Charles University in Prague
Faculty of Mathematics and Physics

Department of Theoretical Computer Science and Mathematical Logic

Computational Intelligence Methods in Metalearning

Summary of Doctoral Thesis
2016

Jakub Šmíd

Branch: Theoretical Computer Science

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Katedra teoretické informatiky a matematické logiky

Metody výpočetní inteligence pro metaučení

Autoreferát doktorské disertační práce
2016

Jakub Šmíd

Obor: Teoretická informatika

Disertační práce byla vypracována na základě výsledků, které byly získány během mého doktorského studia na Matematicko-fyzikální fakulty Univerzity Karlovy v Praze (2012-2016).

Doktorand:	Mgr. Jakub Šmíd
Školitel:	Mgr. Roman Neruda, CSc. Ústav informatiky AV ČR, v.v.i. Pod Vodárenskou věží 271/2 182 07 Praha, Česká republika
Školící pracoviště:	KTIML MFF Malostranské nám. 25 118 00 Praha, Česká republika
Předseda RDSO 4I1:	XXX KTIML MFF Malostranské nám. 25 118 00 Praha, Česká republika
Oponenti:	Mgr. Marta Vomlelová PhD. KTIML MFF Malostranské nám. 25 118 00 Praha, Česká republika
	dr. ir. Joaquin Vanschoren Technische Universiteit Eindhoven 5600 MB Eindhoven, Netherlands

Autoreferát byl rozeslán dne:

Obhajoba se koná dne XXX před komisí pro obhajoby doktorských disertačních oboru 4I1 – Teoretická informatika na MFF UK, Malostranské náměstí 25, 118 00 Praha, v místnosti č. XXX.

S disertací je možno se seznámit na studijním oddělení — doktorské studium MFF UK, Ke Karlovu 3, 121 16 Praha 2.

1 Introduction

The ongoing rapid growth of the available amount of data drives the urge for automated processing of such data. *Data mining* – the means of finding new patterns in *datasets* – is now widely used in medicine, economics, bioinformatics and other important areas of human interest. Many different algorithms exist and are used for this task of pattern extraction. However, even for an expert, it is hard to choose the most suitable algorithm for a particular dataset. According to the *No Free Lunch* theorem (NFL) stated in [18], the average performances of data mining algorithms on all data mining problems are equal. It means that elevated performance of any algorithm over one class of problems is paid for in performance over another class. However, NFL considers all possible problems. We are mostly interested in so called *real world* problems. Therefore, the key to success when dealing with a data mining real world problem is in binding the problem with an algorithm having elevated performance on the class of the problem. Since many fields depend on data mining techniques, it is crucial to propose and improve such bindings.

Metalearning [3] – the learning how to learn – can tackle the issue. The main idea behind metalearning is that *machine learning* methods are supposed to perform similarly on similar datasets. Therefore, the notion of dataset similarity is crucial. In most cases, the similarity is computed using data characterizing the datasets – the *metadata*. Metalearning techniques use the metadata and previous experience to predict the performance of machine learning methods on new datasets. In essence, metalearning does not differ much from the traditional machine learning. The main difference is that metalearning works with the metadata of given datasets instead of the actual data, and that the result of metalearning is a recommendation of a machine learning method to use.

The metadata may contain general information about the dataset, like the number of instances and attributes, the number of classes, performance of some algorithm over the dataset, etc. There is, however, a fixed amount of such general information extracted out of each dataset. Many traditional methods compare datasets based on such measures.

The attribute-specific metadata include more fine-grained information about the dataset. On the other hand, each dataset can have a different number of attributes (and consequently, different amount of attribute-specific metadata), which leads to a non-propositional representation of the datasets using attribute-specific metadata. The problem of defining the similarity on the non-propositional space is non-trivial as stated by Kalousis in [6]. The authors of [3] also explicitly recognize the problem of handling datasets described by the varying amount of metafeatures. In the past there were only few attempts to define the non-propositional similarity. Furthermore, most of these attempts loose important information in the process.

2 Goals of the Thesis

One of the goal of this thesis is to propose distance measures capable of handling a non-propositional representation. We take into consideration other methods dealing with non-propositional data, either directly in the metalearning field or in other fields as well.

Another goal of this thesis is to investigate theoretical properties of proposed distance measures and compare them with the related state of the art methods. We investigate whether the methods satisfy *metric* axioms, alternatively, what properties must be satisfied in order to do so.

The last goal of the thesis is to evaluate the performance of the proposed methods on real data. Several machine learning repositories are reviewed. The emphasis is put on the amount of data provided, types of metadata available, and the fact whether the data are publicly accessible allowing for the independent re-evaluation of our results.

3 Distance Using Attributes

The state of the art metalearning non-propositional approaches are either losing information or lacking properties usually important for the distance functions. Authors of [19] review possible ways of using the distance measures defined on $\mathbb{X}$ in order to define distance measures on the power set $2^{\mathbb{X}}$ of $\mathbb{X}$. The most promising is the distance measure defined as follows:

$$D_M(A,B) = \min_{R_i \in R} \Big(\sum_{(a_k,b_j)\in R_i} \delta(a_k,b_j) + (|B \setminus R_i(A)| + |A \setminus R_i^{-1}(B)|)\frac{M}{2} \Big), \quad (1)$$

where δ is a distance between elements of some space $\mathbb{S}$, $A, B \subseteq \mathbb{S}$, R_i is a matching (each element at most once) on $A \times B$, and M is a maximum distance of δ on $\mathbb{S}$. D_M can be also viewed as a distance that finds optimal mapping between elements of A, B and can decide to omit some elements by getting $\frac{M}{2}$ penalty. Authors of [9] prove that the D_M is a metric if δ is a metric. Furthermore, D_M is computable in the polynomial time. We can treat datasets as composed by attributes and use D_M by defining a distance δ on attribute space. However, there are still reasons that render this approach impractical for metalearning. The distance measure D_M requires M. The computation of M may not be feasible or not known in advance, especially if the attribute distance was not normalized. Additionally, some attributes may be more significant than the others. Not matching some insignificant attributes because of fewer attributes in the second dataset (for example with low entropy for predicting the target attribute) will still result in $\frac{M}{2}$ distance. Also, algorithm can decide not to match some attributes with high difference and get only $\frac{M}{2}$ penalty. For example, when comparing two datasets with single attributes, the results will be the same if the attributes are really distant (M) or somewhat distant ($\frac{M}{2}$). In both cases the D_M will return $\frac{M}{2}$.

In this section, we propose several approaches that can handle the non-propositional dataset representations without any trade-offs. We have already published their descriptions and experiments validating their asset in [13, 14, 12, 15, 16]. Each one is based on the idea of attribute aligning where the order of attributes is not important. By supplying a function measuring the distance between individual attributes (like in [6]), we could try to align the attributes in the way that minimizes the sum of distances between aligned attributes. To avoid confusion, we will always denote attribute distance measure as δ. The name was not chosen randomly. The upper case Δ – as always – will denote the final dataset distance measure. As we will see, we will piece Δ out of smaller δs, hence the name.

Although we are aiming to handle the non-propositional approach, we will start with propositional situation to describe the approach, and extend it later to handle varying amount of attributes. For now let us suppose that there is the same number of attributes in every dataset (and hence the same number of attribute metafeatures).

Definition 1. *Given a distance function between attributes δ, two datasets a, b and a bijection f between the attributes of a, b we define the dataset distance induced by the bijection between the datasets as:*

$$\Delta_f(a,b) = \sum_{k=1}^{n} (\delta(a_k, f(a_k))), \tag{2}$$

where a_k is the k-th attribute of a and $f(a_k)$ corresponding attribute in dataset b given by the bijection f. We will sometimes refer to the Δ_f as the cost *of f.*

We would like to match the attributes as best as possible to get the lowest possible distance Δ_f, so optimally we are looking for f^*:

$$f^* = \text{argmin}_f(\Delta_f). \tag{3}$$

We will denote f^* as an optimal alignment.

From this, the general distance between datasets can be derived:

$$\Delta(a,b) = \Delta_{f^*}(a,b). \tag{4}$$

Now suppose that the number of attributes is different. We transform this case to the previous one by adding dummy attributes into the dataset with less attributes. There are two approaches how to do this. Suppose $\mathbb{A}$ is a space of attributes. Either *dummy* $\in \mathbb{A}$ or *dummy* $\notin \mathbb{A}$. In the former case, nothing is needed, as the distance function between attributes is already defined if the *dummy* attribute is on the input. In the latter case, it is needed to extend the distance function by defining the distance between a regular attribute and the *dummy* one. In the further text, we will refer to the attribute space with the *dummy* attribute as to the *extended attribute space*. If it is clear from the context, we will sometimes refer to extended attribute space simply as attribute space. *Dummy* attribute will be referred to as *dummy* or specifically to

$dummy_a$ if the dummy attribute is already in the attribute space and *$dummy_n$* if the attribute is newly created.

From now on, we can suppose that if we are aligning two datasets that they have the same number of attributes, and that one dataset was to some extent enriched with some number of *dummy* attributes, so the amount of attributes matches.

There are $n!$ bijections from a to b. It is costly to enumerate them, so it is desired to eliminate some of the bijections in advance. In this work, we came up with two approaches that vary in the generality of the attribute distance and computational complexity.

Definition 2. Evaluation function σ *is a function mapping attributes to* $\mathbb{R}$.

In our first approach, we suppose we have σ (for example, number of distinct values), and the attribute distance is defined as follows:

$$\delta(a_k, f(a_k)) = |\sigma(a_k) - \sigma(f(a_k))|. \tag{5}$$

In the thesis, we have derived an algorithm efficiently computing the $\Delta(a, b)$ using such σ. The pseudocode is outlined in Algorithm 1. The total complexity is $\mathcal{O}(n(\log(n) + c(\sigma)))$, where n is the number of attributes and $c(\sigma)$ is a cost of calling the evaluation function for a single attribute.

Algorithm 1: Attribute alignment

```
   // Pseudocode for an attribute alignment algorithm with
      constrained attribute distance function running in
      $\mathcal{O}(n\log(n))$.
   input  : a ← List of attributes
   input  : b ← List of attributes
   input  : $\sigma$ ← Attribute evaluation function: $\mathbb{A} \to \mathbb{R}$
   output: Distance between a and b
1  Add dummy attributes into the list with less attributes;
2  Sort both list of attributes by $\sigma$;
3  totalDistance ← 0;
4  for $i \leftarrow 0$ to $i < len(a)$ do
5  |   totalDistance ← $|\sigma(a[i]) - \sigma(b[i])|$;
6  end
7  return totalDistance;
```

In our second approach, we allow arbitrary function δ measuring the distance between two attributes. Given two datasets a, b and attribute distance function δ, a distance matrix M can be easily built up:

$$M_{i,j} = \delta(a_i, b_j). \tag{6}$$

We can see the distance matrix M as the cost function and the aligning of attributes as an assignment, which leads to an assignment problem. The assignment problem is well known polynomial problem. The $\mathcal{O}(n^3)$ implementation of the Hungarian algorithm was published in [5].

We can use this algorithm to find the best assignment of the attributes. From the assignment the total distance between attributes can be computed as the sum of individual distances defined by the alignment as already seen in Algorithm 1. If δ was defined in the same manner as in the first approach, we would get the same result (Algorithm 1 is a special case of Algorithm 2). The difference is that this version allows for an arbitrary function measuring distance between attributes as an input.

The total complexity of the algorithm is $\mathcal{O}(n^3 + n^2\sigma(n))$, where n is the number of attributes and $\sigma(n)$ is the complexity of the attribute distance function. The whole algorithm that uses the Hungarian Algorithm is outlined in Algorithm 2.

Algorithm 2: Attribute assignment

```
  // Pseudocode for an attribute alignment algorithm using
     Hungarian Algorithm for the alignment.
  input  : $a \leftarrow$ First list of attributes
  input  : $b \leftarrow$ Second list of attributes
  input  : $\delta \leftarrow$ $IAttributeDistance$ Function
  output: Distance between $a$ and $b$
1 Add dummy attributes into the list with less attributes;
2 $M[i,j] \leftarrow \delta(a[i], b[j])$;
3 assignments $\leftarrow$ HungarianAlgorithm($M$);
4 totalDistance $\leftarrow$ 0;
5 for $i \leftarrow 0$ to $i < \text{len}(a)$ do
6     totalDistance $\leftarrow M[i]$[assignments[$i$]];
7 end
8 return totalDistance;
```

It is arguable whether we should come with the distance function between attributes that covers all cases including the distance between categorical and numerical attribute. We suppose that the attribute metafeatures of categorical and numerical attributes will vary, thus making it difficult to specify the distance. To solve this problem, the distance could be split into two parts: the distance between numerical attributes and the distance between categorical attributes. The final distance would be then the total of the sub-distances. To generalize this idea, we could go a step further and define selectors over the list of attributes. The selector would be a function accepting a list of attributes and returning a subset of the list. An attribute distance function could then be provided for each selector. Optionally, weights could be defined for each selector describing the importance of such selector (e.g. categorical attributes could be weighted more than numerical). This is outlined in Algorithm 3.

We have dedicated lots of effort to utilize extra information from the at-

Algorithm 3: Combined Attribute Assignment

```
// Pseudocode for distance measure combining multiple
   attribute assignments for each selectors.
input  : $a \leftarrow$ First list of attributes
input  : $b \leftarrow$ Second list of attributes
input  : selectors $\leftarrow$ List of selectors
input  : $w \leftarrow$ List of weights
input  : distanceMeasures $\leftarrow$ List of $IAttributeDistance$ functions
output: Distance between $a$ and $b$

1 totalDistance $\leftarrow$ 0;
2 for $i \leftarrow 0$ to $i <$ len(selectors) do
3     $a' \leftarrow$ selectors$[i](a)$;
4     $b' \leftarrow$ selectors$[i](b)$;
5     $\delta \leftarrow$ distanceMeasures$[i]$;
6     Add dummy attributes into $a'$ or $b'$ - whichever has less attributes;
7     totalDistance $\leftarrow w[i]*$attributeAssignment$(a', b', \delta)$;
8 end
9 return totalDistance;
```

tributes. We have the whole workflow that builds the distance on the datasets from the attribute distance through aligning and selectors. However, we would like to emphasize the fact that global attributes store useful information as well – this was proven in the literature. Even though the attribute alignment was competitive to global dataset distances, it does not necessarily mean that we have to use them separately from each other. In fact, it could be useful to combine their powers to get even better distance measure between datasets. In order to achieve this we have proposed Algorithm 4.

4 Theoretical Properties

Theorem 1. *Let a, b be lists of attributes, f optimal alignment between a and b. Attribute distance measure δ is a metric on the extended space of attributes. Let $a' = a \cup \{dummy_1\}$ and $b' = b \cup \{dummy_2\}$, then f' defined as*

$$f'(x) = \begin{cases} f(x); & \text{if } x \in a, \\ dummy_2; & \text{otherwise.} \end{cases}$$

is an optimal alignment between a' and b' with the same cost as f.

By applying this theorem multiple times, we can add an arbitrary number of *dummy* attributes without changing the optimality of alignment.

Theorem 2. *Let a, b are list of attributes, δ is a metric on the extended space of attributes. Then Algorithm 2 preserves all metric axioms and resulting distance Δ on the dataset space is a metric.*

Algorithm 4: Dataset Distance Aggregation: *IDatasetDistance*

```
// Pseudocode for combining multiple dataset distance
   measures and producing their weighted combination.
input : distances ← List of dataset distance measures $\Delta$
input : weights ← List of weights
input : $x$ ← First dataset
input : $y$ ← Second dataset
output: Distance between two datasets
1 distance ← 0;
2 for $i$ in $\{1, \ldots, \text{len(distances)}\}$ do
3     distance ← distance + weights[$i$]distances[$i$]($x, y$);
4 end
5 return distance;
```

The previous theorem also holds for Algorithm 3:

Corollary 1. *Let a, b are lists of attributes, $\{\delta_1, \ldots, \delta_n\}$ are metrics on the extended space of attributes, $\{s_1, \ldots, s_n\}$ are selectors, $\{w_1, \ldots, w_n\}$ are positive weights of each selector. Then Algorithm 3 preserves all metric axioms and resulting distance on the dataset space is a metric.*

It may be interesting to see how to extend attribute space by an artificial $dummy_n$ attribute. A following theorem gives us some clue:

Theorem 3. *Let δ' be a metric on a non-empty attribute space $\mathbb{A}$. Let δ be a distance derived from δ' by extending $\mathbb{A}$ by artificial $dummy_n$ attribute. We set the $\delta(dummy_n, x)$ and $\delta(x, dummy_n)$ to some constant k $\forall x \in \mathbb{A}$. Finally, we set $\delta(dummy_n, dummy_n) = 0$. Let $\delta_{max} = \max_{x,y \in \mathbb{A}} \delta(x, y)$. If δ is a metric on the extended attribute space then $0 < k$ and $\frac{\delta_{max}}{2} \leq k$.*

If we want to have metric attribute distance, and at the same time, a constant distance from artificial $dummy_n$ attributes, we have to choose this constant from quite large values. Doing this can create unwanted side affects. The penalty for adding $dummy_n$ attributes can become the most significant part of the distance, especially if the number of attributes varies by large amounts. It can be argued that if some attributes are not very relevant for the prediction of the target (for example attributes with low entropy), the two datasets should not differ too much if one dataset is missing such not very relevant attributes. Furthermore, in some cases it may be impossible to compute δ_{max} or to learn the value in advance.

We get no such problems if we use the element of the attribute space itself, if we already made sure that the attribute distance is a metric. Using the $dummy_a$ attribute from the space will not break the metric axioms (if the attribute distance δ is a metric).

Theorem 2 is also valid in the opposite direction, as shown in the following theorem.

Theorem 4. *Let a, b are list of attributes, δ is a distance between attributes. If Δ is a distance over dataset space produced by Algorithm 2 and Δ is a metric on the dataset space, then δ is a metric on the extended attribute space $\mathbb{A}$.*

Similar theorem holds for Algorithm 3 with addition that all selectors must be distinct:

Corollary 2. *Let a, b are list of attributes, $\{\delta_1, \dots, \delta_n\}$ are the distances on the extended space of attributes, $[s_1, \dots, s_n]$ are selectors, $[w_1, \dots, w_n]$ are weights of each selector. If $\forall k, j \in \{1, \dots, n\}, \forall \mathbb{A}' \subseteq \mathbb{A} : s_k(\mathbb{A}') \bigcap s_j(\mathbb{A}') = \emptyset$ and distance Δ produced by Algorithm 3 is a metric on the dataset space then $\{\delta_1, \dots, \delta_n\}$ are metrics on the extended attribute space defined by the corresponding selector.*

It is intuitive to have each selector distinct, if we think of a selector in the sense we have introduced them: selector selects specific attributes (e.g. categorical or numerical) so more fine grained distance functions can be applied. In this sense the selectors will be distinct as the subset of categorical attributes is clearly disjunct to subset of numerical attributes.

We can wonder whether Corollary 2 holds even without the constraint for distinct selectors.

Observation 1. *The requirement for distinct selectors is an essential part of Corollary 2.*

Proof. We will show a counterexample: let attribute space be the set of two elements: $\mathbb{A} = a_1, a_2$. Let s_1, s_2 be two selectors defined as $s_1(\mathbb{X}) = s_2(\mathbb{X}) = \mathbb{X}$. Let δ_1 be a metric and δ_2 be a distance function defined as $\delta_2(x, y) = -0.1\delta_1(x, y)$. δ_2 is not a metric on the attribute space as it violates non-negativity – $\delta_2(a_1, a_2) = -0.1\delta_1(a_1, a_2) < 0$. It still satisfies symmetry and coincidence though. We can combine δ_1 and δ_2 to form δ_3: $\delta_3(x, y) = \delta_1(x, y) + \delta_2(x, y) = 0.9\delta(x, y)$. Δ induced by δ_1, δ_2, and $s_1 = s_2$ is the same as Δ' induced by δ_3. δ_3 is a metric on the attribute space according to Theorem **??** as it is positively rescaled δ_1. Therefore, it does not matter if we use the combination of δ_1, δ_2 or just δ_3 alone. Resulting Δ is a metric on the dataset space according to Theorem 2 even-though δ_2 is not a metric on the dataset space. □

Theorems 2, 4 and Corollaries 1 and 2 are useful as they state that if we can get a metric on the attribute or dataset space, we get other metric on the other space for free when using attribute aligning. During training, we could define a function measuring metric similarity on the attribute or dataset samples. As the samples would usually be just small finite subsets of attribute or datasets space, we could be wondering whether by optimizing metric on some dataset or attribute subset, we would get also metric on the other subset as Theorems 2, 4 and Corollaries 1, 2 are valid only for the whole spaces. We will try to formalize this motion:

Definition 3. *Let $\mathbb{A}$ be attribute space, let A be the subset of $\mathbb{A}$. Let $\mathbb{D}$ be a dataset space. Then we will denote dataset subspace D as* supported *by attribute subspace A if $D \subseteq \mathbb{D}$ and if $\forall d \in D, \forall att \in d : att \in A$.*

In other words, datasets in D are only composed of elements in A. This definition allow us to investigate metric properties in just the subset of attribute spaces and conclude properties in supported subspaces of datasets (like training and testing subset) and vice-versa.

Definition 4. *Let $\mathbb{A}$ be attribute space, let $\mathbb{D}$ be a dataset space and D its subset. We will call the subset A of $\mathbb{A}$ as the* source *of D if $\forall att \in \mathbb{A} : att \in A \iff \exists d \in D : att \in d$.*

In other words, elements of A are just enough in order to create all elements in D. Similarly to the remark to Definition 3, we can use this definition to reason about whether properties that are valid for some distance function on some set of datasets are also valid for the attribute source of these datasets.

If we optimize metric on subset of attributes, we also get metric on all datasets that can be combined by this subset of attributes when using Algorithm 2:

Theorem 5. *Let $\mathbb{A}$ be extended space of attributes and A its extended subset, δ is a metric on A, $\mathbb{D}$ space of datasets. Then $\forall D \subseteq \mathbb{D}$, D supported by A: Algorithm 2 preserves all metric axioms and resulting distance Δ on the D is a metric.*

Intuitively, the same works for Algorithm 3.

This enables us to define our algorithms in such a way, that if we can create a metric on all the attributes in the training and testing samples, we can also guarantee the resulting metric on all supported subsets of the dataset space. The training and testing datasets must be among them, as they are supported by the testing and training attributes.

Note that Theorem 5 is valid because there is nothing in the proof of Theorem 2 that would require elements (datasets) outside of the subset D and A. As for the other direction, when observing the proof of Theorem 4, we use the trick that we artificially create datasets with a single element. However, such datasets can be outside of D. We can then wonder whether Theorem 5 is valid also in the other direction considering Algorithm 2. It is not according to Observation 2.

Observation 2. *Let $\mathbb{A}$ be space of attributes and $\mathbb{D}$ space of datasets. Let D be a subset of $\mathbb{D}$ and Δ a metric on D. Let A be the source of D and δ be a distance measure on A, such that Algorithm 2 induces Δ using δ. Then δ is not necessarily a metric on A.*

Same counterexample can be found when using Algorithm 3 as this algorithm is a generalization of Algorithm 2, therefore the same argument can be applied. This implies that we cannot guarantee metric on the subspace of attributes appearing in the training and testing datasets even if we can guarantee resulting distance to be a metric on the training and testing dataset subspace.

Another argument favouring a metric for the attributes is that specialized algorithms can be used for the assignment such as [2].

5 Experiments and Results

The OpenML [17] machine learning repository has been selected as the source of data for the experiments. It is a repository of datasets, tasks, machine learning algorithms, and experiment results called runs. When a new dataset is uploaded to a repository, OpenML automatically extracts metafeatures. There are in total 106 different metafeatures that can be extracted, however not every metafeature is extracted for each dataset. An OpenML task defines experiment over some dataset – a goal (e.g. supervised classification over the target attribute), estimation procedure (e.g. 10-fold cross-validation) and evaluation measures used (RMSE, PredictiveAccuracy). An OpenML run is the result of some machine learning algorithm on some task.

As the experiment results are standardized, it is easy for researchers to compare the results of machine learning and metalearning methods. Furthermore, as some metafeatures are automatically computed by OpenML, it makes implementation of other metalearning approaches faster, thus speeding up the research. OpenML also exposes data via its REST API. Specialized connector packages for communication with this interface are available for R, Java, .Net (which was created by us) and Python. However, it is possible to write a custom connector, as almost all languages support REST API. Another benefit is that the whole project is under active development and there is a growing community around it.

Not even OpenML is without drawbacks. All datasets are visible including the testing data. With many experiments, there is no guarantee that users will not eventually carry information out of testing datasets into the training by looking at the results of previous experiments on the testing datasets (the phenomenon referred to as *peeking* by authors of [11]).

The choice of the data is very important for the quality of experiments. Results can be affected by many factors – amount of errors in the data, whether the data are general or domain specific only, etc. It should be taken into consideration that OpenML repository is very general and public. Therefore, it may contain errors or noise, and our algorithms have to handle every type of dataset.

Originally, our OpenML dump contained 791 public datasets. We have placed extra requirements on the datasets in order to fulfil two requirements.

Keep the computation of alignment reasonable for all pairs of datasets in the chosen subset. This will speed up the evaluation of quality of alignment predictor. The computation cost depends on the number of attributes, the goal is to find a good compromise between the amount of datasets that we can use and the cost of alignment of their attributes. From the plot of the distribution of number of attributes among datasets it was clear that only a very small number of datasets have large number of attributes. The alignment of this minority of datasets would take the majority of time. We have decided to filter the datasets with more than 50 attributes.

In order to be able to easily compare metalearning approaches, it is desirable that every dataset has the same types of metafeatures extracted. This may not be possible in every case, as some global metafeatures are only computable for classification/regression tasks. Because of this, some compromises may be needed. Similarly, some attribute metafeatures may be computable only for classification/numerical attributes but this does not concern us too much, as we will be using selectors (see Algorithm 3) to handle different types of attributes). To fulfil the second requirement it was sufficient to take the classification datasets only.

In our experiments the experiment results are also needed, which – in the case of OpenML – are covered by OpenML tasks and runs. First requirement was clear – we can include only those datasets with the results available. We have chosen Predictive Accuracy as the performance criterion because of its frequent availability.

There could be multiple results for a pair of algorithm, dataset. The question was which result to choose when arguing about the actual ranking of the algorithms on datasets. We argued that we want to asses the maximal potential of some algorithm. The best result also usually corresponds with a solution found by a hyper-parameter optimizer. Therefore, we have used the result with the best Predictive Accuracy.

Finally, the results of ensemble algorithms like bagging, boosting, stacking and rotation forests [10] were omitted. These ensembles are encompassing different number of other machine algorithms and can take advantage of such combined power. Such composite behaviour resulted in heavily outperforming every other algorithm in the database, thus changing the results significantly. Furthermore, their performance relies heavily on the parameters that specify what algorithms should be used in the ensemble and as such, every parameter setting should be better treated as a separate algorithm. More careful examination is thus necessary before including ensemble algorithms into our experiments.

Application of the filters resulted in 351 datasets with 20,719 rows of the best results with Predictive Accuracy for some pair of algorithm and dataset. That included 115 unique algorithms.

These amounts may seem low for machine learning experiments, but they are still much bigger that in our previous experiments, and are very high also compared to the rest of the metalearning experiments found in the literature.

We have normalized most metafeatures into the interval $\langle 0, 1\rangle$. Attributes already naturally constrained to that or similar interval (Spearman's Correlation Coefficient) were the exception. The normalization was performed regardless of whether the metafeature was global or attribute specific. We have used the $min - max$ normalization given by the following equation:

$$x'_i = \frac{x_i - \min_{x\in\mathbb{X}}}{\max_{x\in\mathbb{X}} - \min_{x\in\mathbb{X}}},$$

where $\min_{x\in\mathbb{X}}$ and $\min_{x\in\mathbb{X}}$ are minimal and maximal values of the given metafeature $\mathbb{X}$, x_i is specific value of the i-th metafeature before rescaling, and x'_i the value of that metafeature after rescaling.

There is no such public repository as OpenML that would have attribute metadata available for each dataset. Therefore, we have extracted additional attribute specific metadata – 23 attributes for all types, 24 for numerical attributes, and additional 3 for categorical attributes.

For the experiments, we have inserted dataset distance algorithms into the whole metalearning workflow and measured the quality of the metalearning algorithm. As our algorithm requires metric either between datasets or attributes, we have used different weighted p-norms and let the genetic algorithms to optimize the weights. In the case of selectors or combined global and attribute specific approach, we also let the weights of each sub-distance to be optimized. As the algorithms are non-deterministic, every experiment settings were conducted 10 times.

The exact values of different variables influencing the complexities are shown in Table 1.

Table 1: Values of variables influencing the complexity given by the training data.

Variable	Description	Value
n_d	Number of attributes	170
n_a	Number of algorithms	115
n_m	Number of global metafeatures	31
n_{att}	Bound of number of attributes	50
n_{att_met}	Number of attribute metafeatures	47

The total complexity for the whole workflow for different setups of the ranking algorithms are in Table 2.

From the raw results it can be seen that some algorithms are better than others. From the point of view of the median of the results, the order of algorithms is as follows (from best to worst): (Aggregation with $p = 1$), (Aggregation with $p = 2$), (Global with p=∞), (Global with $p = 1$), (Global with $p = 2$), (Aggregation with $p = \infty$), (Assignment with $p = 2$), (Assignment with $p = \infty$), (Assignment with $p = 1$), baseline.

To have a proper comparison, results of each algorithm on the testing set were compared based on the result of two tailed *Mann-Whitney U Test* [4]. This test is a non-parametric test with the null hypothesis that both samples come from the same population. The α statistics used was 0.05. The results of the tests are in Table 3. All algorithms except the Assignment algorithm with the $p = 1$ were significantly better than the baseline. The best results had the aggregation of assignment and global metadata. There was no algorithm that would be significantly better than any of the Aggregated algorithms. On the contrary, Aggregated algorithms with $p = 1$ and $p = 2$ were significantly better than every other algorithm. There was no clear winner between those two. The results proof our hypothesis that algorithms based on attribute assignment produce useful results for ranking prediction. Also, our theory that the best results are probably obtainable by the aggregation of assignment and global distance is also supported by the results. Interesting observation is that the assignment with $p = 1$ produces the worst results but the aggregation with the

Table 2: Total complexity of the whole workflow for ranking quality evaluation for different ranking algorithms.

Algorithm	Total Complexity (in $\mathcal{O}$)	Optimizing
Baseline	$n_d n_a + n_d n_a + n_a \log(n_a)$	No
Attribute Alignment	$n_d n_a + n_d(n_d \log(n_d) + n_a \log(n_a))$ $+ n_d^2 n_{att} \log(n_{att})$	No
Global Distance	$n_d n_a + n_d(n_d \log(n_d) + n_a \log(n_a))$ $+ n_d^2 n_m$	Yes
Attribute Assignment	$n_d n_a + n_d(n_d \log(n_d) + n_a \log(n_a))$ $+ n_d^2(n_{att\_met} n_{att}^2 + n_{att}^3)$	Yes
Aggregation	$n_d n_a + n_d(n_d \log(n_d) + n_a \log(n_a))$ $+ n_d^2(n_{att\_met} n_{att}^2 + n_{att}^3 + n_m)$	Yes

same p provides the best results. Perhaps alignment with $p = 1$ provide best additional information to support decision by the global metadata.

The best results on the testing set were produced by the aggregation of global and attribute metafeatures where the p was set to 1.

To further improve the results, we have allowed switched the genetic algorithms for the genetic programming. TODO

Given an arbitrary distance measure d on some space, we can repair the measure in such a way that it is a semimetric. For example function d' defined as follows:

$$f(x,y) = \frac{|f'(x,y)| + |f'(y,x)|}{2} \tag{7}$$

is always non-negative and symmetric. To enforce coincidence axiom, we can easily detect equal arguments on the input and return 0 otherwise. Similarly, if the d' would return 0 for a non-matching input we could return some small non-negative number ϕ instead. Or we could return $\epsilon + d(x,y)$, for small $\epsilon > 0$. Both approaches results in semimetric, however there is a distinction. The former approach could break triangle inequality in the case d was a metric. If we found three objects k, l, z such that $d(k,z) < \phi/2$, $d(z,l) < \phi/2$, we could break the triangle inequality. That would happen if the ϕ was returned for the tuple k, l instead of 0. For the triangle inequality to hold, it must be the case that the distance $\phi = d(k,l) \leq d(k,z) + d(z,l)$. But at the same time, we have $d(k,z) + d(z,l) < \frac{\phi}{2} + \frac{\phi}{2} = \phi$ from the assumptions, which is a contradiction. We cannot do the same argument for the latter approach, as ϵ was amended to all distances of non equal objects.

However, the GP experiments were only slightly better than the baseline algorithm as they tended to overfit easily. There are many ways how to improve the generalization abilities. The so called *bootstrapping* modifies the initialization phase. Some individuals with already interesting fitness are inserted into

Table 3: Statistical comparison of different algorithms and their ranking quality results on the testing set. Row i defines what algorithms had significantly worse results than algorithm i. N stands for no and Y for yes.

	Aggregation, $p = 1$	Aggregation, $p = 2$	Global, $p =$ inf	Global, $p = 1$	Global, $p = 2$	Aggregation, $p = \infty$	Assignment, $p = 2$	Assignment, $p = \infty$	Assignment, $p = 1$	Baseline
Aggregation, $p = 1$		N	Y	Y	Y	Y	Y	Y	Y	Y
Aggregation, $p = 2$			Y	Y	Y	Y	Y	Y	Y	Y
Global, $p = \infty$				N	Y	N	Y	Y	Y	Y
Global, $p = 1$					Y	N	Y	Y	Y	Y
Global, $p = 2$						N	Y	Y	Y	Y
Aggregation, $p = \infty$							Y	Y	Y	Y
Assignment, $p = 2$								N	N	Y
Assignment, $p = \infty$									N	Y
Assignment, $p = 1$										N

populations and their useful blocks may be distributed over the population. *Regularization* [1] is a set of techniques aiming for boosting the generalization abilities of machine learning model. This is done by penalizing complex hypothesis or by encouraging the properties that we think help in generalization. We have used two regularization techniques that have been already used in our experiments in [15, 16]. The first approach uses technique called *coevolution* during the evolution of the trees. The second approach – called *multi-objectivization* – splits the single objective into multiple objectives in which we can measure some other interesting properties. We have chosen NSGA-II as a *multi-objective optimization* algorithm as it is one of the best for two-objective optimization.

The algorithm with bootstrapping and coevolution managed to beat the baseline and all the pure assignment based algorithms. It also managed to match the level of the first combination of the global metafeatures and assignments. In some runs the overfitting was still present – although the algorithm managed to improve the fitness on the training set, in some cases it did not improve the results on the validation set. That significantly decreased its score in the overall results. However, one run managed to outperform every other run using the global attributes only and produced one of the best results using solely the assignments. This suggest that attribute assignment has a very good potential to be improved with further empowering the generalization abilities of GP algorithm. The runs with the antibloat operator reduced bloating, however we did not observe a big difference in the resulting ranking quality. This suggests that the bloating is not the only factor reducing the generalization abilities.

During the multi-objectivization experiments, slightly different OpenML dump was used. Therefore, we could compare the results only with the baseline algorithm. We have observed the high correlation of the metric similarity and prediction accuracy. This supports the hypothesis that the metric properties are important for the generalization abilities of the induced dataset similarity measure. This was the most obvious during the run where the Spearman's rank correlation coefficient between the first and second criterion of the second run on the testing set was 0.734. This means that in the testing set the individuals more similar to a metric had better results for the prediction of the algorithm ranking.

6 Conclusions and Future Work

In this thesis, we have studied the non-propositional approach for comparing two datasets. We have focused on algorithm ranking problem in metalearning. However, our approach is not limited solely to this area and can be easily reused even in other fields. Specifically, the distance measures defined on some set $\mathbb{X}$ can be transformed by attribute assignment algorithms to a distance measure on the power set $2^{\mathbb{X}}$ of $\mathbb{X}$.

The main contribution lies in the design of multiple algorithms for measuring the distance between two datasets that can handle non-propositional metadata and their unique theoretical properties. The difference between them is in the computational complexity, expression power and guarantees on the resulting dataset distance measure.

When proposing our algorithms, we built our work on the related approaches. The literature suggested several methods to handle the non-propositionality, either directly in the metalearning fields or in some other areas of computational intelligence. However, we have identified several areas that could improve the reviewed literature. For instance, the current approaches are either losing important information, lack metric properties or assume that the order is important (although we can reshuffle attributes in datasets without changing the information). This drove our motivation to propose new methods in the first place. Furthermore, authors of [6] recognized the problem of building the distance over non-propositional datasets as non-trivial.

The main idea behind our algorithms rests upon the idea of attribute assignment. To measure distance between two datasets, we first propose an attribute distance measure. We first amend the datasets so their cardinalities match, by inserting artificial attributes into the dataset with less attributes. Then we find bijection of attributes from the first dataset to another so the sum of the distances defined by the bijection and the attribute is minimized. This sum of distances is also the desired distance between datasets.

We have proven that under certain conditions, the resulting distance between datasets is a metric. The main condition for this is that the underlying attribute distance measure itself is a metric. Other conditions include several restrictions on extending the attribute space by a *dummy* attribute. We have also proven that the above is valid also in the other directions. However, the first direction is somewhat stronger if we have only training data and is optimizing towards a metric on either dataset or attribute subspace defined by training data. Some other ideas were considered, for instance the normalization of the sum of attribute distances by the number of attributes. However, we have proven that this can break the triangle inequality axiom.

We have designed our algorithm to be extensible. It is possible to use the metadata specific for some attribute types, for instance for categorical and numerical metadata. This is achieved by splitting the distance into two, one for each metadata type. We have verified that this does not affect the metric properties of the algorithms. It is also possible to combine multiple metric distance into one. This allows our assignment approach to be combined with propositional approaches. As both approaches return useful information, we argued that by combining them, even more accurate sense of distances between datasets should be obtained.

We have designed generic workflow for measuring quality of ranking. As we made sure that all our algorithms conform to specified interfaces, it is possible to replace different parts of the workflow. This allowed us to compare different algorithms and their combinations proposed in the thesis in a unified way. We have also proven that it is possible to optimize distance based measures without changing their properties. This enabled us to employ optimization methods to boost the performance of our algorithms. We have employed genetic algorithms and genetic programming for this purpose.

Results of genetic algorithm experiments suggest that attribute alignment algorithms can be successfully used for algorithm ranking. Every parameter settings we optimized produced statistically better results than the baseline algorithm with a single exception. The aggregation of global and attribute approach produced the best results. With genetic programming, we traded triangle inequality for higher expression power. It was so powerful that it could overfit the training data very easily. To counter this, we have employed several approaches to improve generalization abilities. With this we managed to further improve the results of the assignment algorithms on our data. Especially coevolution combined with bootstrapping of the population managed to obtain promising results. We also reviewed one of our previous work where we used multi-objectivization to boost the generalization abilities of models being evolved. We introduced metric resemblance as a second criterion. Results suggested that there is a high correlation between the second criterion and generalization abilities of the models.

We have also demonstrated visualization of non-propositional dataset representation using the kernelized PCA. We investigated the visualization in more depth, and the results seems to be plausible as the visualisation rendered similar datasets in the same cluster.

The work presented in this thesis can be extended in several directions. We wanted to focus on the attribute based distance. As the whole workflow of all the pieces plugged together became quite complicated, we did not want to add extra parts that would distract from the main idea by increasing the complexity of the workflow. It would be possible however to use ensemble based learning on top of our models to get the combined accuracy of our models.

As we are using distance based algorithms, it could be beneficial to try better methods, such as weighted k-NN, that would make use of the distances of the nearest neighbours. It would also help to have more data. As we are dealing with high dimensional dataset space, hundreds of datasets is still very small amount to reasonably cover the space.

Our algorithms need lots of parameters. As our resources are limited and the training of models took significant amount of time, we could not dedicate much space to tuning of the parameters, and parameters we used were based on the previous experiments or short preliminary experiments. We would like to tune the parameter of k of the k-NN, different parameters and genetic operators of genetic algorithms used, with different strategies for adding the dummy attribute (either constant or different attribute when selected from attribute space). Even in this thesis, we have observed that parameters can significantly improve the overall results of experiments.

We would also like to enhance Genetic Programming algorithm with types. Typed genetic programming [8, 7] can help reduce the space that is searched and allows for more elaborate constructs and operators.

A room for improvement can be also seen in the metafeatures we are extracting from the attributes. Currently, only simple, statistical and theoretical metafeatures are extracted. It could be interesting whether we can extract some sort of landmarks on the attribute level. For example, we can try to predict target values using only a single attribute and use the results as a new metafeatures. It could also help to find metafeatures that are really useful for the generalization. It may well be the case that some metafeatures are just used to overfit the data and instead of contributing to the generalization abilities of the models, they are just downgrading the validation results.

We also see an opportunity in expanding the theoretical work. After the alignment, we use the sum of individual attribute distances given by the assignments to get the metric on datasets provided that certain conditions hold. It could be interesting to see whether some other aggregations can be used without sacrificing the nice property of metric preservations. It could be also beneficial to define sort of normalization of the resulting dataset distance that does not violate metric axioms.

References

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*. AMLBook, 2012. ISBN: 1600490069, 9781600490064.

[2] Pankaj K. Agarwal and R. Sharathkumar. “Approximation algorithms for bipartite matching with metric and geometric costs”. In: *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. Ed. by David B. Shmoys. ACM, 2014, pp. 555–564. ISBN: 978-1-4503-2710-7. DOI: `10.1145/2591796.2591844`. URL: `http://doi.acm.org/10.1145/2591796.2591844`.

[3] Pavel Brazdil et al. *Metalearning — Applications to Data Mining*. Cognitive Technologies. Springer, 2009, pp. I–X, 1–176. ISBN: 978-3-540-73262-4. URL: `http://dblp.uni-trier.de/db/series/cogtech/index.html%5C#0022052`.

[4] Gregory W. Corder and Dale I. Foreman. *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, 2014. ISBN: 1118840313. URL: `http://www.amazon.com/Nonparametric-Statistics-Step-Step-Approach/dp/1118840313%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1118840313`.

[5] Jack Edmonds and Richard M. Karp. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". In: *Journal of the ACM* 19.2 (Apr. 1972), pp. 248–264. ISSN: 0004-5411. DOI: `10.1145/321694.321699`. URL: `http://doi.acm.org/10.1145/321694.321699`.

[6] Alexandros Kalousis and Melanie Hilario. "Representational Issues in Meta-Learning". In: *ICML*. AAAI Press, 2003, pp. 313–320.

[7] T. Křen and R. Neruda. "Generating lambda term individuals in typed genetic programming using forgetful A*". In: *Evolutionary Computation (CEC), 2014 IEEE Congress on*. July 2014, pp. 1847–1854. DOI: `10.1109/CEC.2014.6900547`.

[8] David J. Montana. "Strongly Typed Genetic Programming". In: *Evolutionary Computation* 3 (1994), pp. 199–230.

[9] Jan Ramon and Maurice Bruynooghe. "A polynomial time computable metric between point sets". In: *Acta Informatica* 37.10 (2001), pp. 765–780. ISSN: 1432-0525. DOI: `10.1007/PL00013304`. URL: `http://dx.doi.org/10.1007/PL00013304`.

[10] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. "Rotation Forest: A New Classifier Ensemble Method". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.10 (Oct. 2006), pp. 1619–1630. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2006.211`.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. 3rd ed. Prentice Hall, Dec. 2009. ISBN: 0136042597. URL: `http://www.worldcat.org/isbn/0136042597`.

[12] J. Šmíd and R. Neruda. "Comparing datasets by attribute alignment". In: *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*. Dec. 2014, pp. 56–62. DOI: `10.1109/CIDM.2014.7008148`.

[13] Jakub Šmíd. "Agent optimization by means of genetic programming". MA thesis. Prague, Czech Republic: Charles University in Prague, 2012.

[14] Jakub Šmíd and Roman Neruda. "Using Genetic Programming to Estimate Performance of Computational Intelligence Models". In: *Adaptive and Natural Computing Algorithms (Proceedings of ICANNGA 2013)*. Ed. by Marco Tomassini et al. Vol. 7824. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 169–178. ISBN: 978-3-642-37212-4. DOI: `10.1007/978-3-642-37213-1_18`. URL: `http://dx.doi.org/10.1007/978-3-642-37213-1_18`.

[15] Jakub Šmíd et al. "Co-evolutionary genetic programming for dataset similarity induction". In: *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE. 2015, pp. 1160–1166.

[16] J. Šmíd et al. "Multi-Objective Genetic Programming for Dataset Similarity Induction". In: *Computational Intelligence, 2015 IEEE Symposium Series on*. Dec. 2015, pp. 1576–1582. DOI: `10.1109/SSCI.2015.222`.

[17] Joaquin Vanschoren et al. "OpenML: Networked Science in Machine Learning". In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60. DOI: `10.1145/2641190.2641198`. URL: `http://doi.acm.org/10.1145/2641190.2641198`.

[18] David H. Wolpert. "The supervised learning no-free-lunch Theorems". In: *In Proc. 6th Online World Conference on Soft Computing in Industrial Applications*. 2001, pp. 25–42.

[19] Adam Woznica, Alexandros Kalousis, and Melanie Hilario. "Distances and (Indefinite) Kernels for Sets of Objects". In: *ICDM*. IEEE Computer Society, 2006, pp. 1151–1156.

List of Publications

[20] O. Kazík, J. Šmíd, and R. Neruda. “Evolutionary optimization of meta data metric for method recommendation”. In: *Cybernetics and Intelligent Systems (CIS), IEEE Conference on.* Nov. 2013, pp. 123–127. DOI: 10.1109/ICCIS.2013.6751590.

[21] Ondřej Kazík et al. “Clustering Based Classification in Data Mining Method Recommendation”. In: *International Conference on Machine Learning and Applications (ICMLA 2013).* IEEE Computer Society, 2013, pp. 356–361. DOI: 10.1109/ICMLA.2011.161.

[22] Klára Pešková et al. “Hybrid Multi-Agent System for Metalearning in Data Mining”. In: *Proceedings of the International Workshop on Meta-learning and Algorithm Selection co-located with 21st European Conference on Artificial Intelligence, MetaSel@ECAI 2014, Prague, Czech Republic, August 19, 2014.* 2014, pp. 53–54. URL: http://ceur-ws.org/Vol-1201/paper-13.pdf.

[12] J. Šmíd and R. Neruda. “Comparing datasets by attribute alignment”. In: *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on.* Dec. 2014, pp. 56–62. DOI: 10.1109/CIDM.2014.7008148.

[14] Jakub Šmíd and Roman Neruda. “Using Genetic Programming to Estimate Performance of Computational Intelligence Models”. In: *Adaptive and Natural Computing Algorithms (Proceedings of ICANNGA 2013).* Ed. by Marco Tomassini et al. Vol. 7824. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 169–178. ISBN: 978-3-642-37212-4. DOI: 10.1007/978-3-642-37213-1_18. URL: http://dx.doi.org/10.1007/978-3-642-37213-1_18.

[15] Jakub Šmíd et al. “Co-evolutionary genetic programming for dataset similarity induction”. In: *Evolutionary Computation (CEC), 2015 IEEE Congress on.* IEEE. 2015, pp. 1160–1166.

[16] J. Šmíd et al. “Multi-Objective Genetic Programming for Dataset Similarity Induction”. In: *Computational Intelligence, 2015 IEEE Symposium Series on.* Dec. 2015, pp. 1576–1582. DOI: 10.1109/SSCI.2015.222.