Mass Spectrum Labeling: Theory and Practice

Z. Huang, L. Chen, J-Y. Cai, D. Gross*, D. Musicant*, R. Ramakrishnan, J. Schauer*, S.J. Wright

Abstract

*We introduce the problem of **labeling a particle's mass spectrum** with the substances it contains, and develop several formal representations of the problem, taking into account practical complications such as unknown compounds and noise. This task is currently a bottle-neck in analyzing data from a new generation of instruments for real-time environmental monitoring.*

1. Introduction

Mass spectrometry is widely used for the identification and quantification of elements, chemicals and biological materials. Historically, the specificity of mass spectrometry has been aided by upstream separation to remove mass spectral interference between different species. However, in the past decade, a wide range of real-time mass spectrometry instruments have been employed, and the nature of these instruments often precludes separation and clean-up steps. The mass spectrum produced for a particle in real-time by one of these instruments, e.g., the Aerosol Time-of-Flight Mass Spectrometer (ATOFMS) [12,9,14,16], is therefore comprised of overlaid mass spectra from several substances, and the overlap between these spectra makes it difficult to identify the underlying substances. The commercially available ATOFMS instrument can obtain mass spectra for up to about 250 particles per minute, producing a time-series with unusual complexity. The data analysis challenges we describe are equally applicable to other real-time instruments that utilize mass spectrometry, such as the Aerosol Mass Spectrometer (AMS).

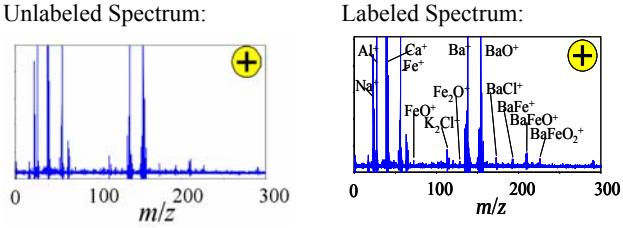

Figure 1: Mass spectrum labeling

Mass spectrum labeling consists of "translating" the raw plot of intensity versus mass-to-charge (m/z) value to a list of chemical substances or ions and their rough quantities (the quantities omitted in Figure 1) present in the particle. Labeling spectra allows us to think of a stream of mass spectra as a time-series of observations, one per collected particle, where each observation is a set of ion-quantity pairs. This is similar to a time-series of transactions, each recording the items purchased by a customer in a single visit to a store [1,4,13]. This analogy makes a wide range of association rule [3] and sequential pattern algorithms [2] applicable to the analysis of labeled mass spectrometry data.

The contributions of this paper include the following: In this and a companion paper [7], we introduce an important class of data mining problems involving mass spectra. The focus in this paper is on the labeling of individual spectra (Section 2), which is the foundation of a class of group-oriented labeling tasks discussed in [7]. We introduce a rigorous framework for labeling and present a theoretical characterization of *ambiguity*, which arises due to overlapped spectra (Section 3). We account for practical complexities such as noise, errors, and the presence of unknown substances (Section 4), and present algorithms together with several optimizations and theoretical results (Section 5). We then present a detailed synthetic data generator that is based on real mass spectra, conforms to realistic problem scenarios, and allows us to produce labeled spectra while controlling several fundamental parameters such as ambiguity and noise (Section 6). Finally, we introduce a metric for measuring the quality of labeling, and evaluate our labeling algorithms, showing that although slower than some machine learning approaches, they achieve uniformly superior accuracy *without the need for training datasets* (Section 7). In many real settings, it is unrealistic to expect labeled training sets (e.g., when deploying an instrument in a new location, or when the ambient conditions change significantly). We also apply our algorithms to a collection of real spectra and compare our results with hand-labeling by domain scientists; they are effective enough (achieving 93% accuracy in detecting true labels) to be immediately useful.

2. Problem formalization

A **mass spectrum** (or spectrum) is a vector $\vec{b} = [b_1, \cdots b_r]$, where $b_i \in R$ is the signal intensity at mass-to-charge (m/z) value i. For simplicity, we assume all spectra have the same 'range' and 'granularity' over the m/z axis; i.e., they have the same dimension r and the i^{th} element of a spectrum always corresponds to the same m/z value i. Intuitively, each m/z ratio corresponds to a particular isotope of some chemical element. The **signature** of an ion is a vector $\vec{s} = [I_1, I_2 \cdots I_r]$, $I_i \in R$ and $\sum I_i = 1$, representing the distribution of isotopes. I_i is the proportion of the isotope with m/z value i. A **signature library** is a set of known signatures $S = \{\vec{s}_1, \vec{s}_2 \cdots \vec{s}_n\}$, in which $\vec{s}_j$ is the signature of ion j. Additionally, there may be ions that appear on particles, and are therefore reflected in mass spectra, but that for which signatures are not included in the signature library.

The spectrum $\vec{b}$ of a particle is a linear combination of the signatures of ions that it contains. $\vec{b} = \sum_j w_j \vec{s}_j$, where w_j is the quantity of ion j in the particle. The task of **mass spectrum labeling** is to find all ions present in the particle as well as their quantities w_i, given an input spectrum. Formally, a **label** for an ion with respect to a given spectrum is an $< ion, quantity >$ pair;

* Profs. Gross and Musicant are at Carleton College, and the remaining authors are at University of Wisconsin-Madison. The contact email is raghu@cs.wisc.edu. Work supported by NSF ITR grant IIS-0326328.

a **label** for the spectrum is the collection of labels for all ions in the signature library. The task of labeling an input spectrum can be viewed as a search for a linear combination of ions that best approximates the spectrum, and the success that is achievable depends on the extent of unknown ions. In Sections 3 to 5, for simplicity we assume that the signature library is complete, i.e., there are no unknown ions. We evaluate the impact of unknowns in Sections 6 and 7.

3. When is labeling hard?

In this section, we formulate the labeling task as solving a set of linear equations, and then discuss the fundamental challenge involved: the interference between different combinations of signatures and the consequent ambiguity in labeling.

3.1. Linear system abstraction

We can represent the signature library $S = \{\vec{s}_1, \vec{s}_2 \cdots \vec{s}_n\}$ as a matrix $A = [\vec{s}_1, \vec{s}_2, ..., \vec{s}_n]$, where $\vec{s}_k$, the k^{th} column of A, is the signature of ion *k*. A spectrum label is an *n*-dimensional vector $\vec{x}$ whose j^{th} component $\vec{x}[j]$ indicates the quantity of ion *j* in the particle. Labeling consists of solving the linear system $A\vec{x} = \vec{b}$, $\vec{x} \geq 0$. Noticing that $A\vec{x} = \vec{b} \Rightarrow A(c\vec{x}) = c\vec{b}$ for any constant c, we can assume without loss of generality that $\vec{b}$ is normalized (i.e., $\sum_i \vec{b}[i] = 1$). By definition of signatures, each column of A also sums to 1. It follows immediately from this fact and $\sum_i \vec{b}[i] = 1$ that $\sum_i \vec{x}[i] = 1$. The exact quantities of all ions can be easily calculated by multiplying the quantity distribution vector $\vec{x}$ by the overall quantity of the particle, which is simply the sum of signal intensities over all m/z values in the original spectrum before normalization.

3.2. Uniqueness

Definition 1: An input spectrum $\vec{b}$ is said to have the **unique labeling** property with respect to signature library A if there exists a unique solution $\vec{x}_0$ to the system $A\vec{x} = \vec{b}$, $\vec{x} \geq 0$.

In general, given library A and input spectrum $\vec{b}$, neither existence nor uniqueness of solutions is guaranteed for the above equation. Our first result identifies a class of libraries for which every input spectrum is guaranteed to have a unique label.

Theorem 1: Consider signature library $A = [\vec{s}_1, \vec{s}_2, ..., \vec{s}_n]$ and a spectrum $\vec{b}$ where $\vec{s}_1, \vec{s}_2, ..., \vec{s}_n$ are linearly independent (i.e., there is no vector $\vec{a} = [a_1, a_2, ..., a_n]$ such that $\sum_{i=1}^{n} a_i \vec{s}_i$ and at least one $a_i \neq 0$). Then, either $\vec{b}$ has the unique labeling property w.r.t. *A*, or the system of equations (1) has no solution. □

Even if a signature library does not satisfy the conditions of Theorem 1, there may still be input spectra $\vec{b}$ for which the solution of (1) is unique, e.g. when

$$A = \begin{pmatrix} 0 & 1 & 1/2 \\ 1 & 0 & 1/2 \end{pmatrix} \quad \vec{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

there is a unique solution $\vec{x}^T = [0,1,0]$.

Conversely, for a given spectrum, there will typically be infinitely many solutions when the signature library does not satisfy the conditions of Theorem 1. Theorem 2 shows an important case with infinite solutions.

Theorem 2: Consider the signature library $A = [\vec{s}_1, \vec{s}_2, ..., \vec{s}_n]$ and a spectrum $\vec{b}$ where $\vec{s}_1, \vec{s}_2, ..., \vec{s}_n$ are *not* linearly independent. If there is a solution $\vec{x} = [x_1, x_2, ..., x_n]$ to $A\vec{x} = \vec{b}$, $\vec{x} \geq 0$ such that $min_{i=1,2,...,n} x_i > 0$, then $\vec{b}$ has infinitely many labels. □

3.3. Spectra with unique labeling

We now present our main theoretical result, which is an elegant characterization of the complete set of spectra that have the unique labeling property with respect to a given signature library. We explain the concept through an example and state a theorem that describes this set.

Suppose the signature library has only four signatures $\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4$. Figure 2(a) shows the case in which $\vec{s}_1, \vec{s}_2, \vec{s}_4$ are linearly dependent. All normalized spectra that can be represented as a conic combination (that is, a linear combination of the vectors $\vec{s}_1, \vec{s}_2, \vec{s}_3, \vec{s}_4$ in which the coefficients are nonnegative) form the triangle $\Delta s_1 s_2 s_3$ in this example. The ambiguity of the labeling comes from the linear dependency among $\vec{s}_1, \vec{s}_2, \vec{s}_4$, since $\vec{s}_4$ is itself a conic combination of $\vec{s}_1$ and $\vec{s}_2$. However, any point lying on the line $\overline{s_1 s_3}$ can be uniquely represented as a conic combination of s_1 and s_3. The intuitive reason for this is clear: Any involvement of a positive fraction of $\vec{s}_2$ or $\vec{s}_4$ (or both) will lift the point out of the line $\overline{s_1 s_3}$. Similarly, the points on the line $\overline{s_2 s_3}$ can be uniquely represented as a conic combination of $\vec{s}_2$ and $\vec{s}_4$. The case in which $\vec{s}_4$ combines all three vectors, $\vec{s}_1, \vec{s}_2, \vec{s}_3$ is shown in Figure 2(b). In this case, any point lying on the boundary of triangle $\Delta s_1 s_2 s_3$ can be uniquely represented as a conic combination of two signatures among $\vec{s}_1, \vec{s}_2, \vec{s}_3$.

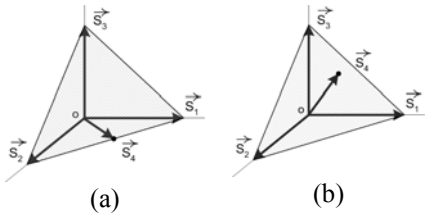

Figure 2: Vector space spanned by signatures

Definition 2: Given a signature library $S = \{\vec{s}_1, \vec{s}_2 \cdots \vec{s}_n\}$, the **convex hull** generated by S is defined as:

$$ch(S) = \{\sum_{i=1}^{n} w_i \vec{s}_i \mid n \geq 1, \sum_{i=1}^{n} w_i = 1, w_i \geq 0, \vec{s}_i \in S, 1 \leq i \leq n\}$$

The following theorem is a necessary and sufficient condition for an input spectrum to have a unique label with respect to a given signature library. The full proof is involved, and is included in an appendix; we provide a proof outline below.

Theorem 3: The set of spectra with the unique labeling property w.r.t. library *S* is the set of points in *ch(S)* that do not lie in the interior of the convex span of some affine dependent subset of *S*. Further, there is a polynomial time algorithm to test whether a given spectrum has a unique spectrum w.r.t. a library *S*.□

4. Handling ambiguity and errors

In practice, signal intensities are not precisely calibrated, and the background noise causes measurement errors and introduces uncertainty. We therefore introduce an error bound *E* and a distance function *D*, and recast the labeling problem in terms of mathematical programming, as an "exhaustive" feasibility task:

Seek all $\bar{a}$ such that $D(A\bar{a},\bar{b}) \le E, \bar{a} \ge 0.$ (1)

Given a library *A* with *n* signatures and input spectrum $\bar{b}$, the search space for problem (1) is an *n*-dimensional space. The **solution space** for input spectrum $\bar{b}$ is defined as follows:

Definition 3 **:** Given a signature library *A*, an input spectrum $\bar{b}$ and an error bound *E* with respect to distance function *D*, the **solution space** of spectrum $\bar{b}$ is $L_{\bar{b}} = \{\bar{a} \mid D(A\bar{a},\bar{b}) \le E, \bar{a} \ge 0\}$.

It is worth noting that the choice of the distance function *D* may affect the complexity of the problem significantly. We use *Manhattan Distance* (also known as ℓ_1 norm) as our distance measurement. The Manhattan distance between two vectors is defined as $\ell_1(\bar{\alpha},\bar{\beta}) = \sum_i |\alpha_i - \beta_i|$. With Manhattan distance, the solution set for (2) can be found using the following linear programming (LP) model:

$$\begin{aligned} &min \sum_i s_i \quad s.t. \\ &\quad A\bar{\alpha} - \bar{b} \le s, \quad A\bar{\alpha} - \bar{b} \ge -s \\ &\quad \alpha_i \ge 0, s_i \ge 0, i = 1,2,3,... \end{aligned} \tag{2}$$

We observe that if the distance function is convex, the solution space of an input spectrum $\bar{b}$ is convex (see below). We will explore this property further in Section 5; for now, we note that *Manhattan Distance* is a convex distance function.

Theorem 4: If the distance function *D* has the form $D(u,v) = d(u-v)$, where *d* is a *convex function*, then the solution space of the search described by Equation (1) is convex. □

4.1. Discretization

Even if an input spectrum has an infinite number of labels (for a given signature library) due to the ambiguity, in practice, we do not need to distinguish between solutions that are very similar. A natural approach to deal with a continuous space is to discretize it into grids, so that the number of possible solutions becomes finite.

Formally, a **threshold vector** $\bar{t} = [t_0, t_1, ..., t_d]$ divides each dimension of the search space into *d* ranges, where t_i and t_{i+1} are the lower bound and upper bound of range *i*. Given a threshold vector, we introduce the notion of **index vector** to represent a continuous subspace.

Definition 4: Given a threshold vector $\bar{t} = [t_0, t_1, ..., t_d]$, an **index vector** I=[(l_1, h_1) ...,(l_n, h_n)], $l_i < h_i, l_i, h_i \in Z$ represents a continuous subspace,

$$S_I = \{\bar{a} \mid \forall i, \bar{t}[l_i] < \bar{a}[i] \le \bar{t}[h_i], \bar{a}[i] \in R\} \tag{4}$$

Since an index vector represents a unique subspace, we will refer to a subspace simply by its corresponding index vector when the context is clear. Using the index vector representation, we in turn define the notion of cell.

Definition 5: A subspace $[(l_1,h_1),(l_2,h_2),...,(l_n,h_n)]$ is a **cell** if $\forall j, l_j + 1 = h_j$.

The cell is the finest granularity of the discretization, which reflects the degree of detail which users care about. A threshold vector $\bar{t} = [t_0, t_1, ..., t_d]$ divides the whole search space into d^n *cells*, where *n* is the number of dimensions (which is equivalent to the total number of signatures). Each cell also corresponds to a distinct *n*-dimensional integer vector

$$\bar{y} = [y_1, y_2, ..., y_n], 1 \le y_i \le d, y_i \in Z$$

which defines a subspace Y corresponding to the index vector $[(y_1, y_1+1),(y_2, y_2+1),...,(y_n, y_n+1)]$.

4.2. Optimization model

We now redefine the task of **spectrum labeling** as follows: *Find all the cells that intersect the solution space of the input spectrum.* A **label** of spectrum $\bar{b}$ is then simply an integer vector $\bar{x}$ representing a cell that intersects the solution space of $\bar{b}$. All such integer vectors form the **label set** of spectrum $\bar{b}$. Formally,

Definition 6: A vector $\bar{x} = (x_1, x_2, ..., x_n)$, $x_i \in [0...d-1]$ is a **label** of spectrum $\bar{b}$ if the subspace defined by the index vector $X = [(x_1, x_1+1),(x_2, x_2+1),...,(x_n, x_n+1)]$ intersects the solution space of spectrum $\bar{b}$. In the other word, $\bar{x}$ is the label if $\exists \bar{\alpha}$, s.t $D(A\bar{\alpha},\bar{b}) \le E$, $\bar{t}[\bar{x}_i] < \alpha_i \le \bar{t}[\bar{x}_i + 1]$. $\bar{b}$'s **label set** is $L = \{\bar{x} \mid \bar{x}$ is a label of $\bar{b}\}$.

To simplify the discussions in the following sections, we also introduce the notion of **feasible space** to describe a subspace that intersects the solution space of the input spectrum. A feasible space is a collection of one or more cells. If the *feasible space* is a cell, it is also called a *label*. Table 1 summarizes our notations and model.

Figure **Error! Bookmark not defined.** illustrates the concepts discussed in this section. Suppose there are only two signatures in the signature library. The whole search space is a two dimensional space *ABCD* within which $S_1S_2S_3S_4$ forms the solution space of an input spectrum. It intersects the *cells LFGM* and *MGHA*, each of which corresponds to a *label*. Subspace *ALFH* intersects with the solution space, so it is a *feasible space*. *MBEG* is also a *feasible space.*

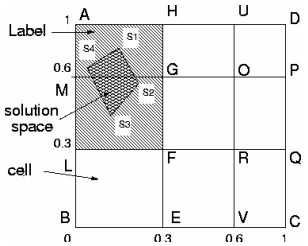

Figure 3: Illustration of concepts

Table 1: Operational definitions of labeling

Notations	$\bar{x}$	An *n*-dimensional integer vector, $0 \le x_i < d$
	$\bar{b}$	Normalized input mass spectrum
	$\bar{t}$	Threshold vector for discretization
	d	Number of ranges per dimension
	L	Label set of input spectrum
	A	Signature library with n signatures
	D	Distance function
	E	Error bound

$L \leftarrow \varnothing$

for each possible *x*, *Seek* $\bar{a}$ s.t.

$D(A\bar{\alpha},\bar{b}) \le E$

$\bar{t}[j] < \bar{a}[i] \le \bar{t}[j+1], j = \bar{x}_i$

if exists such $\bar{\alpha}$, $L = L \bigcup \{\bar{x}\}$

5. Labeling Algorithms

In Section 4, we showed that given n signatures and discretization granularity d, the search space contains d^n cells. A brute force approach that tests the feasibility of each cell is not practical, considering that there are hundreds of signatures. In this section, we propose two algorithms: DFS is a general algorithm which works for any distance function, and Crawling algorithm exploits convexity property of distance functions.

5.1. Feasibility test

Given a subspace S, we use the algorithm shown in Table **Error! Bookmark not defined.** to test the feasibility of the subspace; this module is the building block of the later algorithms. Notice that for each test; exactly one LP call is invoked.

Table 2: Test the feasibility of a subspace

Input:	$\vec{b}$	Input mass spectrum
	E	Error bound
	$\vec{t}$	Threshold vector
Output:	TRUE if the subspace is feasible, FALSE otherwise	

Is_feasible(subspace S)

Seek $\vec{a}$ s.t.

$D(A\vec{a},\vec{b}) \le E$ (*)

$\vec{t}[l_j] < \vec{a}[j] \le \vec{t}[h_j],\ \text{for } j = 1,2,...,n$

$[(l_1,h_1),(l_2,h_2),...,(l_n,h_n)]$ is the index vector of subspace S

if (*) succeeds, return TRUE, otherwise return FALSE

5.2. Depth-First Search (DFS) algorithm

We first state an important property of subspace feasibility which guarantees the correctness of the DFS algorithm. The proof is straightforward and is omitted.

Table 3: DFS Algorithm

Input:	$\vec{b}$	Input mass spectrum
	E	Error bound
	$\vec{t}$	Threshold vector
Output:	Label set $L(\vec{b})$	

Depth_First_Search(subspace S)

```
L ← ∅
if not Is_feasible(S) then
   return ∅
else
   if S is a cell then
      L ← label corresponding to S;
      return L;
   else
      pick_dimension(j)
      {S_i} = split_subspace(S, j)
      for each S_i
         L ← L ∪ Depth_First_Search( S_i )
      return L;
```

Main : *Depth_First_Search(whole search space W)*

Theorem 5: Let a spectrum $\vec{b}$ and a signature library A with n signatures be given. If subspace S is feasible, then any subspace T, with $S \subset T$ is also feasible.□

The DFS labeling algorithm explores a search tree in which each node is associated with a particular subspace, and the subtree rooted at that node corresponds to the subsets of that subspace. At each node, the algorithm first tests the feasibility of the subspace for that node. If not feasible, that node and its subtree are pruned. Otherwise, we select a dimension j that has not been subdivided to the finest possible granularity in the subspace of that node, and divide the subspace further along dimension j. Each smaller space created thus corresponds to a child of the current node.

In Table **3**, the *pick_dimension* method chooses a dimension (which is not already at the finest granularity possible) to split the current subspace and *split_subspace* divides the current subspace into smaller pieces along the chosen dimension. Details of these two methods are discussed in [10].

The correctness of DFS algorithm is proved in [10]. In [7] we show that the complexity of the DFS algorithm is *O(knd)*

5.3. Crawling algorithm

DFS is a general algorithm in which we can use any distance function D, even one that is non-convex. The Crawling algorithm requires the distance function to be convex and exploits the *connectivity property*[1] derived from the convexity of solution spaces, as described in Theorem 4.

Connectivity Property: Given two labels l_1 and l_n, there exists a path of connecting labels (l_1 ,…, l_{i-1} , l_i ,… l_n) in which l_{i-1} , l_i are adjacent, i.e., differ only in one dimension by 1.

The Crawling algorithm first finds a solution to the linear system $D(A\vec{a},\vec{b}) \le E, \vec{a}[i] \ge 0$ by invoking one LP call. The cell that contains solution is a label and is used as the start point to explore other connected cells in a breath-first fashion. If the cell discovered has not been visited before and is a label, its neighbors will be explored subsequently. Otherwise, it is discarded and no further exploration will be incurred by it. The algorithm stops when all labels and their neighbors are visited. The connectivity property guarantees that all labels are connected to the first label we found and can be discovered by "crawling" from that start point. Due to lack of space, we omit details of the algorithm; see [10], which also contains a correctness proof and shows that the time and space complexity are *O(k*n).

Let's take Figure 3 in Section 4 again as an example. The input spectrum's label set contains two labels which are the cells in shade. The crawling algorithm first finds a solution point. Suppose it falls in cell *LFGM*. It then starts from *LFGM* and explores its neighbors *LBEF*, *FROG* and *MGHA*. Among the three, only cell *MGHA* is a label and will incur further exploration. It has only two adjacent cells. One is already visited and the other is not a label. Thereby the algorithm terminates and outputs *LFGM* and *MGHA* as the input spectrum's label set.

Theorem 6 Given an input spectrum $\vec{b}$, a signature database $A = [\vec{S}_1, \vec{S}_2, ..., \vec{S}_n]$ and a threshold vector $\vec{t} = [t_1, t_2, ..., t_{d+1}]$, suppose the number of labels for $\vec{b}$ is k, the Crawling algorithm will find the complete set of the labels for input spectrum. □

[1] Convexity is actually stronger than the connectivity property.

Theorem 7: Given an input spectrum $\vec{b}$, a signature library $A=[\vec{s}_1,\vec{s}_2,...,\vec{s}_n]$ and a threshold vector $\vec{t}=[t_0,t_1,...,t_d]$, suppose the number of labels for $\vec{b}$ is k, then the number of LP calls invoked by the Crawling algorithm is *O(kn)*. The number of index vectors stored in the queue is *O(kn)*. □

6. Data generation

There is a fundamental difficulty in evaluating algorithms for labeling mass spectra: manual labeling of spectra (to create training sets) is laborious, and must additionally be cross-validated by other kinds of co-located measurements, such as traditional filter-based or "wet chemistry" techniques. For any given application, rigorously establishing appropriate "ground truth" datasets can take months of field-work. In this section, we describe a detailed approach to synthetic data generation that allows us to use domain knowledge to create signature libraries and input particle spectra that reflect specific applications and instrument characteristics.

Our generator has two parts: generation of the signature library, and generation of input spectra. We begin with a collection of real ion signatures, and select a set of *n* linearly independent signatures to serve as "seeds". New signatures are generated using a non-negative weighted average of seed signatures. The set of all generated signatures is partitioned into two sets: the *signature library*, and the *unknowns*.

The generation of new signatures for the signature library is done in "groups" as follows, in order to control the degree of (non-)uniqueness, or ambiguity. Each group consists of two "base" signatures from the seeds (chosen such that no seed appears in multiple groups) plus several "pseudo-signatures" obtained using non-negative weighted averages of these two signatures. The generated signatures in each group are effectively treated as denoting new ions in the signature library. Of course, they do not correspond to real ions at all; rather, they represent ambiguity in that it is impossible to distinguish them from the weighted average of base signatures used to generate them when labeling an input spectrum that contains ions from this group. Intuitively, the larger the size of a group, the greater the ambiguity in input spectra that contain ions from the group; observe that interference can only occur within groups. We create a total of *k* groups with *i+1* pseudo-signatures in group *i*.

The set of *n* original signatures plus the $(k+3)\cdot k/2$ pseudo-signatures generated as above constitute our "universe" of all signatures. Next, we select some of these signatures to be unknowns, as follows: We randomly select one signature from each of the *k* groups; these *k* signatures are "interfering unknowns". We also randomly select *u-k* seed signatures that were not used in group generation; these *u-k* signatures are "non-interfering unknowns", giving us a total of *u* unknowns.

The second part of our generator is the generation of input spectra. An input spectrum is generated by selecting *m* signatures from the universe of signatures and adding them according to a weighting vector $\vec{w}$. Ambiguity and unknowns are controlled by the careful selection of signatures that contribute to the spectrum, and the input weighting vector controls the composition of the spectrum as well as the contribution of unknowns. We observe that the effect of many unknowns contributing to an input spectrum can be simulated by aggregating them into a single unknown signature with an appropriate weighting vector; accordingly, we use at most a single unknown signature. Table 4 summarizes the parameters for spectrum generation.

Table 4: Parameters used for spectrum generation

m	number of signatures
q	number of groups
$\vec{w}$	vector for the weight of the signatures
o	whether the unknown signature is used
g	average amount of noise

We begin by randomly selecting two signatures from group *q*. Then, if unknowns are desired in the generated spectrum (*o*=1), we choose either the q^{th} unknown signature, or a randomly selected non-interfering unknown signature, depending on whether or not the unknown is desired to interfere with known ions in the spectrum (*v* = 1 or 0). The contribution of unknowns is controlled by the last component of the weighting vector. Next, we randomly select signatures from the signature library that do not belong to any of the *k* "groups" to get a total of *m* signatures. These signatures are linearly independent seeds, and thus the ambiguity of the generated spectrum will depend solely on the first 2 (or 3, if an interfering unknown is chosen) signatures.

Finally, we select values for *m* random variables following a normal distribution whose means are given by the weighting vector of arity *m*. The values for these variables are used as the weights w_i to combine the *m* signatures: $\sum_{j-1}^{m} w_j s_j$. (We note that when an unknown signature is used in the generation, the last element of the weighting vector is reset to be the relative quantity of the unknown signature and the whole weighting vector is normalized to sum up to 1.)

We account for noise by simply adding a noise value (a random variable following a normal distribution) to each component (i.e., m/z position) of the generated spectrum.

7. Experimental results

We now describe experiments to evaluate our labeling algorithms with respect to both quality and labeling speed. To give the reader an idea of the speed, we observed an average processing rate of about one spectrum per second when we ran our algorithms on over 10,000 real mass spectra collected using an ATOFMS instrument in Colorado and Minnesota; this is adequate for some settings, but not all, and further work is required. Speed and scalability are not the focus of this paper, but are addressed in [7], and extensive experiments are reported. We also tested the accuracy of our labeling algorithm against a small set of manually labeled spectra; all were correctly labeled by the algorithm. Admittedly, this is not an extensive test, but we are limited by the fact that manual labeling is a tedious and costly process. (This underscores the importance of not requiring training datasets.)

In this section, we therefore evaluate our algorithms using the data generator from Section 6; this approach also allows us to study the effect of ambiguity, unknown signatures and noise levels in a controlled fashion. For comparison, we also evaluated machine learning (ML) algorithms. However, the reader should note that our algorithms can label input spectra *given only the signature library,* whereas the ML approaches require extensive training datasets, which is unrealistic with manual labeling. In addition, the ML algorithm ignores equivalent alternatives and only generates one label. Nonetheless, we propose two different quality measures and include the comparison for completeness, and to motivate a promising direction for future work, namely the development of hybrid algorithms that combine the strengths of these approaches.

7.1. Machine learning approach

Our ML algorithm builds upon WEKA classifiers [18]. For each signature in the signature library, we train a classifier to take a given input spectrum and output a presence category (*absent, uncertain* or *present*), i.e., to detect whether or not the ion represented by the signature is present in the particle represented by the spectrum. The predictive attributes are the (fixed set of) m/z locations, taking on as values the signal intensities at these locations in the input spectrum. To label a spectrum, we simply classify it using the classifiers for all signatures in the library. When we are only interested in the presence of a subset of ions, of course, we need only train and run the classifiers for the corresponding signatures. We evaluated four types of classifiers: Decision Trees (J48), Naïve Bayes, Decision Stumps, and Neural Networks. Decision Trees consistently and clearly outperformed the other three, and we therefore only compare our algorithms against this approach in the rest of this section.

7.2. Datasets

The (common) dimension of all signatures and spectra is set to be 255. We used n=78 base signatures of real ions, and generated k=5 groups containing 2 to 6 pseudo-signatures respectively. Including the original 78, we thus obtained 98 signatures, 15 of which were withheld as unknown; the remaining 83 comprised the signature library. For generating input spectra, we set the number of signatures used for spectrum generation to be m=10. The relative proportion of these m signatures was controlled by the weighting vector [0.225, 0.2, 0.2, 0.1, 0.1, 0.06, 0.06, 0.03, 0.01, 0.01].

We generated five testing datasets with controlled ambiguity, unknown signature and noise levels. Each dataset contains several files, each of which contains 1,000 spectra generated by using the same set of parameter values. Dataset 1 is designed to test the effect of noise. It consists of 10 files. Each file corresponds to a distinct noise level from 0% to 360% of a preset error bound, which is 0.01 of the total intensity. No ambiguity or unknown signature is involved in this dataset. Dataset 2 tests the effect of ambiguity. It consists of 5 files corresponding to 5 ambiguity levels. Dataset 3 has no noise or ambiguity, but contains some non-interfering unknown signatures. This dataset contains ten files, with the weight on the unknown signature varying from 0% to 180% of the preset error bound. Dataset 4 is identical to Dataset 3 except that the unknown signatures selected are interfering unknowns. Dataset 5 is designed to test the combined effect of noise and ambiguity. Five ambiguity degrees used in Dataset 2 and five noise levels selected from the 10 noise levels used in Dataset 1 result in 25 different combinations of noise and ambiguity, and 25 files are generated for each such combination. The discretization criteria used for all the datasets above is controlled by a threshold vector [0, 0.08, 0.18, 1], which indicates *absent, uncertain* and *present* respectively.

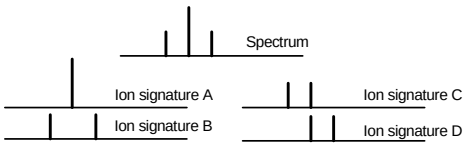

Figure 4: Indistinguishable spectrum labels

7.3. Labeling quality

Given a particle, consider the "*ideal*" version of its spectrum obtained by eliminating noise and unknowns, and is therefore strictly the weighted sum of known ion signatures present in the particle. Even such an ideal mass spectrum might not have unique labels. The spectrum shown in Figure 4 might represent a particle that contains ions A and B, or a particle that contains C and D. Given only the input spectrum, the combinations AB and CD are mathematically indistinguishable, and should be presented to domain experts for further study. The complete set of such "*indistinguishable spectrum labels*" for the "*ideal*" version of an input spectrum is the best result we can expect from labeling; we call each label in this set a **correct** label. Intuitively, it is the set of all feasible combinations of ions in the particle. This is exactly the *label set* of the ideal spectrum defined in Section 4 (with the error bound set to 0). By Theorem 7, our algorithms generate this label set when no unknown or noise is present, i.e., the ideal version is the given input spectrum itself. However, as noise and unknowns are added, the labels found by our algorithm will no longer be the same as the desired set of all correct labels.

Our first proposed metric comparing the result of a labeling algorithm with the set of all correct labels. This metric consists of two ratios: the *hit ratio* and *false ratio*. The *hit ratio* is the percentage of correct labels in the result set of the labeling algorithm. The *false ratio* is the proportion of labels in the result set that are not correct labels. Formally, let the *label set* of a particle's *ideal spectrum* be L_T and let the set of labels found by a labeling algorithm for the particle's real spectrum under the presence of noise and unknowns be L_O :

$$Hit\ Ratio = |L_T \cap L_O| / |L_T| \qquad False\ Ratio = |L_O - L_T| / |L_O|$$

Experiments under this metric will be called *full labeling tests,* and are presented in Section 7.3.1.

Our second metric relaxes the requirement of finding the correct combinations of ions, and focuses on the proportion of individual ions whose presence or absence is correctly classified. Given a collection of *interesting ions*, we aggregate the set of correct spectrum labels to obtain a set of ion labels IL_T as follows: An ion of interest is marked *present* if all correct labels mark it as *present*, *absent* if all correct labels mark it as *absent*, and marked *uncertain* in all other cases. Similarly, we can obtain a set of ion labels IL_O from the result set of the labeling algorithm. Our second metric consists of two ratios based on these ion labels:

$$Partial\ Hit\ Ratio = |IL_T \cap IL_O| / |IL_T|$$

$$Partial\ False\ Ratio = |IL_O - IL_T| / |IL_O|$$

Partial hit ratio is similar to *hit ratio*, and describes the percentage of ions that are correctly labeled, while *partial false ratio* is the proportion of ions that are incorrectly labeled. Experiments under this second metric will be called partial *labeling tests,* and are presented in Section 7.3.2.

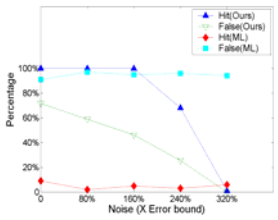

Figure 5: Effect of noise w/o ambiguity

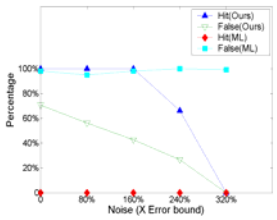

Figure 6: Effect of noise with ambiguity

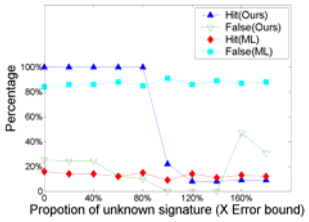

Figure 7: Effect of non-interfering unknown

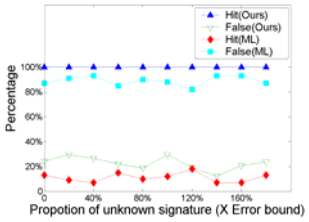

Figure 8: Effect of interfering unknown

It is worth noting that for any given spectrum, our algorithms will generate exactly the same result. Therefore, for quality evaluation, we simply refer to them as "our algorithm" or the "LP" algorithm, since they both build upon linear programming.

7.3.1. Full labeling tests In the following graphs, each data point for our algorithm is the average of results on 1000 spectra, while each data point for the ML algorithm is the result of 5-fold cross validation on the same dataset. Figure **5** shows the result on Dataset 1, which contains no ambiguity or unknowns. Even in this simple case, the ML algorithm performs poorly. Its hit ratio is close to zero while the false ratio is close to one. In contrast, our algorithm shows great strength in identifying a possible combination of ions to explain the spectrum. The hit ratio remains almost perfect when the noise is within 180% of error bound, but drops sharply when noise grows above that threshold. This shows a limitation of our algorithm: *the error bound is the only component that accounts for noise*, and our results are sensitive to the choice of the error bound relative to noise levels. While the error bound helps in accounting for noise, it also introduces a degree of freedom that allows incorrect labels to be included. Surprisingly, the *false ratio*, which measures the percentage of incorrect labels in the result, actually goes down as the noise level increases; the noise intuitively takes up the slack introduced by error bound. This observation suggests that we might be able to automatically tune the error bound by estimating the noise level. Figure **6** shows the results on Dataset 5, which contains both ambiguity and noise. As we can see, the already low hit ratio of the ML algorithm drops further, essentially to zero, and the false ratio goes over 95%. Our algorithm performs consistently well in Figure **6**, demonstrating its ability to handle ambiguity even in the presence of noise. Figures **7** and **8** summarize the experimental results on Datasets 3 and 4, which show the effect of unknowns. Intuitively, if the unknown ion is non-interfering, it acts like additional noise at some m/z positions, which makes it harder to compensate for. The hit ratio of our algorithm drops sharply when the non-interfering unknown proportion exceeds the error bound. The spike in the *false ratio* at the very end is an artifact caused by the fact that the number of labels found is reduced to one essentially, and that one is incorrect. The effect of interfering unknowns is more interesting. While it raises the false ratio as more and more unknowns are added, as expected, surprisingly, it also helps the hit ratio (because it can be interpreted as some linear combination of known signatures that effectively increases the quantity of the known signatures).

7.3.2 Partial labeling tests We run the exact same set of experiments as for the Full Labeling Test, but apply the second metric. The ten signatures of interest are set to be those used to generate the spectrum, so that ambiguity w.r.t. the signatures of interest is still under control. Figures **9** and 10 illustrate the effect of noise and unknowns combined with ambiguity. The figures only show *hit ratio*, since in our setting the *false ratio* is just *1-hit ratio*. In both graphs, the triangle series show the hit ratio of our algorithm and the square/diamond series represent the ML algorithm. Solid lines represent the results on datasets with no ambiguity while dotted lines represent a dataset with ambiguity. The first observation is that the ML algorithm achieves decent performance under this metric, although it is still uniformly dominated by the LP algorithm. The performance degradation of the ML algorithm from diamond curves to square curves in both graphs again shows the weakness of the ML approach, namely its *inability to handle ambiguity*. Both noise and unknowns have a similar effect on our algorithm as in the full labeling tests. On the other hand, the almost horizontal *hit ratio* curves for the ML algorithm illustrate an interesting point: *the ML algorithm tends to be less sensitive to unknowns than our algorithm.* This is because our algorithm assumes complete knowledge of ion signatures and tries to combine all signatures simultaneously, whereas the ML algorithm simply looks at one ion at a time.

Overall, our algorithm clearly beats the ML algorithm in terms of labeling quality, even in partial labeling tests. In addition, the ML algorithm needs substantial training data. This is not realistic to get at all. However, the ML algorithm does show promise in partial labeling, which suggests a promising research direction, namely a hybrid algorithm that combines the speed of ML and the ambiguity-handling ability of our LP-based approach.

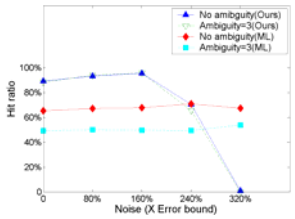

Figure 9: Effect of noise

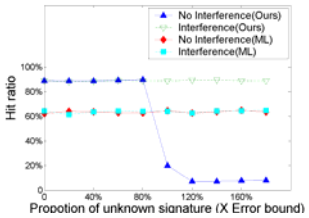

Figure 10: Effect of unknowns

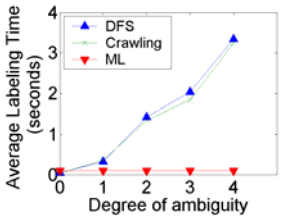

Figure 11: Effect of ambiguity on label time

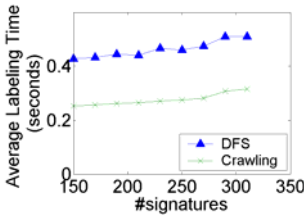

Figure 12: Scalability w.r.t #signatures

7.3.3 Labeling speed We ran efficiency tests on the five datasets described in Section 7.2. Results show that the presence of noise and unknown signature does not affect the performance of our algorithms much, unless the noise or weight on the non-interfering unknown signature is significantly larger than the error bound. When no ambiguity is present, labeling takes about one second for both DFS and Crawling algorithms. However, as more ambiguity is included and the label set size increases sharply, the performance of our algorithms degrades significantly. Figure **11** shows the running time of our algorithms on Dataset 2, which contains five files of spectra with five different degrees of ambiguity. Series 1 and 2 show the performance of DFS and Crawling algorithms. The Crawling algorithm exploits the convexity of the distance function and runs slightly faster than DFS, but both become much slower as

ambiguity is increased. This is mainly due to the dramatic increase in the number of correct labels. The ML algorithm is much faster than our algorithms, but it is worth noting that when no ambiguity is involved and the number of correct labels is small, the running time of our algorithm is almost the same as for ML. In addition, the training time of the ML approach is not reflected at all in these graphs. Further, when we are only interested in detecting a small number of signatures, we can revise our DFS algorithm to only pick the signatures of interest and do partial labeling. This optimization greatly speeds up DFS, to about 100 spectra per second. Figure **12** summarizes the results of algorithm scalability with respect to the number of signatures in the signature library.

7.4. Labeling spectra from a real application

We now present results on data from a real application, comparing our labeling results with manual labeling by a domain expert. The spectra in our experiment come from particles collected in a diesel engine test. Most of the ions in our library are inorganic or simple organic. The signatures of most of ions have a single major peak, i.e., for signature $\bar{s} = \{ I_1, I_2, ... \}$, there exist I_i such that $I_i \gg I_j$, for all $j \neq i$. Hence, most of the ambiguity in the signature library comes from ions which have their major peaks at the same m/z value, although some of the signatures, such as Hg and $TEANO_3$, do have multiple peaks.

We used both our algorithms to label a set of 85 input spectra, with identical results. The labels were evaluated by a chemist who is studying the spectra. In this specific application, the goal is to detect all present ions rather than to quantify their abundance. Our algorithms performed remarkably well, correctly detecting the ions in 93% of the spectra.

When our labeling algorithms failed, it was due to one of three reasons. First, there are ions which exist in the spectra but whose signatures are not included in the signature library. Our algorithms can tolerate some degree of "unknown" ions, in that unknown and "uninteresting" ions do not (usually) prevent us from identifying ions of interest (i.e., in the library). However, if these ions are of interest to the scientist and must be identified when present, we require that they be included in the signature library. Overcoming this limitation requires us to detect certain "missing" signatures by comparing labels from multiple spectra, and is a direction for future work; such a step is currently not included in our model. Even when we do include the correct label as an alternative, it is important to be able to identify the correct label and to distinguish it from the alternatives that arise due to ambiguity. Again, further work is needed in this area.

The second problem is related to the ambiguity of the signature library. Some ions in the library have exact the same signature. It is impossible to distinguish these ions without integrating domain knowledge. This is another important direction for improvement.

The third problem with our labeling result is peak "drifting". Due to the interaction between different ions in the chamber of the mass spectrometer, the peak in the spectrum is actually not a spike that stands on one unique m/z value. Instead, it is a curve that is distributed over multiple m/z values. Our current model is sensitive to this type of error, and additional work is needed.

8. Related work and conclusion

Methods of categorizing aerosol particles using clustering and neural networks have been proposed [1,12,9,16], but none of them deals with the labeling problem directly. The linear programming method used in this paper is standard, see, e.g., [13]. Related nonlinear or integer programming tasks arising from the use of Euclidean distance or discretization are also well studied in the optimization community [6,13,19]. Recent work on knowledge-based optimization and machine learning [8,17] are promising extensions to the framework we propose. Machine learning methods such as clustering [5,20] can be applied to our basic linear programming approach by helping identify better initial points and optimization constraints.

Our future work includes finding better labeling algorithms, utilizing domain knowledge in the labeling process, discovering unknown signatures and validating our algorithm on real data. Interested readers can check the technical report [10] for detailed discussion.

9. References

[1] Agrawal, R., Imielinski, T., Swami, A., Mining Associations between Sets of Items in Massive Databases, *Proc. ACM-SIGMOD*, 1993.

[2] Agrawal, R., Faloutsos, C. and Swami, A. Efficient Similarity Search in Sequence Databases. *FODO*, 1993

[3] Agrawal, R., Mannila, H., et. al., Fast Discovery of Association Rules, *Advances in Knowledge Discovery and Data Mining*, 1995.

[4] Agrawal, R. and Srikant R.: Fast Algorithms for Mining Association Rules, *Proc. VLDB*, 1994

[5] Basu, Sugato, Banerjee, Arindam and Mooney, Raymond J., Semi-supervised Clustering by Seeding. *Proc. ICML,* 2002.

[6] Benson, Steven J., More, Jorge J, A Limited Memory Variable Metric Method, in *Subspaces and Bound Constrained Optimization Problems*, 2001.

[7] Chen, L., Huang, Z. and Ramakrishnan, R., Cost-Based Labeling of Groups of Spectra, *Proc. ACM-SIGMOD*, 2004.

[8] Fung, G., Mangasarian, O.L. and Shavlik, J., Knowledge-Based Support Vector Machine Classifiers. *Proc. NIPS 2002*.

[9] Gard, E., Mayer J.E., et. al., Real-Time Analysis of Individual Atmospheric Aerosol Particles: Design and Performance of a Portable ATOFMS, *Anal. Chem.* 1997, 69, 4083-4091.

[10] Huang, Z., Chen, L., et. al., Spectrum Labeling: Theory and Practice, 2004, Technical Report, UW-Madison.

[11a] Jayne, J.T., D.C. Leard, X. Zhang, et. al., Development of an aerosol mass spectrometer for size and composition analysis of submicron particles, *Aerosol Sci. Tech..,* 2000, 33, 49-70.

[11] McCarthy, J., Phenomenal data mining, *In Communications of the ACM 43 (8)*, 2003

[12] Noble, C.A. and Prather K.A., Real-time Measurement of Correlated Size and Composition Profiles of Individual Atmospheric Aerosol Particles. *Environ. Sci. Technol,* 1996

[13] Nocedal, J. and Wright, S.J., Numerical Optimization, *Springer*, 1st edition, 1999.

[14] Prather, K.A., Nordmeyer, T., and Salt, K. Real-time Characterization of Individual Aerosol Particles Using ATOFMS. *Anal. Chem.*, 1994; 66, 1403-1407.

[15] Srikant, R. and Agrawal, R., Mining Quantitative Association Rules in Large Relational Tables, *Proc ACM-SIGMOD*, 1996.

[16] Suess, D.T. and Prather K.A., Mass Spectrometry of Aerosols, *Chemical Reviews*, 1999, 99, 3007-3035.

[17] Towell, G.G. and Shavlik J., Knowledge-Based Artificial Neural Networks. *Artificial Intelligence*, 1994.

[18] Witten, Ian H. and Frank, Eibe, Practical Machine Learning Tools and Techniques with Java Implementation, *Morgan Kaufmann*, 1999.

[19] Wolsey, L., Integer Programming, John Wiley, 1998.

[20] Zhang, T., Ramakrishnan, R. and Livny M., BIRCH: An Efficient Data Clustering Method for Very Large Databases, *Proc. ACM-SIGMOD*, 1996.