On Exploiting Hierarchical Label Structure with Pairwise Classifiers

Johannes Fürnkranz
Knowledge Engineering Group
TU Darmstadt
juffi@ke.tu-darmstadt.de

Jan Frederik Sima
Cognitive Systems Research Group
Universität Bremen
sima@informatik.uni-bremen.de

ABSTRACT

The goal of this work was to test whether the performance of a regular pairwise classifier can be improved when additional information about the hierarchical class structure is added to the training sets. Somewhat surprisingly, the additional information seems to hurt the performance. We explain this with the fact that the structure of the class hierarchy is not reflected in the distribution of the instances.

1. INTRODUCTION

The pairwise approach, which learns one classifier for each pair of classes and aggregates the results by voting, has shown a good performance in various learning scenarios, including classification [5] and multi-label classification [7; 14]. It is an interesting question whether additional information on the structure of the output space can be used for an improved performance.

In this paper, we report on an experiment that aimed at improving pairwise classification in the presence of hierarchical class structures. The key idea is to augment the training sets for the binary base classifiers with additional examples that utilize the hierarchical structure of the class labels: each binary classifier $\mathcal{M}_{ij}$ discriminating between classes λ_i and λ_j is given additional training examples: examples of classes that are closer to λ_i in the training data are added to the examples for class λ_i, and examples of classes that are closer to λ_j are added to λ_j.

We start with a brief recapitulation of pairwise classification (Section 2) and hierarchical classification (Section 3).

2. PAIRWISE CLASSIFICATION

Pairwise Classification is a method to solve multi-class classification problems by dividing them into several binary problems. These 2-class problems will then be solved independently of each other using a binary base classifier [4]. Contrary to the conventional one-against-all or one-vs-rest approach, the pairwise classifier trains one classifier $\mathcal{M}_{ij}$ for each pair of classes (λ_i, λ_j). This classifier is trained on all examples from these two classes; all other examples are ignored. Thus, for c classes, one has to train $c \cdot (c-1)/2$ binary classifiers. By aggregating the predictions of all base classifiers with voting, one can eventually obtain a prediction for the actual multi-class task. Despite the quadratic number of classifiers, it has been shown that training is even faster than in the one-against-all approach [5], and that classification can be sped up to almost linear in the number of classes [17], making the pairwise approach competitive in terms of efficiency and superior in terms of accuracy.

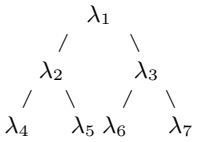

Figure 1: Hierarchical structure corresponding to the following partial order on the class labels $\mathcal{L} = \{\lambda_1, \ldots \lambda_7\}$:
$\lambda_1 \sqsupset \lambda_2$, $\lambda_1 \sqsupset \lambda_3$, $\lambda_2 \sqsupset \lambda_4$, $\lambda_2 \sqsupset \lambda_5$, $\lambda_3 \sqsupset \lambda_6$, $\lambda_3 \sqsupset \lambda_7$.

3. HIERARCHICAL CLASSIFICATION

In hierarchical classification, the set of labels $\mathcal{L} = \{\lambda_1, \ldots \lambda_m\}$ is structured with a partial order relation $\sqsupset$, which imposes a tree structure upon the label set, as shown in in Figure 1. This distinguishes hierarchical from conventional classification, where $\mathcal{L}$ is an unordered set. Many real-world classification problems, in particular text classification problems such as the REUTERS benchmark datasets [11; 12] or classification of Web catalogues [13], exhibit such a hierarchical structure in the labels. Several techniques have been proposed to exploit such a structure [9; 15; 1; 18].

A simple approach to hierarchical classification has become known as the *Pachinko machine* classifier [9]. Its key idea is to associate one classifier with each interior node of the label hierarchy tree. Its task is to decide which path will be followed. For example, in Figure 1, one classifier $\mathcal{M}_1$ is trained to discriminate labels λ_4 and λ_5 from labels λ_6 and λ_7. Depending on the outcome of the prediction, either classifier $\mathcal{M}_2$ decides between labels λ_4 and λ_5 or another classifier $\mathcal{M}_3$ decides between λ_6 and λ_7.

Strictly speaking, there are two different scenarios for hierarchical classification. In the first, only the leaves of the hierarchy (labels λ_4, λ_5, λ_6, and λ_7) can be predicted, in the second the examples can be labeled with all nodes in the hierarchy. As described in the next section, both cases can be tackled with pairwise classification. In the first we only need to train a pairwise classifier for the subset of leaf labels, in the second with all labels.

4. HIERARCHICAL PAIRWISE CLASSIFIER

In many applications of hierarchical classification, the class hierarchy corresponds to an ISA-hierarchy, where higher nodes are superconcepts of the nodes in their sub-trees. A natural assumption to be made in such a case is that the distance of classes within the class tree corresponds to the actual distances between their examples in the training set. We call this the *class fidelity assumption*. If, e.g., we have a topic hierarchy with the concepts *Politics*, *Economy*, *Sports*, etc., it is natural to assume that the subconcepts of the node *Sports* (such as *Basketball*, *Baseball*, *Football*) are closer to each other than to subconcepts of *Politics* or *Economy*.

The key idea of the proposed *augmented pairwise classifier* (APC) is to enforce the assumed class fidelity by using additional training examples of similar classes for the training of each classifier. For example, in Figure 1, we would add the λ_4 examples to the λ_5 examples when training the classifier $\mathcal{M}_{5,6}$. Our expectation is that this approach will outperform a "flat" classifier which simply ignores the hierarchical relationships between the classes because the additional training examples will improve the predictions of the pairwise classifiers on examples of other classes.

To formalize this process, we first have to define a notion of (semantic) closeness relation within the hierarchy.

DEFINITION 1 (MOST SPECIFIC SUPER-CONCEPT). *$\lambda_{mssc} = mssc(\lambda_i, \lambda_j)$ is the* most specific super-concept *of two classes λ_i and λ_j iff*

1. $\lambda_{mssc} \sqsupset \lambda_i \wedge \lambda_{mssc} \sqsupset \lambda_j$
2. $\nexists \lambda$ *s.t.* $(\lambda \sqsupset \lambda_i \wedge \lambda \sqsupset \lambda_j) \wedge \lambda_{mssc} \sqsupset \lambda$

Based on this, we defined the similarity or *closeness* between two nodes in the hierarchy as the depth of the most specific super-concept. In Figure 1, λ_4 and λ_5 have a closeness of 1, whereas λ_4 and λ_6 have closeness of 0.

The key idea of our approach is to train the binary classifier $\mathcal{M}_{ij}$ by adding all examples that are closer to λ_i to this class, and adding all examples that are closer to λ_j to that class.

DEFINITION 2 (AUGMENTED PAIRWISE CLASSIFIER). *The* augmented pairwise classifier *(APC) consists of one classifier $\mathcal{M}_{ij}$ for each pair of labels, which is trained on the examples of the following sets of positive ($\mathcal{P}_{ij}$) and negative ($\mathcal{N}_{ij}$) labels:*

$$\mathcal{P}_{ij} = \{\lambda \in \mathcal{L} | mssc(\lambda, \lambda_j) \sqsupset mssc(\lambda, \lambda_i)\}$$
$$\mathcal{N}_{ij} = \{\lambda \in \mathcal{L} | mssc(\lambda, \lambda_j) \sqsubset mssc(\lambda, \lambda_i)\}$$

For example, for training the classifier $\mathcal{M}_{56}$ with the class structure of Figure 1, the examples labeled as λ_4 are added to those with label λ_5, and the examples labeled as λ_7 are added to λ_6. If the prediction task includes the interior nodes of the hierarchy, label λ_2 will also be added to λ_4 and λ_5, and label λ_3 to λ_6 and λ_7. Examples with label λ_1, which has the same $mssc$ (itself) for both λ_5 and λ_6, are ignored. Note that $\mathcal{P}_{ij}$ trivially includes λ_i and $\mathcal{N}_{ij}$ trivially includes λ_j, i.e., the training sets of the augmented pairwise classifier are super-sets of the regular pairwise classifier.

Many hierarchical classification problems are also multi-label, i.e., each example may be associated with more than one label [19]. Pairwise classification can be easily extended to multi-label classification. In this case, the binary model $\mathcal{M}_{ij}$ is trained on all examples $\vec{x}$ for which one of the two labels λ_i and λ_j is associated with $\vec{x}$ and the other is not. For a new example, we can then predict a ranking of all classes, just as with single-label pairwise classification [14]. The top-portion of the ranking can then be predicted as a multi-label set for this example. For establishing the split-point in the ranking, separate techniques have to be used. Alternatively, the calibrated label ranking algorithm tightly integrates ranking and splitting [7]. In this paper, we will ignore this aspect and only compute a ranking.

For adding hierarchical information to the multi-label pairwise classifier, we adopt the approach of the previous section. In this case, the identification of additional training examples becomes a bit more complicated, because one of the multiple labels of the training example might be closer to λ_i than to class λ_j but at the same time some other label from the same example might be closer to λ_j than to λ_i. Thus, an example is added to the class λ_i if at least one of its classes is closer to λ_i than to λ_j, and no other class is closer to λ_j than to λ_i.

5. EQUIVALENCE TO PACHINKO MACHINE CLASSIFIER

After first experiments with a preliminary implementation it turned out that the predictions of the augmented pairwise classifier are the same as the predictions of the Pachinko machine classifier [9], which we briefly described in Section 3. This came unexpected to us, because while both methods, the Pachinko classifier and the augmented pairwise classifier reduce the hierarchical classification problem to an ensemble of binary classifiers, the Pachinko machine classifier uses fewer binary classifiers, which are arranged in a hierarchy, whereas the pairwise classifier uses one classifier for each pair of labels, each contributing one vote to the final prediction.

Upon closer inspection, it turns out that the two classifiers are, in fact, equivalent. The reason lies in the fact that the augmentation process described in Section 4 makes many of the pairwise classifiers equivalent.

LEMMA 1. *In the augmented pairwise classifier, the binary classifiers $\mathcal{M}_{ij}$ and $\mathcal{M}_{kl}$ receive the same training examples if $\lambda_{ik} \leftarrow mssc(\lambda_i, \lambda_k)$ and $\lambda_{jl} \leftarrow mssc(\lambda_j, \lambda_l)$ are in different subtrees, i.e., $\lambda_{ik} \not\sqsupset \lambda_{jl}$ and $\lambda_{jl} \not\sqsupset \lambda_{ik}$.*

PROOF. The node $\lambda = mssc(\lambda_{jl}, \lambda_{ik})$ separates the two subtrees S_{ik} rooted in λ_{ik} and S_{jl} rooted in λ_{jl}. From $\lambda_{ik} \not\sqsupset \lambda_{jl}$ and $\lambda_{jl} \not\sqsupset \lambda_{ik}$ it follows that $\lambda \neq \lambda_{jl}$ and $\lambda \neq \lambda ik$. Thus, all paths going to the labels in S_{ik} and S_{jl} share the same sub-path up to node λ, and differ from then on (otherwise λ would not be the $mssc$). Similarly, λ_i and λ_k share the same path down to λ_{ik}, which contains the path to λ as a (proper) sub-path. Therefore, $mssc(\lambda_k, \lambda_j) \sqsupset mssc(\lambda_k, \lambda_i)$ must hold and λ_k must be associated with $\mathcal{P}_{ij}$ according to Definition 2. With an analogous argument we can show that λ_l must be in $\mathcal{N}_{ij}$. Thus, $\mathcal{P}_{kl} \subseteq \mathcal{P}_{ij}$ and $\mathcal{N}_{kl} \subseteq \mathcal{N}_{ij}$. By the symmetry of the arguments, it follows that $\mathcal{P}_{kl} = \mathcal{P}_{ij}$ and $\mathcal{N}_{kl} = \mathcal{N}_{ij}$. □

LEMMA 2. *Each binary classifier $\mathcal{M}_{ab}$, which is trained to discriminate between two successor branches S_a and S_b of a node λ, corresponds to a binary classifier of the augmented pairwise classifier, and vice versa.*

PROOF. By the previous lemma, all classifiers $\mathcal{M}_{ij}$ and $\mathcal{M}_{kl}$ for $\lambda_i, \lambda_k \in S_a$ and $\lambda_j, \lambda_j \in S_b$ are identical to each other. Thus, $\mathcal{P}(\mathcal{M}_{ij}) = \mathcal{P}(\mathcal{M}_{kl}) = \bigcup_{\lambda \in S_a} \lambda$ and $\mathcal{N}(\mathcal{M}_{ij}) = \mathcal{N}(\mathcal{M}_{kl}) = \bigcup_{\lambda \in S_b} \lambda$. These are the positive and negative training sets of the Pachinko classifier at node λ.

Conversely, each binary classifier $\mathcal{M}_{ij}$ of the APC must correspond to the binary classifier that discriminates between the corresponding two successor branches of the node $\lambda = mssc(\lambda_i, \lambda_j)$. □

By the previous lemma, we already know that all binary classifiers of the augmented pairwise classifier correspond to a classifier that discriminates between two successor branches of an interior node λ. If the class structure is binary, i.e., each interior node has only two successors, this is the classifier trained by the Pachinko machine.

What remains to be shown is that the voting strategy of the augmented pairwise classifier leads to the same class label as the hierarchical path expansion of the Pachinko machine.

THEOREM 1. *For binary class hierarchies, the augmented pairwise classifier is equivalent to the Pachinko machine.*

PROOF. Each interior node λ of the binary class structure corresponds to one binary classifier $\mathcal{M}_{ab}$ of the Pachinko machine

classifier. This classifier is identical to all pairwise classifiers $\mathcal{M}_{ij}$ with $\lambda_i \in S_a$ and $\lambda_j \in S_b$. If S_a contains a nodes and S_b contains b nodes, we have $a \cdot b$ such identical binary classifiers, which all vote in the same way. Assume that $\mathcal{M}_{ab}$ selects branch S_a. Then, all of the above-mentioned $a \cdot b$ binary classifiers will vote for the class in S_a, i.e., each class in this branch will receive b votes from these classifiers. On the other hand, each class in S_b will receive 0 votes from these classifiers. Thus, classes in S_b can only receive votes from the comparisons among themselves, i.e., each class in S_b can receive at most $b-1$ votes. Thus, all classes in S_a will be ranked above all classes in S_b, which corresponds to the decision that is taken by the binary classifier $\mathcal{M}_{ab}$. □

For general multi-class class hierarchies, the situation is a bit more complex. Assume that node λ has successor subtrees $\{S_{a_i}\}, i = 1 \ldots s$, each branch having a_i nodes. Here, we need a multi-class classifier to decide which branch to follow. If this multi-class classifier is realized with a pairwise classifier, then the equivalence still holds if the selected subtree S_{a_i} is predicted by all pairwise models $\mathcal{M}_{a_i a_j}$ that compare S_{a_i} with some other branch S_{a_j}. If one such model $\mathcal{M}_{a_i a_j}$ makes an inconsistent prediction for the subtree S_{a_j}, i.e., if it predicts S_{a_j} even though the final selection of the pairwise classifier is S_{a_i}, then the hierarchical pairwise classifier may make a different selection (S_{a_j}) if $a_j \gg a_i$. Thus, in case of unbalanced class hierarchies, the hierarchical pairwise classifier may exhibit a bias towards larger subtrees if the binary classifiers do not make consistent predictions.

It should be noted that this result only holds for hierarchical singe-label classification. As discussed in Section 4, the technique can be straight-forwardly extended to multi-label classification. In this case, the equivalence between hierarchical pairwise classifiers and the Pachinko machine classifier no longer holds. In fact, an extension of the Pachinko machine to hierarchical classification is not obvious, so that one interpretation of the above result could be that the augmented pairwise classifier is a generalization of the Pachinko machine classifier to multilabel problems.

6. RESULTS ON THE REUTERS DATASET

In order to evaluate the augmented pairwise classifier, we performed experiments on the REUTERS RCV1 corpus [12]. We emphasize that we were not so much interested in the absolute performance of the method, but only focused on the comparison between regular pairwise classification and augmented pairwise classification. Our expectation was that additional knowledge about the class hierarchy should be able to improve the performance of the pairwise classifier, and we wanted to verify this hypothesis. All experiments were conducted in Weka, using its support vector machine SMO with default parameters as the base classifier.

We used a version of the REUTERS RCV1 corpus which consists of five datasets, each containing 3000 training and 3000 test examples.[1] Each example is encoded with about 40,000 attributes representing the TF-IDF values of the words in the text. From these, we performed a feature selection based on document frequency on the training sets, i.e., for each of the five datasets we only kept the 5000 features which had the highest number of non-zero values in the training set. This performed quite well in the experiments reported in [20]. The dataset is a multi-label dataset, where each example is on average assigned to four label of a total of 101 labels. The class hierarchy is up to 4 levels deep. 23 of the 101 labels correspond to interior nodes in the hierarchy.

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multilabel.html

Table 1: Ranking Error on the REUTERS RCV1 dataset

Dataset	1	2	3	4	5
pairwise classification (PC)					
Precision@4	**0.58**	**0.61**	**0.59**	**0.58**	**0.62**
Recall@4	**0.56**	**0.60**	**0.60**	**0.58**	**0.60**
F1@4	**0.58**	**0.60**	**0.60**	**0.58**	**0.61**
Hamming@4	**0.03**	0.03	0.03	**0.03**	0.03
Margin Loss	**10.30**	**10.28**	**10.57**	**10.97**	**10.48**
One Error	**0.17**	**0.13**	**0.11**	**0.18**	**0.12**
Rank Loss	**0.04**	**0.04**	**0.04**	**0.04**	**0.04**
Avg Precision	**0.73**	**0.75**	**0.75**	**0.73**	**0.76**
First Irrelevant	**2.90**	3.10	3.03	2.89	3.07
augmented pairwise classification (APC)					
Precision@4	0.49	0.57	0.56	0.56	0.59
Recall@4	0.49	0.57	0.57	0.56	0.57
F1@4	0.49	0.57	0.56	0.56	0.58
Hamming@4	0.04	0.03	0.03	0.04	0.03
Margin Loss	24.62	16.50	15.15	18.62	14.63
One Error	0.27	0.17	0.14	0.22	0.17
Rank Loss	0.14	0.08	0.08	0.10	0.07
Avg Precision	0.66	0.73	0.73	0.72	0.75
First Irrelevant	2.83	**3.16**	**3.16**	**3.10**	**3.22**

Table 1 shows the results of pairwise classification (top) and augmented pairwise classification (bottom) in terms of nine evaluation measures. The first four measures assume that first four labels of the ranking are relevant and compute the precision, recall and F1-measures as well as the error (Hamming loss) on the predicted labels. For example, a precision value of 0.6 means that 60% of the predicted labels were actually relevant, a recall value of 0.6 means that 60% of the relevant labels were actually predicted. The remaining five values try to capture the quality of the ranking: *margin loss* is the difference in the ranking position of the first irrelevant labels and the last relevant label, *one error* is the percentage of test instances where the top rank is not a relevant class, *rank loss* is the fraction of label pairs for which the irrelevant label is ranked before the relevant (an adaptation of Kendall's tau for multi-label problems), *average precision* is the averaged precision values at the position of each relevant label, and *first irrelevant* is the ranking position of the first irrelevant label. All reported values are averaged over all test instances.

The results show that according to all but one measures the augmented pairwise classifier does not improve over the regular pairwise classifier (better results are shown in **bold**). Particularly striking is the large difference in margin loss, i.e., the position of the last relevant label is typically much lower for the APC. Interestingly, the APC seems to have a slight advantage in terms of the position of the first irrelevant label. However, this difference is not significant and does not change the overall result that the augmented pairwise classifier did not improve over the pairwise classifier, but, in fact, seems to perform somewhat worse.

7. RESULTS ON ARTIFICIAL DATA

The negative result on the REUTERS data came somewhat surprising and asked for an explanation. Possible explanations are:

1. Augmenting the hierarchical classifiers with additional examples makes the decision surface more complex and thus harder to learn
2. Datasets violate the class fidelity assumption, upon which the idea of the augmented pairwise classifier is based

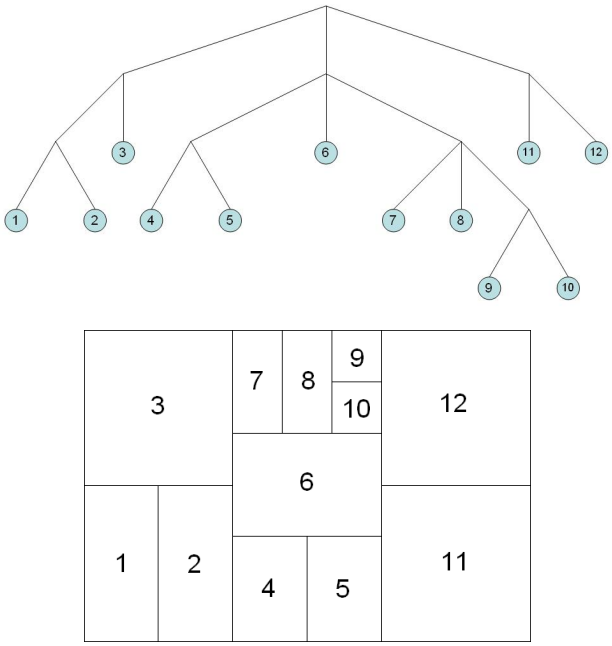

Figure 2: Class hierarchy (top) and spatial layout (bottom) of the artificial dataset

.

To test these two assumptions, we generated an artificial, single-label dataset with 12 classes organized in the hierarchical structure that is shown in the upper part of Figure 2. The examples were assigned labels roughly following the spatial layout shown in the lower part of Figure 2. For each class, we generated 100 training examples using a 2-dimensional Gaussian distribution with the mean in the center of the rectangle and the standard deviations proportional to the side lengths of the enclosing rectangles. We tried five different settings (numbered from 1 to 5), corresponding to $1/3$, $1/2$, 1, $3/2$, 2 times the breadth and width of the rectangle.

The motivation for generating the data in this way was primarily that we wanted to be sure that the hierarchical class structure is reflected in the instance space. This property is mostly true, but one can also find exceptions. For example, the center of class 3 is closer to the center of class 7 than to the center of class 1. Moreover, the classes in each internal node of the hierarchy should be linearly separable. This is the case for the lowest variance level, but with increasing variances this property will be weakened.

Finally, as an additional test for the influence of the class fidelity assumption, we also generated a version of this dataset in which the classes 3 and 10 were swapped in instance space, resulting in a dataset that clearly violates the above assumption.

Figure 3 shows the results of PC and APC over increasing variance around the class centers, on both versions of the dataset. First, we can see that the pairwise classifier dominates the augmented pairwise classifier in both scenarios. Moreover, there is no noticeable difference in performance between the normal dataset and the one with swapped classes. This is not surprising, as PC does not explicitly make use of the class hierarchy. On the other hand, for low variance levels, the performance of APC clearly depends on the class fidelity assumption: APC's performance is en par with PC's for the case where the hierarchy is reflected in instance space, but it is much worse in the case where this assumption is violated.

Both, PC and APC suffer when the variance of the data around the

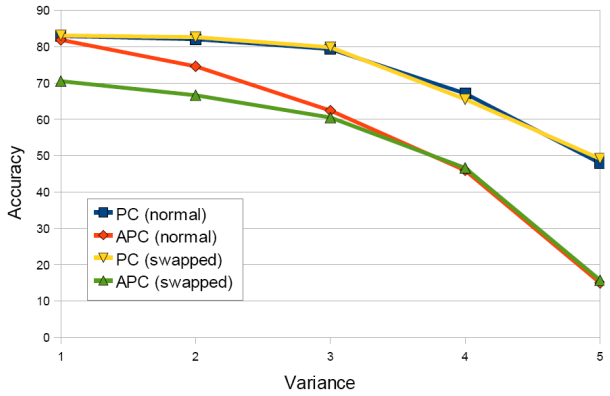

Figure 3: Results on artificial data

center increases. Again, this is not surprising because the problem becomes less and less linearly separable and thus harder to solve. However, it seems that the advantage of PC over APC increases with increasing variance. This seems to indicate that the unaugmented binary classifiers are easier to train, in particular when the augmentation does not respect the class fidelity.

Based on the above observations, we want to verify the class fidelity of the REUTERS dataset. To this end, we tried to measure class fidelity in the following way: If we train a binary classifier $\mathcal{M}_{ij}$ for discriminating classes λ_i and λ_j (without seeing examples of any of the other classes), then classes that are closer to λ_j than to λ_i should be more likely to be classified as λ_j than as λ_i. We used this for computing a *class fidelity index* which is simply the fraction of all examples that are assigned according to the expectation of the hierarchy, averaged over all binary classifiers of the PC (examples where the label is equally likely for both sides are ignored). For multi-label data, we use the same extension as defined in Section 4, namely that an example is assumed to be closer to class λ_i if at least one of its labels is closer to λ_i than to λ_j, and no other class is closer to λ_j than to λ_i.

Table 2 shows the class fidelities for all datasets. We can see that even for the artificial data, the class fidelity is not perfect, because of the minor violations in class fidelity discussed above. However, clearly, the index is worse for the dataset where two labels were swapped in instance space. Also, the class fidelity is clearly decreasing with increasing variance around the class centers. The results for REUTERS, although maybe not directly comparable because this is a multi-label dataset, show an even worse class fidelity index. Note that the expected value for the index for the case when there is no correlation between class hierarchy and location in instance space would be 0.5. Thus, it is safe to conclude that REUTERS does not exhibit the class fidelity upon which the design of the augmented pairwise classifier was based.

Table 2: Class fidelity index for all datasets.

	1	2	3	4	5
dataset	variance level				
normal	0.775	0.746	0.735	0.709	0.649
swapped	0.739	0.713	0.700	0.674	0.616
	fold #				
REUTERS	0.63	0.64	0.64	0.63	0.63

8. DISCUSSION

We think that our primarily negative results are not limited to hierarchical classification, but apply to any attempt to exploiting an order relation on the label structure by enriching the training examples in the way outlined in this paper. For example, in ordinal classification, also called ordinal regression in statistics, the set of class labels $\mathcal{L} = \{\lambda_1, \lambda_2 \dots \lambda_m\}$ is endowed with a natural (total) order relation $\lambda_1 \sqsupset \lambda_2 \sqsupset \dots \sqsupset \lambda_m$. From a learning point of view, the ordinal structure of $\mathcal{L}$ is *additional* information that a learner should try to exploit, and this is what existing methods for ordinal classification essentially seek to do [10; 3; 2]. On the other hand, pairwise classification has been previously shown to work quite well on this problem, even though it disregards this information entirely [6].

Obviously, the order information can be exploited in the same way as sketched above for hierarchical classification: for training the classifier $\mathcal{M}_{ij}$, the examples of class λ_i are enriched with the examples of all classes λ_k for $k < i$, and the examples of class λ_j are enriched with the examples of all classes $\lambda_l, l > j$ because of $\lambda_k \sqsupset \lambda_i \sqsupset \lambda_j \sqsupset \lambda_l$. This approach was tried in [16], but, just as the results reported here, did not yield any improvements over regular pairwise classification. This is consistent with the observation of [8] that the ordering information is not as strongly reflected in the training data as one might expect.

Recently, [21] have also observed that approaches that attempt to exploit the hierarchical class structure of a problem do not improve over approaches that ignore this structure. It remains to be seen whether these results can also be explained with a lack of class fidelity.

9. CONCLUSIONS

The negative result reported in this paper, namely that the augmentation of pairwise classifiers with additional training examples does not improve classification performance, has lead to two interesting insights. First, we have shown that the method is essentially equivalent to a Pachinko-machine classifier, but can be straight-forwardly generalized to multilabel data. Second, we have seen that a key assumption behind the augmentation strategy, namely that examples of classes that are near-by in the class hierarchy are also close in instance space, does not always hold.

Acknowledgments: This research was supported by the German Science Foundation (DFG). We would like to thank Eyke Hüllermeier for inspiring discussions on this subject.

10. REFERENCES

[1] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM-04)*, pp. 78–87, Washington, DC, 2004.

[2] J. S. Cardoso and J. F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8:1393–1429, 2007.

[3] E. Frank and M. Hall. A simple approach to ordinal classification. In L. D. Raedt and P. Flach (eds.) *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*, pp. 145–156, Freiburg, Germany, 2001. Springer-Verlag.

[4] J. H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, Stanford, CA, 1996.

[5] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[6] J. Fürnkranz. Round robin ensembles. *Intelligent Data Analysis*, 7(5):385–404, 2003.

[7] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, June 2008.

[8] J. C. Hühn and E. Hüllermeier. Is an ordinal class structure useful in classifier learning? *International Journal on Data Mining, Modelling and Management*, 1(1):45–67, 2008.

[9] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pp. 170–178, Nashville, 1997.

[10] S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, XXI:1001–1013, 2001.

[11] D. D. Lewis. Reuters-21578 text categorization test collection. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, 1997.

[12] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[13] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.

[14] E. Loza Mencía and J. Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*, pp. 2900–2907, Hong Kong, 2008. IEEE.

[15] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pp. 359–367, 1998.

[16] G. H. Nam. Ordered pairwise classification. Master's thesis, TU Darmstadt, Knowledge Engineering Group, 2007.

[17] S.-H. Park and J. Fürnkranz. Efficient pairwise classification. In *Proceedings of 18th European Conference on Machine Learning (ECML-07)*, pp. 658–665, Warsaw, Poland, 2007. Springer-Verlag.

[18] J. Rousu, C. Saunders, S. Szedmák, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, July 2006.

[19] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[20] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In D. Fisher (ed.) *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pp. 412–420, Nashville, TN, 1997.

[21] A. Zimek, F. Buchwald, E. Frank, and S. Kramer. A study of hierarchical and flat classification of proteins. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7:563–571, 2010.