Efficient Feature Construction by Meta Learning – Guiding the Search in Meta Hypothesis Space

Ingo Mierswa MIERSWA@LS8.CS.UNI-DORTMUND.DE
Michael Wurst WURST@LS8.CS.UNI-DORTMUND.DE
Artificial Intelligence Unit, Department of Computer Science, University of Dortmund, Germany

Abstract

Choosing the right internal representation of examples and hypothesis is a key issue for many learning problems. Feature construction is an approach to find such a representation independently of the underlying learning algorithm. Unfortunately, the construction of features usually implies searching a very large space of possibilities and is often computationally demanding. In this work, we propose an approach to feature construction that is based on Meta Learning. Learning tasks are stored together with a corresponding set of constructed features in a case base. This case base is then used to constraint and guide the feature construction for new tasks. Our proposed method consists essentially of a new representation model for learning tasks and a corresponding two step distance measure. Our approach is unique as it enables us to apply case based feature construction not only on a large scale, but also in distributed learning scenarios in which communication cost plays an important role. Using the two step process, the accuracy of recommendations can be increased while not loosing the benefits of efficiency. The theoretical results are also confirmed by experiments on both synthetical data and data obtained from a distributed learning scenario on audio data.

1. Introduction

Many inductive learning problems cannot be solved accurately by using the original feature space. This is due to the fact that standard learning algorithms cannot represent complex relationships as induced for example by trigonometric functions. For example, if only base features X_1 and X_2 are given but the target function depends highly on $X_c = \sin(X_1 \cdot X_2)$, the construction of the feature X_c would ease learning – or is necessary to enable any reasonable predictions at all. Feature construction can help to deal with such problems by enriching the original feature space with additional features (Blum & Langley, 1997; Dash & Liu, 1997; Koller & Sahami, 1996). Many algorithms can be formulated in terms of kernel functions (Schölkopf & Smola, 2002; Vapnik, 1995). Since the search for an appropriate kernel is equivalent to the search for an appropriate feature space transformation the success of these algorithms also underlines the need for feature construction. Unfortunately, feature construction is a computationally very demanding task often requiring to search a very large space of possibilities (Wolpert & Macready, 1997). In this work we consider a scenario in which several learners face the problem of feature construction on different learning problems. The idea is to transfer constructed features between similar learning tasks to speed up the generation in such cases in which a successful feature has already been generated by another feature constructor. Such approaches are usually referred to as Meta Learning (Vilalta & Drissi, 2002).

The importance of the representation bias, which is closely related to feature construction, was recognized since the early days of Meta Learning research (Baxter, 1995; Baxter, 2000). Inductive learning is often regarded as a search for an optimal hypothesis in a predefined space of possible hypothesis. A major paradigm for Meta Learning is to use other learning tasks to guide or constraint this search for a given learning task. In the context of our approach, the space of possible hypothesis is not fixed. It is optimized itself by generating new features describing input examples in a way that is more suitable for the subsequent learning algorithm. This procedure can there-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

fore be seen as search in a meta hypothesis space, containing all possible representations of a given set of examples. While there are many efficient algorithms for learning in fixed, predefined hypothesis spaces, searching for an optimal hypothesis space can often be only tackled by uninformed search strategies as evolutionary algorithms, which are in turn quite inefficient. It is therefore very attractive to search for an alternative strategy. In contrast to many classical Meta Learning approaches trying to guide the search in a predefined hypothesis space, our approach tries to guide the search in the meta hypothesis space.

As for many other Meta Learning methods (e.g. (Ben-David & Schuller, 2003; Thrun & O'Sullivan, 1996)), the definition of a similarity between different learning tasks is a key issue for our approach. Considering the constraints and challenges of a distributed learning scenario we develop and analyze a method for feature construction based on Meta Learning that estimates the distance of two learning tasks using only base feature weights but not the data supporting the learning tasks. All learners weight a common set of base features according to their relevance to the given learning task. Some of the learners, that have already performed feature construction, send these relevance weights and the constructed features to a case base. From this case base, learners can retrieve possibly relevant features by sending their own base feature weights. The central challenge is to find a measure that compares two learning tasks accurately and efficiently at the same time.

1.1. Outline

To motivate our approach, we discuss the challenges of applying Meta Learning in large, highly distributed domains in section 2. We give some definitions and conceptual ideas in section 3. In section 4 we analyze the problem of comparing learning tasks based only on feature weights by stating some basic conditions. We will see that the weights calculated by Support Vector Machines in combination with the Manhattan distance fulfill all conditions described in section 4. In section 5 we will discuss some enhancements allowing even more efficient and accurate case retrieval for case based feature construction. The theoretical results are also confirmed by experiments that will be discussed in section 6.

2. Applying Meta Learning in Large, Highly Distributed Domains

Our approach is motivated by our work in the field of distributed multimedia learning. Concretely, we face a scenario in which several users independently arrange multimedia items according to some personal classification scheme. New items should be added to this scheme automatically. This is achieved by Machine Learning methods based on a set of audio features. The audio features best suited for a given learning task are chosen from an infinite space of possible features as described in (Mierswa & Morik, 2005). Although this approach is very well suited to achieve a high accuracy, it is computationally very demanding and therefore not applicable to an interactive end user system. Meta Learning is an approach to achieve both high accuracy and efficiency. The basic idea is that although classification schemes created by the users may differ in any possible way, we can assume that at least some of them resemble each other to some extend, e. g. many users arrange their music according to the genre.

In our opinion, this scenario is prototypical for many current scenarios. There is a clear trend to substitute centralized data mining and learning architectures by highly distributed ones (Park & Kargupta, 2002). The wide spread use of mobile devices makes the vision of ubiquitous machine learning very realistic. Corresponding systems are characterized on the one hand by massive amounts of data fragmented in many local, partially overlapping databases. On the other hand they are characterized by a high degree of heterogeneity on different levels. Firstly, the data representation can differ among different nodes. Secondly, the learning tasks and methods at different nodes can differ in any possible way. Connected to this point is the observation, that learning tasks often occur in an ad hoc fashion, based on a current information need of a user or analyst and they may differ strongly from the ones anticipated in the original design of the system. Finally, the computational power and the efficiency of the network connection may vary to a large extend. Mobile devices for example will often have only poor processing speed and an inefficient network connection.

While some of these problems can best be approached by methods developed in the area of Distributed Data Mining, others give rise to new opportunities and challenges in Meta Learning. Strong heterogeneity in learning tasks demands new methods to compare learning tasks effectively. The large number of individual learners and the ad hoc character of many learning problems demands efficient, flexible Meta Learning methods. Minimal exchange of information is often a constraint for the applicability of the system, as to allow the application in areas with poor network connection and high communication costs. Trivial solutions, as sending a whole data set form a mobile device for

example to a computationally more powerful node are ruled out by this constraint.

The methods described in this work represent a new solution to some of these problem and challenges.

3. Basic Concepts

Before we state the conditions which must be met by any method comparing learning tasks using feature weights only, we first introduce some basic definitions. Let T be the set of all learning tasks, a single task is denoted by t_i. Let X_i be a vector of random variables for task t_i and Y_i another random variable, the target variable. These obey a fixed but unknown probability distribution $Pr(X_i, Y_i)$. The components of X_i are called *features* X_{ik}. The objective of every *learning task* t_i is to find a function $h_i(X_i)$ which predicts the value of Y_i. We assume that each set of features X_i is partitioned in a set of *base features* X_B which are common for all learning tasks $t_i \in T$ and a set of *constructed features* $X_i \setminus X_B$.

We now introduce a very simple model of feature relevance and interaction. The feature X_{ik} is assumed to be irrelevant for a learning task t_i if it does not improve the classification accuracy:

Definition 1. *A feature X_{ik} is called* IRRELEVANT *for a learning task t_i iff X_{ik} is not correlated to the target feature Y_i, i. e. if $Pr(Y_i|X_{ik}) = Pr(Y_i)$.*

The set of all irrelevant features for a learning task t_i is denoted by IF_i.

Two features X_{ik} and X_{il} are alternative for a learning task t_i, denoted by $X_{ik} \sim X_{il}$ if they can be replaced by each other without affecting the classification accuracy. For linear learning schemes this leads to:

Definition 2. *Two features X_{ik} and X_{il} are called* ALTERNATIVE *for a learning task t_i (written as $X_{ik} \sim X_{il}$) iff $X_{il} = a + b \cdot X_{ik}$ with $b > 0$.*

This is a very limited definition of alternative features. However, we will show that most weighting algorithms are already ruled out by conditions based on this simple definition.

4. Comparing Learning Tasks Efficiently

The objective of our work is to speed up feature construction and improve prediction accuracy by building a case base containing pairs of learning tasks and corresponding sets of constructed features. We assume that a learning task t_i is completely represented by a feature weight vector w_i. The vector w_i is calculated from the base features X_B only. This representation of learning tasks is motivated by the idea that a given learning scheme approximate similar constructed features by a set of base features in a similar way, e. g. if the constructed feature "$\sin(X_{ik} \cdot X_{il})$" is highly relevant the features X_{ik} and X_{il} are relevant as well.

Our approach works as follows: for a given learning task t_i we first calculate the relevance of all base features X_B. We then use a distance function $d(t_i, t_j)$ to find the k most similar learning tasks. Finally, we create a set of constructed features as union of the constructed features associated with these tasks. This set is then evaluated on the learning task t_i. If the performance gain is sufficiently high (above a given threshold) we store task t_i in the case base as additional case. Otherwise, the constructed features are only used as initialization for a classical feature construction that is performed locally. If this leads to a sufficiently high increase in performance, the task t_i is also stored to the case base along with the locally generated features.

While feature weighting and feature construction are well studied tasks, the core of our algorithm is the calculation of d using only the relevance values of the base features X_B. In a first step, we define a set of conditions which must be met by feature weighting schemes. In a second step, a set of conditions for learning task distance is defined which makes use of the weighting conditions.

Weighting Conditions. *Let w be a* WEIGHTING FUNCTION *$w : X_B \to \mathbb{R}$. Then the following must hold:*

(W1) $w(X_{ik}) = 0$ *if $X_{ik} \in X_B$ is irrelevant*

(W2) *$F_i \subseteq X_B$ is a set of alternative features. Then*

$$\forall S \subset F_i : \sum_{X_{ik} \in S} w(X_{ik}) = \sum_{X_{ik} \in F_i} w(X_{ik}) = \hat{w}$$

(W3) $w(X_{ik}) = w(X_{il})$ *if $X_{ik} \sim X_{il}$*

(W4) *Let AF be a set of features where*

$$\forall X_{ik} \in AF : X_{ik} \in IF_i \quad \vee \quad \exists X_{il} \in X_B : X_{ik} \sim X_{il}.$$

Then

$$\forall X_{il} \in X_B : \nexists X_{ik} \in AF : X_{il} \sim X_{ik} : w'(X_{il}) = w(X_{il})$$

where w' is a weighting function for

$$X'_B = X_B \cup AF$$

These conditions state that irrelevant features have weight 0 and that the sum of weights of alternative features must be constant independently of the actual number of alternative features used. Together with the last condition it will be guaranteed that a set of alternative features is not more important than a single feature.

Additionally, we can define a set of conditions which must be met by distance measures for learning tasks which are based on feature weights only:

Distance Conditions. *A* DISTANCE MEASURE d *for learning tasks is a mapping* $d : T \times T \rightarrow \mathbb{R}^+$ *which should fulfill at least the following conditions:*

(D1) $d(t_1, t_2) = 0 \Leftrightarrow t_1 = t_2$

(D2) $d(t_1, t_2) = d(t_2, t_1)$

(D3) $d(t_1, t_3) \leq d(t_1, t_2) + d(t_2, t_3)$

(D4) $d(t_1, t_2) = d(t'_1, t'_2)$ *if*
$X'_B = X_B \cup IF$ *and* $IF \subseteq IF_1 \cap IF_2$

(D5) $d(t_1, t_2) = d(t'_1, t'_2)$ *if*
$X'_B = X_B \cup AF$ *and*
$\forall X_k \in AF : \exists X_l \in X_B : X_k \sim X_l$

(D1)–(D3) represent the conditions for a metric. These conditions are required for efficient case retrieval and indexing, using e. g. M-Trees (Ciaccia et al., 1997). (D4) states that irrelevant features should not have an influence on the distance. Finally, (D5) states that adding alternative features should not have an influence on distance.

Many common feature weighting algorithms including Relief, information gain, and feature selection do not fulfill the weighting conditions stated above. Furthermore, the Euclidian distance can not be used for learning task distance in our scenario (Mierswa & Wurst, 2005). In this section we will state that the feature weights delivered by a linear Support Vector Machine (SVM) (Schölkopf & Smola, 2002; Vapnik, 1995) obeys the proposed weighting conditions.

Theorem 1. *The feature weight calculation of SVMs with linear kernel function meets (W1)–(W4).*

In order to calculate the distance of learning tasks based only on a set of base feature weights we still need a distance measure that met the conditions (D1)–(D5).

Theorem 2. *Manhattan distance does fulfill the conditions (D1)–(D5).*

Proofs can be found in (Mierswa & Wurst, 2005). We conclude that SVM feature weights in combination with Manhattan distance fulfill the necessary constraints for a learning task distance measure based on feature weights.

5. Exploiting the Similarity of Constructed Features

In the last section we discussed a distance measure on a set of common base features. The calculations can be performed in linear time of both the number of training cases and the number of features. Furthermore, since the measure is a metric, efficient indexing of cases is possible using data structures like M-Trees.

However, if the base features are not sufficient to approximate the target function the base weights alone might be a poor indicator for learning task distance. In the following we introduce two extensions of the case based feature construction approach. First, we incorporate the similarity of constructed features into the distance measure presented above. This follows the idea that for similar learning tasks similar features must be constructed. Without loss of generality, the constructed features can be modeled as function trees. In the case of value series data the constructed features can be given as method trees (Mierswa & Morik, 2005). In order to take the similarity of constructed features into account, the distance between the feature trees must be calculated. It has been shown that calculating the difference of two unordered labeled trees is an NP-complete problem (Zhang et al., 1995). Approximations for this problem using syntactical heuristics are still not feasible for large feature trees and case bases (Klein, 1998). Additionally, the data distribution is not considered by syntactical approaches. For example, $\sin(x)$ is very similar to x for small absolute values of x but of course not for $x \rightarrow \infty$. Therefore, we are using a probabilistic sampling approach considering the ranges of interest instead of syntactical heuristics. The main routine just compares each constructed features of the first learning task with all constructed features of the second learning task. The minimum distances for all features are summed up and returned (Figure 1).

We employ a sampling approach for the routine `calc_distance` which ensures that the feature construction distance is calculated in some range of interest, i.e. with respect to the data distributions in the base dimensions. Since the data is not accessible, the basic idea is to generate m artificial data points in the base space X_B and construct both features X_{ik} and X_{jl} for this small artificial data set. The distance

```
Given: learning tasks t_i and t_j

total = 0;
for all constructed features X_ik \ X_B do {
  dist = Infinity;
  for all constructed features X_jl \ X_B do {
    dist = min(dist, calc_distance(X_ik, X_jl));
  }
  total = total + dist;
}
return total;
```

Figure 1. The main routine to calculate the distance between two different feature sets including constructed attributes.

```
Given: - base feature std. dev. σ_i and σ_j
       - features X_ik and X_jl

// calculate ranges
for (p = 1 until |X_B|) do {
  σ_p = min(σ_ip,σ_jp);
}
// generate small artificial data set
artificial_data = empty;
for (i = 1 until m) do {
  generate base feature data using σ;
  construct X_ik and X_jl on generated data;
}
// calculate correlation and return as distance
r = correlation(X_ik,X_jl) on artificial_data;
return 1 - (r * r);
```

Figure 2. The routine `calc_distance` which calculates the distance between two constructed features using a range sensitive sampling and the squared correlation on a small artificially generated data set with size m.

can then be calculated with help of the correlation coefficient or the absolute deviation of both constructed features. To ensure that the m artificial points are generated in interesting ranges of all dimensions the data ranges of all features must be submitted to the case base in addition to the base feature weights. If the data is standardized only the standard deviation must be transfered. Figure 2 shows the sampling based algorithm `calc_distance` using the squared correlation as feature distance for constructed features.

Using a probabilistic sampling approach allows an estimation how the distance calculation varies with different possible samples. This way we are able to say how close or far from the actual distance our estimation is likely to be (Neal, 1993). Since the construction of features can be done in $O(1)$ (at least in the non-series case), the runtime of this sampling approach is quadratic in the number of constructed features and linear in the number m of sample points and cases, i.e. $O(|X_i \setminus X_B| \cdot |X_j \setminus X_B| \cdot m)$ for each case. To summarize this section, one can say that the sampling based distance measure for constructed features does not only consider the feature ranges but is also feasible for large numbers of constructed attributes and cases.

The second extension to the case based feature construction approach introduced above is to allow learners to query the case base repeatedly. We start using base features only and add further features in each iteration until a given stopping criterion is fulfilled. This can be a desired performance or a maximal amount of time. In each step, a learner can construct features on its own, e.g. with help of an evolutionary feature construction approach. This can also be seen the other way round: for an evolutionary approach the retrieval of similar cases and construction of new features is just another mutation of the input data.

5.1. Decreasing Runtime using a 2-Phase Distance Calculation

We have seen that the runtime of the discussed learning task similarity using constructed features is more efficient than other approaches calculating the syntactical tree distance of constructed features. However, a quadratic runtime might be to slow for real world applications using huge case bases. In this section, we combine both the simple base weight distance using SVM weights with Manhattan distance and the sampling based construction distance. In a first phase we employ only the base feature weight distance to find a candidate set of the k most promising cases. In a second phase we use a weighted combination of both, the weight distance and the construction sampling distance introduced in the last section. This leads to a number of k' cases from this candidate set whose constructed features are also constructed for the case at hand. Of course these steps can be embedded in the iterative learning process described above. The parameter k allows an adaption of the trade-off between runtime and accuracy depending on the application.

6. Experiments

All experiments were performed with the machine learning environment YALE (Fischer et al., 2002). Yale is unique in that for the first time a single environment covers and joins methods of Machine Learning, Meta Learning, and feature construction. Yale is open source and available from our web site[1].

[1] `http://yale.cs.uni-dortmund.de`.

6.1. Synthetical Data

Experiments were performed on synthetical data. All cases have an important property: linear regression schemes are not able to predict the target function without applying some feature construction. A case base containing maximal 1500 cases was generated. For each case the following was done:

Example set generation: 500 examples with five base features $X_1, \ldots, X_5$ were generated. The target variable obeys a randomly created regression function containing the building blocks $+$, $*$, sin, exp, abs and $1/x$. The maximal depth of the target function was 3, the probability for a leafs, i. e. a base attribute, was 0.2.

Weight calculation: The weights of the base features were calculated with a linear SVM.

Feature construction: An evolutionary approach for feature construction was used to generate new features from the base features (Ritthoff et al., 2002) with the following parameters: 400 generations, population size 10, and crossover probability 0.5.

Adding the case: the standard deviations of the base features, the base weights, and a syntactical description of the constructed features were added to the case base. Please note that the case base does not contain the data itself, information about the target variable, or the learned hypothesis.

Figure 3 shows the base feature weights of all cases after a dimensionality reduction onto two dimensions. The five "axes" represents the five base features, the outermost points on each axis derive from the cases $\exp(\exp(\exp(x_i)))$. Cases between two axes contain functional parts depending on both base features.

After creation of the case base the 241 test cases were created in the same manner but without feature construction. It was not possible to reliably determine the averaged relative error for these test cases without using feature construction since most SVM runs does not converge. The case based feature construction was performed with the constructed features of the most similar case only. If this case was randomly selected from the case base without using a distance measure, an averaged relative error of 4.010 and 8.720 respectively could only be achieved for 1500 and 250 cases. For the other case base sizes it was also not possible to calculate an error in a reasonable amount of time. All error estimations were done with a 10-fold cross validation. Table 1 summarizes the results and Figure 4 plots the averaged relative errors of all 241 test cases against the number of used training cases. The plot shows the errors for the combination of SVM feature weighting and Manhattan distance and the new 2-phase construction approach with Manhattan distance on the base feature weights in the first phase (20 cases) and a construction sampling weight of 50. Both approaches dramatically reduced the error compared to the randomly selected cases and the 2-phase distance outperforms the base weight only distance. However, further experiments has shown that increasing the number k' of similar cases reduce the performance gain of the 2-phase approach.

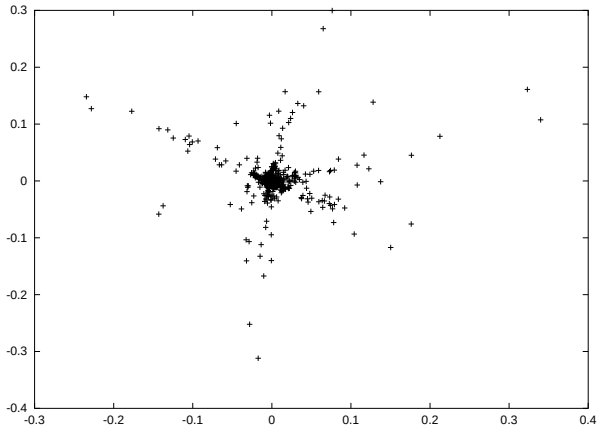

Figure 3. The base feature weights of the synthetical test cases after a dimensionality reduction on two dimensions.

Size	No	Rand.	Eucl.	Manh.	2-phase
50	∞	∞	0.228	0.213	0.145
100	∞	∞	0.201	0.196	0.106
250	∞	8.720	0.125	0.125	0.102
500	∞	∞	0.126	0.125	0.105
1000	∞	∞	0.120	0.119	0.106
1500	∞	4.010	0.121	0.116	0.106

Table 1. The averaged relative errors for the different approaches. The symbol ∞ indicates that no result was produced in a reasonable amount of time.

Our approach fulfills the constraints presented above and can speed up the process of feature construction considerably. The average time needed for the automatic construction of an optimal feature set without using the case base was 216.9 seconds. Using the feature construction induced by the most similar case reduces the time needed for learning to 2.18 seconds for the Manhattan distance and 3.67 seconds for the 2-phase approach. Using only the sampling construction distance also provides a very competitive error of 0.109 but needs a runtime of 58.73 seconds for each case.

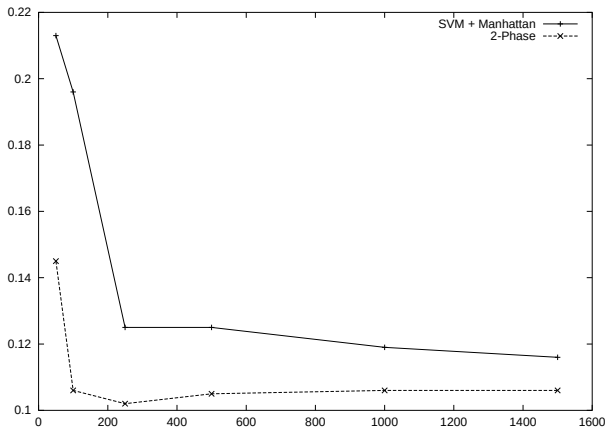

Figure 4. The results of case based feature construction. The averaged relative error of all 241 test cases is plotted against the number of cases used as case base. The 2-phase approach clearly outperforms the SVM plus Manhattan distance.

6.2. Real World Data

We performed additional experiments using real world data based on the application described in section 2. Our experiments are based on 20 taxonomies created by a group of students. The underlying audio data consists of about 2000 audio clips from the Garageband site[2]. The set of 20 seed taxonomies was extended by applying diverse mutation operations on them (such as adding or deleting items or subtrees). Although this leeds to data that is partially synthetic, this approach is justified for the given application area, as users of the system themselves often copy complete taxonomies from other users and modify them to suit their needs. So the data generation reflects the actual mechanisms present in the system. In a second step, the hierarchical classification problems are split into a set of flat, binary classification problems, by using each inner node with two or more children as a split point. If there are more than two sub nodes, two of them are selected at random. Also, all selected sub nodes have to contain more than 30 examples, as very small example sets would add a considerable amount of noise. In this case, random recommendation of features was compared with Manhattan and Euclidian distance using case bases of different sizes. Twelve base features were used to compare the cases. Figure 5 shows the base feature vectors of all cases after a singular value decomposition to transform the data into two dimensions. The accuracy was estimated by 10-fold cross validation using a test set of 100 classification tasks. Table 2 shows the results for different

[2]http://www.garageband.com

Size	Random	Manhattan	Euclidian
0	73.36	73.36	73.36
3	76.31	77.11	76.96
5	78.20	81.80	81.92
10	79.16	82.34	82.48
20	77.23	81.35	81.49
100	77.79	82.88	82.95

Table 2. The accuracy achieved by randomly choosing cases for feature construction, using the Manhattan and Euclidian distances of base feature weights.

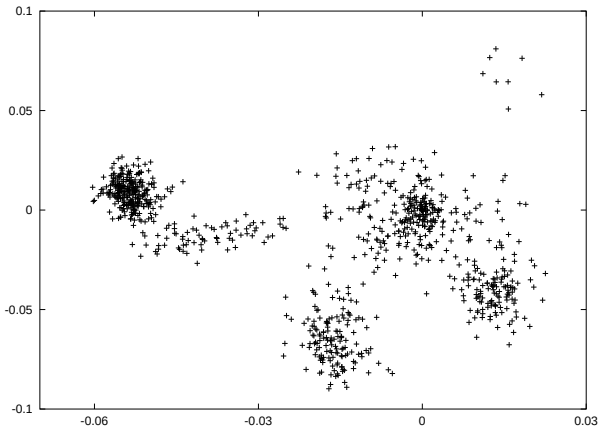

Figure 5. The base feature weights of the audio test cases after a dimensionality reduction on two dimensions.

weighting approaches. Since the "sampling" of series data is not as trivial as for the function experiments discussed above, no results for the 2-phase approach are provided for series data learning for now.

This evaluation is still preliminary, though it shows some basic results. Meta Learning does significantly improve the accuracy of the learners. If the number of prototypical concepts is quite small, only very few cases are sufficient to improve the results. There is no significant difference between Manhattan and Euclidian distance. This is due to the fact, that an optimized set of base features was used, that by construction contains only minimal redundancy. However, alternative features were the reason to prefer Manhattan over Euclidian distance. For our future work, we plan to gather additional empirical user data and to publish it as a benchmark data set for meta- and audio learning. Please contact us for a preliminary version of this data set.

7. Conclusion and Outlook

We presented a Meta Learning approach to feature construction that compares tasks using relevance

weights on a common set of base features only. A SVM as base feature weighting algorithm and the Manhattan distance fulfill some very basic conditions for such a learning task distance measure. We have presented some experimental results on both synthetical and real-world data indicating that our method can speed up feature construction considerably.

This paper introduces some enhancements for this case based feature construction approach. These enhancements include the distance calculation of already constructed features using a sampling procedure in a 2-phase distance measure. For our future work on the subject we plan to incorporate generated features from value series data in our distance measure too. Another direction of work is to analyze the relationship between base feature weights and generated features to get further insight which weighting methods are suitable. Finally, we plan to perform further empirical experiments using other real world data.

References

Baxter, J. (1995). Learning internal representations. *Proc. of the 8th annual conference on Computational Learning Theory* (pp. 311–320). ACM Press.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *Proc. of the Sixteenth Annual Conference on Learning Theory COLT 2003*.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 245–271.

Ciaccia, P., Patella, M., & Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. *Proc. of 23rd International Conference on Very Large Data Bases* (pp. 426–435). Morgan Kaufmann.

Dash, M., & Liu, H. (1997). Feature selection for classification. *International Journal of Intelligent Data Analysis*, *1*, 131–156.

Fischer, S., Klinkenberg, R., Mierswa, I., & Ritthoff, O. (2002). *Yale: Yet Another Learning Environment – Tutorial* (Technical Report CI-136/02). Collaborative Research Center 531, University of Dortmund, Dortmund, Germany.

Klein, P. (1998). Computing the edit-distance between unrooted ordered trees. *Proc. of the 6th Annual European Symposium* (pp. 91–102). Springer, Berlin.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proc. of the Thirteenth International Conference on Machine Learning* (pp. 129–134).

Mierswa, I., & Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine Learning Journal*, *58*, 127–149.

Mierswa, I., & Wurst, M. (2005). *Efficient case based feature construction for heterogeneous learning tasks* (Technical Report CI-194/05). Collaborative Research Center 531, University of Dortmund.

Neal, R. M. (1993). *Probabilistic Inference Using Markov Chain Monte Carlo Methods* (Technical Report). Department of Computer Science, University of Toronto.

Park, B., & Kargupta, H. (2002). Distributed Data Mining: Algorithms, Systems, and Applications. In N. Ye (Ed.), *Data Mining Handbook*, 341–358. IEA.

Ritthoff, O., Klinkenberg, R., Fischer, S., & Mierswa, I. (2002). A hybrid approach to feature selection and generation using an evolutionary algorithm. *Proc. of the 2002 U.K. Workshop on Computational Intelligence* (pp. 147–154).

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels – support vector machines, regularization, optimization, and beyond*. MIT Press.

Thrun, S., & O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC algorithm. *Proc. of the 13th International Conference on Machine Learning*. Morgen Kaufmann.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*, 77–95.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimisation. *IEEE Trans. on Evolutionary Computation*, *1*, 67–82.

Zhang, K., Wang, J. T. L., & Shasha, D. (1995). On the editing distance between undirected acyclic graphs and related problems. *Proc. of the 6th Annual Symposium on Combinatorial Pattern Matching* (pp. 395–407). Springer, Berlin.