Towards the Prediction of Protein Abundance from Tandem Mass Spectrometry Data

Anthony J Bonner[*] Han Liu[†]

Abstract

This paper addresses a central problem of Proteomics: estimating the amounts of each of the thousands of proteins in a cell culture or tissue sample. Although laboratory methods involving isotopes have been developed for this problem, we seek a simpler method, one that uses more-straightforward laboratory procedures. Specifically, our aim is to use data-mining techniques to infer protein levels from the relatively cheap and abundant data available from high-throughput tandem mass spectrometry (MS/MS). This paper develops and evaluates several techniques for tackling this problem. First, we develop three generative models of MS/MS data. Each model represents a different hypothesis about the way MS/MS data is generated, the main difference between them being their treatment of peptide ionization. Second, for each model, we develop a family of methods for efficiently fitting the model to data. Each method is based on optimizing an objective function, and we prove that the methods achieve the optimum values. Finally, to evaluate the biological relevance of the models and methods, we test them on three real-world datasets generated by MS/MS experiments performed on various tissue samples taken from Mouse. The evaluations are carried out on four different representations of the data and involve two distinct evaluation criteria, one based on correlation coefficients and one based on data visualization.

keywords: Bioinformatics, Proteomics, Data Mining, Proteins, Peptides, Tandem Mass Spectrometry.

1 Introduction.

Proteomics is the large-scale study of the thousands of proteins in a cell [10]. In a typical Proteomics experiment, the goal might be to compare the proteins present in a certain tissue under different conditions. For instance, a biologist might want to study cancer by comparing the proteins in a cancerous liver to the proteins in a healthy liver. Modern mass spectrometry makes this possible by enabling the identification of thousands of proteins in a complex mixture [14, 4]. However, *identifying* proteins is only part of the story. It is also important to *quantify* them, that is, to estimate how much of each protein is present in a cell [1, 5]. To this end, a number of laboratory methods have been developed, notably those based on mass tagging with isotopes [6, 13]. However, simpler, more-direct methods may be possible, methods that do not require additional laboratory procedures, but which are simply based on the data provided by tandem mass spectrometers [11]. This paper is an initial exploration of this possibility. In particular, we investigate the use of data-mining techniques to infer protein quantity from tandem mass spectrometry data.

1.1 Tandem Mass Spectrometry. Tandem mass spectrometry involves several phases in which proteins are broken up and the pieces separated by mass [10, 14]. First, a complex mixture of thousands of unknown proteins is extracted from a cell culture or tissue sample. Since proteins themselves are too large to deal with, they are fragmented, producing a mixture of tens of thousands of unknown peptides. The peptides are then ionized and passed through a mass spectrometer. This produces a mass spectrum in which each spectral peak corresponds to a peptide. From this spectrum, individual peptides are selected for further analysis. Each such peptide is further fragmented and passed through a second mass spectrometer, to produce a so-called tandem mass spectrum. The result is a collection of tandem mass spectra, each corresponding to a peptide. Each tandem mass spectrum acts as a kind of fingerprint, identifying the peptide from which it came. By searching a database of proteins, it is possible to identify the protein that produced the peptide that produced the tandem mass spectrum. In this way, the proteins in the original tissue sample are identified. Often, the entire process is completely automatic.

A peptide mixture is not analyzed all at once. Instead, to increase sensitivity, the peptides are "smeared out" over time (often using liquid chromatography), so that different kinds of peptides enter the mass spectrometer at different times. A typical MS/MS experiment may last many hours, with proteins and peptides being identified each second. Copies of a particular peptide

*Department of Computer Science, University of Toronto, bonner@cs.toronto.edu

†Department of Computer Science, University of Toronto

may continue to enter the mass spectrometer for several seconds or minutes. As the copies enter, the peptide will be repeatedly identified, once a second. In this way, a peptide may be identified and re-identified many times, increasing the confidence that the identification is correct. Each identification of a peptide is called a *spectral count*, since it requires the generation of a tandem mass spectrum. A large spectral count indicates that a peptide has been confidently identified.

In general, as protein abundance increases, so does spectral count [11]. However, the exact relationship is not at all clear and seems to depend on many factors, including the amino acid sequence of the peptides and the properties of the experimental set up. At present, there is no complete quantitative theory relating a protein's abundance to the spectral counts of its peptides. This paper is an initial attempt at using data-mining techniques to develop such a theory, and using the theory to estimate protein abundance.

1.2 Data Mining. We develop and evaluate three generative models of MS/MS data, and for each, we develop a family of methods for efficiently fitting the model to data. Because this is an initial study, the models were chosen for their simplicity and tractability, and the goal is to see how well (or poorly) they fit the data, and to quantify the error. Each model predicts the spectral count of a peptide based on two factors: its amino-acid sequence, and the abundance of the protein from which it was derived. The three models differ in their treatment of peptide ionization. However, they each provide an explanation for a recently observed linear relationship between protein abundance and spectral count [11]. More importantly, we show how to use each model to estimate protein abundance from spectral count.

To evaluate the models, the Emili Laboratory at the Banting and Best Department of Medical Research at the University of Toronto has provided us with datasets of several thousand proteins and peptides. The datasets were derived from MS/MS experiments on protein mixtures extracted from various tissue samples of Mouse. Each mixture contains tens of thousands of proteins, and each protein is present in the mixture with a specific (but unknown) abundance. A small sample of the data is shown in Table 1. (Details on how this data was generated can be found in [9].) Each row in the table represents a peptide ion. The first (left-most) column is the Swissprot accession number identifying a protein. The second column is the amino-acid sequence of the peptide. The third column is the spectral count of the peptide, and the last column is its charge. Notice that there may be many entries for the same protein, since a single protein can produce many peptides, and each peptide can produce ions with different amounts of charge. Protein ID, Peptide and Charge define a key for the table, that is, they uniquely identify a row.

Table 1: A fragment of a data file

Protein ID	Peptide	Count	Charge
Q91VA7	$TRHNNLVIIR$	4	2
Q91VA7	$KLDLFAVHVK$	3	2
...	...	...	...

High-throughput MS/MS experiments can provide a large amount of data of this kind on which to train and test data-mining methods. However, they also introduce a complication, since the amount of protein input to the mass spectrometer is unknown. This can be seen in Table 1, where spectral count is provided, but protein abundance is not. Thus, it is in general unclear whether a low spectral count for a peptide is due to the properties of the peptide or to a small amount of protein at the input. One of the challenges is to untangle these two influences. What makes the problem approachable is that we have data on spectral counts for peptides from the *same protein*, so differences in their counts cannot be due to differences in protein abundance. The models and methods developed here were chosen, in part, because of their ability to exploit this information. In effect, they treat protein abundance as a latent, or hidden variable, whose value must be estimated. In addition, they lead to efficient algorithms based on well-developed operators of linear algebra (specifically, matrix inversion and eigenvector decomposition). Finally, we evaluate each of the methods on the MS/MS datasets provided by the Emili Laboratory.

2 Modeling the Data.

In this section, we present our three models of MS/MS data. Each model represents a different hypothesis about the way MS/MS data is generated. The main difference between them is their treatment of peptide ionization. In Section 4, we evaluate the effectiveness of each model.

2.1 Modeling Spectral Counts. To keep track of different proteins and peptides, we use two sets of indices, usually i for proteins and j for peptides. Proteins are numbered from 1 to N, and the peptides for the i^{th} protein are numbered from 1 to n_i. In addition, we use y to denote spectral count, and in to denote the amount of protein input to the mass spectrometer. Each protein has a unique abundance, and each peptide has a unique spectral count. Thus, in_i is the abundance of protein i, and y_{ij} is the spectral count of peptide j of protein i. With this notation, the following equation provides a simple model of spectral count:

$$y_{ij} = in_i \cdot ie_{ij} \tag{2.1}$$

This equation divides spectral count into two factors: in_i, the amount of protein from which peptide ij was generated; and ie_{ij}, the *ionization efficiency* of the peptide. Ionization efficiency can be thought of as the propensity of the peptide to ionize and contribute to a mass spectrum, though in this paper, we are using it to refer to all factors that contribute to a peptide's spectral count *other than* the amount of protein. In this way, we hope to untangle the amount of protein (which we want to estimate) from all other factors. Note that y_{ij} is observed, while in_i and ie_{ij} are both unknown. Of course, Equation 2.1 is not exact. It provides at best an approximate description of the data, and it is not yet clear what the errors look like. The rest of the paper develops the model in three different ways, fits each model to real MS/MS data, quantifies the errors, and estimates values for in_i and ie_{ij} in the process.

It should be noted that with the model and data described above, we can only learn *relative* values of protein abundance and ionization efficiency, not absolute values. This is because any solution to Equation 2.1 is unique only up to a constant: multiplying all the in_i by a constant, and dividing all the ie_{ij} by the same constant gives another, equally good solution. However, estimating the relative amounts of protein is an extremely useful biological result. Moreover, by using a small amount of calibration data, the relative values can all be converted to absolute values.

It is also worth noting that the model already accounts for an experimentally observed property of MS/MS data. Specifically, the abundance of a protein is known to be directly proportional to the total spectral count of its peptides [11]. Formally, $in_i = b_i \sum_j y_{ij}$, where b_i is an (unknown) proportionality constant that depends on the protein. The notion of ionization efficiency provides an explanation for this proportionality and a way of computing the constants b_i. In particular, it follows immediately from Equation 2.1 that

$$in_i = \frac{\sum_j y_{ij}}{\sum_j ie_{ij}} \tag{2.2}$$

In other words, $b_i = 1/\sum_j ie_{ij}$. Thus, in the model of Equation 2.1, learning ionization efficiencies, ie_{ij}, is the central problem in estimating protein abundance.[1]

[1]Recall that in this paper, the term "ionization efficiency" is used broadly to refer to all factors (other than protein abundance) that contribute to a peptide's spectral count.

2.2 Modeling Ionization Efficiency. In order to estimate relative values for these unknowns, we need a model of ionization efficiency. In this paper, we investigate three relatively simple models:

$$\begin{array}{lrcl} \text{Linear:} & ie_{ij} & = & \mathbf{x}_{ij} \bullet \beta \\ \text{Exponential:} & ie_{ij} & = & e^{\mathbf{x}_{ij} \bullet \beta} \\ \text{Inverse:} & ie_{ij} & = & 1/(\mathbf{x}_{ij} \bullet \beta) \end{array} \tag{2.3}$$

Here, β is a vector of parameters (to be learned), $\mathbf{x}_{ij}$ is a vector of (known) peptide properties, and $\bullet$ denotes the dot product (or inner product) of the two vectors. The peptide properties are all derived from the amino-acid sequence and charge of the peptide ion. They could include such things as length, mass, amino-acid composition, and estimates of various biochemical properties such as hydrophobicity, chargeability, pH under the experimental conditions, etc. Section 4.2 spells out the specific properties used in this study.

We investigate linear models because they are directly amenable to the techniques of linear algebra. We investigate exponential models because, by taking logs, they become linear. In addition, exponential models have the advantage that the ionization efficiency is guaranteed to be positive. In contrast, the linear model may produce a preponderance of positive values, but it sometimes produces negative values as well, which are meaningless (though very small negative values can be assumed to be zero).

The inverse model has a different motivation. Spectral counts have a very skewed distribution of values, ranging over several orders of magnitude, with most of the values concentrated at the very low end of the spectrum. In fact, Section 4.1 shows that the distribution is $O(1/y^2)$, where y denotes spectral count. It can be difficult to fit a linear model to data with this kind of distribution, since a small number of very large values tends to dominate the fit. Even if the largest values are removed, the next largest values dominate, ad infinitum. Taking logarithms helps, but even $\log(y)$ has a skewed distribution. However, $1/y$ has a uniform distribution, thus eliminating all skew. This is the motivation for the inverse model: to transform the data to a form that is more manageable. In addition, all the methods we develop for fitting the linear model are easily adapted to fit the inverse model.

Finally, it is worth noting that not all the data contains useful information. In fact, only those proteins that produce at least two observable peptides can be used for learning β. To see this, note that for each model, $y_{ij} = in_i \cdot f(\mathbf{x}_{ij}, \beta)$, for some function, f. If a protein produces only one peptide, then $j = 1$ and the protein yields only one equation: $y_{i1} = in_i \cdot f(\mathbf{x}_{i1}, \beta)$. This is the *only* equation containing the unknown value in_i. Once we know β, we can use this equation to estimate in_i. However, the equation provides absolutely no help in estimating β itself, since it does not constrain β in any way. In fact, the equation is trivially satisfied for *any* β by using $in_i = y_{i1}/f(\mathbf{x}_{i1}, \beta)$. If all the proteins produced only one peptide, then we would have no way to estimate β, since any β would fit the data exactly. For this reason, our methods ignore all proteins that produce only one peptide. In fact, the first method considered below explicitly requires at least two peptides per protein.

3 Fitting the Models to Data.

The models described above each require the estimation of a parameter vector, β. In addition, since the amounts of protein are unknown, each of the in_i is a latent, or hidden variable whose value must be estimated. Since a tissue sample may contain thousands or tens of thousands of proteins, we have thousands of hidden variables to estimate. If the values of these variables *were* known (*i.e.*, if they were included in the training data), then for each model, the problem of estimating β would reduce to multivariate linear regression. Unfortunately, the training data does *not* include protein abundance, as can be seen in Table 1. This makes each model non-linear in the unknowns. This section develops a number of methods for transforming these non-linear models into linear ones and for efficiently fitting them to data. In some cases, the problem still reduces to multivariate linear regression, but in most cases it reduces to generalized eigenvector problems. We consider each model in turn, and for each, we develop a family of methods for fitting the model to data.

3.1 Linear Models. This section develops three methods for fitting the linear model to experimental data, where each method is meant to improve upon the one before. The first two methods are closely related. They have in common that learning is divided into two phases: the first phase estimates a value for β, and the second phase uses β to help estimate values for the in_i. The two methods differ in the optimization criteria they use to fit the model to the data. The third method is different from the first two in that it has only one learning phase, in which all parameters are estimated simultaneously. In this way, we hope to get a better fit to the data, since the estimate of β is now affected by how well the estimates of in_i fit the data, something that is impossible in the two-phase approach.

The linear model is given by equations of the form

$$y_{ij} = in_i \cdot (\mathbf{x}_{ij} \bullet \beta) \tag{3.4}$$

where the parameter vector β and all the in_i are unknown and must be learned. Of course, these equations are not exact, and provide at best an approximate description of the data. The goal is to see how closely they fit the data, and to estimate values for β and in_i in the process. As discussed above, we know it is only possible to estimate *relative* values for these quantities. This effectively means we can determine the *direction* of β but not its *magnitude*. In fact, in the absence of calibration data, the magnitude of β is meaningless. For this reason, the methods described in this section all impose constraints on the magnitude of β in order to obtain a unique solution.

3.1.1 LIN1: Two-Phase Learning. This approach factors out protein abundance, in_i, from the set of Equations 3.4. The result is a set of linear eigenvector equations for the parameter vector β, which we can solve using standard eigenvector methods.

From Equations 3.4, we see that protein i gives rise to the following equations, one equation for each peptide:

$$\begin{aligned} y_{i1} &= in_i \cdot (\mathbf{x}_{i1} \bullet \beta) \\ y_{i2} &= in_i \cdot (\mathbf{x}_{i2} \bullet \beta) \\ &\cdots \\ y_{in_i} &= in_i \cdot (\mathbf{x}_{in_i} \bullet \beta) \end{aligned} \tag{3.5}$$

Note that the unknown value in_i is the same in each equation. Thus, by dividing each equation by the previous one, we can eliminate this unknown value, leaving the parameter vector β as the only unknown quantity. That is, $y_{ij}/y_{i,j-1} = (\mathbf{x}_{ij} \bullet \beta)/(\mathbf{x}_{i,j-1} \bullet \beta)$, for j from 2 to n_i. Cross multiplying gives $y_{ij}(\mathbf{x}_{i,j-1} \bullet \beta) = y_{i,j-1}(\mathbf{x}_{ij} \bullet \beta)$, and rearranging terms gives the following:[2]

$$\mathbf{z}_{ij} \bullet \beta = 0 \tag{3.6}$$

where $\mathbf{z}_{ij} = y_{ij}\mathbf{x}_{i,j-1} - y_{i,j-1}\mathbf{x}_{ij}$, for j from 2 to n_i, and i from 1 to N. Thus, for each value of i, we have eliminated a single unknown, in_i, and reduced the number of equations by 1. Geometrically, these equations mean that the parameter vector β is orthogonal to each of the derived vectors $\mathbf{z}_{ij}$. Note that this is a constraint on the direction of β but not its magnitude, which is to be expected. Equation 3.6 is a restatement of Equation 3.4 with the unknown values in_i removed. Like

[2]Of course, we could generate many more equations of this form by cross multiplying all possible pairs of equations from 3.5, instead of just the successive ones. However, only $n_i - 1$ of the resulting equations would be linearly independent.

Equation 3.4, it is an approximation, and our goal is to see how closely we can fit it to the data.

A simple approach is to choose β so that the values of $\mathbf{z}_{ij} \bullet \beta$ are as close to 0 as possible. That is, we can try to minimize the sum of their squares, $\sum_{i,j}(\mathbf{z}_{ij} \bullet \beta)^2$. Of course, this sum can be trivially minimized to 0 by setting $\beta = 0$. But, as described above, the magnitude of β is meaningless, and only its direction is important. So, without loss of generality, we minimize the sum of squares subject to the constraint that the magnitude of β is 1. To do this, we use Lagrange multipliers. That is, we minimize the following function:

$$F(\beta, \lambda) \;=\; \sum_{i,j}(\mathbf{z}_{ij} \bullet \beta)^2 \;-\; \lambda(\|\beta\|^2 - 1) \tag{3.7}$$

Taking partial derivatives with respect to β and setting the result to 0, we get the equation

$$\sum_{ij} \mathbf{z}_{ij}(\mathbf{z}_{ij} \bullet \beta) \;=\; \lambda\beta \tag{3.8}$$

Taking the inner product of both sides with β gives $\sum_{ij}(\mathbf{z}_{ij} \bullet \beta)^2 = \lambda\beta \bullet \beta = \lambda\|\beta\|^2 = \lambda$, where the last equation follows from the constraint $\|\beta\| = 1$. We have therefore derived the following two equations:

$$\sum_{ij} \mathbf{z}_{ij}\mathbf{z}_{ij}^T \,\beta \;=\; \lambda\beta \qquad\qquad \lambda \;=\; \sum_{ij}(\mathbf{z}_{ij} \bullet \beta)^2$$

In the left equation, all multiplications are matrix multiplications, with all vectors interpreted as column vectors. The left equation says that β is an eigenvector of the matrix $\sum_{ij} \mathbf{z}_{ij}\mathbf{z}_{ij}^T$, and the right equation says that its eigenvalue is just the sum of squares we want to minimize. We should therefore choose the eigenvector with the smallest eigenvalue. This provides an estimate of the parameter vector β, and completes the first phase of learning.

In the second phase, we use β to estimate the abundance of each protein, in_i. This can be done in a number of ways, such as using Equation 2.2. Another possibility is suggested by Equations 3.5. Letting $v_{ij} = \mathbf{x}_{ij} \bullet \beta$, we get $y_{ij} = in_i v_{ij}$, for j from 1 to n_i. Thus, for each value of i, we get a system of n_i linear equations involving in_i, and we can estimate the value of in_i using univariate linear regression. The values of $in_1, \cdots, in_N$ can then be estimated by carrying out N such regressions.

3.1.2 LIN2: Improving the Optimization Criterion.

The method developed above uses the equation $\|\beta\| = 1$ to constrain the magnitude of the parameter vector β. However, the choice of this particular equation was arbitrary, since any equation of the form $\|M\beta\| = 1$ also constrains the magnitude of β, for any non-singular matrix M. Above, we implicitly chose $M = I$, the identify matrix. Here, we use $M = X$, where each row of matrix X is one of the feature vectors $\mathbf{x}_{ij}^T$. We still try to minimize the sum of squares $\sum_{ij}(z_{ij} \bullet \beta)^2$, but subject to the new constraint $\|X\beta\| = 1$. Note that this constraint is equivalent to $\sum_{ij}(x_{ij} \bullet \beta)^2 = 1$.

The advantage of this constraint is that it leads to estimates of ionization efficiency and protein abundance that do not depend on arbitrary choices in data representation. For example, the estimates do not depend on whether we represent peptide mass in milligrams or micrograms. More generally, the estimates are invariant under any one-to-one linear transformation of the feature vectors. That is, suppose we let $\mathbf{x}'_{ij} = A\mathbf{x}_{ij}$, where A is a non-singular matrix. Then our estimates of protein abundance and ionization efficiency will not depend on whether we use $\mathbf{x}_{ij}$ or $\mathbf{x}'_{ij}$ to represent the peptides. This is how things should be. After all, the vectors $\mathbf{x}_{ij}$ and $\mathbf{x}'_{ij}$ contain exactly the same information, since each can be derived from the other. In fact, any change in the estimates would imply that the estimates are somewhat arbitrary, since they depend on arbitrary choices in data representation. In other words, if using $\mathbf{x}_{ij}$ leads to one set of estimates, while using $\mathbf{x}'_{ij}$ leads to another, then which estimates are correct? The estimation method developed here does not have this problem.

Before developing the method, we first show that its estimates will indeed be invariant under linear transformation. By Equation 3.4, it is enough to show that the values of $\mathbf{x}_{ij} \bullet \beta$ are invariant. To do this, first note that if $\mathbf{x}'_{ij} = A\mathbf{x}_{ij}$ and $\beta' = A^{-T}\beta$, then $\mathbf{x}'_{ij} \bullet \beta' = \mathbf{x}_{ij} \bullet \beta$. Thus, it is enough to show that if each feature vector, $\mathbf{x}_{ij}$, is replaced by $A\mathbf{x}_{ij}$, then the estimated parameter vector changes from $\hat{\beta}$ to $A^{-T}\hat{\beta}$. To show that the method behaves this way, first recall from Equation 3.6 that $\mathbf{z}_{ij} = y_{ij}\mathbf{x}_{i,j-1} - y_{i,j-1}\mathbf{x}_{ij}$. Also note that $y_{ij}\mathbf{x}'_{i,j-1} - y_{i,j-1}\mathbf{x}'_{ij} = A^{-T}(y_{ij}\mathbf{x}_{i,j-1} - y_{i,j-1}\mathbf{x}_{ij})$. In other words, $\mathbf{z}'_{ij} = A^{-T}\mathbf{z}_{ij}$. Thus, if $\beta' = A^{-T}\beta$, and $\mathbf{x}'_{ij} = A\mathbf{x}_{ij}$ for all i and j, then $\mathbf{z}'_{ij} \bullet \beta' = \mathbf{z}_{ij} \bullet \beta$ for all i and j. Hence,

$$\begin{array}{rl} \beta = \hat{\beta} & \text{minimizes } \sum_{ij}(\mathbf{z}_{ij} \bullet \beta)^2 \\ & \text{subject to } \sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2 = 1 \end{array}$$

if and only if

$$\begin{array}{rl} \beta = A^{-T}\hat{\beta} & \text{minimizes } \sum_{ij}(\mathbf{z}'_{ij} \bullet \beta)^2 \\ & \text{subject to } \sum_{ij}(\mathbf{x}'_{ij} \bullet \beta)^2 = 1 \end{array}$$

since none of the dot products changes value. Thus, all our estimates of ionization efficiency and protein abundance will remain unchanged under a one-to-one linear transformation of the feature vectors. In fact, this is true for any method that solves this constrained minimization problem.

As in Section 3.1.1, the method developed here is based on Lagrange multipliers. That is, we minimize the following function, which is the new version of function (3.7):

$$F(\beta, \lambda) \;=\; \sum_{i,j} (\mathbf{z}_{ij} \bullet \beta)^2 \;-\; \lambda(\sum_{ij} (\mathbf{x}_{ij} \bullet \beta)^2 - 1)$$

Taking partial derivatives with respect to β and setting the result to 0, we now get

$$\sum_{ij} \mathbf{z}_{ij}(\mathbf{z}_{ij} \bullet \beta) \;=\; \lambda \sum_{ij} \mathbf{x}_{ij}(\mathbf{x}_{ij} \bullet \beta) \tag{3.9}$$

Taking the inner product of both sides with β, we get $\sum_{ij}(\mathbf{z}_{ij} \bullet \beta)^2 = \lambda \sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2 = \lambda$, where the last equation comes from the constraint $\sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2 = 1$. Thus, expressing Equation (3.9) in matrix notation, we have the following two equations:

$$\sum_{ij} \mathbf{z}_{ij}\mathbf{z}_{ij}^T \beta = \lambda \sum_{ij} \mathbf{x}_{ij}\mathbf{x}_{ij}^T \beta \qquad \lambda = \sum_{ij} (\mathbf{z}_{ij} \bullet \beta)^2$$

The left equation is a *generalized* eigenvector equation, that is, an equation of the form $A\beta = \lambda B\beta$, where A and B are square matrices. In this case, since A and B are symmetric, the eigenvectors and eigenvalues are guaranteed to be real. There is therefore a smallest eigenvalue. As before, the right equation says that the eigenvalue is the sum of squares we are trying to minimize. We therefore choose the generalized eigenvector with the smallest eigenvalue. Estimating β in this way (the first phase of learning), we then use it to estimate values for the in_i (the second phase of learning), as in LIN1.

3.1.3 LIN3: Simultaneous Learning.

Here, we outline a single-phase approach to learning, one that estimates values for β and all the in_i simultaneously. Since the estimate for one parameter takes into account the estimates for all the other parameters, we hope to get a better overall fit to the data.

In order to do this, we first transform Equation 3.4. Observe that the right-hand side of this equation contains a product of two unknowns, in_i and β. To eliminate this non-linear term, we divide both sides by in_i, to obtain a model that is linear in all the unknowns: $y_{ij}\alpha_i \;=\; \mathbf{x}_{ij} \bullet \beta$, where $\alpha_i = 1/in_i$. Since both sides of this equation contain unknown parameters, we cannot simply minimize the error between them, since by setting $\alpha_i = 0$ and $\beta = 0$ the error is trivially minimized to 0, which is clearly incorrect. Instead, we minimize the *angle* between two vectors, the vector of values on the right-hand side of the equation, and the vector of values on the left-hand side. Moreover, we do this by maximizing the cosine of the angle between them. In general, the cosine of the angle between two vectors, $\mathbf{v}$ and $\mathbf{w}$, is given by the formula $\mathbf{v} \bullet \mathbf{w}/\|\mathbf{v}\| \cdot \|\mathbf{w}\|$. We must therefore maximize the following expression:

$$\frac{\sum_{ij}(y_{ij}\alpha_i) \cdot (\mathbf{x}_{ij} \bullet \beta)}{\sqrt{\sum_{ij}(y_{ij}\alpha_i)^2}\sqrt{\sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2}} \tag{3.10}$$

Note that this measure of error is insensitive to the absolute magnitude of the parameters, α_i and β, which can all be scaled up or down by the same amount without affecting the angle between the two vectors.

In [3], we develop a method to efficiently carry out this maximization. It is based on Theorem 1 below, which is also proved in [3]. In this theorem, Y_i is the column vector $(y_{i1}, y_{i2}, ..., y_{ik})^T$, and $\mathbf{X}_i$ is the matrix $(\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{in_i})^T$, where each $\mathbf{x}_{ij}$ is viewed as a column vector. They represent, respectively, the spectral counts and peptide property vectors for protein i.

Theorem 1: *Expression (3.10) is maximized when the parameter vector β is a solution of the following generalized eigenvector equation:*

$$\rho^2 \sum_{ij} \mathbf{X}_i^T \mathbf{X}_i \; \beta \;=\; [\sum_{i} \; \mathbf{X}_i^T Y_i Y_i^T \mathbf{X}_i / \|Y_i\|^2] \; \beta$$

Moreover, it is the eigenvector corresponding to the largest eigenvalue, ρ^2. In addition,

$$\alpha_i \;=\; \mathbf{Y}_i^T \mathbf{X}_i^T \beta / \rho \|\mathbf{Y}_i\|^2$$

Finally, ρ is the maximum value of Expression (3.10). (So the minimum angle, θ, between the two vectors is given by $\rho = \cos(\theta)$).

3.2 Inverse Models.

All of the methods developed above for fitting the linear model to data are easily adapted to fitting the inverse model. Recall that in the inverse model, $y_{ij} \;=\; in_i/(\mathbf{x} \bullet \beta)$. By inverting both sides of the equation, we get, $y'_{ij} \;=\; in'_{ij} \cdot (\mathbf{x} \bullet \beta)$ where $y'_{ij} = 1/y_{ij}$ and $in'_{ij} \;=\; 1/in_{ij}$. This is precisely the linear model. In addition, y'_{ij} is known and in'_{ij} is unknown, which is precisely the condition for applying the linear fitting methods developed above. When the linear methods LIN1, LIN2 and LIN3 are adapted in this way to the inverse model, we refer to them as INV1, INV2 and INV3.

3.3 Exponential Models.

In this section, we develop two methods for fitting the exponential model to the data. Recall that in the exponential model, $y_{ij} = in_i \cdot e^{\mathbf{x}_{ij} \bullet \beta}$. Taking logs of both sides converts this to a linear model:

$$\log y_{ij} = \log in_i + \mathbf{x}_{ij} \bullet \beta \tag{3.11}$$

The two methods described below differ in their treatment of the latent variables in_i. As with the linear methods developed above, one method is a two-phase learner, and the other is single-phase. Unlike the linear methods, the workhorse of these methods is linear regression, not eigenvector decomposition.

3.3.1 EXP1: Two-Phase Learning. As in the development of LIN1, we subtract successive instances Equation 3.11 in order to eliminate the unknown value in_i. This gives equations of the form $\log y_{i,j-1} - \log y_{i,j} = (\mathbf{x}_{i,j-1} - \mathbf{x}_{i,j}) \bullet \beta$ for j from 2 to n_i.[3] Equivalently, $Y_{ij} = X_{ij} \bullet \beta$, where $Y_{ij} = \log y_{i,j-1} - \log y_{i,j}$ and $X_{ij} = \mathbf{x}_{i,j-1} - \mathbf{x}_{i,j}$. β can thus be estimated from the new variables X_{ij} and Y_{ij} using linear regression. This is the first phase of learning. Once β is known, its value can be used to help estimate values for the remaining unknowns, $in_1, ..., in_N$. This is the second phase of learning. Equation 3.11 suggests a natural approach. Letting $v_{ij} = \log y_{ij} - \mathbf{x}_{ij} \bullet \beta$, we get $v_{ij} = \log in_i$, for j from 1 to n_i. Like Equation 3.11, this equation is only approximate. The value of $\log in_i$ that minimizes the mean squared error is just the mean of the v_{ij}. We therefore use $\log in_i = (v_{i1} + \cdots + v_{in_i})/n_i$.

Although it is straightforward, a potential problem with this method is in the error that it minimizes. We would like to minimize (the sum of squares of) the errors $\varepsilon_{ij} = \log y_{ij} - \log in_i - \mathbf{x}_{ij} \bullet \beta$. Instead, the method minimizes (the sum of squares of) $\varepsilon_{ij} - \varepsilon_{i,j-1}$, which is not the same thing. The method developed next solves this problem.

3.3.2 EXP2: Simultaneous Learning. In order to minimize the errors ε_{ij} directly, we treat the unknown values $\log in_i$ in Equation (3.11) as regression parameters. To this end, we define β' to be the extended parameter vector $(\beta_1, ..., \beta_p, b_1, ..., b_N)^T$. Here, b_i is a parameter representing $\log in_i$, and $(\beta_1, ..., \beta_p)^T$ is the original parameter vector, β, used above. Values for all the parameters in β' will be estimated simultaneously in a single act of linear regression. To do this, we define a number of matrices. First, we define $\mathbf{O}^T$ to be the following sparse matrix:

$$\left(\begin{array}{ccc|ccc|c|ccc}
1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
0 & \cdots & 0 & 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \ddots & & \vdots & \\
0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 1 & \cdots & 1
\end{array}\right)$$

where the i^{th} vertical block has n_i columns. In this matrix, each row corresponds to a protein, and each column to a peptide. A 1 in the matrix means that the peptide belongs to the protein. In addition, we define $\mathbf{X}$ to be the matrix of predictors $(\mathbf{x}_{11}, ..., \mathbf{x}_{1,n_1}, \mathbf{x}_{21}, ..., \mathbf{x}_{2n_2}, ..., \mathbf{x}_{N1}, ..., \mathbf{x}_{Nn_N})^T$, and Y to be the column vector of responses $(Y_{11}, ..., Y_{1,n_1}, Y_{21}, ..., Y_{2n_2}, ..., Y_{N1}, ..., Y_{Nn_N})^T$, where $Y_{ij} = \log y_{ij}$. With this notation, it is not hard to see that the set of linear Equations (3.11) becomes the single matrix equation $Y = \mathbf{V}\beta'$ where $\mathbf{V}$ is the extended predictor matrix $(\mathbf{X}, \mathbf{O})$. Thus, the problem again reduces to linear regression, though with a far larger set of parameters. The solution is

$$\beta' = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T Y \tag{3.12}$$

Observe that $\mathbf{X}$ is a $M \times p$ matrix, where $M = \sum_i n_i$ is the total number of peptides, and p is the number of features in the vectors $\mathbf{x}_{ij}$. Likewise, $\mathbf{O}$ is a $M \times N$ matrix, where N is the number of proteins. Consequently, $\mathbf{V}$ has dimensions $M \times (p + N)$, and $\mathbf{V}^T\mathbf{V}$ has dimensions $(p + N) \times (p + N)$. Although p is relatively small in our datasets, N is large, so the matrix $\mathbf{V}^T\mathbf{V}$ is extremely large, and inverting it is computationally very expensive.

Fortunately, because of the sparse and regular structure of $\mathbf{O}$, we can develop a more efficient procedure. The first step is to partition the matrix $(\mathbf{V}^T\mathbf{V})^{-1}$ into blocks as follows:

$$(\mathbf{V}^T\mathbf{V})^{-1} = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$$

Here, the submatrices have the following dimensions: A is $p \times p$, B is $p \times N$, C is $N \times p$, and D is $N \times N$. The submatrix A is therefore small, while D is extremely large. The next step is to solve for A, B, C and D, as follows:

$$\begin{aligned}
I &= (\mathbf{V}^T\mathbf{V})(\mathbf{V}^T\mathbf{V})^{-1} \\
&= \left(\begin{array}{c} \mathbf{X}^T \\ \hline \mathbf{O}^T \end{array}\right) (\mathbf{X}, \mathbf{O}) \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \\
&= \left(\begin{array}{c|c} \mathbf{X}^T\mathbf{X}A + \mathbf{X}^T\mathbf{O}C & \mathbf{X}^T\mathbf{X}B + \mathbf{X}^T\mathbf{O}D \\ \hline \mathbf{O}^T\mathbf{X}A + \mathbf{O}^T\mathbf{O}C & \mathbf{O}^T\mathbf{X}B + \mathbf{O}^T\mathbf{O}D \end{array}\right)
\end{aligned}$$

[3] Of course, we could generate many more equations by subtracting different pairs of equations, instead of just the successive ones. However, as with LIN1, only $n_i - 1$ of the resulting equations would be linearly independent.

where I is the identity matrix. We can view this as four matrix equations in four unknowns, A, B, C, D. Solving, we get

$$\begin{aligned} A &= (\mathbf{X}^T\mathbf{X} - \hat{\mathbf{X}}^T P^{-1}\hat{\mathbf{X}})^{-1} \\ B &= -A\tilde{\mathbf{X}}^T \\ C &= B^T \\ D &= P^{-1} - \tilde{\mathbf{X}}A\tilde{\mathbf{X}}^T \end{aligned}$$

where $\hat{\mathbf{X}} = \mathbf{O}^T\mathbf{X}$, $P = \mathbf{O}^T\mathbf{O}$, and $\tilde{\mathbf{X}} = P^{-1}\hat{\mathbf{X}}$. The first point to notice here is that A is a small matrix, of size $p \times p$, so even though it requires a matrix inversion, it is not costly. The second point is that the matrices P and $\mathbf{O}$ are both exremely large, of size $N \times N$ and $M \times N$, respectively. However, neither one needs to be materialized. To see this, note that the rows of $\mathbf{O}^T$ are orthogonal, and P is therefore diagonal. In fact, the i^{th} diagonal element of P is simply n_i, the number of peptides in protein i. The i^{th} diagonal element of P^{-1} is therefore $1/n_i$. Multiplication by P or P^{-1} is therefore a simple operation. Likewise for left multiplication by $\mathbf{O}^T$. For instance, the i^{th} row of $\hat{\mathbf{X}} = \mathbf{O}^T\mathbf{X}$ is simply $\sum_j \mathbf{x}_{ij}^T$. In this way, neither P, P^{-1} nor $\mathbf{O}$ needs to be materialized.

Neither does the extremely large matrix D. In fact, the largest matrix that needs to be materialized is the data matrix, $\mathbf{X}$. To see this, first rewrite Equation 3.12 as

$$\begin{aligned} \left(\begin{array}{c} \beta \\ \hline \mathbf{b} \end{array}\right) &= \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \left(\begin{array}{c} \mathbf{X}^T \\ \hline \mathbf{O}^T \end{array}\right) Y \\ &= \left(\begin{array}{c} A\mathbf{X}^TY + B\mathbf{O}^TY \\ \hline C\mathbf{X}^TY + D\mathbf{O}^TY \end{array}\right) \end{aligned}$$

where we have decomposed the extended parameter vector, β', into the original parameter vector, $\beta = (\beta_1, ..., \beta_p)^T$, and a vector of the new parameters, $b = (b_1, ..., b_N)^T$. Expanding the definition of D, we get two equations:

$$\beta = A\mathbf{X}^TY + B\hat{Y} \tag{3.13}$$

and

$$\begin{aligned} b &= C\mathbf{X}^TY + (P^{-1} - \tilde{\mathbf{X}}A\tilde{\mathbf{X}}^T)\hat{Y} \\ &= C\mathbf{X}^TY + \tilde{Y} - \tilde{\mathbf{X}}A\tilde{\mathbf{X}}^T\hat{Y} \end{aligned} \tag{3.14}$$

where $\hat{Y} = \mathbf{O}^TY$ and $\tilde{Y} = P^{-1}\hat{Y}$. Note that $\hat{Y}$ is a column vector whose i^{th} component is $\hat{Y}_i = \sum_j Y_{ij}$. Likewise, $\tilde{Y}$ is a column vector whose i^{th} component is $\tilde{Y}_i = \hat{Y}_i/n_i$. Thus, it is still unneccessary to materialize the very large matrices P, P^{-1} and $\mathbf{O}$.

Finally, observe that the largest matrix in Equations 3.13 and 3.14 is the data matrix $\mathbf{X}$, which has size $M \times p$. The matrices $\hat{\mathbf{X}}$ and $\tilde{\mathbf{X}}$ both have size $N \times p$, the matrix A has size $p \times p$, and the matrices B and C have sizes $p \times N$ and $N \times p$, respectively. All of these are smaller than $\mathbf{X}$ since $p < N < M$. Moreover, if all the matrix multiplications are carried out from right to left, then all the intermediate results are column vectors, not large matrices. In this way, the parameter vectors β and b can be estimated without having to materialize or manipulate any matrices larger than $\mathbf{X}$. In fact, for fixed p, the total cost of estimating the parameters is linear in M, the number of peptides; *i.e.*, the total cost is linear in the number of data points.

4 Experimental Evaluation.

We conducted an extensive evaluation of the models and methods developed above using both real and simulated datasets [12]. This section presents a representative sample of results based on the real data. We describe the datasets, the experimental design, and the evaluation methods. The main difficulty in evaluating the methods is the distribution of the real data. As shown below, it ranges over several orders of magnitude and is highly skewed, with most data concentrated at very low values. We deal with these difficulties in two ways. First, we use the Spearman rank correlation coefficient to measure the goodness of fit of our estimates to the observed values [2]. Unlike the more common Pearson correlation coefficient, which measures *linear* correlation, Spearman's coefficient measures *monotone* correlation and is insensitive to extreme data values. In addition, we use log-log plots of observed v.s. estimated values to provide an informative visualization of the fit.

4.1 Real-World Datasets. The experimental results in this paper are based on tables of real-world data similar to Table 1. They consist of three datasets derived from tissue samples taken from Mouse and were provided by the Emili Laboratory at the Banting and Best Department of Medical Research at the University of Toronto. We refer to these datasets as **Mouse Brain Data**, **Mouse Heart Data**, and **Mouse Kidney Data**. As noted at the end of Section 2.2, proteins that produce only one observable peptide are not useful for fitting models to data. We identified these proteins and removed them from the data. After removal, the Brain dataset contains 8,527 peptides and 1,664 proteins, the Heart dataset contains 7,660 peptides and 1,281 proteins, and the Kidney dataset contains 7,074 peptides and 1,291 proteins. For the purposes of this paper, each of these three datasets was divided randomly into two subsets, training data and testing data, in a 2:1 ratio. The results are comparable to those in [12], in which we use ten-fold cross validation.

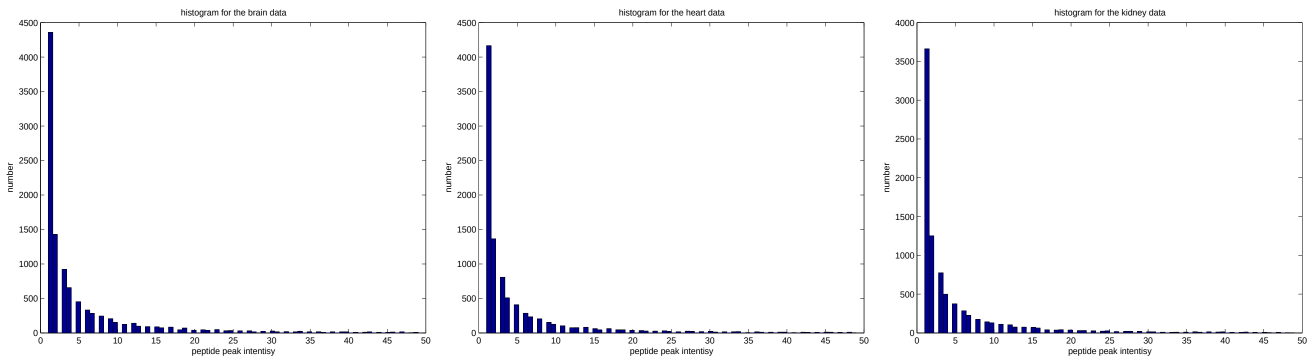

Figure 1: Histograms of peptide spectral count. The left histogram is for the Brain dataset, the middle is for Heart, and the right is for Kidney.

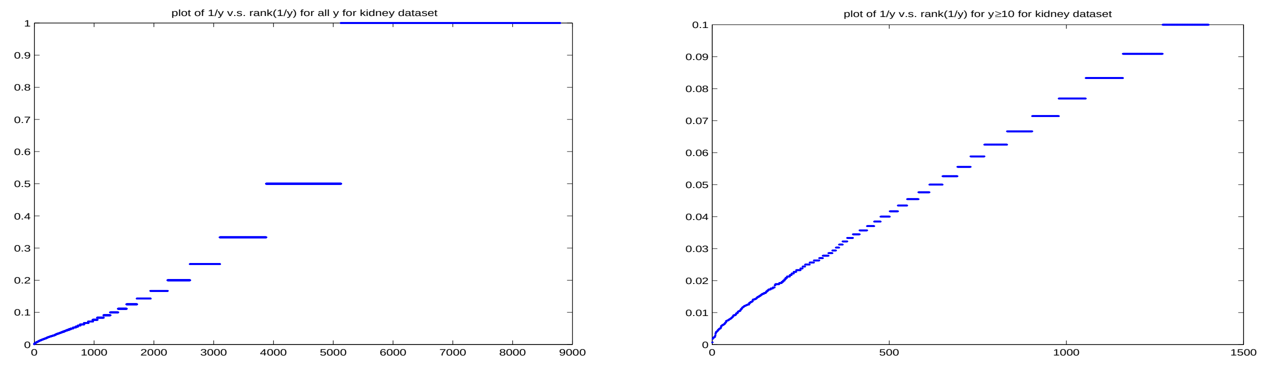

Figure 2: Probability plots of $1/y$ for the Kidney dataset, where y is spectral count. The left plot includes all observed values of y, while the right plot excludes values of y less than 10.

Histograms of spectral count for the three data sets are shown in Figure 1. These show that spectral count is far from being uniformly or normally distributed. Instead, spectral counts are heavily concentrated near the minimum value of 1, and their frequency rapidly falls off as spectral count increases. For example, although the maximum spectral count in the Kidney dataset is 1,491, the median value is only 2! That is, half the peptides have spectral counts of only 1 or 2, while the remaining peptides have spectral counts from 2 to 1,491. Thus, the data values range over several orders of magnitude, with the bulk of the data concentrated at lower values. This distribution is likely due to a number of factors, including the distribution of protein levels in the tissue samples, and the distribution of ionization efficiencies among peptides. Untangling these factors is one of the goals of this work.

A more accurate description of the distribution of spectral count is provided by the probability plots in Figure 2, which display observed spectral counts in sorted order. The two panels in Figure 2 show probability plots for the reciprocal of spectral count; that is, they are probability plots of $1/y$, where y is spectral count. The vertical axis is the value of $1/y$, and the horizontal axis is the rank of this value. The plot in the left panel includes all observed spectral counts, while the plot in the right panel excludes small spectral counts, *i.e.*, counts less than 10. The horizontal line segments that make up the plots, especially at the upper end, are due to the discrete nature of the data (*i.e.*, y is an integer). However, the trend in the right hand plot is clearly a diagonal line, from the lower left corner of the plot to the upper right corner. This indicates that the cumulative distribution function of $1/y$ is a diagonal line, which means that $1/y$ is uniformly distributed, which in turn means that y has an $O(1/y^2)$ distribution [2]. The left hand plot is the same except that it curves up slightly for large values of $1/y$, that is, for small spectral counts. Thus, while the probability density of $1/y$ is flat for $y \geq 10$, it decreases for $y < 10$. This in turn means that, while the distribution of spectral count is $O(1/y^2)$ for $y \geq 10$, it is less than this for $y < 10$. Similar results hold for all three data sets, Kidney, Brain and Heart.

4.2 Peptide Representation. To use the models and methods developed in this paper, each peptide must be represented as a vector, $\mathbf{x}$. This paper evaluates several ways of doing this, using vectors with 22, 42, 62 and 232 components, respectively. The 22-component vector represents the amino-acid composition of a peptide. The vector has 20 components, $(x_1, ..., x_{20})$, one for each amino acid, where the value of x_i is the number of occurrences of a particular amino acid in the peptide. In addition, the vector has a 21^{st} component whose value is always 1, to represent a bias term, as is common in linear regression [7]. The vector also has a 22^{nd} component whose value is the charge of the peptide ion, as given in Table 1. The 232-component vector uses these same 22 components plus an additional 210 quadratic components of the form $x_i x_j$ computed from x_i and x_j for $1 \leq i, j \leq 20$.

These vectors capture the charge and amino-acid composition of a peptide ion, but they do not contain any sequential information. The 42-component vector ameliorates this situation somewhat. It divides a peptide sequence into two subsequences, by cutting it in half, and represents the amino-acid composition of each half. This requires 40 vector components. Again, we add a 41^{st} component to act as a bias term, and a 42^{nd} component to represent charge. The 62-component vector carries this idea one step further by dividing a peptide into three subsequences, thereby capturing more sequential information.

It is worth noting that these four representations subsume many others. For instance, natural peptide features such as mass and length are implicitly included in our vectors. This is because all of our models are based on a linear combination of peptide features, that is, on the quantity $\beta \bullet \mathbf{x} = \sum_i \beta_i x_i$, where the β_i are parameters to be learned. Thus, any peptide feature that can be expressed as a linear combination of other features is already accounted for. For instance, the mass of a peptide is $m = \sum_i m_i x_i$, where m_i is the mass of amino acid i, and x_i is the count of amino acid i in the peptide. If we were to add m as a new feature in our 22-component vector, then our models would be based on the quantity $\beta_m m + \sum_i \beta_i x_i$, where β_m and each of the β_i are parameters to be learned. However, this quantity is equal to $\beta_m \sum_i m_i x_i + \sum_i \beta_i x_i = \sum_i \beta'_i x_i$ where $\beta'_i = \beta_m m_i + \beta_i$. Thus, adding peptide mass as an explicit feature does not increase the expressive power of our models. In fact, including it would simply complicate the learning process by introducing a linear dependence that would make many matrices singular. Likewise for peptide length and any other feature that can be expressed as a linear combination of amino-acid counts.

4.3 Measuring Goodness of Fit. We used each of the four peptide representations with each of the eight fitting methods, LIN1, LIN2, LIN3, EXP1, EXP2, INV1, INV2 and INV3. We applied each combination to the training data to estimate a value for the parameter vector, β. We then used β to predict the spectral counts of all the peptides in the testing data, and then compared the predictions to the observed values by measuring the correlation between them.

To predict spectral counts, y_{ij}, we used Equation 2.1. This in turn required estimates of ionization efficiency, ie_{ij}, and protein abundance, in_i. Given the estimate for β, ionization efficiency was computed directly from Equation 2.3, depending on the model. Estimating protein abundance was only sightly more complicated, and also depended on the model. Recall that in each model, the formula $y_{ij} = in_i \cdot ie_{ij}$ is transformed so that the expression $\mathbf{X} \bullet \beta$ appears as a linear term. The transformed models are given by the following equations:

$$\begin{array}{lrcl}
\text{Linear :} & y_{ij} & = & in_i \cdot (\mathbf{x}_{ij} \bullet \beta) \\
\text{Exponential :} & \log(y_{ij}) & = & \log(in_i) + \mathbf{x}_{ij} \bullet \beta \\
\text{Inverse :} & y_{ij}^{-1} & = & in_i^{-1} \cdot (\mathbf{x}_{ij} \bullet \beta)
\end{array}$$

In each of these models, the y_{ij} and $\mathbf{x}_{ij}$ are known, and β has been estimated from the training data. Estimates of in_i for proteins in the testing data were then computed as follows. In the linear model, for each value of i, we have a set of linear equations, from which in_i was estimated by univariate linear regression. Likewise, in the inverse model, in_i^{-1} was estimated by univariate linear regression, and in_i was then estimated as $1/in_i^{-1}$. Finally, in the exponential model, $\log(in_i)$ was estimated as the mean of $\log(y_{ij}) - (\mathbf{x}_{ij} \bullet \beta)$, and in_i was then estimated by taking exponentials. In what follows, we use $\hat{y}_{ij}$, $\hat{in}_i$ and $\hat{ie}_{ij}$ to denote the estimates of y_{ij}, in_i and ie_{ij}, respectively.

A common way to measure correlation is with the Pearson correlation coefficient, which measures the linear correlation between two random variables [2]. However, because of the distribution of our data—highly skewed and ranging over several orders of magnitude—the Pearson correlation coefficient can be misleading, since its value is dominated by a relatively small number of extremely large data values. In addition, because of the variety of models we use, it is not clear whether we should compute the correlation of y v.s. $\hat{y}$, or of $\log(y)$ v.s. $\log(\hat{y})$, or of $1/y$ v.s. $1/\hat{y}$. Fortunately, all these problems are solved by using the *Spearman* rank correlation coefficient [2]. This is similar to Pearson's coefficient, except that instead of correlating the data values themselves, it correlates their ranks. It is therefore insensitive to the exact values of the data, and in particular, to extremely large values. Moreover, instead of measuring linear correlation, it measures monotone correlation (*i.e.*, the tendency of one variable to increase or decrease with the other variable). In fact, the value of Spearman's coefficient is invariant under monotonic transformations of the data, so it does not matter whether we consider y or $\log(y)$ or $1/y$.

Tables 2, 3 and 4 show experimental results for every combination of the eight fitting methods, four peptide representations, and three Mouse datasets. The columns of each table represent fitting methods, and the rows represent vector representations of peptides (identified by the number of features in the vectors). Each table entry is the Spearman rank correlation coefficient of y and $\hat{y}$, that is, of observed and estimated spectral counts, as measured on the testing portion of the data. An entry of ****** means that the method could not be used because a matrix turned out to be singular.

5 Discussion.

The following observations about the fitting methods are immediately apparent from Tables 2, 3 and 4: the exponential methods, EXP1 and EXP2, almost always perform the best, though INV3 is often competitive; LIN1 and INV1 perform the worst; of the linear methods, LIN3 performs the best; of the inverse methods, INV3 performs the best; INV3 always performs better than LIN3.

Recall that the only difference between LIN1 and LIN2 is the constraints on which they are based. The superior performance of LIN2 suggests that its constraint is much more appropriate than the constraint for LIN1, as argued in Section 3.1.2. Recall also that LIN1 and LIN2 use a two-phase approach to learning, in which the parameter vector β is estimated first, after which protein abundance is estimated from β. In contrast, LIN3 is based on a single-phase approach to learning, in which parameters and protein abundance are all learned simultaneously. The better performance of LIN3 over LIN1 and LIN2 on the real datasets suggests that the single-phase approach is more appropriate, at least for tandem mass spectrometry data, as argued in Section 3.1.3. These conclusions are all corroborated by the performance of INV1, INV2 and INV3, which parallels that of LIN1, LIN2 ad LIN3, upon which they are based. That INV3 always performs better than LIN3 suggests that the inverse transformation upon which INV3 is based had its intended effect, as described in Section 2.2.

The superior performance of the exponential models suggests that the logarithmic transformation on which they are based is the most appropriate. This is perhaps not surprising. Logarithms remove the problem of data ranging over several orders of magnitude, while simultaneously compressing a sparse set of extremely large values into a denser set of smaller values. This effectively deals with two of the main problems in our datasets. In addition, it is reasonable to suppose that in estimating spectral counts, as with many other real-world values, it is the percentage error, not the absolute error that matters. Thus, the appropriate noise model would appear to be multiplicative, not additive. Logarithms conveniently convert multiplicative noise to additive noise, thus making linear regression a natural method to apply to the transformed data, which is exactly what the exponential methods do. In addition, unlike the inverse transformation, logarithms map large values to large values, and small values to small values. In contrast, since the inverse transformation maps large values to small values, a small error in the estimated value of $1/y$ can result in a large error in y when y is large. This may more than compensate for the ability of the inverse model to eliminate all skew from the data.

Finally, we should point out that of the four different vector representations, the 22-feature vector generally works the best, though the 42-feature and 62-feature vectors are often competitive. The generally poor performance of the 232-feature vector suggests that it is overfitting, since it includes all the features of the 22-feature vector, which performs best. However, theses comments apply only when we consider all of the fitting methods. When we consider only EXP1, the best performing method, its performance seems independent of which vector representation is used.

5.1 Visualizing the Goodness of Fit.

To help visualize the goodness of the fit provided by our methods, Figure 3 provides log-log plots of observed and estimated values for a training dataset. Due to space limitations, we show plots only for the Kidney dataset, and only for EXP1, INV3 and LIN3, the best methods for each of our three models.[4] For each method, we provide two figures, a plot of y v.s. $\hat{y}$, and a plot of ie v.s. $\widehat{ie}$. Here, ie and $\widehat{ie}$ are derived from y and $\hat{y}$, respectively, by dividing by $\widehat{in}$, the estimate of protein abundance. ie and $\widehat{ie}$ thus provide two different measures of the ionization efficiency of a peptide. Note that $\widehat{ie}$ is just the estimate derived from Equation 2.3.

The first thing to notice is the strong horizontal lines in the plots of y v.s. $\hat{y}$. These are due to the discrete nature of the observed values, y, most of which take on small, positive integer values. The plots have no such vertical lines because the estimates, $\hat{y}$, are real

[4]The INV3 and LIN3 methods produce a very small number of negative estimates, which we discard before taking logs.

Table 2: Spearman rank correlation coefficients on the testing portion of the Mouse Brain data

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2
22	0.2080	0.4342	0.4259	0.1582	0.4755	0.4755	0.4986	0.5051
42	0.2430	0.2645	0.4245	0.0065	0.4848	0.5588	0.4981	0.5049
62	0.2146	0.2365	0.4265	0.0346	0.4846	0.4840	0.5033	******
232	0.2103	0.2058	0.4218	0.0370	0.0594	0.4785	0.5141	******

Table 3: Spearman rank correlation coefficients on the testing portion of the Mouse Heart data

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2
22	0.2039	0.4965	0.4965	0.2386	0.5635	0.5635	0.5669	0.5699
42	0.1721	0.1844	0.4243	0.0106	0.5611	0.5556	0.5658	0.5656
62	0.1355	0.1691	0.4974	0.0621	0.0667	0.5605	0.5867	******
232	0.1477	0.1399	0.3285	0.0048	0.0307	0.4713	0.5634	******

valued. The second thing to notice is that the plots of ie v.s. $\hat{ie}$ have no strong lines. This is because ionization efficiency is inherently real-valued. Finally, notice that the plot of ie v.s. $\hat{ie}$ for the EXP1 method is by far the most Gaussian-looking of all the plots. In contrast, the plots for LIN3 and INV3 show evidence of residual structure that the methods were unable to fit.

6 Conclusions.

We developed and evaluated eight methods for estimating protein abundance from high-throughput Tandem Mass Spectrometry (MS/MS) data. Each method is based on a simple, generative model of MS/MS data. We evaluated the effectiveness of the methods on MS/MS data derived from tissue samples of Mouse. The evaluations included three different tissue types, four different vector representations of peptides, and two different evaluation measures (one based on correlation coefficients and one based on data visualization). Of the eight methods we developed, the two exponential methods, EXP1 and EXP2, performed the best. This in turn suggests that, of our three models of MS/MS data, the exponential model is the best.

Our results suggest several directions for future research to increase the accuracy of the estimations:

- More-sophisticated models of ionization efficiency could be developed, such as models based on neural nets, kernel methods or regression trees. However, the learning methods will need to deal with the unknown amounts of protein, perhaps by using an EM-like algorithm.

- Although we tested several representations of peptides in this paper, it is not yet clear whether the solution lies in more sophisticated data-mining methods or in better peptide representations. Representations that capture more sequential information (perhaps involving string kernels or HMMs) may be important.

- The effect of errors in peptide sequence could be accounted for. It is reasonable to suppose that not all the peptides are sequenced correctly. In fact, it is possible that for some peptides, the sequences are completely wrong. The senstivity of a method to this kind of error could be determined through studies based on simulated datasets.

- The problem of a fit being dominated by a few extremely large data values might be ameliorated by a weighting scheme that places a large weight on small values. We have taken some initial steps along these lines in [3].

- More sophisticated models of spectral counts could be developed. For instance, the spectral count of a peptide depends on what other peptides are in the mass spectrometer at the same time. The first step in accounting for these dependencies is estimating what peptides enter the mass spectrometer at about the same time, perhaps along the lines of [8]. In addition, we have recently acquired more detailed MS/MS data that allows us to determine that certain peptides did *not* enter the mass spectrometer at the same time.

Table 4: Spearman rank correlation coefficients on the testing portion of the Mouse Kidney data

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2
22	0.1655	0.4599	0.4609	0.2299	0.5250	0.5208	0.5207	0.5220
42	0.2482	0.2339	0.4662	0.0584	0.5027	0.4991	0.5089	0.5188
62	0.2024	0.2080	0.4690	0.0421	0.5033	0.5065	0.5151	******
232	0.1818	0.2357	0.3992	0.0219	0.0124	0.4953	0.5105	******

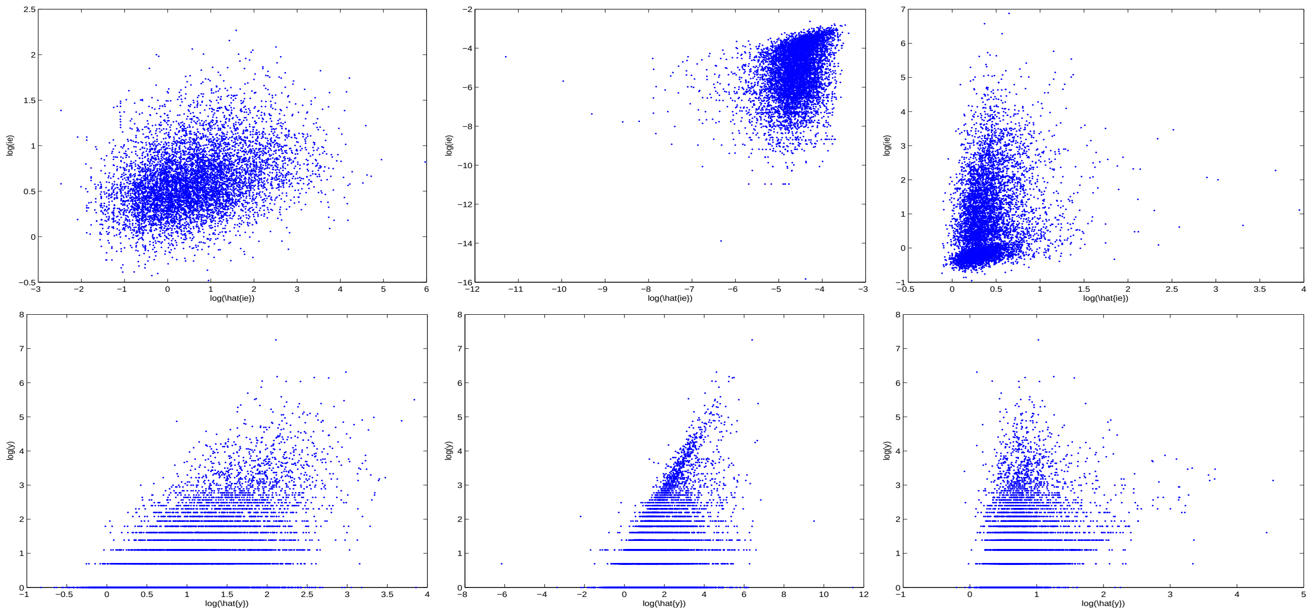

Figure 3: Log-log plots of y v.s. $\hat{y}$ (bottom row) and ie v.s. $\hat{ie}$ (top row) on the Kidney dataset. The two left panels are the results of the EXP1 method, the two middle panels are the results of the LIN3 method, and the two right panels are the results of the INV3 method.

- The experimental methodology used in this paper evaluates a method in terms of its ability to predict the spectral counts of peptides, based on peptide properties and predicted protein abundance. The ability to accurately predict spectral counts in this way would be strong evidence that a method is correct, and would suggest that protein abundance was accurately predicted. However, conclusive proof would require a more direct comparison against some known protein amounts (*e.g.*, as measured by the more laboratory-intensive isotope marker experiments [6, 13]).

Acknowledgments.
The authors would like to thank Dr. Andrew Emili for his comments and discussions and for generous access to his data. We would also like to thank Clement Chung for generating and preprocessing the data. This work was supported in part by a grant from the Natural Sciences and Engineering Researh Council of Canada (NSERC).

References

[1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 2003.

[2] P. Bickel and K. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics.* Holden-Day, Inc, 1977.

[3] A.J. Bonner and Han Liu. Canonical correlation, an approximation, and the prediction of protein abundance. In *Proceedings of the Eighth Workshop on Mining Scientific and Engineering Datasets (MSD'05)*, pages 29–38, Newport Beach, CA, April 2005. Workshop affiliated with the SIAM International Conference on Data Mining (SDM 2005).

[4] Joshua E Elias, Francis D Gibbons, Oliver D King, Frederick P Roth, and Steven P Gygi. Intensity-based protein identification by machine learning from a library of tandem mass spectra. *Nature Biotechnology*, 22(2):214–219, 2004.

[5] S.P. Gygi and R. Aebersold. Mass spectrometry and proteomics. *Current Opinion in Chemical Biology*, 4:489–494, 2000.

[6] S.P. Gygi, B. Rist, S.A. Gerber, F. Turecek, M.H. Gelb, and R. Aebersold. Quantitative analysis of complex protein mixtures using isotope-coded affinity tags. *Nature Biotechnology*, 17(10):994–999, 1999.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2001.

[8] Petritis K, Kangas LJ, Ferguson PL, and et al. Use of artificial neural networks for the accurate prediction of peptide liquid chromatography elution times in proteome analyses. *Analytical Chemistry*, 75(5):1039–1048, 2003.

[9] T. Kislinger, K. Ramin, D. Radulovic, and et al. Prism, a generic large scale proteomics investigation strategy for mammals. *Molecular & Cellular Proteomics*, 2(1), 2003.

[10] D.C. Liebler. *Introduction to Proteomics: Tools for the New Biology.* Humana Press, NJ, 2002.

[11] H. Liu, R.G. Sadygov, and J.R. Yates. A model for random sampling and estimation of relative protein abundance in shotgun proteomics. *Analytical Chemistry*, 76:4193–4201, 2004.

[12] Han Liu. Development and evaluation of methods for predicting protein levels from tandem mass spectrometry data. Master's thesis, Department of Computer Science, University of Toronto, Toronto, ON, Canada, 2005.

[13] S. Ong, B. Blagoev, I. Kratchmarovat, D.B. Kristensen, H. Steen, A. Pandey, and M. Mann. Stable isotope labelling by amino acid in cell culture, silac, as a simple and accurate approach to expression proteomics. *Molecular & Cellular Proteomics*, 1(5), 2002.

[14] G. Siuzdak. *The Expanding Role of Mass Spectrometry in Biotechnology.* Mcc Press, 2003.