CONTEXT-SENSITIVE MARKOV MODELS FOR PEPTIDE SCORING AND IDENTIFICATION FROM TANDEM MASS SPECTROMETRY

by

Himanshu Grover

B.Tech. Information Technology, IIIT - Allahabad (India), 2004

M.S. Biomedical Informatics, University of Pittsburgh, 2008

Submitted to the Graduate Faculty of

School of Medicine in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2012

UNIVERSITY OF PITTSBURGH

SCHOOL OF MEDICINE

This dissertation was presented

by

Himanshu Grover

It was defended on

September 28, 2012

and approved by

Shyam Visweswaran, M.D., Ph.D.,

Assistant Professor, Department of Biomedical Informatics, University of Pittsburgh

Garrick Wallstrom, M.S., Ph.D.,

Assistant Professor, Department of Biomedical Informatics, Arizona State University

Christine C. Wu, Ph.D.,

Associate Professor, Department of Cell Biology & Physiology, University of Pittsburgh

Dissertation Advisor: Vanathi Gopalakrishnan, M.S., Ph.D.,

Associate Professor, Department of Biomedical Informatics, University of Pittsburgh

Copyright © by Himanshu Grover

2012

CONTEXT-SENSITIVE MARKOV MODELS FOR PEPTIDE SCORING AND IDENTIFICATION FROM TANDEM MASS SPECTROMETRY

Himanshu Grover, M.S., PhD

University of Pittsburgh, 2012

Computational methods for peptide identification via tandem mass spectrometry (MS/MS) lie at the heart of proteomic characterization of biological samples. Due to the complex nature of peptide fragmentation process inside mass spectrometers, most extant methods underutilize the intensity information available in the tandem mass spectrum. Further, high noise content and variability in MS/MS datasets present significant data analysis challenges. These factors contribute to loss of identifications, necessitating development of more complex approaches.

This dissertation develops and evaluates a novel probabilistic framework called Context-Sensitive Peptide Identification (**CSPI**) for improving peptide scoring and identification from MS/MS data. Employing Input-Output Hidden Markov Models (IO-HMM), CSPI addresses the above computational challenges by modeling the effect of peptide physicochemical features (‘**context**’) on their observed (normalized) MS/MS spectrum intensities. Flexibility and scalability of the CSPI framework enables incorporation of many different kinds of features from the domain into the modeling task. Design choices also include the underlying parameter representation and allow learning complex probability distributions and dependencies embedded in the data.

Empirical evaluation on multiple datasets of varying sizes and complexity demonstrates that CSPI's intensity-based scores significantly improve peptide identification performance, identifying up to ~25% more peptides at 1% False Discovery Rate (FDR) as compared with popular state-of-the-art approaches. It is further shown that a weighted score combination procedure that includes CSPI scores along with other commonly used scores leads to greater discrimination between true and false identifications, achieving ~4-8% more correct identifications at 1% FDR compared with the case without CSPI features.

Superior performance of the CSPI framework has the potential to impact downstream proteomic investigations (like protein identification, quantification and differential expression) that utilize results from peptide-level analyses. Being computationally intensive, the design and implementation of CSPI supports efficient handling of large MS/MS datasets, achieved through database indexing and parallelization of the computational workflow using multiprocessing architecture.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep gratitude to my advisor, Dr. Vanathi Gopalakrishnan, whose support, guidance and direction taught me a lot and tremendously helped me in developing my thesis. It wouldn't be possible without her utmost patience and constant encouragement over the long process of conceiving, designing and developing the research. I would like to express my heartfelt thanks to Dr. Garrick Wallstrom with whom I have had the good fortune of many long discussions that provided crucial insights into my work. I would like to thank my thesis committee members, Drs. Shyam Visweswaran and Christine Wu for all their guidance, time and effort. Their thoughtful feedback and unique perspectives helped me refine the ideas in this thesis. I am also thankful to Dr. Michael MacCoss from the University of Washington, with whom I had the opportunity to meet only a couple of times but obtained valuable inputs during each meeting.

I am grateful to Drs. Mike Becich, Rebecca Crowley and Roger Day for their support and encouragement on various occasions. I thank Toni Porterfield, Yolanda Dibucci and Charles Dizard for all the valuable help they have provided me over the years.

I acknowledge the financial support from the NIH and NIGMS, without which I wouldn't have been able to pursue my doctoral studies.

I thank all my friends in Pittsburgh and elsewhere for enriching my life with wonderful and memorable moments.

Finally and most importantly, I would like to express my deepest thanks to all my family members, who are the pillars of my being. I am indebted forever: to my dear parents who have always believed in me and encouraged me to pursue my dreams. Through their example they have instilled in me the highest ideals of diligence, integrity and humility, the rewards of which I reap every day; to my brother Harsh whose passion, indomitable spirit and optimism against all odds is a constant source of inspiration; to my wife Radhika who has made plenty of sacrifices

without ever complaining, and whose loving care and moral support at every step of the way provided me with all the strength to go on.

GLOSSARY

b/y-ions: Predominant fragment ion-types produced when peptides are fragmented using collision-induced dissociation (CID).

Collision-induced dissociation (CID): A commonly used method for inducing fragmentation of peptides inside mass-spectrometers by bombarding the gas-phase charged peptides with an inert-gas, like argon.

CSPI: Context-Sensitive Peptide Identification, a framework based on Input-Output Hidden Markov Models, for learning peptide fragmentation intensity patterns from tandem mass-spectrometry data. Context refers to the peptide physicochemical properties, that determine its fragmentation behavior.

Data Preprocessing: Steps performed to "clean" the data before training or applying the models to new samples, with the goal to reduce the effects of unknown sources of variation (like sample handling or during data acquisition) and to aid in learning better models. Some typical examples include data transformation and normalizations, like square-root or log transform, noise-filtering etc.

Database Indexing: A technique to store a database in a structured format amenable to fast search and retrieval. For example, a key-value representation, as in a dictionary, allows storing a "value" with an associated "key" that can later be used to query the data-structure for instant retrieval of the stored value.

Database Search Parameters: Parameters used to constrain the search for candidate peptides from protein databases when evaluating tandem mass spectra. Examples include mass-error tolerance for peptides and their fragments, allowable post-translational modifications and protein digestion enzyme specificity of the theoretical peptides searched.

Database Searching: A paradigm for identification of peptides, and hence proteins contained in samples, using mass-spectrometry technology. Each mass-spectrum is compared against a set of potential candidates derived from an appropriate protein database based on the masses of peptides present in the database, which are then evaluated with a scoring algorithm.

Electrospray Ionization (ESI): A commonly used technique for depositing charge on macromolecules like proteins and peptides by employing electricity to disperse the liquid containing analytes into a fine aerosol. The solvent droplets evaporate leaving behind the charge on the analytes, without fragmenting them. This technique is typically coupled with Liquid Chromatography, a technique for separating peptides which are then charged and introduced into the mass spectrometer directly for analysis.

Expectation Maximization: An Iterative algorithm for obtaining the (locally) maximum likelihood parameter estimates of a statistical model that contains hidden or unobserved variables.

False Discovery Rate (FDR): A global error control measure that corrects for multiple comparisons by directly controlling the type-I errors (incorrectly rejected null hypotheses).

FASTA Database: An ASCII text file containing a list of Protein sequences where each sequence is preceded with a single-line header (identified with the “>” symbol in the beginning) uniquely identifying and describing the sequence, followed by the lines containing the actual amino-acid sequence of the protein.

Inference: Application of a trained model to a new sample for the intended purpose, for example, classification of the sample or enumerating the probability that the new sample contains similar patterns as the data that were used to train the model.

Input-output Hidden Markov Model (IO-HMM): A sequential machine learning algorithm (extending classical Hidden Markov Model) that is used to learn patterns in pairs of sequences, called input and output. The goal is to dynamically map the influence of the input sequence on to the output sequence.

Logistic Regression: A classical machine learning technique used to predict the probability distribution of a categorical outcome variable that can take on a finite set of values, for a sample that is described using a set of observed features (a.k.a. predictor variables).

Machine Learning: A class of techniques, dealing with the design and development algorithms that learn patterns from empirical data representing a complex phenomenon. The goal is typically to describe/summarize the data or provide a means for future predictions, forecasting and decision making in the presence of uncertainty and limited theory.

Markov Assumption: This refers to the property of a stochastic process that limits the dependence of the probability distribution of the current state to the past observations/states. The 'order' of the assumption determines how far back in time or space to look. For example, first-order markov assumption means that the current state of the process depends only on the immediately preceding state and not the ones before that.

Mass Spectrometry: Analytical tool to determine the chemical compounds present in a sample by measuring their masses or m/z-ratio based on their flight or motion inside electromagnetic fields.

Mass-to-charge ratio (m/z): Ratio of mass of an entity to its charge state.

Maximum Likelihood: A method of parameter-estimation of a statistical model, that maximizes the likelihood, in probabilistic sense, of the training data from the model.

Multiple Comparisons or Testing Problem: Occurs when several statistical hypothesis tests are considered simultaneously, as a whole. Under such situations, any testing procedure is more likely to incorrectly reject the null hypothesis by chance and must be accounted for.

Multiprocessing: Use of multiple CPUs or processor cores available on a single machine to speed up processing in computationally intensive applications, by appropriately splitting the task and allocating the sub-tasks to individual units.

Peak: A signal detected by the detector unit of the mass-spectrometer, characterized by the m/z of the entity observed and its abundance.

Peptide: A short sequence of amino-acids, typically ranging in length from 5 to 50 amino acids.

Peptide-Spectrum Match (PSM): A pair of a peptide and an experimental mass-spectrum, either representing a true peptide or a candidate peptide being evaluated against the spectrum.

PSM Score/feature: A score or a feature that evaluates the quality of the match between a peptide and a spectrum.

Q-values: FDR equivalent of the standard p-value for an individual hypothesis test, referring to the minimum FDR at which the given hypothesis test can be called significant.

Tandem Mass Spectrometry: A technique involving two stages of mass-spectrometric analysis, where the first stage generates a precursor mass-spectrum while the second stage selects a small number of abundant precursor signals for further fragmentation and analysis.

Tandem Mass-spectrum: A plot showing the masses (or mass-to-charge ratio) of entities (peptide fragments) on the x-axis vs. their abundance on the y-axis.

Target-decoy strategy: An approach to control the false-discovery rate of peptide identifications in a large-scale experiment. Target database is the actual protein database of interest, while the decoy is a shuffled or reversed protein sequence database which contains false/random sequences. Database search against a decoy yields false peptide identifications, by design, and can be used to determine the null distribution of PSM scores required for controlling FDR.

Training: The process of learning the optimal parameters of a machine learning model, by optimizing a certain criterion or loss function.

Training Dataset: A dataset containing empirical examples to train a machine learning model.

1.0 INTRODUCTION

Proteins are among the most important bio-molecules in all living beings, with numerous physiological and executive roles. They are involved in catalyzing and regulating biochemical processes that maintain life, in transport of molecules within and across cells, or as structural building blocks of many cellular components (Eidhammer et al. 2007). The set of expressed proteins varies extensively with time, the type of tissue or fluid or sub-cellular location, as well as according to the specific environment a cell finds itself in; such study and characterization comprises the field of Proteomics (Eidhammer et al. 2007). Rapid advances in this young and burgeoning field over the last decade are facilitating our understanding of cellular processes at molecular level as enabled through protein expression and interactions, post-translational modifications, and particularly their role as biomarkers of clinical conditions like disease (Vitek 2009).

Towards this end, mass-spectrometry technology has played a key role and continues to provide wealth of information in conjunction with clever experimental designs (Aebersold et al. 2003). Of particular importance is the paradigm of bottom-up or shotgun proteomics, in which proteins in complex mixtures are first cleaved into smaller peptides which are then analyzed using mass-spectrometry (Wysocki et al. 2005). One of the cornerstones of this peptide-centric approach, on which much downstream analysis and interpretation rests, is the process of peptide identification using the information contained in the mass-spectra generated by these peptides

(Nesvizhskii 2007). This approach offers several advantages like high-throughput nature, greater sensitivity and specificity of signal detection, as well as greater dynamic range of detection (de Godoy et al. 2006).

Developments in technology, together with sophisticated computational algorithms to process and analyze the acquired data, have transformed the shotgun method into the routine methodology of choice in proteomic investigations. Several large-scale collaborative projects are currently underway utilizing this approach to characterize the entire proteomic complement of key organs like liver and brain (HLP www.hlpp.org, HBP http://www.hbpp.org/), or in fluids like plasma (Anderson et al. 2004). Beyond identification of sample components, this paradigm is also critical in their characterization, such as protein quantification and identifying post-translational modifications, as well as sample comparison to characterize relative occurrence, abundance and/or differential modification across different populations of cells. The eventual goal of proteomics is to develop such information-rich maps and holds tremendous promise for clinical applications like early diagnostic tests or discovery of new drug targets for diseases (Eidhammer et al. 2007).

1.1 THE PROBLEM

Routine shotgun proteomic experiments yield large datasets of peptide tandem mass-spectra (MS/MS). One primary data-analysis task then is to ascertain the peptide sequence(s) that generated these spectra, and subsequently infer the parent proteins from the resulting peptides (Steen et al. 2004). A typical approach to achieve this goal is called Database Searching, and proceeds via scoring candidate peptides (obtained from a protein sequence database) against experimental spectra for possibility of a match (Nesvizhskii 2007).

A critical step in scoring involves theoretical modeling of peptide fragmentation behavior inside mass-spectrometers. The scorer then compares the candidate theoretical spectra with experimental spectrum to compute agreement. Several scoring algorithms, some heuristic while others probabilistic, have been developed to achieve this task. These algorithms are routinely applied to complex proteomic investigations. However, they rely on over-simplified theoretical fragmentation models and scores that either completely ignore or underutilize the intensity dimension of MS/MS spectra. In large-scale experiments less than 30 % spectra are confidently assigned with peptides, and inadequacies of scoring algorithms is a key contributing factor, among others (Marcotte 2007).

Peptide fragmentation inside mass-spectrometers is a complex process. Although much effort has been invested into deciphering the rules (Paizs et al. 2005; Hubbard et al. 2010), only limited qualitative understanding has been achieved which is hard to encode in deterministic algorithms. Several characteristics of peptide MS/MS spectra complicate their interpretation:

1. High noise content (Ning et al. 2007)**:**

An average peptide has a theoretical spectrum of few tens of most important peaks while typical real peptide MS/MS spectrum can contain several hundreds of peaks. A large fraction of these peaks emerge from uncontrollable electrical and/or chemical noise as well as from unanticipated fragmentation events, all of which can vary from one experiment/laboratory to another, and also across the mass-range of individual spectra. Sparse signal events are then interspersed and sometimes embedded in large stretches of unexplainable noise. Complicating the matters are several spectra in each experiment that emerge from non-peptide species and must be correctly distinguished.

2. Variability:

Another significant challenge is the widely varying intensity profiles of peaks produced by peptide fragmentation, which depends partly on the experiment, but also significantly on the physicochemical properties of the peptide itself (Huang et al. 2008). These profiles vary not only across different peptides, but also from the same peptide generated in different labs or experiments, making accurate and formulaic prediction of peak intensities a challenging task.

3. Low mass accuracy and resolution:

All mass spectrometers have an expected measurement error and an associated limit on the signal resolution, which must be taken into account when selecting candidates to match against a spectrum as well as in their scoring. Database searches typically result in large number of candidate hits per each spectrum, and for many spectra more than a few candidates can randomly achieve comparably high scores. This leads to many spurious or chance matches. While the more recent high-resolution and high mass-accuracy instruments, like the Fourier Transform and Orbitrap spectrometers, reduce the burden of evaluating too many candidates, these are not much widely available due to huge costs involved. The benefit may also be offset if database search constraints, like allowable post-translational modifications, are relaxed to search a larger space of peptides**.** Furthermore, in all cases fragmentation spectra are still acquired at lower resolution posing challenges for accurate identification.

As a result of these factors, the score distributions of true and false peptide identifications from simple scoring systems overlap significantly, making them hard to differentiate (Keller et al. 2002); this is particularly true for peptides that don't fragment extensively or have specific sequence-dependent effects (Hubbard et al. 2010). Therefore, a good scoring function is of primary importance for the accurate evaluation of the quality of matches between spectra and peptides. Along with proper normalization and transformation of spectra, this requires application of statistical and machine learning methods that can automatically account for presence of noise as well as learn complex intensity patterns from data. This will lead to improved peptide identification accuracy as well as downstream interpretation from large-scale experiments, which strongly depends on the confidently identified peptides.

1.2 THE APPROACH

This thesis explores novel computational approaches to address some of the above challenges and the unique aspects of MS/MS data. Specifically, probabilistic models of fragmentation behavior of peptides are developed, taking into account appropriate contextual information from peptide amino-acid sequence, as well as spectrum information. The overall goal is to develop algorithms that perform well across a wide variety of datasets and that are easily extensible to rapidly developing technology and new experimental protocols.

With accumulation and ready availability of large amount of data from both controlled and real-world experiments in proteomic repositories, automated machine learning algorithms are well suited for this problem. This offers a unique opportunity to understand the data produced from these experiments and build models that automatically capture the underlying variability, which is a major issue in mass-spectrometry based proteomics. Several recent studies have tried to learn peptide fragmentation behavior using automated methods and have reported improved identification performance as well as discovery of previously unknown fragmentation rules (Elias et al. 2004; Klammer et al. 2008). Algorithms in this thesis build upon these studies and attempt to effectively learn peptide fragmentation patterns, both the presence/absence of specific peaks as well as their intensity distribution, using Markovian models, thus yielding a robust scoring system.

Particularly, this work develops a probabilistic framework called "**C**ontext-**s**ensitive **P**eptide **I**dentification" (**CSPI**) that uses Input-output Hidden Markov Models (IO-HMM) to capture the influence of peptide physicochemical properties on their observed MS/MS spectra (Bengio et al. 1995). These models have been previously successfully applied to several sequential data-mining tasks, including financial data analysis (Bengio et al. 2001), music processing (Jean-Fran\ et al. 2009), and gene regulation (Ernst et al. 2007). CSPI is a scalable and flexible framework with several modeling choices to learn complex patterns embedded in MS/MS data. This offers advantages as compared to previous attempts on modeling fragmentation spectra, which had limited flexibility. Several local and global properties of peptides and their fragment ions, referred to as '**context'** in this thesis, are used to model their effect on fragmentation behavior. In order to reduce noise and make spectra comparable across experiments, several preprocessing steps are performed. Finally, a state-of-the-art post-processor is implemented that combines several scores and features of peptide-spectrum match (PSM) quality to distinguish true from false identifications, while controlling for false discovery rate (FDR) at a user-defined level.

1.2.1 Thesis

The central thesis of this dissertation is that the CSPI framework is effective for peptide scoring and identification from tandem mass spectrometry.

Based on the experiments performed on several datasets of varying complexity and sizes, from controlled as well as real-world experiments, the following specific claims are made:

Claim 1: CSPI statistically significantly improves peptide identification performance at a user-defined FDR compared with the popular state-of-the-art approaches.

Claim 2: Gains in CSPI performance depend strongly upon the fragment ion-types being modeled as well as data normalization protocol used. For the ion-trap data used in this research, y-ions and local normalization scheme show good performance characteristics

Claim 3: CSPI's intensity-based scores combined with other features commonly used for quantifying peptide-spectrum match quality leads to greater discrimination between true and false peptide identifications.

1.3 SIGNIFICANCE

To the best of my knowledge, this is the first attempt to apply IO-HMMs to score peptide-spectrum matches (PSMs). Since peptide scoring lies at the heart of shotgun proteomics approach, a good scoring system with even slightly better performance can make a significant difference in the downstream interpretation of results in large-scale studies. This necessarily involves effective utilization of the information contained in the spectrum and being able to differentiate true from false identifications in the presence of noise and variation. A key deliverable of this work is the new scoring framework that models the fragmentation behavior taking into account the context of the peptide sequence (both global and local) as well as observed spectral features, thus providing a robust scoring system. Being highly flexible and scalable, it is easy to extend these models with additional features/context (as was demonstrated by exploiting several design choices), thus making them attractive to explore. The immediate impact is seen on the peptide identification accuracies and improved coverage from large-scale MS/MS experiments.

The CSPI framework can be used with different approaches to peptide identification other than Database searching. The methods are very general and can be used for learning fragmentation patterns under different experimental conditions, such as for example from a different spectrometer or a different technique of fragmenting peptides, like Electron Transfer Dissociation (ETD) (Syka et al. 2004).

The implementation of the framework and all evaluation experiments in this dissertation were conducted in the Python programming language (www.python.org), and are made available. For efficient handling of large MS/MS datasets, a multiprocessing version of database search was also developed and is made available. In addition, a simple (and basic) spectrum viewer is provided in order to visualize the effects of different preprocessing steps.

1.4 DISSERTATION OVERVIEW

The rest of the document is organized as follows. Chapter 2 provides background information on the peptide-centric mass-spectrometry analysis pipeline including a review on technology, experimental protocol, and current state-of-the-art algorithms. Also discussed are the analytical and machine-learning methods utilized in this dissertation to address different problems. Chapter 3 describes the CSPI framework in details along with the evaluation protocol used to compare with the state-of-the-art approaches to peptide identification. Chapter 4 discusses the experiments and evaluation methods including description of datasets used in the thesis. Results from evaluation of the CSPI framework are presented in Chapter 5. Chapter 6 concludes with a discussion on open research questions, limitations and potential future developments of the methods presented here.

2.0 BACKGROUND

In this chapter I provide the background material on the techniques relevant to the thesis, beginning with a section on mass-spectrometry (MS) technology and its application to peptide identification via tandem mass spectrometry (MS/MS). Next, I discuss the most popular and successful approach to peptide identification called Database Searching, a sampling of the current state-of-the-art algorithms along with challenges and motivation for my own work. Final sections describe the statistical and machine learning concepts used and developed herein.

2.1 PROTEOMIC MASS SPECTROMETRY: FUNDAMENTALS

Mass-spectrometry (MS) is an analytical technology that has been around for several decades to identify unknown compounds in a sample by measuring their mass or more precisely mass-to-charge ratios (m/z). Figure 1 shows a schematic of the key components involved and the fundamental principle at work. The ionization source deposits charge (protons or electrons) on the sample constituents, which are then transformed into gas phase and introduced into the mass-analyzer unit. Mass-analyzers are of several kinds, but essentially they are all fitted with static or dynamic electromagnetic fields that spatially segregate the ionized components based on their mass and charge status. Finally, as the ions hit the detector unit, they are registered as peaks in a mass-spectrum that has on the horizontal axis the m/z ratio of the ion, and on the vertical axis the number of times the ion was detected, indicative of its relative abundance. Mass information for each component can be derived only if its charge-status is known.

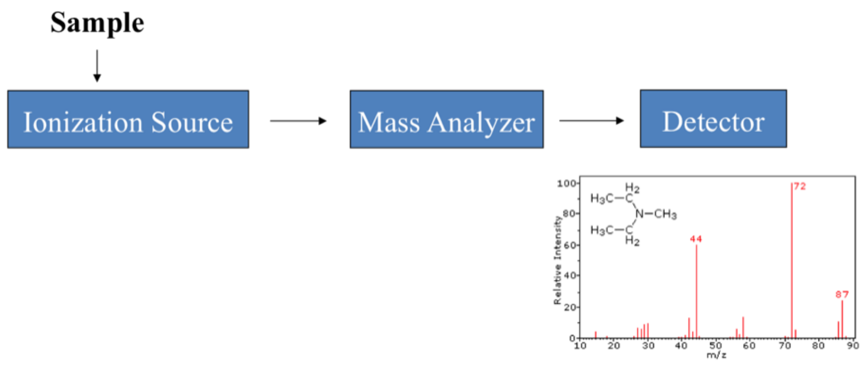


Figure 1. Schematic of components of an MS

Application of MS to large and thermally unstable biomolecules, particularly peptides and proteins, is a relatively recent development. This was made possible largely by development of "soft" ionization techniques like Matrix-assisted Laser Desorption Ionization (MALDI) and Electrospray Ionization (ESI) that can generate stable gas-phase ions from these large and polar molecules, using only minute sample quantity (Aebersold et al. 2001; Aebersold et al. 2003). In conjunction with innovative experimental protocols and instrumentation design, robust data analytics enable comprehensive analysis of simple as well as complex protein mixtures at a global level, including their expression, interactions and post-translational modifications in a high-throughput fashion (Mann et al. 2001; Vitek 2009). As a result, among several technologies available for proteomic investigations, MS-based tools currently play a central role to address a diverse range of research questions.

For simple mixtures of small analytes, information obtained from MS can be sufficient to determine the constituents along with their molecular formulae, with high sensitivity, selectivity and little time. For large intact-proteins however, which can be several kilo-daltons in size, this strategy of structure determination can be rather challenging due to several factors like large number of possible charge states and hence multiplicity of spectrum peaks, complex isotope distributions and several possible locations for unknown post-translational modifications which can shift the m/z of proteins by an unknown amount. This renders exact identification and sequencing (including localization of PTMs) of proteins difficult (Steen et al. 2004). As a result of the aforementioned challenges and for direct sequence determination and related applications, shotgun proteomic methodology is more popular and versatile, involving two stages of mass-spectrometric analysis of constituents, accordingly known as tandem mass-spectrometry.

2.2 TANDEM MASS SPECTROMETRY (MS/MS), A.K.A. BOTTOM-UP OR SHOTGUN PROTEOMICS

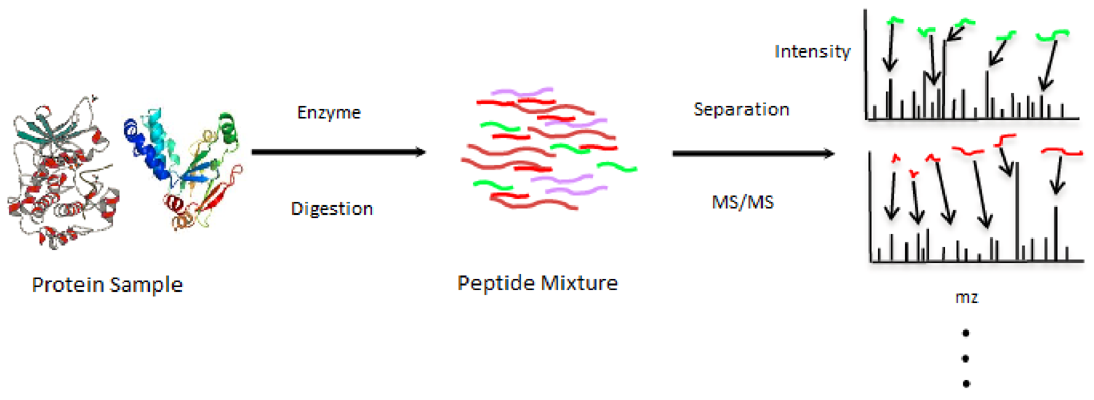


Figure 2. Schematic of Shotgun Proteomics Approach

Typically, shotgun or MS/MS experiments begin with enzymatic digestion of a protein mixture into a mixture of peptides using an enzyme of known specificity, like trypsin (which cleaves each protein at the C-terminus of Lys and Arg residues), chymotrypsin or elastase. This is followed by one, or two stages of separation of peptide mixtures by Liquid Chromatography (LC), and ionization, after which the eluting charged peptides are analyzed via MS/MS (Figure 2). State-of-the-art hybrid systems are fully automated and once the sample has been loaded, they can perform LC followed by MS/MS with minimal human intervention.

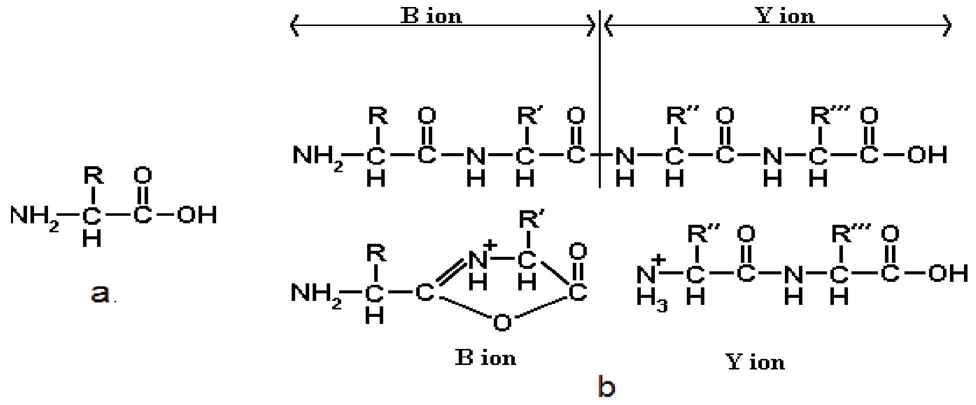


Figure 3. (a) Amino acid structure (R: side-chain identifying the amino acid); (b) b- and y-ion structures (adapted from http://www.weddslist.com/ms/tandem.html)

Mass-spectrometers continuously switch (alternate) between two different scanning modes: in the MS mode the masses (or m/z) of the intact peptides eluting out of the LC column at that instant are measured, while in the subsequent MS/MS scans, few of most abundant peptides (usually three to five) are selected and isolated for fragmentation, generating a MS/MS spectrum for each (Mann et al. 2001). One popular fragmentation protocol is called low-energy Collision-induced dissociation (CID) whereby charged peptides are bombarded with inert gas molecules (like argon) during their flight. This breaks the charged peptide molecule, predominantly at an amide bond, yielding charged fragments (predominantly b- and y-ions (Roepstorff et al. 1984); see Figure 3). Other fragmentation methods include Electron-capture

Dissociation (ECD), Electron-transfer Dissociation (ETD) and High-energy Collisional Dissociation (HCD).

Peptides are most commonly electrospray-ionized which deposits a charge of +1 to +3 (most commonly). For doubly-charged peptides, the resulting fragments usually carry a single charge, while longer peptides which tend to carry higher charges yield higher-charged fragments too. In addition, these fragments may undergo secondary fragmentations and/or loose further neutral molecules like H_2O or NH_3.

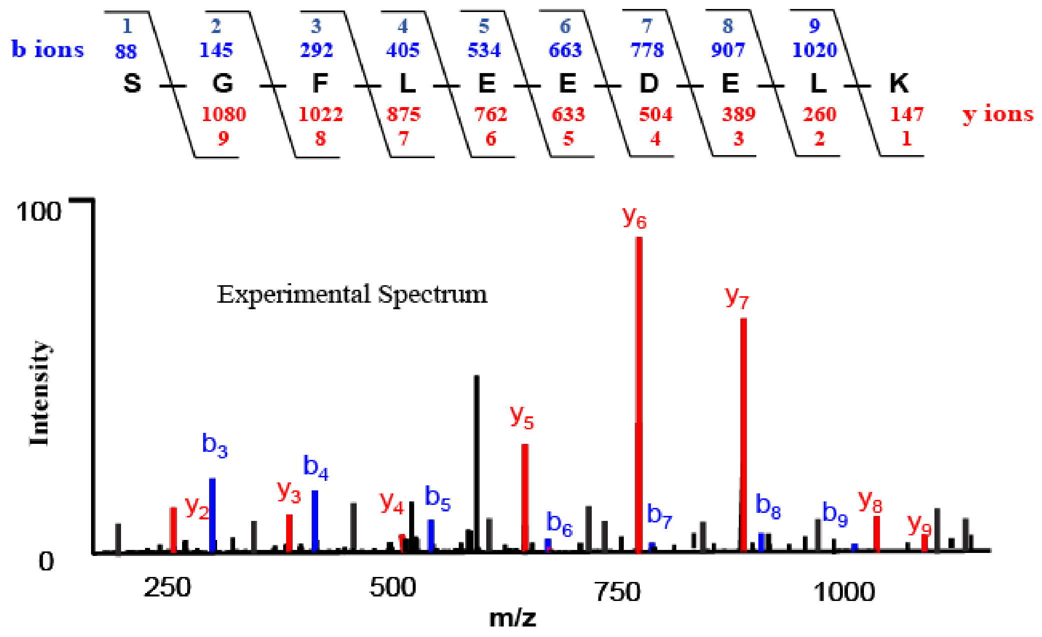


Figure 4. Peptide evaluation against MS/MS spectrum

The resulting fragments are separated in the mass-analyzer and registered as peaks at the appropriate m/z value on the horizontal axis (as described previously) in a fragment-ion spectrum or MS/MS spectrum. The height of the peak represents the relative intensity of the corresponding fragment, and is indicative of the amount of associated cleavage among numerous molecules of the same peptide in the spectrometer. This provides the fundamental data point (an MS/MS spectrum) to be analyzed. A representative (hypothetical) example is shown in Figure 4, with peaks of an experimental spectrum annotated with respective fragment-ion labels of a peptide that generated the spectrum. A typical lab can generate many thousands of such spectra every day, and the goal is to assign peptides to these spectra, followed by relating the peptides back to the parent proteins. This entire protocol is referred to as 'Shotgun Proteomics', in analogy with shotgun genomics (Marcotte 2007).

As described above, and in an ideal situation, the fragment ions from a peptide form a ladder of peaks, with subsequent m/z values separated by mass of some amino acid and the reconstruction of the peptide sequence is only a matter of identifying this contiguous ladder, a rather trivial computational problem even for large-scale analyses. However, real-world MS/MS spectra come with several complications, most predominant being, "Lots of Noise Peaks" (75% peaks are noise (Ning et al. 2007)), "poorly fragmenting peptides", "incomplete sequence of fragments" and "Unknown fragmentation events/pathways". In addition, a large fraction of spectra are either of poor quality or from non-peptide species. These complications make confident assignment of peptides a challenging task. Several approaches to interpreting MS/MS spectra have been developed, which include: 1. Database searching; 2. DeNovo sequencing; 3. Sequence Tags (Steen et al. 2004). All these approaches differ in the way they search the peptide space for each spectrum. The one thing all have in common is a scoring function to evaluate candidate peptides against a spectrum, and lies at the heart of peptide identification algorithms. Of these, database searching is the most common and successful, and is the focus of this work. However, the scoring systems are in principle generic and applicable to DeNovo and Sequence Tag-based approaches as well.

2.3 DATABASE SEARCH

Given a MS/MS spectrum that is experimentally observed, the basic procedure is to search a database of known proteins for candidate peptides, based on the putative peptide's expected m/z, its allowed PTMs and the cleavage enzyme's specificity. Due to the low resolution of commonly used ion-trap tandem mass spectrometers, the experimental mass of putative peptides cannot be determined accurately and may vary slightly from the theoretical true mass calculated from the amino acid sequence. Thus a user-specified mass-tolerance parameter (usually a +/- 3 Da window across the true mass) is applied during the search for candidate peptides, and typically several candidates are returned within the applied mass-window, especially for large organisms or for unconstrained searches. From the known rules of fragmentation, a theoretical spectrum is generated for each of the candidate peptides, which are then scored and ranked based on some form of "agreement" between the theoretical and the experimental spectra. Finally the top ranked peptide is assigned to the spectrum, along with a measure indicating confidence in the assignment (such as the expectation-value, p-value or fixed false-discovery rate, FDR) (see Figure 5).

Such "shared-peak-count" approach was pioneered by Eng et al. in the Sequest algorithm, where agreement was measured with a "cross-correlation" based score (Eng et al. 1994). Reliability of Sequest scores has been evaluated in many studies and the scoring function has been refined over the years (Klammer AA 2009). Mascot, another peptide identification algorithm, is based on a probabilistic scoring (MOWSE), which uses distributions of size of peptide fragments with respect to the size of peptides in the searched database. Peptide assignments are associated with a p-value to differentiate from random matches (Perkins et al. 1999). X!Tandem is a popular open-source algorithm that uses a preliminary intensity-based score (*hyperscore*) which simply sums the intensities of all observed b and y-ions rather than modeling sequence specific fragmentation effects (Craig et al. 2004). Statistical analysis on *hyperscore* is used to compute an E-value, which summarizes the significance of the match. It also optionally allows a two-phase search where the 1st pass can perform fast, constrained searches (for ex. with no PTMs) while the 2nd pass performs a more elaborate search (for ex. with PTMs and relaxed enzyme specificity) but only on the proteins shortlisted from the 1st pass.

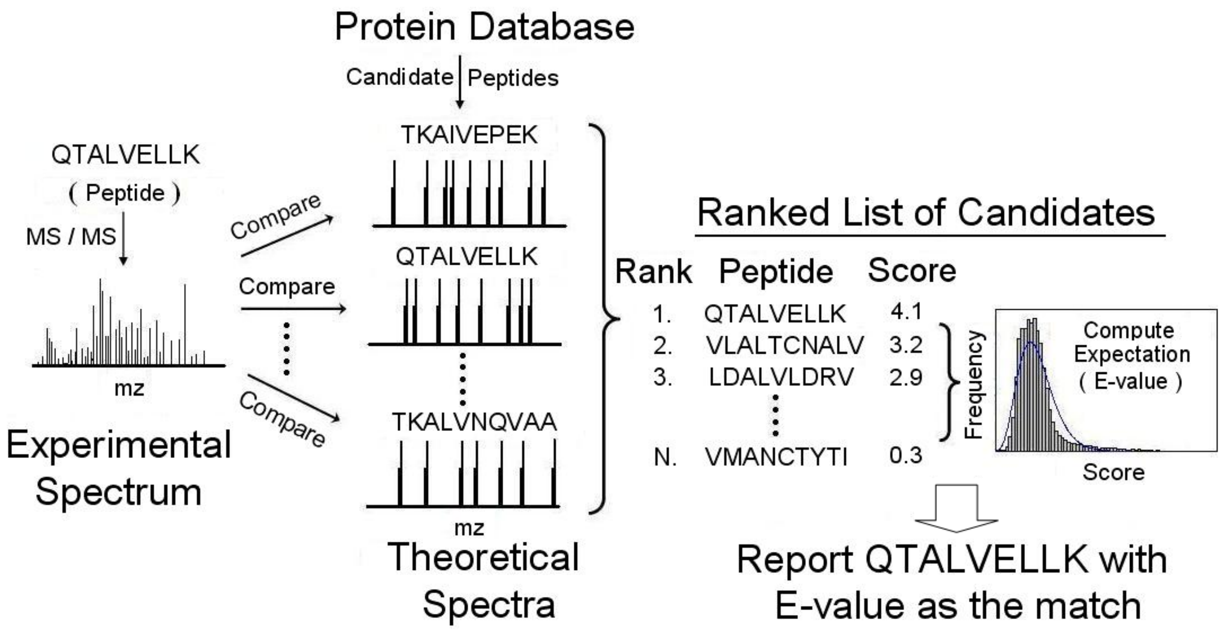


Figure 5. Schematic for Peptide identification by MS/MS via database searching (adapted from (Nesvizhskii et al. 2007))

Thereafter, a host of other algorithms have been developed on similar theme with slightly modified search protocol or features used in the scoring system (Kapp et al. 2005; Kapp et al. 2007). These algorithms are routinely applied to complex proteomic investigations. However, despite their popularity and success in several applications, they all have limitations. In large-scale experiments less than 30 % spectra are confidently assigned with peptides, and inadequacies of scoring algorithms is a key contributing factor, among others (Marcotte 2007).

There is a significant overlap in the score distributions from false and true identifications, which is due to extremely noisy nature of MS/MS data. In order to control false identifications and missed true identifications, most of these algorithms are supplemented with post-processing and statistical validation of scores of top-ranking PSMs (See the section on Evaluation, later), which essentially provides a score threshold above which an identification is 'considered' correct. Attempts have also been made to improve identification confidence by combining scores from more than one database search algorithm (Searle et al. 2008).

In addition to the obvious sources of error such as the existence of novel peptides (that are either not present in the database or contain an unknown PTM), or incorrect charge-state assignment to peptides, incorrect peptide assignment to spectra also occurs due to inadequacies resulting from using over-simplified fragmentation models (Ma 2010). Essentially, most popular scoring algorithms rely on models and scores/features that either completely ignore or underutilize the intensity dimension of MS/MS spectra, where the effect of fragmentation chemistry on peptide fragmentation behavior is responsible for different heights of peaks in an observed spectrum. Instead, a naïve theoretical spectrum (equal intensities for various fragments, or some similar variant) for each candidate peptide is used for comparison with the experimental spectrum. Similarity based on such comparison is understandably prone to error.

As discussed in several recent studies, intensity patterns have been shown to be reproducible under similar experimental conditions, and hence theoretically predictable, at least for certain key amino acid residues like enhanced N-terminal cleavage at Proline (Vaisar et al. 1996; Breci et al. 2003). Influence of presence, as well as positions in the peptide chain, of basic residues like Arginine, Lysine and Histidine, and of acidic residues like Aspartic and Glutamic acids, have also been studied (Kapp et al. 2003; Tabb et al. 2004; Tsaprailis et al. 2004; Huang et al. 2005). In addition to computational approaches from large datasets of PSMs, much research has also been done to enhance the understanding of fundamental biochemical principles of peptide fragmentation in a tandem mass spectrometer (Paizs et al. 2005). Particularly important is the "Mobile Proton Theory" which associates the fragmentation efficiency of a peptide with the mobility of the added charge along the peptide chain, and confirms the influence of physicochemical content of peptides on their observed fragmentation patterns (Wysocki et al. 2000; Huang et al. 2005). In light of these developments, attempts are being made to develop the next generation of scoring systems, which try to capture the influence of these physicochemical properties on the occurrence and intensity patterns of different fragment ion-types.

For example, Elias et. al. used probabilistic decision trees (PDT) (Jensen et al. 2007) which represented peptide fragments as a set of 63 features including fragment ion-type, length of peptide and fragment, gas-phase basicity, hydrophobicity and helicity of flanking amino acids and charge-state among others (Elias et al. 2004; Gibbons et al. 2004). The models were trained using a large set of high-confidence PSMs (assigned using Sequest) for learning the (discretized) intensity distributions for different combinations of the feature values. The learning algorithm automatically picked up the most significant features to explain the intensity distributions, and only few out of the list of 63 were actually utilized. Their likelihood ratio scores, when used in conjunction with Sequest scores were shown to reduce peptide identification error rate significantly. Zhou et. al. used similar properties with Bayesian Artificial neural network (Bishop 1996) for predicting intensities of the commonly observed b- and y-ions and showed that additional features (than those utilized by PDT) were important as well for predicting the peak intensities (Zhou et al. 2008). Although a significant advance, these algorithms assume independence of fragments and ignore any correlations that might exist in series of observed ion intensities.

Klammer et. al. used a Dynamic Bayesian network (DBN) (Murphy 2002) to model the intensities of different fragment ion-types, individually as well as in pairs, and utilized a smaller set of features like flanking amino-acids, position of cleavage in the peptide chain and fragment-ion detectability (Klammer et al. 2008). Normalized ranks of fragment intensities were represented as a mixture of Gaussians, the parameters of which were conditioned on the physicochemical properties. Likelihood ratio scores were computed from all the models (fragment and fragment-pairs) and were then fed as features into a support-vector machine (SVM) (Cortes et al. 1995; Vapnik 1998) to discriminate true from false identifications. Along with superior identification performance, their probabilistic models were also able to discover some new fragmentation patterns, establishing the significance of their approach. Khatun et. al. have used a complex Hidden Markov Model (HMM) (Rabiner 1989) to model intensity dependence on fragment ion types and their mass distributions, as well as on flanking amino acids. Viterbi algorithm was used to automatically determine whether a peak is a noise or a true fragment ion (Khatun et al. 2008).

As is evident from all of the above studies, several properties at peptide and spectrum level are utilized by different algorithms for assigning peptides to MS/MS spectra. Their complex interactions are either heuristically determined or learned automatically from large datasets of validated high-confidence PSMs and utilized in scoring systems. Nevertheless, the performance is still far from optimal in terms of utilization of the large volumes of data generated and a large fraction of spectra remain unconfidently assigned with peptides. Older algorithms like Sequest, Mascot and X!Tandem are quite mature and several studies have performed evaluation and comparison on different datasets demonstrating their similarities and differences (Kapp et al. 2005). They still are the most predominant algorithms in use and a recent study highlighted that more than 90% of investigations use some combination of them to analyze their datasets (Kandasamy et al. 2009). The more recent intensity-based models are still in their infancy and are being developed and refined. These models are powered by automated analyses and model building from large datasets of previously identified spectra, as evidenced by the above sampling of recent algorithmic development in this domain. Since the prime focus of this thesis is to utilize machine learning (ML) methods to model complex peptide fragmentation-intensity patterns from MS/MS data, in the next few sections I discuss the relevant fundamental concepts in ML that were used in developing the CSPI framework.

2.4 MACHINE LEARNING CONCEPTS

Recent surge of technology and experimental protocol in the biomedical domain has radically transformed the field into a quantitative science where large amount of data are routinely utilized to discover or test hypotheses of interest. Towards this end automated Statistical and Machine Learning (ML) methods have become a standard tool in a researcher’s toolbox to deal with and build from these data computer-based models of some partially or completely observable phenomenon. These methods are particularly suitable for complex domains where little prior knowledge is available to develop deterministic mathematical models. However, with certain assumptions models with practical utility can still be constructed using previously observed data. The same is true for the field of Computational Proteomics too, where now there exist several huge data repositories that store raw as well as annotated MS/MS data from simple to complex experiments (Craig et al. 2004; Martens et al. 2005; Desiere et al. 2006).

ML typically involves the steps of identification of the learning task, collection of prior experience (in the form of training examples) and a measure to evaluate the performance of the learner. Some of the end goals include using the models for prediction and forecasting, classification, explanation or grouping of entities involved, and each can usually be put in the form of a well-defined objective function that must be optimized, such as the overall cost of making mistakes (Bishop 2007).

2.4.1 Logistic Regression

Logistic regression (Hosmer et al. 2000) is a classical supervised statistical learning algorithm that is used to predict the probability distribution of a discrete outcome variable Y (i.e., Y takes on a value from a finite set, like {0, 1} in the case of binary classification problems) based on observed values of one or more predictor variables $\mathbf{X} = <X_1, X_2, \ldots, X_n>$, where each X_i could be discrete or continuous, i.e. $P(Y \mid \mathbf{X})$. For example, in a medical diagnosis problem, **X** represents a bunch of symptoms that a patient presents with, while Y is the unknown but desired disease status (healthy or sick) to be predicted. In order to learn such a mapping or classifier function, a set of training examples of the form $\{(Y, \mathbf{X})_i;\ i=1,2,3,\ldots,N\}$ are used with known values of covariates $\mathbf{X_i}$ and corresponding class labels Y_i. The learning task then consists in selecting the functional form of the classifier, a method for training the parameters of the function, and a performance measure. Logistic Regression assumes a parametric form for the distribution P(Y|X), which for binary Y is mathematically represented (in log-odds formulation) as:

$$\text{log-odds} = \ln\left(\frac{\mathrm{p}}{1-\mathrm{p}}\right) = \beta_0 + \sum_{i=1}^{n} \beta_i * X_i \qquad (1)$$
$$\text{where}\{\beta_{\mathrm{i}}\} = \Theta \text{ are the parameters, } \mathrm{p} = \mathrm{P}(\mathrm{Y}=1 \mid \mathrm{X}, \Theta)$$

In the case that Y may take more than two possible values, the model is called ‘multinomial logistic regression’. Suppose Y can take on ‘k’ possible values from the set {0, 1, 2, …, k-1}, the log-odds form is represented as:

$$\text{log-odds} = \ln\left(\frac{\mathrm{p_j}}{\mathrm{p_0}}\right) = \beta_{0j} + \sum_{i=1}^{n} \beta_{ij} * X_i, \mathrm{j} = 1, 2, 3, \ldots, \mathrm{k}-1 \qquad (2)$$
$$\text{where}\{\beta_{\mathrm{ij}}\} = \Theta_j \text{ are the parameters for the j}^{\mathrm{th}}\text{ class}, \mathrm{p_j} = \mathrm{P}(\mathrm{Y}=\mathrm{j} \mid \mathrm{X}, \Theta)$$
$$\text{and } \mathrm{p_0} = P(Y=0 \mid X, \Theta) = 1 - \sum_{j=1}^{k-1} p_j$$

Multinomial logit model is equivalent to (k-1) linear (on log-odds scale) expressions for representing the distribution of ‘k’ possible values of Y. In binary as well as multinomial case, the odds are computed with respect to a base class, which in the present case is Y=0. The models in the above equations can be interpreted as follows: “If x_i increases by one unit, log-odds for the outcome class (Y=j) w.r.t. base class changes by β_{ij} units”. Different values of parameters control the decision boundary learned for classifying the samples into individual classes.

2.4.2 Maximum Likelihood Training

The parameters of a statistical model can be trained using Maximum-Likelihood Estimation (MLE) approach (Casella G 2001). As the name suggests, MLE produces parameter estimates,

$\mathbf{\Theta}^{\mathbf{MLE}}$, that correspond to the probability distribution that generates the observed data with the greatest likelihood.

Suppose the observed Data **D** = {d_i; i=1, 2,…, N} consists of N independent and identically distributed (i.i.d.) observations from a probability density function with an assumed functional form $f_{\mathbf{\Theta}}$, where **Θ** are the parameters of the model. The first step in MLE is to write the joint distribution of **D**:

$$p(D \mid \Theta) = \prod_{i=1}^{N} f(d_i \mid \Theta) \qquad (3)$$

When the data samples are observed and the parameters are unknown, then (3) above is called the Likelihood function ***L*(Θ|*D*)**, and is a function of the parameters **Θ**. The goal is then to maximize ***L*(Θ|*D*)**. In practice, it is much more convenient to work with the (natural) log transformation of ***L*(Θ|*D*)**, called the Log Likelihood Function, ***l*(Θ|*D*)**. The purpose of using log-transformation is to simplify the computation by converting products to summations; this doesn’t affect the final outcome because log function is monotonically increasing. The maximum likelihood parameters’ estimates are then given by:

$$\begin{aligned}\hat{\Theta}^{MLE} &= \underset{\theta \in \Theta}{\arg\max}\, l(\theta \mid D) \\ &= \underset{\theta \in \Theta}{\arg\max}\, \ln(L(\Theta \mid D)) \\ &= \underset{\theta \in \Theta}{\arg\max} \sum_{i=1}^{N} p(d_i \mid \Theta) \qquad (4)\end{aligned}$$

For Logistic Regression, one way to estimate parameters through MLE is to maximize the conditional data likelihood or equivalently its log-transformation, i.e. the probability of the observed Y conditioned on covariates **X**. Mathematically, this can be represented as:

$$\hat{\Theta}^{MLE}_{Logistic\ Regression} = \underset{\theta\in\Theta}{\arg\max}\left(\sum_{i=1}^{N}\ln(P(Y_i \mid X_i,\theta))\right) \qquad (5)$$

The expression $P(Y_i \mid X_i, \theta)$ can be easily obtained from log-odds formulation of Logistic regression in equation 1 and equation 2 above. Since there is no closed form solution for this expression, one must resort to iterative numerical methods based on gradient ascent, like Newton-Raphson.

2.4.3 Input-output Hidden Markov Models (IO-HMMs)

Classic Hidden Markov Models (HMM) have been successfully applied to many sequential data-mining problems in biology and elsewhere that have to deal with data containing sequential structure, like those involving gene and protein sequences (Rabiner 1989; Durbin R 1999). Some representative examples include “Gene-prediction” (Henderson J 1997) and “Protein secondary structure prediction” (Karplus K 1999). Learning and inference from such models incorporates the sequential dependencies that are characteristic of such data. They derive their strength and flexibility from the hidden-state representation of ‘past context’, while restricting direct long-range interactions using Markov assumption. In the discussion that follows, a generic sequence is denoted as $<x_1x_2\ldots,x_t\ldots x_T>$, where ‘t’ refers to the location within the sequences being modeled while ‘T’ is the total length of the sequence. The same length ‘T’ is used for all sequences for notational convenience.

HMMs (Figure 6A) consist of an observation or emission sequence $y_1y_2\ldots y_T$, and represent the joint conditional probability distribution $P(y_1y_2\ldots y_T| \mathbf{\Theta})$, where $\mathbf{\Theta}$ are the model parameters. An intermediate hidden layer (unobserved) $<q_1q_2\ldots q_t\ldots q_T>$ facilitates modeling sequential dependencies. For example, in the problem of identifying protein-coding regions in a nucleotide sequence, the observation sequence can be the nucleotide sequence while the hidden states may represent the possible group labels (intronic region vs. exonic region). The goal would then be to compute the 'most likely' hidden-state sequence providing the desired group labels to individual nucleotides in the sequence. Mathematically, HMMs are represented as the parameters' tuple Θ = (π, A, B) where:

➔ **π :** Initial state probability distribution, $P(q_1)$

➔ **A :** Transition probability distribution matrix, $P(q_t \mid q_{t-1})$

➔ **B :** Emission/observation distribution, $P(y_t \mid q_t)$

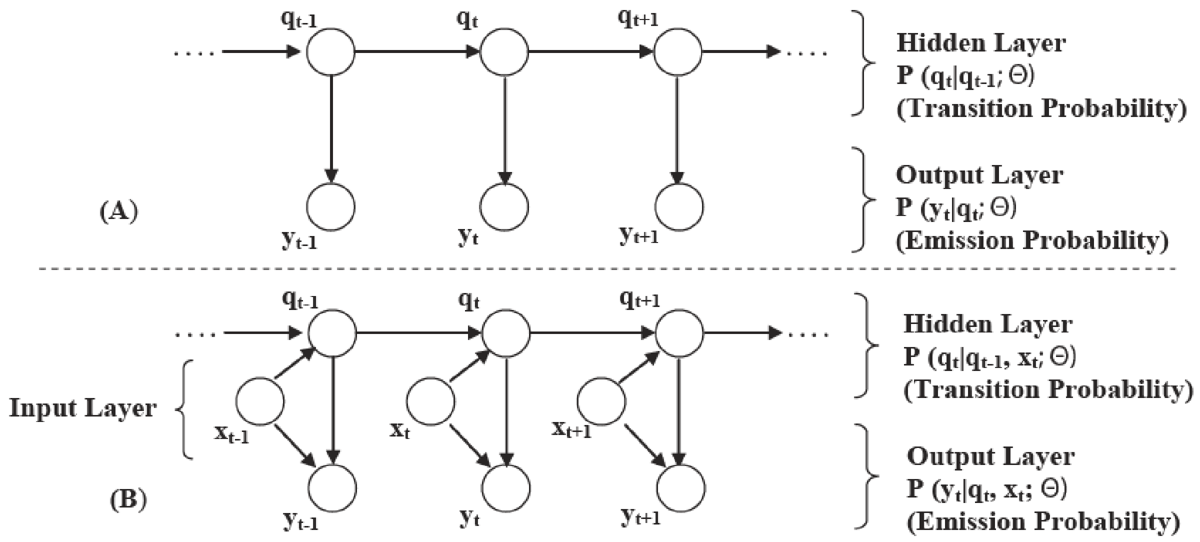


Figure 6. A) Classical Hidden Markov Model; **B)** Input-output Hidden Markov Model

The underlying assumptions in HMMs are: first, the presence of a hidden state-space that can correspond to the different (observation) data generating processes and a first-order markov assumption, which states that the probability distribution of current state is dependent only on the preceding state.

IO-HMMs are an extension of HMMs and are used to stochastically model sequence pairs rather than individual sequences (Bengio et al. 1995). So, in addition to an observation (output) sequence, there is another input sequence (also observed). The graphical structure of a basic IO-HMMs is shown in Figure 6B. As can be seen, IO-HMMs contain extra nodes (than HMMs) for the input sequence $<x_1, x_2, \ldots, x_T>$, which can probabilistically influence the output layer and/or the hidden states, represented as $<y_1, y_2, \ldots, y_T>$ and $<q_1, q_2, \ldots, q_T>$ respectively. They represent the joint conditional probability distribution $P(y_1y_2\ldots y_T| x_1x_2\ldots x_T; \boldsymbol{\Theta})$, where '$\boldsymbol{\Theta}$' are the model parameters**.**

Similar to HMM, an intermediate hidden layer $<q_1q_2\ldots q_t\ldots q_T>$ facilitates modeling sequential dependencies as complex probability distributions. However, the additional conditioning on the input layer makes the transition and/or emission probability distributions potentially non-stationary in location. This means that unlike HMM, instead of a transition matrix (or emission vector) of probabilities that remains fixed throughout the hidden markov chain, there is now a probabilistic function that takes the context (input features x_t) available at the specific location 't' under consideration, thus facilitating dynamic mapping of input-to-output sequences. Both x_t and y_t can be uni-variate or multi-variate, discrete or continuous, whereas the hidden states, q_t, are typically discrete. Additionally, the input sequence can be constructed with arbitrary features (from the domain) that may or may not overlap in location, allowing rich contextual information at local (specific location) as well as global (sequence) level to be incorporated in the sequence mapping tasks. The goal, then, is to learn the sequential dependencies between the input and the output. IO-HMMs have been successfully applied to several challenging sequential data-mining problems (Bengio et al. 2001; Ernst et al. 2007).

The state transition probabilities $q_{t-1} \rightarrow q_t$, and emissions y_t, conditioned on the input layer $x_1x_2 \ldots x_T$, can be represented by arbitrary probabilistic functions, as below:

$$P(q_t \mid q_{t-1}, \mathbf{x_t}) = f(q_{t-1}, \mathbf{x_t}) \qquad (6)$$

$$P(\mathbf{y_t} \mid q_t, \mathbf{x_t}) = g(q_t, \mathbf{x_t}); \qquad (7)$$

where '$\mathbf{x_t}$' is the input or context at location t, '$\mathbf{y_t}$' is the output or emission from the current hidden state, 'f' and 'g' are any linear or non-linear functions with valid probabilistic outputs.

In practice, there is one transition function for each hidden state, to compute the probability distribution of state at current location (q_t) given the state at previous location (q_{t-1}), i.e. $P(q_t \mid q_{t-1}, \mathbf{x_t})$. Likewise, there is one emission function for every hidden state, to compute the probability distribution of the emission/observation at the current location, given the state at current location, i.e. $P(\mathbf{y_t} \mid q_t, \mathbf{x_t})$. The parameterization for the architecture shown in Figure 6(B) is given in Table 1 below.

2.4.3.1 IO-HMM Training

The structure of the IO-HMM model is fixed apriori, in terms of the input and output layer representation as well as the number of hidden states, and should be reflective of the domain being modeled. Training, then, consists of estimating the parameters of the model structure from a training dataset, typically using MLE approach. Depending on the domain being modeled, the training dataset can be one input-output pair of very long sequences, or many such pairs of short sequences. The work in this thesis deals with the latter situation, but the methodology is trivially modified to the former case as well. MLE parameter estimation of IO-HMM models is a little more involved than the procedure for Logistic Regression described above due to presence of hidden variables ($<q_1 q_2 \ldots q_T>$).

Table 1. Notation and Parameterization for the IO-HMM architecture in Figure 6(B)

Symbol	**Description**
s	Number of hidden states
$\mathbf{S} = \{S_1, S_2, \ldots, S_s\}$	Set of hidden states
$\mathbf{X_n} = x_{n,1}, x_{n,2}, \ldots, x_{n,tn}, \ldots, x_{n,Tn}$	Input sequence for nth training sample; T_n: length of nth input training sequence
$\mathbf{Q_n} = q_{n,1}, q_{n,2}, \ldots, q_{n,tn}, \ldots q_{n,Tn}$	State transition sequence for nth training sample; Each of $q_{n,t} \in \mathbf{S}$
$\mathbf{Y_n} = y_{n,1}, y_{n,2}, \ldots, y_{n,tn}, \ldots, y_{n,Tn}$	Output sequence for nth training sample
$\mathbf{\Theta} = (\pi, A, B)$	Model parameters
$\mathbf{\Pi}$	Initial-state probability model parameters
$\mathbf{A} = \{\mathbf{A_i}; i = 1, 2, \ldots, \mathbf{s}\}$	Transition-probability models' parameters; **'A_i'**: set of parameters for transition model for ith hidden state
$\mathbf{B} = \{B_j; j=1,2, \ldots, \mathbf{s}\}$	Emission models' parameters **'B_j'**: set of parameters for emission model for jth hidden state

Let ***D*** be the (***observed***) training dataset comprising of **N** "independent and identically distributed" (iid) samples, which in the present case are pairs of input/output sequences:

$$
\begin{aligned}
D &= \{d_n = (X_n, Y_n); n = 1,2,\ldots,N\} \\
&= \{(<x_{n,1}, x_{n,2}, \ldots, x_{n,T_n}>, <y_{n,1}, y_{n,2}, \ldots, y_{n,T_n}>); n = 1,2,\ldots,N\} \qquad (8)
\end{aligned}
$$

Given the parameters **Θ** and the probability density function *p*(.|**Θ**), MLE parameters can be obtained by maximizing the conditional data log-likelihood as:

$$
\begin{aligned}
\overset{\wedge}{\Theta}_{mle} &= \underset{\theta\in\Theta}{\operatorname{argmax}}\, l(\theta \mid D) \\
&= \underset{\theta\in\Theta}{\operatorname{argmax}} \left(\sum_{n=1}^{N} \ln(P(Y_n \mid X_n; \Theta)) \right) \\
&= \underset{\theta\in\Theta}{\operatorname{argmax}} \left(\sum_{n=1}^{N} \ln(\sum_{q_n} P(Y_n, q_n \mid X_n; \Theta)) \right) \qquad (9) \\
&\text{(Marginalizing over hidden states)}
\end{aligned}
$$

Marginalizing over the hidden (missing) states leads to a summation expression inside the natural log. If the state transitions were known for each of the i.i.d. samples, this summation would vanish and the expression could be optimized directly with any gradient-based algorithm, like conjugate gradients. That not being the case, one must to resort to the numerical optimization method called the "Expectation Maximization" (EM), which is the standard methodology for MLE approach to parameter estimation in the presence of missing data (Dempster A 1977).

2.4.3.2 Expectation Maximization (in context of IO-HMM)

Let us define the '***complete data'***, as:

$$
\begin{aligned}
D^C &= \{d_n^c = (X_n, q_n, Y_n); n = 1,2,\ldots,N\} \\
&= \{(< x_{n,1}, x_{n,2},\ldots,x_{n,T_n} >, < q_{n,1}, q_{n,2},\ldots,q_{n,T_n} >, < y_{n,1}, y_{n,2},\ldots,y_{n,T_n} >); n = 1,2,\ldots,N\} \qquad (10) \\
&\quad \text{where} < q_{n,1}, q_{n,2},\ldots,q_{n,T_n} > \text{is a specific set of state transitions (for n}^{\text{th}} \text{ iid sample)}
\end{aligned}
$$

The corresponding Likelihood and log Likilihood functions are called the "complete data likelihood", [CDL, $\boldsymbol{L^C(\Theta|D)}$], and the "complete data log likelihood" [log(CDL), $\boldsymbol{l^C(\Theta|D)}$], respectively. The optimization operation is broken down into two steps of the EM algorithm, which iteratively improves the parameter values starting from a random initialization. The two steps of EM are mathematically represented as follows:

1. Expectation (E-step):

$$Q(\Theta,\Theta^k) = E_{q|x,y}[l^C(\Theta; D^C \mid x, y, \Theta^k) \qquad (11)$$

2. Maximization (M-step):

$$\Theta^{k+1} = \underset{\Theta}{\arg\max}\; Q(\Theta,\Theta^k) \qquad (12)$$

The E-step takes the expectation of the log of complete data likelihood [log(CDL)], which amounts to estimating the missing data (hidden states), conditioned on the previous estimate of model parameters (assumed correct). The M-step maximizes the resulting function of the parameters to get an improved estimate of the parameter-set. In the standard case, the EM begins with a random initialization of all parameter values in the model. The above two steps are applied iteratively to improve the parameter estimates until a local maxima is obtained. It can be shown that for convergence only an increase in Q function of E-step is required to guarantee an increase in the ***L*(Θ|*D*)** in each subsequent iteration. Hence, in case when M-step cannot be maximized in a closed-form, one needs only to ensure an increase in the Q function value in each iteration rather than maximizing it. This is called the Generalized EM algorithm (GEM) (Dempster A 1977). Several computational steps are required to perform each iteration of EM, general details of which can be found in (Bengio et al. 1995).

Using the 1st order markov assumption and independence relations that follow from the graphical structure, $\boldsymbol{L^C(\Theta|D^C)}$ is factorized as follows:

CDL:

$$L^C(\Theta;D^C)=\prod_{n=1}^{N}P(y_n,q_n\,|\,x_n;\Theta)=\prod_{n=1}^{N}\prod_{t=1}^{T_n}P(y_{n,t},q_{n,t}\,|\,q_{n,t-1},x_{n,t};\Theta)$$

$$=\prod_{n=1}^{N}\prod_{t=1}^{T_n}P(y_{n,t}\,|\,q_{n,,t},x_{n,t};\Theta)*P(q_{n,t}\,|\,q_{n,t-1},x;\Theta) \qquad (13)$$

(Using Markov Assumption)

Now, taking all possible values of hidden state variables $q_{n,t}$ and taking log on both sides, the log(CDL) is obtained as:

log(CDL):

$$l^C(\Theta;D^C)=\log\left(\prod_{n=1}^{N}\prod_{t=1}^{T_n}\prod_{i=1}^{S}\prod_{j=1}^{S}\big(P(y_{n,t}\,|\,q_{n,t}=i,x_{n,t};\Theta)\big)^{z_{n,i,t}}*\big(P(q_{n,t}=i\,|\,q_{n,t-1}=j,x_{n,t};\Theta)\big)^{z_{n,i,t}*z_{n,j,t-1}}\right)$$

$$=\sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S}z_{n,i,t}*\log\big(P(y_{n,t}\,|\,q_{n,t}=i,x_{n,t};\Theta)\big)+\sum_{j=1}^{S}z_{n,i,t}*z_{n,j,t-1}*\log\big(P(q_{n,t}=i\,|\,q_{n,t-1}=j,x_{n,t};\Theta)\big)$$

(where, $z_{n,i,t}$ is an indicator variable which takes the value 1 if $q_{n,t}$ = i, and 0 otherwise)

……(14)

As described above, since $z_{n,i,t}$ and $z_{n,j,t-1}$ are unknown, expectation is evaluated with respect to the distribution of the hidden state transitions conditioned on data ***D*** and current 'guess' of the parameters, $\mathbf{\Theta}^{\mathbf{k}}$. This gives the Q function of the E-step (See (Bengio et al. 1995) and Appendix A for details):

$$Q(\Theta,\Theta^k) = E[l^C(\Theta;D) \mid x,y,\Theta^k]$$
$$= \sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{i=1}^{S} g'_{n,i,t} * \log\left(P(y_{n,,t} \mid q_{n,t}=i, x_{n,t};\Theta)\right) + \sum_{j=1}^{S} h'_{n,i,j,t} * \log\left(P(q_{n,t}=i \mid q_{n,t-1}=j, x_{n,t};\Theta)\right)$$
$$\text{where } g'_{n,i,t} = P(q_{n,t}=i \mid x_n, y_n;\Theta^k) \text{ and } h'_{n,i,j,t} = P(q_{n,t}=i, q_{n,t-1}=j \mid x_n, y_n;\Theta^k) \qquad \text{......(15)}$$

$g'_{n,i,t}$ in (15) above represents the posterior state probability, that tth observation for the nth training sample (input-output pair) appears from the hidden state '*i*'. $h'_{n,i,j,t}$, on the other hand, represents the posterior state-pair probability, that the observation pair at locations (t-1, t) appears from the hidden state-pair (j,i). Both g' and h' are conditioned on the current 'guess' parameters $\mathbf{\Theta}^{\mathbf{k}}$ and the nth training sample, and hence the label 'posterior'. In order to compute g' and h', first the forward and backward recursion matrices for the nth training sample are computed (Rabiner 1989; Bengio et al. 1995). These expressions are very similar for classic HMMs except that everything is now conditional on the input sequence.

Defining forward variable, $\alpha_{n,i,t}$, as the probability of observing the partial sequence $<y_{n,1}, y_{n,2}, \ldots, y_{n,t}>$ and ending in the hidden state $q_{n,i,t} = i$ (conditioned on the partial input sequence $<x_{n,1}, x_{n,2}, \ldots, x_{n,t}>$), the α-matrix can be expressed and filled recursively as:

$$1.\ \alpha_{n,i,t=1} = P(q_{n,t=1}=i \mid x_{n,t=1};\Theta^k)$$
$$2.\ \alpha_{n,i,t=2\ldots T_n} = \left(\sum_{j=1}^{S} \alpha_{n,i,t-1} * P(q_{n,t}=i \mid q_{n,t-1}=j, x_{n,t};\Theta^k)\right) * P(y_{n,t} \mid q_{n,t}=i, x_{n,t};\Theta^k) \qquad (16)$$

Similarly, the backward variable, $\beta_{n,i,t}$, is defined as the probability of observing the partial sequence $<y_{n,t+1}, y_{n,t+2}, \ldots, y_{n,Tn}>$ given that the hidden state at time t is '$q_{n,t} = i$'. The β-matrix can be expressed and filled recursively as:

$$1. \beta_{n,i,T_n} = 1$$

$$2. \beta_{n,i,t=T_n-1....1} = \sum_{j=1}^{S} \left(\beta_{n,j,t+1} * P(q_{n,t+1} = j \mid q_{n,t} = i, x_{n,t}; \Theta^k) \right) * P(y_{n,t} \mid q_{n,t+1} = j; \Theta^k) \qquad (17)$$

Given the Forward and Backward matrices, the posterior state and state-pair probabilities can be computed as follows (See details of derivations in (Bengio et al. 1995)):

$$g'_{n,i,t} = \frac{\alpha_{n,i,t} * \beta_{n,i,t}}{\sum_{j=1}^{S} \alpha_{n,j,t} * \beta_{n,j,t}} \qquad (18)$$

$$h'_{n,i,j,t} = \frac{\alpha_{n,j,t-1} * P(q_{n,t} = i \mid q_{n,t-1} = j, x_n; \Theta^k) * P(y_{n,t} \mid q_{n,t} = i, x_n; \Theta^k) \beta_{n,i,t}}{\sum_{u=1}^{S} \sum_{v=1}^{S} \alpha_{n,v,t-1} * P(q_{n,t} = u \mid q_{n,t-1} = v, x_n; \Theta^k) * P(y_{n,t} \mid q_{n,t} = u, x_n; \Theta^k) \beta_{n,u,t}} \qquad (19)$$

Computing (18) and (19) for each training sample gives the expression for the Q-function and completes the E-step of the EM algorithm. We now have a function of the parameters of the model (like in regular MLE with no missing data). The M-step then proceeds by maximizing (or increasing) the Q-function and substituting the old guess parameters $\mathbf{\Theta^k}$ with new parameter values $\mathbf{\Theta^{k+1}}$.

It is worth noting that in practice, the parameters for each sub-component of the model (**'s'** transition functions and **'s'** emission functions) split nicely in the Q-function so that each sub-component can be optimized independently of the other in the M-step. Training the IO-HMM via GEM requires the model to be initialized to some random initial parameter values which are then iteratively improved until the model converges to a local maximum in the likelihood space. Since the model is quite complex, it is possible that the likelihood surface is not unimodal with only one unique maximum-likelihood estimate of parameter set. The usual practice under these circumstances is to perform multiple rounds of training starting from a different random initial seed and choosing the parameters that maximize the likelihood among all rounds.

2.5 EVALUATION, SCORE COMBINATION AND POST-PROCESSING OF DATABASE SEARCH RESULTS

Evaluation is a critical step in automated methods for data analysis, and involves several aspects depending on the application and domain. Typically, one would like to establish how well the model either describes the data it was learned from, or how well it predicts on future unseen data. The main objective of the CSPI framework is confident assignment of peptides to MS/MS spectra and hence, the most important evaluation deals with how well the overall framework performs the task of differentiating true from false peptide identifications. Several methods have been described in the literature to deal with this problem and are discussed next.

As described above, each database search for a spectrum yields a list of candidates ranked according to their PSM scores. Typically only the top-ranking peptide is considered further for protein inference, although sometimes the true peptide appears at a lower rank. Due to multiple steps involved in the shotgun proteomics pipeline, numerous factors introduce biases and noise, making all scoring algorithms prone to errors. In typical large-scale proteomics experiments, over 75% of spectra are of poor quality or from non-peptide species and must be correctly distinguished from real signals (Ning et al. 2007). Due to high amount of noise in MS/MS spectra, false peptide assignments to these spectra can attain reasonably good PSM scores, and consequently, the distributions of scores from True PSMs and random/False identifications exhibit significant overlap, substantially increasing the identification error rates of search algorithms. It is particularly important for the Database searching algorithms, because they always return "the best available" answer even if incorrect.

Hence, in order to interpret and effectively utilize peptide identification results in downstream analyses, separating the two is a crucial step in the overall analysis and researchers are interested in knowing precisely the error rates of the algorithms used for their experiments, so as to fine-balance false-identifications with missed (estimated) true-identifications. This amounts to choosing a score threshold above (or below, depending on the score) which the PSM is considered significant (or true).

Arbitrary choice of score thresholds (derived empirically) to call a PSM true or false is an inadequate solution and does not perform well across different datasets. Additionally, the vast numbers of different scoring schemes described in the literature are quite varied in terms of what they represent and their scales of measurement. Such variation makes the scoring schemes and their thresholds incomparable directly. In order to better control the above aspects, statistical validation that transforms these arbitrary scores to a statistical significance measure is an essential analysis step.

This problem is that of hypothesis testing with null H_0: "PSM is a random match" and alternative H_a: "PSM is a true match". A test is considered significant if, under the null hypothesis/distribution, the observed value of the PSM-score is better than some threshold specified by the desired significance α-level. Alternatively, a traditional p-value indicates the probability that the under H_0, the PSM-score is at least as good as observed, while an e-value indicates the expected number of PSM-scores better than observed. Since for each spectrum it is assumed that only one candidate peptide can be possibly true, null distribution can be estimated from the scores for all except the top-scoring candidate. These procedures however provide an unsatisfactory solution due to large numbers of tests involved in a single experiment (classic multiple hypothesis-testing problem). As a result, a sizable proportion of tests can emerge significant just by random chance. Simple procedures for multiple-testing correction, like Bonferroni correction, will be too stringent.

2.5.1 False Discovery Rate (FDR) and Q-values

FDR is an alternative way of correcting for multiple comparisons and is a global error control measure, unlike p- or e-values, that directly controls type-I errors (incorrectly rejected null hypotheses) (Benjamini et al. 1995). In the context of peptide MS/MS, it is defined as the expected proportion of 'false' identifications in the entire set of 'significant' PSMs at a specified score threshold (Choi et al. 2008). For example, if a 1000 PSMs obtain a score better than the threshold 's', and the FDR is controlled at a level of 0.01, then at max 10 of these PSMs are expected to be false positives.

Estimating FDR requires a good choice of "null distribution" of PSM scores. One commonly used null model is that from a decoy database search (Elias et al. 2007). The database of true protein sequences (of the organism under consideration) is called the "Target" database. A decoy database is derived from target by some operation like reversing or shuffling all protein sequences, or using markov models to generate sequences that have the same distribution and dependencies of amino acids as the target database (Feng et al. 2007). A search against such a decoy database will return top-ranking hits, which by design, are random or false identifications, and their scores can be used as a representative for the null distribution.

One simple strategy for computing and controlling False Discovery Rates (FDR), based on target-decoy strategy is described in Kall et. al. (Kall et al. 2008). Briefly, after performing separate target and decoy searches, FDR at a score threshold, ***t***, is approximated as:

$$E(FDR(t)) = \pi_0 * (N_t / N_d) * (n_d / n_t) \qquad (20)$$

$$\begin{aligned}
&\text{where } n_{t(d)} : \text{\# top-ranked target (decoy) peptides with score} > \text{t} \\
&\qquad N_{t(d)} : \text{total number of targets (decoys)} \\
&\qquad \pi_0 : \text{estimated proportion of target PSMs that are incorrect} \\
&\qquad\quad \text{(fixed at 0.9 as described in Refs. 28, 34)}
\end{aligned}$$

The underlying assumption in target-decoy strategy is that the score distribution of incorrect target peptides is the same as that of decoy peptides. The usual practice is to keep the estimated FDR to as low as 1-5% or lower, obtain the corresponding score threshold, and determine how many peptides are identified with scores above the chosen threshold. These are then considered as “estimated true” identifications, and are used in downstream inference for protein identification. The significant advantage of this approach is its conceptual simplicity and minimal effort towards implementation.

Since FDR as computed above is associated with an entire set of PSMs, it loses a desirable property of being a monotonically non-increasing function of score, i.e. as the score threshold increases, FDR should not increase. A more useful measure, that is also associated with each individual PSM is the FDR analogue of p-value, called the q-value, and refers to the minimum FDR at which a given PSM is called significant, or a true PSM (null-hypothesis rejected) (Storey et al. 2003).

2.5.2 Score Combination, Post-processing

The procedures described above can be used to compare performance of different scoring algorithms, each of which yields some primary score of quality of match based on which candidate PSMs are ranked. However, these scores are far from perfect due to factors described earlier. Additionally, they vary a lot from one spectrum to another depending upon PSM properties (spectrum quality and noise-level, peptide’s propensity to fragment, charge-state etc.). As a result, several potentially true PSMs fall in the region of overlapping score distributions, particularly because the scores usually have arbitrary scales and may not be absolutely comparable from one instance to another. One definitive way to improve identification accuracies is to combine the primary score with other features of PSM match-quality, which are also reported alongside, or by combining the scores from multiple different algorithms (Searle et al. 2008). Often such combination provides additional complementary information and can significantly boost the performance. Several approaches of such post-hoc processing and combination have been developed to address this aspect; most popular ones are described next.

Peptide Prophet was developed and optimized for the Sequest database search algorithm and utilizes four numeric PSM quality features as well as observable discrete peptide properties (Keller et al. 2002). Using a manually-verified training dataset in the first stage, the algorithm learns a linear discriminant function to combine the features into a composite score. The next step combines the composite score with peptide properties using the Empirical Bayesian framework, assuming conditional independence between the composite score and peptide properties given the class label (which is essentially a Naïve Bayesian Classifier with hidden class variable). Two drawbacks of the original formulation – fixed discriminant function parameters across datasets and inflexible composite score distributions – were addressed in later extensions (Choi et al. 2008; Ding et al. 2008). Another extension also improved their performance using a semi-supervised approach utilizing decoys to learn a better null distribution (Choi et al. 2008).

Unlike Peptide Prophet, which is an unsupervised generative algorithm, the Percolator algorithm uses a more discriminative approach in semi-supervised setting. It uses target-decoy strategy together with Support Vector Machine (SVM) classifier to combine scores/features, learning new parameters for each new dataset (Kall et al. 2007). Percolator iterates over the following steps until convergence: a) Identify a set of high-confidence target PSMs to use as positive training data; b) using decoy PSMs as negative training data, train an SVM classifier; c) score the entire set of target PSMs using the trained SVM. The iterations are initialized using high-confidence targets based on SEQUEST cross-correlation score, while subsequent iterations use SVM-based discriminant score. Confidence is measured using q-value (as described earlier) based on these scores. The procedure converges when no new targets are identified at high confidence. According to the authors, this approach does better than Peptide Prophet due to a larger feature set and adaptive discrimination using SVMs that adjusts to peculiarities of each individual dataset.

3.0 CONTEXT-SENSITIVE PEPTIDE IDENTIFICATION FRAMEWORK

A novel Context-sensitive Peptide Identification (CSPI) Framework is proposed in this thesis for improving peptide scoring and identification from MS/MS data through modeling their fragmentation ion intensities. CSPI utilizes an instance of the flexible IO-HMM class of models to represent the complex peptide fragmentation intensity patterns in mass-spectrometers under low energy CID. The specific constrained structure of the model used for all analyses presented in this thesis, which is a special case of Figure 6(B) in section 2.4.3, is presented in Figure 7. For the application to peptide identification, the input contextual features (x_t) are derived from the peptide sequence while the output variables (y_t) are derived from spectrum intensities.

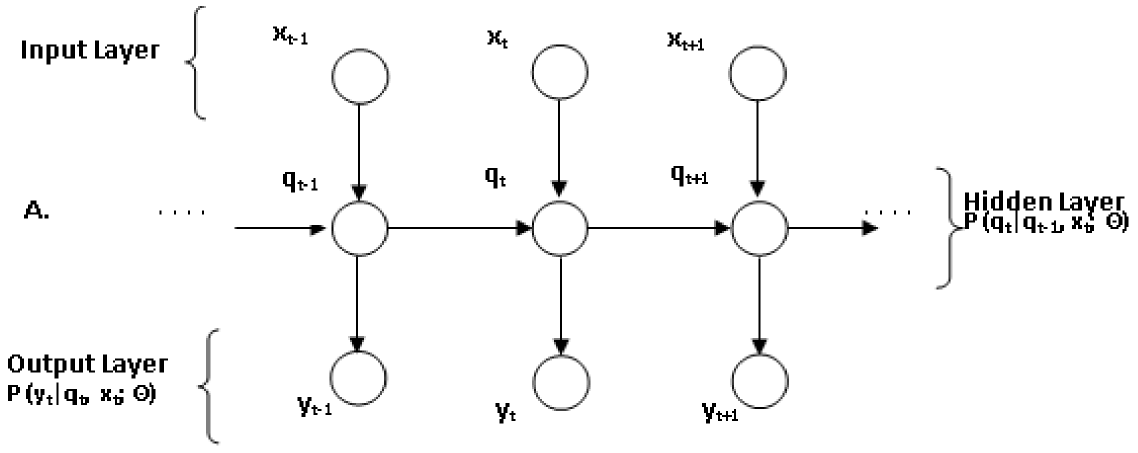


Figure 7. IO-HMM Structure used in the CSPI Framework. Both b- and y-ion models have the same structure with y_t representing observed b- and y-ion intensities respectively

This structure implies that the contextual features influence only the transition functions and not the emission functions. In effect, through the hidden states, the model learns complex mixture distributions of the output variable, conditioned on the input layer features and the previous hidden state distribution. The model can then be used to probabilistically evaluate how well a peptide's physicochemical properties are able to explain the observed fragment-ion intensity series in a spectrum, resulting in an intensity-based score.

As described earlier, a fragmentation event can produce several different ion types. From low-energy CID, b- and y-ions dominate and were used to develop and evaluate models in CSPI. In order to capture their distinctive characteristics and distributions, the b- and y-ions are modeled separately, and are called CSPI_b and CSPI_y, respectively.

Given the model structure, the next step is to transform a PSM pair into appropriate input-output format, requiring several preprocessing steps and fixing the functional forms of the components of the model, as are described next.

3.1 INPUT LAYER (<$X_1X_2…X_T$>)

In the current work, input layer is a sequential representation of the peptide sequence being evaluated. Each amide-bond position in the peptide sequence (from N- to C-terminus) is represented as a feature vector that forms the 'input' x_t at the corresponding location, and represents the global (peptide- or fragment-level) and local (fragmentation site-level) context influencing observed fragmentation. For example, a peptide of length 10 has 9 amide bond positions and is represented in the input layer as a sequence of 9 feature vectors, each being of same length as the number of features used. The same input features and representation are used for both b- and y-ion models. The directionality used in the b-ion models is from N-terminus to C-terminus, while that for y-ion models is from C-terminus to N-terminus.

The features used in the model are described in Table 2. The “Mob” feature uses an accepted definition of mobility [ChargeState – Number of Arg – 0.5*(Number of His + Number of Lys) (Huang et al. 2008). The mobility values were grouped into 4 bins as shown in the table. Similarly, ‘length’ feature was grouped into 3 categories: short (< 13), medium (between 13 and 22) and long (> 22), binned roughly at 25th and 75th percentiles of peptide lengths in the training dataset.

3.2 OUTPUT LAYER ($<Y_1Y_2\ldots Y_T>$)

The output layer consists of the sequence of observed intensities of the b- and y-ions of the peptide. In order to handle wide variation in the observed fragment intensities as well as to reduce the dominance of few high-abundance peaks, as part of building CSPI models certain pre-processing steps are performed on the MS/MS spectra before they are used either for learning model parameters or searched against databases for candidate peptides. This is crucial so as to make spectra more comparable across each other as well as across multiple datasets, and includes the following steps (in order of operation): a) Remove the peak corresponding to the precursor as this can be very intense and thus overshadow many other shorter peaks; b) Square-root transform all peaks in order to reduce the influence of very intense peaks; c) Normalize all the peaks so that the intensity of the tallest (base) peak is 100 while all other peaks are scaled accordingly; d) Filter noise peaks, where noise threshold is user-defined (default is set to 0.025,

Table 2. Contextual Features used in the input layer of CSPI models

Feature	**Type (Length)**	**Description**	**Influence**
N_{AA}	Binary (19)	Flanking N-terminal Amino acid (Considering Pro as baseline; 1 binary feature for remaining 19 possible)	Local
C_{AA}	Binary (19)	Same as N_{AA}	Local
FracMz	Numeric (1)	Fractional mass-to-charge (m/z) of fragment relative to the m/z of the parent peptide; Range: (0,1)	Local
Mob	Binary (3)	Mobility value of the peptide (<=0: baseline; one binary feature for 0.5, 1, >1)	Global
CTerm=R?	Binary (1)	Is the C-terminus of peptide Arg?	Global
K/H in b-fragment	Binary (1)	Is there a Lys or His in the b-fragment (other than N_{AA}/C_{AA})	Fragment
R in b-fragment	Binary (1)	Is there an Arg in the b-fragment (other than N_{AA}/C_{AA})?	Fragment
Length	Binary (2)	Length of the peptide, discretized into 3 bins (length<13: baseline; one binary feature each for 13<= len < 23 and 23<= len)	Global
Total	47		

i.e. 2.5% or lesser of the base peak); e) Remove the peaks below the low-mass cut-off region (default threshold used is 0.3 times the m/z value of the precursor; ion-trap instruments typically do not retain peaks in this region and filtering reduces the chance of modeling noise); f) select 200 most intense peaks (at max) of those that remain. Finally a normalized intensity value (described next) is used for each observed fragment at each fragmentation site. It is important to note that these spectrum pre-processing steps are a part of model building process and were applied (as described) only to the CSPI framework. Other algorithms that were used for comparison follow their own pre-processing protocols.

3.3 NORMALIZATION

Two different normalization schemes, called "Rank-norm" and "Window-norm", were evaluated.

For 'Rank-norm' scheme (after spectrum pre-processing), the peaks of the spectrum are assigned ranks, which are then normalized to range [0.001, 0.999], 0.001 being the highest intensity and 0.999 being the lowest. This normalization range was chosen instead of [0, 1] to avoid difficulties in parameter estimation for the emission distributions used for rank-norm scheme (see section 3.4 below). Such rank-based normalization has been used in recent studies in order to reduce variation, and makes intensities comparable across spectra (Wan et al. 2006; Klammer et al. 2008). Fragment ions that are not observed in the experimental spectrum are represented as "Null" in the output layer.

For the "Window-norm" scheme, (after spectrum pre-processing) the output/emission value used is the logarithm of the fraction of intensity explained by the fragment within +/-75 Da window around its m/z value, rationale being that the fragments from the true peptide should explain more abundant peaks than those from false peptides. Again, if a fragment is not observed the observation at that location is designated as “Null”.

3.4 PARAMETER REPRESENTATION

In the current implementation (Figure 7), the emission functions are represented as simple distributions, conditioned only on the hidden state value (q_t), i.e. $P(\mathbf{y_t} \mid q_t, \mathbf{x_t}) = P(\mathbf{y_t} \mid q_t)$. In CSPI models, IO-HMMs with **four hidden-state values** are used, out of which one is reserved for “Null” emission (i.e. when the fragment is not observed), and has emission probability of 1. The other three values correspond to observed fragments with continuous emission distributions. These can be thought of as states producing “Low”, “Medium” and “High” intensity observations (on average, determined by the “mean” of the emission distribution used). The distributions of observed (normalized) intensities of b- and y-ions from a large set of validated PSMs is shown in Figure 8 and Figure 9.

Similar patterns are observed for other datasets of validated PSMs and have guided the choice of appropriate emission distributions used in this thesis. For the rank-norm scheme, Exponential emission distribution was used for b/y-ions from true peptides and Beta distribution for those from false/random peptides. For the window-norm scheme, Gaussian emission distribution was used for both b/y-ions and from true and false peptides.

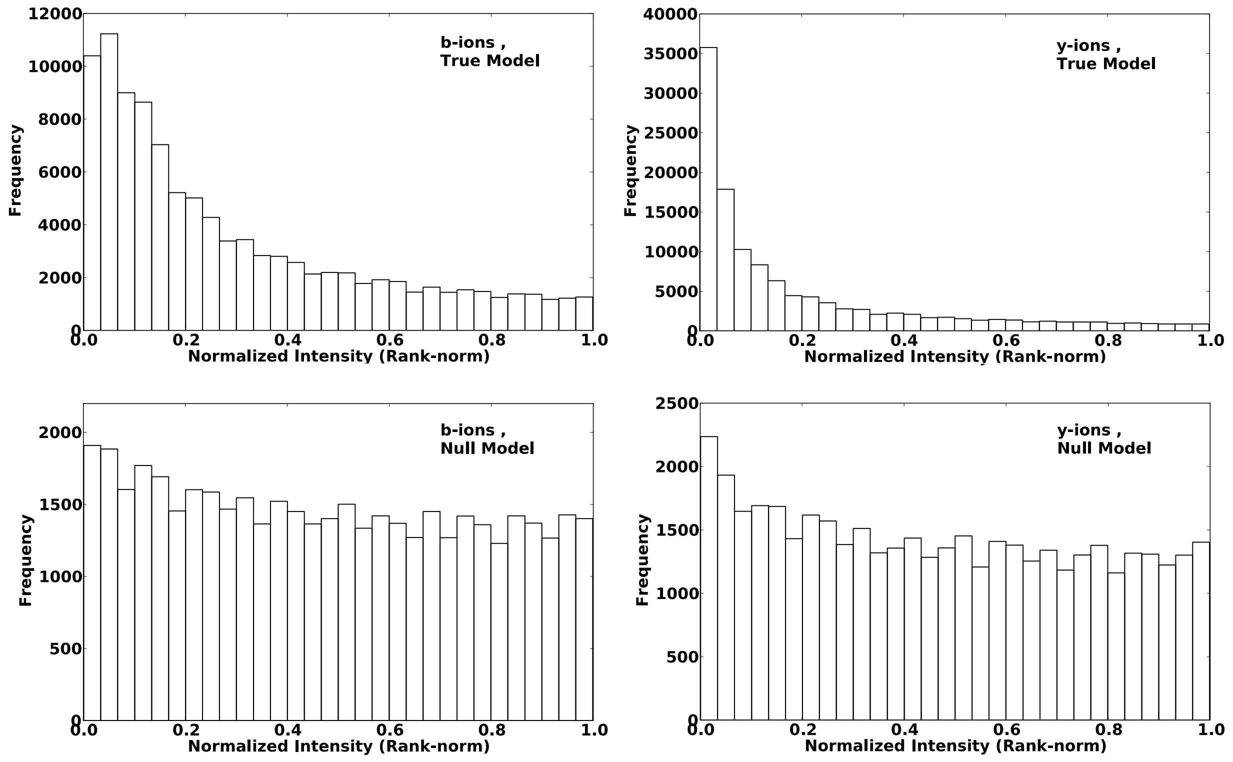


Figure 8. Distribution of observed rank-normalized intensities of b- and y-ions from True and False/Random PSMs, for SO-DR dataset (See Chapter 4 for dataset description).

3.5 TRANSITION FUNCTIONS

Each CSPI model structure results in one transition function for each hidden-state value. Given the hidden-state value $q_{t-1} = q$ ($t > 1$) and the context (input $\mathbf{x_t}$), the corresponding function provides the probability distribution over hidden-state values at current location t, i.e. $P(q_t \mid q_{t-1} = q, \mathbf{x_t})$. The output of this function changes as the input $\mathbf{x_t}$ varies along the peptide sequence. Additionally, there is an initial-state function for computing the distribution over hidden-state values at the start of the sequence, i.e. $P(q_{t=1} \mid x_{t=1})$. All these distributions are modeled using logistic functions.

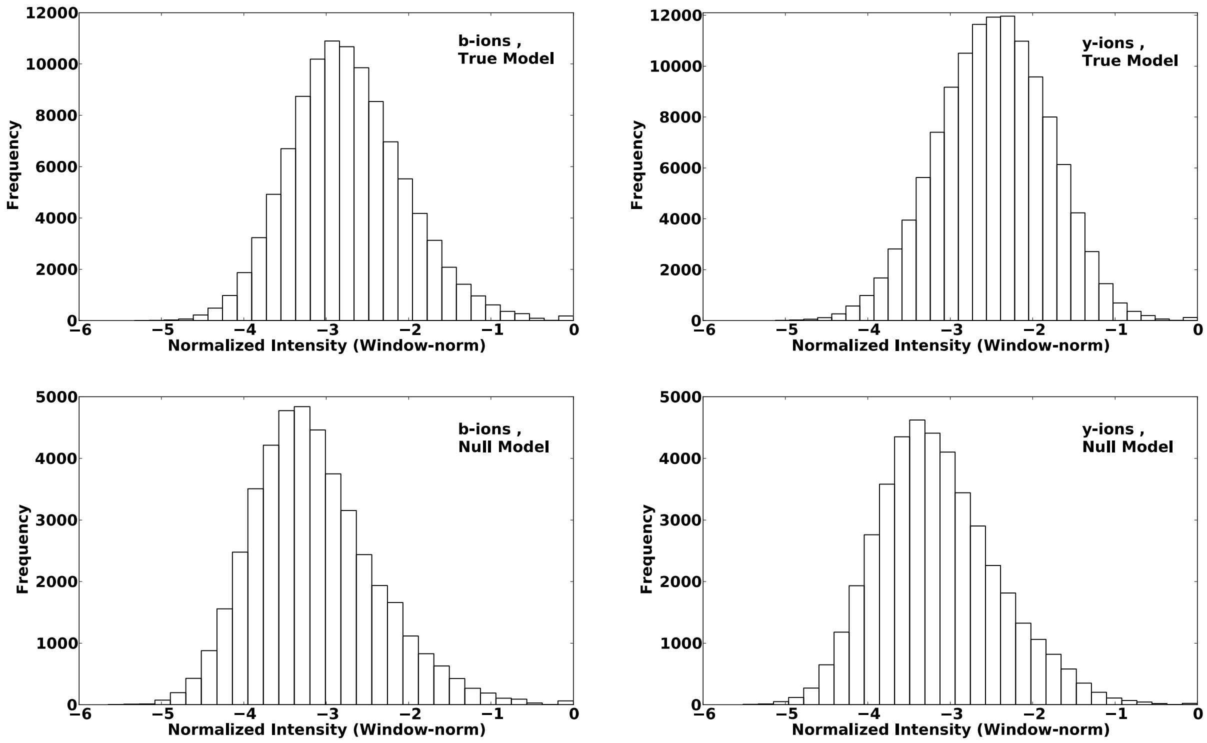


Figure 9. Distribution of observed rank-normalized intensities of b- and y-ions from True and False/Random PSMs, for SO-DR dataset (See Chapter 4 for dataset description).

In the current implementation, CSPI models with s=4 hidden-state values are used (For parameter notation, see Table 1). Mathematically, output of the i th transition logistic function is represented as:

$$P(q_t = j \mid q_{t-1} = i, x_t; \Theta_{A_i}) = \begin{cases} \dfrac{1}{1+\sum_{k=1}^{S}\exp(\mathbf{w}_k^T\mathbf{x}_t)} \; if\, y_t = "NA", j = 1 \\ \dfrac{\exp((\mathbf{w}_{j-1}^T\mathbf{x}_t))}{1+\sum_{k=1}^{S}\exp(\mathbf{w}_k^T\mathbf{x}_t)} \; if\, y_t != "NA", j = 2,3,\ldots,s \end{cases} \tag{21}$$

where $\mathrm{w}_{j(k)}^{\mathrm{T}}$ are the i th logistic transition function weight vectors, $\mathrm{A_i}$

Initial-state probability is computed in a similar fashion and uses its own logistic function model, the output of which gives the distribution over the hidden states at location t=1, i.e. $P(q_{t=1} \mid \mathbf{x}_{\mathbf{t=1}})$.

The initial-state and transition functions together predict the hidden state transition probabilities along the peptide sequence. Based on the sequence (context), some state transitions will be more likely than others. CSPI models compute probabilities over all such possible state transitions over the entire peptide chain, in order to compute the contribution to the overall score (See section 3.7 for details).

Each logistic model has ($s - 1 = 4\text{-}1 = 3$) weight vectors. Each weight vector is of size=(#input features + 1)=48. This leads to a total of 48*3=144 tunable parameters per logistic regression component. Since each CSPI model has 5 such logistic models (1 for initial state and 4 for each hidden state value), the total number of logistic functions' parameters per CSPI model is 144*5=720.

Further, each hidden-state value corresponds to an emission distribution. For s=4, and rank-norm scheme, this leads to 3 exponential distributions (for True models) and 3 beta distributions (for Null Models), for a total of 3+6=9 emission parameters. For window-norm scheme, three Gaussian emission models for each, True and Null models, are used for a total of 12 tunable parameters.

3.6 CSPI TRAINING

For the apriori fixed structure, in terms of input-output representation and number of hidden states, training consists of estimating the parameters of the model structure from a training dataset which comprises of a set of input-output sequence pairs $<(\mathbf{x_1 x_2 \ldots x_T}); (\mathbf{y_1 y_2 \ldots y_T})>_{\mathbf{i}}$, i=1,2,3…,N, where N is the size of the training dataset. These are derived from high-confidence and validated PSMs, with representations as described above. Parameter estimation in CSPI is done using the "Maximum-Likelihood" approach. Due to presence of hidden variables (<$q_1q_2…q_T$>) and absence of a closed-form solution, this is achieved using the iterative numerical optimization method called "Generalized Expectation Maximization" (GEM) (Dempster A 1977), as described in Section 2.4.3.1. For detailed derivation, see Appendix A**.**

3.7 CSPI INFERENCE

Trained CSPI models are used to score and rank candidate peptides obtained via Database Search for each spectrum. Inference involves evaluating the joint probability of observing the spectrum (a particular fragment ion series, b- or y-) given the peptide and the model (learned parameters), i.e., P($\mathbf{y_1y_2…y_T}$ | $\mathbf{x_1x_2…x_T}$; $\mathbf{\Theta}$). Let us consider a specific input-output sequence pair x = <$x_1x_2…,x_t…x_T$> and y = <$y_1y_2…y_t…y_T$>. Suppose we also know the hidden state transitions q=<$q_1q_2…q_t…q_T$> that generated the output y for this pair. The joint probability P(y, q | x; $\mathbf{\Theta}$) can be computed as:

$$\begin{aligned} P(y,q\,|\,x;\Theta) &= P(<y_1y_2..y_T>,<q_1q_2..q_T>|<x_1x_2..x_T>;\Theta) \\ &= [P(q_1\,|\,x_1;\Theta)*P(y_1\,|\,q_1;\Theta)]*[P(q_2\,|q_1,x_2;\Theta)*P(y_2\,|\,q_2;\Theta)]*..... \\ &\quad \text{(Using first order markov assumption on the hidden markov chain)} \\ &= \underbrace{P(q_1\,|\,x_1;\Theta)}_{(\,\mathbf{I}\,)}*\underbrace{\left(\prod_{t=2}^{T}P(q_t|q_{t-1},x_t;\Theta)\right)}_{(\,\mathbf{II}\,)}*\underbrace{\left(\prod_{t=1}^{T}P(y_t\,|\,q_t;\Theta)\right)}_{(\,\mathbf{III}\,)} \end{aligned} \qquad (22)$$

In expression (22), factor (I) is computed using the initial-state logistic function. Remember that for each hidden-state there is one transition logistic function and one emission function. So, factor (II) is computed using the corresponding transition logistic functions for state value q_{t-1} at each location t. Similarly factor (III) is computed using the corresponding emission function for the state value q_t.

Scoring a PSM involves computing the desired probability P(spectrum | peptide; **Θ**) or more generally P(**y** | **x** ; **Θ**). This expression requires summing over all possible hidden state transitions and can be computed using (22) above as:

$$P(y|x;\Theta)=\sum_{q} P(y,q|x;\Theta) \qquad (23)$$

To compute expression (23) efficiently, an extension of the Forward procedure used in classic HMMs is used, which follows similar mechanics except extra conditioning on the input layer at each step requiring computing the transition probability (Bengio et al. 1995).

In order to discriminate between true and false peptide identifications, two different models, one for true PSMs and one for random/false PSMs, are learned for each fragment ion-type. In each random PSM, the peptide sequence (input) used is a random/false sequence of (nearly) same mass as the true peptide. These models are called the True and the Null models, with parameters $\mathbf{\Theta}^{\mathbf{True}}$ and $\mathbf{\Theta}^{\mathbf{Null}}$, respectively. The score for a candidate PSM then is computed as the log of likelihood ratio of the spectrum conditioned on the input peptide, from the true and null models:

$$CSPI\_Score = \log\left(\frac{P(y_{1,2,\ldots,T} \mid x_{1,2,\ldots,T};\Theta^{True})}{P(y_{1,2,\ldots,T} \mid x_{1,2,\ldots,T};\Theta^{Null})}\right) \qquad (24)$$

(numerator and denominator are computed as described above, from respective models)

Scores from b- and y-ion models are computed in similar fashion, by replacing the output sequence $<\mathbf{y_1 y_2 \ldots y_T}>$ with the normalized intensities of appropriate fragment ion series, b or y. Peptide identification performance is evaluated for three PSM scores: (i) $CSPI\_Score^{b}$ (from model CSPI_b), (ii) $CSPI\_Score^{y}$ (from model CSPI_y), and (iii) composite $CSPI\_Score^{byAdded}$ $(= CSPI\_Score^{b} + CSPI\_Score^{y})$.

3.8 SCORE COMBINATION WITH LOGITPERCOLATOR

Scores from individual fragment-ion models provide complementary information that must be used together to perform inference on any PSM pair being evaluated. Often times, other features of match quality are also available and can be used as additional sources of information, as is typically done during manual interpretation of spectra. In that case, the simple composite score, that adds the individual scores, is not the most optimal as it attaches equal weight to both components. This can be addressed by combining scores using machine-learning approaches that appropriately weight the contributions of individual components. Two prominent examples of this approach are PeptideProphet (Keller et al. 2002) and Percolator (Kall et al. 2007), both of which are automated methods to post-process database search results.

The goal in this part of the research was to develop a post-processor to appropriately weight and combine CSPI's individual intensity-based scores, as well as to demonstrate the utility of these features towards improving peptide identification performance when used in conjunction with other features popularly used in large-scale proteomics. To achieve this goal, a similar, albeit simplified, strategy as outlined in the Percolator algorithm (see section 2.5.2) is followed, with the following two differences: a) Instead of SVM, Logistic Regression classifier is used, and the posterior probability of target PSMs from the model is used as the composite score from which FDR and q-values are computed; b) no cost-matrix for errors in classification is learned. Due to these differences, the current implementation, which is called "LogitPercolator" for the remainder of this thesis, can be considered as a baseline.

3.9 EFFICIENT PROCESSING OF LARGE MS/MS DATASETS

As described earlier, associating the spectra with their true peptide identification involves searching large protein databases to score and rank potential candidates. Depending upon the size of the database and constraints applied on the search, like allowable post-translational modifications, enzyme specificity and possible charge-states, each spectrum may have to be evaluated against several thousand candidates to select the one that best explains the observed data. Additionally, a single MS/MS experiment from a modern mass-spectrometer can generate up to the order of 5-10K MS/MS spectra in less than an hour, resulting in several GB of data each day from even a moderate-sized proteomics lab. Analyzing such large datasets requires significant computation time, particularly when using complex scoring systems like the CSPI framework presented in this thesis. Hence, in order to keep pace with the volume and rate of data generation, the software system implementation must support efficient data processing. Efficiency was achieved for the CSPI using two strategies: a) Protein database indexing, and b) Parallel implementation using multiprocessing.

3.9.1 Protein Database Indexing

The first step in analysis is the database search component that involves extracting candidate peptides for each spectrum by querying a protein database, which is a simple ASCII text file with a list of protein sequences or character strings (the protein alphabet is of size 20, with each character being of a different mass). This amounts to a range query on the "expected mass" of the true peptide. Protein databases can be large and the naïve approach of scanning them afresh for each spectrum for retrieving strings of required mass will be prohibitive in terms of time. Most systems pre-compute once some form of index for fast querying, and similar strategy was followed within CSPI.

In order to create indexes for protein databases, the appropriate protein FASTA file was preprocessed to generate the list of all possible peptides satisfying the desired search constraints for database search. These are then indexed used the python interface for the Berkeley DB key-value database (Olsen et al. 1999), where the 'key' is the string representation of peptide mass up to one decimal point; and 'value' is the string concatenation of peptide's location in the database (protein number as it appears in the FASTA file, and position number within the protein sequence) and length of the peptide. Additionally, in order to keep the size of index files small, the entire range of expected peptide masses is split into bins of size 25 units (arbitrarily chosen and may be optimized further), leading to multiple index files each storing a different mass region. Values of candidates with same keys are concatenated with a separator. Now, for every new query, the index allows for fast retrieval of candidates, by first mapping the query mass ("key") to the appropriate index file, followed by retrieval of candidates in the corresponding mass-region that meet the mass-tolerance search criterion, and reconstruction of the peptide sequences using the corresponding information stored in the "value" part of the key-value pair.

3.9.2 Parallel implementation using multiprocessing

The next step in database searching evaluates all the candidates retrieved for each spectrum. This is computationally the most expensive step in the peptide identification workflow. However, fortunately, this particular step is amenable to massive parallelization and can exploit large multiprocessor and/or distributed computing architectures to alleviate the computational bottleneck. Specifically, for each spectrum in the dataset, searching and scoring/ranking candidate peptides can be performed in parallel, independent of other spectra. This approach was followed for evaluating the CSPI framework.

A simple multiprocessing application design based on shared synchronized queues for inter-process communication is used. The flow diagram is shown in Figure 10. The main process reads in and preprocesses the spectra, queries the protein database stored as a pre-computed index on the hard disk (as described above) and places the retrieved candidates along with the preprocessed spectrum on a shared queue. From this queue, all the worker processes extract the objects, compute the CSPI scores, and store the results onto a shared output queue. Another child process extracts the results from this output queue and stores them in an output file.

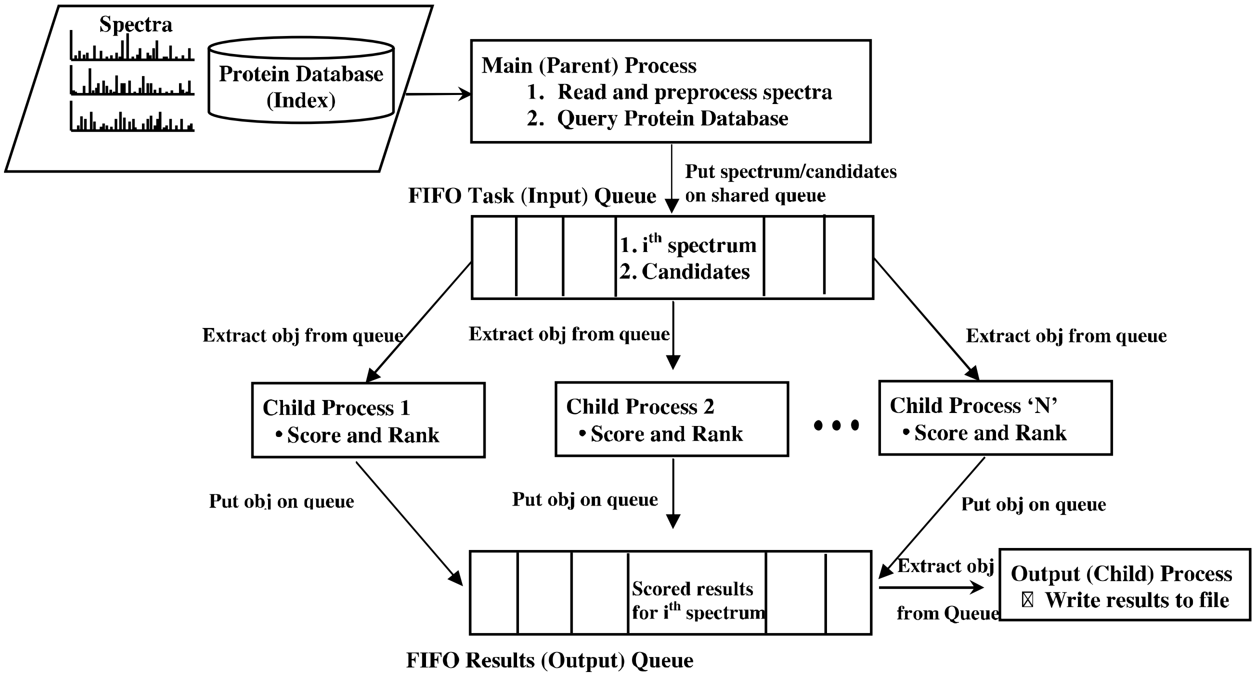


Figure 10. Workflow of the multiprocessing version of CSPI scoring framework

4.0 EXPERIMENTS AND EVALUATION METHODS

This chapter describes the experiments done to evaluate the CSPI framework beginning with description of datasets section 4.1, followed by the PSM properties and Database Search parameters used for all the analyses in section 4.2. Section 4.3 discusses the performance evaluation protocol.

4.1 DATASETS

In order to evaluate the performance of the CSPI framework, several MS/MS datasets of different sizes, complexity and nature were utilized, as briefly summarized in Table 3. The LTQ and LCQ instruments selectively isolate and detect precursor peptides as well as their corresponding fragments at low resolution and mass accuracy. Details of experimental protocol for each dataset can be found in the respective references. All samples were processed using Trypsin enzyme prior to separation via liquid chromatography and analysis using MS/MS.

Since CSPI models contain many tunable parameters, a large training dataset is required to avoid over-fitting. In the absence of such large, expert-validated 'gold-standard' identifications, a common strategy is to use a set of high-confidence identifications. Dataset 1 (SO-DR) contains high-scoring identifications made initially using the Sequest algorithm, and further validated via

Table 3. Characteristics of MS/MS datasets used for comparing algorithms

#	Name	Usage	Size	Instr	Validation	Source
1	SO-DR	Train	13, 249	LCQ + LTQ	FT-ICR	***S**hewanella **O**neidensis*, ***D**einococcus **R**adiodurans* (Huang et al. 2008) (Real world)
2	18Mix1_LCQ	Test	19, 822	LCQ	FDR	Standard 18 protein mix (Mix1) (Klimek et al. 2008) (Controlled)
3	18Mix1_LTQ	Test	53, 507	LTQ	FDR	Standard 18 protein mix (Mix1) (Klimek et al. 2008) (Controlled)
4	Yeast_LTQ	Test	34, 499	LTQ	FDR	Yeast whole cell lysate (Kall et al. 2007) (Real world)

accurate mass detection at the same retention time by FT-ICR and under identical chromatographic conditions. All identifications that could not be validated were removed from the dataset. As a consequence, a large fraction of these identifications are expected to be correct and hence form a good source for learning the parameters of the models. Roughly two-thirds of this data came from an LTQ and remainder from an LCQ instrument. Other possibilities for training datasets include using: a) validated PSMs from large spectral libraries of identifications, which typically contain, for each peptide (in the repository) a consensus spectrum obtained by some form of averaging over multiple copies; b) high-confidence assignments made on various large publicly available datasets**.** For the current thesis, all the CSPI models were trained using the SO-DR dataset.

While SO-DR consisted of only validated high-confidence PSMs, other datasets (18Mix1_LCQ, 18Mix1_LTQ, Yeast_LTQ) are large collections of MS/MS spectra and represent either a controlled or a real-world scenario where the goal is to assign peptides to the spectra and assess significance of the matches. These were also generated using low-resolution and low mass-accuracy LTQ instruments. All these additional datasets were used for testing the performance of the CSPI framework.

4.2 PSM PROPERTIES AND DATABASE SEARCH PARAMETERS

In this work, all analysis was restricted to a constrained but significant set of peptides. First, only tryptic peptides with both ends adhering to Trypsin cleavage specificity are used for all evaluations. Considering imperfect efficiency of Trypsin digestion, up to three internal Lys/Arg residues in peptides were allowed where trypsin misses to cleave. Second, only precursor charge state of +2 was modeled since these peptides fragment well while generating relatively less complex spectra than higher charge states. This class also constitutes the majority for Electospray Ionization, which is widely used for ionizing peptides. Finally, under low-energy CID peptides largely fragment at amide bonds along the peptide backbone, (most commonly) yielding singly charged N-terminal fragments (b-ions) and/or a singly charged C-terminal fragments (y-ions). Only these ions were modeled within CSPI.

Candidate sequences were searched using constraints as described above, with a fixed carbamidomethylation modification applied on Cysteine residues. A precursor tolerance of +/- 3.0 Da and fragment-ion tolerance of +/- 0.5 Da was used throughout. Since CSPI models are computationally expensive a simple filter was applied, which picks only top 500 unique candidates for each spectrum, based on their number of theoretical fragments observed in the experimental spectrum. Only these shortlisted peptides are scored using CSPI. All these search parameters were kept the same to the extent possible for all algorithms compared in this work.

4.3 PERFORMANCE EVALUATION AND SCORE COMBINATION

Peptide identification problem does not fit the traditional machine learning paradigm where the goal is classification of each sample into, say, a binary class, for which established methods of evaluation work well. Rather, each sample here (PSM) is represented with a bunch of scores or features and represents a mixture of correct and incorrect identifications. The goal then is to be able to differentiate, based on these features, between these identifications keeping the error (or false-discovery) rate within a user-defined level. As described earlier (section 2.5.1), a simple strategy based on target/decoy database search was used to address this hypothesis-testing problem. Briefly, the primary evaluation procedure is to control FDR at a user-specified value, which yields a score threshold and an estimate of the number of correct peptide identifications at that threshold. Whichever algorithm/score estimates higher number of correct peptide identifications at the same controlled FDR is reported superior.

The CSPI framework was compared with two widely used algorithms: Crux (version 1.33) (Park et al. 2008) which is a re-implementation of the original Sequest algorithm, and

X!Tandem (version CYCLONE 2010.12.01.1), which is another popular open-source peptide identification algorithm. Simple FDR and q-values were computed for Crux, X!Tandem and CSPI models using their primary search scores XCorr, Hyperscore and *CSPI_Score*^{b} (or *CSPI_Score*^{y} or the composite score *CSPI_Score*^{byAdded}), respectively.

Since SO-DR dataset was obtained from *Shewanella Onedensis* (SO) and *Deinococcus radiodurans* (DR), the target database used for these spectra was the concatenated protein FASTA sequences for SO and DR (~7000 proteins). Datasets 18Mix1_LCQ and 18Mix1_LTQ were obtained from controlled mixture of 18 proteins (see reference for details). Hence the target database for these spectra was the corresponding set of protein FASTA sequences appended with commonly observed contaminant proteins **(**http://www.thegpm.org/crap/index.html**)**. Likewise, for the Yeast_LTQ the target used was Yeast FASTA (~6,500 proteins) sequences, appended with common contaminants. Q-values for SO-DR were estimated using two different decoys: reversed SO/DR FASTA and a much larger reversed Human FASTA. Using a large decoy provides a more rigorous test of performance due to much larger number of candidates being evaluated for each spectrum. For all other test datasets reversed Human FASTA appended with corresponding reversed target database was used as the decoy.

Similar target-decoy strategy and FDR control was applied to the score-combination approach using LogitPercolator. After each iteration of LogitPercolator the primary composite score/feature used for computing FDR and q-values is the posterior probability of target PSMs from the Logistic Regression model.

5.0 EVALUATION OF THE CSPI FRAMEWORK

In this chapter, I present and discuss the results from evaluation of the CSPI framework using several MS/MS datasets of varying complexity. Section 5.1 and 5.2 present performance comparison of raw scores from CSPI (CSPI_Score^b and CSPI_Score^y and CSPI_Score^{byAdded}) with those of Crux (XCorr) and X!Tandem (Hyperscore), while Section 5.3 presents the results of score combination using the LogitPercolator procedure described in section 3.8. Section 5.5 discusses the efficiency aspects of CSPI implementation for handling large datasets.

5.1 CROSS-VALIDATION EXPERIMENT (SO-DR TRAINING DATASET)

Cross-validation is a commonly used re-sampling strategy, based on splitting the training dataset, to estimate the average performance of statistical models on unobserved data. Five-fold cross-validation (5-CFV) was performed on SO-DR dataset by splitting it into five equal parts. Of these four parts are used for training CSPI models while the remaining one-fifth samples are used for testing. The process is repeated five times so that each part becomes the test set once. X!Tandem and Crux do not involve any training, but each time they are evaluated on the same set of one-fifth samples as CSPI to facilitate performance comparison. Database search is performed on each of these test sets as described in section 4.2. For CSPI framework three scoring schemes were used for computing q-values: (i) *CSPI_Score*^{b}, (ii) *CSPI_Score*^{y}, and (iii)

$CSPI\_Score^{byAdded}$. For X!Tandem and Crux, their respective primary scores, Hyperscore and XCorr, were used for computing q-values.

Based on the size of the training dataset, each test set consisted of 2649 MS/MS spectra. After performing database search on these and controlling the FDR at q-value <= 1%, the percentage of the (assumed known) correct peptide identifications, retrieved correctly by each scoring feature was computed. Here, 'correctness' refers to the case that the peptide sequence identified is the same as the original high-confidence assignment provided in the dataset. For example, if in a test set 'n' is the actual number of correct identifications among all those 'estimated as correct' at 1% q-value, the reported performance is computed as n/2649. Table 4 reports this performance averaged over 10 test sets obtained by executing 5-CFV twice (2 x 5-CFV) on the SO-DR dataset.

It is observed that over both normalization schemes and decoys, byAdded models perform better than y-ion models, which in turn perform better than b-ion models ($p < 0.001$ from one-sided two-sample paired Wilcoxon signed-rank test). Also, within each group (b-, y- or byAdded models) window-norm scheme outperforms rank-norm ($p < 0.001$), providing a significant boost in the number of correct identifications.

Individually, b-ion models perform unfavorably as compared to both Crux and X!Tandem for both normalization schemes and decoys **(**$p < 0.001$**)**. On the other hand, y-ion models perform much more favorably **(**better than X!Tandem for window-norm/decoy-1/2, $p < 0.001$ **;** better than Crux for window-norm/decoy-2, $p < 0.001$ **;** worse than Crux for rank-norm/decoy-1, $p < 0.001$**;** worse than Crux for window-norm/decoy-1, $p < 0.05$ **;** no significant difference in remaining cases**)**. The composite byAdded models perform the best, particularly for window-norm scheme **(**better than both Crux and XTandem for window-norm/decoy-1/2, p <

0.001**;** better than Crux and X!Tandem for rank-norm/decoy-2 and rank-norm/decoy-1 respectively, $p < 0.005$**;** worse than Crux for rank-norm/decoy-1, $p < 0.01$; no significant difference from XTandem for rank-norm/decoy-2**)**. It is worth noting that the commercial version of Crux (i.e. SEQUEST) was used to originally identify the peptides in this dataset, and only validated high-confidence PSMs were retained. Despite this bias in favor of Crux, CSPI models show superior performance, particularly for the composite score and window-norm scheme.

Table 4. Cross-validation experiment on SO-DR dataset, reporting % of (assumed known) correct identifications, correctly retrieved by respective scoring feature; All values are averaged over 2-times 5-fold cross-validation (2649 test MS/MS spectra per fold; q-value = 0.01)

Algorithm	**Decoy 1 (Reversed SO-DR)**		**Decoy 2 (Reversed Human)**	
	Rank-norm	**Window-norm**	**Rank-norm**	**Window-norm**
CSPI_Score^{b}	26.8 %	39.4 %	17.7 %	26.6 %
CSPI_Score^{y}	71.0 %	75.4 %	62.8 %	66.5 %
CSPI_Score^{byAdded}	74.4 %	**81.6%**	64.7 %	**72.8 %**
Crux	77.2 %		62.0 %	
X!Tandem	71.7 %		63.2 %	

5.2 INDEPENDENT TEST DATASET VALIDATION

A more reliable evaluation is to train and test on completely different datasets.

Figure 11 reports the q-value plots for different algorithms compared when the CSPI models were trained on SO-DR dataset while tested on the 18Mix1_LCQ and 18Mix1_LTQ datasets.

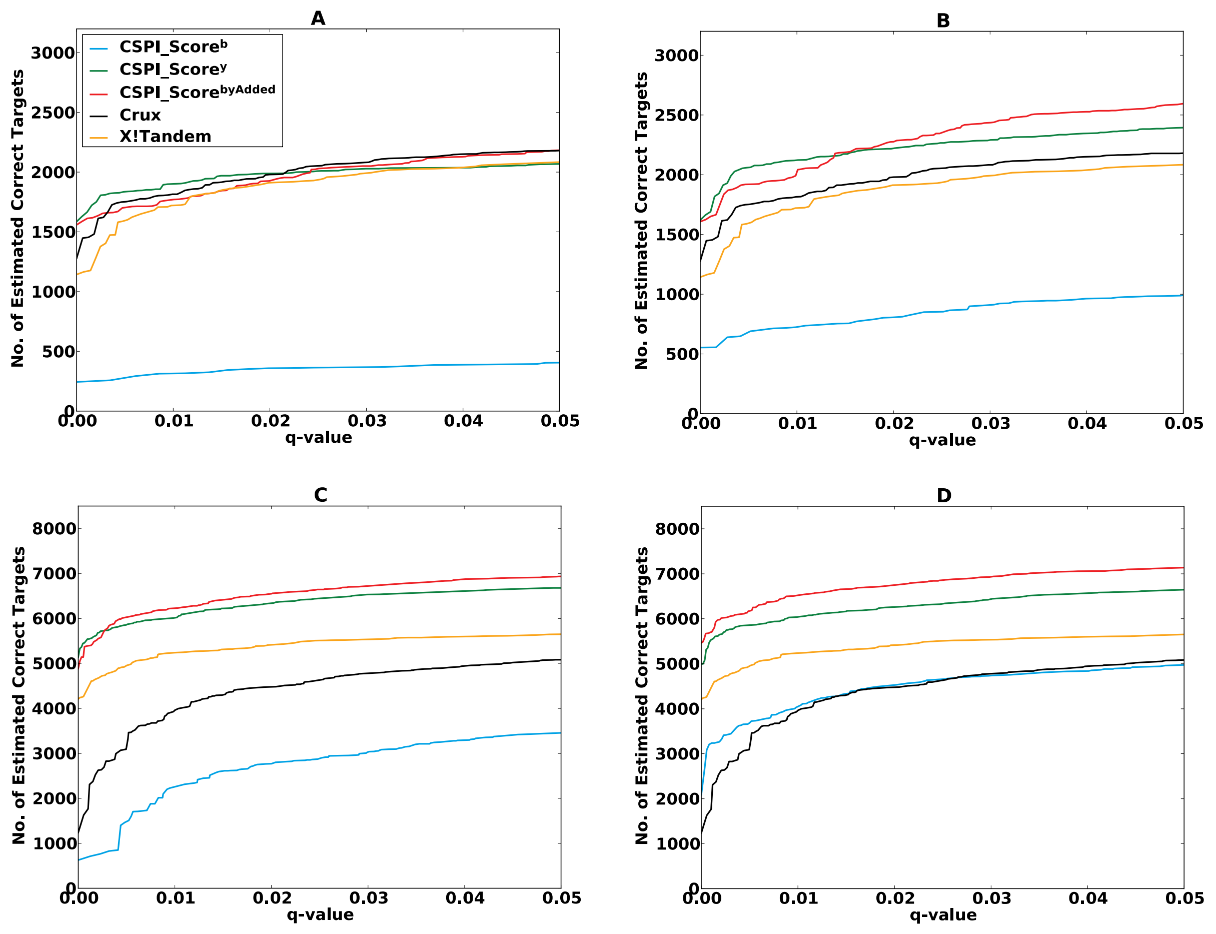
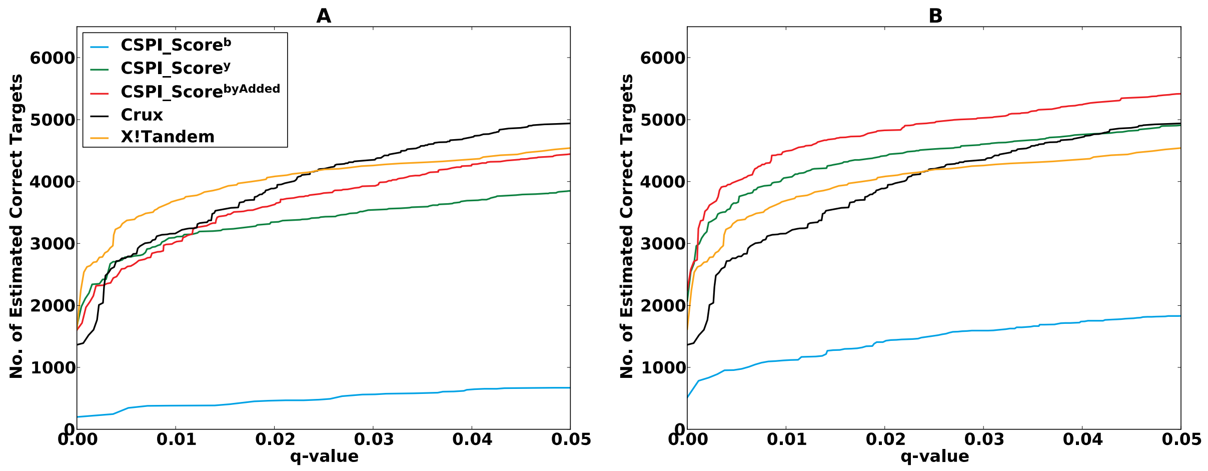


Figure 11. FDR curves; Train on SO-DR dataset, test on: A) 18Mix1_LCQ, Rank-normalization; B) 18Mix1_LCQ, Window-normalization; C) 18Mix1_LTQ, Rank-normalization; D) 18Mix1_LTQ, Window-normalization.

Here similar trends are observed in terms of relative performance of features based on IO-HMM models. Specifically, the composite score CSPI_Score^{byAdded} (red) outperforms the individual b- or y-ion models (blue and green, respectively) for both normalization schemes except for 18Mix1_LCQ dataset (upper panel) for which y-ion models perform better at lower q-values. Comparing rank-norm (left panel) with window-norm (right-panel) scheme, a significant performance improvement is seen in both b- and y-ion models, and therefore the composite byAdded score, except for y-ions (green) for 18Mix1_LTQ (lower panel) which perform comparably for both normalizations. It is noted that the contribution of b-ion models to the composite score appears to be limited and needs further investigation and fine-tuning. Both y-ion and byAdded models outperform Crux and X!Tandem by a wide margin over an acceptable range of q-values (< 0.05) for window-norm scheme on both datasets, and for rank-norm scheme



on 18Mix1_LTQ dataset. Specifically, at q-value = 0.01, CSPI models can achieve over ~25% improvement in the number of estimated correct peptide identifications.

Figure 12. FDR curves; Train on SO-DR dataset, test on: A) Yeast_LTQ, Rank-normalization; B) Yeast_LTQ, Window-normalization.

In order to test the generalization of results from controlled protein mixture to a real-world dataset, performance was evaluated on an additional dataset, Yeast_LTQ. This dataset was generated from yeast whole-cell lysate and consists of a set of ~ 35K spectra. CSPI models were trained on SO-DR dataset, and Figure 12 shows the corresponding q-value plots. Again, a significant performance boost is seen in the byAdded models using window-norm scheme, with contribution from improvement in both b- and y-ion models as compared with rank-norm scheme. Additionally, for the window-norm scheme y-ion and byAdded models significantly outperform both X!Tandem and Crux, with ~11% and 22% more estimated correct identifications (than X!Tandem, which does better than Crux in this case) at q-value = 0.01, respectively.

With a good choice of features representing the problem at hand, machine learning methods have the potential to learn complex patterns even with noisy data like that obtained from MS/MS experiments. Incorporating several peptide properties in our models, it has been shown how arbitrary features can be easily plugged into and tested with the CSPI framework. Although each feature was not evaluated individually, the prototype appears to model y-ion intensities well, providing good discrimination between correct and incorrect peptides. The b-ion models clearly need additional fine-tuning of input layer features, normalization or both.

Based on the experience in analyzing these datasets, one reason for inadequate performance of b-ion models is the nature of the datasets used. Specifically, for ion-trap data from trypsin-digested proteins, y-ions are preferably more abundant in number and intensity than b-ions, which, in many cases, are much harder to discriminate from random noise matches. For most correctly identified peptides, several fragments from at least one ion-series (b- or y-) are observed. Since this information is lost when each ion-series is modeled separately, it might be beneficial to build joint models from b- and y-ion series.

The above results also suggest that local normalization schemes may be superior to global approaches, possibly due the fact that different regions of the m/z range of MS/MS spectra show wide variability in both the density as well as intensity of peaks. This is well established in the literature, specifically for ion-trap data, where more and abundant peaks are generally observed from the middle of the peptides. Although the window-norm procedure is conceptually reasonable and achieves good performance, the existence of several other pre-processing methods in the literature is acknowledged, for example (Ning et al. 2007; Renard et al. 2009), that could be worth investigating within the CSPI framework.

5.3 SCORE COMBINATION

As described earlier, multiple features, either from the same or different search algorithms, can be combined to achieve greater performance. This experiment evaluates the benefit achieved by adding CSPI's intensity-based scores on top of other features/scores. Top-ranking PSMs (both targets and decoys) were first extracted from Crux results' files using in-house python scripts, after which CSPI models trained on SO-DR dataset were applied to them. For this experiment, random decoy peptide sequences generated by Crux were used instead of those from reversed human FASTA. Different sets of features were combined using LogitPercolator and also compared with original Percolator applied on Crux results. Since, from previous results, it is clear that window-norm is superior to rank-norm scheme, this section evaluates LogitPercolator only on window-norm scheme.

Figure 13 shows the q-value plots for various combinations of features, from 18Mix1_LCQ, 18Mix1_LTQ and Yeast_LTQ datasets. As expected, a dramatic increase in performance is observed as compared with results in the previous section where a single feature was used (up to ~63 % extra estimated correct identifications at q-value=0.01 than the best performing individual feature). This corroborates earlier findings on the utility of post-processing and score combination approaches (Keller et al. 2002; Higdon et al. 2004; Kall et al. 2007).

Without CSPI scores, comparable performance was achieved for 18Mix1_LTQ and Yeast_LTQ datasets by the baseline LogitPercolator(Crux+) compared to that of the original Percolator, which provides confidence in the comparison and interpretation. Further addition of CSPI's intensity-based features provides up to ~4-8% additional estimated number of correct identifications than without them, at q-value=0.01. However it is noted that 'delta_CSPI' scores (i.e. difference in primary CSPI scores between top-ranking and the next best candidate) in LogitPercolator(Crux+,IOHMM+) do not always provide significant additional benefit and require further experimentation.

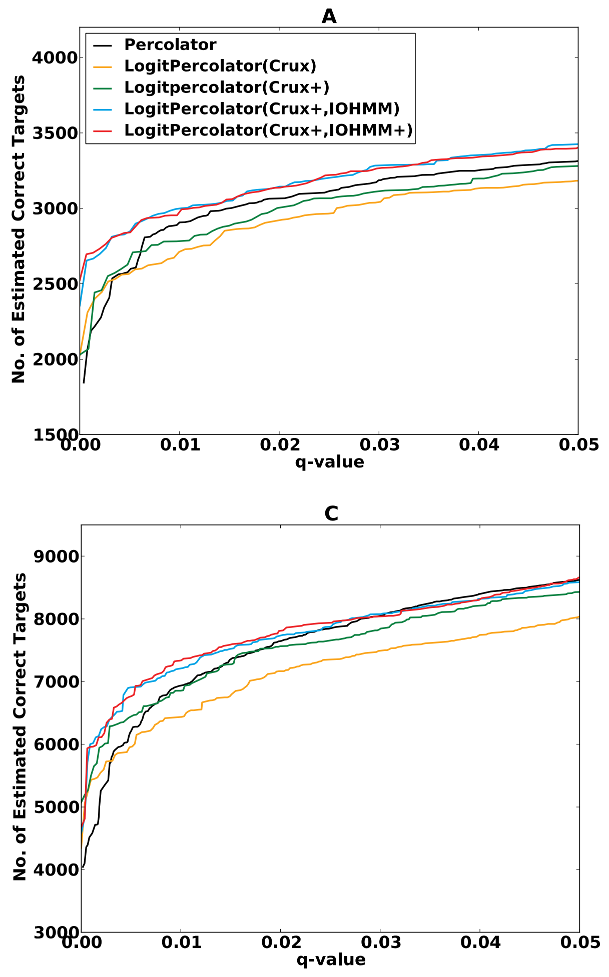


Figure 13. FDR curves; Train on SO-DR dataset, apply on top-ranking targets/decoys from Crux; LogitPercolator: Implementation of Percolator developed in this thesis using Logistic Regression Classifier; Percolator: Original Percolator; Crux: features from Crux {XCorr, deltaCn, SpScore}; Crux+: features {Crux, fracMatch (fraction of peptide fragments observed), fracExp (fraction of explained spectrum intensity)}; IOHMM: features {Crux+, CSPI_Score^b, CSPI_Score^y}; IOHMM+: features{IOHMM, delta_CSPI_Score^b, delta_CSPI_Score^y}, where delta is the difference between scores from top-ranking and the next best peptide (from original crux ranking); A) 18Mix1_LCQ, Window-normalization; B) 18Mix1_LTQ, Window-normalization; C) Yeast_LTQ, Window-normalization.

5.4 DATABASE SEARCH LOGISTICS

As discussed earlier, a key feature of shotgun proteomic data is their high-throughput aspect. Effective utilization of these complex datasets requires intricate algorithms with good performance characteristics, but that typically require significant computation time, the CSPI framework being a case in point. As seen above, CSPI can confidently identify more spectra at a controlled FDR as compared with popular state-of-the-art methods. However, it takes ~5-8 seconds for evaluating a spectrum (against the human protein database), and under constrained searches (as described in section 4.2), which is at least 2 orders of magnitude more than the closest competitor (Crux). Keeping pace with volume and rate of data generation will become even more challenging when search constraints are removed or reduced, as will be necessary for more thorough analysis.

5.4.1 Indexing Challenge

One commonly used strategy, also used in CSPI, for faster database search is indexing the FASTA database file. The approach works well for constrained database searches (total of ~10 million peptides in the index, and ~10-20K candidates per spectrum) employed in the current implementation and analysis in this thesis, and took (on average) less than a second to retrieve candidates per query. However, unconstrained searches can yield a total space of several billion peptides, leading to larger index files and increased index generation as well as querying time. A **potential scalable solution** is a distributed index with capability for parallel generation and querying (using simple synchronization primitives) which is facilitated by splitting the index into multiple files by mass region (as described earlier) as well as the fact that each spectrum can be queried independently of others. Such schemes or variants thereof will be crucial for future large-scale proteomics and must be explored.

5.4.2 Parallelization Challenge

Although, database search and candidate evaluation time depend upon the size of the MS/MS datasets as well as the number of candidates evaluated per spectrum (which in turn depends upon the search constraints applied), each spectrum can be evaluated independently of others. The CSPI framework takes advantage of this characteristic to parallelize the computational workflow using multiprocessing architectures. Figure 14 shows how CSPI scales with addition of processor units. Specifically, the constrained searches performed resulted in between 10K and 20K candidates to be evaluated per spectrum.

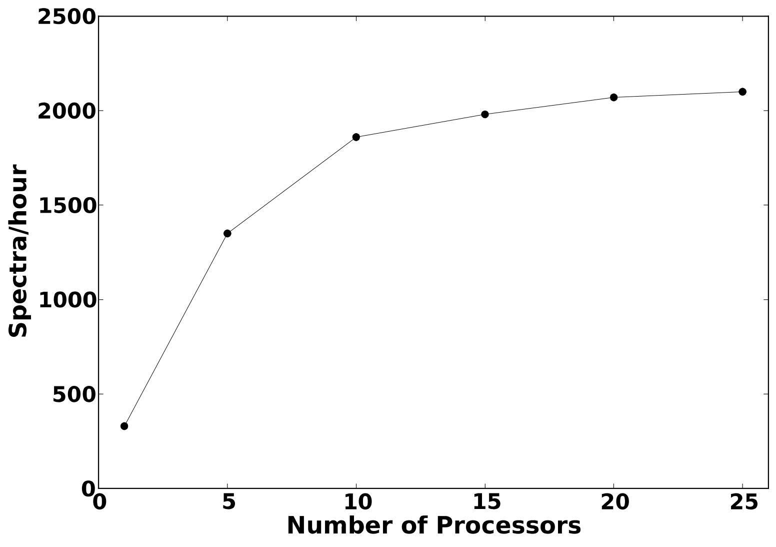


Figure 14. Scalability of the multiprocessing version of CSPI scoring algorithm

It is seen that the throughput increases rapidly initially, although not linearly, but saturates at about 15-20 processors. Although simpler scoring systems can achieve much higher performance gains through parallelization (Xu et al. 2009), the gap can be possibly reduced with alternate schemes for task-distribution.

As described above, the current workflow breaks the tasks at the individual spectrum level, which means once a spectrum and its potential candidates are assigned to a child process, they are evaluated sequentially within the same process. However, since evaluation of each candidate against a spectrum itself requires several steps and can be performed independently of all other candidates for all other spectra, there is scope for much further optimization. It is important to note that although the entire process of peptide identification is inherently parallelizable, optimum task distribution and sharing between processes will need careful profiling of processing needs of individual steps and will also depend critically upon such factors as the size of the database searched as well as search constraints applied. Further, with greater granularity of tasks and number of processes, overhead due to inter-process communication will become an important factor to consider (Xu et al. 2009). Automatically adjusting for all these dependencies within resource constraints is a non-trivial but interesting problem to investigate.

6.0 CONCLUSIONS AND FUTURE WORK

Scoring and confident identification of peptides and proteins lies at the heart of current mass-spectrometry-based proteomics. The primary hypothesis of this dissertation was that CSPI framework is effective for peptide scoring and identification from tandem mass spectrometry. In order to test the hypothesis, CSPI was developed and empirically evaluated on several datasets of different complexity.

(Claim 1) Increased peptide identification performance was demonstrated, in terms of number of correct identifications at a fixed (user-defined) FDR as compared with popular state-of-the-art algorithms (see Sections 5.1 and 5.2). The framework is highly flexible and scalable, and can exploit different feature types and representations, as well as choice of component functions, in order to learn and represent complex probability distributions.

(Claim 2) Variable performance characteristics were observed for the two different fragment ion-types modeled in CSPI (see Sections 5.1 and 5.2). Particularly, y-ion models showed much superior performance than their complementary b-ion models. As was pointed out earlier, one reason for this discrepancy is the nature of data from ion-trap mass-spectrometers, which strongly favor y-ions. Nevertheless, the b-ions do contribute some additional information as was seen in superior performance of the simple composite score ($CSPI\_Score^{byAdded}$).

Utility of CSPI's intensity based features was further evidenced by better performance in a state-of-the-art score-combination procedure. (Claim 3) It was demonstrated that addition of

CSPI scores to other complementary scores and features leads to better discrimination between true and false peptide identifications, thus leading to greater number of correctly identified peptides at a fixed (low) FDR (see section 5.3). This approach is also a much superior composite scoring scheme it appropriately weights the different features based on how much predictive they are of the class label.

Since for most identifiable peptides, several fragment ions are observed from at least one of the two ion-series, a possible direction for future work is to construct an additional feature from jointly modeling the two observations at a specific fragmentation site. Since these are complementary fragments, this will allow modeling the dependencies between their intensity distributions. A couple of ideas of simple dependency models are shown in Figure 15.

A further, more challenging extension can include several other fragment-ion types (like those carrying higher charge states, and neutral losses) in the output layer and learning their complex dependencies. Modeling fragmentation is fundamental to the shotgun proteomics approach. This general methodology can be adapted for modeling, either individually or together, ion intensities from other fragmentation modalities than CID, like ETD or ECD, all of which have their unique advantages and are sometimes generated as complementary sources of information.

For the models developed in this thesis, the same set of features was used in the input layer. It would be worth investigating if different sets of features are relevant for each kind of observation sequence being modeled. Additionally, the models developed comprised only one instance of a large class where the input features influenced the transition functions alone. A more general model can also include their effect on the output distribution function, as well as allow alterations in (currently fixed) model topology and component distributions. Training in such rich and expressive model space will require clever new search strategies as well as significant domain knowledge, and may become possible in the future as peptide fragmentation behavior is understood at a finer level.

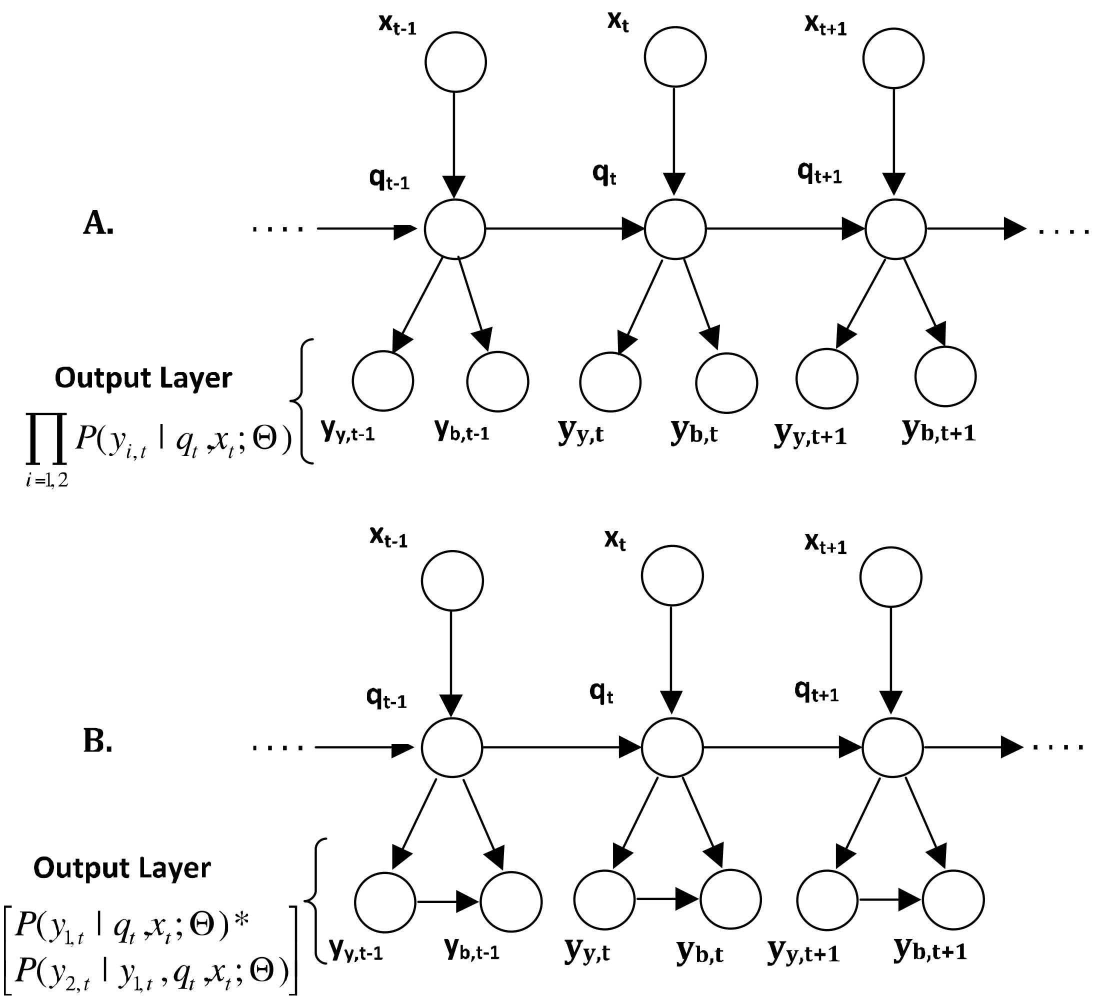


Figure 15. Extensions to CSPI model structure used in this thesis; A) Joint b/y-ion models, with $y_{b,t}$ and $y_{y,t}$ representing observed b- and y-ion intensities. Conditioned on hidden state q_t, $y_{b,t}$ and $y_{y,t}$ are independent; B) Joint b/y-ion models, with $y_{b,t}$ and $y_{y,t}$ representing observed b-ion and y-ion intensities, respectively. Here the b-ion intensity depends on both the hidden state and the observed y-ion intensity

The findings from experiments show the importance of appropriate normalization protocol for effective modeling. Local approaches (for ex. the window-norm procedure) seem to perform better. One possible argument in support of this observation is that noise level varies from one region to another on the m/z (x)-axis. This information is lost in the global rank-based approaches, which have been more widely researched and used to date. It is conceivable that these two approaches provide complementary pieces of information and that a hybrid strategy might be superior to each alone. A related line of research that by far remains unexplored in the current domain is the explicit modeling of variability in signal as well as noise intensities in data replicated across different laboratories. Although the current models were trained on PSM pairs with unique peptide sequences, it is possible to obtain multiple spectra per peptide and to account for the variability in peak intensities within the training phase. It would be useful to quantify the effect similarity (or differences) in MS/MS spectra on the score assigned to a PSM.

Confirmation of the robustness and utility of the CSPI's intensity-based modeling approach was further demonstrated in conjunction with a state-of-the-art score combination procedure, LogitPercolator, which appropriately weights and combines several features of match quality to boost performance. LogitPercolator provides a dramatic improvement over the simple composite score and was shown to significantly enhance performance with the addition of CSPI's intensity-based features to other features. An immediate extension to this baseline version would be to allow cost-sensitive learning as was done in the original Percolator algorithm, perhaps after factoring in features like the spectrum quality and signal-to-noise level. It is quite easy to add other features into the algorithm. These features generally exhibit strong correlations which are currently not exploited and can potentially improve performance if modeled appropriately. The PSM scores and other features obtained depend upon how well different peptides fragment. Although this idea was exploited in the CSPI framework, conditioning on physicochemical context at the score-combination stage might improve performance further and also has the potential to elucidate dependencies and biases in individual features in relation to the peptide sequences being evaluated.

Finally, much of peptide-centric analyses are utilized in further downstream proteomic investigations like protein identification, quantification and differential expression. It would be interesting to compare the effects of differences in peptide identification performance in each of these analysis stages. However, in order for such comparisons to be complete and practically useful, the CSPI framework must first be extended to handle other data characteristics like higher precursor peptide charge-states, post-translational modifications and digestion enzymes, all of which were excluded from the current research.

APPENDIX A

CSPI TRAINING: EM ALGORITHM

Let the nth observation sequence pair be represented by $d_n = (X_n, Y_n) = (<x_{n,1}, x_{n,2}, \ldots, x_{n,Tn}>, <y_{n,1}, y_{n,2}, \ldots, y_{n,Tn}>)$, and the corresponding hidden state-sequence by $q_n=<q_{n,1}, q_{n,2}, \ldots, q_{n,Tn}>$. Then the conditional distribution $P(Y_n, q_n \mid X_n, \Theta)$, where Θ are the model parameters, is given by:

$$\begin{aligned}
P(Y_n, q_n \mid X_n, \Theta) &= P(Y_n \mid q_n, X_n, \Theta) * P(q_n \mid X_n, \Theta) \\
&= P(y_{n,1} \mid q_{n,1}) * P(y_{n,2} \mid q_{n,2}) * \ldots * P(y_{n,T_n} \mid q_{n,T_n}) \\
&\quad * P(q_{n,1} \mid X_{n,1}) * P(q_{n,2} \mid q_{n,1}, X_{n,2}) * \ldots * P(q_{n,T_n} \mid q_{n,T_n-1}, X_{n,T_n}) \\
&\quad \text{(using 1}^{\text{st}}\text{ order Markov assumption and independence properties of the} \\
&\quad \text{graphical structure in Fig 7)} \\
&= P(q_{n,1} \mid X_{n,1}) * P(y_{n,1} \mid q_{n,1}) * \prod_{t_n=2}^{T_n} P(q_{n,t_n} \mid q_{n,t_n-1}, X_{n,t_n}) * P(y_{n,t_n} \mid q_{n,t_n}) \qquad \text{(A.1)} \\
&\quad (\Theta \text{ is suppressed for notational convenience})
\end{aligned}$$

Then, for a dataset D of N independently and identically distributed (iid) sequences, the joint distribution is given by:

$$\begin{aligned}
P(D, q \mid \lambda) &= \prod_{n=1}^{N} P(Y_n, q_n \mid X_n, \Theta) \\
&= \prod_{n=1}^{N} P(q_{n,1} \mid X_{n,1}) P(y_{n,1} \mid q_{n,1}) \prod_{t_n=2}^{T_n} P(q_{n,t_n} \mid q_{n,t_{n-1}}, X_{n,t_n}) * (y_{n,t_n} \mid q_{n,t_n}) \qquad \text{(A.2)}
\end{aligned}$$

When considered a function of the parameters Θ, (A.2) is also known as the complete data likelihood (CDL). Taking natural log, we get the log(CDL) as:

$$\begin{aligned}
\log CDL &= \log \prod_{n=1}^{N} \{P(q_{n,1} \mid X_{n,1}) P(y_{n,1} \mid q_{n,1}) * \prod_{t_n=2}^{T_n} P(q_{n,t_n} \mid q_{n,t_{n-1}}, X_{n,t_n}) * P(y_{n,t_n} \mid q_{n,t_n})\} \\
&= \sum_{n=1}^{\mathrm{N}} \log\{P(q_{n,1} \mid X_{n,1}) P(y_{n,1} \mid q_{n,1}) \prod_{t_n=2}^{T_n} P(q_{n,t_n} \mid q_{n,t_{n-1}}, X_{n,t_n}) * P(y_{n,t_n} \mid q_{n,t_n})\} \\
&= \sum_{n=1}^{\mathrm{N}} \log P(q_{n,1} \mid X_{n,1}) + \log P(y_{n,1} \mid q_{n,1}) + \sum_{t_{n=2}}^{\mathrm{T_n}} \log P(q_{n,t_n} \mid q_{n,t_{n-1}}, X_{n,t_n}) + \sum_{t_n=2}^{T_n} P(y_{n,t_n} \mid q_{n,t_n}) \\
&= \underset{\substack{\uparrow \\ \mathrm{A}}}{\sum_{n=1}^{\mathrm{N}} \log P(q_{n,1} \mid X_{n,1})} + \underset{\substack{\uparrow \\ \mathrm{B}}}{\sum_{n=1}^{\mathrm{N}} \sum_{t_n=1}^{\mathrm{T_n}} \log P(y_{n,t_n} \mid q_{n,t_n})} + \underset{\substack{\uparrow \\ \mathrm{C}}}{\sum_{n=1}^{\mathrm{N}} \sum_{t_n=2}^{\mathrm{T_n}} \log P(q_{n,t_n} \mid q_{n,t_{n-1}}, X_{n,t_n})} \qquad \text{(A.3)}
\end{aligned}$$

Expectation of the log(CDL) is computed w.r.t. the distribution P(***q*** | *D*, ***Θ****^k^*), where ***Θ****^k^* are the parameters in the previous iteration, and can be computed independently for each of the terms A, B and C in (A.3). Here '**q**' is hidden state transition sequence for all the samples in the dataset ***D.***

$$\begin{aligned}
A &= \sum_{n=1}^{N} \log P(q_{n,1} \mid X_{n,1}, \Theta) \\
E_{[q|D,\Theta^{'}]}[A] &= \sum_{q} \left(\sum_{n=1}^{N} \log P(q_{n,1} \mid X_{n,1}, \Theta) \right) * P(q \mid D, \Theta^k) \\
&= \sum_{n=1}^{N} \sum_{q} \left(\log P(q_{n,1} \mid X_{n,1}, \Theta) \right) * P(q \mid D, \Theta^k) \\
&= \sum_{n=1}^{N} \sum_{q_n} \sum_{r=q \backslash q_n} \log P(q_{n,1} \mid X_{n,1}, \Theta) * P(q_n \mid D, \Theta^k) * P(r \mid D, \Theta^k) \\
&\quad (\text{since } \mathrm{q_i} \text{ is independent of } \mathrm{q_j} \text{ for } \mathrm{i} \neq \mathrm{j}) \\
&= \sum_{n=1}^{N} \sum_{q_n} \log P(q_{n,1} \mid X_{n,1}, \Theta) * P(q_n \mid D, \Theta^k) * \sum_{r} P(r \mid D, \Theta^k) \\
&= \sum_{n=1}^{N} \sum_{q_n} \log P(q_{n,1} \mid X_{n,1}, \Theta) * P(q_n \mid D, \Theta^k) \qquad (\because \sum_{r} P(r \mid D, \Theta^k) = 1) \\
&= \sum_{n=1}^{N} \sum_{q_n} \log P(q_{n,1} \mid X_{n,1}, \Theta) * P(q_{n,1} \mid D, \Theta^k) * P(q_{n,2 \ldots T_n} \mid D, \Theta^k) \\
&= \sum_{n=1}^{N} \sum_{q_{n,1}} \log P(q_{n,1} \mid X_{n,1}, \Theta) * P(q_{n,1} \mid D, \Theta^k) * \sum_{m=2}^{T_n} P(q_{n,m} \mid q_{n,1}, D, \Theta^k) \\
&= \sum_{n=1}^{N} \sum_{q_{n,1}} \underset{\substack{\nearrow \\ \textbf{Function of Parameters}}}{\log P(q_{n,1} \mid X_{n,1}, \Theta)} * \underset{\substack{\nwarrow \\ g^{'}_{n,1} \textbf{, as in Eq}^{\mathbf{n}} \textbf{ (15) in Chapter 2}}}{P(q_{n,1} \mid D, \Theta^k)} \qquad \text{(A.4)}
\end{aligned}$$

Similarly we can compute the expectation of the terms ‘B’ and ‘C’ in (A.3)

$$B=\sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)$$

$$\begin{aligned}
E_{[q|D,\Theta^{'}]}[B] &= \sum_{q}\left(\sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta\right)*P(q\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{q}\left(\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)\right)*P(q\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{q_n}\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)*P(q_n\,|\,D,\Theta^{k})*\sum_{r=q\backslash q_n}P(r\,|\,q_n,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{q_n}\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)*P(q_n\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{q_{n,t_n}}\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)*P(q_{n,t_n}\,|\,D,\Theta^{k})*\sum_{r_n=q_n\backslash q_{n,t_n}}P(r_n\,|\,q_{n,t_n},\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{q_{n,t_n}}\underbrace{\log P(y_{n,t_n}\,|\,q_{n,t_n},\Theta)}_{\textbf{Function of Parameters}}*\underbrace{P(q_{n,t_n}\,|\,D,\Theta^{k})}_{g^{'}_{n,t_n}\textbf{, as in Eq}^{\textbf{n}}\textbf{ (15) in Chapter 2}} \qquad \text{(A.5)}
\end{aligned}$$

$$C=\sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)$$

$$\begin{aligned}
E_{[q|D,\Theta^{'}]}[C] &= \sum_{q}\left(\sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)\right)*P(q\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)*P(q\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q_n}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)*P(q_n\,|\,D,\Theta^{k})*\sum_{r=q\backslash q_n}P(r\,|\,q_n,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q_n}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)*P(q_n\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q_{n,t_n},q_{n,t_n-1}}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)*P(q_{n,t_n},q_{n,t_{n-1}}\,|\,D,\Theta^{k})*\sum_{r_n=q_n\backslash\{(q_{n,t_n}),(q_{n,t_{n-1}})\}}P(r_n\,|\,q_{n,t_n},q_{n,t_{n-1}},D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q_{n,t_n},q_{n,t_n-1}}\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)*P(q_{n,t_n},q_{n,t_{n-1}}\,|\,D,\Theta^{k})\\
&= \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{q_{n,t_n}}\sum_{q_{n,t_n-1}}\underbrace{\log P(q_{n,t_n}\,|\,q_{n,t_{n-1}},X_{n,t_n},\Theta)}_{\textbf{Function of Parameters}}*\underbrace{P(q_{n,t_n},q_{n,t_{n-1}}\,|\,D,\Theta^{k})}_{h^{'}_{n,t_n}\textbf{, as in Eq}^{\textbf{n}}\textbf{ (15) in Chapter 2}} \qquad \text{(A.6)}
\end{aligned}$$

This completes the E-step, and the Q-function is given by:

$$\begin{aligned} Q(\Theta,\Theta^k) &= E[A]+E[B]+E[C] \\ &= Q_A + Q_B + Q_C \qquad \text{(A.7)} \end{aligned}$$

As described earlier, in the M-step only an increase in the Q-function is required. This can be achieved by a conjugate gradient method, and requires computing the partial derivatives of the Q-function w.r.t. each of the parameters in the model. These are described next for the functional forms used in CSPI framework (Logistic functions for initial state and transition probabilities, and Gaussian/Exponential/Beta distributions for the emission probabilities).

$$Q_A = \sum_{n=1}^{N}\sum_{i=1}^{S} \log P(q_{n,1} = i \mid X_{n,1},\Theta) * g^{'}_{n,i,1} \qquad \text{(Here } \Theta \text{ are the parameters for the initial - state Logistic Functions)}$$

For the nth sample,

$$\begin{aligned} Q_{A,n} &= \sum_{i=1}^{S} \log P(q_{n,1} = i \mid X_{n,1},\Theta) * g^{'}_{n,i,1} \\ &= g^{'}_{n,1,1} * \log\left(\frac{1}{1+\sum_{i=2}^{S}\exp(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v})} \right) + \sum_{j=2}^{S} \log\left(\frac{\exp\sum_{v=1}^{V}\beta_{j,v}x_{n,1,v}}{1+\sum_{i=2}^{S}\exp(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v})} \right) * g^{'}_{n,j,1} \qquad \text{(A.8)} \end{aligned}$$

where 'β' are the logistic function weight parameters, and 'x' are input layer features

Taking partial derivatives w.r.t. a specific β:

$$\frac{\partial Q_{A,n}}{\partial \beta_{l,v'}} = \underset{\overset{\uparrow}{\mathbf{A}}}{g^{'}_{n,1,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{1}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\}} + \underset{\overset{\uparrow}{\mathbf{B}}}{g^{'}_{n,l,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\}} + \underset{\overset{\uparrow}{\mathbf{C}}}{\sum_{\substack{j=2 \\ j\neq l}}^{S} g^{'}_{n,j,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{j,v}x_{n,1,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\}}$$

where $l = 2,3,\ldots,S \;\; v^{'} = 1,2,\ldots,V$ -------- (A.9)

$$\mathrm{A} = g'_{\mathrm{n},1,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{1}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\} = g'_{n,1,1} * \frac{(-1)*\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)*x_{n,1,v'}}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)}$$

$$= -x_{n,v'} * g'_{n,1,1} * P(q_{n,1} = l \mid X_{n,1},\Theta) \qquad \text{-----------(A.10)}$$

$$\mathrm{B} = g'_{n,l,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\}$$

$$= g'_{n,l,1} * \frac{1}{\left(\frac{\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)\right\}} \right)}$$

$$* \frac{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)\right\}*\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)*x_{n,1,v'} - \exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)*\exp\left(\sum\beta_{l,v}x_{n,1,v}\right)*x_{n,1,v'}}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)\right\}^2}$$

$$= \mathrm{x}_{\mathrm{n},1,\mathrm{v}'} * g'_{n,l,1} * \{1 - p(q_{n,1} = l \mid X_{n,1},\Theta)\} \qquad \text{------------(A.11)}$$

$$\mathrm{C} = \sum_{\substack{j=2\\ j\neq l}}^{S} g'_{n,j,1} * \frac{\partial}{\partial \beta_{l,v'}} \log\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{j,v}x_{n,1,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)} \right\}$$

$$= \sum_{\substack{j=2\\ j\neq l}}^{S} g'_{n,j,1} * \frac{1}{\left(\frac{\exp\left(\sum_{v=1}^{V}\beta_{j,v}x_{n,1,v}\right)}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)\right\}} \right)} * \frac{\exp\left(\sum_{v=1}^{V}\beta_{j,v}x_{n,1,v}\right)*\left(-\exp\left(\sum_{v=1}^{V}\beta_{l,v}x_{n,1,v}\right)\right)*x_{n,1,v'}}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{i,v}x_{n,1,v}\right)\right\}^2}$$

$$= \sum_{\substack{j=2\\ j\neq l}}^{S} - g'_{n,j,1} * x_{n,v'} * P(q_{n,1} = l \mid X_{n,1},\Theta) \qquad \text{----------(A.12)}$$

$$\frac{\partial Q_{A,n}}{\partial \beta_{l,v'}} = A+B+C$$

$$= \left[-x_{n,1,v'} * g'_{n,1,1} * P(q_{n,1}=l \mid X_{n,1},\Theta)\right] + \left[x_{n,1,v'} * g'_{n,l,1} * \{1 - P(q_{n,1}=l \mid X_{n,1},\Theta)\}\right]$$

$$+ \left[\sum_{\substack{j=2 \\ j\neq l}}^{S} - g'_{n,j,1} * x_{n,1,v'} * P(q_{n,1}=l \mid X_{n,1},\Theta)\right]$$

$$= -x_{n,1,v'} * \left\{P(q_{n,1}=l \mid X_{n,1},\Theta) - g'_{n,l,1}\right\} \quad \text{---------(A.13)}$$

Hence, considering the entire dataset,

$$\frac{\partial Q_A}{\partial \beta_{l,v'}} = \sum_{n=1}^{N} - x_{n,1,v'} * \left\{P(q_{n,1}=l \mid X_{n,1},\Theta) - g'_{n,l,1}\right\} \quad \text{--------(A.14)}$$

The term Q_B in the Q-function of E-step is a function of the parameters of the emission distributions. For the purpose of demonstration, the following derivations correspond to Gaussian Emission distributions, and can be easily extended for other emission distributions. Let Z ~ N(μ, σ^2) be a normally distributed variable. The following results are used for performing the M-step:

$$w = \log(f(Z=z;\mu,\sigma)) = \log\left(\frac{1}{\sqrt{2\Pi\sigma^2}} e^{\frac{-1}{2}*\left(\frac{z-\mu}{\sigma}\right)^2}\right)$$

$$= \frac{-1}{2}\log(2\Pi\sigma^2) - \frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2$$

$$= -\log(\sigma) - \frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2 + const$$

$$\frac{\partial w}{\partial \mu} = \left(\frac{z-\mu}{\sigma^2}\right) \quad \text{--------(A.15)}$$

$$\frac{\partial w}{\partial \sigma} = \frac{-1}{\sigma} + \frac{(z-\mu)^2}{\sigma^3} \quad \text{--------(A.16)}$$

Now,

$$Q_B = \sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\sum_{i=1}^{S}\log[P(y_{n,t_n} \mid q_{n,t_n}=i,\Theta)] * g'_{n,i,t_n} \qquad \text{(Here } \Theta \text{ are the parameters for the emission distributions)}$$

Taking partial derivatives w.r.t. the parameters of the Gaussian emission distribution, and using results from (A.15) and (A.16) above, we get,

$$\frac{\partial Q_B}{\partial \mu_s} = \sum_{n=1}^{N}\sum_{t_n=1}^{T_n} g'_{n,s,t_n} * \left(\frac{y_{n,t_n} - \mu_s}{\sigma_s^2}\right)$$

For the case of Gaussian distribution, we can maximize the parameters (M-step) by equating the partial derivative to zero, which gives the following MLE for the mean of the distribution:

$$\overset{\Lambda}{\mu}_{s_{MLE}} = \frac{\sum_{n=1}^{N}\sum_{t_n=1}^{T_n}\left(g'_{n,s,t_n}\right) * y_{n,t_n}}{\sum_{n=1}^{N}\sum_{t_n=1}^{T_n} g'_{n,s,t_n}} \qquad \text{---------(A.17)}$$

Similarly, taking the partial derivative w.r.t. the variance parameter and equating to zero, we get the MLE for the variance of the distribution:

$$\overset{\Lambda}{\sigma}_{s_{MLE}}{}^2 = \frac{\sum_{n=1}^{N}\sum_{t_n=1}^{T_n} g'_{n,s,t_n} * (y_{n,t_n} - \overset{\Lambda}{\mu}_{s_{MLE}})^2}{\sum_{n=1}^{N}\sum_{t_n=1}^{T_n} g'_{n,s,t_n}} \qquad \text{-----------(A.18)}$$

The term Q_C in the Q-function of the E-step is a function of the parameters of the transition probability functions.

$$Q_C = \sum_{n=1}^{N}\sum_{t_n=2}^{T_n}\sum_{p=1}^{S}\sum_{q=1}^{S}\log P(q_{n,t_n} = q \mid q_{n,t_{n-1}} = p, X_{n,t_n}, \Theta) * h'_{n,q,p,t_n}$$

For the nth sample and t_nth position,

$$Q_{C,n,t_n} = \sum_{p=1}^{S}\sum_{q=1}^{S}\log P(q_{n,t_n} = q \mid q_{n,t_{n-1}} = p, X_{n,t_n}, \Theta) * h'_{n,q,p,t_n}$$

$$= \sum_{p=1}^{S} h'_{n,1,p,t_n} * \log\left\{\frac{1}{1+\sum_{i=2}^{S}\exp\sum_{v=1}^{V}\left(\beta_{p,i,v}x_{n,t_n,v}\right)}\right\} + \sum_{q=2}^{S} h'_{n,q,p,t_n} * \log\left\{\frac{\exp\sum_{v=1}^{V}\left(\beta_{p,q,v}x_{n,t_n,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\left(\beta_{p,i,v}x_{n,t_n,v}\right)\right)}\right\}$$

-----------(A.19)

Taking partial derivatives w.r.t. a specific weight parameter β of a logistic function, we get:

$$\frac{\partial Q_{C,n,t_n}}{\partial \beta_{k,l,v'}} = h'_{n,1,k,t_n} * \frac{\partial}{\partial \beta_{k,l,v'}}, \log\left\{\frac{1}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)}\right\} \longleftarrow \quad \textbf{A}$$

$$+ h'_{n,l,k,t_n} * \frac{\partial}{\partial \beta_{k,l,v'}} \log\left\{\frac{\exp\sum_{v=1}^{V}\left(\beta_{k,l,v}x_{n,t_n,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)}\right\} \longleftarrow \quad \textbf{B}$$

$$+ \sum_{\substack{j=2 \\ j\neq l}}^{S} h'_{n,j,k,t_n} * \frac{\partial}{\partial \beta_{k,l,v'}} \log\left\{\frac{\exp\sum_{v=1}^{V}\left(\beta_{k,j,v}x_{n,t_n,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)}\right\} \longleftarrow \quad \textbf{C}$$

----------(A.20)

Simplifying each of the terms, we get:

$$\mathrm{A} = h'_{n,1,k,t_n} * \frac{\partial}{\partial \beta_{k,l,v'}} \log\left\{ \frac{1}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)} \right\}$$

$$= h'_{n,1,k,t_n} * \frac{1}{\left\{ \frac{1}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)} \right\}} * \frac{(-1)*\exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)*x_{n,t_n,v'}}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)\right\}^2}$$

$$= -x_{n,t_n,v'} * h'_{n,1,k,t_n} * P(q_{n,t_n} = l \mid q_{n,t_{n-1}} = k, X_{n,t}\ \Theta) \qquad \text{---------(A.21)}$$

$$\mathrm{B} = h'_{n,l,k,t_n} * \frac{\partial}{\partial \beta_{k,l,t_n}} \log\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)} \right\}$$

$$= h'_{n,l,k,t_n} * \frac{1}{\left\{ \frac{\exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)}{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)} \right\}}$$

$$* \left[\frac{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)\right\}*\exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)*x_{n,t_n,v'} - \exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)*\exp\left(\sum_{v=1}^{V}\beta_{k,l,v}x_{n,t_n,v}\right)*x_{n,t_n,v'}}{\left\{1+\sum_{i=2}^{S}\exp\left(\sum_{v=1}^{V}\beta_{k,i,v}x_{n,t_n,v}\right)\right\}^2} \right]$$

$$= x_{n,t_n,v'} * h'_{n,l,k,t_n} * \left\{1 - P(q_{n,t_n} = l \mid q_{n,t_{n-1}} = k, X_{n,t_n}, \Theta)\right\} \qquad \text{-------(A.22)}$$

$$\mathrm{C} = \sum_{\substack{j=2 \\ j \neq l}}^{S} h'_{n,j,k,t_n} * \frac{\partial}{\partial \beta_{k,l,v'}} \log \left\{ \frac{\exp \sum_{v=1}^{V} \left(\beta_{k,j,v} x_{n,t_n,v} \right)}{1 + \sum_{i=2}^{S} \exp \left(\sum_{v=1}^{V} \beta_{k,i,v} x_{n,t_n,v} \right)} \right\}$$

$$= \sum_{\substack{j=2 \\ j \neq l}}^{S} h'_{n,j,k,t_n} * \frac{1}{\left\{ \frac{\exp \sum_{v=1}^{V} \left(\beta_{k,j,v} x_{n,t_n,v} \right)}{1 + \sum_{i=2}^{S} \exp \left(\sum_{v=1}^{V} \beta_{k,i,v} x_{n,t_n,v} \right)} \right\}} * \frac{\exp \left(\sum_{v=1}^{V} \beta_{k,j,v} x_{n,t_n,v} \right) * \left\{ -\exp \left(\sum_{v=1}^{V} \beta_{k,l,v} x_{n,t_n,v} \right) \right\} * x_{n,t_n,v'}}{\left\{ 1 + \sum_{i=2}^{S} \exp \left(\sum_{v=1}^{V} \beta_{k,i,v} x_{n,t_n,v} \right) \right\}^2}$$

$$= \sum_{\substack{j=2 \\ j \neq l}}^{S} - h'_{n,j,k,t_n} * x_{n,t_n,v'} * P(q_{n,t_n} = l \mid q_{n,t_{n-1}} = k, X_{n,t_n}, \Theta) \qquad \text{----------(A.23)}$$

Hence,

$$\frac{\partial Q_{C,n,t_n}}{\partial \beta_{k,l,v'}} = \mathrm{A} + \mathrm{B} + \mathrm{C}$$

$$= -x_{n,t_n,v'} * P(q_{n,t_n} = l \mid q_{n,t_{n-1}} = k, X_{n,t_n}, \Theta) * \left(\sum_{j=1}^{S} h'_{n,j,k,t_n} \right) + x_{n,t_n,v'} * h'_{n,l,k,t_n}$$

Adding contribution from all samples and all positions within samples, we get,

$$\frac{\partial Q_C}{\partial \beta_{k,l,v'}} = \sum_{n=1}^{N} \sum_{t_n=2}^{T_n} - x_{n,t_n,v'} * \left\{ P(q_{n,t_n} = l \mid q_{n,t_{n-1}} = k, X_{n,t_n}, \Theta) * \sum_{j=1}^{S} h'_{n,j,k,t_n} - h'_{n,l,k,t_n} \right\} \qquad \text{------(A.24)}$$

APPENDIX B

CSPI MANUAL

B.1 DESIGN AND DESCRIPTION OF PYTHON SCRIPTS

B.1.1 Domain Objects (DO)

a. **DO/spectrumDO.py:** Classes for MS/MS spectrum and spectrum peaks. These contain data and methods for handling spectrum files, including spectrum pre-processing steps.

b. **DO/sequenceDO.py:** Generic sequence class to store an amino-acid sequence and the corresponding FASTA header.

c. **DO/proteinDO.py:** Inherits Sequence class, containing protein sequence-specific data/methods.

d. **DO/peptideDO.py:** Generic peptide class that inherits from the Sequence class.

e. **DO/candidatePeptideDO.py:** Inherits the Peptide class, and contains information for a peptide in context of a specific Spectrum, like charge-state.

f. **DO/fragmentDO.py:** Generic peptide fragment class.

g. **DO/matchedFragmentDO.py:** Inherits Fragment class, and contains information for a fragment in context of a specific spectrum, like a Boolean variable "observed/not observed".

h. **DO/constantsDO.py:** Contains data values that remain fixed, for ex. properties of amino acids like their masses, hydrophobicity, gas-phase basicity etc.

B.1.2 Data Access Objects (DAO)

a. **DAO/fastaReader.py:** Parser for protein FASTA databases. Given a FASTA file, this script is used to read in protein sequences and return Protein objects.

b. **DAO/peptideIndexerDAO.py:** Script for creating and querying indexes generated from protein FASTA files, for fast retrieval of candidates during Database Search.

c. **DAO/resultsFileParserDAO.py:** Contains parsers for extracting relevant data from results files of CSPI, Crux and X!Tandem.

d. **DAO/searchResultRecordDAO.py:** Container for a particular peptide-spectrum match, used in post-processing of database search results for storing the scores from database search as well as FDR/q-value.

B.1.3 Processing Scripts (BO)

a. **BO/trainingEngineBO.py:** Main script for initializing trainer, reading in training data and starting the trainer.

b. **BO/searchEngineBO(_mp).py:** Main script for performing database search. “*_mp*” version utilizes multiprocessing to speed up processing of large MS/MS datasets.

c. **BO/ioHmmBO_mp.py:** Script containing the details of the EM algorithm as well as the scorer class that scores PSMs.

d. **BO/modelFamilyBO_mp.py:** Script containing classes for transition and emission models used as components in the CSPI framework. Their methods include computation of maximum likelihood parameter estimates as well as relevant methods for computing probability density/distributions.

e. **BO/psmEngineBO.py:** Script for evaluating a PSM, including matching theoretical with experimental spectrum, computing input and output layers of CSPI models and computing fragmentation statistics (fracMatch and fracExplained)

f. **BO/fdrAnalysisBO.py:** Script for performing fdr analysis on search results from various algorithms (CSPI, Crux, X!Tandem)

B.1.4 Parameters (Params)

a. **Params/applicationParams.py**: Contains all the parameters used in training and applying CSPI models, as described above.

B.2 PARAMETERS (TO BE SPECIFIED IN THE SCRIPT PARAMS/APPLICATIONPARAMS.PY)

B.2.1 Training CSPI models

a. **maxNoOfIterations:** Maximum number of iterations for GEM training (integer; default = 500)

b. **relDiff_dataLogLik_thresh:** Relative difference in data log likelihood in order for EM to converge (floating point; default = 0.0001)

c. **maxIter:** Number of steps in the conjugate gradient used in the M step of the GEM algorithm (integer; default = 2)

d. **seed:** Floating point seed for random initialization (numeric; default = “None”, in which case system clock is used)

e. **paramEvolutionFile_TP(FP)_b(y):** File name storing the concatenated list of parameters for each iteration of the GEM algorithm, for true (TP) and Null (FP) models for b (or y) fragment ion-types

f. **params_IoHmm_TP(FP)_filename_b(y)ions:** parameter filenames for storing the final parameters for trained CSPI models

g. **TP(FP)_psmMap_file:** Path to the training data files for True (TP) and Null (FP) models

h. **spectrumParentDir:** Path to the directory containing spectrum directories

i. **spectrumDirName:** Name of the directory containing spectrum files

j. **spectrumDir:** Path to the directory containing MS/MS spectrum files

k. **modelFamilyDict_True(Null)_b(y):** Dictionary storing the model types for emission and transition functions

l. **results_parentDir:** Parent directory for storing all the outputs to various scripts

m. **trainingResultsDir:** Directory for storing parameters and parameter-evolution files

n. **noOfProcs:** Number of child processes to create for training and searching (integer).

B.2.2 Database search and CSPI Scoring

a. **dbFilename:** Protein database file (full path)

b. **MH_Lower:** Lower threshold of peptide (+ proton) mass in daltons, i.e. ($M+H^+$)

c. **MH_Upper:** Upper threshold of peptide (+ proton) mass in daltons, i.e. ($M+H^+$)

d. **precursorMassType:** Type of mass used to compute peptide mass (Average or monoisotopic) (0 or 1; default=1, for Avg)

e. **precursorPepError:** Error tolerance to search candidate peptides from database (floating point; default=+/- 3 Da)

f. **fragMassType**: Type of mass used to compute fragment mass (Average or monoisotopic) (0 or 1; default=0, for Mono)

g. **fragmentError:** Error tolerance to match peptide fragments with spectrum peaks (floating point; default=+/- 0.5 Da)

h. **enzyme**: Enzyme used in Protein digestion (char string; typically “Trypsin”)

i. **cleavageMode:** Extent of cleavage enzyme specificity to use for searching candidates (0,1 or 2; default=2, i.e. full enzyme specificity)

j. **maxMissCleavage:** number of allowable internal enzyme-specific sites in peptides (integer; default=3)

k. **candidateFilterLevel:** Size of filtered candidate peptides' list to evaluate using CSPI models

l. **searchResultsDir:** Directory for storing database search results

m. **searchResultsFileName_b(y or byAdded)Model:** Filename to store database search results

n. **noOfTopRanksToReport:** Number of top-ranking candidate peptides to report for each spectrum (integer; default=10)

B.2.3 Protein FASTA Database Indexing

a. **mzBinSize:** m/z range to cover per index file (float; default=25 Da). Peptides in the mass range (MH_Lower, MH_upper) are considered in database search. Multiple index files are generated covering subsequent 'mzBinSize' Da units.

b. **Index_parentDir:** Parent directory where index files are stored

B.3 RUNNING THE SCRIPTS

B.3.1 Protein FASTA Indexing

Depending upon the task, the following lines of code are added inside DAO/peptideIndexerDAO.py, which can then be run as the main script:

-- Index Generation:

```
> writer = PeptideIndexWriter()

> writer.index_proteins()
```

-- Index Query:

```
> reader = PeptideIndexReader()

> reader.search(<queryMH>)
```

(where where, ‘PeptideIndexReader’ and ‘PeptideIndexWriter’ are classes defined in DAO/peptideIndexerDAO.py; first line instantiates an object of the class while the second line calls a method defined in the class; queryMH = Expected Mass of peptide + Proton that is extracted from the spectrum file being evaluated)

B.3.2 CSPI models’ training

The following lines of code are added inside BO/trainingEngineBO.py, which can then be run as the main script:

```
> trainingEngine = pepIoHmm_Train(<Model Type>, <Ion Type>,
                                   <paramEvolutionFile>, <trainedParamsFile>)
```

```
>   trainingEngine.start()
```

(where, ‘pepIoHmm_Train’ is a class defined in BO/trainingEngineBO.py; first line instantiates an object of the class while the second line calls a method defined in the class; Model Type can be “True” or “False” and Ion Type can be “b” or “y”; arguments paramEvolutionFile and trainedParamsFile are stated in the Params/applicationParams.py file)

B.3.3 Database Searching

The following lines of code are added inside BO/searchEngineBO_mp.py, which can then be run as the main script:

```
> searchEngine = SearchEngine(<spectrumDir>, <paramsFile_b_TP>,
                              <paramsFile_y_TP>, <paramsFile_b_FP>,
                              <paramsFIle_y_FP>, <searchResultsFile_bModel>,
                              <searchResultsFile_yModel>,
                              <searchResultsFile_byAdded>)
> searchEngine.start(False)
```

(where, ‘SearchEngine’ is a class defined in BO/searchEngineBO_mp.py; first line instantiates an object of the class while the second line calls a method defined in the class; all the arguments are specified in the Params/applicationParams.py file)

B.4 FILE FORMATS

B.4.1 Spectrum files

Currently, CSPI framework supports spectrum files in the SEQUEST ‘dta’ format

B.4.2 Files generated from CSPI Training (Location of the following files is specified in Params/applicationParams.py)

a. **paramEvolutionFile**

This file contains the record of CSPI models’ parameters as they evolve through the iterative EM algorithm. Each record consists of concatenated parameter values from the Initial-state logistic model, transition logistic models and emission models. The number of records in the file is the same as the number of iterations it took for the training procedure to converge.

b. paramsFile

This file contains the record of trained CSPI models’ parameters, listed in the following order: Initial state logistic function, transition probability logistic functions and emission function parameters. For logistic function models, all the parameters/weights from one weight vector are concatenated (with comma-separator) and stored on one line. Hence, for ex., with four hidden states each logistic model contains three weight vectors, each of which are listed on a separate row. Similarly, parameters for each emission model are concatenated and put on one row.

B.4.3 Search Results' files

CSPI results files are simple ASCII text with the following components:

a. **Header** for the following column labels: SpectrumIndex (unique id for a spectrum in the dataset), SpectrumName (spectrum filename), LogLR_b (CSPI_Score^{b}), True_LogLik_b (CSPI_Score^{bTrue}), Null_LogLik_b (CSPI_Score^{bNull}), LogLR_y (CSPI_Score^{y}), True_LogLik_y (CSPI_Score^{yTrue}), Null_LogLik_y (CSPI_Score^{yNull}), Score (CSPI_Score^{byAdded}), MH (Mass of Peptide + proton), FrontChar (Amino acid of the Protein just ahead of the peptide sequence), Sequence (Peptide Amino Acid Sequence), EndChar (Trailing Amino Acid of the peptide sequence), FastaHeader (FASTA header sequence of the parent protein))
b. **Results Records**: containing values for each of the columns (see header) for top **'n'** candidates for each spectrum, where **'n'** is specified in the Params/applicationParams.py script

BIBLIOGRAPHY

Aebersold, R. and D. R. Goodlett (2001). "Mass spectrometry in proteomics." Chem Rev **101**(2): 269-295.

Aebersold, R. and M. Mann (2003). "Mass spectrometry-based proteomics." Nature **422**(6928): 198-207.

Anderson, N. L., M. Polanski, et al. (2004). "The human plasma proteome: a nonredundant list developed by combination of four separate sources." Mol Cell Proteomics **3**(4): 311-326.

Bengio, Y. and P. Frasconi (1995). An Input Output HMM Architecture. Advances in Neural Information Processing Systems.

Bengio, Y., V.-P. Lauzon, et al. (2001). "Experiments on the Application of IOHMMs to Model Financial Returns Series." IEEE Trans. Neural Networks **12**(1): 113-123.

Benjamini, Y. and Y. Hochberg (1995). "Controlling the false discovery rate: a practical and powerful approach to multiple testing." Journal of the Royal Statistical Society **57**(1): 289-300.

Bishop, C. (1996). Neural Networks for Pattern Recognition. New York, Oxford University Press.

Bishop, C. (2007). Pattern Recognition and Machine Learning, Springer.

Breci, L. A., D. L. Tabb, et al. (2003). "Cleavage N-terminal to proline: analysis of a database of peptide tandem mass spectra." Anal Chem **75**(9): 1963-1971.

Casella G, B. R. (2001). Statistical Inference, Duxbury Press.

Choi, H., D. Ghosh, et al. (2008). "Statistical validation of peptide identifications in large-scale proteomics using the target-decoy database search strategy and flexible mixture modeling." J Proteome Res **7**(1): 286-292.

Choi, H. and A. I. Nesvizhskii (2008). "False discovery rates and related statistical concepts in mass spectrometry-based proteomics." J Proteome Res **7**(1): 47-50.

Choi, H. and A. I. Nesvizhskii (2008). "Semisupervised model-based validation of peptide identifications in mass spectrometry-based proteomics." J Proteome Res **7**(1): 254-265.

Cortes, C. and V. Vapnik (1995). "Support-Vector Networks." Machine Learning **20**(3): 273-297.

Craig, R. and R. C. Beavis (2004). "TANDEM: matching proteins with tandem mass spectra." Bioinformatics **20**(9): 1466-1467.

Craig, R., J. P. Cortens, et al. (2004). "Open source system for analyzing, validating, and storing protein identification data." J Proteome Res **3**(6): 1234-1242.

de Godoy, L. M., J. V. Olsen, et al. (2006). "Status of complete proteome analysis by mass spectrometry: SILAC labeled yeast as a model system." Genome Biol **7**(6): R50.

Dempster A, L. N., Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." J of the Royal Statistical Society **39**(1): 1-38.

Desiere, F., E. W. Deutsch, et al. (2006). "The PeptideAtlas project." Nucleic Acids Res **34**(Database issue): D655-658.

Ding, Y., H. Choi, et al. (2008). "Adaptive discriminant function analysis and reranking of MS/MS database search results for improved peptide identification in shotgun proteomics." J Proteome Res **7**(11): 4878-4889.

Durbin R, E. S., Krogh A, Mitchison G (1999). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press.

Eidhammer, I., K. Flikka, et al. (2007). Computational Methods for Mass Spectrometry Proteomics, John Wiley and Sons Ltd.

Elias, J. E., F. D. Gibbons, et al. (2004). "Intensity-based protein identification by machine learning from a library of tandem mass spectra." Nat Biotechnol **22**(2): 214-219.

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

Eng, J., A. McCormack, et al. (1994). "An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database." Journal of the American Society for Mass Spectrometry **5**(11): 976-989.

Ernst, J., O. Vainas, et al. (2007). "Reconstructing dynamic regulatory maps." Molecular Systems Biology **3**: 74.

Feng, J., D. Q. Naiman, et al. (2007). "Probability-based pattern recognition and statistical framework for randomization: modeling tandem mass spectrum/peptide sequence false match frequencies." Bioinformatics **23**(17): 2210-2217.

Gibbons, F. D., J. E. Elias, et al. (2004). "SILVER helps assign peptides to tandem mass spectra using intensity-based scoring." J Am Soc Mass Spectrom **15**(6): 910-912.

Henderson J, S. S., Fasman KH (1997). "Finding genes in DNA with a Hidden Markov Model." J Comput Biol **4**(2): 127-141.

Higdon, R., N. Kolker, et al. (2004). "LIP index for peptide classification using MS/MS and SEQUEST search via logistic regression." OMICS **8**(4): 357-369.

Hosmer, D. W. and S. Lemeshow (2000). Applied Logistic Regression, Wiley-Interscience.

Huang, Y., J. M. Triscari, et al. (2005). "Statistical characterization of the charge state and residue dependence of low-energy CID peptide dissociation patterns." Anal Chem **77**(18): 5800-5813.

Huang, Y., G. C. Tseng, et al. (2008). "A data-mining scheme for identifying peptide structural motifs responsible for different MS/MS fragmentation intensity patterns." J Proteome Res **7**(1): 70-79.

Hubbard, S. J., A. R. Jones, et al. (2010). Understanding and Exploiting Peptide Fragment Ion Intensities Using Experimental and Informatic Approaches. Proteome Bioinformatics, Humana Press. **604:** 73-94.

Jean-Fran\, \#231, et al. (2009). "Probabilistic models for melodic prediction." Artif. Intell. **173**(14): 1266-1274.

Jensen, F. V. and T. D. Nielsen (2007). Bayesian networks and decision graphs. New York, Springer.

Kall, L., J. D. Canterbury, et al. (2007). "Semi-supervised learning for peptide identification from shotgun proteomics datasets." Nat Methods **4**(11): 923-925.

Kall, L., J. D. Storey, et al. (2008). "Assigning significance to peptides identified by tandem mass spectrometry using decoy databases." J Proteome Res **7**(1): 29-34.

Kandasamy, K., A. Pandey, et al. (2009). "Evaluation of Several MS/MS Search Algorithms for Analysis of Spectra Derived from Electron Transfer Dissociation Experiments." Analytical Chemistry **81**(17): 7170-7180.

Kapp, E. and F. Schutz (2007). "Overview of tandem mass spectrometry (MS/MS) database search algorithms." Curr Protoc Protein Sci **Chapter 25**: Unit25 22.

Kapp, E. A., F. Schutz, et al. (2005). "An evaluation, comparison, and accurate benchmarking of several publicly available MS/MS search algorithms: sensitivity and specificity analysis." Proteomics **5**(13): 3475-3490.

Kapp, E. A., F. Schutz, et al. (2003). "Mining a tandem mass spectrometry database to determine the trends and global factors influencing peptide fragmentation." Anal Chem **75**(22): 6251-6264.

Karplus K, B. C., Cline M, Diekhans M, Grate L, Hughey R (1999). "Predicting protein structure using only sequence information." Proteins: Structure, Function, and Bioinformatics **37**: 121-125.

Keller, A., A. I. Nesvizhskii, et al. (2002). "Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search." Anal Chem **74**(20): 5383-5392.

Khatun, J., E. Hamlett, et al. (2008). "Incorporating sequence information into the scoring function: a hidden Markov model for improved peptide identification." Bioinformatics **24**(5): 674-681.

Klammer AA, P. C., Noble WS (2009). "Statistical Calibration of the Sequest XCorr Function." J Proteome Res **8**(4): 2106-2113.

Klammer, A. A., S. M. Reynolds, et al. (2008). "Modeling peptide fragmentation with dynamic Bayesian networks for peptide identification." Bioinformatics **24**(13): i348-356.

Klimek, J., J. S. Eddes, et al. (2008). "The standard protein mix database: a diverse data set to assist in the production of improved Peptide and protein identification software tools." J Proteome Res **7**(1): 96-103.

Ma, B. (2010). "Challenges in Computational Analysis of Mass Spectrometry Data for Proteomics." Journal of Computer Science and Technology **25**(1).

Mann, M., R. C. Hendrickson, et al. (2001). "Analysis of proteins and proteomes by mass spectrometry." Annu Rev Biochem **70**: 437-473.

Marcotte, E. (2007). "How do shotgun proteomics algorithms identify proteins?" Nature Biotechnology **25**(7): 755-757.

Martens, L., H. Hermjakob, et al. (2005). "PRIDE: the proteomics identifications database." Proteomics **5**(13): 3537-3545.

Murphy, K. (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. Computer Science Ph.D. Thesis, UC Berkeley.

Nesvizhskii, A. I. (2007). "Protein identification by tandem mass spectrometry and sequence database searching." Methods Mol Biol **367**: 87-119.

Nesvizhskii, A. I., O. Vitek, et al. (2007). "Analysis and validation of proteomic data generated by tandem mass spectrometry." Nat Methods **4**(10): 787-797.

Ning, K. and H. W. Leong (2007). "Algorithm for peptide sequencing by tandem mass spectrometry based on better preprocessing and anti-symmetric computational model." Comput Syst Bioinformatics Conf **6**: 19-30.

Olsen, M. A., K. Bostic, et al. (1999). Berkeley DB. Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference.

Paizs, B. and S. Suhai (2005). "Fragmentation pathways of protonated peptides." Mass Spectrom Rev **24**(4): 508-548.

Park, C. Y., A. A. Klammer, et al. (2008). "Rapid and accurate peptide identification from tandem mass spectra." J Proteome Res **7**(7): 3022-3027.

Perkins, D. N., D. J. Pappin, et al. (1999). "Probability-based protein identification by searching sequence databases using mass spectrometry data." Electrophoresis **20**(18): 3551-3567.

Rabiner, L. (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." Proceedings of the IEEE **77**(2): 257-286.

Renard, B. Y., M. Kirchner, et al. (2009). "When less can yield more - Computational preprocessing of MS/MS spectra for peptide identification." Proteomics **9**(21): 4978-4984.

Roepstorff, P. and J. Fohlman (1984). "Proposal for a common nomenclature for sequence ions in mass spectra of peptides." Biomed Mass Spectrom **11**(11): 601.

Searle, B. C., M. Turner, et al. (2008). "Improving sensitivity by probabilistically combining results from multiple MS/MS search methodologies." J Proteome Res **7**(1): 245-253.

Steen, H. and M. Mann (2004). "The ABC's (and XYZ's) of peptide sequencing." Nat Rev Mol Cell Biol **5**(9): 699-711.

Storey, J. D. and R. Tibshirani (2003). "Statistical significance for genomewide studies." Proc Natl Acad Sci U S A **100**(16): 9440-9445.

Syka, J. E., J. J. Coon, et al. (2004). "Peptide and protein sequence analysis by electron transfer dissociation mass spectrometry." Proc Natl Acad Sci U S A **101**(26): 9528-9533.

Tabb, D. L., Y. Huang, et al. (2004). "Influence of basic residue content on fragment ion peak intensities in low-energy collision-induced dissociation spectra of peptides." Anal Chem **76**(5): 1243-1248.

Tsaprailis, G., H. Nair, et al. (2004). "A mechanistic investigation of the enhanced cleavage at histidine in the gas-phase dissociation of protonated peptides." Anal Chem **76**(7): 2083-2094.

Vaisar, T. and J. Urban (1996). "Probing the proline effect in CID of protonated peptides." J Mass Spectrom **31**(10): 1185-1187.

Vapnik, V. N. (1998). Statistical learning theory. New York, Wiley.

Vitek, O. (2009). "Getting started in computational mass spectrometry-based proteomics." PLoS Comput Biol **5**(5): e1000366.

Wan, Y., A. Yang, et al. (2006). "PepHMM: a hidden Markov model based scoring function for mass spectrometry database search." Anal Chem **78**(2): 432-437.

Wysocki, V. H., K. A. Resing, et al. (2005). "Mass spectrometry of peptides and proteins." Methods **35**(3): 211-222.

Wysocki, V. H., G. Tsaprailis, et al. (2000). "Mobile and localized protons: a framework for understanding peptide dissociation." J Mass Spectrom **35**(12): 1399-1406.

Xu, H. and M. A. Freitas (2009). "MassMatrix: a database search program for rapid characterization of proteins and peptides from tandem mass spectrometry data." Proteomics **9**(6): 1548-1555.

Zhou, C., L. D. Bowler, et al. (2008). "A machine learning approach to explore the spectra intensity pattern of peptides using tandem mass spectrometry data." BMC Bioinformatics **9**: 325-341.