Bioinformatics, 35, 2019, i164–i172
doi: 10.1093/bioinformatics/btz319
ISMB/ECCB 2019

ADAPTIVE: leArning DAta-dePendenT, conclse molecular VEctors for fast, accurate metabolite identification from tandem mass spectra

Dai Hai Nguyen[1,*], Canh Hao Nguyen[1] and Hiroshi Mamitsuka[1,2]

[1]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan and [2]Department of Computer Science, Aalto University, Espoo, Finland

*To whom correspondence should be addressed.

Abstract

Motivation: Metabolite identification is an important task in metabolomics to enhance the knowledge of biological systems. There have been a number of machine learning-based methods proposed for this task, which predict a chemical structure of a given spectrum through an intermediate (chemical structure) representation called molecular fingerprints. They usually have two steps: (i) predicting fingerprints from spectra; (ii) searching chemical compounds (in database) corresponding to the predicted fingerprints. Fingerprints are feature vectors, which are usually very large to cover all possible substructures and chemical properties, and therefore heavily redundant, in the sense of having many molecular (sub)structures irrelevant to the task, causing limited predictive performance and slow prediction.

Results: We propose ADAPTIVE, which has two parts: learning two mappings (i) from structures to molecular vectors and (ii) from spectra to molecular vectors. The first part learns molecular vectors for metabolites from given data, to be consistent with both spectra and chemical structures of metabolites. In more detail, molecular vectors are generated by a model, being parameterized by a message passing neural network, and parameters are estimated by maximizing the correlation between molecular vectors and the corresponding spectra in terms of Hilbert-Schmidt Independence Criterion. Molecular vectors generated by this model are compact and importantly adaptive (specific) to both given data and task of metabolite identification. The second part uses input output kernel regression (IOKR), the current cutting-edge method of metabolite identification. We empirically confirmed the effectiveness of ADAPTIVE by using a benchmark data, where ADAPTIVE outperformed the original IOKR in both predictive performance and computational efficiency.

Availability and implementation: The code will be accessed through http://www.bic.kyoto-u.ac.jp/pathway/tools/ADAPTIVE after the acceptance of this article.

Contact: hai@kuicr.kyoto-u.ac.jp

1 Introduction

Metabolites are small molecules, having many important functions in living cells such as energy transport, signaling, building blocks of cells and so on (Wishart, 2007). Identifying their biochemical characteristics or so-called metabolite identification is an essential task in metabolomics to increase the knowledge of biological systems. Yet, it is still a challenging task due to the size or coverage of spectra libraries.

Mass spectrometry (MS) is one of the most common techniques in analytical chemistry for dealing with metabolite identification (de Hoffmann and Stroobant, 2007). In more detail, a chemical compound is decomposed into fragments, of which mass-to-charge ratios (m/z) are continuously measured to obtain a mass spectrum. One MS spectrum can be represented by a list of peaks, each of which corresponds to a fragment captured by MS. Figure 1 shows a real example of a MS spectrum. In practice, tandem MS (also known as MS/MS or MS2) is widely used, in which precursor ions of specific m/z values from MS spectra are selected and further fragmented to produce other groups of product ions [see, e.g. Vaniya and Fiehn (2015) for more details]. The MS/MS spectra provide structural information about the measured compound, which makes MS/MS more useful for tackling metabolite identification.

© The Author(s) 2019. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

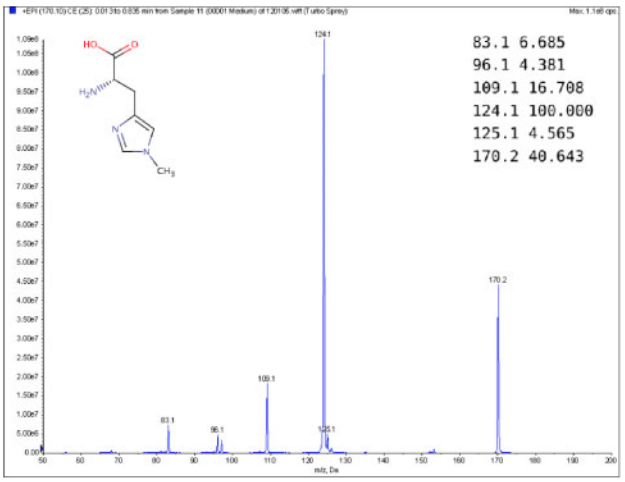

Fig. 1. Example MS spectrum from Human Metabolome Database (Wishart *et al.* 2013) for 1-Methylhistidine (HMBD00001), with the corresponding chemical structure (top-left) and peak list (top-right)

A number of computational methods have been proposed for identifying unknown metabolites from MS/MS spectra data. In general, they are classified into three main categories: (i) spectral library search; (ii) *in silico* fragmentation; and (iii) machine learning (Nguyen *et al.*, 2018a). Recent advances in metabolite identification have been led by the machine learning category (e.g. Brouard *et al.*, 2016; Dührkop *et al.*, 2015; Nguyen *et al.*, 2018b). This category can be further divided into two key groups: supervised learning for substructure prediction and unsupervised learning for substructure annotation. While the former is to find a mapping from inputs (e.g. spectra) to outputs (e.g. fingerprints), the latter extracts underlying substructures of metabolites. Our research focuses on supervised learning, where the common scheme is to learn a mapping from spectra to structures.

The prediction can be divided into two steps: (i) fingerprint prediction: predicting fingerprints of a given test spectrum with supervised learning; (ii) candidate retrieval: retrieving chemical compound (from database) which is closest to the predicted fingerprints (Nguyen *et al.*, 2018b).

Kernel methods have been shown to be effective for fingerprint prediction, such as methods include FingerID (Heinonen *et al.*, 2012), CSI:FingerID (Dührkop *et al.*, 2015) and input output kernel regression (IOKR, Brouard *et al.*, 2016). In particular, IOKR is recognized as the current cutting-edge method for metabolite identification due to the following advantages: (i) structures (e.g. feature interaction in the molecular fingerprint vectors) in the output can be incorporated into the learning model by the kernel defined in the output space, leading to accuracy improvement; (ii) fingerprints are simultaneously predicted by the learned model, rather than being considered as a set of separate tasks, resulting in faster computation. One can take structures of the metabolites into account by using graph kernels (path, shortest-path and graphlet kernels) or kernels defined on molecular fingerprints. It is also known that kernels based on fingerprint vectors obtained the best performance (Brouard *et al.*, 2016). However, a limitation of using molecular fingerprints as the intermediate representation vectors is that they are general-purpose and very large in size to encode all possible substructures and chemical properties related with metabolites. Consequently, such vectors are neither necessarily specific to any task nor data, and therefore redundant in the sense that these vectors might contain information irrelevant to the task, resulting in limited predictive performance. Moreover, the large size of fingerprints causes slow prediction in the first step of the above two steps.

Generally, in machine learning, deep learning has been proven successful recently in many application domains. Deep learning is useful for regular data, say a table, in which rows are instances and columns are features, and vice versa. However, semistructured data, particularly graphs, for example, chemical (or biological) molecules, which are irregular types of data, are difficult to be used with deep learning. A number of research efforts have been devoted to applying deep learning to semistructured data, proposing models to learn representations of graphs, such as Duvenaud *et al.* (2015), Li *et al.* (2015) and Nguyen *et al.* (2017). Importantly, Gilmer *et al.* (2017) showed that a lot of research on graphs can be formulated in a unified model, namely message passing neural network (MPNN), with the following three components: *message passing*, *update* and *readout* functions. In other words, one way of defining such functions results in a different model for learning graphs. Furthermore, another attractive property of MPNN is that it allows to learn meaningful representations specific to each task for graphs in an end-to-end manner.

We propose a powerful machine learning framework for metabolite identification, named ADAPTIVE, which has two subtasks: (i) learning a mapping from structures to molecular vectors and (ii) learning a mapping from spectra to molecular vectors. Figure 2 shows a schematic picture of ADAPTIVE, where the left and right blue boxes correspond to the first and second subtasks, respectively. In Subtask 1, ADAPTIVE learns a model to generate molecular vectors for metabolites using their chemical structures, where these vectors are specific to both data and the task of metabolite identification, and therefore nonredundant. The model in Subtask 1 is parameterized by MPNN for mapping metabolite structures to the molecular vectors. The *main contribution* of this article is in the Subtask 1, that is, to learn the correspondence between given pairs of spectra and structures for metabolites.

Thus, the parameters of MPNN are trained so that the correlation between the spectra and the vectors mapped from the structures is maximized. We use Hilbert-Schmidt Independence Criterion (HSIC, Gretton *et al.*, 2005) for evaluating the correlation, due to its theoretically nice properties and kernel-based calculation. Specifically, we formulate an objective function for the maximization problem through HSIC and solve this problem to have the best molecular vectors adapted to given data. For Subtask 2, ADAPTIVE uses IOKR to learn a mapping from spectra to molecular vectors generated by the Subtask 1.

We emphasize that the key difference between ADAPTIVE and the original IOKR is that IOKR uses 'manually designed' fingerprints, which are large in size, possibly redundant and nonspecific to metabolite identification (and given data), while ADAPTIVE learns representations for metabolites from given data, as molecular vectors, resulting in that the molecular vectors generated by ADAPTIVE are data-driven and concise.

In order to validate the performance of ADAPTIVE, we conducted extensive experiments using a benchmark data. Experimental results showed the following two main advantages of ADAPTIVE over existing methods, including the original IOKR:

- *Predictive performance*

ADAPTIVE achieved the best performance, followed by IOKR, CSI:FingerID and FingerID. For example, the top-20 accuracy of ADAPTIVE was 78.52% with the parameters of Gaussian kernel,

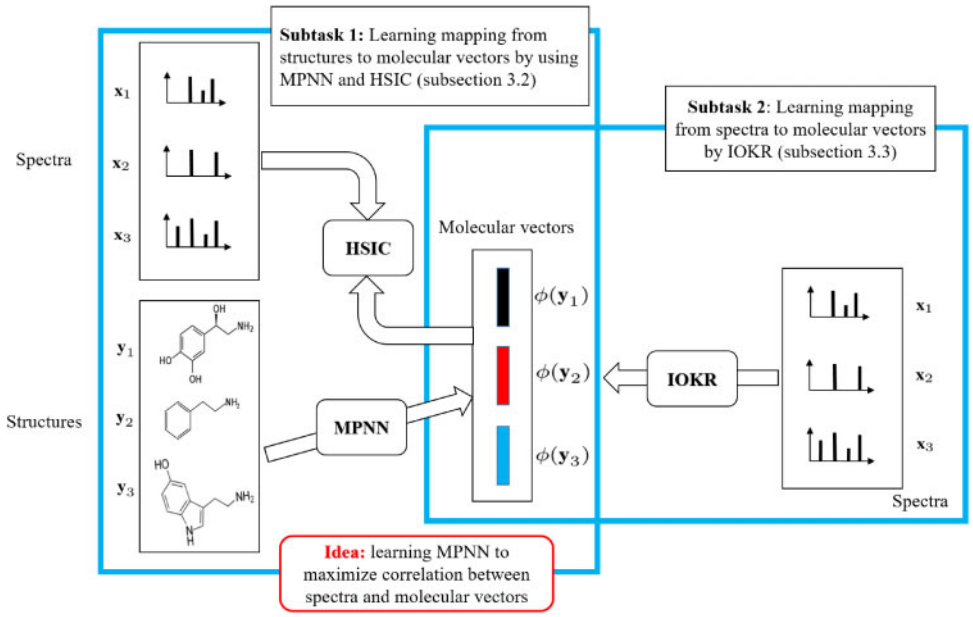

Fig. 2. Overview of ADAPTIVE for metabolite identification. ADAPTIVE has two components: (i) Subtask 1: estimates parameters of a function mapping metabolites from structures to molecular vectors, given a set of spectra-structure pairs; (ii) Subtask 2: learns a function mapping from spectra to molecular vectors (generated by Subtask 1), given a set of spectrum-vector pairs

ALIGNF and molecular vector size of 300. On the other hand, IOKR, CSI:FingerID and FingerID achieved 74.79%, 73.07% (or 68.20%) and 58.17%, respectively, using Gaussian kernel (for IOKR) and ALIGNF. The top-*k* accuracy was computed by the average over all trials of 10-fold cross-validation (CV), and so the performance advantage of ADAPTIVE was significant and very clear.

- *Computational efficiency for prediction*

Under the same experimental setting, ADAPTIVE was four to seven times faster than IOKR, which was already known as the fastest method. We can then say that ADAPTIVE is the current fastest method while keeping the highest predictive performance for metabolite identification.

2 Related work

As mentioned in the Introduction section, fingerprint prediction is important in supervised learning for metabolite identification, because we can retrieve metabolite candidates more reliably if fingerprints are predicted more accurately. For fingerprint prediction, kernel learning has been shown to be the most powerful approach. For example, a typical approach, FingerID (Heinonen *et al.*, 2012) uses probability product kernel (PPK, Jebara *et al.*, 2004), which can be directly computed from spectra and runs support vector machine with this kernel for solving fingerprint prediction as a classification problem. CSI:FingerID (Dührkop *et al.*, 2015), an extension of FingerID, uses not only spectra but also fragmentation trees (FTs, Rasche *et al.*, 2011) as input to generate kernels over spectra and FTs, which are then combined via multiple kernel learning (MKL, Gönen and Alpaydin, 2011). FTs may capture structural information behind spectra which is missing in the approach of FingerID. This is the motivation of CSI:FingerID. However, the computational cost for converting FTs from MS/MS spectra is very expensive, leading to heavy computational load, which causes a problem particularly in prediction. Thus, we can say that kernel-based supervised learning, particularly complex kernels, have a computation issue, regardless of high performance in prediction. On the other hand, a sparse learning model, namely SIMPLE (Nguyen *et al.*, 2018b), considers a simpler function than kernels for fingerprint, while interactions of peaks in spectra can be incorporated into learning models explicitly. SIMPLE achieved a comparable performance against kernel-based learning, reducing the computational cost drastically. A key point of SIMPLE is to take advantage of sparsity of spectra, which results in faster prediction and interpretability, showing clear advantages over kernel-based methods.

Among the series of kernel-based approaches, IOKR (Brouard *et al.*, 2016) has been shown to outperform the previous methods, in terms of both predictive performance and computational speed.

It learns a mapping from spectra, i.e. input $\mathcal{X}$, to molecular fingerprints (or structures behind fingerprints), i.e. output $\mathcal{Y}$. In order to do this mapping, IOKR defines kernels to encode similarities in the input space (e.g. spectra and/or FTs) and the output space (molecular fingerprints or structures). Then, the advantage of IOKR comes from the following two points: (i) unlike previous kernel-based methods, IOKR handles the structured output space by the kernel defined for the output, which improves the predictive performance; (ii) IOKR simultaneously predicts fingerprints rather than considering fingerprint prediction as a set of separate tasks, leading to an efficient computation in prediction. Some part (mapping from spectra to feature vectors) of IOKR is a part of ADAPTIVE, and so further technical details of the corresponding part of IOKR is described more in Section 3.

Conventionally, molecular fingerprints for fingerprint prediction have been manually designed feature vectors to encode a predefined set of substructures or chemical properties, which are possibly found in metabolites. However, recently, machine learning-based (or data-driven) algorithms for generating fingerprints have been proposed. A typical approach is neural fingerprint (NFP, Duvenaud *et al.*, 2015), which takes graphs with arbitrary sizes and shapes as inputs. NFP uses the idea of *graph convolution*, an extension of convolution operation from multidimensional arrays, like images or texts, to

graph structures. NFP is then trained in a supervised manner by using available labels, such as log mol/L for solubility, EC_{50} for drug efficacy. Finally, NFP results in fingerprint vectors (for molecules) specific to given task and data. An extension of NFP is for unsupervised (as well as semisupervised) settings to learn representations of molecular graph without labels (Nguyen *et al.*, 2017), since label information can be experimentally obtained and precious.

More recently, Gilmer *et al.* (2017) showed that several graph convolution-based models, including NFP, Gated Graph Neural Networks (Li *et al.*, 2015), spectral graph convolutional network (Kipf and Welling, 2016), etc., can be formulated in an unified model, namely MPNN, with the following three functions: *message passing*, *update* and *readout*. A key advantage of MPNN is that defining the above components generates a proper model for learning graphs, depending on a given task. Also, another advantage of MPNN as well as other neural network-based methods in this paragraph is that they adopt differentiable operations, and thus their parameters can be effectively trained by using a stochastic gradient descent algorithm.

3 Materials and methods

3.1 ADAPTIVE: overview

We first introduce the framework of ADAPTIVE for metabolite identification. This is also the general framework of approaches using machine learning for metabolite identification. It has two subtasks. *Subtask 1*: learning a function which maps metabolites from their structures to molecular vectors and *Subtask 2*: learning a function which maps metabolites from spectra to the vectors generated in Subtask 1. Figure 2 shows an illustration of the entire framework of ADAPTIVE. In this figure, the left and right blue boxes correspond to Subtasks 1 and 2, respectively.

For Subtask 1, given pairs of metabolite structure-spectrum, we estimate parameters of a function which maps metabolites from their structures to molecular vectors by maximizing the correlation between the vectors mapped from the structures and also the corresponding spectra. In more detail, we model the mapping function by MPNN and evaluate the correlation between the vectors and spectra by using HSIC due to the computational simplicity and provably theoretical properties of HSIC. For Subtask 2, we simply borrow the corresponding part of IOKR to learn a function mapping metabolites from spectra to vectors generated by Subtask 1.

We explain these two subtasks in the following subsections, being followed by the subsection on kernels we used in ADAPTIVE.

3.2 Subtask 1: learning molecular vectors for metabolites via HSIC

For this subtask, we need to estimate a function to map metabolites from structures to molecular vectors, given spectrum-structure pairs. For this problem, we use MPNN as the mapping function, which can extract meaningful representation for graphs (molecules for our problem) by supervised learning from training data. That is, MPNN requires labeled training data, which are, however, unavailable for this subtask. Then we manage this problem by taking advantage of given spectrum-structure pairs. We estimate parameters of MPNN by using the idea of maximizing the correlation between the given spectra and vectors (mapped from structures). The correlation is evaluated by HSIC. We describe the detail of MPNN, HSIC and related optimization procedures in the following subsections.

3.2.1 Message passing neural network

MPNN is a framework, which takes graphs of arbitrary sizes and structures as inputs, to learn their representation vectors at different levels (i.e. nodes, subgraphs and the whole graph) in a supervised manner (Gilmer *et al.*, 2017). A key advantage is that MPNN allows to learn features specific to the given task from the given data. Below, we explain the procedure of MPNN.

First let G be an undirected graph, and v and vw be a node (atom in molecules) and an edge (bond in molecules), respectively. Each node v is assigned with *state vectors* at different levels, where each level represents a substructure (or subgraph) rooted at the corresponding node, denoted by h_v^r, where r shows a level. We can compute state vector h_v^r as well as *message* m_v^r in a hierarchical manner, by using the following two functions: *message passing* (1) and *update* (2):

$$m_v^{r+1} = \sum_{w \in \mathcal{N}(v)} H_{e(vw)}^r h_w^r, \tag{1}$$

$$h_v^{r+1} = g(h_v^r + m_v^{r+1}), \tag{2}$$

where $\mathcal{N}(v)$ denotes the set of neighbors of node v in graph G; $e(v, w)$ indicates the type of edge between two nodes v and w (this edge type is like a single, double, triple or aromatic bond); $H_{e(vw)}^r$ is a (square) weight matrix to be learned, specific to the edge type $e(vw)$ at the rth level; g is a nonlinear activation function (e.g. ReLU or sigmoid).

Intuitively, the *message passing* function (1) on node v plays the role of collecting information from the neighbors of node v and *update* function (2) on node v is to update the state of node v based on the collected information and the former state of node v. Thus, by applying two functions (1) and (2) multiple times, the updated features at node v (e.g. h_v^{r+1}) can be used to represent a certain number of substructures with the root of node v. Then, the values for these series of substructures can be used to generate a vector at node v with different levels (sizes) of substructures. Figure 3 shows a schematic and illustrative picture of this procedure [Fig. 3 is from Nguyen *et al.* (2017)].

After obtaining the state vectors of substructures rooted at node v, i.e. h_v^r, we have the *readout* phase to combine all vectors at different levels into a single representation vector of the whole molecule (namely, NFPs). Figure 4 shows a schematic picture of summing up the state vectors at different levels. As in Duvenaud *et al.* (2015), we adopt the softmax operation on the states and then perform linear projections (parameterized by different weight matrices W_r) and finally sum them up to obtain a single vector over different levels which represents the whole graph. In short, the molecular vector for the entire molecule can be written as following:

$$\sum_r \sum_v \mathrm{softmax}(W_r h_v^r). \tag{3}$$

We note that operations are all differentiable with respect to parameters, which makes learning the parameters possible, given an objective function, by a stochastic or minibatch gradient descent algorithm. Algorithm 1 shows a pseudocode of the procedure of repeating the *message passing* and *update* functions.

3.2.2 HSIC-based objective function

We estimate parameters of MPNN by maximizing the correlation (dependency) between given spectra and molecular vectors. A lot of measures can be used to evaluate and estimate the correlation, while we use HSIC due to its theoretically sound properties. More

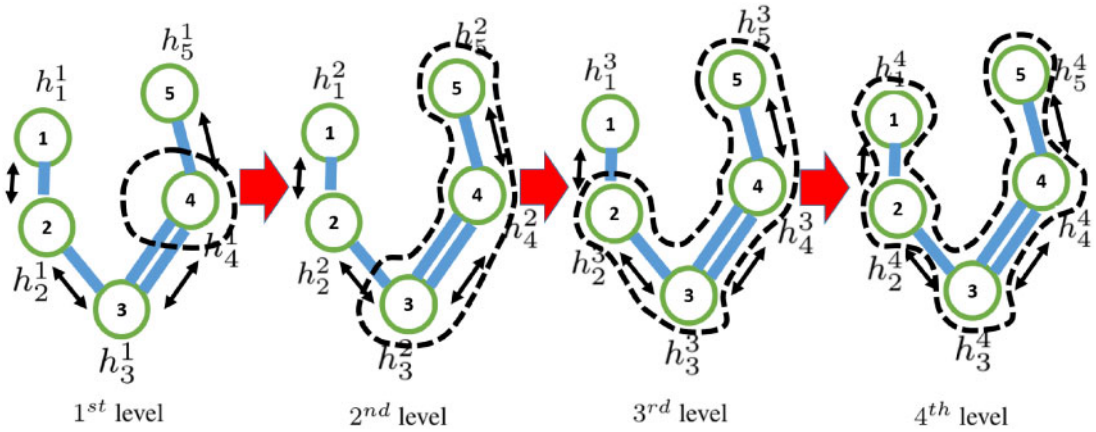

Fig. 3. Message passing and update functions are used to represent rooted substructures in a hierarchical manner. At the first level (left-most graph), each node is represented by feature vector, with only information of the node itself. We note that by repeatedly applying message passing and update functions (from left to right), more neighboring information are incorporated. For example, the updated feature (second level) has information on nodes 3 and 5, and then third level has that on nodes 2 to 5. Finally, the whole graph is covered

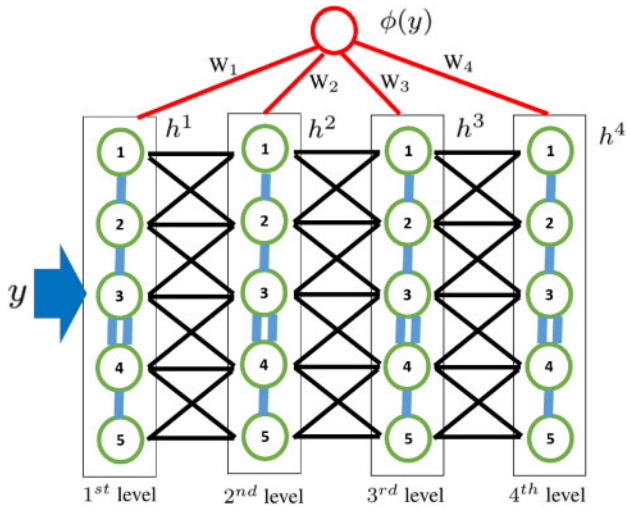

Fig. 4. Representation vectors of substructures, which are rooted at nodes, are computed from the input graph by the message passing and update functions. These functions contribute to computing the molecular representation vector of the whole molecule

importantly, estimation of HSIC is based on kernel calculation, which can effectively deal with the uncertainty of peaks in spectra caused by measurement errors.

Formally, we are given dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i, \mathbf{y}_i$ are spectrum and molecular structure, respectively, of the ith metabolite. First, for the spectra, i.e. x, we consider kernels which combine spectra with FTs, namely $k(\mathbf{x}_i, \mathbf{x}_j)$. We describe the detail of the kernels for spectra in Section 3.4. Then, given the kernel over $\mathcal{X}$ is fixed, the goal is to learn the function $\phi : \mathcal{Y} \mapsto \mathcal{F}_d$ from $\mathcal{D}$ such that the correlation between the input and output is maximized. The $\phi(\mathbf{y})$ is the output of MPNN (or molecular vectors) which belongs to space $\mathcal{F}_d$. The linear kernel function induced by this space can be written as follows:

$$l(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle \tag{4}$$

To evaluate the correlation between spectra and molecular vectors (output of MPNN), we use an unbiased empirical estimate of HSIC (Gretton *et al.*, 2005), which can be given as follows:

Algorithm 1. Message Passing Neural Network (MPNN).

Inputs:
minibatch of molecular structures $\mathbf{Y}_b = \{\mathbf{y}_i\}_{i=1}^B$, radius R
weight matrices of edges: $\mathrm{H}_1^1, \mathrm{H}_2^1, \ldots, \mathrm{H}_4^R$,
weight matrices of *readout* function: W_1, W_2,..., W_R
Outputs:
molecular vectors $\phi(\mathbf{Y}_b)$
for $i \leftarrow 1$ to B **do**
  for each atom v in $\mathbf{y}_i$ **do**
    $h_v \leftarrow$initial hidden rep. vector of v ▷ atom feature
  end for
    $\phi_i \leftarrow 0_d$ ▷ Initialize each molecular vector with a zero vector
  for $r \leftarrow 1$ to R **do**
    for each node v in $\mathbf{y}_i$ **do**
      $m_v^{r+1} = \sum_{w \in \mathcal{N}(v)} \mathrm{H}_{vw}^r h_w^r$ ▷ *message* function
      $h_v^{r+1} = g(h_v^r + m_v^{r+1})$ ▷ *update* function
      $\phi_i = \phi_i + \mathrm{softmax}(\mathrm{W}_{r+1} h_v^{r+1})$ ▷ *readout* function
    end for
  end for
end for
$\phi(\mathbf{Y}_b) = [\phi_1, \phi_2, \ldots, \phi_B]$

$$\mathrm{uHSIC}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n(n-3)} [\mathrm{trace}(\bar{\mathrm{K}}_n \bar{\mathrm{L}}_n) + \frac{\mathbf{1}_n^\top \bar{\mathrm{K}}_n \mathbf{1}_n \mathbf{1}_n^\top \bar{\mathrm{L}}_n \mathbf{1}_n}{(n-1)(n-2)} - \frac{2}{n-2} \mathbf{1}_n^\top \bar{\mathrm{K}}_n \bar{\mathrm{L}}_n \mathbf{1}_n], \tag{5}$$

where $\bar{\mathrm{K}}_n = \mathrm{K}_n - \mathrm{diag}(\mathrm{K}_n)$ denotes the kernel matrix for the set of n spectra $\mathcal{X}$ with diagonal elements set to zero; $\mathbf{1}_n$ is a vector of 1 s of n dimensions. Likewise $\bar{\mathrm{L}}_n = \mathrm{L}_n - \mathrm{diag}(\mathrm{L}_n)$, where L_n is the kernel matrix of n molecular vectors output by MPNN. By arranging terms in (5), we can rewrite (5) as the objective function to learn parameters as follows:

$$\mathrm{uHSIC}(\mathcal{X}, \mathcal{Y}) = \mathrm{trace}(\mathrm{S}_n \bar{\mathrm{L}}_n) \tag{6}$$

where

$$\mathrm{S}_n = \frac{1}{n(n-3)}\left[\bar{\mathrm{K}}_n + \frac{1_n 1_n^\top \bar{\mathrm{K}}_n 1_n 1_n^\top}{(n-1)(n-2)} - \frac{2}{n-2} 1_n 1_n^\top \bar{\mathrm{K}}_n\right] \tag{7}$$

However, directly optimizing (6) is prohibitively expensive in computation, particularly for large-scale data, since the complexity reaches $\mathrm{O}(n^2)$, both in space and time. In order to overcome this limitation, following Zhang *et al.* (2018), we disjointly divide samples $(\mathcal{X}, \mathcal{Y})$ into n/B blocks with the size of B, $\{\{(\mathbf{x}_i^{(b)}, \mathbf{y}_i^{(b)})\}_{i=1}^{B}\}_{b=1}^{n/B}$ and then apply HSIC on each block independently. An empirical estimate of the unbiased block HSIC can be defined by:

$$\mathrm{ubHSIC}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n/B}\sum_{b=1}^{n/B} \mathrm{trace}(\mathrm{S}_b \bar{\mathrm{L}}_b), \tag{8}$$

where S_b can be defined by a similar manner to (7), and $\bar{\mathrm{L}}_b$ is the kernel matrix for the bth block.

Furthermore, in order to avoid the effect by biased partition of the dataset, following Yamada *et al.* (2018), we repeat shuffling dataset T times, compute ubHSIC on each permutation and take the average over them. HSIC by this procedure is known as bagging block HSIC, which can be written as follows:

$$\mathrm{ubHSIC}(\mathcal{X}, \mathcal{Y}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n/B}\sum_{b=1}^{n/B} \mathrm{trace}(\mathrm{S}_{t,b} \bar{\mathrm{L}}_{t,b}), \tag{9}$$

We use (9) as objective function J to learn parameters.

3.2.3 Optimization algorithm

An advantage of objective function (9) is that we can use the gradient descent (minibatch gradient descent) for estimating parameters of MPNN. We here explain details on how to conduct the minibatch gradient descent procedure for the HSIC-based loss, which has three steps.

Step 1: Feed forward and loss calculation.

For samples of size n, at each iteration, we perform random permutation and then split all samples into batches, where the size of each batch is B. Batches are sequentially fed into MPNN. The output of MPNN for the bth batch at the tth iteration is denoted by $\phi(\mathbf{Y}_{t,b}) = (\phi(\mathbf{y}_1^{t,b}), \phi(\mathbf{y}_2^{t,b}), \ldots \phi(\mathbf{y}_B^{t,b}))$. Then using these outputs, the objective function on the whole samples can be calculated as in (9).

Step 2: Gradient calculation of the loss layer.

As we can compute the loss directly with the output of MPNN (i.e. $\phi(\mathbf{Y}_{t,b})$), we need to compute the gradient of J with respect to $\phi(\mathbf{Y}_{t,b})$. Suppose that the output of MPNN is already normalized, i.e. $\phi(\mathbf{y})^\top \phi(\mathbf{y}) = 1$ for all $\mathbf{y} \in \mathcal{Y}$, the gradient can be obtained by the following:

$$\frac{\partial J}{\partial \phi(\mathbf{Y}_{t,b})} = \frac{B}{Tn}\mathrm{S}_{t,b}\phi(\mathbf{Y}_{t,b}) \tag{10}$$

Step 3: Gradient calculation of the MPN and weight update.

Having calculated the gradient of J, i.e. (10), the next step is to compute the gradient of $\phi(\mathbf{Y}_{t,b})$ with respect to model parameters θ, namely $\frac{\partial \phi(\mathbf{Y}_{t,b})}{\partial \theta}$, to update the whole parameters for each batch at each step.

Algorithm 2 is a pseudocode of the entire algorithm of learning parameters of MPNN.

Algorithm 2. Learning molecular representation vectors via HSIC.

Inputs:
set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ of spectra-structure pairs,
T: number of iterations, B: size of minibatch
Outputs:
$\theta = \{\mathrm{H}_1^1, \mathrm{H}_2^1, \ldots, \mathrm{H}_4^R, \mathrm{W}_1, \mathrm{W}_2, \ldots, \mathrm{W}_R\} \cup \{h_v | v \in$ set of atoms$\}$
for $t \leftarrow 1$ to T **do**
  $\mathcal{D}_t = \{\{(\mathbf{x}_i^{t,b}, \mathbf{y}_i^{t,b})\}_{i=1}^{B}\}_{b=1}^{n/B}$ ▷ shuffled and split
  for $b \leftarrow 1$ to B **do**
    $\mathbf{Y}_{t,b} = \{\mathbf{y}_i^{t,b}\}_{i=1}^{B}$, $\mathbf{X}_{t,b} = \{\mathbf{x}_i^{t,b}\}_{i=1}^{B}$
    $\mathrm{F}_{t,b} = \phi(\mathbf{Y}_{t,b})$ ▷ Call Algorithm 1
    $\mathrm{S}_{t,b}$ is calculated from $\mathbf{X}_{t,b}$ by (7)
    $\mathrm{J}_{t,b} = \mathrm{trace}(\mathrm{S}_{t,b}\mathrm{F}_{t,b}^\top \mathrm{F}_{t,b})$
    gradient of loss layer $\leftarrow \mathrm{S}_{t,b}\mathrm{F}_{t,b}$
    Grad_θ is calculated by chain rule
    $\theta \leftarrow \theta - \gamma \mathrm{Grad}_\theta$ ▷ Update the whole parameters
  end for
end for

3.3 Subtask 2: learning a mapping from spectra to molecular vectors by IOKR

For Subtask 2, we use IOKR. That is, we learn a mapping from spectra to molecular vectors generated in Subtask 1 by using IOKR. Again, we explain two technical reasons why we use IOKR for this mapping below: (i) IOKR allows to incorporate the structures behind outputs, such as feature interactions in molecular vectors, into the learning model, by which the prediction accuracy can be improved. (ii) Furthermore, all features in molecular vectors are predicted simultaneously, which is not like separate tasks in prediction. This leads to faster computation.

We now present the technical detail of IOKR below, which has two consecutive steps.

3.3.1 Step 1: Learning spectra-vectors mapping

Once parameters, i.e. function ϕ, are learned, we convert the structures of metabolites into their molecular vectors to obtain a new set of pairs, $\{(\mathbf{x}_i, \phi(\mathbf{y}_i))\}_{i=1}^{n}$. Now the goal is to find the optimal function $h : \mathcal{X} \to \mathcal{F}_d$ by minimizing the following objective function:

$$\hat{h} = \underset{h \in \mathcal{H}}{\mathrm{argmin}} \sum_{i=1}^{n} ||h(\mathbf{x}_i) - \phi(\mathbf{y}_i)||_{\mathcal{F}_d}^2 + \lambda ||h||_{\mathcal{H}}^2, \tag{11}$$

where λ (> 0) is a regularization parameter to prevent overfitting and $\mathcal{H}$ is an approximate functional space that contains h; $\mathcal{F}_d$ is a space of molecular vectors of dimension d.

By using the representer theorem in Micchelli and Pontil (2005), optimal solution $\hat{h}$ of (11) can be represented by a linear combination of vector-valued kernels on training set $\mathcal{X}$:

$$\hat{h}(\mathbf{x}_i) = \sum_{i=1}^{n} \mathcal{K}_n(\mathbf{x}_i, \mathbf{x}_j) c_j, \tag{12}$$

where $c_i (i = 1, \ldots, n)$ are vectors in $\mathcal{F}_d$; $\mathcal{K}_n$ is an operator-valued kernel, defined on spectra $\mathcal{X}$, satisfying certain constraints (see Micchelli and Pontil, 2005). As dimensionality d of space $\mathcal{F}_d$ is finite, the kernel is a matrix with the size of $d \times d$.

By replacing $\hat{h}(\mathbf{x}_i)$ in (11) with (12), $c_i (i = 1, \ldots, n)$ can be estimated in the following:

$$\mathrm{vec}(\mathbf{C}_n) = (\lambda \mathbf{I}_{nd} + \mathcal{K}_n)^{-1} \mathrm{vec}(\phi(\mathbf{Y}_n)), \tag{13}$$

where $\mathbf{C}_n = (c_1, c_2, \ldots, c_n)$ and $\phi(\mathbf{Y}_n) = (\phi(\mathbf{y}_1), \phi(\mathbf{y}_2), \ldots, \phi(\mathbf{y}_n))$ are both matrices with the size of $d \times n$, and vec(.) is the

vectorization of the input matrix, where the output is a vector obtained by repeatedly stacking each column of the input matrix on the top of the next column.

3.3.2 Step 2: Candidate retrieval

Given mapping $\hat{h}$ learned in Step 1, we now turn to the problem of finding the output metabolite in the database which corresponds to the query spectrum $\mathbf{x}$. To this end, we search metabolite $\mathbf{y}$ in the list of given candidates $\mathcal{Y}^*$, such that the squared distance between $\phi(\mathbf{y})$ and $\hat{h}(\mathbf{x})$ can be minimized:

$$f(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}^*}{\operatorname{argmin}} ||\hat{h}(\mathbf{x}) - \phi(\mathbf{y})||^2_{\mathcal{F}_d} \tag{14}$$

Considering that the output kernel is normalized and the operator-valued kernel keeps $\mathcal{K}_n(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}') * \mathbf{I}_d$, the optimal solution of $f(\mathbf{x})$ can be estimated as the following:

$$\hat{f}(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}^*}{\operatorname{argmax}}\, l(\mathbf{Y},\mathbf{y})^\top (\lambda \mathbf{I}_n + \mathrm{K}_n)^{-1} k(\mathbf{X},\mathbf{x}), \tag{15}$$

where $l(\mathbf{Y},\mathbf{y}) = \begin{bmatrix} l(\mathbf{y}_1,\mathbf{y}) \\ \vdots \\ l(\mathbf{y}_n,\mathbf{y}) \end{bmatrix}$ and $k(\mathbf{X},\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}) \\ \vdots \\ k(\mathbf{x}_n,\mathbf{x}) \end{bmatrix}$ are column vectors.

Practically, the values given by objective function (15) are used as scores for ranking candidate metabolites in Step 2: candidate retrieval.

3.4 Kernels

ADAPTIVE uses kernels for the input and output.

3.4.1 Kernels for input

A various types of kernels are already defined and used for the input from MS/MS spectra. These kernels are typically divided into the following two groups: (i) kernels defined for spectra such as PPK (Jebara *et al.*, 2004) and (ii) kernels defined for FTs (Rasche *et al.*, 2011). Details on these kernels can be found in Dührkop *et al.* (2015).

In fact, Dührkop *et al.* (2015) suggested 24 different input kernels. ADAPTIVE combines these input kernels into a single kernel through MKL (Gönen and Alpaydin, 2011). ADAPTIVE uses two options for MKL: (i) UNIMKL (uniform MKL): assigns the same weights to all component kernels, and (ii) ALIGNF: uses weights over kernels to combine. That is, in ALIGNF, weights over component kernels are optimized (trained) by maximizing the centered kernel alignment between the combined kernel and the target kernel defined on the molecular vectors, which generate trained parameters (model).

3.4.2 Kernels for output

After learning parameters (model) to generate the molecular vectors for structures, we define kernels for output $\mathcal{Y}$ by directly computing kernels on the corresponding molecular vectors. In our experiments, we consider the following two typical kernels:

- Linear kernel: $l(\mathbf{y},\mathbf{y}') = \phi(\mathbf{y})^\top \phi(\mathbf{y}')$.
- Gaussian kernel: $l(\mathbf{y},\mathbf{y}') = \exp(-\gamma||\phi(\mathbf{y}) - \phi(\mathbf{y}')||^2)$,

  where $\mathbf{y}$ and $\mathbf{y}'$ are molecular structures in $\mathcal{Y}$.

4 Experimental results

4.1 Dataset and evaluation measures

We used a benchmark dataset in Brouard *et al.* (2016) to evaluate ADAPTIVE and compare with existing methods. The dataset consists of 4138 MS/MS spectra extracted from the GNPS (Global Natural Products Social) public spectra library (https://gnps.ucsd.edu/ProteoSAFe/libraries.jsp).

Table 1. Parameter values used for experiments

Notations	Parameter	Values
T	#epoch	100
B	Batchsize	100
R	#updates	6
d	#dim of molecular vectors	100,200,300
m	#dim of atom feature	50
	#atom types	12 (C, O, N, P, S, etc.)
	#bond types	4 (single, double, triple, acromatic)

To compare ADAPTIVE with existing methods, we used the same setting for all competing methods. Specifically we used 10-fold CV, and the results are averaged over all 10-folds. The performance was checked by the top-k accuracies (where $k = 1$, 10, 20), which is the ratio of the number of the cases that the true structures are ranked at lower than or equal to k to the number of all cases. Also the speed was checked by computation time for prediction, measured by milliseconds per example (ms/example).

Hyperparameters, such as regularization parameter λ and parameter γ of the output kernel, were chosen by using leave-one-out CV on each training fold. For prediction in ADAPTIVE, at the retrieval stage, given test example $\mathbf{x}$, we computed the molecular vectors of $\mathbf{x}$, $\hat{h}(\mathbf{x})$ [see (11)] and those of all candidates $\phi(\mathbf{y})$ (see Algorithm 1). These candidates including the correct molecular structure of test example $\mathbf{x}$ were ranked, according to their distances to $\hat{h}(\mathbf{x})$ (from the smallest to the highest). These ranked candidates were used for computing the top-k accuracy. Table 1 shows a set of parameter values, which were used to train MPNN of generating molecular vectors. *State vectors* of identical atoms at the lowest (atomic) level were initialized with the same random vector sampled from the standard normal distribution and updated during the training stage.

All experiments were performed on a server with 2.7 GHz Intel Core i5 CPU and 8GB memory. The code was written in Python and Matlab with the support of the Chainer framework (Tokui *et al.*, 2015).

4.2 Performance results

4.2.1 Predictive performance

We compared the predictive performances of ADAPTIVE with three existing methods: FingerID (Heinonen *et al.*, 2012), CSI:FingerID (Dührkop *et al.*, 2015) and IOKR (Brouard *et al.*, 2016) in terms of the top-k accuracy ($k = 1$, 10 and 20). Table 2 shows the top-k accuracies of the competing methods with UNIMKL and ALIGNF for MKL and linear and Gaussian kernels for the output kernel, changing k from 1 to 20 and also changing the size of fingerprints from 100 to 300 (for ADAPTIVE only). This table first shows that ADAPTIVE achieved the best performance, being followed by IOKR, CSI:FingerID and FingerID. For example, ADAPTIVE with ALIGNF, Gaussian kernel and the fingerprint size of 300 achieved 31.03% for $k = 1$, while IOKR with ALIGNF and Gaussian kernel was 29.59% and CSI:FingerID with ALIGNF was 28.84% or 24.82%. That of Finger: ID was only 17.74%. Interestingly, for $k = 1$, the performance advantage of ADAPTIVE against IOKR was rather slight, while $k = 10$ and 20, ADAPTIVE outperformed IOKR much more clearly, with the difference of around 3–5% under the same condition for the two methods.

Table 2. Comparison of the top-k accuracy ($k = 1$, 10 and 20) of FingerID, CSI:FingerID, IOKR and ADAPTIVE

Method	Vec. size	MKL	Accuracies (mean/SD %)		
			Top 1	Top 10	Top 20
FingerID	2765	None	17.74	49.59	58.17
CSI:FingerID unit	2765	ALIGNF	24.82	60.47	68.20
CSI:FingerID mod	2765	ALIGNF	28.84	66.07	73.07
Platt					
IOKR linear	2765	UNIMKL	30.58/2.23	65.99/2.46	73.53/2.47
		ALIGNF	28.54/2.54	65.77/2.39	73.19/3.11
ADAPTIVE	100	UNIMKL	29.42/2.83	70.01/2.79	77.48/2.98
linear		ALIGNF	29.19/3.21	69.52/2.89	77.64/3.23
	200	UNIMKL	29.57/3.96	69.38/3.05	76.95/2.98
		ALIGNF	29.11/3.45	69.53/2.52	77.56/2.43
	300	UNIMKL	30.22/3.47	70.48/2.72	78.18/2.67
		ALIGNF	**30.61/3.23**	**70.51/2.52**	**78.23/2.75**
IOKR Gaussian	2765	UNIMKL	30.66/2.34	66.51/2.87	73.94/2.54
		ALIGNF	29.59/2.58	66.13/2.09	73.62/1.85
ADAPTIVE	100	UNIMKL	29.47/3.21	70.01/2.83	77.51/2.11
Gaussian		ALIGNF	29.37/3.21	69.91/2.64	77.48/2.33
	200	UNIMKL	29.44/3.86	69.84/2.78	77.08/2.95
		ALIGNF	28.98/3.32	69.65/2.71	77.15/2.74
	300	UNIMKL	30.31/3.48	**71.10**/2.73	78.51/2.65
		ALIGNF	**31.03**/3.40	70.89/2.74	**78.52**/2.52

Note: The highest value (indicating the most accurate prediction) are in boldface for each k.

We used one-sided paired t-test to verify if the differences between ADAPTIVE and IOKR are statistically significant. For example, considering the top 10 accuracy with Gaussian kernel and ALIGNF, the calculated P-value was $P = 0.0012$. Since it is less than the significance level of $\alpha = 0.01$, we can claim the statistical significance of the advantage of ADAPTIVE in terms of the top 10 accuracy over IOKR under Gaussian kernel and ALIGNF. We conclude that the performance advantage of ADAPTIVE was confirmed by checking a larger number of top candidates. Another finding is the performance difference between linear and Gaussian kernels was very slight (almost nothing) for ADAPTIVE under the same other conditions. This is also true with the settings of UNIMKL and ALIGNF, the performance for them was rather the same. However, the size of fingerprints strongly affected the performance in the sense that a larger size of fingerprints achieved a higher performance. In summary, ADAPTIVE clearly outperformed competing methods with, for example, for $k = 20$, the difference of 3–5%, which is very sizable.

4.2.2 Computation time for prediction

IOKR was already shown to be faster than previous kernel-based methods in prediction (Brouard *et al.*, 2016). Thus, we consider only IOKR as a competing method for examining computational efficiency. Table 3 shows the computation time of ADAPTIVE and IOKR with linear and Gaussian kernels for prediction. The computation time was averaged over the 10-fold CV. This table shows that ADAPTIVE was significantly faster than IOKR. Specifically, under both linear and Gaussian kernels, ADAPTIVE with the fingerprint size of 100 was four to seven times faster than IOKR. This is because molecular vectors by ADAPTIVE are much more precise and adaptive to given data than those used in IOKR.

Table 3. Computation time for prediction by ADAPTIVE and IOKR

Method	Mol. vec. size	prediction time (ms/example)	
		Linear	Gaussian
IOKR	2765	140.22	3352.4
ADAPTIVE	100	**20.32**	**802.6**
	200	39.88	844.33
	300	54.14	1071.8

Note: The smallest values (indicating the fastest) were in boldface for linear and Gaussian kernels.

4.3 Case study

To understand the results obtained by ADAPTIVE more, in the obtained molecular vectors, we examined substructures rooted at atoms, which activated several example features most. As shown in Section 3.2.1, each substructure rooted at an atom is represented by a state vector and contributes to computing the molecular vector of the whole molecule. Then, given a feature, we can estimate the contribution of each substructure by simply computing the softmax value from the corresponding state vector. We use these values of substructures as scores to rank substructures to activate the given feature.

Figure 5 shows three example features (#2, #39 and #83). For each feature, we show three substructures with the highest scores (each score is shown above the substructure). The first row shows three substructures which activated feature #2 most. Interestingly, we can see that these substructures share a further smaller, similar group of atoms: O, P and S (highlighted in blue). Similarly, the second row shows three substructures sharing a group of atoms: O and N, where these substructures activated feature #39 most. Also the third row shows substructures which activated feature #83, all having atom: Cl. Thus, Figure 5 shows that each feature of ADAPTIVE is activated by multiple different substructures sharing some similar properties, which must be important in data and probably for prediction. In contrast, each feature in regular molecular fingerprints is activated by only one predefined substructure. In summary, from this case study, learned features of ADAPTIVE are more concise and specific to the task of metabolite identification than regular molecular fingerprints, leading to the advantage of predictive performance and computation time.

5 Discussion and conclusion

Supervised learning for metabolite identification uses fingerprints as intermediate representation vectors between spectra and metabolites, while such fixed vectors are too redundant to cover all possible substructures and chemical properties in metabolites, causing limitations in predictive performance and high computational costs. To overcome this problem, we have proposed ADAPTIVE, which generates representations of metabolites specific to given spectrum-structure pairs. ADAPTIVE learns a model to generate molecular vectors for metabolites, which is parameterized by a MPNN over given molecular structures and trained through optimizing the objective function to maximize the correlation between molecular vectors and corresponding spectra. Our empirical validation of ADAPTIVE with the benchmark dataset showed the advantage of ADAPTIVE over existing methods including IOKR, the current cutting-edge method, both in predictive performance and computation time for prediction.

A drawback of ADAPTIVE would be interpretability, because structural information is implicitly encoded in compact vectors in ADAPTIVE and cannot be made explicit easily. In metabolite identification, it would be desirable to connect the set of peaks to the

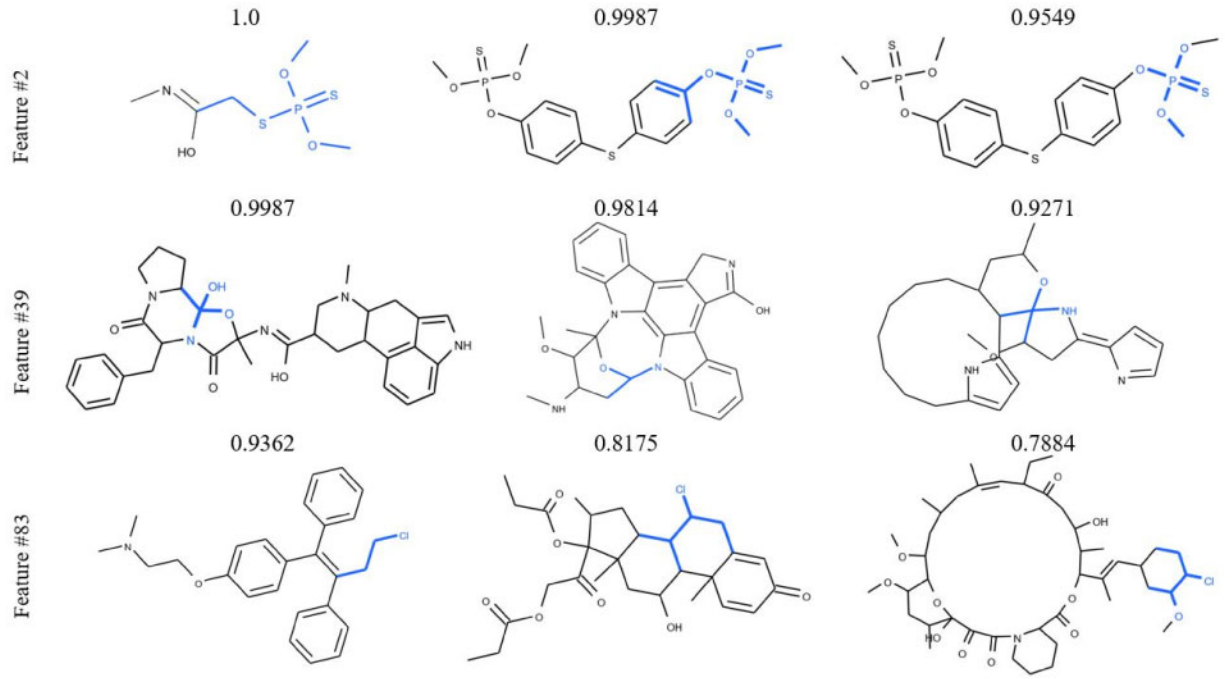

Fig. 5. Example features (#2, #39 and #83) and their three substructures (and their scores) which activated the corresponding feature most. Note that three substructures of each feature share a similar group (set) of atoms which are shown in blue

corresponding substructures/chemical properties of metabolites (Nguyen *et al.*, 2018b). Developing a model with such interpretability would be interesting future work.

Funding

D.H.N. has been supported in part by Otsuka Toshimi scholarship and JSPS KAKENHI [grant number 19J14714]. C.H.N. has been supported in part by MEXT Kakenhi 18K11434. H.M. has been supported in part by JST ACCEL [grant number JPMJAC1503], MEXT Kakenhi [grant numbers 16H02868 and 19H04169], FiDiPro by Tekes (currently Business Finland) and AIPSE program by Academy of Finland.

Conflict of Interest: none declared.

References

Brouard,C. *et al.* (2016) Fast metabolite identification with input output kernel regression. *Bioinformatics*, **32**, i28–i36.

de Hoffmann,E. and Stroobant,V. (2007). *Mass Spectrometry, Principles and Applications*. 3rd edn. John Wiley & Sons, Hoboken, New York.

Dührkop,K. *et al.* (2015) Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc. Natl. Acad. Sci.*, **112**, 12580–12585.

Duvenaud,D.K. *et al.* (2015). Convolutional networks on graphs for learning molecular fingerprints. In: Cortes,C., Lawrence,N.D., Lee,D.D., Sugiyama,M., and Garnett,R. (eds.) *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Vol. 2. Curran Associates, Inc., Montreal, Canada, pp. 2224–2232.

Gilmer,J. *et al.* (2017). Neural message passing for quantum chemistry. In: Precup,D. and Teh,Y.W. (eds.) *Proceedings of the 34th International Conference on Machine Learning, Volume 70 of Proceedings of Machine Learning Research*. International Convention Centre, PMLR, Sydney, Australia, pp. 1263–1272.

Gönen,M., and Alpaydin,E. (2011) Multiple kernel learning algorithms. *J. Mach. Learn. Res.*, **12**, 2211–2268.

Gretton,A. *et al.* (2005). Measuring statistical dependence with Hilbert-Schmidt norms. In: *Proceedings of the 16th International Conference on Algorithmic Learning Theory, ALT'05*. Springer-Verlag, Berlin, Heidelberg, pp. 63–77.

Heinonen,M. *et al.* (2012) Metabolite identification and molecular fingerprint prediction through machine learning. *Bioinformatics*, **28**, 2333–2341.

Jebara,T. *et al.* (2004) Probability product kernels. *J. Mach. Learn. Res.*, **5**, 819–844.

Kipf,T.N. and Welling,M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv: 1609.02907*.

Li,Y. *et al.* (2015) Gated graph sequence neural networks. *CoRR*, abs/1511.05493.

Micchelli,C.A. and Pontil,M.A. (2005) On learning vector-valued functions. *Neural Comput.*, **17**, 177–204.

Nguyen,D.H. *et al.* (2018a). Recent advances and prospects of computational methods for metabolite identification: a review with emphasis on machine learning approaches. *Brief. Bioinf.* doi: 10.1093/bib/bby066. [Epub ahead of print].

Nguyen,D.H. *et al.* (2018b) Simple: sparse interaction model over peaks of molecules for fast, interpretable metabolite identification from tandem mass spectra. *Bioinformatics*, **34**, i323–i332.

Nguyen,H. *et al.* (2017). Semi-supervised learning of hierarchical representations of molecules using neural message passing. *CoRR*, abs/1711.10168.

Rasche,F. *et al.* (2011) Computing fragmentation trees from tandem mass spectrometry data. *Anal. Chem.*, **83**, 1243–1251. PMID: 21182243.

Tokui,S. *et al.* (2015) Chainer: a next-generation open source framework for deep learning. In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*, Vol. 5, pp. 1–6.

Vaniya,A. and Fiehn,O. (2015) Using fragmentation trees and mass spectral trees for identifying unknown compounds in metabolomics. *Trends Analyt. Chem.*, **69**, 52–61.

Wishart,D.S. (2007) Current progress in computational metabolomics. *Brief. Bioinf.*, **8**, 279–293.

Wishart,D.S. *et al.* (2013) HMDB 3.0—the human metabolome database in 2013. *Nucleic Acids Res.*, **41**, D801–D807.

Yamada,M. *et al.* (2018). Post selection inference with kernels. In: Storkey,A. and Perez-Cruz,F. (eds.) *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, volume 84 of Proceedings of Machine Learning Research*. Playa Blanca, PMLR, Lanzarote, Canary Islands, pp. 152–160.

Zhang,Q. *et al.* (2018) Large-scale kernel methods for independence testing. *Stat. Comput.*, **28**, 113–130.