In Jones & Somers (eds) New Methods in Language Processing, London: UCP, 3-15, 1997

Skousen's Analogical Modeling Algorithm: A comparison with Lazy Learning

Walter DAELEMANS
ITK, Tilburg University
The Netherlands
Walter.Daelemans@kub.nl

Steven GILLIS and Gert DURIEUX
Linguistics, University of Antwerp
Belgium
{Steven.Gillis,Gert.Durieux}@uia.ac.be

Abstract

We provide a qualitative and empirical comparison of Skousen's Analogical Modeling algorithm (AM) with Lazy Learning (LL) on a typical Natural Language Processing task. AM incorporates an original approach to feature selection and to the handling of symbolic, unordered feature values. More specifically, it provides a method to dynamically compute an optimally-sized set of nearest neighbours (the analogical set) for each test item, on the basis of which the most plausible category can be selected. We investigate the algorithm's generalisation accuracy and its tolerance to noise and compare it to Lazy Learning techniques on a primary stress assignment task in Dutch. The latter problem is typical for a large amount of classification problems in Natural Language Processing. It is shown that AM is highly successful in performing the task: it outperforms Lazy Learning in its basic scheme. However, LL can be augmented so that it performs at least as well as AM and becomes as noise tolerant as well.

Keywords: Analogy-based NLP, Example- and Memory-based NLP, Statistical methods.

1 Introduction

Machine Learning techniques may help in solving *knowledge acquisition bottlenecks* in Natural Language Processing: instead of handcrafting linguistic knowledge bases like rule bases and lexicons almost from scratch for each new domain, language, application, and theoretical formalism, we can apply machine learning algorithms to a corpus of examples, and automatically induce the required knowledge.

This paper explores one such class of algorithms, *Lazy Learning*, for which there has been an increased interest in Machine Learning recently. In this type of similarity-based learning, classifiers keep in memory (a selection of) examples without creating abstractions in the form of rules or decision trees. Generalisation to a new (previously unseen) input pattern is achieved by retrieving the most similar memory items according to some distance metric, and extrapolating the category of these items to the new input pattern. Instances of this form of *nearest neighbour method* include instance-based learning (Aha et al., 1991), exemplar-based learning (Salzberg, 1991; Cost and Salzberg, 1993), and memory-based reasoning (Stanfill and Waltz, 1986).

The approach has been applied to a wide range of problems using not only numeric and binary values (for which nearest-neighbour methods are traditionally used), but also using symbolic, unordered features. Advantages of the approach include an often surprisingly high classification accuracy, the capacity to learn polymorphous concepts, high speed of learning, and perspicuity of algorithm and classification (Cost and Salzberg, 1993). Aha et al. (Aha et al., 1991) have shown that the basic instance-based learning algorithm can pac-learn any concept whose boundary is a union of a finite number of closed hyper-curves of finite size (a class of concepts similar to that which ID3 and backpropagation can learn). Training speed is extremely fast (it consists basically of storing patterns), and classification speed, while relatively slow on serial machines, can be considerably reduced by using k-d trees on serial machines (Friedman et al., 1977), massively parallel machines (Stanfill and Waltz, 1986), or Wafer-Scale Integration (Kitano, 1993).

In Natural Language Processing, lazy learning techniques are currently being applied by various Japanese groups to parsing and machine translation under the names *exemplar-based translation* or *memory-based translation and parsing* (Kitano, 1993). In work by Cardie (Cardie, 1993) and by the present authors (Daelemans, 1995; Daelemans et al., 1994), variants of lazy learning are applied to disambiguation tasks at different levels of linguistic representation (from phonology to semantics).

One lazy learning variant, Analogical Modeling (Skousen, 1989) was explicitly developed as a linguistic model. On the one hand, despite its interesting claims and properties, it seems to have escaped the attention of the Computational Linguistics and Machine Learning communities while on the the other hand, the AM literature seems to be largely oblivious of the very similar lazy learning and nearest neighbour techniques. In this paper we provide a qualitative and empirical comparison of analogical modeling to the more mainstream variants of lazy learning.

In the next section the concept of Lazy Learning is elaborated further and a basic Lazy Learner is described. A number of improvements to that basic scheme will be discussed. Next, Analogical Modeling is presented in considerable detail. We will then turn to the main theme of this paper: a comparison of the basic lazy learner and analogical modeling. This comparison will mainly focus on two aspects: (i) the global performance of both algorithms on a selected linguistic task, and (ii) the algorithms' resistance to noise: the performance of the algorithms is monitored in conditions in which the learning material contains progressively more 'noise'.

2 Variants of Lazy Learning

Lazy Learning is a form of *supervised learning*. Examples are represented as a vector of feature values with an associated category label. Features define a pattern space, in which similar examples occupy regions that are associated with the same category (note that with symbolic, unordered feature values, this geometric interpretation does not make sense). Several regions can be associated with the same category, allowing polymorphous concepts to be represented and learned.

During training, a set of examples (the training set) is presented in an incremental fashion to the classifier, and added to memory. During testing, a set of previously unseen feature-value patterns (the test set) is presented to the system. For each test pattern, its distance to all examples in memory is computed, and the category of the least distant instance is used as the predicted category for the test pattern.

In lazy learning, performance crucially depends on the distance metric used. The most straightforward distance metric would be the one in equation (1), where X and Y are the patterns to be compared, and $\delta(x_i, y_i)$ is the distance between the values of the i-th feature in a pattern with n features.

$$\Delta(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{1}$$

Distance between two values is measured using equation (2) for numeric features (using scaling to make the effect of numeric features with different lower and upper bounds comparable), and equation (3), an overlap metric, for symbolic features.

$$\delta(x_i, y_i) = \frac{|x_i - y_i|}{max_i - min_i} \tag{2}$$

$$\delta(x_i, y_i) = 0 \; if \; x_i = y_i, \; else \; 1 \tag{3}$$

2.1 Feature weighting

In the distance metric described above, all features describing an example are interpreted as being equally important in solving the classification problem, but this is not necessarily the case. Elsewhere (Daelemans and van den Bosch, 1992) we introduced the concept of information gain, which is also used in decision tree learning (Quinlan, 1986; Quinlan, 1993), into lazy learning to weigh the importance of different features in a domain-independent way. Many other methods to weigh the relative importance of features have been designed, both in statistical pattern recognition and in machine learning (Aha, 1990; Kira and Rendell, 1992, etc. are examples), but the one we used will suffice as a baseline for the purpose of this paper.

The main idea of *information gain weighting* is to interpret the training set as an information source capable of generating a number of messages (the different category labels) with a certain probability. The information entropy of such an information source can be compared in turn for each feature to the average information entropy of the information source when the value of that feature is known.

Database information entropy is equal to the number of bits of information needed to know the category given a pattern. It is computed by equation (4), where p_i (the probability of category i) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_{p_i} p_i log_2 p_i \tag{4}$$

For each feature, it is now computed what the information gain is of knowing its value. To do this, we compute the average information entropy for this feature and subtract it from the information entropy of the database. To compute the average information entropy for a feature (equation 5), we take the average information entropy of the database restricted to each possible value for the feature. The expression $D_{[f=v]}$ refers to those patterns in the database that have value v for feature f, V is the set of possible values for feature f. Finally, $|D|$ is the number of patterns in a (sub)database.

$$H(D_{[f]}) = \sum_{v_i \in V} H(D_{[f=v_i]}) \frac{|D_{[f=v_i]}|}{|D|} \tag{5}$$

Information gain is then obtained by equation (6), and scaled to be used as a weight for the feature during distance computation.

$$G(f) = H(D) - H(D_{[f]}) \tag{6}$$

Finally, the distance metric in equation (1) is modified to take into account the information gain weight associated with each feature.

$$\Delta(X, Y) = \sum_{i=1}^{n} G(f_i)\delta(x_i, y_i) \tag{7}$$

2.2 Symbolic Features

A second problem confronting the basic lazy learning approach concerns the overlap metric (equation 3) for symbolic, unordered feature values. When using this metric, all values of a feature are interpreted as equally distant to each other. This may lead to unsufficient discriminatory power between patterns. It also makes impossible the well-understood 'Euclidean distance in pattern space' interpretation of the the distance metric. Stanfill and Waltz (1986) proposed a *value difference metric* (VDM) which takes into account the overall similarity of classification of all examples for each value of each feature. Recently, Cost and Salzberg (Cost and Salzberg, 1993) modified this metric by making it symmetric. In their approach, for each feature, a matrix is computed with the differences (represented as a real value) between each value using equation (8), where V_1 and V_2 are two possible values for feature f, the distance is the sum over all n categories, C_{1i} is the total number of times value V_1 was classified as category i, and C_1 is the total number of times V_1 occurred.

$$\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right| \tag{8}$$

The value difference matrices, computed for each feature, are used to compute the distance between two patterns with symbolic values, so that the distance metric in equation (2) can be used instead of equation (3), thereby safe-guarding the Euclidean distance in pattern space concept. Cost and Salzberg report excellent classification accuracy on a number of tasks with symbolic, unordered features with this modified Stanfill-Waltz VDM (MVDM).

2.3 Variants not considered

Apart from weighting the relative importance of different features, and computing the numeric distance between symbolic feature values, several other improvements and modifications to the basic lazy learning scheme have been proposed: weighting the examples themselves, based on typicality of performance, minimising storage by keeping only a selection of examples, etc. These variations will not concern us here. We will compare the analogical modeling approach to be discussed shortly to (1) the basic lazy learning scheme, (2) lazy learning with information gain weighting of features, and (3) lazy learning using the MVDM. We will consider both normal circumstances and the presence of noise.

3 The Analogical Modeling Algorithm

3.1 Background

Analogical Modeling has been proposed as a model of language usage, as an alternative to current rule-based linguistic descriptions. The main assumption underlying this approach is that many aspects of speaker performance are better accounted for in terms of 'analogy', i.e. the identification of similarities or differences with forms in memory (the lexicon), than by referring to explicit but inaccessible rules. See (Derwing and Skousen, 1989; Chandler, 1993) for psycholinguistic research supporting this assumption. As in Lazy Learning, the notion of 'analogy' is given an operational definition in terms of a matching process between an input pattern and a database of stored exemplars. The result of this matching process is a collection of examples called the analogical set, and classification of the input pattern is achieved through extrapolation from this set. AM thus shares some important characteristics with Lazy Learning: the main source of knowledge in both approaches is a database of stored exemplars. These exemplars themselves are used to classify new items, without intermediate abstractions in the form of rules. In order to achieve this, an exhaustive database search is needed, and during this search, less relevant examples need to be discarded.

The main difference between both approaches lies in the way this selection is made. In Lazy Learning with feature selection, the different features are assigned a relative importance, which is used during matching to filter out the influence of irrelevant features. In AM, essentially the same effect is achieved without precomputing the relative importance of individual features. Instead, all features are equally important initially, and serve to partition the database into several disjoint classes of examples. Filtering out irrelevant exemplars is done by considering properties of these classes rather than by inspecting individual features that their members may share with the input pattern. To explain how this works, we will describe the matching procedure in some more detail.[1]

3.2 The Algorithm

The first stage in the matching process is the construction of *subcontexts*; subcontexts are sets of examples, and they are obtained by matching the input pattern, feature by feature, to each item in the database, on an equal/not equal basis, and classifying the database exemplars accordingly. Taking an input pattern ABC as an example, eight (2^3) different subcontexts would be constructed, ABC, $\overline{A}BC$,$A\overline{B}C$, $AB\overline{C}$, $\overline{AB}C$, $\overline{A}B\overline{C}$, $A\overline{BC}$ and $\overline{ABC}$, where the overstrike denotes complementation. Thus, exemplars in the class ABC share all their features with the input pattern, whereas for those in $\overline{AB}C$ only the value for the third feature is shared. In general, n features yield 2^n mutually disjoint subcontexts. Subcontexts can be either *deterministic*, which means that their members all have the same associated category, or *non-deterministic*, when several categories occur.

In the following stage, *supracontexts* are constructed by generalising over specific feature values. This is done by systematically discarding features from the input pattern, and taking

[1] Our description of the algorithm is based on the PASCAL code in Skousen (1989).

the union of the subcontexts that are subsumed by this new pattern. Supracontexts can be ordered with respect to generality, so that the most specific supracontext contains examples which share all n features with the input pattern, less specific supracontexts contain items which share at least $n-1$ features, and the most general supracontext contains all database exemplars, whether or not they have any features in common with the input pattern. In the table below the supracontexts for our previous example are displayed, together with the subcontexts they subsume.

Supracontext			Subcontexts
A	B	C	ABC
A	B	-	ABC $AB\overline{C}$
A	-	C	ABC $A\overline{B}C$
-	B	C	ABC $\overline{A}BC$
A	-	-	ABC $A\overline{B}C$ $AB\overline{C}$ $A\overline{BC}$
-	B	-	ABC $\overline{A}BC$ $AB\overline{C}$ $\overline{A}B\overline{C}$
-	-	C	ABC $\overline{A}BC$ $A\overline{B}C$ $\overline{AB}C$
-	-	-	ABC $\overline{A}BC$ $A\overline{B}C$ $AB\overline{C}$ $\overline{AB}C$ $\overline{A}B\overline{C}$ $A\overline{BC}$ $\overline{ABC}$

An important notion with respect to supracontexts is *homogeneity*. A supracontext is called homogeneous when any of the following conditions holds:

- The supracontext contains nothing but empty subcontexts.
- The supracontext contains only deterministic subcontexts with the same category.
- The supracontext contains a single non-empty, non-deterministic subcontext.

Heterogeneous supracontexts are obtained by combining deterministic and non-deterministic subcontexts. Going from least to most general, this means that as soon as a supracontext is heterogeneous, any more general supracontext will be heterogeneous too.

In the final stage, the analogical set is constructed. This set contains all of the exemplars from each of the homogeneous supracontexts. Two remarks are in order here. First, since some exemplars will occur in more than one supracontext, each exemplar is weighted according to its distribution across different supracontexts. This is accomplished by maintaining a score for each exemplar. This score is simply the summed cardinality of each of the supracontexts in which the exemplar occurs. The intent of this scoring mechanism is to favor frequent patterns over less frequent ones, and patterns closer to the input pattern over more distant patterns, since the former will surface in more than one supracontext. Second, banning heterogeneous supracontexts from the analogical set ensures that the process of adding increasingly dissimilar exemplars is halted as soon as those differences may cause a shift in category. Exactly when this happens depends largely on the input pattern. To finally categorise the input pattern, either the predominant category in the analogical set or the category of a probabilistically chosen member of this set is chosen. In our experiments, we adopted the former approach.

4 Main Stress Assignment to Dutch Words

4.1 Data and Experiments

The experimental set-up used in all experiments consisted of a ten-fold cross-validation experiment (Weiss and Kulikowski, 1991). In this set-up, the database is partitioned ten times, each with a different 10% of the dataset as the test part, and the remaining 90% as training part. For each of the ten simulations in our experiment, the test part was used to test generalisation performance. The success rate of an algorithm is obtained by calculating the average accuracy (number of test pattern categories correctly predicted) over the ten test sets in the ten-fold cross-validation experiment.

In order to insert additional noise to the data, the datasets are subjected to the following procedure: for a given noise rate, every feature value and category in the dataset was replaced by another possible feature or category value with a probability amounting to the noise rate. A replaced value can be the same as the original one (with a probability depending on the number of different values for that feature or category). After noise addition to the dataset, the training and test sets were created in the way described above.

4.2 Task Description

The task consisted of predicting the stress pattern of individual (non-complex) Dutch words. For example, the main stress of the word *panama* is on the first syllable: *PAnama.* The task of determining main stress in Dutch is notoriously difficult. According to recent linguistic analyses of the domain, not more than 80% of Dutch monomorphemes are regular in metrical terms (Daelemans et al., 1994, for an overview). As such, the problem is typical for many natural language processing tasks: regularities, sub-regularities and exceptions account for data that is ambiguous and noisy (from the point of the modeler: human or learning algorithm).

The experiments are conceived as follows: in a training phase, words are presented together with their target stress patterns. Next, the algorithm is tested with new words (i.e. words not encountered during training) the stress pattern of which the algorithm has to predict. More specifically, the data were presented to the algorithms as strings of phonemes (representing their pronunciation). Following is a sample of the training input for the words *acroniem* (acronym), *actie* (action), *actief* (active), and *acuut* (acute), respectively. The first column denotes the category (A means stress on the last syllable, B on the penultimate syllable, C on the antepenultimate syllable). The remaining columns show for each of the three last syllables of a word its syllable structure (onset, nucleus, and coda, respectively). When part of the syllable structure is absent, the symbol '=' is used.

CAT	Antepenult			Penult			Last		
A	=	a	=	kr	o	=	n	i	m
B	=	=	=	=	A	k	s	i	=
A	=	=	=	=	A	k	t	i	v
A	=	=	=	=	a	=	k	y	t

A full discussion of data selection and encoding is irrelevant for this study, and can be found elsewhere (Daelemans et al., 1994). In the results reported here, we were not interested in optimal generalisation performance, but in a comparison of different classifier systems and input conditions. Optimal performance on this task is reported in Daelemans et al. (1994) as well.

4.3 Results

4.3.1 Comparison of AM and LL's basic scheme.

The first question to be addressed relates to the overall accuracy of AM as compared to the basic scheme of LL (k-nearest neighbour matching with different values of k and an overlap metric for symbolic features). AM clearly outperforms LL: AM attains a global success score of 80.5% while LL (k=1) reaches a success score of 75.4%, a difference that is highly significant as assessed by a Chi-square test ($\chi^2 = 7.5663$, $p < .006$**). When we extend LL to take into account more neighbours than just the closest matching one, the difference in performance between AM and LL still remains highly significant. In Table 1 the results are displayed comparing the success score of AM with an implementation of LL that compares a test item with its nearest neighbour (LL-1nn), its two nearest neighbours (LL-2nn), its five and ten nearest neighbours (LL-5nn and LL-10nn).

Table 1: Comparison of Success Scores of AM and LL k-Nearest Neighbours: Assessment of difference with AM

System	Success Score	$\chi^2 =$	p <
AM	80.5	/	/
LL-1nn	75.4	7.5663	.006 **
LL-2nn	75.8	6.4682	.0110 *
LL-5nn	76.1	5.6971	.0171 *
LL-10nn	75.1	8.4416	.0037 **

From Table 1 it appears that AM is significantly more successful than LL irrespective of the number of nearest neighbours considered in the latter algorithm. All comparisons of the success score reveal a significantly better performance for AM. The superior performance of

AM does not appear to be resistant to the addition of additional noise to the data. When additional noise is added, AM still outperforms LL in the conditions considered, but the difference is not significant anymore (Table 2). The significance of the differences in performance between AM and LL, as assessed by a Chi-square test, is indicated in Table 2.

Table 2: Performance of AM and LL Relative to Different Noise Levels

	Noise Level				
System	0%	20%	40%	60%	80%
AM	80.5	60.3	48.1	38.3	35.1
LL 1-nn	75.4 **	56.2	44.8	34.9	33.2
LL 2-nn	75.8 *	56.2	44.7	37.5	33.3
LL 5-nn	76.1 *	59.3	49.9	38.6	32.3
LL 10-nn	75.1 **	60.5	48.0	39.9	32.9

The results in Table 2 indicate that when additional noise is introduced into the training data, the performance of AM becomes similar to the performance of LL. Even if LL only takes into account the single nearest neighbour of the test item, it does not perform significantly worse than AM. This observation holds irrespective of the amount of noise added to the training data as well as the number of nearest neighbours taken into account.

4.3.2 Comparison of AM and IBL's augmented implementations.

In section 2 a number of enhancements of the basic LL algorithm were discussed. Two extensions were described: the basic LL scheme can be enriched with (i) a differential weighing of features using 'information gain' (IG), and (ii) a value difference metric which takes into account the overall similarity of classification of all examples for each value of each feature (MVDM). In this section we compare the performance of AM with that of LL augmented with (i) IG (LL-IG), (ii) MVDM (LL-MVDM) and (iii) a combination of the two (LL-IG-MVDM). In the data reported, LL was tested in the condition in which only the single nearest neighbour was considered.

In Table 3 the success score of AM is compared with the performance of LL's basic scheme (LL), and the three augmentations of LL, viz. LL-IG, LL-MVDM and LL-IG-MVDM. The comparison shows that only in its basic scheme LL attains a significantly lower success rate than AM. All extensions of LL (taking into account only nearest neighbour) yield success rates that are highly similar to those of AM, and in some cases even higher.

Table 3: Comparison of AM and Augmented Versions of LL: Assessment of difference with AM

System	Success	$\chi^2 =$	p <
AM	80.5	/	/
LL	75.4	7.5663	.006 **
LL-IG	81.8	0.5524	.4559 NS
LL-MVDM	79.4	0.3774	.5372 NS
LL-IG-MVDM	81.4	0.2626	.6065 NS

The results displayed in Table 3 pertain to a training set in which no additional noise is added. When additional noise is present, the comparisons of AM and LL do not show many discrepancies: in all conditions, there are hardly any results that can be marked as significantly different. The noise tolerance of AM is not more robust than the noise tolerance of LL: out of 16 comparisons only three yield a statistically significant difference (Table 4). This means that, at least for this task, AM is not more noise tolerant than LL. Extended with IG and MVDM, LL attains a highly similar success rate to AM. It should be noted, however, that we have only studied and compared overall sucess score, and have not, e.g., compared the errors made by the different variants (in terms of error types or relative markedness).

Table 4: Comparison of AM and Augmented Versions of LL: Assessment of difference with

AM when noise is included.

	Noise Level				
System	0%	20%	40%	60%	80%
AM	80.5	60.3	48.1	38.3	35.1
LL	75.4 **	56.2	44.8	34.9	33.2
LL-IG	81.8	57.3	42.5 *	38.2	34.6
LL-MVDM	79.4	55.5 *	44.1	36.3	34.4
LL-IG-MVDM	81.4	54.4 **	46.3	36.5	34.6

5 Conclusion

An important limitation of the Analogical Modeling algorithm is that it requires exponential time in the number of attributes. When repairing this using massive parallelism, the algorithm would be extraordinarily memory greedy. Using a very economic scheme, explained in Skousen (1987), the number of processors required for a 22 feature problem would be larger than the 2^{16} processors currently available on the Connection Machine. In lazy learning, processing time is linear in the number of features.

A second restriction is that AM can only usefully be applied to unordered symbolic or Boolean features: with numeric features or ordered symbolic features, the ordering is not taken into account during classification. Lazy Learning is more general in that all types of features can be handled by a uniform distance metric based on distance in a Euclidean space.

In the experimental tests reported in this paper, it was shown that AM outperforms the most basic implementation of a Lazy Learner in predicting the stress pattern of underived words. The addition of more nearest neighbours to the set of memorised items a test item is compared with, does not yield significant improvements: AM still performs better. However this advantage disappears when additional noise is entered in the training data. It was also shown that augmented versions of the Lazy Learner (that incorporate feature weighting and/or value difference matrices) perform equally well as AM even when additional noise is added in the training material. We conclude that, at least for this type of classification task, there is no clear motivation to use the complex and costly AM algorithm instead of the more general and less complex class of lazy learning algorithms.

References

Aha, D. (1990). A study of instance-based algorithms for supervised learning task. Technical Report 90-42, University of California at Irvine.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Cardie, C. (1993). A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *AAAI-93*, pages 798–803.

Chandler, S. (1993). Are rules and modules really necessary for explaining language? *Journal of Psycholinguistic Research*, 22(6):593–606.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Daelemans, W. (1995). Memory-based lexical acquisition and processing. In Steffens, P., editor, *Machine Translation and the Lexicon*, Lecture Notes in Artificial Intelligence 898, Springer, pages 85–98.

Daelemans, W., Gillis, S., and Durieux, G. (1994). The acquisition of stress, a data-oriented approach. *Computational Linguistics*, 20(3):421–451.

Daelemans, W. and van den Bosch, A. (1992). Generalization performance of backpropagation learning on a syllabification task. In *Proceedings Third Twente Workshop on Language Technology*, pages 27–38.

Derwing, B. and Skousen, R. (1989). Real time morphology: symbolic rules or analogical networks. In *Berkeley Linguistic Society 15*, pages 48–62.

Friedman, J., Bentley, J., and Ari Finkel, R. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3).

Kira, K. and Rendell, L. (1992). A practical approach to feature selection. In *International Conference on Machine Learning*.

Kitano, H. (1993). Challenges of massive parallelism. In *IJCAI*, pages 813–834.

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

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

Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276.

Skousen, R. (1989). *Analogical modeling of language*. Kluwer, Dordrecht.

Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29:1212–1228.

Weiss, S. and Kulikowski, C. (1991). *Computer systems that learn*. Morgan Kaufmann, San Mateo.