

# Coactive Learning for Distributed Data Mining

Dan L. Greco, Lee A. Becker

Department of Computer Science  
Worcester Polytechnic Institute  
Worcester, MA 01609, USA  
dgreco, lab@cs.wpi.edu

## Abstract

We introduce coactive learning as a distributed learning approach to data mining in networked and distributed databases. The coactive learning algorithms act on independent data sets and cooperate by communicating training information, which is used to guide the algorithms' hypothesis construction. The exchanged training information is limited to examples and responses to examples. It is shown that coactive learning can offer a solution to learning on very large data sets by allowing multiple coacting algorithms to learn in parallel on subsets of the data, even if the subsets are distributed over a network. Coactive learning supports the construction of global concept descriptions even when the individual learning algorithms are provided with training sets having biased class distributions. Finally, the capabilities of coactive learning are demonstrated on artificial noisy domains, and on real world domain data with sparse class representation and unknown attribute values.

## Introduction

With the growth in the use of networks has come the need for pattern discovery in distributed databases (Uthurusamy 1996; Bhatnagar 1997; Bhatnagar and Srinivisan 1997). This paper addresses the problem of data mining in distributed databases, in particular learning models for classification or prediction. Certain machine learning methods, referred by the terms integrating multiple models and ensemble methods, can potentially be used with distributed databases. In these methods, individual learners complete their learning tasks independently. Their results or patterns are integrated either when the learned knowledge is used, for example by voting, or before use, through stacked generalization or metalearning (Wolpert 1992; Chan and Stolfo 1995; Fan, Chan, and Stolfo 1996; Shaw 1996). These approaches may be scaled up on massive data sets by reducing space and time requirements through parallelization.

As opposed to these post-learning integration approaches, distributed learning algorithms can cooperate *during* learning (Provost & Hennessey 1996). Coacting is one particular type of cooperative learning. In the following we will describe coactive learning. We will explain how it can be

applied to instance-based learning, an example-based method using a nearest neighbor similarity function; such representations have been popular for data mining (Fayyad, Piatetsky-Shapiro, and Smyth 1966). We will then demonstrate how coactive learning can help the data mining of massive data sets by parallelization. Next we will show how coacting can compensate for skewed distributions of data in distributed databases. Finally we will show how coacting can provide noise filtering.

## Coactive Learning

Coacting is used in the psychological literature to refer to an individual's performance in the presence of other individuals performing the same task (Hill 1982). In the context of this paper we use the term coactive learning to describe a new type of distributed learning. Coactive learning algorithms perform the same learning task. Coacting emphasizes the possibility of modifying the individual learning algorithms through communication of potentially relevant information that results during their individual training tasks. During their communication agents may exchange training examples and how they responded to the examples. For example, they may communicate how the current training example related to their current concept description or what learning operation was triggered by that example. In this paper the coacting learning algorithms will be all assumed to operate in an incremental manner.

Figure 1 depicts two coacting learning algorithms, each performing incremental concept learning from examples. In isolation each algorithm would use its performance engine to make a classification prediction based on its current concept description. Its learning algorithm, based on the classification prediction, the correct classification, and the current hypothesized concept representation, would decide how or whether to apply a learning operation to modify the current concept representation. In coacting, the learning algorithm makes a learning decision based not only on local information, but also on learning information received from one or more coactors. For example, learning algorithm 1 (LA1) could decide to send the current training example to its coactor – learning algorithm 2 (LA2). LA1's algorithm

may then take into consideration in its choice of learning operation the coactor's response to that example. Although Figure 1 depicts only two coacting learning algorithms, in coactive learning there can be any number of coactors.

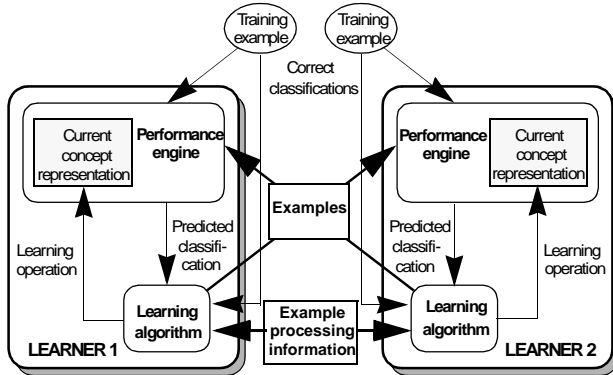


Figure 1: Coactive learning framework

It should be noted that the learning algorithm of one coactor may send an example to its coactor just for the benefit of its coactor. Receiving critical examples may be of significant use to a learning algorithm, especially if the coactor that sends them knows what kind of examples would be of interest. Such examples may reduce the amount of training needed by an agent to achieve a certain level of performance or to guide its hypothesis development. A learning algorithm which is trained on examples sent from a coacting learning algorithm as well as on its own examples emulates a learning paradigm in which the learner is guided through examples supplied by an external teacher.

## Instance-Based Learning

Instance-based learning (IBL) is an inductive learning model, that generates concept descriptions by storing specific training instances (Aha, Kibler, and Albert 1991). This approach improves prior work on nearest neighbor classification (Cover and Hart 1967) which stores all training instances, and thereby has larger memory requirements, slow execution speed, and is sensitive to noise.

Aha, Kibler, and Albert (1991) present and compare three algorithms for IBL: IB1, IB2 and IB3. A prediction about a new instance results from the classification given by the most similar  $k$  stored instances. The algorithms use a similarity function, such as the square root of the sum of the squared differences of the attribute values and which allows for nominal attributes and missing values. The algorithms differ in their storage update strategies. IB1 simply stores all training instances. IB2 stores only misclassified training instances. This reduces storage require-

ments significantly, but makes the algorithm sensitive to noise. IB3 stores all misclassified training instances, but maintains a classification record of correct vs. incorrect predictions, discarding significantly poor predictors. IB3 is significantly more noise tolerant than IB1 and IB2 and, among the three algorithms, uses the least number of instances in its concept description. As a result, in noisy domains IB3 significantly outperforms IB1 and IB2.

## Coacting in Instance-Based Learning

### Coactive Learning with Two IB2 Learners

To illustrate the coactive learning approach consider two IB2 algorithms (IB2-LA1 and IB2-LA2) coacting. IB2-LA1 and IB2-LA2 are learning on different training instances. Certain training instances are sent by IB2-LA1 to IB2-LA2. The training instances can be sent for two distinct purposes. The first purpose is for IB2-LA1 to get a second 'opinion', i.e., to get feedback from IB2-LA2. This feedback is meant to aid IB2-LA1's learning. The second purpose is for the edification of IB2-LA2, i.e., to send an interesting or critical example to aid IB2-LA2's learning.

Scheme	Learner	Storage pattern				Obs.	
1	IB2-LA1	-NS	-S	GCI +NS	+NS	modif.	
	IB2-LA2	-S	+NS	-S	+NS	unmodif	
2	IB2-LA1	-S	PSI	-S	PSI +NS	+NS	unmodif
	IB2-LA2	-S	PSI	+NS	-S	PSI +NS	unmodif

Table 1: Some possible coaction schemes for two IB2 agents (+/- means correctly/incorrectly classified instance, S/NS means stored/not stored instance, GCI means getting the classifying instance from the actor and storing it, PSI means pass the stored training instance)

We consider first the interaction for feedback purposes. When IB2-LA2 is asked to comment on a training instance provided by IB2-LA1, the most immediate type of response from IB2-LA2 would be whether its current concept description correctly classified the training instance. If IB2-LA1 misclassified an instance and it knew that IB2-LA2 also misclassified it, the IB2-LA1 algorithm may decide that the instance is likely to be noise and decide not to store that instance. In addition to the classification prediction made by a coactor, it might also prove of value to an IB2 learning algorithm to know the most similar instance used by its coactor to make a correct classification prediction. If IB2-LA1 misclassifies a training instance and the coacting IB2-LA2 correctly classifies the same instance, IB2-LA1 may wish to store the instance which IB2-LA2 used to make the correct prediction.

Table 1 presents two coacting schemes. In both schemes the coactors are learning on different data. In scheme 1 IB2-LA1 sends each of the training instances to IB2-LA2 for a second ‘opinion’, i.e. for feedback. IB2-LA2 is learning on its own data set using the unmodified IB2 algorithm. Its purpose is to *help* IB2-LA1, where the learned model will ultimately reside. IB2-LA1 makes use of the information sent back from IB2-LA2; in particular, when IB2-LA2 has also misclassified a training instance that IB2-LA1 misclassified, IB2-LA1 does not store the training instance. If IB2-LA2 correctly classifies the training instance that IB2-LA1 has misclassified, it sends to IB2-LA1 the exemplar which it has used for classification and IB2-LA1 stores that instance.

Scheme 2 illustrates coacting interaction for the edification of a coactor. Both IB2-LA1 and IB2-LA2 use unmodified IB2 learning algorithms, and send any misclassified instances to their coactor. The coactor stores the passed training instances. In this case, all the coactors will eventually develop the same learned model through integration of the training instances received by each individual learner.

The schemes in Table 1 involve only two coactors, but coactive learning can involve any number of coactors. In the experiments carried out below the number of coactors will vary. Scheme 2 will be used for the experiments dealing with scaling up through parallelization and those dealing with handling skewed distributions at the nodes of a distributed database, while scheme 1 will demonstrate the noise filtering capabilities of coactive learning.

### Coaction for Distributed or Parallelized Learning

For the purpose of this discussion we will use an artificial test domain. The instances represent points with integer coordinates lying inside a 100x100 square, subdivided into four equal-sized subsquares. The prediction task consists of determining the subsquares to which the test instances belong. Experiments are averaged over 50 runs. For each run a testing set had 200 instances, while the dimension of the training set was varied up to 4000 instances. In each run the total pool of instances was randomly separated into a training and testing set, and each set was randomly ordered. The IB2 algorithms performed classification based on the closest classifying instance ( $k = 1$ ).

Table 2 presents results from experiments with multiple IB2 learners using scheme 2 from Table 1. Each coactor is an unmodified IB2 algorithm learning independently on its own data. At the same time each learning algorithm sends each training instance which it misclassifies to all its coactors for edification, and each coactor stores all instances passed to it. Column 1 shows the total number of training

instances. These instances are equally divided among the coactors. For example, on the row with 400 training instances, one IB2 algorithm learns on all 400 instances, two IB2 learners have 200 instances each, four IB2s have 100 each, and eight IB2s have only 50 each.

Tr. inst.	1 IB2 alg.	2 IB2 alg.	4 IB2 alg.	8 IB2 alg.
30	84.40 (7.78)	85.29 (7.94)	84.59 (8.74)	82.16 (8.28)
40	86.31 (9.78)	86.17 (10.52)	85.26 (10.22)	84.74 (9.82)
50	87.76 (11.10)	86.75 (11.24)	86.83 (11.48)	87.39 (12.40)
75	89.22 (13.92)	89.48 (14.00)	89.26 (14.02)	89.13 (14.58)
100	91.65 (15.34)	91.41 (16.04)	91.55 (16.36)	91.00 (16.86)
200	93.77 (22.68)	94.17 (24.06)	94.12 (23.56)	93.78 (23.78)
300	95.19 (28.46)	94.97 (29.02)	94.84 (29.89)	94.67 (28.66)
400	94.57 (33.74)	94.72 (34.74)	94.75 (35.26)	94.86 (35.40)
800	96.62 (47.52)	96.53 (48.76)	96.37 (48.78)	96.69 (50.10)
1200	97.25 (61.14)	97.42 (60.86)	97.55 (60.94)	97.60 (64.38)
1600	97.83 (72.10)	97.85 (71.16)	97.87 (72.74)	97.97 (74.60)
2000	98.02 (78.06)	98.16 (76.64)	98.10 (77.08)	98.16 (77.90)
2400	97.85 (85.20)	97.98 (83.02)	97.82 (84.16)	98.24 (86.58)
2800	98.17 (93.68)	98.2 (92.98)	98.24 (94.08)	98.22 (93.54)
3200	98.51 (105.2)	98.36(102.68)	98.29(103.02)	98.07(104.94)
3600	98.37(110.16)	98.29(107.5)	98.36(107.18)	98.17(109.04)
4000	98.37(119.04)	98.39(116.92)	98.38(116.4)	98.52(119.02)

Table 2. Prediction accuracies of the independent IB2 algorithm and of 2, 4, and 8 coacting algorithms, using scheme 2 from Table 1 (numbers in parentheses represent the number of stored exemplars in the concept description)

Except for the first two rows the prediction accuracies for differing numbers of IB2 learners are virtually identical. The eight learners in the column at the right, although they learned individually on one-eighth of the data used by the single learner in the column at the left, achieved the same accuracy. This scheme allows to parallelize learning tasks on large instance sets and to achieve a learning speed up roughly proportional to the number of processors. The scheme is also useful in inherently distributed environments, with training data collected at different locations.

Note that in any row the number of instances stored by the learners is virtually identical. Thus the number of instances stored is independent of the size of the individual training sets. The instances sent to a learner from its coactors reduce the number of misclassifications the learner makes on its own dataset, and thus the number of instances stored from its own dataset. For example, one independent IB2 learner training on 200 instances misclassifies and

stores on average 22.68 instances. The eight learners in the row with 1600 instances, each store 74.60 instances, and not the 181.44 they would store if they each misclassified 22.68 and received 22.68 from each of its coactors.

In the situation where the data sets are distributed over a network coacting can be of significant value. As an alternative to sending all the data sets to one node for data min-

ing, the amount of data that must be passed is considerably diminished. The total number of instances passed between coactors equals the number of instances stored by a coactor multiplied by the number of learners less one. For the case with 4000 training instances and 8 data locations, this requires a transfer of 860 instances vs. 3500 instances that have to be sent if the training is to happen in one node.

Train. inst.	Percentage of biasing training instances relatively to the total number of training instances received by an agent						
	60%	50%	40%	30%	20%	10%	0%
100	90.1 (17.98)	90.39 (17.36)	91.35 (17.1)	90.82 (16.9)	91.01 (16.44)	90.48 (17.4)	89.68 (17.72)
200	93.62 (26.68)	93.69 (26.56)	92.98 (25.28)	93.05 (25.88)	92.78 (25.0)	92.79 (26.4)	92.67 (26)
300	94.82 (30.42)	94.43 (30.42)	94.41 (32.4)	94.78 (33.34)	95.18 (31.28)	95.12 (31.72)	95.5 (31.84)
400	94.87 (39.2)	94.72 (38.9)	95.26 (36.84)	95.14 (36.38)	94.76 (36.96)	95.02 (36.2)	94.86 (36.66)
800	96.16 (53.36)	96.75 (54.24)	96.46 (52.84)	96.33 (57.42)	96.55 (53.9)	96.4 (54.08)	96.59 (54.26)
1200	96.32 (67.4)	96.53 (65.56)	96.53 (64.16)	96.99 (65.22)	97.13 (64.8)	97.24 (64.98)	96.8 (67.48)
1600	97.28 (85.58)	97.01 (83.42)	97.44 (76.06)	97.3 (69.4)	97.6 (74.84)	97.77 (73.0)	97.62 (65.16)
2000	97.98 (88.76)	98.12 (82.28)	98.16 (83.74)	98.02 (88.7)	98.17 (86.8)	98.32 (81.46)	98.39 (78.24)

Table 3. Sensitivity of a coacting scheme involving 4 IB2 learners to various degrees of class bias of the training instances (numbers in parentheses represent the number of stored exemplars in the concept description)

### Coaction to Compensate for Biased Class Distribution in the Training Data

When data is gathered and stored at different sites, the data at any site may not reflect the natural global distribution of the data. The experiments summarized in Table 3 aimed to test whether coacting compensates for these biases. We used 4 coacting IB2 algorithms coacting based on scheme 2 from Table 1. Recall that the artificial domain described above had 4 classes. The columns represent the amount of bias in terms of class distribution in the data sets of the four learners. 0% represents a balanced class distribution in the training set received by each learner. 60% bias means that 60% of the training instances were selected to be from one class, the rest of the training instances being evenly distributed among the four classes. Each of the four learners had the training data biased for a different class. The prediction accuracy and the required storage were virtually identical across any row. This indicates that coacting did indeed compensate for the biased distributions.

### Coacting for Noise Reduction and on Real World Domains

**Noisy domains.** Table 4 shows experiments with two coactive learners that were using the feedback scheme in Table 1 (scheme 1). The experiments used 320 training instances from the previously described artificial domain, equally divided between the two learners, and 80 testing instances. The degree of noise was 10% – there was a 10%

chance for a classifying feature of a training instance to be affected by noise.

Agent	Coacting protocol				No noise	10% noise
IB2 (A)	-NS	-S	GCI	+NS +NS	92.7 (23.6)	84 (28.1)
IB2 (B)	-S	+NS	-S	+NS	94.5 (20.6)	74.7 (56.2)

Table 4: Scheme 1 coacting results on the artificial domain (GCI means get classifying instance from coactor)

The cells with an enhanced border in the table represent situations where IB2-LA1 consults its coactor IB2-LA2 on how it would classify the current instance of IB2-LA1. IB2-LA2 doesn't store IB2-LA1's training instance, no matter what the result of the classification is. However, IB2-LA1, depending on IB2-LA2's response on the forwarded training instance, may request IB2-LA2 to provide the classifying instance it has used for prediction (such situations are marked by a GCI code). IB2-LA2 runs a standard IB2 algorithm. On noisy data sets, IB2-LA1 outperforms IB2-LA2 in terms of accuracy by 9.3%, while storing only half as many instances as IB2-LA2 (28 vs. 56 instances). On the no-noise data IB2-LA1 eliminates some instances that it falsely assumes to be noise, and therefore its accuracy is 1.8% lower than that of IB2-LA2.

**Sparse databases and training sets with unknown attributes.** To examine coactive learning in real world domains, the previous scheme was also run on four of the databases from the University of California at Irvine Machine Learning Repository. We used the following training/testing ratios: Cleveland database – 250/53, Hun-

garian database – 250/44, Tumor database – 275/64, Voting database – 350/85. The databases are representative for data characterized by large numbers of attributes, by sparsely distributed training instances, by the presence of unknown attribute values, and by nominal and numeric attribute values. Distances to instances having unknown attributes were considered infinite.

IB2-LA1 performs better than the standard IB2 (represented by IB2-LA2) in three of the four domains. This shows that the coacting IB2-LA1 is not only more robust to noise, but is also more robust on non-noisy training data with missing attributes and sparse class distributions.

Alg.	Cleveland	Hungarian	Tumor	Voting
IB2-LA1	51.3 (32.5)	77.4 (24.0)	33.1 (32.8)	91.6 (18.0)
IB2-LA2	50.4 (132.6)	71.5 (63.5)	34.1 (190.2)	89.9 (36.5)

TABLE 5. Coactive IB2-IB2 results on the repository databases (scheme 1). Results are averaged over 50 runs

## Discussion

In using coactive learning for data mining, the coactors can differ in several possible ways. They may use the same (SR) or different (DR) representational forms. In the latter case, they may use the same (SA) or different (DA) learning algorithms. The differences in the representational form and the algorithm can be combined into three types: SR-SA, SR-DA, DR-DA.

In the coactive learning experiments presented above the coactors used a SR-SA scheme, with the same representational form (instances) and the same learning algorithm (IB2). We have also investigated the SR-DA scheme, using instances as the representational form, but with IB2 coacting with IB1. From this interaction emerged a noise reduction capability surpassing that of IB3, and a better performance than IB3's on real world domain data. We believe that coactive learning with different representational forms and different algorithms will also prove useful

## Conclusion

Coacting is a particular type of cooperative learning. After describing coactive learning and how it can be applied to instance-based learning, we conduct experiments using artificially generated data, as well as real domain data from the data repository at the University of California at Irvine. We have shown how coactive learning can support scaling up of data mining on massive data sets through parallelization. Coactive learning was also shown to compensate for class bias when learning on distributed data, and was shown to provide noise filtering.

**Acknowledgments.** The authors would like to thank Prof. David Brown from the WPI Computer Science Department for the insightful comments on this paper and the reviewers for their helpful suggestions.

## References

- Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning*, 6: 37-66.
- Bhatnagar, R. 1997. Learning by Cooperating Agents. In *AAAI-1997 Workshop on Multiagent Learning*, Technical Report WS-97-03, Menlo Park, CA: AAAI Press, 1-6.
- Bhatnagar, R.; and Srinivasan, S. 1997. Pattern Discovery in Distributed Databases. In *Proc. of the 14th Nat. Conf. on Artif. Intell.*, Menlo Park, CA: AAAI Press, 503-508.
- Chan, P. K.; and Stolfo, S. J. 1995. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. of the 12th Int. Conference on Machine Learning*, Mahwah, NJ: Morgan Kaufmann, 90-98.
- Cover, T.M.; and Hart, P.E. 1967. Nearest neighbor pattern classification. *IEEE Trans. on Inf Theory*, 13(1): 21-27.
- Fan, D. W.; Chan, P. K.; and Stolfo, S. J. 1996. A comparative evaluation of combiner and stacked generalization. In *Proc. of AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, 40-46.
- Fayyad, U.; Piatetsky-Shapiro, G.; and Smyth, P. 1996. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Comm. of the ACM*, 39(11): 27-34.
- Hill, G.W. 1982. Group vs. individual performance: Are N+1 heads better than one? *Psych. Bull.*, 91(3): 517-539.
- Iba, W.; Wogulis, J.; and Langley, P. 1988. Trading off simplicity and coverage in incremental concept learning. In *Proc. of the 5th Int. Conference on Machine Learning*, Mahwah, NJ: Morgan Kaufmann, 148-156.
- Provost, J.P.; and Hennessy, D.N. 1996. Scaling up distributed machine learning with cooperation. In *Proc. 13th Nat. Conf. on Artif. Intell.*, Menlo Park, CA: AAAI Press.
- Shaw M. 1996. Cooperative Problem-Solving and Learning in Multi-Agent Information Systems. *Int. Journal on Comput. Intelligence and Organizations*, 1(1): 21-34.
- Uthurusamy, R. 1996. From data mining to knowledge discovery: Current challenges and future directions. In Fayyad U.M.; Piatetsky-Shapiro G.; Smyth P.; and Uthurusamy R. eds., *Advances in knowledge discovery and data mining*, Menlo Park, CA: AAAI Press/MIT Press.
- Wolpert, D. H. 1992. Stacked generalization. *Neural Networks*, 5: 241-259.