_1 \geq X_2$ or $X_2 \geq X_3$. In this case, a measure function is used to decide which should be ranked higher.

2.3.4 Linear Programming methods

In this section, we detail models which frame protein inference as a LP. All LPs are solved using `CPLEX` version 12.10.0 [59]

Minimum Set-cover LP The minimum set-cover model is a protein inference model that attempts to explain all peptides in the dataset using as few proteins as possible. Two different implementations of this model are presented. In both models, a free parameter, τ is introduced. In the first version, τ represents a threshold at which all peptides whose peptide score is below this threshold are discarded and thus do not need to be covered. In the second version of this model, τ is interpreted as a minimum peptide score sum that must be achieved for a protein to be considered present. These are then formulated as an LP and solved using `CPLEX`. Results in this

paper were obtained using an in-house `python` implementation of both models.

To assign protein scores, the model is ran several times according to a golden search like parameter schedule. The results of each of these parameterisations are then ranked by `quality`. If a protein is identified as present in the set of results with the best `quality` metric, it is assigned the value of it's top scoring peptide. This peptide is then removed from consideration, this process can be thought of as a protein "claiming" ownership over this peptide. In the event that two proteins which are identified as present in the same parameterisation of the model attempt claim the same peptide, whichever protein is seen first gets the peptide. This routine proceeds down the list of results. If a protein is never identified as present in any parameterisation of the model, or if all of the peptides adjacent to this protein have been previously claimed, the protein is assigned a score of 0.0.

Peptide-centric graph cuts This model re-formulates the protein inference problem as a graph cut problem. The source is represented as present while the sink is represented as absent. The edge weights between the nodes are decided according to a parameterisation similar to that of Fido. There are three free parameters: α, β and γ each of which have a similar interpretation as they do in Fido. The resulting LPs are sent to `CPLEX` and solved as a max-flow problem which is used to find the graph cut [60]. It should be noted since this model is solving a graph cut LP, the initial results are a collection of binary indicator variables. Protein scores are assigned in the same manner as the Minimum Set-cover LP. Results in this paper were obtained using an in-house `python` implementation. A small example illustrating this model is shown in Figure 2.6.

2.4 Semi-Supervised

In this section, a novel protein inference method `Semi-supervised` is presented, and specified in detail.

`Semi-supervised` is an iterative method. Each iteration consists of two steps. The first step involves using a model to generate protein-level confidence scores, which are then used to create a set of positive and negative examples for training that machine learner. In the initial iteration, we use protein-level confidence scores produced by an external model. In subsequent iterations, we use a machine learner trained on the positive and negative examples. The set of positive examples are referred to as the high-confidence target (HCT) set and are a subset of the target proteins. Since they are known to be absent, the negative examples consist of the entire decoy set. The machine learner is then trained on the features of these positive and negative examples. Finally, the trained machine learner is used to produce protein-level confidence scores on the entire dataset. These scores are then used as input to the next iteration. This continues for a user specified number of iterations. Ideally, the scores produced by the machine learner should become more accurate as the process iterates; however, this may not be the case if the features used to represent the examples are not discriminatory.

Semi-supervised learning is not a novel idea; however, as far as we are aware, the application to protein inference in this manner is. Similar software, such as Percolator [3], solves the same problem, but at the peptide level rather than the protein level. The initial scores for Percolator come from the peptide search program used, typically the cross-correlation score which represents how well the theoretical spectrum matched the observed spectrum. These scores are then used to compute the q-value for each peptide. The target peptides below a given q-value threshold are

considered the positive examples for training the machine learner, a support vector machine [61] (SVM), while decoys are used as negative examples.

We use the protein-level confidence scores produced by the p−norm model as the initial scores for computing the HCT set. Just as Percolator does, the HCT set is created by computing q-values for all the target proteins and retaining those below a user specified threshold. In practice, it was found that a threshold of 0.05 tends to work well. We then use the HCT set as the positive examples for training a Convolutional Neural Network (CNN). A CNN was chosen specifically due to it's ability to aggregate information in a spatial manner, allowing greater utilization of information.

2.4.1 Protein Features

Creating feature vectors for individual proteins is one of the most important parts of this process. This is not a trivial task, especially since a protein's relation to the peptides it may have emitted is typically realized as a bipartite graph. Since graphs do not have an ordering on the adjacencies it becomes nearly impossible to come up with an ordered vector of features that represent that particular protein. The peptides are assumed to have scores associated with them, so it is possible to sort the peptides adjacent to a particular protein, but not all proteins have the same number of peptides adjacent to them. The feature vector could be padded with zeros or some other symbol but it then becomes unclear how this will be rectified by a machine learner. For instance, the network may learn to discriminate simply by counting the non-zero entries in the feature vector). Ideally, we would like some set of one or more functions, F_0, F_1, ..., F_m that takes some unordered set V and maps it to a numeric value such that for any ordering of V the value of F_j is the same.

A perhaps non-obvious choice for a such a function would be the p-norm function.

Let p be a real number, then the p-norm of a vector $\vec{x} = (x_0, x_1, \ldots, x_n)$ is defined as:

$$||\vec{x}||_p = \left(\sum_{i=0}^{n} |x_i|^p \right)^{1/p}$$

.

Since p-norms are equivalent to moments of a distribution, they provide a great deal of information about the distribution of peptide scores. For instance, the 0-norm counts the number of non-zero elements in the vector, while the ∞-norm gives the maximum value of the vector. It is known that with enough moments of a distribution, it is possible to recover the distribution itself [62]. Hence, from this perspective, p-norms seem like a good candidate for F_0. Since it has been shown that expectations on the top k scores in a peptide score vector are also quite informative [63], it is utilized as the second function in the set, F_2.

In addition to utilizing the peptide score vector under various p-norms as features, we also look at the peptide score vector at various τ thresholds. This modifies the peptide score vector to only consist of peptides whose peptide-level confidence score is $\geq \tau$ (note that by setting $\tau = 0$ we also include the unmodified peptide score vector).

Since the overall performance of a method can depend heavily on how degenerate peptides are handled, the partitioning scheme the model uses greatly effects the final results of the method; however it is not trivial to decide which partitioning scheme is the best for each particular dataset, or even a particular protein. It is almost certainly the case that in order to correctly solve multiple different datasets, multiple different partitioning schemes will be needed. One could try to train several machine learners at once, each of which uses a different peptide partitioning scheme to solve the inference problem; however, it is then unclear how to best intelligently aggregate the results of these networks. An obvious alternative to this would be to modify

the feature vector in such a way that it can utilize multiple partitioning schemes at once, and potentially even disregard schemes it deems uninformative. The answer to this is to create a 2D feature matrix for each protein, where the rows represent different partitioning schemes and the columns represent the p-norm of the protein's peptide score vector under that partitioning scheme. This feature matrix, along with a visualization of the first two convolutions that will be on it, is pictured in Figure 2.7. By using a CNN, it should learn the optimal subset of partitioning schemes for each dataset.

The set of features for all proteins then forms a 3D cube with proteins on one axis, p-norms on the second axis, and partitioning schemes the third axis.

2.4.1.1 Partitioning Schemes

Here, a handful of partitioning schemes used in the feature matrix for `Semi-supervised` are presented. While this is not an exhaustive list of all possible partitioning schemes, these are believed to be the most sensible and informative, from a protein inference perspective. Each of these partitioning schemes can be viewed as different possible solutions to the peptide degeneracy problem. Hence, from this perspective it makes sense that utilizing several different schemes in an intelligent manner would be beneficial. Note that the "all peptides", "vertex cover" and "only unique" partitioning schemes are static (that is, they do not change throughout inference) while the "greedy" and "ProteinProphet-like" partitioning schemes are updated each iteration according to the protein-level confidence scores computed in the previous iteration.

All peptides Under this partitioning scheme, proteins are assigned the peptide score vector consisting of the scores associated with all the observed peptides adjacent to this protein. This is the most permissive of all of the partitioning schemes and is

essentially equivalent to ignoring the peptide degeneracy problem altogether.

ProteinProphet-like The peptide score vector for this partitioning scheme is based on a weighted assignment scheme exactly like the one used in ProteinProphet. That is, peptide scores are partitioned among proteins in proportion to their protein-level confidence score. The higher the score of protein X_i relative to other proteins which may emitted this peptide, the larger proportion of the peptide score this protein will receive. This scheme is slightly less permissive. It attempts to solve the degeneracy problem in a more intelligent manner, essentially allowing multiple different proteins to emit the same peptide (which is indeed possible); however, this is predicated upon the model producing the scores being discriminative (which is not necessarily true).

Vertex cover Assigns peptide score vector according to our in-house vertex-cover algorithm. It takes all peptides with a score $\geq \tau$ as present and attempts to explain these peptides using as few proteins as possible. Hence, proteins which are considered present are assigned their adjacent peptides and proteins which are absent are assigned any of their remaining adjacent peptides. This scheme is less permissive than the previous two in the respect that this scheme no longer entertains the idea that multiple proteins may have emitted the same peptide.

Greedy The peptide score vector for this partitioning scheme is based on the protein scores from the previous iteration. The highest ranking protein is assigned all of it's peptides. These peptides are then removed from the set of peptides which may be assigned. Hence, the second highest ranking protein is assigned all of it's peptides which were not assigned to the previous protein. This proceeds down the ranks until each protein is assigned it's remaining unclaimed peptides. In the event the protein

in question has no peptides to claim, it receives a value of zero for this feature. This is equally as permissive as the "vertex cover" scheme due to the fact that it also does not entertain the idea that multiple proteins may have emitted the same peptide, but the routine governing how the peptides are partitioned is drastically different.

Only Unique Under this partitioning scheme, protein are assigned the peptide score vector consisting of the scores associated with all the unique observed peptides adjacent to this protein. This is the least permissive of all of the partitioning schemes, as it does not even entertain the idea of shared peptides.

2.4.2 Model

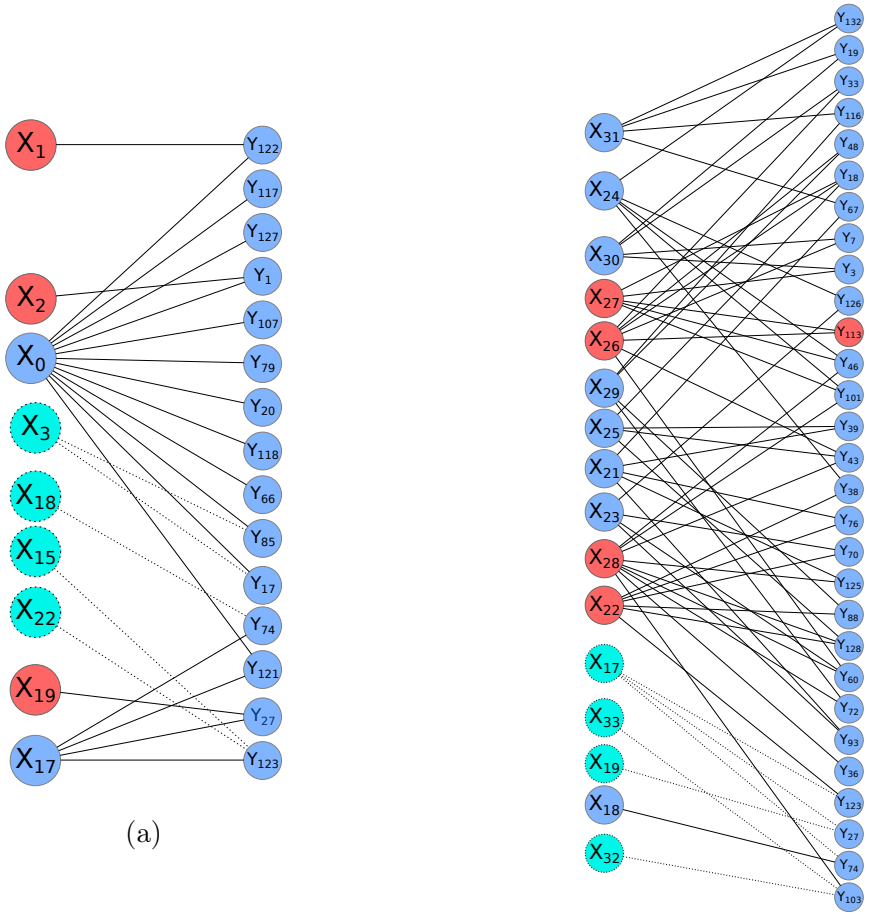
The model architecture is as follows: convolutional layer with 1x3 filter, 1x3 stride and 12 output channels; convolutional layer with 1x2 filter, 1x2 stride with 24 output channels; fully connected layer with 120 output nodes; fully connected layer with 84 input nodes and 1 output node. The output activation function is a sigmoid, while all other activation functions are rectified linear units (ReLUs). ReLUs were chosen due to the fact that they work quite well in practice [64]. Binary cross entropy (BCE) is used as the loss function and was chosen as it tends to work well for classification. For the convolutional layers, padding was turned off as it is unclear what effect this would have on the results. Further, it is not clear what the interpretation of artificially padding the feature would be. For instance, if zeros are chosen for the fill value in the padding, it is unclear what this is indicative of.

The values of p that are used to generate the feature matrices in this implementation are: 0, 1, 2, 3, 5, 10, 40, and inf. The values of τ that are used to generate the feature matrices in this implementation are: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 0.995, and 0.999. The last four τ values are intentionally close to one another

in value due to the fact that, for present target proteins, many of the peptide scores tend to be in this range.

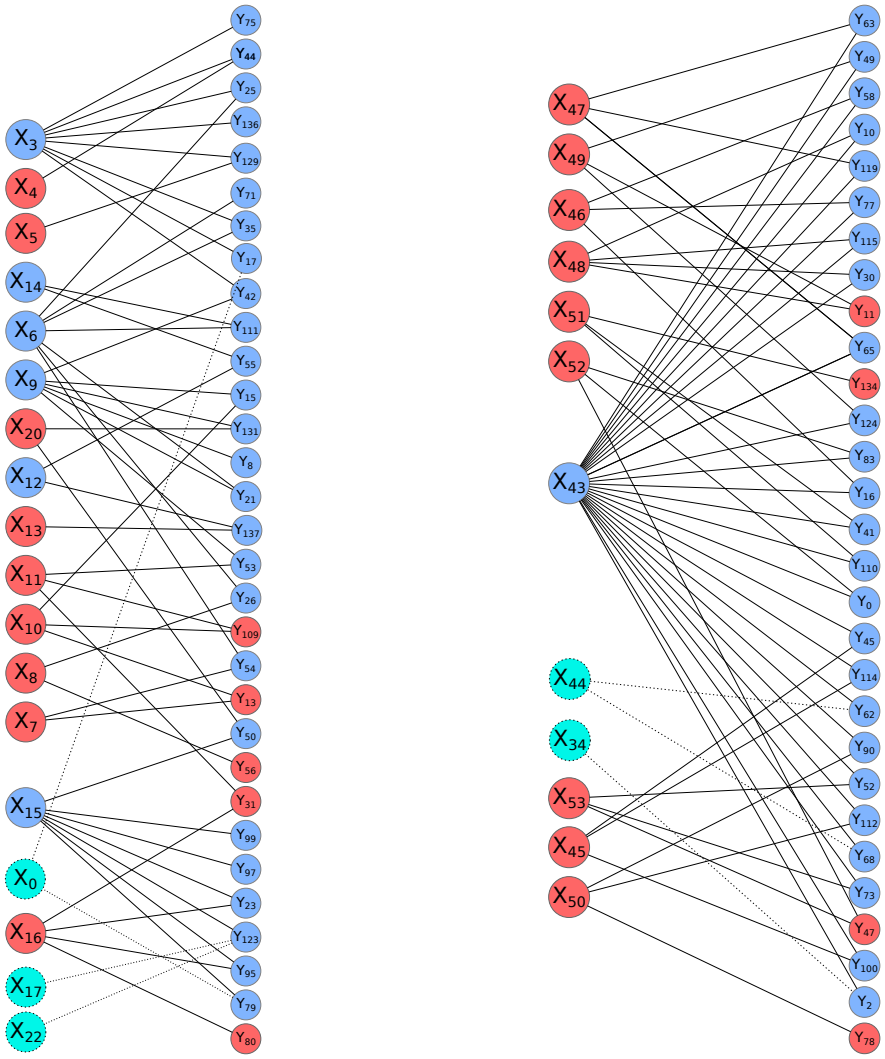
The optimal number of epochs is fit using k-fold cross-validation with 10% of the training set held out for validation and 100 epochs being the maximum number epochs possible. Specifically, the accuracy on the validation dataset is monitored throughout training, and when the validation accuracy begins to drop, the training is stopped. This process is then repeated k times with k different training and validation sets, and the minimal number of epochs is taken. In this case, the minimum observed number of epochs is taken rather than the average, as this helps prevent overfitting. It should also be noted that, in general, the quality of training instances varies (*e.g.* it is typically the case that not all decoys are created equal). A mini-batch size of 32 is used to train the network, while a mini-batch of size 16 is used when evaluating the validation set. For training the network, an initial learning rate of 0.001 with the Adam optimizer [65] is used.

A CNN was chosen because we believe the fact that the weights are shared adds additional discriminatory power, as it is allowed to "see" the peptide score vector from multiple views at once. It was also shown to perform better in general than a DNN using similar parameters and the same features (in this case, the feature matrix was simply flattened before being used as input to the network).



(a) (b)

Figure 2.2: Bipartite graph representation of a collection of target proteins found in the PHPP dataset. An edge between nodes indicates that the peptide is found in the protein sequence. Blue nodes signify present proteins or peptides. Red nodes signify present or absent peptides. Turquoise nodes represent proteins which are presented in another sepset. The dashed edges represent relations between peptides in this sepset and proteins found in another sepset. These figures represent proteins found in different sepsets of the dataset. **(a)** Proteins like X_0 are classified as Semi-Simple. We expect these proteins to push inference methods towards a more parsimonious scheme for handling shared peptides. **(b)** Proteins like X_{26} and X_{27} are classified as Hitchhiker proteins. We expect these proteins to push inference methods towards a less parsimonious scheme for handling shared peptides.



(a) (b)

Figure 2.3: Node coloring in this figure is the same as in Figure 2.2. These figures represent proteins found in different sepsets of the dataset. **(a)** Proteins like X_{14} are classified as Subset proteins. We expect these proteins to push inference methods towards a less parsimonious scheme for handling peptide sharing. This is because both X_{14} and X_6 should be identified; however, because X_6 has more pieces of evidence, a parsimonious inference method would identify only X_6 as present. **(b)** Proteins like X_{43} are classified as a Driver. Proteins such as this should push inference methods towards a more parsimonious scheme for handling shared peptides since it is only this protein which should be identified.

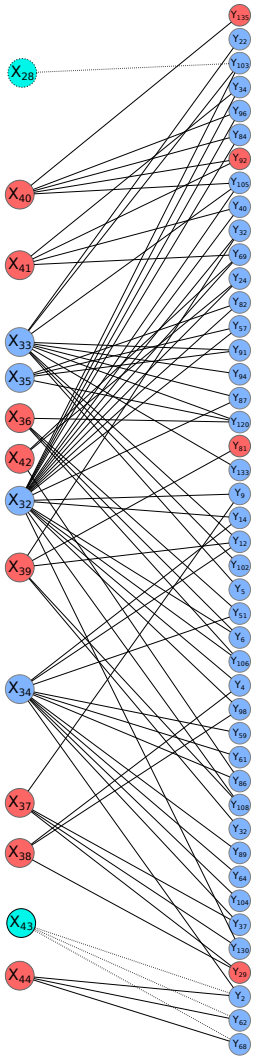


Figure 2.4: Node coloring in this figure is the same as in Figure 2.2. Proteins in this graph represent a different sepset of the dataset than all other figures. Proteins like X_{33} are classified as Leftover. These proteins do not belong under any other classification.

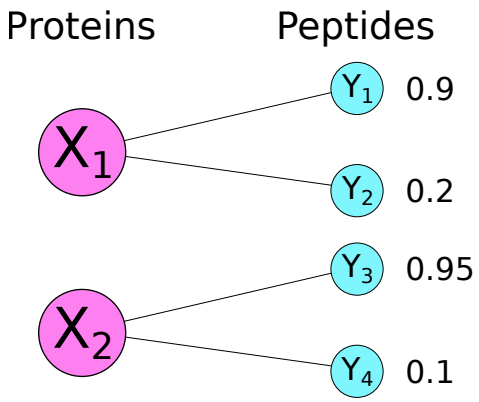


Figure 2.5: **An illustration of a situation on which a partial ordering on protein ranks is not possible.** X_1 and X_2 are two proteins with only unique peptide evidence. peptides Y_1, Y_2, Y_3 and Y_4 were identified by peptide search and assigned a peptide-level confidence score of 0.9, 0.2, 0.95 and 0.1, respectively. Though both of these proteins have only unique peptide evidence, since $s_1 < s_3$ and $S_2 > s_4$ it is unclear based on this information alone which protein should receive a higher rank.

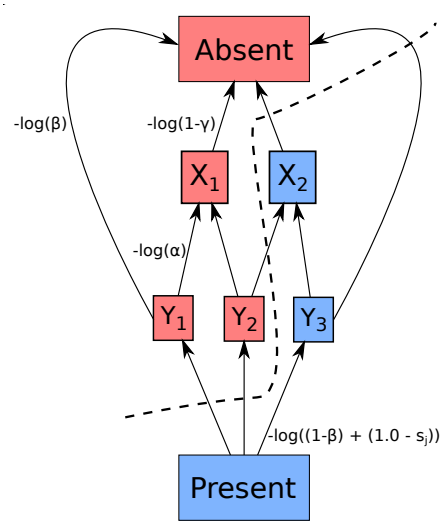


Figure 2.6: **An illustration of the Peptide-centric graph cuts model.** The blue protein, X_2, and the blue peptide, Y_3, are identified as present. The red protein, X_1, and the red peptides Y_1, Y_2 are identified as absent. The free parameters, α, β, and γ, are fit using a golden search like parameter schedule. Edges between peptide and protein nodes indicate that protein could have emitted that peptide. The dashed line indicates the optimal cut. s_j represents the peptide-level confidence score obtained during the peptide search for the j^{th} peptide. Blue nodes indicate present, red nodes indicate absent.

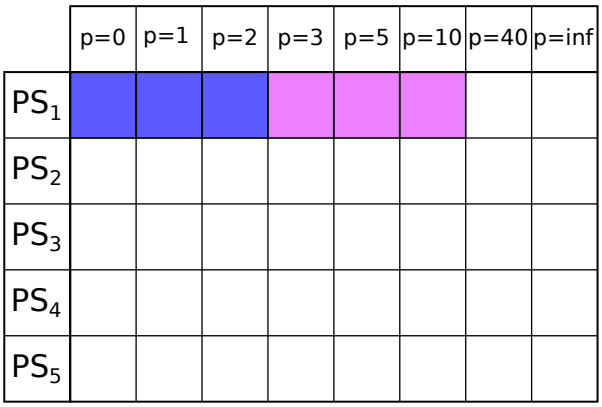


Figure 2.7: **An illustration of the matrix of features for an individual protein. The blue and magenta blocks represent the convolutional filter before and after a stride.** The rows are indexed by the partitioning schemes and the i^{th} partitioning scheme is denoted by PS_i, while the columns are indexed by the p-norms. A convolution between these feature values and the filter is performed, producing a single numeric value that will be evaluated the activation function and then fed forward to the next layer. The filter is then slid over by the stride amount, which, in this case, is the same size as the convolutional filter. Hence, the information in the feature vector is essentially downsampled across the convolutional layers. Note that although in practice there are columns for each value of τ paired with each value of p-norms, we only use the p-norms for illustrative purposes.

CHAPTER 3 RESULTS

In this chapter, some results are presented, including results of several models ran on the PTHPP and results from BIS on several models.

3.1 ProteomeTools Hitchhiking Peptide Standard

Fido, ProteinProphet, and several variations of the 1-peptide and 2-peptide models were all ran on the PHPP dataset and evaluated against the gold standard target-decoy database under the AUC and CE metrics. The results of these methods are shown in Table 3.1.

Method	**AUC**	**CE**
Fido, $p = 1$	0.29	0.001292
ProteinProphet	0.0005357	6.25e-05
`1-peptide (with sharing)`	0.002976	6.25e-05
`1-peptide (without sharing)`	0.1431	5.487e-05
`2-peptide (with sharing)`	0.29	5.544e-05
`2-peptide (without sharing)`	0.1429	4.072e-05

Table 3.1: **Table of results for various methods evaluated against the ProteomeTools Hitchhiking Peptide Standard**. All methods, where applicable, were allowed to use the ground truth target-decoy database to fit any parameters. To produce the metrics area under the receiver operator characteristics curve (AUC) and calibration error (CE), the method results were also evaluated against the ground truth target-decoy database. Each metric was evaluated at a threshold 0.05 and the resulting values were rounded to 4 significant digits.

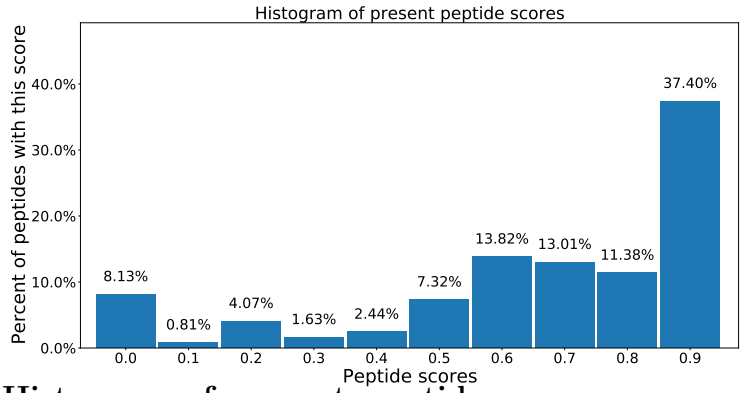


Figure 3.1: **Histogram of present peptide scores as a percentage of all present peptides.** There are 123 unique peptides with scores $\in [0.0, 1.0]$. Scores in the 0.0 bin are $\in [0.0, 0.1)$, scores in the 0.1 bin are $\in [0.1, 0.2)$ and so on. A majority (83%) of the observed peptides received scores of 0.5 or greater. Actual percentage values are displayed above each of the bars.

3.1.1 Analysis

While it is clear from Table 3.1 that none of the methods performed very well, `2-peptide (with sharing)` and Fido, $p = 1$ achieved the best AUC overall, while the `2-peptide (without sharing)` achieved the best CE overall. In contrast, ProteinProphet received the worst AUC and Fido received the worst CE. The fact that the models `2-peptide (with sharing)` and the `2-peptide (without sharing)` did so well (relative to other models) in different categories further underscores the fact that, in order to perform well on this particular dataset, a method cannot simply disregard shared peptides, nor treat them in a parsimonious manner: doing so results in poor performance in one or more metrics.

The fact that none of the methods were able to achieve perfect (or near perfect) AUC and CE scores, even when using the ground truth target-decoy database, is indicative of the complexity and difficulty of the dataset.

3.1.2 Peptide Coverage

An important metric to look at when evaluating the quality of results for a protein or peptide standard dataset is to look at the quality of scores on the present peptides. Ideally, the peptide-level confidence scores of the present peptides will be quite high for all of these peptides. Of course, this will not always be the case due to things like contamination and imperfections in the mass spectrometry process. As shown in Figure 3.1, this dataset has decent coverage with more than 37% of the peptides in the dataset having a peptide-level confidence score ≥ 0.9. Further, about 83% or so of the peptides have a peptide-level confidence score of ≥ 0.5. Though there are 10 or so peptides that have a peptide-level confidence score of 0.0, this is somewhat expected, as the mass spectrometry process is not perfect.

3.2 Best In Show

The results shown in Table 3.2. All presented results were produced on a machine with two Epyc 7351 processors with 32 threads each (64 threads in total) and 256GB of ram. Where applicable, version 1.5.4 of `SciPy` [44], version 1.19.4 of NumPy [45], and lmxl [46] version 4.6.1 were used.

`Semi-supervised` and the p-norm model tie for first place overall; however, `Semi-supervised` outperforms the p-norm model in terms of discrimination (0.0), taking first place while the p-norm model takes second place in discrimination. The p-norm model does slightly better in terms of calibration, coming in 4th place, while `Semi-supervised` takes 5th in calibration. The fact that they tie is likely indicative of some of the datasets not being difficult enough to properly utilize the different partitioning schemes for `Semi-supervised`. It should also be noted that with an NRS of 0.07692, `Semi-supervised` and p-norm significantly outperform the next

Methods	Cal.	Disc.	NRS	R
semi_supervised.py	4.0	0.0	0.07692	15m 14.988s
p_norm_model.py parallel	3.0	1.0	0.07692	1m 2.0731s
n_peptide.py 2 –consider-shared –expectation	5.0	2.0	0.1346	22.5422s
peptide_centric_em_graphcuts.py	0.5	7.0	0.1442	11m 19.1174s
n_peptide.py 2 –consider-shared –n-peptide	6.0	6.0	0.2308	22.5503s
fido_golden_search.py 1.0	13.0	5.0	0.3462	30m 51.2819s
n_peptide.py 2 –classic –expectation	2.0	17.0	0.3654	22.6914s
epifany.sh	11.0	9.0	0.3846	14m 13.5538s
measure_model.py	9.0	12.0	0.4038	27.7418s
fido_golden_search.py inf	17.0	4.0	0.4038	38m 37.0536s
p_norm_dynamic_punishment.py true	14.0	8.0	0.4231	12m 45.0867s
n_peptide.py 2 –classic –n-peptide	0.5	22.5	0.4423	22.5596s
n_peptide.py 1 –consider-shared –expectation	15.5	10.5	0.5	22.4593s
n_peptide.py 1 –consider-shared –n-peptide	15.5	10.5	0.5	22.575s
n_peptide.py 1 –classic –expectation	7.5	19.5	0.5192	22.6837s
n_peptide.py 1 –classic –n-peptide	7.5	19.5	0.5192	22.6501s
new_model_em_random_starts.py 3 1.0	25.0	3.0	0.5385	8m 46.6753s
p_norm_iterative.py	12.0	18.0	0.5769	17m 36.5389s
min_set_cover.py 0.001 IncludeAllPeptidesAboveTau	10.0	24.0	0.6538	9m 51.8765s
protein_prophet_simplified.py 0.001	22.0	15.0	0.7115	14m 14.4738s
fido_em_random_starts.py 5 inf	21.0	16.0	0.7115	15m 36.9453s
fido_em_random_starts.py 6 1.0	24.0	13.0	0.7115	11m 19.7434s
new_model_em_random_starts.py 3 inf	23.0	14.0	0.7115	9m 52.1816s
min_set_cover.py 0.01 ConstrainSum	18.0	21.0	0.75	6m 25.8321s
protein_prophet_wrapper_from_evaluator.sh	20.0	22.5	0.8173	1m 18.2413s
protein_lp.py	19.0	25.0	0.8462	2m 8.7807s

Table 3.2: **Table of results produced by the Best In Show protein inference evaluation engine.** Presented results are obtained by running each method with ten different obfuscated target-decoy databases, evaluating each of them under the gold standard target-decoy database and averaging their ranks. The value in the **Method** column represents the command issued to run the inference method. BIS has several metrics it uses to evaluate protein inference performance, each of which measures either how well calibrated the method is or how well it is discriminating between TPs and FPs. The value in the **Cal.** column represents the overall calibration rank this method received. The value in the **Disc.** column represents the overall discrimination rank this method received. The value in the **NRS** column represents the value of the method's rank sum (*e.g.* calibration rank + discrimination rank), normalized by the size of the table. The value in the **R** column represents the total time this method took to perform inference on all data sets.

best model, N-`peptide (expectation, with shared peptides)` model which has an NRS of 0.1346. Hence, `Semi-supervised` and p-norm perform ≈ 1.8 times better than N-`peptide (expectation, with shared peptides)`.

While the current parameter re-estimation scheme for the EM models does not seem to optimize the parameters to performant values, a better parameter re-estimation scheme could certainly be devised that should work better. An alternative optimiza-

tion method, such as golden search could be used; however, a naive version of this may be prohibitively computationally expensive for the GCM as it has six free parameters and golden search scales exponentially with the number of variables to be fit.

There are several models present in the results which are slight variations on the `1-peptide` model, all of which are ran via the `n_peptide.py` script. The canonical `1-peptide` model identifies all proteins with at least 1 unique piece of peptide evidence as present, and all other proteins as absent. This model is represented by: `n_peptide.py 1 -classic -n-peptide`. The 1 indicates how many pieces of peptide evidence should be considered, the `-classic` flag indicates that shared peptides should not be considered, while the `-n-peptide` flag indicates that the peptide-level confidence score of the N^{th} highest scoring peptide should be assigned to the protein if it is identified as present. The `-consider-shared` flag is mutually exclusive to the `-classic` flag and indicates that shared peptide evidence should be considered. The `-expectation` flag is mutually exclusive to the `-n-peptide` flag and indicates the protein-level confidence score assigned by this model will be the mean of the top the top k peptide scores.

3.3 `Semi-Supervised` Lysate Dataset Performance

In this section, the performance of `Semi-supervised` on several real world datasets, against several other models, is presented. All of these datasets were derived from a lysate. This means the the entire cell or organism was digested and ran through the mass spectrometer. Since the entire proteome of the organism is not likely to be expressed at the same time, it is likely the case that at least some proteins under consideration are absent. Each dataset was searched against a target-decoy database consisting of proteome of the species as the target set with the decoy

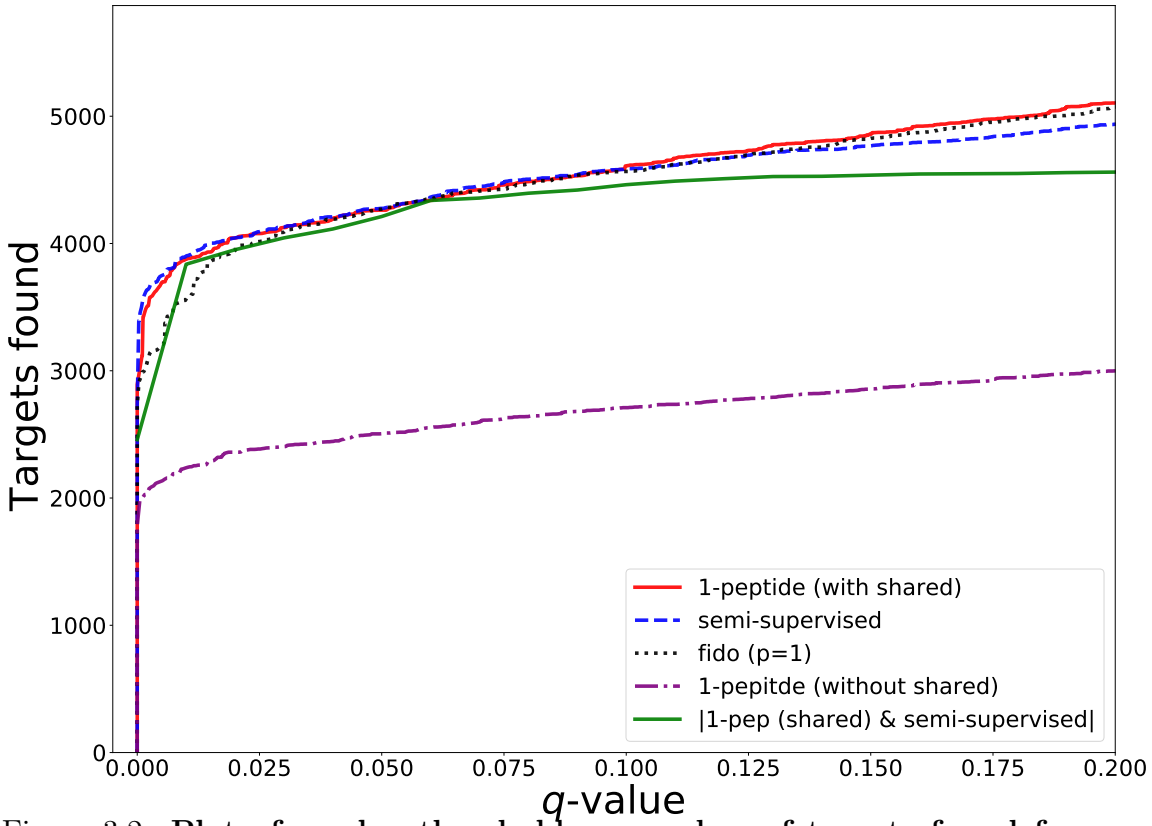


Figure 3.2: **Plot of q-value threshold vs number of targets found for various models on a *C. elegans* dataset.** The solid green line is the intersection of the target sets found by `Semi-supervised` and `1-peptide (with shared`. Although their performance appears somewhat similar, the target sets identified at various thresholds are quite different.

set consisting of the reversed target sequences. The receiver operator characteristic (ROC) curves are shown in Figures 3.2, 3.3, 3.4, 3.5, and 3.6.

C. elegans The *C. elegans* dataset [3] is a dataset derived from the lysate of a *C. elegans*. The data was searched against the target database included with the data, with the decoy database being composed of the reversed target sequences.

S. cerevisiae The *S. cerevisiae* dataset [3] is a dataset derived from the lysate of a *S. cerevisiae*. The data was searched against the target database included with the data, with the decoy database being composed of the reversed target sequences.

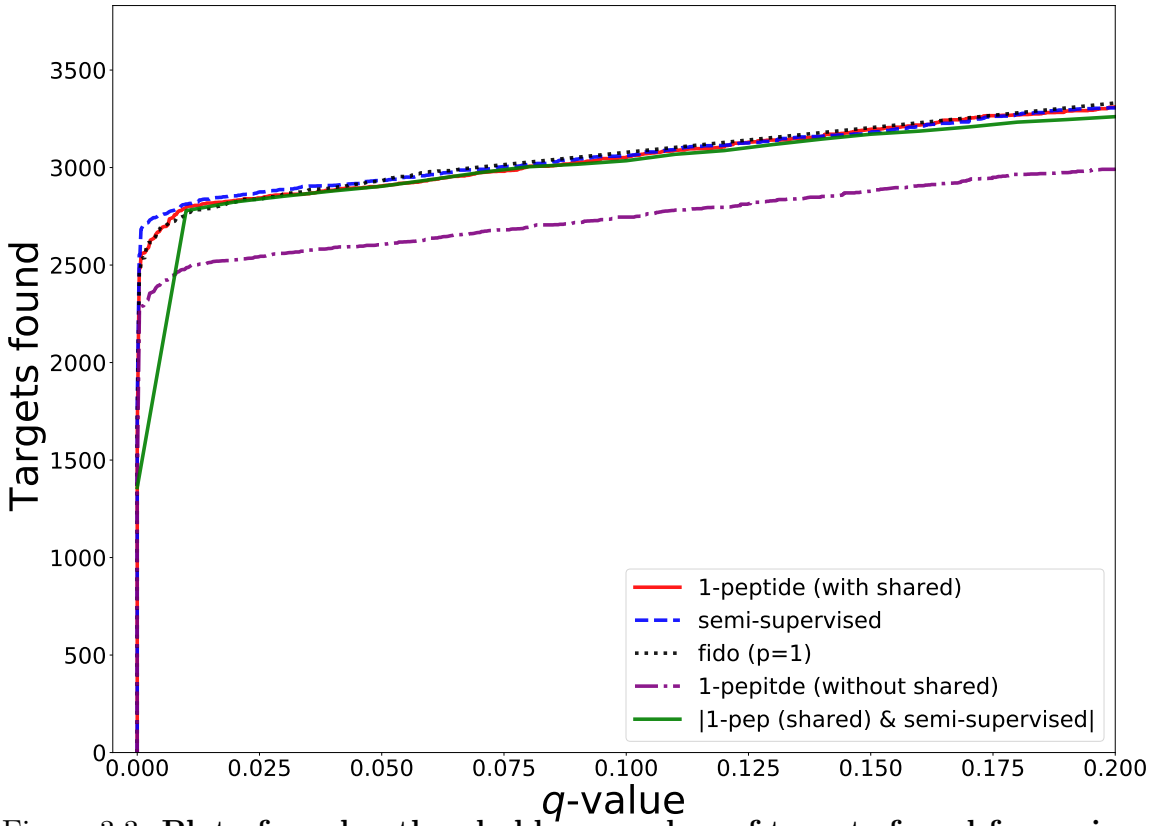


Figure 3.3: **Plot of q-value threshold vs number of targets found for various models on a *S. cerevisiae* dataset.** The solid green line is the intersection of the target sets found by `Semi-supervised` and `1-peptide (with shared)`. Although their performance appears somewhat similar, the target sets identified at various thresholds are quite different.

HumanMD The HumanMD dataset [55] is a dataset derived from the lysate of a dataset derived from Human Medulloblastoma daoy cells. The data was searched against the provided target-decoy database. The target set consisted of the longest transcription of the human proteome, retrieved from [66].

HumanMD (Trembl) The HumanMD (Trembl) dataset contains the same data as the HumanMD; however, for this dataset, the Trembl [67] protein database was used as the target set, with the reversed target sequences serving as the decoys.

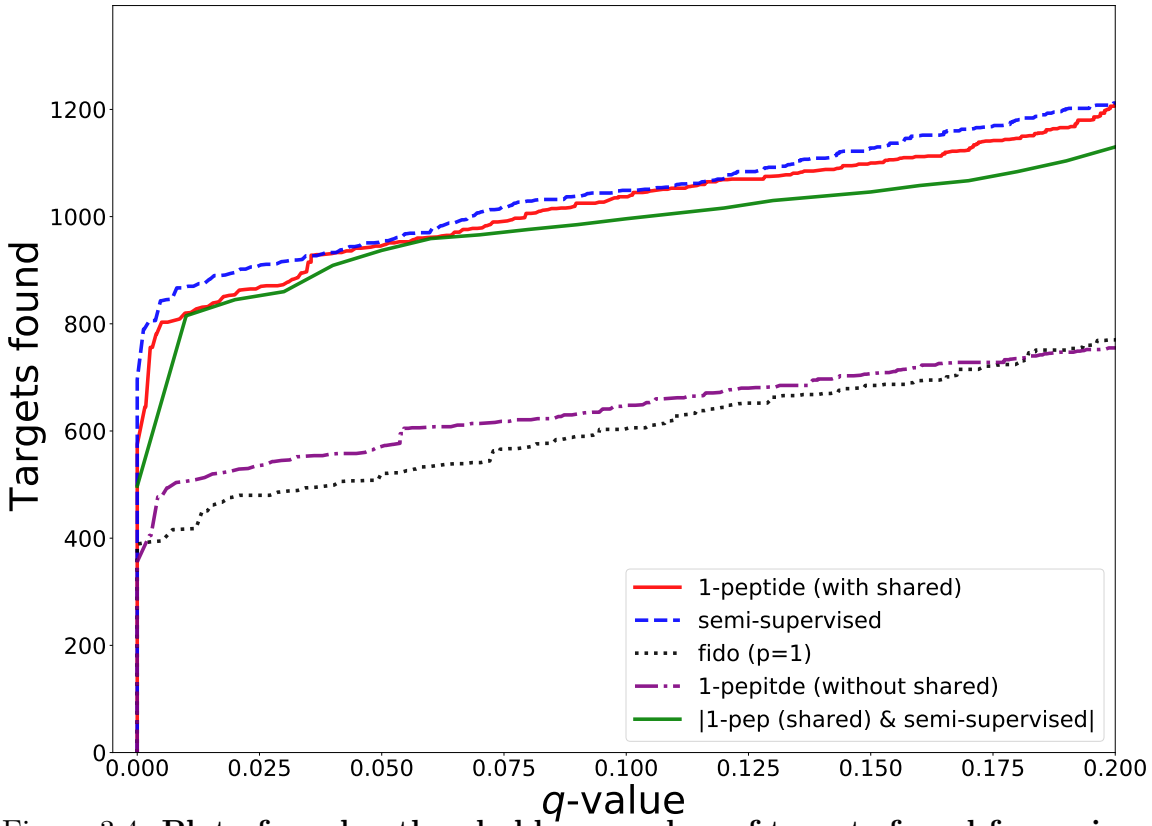


Figure 3.4: **Plot of q-value threshold vs number of targets found for various models on a dataset derived from a human medulloblastoma tumor.** The solid green line is the intersection of the target sets found by the `Semi-supervised` model and the `1-peptide (with shared)` model. Although their performance appears somewhat similar, the target sets identified at various thresholds are quite different.

HumanEKC The HumanEKC dataset [68] is a dataset derived from the lysate of a human embryonic kidney cell T293. This data was searched against a target-decoy database with the Trembl protein database composing the target set and the reversed target sequences composing the decoy set.

Though `Semi-supervised` and `1-peptide (with shared)` appear to perform similarly, the receiver operator characteristic curve composing their intersection shows that the target set they identify is different, especially at extremely low q-values. Further, `Fido` and `1-peptide (without shared)` perform much more conservatively. Note that Fido is not pictured in Figure 3.5 because it did not produce

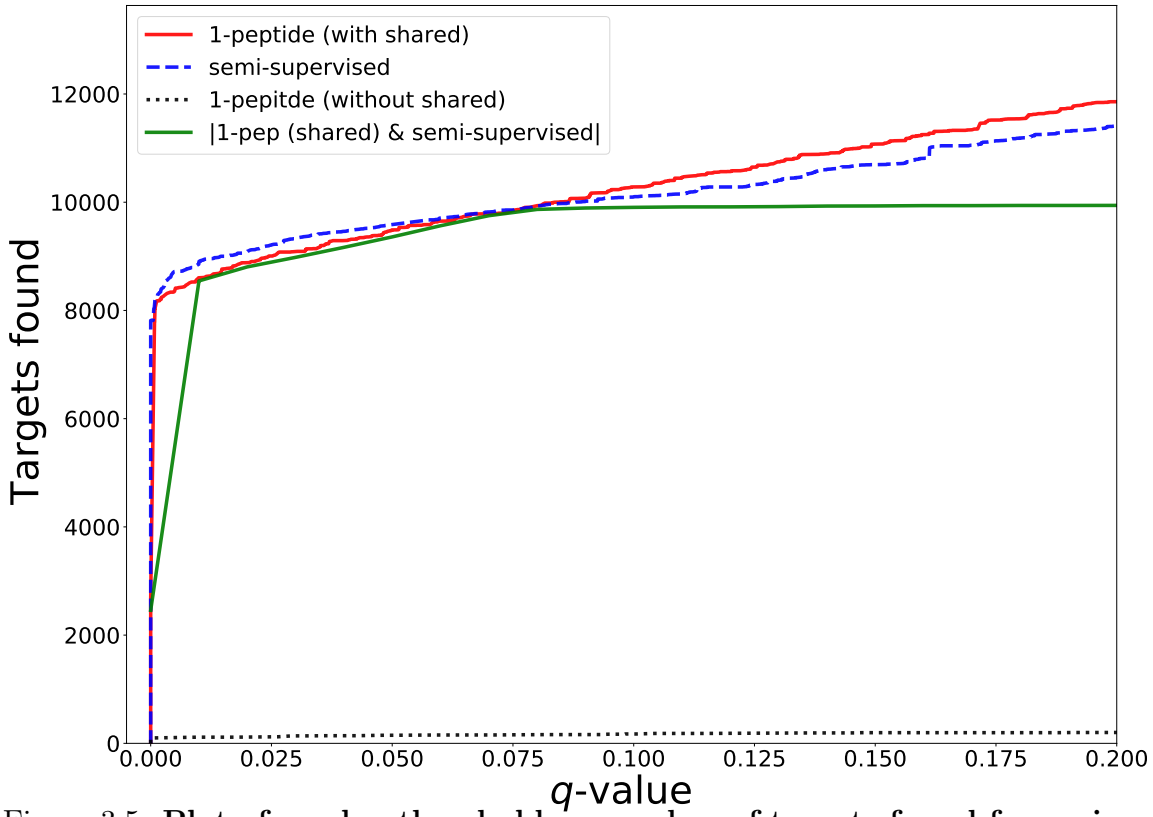


Figure 3.5: **Plot of q-value threshold vs number of targets found for various models on a dataset derived from a human medulloblastoma tumor.** The solid green line is the intersection of the target sets found by `Semi-supervised` and `1-peptide (with shared)`. Although their performance appears somewhat similar, the target sets identified at various thresholds are quite different. Note: Fido is not pictured as was not able to find reasonable parameter values.

an output due to unreasonable parameter values.

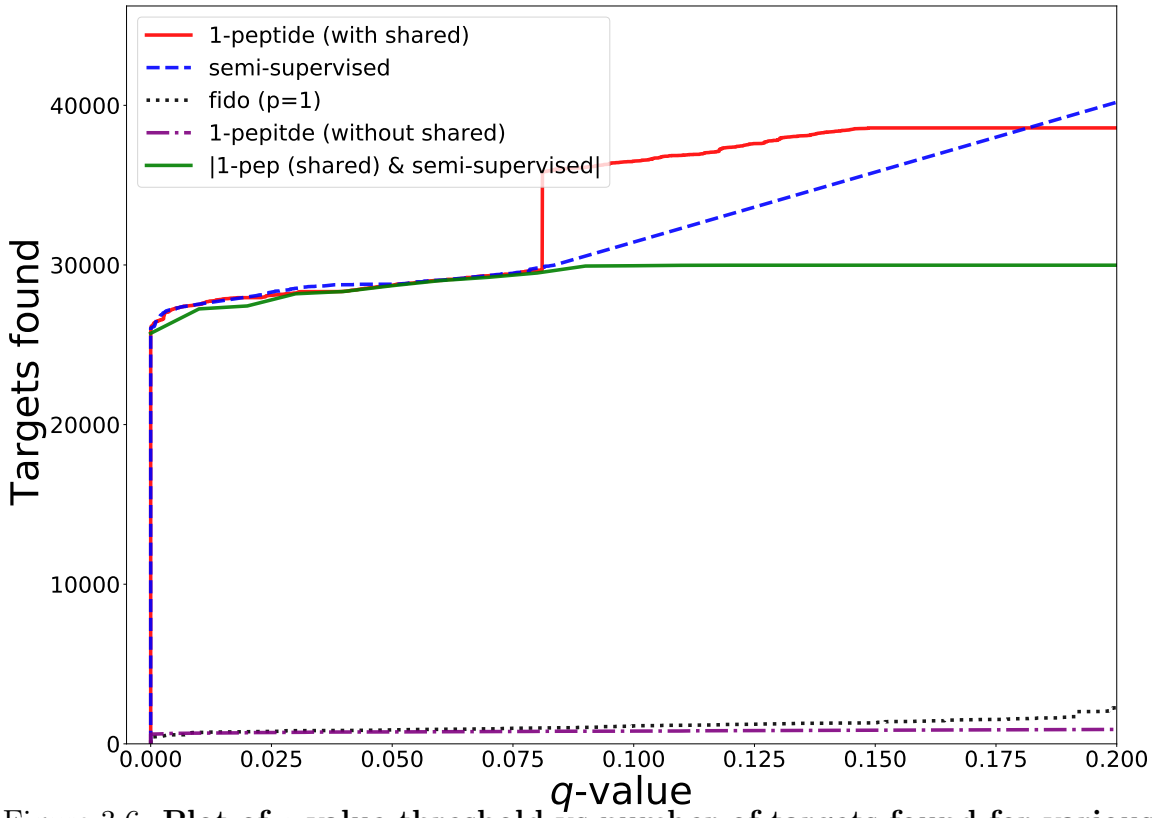


Figure 3.6: **Plot of q-value threshold vs number of targets found for various models on a dataset derived from a human kidney cell.** The solid green line is the intersection of the target sets found by the `Semi-supervised` model and the `1-peptide (with shared)` model. Although their performance appears somewhat similar, the target sets identified at various thresholds are quite different.

CHAPTER 4 DISCUSSION

Protein inference is an important but difficult problem. This is largely due the fact that current evaluation metrics rely on an imperfect target-decoy database. This imperfection results from the fact that the current evaluation metrics assume the target set of proteins are all present in the sample, when in fact they are typically a mix of present and absent proteins. Moreover, there are many ways of measuring discrimination and calibration and metrics like AUC and CE do not adequately capture all the quantitative properties we believe a good inference method should have. Moreover, AUC and CE can easily be cheated by assigning target proteins a score of 1.0 and decoy proteins a score of 0.0.

One way around these often imperfect target-decoy databases is to use a protein standard dataset. Since these samples are carefully prepared to contain exactly the proteins specified in the target set, we can be significantly more confident in the quality of the target-decoy database.

4.1 ProteomeTools Hitchhiking Peptide Standard

While all of the currently existing protein standards are important for validating and sanity checking models, not all such datasets have so many shared peptides nor chances for hitchhiking to occur as they would in lysates of higher order eukaryotes. The PHPP aims to address this problem by being intentionally complex by balancing

shared peptides and chances for hitchhiking. Further, the target protein set was chosen in such a way that a method cannot use parsimony or disregard shared peptides altogether (*e.g.* this dataset is not setup to always penalize nor reward hitchhiking). In this way, this dataset incentives algorithms to solve the set-cover like generalization that exists in proteomics at the protein level. It is clear from the results that the PHPP has sufficient complexity to trick even the current, best inference methods under current metrics.

4.2 Best In Show

Though the results for BIS are quite robust, there are of course shortcomings to this methodology. How well a model appears to perform overall is inherently dependent upon the performance of the other models it is competing against, hence, a user could intentionally make their model look good by including a lot of poorly performing methods; however, such a model is not likely to stand up to the scrutiny of the community. Further, by evaluating the methods in this manner, it is nonparametric. Hence, there is no chance of a circular 'rock-paper-scissors' situation happening (*e.g.* we will always be able to impose a total ordering in this manner).

There is also an interesting phenomena that occurs when a handful of similar methods are evaluated together: the qualitatively similar models appear to perform worse. This is likely due to the increased number of ties that occur, thereby overinflating the rank sum of the qualitatively similar methods.

While Epifany loses to Fido in terms of discrimination, this not due to the fact that Epifany performs worse in terms of AUC. Rather, it is a result of Epifany losing out to Fido in other discrimination metrics. This indicates that AUC alone is not a robust representative of how well a model discriminates in general.

Even though we go to significant lengths to disallow protein inference methods from gaining access to any useful information about the ground truth data, there is a possibility that a user could cleverly run a method many times in an attempt to effectively scam the obfuscated target-decoy database and recover the set of ground-truth targets. While this is certainly a possibility, it would be prohibitively time consuming for all but the smallest datasets.

Though the notion of using an obfuscated target-decoy database for inference evaluation is cheap, effective, and seemingly robust, an alternative methodology to this would be to augment the target-decoy databases with different types of proteins (like those detailed in 2.1) before performing the peptide search. In this way, ground truth datasets could be augmented with absent target proteins to more closely emulate a real-world biology experiment. To do this locally in an online manner would be a bit tricky as the user would need access to the RAW files, and the peptide-search would have to be performed on the fly; however, with disk space becoming increasingly cheap, processors becoming more powerful, and peptide-search and post-processing algorithms becoming increasingly efficient, doing this is not out of the question.

Another interesting metric that could be incorporated into BIS would be to examine how the method performs with respect to different levels of obfuscation. For example, two different obfuscated target-decoy databases could be constructed: one with 20% of the decoys relabeled and one with 40% of the decoys relabeled.

4.3 Semi-supervised

The fact that `Semi-supervised` performs better in terms of discrimination than the p-norm model overall indicates that different partitioning schemes are beneficial for properly solving different datasets in terms of discrimination. While this method

only utilizes a small handful of partitioning methods, the model can easily be extended to include any novel partitioning schemes. The tie for first place overall between p-norm and `Semi-supervised` could be due, at least in part, to `Semi-supervised` overfitting. This overfitting could likely be solved by reducing the number of epochs used during training, though the best way to do this is still unclear. Further, it should also be possible to use software such as `qvality` to produce well-calibrated scores for `Semi-supervised`, thereby causing giving it a significant win over p-norm in terms of both discrimination and calibration.

The design of the features for `Semi-supervised` makes them easily extensible: new features could either be appended to the end of the feature matrix, or they could occupy a new axis in the features. For instance, additional peptide level information such as the mass difference between the observed and theoretical peptides, the charge state of the spectra in the PSMs, hydrophobicity information, detectability of the peptides [6], *etc.* could be easily incorporated.

Another feature that could be easily incorporated is rank based information about other proteins. This information is not currently utilized in any meaningful manner. For example, information about the current protein-level confidence score, as well as the rank of the protein in question could be useful, as this protein would then have an idea of how good it is overall. Further, information about the protein-level confidence score and rank of the next few proteins that are better and worse than this protein would also be beneficial, as this would signify how much better or worse this protein is relative to other proteins. An alternative way to incorporate this information would be have two networks, one that uses the feature tensor from the previous iteration, and one that uses the feature tensor for the current iteration. These results could then be reconciled by a final node.

Note that while this method currently uses a CNN as the machine learner, this

could also be done with Graph Neural networks [69] (GNNs), a subset of Geometric deep learning models [70]. GNNS attempt to perform learning on data that can be represented as a graph, while Geometric deep learning attempts to perform learning on data that is in any non-euclidean space (something that is typically required for ANNs). Further, when using a GNN, p-norms are not required to transform the graphical data into features vectors, as a GNN us able to learn the graph structure via message passing between neighboring vertices. Another avenue of improvement could be to use the PEPs calculated by the software `qvality` (or the q-values computed when creating the HCT set) to weight the training examples when training the machine learner.

A perhaps more robust alternative to using one machine learner is to use an ensemble of machine learners. It is not clear what kind of machine learners would be best, nor it is clear what the best way to reconcile the results from the different learners would be; however, ensemble machine learners have seen good success in the past [71]. Further, model selection could be performed by some sort of "meta-learner" to select beneficial machine learners while ignoring detrimental ones. Such learners could be entirely different classifiers (e.g. there could be an SVM, a neural network, and a CNN). A perhaps more clever alternative to this would be to use several CNNS, each of which utilize different norm spaces (*e.g.* different sets of p-norms).

4.3.1 Lysate Dataset Proteins

In this section, several proteins which were found by the `1-peptide (with shared)` model at a low q-value, but not by the `Semi-supervised` model are examined. One example of such a protein from each of the lysate datasets is presented.

Here, several target proteins identified at a very low q-value by `1-peptide (with shared peptides)`, but not `Semi-supervised` are examined. The blue nodes

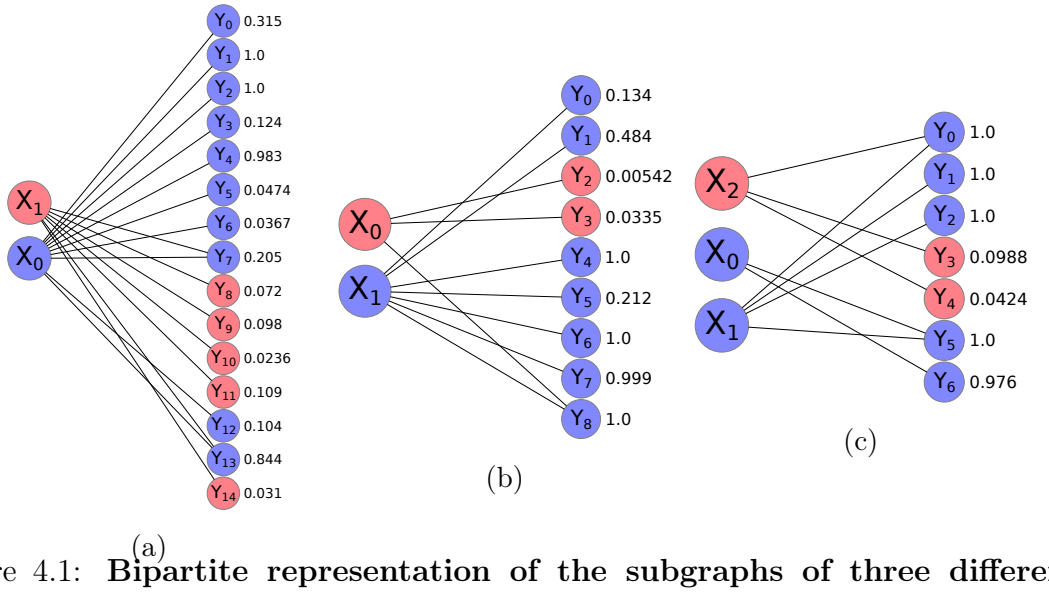
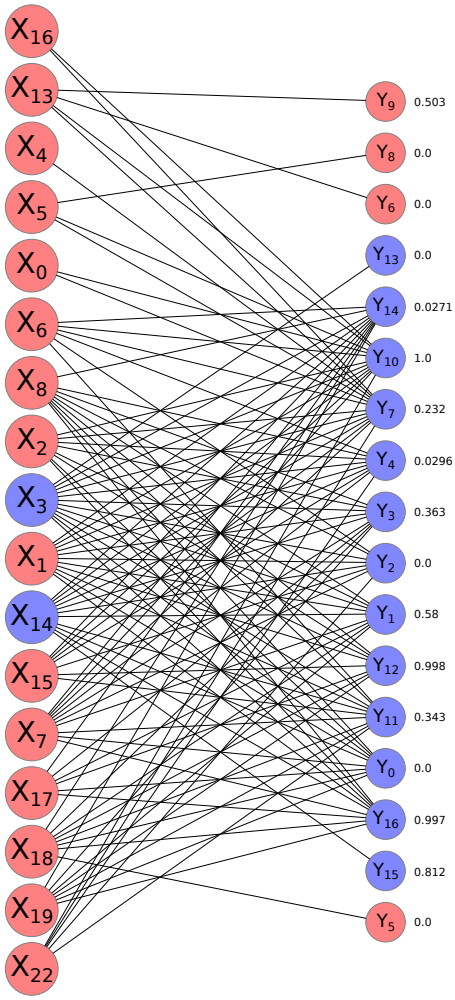
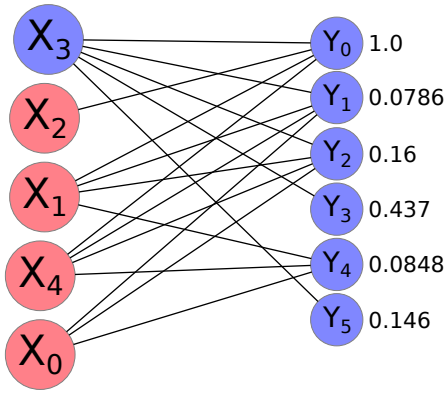


Figure 4.1: **Bipartite representation of the subgraphs of three different target proteins that are likely absent.** The proteins which are likely absent were identified at a q-value of 0.0 (with the exception of protein ZK563.7, which was identified at a q-value of 0.2) by `1-peptide (with shared peptides)` but not by `Semi-supervised`. Proteins are denoted as X_i, while peptides are denoted as Y_j. An edge between a protein X_i and a peptide Y_j indicates that protein X_i could have emitted peptide Y_j. The numbers next to the peptides indicate their peptide-level confidence scores that were produced by using Percolator to post-process a peptide search conducted with Comet. Blue nodes represent present proteins and peptides while red nodes indicate absent proteins and peptides. **(a)** Subgraph representing proteins ZK563.7 (X_1) and F08C6.6 (X_0) from the *C. elegans* lysate dataset. **(b)** Subgraph representing proteins YGR143W (X_0) and YGR159W (X_1) from the *S. cerevisiae* lysate dataset. **(c)** Subgraph representing proteins ENSP00000346209 (X_2), ENSP00000346037 (X_1), and one other protein which is also in the same subgraph from the HumanMD dataset.

represent proteins and peptides believed to be present, while the red nodes represent proteins and peptides which are believed to be absent. In Figure 4.1a, proteins X_1 and X_0 share a few pieces of peptide evidence, Y_7 and Y_{13}. Notice that Y_{13} is the highest scoring peptide adjacent to X_1. Similarly, in Figure 4.1b, the shared peptide, Y_8, is the highest scoring piece of peptide evidence for X_0. This is also the case for Figures 4.1c, 4.2a, and 4.2b. Moreover, each of the proteins which are likely present have at least one piece of unique high scoring evidence (and most have several). Hence, it



(a)



(b)

Figure 4.2: **Bipartite representation of subgraphs of two different proteins which are likely absent.** Notation, node coloring, and edges in this figure have the same the meaning as in Figure 4.1. **(a)** Subgraph representing proteins tr|A0A7I2YQP1|A0A7I2YQP1_HUMAN (X_{13}), tr|A0A7I2YQV4|A0A7I2YQV4_HUMAN (X_{14}), and several other proteins which are also in the same subgraph from the HumanMD dataset searched against the Trembl protein database. Proteins which had identical observed peptide sets are represented by one protein (in bold): $\{\mathbf{X_1}, X_9\}$, $\{\mathbf{X_6}, X_{11}\}$, $\{\mathbf{X_8}, X_{10}, X_{12}\}$, and $\{\mathbf{X_{16}}, X_{20}, X_{21}, X_{23}\}$. **(b)** Subgraph representing proteins tr|X5DP03|X5DP03_HUMAN (X_4), tr|X5D7P8|X5D7P8_HUMAN (X_3), and several other proteins which are also in the same subgraph from the HumanEKC dataset, searched against the human proteome in the Trembl protein database.

is much more likely the case that the proteins with unique, high scoring evidence are actually present, while the other proteins are absent.

CHAPTER 5 SOURCE CODE AVAILABILITY

Source code for the inference engine presented in this manuscript can be found here.

BIBLIOGRAPHY

[1] Y. F. Li and P. Radivojac. Computational approaches to protein inference in shotgun proteomics. *BMC bioinformatics*, 13(Suppl 16):S4, 2012.

[2] A. Doerr. Dia mass spectrometry. *Nature Methods*, 12(1):35–35, 2015.

[3] L. Käll, J. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss. A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4:923–25, 2007.

[4] A. Keller, A. I. Nesvizhskii, E. Kolker, and R. Aebersold. Empirical statistical model to estimate the accuracy of peptide identification made by MS/MS and database search. *Analytical Chemistry*, 74:5383–5392, 2002.

[5] O. Serang, M. J. MacCoss, and W. S. Noble. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *Journal of Proteome Research*, 9(10):5346–5357, 2010.

[6] H. Tang, R. J. Arnold, P. Alves, Z. Xun, D. E. Clemmer, M. V. Novotny, J. P. Reilly, and P. Radivojac. A computational approach toward label-free protein quantification using predicted peptide detectability. *Bioinformatics*, 22:e481–e488, 2006.

[7] Y. F. Li, R. J. Arnold, H. Tang, and P. Radivojac. The importance of peptide detectability for protein identification, quantification, and experiment design in ms/ms proteomics. *Journal of proteome research*, 9(12):6288–6297, 2010.

[8] A. D. Catherman, O. S. Skinner, and N. L. Kelleher. Top down proteomics: Facts and perspectives. *Biochemical and Biophysical Research Communications*, 445(4):683–693, 2014. Advances in OMICs-based disciplines.

[9] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.

[10] J. D. Storey. The positive false discovery rate: A bayesian interpretation and the q-value. *The Annals of Statistics*, 31(6):2013–2035, 2003.

[11] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[12] S. Houel, R. Abernathy, K. Renganathan, K. Meyer-Arendt, N. G. Ahn, and W. M. Old. Quantifying the impact of chimera MS/MS spectra on peptide identification in large-scale proteomics studies. *Journal of Proteome Research*, 9(8):4152–4160, 2010.

[13] A. I. Nesvizhskii, A. Keller, E. Kolker, and R. Aebersold. A statistical model for identifying proteins by tandem mass spectrometry. *Analytical Chemistry*, 75:4646–4658, 2003.

[14] A. Keller, J. Eng, N. Zhang, X. Li, and R. Aebersold. A uniform proteomics ms/ms analysis platform utilizing open xml file formats. *Molecular systems biology*, 1(1):2005–0017, 2005.

[15] T. L. Thompson and R. M. Peart. Useful search techniques to save research time. *Transactions of the ASAE*, 11(4):461–0467, 1968.

[16] O. Serang. The probabilistic convolution tree: Efficient exact Bayesian inference for faster LC-MS/MS protein inference. *PloS one*, 9(3):e91507, 2014.

[17] J. Pfeuffer, T. Sachsenberg, T.M.H. jeerd T. Dijkstra, O. Serang, K. Reinert, and O. Kohlbacher. Epifany: A method for efficient high-confidence protein inference. *Journal of proteome research*, 19(3):1060–1072, 2020.

[18] H. L. Röst, T. Sachsenberg, S. Aiche, C. Bielow, H. Weisser, F. Aicheler, S. Andreotti, H. Ehrlich, P. Gutenbrunner, E. Kenar, et al. Openms: a flexible open-source software platform for mass spectrometry data analysis. *Nature methods*, 13(9):741–748, 2016.

[19] T. Huang and Zengyou Z. He. A linear programming model for protein inference problem in shotgun proteomics. *Bioinformatics*, 28(22):2956–2962, 09 2012.

[20] V. Dancik, T.A. Addona, K.R. Clauser, J.E. Vath, and P.A. Pevzner. *De novo* peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 6(3-4):327–342, 1999.

[21] S. Kim, N. Gupta, and P. A. Pevzner. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *Journal of Proteome Research*, 7:3354–3363, 2008.

[22] E. R. Schroeter, C. J. DeHart, T.P. Cleland, W. Zheng, P.M. Thomas, Neil L N. L. Kelleher, Marshall Bern, and Mary H Schweitzer. Expansion for the brachylophosaurus canadensis collagen i sequence and additional evidence of the preservation of cretaceous protein. *Journal of proteome research*, 16(2):920–932, 2017.

[23] W. S. Noble. Mass spectrometrists should search only for peptides they care about. *Nature methods*, 12(7):605–608, 2015.

[24] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[25] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[26] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[27] M. Wilhelm, J. Schlegl, H. Hahne, A. M. Gholami, M. Lieberenz, M. M. Savitski, E. Ziegler, L. Butzmann, S. Gessulat, H. Marx, et al. Mass-spectrometry-based draft of the human proteome. *Nature*, 509(7502):582–587, 2014.

[28] J. D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society*, 64:479–498, 2002.

[29] L. Käll, J. D. Storey, and W. S. Noble. QVALITY: Nonparametric estimation of q values and posterior error probabilities. *Bioinformatics*, 25(7):964–966, 2009.

[30] J. E. Elias and S. P. Gygi. Target-decoy search strategy for mass spectrometry-based proteomics. *Proteome bioinformatics*, pages 55–71, 2010.

[31] J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*, 4(3):207–214, 2007.

[32] J. Klimek, J. S. Eddes, L. Hohmann, J. Jackson, A. Peterson, S. Letarte, P. R. Gafken, J. E. Katz, P. Mallick, H. Lee, A. Schmidt, R. Ossola, J. K. Eng, R. Aebersold, and D. B. Martin. The standard protein mix database: a diverse data set to assist in the production of improved peptide and protein identification software tools. *Journal of Proteome Research*, 7(1):96–103, 2008.

[33] B. Zhang, M. C. Chambers, and D. L. Tabb. Proteomic parsimony through bipartite graph analysis improves accuracy and transparency. *Journal of Proteome Research*, 6(9):3549–3557, 2007.

[34] J. Lee, H. Choi, C. M. Colangelo, D. Davis, M.R. Hoopmann, L. Käll, H. Lam, S.H. Payne, Y. Perez-Riverol, M. The, et al. Abrf proteome informatics research group (iprg) 2016 study: Inferring proteoforms from bottom-up proteomics data. *Journal of biomolecular techniques: JBT*, 29(2):39, 2018.

[35] D. P. Zolg, M. Wilhelm, K. Schnatbaum, J. Zerweck, T. Knaute, B. Delanghe, D. J. Bailey, S. Gessulat, H. Ehrlich, M. Weininger Maximilian, et al. Building proteometools based on a complete synthetic human proteome. *Nature methods*, 14(3):259–262, 2017.

[36] H. Wenschuh, R. Volkmer-Engert, M. Schmidt, M. Schulz, J. Schneider-Mergener, and U. Reineke. Coherent membrane supports for parallel microsynthesis and screening of bioactive peptides. *Peptide Science*, 55(3):188–206, 2000.

[37] H. Hahne, F. Pachl, B. Ruprecht, S. K. Maier, S. Klaeger, D. Helm, G. Médard, M. Wilm, S. Lemeer, and B. Kuster. Dmso enhances electrospray response, boosting sensitivity of proteomic experiments. *Nature methods*, 10(10):989–991, 2013.

[38] Eric W Deutsch. Mass spectrometer output file format mzml. In *Proteome bioinformatics*, pages 319–331. Springer, 2010.

[39] N. Hulstaert, J. Shofstahl, T. Sachsenberg, M. Walzer, H. Barsnes, L. Martens, and Y. Perez-Riverol. Thermorawfileparser: modular, scalable, and cross-platform raw file conversion. *Journal of proteome research*, 19(1):537–542, 2019.

[40] S. A. Leonard, T. G. Littlejohn, and A. D. Baxevanis. Common file formats. *Current protocols in bioinformatics*, 16(1):A–1B, 2006.

[41] J. K. Eng, T. A. Jahan, and M. R. Hoopman. Comet: An open-source ms/ms sequence database search tool. *PROTEOMICS*, 13:22–24, 2013.

[42] C. Y. Park, A. A. Klammer, L. Käll, M. P. MacCoss, and W. S. Noble. Rapid and accurate peptide identification from tandem mass spectra. *Journal of Proteome Research*, 7(7):3022–3027, 2008.

[43] Indra M Chakravarty, JD Roy, and Radha Govind Laha. Handbook of methods of applied statistics. pages 392–394, 1967.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C.R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[45] C.R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane Allan, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[46] S. Behnel, M. Faassen, and I. Bicking. lxml: Xml and html with python, 2005.

[47] U. Kaempf. The binomial test: a simple tool to identify process problems. *IEEE Transactions on Semiconductor Manufacturing*, 8(2):160–166, 1995.

[48] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[49] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[50] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[51] B. Efron, R. Tibshirani, J.D. Storey, and V. Tusher. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*, 96(456):1151–1161, 2001.

[52] M. M. Savitski, M. Wilhelm, H. Hahne, B. Kuster, and M. Bantscheff. A scalable approach for protein false discovery rate estimation in large proteomic data sets. *Molecular & Cellular Proteomics*, 14(9):2394–2404, 2015.

[53] Dattatreya Mellacheruvu, Zachary Wright, Amber L Couzens, Jean-Philippe Lambert, Nicole A St-Denis, Tuo Li, Yana V Miteva, Simon Hauri, Mihaela E Sardiu, Teck Yew Low, et al. The crapome: a contaminant repository for affinity purification–mass spectrometry data. *Nature methods*, 10(8):730–736, 2013.

[54] P. Jones, R. G. Côté, L. Martens, A. F. Quinn, C. F. Taylor, W. Derache, H. Hermjakob, and R. Apweiler. Pride: a public repository of protein and peptide identifications for the proteomics community. *Nucleic acids research*, 34(suppl_1):D659–D663, 2006.

[55] S.R. Ramakrishnan, C. Vogel, J.T. Prince, R. Wang, Z. Li, L.O. Penalva, M. Myers, E.M. Marcotte, and D.P. Miranker. Integrating shotgun proteomics and mrna expression data to improve protein identification. *Bioinformatics*, 25(11):1397–1403, 2009.

[56] M .Vaudel, J. M. Burkhart, D. Breiter, R. P. Zahedi, A. Sickmann, and L. Martens. A complex standard for protein identification, designed by evolution. *Journal of proteome research*, 11(10):5065–5071, 2012.

[57] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–22, 1977.

[58] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1998.

[59] IBM ILOG. Ibm ilog cplex.

[60] L.R. Ford Jr. and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[61] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[62] J. John, I. Angelov, A. A. Öncül, and D. Thévenin. Techniques for the reconstruction of a distribution from a finite number of its moments. *Chemical Engineering Science*, 62(11):2890–2904, 2007.

[63] K. Lucke, J. Pennington, P. Kreitzberg, B. Kuster, M. Wilhelm, and O. Serang. Best in show: ensemble evaluation of protein inference engines. *bioarvix*, 2021.

[64] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] K. L. Howe, P. Achuthan, J. Allen, J. Allen, J. Alvarez-Jarreta andM. R. Amode, I. M. Armean, A. G. Azov, R. Bennett, J. Bhai, K. Billis, S. Boddu, M. Charkhchi, C. Cummins, L .Da Rin Fioretto, C. Davidson, K. Dodiya, B. El Houdaigui, R. Fatima, A. Gall, C. Garcia Giron, T. Grego, C. Guijarro-Clarke, L Haggerty, A. Hemrom, T. Hourlier, O. G. Izuogu, T. Juettemann, V. Kaikala, M. Kay, I. Lavidas, T. Le, D. Lemos, J. Gonzalez Martinez Jose, J. C. Marugán, T. Maurel, A. C. McMahon, S. Mohanan, B. Moore, M. Muffato, D. N. Oheh, D. Paraschas, A. Parker, A. Parton, I. Prosovetskaia, M. P. Sakthivel, A. I. A. Salam, B. M. Schmitt, H. Schuilenburg, D. Sheppard, E. Steed,

M. Szpak, M. Szuba, K. Taylor, A. Thormann, G. Threadgold, B. Walts, A. Winterbottom, M. Chakiachvili, A. Chaubal, N. De Silva, B. Flint, A. Frankish, S. E. Hunt, G. R. IIsley, N. Langridge, J. E. Loveland, F. J. Martin, J. M. Mudge, J. Morales, E. Perry, M. Ruffier, J. Tate, D. Thybert, S. J. Trevanion, F. Cunningham, A. D. Yates, D. R. Zerbino, and P. Flicek. Ensembl 2021. *Nucleic Acids Research*, 49(D1):D884–D891, 11 2020.

[67] Amos Bairoch and Rolf Apweiler. The swiss-prot protein sequence data bank and its supplement trembl. *Nucleic acids research*, 25(1):31–36, 1997.

[68] Smriti R Ramakrishnan, Christine Vogel, Taejoon Kwon, Luiz O Penalva, Edward M Marcotte, and Daniel P Miranker. Mining gene functional networks to improve mass-spectrometry-based protein identification. *Bioinformatics*, 25(22):2955–2961, 2009.

[69] F. Scarselli, M. Gori, A. H. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[70] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[71] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.