Deciding Weights for Nearest Neighbour Algorithms using Decision Trees

Charles X. Ling* John J. Parry Handong Wang
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract

Nearest Neighbour (NN) algorithms utilize a distance function to determine the classification of testing examples. The attribute weights in the distance function should be set appropriately. We study situations where a simple approach of assigning weights for NN algorithms using decision trees (Kibler & Aha, 1987; Cardie, 1993) does not work well, and design three new methods that also consider the attributes that do not appear in the tree, and the positions of the attributes in the tree. We test these methods thoroughly using artificially generated data sets and data sets from the machine learning repository. We identify several conditions under which our new methods are advantageous over the previous method.

Key Words: Nearest neighbour algorithms, Instance-based learning, decision trees, inductive learning from examples.

*Contact author. Tel: 519-661-3341. Fax: 519-661-3515. E-mail: ling@csd.uwo.ca.

1 Introduction

Decision tree induction learning algorithms, such as C4.5 (Quinlan, 1986, 1993), and nearest neighbor (NN) algorithms (Cover & Hart, 1967; Dasarathy, 1991) are two popular machine learning programs. They take very different approaches to generalization. C4.5 builds a small decision tree which consists of a sequence of logical decisions based on the attributes. NN algorithms originated in the field of pattern recognition (Cover & Hart, 1967, 1968), and are recently studied and applied in classification tasks in machine learning (e.g., (Stanfill & Waltz, 1986; Aha, Kilber, & Albert, 1991; Cost & Salzberg, 1993; Ling & Marinov, 1993; Ling, 1994; Schmidt & Ling, 1994))[1] They stores all (or selected) training examples and utilizes a similarity function to divide the instance space into regions of different classes.

C4.5, an improved implementation of ID3 (Quinlan, 1986), builds decision trees by employing a simple divide and conquer strategy. It first chooses an attribute as the current root, divides the training examples into subsets, and is recursively applied to the subsets, until all remaining examples belong to a single class. The choice of the attribute is based on the information gain, which estimates how much randomness is taken or absorbed by the attribute. C4.5 always chooses the attribute with maximum information gain[2] as the current root; such attributes tend to be most discriminative or informative for classification at that point. The heuristic usually results in a small decision tree.

The distance function of original NN algorithms takes the form:

$$dist(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

where x is the testing example, y is the stored training example, and x_i and y_i are the (normalized) values of the i^{th} attribute[3]. However, the above distance function treats every attribute with equal contribution, which for many cases is not ideal. Sometimes an attribute can be completely irrelevant to the classification, so it should not be included in the distance function. Therefore, a more realistic distance function is

$$dist(x, y) = \sqrt{\sum w_i \times (x_i - y_i)^2}$$

[1]NN algorithms are usually called instance-based learning (IBL) algorithms (Aha et al., 1991) in the machine learning community. We will use NN and IBL algorithms interchangeably in this paper.

[2]To balance different numbers of values among attributes, C4.5 actually uses a normalized gain called gain ratio (Quinlan, 1993). In this paper, the term information gain is the normalized information gain.

[3]If the attribute is symbolic, then $x_i - y_i = 0$ if $x_i = y_i$; 1 otherwise.

where w_i is a constant (to be determined) that weighs the role of the i^{th} attribute in the similarity function.

Setting the weights in the similarity function appropriately can improve the learning rate of NN algorithms, especially in domains with many irrelevant attributes (e.g., (Aha, 1989; Kelly & Davis, 1991; Cardie, 1993; Mohri & Hanaka, 1994; Ling & Wang, 1995)). More important attributes should have larger weights, while totally irrelevant attributes should have 0 weight. Kibler and Aha (1987) first presented a simple approach of combining C4.5 and NN algorithms. The method uses the presence and absence of attributes in the decision tree built by C4.5 on the same training examples to determine the weights in the NN similarity function. If an attribute is present in the decision tree, then its weight is 1; otherwise, its weight is 0. The method is very efficient since it only involves running C4.5 once over the training set. We call this method *binary-dtree* in this paper. Cardie (1993) independently rediscovered the same method, and showed that the method improves the predictive accuracy by a small amount in a lexical acquisition task. Cardie pointed out that a strategy of considering positions of attributes in the decision tree may work better, and the method needs to be tested on additional data bases (this is what we have done in this paper). Daelemans, Gillis, Durieux, and van den Bosch (1993) and Wettschereck and Dietterich (1995) presented an approach of assigning continuous weights of all attributes in NN algorithms simply by their information gain values. A few other methods that assign binary weights (Aha & Bankert, 1994; Caruana & Freitag, 1994; Skalak, 1994) and continuous weights (Aha, 1989; Kelly & Davis, 1991) for the NN algorithms have been studied.

In Section 2 we examine the method of (Kibler & Aha, 1987; Cardie, 1993) and extend it to three new methods of combining C4.5 into NN algorithms. We then thoroughly test all four approaches using artificial data sets and "real world" data sets from the UCI repository of machine learning databases in Section 3. We identify situations that our new methods outperform the previous method.

2 Three New Methods of Combining C4 and NN Algorithms

There are two reasons where the binary-dtree method (recall that binary-dtree is an NN algorithm that assigns attributes a weight of 1 if they are in the decision tree, 0 otherwise) may not work well. First, considering only attributes appearing in a decision tree constructed from a particular set of training examples diminishes the advantage of NN algorithms. At each point during the tree construction, other attributes also have information gain, but only the one with maximum information gain is chosen by C4.5. In addition, since C4.5 attempts to construct a small decision tree, the number of attributes used in the decision tree tends to be relatively small. However, the advantage of NN algorithms is that they can take partial contribution of many attributes into account by assigning continuous weights to those attributes. In the situation where the training sample is very small (comparing to the number of attributes), the decision tree constructed is often very small too, and thus, relevant attributes would not be included in the decision tree. The binary-dtree method would ignore completely those relevant attributes and thus can have even higher predictive errors than the original 1-NN.

The second reason is that the positions where attributes appear in the tree can reflect the relative importance of attributes, but are ignored in the binary-dtree method. Attributes at the upper level (i.e., close to the root) of the tree are usually more informative and more important than those at the lower level. In the situation where there are many irrelevant attributes, C4.5 may inevitably select some irrelevant attributes at lower levels of the tree because of the sample variation. These attributes would be taken as important as other relevant attributes in the binary-dtree method.

To overcome problems that the binary-dtree method encounters in the two situations described above, we propose three new methods of using information gain to determine the weights of NN algorithms. The first method attempts to improve NN algorithms' performance for the first situation, the second method for the second situation, and the third one combines the two.

Rather than considering only attributes with the maximum information gain (i.e., those appeared in the tree), our first method considers information gain of *all* attributes at the top level; that is, information gain of all attributes based on the the whole training set (thus there is no need to construct the decision tree). It uses these information-gain values as the weights in the similarity function of NN algorithms directly[4]. Clearly, the attribute with maximum information gain is assigned a maximum weight, but other attributes can have some smaller effects in the similarity function as well, rather than being completely ignored. Thus we call this first method *info-top*.

[4]This method is similar to the one studied by Daelemans et al. (1993). We independently rediscovered it.

Our second method takes into account the places that attributes appear in the decision tree (thus a decision tree is first constructed from the training examples). For each attribute, which may appear in several places in the tree, we take the sum of its information gain at each appearance multiplied by the percent of training examples classified by that attribute, as the weight for that attribute in the NN algorithms. For example, if an attribute appears three times in the tree, with information gain values 0.9, 0.8, 1.0, and with 40%, 20%, and 10% of training examples classified by the attribute respectively, then the weight of this attribute for NN algorithms is $0.9 \times 0.4 + 0.8 \times 0.2 + 1.0 \times 0.1 = 0.62$. Clearly, attributes at lower levels contribute less to their weights because the number of training examples they classify is smaller. Note, however, that this method, like binary-dtree, considers only (the information gain of) those attributes that appear in the tree. We called this second method *info-dtree*.

Our third method simply combines our first two methods. First, the information gain for all attributes is calculated (as in our first method info-top) at each decision node of the decision tree. Second, we take the sum of information gain of an attribute in a node multiplied by the percent of training examples classified by that node over all nodes (as in our second method info-dtree). We call this third method *info-all*.

3 Experimental Results

We perform extensive experiments comparing a total of 6 algorithms — C4.5, the original 1-NN algorithm, binary-dtree, and our three new methods — both on artificially created data sets and on data sets in the machine learning repository. C4.5 (Quinlan, 1993) is applied with the default parameter setting (i.e., $m = 2$) and the unpruned trees are used for the weight calculation. For 1-NN and the new NN methods, all training examples are stored, and only the closest training example is used to make predictions (i.e., the 1-nearest neighbour algorithm).

3.1 Artificial Data Sets

These data sets are created to test the two situations discussed in Section 2 in order to demonstrate the shortcomings of the previous NN methods. Because those data sets have clear geometry shape with no noise, it is actually possible to calculate theoretically the predictive errors for the 1-NN algorithm with given weights. Such theoretical results have been derived (Ling & Wang, 1995), however they are outside the scope of this paper.

As for each artificial data set, we consider several different sizes of randomly created training examples against 10,000 randomly created testing instances. For each training size, 5 training sets are created for the comparison. For each training set, each of the six methods is trained (on the same training examples), and tested on the the same testing set (of size 10,000). The average predictive errors of 5 runs are recorded in the tables.

Database that contains many irrelevant attributes.

This data set reflects the second situation discussed in Section 2 when binary-dtree does not work well: domains with many irrelevant attributes. Instances of this artificial data set are described by ten continuous attributes ranging from 0 to 10, but only two attributes define the classification, while the other eight are totally irrelevant. The two defining attributes describe five squares connected together as in Figure 1. Everything inside one of the five squares is positive, and everything outside is negative. The size of each square is $\sqrt{10}$, and therefore, there are overall 50% positive examples and 50% negative examples. Sizes of the training sets tested are 100, 200, 300, 400, 800, 1600, 3200, and 6400. The average predictive accuracies (over 5 runs) with 10,000 testing examples are reported in Table 1.

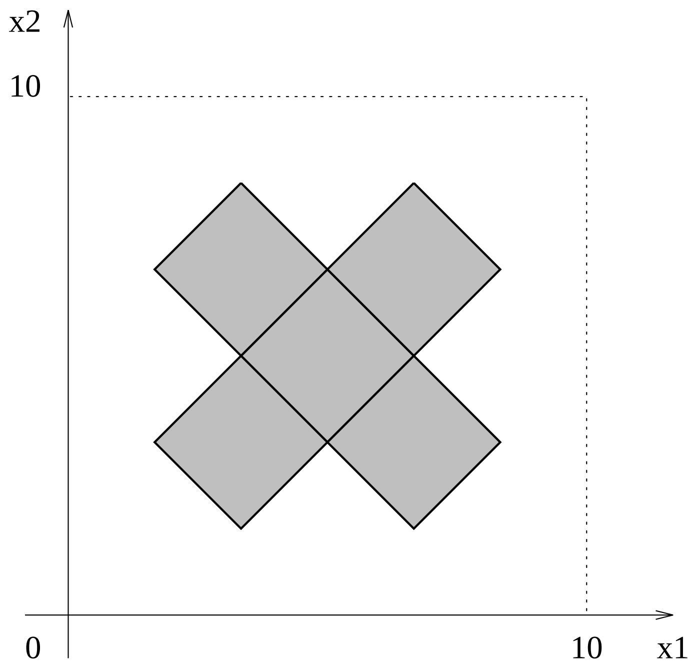

Figure 1: The classification region of the first artificial data set.

Clearly, 1-NN predicts poorly for tasks with many irrelevant attributes. The binary-dtree method performs poorly as well because C4.5 often selects some irrelevant attributes at the lower levels of the decision tree. C4.5 does not do well either in this case because the complicated classification boundaries for this data set are not parallel to any axis, posing difficulty for C4.5. Our new methods, info-dtree (which takes into account the position of attributes in the tree) and info-top (which takes into account the importance of attributes for the data set) have much lower testing errors than binary-dtree (an order of magnitude smaller) and than C4.5 (by as much as 50%). Our third method, info-all, seems to have overly considered irrelevant attributes (since all attributes are weighted for all nodes in the tree), and does not predict as well as C4.5.

Table 1: Percent of predictive errors on the crossing rectangles data set.

Training Size	C4.5	1-NN	binary-dtree	info-dtree*	info-top	info-all
100	20.8	42.6	35.5	18.7 (10.1)	22.6	31.0
200	18.8	41.3	37.9	13.5 (28.2)	14.0	26.1
400	14.8	39.1	37.7	8.8 (40.5)	9.0	21.6
800	10.2	37.1	33.2	5.8 (43.1)	5.5	16.5
1600	6.1	34.8	31.5	3.2 (47.5)	3.8	11.5
3200	4.5	32.4	30.0	2.3 (48.9)	2.2	8.2
6400	3.1	29.9	29.3	1.5 (51.6)	1.5	6.2

* The numbers in parentheses in this column are the percents of reduction in info-dtree's error rates over C4.5's error rates. We have conducted paired t-test between C4.5 and info-dtree, and info-dtree is significantly better than C4.5 ($p < 0.01$).

To further compare how the number of irrelevant attributes affect the predictive accuracy of C4.5, 1-NN, binary-dtree, and our methods, we ran the same experiment with 400 training examples (and 10,000 testing examples) except the number of irrelevant attributes is varied (from 0 to 16). See Figure 2 for the result of the comparison. It is clear that when there is no irrelevant attribute, all of the NN family methods have very similar predictive errors. However, info-dtree remains immune to the increasing number of irrelevant attributes, and is advantageous over the other methods.

Database that contains attributes of different importance.

This database has three continuous attributes ranging from 0 to 10, all of which contributes to the classification (thus, all attributes are relevant). The classification is defined by a three-dimensional block of size $9 \times 1 \times 9$. Since the lengths of the sides vary greatly, the attributes do

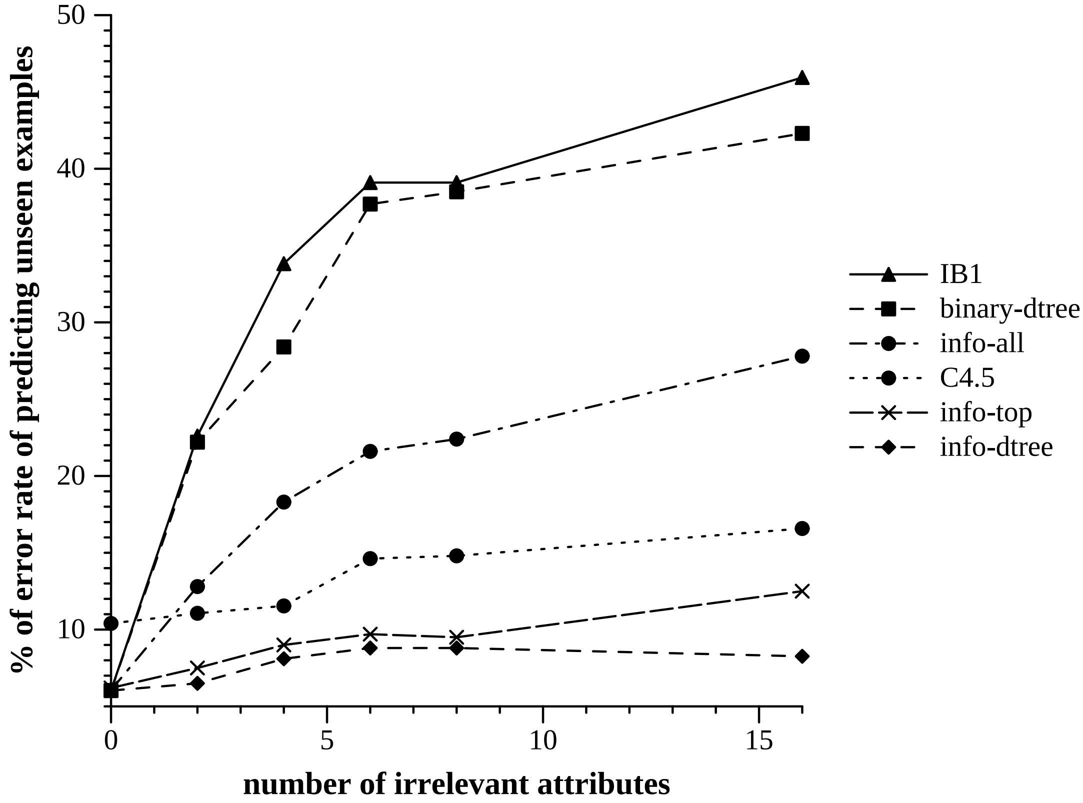

Figure 2: something here.

not equally contribute to the classification. If all of the classification boundaries are parallel to axes, C4.5 would construct a very small decision tree and thus have very low predictive errors. In order to demonstrate that it is still possible for our new NN methods to outperform C4.5 in the case of block regions, the block is placed in parallel to two axes but is rotated 10° around the third axis.

The testing accuracy shows that all of our methods outperform 1-NN which does not consider attribute weights significantly (by as much as 46%). Therefore, utilizing C4.5 for determining attribute importance improves the original 1-NN even when all of the attributes are relevant. In addition, our new methods, info-dtree and info-top, outperform C4.5 (which is the best of binary-dtree, 1-NN, and C4.5) by as much as 26%. See Table 2.

We can conclude the following from the results of the artificial data sets:

- When the domain has many irrelevant attributes, info-dtree performs the best.

Table 2: Percent of predictive errors on the block data set.

Training size	C4.5	1-NN	binary-dtree	info-dtree*	info-top	info-all
60	8.1	12.2	9.6	7.7 (4.9%/36.9%)	8.5	9.9
100	7.6	10.3	10.3	6.6 (13.2%/35.9%)	6.8	7.8
150	6.7	9.6	7.6	5.2 (22.4%/45.8%)	5.7	6.8
200	6.2	8.6	7.9	5.2 (16.1%/39.5%)	5.4	6.4
250	6.0	8.2	6.9	4.4 (26.7%/46.3%)	4.9	6.0
300	6.2	7.9	7.2	4.7 (24.2%/40.5%)	4.9	5.7

* The numbers in parentheses in this column are the percents of reduction in info-dtree's error rates over C4.5's and 1-NN's error rates. We have conducted paired t-test between C4.5 and info-dtree, and between 1-NN and info-dtree; info-dtree is significantly better than both of them ($p < 0.01$).

- If the domain has few or no irrelevant attributes, our new methods still have some improvements over 1-NN and binary-dtree methods.

3.2 Real World Data Sets

In the previous section we compared C4.5, 1-NN, binary-dtree, and new NN methods on artificially created data sets which we know how to bring up advantages of certain methods over the others. In this section we compare C4.5, 1-NN, and new NN methods on real world data sets, which normally do not have a clear geometric interpretation and may also contain noise. Fifteen data sets from the UCI machine learning Repository (Murphy & Aha, 1992) are selected randomly for comparison. Many of those data sets were used in other comparative studies of learning algorithms as well. To avoid using heuristic methods of handling unknown data, instances with unknown attribute values are removed. Each data set is then randomly split into 25%/75%, 50%/50%, and 75%/25% of training and testing samples, and each method is applied on the same training and testing sets 10 times. Only results of the average predictive errors on the testing sets with 25%/75% split are presented in Table 3; more detailed results of other splits can be found in (Parry, 1994).

Clearly, the results on real-world data sets are not as clear-cut as on the artificial data sets. As we expected, no single method outperforms other methods in all data sets — a now well-known law in machine learning (Schaffer, 1994). If some prior knowledge of the domains

Table 3: Comparison of our new NN methods with C4.5, 1-NN, and the binary-dtree method. Numbers are predictive error rates.

Data set	C4.5	1-NN	binary-dtree	info-top	info-dtree	info-all
Auto-cost (193/15/11)	28.9	25.1	30.9	24.0	29.1	23.7
Auto-symbolling (193/15/11)	53.5	44.2	47.3	46.2	47.9	43.1
Breast Cancer (675/0/10)	7.7	5.8	6.5	5.2	6.5	5.1
Heart Disease (297/5/8)	49.7	46.4	47.3	46.1	47.7	46.8
Diabetes (768/8/0)	30.5	31.5	31.8	30.4	30.8	30.9
Solar Flare (323/0/10)	14.0	16.8	15.5	19.0	16.2	17.3
Glass (214/9/0)	10.9	16.6	5.8	10.3	5.8	9.8
Image Segmentation (210/19/0)	23.8	19.8	19.0	15.4	17.9	16.5
Iris (110/4/0)	11.2	8.1	8.1	5.9	8.2	6.3
Liver Disorders (345/6/1)	39.7	42.6	41.8	41.7	40.5	41.6
Wine (178/13/0)	14.8	6.5	13.0	5.2	12.8	5.0
Zoo (101/1/15)	27.2	11.4	21.6	13.5	22.0	12.8
Soybean (266/0/35)	50.3	26.1	45.2	28.1	50.6	30.5
Credit card (653/6/9)	18.0	20.9	20.2	18.1	19.2	18.1
Mushroom (5644/0/22)	1.5	1.3	2.6	1.2	1.4	1.6

The data set column indicates the data set name, with the number of total examples, the number of continuous attributes, and the number of discrete attributes in parentheses.

and data sets is known, one can apply the appropriate method accordingly. We have identified the following situations where conclusions can be drawn after analyzing results in Table 3:

- For data sets with many irrelevant attributes, info-dtree performs the best. We can see this clearly from the Glass data set. The decision tree built from the whole training set (214 examples) uses only one attribute; therefore, info-dtree (and binary-dtree, since only one attribute is used in the tree) has the lowest predictive error than other methods.
- As discussed in Section 2, if the number of available training examples is too small, C4.5 constructs an overly simple decision tree, overlooking relevant attributes. This happens in five data sets (wine, auto-cost, auto-symbolling, zoo, and soybean) here. In the wine data set, for example, there are 13 continuous attributes but there are only 178 examples.

When using 25% for training, only about 45 examples are selected; a decision tree built on such small set of examples normally looks only two attributes. When this happens, the info-dtree and binary-dtree (which only consider attributes in the decision trees) both predict even worse than 1-NN. However, our new methods, info-top and info-all, normally perform better since they do not just consider attributes in the decision tree.

Note that the above two situations are not distinguishable from the data sets alone; in both situations the decision trees constructed are very simple, but attributes unused in the decision tree may or may not be irrelevant. However, the effect of the two situations on info-dtree is opposite (that is why we need prior domain knowledge). Therefore, if a real-world data set is somewhat between the two situations above, the improvement with our new methods may not be clear-cut and dramatic. However, we can see from Table 3 that whenever 1-NN has a lower predictive error than C4.5 — a problem more suitable for the NN program — at least one of our new methods has even lower or similar predictive error. Ten (auto-cost, auto-symbolling, breast cancer, heart disease, glass, image segmentation, iris, wine, zoo, and soybean) out of the 15 data sets belong to this case. Therefore, we can conclude empirically that our methods tend to improve the original NN program for problems suitable for NN.

4 Conclusions

In this paper, we extend an original approach of combining C4.5 and NN algorithms to three new approaches that also consider the attributes that do not appear in the tree, and the positions of the attributes in the tree. We tested these methods thoroughly using data sets from the machine learning repository. We conclude that info-dtree is immune to irrelevant attributes, and that info-top and info-all are suitable for cases where there are not enough training examples. In other cases (when all attributes are relevant and training examples are ample), our new methods tend to predict better than the original NN and binary-dtree since they take into account the importance of attributes.

In our future research, we will study the effect of taking k-nearest neighbours in our methods, and more suitable ways of calculating weights for the continuous attributes in the decision tree.

Acknowledgements

We gratefully thank David Aha and Claire Cardie for their detailed comments on the early draft of the paper, and Ross Quinlan for providing us with the C4.5 learning program.

Reference

Aha, D. W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. In *Proceedings of the 1989 International Workshop on Machine Learning*, pp. 387 – 391.

Aha, D. W., & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, pp. 106 – 112.

Aha, D. W., Kilber, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the 1993 International Conference on Machine Learning*, pp. 25 – 32.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. In *Proceedings of the 1994 International Conference on Machine Learning*, pp. 28 – 36.

Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, *10*.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*, 21–27.

Cover, T., & Hart, P. (1968). Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, *14*, 50–55.

Daelemans, W., Gillis, S., Durieux, G., & van den Bosch, A. (1993). Learnability and markedness in data-driven acquisition of stress. Tech. rep. ITK Research Report No. 43, Institute for Language Technology and AI (ITK), Tilburg University.

Dasarathy, B. (1991). *Nearest neighbor (NN) norms: NN pattern classification techniques*. Los Alamitos, CA: IEEE Computer Society Press.

Kelly, J. D., & Davis, L. (1991). A hybrid algorithm for classification. In Mylopoulos, J., & Reite, R. (Eds.), *Proceedings of IJCAI-91 (Thirteenth International Conference on Artificial Intelligence)*, pp. 645–650. Morgan Kaufmann, San Mateo, CA.

Kibler, D., & Aha, D. W. (1987). Learning representative exemplars of concepts: An initial case study. In *Proceedings of the 1987 International Workshop on Machine Learning*.

Ling, C. X. (1994). Learning the past tense of English verbs: the Symbolic Pattern Associator vs. connectionist models. *Journal of Artificial Intelligence Research*, *1*, 209 – 229. Anonymous ftp: /pub/ling/papers/jair-verb.ps.Z on ftp.csd.uwo.ca.

Ling, C. X., & Marinov, M. (1993). Answering the connectionist challenge: a symbolic model of learning the past tense of English verbs. *Cognition*, *49*(3), 235–290. Anonymous ftp: /pub/ling/papers/cognition-verb.ps.Z on ftp.csd.uwo.ca.

Ling, C. X., & Wang, H. (1995). Deciding optimal weights for IBL algorithms. Submitted to *Artificial Intelligence Review*.

Mohri, M., & Hanaka, H. (1994). An optimal weighting criterion of case indexing for both numeric and symbolic attributes. Tech. rep. WS-94-01, Case-Based Reasoning: Papers from the 1994 Workshop. Menlo Park, CA: AAAI Press.

Murphy, P. M., & Aha, D. W. (1992). *UCI Repository of machine learning databases [Machine-readable data repository]*. Irvine, CA: University of California, Department of Information and Computer Science.

Parry, J. J. (1994). Algorithms to calculate weights for the nearest neighbour learning. Master's thesis, Department of Computer Science, University of Western Ontario.

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

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

Schaffer, C. (1994). A conservation law for generalization performance. In *Proceedings of the 1994 International Conference on Machine Learning*.

Schmidt, W. C., & Ling, C. X. (1994). A decision-tree model of balance scale development. To appear in *Machine Learning*.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the 1994 International Conference on Machine Learning*, pp. 293 – 301.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, *29*, 1213–1228.

Wettschereck, D., & Dietterich, T. (1995). An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*. To appear.