Appears in *Proceedings of the 1992 AAAI Spring Symposium on Knowledge Assimilation,*
Standford, CA, March, 1992

Batch versus Incremental Theory Refinement *

Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, TX 78712
mooney@cs.utexas.edu

Abstract

Most existing theory refinement systems are not incremental. However, any theory refinement system whose input and output theories are compatible can be used to incrementally assimilate data into an evolving theory. This is done by continually feeding its revised theory back in as its input theory. An incremental batch approach, in which the system assimilates a batch of examples at each step, seems most appropriate for existing theory revision systems. Experimental results with the EITHER theory refinement system demonstrate that this approach frequently increases efficiency without significantly decreasing the accuracy or the simplicity of the resulting theory. However, if the system produces bad initial changes to the theory based on only small amount of data, these bad revisions can "snowball" and result in an overall decrease in performance.

Introduction

Recently, a number of machine learning systems have been developed that use examples to revise an approximate (incomplete and/or incorrect) domain theory [Ginsberg, 1990; Ourston and Mooney, 1990; Towell and Shavlik, 1991; Danyluk, 1991; Whitehall *et al.*, 1991; Matwin and Plante, 1991]. However, these systems are "batch" learners, which process all of the training instances at once. Knowledge assimilation requires the ability to incrementally revise a domain theory as new data is encountered. Incremental processing allows for continual responsiveness to the environment and the potential for improved efficiency and the ability to deal with *concept drift* [Schlimmer and Granger, 1986]. Consequently, there has been a growing body of work on incremental empirical learning systems [Schlimmer and Fisher, 1986; Utgoff, 1989; Reinke and Michalski, 1988].

Unlike a purely empirical system, a theory revision system takes an initial domain theory as well a set of training examples and produces a revised theory represented in the same language. Consequently, a general theory revision system is "input-output compatible," so it is a trivial matter to make it incremental by continually feeding its output from processing one or more examples back into the input for processing additional examples. It is important to notice that theory revision systems that assume a strictly overly-specific (incomplete) initial theory [Wilkins, 1988; Danyluk, 1991; Whitehall *et al.*, 1991] or a strictly overly-general (promiscuous) initial theory [Flann and Dietterich, 1989; Mooney and Ourston, 1989; Cohen, 1990] are not "input-output compatible" since the output theory may not meet the required restrictions on the input theory. However, systems that can handle arbitrarily incorrect initial theories [Ginsberg, 1990; Ourston and Mooney, 1990; Towell and Shavlik, 1991] can easily be made incremental.

This paper presents empirical results on an *incremental batch* [Clearwater *et al.*, 1989] version of EITHER, a revision system for refining arbitrarily incorrect propositional Horn-clause theories [Ourston and Mooney, 1990; Mooney and Ourston, 1991b]. After processing a small batch of training examples, the resulting revised theory is fed back as the input theory for processing the next batch. In the limit, the system can be made completely incremental by setting the batch size to one. Using a theory revision system in incremental batch mode can be done in several ways depending on the extent to which previous training examples are retained. In particular, one can take a *full-memory* approach in which the examples from all previous batches are input to the processing of a given batch, or a *no-memory* approach in which only the current batch of examples is given to the system. This paper presents results on three versions of EITHER: BATCH, FULL-MEM, and NO-MEM. Generally, as the system is made more incremental (i.e. moving from BATCH to FULL-MEM

*This research was supported by the NASA Ames Research Center under grant NCC 2-629 and by the National Science Foundation under grant IRI-9102926. Equipment used was donated by the Texas Instruments Corporation.

to NO-MEM), training time is decreased at the cost of slightly decreasing accuracy and increasing theory complexity. However, the exact trade-offs involved depend on the details of the domain and the initial theory.

Overview of EITHER

Problem Definition

EITHER solves the following theory refinement problem for classification tasks:

> **Given**: An imperfect domain theory for a set of categories and a set of classified examples each described by a set of observable features.
>
> **Find**: A minimally revised version of the domain theory that correctly classifies all of the examples.

EITHER is restricted to Horn-clause theories expressed in an extended propositional logic that allows numerical and multi-valued features as well as binary attributes. In addition, domain theories are required to be acyclic and therefore a theory defines a directed acyclic graph (DAG). For the purpose of theory refinement, EITHER makes a closed-world assumption. If the theory does not prove that an example is a member of a category, then it is assumed to be a negative example of that category. Propositions that are used to describe the examples (e.g. (color black)) are called *observables*. To avoid problems with negation as failure, only observables can appear as negated antecedents in rules. Propositions that represent the final concepts in which examples are to be classified are called *categories*. EITHER assumes the categories are mutually disjoint. Propositions in the theory that are neither observables nor categories are called *intermediate concepts*.

It is difficult to precisely define the adjective "minimal" used to characterize the revision to be produced. Since it is assumed that the original theory is "approximately correct" the goal is to change it as little as possible. Syntactic measures such as the total number of literals added or deleted are reasonable criteria. EITHER uses various heuristic methods to help insure that its revisions are minimal in this sense. However, finding a revision that is guaranteed to be syntactically minimal is clearly computationally intractable. When the initial theory is empty, the problem reduces to that of finding a minimal theory for a set of examples.

Figure 1 shows a sample domain theory for animals. This theory is an extended version of a set of rules given in [Winston and Horn, 1989, pages 388-390]. Leading question marks denote variables, which are only used to define thresholds on numerically-valued features. Given a set of randomly generated training examples, and a buggy version of this theory, EITHER can regenerate the correct theory. The initial theory usually used to test EITHER in this domain includes the bugs shown in Figure 2. Items shown in small-caps were added to the theory whereas items shown in italics were deleted.

(mammal) ← (body-covering hair)
(mammal) ← (feed-young milk)
(mammal) ← (birth live)
(bird) ← (body-covering feathers)
(bird) ← (birth egg) (fly)
(ungulate) ← (mammal) (foot-type hoof)
(ungulate) ← (mammal) (ruminate)
(carnivore) ← (eat-meat)
(carnivore) ← (teeth pointed) (foot-type clawed)
(giraffe) ← (ungulate) (neck-length ?n) (≥ ?n 5) (≤ ?n 6) (color tawny) (pattern spots) (pattern-color black)
(zebra) ← (ungulate) (color white) (pattern stripes)(pattern-color black)
(cheetah) ← (mammal)(carnivore)(color tawny) (pattern spots)(pattern-color black)
(tiger) ← (mammal) (carnivore) (color tawny) (pattern stripes)(pattern-color black)
(dolphin) ← (mammal) (fore-appendage fin) (color gray)(body-covering moist-skin) (body-length ?b) (≥ ?b 4) (≤ ?b 6)
(whale) ← (mammal) (fore-appendage fin) (color gray) (body-covering moist-skin) (body-length ?b) (≥ ?b 10) (≤ ?b 60)
(bat) ← (mammal) (color black) (pattern none) (pattern-color none) (fly)
(platypus) ← (mammal) (birth egg) (foot-type webbed)
(ostrich) ← (bird) (neck-length ?n) (≥ ?n 3) (≤ ?n 4) (color white) (pattern patch) (pattern-color black) (not (fly))
(penguin) ← (bird) (color white) (pattern patch) (pattern-color black) (foot-type webbed) (not (fly))
(duck) ← (bird) (foot-type webbed) (fly)
(grackle) ← (bird) (color black) (pattern none) (pattern-color none) (fly)

Observable Features: feed-young, body-covering, birth, eat-meat, fly, teeth, fore-appendage, foot-type, neck-length, body-length, color, pattern, pattern-color, ruminate.

Intermediate Concepts: mammal, bird, ungulate, carnivore.

Categories: giraffe, zebra, cheetah, tiger, dolphin, whale, bat, platypus, penguin, ostrich, duck, grackle.

Figure 1: Animal Theory

(mammal) ← (body-covering hair) (FORE-APPENDAGE LEG)
(mammal) ← (feed-young milk) (FORE-APPENDAGE LEG)
(mammal) ← (birth live) (FORE-APPENDAGE LEG)
(bird) ← *(body-covering feathers)*
(bird) ← *(birth egg)* (fly)
(ostrich) ← ... *(not (fly))*
(penguin) ← ... *(not (fly))*
(duck) ← (bird) (foot-type webbed) *(fly)*

Figure 2: Standard Animal Theory Bugs

The faults introduced include missing rules, additional antecedents, and missing antecedents. In most trials, one hundred random training examples are sufficient to produce a fully corrected animal theory.

Refinement Algorithm

EITHER's theory refinement algorithm is presented in various levels of detail in [Ourston and Mooney, 1990; Mooney and Ourston, 1991b; Ourston, 1991]. It was designed to correct theories that are either overly-general or overly-specific or both. An overly-general theory is one that causes an example (called a *failing negative*) to be classified in categories other than its own. EITHER specializes existing antecedents, adds new antecedents, and retracts rules to fix these problems. An overly specific theory causes an example (called a *failing positive*) not to be classified in its own category. EITHER retracts and generalizes existing antecedents and learns new rules to fix these problems. Unlike other theory revision systems that perform hill-climbing (and therefore subject to local maxima), EITHER is guaranteed to fix any arbitrarily incorrect propositional Horn-clause theory [Ourston, 1991].

During theory generalization, EITHER uses a greedy covering algorithm to find a near-minimum set of leaf-rule [1] antecedent retractions that correct all of the failing positive examples. At each iteration of the covering algorithm, the system calculates a benefit-to-cost ratio for each set of antecedent retractions that would complete a proof for a failing positive, and the set with the most examples covered per antecedent retracted is added to the cover. This continues until all of the failing positives have been covered. If retracting antecedents from a given rule over-generalizes by creating additional failing negatives, EITHER uses the failing positive examples for the rule, and the negative examples that become provable when the consequent of the rule is assumed true, to inductively[2] form a new rule that correctly classifies these examples.

During theory specialization, EITHER uses a greedy covering algorithm to identify a near-minimum set of leaf-rule retractions that fixes all of the failing negatives. At each iteration of the covering algorithm, the system determines the number of faulty proofs in which each rule participates and the rule retraction that removes the most proofs is added to the cover. This continues until all faulty proofs for all failing negatives are removed. If a given rule retraction over-specializes by causing additional failing positives, additional antecedents are inductively learned that discriminate between the positive examples for the category and the erroneously proven negative examples.

Although the current version of EITHER is efficient enough to run on several real-world problems, computing all possible abductive proofs of failing positives and all deductive proofs of failing negatives makes the worst-case time and space complexities exponential. Consequently, we are continuing to develop techniques for improving the efficiency of the algorithm.

The most recent version of EITHER also includes various additional techniques for modifying higher-level rules and dealing with multiple categories [Ourston and Mooney, 1991], using constructive induction to learn intermediate rules and create new intermediate concepts [Mooney and Ourston, 1991a], and handling noisy data [Mooney and Ourston, 1991c].

Incremental Batch Theory Revision

EITHER was originally intended to operate in batch mode, processing all of the training examples at once. However, as previously discussed, it is trivial to run a comprehensive theory revision system incrementally. Assume the function `either(theory, examples)` runs the original batch algorithm and returns the revised theory. Below is a description of a batch and two incremental batch versions of EITHER, where `batch(i)` refers to the ith batch of training examples and `batches(1..i)` refers to the union of the first i batches.

```
batch(theory, examples)
    for i = 1 to n
        let current-theory =
            either(theory, batches(1...i))

full-mem(theory, examples)
    let current-theory = theory
    for i = 1 to n
        let current-theory =
            either(current-theory, batches(1...i))

no-mem(theory, examples)
    let current-theory = theory
    for i = 1 to n
        let current-theory =
            either(current-theory, batch(i))
```

[1] A leaf rule is a rule whose antecedents include an observable or an intermediate concept that is not the consequent of any existing rule.

[2] EITHER currently uses a version of ID3 [Quinlan, 1986] as its inductive component.

What might one predict about the relative performance of BATCH, FULL-MEM, and NO-MEM? First, it is important to realize that the fewer errors there are in the input theory, the faster EITHER runs because it encounters fewer failing examples, meaning it needs to generate a smaller cover of fixes. Since FULL-MEM and NO-MEM start processing later batches with a (hopefully) improved theory generated from previous batches, one might expect them to run faster. Since, after the first batch, NO-MEM gives EITHER a smaller set of training examples, one would obviously expect it to run faster than FULL-MEM.

Since EITHER attempts to minimally refine its input theory, one might expect that if earlier batches result in incorrect changes to the theory, that these changes would tend to persist in FULL-MEM's and NO-MEMS's revised theories. In other words, the incremental systems may exhibit a form of "inertia" which may prevent them from finding the minimal change to the original theory. Consequently, they may produce more complicated theories that do not generalize as well to novel test examples. Since NO-MEM does not even guarantee consistency with the whole training set, we would expect it to be even less accurate than FULL-MEM. In general, one might expect that making the system more incremental is trading off accuracy and minimality of change for an increase in speed.

Experimental Results

This section presents an empirical comparison of the efficiency, accuracy, and theory complexity of BATCH, FULL-MEM, and NO-MEM in three separate domains. The first two domains are artificially created ones involving the classification of animals and computers. The third domain involves a real theory and data set for the DNA promoter problem introduced by [Towell *et al.*, 1990].

Artificial data was automatically generated for the animal theory in Figure 1 and a similar theory for classifying computers based on their appearance (categories: pc, mac, macII, sun, hp, explorer, symbolics; intermediate concepts: workstation, micro, lisp-machine, unix-workstation, macintosh). Thirty examples of each category were generated by first forming "core" examples, which contain just the observables needed to complete a proof. For numerically-valued features, a value is chosen randomly from the range required for a proof. Next, random values for the remaining observable features were added to the core examples to create full examples. However, adding random values can sometimes make an example provable in another category as well. Consequently, each example was checked to make sure it was provable in only one category before adding it to the final data set. A total of 360 examples of animals and 210 examples of computers were created in this manner. The initial animal theory included the faults shown in Figure 2 and a similar set of faults was used for the initial computer theory.

Learning curves were generated by giving each version of the system the same training examples and, after each batch of training examples, recording the training time, the complexity of the revised theory (in terms of the total number of literals), and classification accuracy on the same disjoint test set (all of the remaining examples not used for training). Figure 3 shows the results for the animal domain. These results are averaged over 15 separate trials with different randomly selected training and test sets. Each point plotted on the curves shows the results for processing a single batch of examples including all of the examples since the last plotted point. The plotted points were taken from original tests in this domain and do not always represent equal size batches.[3] Testing with various fixed batch sizes is an obvious area for future experimentation; however, running BATCH with a small batch size is computationally expensive since it must reprocess all of the examples for each batch.

The results in the animal domain are basically as expected. Accuracy decreases slightly going from BATCH, to FULL-MEM, to NO-MEM; however, training time is substantially reduced. NO-MEM processes the last batch 2.5 times faster than BATCH. However, theory complexity increases somewhat with speed because some early incorrect modifications are retained or complicate later revisions. Therefore, the hypothesis that incremental processing trades off accuracy and minimality of change for an increase in speed is supported in this domain.

Figure 4 shows the results for the computer domain averaged over 25 trials. In this domain, the incremental systems do not perform as well. With only a few examples, EITHER makes some bad initial revisions to the theory which, in FULL-MEM and NO-MEM, get carried into subsequent batches and greatly complicate revision. Initial bad revisions have a tendency to "snowball" in the incremental systems and lead to highly non-optimal final solutions. As a result, NO-MEM's theory after the final batch is almost twice as complicated as BATCH's. Notice that BATCH increases the complexity of the theory after the first batch due to some badly chosen revisions; however, since these initial revisions to not get carried into subsequent batches, it eventually recovers and produces the correct theory. The resulting theory complexity for the incremental systems not only decreases accuracy but actually increases their run-time slightly above BATCH's.

As an example of "snowballing", consider an abstract version of a typical scenario in this domain. Assume a rule $A \leftarrow B \wedge C$ is missing the antecedent C. Based on only a few examples in the first batch, an incorrect inductive specialization is made in which the an-

[3] Originally, the distance between data points was increased as the number of examples increased since further out on the learning curve, accuracy changes more gradually.

tecedent D is added to the overly-general rule: A ← B. When processing the second batch, A ← B ∧ D is found to be overly-specific; however, removing D is an over-generalization since then the negatives that D successfully removed from the first batch become provable again. Consequently , the system decides it needs to learn a new rule for A. Since this rule is totally unnecessary, it is unlikely to be correct and will need to be complicated in subsequent batches. Although each batch results in only a small change, the theory can become quite complicated after several such batches. BATCH, on the other hand, always starts with the initial rule A ← B, and once the training set is large enough, it adds the correct antecedent, C, directly to this rule.

Figure 5 shows the results for the DNA domain averaged over 25 trials. The original theory is described in [Towell *et al.*, 1990], it contains 11 rules with a total of 76 literals. The purpose of the theory is to recognize *promoters* in strings of nucleotides (one of A, G, T, or C). A promoter is a genetic region which initiates the first step in the expression of an adjacent gene by RNA polymerase. The data consists of 53 positive and 53 negative examples described by 57 sequential DNA nucleotides and assembled from the biological literature. The initial theory classifies none of the positive examples and all of the negative examples correctly, thus the initial theory is entirely overly specific.

The incremental systems perform very well in this domain. FULL-MEM and NO-MEM's accuracies are comparable to BATCH's while their training time is significantly less. NO-MEM's accuracy is actually better than BATCH's after 40 examples and it processes the last batch in less than 1/10th the time. Finally, the revised theories produced by the incremental systems are actually less complex than BATCH's. The reason for these results seems to be that the promoter theory needs to be substantially generalized, primarily by deleting antecedents in various ways. Antecedent deletions performed during the initial batches are likely to be correct and, since the deletions are minimal and the theory needs significant generalization, EITHER rarely over-generalizes, even when given relatively few negative examples. Consequently, due to more deletions, the incremental systems produce simpler theories that are quite accurate and therefore process subsequent batches significantly faster. BATCH's stricter adherence to minimal change may actually be hurting it in this domain.

Overall, the results indicate that using a theory revision system in incremental batch mode can greatly increase efficiency without producing a significant loss in accuracy or simplicity. However, if revisions based on small amounts of data can be misleading, then the "inertia" of incremental theory refinement can prevent the system from recovering and actually decrease accuracy, simplicity, *and* efficiency.

Future Research

As previously mentioned, the effect of batch size on the performance of incremental batch theory revision needs to be explored. If the batch size is too small, changes will be based on very little evidence and will not be very accurate. With the FULL-MEM version of EITHER, setting the batch size to one would mean that induction would always be called with only one positive example during rule learning and one negative example during antecedent addition. With a no-memory approach, the system may even make changes that render it inconsistent with a large number of previously encountered training examples. With the NO-MEM version of EITHER, setting the batch size to one would always allow the system to delete rules or antecedents to fix the example if it is incorrectly classified and induction would never even be invoked. On the other hand, if the batch size is too big, the efficiency and responsiveness of incremental learning will be completely lost.

Finding an appropriate compromise between full-memory and no-memory is also an important problem. One approach is to retain only a subset of the previously seen examples, such as those that were incorrectly classified or those from the last few batches. Another approach is to retain statistics about previously seen examples rather than the examples themselves [Schlimmer and Fisher, 1986].

Perhaps the most important problem is preventing bad initial changes from "snowballing," as in the computer domain. EITHER treats all parts of the initial theory equally when determining a minimal change. Changes made to the theory during previous batches (perhaps based on very little evidence) are no more suspect than rules from the initial theory. An incremental theory revision system needs ways of representing its confidence in various portions of the theory and then weighting changes accordingly [Feldman *et al.*, 1991].

EITHER's results need to be compared to other theory revision systems [Towell and Shavlik, 1991; Ginsberg, 1990] used in incremental batch mode. The ability of incremental batch theory refinement to track concept drift also needs to be explored. Of course, since a full-memory system maintains consistency with all of the data, it is incapable of tracking concepts that actually change over time. Therefore, some form of partial-memory is required for this task.

Finally, incremental theory revision needs to be integrated with *knowledge integration* [Murray and Porter, 1989], which is concerned with incrementally incorporating new pieces of abstract knowledge (rules) rather than data. Many of the same issues, such as resolving contradictions, arise in the context of integrating new rules into a theory. A complete knowledge assimilation system should be able to incrementally incorporate an arbitrary mix of both rules and data.

Conclusions

Any comprehensive theory refinement system whose input and output theories are represented in the same language can be used to incrementally assimilate data into an evolving theory. This is done by continually feeding its revised theory back in as its initial theory. An incremental batch approach, in which a system incorporates a small batch of examples at each step, is the probably the most effective for existing refinement systems.

Experimental results with the EITHER theory refinement system demonstrate that this approach frequently increases efficiency without significantly decreasing the accuracy or the simplicity of the resulting theory. However, if the system produces bad initial changes to the theory based on only small amount of data, these bad revisions can "snowball" and result in an overall decrease in performance.

Acknowledgements

I would like to thank Dirk Ourston as the primary developer of EITHER and Michiel Noordewier and Jude Shavlik for providing the DNA theory and data.

References

[Clearwater *et al.*, 1989] S. H. Clearwater, T. P. Cheng, H. Hirsh, and B. G. Buchanan. Incremental batch learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 366–370, Ithaca, NY, June 1989.

[Cohen, 1990] William W. Cohen. Learning from textbook knowledge: A case study. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 743–748, Boston, MA, July 1990.

[Danyluk, 1991] A. D. Danyluk. Gemini: An integration of analytical and empirical learning. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 191–206, Harper's Ferry, W.Va., Nov. 1991.

[Feldman *et al.*, 1991] R. Feldman, A. Segre, and M. Koppel. Incremental refinement of approximate domain theories. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 500–504, Evanston, IL, June 1991.

[Flann and Dietterich, 1989] N. S. Flann and T. G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4(2):187–226, 1989.

[Ginsberg, 1990] A. Ginsberg. Theory reduction, theory revision, and retranslation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 777–782, Detroit, MI, July 1990.

[Matwin and Plante, 1991] S. Matwin and B. Plante. A deductive-inductive method for theory revision. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 160–174, Harper's Ferry, W.Va., Nov. 1991.

[Mooney and Ourston, 1989] R. J. Mooney and D. Ourston. Induction over the unexplained: Integrated learning of concepts with both explainable and conventional aspects. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 5–7, Ithaca, NY, June 1989.

[Mooney and Ourston, 1991a] R. Mooney and D. Ourston. Constructive induction in theory refinement. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 178–182, Evanston, IL, June 1991.

[Mooney and Ourston, 1991b] R. J. Mooney and D. Ourston. A multistrategy approach to theory refinement. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 115–130, Harper's Ferry, W.Va., Nov. 1991.

[Mooney and Ourston, 1991c] R. J. Mooney and D. Ourston. Theory refinement with noisy data. Technical Report AI91-153, Artificial Intelligence Laboratory, University of Texas, Austin, TX, March 1991.

[Murray and Porter, 1989] K. S. Murray and B. W. Porter. Controlling search for the consequences of new information during knowledge integration. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 290–295, Ithaca, NY, June 1989.

[Ourston and Mooney, 1990] D. Ourston and R. Mooney. Changing the rules: a comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 815–820, Detroit, MI, July 1990.

[Ourston and Mooney, 1991] D. Ourston and R. Mooney. Improving shared rules in multiple category domain theories. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 534–538, Evanston, IL, June 1991.

[Ourston, 1991] D. Ourston. *Using Explanation-Based and Empirical Methods in Theory Revision*. PhD thesis, University of Texas, Austin, TX, August 1991.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Reinke and Michalski, 1988] R. E. Reinke and R. S. Michalski. Incremental learning of concept descriptions. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence (Vol. 11)*. Oxford University Press, Oxford, England, 1988.

[Schlimmer and Fisher, 1986] J. C. Schlimmer and D. Fisher. A case study of incremental concept induction. In *Proceedings of the Fifth National Conference*

on Artificial Intelligence, pages 496–501, Philadelphia, PA, Aug 1986.

[Schlimmer and Granger, 1986] J. C. Schlimmer and R. H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–334, 1986.

[Towell and Shavlik, 1991] G. Towell and J. Shavlik. Refining symbolic knowledge using neural networks. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 257–272, Harper's Ferry, W.Va., Nov. 1991.

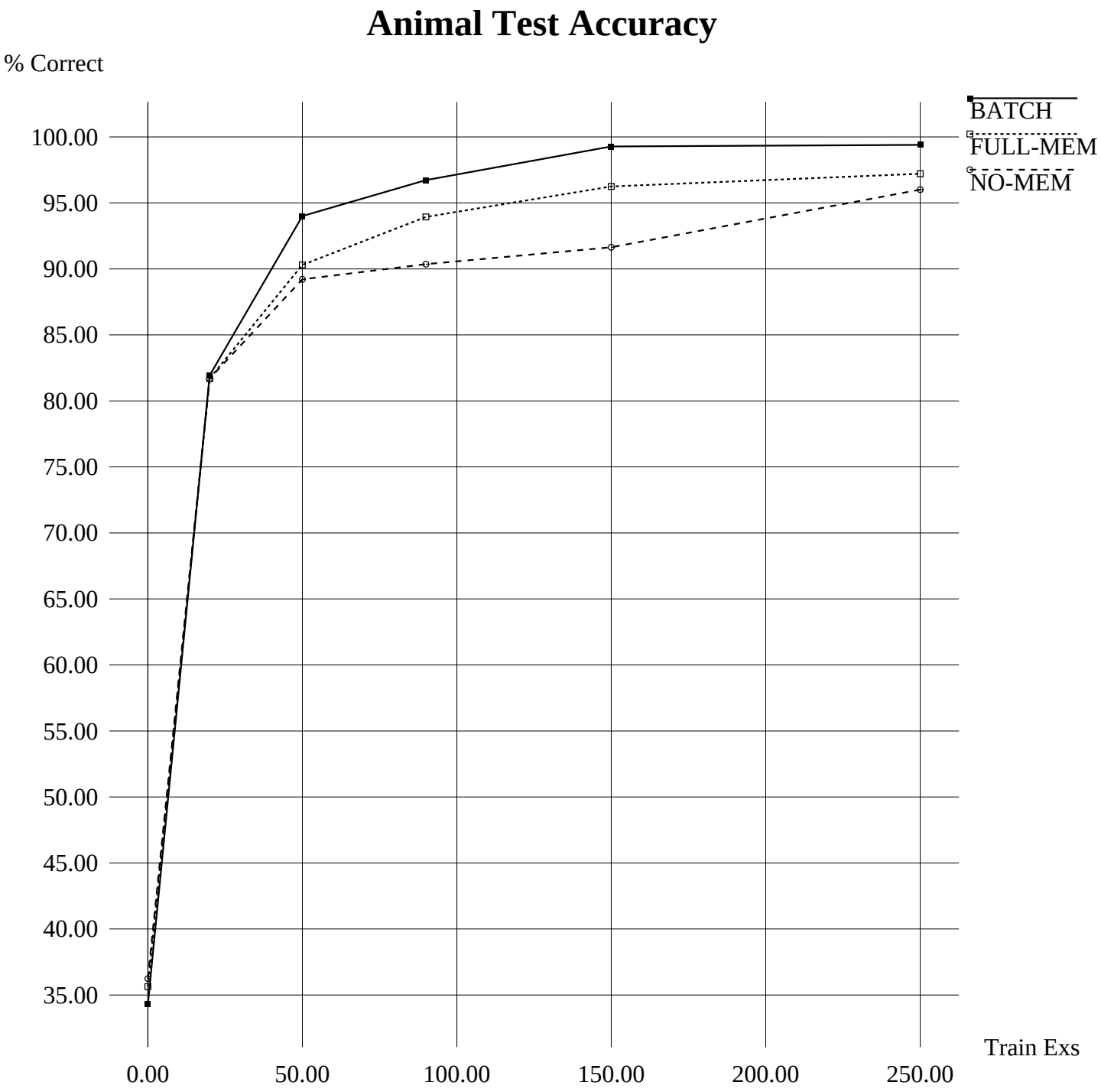
[Towell *et al.*, 1990] G. G. Towell, J. W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, July 1990.

[Utgoff, 1989] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2):161–186, 1989.

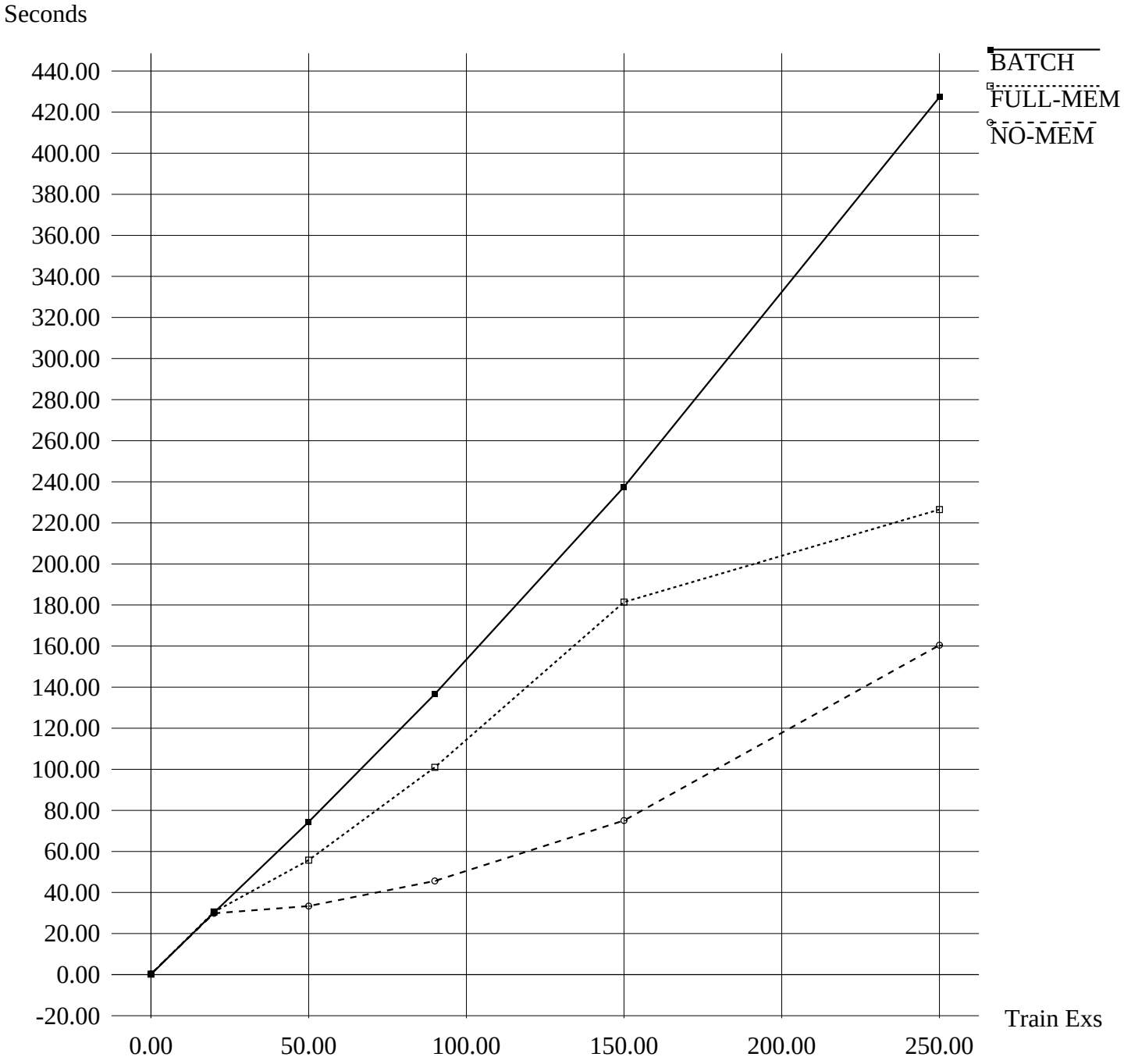
[Whitehall *et al.*, 1991] B. L. Whitehall, S. C. Lu, and R. E. Stepp. Theory completion using knowledge-based learning. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 144–159, Harper's Ferry, W.Va., Nov. 1991.

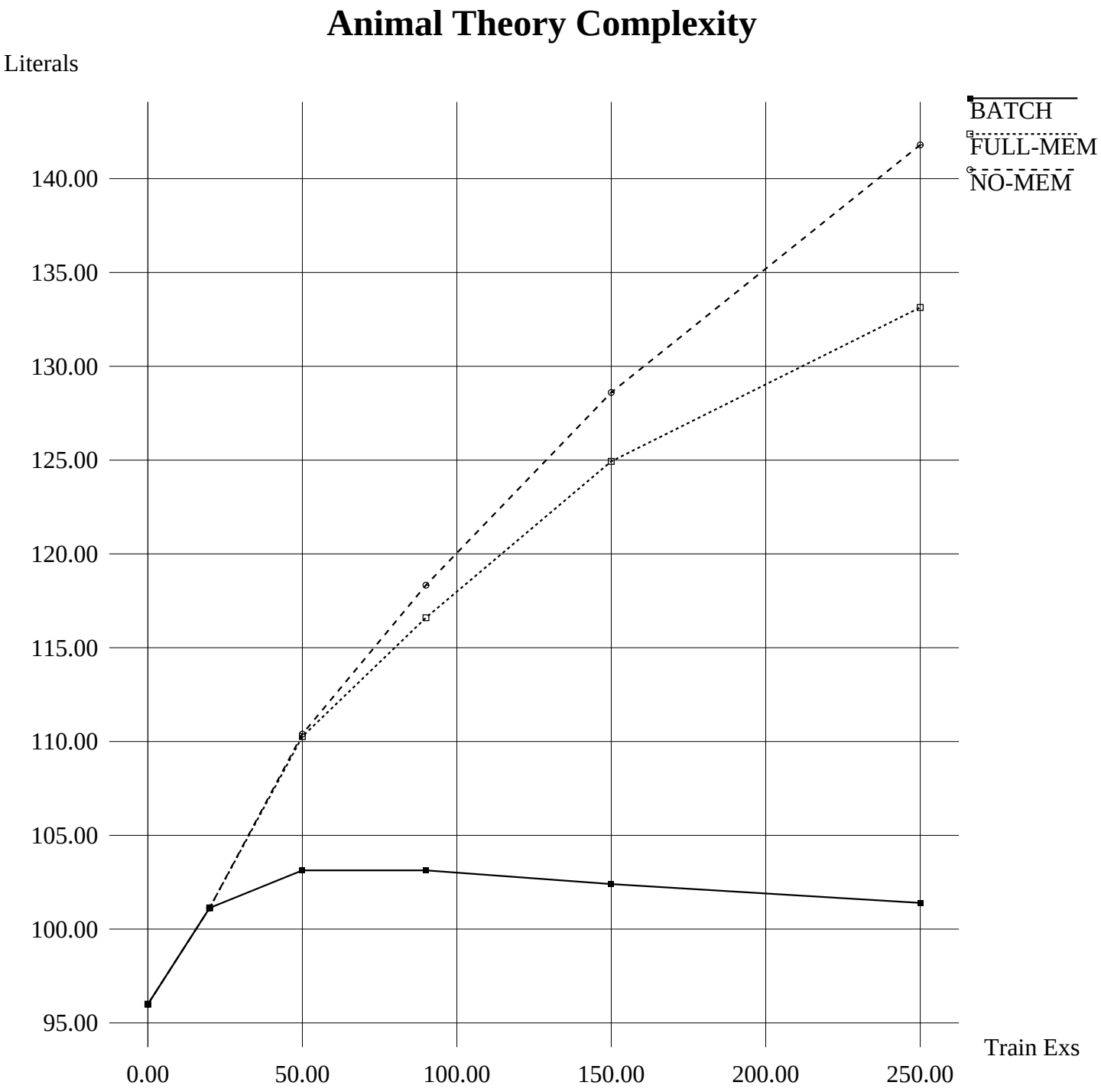
[Wilkins, 1988] D. C. Wilkins. Knowlege base refinement using apprenticeship learning techniques. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 646–651, St. Paul, MN, August 1988.

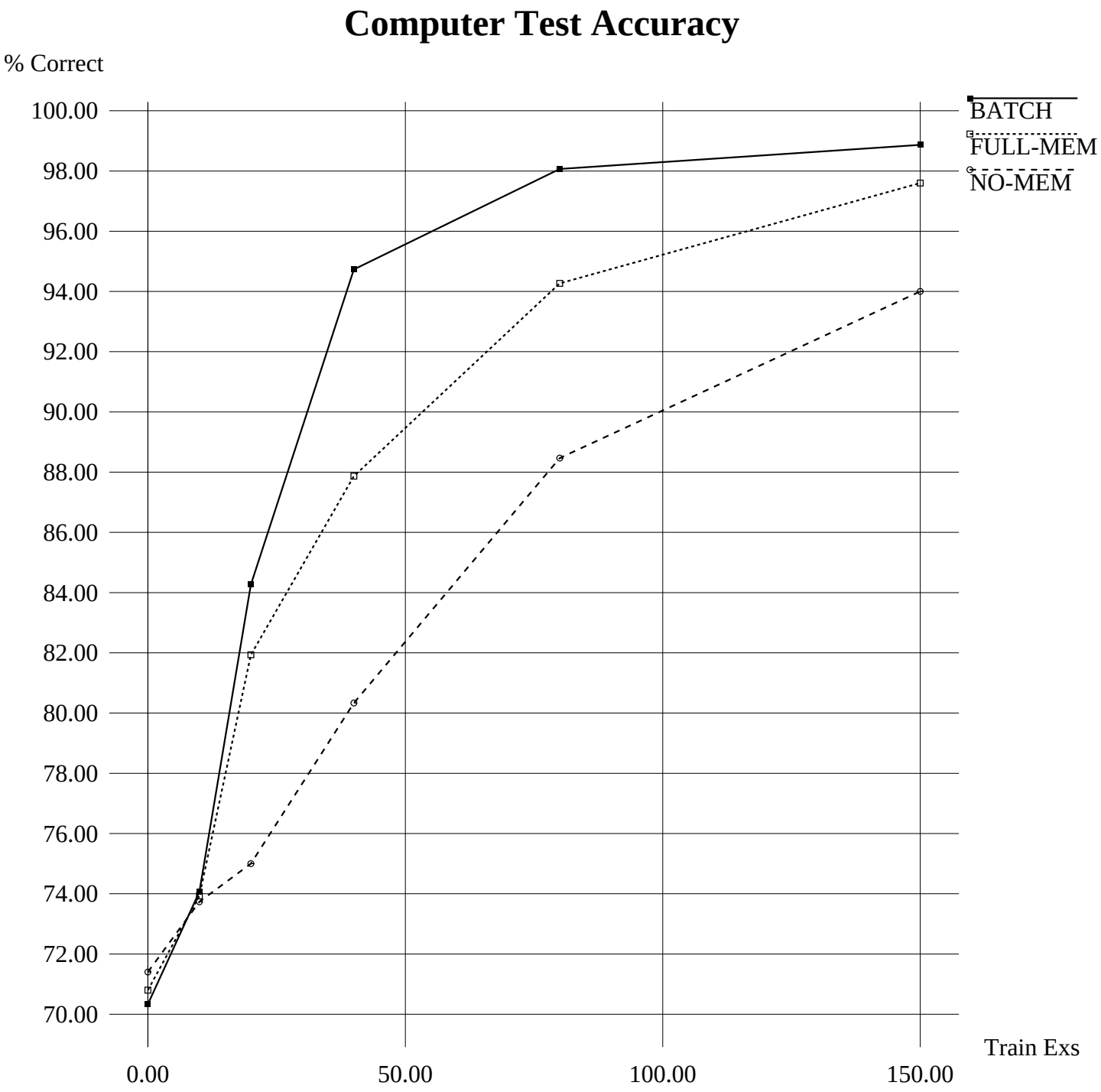
[Winston and Horn, 1989] P. H. Winston and B. K. P. Horn. *Lisp*. Addison-Wesley, Reading, MA, 1989.


Animal Test Accuracy
% Correct
100.00
95.00
90.00
85.00
80.00
75.00
70.00
65.00
60.00
55.00
50.00
45.00
40.00
35.00
0.00
50.00
100.00
150.00
200.00
250.00
Train Exs
BATCH
FULL-MEM
NO-MEM

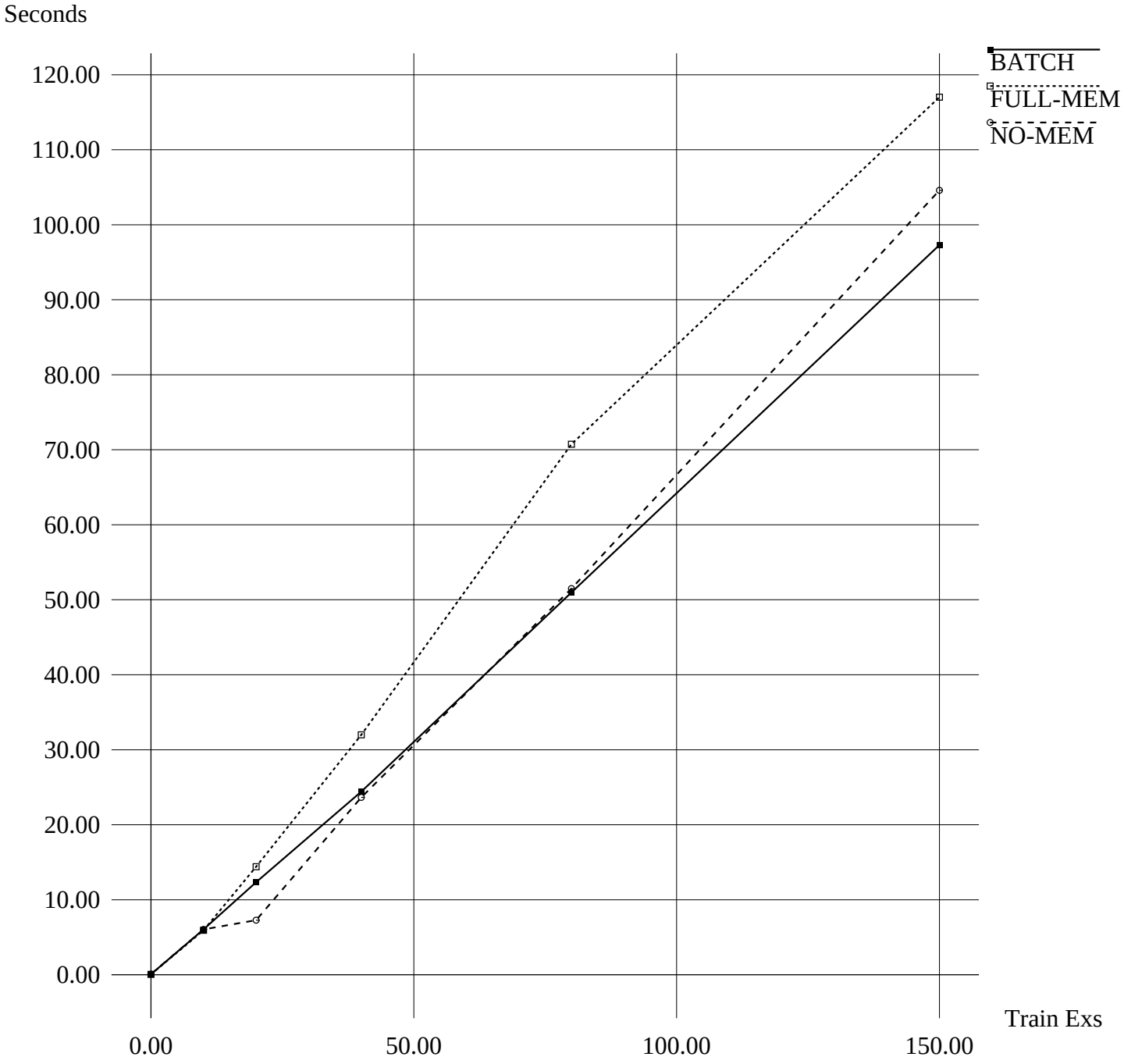
Animal Train Time



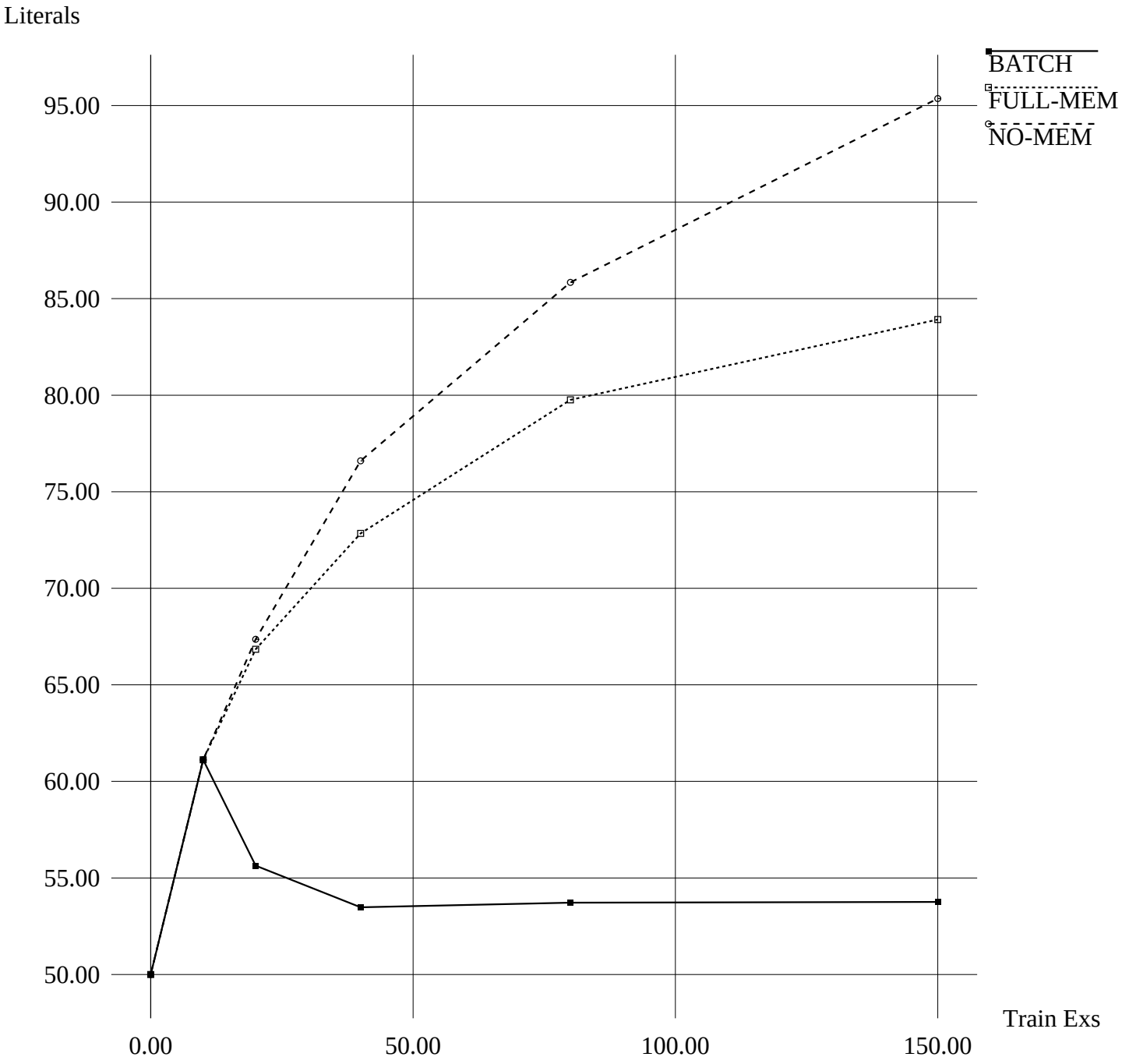

Animal Theory Complexity
Literals
BATCH
FULL-MEM
NO-MEM
140.00
135.00
130.00
125.00
120.00
115.00
110.00
105.00
100.00
95.00
0.00
50.00
100.00
150.00
200.00
250.00
Train Exs


Computer Test Accuracy
% Correct
100.00
98.00
96.00
94.00
92.00
90.00
88.00
86.00
84.00
82.00
80.00
78.00
76.00
74.00
72.00
70.00
0.00
50.00
100.00
150.00
Train Exs
BATCH
FULL-MEM
NO-MEM

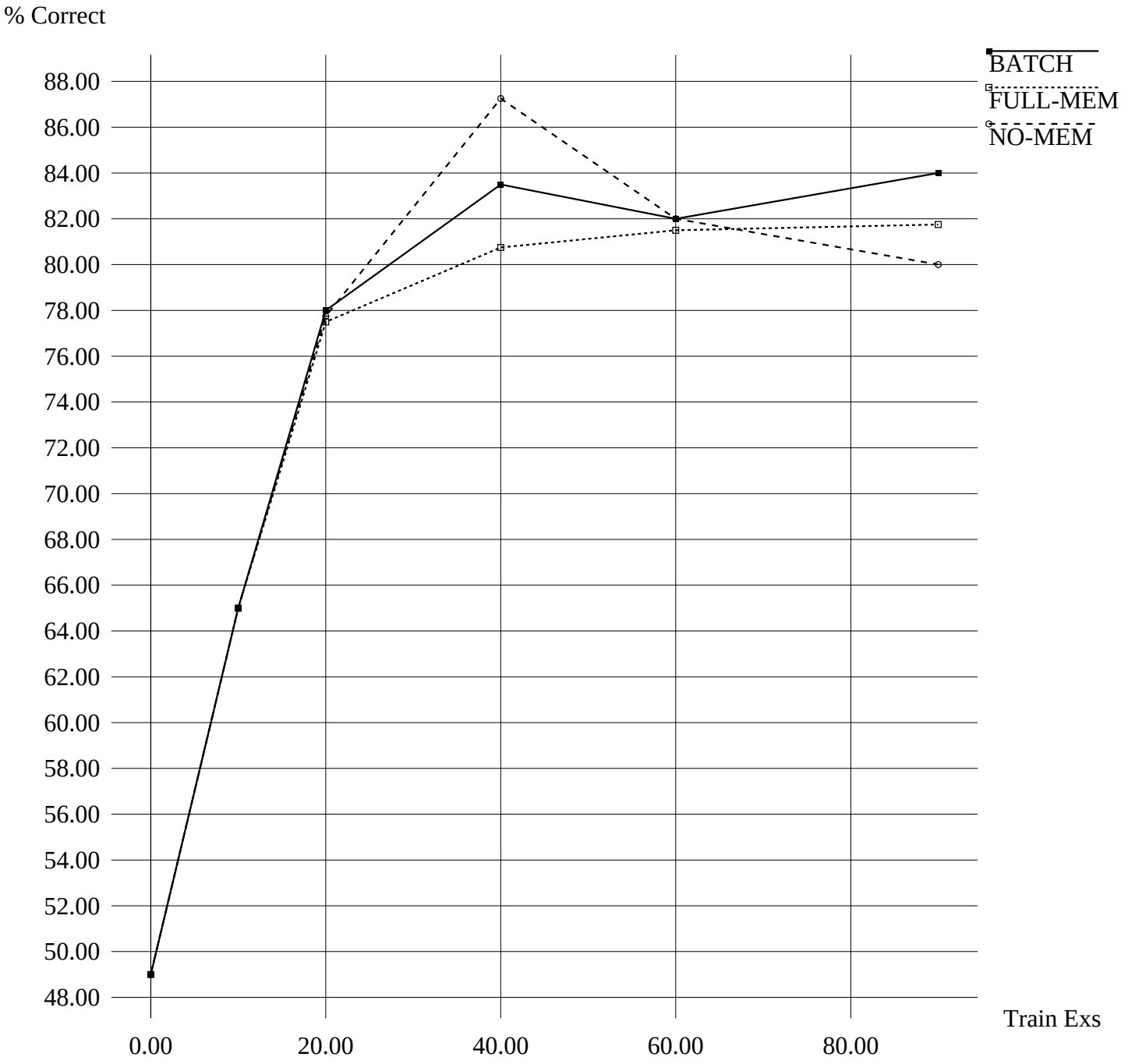
Computer Train Time

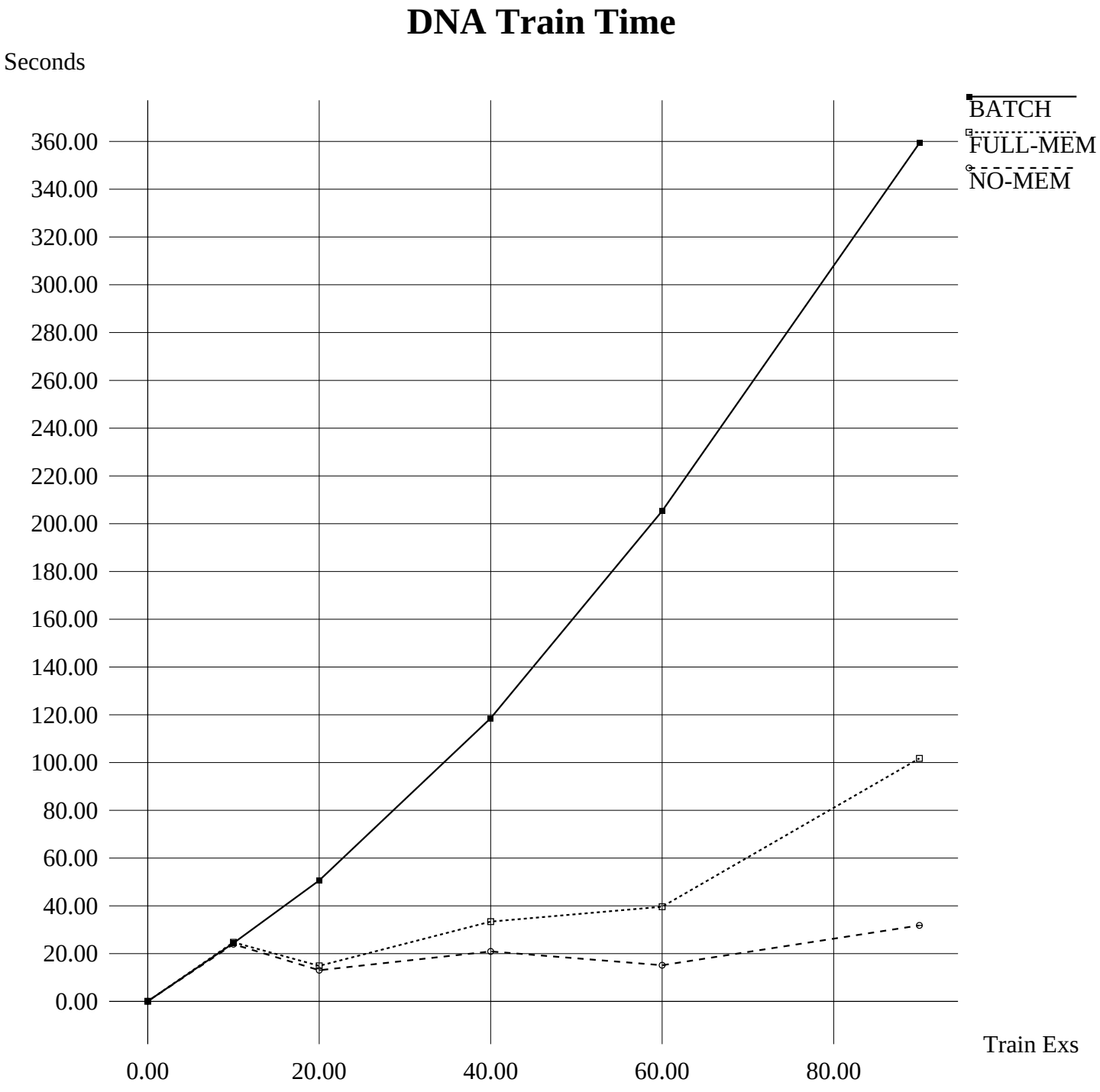


Computer Theory Complexity



DNA Test Accuracy




DNA Train Time
Seconds
360.00
340.00
320.00
300.00
280.00
260.00
240.00
220.00
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
0.00
20.00
40.00
60.00
80.00
Train Exs
BATCH
FULL-MEM
NO-MEM

DNA Theory Complexity