Supervised Learning with Minimal Effort

Eileen A. Ni and Charles X. Ling

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{ani,cling}@csd.uwo.ca

Abstract. Traditional supervised learning learns from whatever training examples given to it. This is dramatically different from human learning; human learns simple examples before conquering hard ones to minimize his effort. Effort can equate to energy consumption, and it would be important for machine learning modules to use minimal energy in real-world deployments. In this paper, we propose a novel, simple and effective machine learning paradigm that explicitly exploits this important simple-to-complex (S2C) human learning strategy, and implement it based on C4.5 efficiently. Experiment results show that S2C has several distinctive advantages over the original C4.5. First of all, S2C does indeed take much less effort in learning the training examples than C4.5 which selects examples randomly. Second, with minimal effort, the learning process is much more stable. Finally, even though S2C only locally updates the model with minimal effort, we show that it is as accurate as the global learner C4.5. The applications of this simple-to-complex learning strategy in real-world learning tasks, especially cognitive learning tasks, will be fruitful.

Keywords: supervised learning, decision tree, minimal effort.

1 Introduction

Traditional supervised learning learns from whatever training examples given to it. This is dramatically different from human learning; human learns simple examples before conquering hard ones to minimize his effort. For example, infants learn from simple words and sentences first and gradually learn complex ones through repeated exposures from parents and care takers. Adults, on the other hand, often learn from simple to complex through repeated reviews of the same materials.

In the human learning research area, the "i+1" education theory suggests that human learns a small piece of new knowledge ("1") based on a large body of previously learned knowledge ("i") [1]. Originally, the "i+1" theory describes that the best methods to acquire a second language are those that supply "comprehensible input" and allow students to produce when they are ready [1,2]. We show that the "i+1" human learning theory can be applied in supervised machine learning: a learning algorithm will take less effort when it learns the next easiest thing based on the current learned knowledge.

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 476–487, 2010.

In this paper, we propose a novel, simple and effective machine learning paradigm that explicitly exploits this important simple-to-complex learning strategy, called *S2C* (Simple to Complex). S2C builds its model with simple examples first. More specifically, it selects those examples that are close to the current model's prediction (thus simple), and updates the model with them. S2C works iteratively in this way which is almost the same as the human learning from simple to complex process.

Experiment results show that our new learning paradigm S2C has several distinctive advantages over C4.5 that takes the training examples in random order. First of all, S2C does indeed take much less effort in building its model than C4.5. This would be important in building machine learning modules that use minimal energy. Energy consumption control is highly important in real-world field deployments [3]. [4] indicates that when modules consume too much energy, they may run out of the batteries quickly and must be aggressively cooled; otherwise they would be unreliable and oftentimes be unavailable for use by the application scientists.

Second, minimal effort learning implies that the process of learning and the final learned model are more stable and reliable. This is certainly crucial for human learning, as well as for machine learning applications. To contrast the difference between S2C and C4.5 taking examples randomly, we also implement an "aggressive" learner who chooses examples from complex to simple. We show that the "aggressive" learner takes even more effort than C4.5.

Finally, even though S2C only locally updates the model with minimal effort, we show that it is as accurate as the global learner C4.5. One might think that as S2C always takes the simplest example to update its model locally and incrementally, it may not predict as accurately as the global learner C4.5 which builds its model on the whole dataset. Our experiment results show that S2C predicts only slightly worse than C4.5. In addition, we design a "mini-review" process, and add it into S2C. S2C then becomes less myopic and is shown to predict just as well as C4.5.

We explicitly exploit and implement this simple-to-complex learning strategy, a ubiquitous human learning strategy, in the machine learning research. Its applications in human-oriented learning tasks, especially cognitive learning tasks, will be fruitful.

The rest of this paper is organized as follows. Several previous works related to learning from simple to complex are reviewed in Section 2. Section 3 describes a generic S2C paradigm. We discuss an efficient implementation of S2C based on the decision tree learning algorithm in Section 4. Experiment results are shown in Section 5. We conclude our work in Section 6.

2 Related Work

Learning from simple to complex gradually may look similar to incremental learning previously studied [5,6], but they are much different. Incremental learning builds classifiers gradually based on whatever the data given passively. However, the S2C learning paradigm actively selects simpler examples first to learn,

and then complex ones. However, the S2C learning paradigm is different from the traditional active learning [7,8]. Active learning selects the unlabeled examples that are most uncertain and acquires their labels from an oracle. Then it rebuilds a completely new model with all the labeled examples. The object of active learning is to learn a model with fewer labelings from the oracle. The S2C learning paradigm, on the other hand, builds a model with simple-to-complex labeled examples (no oracle is required) to reduce the effort.[1]

Ferr, etc. proposed delegating classifiers [9], and its main idea is divide-and-conquer. The learner builds the first classifier with all the training examples, and delegates those examples that cannot be predicted well (complex examples) to build delegating classifiers. However, it is difficult to determine how complex for examples to be delegated. Another potential issue is that the delegated examples probably are not sufficient which may affect the reliability of the delegating classifiers. However, S2C learns examples from simple to complex gradually. Thus S2C has neither threshold nor the difficulty of insufficient examples.

A similar idea to S2C is curriculum learning [10]. It learns by assigning the easier examples with higher weights first and then increasing the weights of the complex examples gradually, which is based on the idea that human or animal learns much better when the examples are organized in gradually more complex order [11]. The difficulty in this method is that the so-called easier or harder examples are given by human, which may be unreasonable and infeasible. However, S2C is a learner-centric paradigm. It means that it is the learner who decides what examples are simple or complex, which is much more realistic and appropriate.

Lifelong learning [12] addresses the situations in which a learner faces a series of different learning tasks providing the opportunity to transfer knowledge. Recently, a number of the transfer knowledge researches have been done in different applications, such as, Web document classification [13,14], sentiment classification [15], reinforcement learning [16], etc. The object of transfer learning is to transfer knowledge of other related, but different source data to the current learning data, such that a good model can be learned with fewer examples. The simple-to-complex strategy in S2C is similar to transfer learning. The difference is that it transfers knowledge that is learned from simple examples to complex examples, such that complex ones can be learned easier. From this perspective, S2C belongs to the vertical transfer learning more; the traditional transfer learning is more similar to the horizontal transfer learning in the psychology research area [17,18].

Deep structure learning attempts to learn high level features by the composition of lower level features [10]. It starts training on simpler human-crafted features and tries to get abstract high level features. One conceivable method is by training each layer one after the other [19,20]. It is also very different from S2C, as S2C learns simpler examples first and then complex examples.

[1] Reducing effort means less energy, e.g. power, that the algorithm needs to consume. IBM Research shows that power consumption is a major problem in designing computers to simulate human brain. `http://spectrum.ieee.org/computing/hardware/ibm-unveils-a-new-brain-simulator`

3 S2C Learning

Our novel simple-to-complex learning strategy models the adult iterative-learning process in which the same set of materials is repeatedly studied, and for each iteration only the currently simple cases are learned.

The rationale behind S2C is that learning is a gradual process of accumulating knowledge. Learning simple examples first may make the learning of harder examples easier, thus the whole learning process becomes easier (less effort) and more reliable. (See Section 3.2 for evaluation metrics for S2C).

3.1 S2C Learning Paradigm

At a high level, S2C is very simple, and it can use any classifier that can produce a refined class probability estimation (see later for details) as its base learner. Generally speaking, for each iteration, S2C selects the simplest example based on the current model and updates the model locally with the selected example, until all examples have been learned.

However, several important issues deserve further explanations. First, how can S2C select the simplest example? Second, how can S2C select the first simplest example before any model is built? Third, how can S2C select the simplest example when tie happens?

The first issue, how to select the simplest example, is crucial for the S2C learning paradigm. Without a proper measurement, S2C cannot choose the simplest example correctly. Here we propose a simple and effective measurement, which uses the prediction error of the current model for an example to measure how simple the example is. The smaller the error, the simpler the example is for the current model. Thus a base learner that can generate a refined class probability estimation is a requirement of S2C, as we mentioned before. Also the measurement is consistent with human intuition about simplicity: the less the surprise (i.e., difference or error), the simpler it is.

The second issue of selecting the first simplest example can be tricky, as the current model is empty. We design a simple and effective strategy for S2C. S2C scans over all the training examples to pick up the most frequent example. If no example appears more than once in the training set, then an example from the majority class will be chosen randomly.

The third issue, the tie-breaking strategy, can be crucial, if tie happens often when S2C selects the simplest example with the current model. If ties happen, S2C must choose one randomly. This may affect the performance of S2C. Indeed, for some algorithms, such as C4.5 (the base learner for S2C in this paper), ties do happen often. This is because C4.5 predicts all examples falling in the same leaf with the same prediction (i.e., same label and same probability estimation). For those algorithms, effective tie-breaking strategies are necessary. For C4.5, we use the Euclidean distance between positive and negative examples and the numbers of positive and negative examples in each node as heuristic information for the tie breaking. Details are presented in Section 4.

3.2 Measurements

In this subsection, the key issue we will discuss is how to evaluate S2C and compare it with other supervised learning algorithms. Since minimal effort is our target, effort is the crucial measurement in our paper. In addition, volatility is also important to measure how stable an algorithm is. Therefore, we present two measurements, effort and volatility, in the following.

Effort is a crucial measurement in this paper. *Effort* indicates how much work a learner has to do to learn certain examples. Intuitively the effort can be computed through energy usage, or computer time, etc. The problem is that it is difficult to measure them since they are related to hardware performances, the efficiency of programming language, etc. As a result, we choose two different methods to reflect the effort indirectly.

One method is using the prediction error to approximate the effort. We call it *error-based effort*, which is almost the same as the measurement for selecting the simplest example (in Section 3.1). The difference is that the measurement in Section 3.1 is used to measure how simple an example is, while the error-based effort here is about a learning model. At the same time, they are close related. If an example is the simplest one for a model, the model will take the least effort to learn it. The other method to reflect the effort is the size of the model being built, called *size-based effort*. Size can reflect how much effort is needed to build the model. Usually the larger the size of a model is, the more effort the model needs. For a decision tree, size-based effort can be the number of the nodes in the tree. These two (error-based effort and size-based effort) are good, universal approximation because they are independent of hardwares or other factors, and can be used to evaluate any learning algorithms easily.

The other important measurement, volatility, is used to evaluate how stable a learner is. If the error rate (accuracy) of a learner varies greatly from different runs, its volatility should be high. Volatility is thus the average value of the standard deviation of the prediction error rate for all of the current models. The volatility reflects the varying range of the error of a learner from different runs. Thus it is also one of the essential measurements to assess a learner.

We have presented the main strategies of the S2C learning paradigm in a high level. In the next section, we will present it with a specific base learner, C4.5.

4 S2C with Decision Tree

As we mentioned, the S2C learning paradigm can take most of the classification algorithms as its base learner easily. In this paper, we combine it with one of the most popular algorithms, C4.5. Originally, C4.5 builds a global tree in "one shot". However, S2C only provides C4.5 examples one by one in a simple-to-complex manner, such that it builds a tree locally and gradually. In the following, we will introduce this cooperation between S2C and C4.5 specifically.

As we discussed in Section 3.1, S2C selects one of the majority examples randomly (assume no repeated examples in datasets) for its base learner, C4.5, to build the first tree model - only a one-leaf tree. Then based on the current

tree, S2C selects the simplest example, which is the one that is predicted most correctly by the current tree, and updates the current tree with it. S2C iterates this selecting and updating process until all the training examples are learned.

However, tie can happen often in decision trees, because a tree returns the label prediction and probability of an example according to the numbers of the positive and negative examples in the node that the example falls in. Thus, two different kinds of tie may happen when S2C selects the simplest example. One is that those examples that fall into the same leaf node. The other one is that those examples that fall into different nodes but have the same probability estimation. S2C could solve these tie problems by selecting one example randomly. However, inevitably this random manner would affect its performance severely. Better strategies are needed to judge which of the equally simple examples is simpler than the others. Two different tie-breaking strategies are given as follows.

Firstly, we design a novel tie-breaking strategy to solve the tie happening in the same leaf. In this situation, the equally simple examples belong to one class, say positive. The simplest positive example should be far away from any negative examples. Thus the tie-breaking strategy is to choose the example that has the furthest distance to its closest negative example comparing to other equally simple ones. That is to say, S2C calculates the Euclidean distance for each equally simple positive example to all the negative examples. It records the closest distance for each positive example, and selects the example having the greatest distance.

$$x_i = \arg\max_i(\min_j(d_{ij})) \quad 0 \leq i < n_1; 0 \leq j < n_0. \tag{1}$$

where, d_{ij} is the distance from example x_i to x_j; x_i and x_j belongs to positive and negative respectively; n_1 is the number of the equally simple examples and n_0 is the number of the negative training examples.

In addition, an efficient strategy also is needed to break the ties among the examples that fall in different leaves but with the same probability estimation. Two factors can be considered for this tie-breaking. One is the number of the examples in the leaves that tie happens. Intuitively, the more examples a leaf has, the more reliable the leaf is in terms of probability estimation. The other factor is the different depths of leaves in the tree. Intuitively the closer a leaf is to the root, the more preferred the example that falls in the leaf is. For the first factor, if two positive examples fall into two positive leaves, say, A and B, and A has *9* positive examples and *1* negative, and B has *18* positive examples and 2 negative, the predictions for the two examples would be tied . However, B should be preferred over A when breaking this tie because B is more reliable. For the second factor, if a leaf C is on the first level and D is on the fifth level, and both C and D have, say, *9* positive examples and 1 negative, the example that falls in C should be preferred over the one that falls in D, since only one attribute is needed to predict its label. By combining the two factors, we propose Formula 2 as a heuristic for the tie-breaking. The preference of an example x_i can be defined as:

$$Preference = \frac{1 + \frac{N_1 - N_0}{N_{root}}}{2} * \frac{1}{L_{path}} \quad (2)$$

where, N_1 and N_0 are the numbers of the positive and negative examples respectively in the leaf node that x_i falls in; N_{root} is the number of the examples in the root node; L_{path} is the length from the root to the leaf that x_i falls in.

In Formula 2, the value of $\frac{1+\frac{N_1-N_0}{N_{root}}}{2}$ corresponds to the first factor we discussed. It indicates how positive x_i is. If $N_1 = N_0$, the value will be *1/2*; if N_1 is very small, its value will be close to zero (more negative); otherwise, it will be close to *1* (more positive). *1/Lpath* corresponds to the second factor. The closer a leaf to its root, the more preferred the example is.

During the iterative process of selecting the simplest example and updating the current tree, two issues need to be explained further. One is that if tie still happens with Formula 1 or Formula 2, S2C will select one example randomly. The other one is that S2C will only split the fringe node if needed when updating the current tree model. That is to say, if an example agrees with the fringe node it enters, S2C will not change the tree structure; otherwise it will split the fringe node according to the traditional C4.5 algorithm. This fringe-node-split strategy reduces the size-based effort effectively.

As we have discussed earlier, two measurements, effort (error-based and size-based effort) and volatility, can be used to evaluate the S2C paradigm effectively. Here we provide the details of how to evaluate the tree-based S2C in terms of effort first. The error-based effort can be calculated by the prediction error of the current model. For a decision tree, the prediction error of the current tree T_i for a positive example x_i is $(1 - p(1|x_i))$, where $p(1|x_i)$ is calculated by k/n. k is the number of the positive examples and n the total number of the examples in the node that x_i falls in. As discussed, another way to measure effort is the size-based effort. For a decision tree T_i, the size-based effort can be the number of the nodes in T_i. The more nodes in a tree, the more size-based effort is needed to build the tree.

The other important measurement, volatility, as discussed in Section 3.2, is the standard deviation of the prediction error rate . It indicates how volatile a learner is.

As we have presented, the S2C learning paradigm can easily take C4.5 as its base learner and works in a simple and effective manner. With the two new measurements, we will show that S2C indeed has several advantages over other learning algorithms in the next section.

5 Experiments

To illustrate the performance of S2C, we choose 10 UCI [21] datasets, including anneal, autos, breast_cancer, colic, diabetes, ecoli, glass, heart-h, sonar and vote, which are commonly used in the supervised learning research area. Originally autos and glass are multi-class. However, to compare with the other binary-class

datasets, we transform them to binary-class datasets, autos_new and glass_new, by keeping the majority class and merging all the other classes. All the algorithms are implemented based on the WEKA [22] source code. In the experiments, 5-fold cross validation is used and the t-test results are with 95% confidence.

5.1 Comparison of Effort

We first compare S2C with C4.5 in terms of effort. C4.5 selects a new example from the training data randomly (not from simple to complex). In addition, instead of updating the current tree locally in S2C, C4.5 rebuilds a tree based on the current examples it has.[2]

To further contrast the differences between S2C and C4.5, we also implement a maximal effort learner, called *Jump-start*, which is very similar to S2C in framework but works in an opposite way. The Jump-start learner selects the most complex example (be predicted most wrongly) based on the current model and updates the current model with it iteratively until all the examples are learned. The rationale behind Jump-start is that the complex examples are more informative and useful to improve the current model. To make the comparison meaningful, Jump-start also takes C4.5 as its base learner.

As discussed in Section 4, we concern two kinds of effort, error-based effort and size-based effort. The error-based effort on the whole training set is the sum of the error when the learning algorithms process and learn every training example. Similarly, size-based effort of decision trees on the whole training set is the sum of the node number of the trees when each training example is learned.

We conduct the t-test on all the 10 datasets to compare the error-based effort between S2C, C4.5 and Jump-start. The results show that S2C takes much less error-based effort than C4.5 on 9 datasets, ties with C4.5 on only one dataset and loses on no dataset. However, the performance of Jump-start is poor. It loses to S2C and C4.5 on all of the 10 datasets. To further show the differences among the three algorithms, we present the average of error-based effort of the 5 runs on each dataset in the upper part of Table 1. On average C4.5 and Jump-start take about 1.3 times and 2 times as much as the error-based effort of S2C respectively. Thus it is clear that S2C needs much less error-based effort than C4.5 to learn all the training examples.

We also conduct the t-test on all the 10 datasets to compare the size-based effort. The results show that S2C wins C4.5 and Jump-start on all the 10 datasets significantly without exception, while Jump-start loses to S2C and C4.5 on all the datasets. We also present the average of size-based effort of the 5 runs on each dataset in the lower part of Table 1. On average, C4.5 and Jump-start take about 2.6 times and 6 times as much as the size-based effort of S2C. Obviously, S2C takes the least size-based effort to build a model, because it updates tree models locally instead of rebuilding trees, which saves much effort.

[2] We may also use ID5 [23], an incremental version of C4.5 to update the tree. However, it is shown [23] that ID5 produces identical tree as C4.5; thus we use C4.5 on the current training dataset directly.

Table 1. The comparison of effort among the three learning algorithms

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
Error-based Effort										
S2C	6.9	7.6	10.0	16.0	41.0	7.0	11.6	13.3	12.5	7.1
C45	7.1	10.5	12.0	22.0	59.0	8.0	16.0	20.3	16.2	9.0
JS	23.7	14.5	23.0	30	73.5	19.3	20.1	22.3	21.4	15.9
Size-based Effort										
S2C	3357	1198	3141	2841	21157	1478	1470	1792	2711	465
C45	14747	2937	12163	7482	49042	4399	4309	4665	3824	1034
JS	51327	7427	27501	14019	95623	12057	7343	9055	10107	3721

The experiment results have convinced us that the performances of S2C are excellent on both kinds of effort. Jump-start is the worst and C4.5 is in-between. Both the error-based effort and the size-based effort can be translated directly to how much energy that the computer running the program will take; however it is beyond this paper and will be our future work.

5.2 Comparison of Volatility

We have already learned that S2C needs much less effort to learn all the training examples. Less effort implies that the learned model should be less volatile. As discussed in Section 4, to compute the volatility, we run the each algorithm on each dataset for 5 times and compute the standard deviation of the error rate. We compare S2C with C4.5 and Jump-start and show the results in Table 2.

Table 2 shows the volatility of the three learning algorithms on each dataset. The volatility of S2C is almost 0 on all the datasets. On average, C4.5 is thousands of times more volatile than S2C, and Jump-start is about 3 times as volatile as C4.5. Thus it is clear that S2C is stable, while C4.5 and Jump-start are much more volatile. The reason is that no matter what sequence of the training examples is, S2C always results in a similar model. Learning algorithms with small volatility are important in learning real time data.

Also in Figure 1, we show that the error rate of S2C decreases more stably than C4.5 and Jump-start with more training examples taken in. We conduct the experiments on the 10 datasets, however, because of the limited space we only show the typical results of two datasets, colic and heart-h, in Figure 1. The x axis indicates the times of updating the trees and the y axis is about the error rate. Since the S2C model takes in all the examples of majority class first, it is only a one-leaf tree. The true building tree process only starts from taking in the first minority example. Thus its updating time is much less than that of C4.5 and Jump-start. From the figure, we can tell that the learning process of S2C is much shorter than the other two.

Figure 1 shows that with the increasing number of examples taken in, the error rate of Jump-start fluctuates very often at the beginning. C4.5 is a little

Table 2. The comparison of volatility among the three learning algorithms

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
S2C	5.6E-7	0.0	2.4E-8	3.1E-6	0.0	1.2E-8	0.0	4.7E-7	3.8E-6	5.6E-6
C45	0.011	0.032	0.016	0.003	0.024	0.039	0.039	0.032	0.067	0.020
JS	0.030	0.051	0.012	0.025	0.027	0.023	0.024	0.028	0.068	0.071

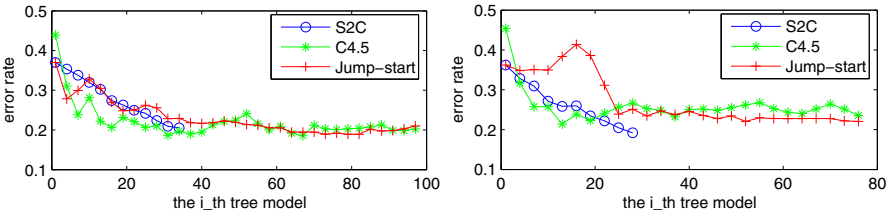

Fig. 1. The error rate on two datasets: (a) colic and (b) heart-h

better than Jump-start; however it still trembles at the beginning on most of the data. On the other hand, the error rate of S2C decreases more stably than the other two.

5.3 Comparison of Error Rate

As discussed in Section 4, S2C only updates the current tree at its fringe when the new simplest example is selected. It is extremely local and myopic. On the other hand, C4.5 rebuilds a tree with all the examples it has, and its final tree is built on the whole dataset. Thus C4.5 is a "global" tree. One might expect that S2C does not predict as well as C4.5. To compare the final error rate between S2C, C4.5 and Jump-start, we perform the t-test on the 10 datasets, after taking in the whole training set. We find that S2C is slightly worse than C4.5. It ties with C4.5 on 7 datasets, but loses on 3 datasets. The results were shown in the upper part of Table 3. This is expected as S2C updates the tree locally.

To avoid the greedy updating strategy of S2C without increasing the effort too much, we design a "mini-review" strategy. The strategy is that after learning K examples S2C will "review" them and use them together to expand the fringe of the tree. This "mini-review" process is similar to the summarizing process in human learning. The lower part of Table 3 gives the experiment results of the 10 datasets when K=*10*. It shows that S2C has almost the same low error rate as the "global" tree C4.5. Also we find that the error-rate performance of Jump-start is fairly good, and only loses to S2C and C4.5 on one dataset respectively. However, this similar performance costs Jump-start extraordinary effort.

The experiment results illustrate that S2C can work as well as the global algorithm C4.5 on error rate. However, it does take much less effort than C4.5 and Jump-start. In addition, S2C is much more reliable than the other two

Table 3. The t-test results on error rate (L: lose; T: tie; W: win)

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
					K = 1					
S2C vs. C45	L	L	T	T	T	L	T	T	T	T
C45 vs. JS	W	T	T	T	T	W	T	T	W	T
JS vs. S2C	T	T	T	T	T	L	T	T	T	T
					K = 10					
S2C vs. C45	T	T	T	T	T	T	T	T	T	T
C45 vs. JS	W	T	T	T	T	T	T	T	T	T
JS vs. S2C	L	T	T	T	T	T	T	T	T	T

algorithms. These are crucial to learning algorithms. All the experimental results show that the simple-to-complex learning strategy has distinctive advantages in the machine learning area.

6 Conclusions

In this paper we present the S2C learning paradigm, which exploits the simple-to-complex human learning strategy in the supervised learning research area, and implement it with C4.5 as its base learner. Experimental results show that S2C does take much less effort to learn a model than C4.5 and reaches very similar low error rate. Furthermore, S2C is much less volatile than C4.5. In our future work, we will apply S2C to more learning tasks, especially cognitive learning tasks.

References

1. Krashen, S.: The input hypothesis: Issues and implications. Addison-Wesley Longman Ltd., Amsterdam (1985)
2. Dong-lin, Z.: Krashen's Input Hypothesis and English classroom teaching
3. Mohiyuddin, M., Murphy, M., Oliker, L., Shalf, J., Wawrzynek, J., Williams, S.: A design methodology for domain-optimized power-efficient supercomputing. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pp. 1–12. ACM, New York (2009)
4. Hsu, C., Feng, W., Archuleta, J.: Towards efficient supercomputing: A quest for the right metric. In: Proceedings of the High-Performance Power-Aware Computing Workshop, Citeseer (2005)
5. Lange, S., Grieser, G.: On the power of incremental learning. Theoretical Computer Science 288(2), 277–307 (2002)
6. Giraud-Carrier, C.: A note on the utility of incremental learning. AI Communications 13(4), 215–223 (2000)
7. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. The Journal of Machine Learning Research 5, 255–291 (2004)
8. Cohn, D., Ghahramani, Z., Jordan, M.: Active learning with statistical models, Arxiv preprint cs/9603104 (1996)

9. Ferri, C., Flach, P., Hernández-Orallo, J.: Delegating classifiers. In: Proceedings of the twenty-first international conference on Machine learning. ACM, New York (2004)
10. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, New York (2009)
11. Krueger, K., Dayan, P.: Flexible shaping: how learning in small steps helps. Cognition 110(3), 380–394 (2009)
12. Thrun, S.: Is learning the n-th thing any easier than learning the first? Advances in Neural Information Processing Systems, 640–646 (1996)
13. Fung, G., Yu, J., Lu, H., Yu, P.: Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering 18(1), 6–20 (2006)
14. Sarinnapakorn, K., Kubat, M.: Combining Subclassifiers in Text Categorization: A DST-Based Solution and a Case Study. IEEE Transactions on Knowledge and Data Engineering 19(12), 1638–1651 (2007)
15. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Annual Meeting-Association For Computational Linguistics, vol. 45, p. 440 (2007)
16. Taylor, M., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th international conference on Machine learning, p. 886. ACM, New York (2007)
17. Rebello, N., Cui, L., Bennett, A., Zollman, D., Ozimek, D.: Transfer of learning in problem solving in the context of mathematics and physics. In: Learning to Solve Complex Scientific Problems. Lawrence Earlbaum, Mahwah (2007)
18. Gagné, R.: The conditions of learning. Holt, Rinehart & Winston, New York (1970)
19. MarcAurelio Ranzato, Y., Boureau, L., LeCun, Y.: Sparse feature learning for deep belief networks. Advances in neural information processing systems 20
20. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. The Journal of Machine Learning Research 10, 1–40 (2009)
21. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
22. WEKA Machine Learning Project: Weka, `http://www.cs.waikato.ac.nz/~ml/weka`
23. Utgoff, P.: Improved training via incremental learning. In: Proceedings of the sixth international workshop on Machine learning table of contents, pp. 362–365. Morgan Kaufmann Publishers Inc., San Francisco (1989)