Metabolite Identification through Multiple Kernel Learning on Fragmentation Trees

Huibin Shen\textsuperscript{1,2}, Kai Dührkop\textsuperscript{3}, Sebastian Böcker\textsuperscript{3} and Juho Rousu\textsuperscript{1,2}

\textsuperscript{1}Department of Information and Computer Science, Aalto University, Finland.
\textsuperscript{2}Helsinki Institute for Information Technology, Finland.
\textsuperscript{3}Chair for Bioinformatics, Friedrich-Schiller-University Jena, Germany.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

1 INTRODUCTION

Metabolomics deals with the analysis of small molecules and their interactions in living cells. A central task in metabolomics experiments, is the identification and quantification of the metabolites present in a sample. This is mandatory for subsequent analysis steps such as metabolic pathway analysis and flux analysis (Pirkanen et al., 2010). Mass spectrometry (MS) is one of the two predominant analytical technologies for metabolite identification. Identification is done by fragmenting the metabolite, for example by tandem MS (MS/MS), and measuring the mass-to-charge ratios of the resulting fragment ions. The measured mass spectra contain information about the metabolite, but extracting the relevant information is a highly nontrivial task.

Several computational methods have been suggested to identify the metabolites from MS/MS spectra. Mass spectral databases (spectral libraries) have been created (e.g., Smith et al., 2005; Hisayuki et al., 2010; Oberacher et al., 2009; Tautenhahn et al., 2012) which allow us to search measured mass spectra. Unfortunately, this approach can only identify “known unknowns” where a reference measurement is available.

Fragmentation trees are combinatorial models of the MS/MS fragmentation process. Böcker and Rasche (2008) suggested fragmentation trees for identifying the molecular formula of an unknown compound. Later, fragmentation trees were shown to contain valuable structural information about the compound (Rasche et al., 2011, 2012).

The relation between spectral similarity and structural similarity has been studied by Demuth et al. (2004). A kernel-based machine learning approach for metabolite identification was recently introduced by Heinonen et al. (2012), relying on predicting the molecular fingerprints as an intermediate step. Molecular fingerprints are given as bit vectors with each bit describes the existence of certain molecular property such as sub-structures in the molecule. After the prediction, imposing some scoring strategy, the predicted molecular fingerprints are used for searching some chemical database and finally a ranked list of candidates are generated (Shen et al., 2013; Heinonen et al., 2012).

Besides these two approaches, methods have been suggested for predicting MS/MS spectra from molecular structures (Allen et al., 2013; Kangas et al., 2012); there also exist commercial software packages for this task. Such simulated spectra can be used to replace the notoriously incomplete spectral libraries by molecular structure databases (Hill et al., 2008). Combinatorial fragmentation of molecular structure serves the same purpose (Wolf et al., 2010; Gerlich and Neumann, 2013). Finally, we can search spectral libraries for similar compounds, by comparing either MS/MS spectra (Demuth et al., 2004; Gerlich and Neumann, 2013) or fragmentation trees (Rasche et al., 2012). See (Scheubert et al., 2013) and (Hufsky et al., 2014) for recent reviews.

We propose a joint strategy which combines fragmentation trees and multiple kernel learning (MKL) to improve molecular fingerprint prediction and, subsequently, the metabolite identification. We first outline the metabolite identification framework and introduce fragmentation trees and their computation. Next, we introduce a family of kernels for fragmentation trees, consisting of simple node and edge statistics kernels as well as path and subtree kernels that use dynamic programming for efficient computation. We then describe state-of-the-art methods for multiple kernel learning. In the experiments, we evaluate different MKL algorithms.
The metabolite identification framework through multiple kernel learning. First, we construct the fragmentation tree from the MS/MS spectrum. Second, we compute kernels for both MS/MS data and fragmentation trees. Third, multiple kernel learning is used to combine kernels and predict molecular fingerprints. Finally, fingerprints are used for molecular structure database retrieval.

with regards to the fingerprint prediction and the metabolite identification.

2 METHODS

Fig. 1 gives an overview for our metabolite identification framework through multiple kernel learning. Fragmentation trees are computed first, followed by the computation of kernels. Multiple kernel learning approaches are used to integrate different kernels for molecular fingerprint prediction. The final step of the framework is to query molecular structure databases with the predicted molecular fingerprint using a probabilistic scoring function.

The advantages of the kernel-based machine learning framework are that it easily allows to incorporate the combinatorial fragmentation trees by kernelizing the model; that it can query molecular structure databases which are much larger than MS/MS spectral libraries; and, that molecular fingerprints can help to characterize the unknown metabolite and may shed light for de-novo identification.

2.1 Fragmentation trees

Böcker and Rasche (2008) introduced fragmentation trees to predict the molecular formula of an unknown compound using its MS/MS spectra. A fragmentation tree annotates the MS/MS spectra of a compound via assumed fragmentation processes. Nodes are molecular formulas, representing the unfragmented molecule and its fragments. Edges represent fragmentation reactions between fragments, or the unfragmented molecule and a fragment. Details on the computation can be found in (Böcker and Rasche, 2008; Rasche et al., 2011). Here, we quickly recapitulate the method. We assume that MS/MS spectra recorded at different collision energies have been amalgamated into a single spectrum, as described in the Results section. We decompose all peaks in the amalgamated spectrum, finding all molecular formulas that are within the mass accuracy of the measurement. For each decomposition of the parent peak, we build a fragmentation graph which contains all possible explanations for each peak, where nodes are colored by the peaks they originate from. We insert all edges between nodes that are not ruled out by the molecular formulas: That is, a product fragment can never gain atoms of any element through the fragmentation. Edges of this graph are then weighted, taking into account the intensity and mass accuracy of the product fragment, the mass of the loss, and prior knowledge about the occurrence of certain losses.

Under the parsimony assumption, we then compute a colorful subtree of this graph with maximum weight. Unfortunately, finding this tree is an NP-hard problem (Rauf et al., 2012). Nevertheless, we can compute optimal trees in a matter of seconds using Integer Linear Programming (Rauf et al., 2012). For each peak, this tree implicitly decides whether it is noise or signal and, in the later case, assigns the molecular formula of the corresponding fragments and the fragmentation reaction it resulted from. The score of the tree is the sum of its edge weights. Candidate molecular formulas of the parent peak are ranked by this score, which is the maximum score of any tree that has this molecular formula as its root.

Different from (Böcker and Rasche, 2008; Rasche et al., 2011) we use a modified weighting function for the edges of the fragmentation graph. With this new weights, the above optimization can be interpreted as a maximum a posteriori estimator of the observed data. We weight edges by the logarithmic likelihood that a certain fragmentation reaction occurs: For this, we consider the intensity and mass deviation of the product fragment peak, the loss mass, and chemical properties of the molecular formula as proposed in Kind and Fiehn (2007): namely, the ring double bond equivalent and the hetero atoms and carbon atoms ratio. Furthermore, we favor a few common losses that were learned from the data, and penalize implausible losses and radicals. Such weights have already been used in (Böcker and Rasche, 2008; Rasche et al., 2011); different from there, we did not choose parameters ad hoc but rather learned them from the data. Details about these new weights will be published elsewhere.

2.2 Kernels for Fragmentation trees and MS/MS spectra

2.2.1 Probability product kernel. Heinonen et al. (2012) compared several kernels that can be computed directly from the MS/MS spectra without the knowledge of the fragmentation trees. In their studies, simple peak and loss matching kernels were found inferior to the probability product kernel. Thus, we use the probability product kernel as the baseline comparison to the fragmentation tree kernels. The idea of the probability product kernel is the following: each peak in a spectrum is modeled by
a two-dimensional Gaussian distribution with the mass-to-charge ratio as one dimension, and the intensity as the other. All against all matching between the Gaussians is performed to avoid problems arising from alignment errors.

Formally, a spectrum is defined as \( \chi = \{\chi(1), \ldots, \chi(\ell)\} \), a set of \( \ell \) peaks \( \chi(k) = (\mu(k), \epsilon(k)) \in \mathbb{R}^2, (k = 1, \ldots, \ell) \) consisting of the peak mass \( \mu(k) \) and the normalized peak intensity \( \epsilon(k) \). The \( k \)th peak of the mass spectrum \( \chi \) is represented by \( p_k = N(\chi(k), \Sigma) \) centered around the peak measurement and with covariance shared with all peaks

\[
\Sigma = \begin{bmatrix}
\sigma^2_x & 0 \\
0 & \sigma^2_\epsilon
\end{bmatrix}
\]

where the variances \( \sigma^2_x \) for the mass is estimated from data and and \( \sigma^2_\epsilon \) is tuned by cross validation. No covariance is assumed between peak distributions. The spectrum \( \chi \) is finally represented as a mixture of its peak distributions \( p_k \) as \( \frac{1}{\ell} \sum_{k=1}^{\ell} p_k \).

The probability product kernel (PPK) \( K_{\text{peaks}} \) [Jebara et al., 2004] between the peaks of two spectra \( \chi, \chi' \) is given by:

\[
K_{\text{peaks}}(\chi, \chi') = K(p_{\chi}, p_{\chi'})
= \int_{\mathbb{R}^2} p_{\chi}(x)p_{\chi'}(x)dx
= \frac{1}{\ell_x \ell_{\chi'}} \sum_{k,k'=1}^{\ell_x, \ell_{\chi'}} \frac{1}{\sqrt{4\pi\sigma_x \sigma_{\epsilon}}} \exp\left(-\frac{1}{2}\right) (\chi(k) - \chi'(k'))^T \Sigma^{-1} (\chi(k) - \chi'(k')).
\]

The precursor ion is the compound selected in the first round of MS/MS and further fragmented in the second round. As a result, the difference (loss) between the peak \( \chi(k) \) and the precursor ion \( p_{\chi}(\cdot, 0) \) are also important where \( \mu(p) \) is the mass of the precursor ion. We can model the difference with distribution \( p_k = N(\chi(k), \Sigma) \) where \( \chi(k) = |\text{prec}(\chi) - \chi(k)| \). This feature is denoted as loss and corresponding kernel matrix as \( K_{\text{loss}} \).

Experiments in [Heinonen et al., 2012; Shen et al., 2013] showed that the combined kernel \( K_{\text{peaks}} + K_{\text{loss}} \) achieved best accuracy and computational efficiency among the spectral kernels.

2.2.2 Fragmentation tree kernels. Fragmentation trees can be considered as an annotated representation of the original MS/MS spectra. Recent advancement [Rojas-Chertó et al., 2012; Rasche et al., 2012] in comparing and aligning the fragmentation trees enables similarity metrics to be defined between fragmentation patterns for small molecules. Rasche et al. [2012] introduced fragmentation tree alignment, and showed alignment scores to be correlated with chemical similarity. However, alignment scores of this type do not in general yield positive semidefinite kernels. In the following, we define a set of kernels for fragmentation trees that will allow us to transfer the power of the fragmentation tree approach to the kernel-based learning algorithms for molecular fingerprint prediction and metabolite identification.

A fragmentation tree \( T = (V, E) \) consists of a nodes set \( V \) of molecular formulas (corresponding to the fragments) and an edges set \( E \subseteq V \times V \) (corresponding to the losses). Let \( v \) denote the root of \( T \). For an edge \( e = (u, v) \in E \) let \( \lambda(e) = \lambda(u, v) := u - v \) be the molecular formula of the corresponding loss. Clearly, different edges may have identical losses; let \( \lambda(E) \) be the multiset of all losses. For some loss molecular formula \( l \), let \( N(l) \) be the number of edges \( e \in E \) with \( \lambda(e) = l \). Each path from the root \( r \) to a node \( v \) implies a root loss \( r - v \) let \( E := \{ r - v \mid v \in V \} \) be the set of all root losses. For a MS/MS spectrum \( x \) let \( T_x = (V_x, E_x) \) be the corresponding fragmentation tree, with root losses \( E_x \) and loss multiplicities \( N_x(\cdot) \). For any node \( v \in V_x \) let \( \lambda_x(v) \) be the corresponding peak intensity; for an edge \( e = (u, v) \in E_x \) let \( \lambda_x(e) \) be the intensity of the terminal node \( v \). For the loss-based kernels and node-based kernels, feature vectors \( \phi \) are constructed and the kernel function is just a simple dot product between two feature vectors. Path-based kernels are more complicated, and details on their computation will be given below.

Loss-based kernels. Edges in the fragmentation trees represent the losses from the parent node to the child node. The following feature vectors are devised based on the losses in a fragmentation tree \( T_v \):

- LB: Loss binary, indicates the presence of a loss \( l \) in a fragmentation tree \( T_v \), that is, \( \phi_{LB}^l(x) = \mathbb{I}_{\lambda(l)(x)} \).
- LC: Loss count, counts the number of occurrences of a loss \( l \) in a fragmentation tree \( T_v \), that is, \( \phi_{LC}^l(x) = N_x(l) \).
- LI: Loss intensity, uses the average intensity of the terminal nodes with loss \( l \) in a fragmentation tree \( T_v \), that is, \( \phi_{LI}^l(x) = \frac{1}{N_x(l)} \sum_{e \in E_x, \lambda(e) = l} \lambda_x(e) \).
- RLB: Root loss binary, indicates the presence of a root loss \( l \) in a fragmentation tree \( T_v \), that is, \( \phi_{RLB}^l(x) = \mathbb{1}_{\lambda(l)(x)} \).
- RLI: Root loss intensity, uses the intensity of the terminal node of a root loss if it is present in a fragmentation tree \( T_v \). For root \( r \) we set \( \phi_{RLI}^{r-l}(x) = \lambda_x(r-l) \) if \( r - l \in V_x \), and \( \phi_{RLI}^{r-l}(x) = 0 \) otherwise.

Node-based kernels. The nodes in the fragmentation tree explain peaks in the MS/MS by some chemical formula of the hypothetical fragment. The nodes are unique in a fragmentation tree \( T_v \), and so are the root losses. To this end, we can omit root losses from the feature vectors.

- NB: Nodes binary, indicates the presence of a node \( v \) in a fragmentation tree \( T_v \), that is, \( \phi_{NB}^v(x) = \mathbb{I}_{\text{in}v} \).
- NI: Nodes intensity, uses the intensity of the node if it is present in a fragmentation tree \( T_v \), that is, \( \phi_{NI}^v(x) = \lambda_x(v) \) for \( v \in V_x \), and \( \phi_{NI}^v(x) = 0 \) otherwise.

Path-based kernels. These kernels are count common path between two fragmentation trees: Here, “common path” refers to an identical sequence of losses in the two trees. We use dynamic programming (DP) to efficiently count the number of common paths, that is, the dot product of two feature vectors which are not explicitly constructed. For two fragmentation trees \( T_1 = (V_1, E_1) \) and \( T_2 = (V_2, E_2) \) we compute a DP table \( D[u, v] \) for all \( u \in V_1 \) and \( v \in V_2 \). In all cases, the number of common paths is \( D[r_1, r_2] \) where \( r_i \) is the root of \( T_i \). We initialize

\[
D[u, v] = 0, \forall u \in \mathcal{L}(T_1), v \in T_2
D[u, v] = 0, \forall u \in T_1, v \in \mathcal{L}(T_2)
\]
where \( \mathcal{L}(T) \) denotes the leaves of a tree \( T \). Let \( C(v) \) be the children of a node \( v \).

- **Common path counting (CPC).** The DP table entry \( D[u,v] \) records the count of common path for the subtrees rooted in \( u \) and \( v \), respectively. This leads to the following recurrence:
  \[
  D[u,v] = \sum_{a \in C(u), b \in C(v), \lambda(u,a)=\lambda(v,b)} (1 + D[a,b]).
  \]

- **Common paths of length 2 (CP2).** In this case, only common losses for paths of length two are considered:
  \[
  D[u,v] = \sum_{a \in C(u), b \in C(v), \lambda(u,a)=\lambda(v,b)} (1 + D[a,b]).
  \]

- **Common path with \( K_{peaks} \) (CPK).** Instead of simply counting the common paths, we use the probability product kernel \( K_{peaks} \) to score the terminal peaks. We omit the straightforward but somewhat tedious details.

- **Common subtree counting (CSC).** In this case, we count the number of “common subtrees” between \( T_1 \) and \( T_2 \), which can be defined analogously to the common paths above. Entry \( D[u,v] \) now counts the number of common subtrees for the two subtrees rooted in \( u \) of \( T_1 \), and \( v \) of \( T_2 \). We have to consider three cases: For each pair of children \( a \in C(u) \) and \( b \in C(v) \) with \( \lambda(u,a)=\lambda(v,b) \) we can either attach the subtrees rooted in \( a \) and \( b \); we can use solely the edges \( (u,a) \) and \( (v,b) \) as a common subtree; or, we can attach no common subtree for this pair of children. But if we choose no subtree for all matching pairs of children, the result would be a tree without edges and, hence, not a valid common subtree. Thus, we have to correct for this case by subtracting one. Hence, the recurrence is:
  \[
  D[u,v] = \prod_{a \in C(u), b \in C(v), \lambda(u,a)=\lambda(v,b)} (2 + D[a,b]) - 1.
  \]

2.3 Multiple kernel learning

In many applications, multiple kernels from different kernel functions or multiple sources of information are available. Multiple Kernel Learning (MKL) becomes a natural way to combine information contained in the kernels. Instead of choosing the best kernel via cross validation as in [Heinonen et al., 2012; Shen et al., 2013], multiple kernel learning seeks a linear, convex or even non-linear combination of the kernels. An overview of MKL algorithms can be found in a survey by Gönen and Alpaydîn [2011].

In practice, it is often difficult for MKL algorithms to outperform the uniform combination of the kernels (UNIMKL) where the weights for kernels are equal. However, in some cases, some methods have seen improvements over the uniform combinations. Three algorithms coupled with SVM are considered in the following: centered alignment-based algorithms [Cortes et al., 2012], quadratic combination of the kernels [Li and Sun, 2010] and \( \ell_p \)-norm \( p \geq 1 \) for the kernel weights [Kloft et al., 2011].

For all three algorithms, the input will be a set of kernels \( K = \{K_i, K_i \in \mathbb{R}^{n \times n}, k = 1, \ldots, q \} \) computed from a data points. The output is a set of \( m \) fingerprint properties \( Y \in \{-1, +1\}^{n \times m} \) which is a multi-label prediction task and each label is trained independently in the experiments.

2.3.1 Centered alignment-based multiple kernel learning. The centered alignment based multiple kernel learning algorithms are based on the observation that the centered alignment score with the target kernel \( K_Y = YY^T \) correlates very well with the performance of the kernel, where \( y \) is a single label. Experiments by [Cortes et al., 2012] show consistently improvements over the uniform combination. In the molecular fingerprint prediction setting, the target kernel is defined as \( K_Y = YY^T \).

Two-stage model are considered in which the kernel weights are learned first and then can be applied to all kernel-based learning algorithms (SVM in this work). The centered kernel matrices are defined by equation (1):

\[
K_c = [I - \text{ee}^T/n][I - \text{ee}^T/n]
\]

where \( I \) is the identity matrix and \( e \) is the vector with all ones. \( \forall A, B \in \mathbb{R}^{n \times n}, \langle \cdot, \cdot \rangle_F \) denotes the Frobenius product and \( \| \cdot \|_F \) denotes the Frobenius norm which are defined by

\[
\langle A, B \rangle_F = Tr[A^T b] \quad \text{and} \quad \|A\|_F = \sqrt{\langle A, A \rangle_F}.
\]

Let now \( K \in \mathbb{R}^{n \times n} \) and \( K' \in \mathbb{R}^{n \times n} \) be two kernel matrices such that \( \|K\|_F \neq 0 \) and \( \|K'\|_F \neq 0 \). Then the centered alignment between \( K \) and \( K' \) is defined by

\[
\hat{b}(K, K') = \frac{\langle K, K' \rangle_F}{\|K\|_F \|K'\|_F}.
\]

The simple independent centered alignment-based algorithm (ALIGN) [Cortes et al., 2012] computes the alignment score between each kernel matrix \( K_i \) and the target kernel matrix \( K_Y \) and combine the kernels as

\[
K_{\mu} \propto \sum_{i=1}^{q} \hat{b}(K_i, K_Y)K_i
\]

\[
= \frac{1}{\|K_Y\|_F} \sum_{i=1}^{q} \langle K_i, K_Y \rangle_F K_i.
\]

The alignment maximization algorithm (ALIGNF) [Cortes et al., 2012] jointly seeks the weight \( \mu_i \) to maximize the alignment score defined by equation (2) between the convex combination of the kernel in \( K \) and the target kernel \( K_Y = YY^T \), that is, the following optimization problem:

\[
\max_{\mu \in \mathcal{M}} \frac{\langle K_{\mu}, K_Y \rangle_F}{\|K_{\mu}\|_F}
\]

where \( \mathcal{M} : \|\mu\|_2 = 1, \mu \geq 0 \).

2.3.2 Quadratic combination multiple kernel learning. In this setting, the quadratic combination of kernels (QCMKL) is included in the formulation and the multiple kernel learning problem is solved by semidefinite programming [Lanckriet et al., 2002; Li and Sun, 2010]. The kernels in \( K \) are enriched to a new set \( \tilde{K} = \{K_i|\tilde{K}_i|\tilde{K} = 1, \ldots, q(q + 1)/2\} \) by the following transformation:

\[
\tilde{K}_{i,j} = \begin{cases} K_i \circ K_j, & i \neq j \\ K_i, & i = j \end{cases}
\]

where \( i, j = 1, \ldots, q \) and \( \circ \) denotes the Hadamard product.
The convex combinations of the kernels is given by $\tilde{K}_p = \sum_{\ell=1}^{q^{(\ell+1)/2}} \mu_\ell K_\ell$ with $\mu \geq 0$ and $e^T \mu = 1$. Adapting the soft margin SVM formulation reveals the following dual problem (in epigraph form) (Li and Sun [2010]):

$$ \max_{\alpha, \mu} \quad u $$

s.t. $u \geq \alpha^T e - \frac{1}{2} \alpha^T G(\tilde{K}_p) \alpha$

$$ 0 \leq \alpha \leq C e, \alpha^T y = 0, \mu \geq 0, e^T \mu = 1. $$

The derived Lagrangian for the problem is (Li and Sun [2010]):

$$ L(\alpha, \beta, \delta, \gamma) = \alpha^T e - \frac{1}{2} \alpha^T G(\tilde{K}_p) \alpha + \beta \alpha^T \gamma + \delta(\beta e - \alpha) $$

with $\alpha, \beta \geq 0, \delta \geq 0, \gamma$ as dual variables, and $G(K) = \alpha^T \text{diag}(y) K \text{diag}(y) \alpha$. Applying Schur’s lemma to convert the first inequality constraint to Linear Matrix Inequality (LMI) unveils the following semidefinite programme (SDP) (Li and Sun [2010]):

$$ \min_{\alpha, \mu} \quad u $$

s.t. $G(\tilde{K}_p) e + \beta \gamma - \delta^T u - 2C \delta^T e \geq 0$

$$ \mu \geq 0, e^T \mu = 1, \beta \geq 0, \delta \geq 0. $$

Many standard SDP solvers can be used to find the optimal solutions such as cvx.

### 2.3.3 $p$-norm multiple kernel learning

While $\ell_1$ norm on the kernel weights $\mu$ produces sparse solutions, higher norms $p > 1$ produces non-sparse solutions which may be beneficial. A general framework for $p$-norm multiple kernel learning ($\ell_p$-MKL) was proposed by Kloft et al. [2011]. The $q$ kernels corresponds to $q$ feature mappings $\Psi_k : \chi \to \mathcal{H}_k, k = 1, \ldots, q$ and $l$ is some convex loss function and the primal problem is then:

$$ \min_{w, b, \mu} \quad C \sum_{i=1}^l \sum_{k=1}^q (w_k, \Psi_k(x_i))_H + b_i y_i + \frac{1}{2} \sum_{k=1}^q \frac{|w_k|^2}{\mu_k} $$

s.t. $\mu \geq 0, ||\mu||^p \leq 1.$

When the optimization is coupled with hinge loss, the problem has a simple dual form (Kloft et al. [2011]):

$$ \max_\alpha \quad \alpha^T e - \frac{1}{2} \sum_{k=1}^q ||(\alpha^T G(\tilde{K}_k) \alpha)^T||^p, $$

where all the variables are all as defined before but $p^* = \frac{p}{p-1}$. The optimization problem can be solved by alternating the dual variables $\alpha$ and the kernel weights $\mu$ via the squared norm on $w$ by the following equations:

$$ ||w_k||^2 = \mu_k^2 \alpha^T K_k \alpha, \forall k = 1, \ldots, q. $$

$$ \mu_k = \frac{||w_k||^2}{(\sum_{k=1}^q ||w_k||^2)^{\frac{p}{2}}}, \forall k = 1, \ldots, q. $$

Based on the above equations, a simple alternating algorithm has been proposed by Kloft et al. [2011] as Algorithm 1.

### 2.4 Probabilistic scoring of candidate metabolites

Given a predicted fingerprint associated with a mass spectrum, for metabolite identification, we need to retrieve metabolites with similar fingerprints from a molecular database. Assume $\hat{y} \in \{-1, +1\}^m$ is a predicted fingerprint and an arbitrary fingerprint $y \in \{-1, +1\}^m$ for some molecule in some molecular database, one can score the $y$ by the following equation as used in FingerID [Shen et al., 2013] and Heinonen et al. [2012]:

$$ P_{FB}(y | \gamma) = \prod_{j=1}^m \gamma_j^y v_j = \gamma_j \prod_{j=1}^m (1 - \gamma_j)^{1 - v_j} $$

that is, the Poisson-Binomial probability for the fingerprint vector $y$ where the cross-validation accuracies $(\gamma_j)_{j=1}^m \in [0.5, 1]$ of the fingerprints prediction are taken as the reliability scores.

### 3 RESULTS

Two MS/MS datasets, 978 compounds downloaded from METLIN [Tautenhahn et al., 2012] and 402 compounds from MassBank [Hisayuki et al., 2010], both measured by QTOF MS/MS instruments are tested. For each compound, mass spectra recorded at different collision energies were amalgamated before further processing; We normalize MS/MS spectra such that intensities sum up to 100%. We merge peaks from different collision energies with m/z difference at most 0.1, using the m/z of the highest peak and summing up intensities. We discard all but the 30 highest peaks, as well as peaks with relative intensity below 0.5%.

Next, we compute the fragmentation tree. We assume that we can identify the correct molecular formula from the data; Limiting candidate molecular formulas to those present in KEGG [Kanehisa and Goto, 2000], which is used for searching molecular structures.

below, the best-scoring fragmentation tree identified the correct molecular formula of the compound in 97.1% (96.0%) of the cases for the METLIN (MassBank) dataset. Integrating other sources of information such as MS1 isolate patterns (Bocker et al. 2009) or retention times would reach even better identification rates. To allow for a meaningful comparison of the power of the different kernels, we therefore use the best-scoring fragmentation tree of the correct compound molecular formula.

All 11 fragmentation tree kernels proposed in the previous section were computed, along with probability product kernel used in Heinonen et al. [2012] Shen et al. [2013] computed directly from MS/MS, resulting in 12 kernels to be evaluated.

Molecular fingerprints were generated using OpenBabel (O’Boyle et al. 2011) which contains four types of fingerprint[1] FP3, FP4 and MACCS fingerprints (528 bits in total) were generated based on the software predefined SMARTS patterns. In our dataset, more than half of the fingerprint properties have high class bias rate, with a large majority of the dataset belonging to the positive class (most compounds match the property) or respectively the negative class (most compounds do not match the property). For such fingerprints, the default classifier, one that always predicts the majority class, has high accuracy, although the model is not meaningful. For our performance comparisons we opted to only include fingerprints with class bias rate below 0.9.

For each fingerprint property, we separately trained a SVM; for all properties, we used identical training and testing compounds. Five fold cross validation was performed and the SVM margin softness parameter ($C \in \{2^{-3}, 2^{-2}, \ldots, 2^{6}, 2^{7}\}$) was tuned based on the training accuracy.

### 3.1 Fingerprint prediction performance

The micro-average (simultaneous average over fingerprint properties and compounds) accuracy and F1 of the individual kernels on the predictions of fingerprint properties with bias rate less than 0.9 are shown in Table 1 with the standard deviations computed from all 11 cross validation folds. The kernel NB achieves the best accuracy and F1 on both METLIN and MassBank. Compared to the probability product kernel, the fragmentation tree kernels are markedly more accurate on average.

The improvement of MKL approaches over single kernel SVMs are clear. The t-test between NB and ALIGNF shows the differences of mean accuracy and F1 are indeed very significant with p-values of $4 \times 10^{-8}$ and $1.7 \times 10^{-3}$, respectively.

The micro-average accuracy and F1 of the MKL algorithms on the fingerprint properties predictions are shown in Table 2 where it can be concluded that averaged over all fingerprints the MKL methods are quite close. We conducted further pairwise difference testing, where the performance difference of each method on each individual fingerprint property is evaluated. Table 2 shows the significance level of the sign test on the accuracy and F1 on the METLIN and MassBank datasets using the different MKL methods. The sign test describes whether one of the methods has higher probability of success (better than the other on a fingerprint) than the other (alternative hypothesis) or not (null hypothesis). From the table we can deduce that ALIGN and ALIGNF rise slightly above the competition whereas $\ell_2$-MKL and QCMKL are slightly inferior to the rest. The performance of UNIMKL is also respectable.

### 3.2 Metabolite identification performance

The molecular fingerprint prediction can serve as an intermediate step for metabolites identification, and can be used to search a molecular structure database (Shen et al. 2013, Heinonen et al. 2012). We want to evaluate whether improvements in fingerprint prediction propagate to better metabolites identifications. We will search for molecular structures from the KEGG database. As we assume to know the correct molecular formula, we may filter based on this information to generate our candidate lists. But it turns out that this filter is too strict for a meaningful evaluation, as the number of candidates for each MS/MS spectrum becomes very small and, hence, all kernels show good performance. For a more discriminative evaluation of the kernels, we artificially enlarge the set of candidates: We use all molecular structures in KEGG with large mass accuracy $\mu_M$ and $\Delta$ this results in candidate lists that allow a meaningful comparison of the kernels.

For identification, we want the true molecular structure to be ranked as high as possible in the candidates list. Fig. 2(a) and (b) show the fraction of compounds that were ranked higher than certain

#### Table 1. Micro-average performance of individual kernels. PPK is the method from Heinonen et al. [2012] we compare against.

<table>
<thead>
<tr>
<th></th>
<th>METLIN Acc(%)</th>
<th>MassBank Acc(%)</th>
<th>METLIN F1(%)</th>
<th>MassBank F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>79.5 ± 0.5</td>
<td>69.9 ± 0.9</td>
<td>78.9 ± 1.0</td>
<td>69.0 ± 2.2</td>
</tr>
<tr>
<td>LC</td>
<td>79.4 ± 0.3</td>
<td>69.6 ± 0.4</td>
<td>78.5 ± 1.2</td>
<td>68.4 ± 2.7</td>
</tr>
<tr>
<td>LI</td>
<td>77.8 ± 0.5</td>
<td>66.8 ± 0.7</td>
<td>77.4 ± 1.0</td>
<td>66.7 ± 2.0</td>
</tr>
<tr>
<td>RLB</td>
<td>81.6 ± 0.8</td>
<td>73.2 ± 1.1</td>
<td>78.6 ± 1.0</td>
<td>68.4 ± 1.2</td>
</tr>
<tr>
<td>RLI</td>
<td>78.4 ± 0.6</td>
<td>68.5 ± 0.8</td>
<td>76.7 ± 0.9</td>
<td>65.4 ± 1.6</td>
</tr>
<tr>
<td>NB</td>
<td>81.9 ± 0.4</td>
<td>73.9 ± 0.3</td>
<td>81.4 ± 0.7</td>
<td>73.2 ± 1.2</td>
</tr>
<tr>
<td>NI</td>
<td>80.3 ± 0.7</td>
<td>71.1 ± 0.8</td>
<td>79.8 ± 1.0</td>
<td>70.5 ± 0.9</td>
</tr>
<tr>
<td>CPC</td>
<td>80.6 ± 0.5</td>
<td>71.6 ± 0.7</td>
<td>78.7 ± 1.4</td>
<td>68.9 ± 2.4</td>
</tr>
<tr>
<td>CP2</td>
<td>78.7 ± 0.7</td>
<td>68.4 ± 1.2</td>
<td>76.4 ± 1.0</td>
<td>65.5 ± 1.1</td>
</tr>
<tr>
<td>CPK</td>
<td>72.9 ± 0.3</td>
<td>58.8 ± 0.5</td>
<td>72.2 ± 0.6</td>
<td>57.9 ± 0.5</td>
</tr>
<tr>
<td>CSC</td>
<td>74.9 ± 0.4</td>
<td>61.9 ± 0.8</td>
<td>77.8 ± 0.8</td>
<td>67.2 ± 2.0</td>
</tr>
<tr>
<td>FPK</td>
<td>76.7 ± 0.6</td>
<td>64.0 ± 0.7</td>
<td>72.9 ± 1.1</td>
<td>58.6 ± 1.2</td>
</tr>
</tbody>
</table>

#### Table 2. Micro-average performance of MKL algorithms.

<table>
<thead>
<tr>
<th></th>
<th>METLIN Acc(%)</th>
<th>MassBank Acc(%)</th>
<th>METLIN F1(%)</th>
<th>MassBank F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIMKL</td>
<td>85.0 ± 0.6</td>
<td>78.3 ± 0.7</td>
<td>82.2 ± 0.6</td>
<td>73.9 ± 1.5</td>
</tr>
<tr>
<td>ALIGN</td>
<td>85.2 ± 0.6</td>
<td>78.6 ± 0.7</td>
<td>82.4 ± 0.7</td>
<td>74.4 ± 1.4</td>
</tr>
<tr>
<td>ALIGNF</td>
<td>85.0 ± 0.5</td>
<td>78.6 ± 0.4</td>
<td>82.8 ± 0.4</td>
<td>75.2 ± 1.2</td>
</tr>
<tr>
<td>QCMKL</td>
<td>84.9 ± 0.5</td>
<td>77.8 ± 0.5</td>
<td>82.1 ± 0.6</td>
<td>74.0 ± 0.7</td>
</tr>
<tr>
<td>$\ell_2$-MKL</td>
<td>84.7 ± 0.5</td>
<td>77.5 ± 0.5</td>
<td>82.2 ± 0.5</td>
<td>74.0 ± 0.9</td>
</tr>
<tr>
<td>$\ell_3$-MKL</td>
<td>85.2 ± 0.6</td>
<td>78.5 ± 0.7</td>
<td>82.4 ± 0.6</td>
<td>74.4 ± 1.3</td>
</tr>
<tr>
<td>$\ell_4$-MKL</td>
<td>85.2 ± 0.6</td>
<td>78.5 ± 0.8</td>
<td>82.3 ± 0.6</td>
<td>74.2 ± 1.0</td>
</tr>
<tr>
<td>$\ell_5$-MKL</td>
<td>85.1 ± 0.6</td>
<td>78.5 ± 0.7</td>
<td>82.3 ± 0.6</td>
<td>74.1 ± 1.3</td>
</tr>
</tbody>
</table>

Fig. 2. Figure (a) and (b) show the performance for identification when searching KEGG using 300 ppm mass window with predicted molecular fingerprints, with fingerprints trained with METLIN and MassBank datasets, respectively. NUM denotes the number of candidate molecules returned per query. Figure (c) and (d) show the proportion of data that were correctly identified in the top 1 rank against a series of mass windows.

We notice that the NB kernel is consistently more accurate than PPK. In addition, Multiple kernel learning clearly improves the identification performance, especially the number of top ranked identifications increases significantly. T-test between the ranks of the ALIGNF and PPK shows a p-value of 0.06 which verifies the improvements in identification by ALIGNF over the PPK is indeed significant. ALIGNF comes on top of the MKL approaches, which is in line with its good fingerprint prediction accuracy and F1 score.

The effect of mass accuracy windows during the database retrieval are shown in Figure (c) and (d). A narrower 20ppm mass search window filters out many false candidates and thus significantly elevate the identification accuracies to 60% on METLIN dataset and 40% on MassBank dataset. However, the effect of improved molecular fingerprint prediction is softened due to the fewer but possibly more similar candidates. An extreme case is observed in Fig. (d) in which all the methods shrink to the same result when searching with 20ppm mass accuracy window.
Table 3. Sign test for the performance of MKL algorithms on the METLIN and MassBank datasets. “+” indicates the method in the row is better than the method in the column (“−” otherwise) with significance p-value between 0.01 and 0.05; Blank indicates no significance. Similarly, “++” and “−−” indicate significance with p-value less than 0.01. Upper table is for accuracy and lower table is for F1.

<table>
<thead>
<tr>
<th></th>
<th>METLIN</th>
<th>MassBank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIMKL</td>
<td>ALIGNF</td>
</tr>
<tr>
<td>UNIMKL</td>
<td>−−−−−</td>
<td>+++</td>
</tr>
<tr>
<td>ALIGN</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>ALIGNF</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>QCMLK</td>
<td>−−−−−</td>
<td>−−−−−</td>
</tr>
<tr>
<td>ℓ3-MKL</td>
<td>−−−−−</td>
<td>−−−−−</td>
</tr>
<tr>
<td>ℓ4-MKL</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>ℓ5-MKL</td>
<td>++</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>METLIN</th>
<th>MassBank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UNIMKL</td>
<td>ALIGNF</td>
</tr>
<tr>
<td>UNIMKL</td>
<td>−−−−−</td>
<td>−−−−</td>
</tr>
<tr>
<td>ALIGN</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>ALIGNF</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>QCMLK</td>
<td>−−−−−</td>
<td>−−−−−</td>
</tr>
<tr>
<td>ℓ3-MKL</td>
<td>−−−−−</td>
<td>−−−−−</td>
</tr>
<tr>
<td>ℓ4-MKL</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>ℓ5-MKL</td>
<td>++</td>
<td>+</td>
</tr>
</tbody>
</table>

4 DISCUSSION
The present work combines the combinatorial fragmentation tree approach with machine learning through a kernel-based approach. We suggest several kernels for fragmentation trees, and show how to fuse their information through multiple kernel learning. The result significantly enhances molecular fingerprint prediction and metabolite identification.

The most close analogs to our fragmentation tree kernels in literature are those defined for parse trees in natural language processing [Collins and Duffy (2001)]; our fragmentation trees can be seen as parses of the MS/MS spectra. Dynamic programming techniques similar to ours are used there for computing kernels between trees [Collins and Duffy (2001) Kuboyama (2007)]. However, fragmentation trees have important differences to the trees defined between parses of natural language and to kernels comparing molecular structures [Mahé and Vert (2009)]. Differently from natural language parses, the node labels have partial order (via their molecular weights) and also the edges have labels. Differently from kernels for molecular graphs, the label spaces of both nodes and edges are vast (subsets of molecular formulae).

The comparison to the probability product kernel employed by the FingerID [Heinonen et al. (2012)] software, show that the fragmentation tree kernels are able to extract more information out of the MS/MS spectra. Improvements are seen in both the prediction accuracy and the F1 score. Comparing to FingerID, already the uniform combination of the kernels (UNIMKL) improves the prediction performance significantly. As witnessed by many MKL applications, the UNIMKL algorithm is hard to beat. In our result, several MKL algorithms such as ALIGNF and ℓ3-norm MKL can give slightly better result than UNIMKL. The improvements in the molecular fingerprint prediction translate to improved metabolite identification.

There are several possible routes forward with the current metabolite identification framework. First, post-processing on the candidates list, such as the one proposed by [Allen et al. (2013)], is necessary when searching a large compound database such as PubChem, because the returned candidates (hundreds to thousands) may share the same fingerprints and there is no way to differ them based only on molecular fingerprints. Second, training a separate SVM for each fingerprint property is clearly an aspect that can be improved upon, for example by a multi-label classification approach. A still more tempting yet challenging direction would be to replace the two-step identification by an integrated prediction approach. Such an approach would potentially learn to predict the fingerprint properties that are important for discriminating metabolites from each other.

ACKNOWLEDGEMENT
This work was financially supported by Academy of Finland grant 268874 (MIDAS) and Deutsche Forschungsgemeinschaft grant BO 1910/16-1 (IDUN).

REFERENCES