Development and Evaluation of Methods for Predicting Protein Levels and Peak Intensities from Tandem Mass Spectrometry Data

Anthony J. Bonner   Han Liu[*]
Department of Computer Science, University of Toronto
Email: hanliu@cs.toronto.edu

July 20, 2004

ABSTRACT

Tandem mass spectrometry (MS/MS) of peptides is a central technology for proteomics, enabling the identification of thousands of proteins and peptides from a complex mixture. With the increasing acquisition rate of tandem mass spectrometers, it has become possible to use data-mining techniques to attempt to solve important biological problems using MS/MS data. These problems include (*i*) estimating the levels of the thousands of proteins in a tissue sample, (*ii*) predicting the intensity of the peaks in a mass spectrum, and (*iii*) explaining why different peptides from the same protein have different peak intensities. In this paper, we develop and evaluate several simple data-mining techniques for tackling these biological problems. The main data-mining problem is to untangle the various factors that affect the intensity of a peak in a mass spectrum. To this end, we develop three statistical models of MS/MS data: linear, exponentiated linear, and inverse linear. For each model, we develop a family of methods for fitting the model to the data. We test each method on both simulated and real-world data, where the real data consists of three datasets generated by MS/MS experiments performed on various tissue samples taken from Mouse. Because of the highly skewed distribution of the data, which also ranges over several orders of magnitude, we measure the fit of each model to the data using Spearman's rank correlation coefficient, instead of the more common Pearson correlation coefficient. Finally, we compare the methods to a number of naive methods, and report on their performance.

keywords: Proteomics, Data Mining, Machine Learning, Peptide Tandem Mass Spectrometry.

[*]SDM 06

1 Introduction

Tandem Mass Spectrometry (MS/MS) of peptides is a central technology of proteomics, enabling the identification of thousands of peptides and proteins from a complex mixture [21, 18, 1]. In a typical experiment, thousands of proteins from a tissue sample are fragmented into tens of thousands of peptides, which are then fractionated, ionized and passed through a tandem mass spectrometer. The result is a collection of spectra, one for each protein, where each peak in a spectrum represents a single peptide [16, 22]. With the increasing acquisition rate of tandem mass spectrometers, there is an increasing potential to solve important biological problems by applying data-mining and machine-learning techniques to MS/MS data [10, 13]. These problems include (*i*) estimating the levels of the thousands of proteins in a tissue sample, (*ii*) predicting the intensity of the peaks in a mass spectrum [10], and (*iii*) explaining why different peptides from the same protein have different peak intensities [13].

In this paper, we develop and test a number of data-mining methods for tackling these problems, including methods based on linear and exponential models with various optimization criteria. These models all focus on the problem of estimating the intensities of the peaks in the mass spectrum of a protein. One important factor is clearly the amount of protein input to the mass spectrometer (since more input implies more output). However, other factors are important as well, such as the efficiency with which various peptides are produced, fractionated and ionized. Our goal is to determine how all these other factors are influenced by a peptide's amino-acid sequence. Once this is understood, it may be possible to predict the exact MS/MS spectrum of a protein, including the intensities of all its peaks. More importantly, it may also be possible to solve the inverse problem: given the MS/MS spectrum of a protein, estimate the amount of protein that was input to the mass spectrometer. This is a fundamental problem whose solution would enable biologists to determine the levels of the thousands of proteins in a tissue sample [24].

As a first step in addressing this problem, we obtained a set of several thousand MS/MS spectra.[1] This data is the result of MS/MS experiments conducted on protein mixtures taken from various tissue samples of Mouse [24]. These high-throughput experiments provide a large amount of data 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. Thus, it is in general unclear whether a high-intensity peak is due to the properties of the peptide or to a large amount of protein at the input. One of our challenges is to untangle these two influences. This distinguishes our work from other research in which the amount of protein is known *a priori* [10]. In effect, we must solve two problems at once: the forward problem of estimating the spectrum of a protein, and the inverse problem of estimating the amount of protein given its spectrum. To deal with this complication, we treat the amount of protein as a latent, or hidden

[1]courtesy of the Emili Laboratory at the Banting and Best Department of Medical Research at the University of Toronto.

variable, whose value must be estimated. The data-mining methods presented in this paper were developed, in part, because of the ease and simplicity with which this can be done. In addition, they lead to efficient algorithms based on well-developed operators of linear algebra (specifically, matrix inversion and eigenvector decomposition).

The paper is organized as follows. Section 2 provides biological background. Section 3 introduces our three models of MS/MS data. Section 4 outlines our methods for fitting these models to data. Section 5.1 introduces the data we use to train and test our methods. Section 6 uses the datasets to test and compare our methods. Finally, Section 7 summarizes the results and suggests possible extensions for future work.

2 Peptide Tandem Mass Spectrometry

In Tandem Mass Spectrometry, a mixture of tens of thousands of unknown proteins are taken from a tissue sample. By adding a specific amount of the enzyme trypsin, the proteins are digested and fragmented into peptides. The resulting mixture is then fractionated, ionized and sent at high speed through an electric field, where the paths of the different peptides are bent by different amounts before hitting a plate. This produces a so-called "mass spectrum" consisting of various peaks [9]. Each peak occurs at a particular location on the plate, as determined by the peptide's mass-to-charge ratio, and the intensity of the peak represents the number of peptide molecules to hit the plate at that location [21, 18, 1].

The intensity of a peak is influenced by the amount of protein in the input mixture. However, the exact mechanism determining the intensity of a spectral peak is poorly understood [10]. In an ideal experiment, there would be no loss during digestion, fractionation and ionization. So, in the MS/MS spectrum for a given protein, one would expect that each peak would contain one peptide molecule for each protein molecule in the input. Consequently, an ideal spectrum for a given protein would consist of peaks of equal intensity. However, this is not observed experimentally. Figure 1 illustrates the differences between an ideal MS/MS spectrum and an experimental one.

Numerous reports in the literature address the question of what factors affect the quality of a MS/MS experiment [16] [22]. An obvious factor influencing peak intensity is the concentration of the peptides in the sample. However, it is not the sole factor. Other factors include sample preparation methods, the pH and composition of the solution containing the protein mixture [6] [22] [8] [4], the characteristics of the MS/MS apparatus [15] [20] [19], and the characteristics of the tissue sample being analyzed [10]. These factors and others all affect the intensities of peaks in a MS/MS spectrum. Unfortunately, no model exists for accurately predicting these peak intensities. Explaining why some peptides produce high-intensity peaks and some produce low-intensity peaks is a first step towards developing such a model, and is the goal of this paper.

To this end, the Emili Laboratory at the Banting and Best Department of Medical

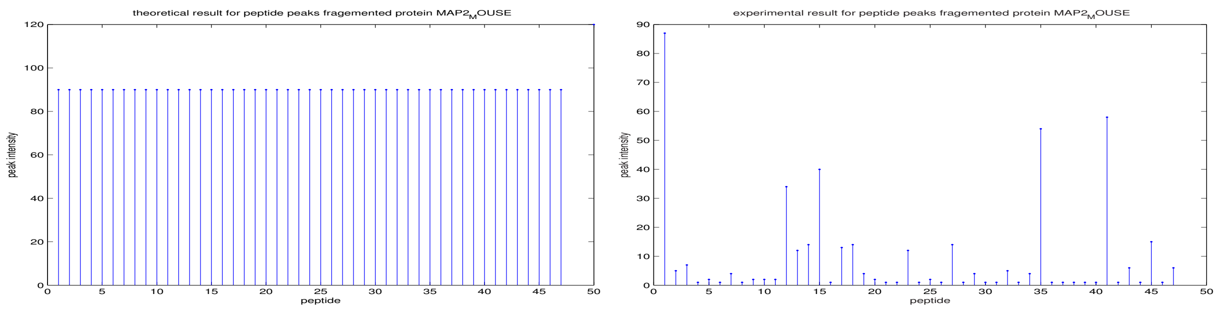


Figure 1: The left panel is the theoretical MS/MS spectrum for the 47 peptides fragmented from protein MAP2-MOUSE digested with trypsin. The right panel is an experimentally derived spectrum of the same protein.

Table 1: A fragment of the original data file

Protein ID	Peptide Sequence	Peak Count	Charge
Q91VA7	*TAAARHCCNNLVIIR*	4	2
Q91VA7	*KLDCCCLFACAVHVK*	3	2
⋮	⋮	⋮	⋮

Research at the University of Toronto has provided us with several thousand MS/MS spectra. Table 1 shows a tiny sample of this data. (Details on how this data was generated can be found in [24].) The first column in the table is the Swissprot accession number identifying a protein. The second column is the amino-acid sequence of a peptide fragmented from the protein. The third column correlates well with the intensity of the spectral peak produced by the peptide. (Its values are integers because it represents a count of peptide molecules.) The last column represents the charge of the peptide ion, which is typically 1 or 2. Notice that there may be many entries for the same protein, since a single protein can produce many peptides.

In fragmenting the proteins to produce peptides, the proteins were digested by the enzyme trypsin, which cuts c-terminal to lysine (K) or arginine (R). Trypsin may occasionally cut at other locations, and other enzymes may nick the protein; hence, we sometimes see partial tryptic peptides (a K or R at the c-terminus, or flanking the peptide at the N-terminus).

3 Modelling the Data

The data from tandem mass spectrometry experiments consists of a collection of spectra, one for each protein. Each spectrum consists of a number of peaks, where each peak is produced by a peptide, *i.e.*, a fragment of the protein. The observables are the intensity of the peak, the amino acid sequence of the peptide, and the amino acid sequence of the protein. In the high-throughput experiments we are dealing with, the amount of protein input to the mass spectrometer is unknown, and one of our goals is to estimate it.

In this paper, our working hypothesis is that for a given MS/MS experiment, each peak in a spectrum is produced independently of the others. That is, we assume that the intensity of a peak depends on only two factors: (*i*) the peptide responsible for the peak, and (*ii*) the (unknown) amount of protein input to the mass spectrometer. Although interactions between peptides do sometimes occur in mass spectrometer experiments[2], we use independence as a simplifying approximation in this initial study. More specifically, we assume that each peptide has an associated *ionization efficiency* that accounts for the relative heights of the various peaks in a spectrum. The ionization efficiency represents the probability that a given peptide will enter the mass spectrometer as an ion. Since only these peptides contribute to the intensity of a peak, we expect that peak intensity will be proportional to ionization efficiency. The ionization efficiency can be thought of as the propensity of a peptide to ionize, though it can include other factors as well, such as the propensity of the peptide to be produced in the first place, *i.e.*, to be cleaved from the original protein. This idea is expressed formally by the equation $output = input \times ie$, where *output* is the intensity of a spectral peak (measured in number of peptide molecules), *input* is the amount of protein from which the peptide was derived (measured in number of protein molecules), and *ie* is the ionization efficiency, a number between 0 and 1.

Under ideal conditions, each protein molecule would fragment into a set of peptide molecules, each of which would then ionize, enter the mass spectrometer, and contribute to a peak [25]. In this case, the ionization efficiency of each peptide would be 1, and the peaks for a given protein would all have the same intensity (equal to the amount of protein input). In practice, though, peptides do not all ionize with equal efficiency, and they produce peaks of different intensity [5]. In fact, some peptides do not ionize at all, so their ionization efficiency is 0 and they produce no observable peaks [11]. The problem is to account for these differences and, more quantitatively, to estimate the ionization efficiency of each peptide. In data-mining terms, we want to learn a function, f, from peptides to the interval $[0, 1]$, where $f(p)$ is the ionization efficiency of peptide p. We shall identify peptides by their amino-acid sequence, *seq*, so $f(p)$ is really $f(seq)$.[3] The equation for peak intensity therefore becomes $output = input \times f(seq)$, where *output* and *seq* are observed, *input* is unobserved, and f is to be learned.

Varying the form of f leads to a variety of different models of the data and to different

[2]In particular, the suppression effect is the phenomena of chemical interactions between peptides during an MS/MS experiment that can lead to the absence of MS/MS peaks or to new peaks corresponding to unexpected peptides [14, 7].

[3]In fact, we are interested not just in peptides, but in peptide *ions*, which are identified by their sequence and their charge. However, to simplify the presentation, we leave it as understood that charge is included. Charge is explicitly included in our experiments in Section 5.3.

learning problems. In this paper, we consider three types of function:

$$\text{Linear}: f(seq) = \mathbf{x}\bullet\beta \qquad \text{Exponential}: f(seq) = e^{\mathbf{x}\bullet\beta} \qquad \text{Inverse}: f(seq) = 1/(\mathbf{x}\bullet\beta)$$

Here, $\mathbf{x}$ is a vector of peptide properties (derived from seq), β is a vector of parameters (to be learned), and $\bullet$ represents the dot product (or inner product) of the two vectors. We investigate linear models because of their simplicity and because they are 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 (as we shall see in Section 6), 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). As we shall see, both models allow for efficient estimation of the (unknown) amount of protein input, though by very different means.

The inverse model has a different motivation. As we show in Section 5.1, spectral peak intensities 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, we show that the distribution is $O(1/y^2)$, where y denotes peak intensity. It is very 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, as we shall see in Section 4, all the methods we develop for fitting the linear model can easily be adapted to fit the inverse model.

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. Thus, seq_{ij} denotes the amino acid sequence of peptide number j of protein number i. Likewise, $\mathbf{x}_{ij}$ denotes the vector of properties for peptide j of protein i. In the sequel, we shall use y to denote the intensity of a spectral peak. Thus, y_{ij} is the peak intensity of peptide j of protein i. We shall also use in to denote the (input) amount of a protein. Thus, in_i is the amount of protein i. With this notation, the equation for peak intensity described above becomes a set of equations:

$$y_{ij} = in_i \times f(seq_{ij}) \qquad \textit{for } i \textit{ from } 1 \textit{ to } N, \quad \textit{and } j \textit{ from } 1 \textit{ to } n_i. \tag{1}$$

When instantiated with linear, exponential and inverse functions, they become, respectively,

$$y_{ij} = in_i \times (\mathbf{x}_{ij} \bullet \beta) \qquad y_{ij} = in_i \times e^{\mathbf{x}_{ij}\bullet\beta} \qquad y_{ij} = in_i/(\mathbf{x}_{ij} \bullet \beta)$$

Not all these equations contain useful information. In fact, only those proteins that produce at least two peptides can be used for estimating f. If a protein produces only one peptide, then $j = 1$ and the protein yields only one equation: $y_{i1} = in_i \times f(seq_{i1})$. This is the only equation containing the unknown value in_i. Once we know f, we can use this equation to estimate in_i. However, the equation provides absolutely no help in estimating f itself, since it does not constrain f in any way. In fact, it is trivially satisfied for *any* f by using $in_i = y_{i1}/f(seq_{i1})$. If all the proteins produced only one peptide, then we would have no way to estimate f, since any f would do. For this reason, our data-mining methods ignore all those proteins that produce only one peptide. In fact, the first method considered below explicitly requires at least two peptides per protein.

Finally, we note a fundamental limit to what can be learned from the data available to us. In particular, we can only learn the *relative* amount of a protein in a tissue sample, not the *absolute* amount. For instance, we could infer that the amount of protein 1 is twice that of protein 2, but we cannot infer exactly how much there is of either protein. This follows immediately from equation 1. Since in_i and f are both unknown, we can always scale one up as long as we scale the other down by the same amount. Specifically, suppose that $in_1, \cdots, in_N$ and f is a solution to the equation. Then $in'_1, \cdots, in'_N$ and f' is an equally good solution, where $in'_i = in_i \times c$ and $f'(seq) = f(seq)/c$, for all i and seq, and any constant c. Thus, there are infinitely many solutions, each predicting different absolute amounts for the proteins and different ionization efficiencies for the peptides. However, all these solutions predict the same relative values, since $in_i/in_j = in'_i/in'_j$ for any two proteins i and j, and $f(seq_i)/f(seq_j) = f'(seq_i)/f'(seq_j)$ for any two peptides i and j. In this way, the relative amounts of protein and the relative ionization efficiencies of peptides can be learned. Moreover, by using a small amount of calibration data, we can convert these relative values into absolute ones. That is, if we are given the absolute amount of just a few proteins, we can then use the ratios in_i/in_j to estimate absolute values for all the proteins, from which we can estimate absolute values for all the ionization efficiencies.

4 Fitting the Models to Data

In the previous section, we developed three generative models of MS/MS data, linear, exponential and inverse. In this section, we develop methods for fitting these models to data.

An important parameter in any model of MS/MS data is the amount of each protein, in_i, input to the mass spectrometer. If this amount were known, then fitting our models to the data would be a straightforward problem of regression. If the fit is good, this would solve the biological problem of predicting the intensity of each spectral peak given the amino-acid sequence of the peptide and the amount of protein at the input. Unfortunately, the amounts of each protein are unknown. Moreover, there are thousands of proteins, each with a different input level, and the various input levels differ by several orders of magnitude. To deal with these complications, we treat the input levels, in_i, as latent, or hidden variables, whose values must be estimated. We therefore have thousands of hidden values to estimate, in addition to the parameters of our models. An accurate estimation of these values would solve another important biological problem: estimating the amounts of each of the thousands of proteins in a tissue sample given their MS/MS spectra. Moreover, since the protein levels differ so dramatically, even an approximate estimate would be biologically useful.

4.1 Linear Models

In this section, we develop a family of 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 model 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.

Recall that the linear model is given by equations of the form

$$y_{ij} \;=\; in_i \cdot (\mathbf{x}_{ij} \bullet \beta) \tag{2}$$

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. From the discussion at the end of Section 3, 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.

4.1.1 LIN1: Two-Phase Learning

This approach factors out the amount of protein, in_i, from the set of Equations 2. The result is a set of linear eigenvector equations for the parameter vector β, which we can solve using standard eigenvector methods. With this estimate in hand, the value of each in_i can be estimated by linear regression.

From Equations 2, we see that protein i gives rise to the following set of equations, one equation for each peptide:

$$y_{i1} = in_i \cdot (\mathbf{x}_{i1} \bullet \beta) \qquad y_{i2} = in_i \cdot (\mathbf{x}_{i2} \bullet \beta) \qquad \cdots \qquad y_{in_i} = in_i \cdot (\mathbf{x}_{in_i} \bullet \beta) \tag{3}$$

Note that the unknown value in_i is the 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:

$$\mathbf{z}_{ij} \bullet \beta = 0 \qquad \textit{where} \quad \mathbf{z}_{ij} \;=\; y_{ij}\mathbf{x}_{i,j-1} - y_{i,j-1}\mathbf{x}_{ij} \quad \textit{for } j \textit{ from } 2 \textit{ to } n_i, \textit{ and } i \textit{ from } 1 \textit{ to } N. \tag{4}$$

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 4 is a restatement of Equation 2 with the unknown values in_i removed. Like Equation 2, 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 the method of Lagrange multipliers. That is, we minimize the following function:

$$F(\beta, \lambda) \;=\; \sum_{i,j}(\mathbf{z}_{ij} \bullet \beta)^2 \;-\; \lambda(\|\beta\|^2 - 1) \tag{5}$$

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

$$\sum_{ij} \mathbf{z}_{ij}(\mathbf{z}_{ij} \bullet \beta) \;=\; \lambda\beta \tag{6}$$

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:

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

where the left equation is just Equation (6) expressed in matrix notation, 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.

Having estimated a value for β, we must now estimate values for the remaining unknowns, $in_1, \cdots, in_N$. Equations 3 suggests a natural way to do this. 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 a simple univariate linear regression. The values of $in_1, \cdots, in_N$ can thus be estimated by carrying out N such regressions.

4.1.2 LIN2: Improving the Optimization Criterion

Here we improve on the method described above by using a different optimization criterion for fitting the linear model to the data. We still try to minimize the sum of squares $\sum_{ij}(z_{ij} \bullet \beta)^2$, but subject to a different constraint: instead of $\|\beta\|^2 = 1$, we use the constraint $\sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2 = 1$. To see why this is more appropriate, recall that $\mathbf{x}$ is a vector of properties for a peptide. Our estimates of ionization efficiency and amounts of protein should not depend on what units we use to measure peptide propeties. For example, the ionization efficiency of a peptide should not depend on whether we measure peptide mass in milligrams or micrograms (just as the engine efficiency of a car does not depend on whether we measure the mass of the car in grams or kilograms). As long as we retain the same units of measurement for ionization efficiency, its estimated value should not depend on the units of measure for other quantities. The new constraint satisfies this requirement, but the old constraint does not.

To see this, suppose that $\mathbf{x} = (x_1, ..., x_k)$, and suppose we change the units of measure of x_1 so that all their values double. The correct response is to compensate by halving the value of β_1 while leaving the values of all the other β_i unchanged. This leaves the value of $\mathbf{x} \bullet \beta$ unchanged, and hence it leaves all our estimates unchanged, since in the linear model, they depend only on $\mathbf{x} \bullet \beta$. This requirement cannot be satisfied with the constraint $\|\beta\|^2 = 1$, for if β_1 decreases, then some other β_i *must* increase in order to maintain the constraint. In contrast, the new constraint has no such problems, since it depends only on the values of $\mathbf{x} \bullet \beta$, which can remain constant.

More formally, recall from Section 3 that without any constraint, there is no single model that fits the data best, since the estimates of protein level, in_i, can all be scaled up by a constant amount, as long as all the estimates of ionization efficiency are scaled down by the same amount. In other words, if $\mathbf{I}$ is the vector of ionization-efficiency estimates for all peptides, then $c \cdot \mathbf{I}$ is another vector of equally good estimates, for any constant c. Thus, for the best estimates, the direction of $\mathbf{I}$ may be unique, but its magnitude is not. In fact, any magnitude can be chosen arbitrarily. This is just what the new constraint does: it chooses a magnitude of 1. To see this, recall that in the linear model, the ionization efficiency of peptide ij is $\mathbf{x}_{ij} \bullet \beta$, so the new constraint is just $\|\mathbf{I}\|^2 = \sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2 = 1$.

With this constraint, the optimizing function (5) becomes

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

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

$$\sum_{ij} \mathbf{z}_{ij}(\mathbf{z}_{ij} \bullet \beta) \;=\; \lambda \sum_{ij} \mathbf{x}_{ij}(\mathbf{x}_{ij} \bullet \beta) \tag{7}$$

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 (7) in matrix notation, we have the following two equations:

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

The left equation is a *generlalized* 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 using univariate linear regression, as before (the second phase of learning).

4.1.3 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 2. 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 $\alpha_i = \beta = 0$ 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:

$$\frac{\sum_{ij}(y_{ij}\alpha_i) \cdot (\mathbf{x}_{ij} \bullet \beta)}{\sqrt{\sum_{ij}(y_{ij}\alpha_i)^2}\sqrt{\sum_{ij}(\mathbf{x}_{ij} \bullet \beta)^2}} \tag{8}$$

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 peak intensities and peptide property vectors for protein i.

Theorem 1: *Expression (8) 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 (8). (So the minimum angle, θ, between the two vectors is given by $\rho = \cos(\theta)$).

4.2 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,

$$y'_{ij} \;\; = \;\; in'_{ij} \times (\mathbf{x} \bullet \beta)$$

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 in Section 4.1. 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.

4.3 Exponential Models

In this section, we develop two methods for fitting the exponential model to the data. Recall that in the exponential model,

$$y_{ij} \;\; = \;\; in_i \times e^{\mathbf{x}_{ij} \bullet \beta}$$

Taking logs of both sides converts this to a linear model:

$$\log y_{ij} \;\; = \;\; \log in_i + \mathbf{x}_{ij} \bullet \beta \tag{9}$$

The two methods described below differ in their treatment of the latent variables in_i. As with the linear methods of Section 4.1, one method is a two-phase learner, and the other is single-phase. The first method operates in two phases: it estimates a value for the parameter vector, β, and then it uses this estimate to determine values for the protein input levels, in_i. In contrast, the second method operates in a single phase that estimates values for β and in_i simultaneously, finding the combination of values that best fits the data. Unlike the linear methods, the workhorse of these methods is linear regression, not eigenvector decomposition.

4.3.1 EXP1: Two-Phase Learning

From the above equations, we see that for the peptides from protein i,

$$\begin{cases} \log y_{i1} = \log in_i + \mathbf{x}_{i1}\beta \\ \log y_{i2} = \log in_i + \mathbf{x}_{i2}\beta \\ \qquad \cdots \\ \log y_{in_i} = \log in_i + \mathbf{x}_{in_i}\beta \end{cases} \tag{10}$$

From this, we get that $\log y_{i,j-1} - \log y_{i,j} = (\mathbf{x}_{i,j-1} - \mathbf{x}_{i,j})\beta$ for j from 2 to n_i, or equivalently $Y_{ij} = X_{ij} \bullet \beta$, where

$$Y_{ij} = \log y_{i,j-1} - \log y_{i,j} \qquad\qquad X_{ij} = \mathbf{x}_{i,j-1} - \mathbf{x}_{i,j} \tag{11}$$

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 9 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 9, 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 $\varepsilon_{ij} = logy_{ij} - \log in_i - \mathbf{x}_{ij} \bullet \beta$. Instead, however, 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.

4.3.2 EXP2: Simultaneous Learning

In order to minimize the errors ε_{ij} directly, we treat the unknown values $\log in_i$ in Equation (9) 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 $\mathbf{O}$ to be the following sparse matrix:

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

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. $\mathbf{X} = (\mathbf{x}_{11}, ..., \mathbf{x}_{1,n_1}, \mathbf{x}_{21}, ..., \mathbf{x}_{2n_2}, ..., \mathbf{x}_{N1}, ..., \mathbf{x}_{Nn_N})^T$, 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 (9) 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

$$\beta' = (\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T Y \tag{12}$$

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, $\mathbf{x}_{ij}$, representing peptide sequences. 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:

$$(\mathbf{V}^T\mathbf{V})^{-1} \quad = \quad \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \tag{13}$$

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:

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

We can view this as four matrix equations in four unknowns, A, B, C, D. Solving, we get

$$A \;=\; (\mathbf{X}^T\mathbf{X} - \mathbf{X}^T Q\mathbf{X})^{-1} \qquad B \;=\; -AU^T \qquad C \;=\; -U^T A \qquad D \;=\; P^{-1} - U^T AU^T$$

where $P = \mathbf{O}^T\mathbf{O}$, $Q = \mathbf{O}P^{-1}\mathbf{O}^T$ and $U = P^{-1}\mathbf{O}^T\mathbf{X}$. The important point 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. Moreover, since the rows of $\mathbf{O}^T$ are orthogonal, P is diagonal. Thus, even though P is an extremely large matrix, of size $N \times N$, it is easily inverted. In fact, the i^{th} diagonal element is simply n_i, the number of peptides in protein i, so the i^{th} diagonal element of P^{-1} is $1/n_i$.

In this way, we can efficiently invert the matrix $\mathbf{V}^T\mathbf{V}$ and solve for the extended parameter vector β' in Equation (12).

5 Experiments

We evaluated the methods and models developed above on real and simulated datasets. This section describes the datasets, the design of the experiments, 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.

5.1 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 data sets as **Mouse Brain Data**, **Mouse Heart Data**, and **Mouse Kidney Data**. The Brain data set contains 10,786 peptides, with peak intensities ranging from 1 to 2,500; the Heart data set contains 9,623 peptides, with peak intensities from 1 to 1,996; and the Kidney data set contains 8,791 peptides, with peak intensities from 1 to 1,491.

Histograms of the peak intensities for the three data sets are shown in Figure 5.1. These show that the peak intensities are far from being uniformly or normally distributed. Instead, peak intensities are heavily concentrated near the minimum value of 1, and their frequency rapidly falls off as peak intensity increases. For example, although the maximum peak intensity in the Kidney dataset is 1,491, the median value is only 2! That is, half the peptides have peaks with intensities of only 1 or 2, while the remaining peptides have peaks with intensities 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 peak intensities is provided by the probability plots in Figure 5.1, which display values in sorted order. The two panels in Figure 5.1 show probability plots for the reciprocal of peak intensity; that is, they are probability plots of $1/y$, where y is peak intensity. 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 peak intensities, while the plot in the right panel excludes peaks of low intensity, that is, with an intensity below 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 [23]. The left hand plot is the same except

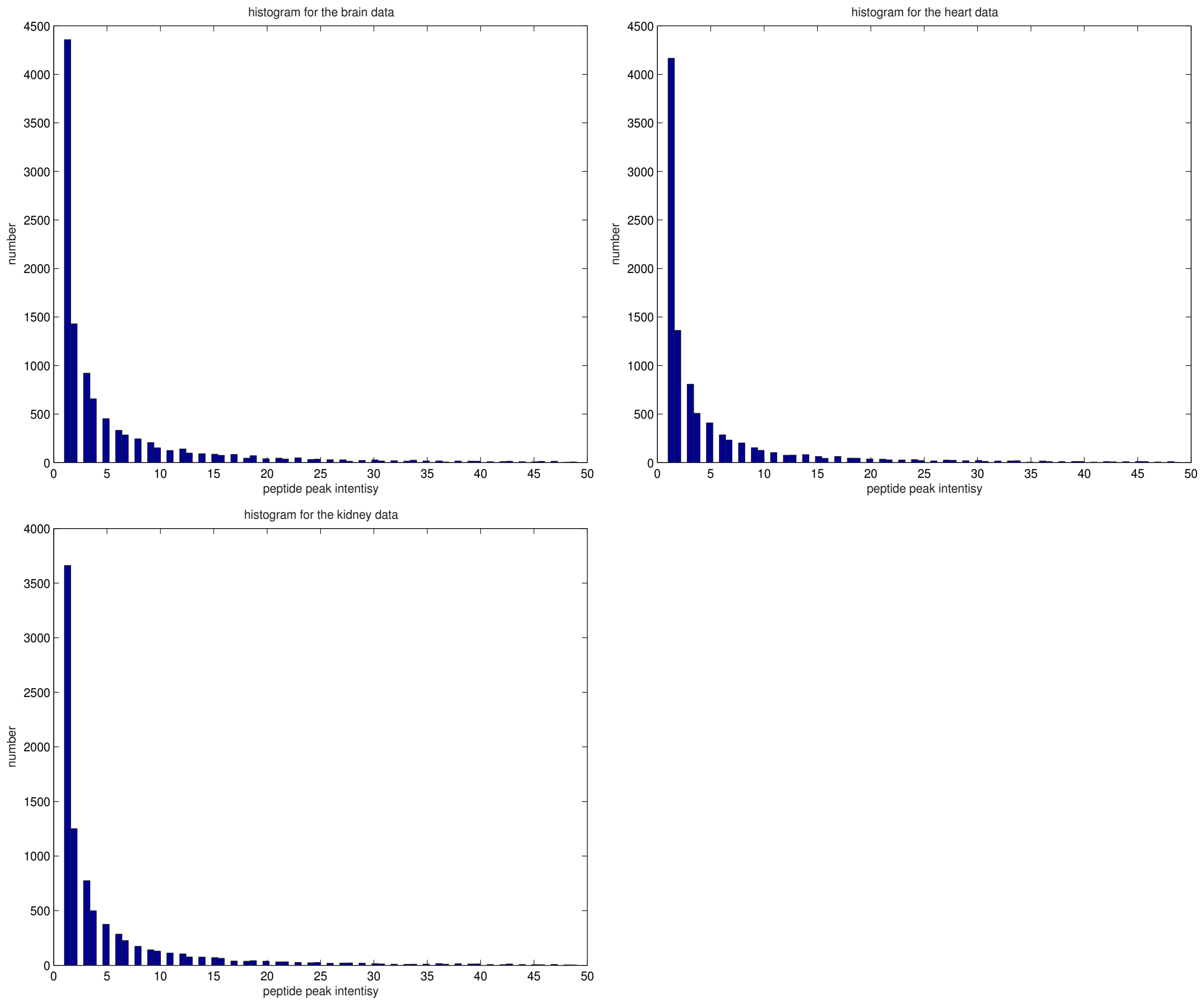
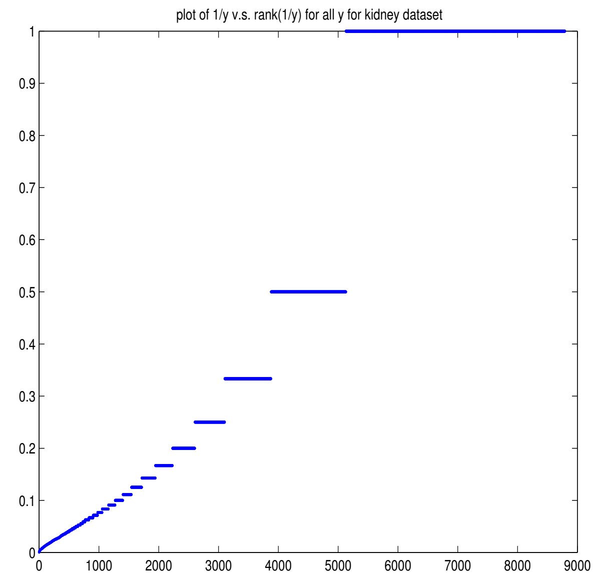


Figure 2: Histograms of peptide peak intensities. The left histogram is for the Brain dataset, the middle is for Heart, and the right is for Kidney.

that it curves up for large values of $1/y$, that is, for low peak intensities. 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 peak intensities 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.

5.2 Simulated Datasets

In addition to real-world datasets (described above), we generated simulated data on which to test the learning models and methods developed in Sections 3 and 4. While real-world data tests the biological relevance of our methods, simulated data allows us to test their mathematical and computational correctness. To this end, we use simulated data that is based on the same statistical models as our learning methods. This serves



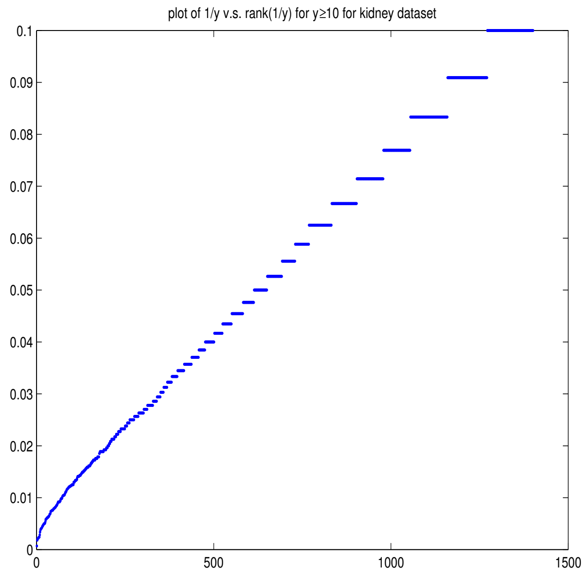


Figure 3: Probability plots of $1/y$ for the Kidney dataset, where y is spectral peak intensity. The left plot includes all observed values of y, while the right plot excludes values of y less than 10.

two purposes. First, it tells us the performance we can expect from the methods under ideal circumstances, and with controllable amounts of noise. Second, it acts as a sanity check, since errors in either the mathematics or the programming of the methods can easily appear as unexpected or bizarre behavior. In [17], we perform a thorough set of simulated experiments. In this paper, we present a small sample of results to illustrate the ability of our methods to estimate protein levels.

The simulator generates data for the linear, inverse and exponential models. Although different, these models each have the same kinds of parameters: an input amount, in_i, for each protein; a feature vector, $\mathbf{x}_{ij}$, for each peptide; and a parameter vector, β, with which to compute ionization efficiencies. Protein amounts are generated randomly from a uniform distribution ranging from 20 to 250. Peptides are represented by vectors with 22 components, as in the real datasets. For each component of a peptide vector, values are generated randomly from a normal distribution with mean 4 and standard deviation 2. For each component of the vector β, values are generated randomly from a uniform distribution ranging from 0 to 1. From this data, we compute ionization efficiencies and spectral peak intensities, according to our models of MS/MS data. In each model, we add noise to $\mathbf{x}_{ij} \bullet \beta$, since this is what our methods minimize. We then compute ionization efficiencies from these values. Thus, for the linear model, the ionization efficiency of peptide i of protein j is $ie_{ij} = \mathbf{x}_{ij} \bullet \beta + noise$, for the inverse model it is $ie_{ij} = 1/(\mathbf{x}_{ij} \bullet \beta + noise)$, and for the exponential model it is $ie_{ij} = e^{\mathbf{x}_{ij} \bullet \beta + noise}$. In all these cases, the noise is generated from a normal distribution with mean 0. Finally, spectral peak intensities are generated, and in all three models, the peak intensity of peptide ij is $y_{ij} = in_i \cdot ie_{ij}$. For the simulated data, the standard deviation of the noise is constant, but different experiments use different amounts of noise, *i.e.*, noise with different standard deviations. Here, noise level is measured with respect to the variation within the peptide data. For example, a noise level of 50% means that the standard deviation of the noise if 50% of the standard deviation of the components of the peptide vectors $\mathbf{x}_{ij}$. For each simulated experiment, we simulated 100 different proteins, each fragmenting into 10 different peptides, for a total of 1000 peptides. In addition, in order to obtain a nice visual representation of the accuracy of our estimates of protein level, we divided the proteins into ten groups, where each protein in a group has exactly the same input amount, in_i.

Each simulated experiment uses the same protein levels, in_i, the same peptide feature vectors, $\mathbf{x}_{ij}$, the same parameter vector, β, and often the same noise. Of course, the datasets for the three models—linear, inverse, and exponential— will still differ, since, according to the formulas given above, the ionization efficiencies, ie_{ij}, and spectral peak intensities, y_{ij}, will be different in the three models.

5.3 Experimental Design

As described in Section 3, only those proteins that produce at least two peptides with observable spectral peaks are useful for fitting models to data. We first identified these proteins. We then built an index for these proteins, and a separate index for their peptides. For the Brain dataset, the indexes contain 8,527 peptides and 1,664 proteins, respectively. For the Heart dataset, the indexes contain 7,660 peptides and 1,281 proteins, respectively. For the Kidney dataset, the indexes contain 7,074 peptides and 1,291 proteins, respectively. In addition, in order to estimate the generalization error of the fitted models, each of these three data sets is divided randomly into two subsets, training data and testing data, in a 2:1 ratio.

Finally, to use the models developed in Section 3, 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. Recall that a peptide is a sequence of amino acids. Since there are twenty different amino acids, the vector has 20 components, $(x_1, ..., x_{20})$, 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, x_0, whose value is always 1, to represent a bias term, as is common in linear regression [12]. 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, 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 41st component to act as a bias term, and a 42nd 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.

These four different peptide representations can be used with any of the eight fitting methods developed in Section 4, namely, LIN1, LIN2, LIN3, EXP1, EXP2, INV1, INV2 and INV3. When applied to the training data, these methods each estimate a vector of parameters, β, from which we can estimate the ionization efficiency of any peptide. As described in Section 3, the ionization efficiency of a peptide is given by the following formulas, depending on the model:

$$\text{Linear}: \; ie = \mathbf{x}\bullet\beta \qquad \text{Exponential}: \; ie = e^{\mathbf{x}\bullet\beta} \qquad \text{Inverse}: \; ie = 1/(\mathbf{x}\bullet\beta) \tag{14}$$

where $\mathbf{x}$ is the vector representation of the peptide, as described above. To estimate the accuracy of these models, we use them to predict the spectral peak intensities of all the peptides in the testing data. We then compare the predictions to the observed values.

The first step is to estimate the amount of each protein in the testing data. Recall that $y_{ij} = in_i \cdot ie_{ij}$, where in_i is the amount of protein i, ie_{ij} is the ionization efficiency of peptide ij, and y_{ij} is the intensity of the peptide's spectral peak. The goal is to estimate a value for in_i given the observed values of y_{ij} and the estimated values of ie_{ij}. There are numerous ways this can be done, and the most appropriate depends on the statistical model being used. Recall that in each model, the formula $y_{ij} = in_i \cdot ie_{ij}$ is transformed so that $\mathbf{X} \bullet \beta$ appears as a linear term. In particular:

$$\text{Linear}: \quad y_{ij} = in_i \cdot (\mathbf{x}_{ij} \bullet \beta) \qquad \text{Exponential}: \quad \log(y_{ij}) = \log(in_i) + \mathbf{x}_{ij} \bullet \beta$$

$$\text{Inverse}: \quad y_{ij}^{-1} = in_i^{-1} \cdot (\mathbf{x}_{ij} \bullet \beta)$$

In each of these models, y_{ij}, $\mathbf{x}_{ij}$ and β are known, and in_i must be estimated. Thus, in the linear model, in_i can be estimated by univariate linear regression. Likewise, in the inverse model, after first computing values for y_{ij}^{-1}, a value for in_i^{-1} can be estimated by univariate linear regression, from which the value of in_i can be estimated as $1/in_i^{-1}$. Finally, in the exponential model, a value for $\log(in_i)$ can be estimated as the mean of $\log(y_{ij}) - (\mathbf{x}_{ij} \bullet \beta)$, from which the value of in_i can be estimated by taking exponentials.

Let $\hat{in}_i$ denote the estimate of in_i, and let $\hat{ie}_{ij}$ be the estimate of ie_{ij}. If these were the true amount of protein i and the true ionization efficiency of peptide ij, then the intensity of the peptide's spectral peak would be $\hat{y}_{ij} = \hat{in}_i \times \hat{ie}_{ij}$. By comparing this estimate to the observed value, y_{ij}, for each peptide in the test set, we obtain an estimate of the generalization error of our methods.

A common way to make such a comparisons is to use 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 the Pearson correlation 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 the Spearman rank correlation coefficient is invariant under monotonic transformations of the data. Thus, it does ot matter whether we consider y or $\log(y)$ or $1/y$.

Finally, in order to judge the significance of our results, we compare them to a set of naive models. We use naive versions of the linear, exponential and inverse models, denoted LIND, EXPD and INVD, respectively (where D stands for "Dumb"). In each naive model, each peptide is simply assumed to have an ionization efficiency of 1. From Equations (14), this is equivalent to assuming that, for all peptides, $\mathbf{x} \bullet \beta$ is 1 in the linear model, 0 in the exponential model, and 1 in the inverse model. We then specialize the methods described above for estimating in_i, as follows:

- in the linear model, in_i is estimated to be the mean of y_{ij};
- in the exponential model, $\log(in_i)$ is estimated to be the mean of $\log(y_{ij})$;
- in the inverse model, $1/in_i$ is estimated to be the mean of $1/y_{ij}$.

Using these estimates of protein input, the spectral peak intensity of each peptide is estimated to be $\hat{y}_{ij} = \widehat{in}_i \cdot \widehat{ie}_{ij} = \widehat{in}_i$, since $\widehat{ie}_{ij} = 1$ in all of the naive models. Finally, we compare these estimated peak intensities to the observed peak intensities using Spearman's rank correlation coefficient. Note that for the naive models, the Spearman coefficient will not depend on which vectors we use to represent peptide sequences, since the models themselves are independent of this.

6 Results and Analysis

6.1 Experiments on Simulated Data

Since the protein levels are unknown in the real datasets, we must resort to simulated data in order to directly compare the estimated protein levels to their true values. Figures 4 and 5 illustrate the estimation of protein input levels for the simulated data. The straight blue lines in the figures represent the true protein levels, while the jagged red lines represent their estimated values.[4] Notice that the curve of estimated values has the same general shape as the curve of real values: when the real values rise or fall dramatically, so to the estimated values; and when the real values remain constant, the estimated values vary around the true value. However, the amount of variation varies considerably from model to model.

The upper panels in Figure 4 clearly show that the variance in the estimated protein levels is much greater for LIN1 than for the other two linear models, LIN2 and LIN3, which have about equal variance. In addition, the lower panels in the figure show that the variance is about equal for all three inverse models, INV1, INV2 and INV3. The upper and lower panels cannot be directly compared, however, since the linear and inverse models are based on different noise models. (As described in Section 5.2, in the linear model, noise is added to ionization efficiency, ie, while in the inverse model, noise is added to $1/ie$.)

Figure 5 shows a similar trend for the two exponential models, EXP1 and EXP2: the curve of estimated protein levels has the same general shape as the curve of true protein levels. In addition, we can see that the variance in the estimated protein levels is about the same for the two exponential models. Note, however, that the ostensible noise level is half of that used in Figure 4 for the linear and inverse models, yet the variance in protein estimates appears to be much higher. This is because the noise model is multiplicative, not additive. In particular, as described in Section 5.2, the ionization efficiency is multiplied by e^{noise}. Thus, an ostensible noise level of 25% has a much greater impact on the exponential model than on the linear model.

A more comprehensive set of simulated experiments is presented in [17].

6.2 Experiments on Real-World Data

Tables 2, 3 and 4 show experimental results for every combination of eleven 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 peptide peak intensity and estimated peptide peak intensity. An entry of ****** means that the method could not be used because a matrix turned out to be singular. The performance of the three best methods, along with the naive methods, is illustrated graphically in Figure 6, which plots the Spearman rank correlation coefficient against the size of the feature vectors. Each curve in the figure represents a different method.

[4] As emphasized in Sections 3 and 4, it is only possible to estimate the relative values of protein levels, not their absolute values. Thus, the true and estimated protein levels may differ by a possibly-large constant factor. We have compensated for this in the figures by scaling the estimates so that they have the same sum of squares as the true values.

The following observations about the fitting methods are immediately apparent from these tables: 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; the exponential methods perform the best of all; the naive methods perform better than many of the other methods; and of the naive methods, INVD performs the worst.

Recall that the only difference between LIN1 ad LIN2 is the constraints on which they are based. The poor performance of LIN1 suggests that its constraint is much less appropriate than the constraint for LIN2, as argued in Section 4.1.2. 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 input levels are estimated from β. In contrast, LIN3 is based on a single-phase approach to learning, in which parameters and protein levels 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 suggested in Section 4.1. (Interestingly, in our experiments on simulated data, LIN2 and LIN3 performed comparably, and clearly better than LIN1 [17].) 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. However, the result is not conclusive, since the naive inverse method, INVD, performs significantly worse than the other two naive methods, LIND and EXPD. In the naive methods, there is no attempt to model the ionization efficiency of peptides, so that data transformation is the only factor that distinguishes them.

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 peptide peak intensities, 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 two 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.

However, even the exponential methods perform only slightly better than the best naive method (which, as it turns out, is the exponential naive method, EXPD). The naive methods essentially estimate the protein level by taking an average of the peptide peak intensities for that protein. That they perform as well as they do can be interpreted in a straightforward way. Higher protein levels will certainly give rise to greater peptide peak intensities. Apparently, the reverse is also true to an extent: greater peptide peak intensities are, on average, a reflection of higher protein levels. Put another way, two factors contribute to peak intensity: the input level of the protein, and the ionization efficiency of the peptide. The naive methods focus entirely on the former factor, which by itself is enough to garner a correlation coefficient of about 0.5. However, the naive methods do nothing to explain the very large differences in peak intensity between different peptides from the same protein. Instead, they assume the peaks are all of equal intensity, something that is not observed experimentally.

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. This suggests that sequential information (as captured in these latter two vectors) is not as important as good statistics on amino-acid counts (as captured in the 22-feature vector). 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.

Table 2: Spearman rank correlation coefficients on the Mouse Brain dataset

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2	LIND	INVD	EXPD
22	0.2080	0.4342	0.4259	0.1582	0.4755	0.4755	0.4986	0.5051	0.4591	0.3949	0.4691
42	0.2430	0.2645	0.4245	0.0065	0.4848	0.5588	0.4981	0.5049	0.4591	0.3949	0.4691
62	0.2146	0.2365	0.4265	0.0346	0.4846	0.4840	0.5033	******	0.4591	0.3949	0.4691
232	0.2103	0.2058	0.4218	0.0370	0.0594	0.4785	0.5141	******	0.4591	0.3949	0.4691

6.3 Visualization of the Goodness of Fit

As mentioned above, the three methods with the best correlation coefficients are EXP1, INV3 and LIN3. To help visualize the goodness of the fit provided by these methods, Figure 7 provides log-log plots of observed and estimated values for a training dataset.[5] For each method, we provide two figures, a plot of y v.s. $\hat{y}$, and a plot of ie v.s. $\hat{ie}$. Here,

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

Table 3: Spearman rank correlation coefficients on the Mouse Heart dataset

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2	LIND	INVD	EXPD
22	0.2039	0.4965	0.4965	0.2386	0.5635	0.5635	0.5669	0.5699	0.5392	0.4861	0.5418
42	0.1721	0.1844	0.4243	0.0106	0.5611	0.5556	0.5658	0.5656	0.5392	0.4861	0.5418
62	0.1355	0.1691	0.4974	0.0621	0.0667	0.5605	0.5867	******	0.5392	0.4861	0.5418
232	0.1477	0.1399	0.3285	0.0048	0.0307	0.4713	0.5634	******	0.5392	0.4861	0.5418

Table 4: Spearman rank correlation coefficients on the Mouse Kidney dataset

	LIN1	LIN2	LIN3	INV1	INV2	INV3	EXP1	EXP2	LIND	INVD	EXPD
22	0.1655	0.4599	0.4609	0.2299	0.5250	0.5208	0.5207	0.5220	0.4908	0.4413	0.4909
42	0.2482	0.2339	0.4662	0.0584	0.5027	0.4991	0.5089	0.5188	0.4908	0.4413	0.4909
62	0.2024	0.2080	0.4690	0.0421	0.5033	0.5065	0.5151	******	0.4908	0.4413	0.4909
232	0.1818	0.2357	0.3992	0.0219	0.0124	0.4953	0.5105	******	0.4908	0.4413	0.4909

ie and $\hat{ie}$ are derived from y and $\hat{y}$, respectively, by dividing by $\hat{in}$, the estimate of protein input level. ie and $\hat{ie}$ thus provide two different measures of the ionization efficiency of a peptide. Note that $\hat{ie}$ is the same as the estimate described in Section 5.

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 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. Formally, dividing y by $\hat{in}$ produces a real number, since $\hat{ie}$ is 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. This is another measure by which it provides the best fit to the experimental data, in addition to its relatively high correlation coefficient. In contrast, the plots for LIN3 and INV3 all show evidence of residual structure that the methods were unable to fit.

7 Conclusion and Future Work

We developed and evaluated a number of methods for estimating protein levels and peptide peak intensities in Tandem Mass Spectrometry (MS/MS) data. Other researchers have attempted to estimate the peptide peak intensities in an MS/MS spectrum given the amount of protein input to a mass spectrometer [10]. However, in this paper, we addressed the reverse problem of estimating the amount of protein input given an MS/MS spectrum. A solution to this problem would allow biologists to efficiently estimate the amounts of the thousands of proteins in a tissue sample. To our knowledge, this is the first attempt to solve this problem using large amounts of data.

The methods we developed are based on simple, generative models of MS/MS data. In fitting the models to the data, the methods attempt to solve two problems simultaneously: estimating protein input levels, and explaining why different peptides produce peaks of different intensity. Of the eight methods developed, the two exponential methods, EXP1 ad EXP2, performed the best. However, this research is just a first step, and additional work is needed before protein levels and peak intensities can be predicted accurately.

The results in this paper suggest several directions for future work:

- More sophisticated regression methods, such as kernel methods and regression trees could be tried. String kernels, in particular, could extract more information from the peptide sequences. Such methods could easily be adapted to the framework of our EXP1 method, which transforms the data into a more manageable form while simultaneously factoring out the unknown amount of protein input.
- 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.
- Although we tested several representations of peptides in this paper, it is not yet clear whether the solution lies in more sophisticated regression schemes or in better representations of peptides. Representations that capture more sequential information (such as string kernels) may be important. In addition, a more refined representation of peptide peaks may be needed. For instance, peak shape, not just peak intensity, may be important. To this end, we have recently acquired more detailed MS/MS data that allows us to estimate peak shape.
- The possible problem of overfitting can be addressed by regularization and bagging. In particular, regularized linear regression could easily be incorporated into our EXP1 method in order to test the utility of large vector representations of peptides.

Acknowledgment

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. Mr Liu is supported by the University of Toronto.

References

[1] Aebersold,R., Mann,M. *Mass spectrometry-based proteomics*, Nature 422, pp198-207, 2003

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

[3] Bonner,A., Liu,H. *Canonical Correlation, an Approximation, and its Application to the Mining of Tandem Mass Spectrometry Data*. Submitted for publication. Available at `www.cs.toronto.edu/~bonner/publications`.

[4] Bornsen,K.O., Gass,M.A.,Bruin,G.J.,Von Adrichem, J.H.et al. *Rapid Commun*,Mass Spectrom 11, pp603-609 1997

[5] Corthals,G.L., Wasinger,V.C., Hochstrasser,D.F.and Sanchez, J.C. *The dynamic range of protein expression: a challenge for proteomic research*,Electrophoresis 21, pp1104-1115 2000

[6] Dogruel,D., Nelson,R.W., Williams,P., *Rapid Commun*,Mass Spectrom. 2,pp695-700, 1998

[7] Eng,J., McCormack,A., Yates,J.R. *An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database*,j.Am.Soc.Mass Spectrom.5,, pp976-989, 1994

[8] Figueroa,I.D.,Torres, O.,Russell,D.H.,Annal Chem. 70, pp4527-4533 1998

[9] Gaskell,S.*Electrospray: Principles and practices*. Journal of Mass Spectrometry, 32, pp677-688 1997

[10] Gay,S. ,Binz, P.A., Hochstrasser,D.F., Appel,R.D. *Peptide mass fingerprinting peak intensity prediction: extracting knowledge from spectra*,Proteomics 2, pp1374-1391, 2002

[11] Gygi,S.P., Corthals, G.L., Zhang,Y., Rochon,Y., and Aebersold,R. *Evaluation of two-dimensional gel electrophoresis-based proteome analysis technology*,Proc.Natl.Acad.Sci.U.S.A. pp9930-9395, 2000

[12] Hastie, H., Tibshirani, R., Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Statistics, 2001.

[13] Joshua E Elias, Francis D Gibbons, Oliver D King, Frederick Roth, Steven P Gygi *Intensity-based protein identification by machine learning from a library of tandem mass spectra*,Nature, biotechnology. Volume 22 Number 2, 2004

[14] Knochenmuss, R., Karbach,V.,Wiesli,U.,Breuker,K.,Zenobi,R.,Rapic Commun.Mass Spectrom.12, pp529-534, 1998

[15] Kocher,F.,Favre,A,Gonnet,F.,Tabet,J.C.et al., Jouranl of Mass Spectrom.33, pp921-935, 1998

[16] Kussmann, M., Nordhoff,E., Rahbek-Nielsen, H., Haebel,S.et al., Journal of Mass Spectrom. 32 pp593-601, 1997

[17] Liu, H. *Mining Tandem Mass Spectrometry Data*, Masters thesis, Department of Computer Science, University of Toronto. Forthcoming.

[18] Mann,M., Hendrickson, R.C., Pandey, A. *Analysis of proteomes by mass spectrometry*, Annu.Rew.Biochem.70, pp437-473, 2001

[19] Martin,K., Spickermann,J., Rader,H.J., Mullen,K.,,*Rapid Commun.*Mass Spectrom.10, pp1471-1474, 1996

[20] Mathurin, J.C., Gregoire,S.,Brunot,A. Tabet,J.C.et al. Journal of Mass Spectrom.32, pp829-837, 1997

[21] Pandey, A., Mann,M. *Protemics to study genes and genomes*, Nature 405, pp837-846, 2000

[22] Purves, R. W., Gabryelski,W., Li,L.,Rapic Commun.Mass Spectrom.2, pp695-700, 1998

[23] Rice, J. *Mathematical Statistics and Data Analysis*, second edition. Duxbury Press, 1995.

[24] Thosma K., Khaled R.,Dragan Radulovic., *PRIM, a Generic Large Scale Proteomics investigation Strategy for Mammals*, Molecular& Cellular Proteomics 2.1 2003

[25] Washburn,M.P., Wolters,D., and Yates, J.R., III *Large-scale analysis of the yeast proteome by multidimensional protein identification technology*, Nat. Biotechnol. 19, pp242-247

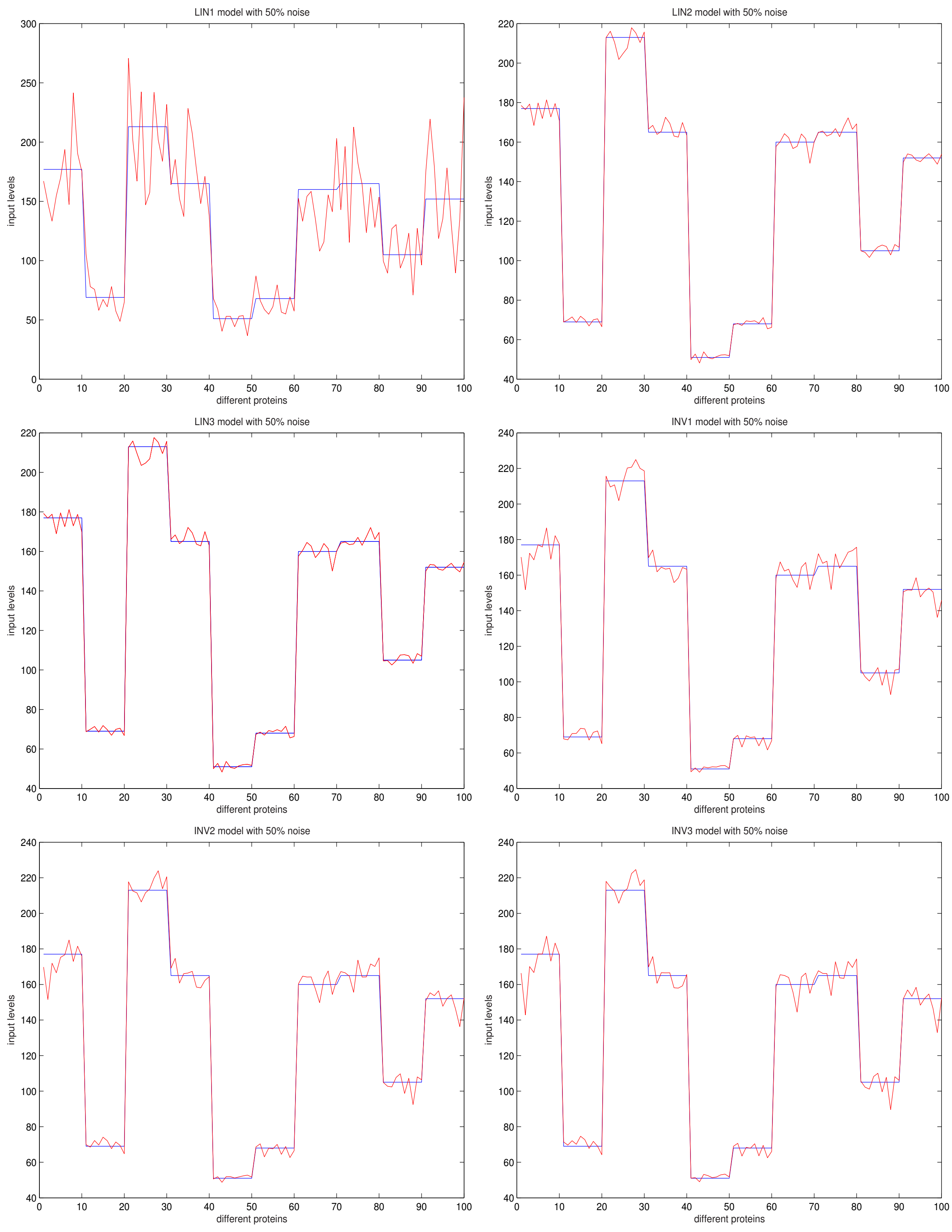


Figure 4: Illustration of the estimated input levels for different linear and inverse models with noise level 50%, The upper panel shows the fitting result for LIN1,LIN2,LIN3, while the lower panel shows the results for INV1,INV2 and INV3

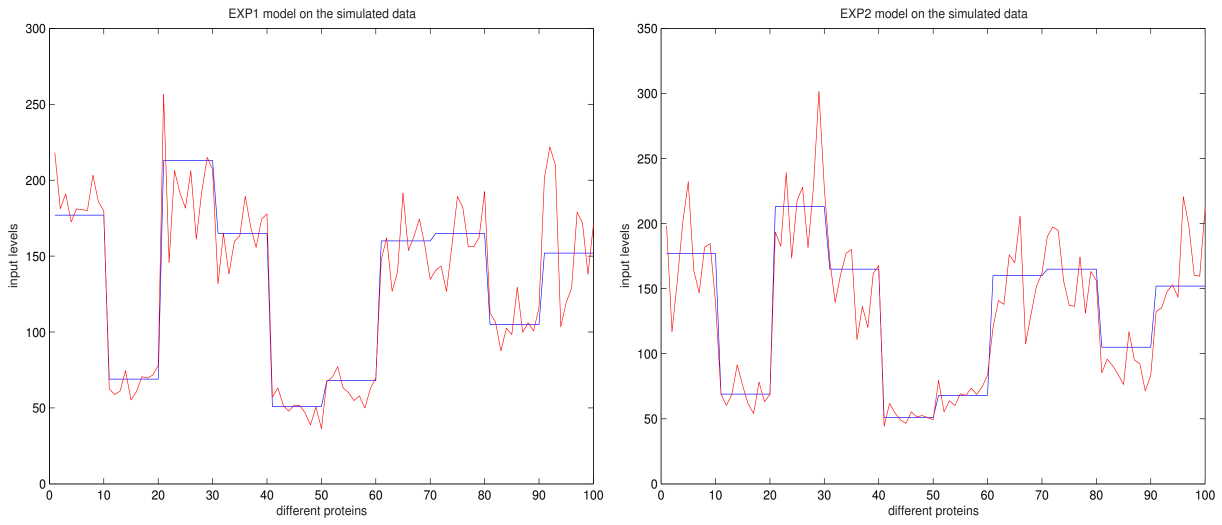


Figure 5: Illustration of the estimated input levels for different exponential models with noise level 25%. The left panel shows results for EXP1, while the right panel shows results for EXP2

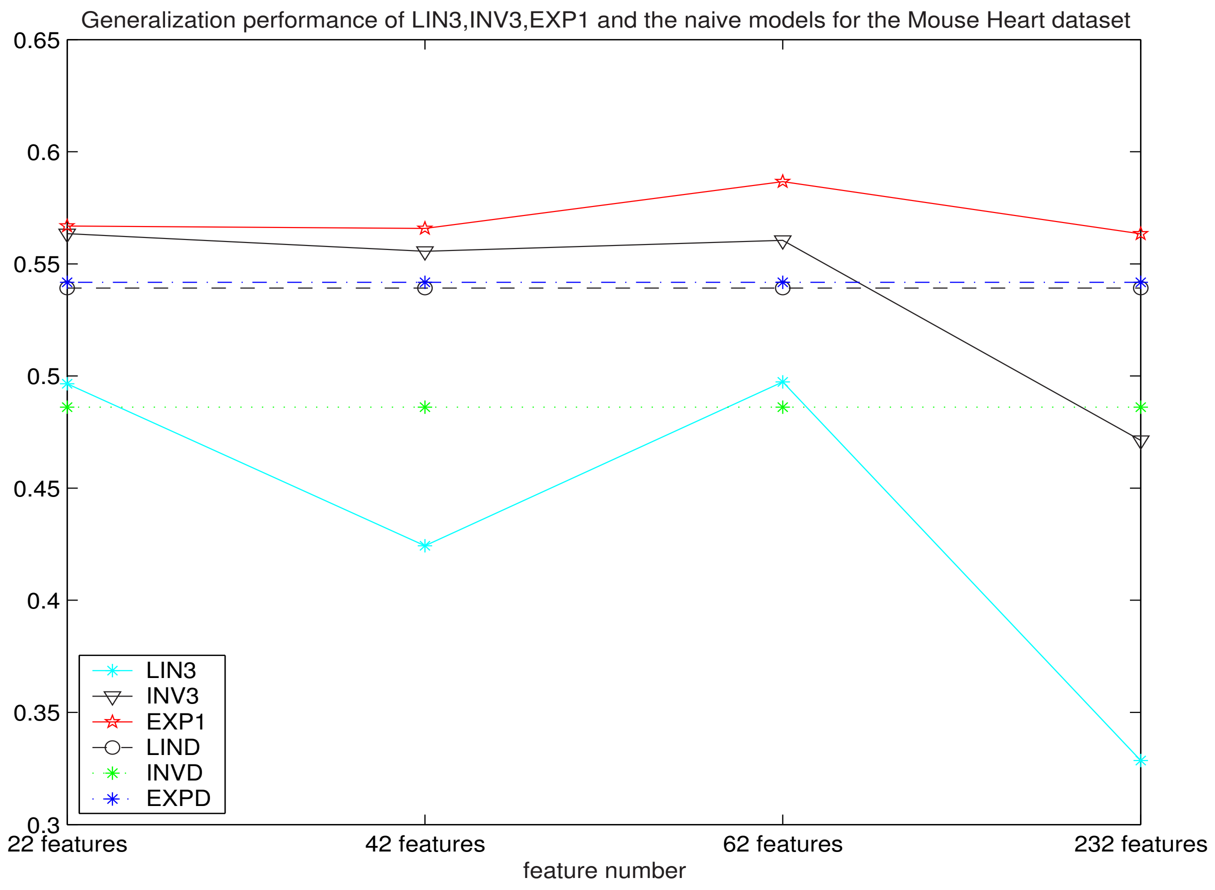


Figure 6: A comparison of the generalization performance of different models on the Mouse Heart dataset

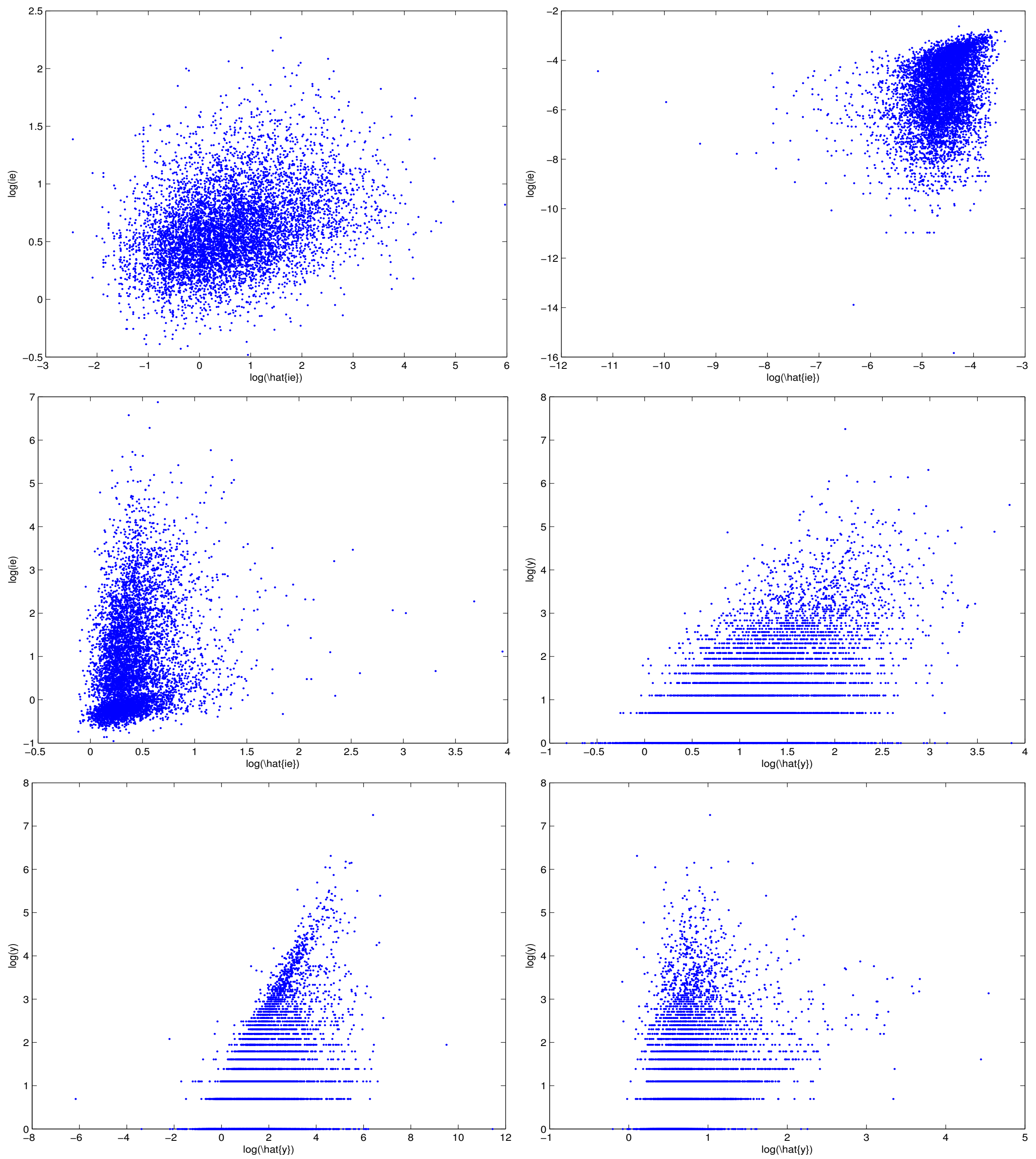


Figure 7: Log-log plots of y v.s. $\hat{y}$ (bottom row) and ie v.s. $\hat{ie}$ (top row) on the Kidney dataset. The two left panels are the results of EXP1 method, the two middle panels are the results of the LIN3 method, and the two right panels are the results of the INV3 method.