Committee-Based Selective Sampling with Parameters Set by Meta-Learning

Miloslav Nepil and Luboš Popelínský

KD Group at Faculty of Informatics
Masaryk University in Brno, Czech Republic
{nepil,popel}@fi.muni.cz

Abstract. This contribution presents a parametric variant of committee-based selective sampling. The committee members learned on small subsets obtained by random sampling from the original dataset are used to classify the rest of the dataset. Those examples on which the committee came to consensus are considered to be easy, the others to be hard. The main idea is to select the resulting training subset with a different ratio of easy to hard examples. In the second part of the paper meta-learning technique for parameter setting is introduced and experimental results obtained with it are discussed. This selective sampling method has been proven useful in reducing the learning time while keeping the accuracy at a better level than random selection does. The meta-learning method for parameter settings displays fairly low ranking error and is sufficient for a reliable and immediate prediction of parameters.

1 Introduction

Instance selection methods are aimed at finding a representative subset of training data which would be smaller than the original dataset, but still would provide enough information to achieve an accurate model. Three main motivations can be found for reducing the number of training examples. The first reason could be that a sufficient amount of labeled examples is difficult to obtain; a problem frequently faced in natural language processing. If we cannot rely on unsupervised learning and examples should be annotated manually by a human, then we need to save the annotation costs [1, 2].

The second, quite an opposite situation arises when we have a cheap access to a large, or even potentially unlimited amount of training data. It happens when our data mining task can be solved with an unsupervised or implicitly supervised approach [3]. But still, we need to select a finite training subset of reasonable size since we are always limited in learning time.

And the third convenience for a selection of training examples comes up when the learning on a whole dataset leads to a huge model. This can be caused either by presence of noisy examples or outliers in the training data [4], or by an inherent property of the given learning algorithm. Instance-based learning algorithms are clearly the case, but also for several other algorithms (including tree construction ones such as C4.5, and rule construction ones such as C4.5rules) it has been observed [5] that increasing the amount of data used to build a model often results in a linear increase in model size, although that additional complexity results in no significant increase in model accuracy. Therefore, a selection of training data could help us to make the model more compact and concise.

This work was motivated by the fact that for large datasets which are being treated in data mining the experiments took too much time. Therefore, our primary goal was to decrease the learning time (and perhaps the model size) while keeping the error rate as low as possible; a goal perfectly addressed by selective sampling. Selective sampling proceeds, in general, by measuring the information content of each training example. The objective is to select those examples which could provide the most informative description of a target concept being learned. The measure of information content can be either uncertainty-based [6] or committee-based [7]. Approaches based on uncertainty often derive an explicit measure of the expected information gained by using the example. However, the main drawback of these approaches is that they are usually dependent on a particular learning algorithm. Since we have been concerned with a simple selective sampling technique which could be easily applied to many different learning algorithms, we gave precedence to the committee-based approach.

The structure of this paper is as follows. In Section 2 we first explain the main idea of our variant of selective sampling and then describe the algorithm. Discussion on speed up of this way of sampling follows. Section 3 brings experimental verification of usefulness of this selective sampling method. The second part of this paper concerns settings of parameters of the method. In Section 4 we describe the meta–learning method used. Section 5 displays the results obtained by meta-learning.

2 Committee-Based Selective Sampling

2.1 General scheme

The general scheme of our variant of selective sampling driven by committee of classifiers is as follows. In the beginning, we learn a set of several fast classifiers – members of the committee. Then, we let the committee make a decision about each given training example, which means that each member has to classify the example according to its own knowledge about the target concept. Thus, we get several (possibly different) class predictions for each example. Hence, the information content of the example is evaluated as a measure of disagreement among the committee members. For final training we select a subset of examples with highest information content.

If we want to devise a particular variant of committee-based selective sampling, several questions should be answered:

1. How many committee members do we need?
2. How to choose the committee members?
3. How to measure the disagreement among committee members?

4. How to select the resulting subset of training examples?

Our parametric variant of committee-based selective sampling adopts the following solution. It presumes that we have a fast (low complexity) learning algorithm $\mathcal{A}_{init}$ which we use for training initial classifiers (committee members) and a slow (but robust) learning algorithm $\mathcal{A}_{final}$ which we use for training the final classifier. Both training and prediction times of the initial classifiers are important, due to the fact that predictions on the whole dataset have to be obtained.[1] Our method treats the number of committee members as a fixed parameter N. The committee members are established by learning on small subsets obtained by random sampling from the original dataset. The size of these small subsets is given by another parameter I. Our measure of disagreement is rather rough, since we distinguish only two categories: a complete consensus and a dissension. Those examples on which the committee came to a consensus are considered to be *easy*, while the others are considered to be *hard*. The main idea of our method is to select the resulting training subset in such a way that the ratio of *easy* to *hard* examples in the resulting subset is computed as a function of the corresponding ratio which was observed in the initial dataset. As this function we simply took a multiplication by a coefficient X. The values $0 \leq X < 1$ mean that we want to decrease the ratio of easy examples in the final subset (actually, $X = 0$ implies no easy examples there). On the other hand, the values $X > 1$ mean that we intend to add even more easy examples to the final subset. Note that the value $X = 1$ results in no change of the easy/hard ratio, therefore this setting corresponds to random sampling. Another parameter F determines the size of the final training subset.

2.2 Algorithm

We can already see that our selective sampling technique is parameterised by four numerical values:

N – a number of initial classifiers (members of the committee)
I – a size of the initial training subset used for learning initial classifiers
F – a size of the final training subset used for learning a final classifier
X – a coefficient for modifying the original ratio of *easy* to *hard* examples

More formally, our example selection works as follows:

1. The number of committee members is given by a parameter N, $N \geq 2$.
2. From the given training set we draw randomly an initial subset of the relative size I, $0 < I < 1$, as a fraction of the original dataset. The initial subset is randomly split into N blocks and each block is used for training one initial classifier with a learning algorithm $\mathcal{A}_{init}$.

[1] However, this need not be the case. In Section 2.3 we describe an improvement of our basic method which estimates the size of a subset sufficient for submitting to the committee.

3. Each initial classifier is applied to the whole training set. Therefore, we obtain N class predictions for each example. Those examples which were classified consistently (it means that all N predictions were identical) are considered as *easy* ones while the others are considered as *hard* ones. Let's denote the ratio of *easy* to *hard* examples as e/h.
4. We select randomly a final training subset so that its ratio of *easy* to *hard* examples is given by the expression $X \cdot e/h$ where the coefficient X, $X \geq 0$, is another fixed parameter. The final subset's size is determined by a parameter F, $0 < F < 1$, as a fraction of the original dataset. The final subset is used for training a final classifier with a learning algorithm $\mathcal{A}_{\mathit{final}}$.

It is not difficult to guess that a particular setting of the parameters presented above has an important impact on performance of the method. The appropriate parameter setting is not a trivial task since it depends not only on properties of the dataset at hand, but also on our preferences with regard to the learning time, the precision of learned model, and the model size.

2.3 Speeding up the Sampling

In the previous description of our basic sampling algorithm we have stated – for the sake of simplicity – that the committee should give class predictions on the whole dataset. On the contrary to this, for the final subset we want to select only a certain (possibly small) amount of original training examples. Of course, it makes many computed predictions redundant and, as a consequence, it means that we would waste computational time for the sampling. Although the initial classifiers are assumed to be fast, they need some time to predict the target class. Considering that we want to sample from large datasets and the number of committee members can be higher as well, we have concerned us with the question if there is a possibility to estimate the size of a subset of original dataset which would be sufficient for subsequent processing.

Let s denotes the size of an original dataset. Then we know that the final subset must contain $F \cdot s$ examples.[2] It implies that the committee should give class predictions on $F \cdot s$ examples, at least. So, we run the committee on these $F \cdot s$ examples and find out that e_1 examples out of them are easy and h_1 are hard, $e_1 + h_1 = F \cdot s$. From the fourth step of the basic algorithm in Section 2.1 we already know that $e_2/h_2 = X \cdot e_1/h_1$ where e_2 and h_2 denote the required numbers of easy and hard examples in the final subset, respectively. But, at the same time, $e_2 + h_2 = F \cdot s$. It follows that

$$e_2 = \frac{X \cdot e_1 \cdot (e_1 + h_1)}{X \cdot e_1 + h_1}\,, \qquad h_2 = \frac{h_1 \cdot (e_1 + h_1)}{X \cdot e_1 + h_1}\,.$$

Now if $h_2 > h_1$ (which means $X < 1$) then we need to find some additional hard examples, otherwise, if $e_2 > e_1$ (which means $X > 1$) then we need to find some additional easy examples. Therefore, in the former case, it suffices to let the

[2] Rounding to whole numbers is omitted in this section.

committee judge $(h_2 - h_1) \cdot (e_1/h_1 + 1)$ additional examples to obtain $h_2 - h_1$ new hard examples, and, in the latter case, it suffices to judge $(e_2 - e_1) \cdot (h_1/e_1 + 1)$ additional examples to obtain $e_2 - e_1$ new easy examples. Of course, an important point here is that we assume the distribution of easy and hard examples to be the same on the whole original dataset.

This improvement of our basic method makes the sampling algorithm two--fold: at first, the committee is applied to $F \cdot s$ examples, and then it is run on an additional block of data, whose size is determined by the result of the first run. As an asset, the committee does not need to explore the whole given dataset, which significantly saves sampling time in many cases.

3 Experimental Results of Selective Sampling

Table 1. The comparison of results achieved on whole dataset (WD), by selective sampling (SS), and by random sampling (RS). The initial algorithm $\mathcal{A}_{init}$ was c50tree and the final algorithm $\mathcal{A}_{final}$ was c50boost. The parameters of selective sampling were set as follows: $N = 2$, $I = 0.2$, $F = 0.3$, and $X = 0.1$. The random sampling was set to select the same resulting fraction of data (30 %).

Dataset	Total Time (sec)			Model Size			Error Rate (%)		
	WD	SS	RS	WD	SS	RS	WD	SS	RS
adult	139.3	22.3	22.1	23768	5977	8751	14.49	14.40	15.33
letter	81.1	21.3	15.3	11691	6966	5484	4.69	7.85	9.73
optical	53.0	11.9	7.8	1771	1009	788	2.47	3.83	4.48
pendigits	32.0	8.8	6.2	1703	1122	904	1.15	1.49	2.27
quisclas	19.2	8.3	6.9	5447	1829	1713	35.24	36.77	36.04
satimage	51.9	12.5	8.5	2652	1346	959	9.54	9.88	11.31

Table 2. The similar experiment as above, but for the final algorithm c50rules. The parameters of selective sampling were set here as follows: $N = 2$, $I = 0.1$, $F = 0.3$, and $X = 0.2$. The random sampling was set again to select the same resulting fraction of data (30 %).

Dataset	Total Time (sec)			Model Size			Error Rate (%)		
	WD	SS	RS	WD	SS	RS	WD	SS	RS
adult	89.3	16.2	12.8	327	218	215	13.67	14.34	14.80
letter	231.5	29.1	20.3	1177	721	548	11.03	18.37	19.56
optical	16.0	3.3	1.6	209	108	95	8.64	11.17	13.49
pendigits	18.1	3.6	2.1	188	121	106	3.22	4.80	5.99
quisclas	16.9	3.5	2.8	475	179	180	35.34	37.48	37.16
satimage	24.0	3.9	1.9	281	138	108	13.36	14.65	15.77

At first, we shall show that the selective sampling method really selects representative subsets of the training data. A better quality of the dataset obtained by selective sampling displays a better accuracy of the learned model when compared to the random sampling. In our experiments we tried a family of C5.0 [8] algorithms.[3] While c50tree has been used as an initial learner, we have used c50boost and c50rules as final learners: corresponding results on several datasets explored inside the MetaL project[4] are shown in Tables 1 and 2, respectively. These tables show results concerning the total time, the size of learned model, and its error rate on a test set. All numbers were computed through 10-fold cross-validation. We can see that the error rate achieved by selective sampling remains in many cases close to the original error rate. For adult dataset it even decreased, when c50boost has been used as a final learner. Furthermore, selective sampling is always better than random sampling in terms of accuracy, except for quisclas dataset (which has an excessive error rate on the whole dataset, either). However, it should be noted that this singularity of quisclas dataset does not mean that the selective sampling is not useful for it at all. If we use c50boost as the final learner and choose a different setting, namely $N = 4$, $I = 0.3$, $F = 0.3$, and $X = 0.3$, then we get the following results: Total Time 9.1, Model Size 1831, and Error Rate 35.58. Thus, the accuracy of selective sampling is better than of random sampling for quisclas as well, but we must hit the appropriate parameter setting.

As for the reduction of model size, often the selective sampling is almost as successful as the random sampling. For adult dataset with c50boost as a final learner and quisclas dataset with c50rules as a final learner, the selective sampling produced even smaller model than random sampling did.

Nevertheless, the most significant is the reduction of total time. The total time comprises the time taken by sampling, training and testing together. It means that in the case of selective sampling the total time subsumes also the time spent on learning and application of initial classifiers. Consequently, the random sampling is a bit faster, but the extra time spent on selective sampling seems to be really useful, considering the better preserved accuracy. The main asset of time reduction does not rest in the fact that we are able to shrink the total time from 51.9 to 12.5 seconds, but the important thing is that 5:1 -‑time reduction with no considerable decrease in accuracy can help the learning algorithm to scale up to significantly larger datasets.

The results and discussion presented above concern the settings $X < 1$, when hard examples are being added to the final subset. We have also tried the settings $X > 1$, which means to add easy examples. These settings resulted in a greater reduction of the total time as well as the model size, however, the accuracy was worse than that of random sampling.

4 Meta-Learning for Parameter Setting

4.1 Ranking Function

As we could see earlier, particularly in the discussion about "abnormal" quisclas dataset, the performance of our selective sampling method strongly depends on

[3] `http://www.rulequest.com`
[4] `http://www.metal-kdd.org`

the setting of its parameters. Table 3 demonstrates the impact of parameter X (the coefficient for modifying the original ratio of *easy* to *hard* examples) on the performance criteria. It is not surprising that the demand on a fast processing and small model goes against the demand on a high accuracy.

Table 3. The impact of parameter X on the resulting time, model size and error rate, shown on satimage dataset with c50tree as an initial learner and c50boost as a final learner. The resting parameters are fixed to these values: $N = 2$, $I = 0.2$, and $F = 0.3$. The expression e_1/h_1 denotes the original (observed) ratio of *easy* to *hard* examples whereas the expression e_2/h_2 refers to the resulting (computed) ratio. The following time values are listed: T_1 – sampling time, T_2 – training time, T_3 – testing time, and T – total time. Selective sampling with the setting $X = 1.0$ corresponds to random sampling, therefore the sampling time is considered to be zero.

X	e_1/h_1	e_2/h_2	T_1	T_2	T_3	T	Size	Error
0.1	1389/348	495/1186	1.9	10.5	0.1	12.5	1346	9.88
0.2	1389/348	771/965	1.7	10.0	0.1	11.8	1259	10.46
0.3	1389/348	946/790	1.5	9.4	0.1	11.1	1176	10.57
0.4	1389/348	1067/668	1.4	9.0	0.1	10.5	1124	10.41
0.5	1389/348	1157/577	1.4	8.9	0.1	10.4	1094	10.69
0.6	1389/348	1225/505	1.3	8.3	0.1	9.7	1056	10.86
0.7	1389/348	1279/456	1.2	8.4	0.1	9.8	1025	11.31
0.8	1389/348	1322/410	1.2	8.1	0.1	9.4	1010	11.20
0.9	1389/348	1358/370	1.2	8.1	0.1	9.4	971	11.73
1.0	1389/348	1389/348	0.0	8.4	0.1	8.5	959	11.31

Therefore, it is clear that search for the best parameters needs to take into consideration not only the properties (the meta-characterisations) of the particular dataset, but also our preferences with regard to some performance criteria (time, accuracy, model size). This observation naturally leads to ranking techniques. Considering our experimental purposes we have resorted to very simple ranking function which takes into account just time and error rate and not model size:

$$R(K, T_s/T_w, E_s/E_w) = K \cdot (T_s/T_w) + (1 - K) \cdot (E_s/E_w)$$

Here T_s and T_w are the total times achieved by learning from a sample and by learning from a whole dataset, respectively. Similarly, E_s and E_w are the error rates achieved by learning from a sample and by learning from a whole dataset, respectively. And finally, K, $0 \leq K \leq 1$ is a balance parameter: $K = 0$ means that we are interested only in accuracy and, on the contrary, $K = 1$ means that we regard the total time only. We always want to minimise this ranking function's value.

4.2 Generation of Meta-Learning Data

As the meta-data characteristics of a dataset we have exploited a learning time T_p, a model size S_p, and an error rate E_p of a pilot classifier. The pilot classifier was a classifier attained by learning with the initial algorithm $\mathcal{A}_{init}$ on a random sample of a small, fixed size (10 %). The choice of these meta–attributes was motivated by the fact that these characteristics are cheaper than most of statistical and information-theory measures to obtain, and also by our belief that for our purpose they will serve comparably well.

Then, we have run the selective sampling on six datasets with various settings to find out corresponding values of total time T_s and error rate E_s. As for the tested settings, the values of their particular parameters were drawn from these enumerations: $N \in \{2,3,4\}$, $I \in \{0.1, 0.2, 0.3\}$, $F = 0.3$, and $X \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. All combinations of these values were tested. From all the obtained results we have compiled examples for learning a meta-model. Figure 4 shows an example from a training meta-dataset. Each training example for meta-learning consisted of attributes which could be divided into four groups: 1. data characteristics (pilot classifier results): T_p, S_p, E_p, 2. selective sampling parameters: N, I, F, X, 3. balance parameter K, and 4. the corresponding value of ranking function $R(K, T_s/T_w, E_s/E_w)$. The groups 1,2,3 represent independent (predictive) attributes, while the last attribute (group 4) is dependent (predicted).

Table 4. An example from a meta-dataset compiled from `adult` dataset. Using `c50tree` and `c50boost` as an initial and a final learner, respectively, for the setting $N = 2$, $I = 0.2$, $F = 0.3$, and $X = 0.1$ we got the total time $T_s = 22.3$ and error rate $E_s = 14.40\,\%$. Corresponding results on the whole dataset (without sampling) are $T_w = 139.3$ and $E_w = 14.49\,\%$. Therefore, the resulting value of ranking function is $R(0.1, 22.3/139.3, 14.40/14.49) = 0.911$.

group 1			group 2				group 3	group 4
T_p	S_p	E_p	N	I	F	X	K	R
0.56	72	0.152	2	0.2	0.3	0.1	0.1	0.911

4.3 Learning the Meta-Model

We do not aim at predicting the best parameter setting directly. Instead, a meta-model is designed to predict the value of ranking function. When processing an unseen data, the data characteristics (attributes from the group 1) and the balance parameter (group 3) are known and we need to find such selective sampling parameters (attributes from the group 2) which would minimise the ranking function value (output of the meta-model, group 4).

We decided to use regression trees as a meta-model for our purpose since in a regression tree it is easy to find values of unknown attributes which would minimise the output function. For learning the meta-model we utilised the system RT4.1 [9] which generates regression trees.[5] Figure 1 shows a part of the

[5] `http://www.liacc.up.pt/~ltorgo/RT`

regression tree which we have obtained. If the condition in a node holds than the left branch is chosen, otherwise the right one.

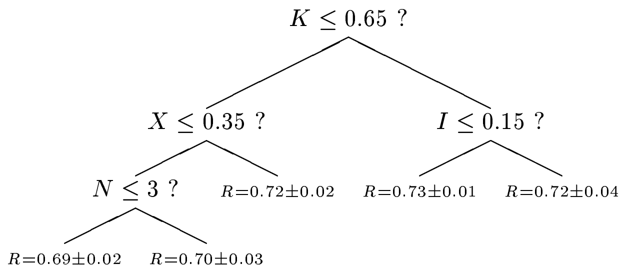

Fig. 1. Example of a regression tree used as a meta-model for parameter setting.

5 Experimental Results of Meta-Learning

Table 5 presents the results of our regression meta-model. We suppose that the most important performance measure is a Relative Increase in Ranking Value (RIRV). It is a comparison of ranking values of the predicted parameter setting and the best known parameter setting for a particular dataset and a learning algorithm. Hit Rate (HI) is a percentage of those cases when the best known parameter setting was also predicted. Relative Increase in Error Rate (RIER) is a comparison of error rates of a resulting (non-meta) classifier obtained from the predicted parameter setting and a resulting classifier obtained from the best known parameter setting. Finally, Relative Increase in Total Time (RITT) is a comparison of total time of a resulting (non-meta) classifier obtained for the predicted parameter setting and a resulting classifier obtained for the best parameter setting.

As we can see in Table 5 RITT is negative and RIER is positive. It means that our regression model tends to predict settings resulting in faster processing, but worse error rate. All numbers were computed from leave-one-out validation on six different datasets: adult, letter, optical, pendigits, quisclas, satimage. As we can see, the meta-attributes T_p, S_p, E_p made the parameter prediction surprisingly worse. We obtained more accurate meta-model (especially for c50boost) by not using those meta-attributes. This is probably due to the fact that the ranking function has a very similar curve for different datasets and thus the meta-characteristics do not bring additional information – they mislead the regression model instead. On the other hand, it should be noted that we have learned our regression model from relatively small amount of datasets. In fact, the training set in each fold of the leave-one-out validation consisted of five datasets. However, the presence of meta-attributes in the case of c50rules final algorithm led to a significant increase in hit rate, whereas the ranking deviation stayed at almost the same level. Therefore, we suppose exploitation of the meta-attributes to be promising.

Table 5. The results of meta-learning for selective sampling with c50tree as an initial algorithm and c50boost and c50rules as final algorithms.

Algorithm	Meta Att.	RIRV	HI	RITT	RIER
c50boost	absent	7.1257 %	27.3 %	-0.5651 %	9.7485 %
c50boost	present	9.9497 %	19.7 %	-6.9764 %	13.5287 %
c50rules	absent	5.9469 %	21.2 %	-3.1301 %	7.3770 %
c50rules	present	5.9957 %	30.3 %	-10.6430 %	8.0315 %

6 Conclusion

We have presented a new parametric variant of committee-based selective sampling and a meta-learning technique for setting its parameters. The selective sampling has been proven useful in reducing the learning time while keeping the accuracy at a better level than random selection does. The main contribution of the meta-learning is that its ranking error is fairly low (7.13% for c50boost and 5.95% for c50rules) which is sufficient for a reliable and immediate prediction of the right parameters setting for selective sampling.

Acknowledgement

Our thanks go to Pavel Brazdil and other members of LIACC group for their help and also to the anonymous referee for helpful comments. This research has been partially supported by Esprit LTR Project MetaL and by the Czech Ministry of Education under the grant JD MSM 14330003.

References

1. Dagan, I., Engelson, S.P.: Selective sampling in natural language learning. In: Proceedings of the Workshop on New Approaches to Learning for Natural Language Processing at IJCAI 1995, Montreal, Canada, Morgan Kaufmann (1995)
2. Thompson, C.A., Califf, M.E., Mooney, R.J.: Active learning for natural language parsing and information extraction. In: Proceedings of 16^{th} International Conference on Machine Learning, ICML 1999, Bled, Slovenia, Morgan Kaufmann (1999) 406–414
3. Hirota, K., Pedrycz, W.: Implicitly-supervised learning and its application to fuzzy pattern classifiers. Information Sciences **106** (1998) 71–85
4. Gamberger, D., Lavrač, N.: Filtering noisy instances and outliers. In Liu, H., Motoda, H., eds.: Instance Selection and Construction for Data Mining. Kluwer Academic Publishers, Boston/Dordrecht/London (2001) 375–394

5. Oates, T., Jensen, D.: Large datasets lead to overly complex models: An explanation and a solution. In: Proceedings of The Fourth International Conference on Knowledge Discovery and Data Mining. (1998) 294–298
6. Lewis, D.D., Catlett, J.: Heterogeneous uncertainty sampling for supervised learning. In Cohen, W.W., Hirsh, H., eds.: Proceedings of 11^{th} International Conference on Machine Learning, Morgan Kaufmann (1994) 148–156
7. Freund, Y., Seung, H.S., Shamir, E., Tishby, N.: Selective sampling using the query by committee algorithm. Machine Learning **28** (1997) 133–168
8. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
9. Torgo, L.: Inductive Learning of Tree-based Regression Models. PhD thesis, Department of Computer Science, Faculty of Sciences, University of Porto (1999)