Dynamic Learning Bias Selection

Christopher J. Merz
Information and Computer Science Department
University of California, Irvine
Irvine, CA 92717
cmerz@ics.uci.edu
(714)824-3491

Abstract

Determining the conditions for which a given learning algorithm is appropriate is an open problem in machine learning. Methods for selecting a learning algorithm for a given domain or for a portion of the domain have met with limited success. This paper proposes a new approach to predicting a given example's class by locating it in the "example space" and then choosing the best learner(s) in that region of the example space to make predictions. The regions of the example space are defined by the prediction patterns of the learners being used. The learner(s) chosen for prediction are selected according to their past performance in that region. This dynamic approach to learning bias selection is compared to other methods for selecting from multiple learning algorithms.

1 Introduction

Determining the conditions for which a given learning algorithm is appropriate is an open problem in machine learning. Methods for selecting a learning algorithm for a given domain (e.g. Aha, 1992; Breiman, et al, 1984) or for a portion of the domain (Brodley, 1993 and 1994) have met with limited success. This paper proposes a new approach which dynamically selects a learning algorithm for each test example by locating it in the "example space" and then choosing the best learner(s) in that part of the example space for prediction. The regions of the example space are formed by the observed prediction patterns of the learners being used. The learner(s) chosen for prediction are selected according to their past performance in that region which is defined by the "cross-validation history."

This paper introduces DS, a method for the dynamic selection of a learning algorithm(s). DS has been evaluated on several real-world domains and frequently outperforms a cross-validation algorithm for selecting a learning algorithm and occasionally outperforms the single algorithm with the best test accuracy.

The paper begins by discussing the limitations of previous work (Section 2) and giving motivation for the DS algorithm (Section 3). Section 4 describes the method for building a "cross-validation history" for a collection of learning algorithms which is then used in the DS algorithm for making predictions on novel examples (Section 5). Experiments comparing DS to two other methods for selecting a learning algorithm(s) are described and analyzed in Sections 6 and 7. Future research issues in dynamic bias selection are outlined in Section 8. Finally, the conclusion section provides a summary of the work.

2 Related Work

A common method for deciding which classification algorithm to use on a given domain is cross-validation (Breiman, et al, 1984) where the examples are divided randomly into v (approximately) equal partitions or folds. Each algorithm being evaluated is then trained on $v-1$ partitions and tested on the remaining partition v times. For that domain, the appropriateness of each algorithm's learning bias is judged by the corresponding average cross-validation accuracy and the algorithm with the best accuracy is chosen for that domain.

There are a couple of limitations to this approach. One is that a *single* learning algorithm is selected as the "appropriate" one for an entire domain. An alternative we explore here is to select a learning algorithm(s) for a given example. Another potential drawback to the cross-validation approach is that several algorithms may have the same cross-validation accuracy but may give different classifications for many examples. Which one is really the "best" if they are not classifying the same examples correctly? Ideally, we would like to use these patterns of predictions to identify regions in the example space where each learning bias is more (or less) accurate. This paper addresses both of these limitations by considering the actual prediction patterns of a set of learning biases rather than a summary of those predictions (i.e., accuracy) to dynamically select the algorithm(s) with the best performance history for similarly classified training examples.

Another approach for selecting the best algorithm for a given domain is given by Aha (1992) where rules are derived which characterize the conditions under which various learning algorithms do well based on certain high-level properties of a domain (e.g., number of examples, number of attributes, number of classes, types of attributes, class and attribute noise). However, this method relies on having a good characterization of the domain, which is often difficult to obtain. DS emphasizes the selection of a learning algorithm at the example level rather than at the domain level since it avoids the need for characterization rules by instead considering each learning algorithm's past performance on "similar" examples from *that* domain.

Brodley (1993) proposes a knowledge-based approach to building hybrid decision structures by using heuristics to select the best algorithm at a given stage of learning. These heuristics are created from practical knowledge about how to detect when a generalization is a good fit or when to switch to a different search bias during the course of learning. Conditions in these rules include checks for certain pathologies in the learning process such as how often features are tested. Also, they use information about the domain such as the ratio of the number of examples to the number features. The development of these heuristics is a difficult ad hoc manual process based on the practitioner's past experience. DS avoids this process by building a collection of models and dynamically choosing which one(s) to use based on past performance in the given region of the domain.

Kwok and Carter (1990) select several "good" models (i.e., rules/trees with high posterior probabilities) manually and average their predictions. This avoids the difficult problem of calculating posterior probabilities for rules (Buntine 1989). The two main drawbacks to this approach are the manual derivation of the models, and the assumption that the majority prediction is always the best one to use. Our approach builds the models automatically and retains models which may not have high posterior probabilities, but still may be useful in certain regions of the example space.

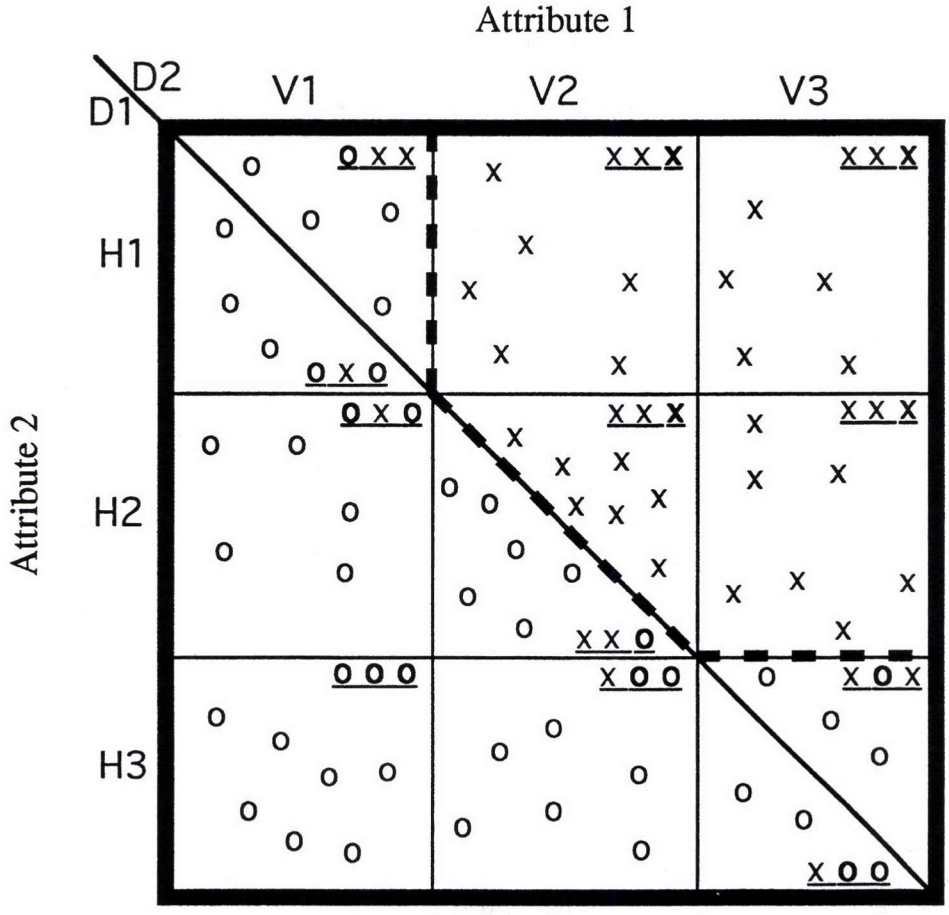

Figure 1. Dividing up the example space by response patterns.

3 Motivation

The main goal of this research was to take a collection of learning algorithms and determine their strengths and weaknesses within a given domain. The approach taken was to characterize the example space in terms of the performance of each algorithm. Given this information, novel examples can be placed in the example space and the strongest learner(s) for that region can be referred to for a prediction.

To characterize the example space, we simply use the patterns of predictions from the constituent learners to define regions in the example space. For example, Figure 1 shows how the models built by three learning algorithms can be used to create these regions. Model V is made up of two vertical splits on Attribute 1 with three "leaves"; V1, V2, and V3. Model H has two horizontal splits on Attribute 2 with three leaves; H1, H2, and H3. And model D has one split which is an oblique split on both attributes forming two leaves; D1 and D2. The training examples are plotted according to their respective values for Attribute 1 and 2, and the symbol used to plot each example is its class (i.e., "X" or "O"). The leaves of the three models divide the training examples up into twelve areas. Each area has an underlined triplet showing the prediction of V, H, and D, respectively. These triplets form seven distinct "regions" characterizing the example space (i.e., OXX, XXX, OXO, XXO, OOO, XOO, and XOX).

Although Figure 1 is a contrived example, it does show the potential utility of having this characterization of the example space. The bold dashed line highlights the learned

boundaries we would like to use to partition the examples. For regions near the left side of the top leg of the boundary, we would like to rely on model V's predictions more. Model D's prediction would be the preferred for the diagonal leg of the boundary, and Model H would be the preferred predictor for regions just below the right leg of the boundary. The models are bolded in regions where they are more likely to predict accurately. Regions further from the boundary may have more than one "reliable" predictor where there is more agreement between the models.

Next, a way of determining which learner(s) are superior within a given region was needed. Since most learners are capable of attaining high training accuracies (i.e., decision trees can continue to partition until purity or inconsistency within each partition is reached), the strengths of the learners being used may not appear all that different for the examples of a training partition. To obtain a more robust evaluation of how well each learner does on each training example, several cross validation runs were done on the training data. As each training example became a "test" example in these cross validation runs, the accuracy of each learner was recorded to accumulate a history of how well each learner classifies each example.

Returning to the example in Figure 1, this approach would show that the training examples which fell into the OXX region tended to be more accurately classified by learner V than the other two learner. Thus, we would defer to model V for test examples which produced an OXX prediction pattern. The next two sections elaborate on how this "cross-validation history" is built and how it is used.

4 Generating a Cross-Validation History for DS

The DS method relies on a **cross-validation history** to decide which algorithm(s) to use for classifying a new (test) example. This history is generated from the examples in the training partition. Given a set of training examples we form the history by building two m by n matrices where m is the number of training examples and n is the number of learning algorithms. The first is the **response matrix** which contains the predictions of the models built from the training data on the training data (i.e., the (i,j)th cell contains model j's prediction for training example i).

The second matrix, the **performance matrix**, contains the number of times each model was correct when that training example appeared as a test example in a cross-validation run on the training partition. That is, k v-fold cross validations are run for each model on the training partition so the (i,j)th cell in the performance matrix is the number of times (out of k) that example i was correctly predicted by model j when it appeared as a test example in a cross-validation run.

Using the models built from the training data we can get a row vector of model responses for a single test example. This row vector can be compared to the row vectors for the training examples in the response matrix. The set of row vectors with the most similar[1] response patterns then represent a region defined from the perspective of the models built. The corresponding rows in the performance matrix are then used to determine the accuracy of each of the models for the region defined by the examples with similar response patterns. Section 5 describes the algorithm.

5 The DS Algorithm

The algorithm for dynamically selecting models, DS, for a given test example is described in Figure 2. The first step is to let all of the learned models classify the test example to form

[1] We use the number of matched responses to measure similarity.

a row-vector of responses, $\mathbf{R_{EX}}$. In step 2, the "closest" rows are all the rows tied with the highest level of agreement with $\mathbf{R_{EX}}$. The local accuracy computed in step 3 is calculated by averaging the performances only on the rows selected in step 2. The algorithm with the highest local accuracy is then selected in step 4. If a tie occurs, the algorithms having the highest local accuracy vote with equal weighting.

Given: **R:** An m X n **response matrix** where
m = the number of examples and
n = the number of models
P: An m X n **performance matrix**
EX: A test example
Return: C, the predicted class of **EX**

1. Get row vector of responses, $\mathbf{R_{EX}}$, by polling each of the n models for **EX**.
2. Find set of "closest" row(s) to $\mathbf{R_{EX}}$ in **R**.
3. Use corresponding rows in **P** to compute local accuracy for each model.
4. Let models tied with highest local accuracy vote to choose **C**.

Figure 2. The DS classification algorithm.

6 Experimentation and Analysis

For our experimentation, DS used 24 variations of the OC1 program (Murthy, et. al., 1993) as its different learning algorithms. The different entropy measures (6), pruning options (off/on with 20% of training data for pruning), and types of splits (univariate/multivariate) combined for 24 unique configurations of OC1. All other parameters had the default settings. In forming the history, 3-fold cross-validation was done ten times.

Three domains with all numeric attributes (Glass, Iris, Breast Cancer Wisconsin, and Liver Disorders) were chosen from the UCI repository for evaluation because OC1 is limited to datasets with numeric features. The Glass dataset consists of 214 examples with nine attributes describing forensic information of a glass sample. The class attribute specifies ten possible glass types. The Iris dataset contains 150 examples, each with four numeric attributes describing an Iris flower's sepal and petal measurements, and a class attribute specifying the three types of Iris flowers. In the Breast Cancer dataset of 470 examples, there were nine attributes describing a tumor, and the class attribute specifying whether it was malignant or benign. For the Liver Disorders dataset, there are five blood test attributes and one alcohol consumption attribute and a class attribute specifying whether a disorder existed.

DS was compared to two other methods for selecting learning algorithms. The Select-All Majority (SAM) method weighs each algorithm's prediction equally and returns the most frequent prediction as its prediction. The cross-validation majority (CVM) method returns the prediction of the algorithm having the highest cross-validation accuracy in the performance table, and, in the case of ties, returns the majority vote of the tied algorithms. Tables 1 through 4 report the test accuracies for each methods for each domain where 25% of the examples were held back for testing and various numbers of training examples were used. Each cell entry is an accuracy averaged over 30 runs. Bold cell entries in the SAM and CVM rows indicate a statistically significant difference between that method and DS[2]. Also included in the tables is a row reporting the best individual model's performance. The

[2] Using a paired two-tailed t-test at the .05 confidence level.

number in parentheses next to these entries is the corresponding configuration of OC1 which had that performance and is meant to track which configuration is best as the number of examples changes.

The Glass domain results given in Table 1 show a significantly better performance for DS over SAM with fewer examples and for CVM for larger example sets. DS has the best a performance overall for all example set sizes. The best individual model changes for each training set size, thus no one configuration is doing well across the learning curve. The performance of CVM is not as good as the best individual model because that configuration is not identified as the *unique* best model based on cross-validation performance if it is selected as one of the best models at all. Therefore, its predictions may be considered along with other models' predictions.

Method	Number of Examples		
	50	100	150
SAM	**54.01**	65.25	69.57
CVM	55.65	63.23	**67.12**
DS	58.90	65.25	70.48
Best Individual Model	58.62 (#9)	64.35 (#12)	68.09 (#10)

Table 1. Glass domain results.

Table 2 shows that DS does as well as or better than the other selection methods in the Iris domain. This difference is most prevalent with fewer examples. The best individual model configuration changes as the training size increases and consistently does the best. However, CVM fails once again to recognize the best individual learning algorithm.

Method	Number of Examples		
	25	50	100
SAM	88.22	94.66	95.33
CVM	89.55	94.11	95.66
DS	89.55	95.00	95.33
Best Individual Model	90.66 (#11)	95.77 (#2)	96.45 (#2)

Table 2. Iris domain results.

In Table 3, we see that for the Breast Cancer dataset DS consistently outperforms CVM. However, the SAM method maintains a slight (1% or less) but significant edge over DS. One possible explanation for this is that the majority of OC1 configurations are well suited for this domain giving SAM a consistent edge. Whether the SAM approach is robust in general will be tested in future research where heterogeneous constituent learners will be used and are less likely to be in agreement.

Method	Number of Examples		
	100	200	300
SAM	**93.65**	**94.59**	**95.50**
CVM	91.95	93.38	94.16
DS	92.61	93.79	94.81
Best Individual Model	92.37 (#17)	94.08 (#17)	94.51 (#18)

Table 3. Breast Cancer domain results.

Performance in the Liver Disorders domain is similar to the Breast Cancer domain. DS does better than CVM and significantly worse than SAM. The performance of SAM on the last two domains led to further experimentation discussed in the next section.

Method	Number of Examples	
	100	200
SAM	**67.02**	**70.25**
CVM	64.42	67.37
DS	64.65	68.62
Best Individual Model	65.19	68.75

Table 4. Liver Cancer domain results.

A couple more comments are in order for these results. First, one can argue that the CVM method could do better with a greater number of folds and/or runs. However, in at least two of the domains, even if CVM were to choose the best model configuration, DS would consistently do as well or better. Second, for two of the domains evaluated (Glass and Iris), DS has better relative performance when fewer examples are available.

7 Further Experimentation

The significant edge that SAM has for the Breast Cancer and Liver Disorder domains in the initial experimentation led to further experimentation and analysis to determine if DS would take advantage of SAM if it were also a constituent learner. Table 5 shows that DS maintains its edge in the glass domain indicating that the inclusion of a learner which does worse for that domain does not interfere with DS's performance. Table 6 shows that DS is capable of taking advantage of SAM to narrow the difference between the accuracies of the two approaches to an amount which is no longer statistically significant.

Method	Number of Examples		
	50	100	150
SAM	**55.26**	**65.25**	**69.60**
CVM	**56.80**	**63.13**	**67.69**
DS	62.42	70.71	72.51
Best Individual Model	**58.34**	**65.25**	**69.50**

Table 5. Glass domain results.

Method	Number of Examples	
	100	200
SAM	67.02	70.25
CVM	64.71	70.28
DS	65.83	69.07
Best Individual Model	67.02	70.25

Table 6. Liver Cancer domain results.

8 Future Research

A number of ideas remain unexplored in dynamic bias selection. The main thrust of the next phase of research will be to do further evaluation on a more heterogeneous suite of

constituent learners (i.e., CN2 (Clark and Niblett, 1989), C4.5 (Quinlan, 1993), Backpropagation (Rumelhart, et. al., 1986), AUTOCLASS (Cheeseman, et. al., 1988), PEBLS (Cost and Salzberg, 1993), etc.). The inclusion of such learners will broaden the number of domains which can be evaluated and allow for a more direct comparison between DS and other methods for learning bias selection such as MCS (Brodley, 1993 and 1994). In addition, this broader range of biases will test the robustness of SAM.

Another future research issue is that of dynamic *weighting* rather than dynamic selection. The current approach can be viewed as a restricted weighting scheme where the weights are either zero or one. Methods for weighting the model's votes need to be developed and compared to DS to see how performance is effected.

Also, DS is unique in its handling of multiple models because it attempts to select dynamically an appropriate learner(s) for a subset of the example space rather than the entire example space as in cross validation. Currently, the example space is divided up according to the responses of each of the learners. An interesting problem can occur when this approach to dividing up the example space results in different areas of the example space having the same response pattern. Figure 3 shows a modified version of Figure 1 with the same three models, V, H, and D, but the center square has two less X's. This leads to three areas having the response pattern OXX (for models V, H, and D, respectively). But in one area we would like to rely on model V and in the other two we would like to rely on model D. This problem can be overcome by introducing another level of granularity by labeling the areas of the example space according to the rule or leaf of each model which made the prediction. For example, in the top right region, leaf 3 of model V predicts an X, leaf 1 of model H predicts an X, and leaf 2 of model D predicts an X. Obviously, this approach to dividing up the example space can lead to many more regions which may not always be useful or necessary. Methods for (dynamically) choosing the appropriate granularity of the example space characterization need also to be investigated. One approach might be to choose the granularity according to a confidence measure of the models selected in each characterization.

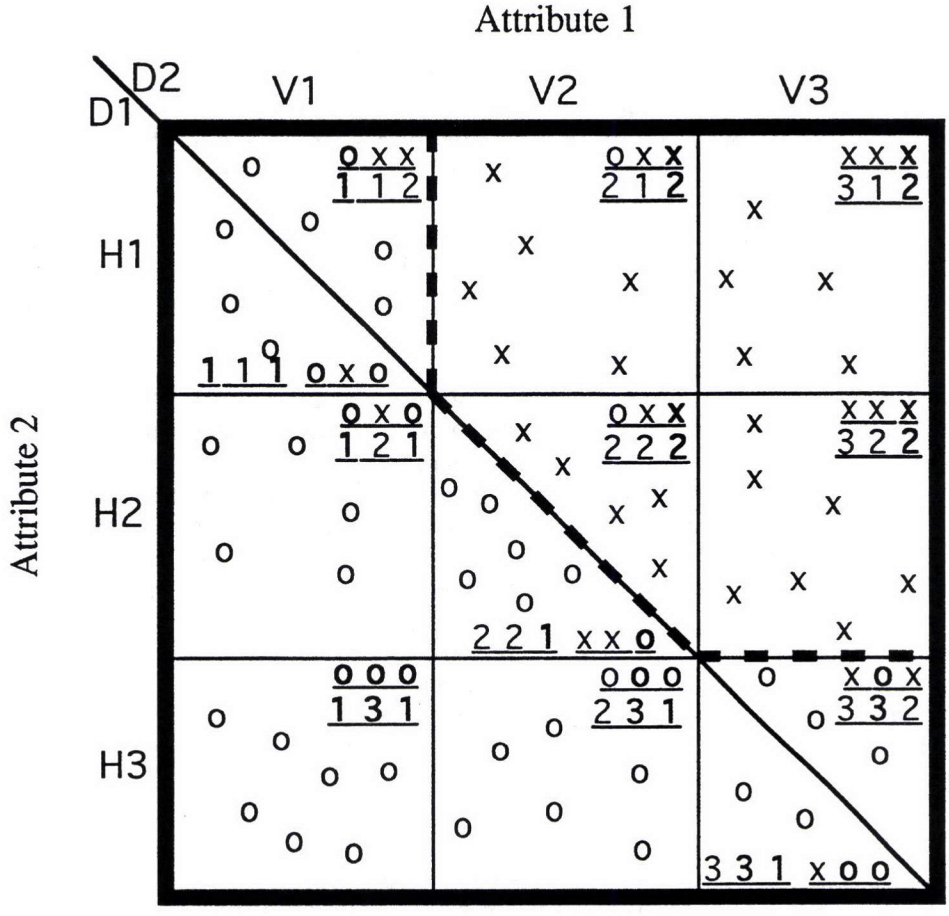

Figure 3. Dividing up the example space by leaf/rule label patterns.

9 Conclusion

A method for dynamically selecting a learning bias, DS, based on past prediction patterns was described. This approach differs from previous work by avoiding ad hoc construction of heuristics for matching learning algorithms to domains. The experiments conducted on four real-world domains demonstrate the synergistic effect of dynamic bias selection as DS did as well and frequently did better than any individual learning bias and a cross-validation algorithm. Although this trend was not always statistically significant, it was consistent for varying training set sizes while the best individual learner changed. This indicates that even an omniscient single learning bias selector would not do as well on these domains. Plans for the expansion of the method include the incorporation of more (heterogeneous) learners, the experimentation on more domains, and the development of other methods for characterizing the example space.

References

Aha, D. W. (1992). Generalizing from case studies: A case study. *Machine Learning: Proceedings of the Ninth International Conference* (pp. 1-10). San Mateo, CA: Morgan Kaufmann.

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and Regression Trees*, Wadsworth, Belmont, CA.

Brodley, C. E. (1993). Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection. *Machine Learning: Proceedings of the Tenth International Conference* (pp. 17-24). San Mateo, CA: Morgan Kaufmann.

Brodley, C. E. (1994). Recursive Automatic Bias Selection for Classifier Construction. To appear in the special issue of *Machine Learning* on "Bias Evaluation and Selection." Kluwer Academic Publishers.

Buntine, W. (1989). Decision tree induction systems: a bayesian analysis. *Uncertainty in Artificial Intelligence 3* (pp. 109-127). North-Holland, Amsterdam.

Cheeseman, P., Kelly, j., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). AUTOCLASS: A Bayesian Classification System. *Proceedings of the Fifth International Conference on Machine Learning*. San Mateo, CA. Morgan Kaufmann.

Clark, P., and Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning 3* (261-284). Kluwer Academic Press.

Cost, S. and Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning 10* (57-78). Kluwer Academic Press.

Murthy, S., Kasif, S., Salzberg, S., & Beigel, R. (1993). OC1: Randomized induction of oblique decision trees. In *Proceedings of AAAI-93* (pp. 322-327). Washington DC. AAAI Press.

Kwok, S. W., & Carter, C. (1990). Multiple decision trees. *Uncertainty in Artificial Intelligence 4* (pp. 327-335). North-Holland, Amsterdam.

Quinlan, R. (1993). *C4.5 Programs for Machine Learning*. San Mateo, CA. Morgan Kaufmann.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Interior Representation by Error Propagation. *Parallel Distributed Processing 1*(318-362). Cambridge, MASS., MIT Press.