Published in final edited form as:

J Proteome Res. 2016 August 5; 15(8): 2749–2759. doi:10.1021/acs.jproteome.6b00290.

A dynamic Bayesian network for accurate detection of peptides from tandem mass spectra

John T. Halloran[†], Jeff A. Bilmes[†], and William S. Noble[‡]

John T. Halloran: halloj3@uw.edu; William S. Noble: william-noble@uw.edu

[†]Department of Electrical Engineering, University of Washington, Seattle, WA

[‡]Department of Genome Sciences, University of Washington, Seattle, WA

Abstract

A central problem in mass spectrometry analysis involves identifying, for each observed tandem mass spectrum, the corresponding generating peptide. We present a dynamic Bayesian network (DBN) toolkit that addresses this problem by using a machine learning approach. At the heart of this toolkit is a DBN for Rapid Identification (DRIP), which can be trained from collections of high-confidence peptide-spectrum matches (PSMs). DRIP's score function considers fragment ion matches using Gaussians rather than fixed fragment-ion tolerances and also considers all possible alignments between the theoretical and observed spectrum to find the optimal such alignment. This function not only yields state-of-the-art database search accuracy but also can be used to generate features that significantly boost the performance of the Percolator post-processor. The DRIP software is built upon a general purpose DBN toolkit (GMTK), thereby allowing a wide variety of options for user-specific inference tasks, as well as facilitating easy modifications to the DRIP model in future work. DRIP is implemented in Python and C++, and is available under Apache license at <http://melodi-lab.github.io/dripToolkit>.

Graphical abstract

Keywords

tandem mass spectrometry; machine learning; Bayesian network; peptide detection

Introduction

Given a complex sample, liquid chromatography followed by tandem mass spectrometry, often referred to as shotgun proteomics, produces a large collection of output spectra, typically numbering in the tens or hundreds of thousands. Ideally, each spectrum represents of a single peptide species that was present in the original complex sample. Most often, the generating peptides of these observed spectra are identified by performing a database search where, for each candidate peptide within the specified mass tolerance window of a spectrum's observed precursor mass, a score function computes the quality of the match between the observed spectrum and the candidate peptides' theoretical spectrum. Typically only the top-scoring peptide-spectrum match (PSM) for a given observed spectrum is retained for further analysis.

Thus, a critical task in shotgun proteomics analysis is designing a scoring algorithm that accurately identifies the peptide responsible for generating an observed spectrum. Many scoring algorithms exist, some of the most popular and accurate of which are SEQUEST¹ and its open source variants,^{2,3} Mascot,⁴ X!Tandem,⁵ MS-GF+,⁶ OMSSA,⁷ MS-Amanda⁸ and Morpheus.⁹ Recently, we described a score function called DRIP¹⁰ that uses techniques developed in the field of machine learning to achieve state-of-the-art search accuracy. Critically, DRIP includes parameters that can be learned automatically using a training set of high-confidence PSMs. Furthermore, DRIP explicitly models two important phenomena prevalent in shotgun proteomics data: spurious observed peaks and missing theoretical fragment ions. We refer to these as *insertions* and *deletions*, respectively, relative to a theoretical peptide spectrum. DRIP scores PSMs by computing the most probable sequence of insertions and deletions, dynamically aligning a theoretical spectrum and observed spectrum in a manner similar to classical methods for minimizing edit distances between two strings¹¹ or bioinformatics techniques like BLAST¹² for protein or DNA string alignment.

In this work, we extend the DRIP search algorithm in several important ways, adding better support for charge state variation and improving the scoring of high-resolution fragmentation spectra. We also demonstrate that DRIP's score function provides value beyond simply ranking PSMs. The popular machine learning post-processor, Percolator re-ranks a given collection of target and decoy PSMs by using a semi-supervised machine learning strategy.¹³ We show that using DRIP in an intermediate step after a search but before Percolator analysis significantly improves post-processing accuracy. In particular, in conjunction with two other search engines (MS-GF+ and Tide), DRIP processing improves Percolator's results by ~13% at a false discovery rate (FDR) threshold of 1%.

The DRIP Toolkit is freely available under an Apache open source license at <http://melodi-lab.github.io/dripToolkit>.

Methods

Bayesian networks

A Bayesian network (BN) is a type of probabilistic graphical model consisting of a directed acyclic graph, in which the nodes correspond to random variables and the directed edges correspond to potential conditional dependencies in the graph. A BN encodes the manner in which the joint distribution over a set of random variables *factorizes*, i.e., decomposes into a product of conditional distributions. The manner in which this factorization occurs may be intuitively read from the BN, such that each factor involved in the overall product is the probability of a random variable conditioned on its parents. For example, the joint distribution of the length 4 Markov chain in Figure 1(a) factorizes as $p(X_1, X_2, X_3, X_4) = p(X_1)p(X_2|X_1)p(X_3|X_2)p(X_4|X_3)$. The joint distribution of the naive Bayes model in Figure 1(b) factorizes as $p(X_1, X_2, X_3, X_4) = p(X_1|X_4)p(X_2|X_4)p(X_3|X_4)p(X_4)$. And the joint distribution of the BN in Figure 1(c) factorizes as $p(X_1, X_2, X_3, X_4) = p(X_1)p(X_2)p(X_3)p(X_4|X_1, X_2, X_3)$.

The graphical structure and factorization of the joint distribution encoded by a BN allow efficient algorithms for computing probabilistic quantities of interest regarding the underlying random variables. For instance, consider the Markov chain in Figure 1(a) and assume that we would like to derive the marginal distribution $p(X_4)$. Assume that $n = |\mathcal{X}_1| = |\mathcal{X}_2| = |\mathcal{X}_3| = |\mathcal{X}_4|$, where we denote the “state spaces” of X_1, X_2, X_3 , and X_4 as $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$, and $\mathcal{X}_4$, respectively. In other words, X_i is a random variable whose values are drawn from the alphabet $\mathcal{X}_i$, which is typically a set of integers from 0 through $|\mathcal{X}_i| - 1$. Computing $p(X_4)$ involves the marginalization of variables X_1, X_2 , and X_3 , i.e., $p(X_4) = \sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \sum_{x_3 \in \mathcal{X}_3} p(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4)$. In general, such a computation will require $O(n^4)$ operations. However, by the factorization of the BN, we have

$$\begin{aligned} p(X_4) &= \sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \sum_{x_3 \in \mathcal{X}_3} p(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4) \\ &= \sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \sum_{x_3 \in \mathcal{X}_3} p(X_1)p(X_2|X_1)p(X_3|X_2)p(X_4|X_3) \\ &= \sum_{x_3 \in \mathcal{X}_3} p(X_4|X_3) \sum_{x_2 \in \mathcal{X}_2} p(X_3|X_2) \sum_{x_1 \in \mathcal{X}_1} p(X_1)p(X_2|X_1). \end{aligned}$$

Computing $\psi(X_2) = \sum_{x_1 \in \mathcal{X}_1} p(X_1)p(X_2|X_1)$ requires $O(n^2)$ operations, as do $\phi(X_3) = \sum_{x_2 \in \mathcal{X}_2} p(X_3|X_2)p(X_4|X_3)$ and $p(X_4) = \sum_{x_3 \in \mathcal{X}_3} p(X_4|X_3)\phi(X_3)$, where $\psi(X_2)$ and $\phi(X_3)$ may be thought of as messages being passed in the network. Thus, by exploiting the factorization of the BN, we have gone from a computational cost of $O(n^4)$ down to $O(n^2)$. Exploiting the factorization properties encoded by a graph to compute quantities possibly involving only subsets of all nodes is a foundation behind the inference algorithms utilized in graphical models, and may be used to efficiently derive quantities such as marginal distributions or the most probable configuration of random variables.

Dynamic Bayesian networks (DBNs) are BNs for which the number of variables is allowed to vary. DBNs are most often used to model sequential data, such as speech,¹⁴ biological,¹⁵ or economic¹⁶ sequences. Each time instance of the sequence is modeled by a *frame*

(sometimes called a *time slice*), which consists of a set of random variables, edges amongst these variables, and incoming/outgoing edges from/to variables in adjacent frames. As an example, a variable length *T hidden Markov model* (HMM) is depicted in Figure 2, where the sequence of random variables $X_1, X_2, \dots, X_T$ are *hidden*, i.e., stochastic, and often referred to as the *hidden layer*. The sequence of shaded variables $O_1, O_2, \dots, O_T$ are *observed*, i.e., each variable takes on a single value (which could be vectorial but typically is a scalar), and are often referred to as the *observed layer*. The observed layer is typically used to model a temporal sequence of observed phenomena, while the hidden layer is typically used to model the underlying random process assumed to produce the observations. Given a length T sequence of observations, we may instantiate the BN in order to perform inference. In practice, however, T may be quite large. As such, highly optimized inference algorithms designed specifically for DBNs may be run, such as the *forward* and *backward recursions*, which allow one to recursively compute forward and backward messages, thus avoiding the memory overhead of the instantiated unrolled graph. For an in depth discussion of DBNs and their inference algorithms, and in particular this sort of model, the reader is directed to Ref. 17.

DBN for Rapid Identification of Peptides (DRIP)

Given an observed spectrum and a peptide, DRIP dynamically aligns the observed and theoretical spectra without quantization of the m/z axis, in a manner similar to minimizing edit distance between two strings.¹¹ To perform this alignment, DRIP explicitly models *insertions* (spurious observed peaks) and *deletions* (theoretical peaks not found in the observed spectrum). Each frame of the model corresponds to an observed peak, so that each frame contains an m/z and intensity observation. Figure 3 depicts the DRIP graph for an example observed spectrum.

DRIP considers all possible traversals of the theoretical spectrum, from left to right, and all possible scorings of observed peaks, either by matching an observed peak to a nearby theoretical peak or by treating the observed peak as an insertion. Importantly, observed-theoretical matches are scored using Gaussians, the means of which may be learned. The alignment strategy used in DRIP allows one to encode and learn different alignment properties, such as larger deletions taking on smaller probabilities and closer observed peaks (to theoretical peaks) receiving higher scores than insertions. Model variables for the traversal of the theoretical spectrum and the scoring of observed peaks are highlighted in light blue and light red, respectively, in Figure 3. We now further describe these two mechanisms in detail. For further DRIP model details, including the exact form of DRIP's scoring function, the reader is directed to Ref. 10.

Traversing theoretical peaks—To determine the alignment between a theoretical and observed spectrum, DRIP considers all possible traversals of the theoretical spectrum, where we define a *traversal* to be an increasing (i.e., left-to-right) sequence of theoretical peaks for all frames. As an example, consider the observed spectrum in Figure 3, the DRIP graph present in the same figure, and a low-resolution MS2 theoretical spectrum (denoted as a vector) $v = (113, 146, 247, 300, 510)$, where $v(i)$ is the i th element of the vector for $1 \leq i \leq 5$. Theoretical peaks in DRIP correspond to Gaussians centered near theoretical peak m/z

values. Traversal of the theoretical peaks is controlled by the variables highlighted in light blue in Figure 3.

The traversal proceeds as follows. Let t be an arbitrary frame number. The variable n is observed to be the number of total theoretical peaks. The discrete random variable $\delta_t \in [0, n - 1]$ dictates the number of theoretical peaks DRIP moves down in a particular frame. The variable K_t , a deterministic function of its parents, denotes the index of the theoretical peak being considered in a particular frame, such that $p(K_0 = \delta_0 | \delta_0) = 1$ and, for $t > 1$, $p(K_t = K_{t-1} + \delta_t | K_{t-1}, \delta_t) = 1$. To ensure that δ_t does not increment past the number of remaining theoretical peaks to consider, we have $p(\delta_t = n | n) = 0$ and, for $t > 1$, $p(\delta_t = n - K_{t-1} | K_{t-1}, n) = 0$. For a given theoretical peak index K_t , DRIP accesses the K_t th theoretical peak $v(K_t)$ and uses this to obtain a Gaussian centered near this theoretical peak. For instance, when $\delta_1 = 2$, we have $K_1 = 2$, $v(K_1) = 146$ and the second Gaussian in Figure 4 is considered; when $K_3 = 2$, $\delta_4 = 3$, we have $K_4 = 5$, $v(K_4) = 510$ and the last Gaussian in Figure 4 is considered. Thus, $\delta_1 > 0$ or, for $t > 1$, $\delta_t > 1$ correspond to deletions. In order to penalize larger deletions, the distribution over δ_t is monotone decreasing. Ultimately, the sequence of $\delta_1, \delta_2, \dots, \delta_T$ dictates the sequence of deletions and, equivalently, the theoretical spectrum traversal (for an example, see Figure 5(a)). All possible sequences of deltas are considered so that DRIP iterates through all possible traversals (displayed in Figure 5(b)).

Scoring observed m/z and intensity values—We now describe how observed peaks are scored using Gaussians. For each element in a theoretical spectrum traversal (i.e., the theoretical peak considered in a particular frame), DRIP must decide whether to treat the observed peak as a noise peak (insertion) or not. Let K_t be the index of the theoretical peak considered in frame t , and let $v(K_t)$ be the theoretical peak value. The variable O_t is an observed random vector containing the t^{th} observed peak's m/z and intensity values. The Bernoulli random variable i_t (Figure 3) denotes whether the t^{th} peak is an insertion. When $i_t = 0$, we score the t^{th} observed peak's m/z value using a Gaussian centered near $v(K_t)$. When $i_t = 1$, the observed peak is considered an insertion, and the m/z observation is scored using a constant insertion penalty (Figure 6). The insertion penalty c is such that m/z observations receive no worse a score than a value $d/2$ Thomsons away from the Gaussian center. Thus, m/z observations closer to the Gaussian center (i.e., closer to theoretical peak locations) receive higher scores. The constant insertion penalty limits the dynamic range of scores, thereby ensuring that different PSM scores are comparable to one another. Similarly, all intensity observations (which are unit normalized) are scored using a unit-mean Gaussian when $i_t = 0$, and scored using a constant insertion penalty when $i_t = 1$. Note that, as a sequence of deltas dictates a sequence of deletions, $i_1, i_2, \dots, i_T$ denotes the sequence of insertions.

For low-resolution MS2 spectra, the m/z bin size d is set to 1.0005079 and the m/z means and intensity variance may be generatively learned using the expectation-maximization algorithm.¹⁸ Note that the learned intensity variance used for the results in the sequel is an order of magnitude larger than the m/z variance, so that matching observed peaks closer to learned means is prioritized over simply matching high intensity peaks during inference. For high-resolution MS2 spectra, $d = 0.05$ by default and may be user-specified, and the m/z means are set to the real values of the respective theoretical peak's fragment ion. When d

and, subsequently, the m/z variance are set smaller than the low-resolution MS2 case (i.e., ~ 1 Thomson), matching observed peaks closer to m/z Gaussian means is prioritized even more highly than the low-resolution MS2 case.

Dynamically aligning the theoretical and observed spectra—Having established how individual peaks are scored, we now consider how the full observed-theoretical alignment is created. In particular, we refer to a particular traversal of the theoretical spectrum and the subsequent scoring of observed peaks as an *alignment* (Figure 7). A particular alignment is thus uniquely determined by the values of $i_1, i_2, \dots, i_7$ and $\delta_1, \delta_2, \dots, \delta_7$, i.e., the sequences of insertions and deletions. DRIP considers every possible alignment to calculate the most probable sequence of insertions and deletions, which corresponds to the maximal-scoring alignment between the observed and theoretical spectra. The alignment between theoretical and observed spectra is one of the strengths of using a DBN, and of DRIP, in particular, as the alignment may be non-linear. The local alignment scores for various alignment hypotheses of one region of the pair of spectra, therefore, may depend on the results of alignment in other local regions. This is not possible with purely additive scores, such as the scores used by SEQUEST,¹ X!Tandem,⁵ Morpheus,¹⁹ MS-GF+⁶ and OMSSA.⁷ More importantly, in DRIP, as mentioned above, these alignment scores may be learned automatically based on training data.

Note that methods which quantize or match fragment ions within fixed m/z widths of observed/theoretical peaks may be thought of as considering a static alignment, since, given such a scheme, there exists only one alignment between the theoretical and observed spectra. In contrast, DRIP may be thought of as dynamically aligning the theoretical and observed spectra. The dynamic alignment may be equal to the static alignment given a fixed fragment-match-error strategy if the static alignment happens to be the most probable one, but may also be drastically different (e.g., when the optimal fragment match error tolerance varies for different datasets, such as those produced by different high-resolution MS2 experiment configurations).

Approximate inference via beam pruning—Although the number of possible alignments grows exponentially in both the number of theoretical peaks and the number of observed peaks, posing this problem as inference in a DBN enables efficient algorithms to compute the maximal alignment while also affording avenues to effectively speed up runtimes via approximate inference algorithms. In particular, DRIP's dynamic alignment strategy is ideal for a particular class of approximate inference methods called *beam pruning*. In k -beam pruning (called “histogram pruning” in Ref. 20), assuming a beam width of k (an integer), only the top k most probable states in a frame are allowed to persist. During inference in DRIP, the scoring of the last several observed peaks by the first few theoretical peaks is likely to produce low-probability alignments, as is the scoring of the first few observed peaks by the last several theoretical peaks (and so on). This necessarily means that, for the current frame (say, t) for which messages are being passed during inference, we may prune low-probability states (i.e., filter from further consideration).

Such pruning has a profound effect on the state space of the DBN since, by filtering the partial-alignment evaluated up to t , we are filtering all alignments which are produced by

this partial alignment. For instance, in Figure 5 at frame $t = 1$, by pruning the hypothesis that the second theoretical peak scores the first observed peak, we prune away all alignments where the second theoretical peak scores the first observed peak (this event is assigned zero probability). Such pruning may lead to significant computational savings, because we avoid considering an exponential number (in T) of low-probability alignments. On the other hand, in practice, care must be taken so that k is not made too small such that the most probable alignment is not filtered.

DRIP extracted features

Inference in DRIP not only returns the probability of the maximal alignment (used to score peptides during database search), but also detailed information regarding a PSM in the form of the most probable sequences of insertions and deletions. In this work, we demonstrate that the insertions and deletions inferred by DRIP provide features that can be effectively employed by Percolator¹³ to better recalibrate PSMs. These features are described in Table 1, along with all other features used for Percolator analysis in this work. In this setting, we employ DRIP as a feature extractor in conjunction with any search method: the search method's target and decoy PSMs are fed into DRIP, DRIP features are extracted and appended to a larger set of Percolator features, and Percolator analysis is then performed.

Software details

The open-source DRIP Toolkit is written in Python 2.7. DBN inference is performed by the Graphical Models Toolkit (GMTK),^{22,23} which supports many different DBN inference algorithms, such as Viterbi inference (currently used in DRIP), sum-product inference, and many approximate inference algorithms for speeding up runtime. The use of GMTK allows easy alterations to the DRIP model, so that in future DRIP may be easily tailored for specific inference tasks of interest. The DRIP Toolkit is available for download at <http://melodi-lab.github.io/dripToolkit>.

Prior to a search, the dripDigest module must be run to digest the protein database. dripDigest accepts as input a FASTA file of proteins, digests the proteins according to a user-specified memory budget (so that variable modifications, missed cleavages, and partial digestions may be efficiently evaluated), and outputs digested peptides to a binary file. Prior to a low-resolution fragment ion search, DRIP parameters may be learned given a tab-delimited file of high-confidence PSMs and the corresponding MS2 spectra in the form of an .ms2 file using the dripTrain module.

Searches are performed using the dripSearch module, which accepts as inputs MS2 spectra in the form of an .ms2 file, the output of dripDigest, and (optionally) the output of dripTrain. dripSearch performs a database search and writes DRIP PSMs to a tab-delimited file. To help speed up search times, dripSearch supports multithreading, data division for easy cluster distribution, and approximate inference options as supported by GMTK.

Features for post-search Percolator analysis are extracted using the dripExtract module, which accepts as inputs a Percolator PIN file containing the PSMs and features of a search algorithm (such as Tide or MS-GF+) and the corresponding MS2 spectra in the form of an .ms2 file. dripExtract performs inference to compute DRIP features and writes these

features, along with any input features, to a PIN file. dripExtract supports multithreading and approximate inference options, as provided by GMTK.

DRIP also supports analysis of PSMs using the Python interactive shell via the dtk module. dtk allows instantiation of PSM objects, command-line inference of DRIP PSMs (using GMTK), and plotting the most probable alignments of PSMs (via matplotlib). PSMs may be instantiated one at a time or as a file of PSMs specified in DRIP's tab-delimited output format.

Tandem mass spectra

The yeast (*Saccharomyces cerevisiae*) and worm (*C. elegans*) data sets were collected using tryptic digestion followed by acquisition using low-resolution precursor scans and low-resolution fragment ions. Each dataset exhibited charge 1+, 2+, and 3+ spectra. Each search was performed using a ± 3.0 Thomson (Th) tolerance for selecting candidate peptides. Peptides were derived from proteins using tryptic cleavage rules without proline suppression and allowing no missed cleavages. A single fixed carbamidomethyl modification was included. Further details about these data sets, along with the corresponding protein databases, may be found in Ref. 13.

The *Plasmodium falciparum* sample was digested using Lys-C, labeled with an isobaric tandem mass tag (TMT) relabeling agent, and collected using high-resolution precursor scans and high-resolution fragment ions. The data set consists of 12,594 spectra with charges ranging from 2+ through 6+. Searches were run using a 50 ppm tolerance for selecting candidate peptides, a fixed carbamidomethyl modification, a fixed TMT labeling modification of lysine and N-terminal amino acids. Further details may be found in Ref. 24.

The relevant features of all datasets are summarized in Table 2.

Search algorithm settings

All XCorr scores and XCorr *p*-values were collected using Crux v2.1.16567.²⁵ Both types of scores were collected using tide-search with flanking peaks not allowed. X!Tandem version 2013.09.01.1 was used, with PSMs scored by E-value. MS-GF+ scores were collected using MS-GF+ version 9980, with PSMs ranked by E-value. Default MS-GF+ parameters were used except that, to make a fair comparison with other methods, isotope peak errors were not allowed and methionine clipping was turned off. DRIP scores were collected using dripSearch, parameters trained using dripTrain and a high-confidence set of PSMs²⁶ (except for the "DRIP default" results in Figure 8, which employ the initial parameters used for training), and default settings.

When searching the *Plasmodium* dataset, the following settings were used to take advantage of the high-resolution fragment ions. XCorr used a fragment ion error tolerance of 0.03 (mz-bin-width=0.03) and fragment ion offset of 0.5 (mz-bin-offset=0.5). XCorr *p*-values are currently not designed to take advantage of the increased fragment ion resolution and so were collected with default settings. MS-GF+ was run with -inst 1, denoting a high resolution fragment ion instrument. X!Tandem was run with fragment monoisotopic mass

error equal to 0.03 Th. dripSearch was run with --precursor-filter true and --high-res-ms2 true.

Results

Learning model parameters boosts statistical power to detect peptides

Employing maximum likelihood estimation via the expectation-maximization algorithm^{18,27} and a set of high-confidence PSMs,²⁶ we learn DRIP's m/z Gaussian means and intensity Gaussian variance for low-resolution MS2 spectra. This learning procedure improves identification accuracy, relative to the default, evenly spaced Gaussian means. In all searched data sets (Figure 8), the learned Gaussian parameters lead to more identifications than the default parameters. Furthermore, the trained model is more accurate at identifying highly confident PSMs (Figure 9).

DRIP is competitive with state-of-the-art search algorithms

Searching using DRIP is competitive with other state-of-the-art methods (Figure 10). DRIP yields significantly more identifications than all other methods for the worm and *Plasmodium* datasets, which contain much lower percentages of identified peaks (i.e., many more insertions) than the other data sets (percentages shown in Table 3). We hypothesize that this difference is due to DRIP's ability to accurately estimate insertions, as shown below (Figure 13). We demonstrate in Figure 11 that the DRIP's score function does a better job of assigning target peptides as the top-ranked match when no score threshold is applied. We call the percentage of assigned peptides that are targets the "Relative ranking percentage," because this percentage reflects the ability of the score function to rank candidate peptides relative to a given spectrum, irrespective of calibration of scores between spectra.

Faster search with *k*-beam pruning—A DRIP search may be significantly sped up by adjusting the beam-pruning width *k* (described in Methods), without compromising search accuracy. Figure 12 illustrates search accuracy using DRIP with various beam widths, while Table 4 shows dripSearch runtimes for various beam widths. All reported runtimes were averaged over five runs on the same machine, with an Intel Core i7-3770 3.40GHz processor and 16 GB of memory using eight threads (set using num-threads in dripSearch). A beam width of 75 allows us to achieve the same performance as exact inference (the default setting for dripSearch) while reducing runtime by 22.6%. Reducing *k* below 75 further reduces runtimes, but degrades the inferred maximal alignment and reduces search performance. For example, *k* = 25 reduces exact inference runtime by nearly 50% but significantly diminishes search accuracy.

DRIP-derived features improve Percolator post-processing

We demonstrate that the features derived using the inferred DRIP sequences of insertions and deletions are highly accurate and beneficial for Percolator analysis. Figure 13 displays Percolator postprocessing performance of MS-GF+ (solid lines) and Tide *p*-value (dashed lines) PSMs for all yeast and worm datasets. DRIP derived features are appended to standard feature sets commonly used for the respective search algorithm (Table 1) and used in Percolator analysis. We show the utility of deriving these features in DRIP by comparing

against analogous features derived by quantizing the observed spectrum. In all data sets presented in Figure 13, the DRIP extracted features allow Percolator to better distinguish between target and decoy PSMs, as evidenced by the significantly improved number of identifications returned by Percolator postprocessing.

Dynamic alignments and lack of fixed fragment-ion-match tolerance improve DRIP feature extraction—Many existing score functions match observed and theoretical fragment ion peaks using either a discretized m/z axis or a discrete matching function (e.g., a window of ± 0.5 Th around the theoretical peak). DRIP's dynamic alignment strategy and its determination of fragment ion matches via Gaussians allows detection of matches outside regularly used fragment-ion-match tolerances (Figure 14). Such information does not account for the bulk of matches (these matches account for the tails of the y -axis histograms in Figure 14) and, thus, do not account for the bulk of improvement extracting features using DRIP. However, detecting such matches are beneficial for Percolator analysis, displayed in Figure 15.

Conclusions

We have shown that DRIP may be effectively trained for low-resolution MS2 spectra, leading to significantly improved search accuracy. We note that the parameters utilized herein were learned via maximum likelihood (ML) estimation and display the flexibility of effective parameter learning afforded by the DRIP model. Future investigation of effective strategies for learning high-resolution MS2 DRIP parameters may be explored by altering the open-source DRIP Toolkit.

We have shown that searching both low and high-resolution spectra using DRIP is competitive with state-of-the-art database search algorithms, only producing less identifications per q -value than the methods MS-GF+ and Tide p -values on four of the nine considered datasets. Furthermore, we've shown that the information detailing PSMs returned by performing inference in DRIP (i.e., a PSM's most probable sequences of insertions deletions) improve Percolator postprocessing analysis for the highly accurate search methods Tide p -value and MS-GF+. All presented DRIP features are available in the DRIP Toolkit. The DRIP Toolkit provides an easily modifiable platform for researchers to utilize DRIP as well as customize the DRIP model and/or inference algorithm to suit specific needs, thanks to the use of the DBN-inference engine GMTK. With the generality afforded by GMTK, users may easily introduce new variables to model particular phenomena in tandem mass spectra or to switch from Viterbi inference to sum-product inference in order to, for example, compute posteriors with respect to particular model variables or sets of model variables.

In addition to the generative model parameter estimation described above, many machine learning frameworks for highly accurate parameter estimation exist, such as max-margin learning²⁸ and conditional maximum likelihood estimation.²⁹ These may also be used for training the parameters of peptide identification models. While we do not discuss it above, DRIP also supports discriminative (i.e., conditional likelihood) based parameter training via

the discriminative training mechanism that exists in GMTK.²³ Future work will elaborate on the benefits of this approach to tandem mass spectrometry-based peptide identification.

Acknowledgments

This work was funded by National Institutes of Health awards R01 GM096306 and P41 GM103533.

References

1. Eng JK, McCormack AL, Yates JR III. Journal of the American Society for Mass Spectrometry. 1994; 5:976–989. [PubMed: 24226387]
2. Eng JK, Jahan TA, Hoopmann MR. Proteomics. 2012; 13:22–24. [PubMed: 23148064]
3. Diamant BJ, Noble WS. Journal of Proteome Research. 2011; 10:3871–3879. [PubMed: 21761931]
4. Perkins DN, Pappin DJC, Creasy DM, Cottrell JS. Electrophoresis. 1999; 20:3551–3567. [PubMed: 10612281]
5. Craig R, Beavis RC. Bioinformatics. 2004; 20:1466–1467. [PubMed: 14976030]
6. Kim S, Pevzner PA. Nature communications. 2014; 5
7. Geer LY, Markey SP, Kowalak JA, Wagner L, Xu M, Maynard DM, Yang X, Shi W, Bryant SH. Journal of Proteome Research. 2004; 3:958–964. [PubMed: 15473683]
8. Dorfer V, Pichler P, Stranzl T, Stadlmann J, Taus T, Winkler S, Mechtler K. Journal of Proteome Research. 2014; 13:3679–3684. [PubMed: 24909410]
9. Wenger CD, Coon JJ. Journal of Proteome Research. 2013; 12:1377–1386. [PubMed: 23323968]
10. Halloran, JT.; Bilmes, JA.; Noble, WS. Learning Peptide-Spectrum Alignment Models for Tandem Mass Spectrometry. Uncertainty in Artificial Intelligence (UAI); Quebec City, Quebec Canada: 2014.
11. Levenshtein, V. Soviet Physics Doklady. 1966. Binary codes capable of correcting deletions, insertions and reversals; p. 707-710.
12. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Journal of Molecular Biology. 1990; 215:403–410. [PubMed: 2231712]
13. Käll L, Canterbury J, Weston J, Noble WS, MacCoss MJ. Nature Methods. 2007; 4:923–25. [PubMed: 17952086]
14. Bilmes J, Bartels C. IEEE Signal Processing Magazine. 2005; 22:89–100.
15. Hoffman MM, Buske OJ, Wang J, Weng Z, Bilmes JA, Noble WS. Nature Methods. 2012; 9:473–476. [PubMed: 22426492]
16. Hassan, MR.; Nath, B. Stock market forecasting using hidden Markov model: a new approach. Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on; 2005; p. 192-196.
17. Bilmes J. Signal Processing Magazine, IEEE. 2010; 27:29–42.
18. Dempster AP, Laird NM, Rubin DB. Journal of the Royal Statistical Society. Series B (Methodological). 1977; 39:1–22.
19. Wenger CD, Coon JJ. Journal of proteome research. 2013
20. Volker Steinbiss HN, Tran Bach-Hiep. ICSLP. 1994
21. Granholm V, Kim S, Navarro JC, Sjölund E, Smith RD, Käll L. Journal of proteome research. 2013; 13:890–897. [PubMed: 24344789]
22. Bilmes, J.; Zweig, G. The Graphical Models Toolkit: An Open Source Software System for Speech and Time-Series Processing. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing; 2002;
23. Bilmes, J. The Graphical Models Toolkit (GMTK) Documentation. 2015. <https://melodi.ee.washington.edu/gmtk/>
24. Wu L, Candille SI, Choi Y, Xie D, Jiang L, Li-Pook-Than J, Tang H, Snyder M. Nature. 2013; 499:79–82. [PubMed: 23676674]

25. McIlwain S, Tamura K, Kertesz-Farkas A, Grant CE, Diamant B, Frewen B, Howbert JJ, Hoopmann MR, Käll L, Eng JK, MacCoss MJ, Noble WS. *Journal of Proteome Research*. 2014; 13:4488–4491. [PubMed: 25182276]
26. Klammer AA, Reynolds SM, Bilmes JA, MacCoss MJ, Noble WS. *Bioinformatics*. 2008; 24:i348–356. [PubMed: 18586734]
27. Bilmes JA. *International Computer Science Institute*. 1998; 4:126.
28. Taskar, B.; Guestrin, C.; Koller, D. *Advances in Neural Information Processing Systems*. Cambridge, MA: 2003. Max margin Markov networks.
29. Povey, D. PhD thesis. University of Cambridge; 2003. Discriminative training for large vocabulary speech recognition.

Figure 1.
Examples of Bayesian networks.

Figure 2.
Length T hidden Markov model.

Figure 3.

DRIP graph for a single observed spectrum, where each frame in the DBN corresponds to an observed peak. Short variable descriptors appear to the far left on the same horizontal level as the described variable. Variable subscripts denote the frame number. Red edges in the DBN denote Gaussian conditional distributions, black edges denote deterministic functions of parents, and blue edges denote switching parents (i.e., parents whose values may change the conditional distributions of their children).

Figure 4.

Example DRIP theoretical spectrum, where theoretical peaks in DRIP correspond to Gaussians centered near theoretical peak m/z values.

Figure 5.

Demonstration of all theoretical spectrum traversals in DRIP. For the theoretical spectrum in Figure 4, we color code each of DRIP's theoretical peak Gaussians. We then color the observed peaks in Figure 3 according to the theoretical peak currently active in a frame.

Figure 6.

Gaussian score and insertion penalty used to score m/z observations. The m/z Gaussian variance is set such that 99.9% of the Gaussian mass lies within d . The insertion penalty, c , is such that m/z observations receive no worse a score than a value $d/2$ away from the Gaussian center.

Figure 7.

Alignment for theoretical spectrum traversal corresponding to $\delta_1 = 0$, $\delta_2 = 2$, $\delta_3 = 0$, $\delta_4 = 1$, $\delta_5 = 0$, $\delta_6 = 0$, $\delta_7 = 1$ and scoring of the second, sixth observed peaks as insertions and all other peaks using Gaussians, i.e., $i_1 = 0$, $i_2 = 1$, $i_3 = 0$, $i_4 = 0$, $i_5 = 0$, $i_6 = 1$, $i_7 = 0$. Observed peaks in gray denote insertions, while other observed peak's colors denote which color-coded theoretical spectrum Gaussian is used to score them.

Figure 8.

Each panel plots the number of accepted PSMs as a function of q -value threshold. Series correspond to searching using default DRIP parameters and searching using DRIP parameters learned using a previously described set of high-confidence PSMs.²⁶ Eight data sets are evaluated, four from *C. elegans* (Worm) and four from *Saccharomyces cerevisiae* (Yeast).

Figure 9.

Bar plot displaying the percentage of PSMs accepted at $q < 0.01$ by DRIP that are also accepted by four other search methods (MS-GF+, Tide XCorr, Tide XCorr p -value, and X! Tandem). Bars correspond to the DRIP trained model versus DRIP with default parameters on four worm and four yeast data sets.

Figure 10.

The figure plots the number of accepted PSMs as a function of q -value threshold for four search methods: DRIP, Tide with exact p -values, MS-GF+, and X!Tandem. The panels correspond to the worm, yeast, and *Plasmodium* data sets.

Figure 11.

Bar plot displaying, for the nine data sets in Figure 10, the percentage of spectra for which a target peptide wins the target-decoy competition.

Figure 12.
Searching Yeast-1 with varying beam-pruning widths used.

Figure 13.

Plots displaying the number of accepted PSMs as a function of q -value threshold for searches using MS-GF+ (solid lines) and Tide with exact p -values (dashed lines) followed by Percolator analysis. Features used in these analyses are described in Table 1. The three MS-GF+/Tide exact p -value series in each figure correspond to analyses with (1) the standard feature set, (2) the feature set augmented with DRIP-extracted features, and (3) the feature set augmented with features analogous to the DRIP-extracted features but derived by quantizing the m/z axis.

Figure 14.

Histograms displaying the fragment match difference (FMD), i.e., the difference between the theoretical peak center and scored m/z observation for a fragment ion match, computed using DRIP for Worm-1, Worm-2 charge +2 target PSMs. The top PSMs per spectrum were selected using Tide p -values and fed into dripExtract.

Figure 15.

Figures displaying the number of accepted PSMs at $q < 0.1$, identified using Tide with exact p -values followed by Percolator processing for the four worm datasets. The two series correspond to Percolator analysis using DRIP features calculated with and without including detected matches greater than ~ 0.5 Thomsons from the DRIP-matched Gaussian mean.

Table 1

Features used for Percolator analysis. DRIP features (denoted by column D) are extracted using DRIP's inferred sequences of insertions and deletions for each PSM. Tide features (denoted by column T) are the ones used by the stand-alone Percolator application, as computed by Crux. MS-GF+ features (denoted by column M) are those described in Ref. 21.

Feature	Description	D	T	M
insertions	Number of inserted observed peaks	X		
deletions	Number of deleted theoretical peaks	X		
scoredPeaks	Number of non-inserted observed peaks	X		
usedTheoPeaks	Number of non-deleted theoretical peaks	X		
sumScoredIntensities	Sum of the intensities of non-inserted observed peaks	X		
sumScoredMz	Sum of the absolute differences between non-inserted observed peaks and the DRIP Gaussian mean used to score those peaks	X		
enzN	Is the peptide preceded by an enzymatic (tryptic) site?		X	X
enzC	Does the peptide have an enzymatic (tryptic) C-terminus?		X	X
enzInt	Number of missed internal enzymatic (tryptic) sites		X	X
ChargeC	Boolean indicating if PSM charge is C		X	X
PepLen	Peptide length		X	X
dm	Difference between the observed precursor and peptide mass		X	X
absdM	Absolute value of the difference between the observed precursor and peptide mass		X	X
Sp	Sp score		X	
lnrSp	Natural logarithm of sp score		X	
deltLCn	Difference between a PSM's XCorr and the XCorr of the last-ranked PSM, divided by the PSM's XCorr or 1, whichever is larger		X	
deltCn	Difference between a PSM's XCorr and the XCorr of the next-ranked PSM, divided by the PSM's XCorr or 1, whichever is larger		X	
IonFrac	Estimated fraction of b and y ions theoretical ions matched to the spectrum		X	
Mass	The observed mass [M+H] ⁺		X	
lnNumSP	Natural logarithm of the number of database peptides within the specified precursor range		X	
RawScore	MS-GF+ base score (dot-product)			X
DeNovoScore	Maximum possible RawScore for observed spectrum			X
ScoreRatio	RawScore/DeNovoScore			X
Energy	RawScore – DeNovoScore			X
lnEValue	– log(MS-GF+ E value)			X
lnSpecEValue	– log(MS-GF+ Spectral E value)			X
IsotopeError	Number of additional neutrons in peptide			X
LnExplainedCurrentRatio	$\log \frac{\sum \{\text{intensity of matched fragment ions}\}}{\sum \{\text{intensity of all fragment ions}\}}$			X
LnNtermIonCurrentRatio	$\log \frac{\sum \{\text{intensity of matched N- terminal fragments}\}}{\sum \{\text{intensity of all fragment ions}\}}$			X

Feature	Description	D	T	M
LnCtermIonCurrentRatio	$\log \frac{\sum \{\text{intensity of matched C-terminal fragments}\}}{\sum \{\text{intensity of all fragment ions}\}}$			X
LnMs2IonCurrent	$\log \sum \{\text{intensity of all fragment ions}\}$			X
Mass	Peptide mass			X
MeanErrorTop7	$\frac{1}{7} \sum \{\text{Mass errors of the 7 highest intensity fragment ion peaks}\}$			X
sqMeanErrorTop7	$(\text{MeanErrorTop7})^2$			X
StdevErrorTop7	Standard deviation of mass errors of the 7 highest intensity fragment ion peaks			X

Table 2

Summary of presented datasets.

Data set	Spectra	Charges	precursor mass tolerance
Worm-1	22,693	1–3	±3.0 Th
Worm-2	21,862	1–3	±3.0 Th
Worm-3	20,011	1–3	±3.0 Th
Worm-4	23,697	1–3	±3.0 Th
Yeast-1	35,236	1–3	±3.0 Th
Yeast-2	37,641	1–3	±3.0 Th
Yeast-3	35,414	1–3	±3.0 Th
Yeast-4	35,467	1–3	±3.0 Th
<i>Plasmodium</i>	12,594	2–6	±50 ppm

Table 3

Average percentage of identified peaks percentage for the yeast, worm, and *Plasmodium* datasets found in Figure 10, calculated using the b/y ions matched output column of crux tide-search divided by the number of observed spectrum peaks.

Dataset	Average <i>PIP</i>
Worm-1	1.7856
Worm-2	1.9210
Worm-3	1.9042
Worm-4	1.9648
Yeast-1	4.4077
Yeast-2	4.3373
Yeast-3	4.3365
Yeast-4	4.3353
<i>Plasmodium</i>	3.8700

DRIP search times (per spectrum per charge) for 200 randomly chosen charge 2+ and 3+ Yeast-1 spectra with various beam widths (beam option of dripSearch).

Table 4

beam width	$k = \infty$	$k = 100$	$k = 75$	$k = 50$	$k = 25$
Runtime in seconds	3.9683	3.313	3.0734	2.7031	2.0484