Feature Selection in Taxonomies with Applications to Paleontology

Gemma C. Garriga, Antti Ukkonen, and Heikki Mannila

HIIT
Helsinki University of Technology and University of Helsinki, Finland

Abstract. Taxonomies for a set of features occur in many real-world domains. An example is provided by paleontology, where the task is to determine the age of a fossil site on the basis of the taxa that have been found in it. As the fossil record is very noisy and there are lots of gaps in it, the challenge is to consider taxa at a suitable level of aggregation: species, genus, family, etc. For example, some species can be very suitable as features for the age prediction task, while for other parts of the taxonomy it would be better to use genus level or even higher levels of the hierarchy. A default choice is to select a fixed level (typically species or genus); this misses the potential gain of choosing the proper level for sets of species separately. Motivated by this application we study the problem of selecting an antichain from a taxonomy that covers all leaves and helps to predict better a specified target variable. Our experiments on paleontological data show that choosing antichains leads to better predictions than fixing specific levels of the taxonomy beforehand.

1 Introduction

Prediction and classification are popular tasks in data analysis. The input examples to these tasks are typically described in a vector space with the dimensions of a set of features. The goal of the learning algorithm is to construct a model that will predict a target variable or a class associated to the given input examples. The set of features describing the examples is crucial for the performance and accuracy of the final learned model. Often, combinations of the input features can provide a much better way to explain the structure of the data. The problem of feature selection and feature construction is nowadays an important challenge in data mining and machine learning [2, 4, 10].

We study feature selection problems on taxonomies. Taxonomies occur in many real-world domains and add a richer representation to the plain set of features. Consider a paleontological application, with a set of fossil species observed across different sites. Commonly, species are categorized into several taxonomic levels exhibiting the structure of a tree. A simple snapshot of a part of the primate taxonomy is shown in Figure 1. We have, for instance, that *Pliopithecidae* and *Hominidae.* are primates; and the genus *Homo* belongs to the family *Hominidae.* A family or genus is considered to occur at a site if at least one of the

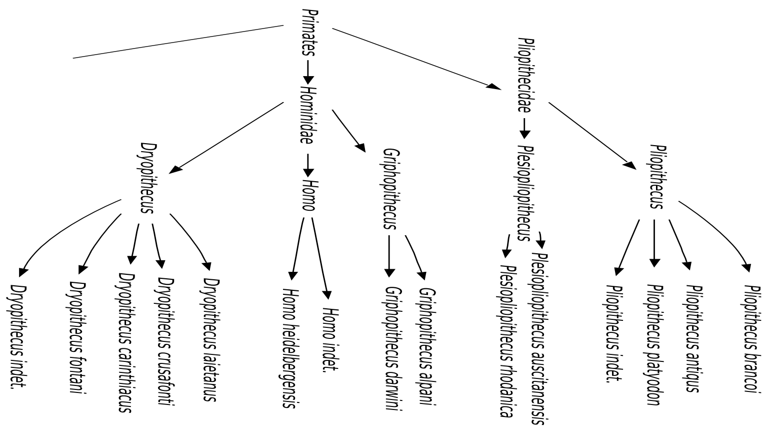


Fig. 1. Taxonomy of the primates species for a paleontological application.

species below it in the taxonomy occurs there. In general, an occurrence of a internal node occurs if and only if at least one of the leaves below it occurs.

A fundamental task in this paleontological application is to predict the age of the site from the observed taxa in the data [7–9, 12]. Only few sites have accurate age determinations from radioisotopic methods, and stratigraphic data (information about the layers in which the fossils are found) is also often missing. Thus the data about the taxa found at the site is all we have for determining the age of the site. The prediction task can be based on using different levels of the taxonomic hierarchy. Selecting aggregates of the features at the proper level of the taxonomy tree is critical. For example, combining the observed species at the genus level of the taxonomy can provide a much better predicting accuracy than using directly the leaf (species) level.

A common solution in practice is to choose a fixed level of the taxonomy to represent the new aggregated features; this implies combining all the species at the same height of the taxonomy tree. Although this solution is natural, it misses the potential gain of aggregating sets of species at different levels of the taxonomy separately. A toy illustration is given in Figure 2: in (a) we have small binary dataset with features from a to e and a given taxonomy on top of them; the variable we wish to predict is the real-valued named Age. In the paleontological example features are species and the age would represent how old each site is. Choosing nodes y and z from the taxonomy with the aggregator of logical "OR" we can better uncover the hidden structure in the data, shown in (b). We have that a, b, c have been aggregated at height 2 and d, e, at height 1 of the taxonomy level. For this example, having the different levels of aggregation is much better than selecting always the level of species (leaf nodes of the taxonomy) or any fixed level.

Motivated by the paleontological application [7–9, 12], the general computational problem we consider is the following: select the best subset of the nodes in the taxonomy which are not comparable (i.e., form an antichain) and still cover all leaves, in order to improve the prediction accuracy of a target variable. Note that this definition is also useful for other applications, for instance market basket data analysis, where we may wish to predict the age of a customer from the products in the shopping basket. Also for the market basket data application we have taxonomies available at the level of the items: e.g., wines and beers both belong to the category of alcoholic beverages.

We address the complexity of the problem, present several algorithms inspired by traditional feature selection approaches, yet exploiting the taxonomy tree structure, and show how to sample uniformly at random non-comparable nodes from the taxonomy tree. Our experiments on real paleontological data show that antichains are natural for this application: they often represent a better refined antichain than the one obtained by fixing the level of the taxonomy beforehand, and typically, the predictions are then more accurate.

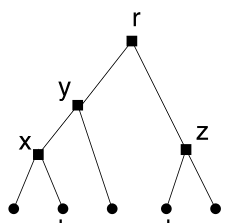
2 Problem definition

Let F be a set of features. For the paleontological application, F would correspond to the set of species observed in the data. A *taxonomy* $\mathcal{T}$ on the feature set F is a rooted tree whose leaf nodes are exactly the elements of F. For any node $x \in \mathcal{T}$ we define $\mathcal{T}(x)$ to be the set of leaf nodes whose ancestor x is. For the root node r of T we have that $\mathcal{T}(r) = F$. A taxonomy T defines a partial order between nodes $x, y \in \mathcal{T}$, as follows. We say that x precedes y, denoted $x \preceq y$, whenever $\mathcal{T}(y) \subseteq \mathcal{T}(x)$, i.e., x is an ancestor of y. If $x \not\preceq y$ and $y \not\preceq x$ we say the nodes are not comparable. Additionally, we denote with children(x) the children of a node $x \in \mathcal{T}$, and with parent(x) its parent.

An example of a taxonomy $\mathcal{T}$ is shown on top of Figure 2(a). The leaf nodes are the set of species $F = \{a, b, c, d, e\}$. The internal nodes of the taxonomy x, y, z and r represent possible categorizations of the data attributes: for instance $\mathcal{T}(y) = \{a, b, c\}$ could correspond to the grouping of carnivores; $\mathcal{T}(z) = \{d, e\}$ could correspond to herbivores and $\mathcal{T}(r) = \{a, b, c, d, e\}$ to all animals. We have that $y \preceq x$; y and z are not comparable; also, children$(r) = \{y, z\}$.

An *antichain* $X = \{x_1, \ldots, x_k\}$ of a taxonomy $\mathcal{T}$ is a subset of nodes from $\mathcal{T}$ that are not comparable. A *covering antichain* X is an antichain that covers all leaves, i.e. $\cup_{x \in X} \mathcal{T}(x) = F$. In the toy example in Figure 2, for instance $\{x, z\}$ is an antichain; an example of a covering antichain would be the set of nodes $\{x, c, z\}$.

Additionally, consider an $n \times m$ data matrix D where each row vector is defined along m dimensions of the feature set F. In the paleontological application the rows of the matrix correspond to sites and columns to species. We denote by D^f the $n \times 1$ column vector of D of a feature $f \in F$. In our application, D^f is a column vector with information of the absence/presence of species f in each site of the data. A useful alternative description of the matrix D is to see it as the the collection of column vectors D^f from $f \in F$, that is: $D = D^{f_1} : \ldots : D^{f_m}$ for all $f_i \in F$, where ":" denotes juxtaposition of column vectors.



a	b	c	d	e	Age
1	0	0	0	0	29
0	1	0	0	0	31
0	0	1	0	0	27
1	0	1	0	0	32
0	1	1	0	0	29
0	0	0	1	0	50
0	0	0	0	1	62
0	0	0	1	1	55

(a)

D(y)	D(z)	Age
1	0	29
1	0	31
1	0	27
1	0	32
1	0	29
0	1	50
0	1	62
0	1	55

(b)

Fig. 2. An example of a taxonomy tree on top of a set of features $F = \{a, b, c, d, e\}$ (a), and after projecting the data on nodes y and z of the taxonomy by means of an "OR" aggregator over the columns covered by these nodes (b).

Given a node $x \in \mathcal{T}$, denote by $D(x)$ the aggregation of the columns in D covered by x. Formally, $D(x) = \alpha_x(D^{f_1}, .., D^{f_k})$ with $f_i \in T(x)$, where α_x is a function computed over the input columns selected by x and returning a $n \times 1$ column. For example, α_x could be "OR" or "AND", over the covered columns; in the paleontological application it is typically "OR". In general, for a set of nodes $X = \{x_1, \ldots, x_k\}$ from $\mathcal{T}$ we let $D(X) = D(x_1) : \ldots : D(x_k)$ be the concatenation of projections from each node in X. Consider the example in Figure 2 and assume that the aggregation function α_x associated to every node $x \in \mathcal{T}$ is a logical "OR". The result of $D(\{y, z\})$ is shown in Figure 2(b).

Projecting the data on a set of nodes of the taxonomy often returns much better aggregates than the original values. To evaluate the quality of the different data projections given by a set of nodes we will consider a target variable $v \notin F$ whose value we wish to predict, i.e., we are interested in predicting the values of the column vector D^v. In Figure 2 the variable v corresponds to the age of the site. The goal of a learning algorithm $A(D, v)$ (e.g., a predictor or classifier depending on the domain of v) is to construct a model from D, in order to predict v for new unseen data points. We denote with $\text{err}[A(D, v)]$ the error returned by the inferred model. The error can be calculated, e.g., as the square of the differences between the predictions of the model $A(D, v)$ and the real value of v in a separate test set.

Following the example from Figure 2, the projection $D(\{y, z\})$ would allow a learning algorithm to clearly distinguish between two different segments of age: those that are younger and those that are older than 40 million years.

We study the following problem on feature selection on taxonomies.

Algorithm 1 `Greedy` algorithm

```
1: **Input:** Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
2: **Output:** Antichain $X$ from $\mathcal{T}$ and $\mathrm{err}[A(D(X), v)]$
3: $X = \emptyset$
4: $N =$ all nodes from $\mathcal{T}$
5: **repeat**
6:     $x^* = \arg\min_{x \in N}\ \mathrm{err}[A(D(X \cup \{x\}), v)]$ {Take the best next node from $\mathcal{T}$}
7:     $X = X \cup \{x^*\} \cup \{n \in N | n$ is sibling leaf of $x^*\}$
8:     $N = N \backslash \{n \in N | n \preceq x^*$ or $x^* \preceq n\}$
9: **until** $N$ is empty
10: Return $X$ and $\mathrm{err}[A(D(X), v)]$
```

Problem 1 (TAXONOMY ANTICHAIN SELECTION). Given a dataset D defined over attributes $\mathcal{F} \cup \{v\}$, and let $\mathcal{T}$ be a taxonomy tree on the set F. Select a covering antichain X of T that minimizes $\mathrm{err}[A(D(X), v)]$ for the variable v.

We refer to the TAXONOMY ANTICHAIN SELECTION problem as TAS. The idea of finding an antichain that covers all leaf nodes is natural in several applications: antichains represent sets of nonredundant features that, potentially, will explain the structure of the data much better than having an unmanageable number of leaf features F. This is especially the case in paleontology, where we would like aggregations of the species at different levels.

Proposition 1. *The* TAS *problem is NP-complete.*

This proposition is proven via a reduction from the SATISFIABILITY problem for certain choices of the aggregation function. Details are omitted due to space constraints.

3 Algorithms

This section describes four algorithms for the TAS problem. As a baseline for our proposals we also show how to sample antichains uniformly at random. Without loss of generality we assume that the taxonomy $\mathcal{T}$ is rooted, i.e., there is a node x such that $\mathcal{T}(x) = F$.

3.1 The greedy algorithm

The scheme of `Greedy` approach is shown in Algorithm 1. The idea is simple and inspired by a forward selection technique: at every step `Greedy` takes the node $x \in \mathcal{T}$ with the least error increase (line 6). To enforce the antichain constraint, all nodes in the path from/to the selected node x have to be removed as they cannot occur together with x in the same solution set. An incremental solution is constructed until all leaf nodes have been covered. Notice that when selecting a leaf node, this will always enforce the addition of all its sibling leaves

Algorithm 2 `Bottom-up Selection` algorithm

```
 1: Input: Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
 2: Output: Antichain $X$ from $\mathcal{T}$ and $\mathrm{err}[A(D(X), v)]$
 3: $X = F$ {Initialize with leaves from $\mathcal{T}$}
 4: $e = \mathrm{err}[A(D(X), v)]$
 5: $B^+ = \{y \mid y \in \mathcal{T},\ \mathrm{children}(y) \subseteq X\}$ {Nodes from $\mathcal{T}$ upper bordering $X$}
 6: repeat
 7:   for all $n \in B^+$ do
 8:     $X'_n = X \backslash \mathrm{children}(n) \cup \{n\}$ {Swap children of $n$ with $n$ in the antichain}
 9:     $e'_n = \mathrm{err}[A(D(X'_n), v)]$
10:   end for
11:   $n^* = \arg\min_{n \in B^+} e'_n$ {Take the best from the above swaps}
12:   if $e'_{n^*} \le e$ then
13:     $X = X'_{n^*}$
14:     $e = e'_{n^*}$
15:     $B^+ = B^+ \backslash n^* \cup \{y \mid y = \mathrm{parent}(n^*),\ \mathrm{children}(y) \subseteq X\}$ {Update $B^+$}
16:   end if
17: until Error $e$ cannot be further reduced
18: Return $X$ and $\mathrm{err}[A(D(X), v)]$
```

into the solution set (line 7); that selecting a leaf node will always enforce the addition of all its siblings into the solution set, this is inherent to the requirement of finding an antichain that covers all leaf nodes.

3.2 Top-down and bottom-up selection algorithms

The following solutions are inspired by traditional backward elimination algorithms. In our problem though, the steps for feature selection have to take into account the topology of the taxonomy tree and ensure the antichain constraint of the selected set of nodes. The scheme of the `Bottom-up Selection` algorithm is given in Algorithm 2. `Bottom-up Selection` starts by taking all the leaf nodes F as an initial antichain X (line 3). At all times, the positive border B^+ maintains the nodes from the taxonomy tree located right above the current antichain X (lines 5 and 15). The algorithm iterates to improve the solution in a bottom-up fashion: first, it evaluates all swaps between a node in the border B^+ and its children (currently belonging to the solution set) by calculating and storing the error loss (lines 7–10); then, the antichain X is updated with the best of those swaps (lines 12–16). The algorithm stops when the error cannot be further reduced with any of the swaps. A complementary approach is the `Top-down Selection` algorithm. The starting point X is initialized to be the root node of the taxonomy, and the negative border B^- maintains at all times nodes right below the current antichain solution. Similarly as before, the algorithm would try to find a better antichain by taking the best swap in a top-down fashion from the taxonomy tree.

Algorithm 3 `Taxonomy Min-Cut` algorithm

```
1: **Input:** Data $D$; a taxonomy $\mathcal{T}$; a learning algorithm $A$.
2: **Output:** Antichain $X$ from $\mathcal{T}$ and $\text{err}[A(D(X), v)]$
3: Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be the vertices and edges of the taxonomy $\mathcal{T}$
4: Let $G = (V, E)$ be an undirected weighted graph as follows,
5:     $V = V_{\mathcal{T}} \cup \{s, t\}$ {Nodes from $\mathcal{T}$, plus a source $s$ and a target $t$}
6:     $E = E_{\mathcal{T}} \cup \{(s, r) \mid r \text{ is root of } \mathcal{T}\} \cup \{(f, t) \mid f \in F\}$
7: Set the weight $w(e)$ of edges $e \in E$
8: **for** edge $e = (x, y) \in G$ **do**
9:   **if** $y$ is the target node $t \in G$ **then**
10:      $w(e) = +\infty$
11:   **else**
12:      $w(e) = \text{score}(y) \times h(y)$
13:   **end if**
14: **end for**
15: Find the minimum cut edges $C$ of $G$ {E.g. with Max-flow algorithms}
16: $X = \{y \mid (x, y) \in C\}$ {$X$ is the set of nodes selected below the min-cut}
17: Return $X$ and $\text{err}[A(D(X), v)]$
```

3.3 Minimum cut based algorithm

The algorithm `Taxonomy Min-Cut` is based on finding the minimum cut of a graph G derived from $\mathcal{T}$. The Minimum Cut problem on a weighted undirected graph asks for a partition of the set of vertices into two parts such as the cut weight (sum of the weights on the edges connecting the two parts) is minimum. This problem can be solved using the max-flow min-cut theorem [1, 6].

We map the antichain selection problem into a Minimum Cut problem by constructing an undirected graph G which is a simple augmented version of $\mathcal{T}$. A scheme is shown in Algorithm 3. First, we extend taxonomy $\mathcal{T}$ with two extra nodes: source node s and target node t (line 5); second, we add edges between the root of $\mathcal{T}$ and the source s, and between leaves of $\mathcal{T}$ and the target node t (line 6). For notational convenience we consider the undirected edges $(x, y) \in G$ to be implicitly directed towards the target node t, that is, whenever $x \preceq y$ we will always write the edge as (x, y). For example, we always have $(s, r) \in G$, being r the root of $\mathcal{T}$; also $(f, t) \in G$ for all leaves $f \in F$ of $\mathcal{T}$.

An s, t-cut is then a set of edges in G whose removal would partition G into two components, one containing s and the other containing t. The following property follows.

Proposition 2. *Let $\mathcal{T}$ be a taxonomy. Then every s, t-cut from G not containing edges from $t \in G$ corresponds to a covering antichain and vice versa.*

Briefly, for a set of s, t-cut edges C of G, we have a covering antichain $X = \{y \mid (x, y) \in C\}$. That is, nodes belonging to the antichain are just below the s, t-cut. Similarly, each antichain X in $\mathcal{T}$ identifies an s, t-cut C separating source and target in G: we only need to select the edges $C = \{(x, \text{parent}(x)) \mid x \in X\}$.

Algorithm 4 `Antichain Sampler` algorithm

```
 1: Input: A taxonomy $\mathcal{T}$ rooted at $r$
 2: Output: Random antichain $X$ from $\mathcal{T}$
 3: Flip a biased coin that comes up heads with probability $1/\beta(r)$
 4: if heads then
 5:    $X = \{r\}$
 6: else
 7:    for nodes $y \in$ children($r$) do
 8:       Let $Z_y$ = Antichain Sampler($y$) {Recursive call with a tree rooted at $y$}
 9:    end for
10:    $X = \bigcup_{y \in \text{children}(r)} Z_y$
11: end if
12: Return $X$
```

To use the min-cut idea, it only remains to set the weights of the edges in G appropriately (lines 8–14, Algorithm 3). We define the weight of an edge $(x, y) \in G$ to be the product of two factors: (1) the score of y, namely score(y), and (2) the height factor of y, namely $h(y)$. The score (1) of a node will depend on the data D. Because a min-cut algorithm looks for a minimum weight cut in G, the small values of score of a node indicate good quality of the node. For example, as the score for a node $x \in \mathcal{T}$ we can use the inverse of the correlation coefficient between x and the target variable v, that is $1/\rho(x, v)$, assuming $+\infty$ when $\rho(x, v) = 0$; or also, we can take directly $\text{err}[A(D(x), v)]$ as the score.

The height factor (2) of a node has to be inversely proportional to the distance from that node to the root of $\mathcal{T}$. The reason is that, by construction, the s, t-cuts close to the root contain less edges. E.g., the single edge $(s, r) \in G$ forms a very small s, t-cut on its own and might be selected even with bad score. Typically, we use as $h(x)$ either $1/\text{height}(x)$, or directly $h(x) = \mathcal{T}(x)$ (that is, the number of leaves covered by x). This last choice of $h(x)$ maintains the natural property of making all the s, t-cuts equally costly if nodes in the taxonomy are all equally good in their score function. Finally, the weight of edges involved with the target node t are set to $+\infty$ (line 10, Algorithm 3) accordingly with Proposition 2.

In practice, the `Taxonomy Min-cut` only requires to evaluate the model once per node; this is done when setting the scores of the edges (line 7, Algorithm 3), so the number of calls to the classifier is linear. On the other hand, the algorithms such as `Greedy` or `Bottom-up` have to call the classifier at each evaluation step.

3.4 Sampling a random antichain

As a baseline to compare the previous approaches we use random antichains. Sampling them uniformly turns out to be an interesting problem in its own right. Formally, given the taxonomy tree $\mathcal{T}$ rooted at $r \in \mathcal{T}$, let $\beta(r)$ be the total number of covering antichains of $\mathcal{T}$. Immediately, the recursion follows: $\beta(r) = \prod_{x \in \text{children}(r)} \beta(x) + 1$. For every $x \in \mathcal{T}$ we have that $\beta(x)$ corresponds to the number of antichains that can be sampled from the subtree of $\mathcal{T}$ rooted at x. For the leaf nodes $f \in F$ we have always that $\beta(f) = 1$. The scheme of the `Antichain Sampler` algorithm is shown in Algorithm 4. The proof of the following proposition comes naturally by induction on the recursion. Details are omitted in this paper due to space constraints.

Proposition 3. *The* `Antichain Sampler` *algorithm samples antichains from $\mathcal{T}$ uniformly at random.*

4 Experiments

The paleontological data contains information of fossil mammals in Europe and Asia [8]. Columns correspond to species and rows correspond to sites. There are originally around 2000 sites and 900 species. Because the raw data contains rare species, as well as sites with only few species, we omit some of these from consideration by creating variations of the original dataset. In the variations named PALEO_X_Y, we first remove species that occur less than X times in the raw data; after this, we remove sites that have less than Y species after the elimination of the rare species. For the experiments we use the datasets PALEO_10_10, PALEO_5_5 and PALEO_2_2. Note that by removing species and sites we are making the data more sparse, hence PALEO_2_2 is the sparsest data matrix of the three variations. The sizes of the taxonomies for these three datasets ranges from 700 and 2500 nodes. Also, in addition to the entire taxonomy, we will consider different subtrees separately, each one of which corresponds to an order, such as carnivores or rodents. The task is always to estimate the age of fossil discovery sites with linear regression.

We implemented the proposed algorithms as components of the Weka machine learning software.[1] For the calculation of the error function $\mathrm{err}[A(D(x), v)]$ and evaluation of the models, we divide the dataset in two folds, training and test. The error used in the model construction phase is based on error in the test data. After that, we compare the solutions of the different algorithms with three criteria: (1) size in number of nodes of the returned antichain; (2) number of calls to the linear regressor needed to learn the final model (this provides an idea of its running time); (3) the correlation coefficient between predictions and actual ages in the test data; and (4) the root mean squared error of the model (RMSE, again in the test data).

The baseline for algorithms is always the random antichain. We complement each one of the experiments with a histogram of the correlation coefficients of the real and predicted ages of 100 random antichains when necessary. We also compare the algorithms with models that are based on selecting a certain level of the taxonomy. These levels are (from general to specific) FAMILY, GENUS and SPECIES. The SPECIES level consists of the leaf nodes, while the GENUS (FAMILY) level is located one (two) step(s) above the leaf level.

Table 1 reports the results when using the complete taxonomy tree available over the features. In PALEO_10_10 fixing the antichain at the level of the leaves

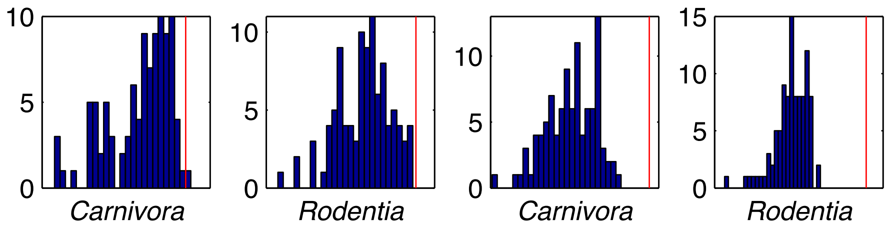
[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Table 1. Results for the complete taxonomies with Paleo_10_10 and Paleo_5_5. *Columns:* "size" is the number of nodes of the discovered antichain; "calls" is the number of calls to the linear regressor in the model construction phase; "corr" corresponds to the the correlation coefficient of the linear regression when using the corresponding antichain (in the test data); "RMSE" reports the root mean squared error of the linear regression when using the corresponding antichain (in the test data). *Rows:* Family, Genus and Species select a fix level of the taxonomy as the antichain. `Random` reports the mean values over 100 antichains.

	Paleo_10_10				Paleo_5_5			
	size	calls	corr	RMSE	size	calls	corr	RMSE
Family	49	-	0.81	3.40	63	-	0.76	3.68
Genus	237	-	0.20	16.4	373	-	0.60	6.53
Species	428	-	0.88	2.62	867	-	0.65	6.04
`Greedy`	246	46534	0.67	4.89	464	140488	0.33	12.89
`Top-down`	145	1817	0.82	3.09	197	2619	0.79	3.91
`Bottom-up`	375	3737	0.86	2.65	794	10409	0.75	4.91
`Tax Min-cut`	201	726	0.51	6.62	306	1329	0.77	4.55
`Random`	325	-	0.83	3.04	615	-	0.1	150

(Species) is the best obtained solution. In this case, algorithm `Bottom-up` is the closest to this species level. As we turn data into a sparser format with Paleo_5_5, we observe a clear gain given by `Tax Min-cut`, which still it does not beat `Top-down` but seems to gain learning power as the data goes sparser. Indeed, we observed that `Tax Min-cut` typically refines the best of the three fixed antichains (i.e., it upgrades the best of the three), typically this is the antichain at the Genus level, and so, it can adapt better to sparser formats. Random antichains perform in average surprisingly well on Paleo_10_10; as the data turns sparser, random antichains are not able to predict as good as our proposed algorithms.

Finally we also ran the same experiments with Paleo_5_5 and Paleo_2_2 using different subtrees of the taxonomy. These correspond to taxonomies of two orders of mammals; the *Carnivora* and *Rodentia.* Comparison with random antichains can be seen in Figure 3. Here the vertical line indicates the correlation obtained by the `Tax Min-cut` algorithm, which we expect to perform well on sparse inputs. With Paleo_5_5 we observe that `Random` can give results that are as good (or maybe even better) than those obtained with `Tax Min-cut`. However, in case of Paleo_2_2 the difference is more pronounced and `Tax Min-cut` gives consistently better results than simply selecting a random antichain. This is also the case with `Bottom-up`. The results reported in Table 2 show the same conclusion as above: `Tax Min-cut` and `Bottom-up` are able to generalize much better than other algorithms and select typically a better refinement of the Genus level. From the computational perspective we should highlight that `Tax Min-cut` tends to run faster than the other algorithms, which typically call the linear regressor more times than the number of nodes in the taxonomy.



(a) PALEO_5_5 (b) PALEO_2_2

Fig. 3. Histograms of the correlation between predicted and real age from 100 random antichains in the *Carnivora* and *Rodentia* subtrees of the taxonomy using PALEO_5_5 and PALEO_2_2. The vertical line indicates the performance of the solution given by the `Tax Min-cut` algorithm in each case.

Table 2. Results for the *Carnivora* and *Rodentia* subtrees of the taxonomy with PALEO_5_5 and PALEO_2_2. `Random` reports the mean values over 100 antichains.

								PALEO_5_5								PALEO_2_2
				Carnivora				*Rodentia*				*Carnivora*				*Rodentia*
	size	calls	corr	RMSE	size	calls	corr	RMSE	size	calls	corr	RMSE	size	calls	corr	RMSE
FAMILY	12	-	0.47	4.90	10	-	0.52	4.74	17	-	0.41	5.45	8	-	0.44	5.37
GENUS	61	-	0.48	4.86	121	-	0.79	3.45	140	-	0.36	5.87	75	-	0.63	4.74
SPECIES	112	-	0.19	6.74	344	-	0.39	12.5	347	-	0.20	8.21	215	-	0.12	23.9
`Greedy`	67	3228	0.45	5.42	163	14017	0.73	4.40	139	9501	0.31	6.11	294	36640	0.60	5.38
`Top-down`	25	99	0.46	5.34	151	1176	0.74	4.20	75	258	0.40	5.58	190	2338	0.69	4.48
`Bottom-up`	48	578	0.45	5.38	232	2050	0.65	5.45	221	2073	0.35	5.89	500	4222	0.27	11.9
`Tax Min-cut`	33	188	0.47	5.30	106	476	0.78	3.77	96	460	0.42	5.55	234	861	0.67	4.67
`Random`	84	-	0.41	5.16	231	-	0.67	4.74	222	-	0.24	7.36	178	-	0.52	5.34

5 Related work

The problem of selecting relevant features is a recurring theme in machine learning [2, 4, 10, 11]. Typical strategies rely on a heuristic search over the combinatorial space of all features, e.g.: backward elimination, forward selection, the filter approach, and the wrapper approach. The algorithms presented in this paper are inspired by some of these techniques: `Greedy` can be seen as a taxonomic forward selection algorithm; `Top-down` and `Bottom-up` have the flavour of backward elimination approaches that take into account the space of the taxonomy; finally, the `Taxonomy Min-Cut` solution can be seen as a filter approach.

There is also relevant work in machine learning on the problem of learning classifiers from attribute valued taxonomies [5, 15]. The work in [5] focuses on Bayesian Networks. The approaches in [15] focus on naïve Bayes classifiers and decision trees. Features are typically selected in a top-down fashion through the hypothesis space given by the taxonomy; also, the problem is studied specifically for the two mentioned learning algorithms. Our proposals complement those approaches by presenting feature selection in taxonomies as a general problem, independent of the learning algorithm one wishes to use. Other different scenarios for taxonomies occur when learning classifiers with class labels exhibiting a predefined class hierarchy, e.g. [3]. Finally, taxonomies have been the target of data mining as well: e.g. in association rules [13] or clustering [14].

6 Conclusions

We have considered the feature selection problem in taxonomies. Our motivating application is the problem of determining the age of fossil sites on the basis of the taxa found in the sites. We formulated the problem of finding the best selection of features from a hierarchy, showed that it is NP-complete, and gave four algorithms for the task. Of the algorithms, `Greedy`, `Bottom-up` and `Top-down` are inspired by previous feature selection approaches where taxonomies where not yet considered; `Tax Min-cut` uses the well-known min-cut max-flow theorem to provide with a quick and rather good choice of an antichain.

The empirical results show that from the proposed methods, especially `Tax Min-cut` works well; it performs at least as good as the best antichains that are based on a fixed level of the taxonomy. Future work involves applying the method to some interesting subsets of the paleontological sites, investigation of other applications, and further study of the properties of the algorithms.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice-Hall, Inc., 1993.
2. A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.
3. L. Cai and T. Hofmann. Exploiting known taxonomies in learning overlapping concepts. In *IJCAI '07*, pages 714–719, 2007.
4. M. Charikar, V. Guruswami, R. Kumar, S. Rajagopalan, and A. Sahai. Combinatorial feature selection problems. In *FOCS '00*, page 631, 2000.
5. M. desJardins, L. Getoor, and D. Koller. Using feature hierarchies in Bayesian network learning. In *SARA '02*, pages 260–270, 2000.
6. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
7. M. Fortelius, A. Gionis, J. Jernvall, and H. Mannila. Spectral ordering and biochronology of european fossil mammals. *Paleobiology*, 32:206–214, 2006.
8. Mikael Fortelius. Neogene of the old world database of fossil mammals (NOW). http://www.helsinki.fi/science/now/, 2008.
9. J. Jernvall and M. Fortelius. Common mammals drive the evolutionary increase of hypsodonty in the neogene. *Nature*, 417:538–540, 2002.
10. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.
11. N. Lavrač and D. Gamberger. Relevancy in constraint-based subgroup discovery. In *Constraint-Based Mining and Inductive Databases*, pages 243–266, 2004.
12. L.H. Liow, M. Fortelius, E. Bingham, K. Lintulaakso, H. Mannila, L. Flynn, and N.Chr. Stenseth. Stenseth higher origination and extinction rates in larger mammals. *PNAS*, 105:6097–6102, 2008.
13. R. Srikant and R. Agrawal. Mining generalized association rules. *Future Gener. Comput. Syst.*, 13(2-3):161–180, 1997.
14. C. Yun, K. Chuang, and M. Chen. Using category-based adherence to cluster market-basket data. In *ICDM '02*, page 546, 2002.
15. J. Zhang, D.-K. Kang, A. Silvescu, and V. Honavar. Learning accurate and concise naïve bayes classifiers from attribute value taxonomies and data. *Knowl. Inf. Syst.*, 9(2):157–179, 2006.