Optimal Subset Selection for Classification through SAT Encodings

Fabrizio Angiulli and Stefano Basta

Abstract In this work we propose a method for computing a minimum size training set consistent subset for the Nearest Neighbor rule (also said CNN problem) via SAT encodings. We introduce the SAT–CNN algorithm, which exploits a suitable encoding of the CNN problem in a sequence of SAT problems in order to exactly solve it, provided that enough computational resources are available. Comparison of SAT–CNN with well-known greedy methods shows that SAT–CNN is able to return a better solution. The proposed approach can be extended to several hard subset selection classification problems.

1 Introduction

Most useful classification tasks can be formulated as subset selection problems [6, 17, 12, 26, 28]. Subsets to be singled out have to posses certain properties guaranteeing that they represent a model of the whole training set, according to the specific classification rule. Often the number of potential models is exponential in the training set size and, among all the training set subsets, the *optimal model* is that composed of the minimum number of objects. Indeed, a small model improves both response time and (according to the Occam's razor) generalization.

For example, a *sample compression scheme* [12] is defined by a fixed rule $\rho : T \mapsto \rho(T)$ for constructing a classifier from a given set of data T. Given a training set T, it is compressed by finding the smallest subset $S \subseteq T$ for which the classifier $\rho(S)$ correctly classifies the whole set T. It is known that the size of a sample compression scheme can be used to bound generalization.

Fabrizio Angiulli
DEIS, Università della Calabria, Via P. Bucci 41C, 87036 Rende (CS), Italy, e-mail: f.angiulli@deis.unical.it

Stefano Basta
ICAR-CNR, Via P. Bucci 41C, 87036 Rende (CS), Italy, e-mail: basta@icar.cnr.it

Please use the following format when citing this chapter:

Angiulli, F. and Basta, S., 2008, in IFIP International Federation for Information Processing, Volume 276; *Artificial Intelligence and Practice II*; Max Bramer; (Boston: Springer), pp. 309–318.

Unfortunately, minimum cardinality subset selection problems often turn out to be intractable (e.g., [27]). Consequently, authors provide greedy heuristics (e.g., [18]) or attempt to search for near optimal solutions using non exhaustive search methods (e.g., [7]) or semi-naive enumeration methods (e.g., [23]) .

Nearest neighbor condensation. The Nearest Neighbor (NN rule for short) decision rule [6] is a widely employed classification rule. The NN rule assigns to an unclassified sample point the classification of the nearest of a set of previously classified points. For this decision rule, no explicit knowledge of the underlying distributions of the data is needed. A strong point of the NN rule is that, for all distributions, its probability of error is bounded above by twice the Bayes probability of error [6, 24, 10].

Naive implementation of the NN rule requires storage of all the previously classified data, and then comparison of each sample point to be classified to each stored point. In order to reduce both space and time requirements, several techniques to reduce the size of the stored data for the NN rule have been proposed (see [28] and [25] for a survey) referred to as training set condensation algorithms. In particular, among these techniques, *training set consistent* ones, aim at selecting a subset of the training set that classifies the remaining data correctly through the NN rule.

According to the discussion above, using a training set consistent subset, instead of the entire training set, to implement the NN rule, has the additional advantage that it may guarantee better classification accuracy. Indeed, [19] showed that the VC dimension of an NN classifier is given by the number of reference points in the training set. Moreover, computing a minimum cardinality training set consistent subset for the NN rule has been shown to be intractable [27].

A number of greedy training set condensation algorithms have been proposed that extract a consistent subset of the overall training set, namely CNN, RNN, MCNN, NNSRM, FCNN, and others [18, 15, 19, 9, 1, 3]. Approximate optimization methods, such as tabu search, gradient descent, evolutionary learning, and others, have been used to compute subsets close to the minimum cardinality one: [20] provides a comparison of a number of these techniques. However, none of these algorithms guarantees that the solution returned is of minimum size.

SAT Encodings. The SAT Problem [5] consists in deciding whether for a given Boolean formula there exists a truth value assignment to its variables that makes the formula true. SAT is the archetypical problem for the NP complexity class [14] and, therefore, many problems of practical interest in, among other examples, artificial intelligence, operations research, and electronic design engineering, can be SAT encoded, that is translated in suitable instances of SAT.

SAT solver technology is emerging, as witnessed by the annual conference devoted to this theme (the International Conferences on Theory and Applications of Satisfiability Testing are the primary annual meetings for researchers studying the SAT problem[1]), by several SAT solver implementations (e.g., [11, 21, 22]), and by

[1] See `http://www.satisfiability.org/`.

the annual competition (the international SAT Competitions identify new challenging benchmarks, promote new solvers for the SAT problem as well as compare them with state-of-the-art solvers[2]).

Proposed approach. In this work we investigate the possibility of computing a minimum size training set consistent subset for the NN rule (the CNN problem) via SAT encoding. The CNN problem is NP-hard [27] and belongs to the complexity class $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$, that is, loosely speaking, the class of the problems that can be solved in polynomial time by invoking at most a logarithmic number of times a procedure able to solve a problem in NP and which is assumed to reply instantaneously.

Basing on this property, we introduce the SAT–CNN algorithm, which exploits a suitable encoding of the CNN problem in a sequence of SAT problems in order to exactly solve it, provided that enough computational resources are available. The proposed approach can be extended to several intractable subset selection classification problems, such as SNN [23], k-NN [13], k-center [17], CNNDD [2], and others.

The rest of the work is organized as follows. In Section 2 some preliminary definitions are provided. Section 3 describes the SAT–CNN algorithm. Section 4 reports some experimental results. Finally, Section 5 depicts conclusions and future works.

2 Preliminary Definitions

In the following by T a labelled training set from a space $\mathscr{S}$ with distance d is denoted.

Let x be an element of T. By $nn(x,T)$ the nearest neighbor of x in T according to the distance d is denoted. By $\ell(x)$ the label associated to x is denoted.

Given a labelled data set T and an element y of $\mathscr{S}$, the *nearest neighbor rule* $\mathrm{NN}(y,T)$ assigns to y the label of the nearest neighbor of y in T, i.e. $\mathrm{NN}(y,T) = \ell(nn(y,T))$ [6].

A subset S of T is said to be a *training set consistent subset of* T if, for each $x \in T$, $\ell(x) = \mathrm{NN}(x,S)$ [18].

Given a training set T, the Minimum Training Set Consistent Subset Problem (or CNN problem) $\langle T \rangle$ is as follows: return a training set consistent subset S^* of T such that, for any other training set consistent subset S of T, $|S^*| \leq |S|$.

Given a training set T and a positive integer number k, the Training Set Consistent Subset Problem (or k-CNN problem) $\langle T,k \rangle$ is as follows: return a training set consistent subset S of T such that $|S| \leq k$, if at least one exists, and the empty set, otherwise.

Given a training set T and a positive integer number k, the decision version $\langle T,k \rangle_D$ of the problem $\langle T,k \rangle$ is as follows: return "no" if the answer of $\langle T,k \rangle$ is the empty set, and "yes" otherwise.

[2] See `http://www.satcompetition.org/`.

3 The SAT–CNN Algorithm

The algorithm SAT–CNN computes a minimum size training-set consistent subset of the input training set T. It accomplishes its task by encoding the problem of computing a training set consistent subset in a sequence of suitable instances of the SAT problem. Without loss of generality the Boolean formula is in conjunctive normal form (CNF), that is it is the conjunction of one or more clauses. A clause is the disjunction of one or more literals. A truth value assignment σ to the set of variables $X = \{x_1, \ldots, x_n\}$ is a function $\sigma : X \mapsto \{true, false\}$.

3.1 SAT Encoding

Given a labelled training set $T = \{o_1, \ldots, o_n\}$ and a positive integer number k, in the following $\Psi^k(T)$ denotes the SAT encoding of the decision problem $\langle T, k \rangle_D$.

More precisely, $\Psi^k(T)$ is a Boolean formula in conjunctive normal form[3] defined on the set of variables $x_1, x_2, \ldots, x_n$ and on some others auxiliary variables that will be introduced next. In particular, each variable x_i is associated with the object o_i of the training set T ($1 \leq i \leq n$). Indeed, if a truth value assignment for the variables in the formula $\Psi^k(T)$ makes the formula true, then the variables $x_1, \ldots, x_n$ encode a training set consistent subset S of T. In particular, the variable x_i being true (false, resp.) means that the corresponding object o_i belongs (does not belong, resp.) to S.

The formula $\Psi^k(T)$ consists of two sets of clauses, namely $\Psi_{cons}(T)$, also called *constraint clauses*, and $\Psi^k_{size}(T)$, also called *cardinality clauses*.

The clauses in the set $\Psi_{cons}(T)$ serve the purpose of guaranteeing that the subset S encoded by the truth assignment for the variables x_i is indeed a training set consistent subset of T. These clauses do not depend on the positive integer k.

The clauses in the set $\Psi^k_{size}(T)$ serve the purpose of guaranteeing that the size of subset S encoded by the truth assignment for the variables x_i does not exceed the value k.

Next we describe the structure of the two above introduced set of clauses. We assume that the training set T consists of $m \geq 2$ class labels $l = 1, 2, \ldots, m$, and that n_l represents the number of objects belonging the the class l (clearly, $n_1 + n_2 + \ldots + n_m = n$).

3.1.1 Constraint clauses

Before describing the constraint clauses, the following preliminary definition is needed.

[3] In the following we will use the terms *boolean formula in conjunctive normal form* and *set of clauses* interchangeably .

Given a labelled training set T and two objects o_i and o_j of T, having different class labels, by $c(o_i, o_j)$ we denote the set of objects of T which have the same class label of o_i and whose distance from o_i is not greater than the distance from o_i to o_j, that is

$$c(o_i, o_j) = \{o \in T \mid \ell(o) = \ell(o_i) \wedge \mathrm{d}(o, o_i) \leq \mathrm{d}(o_j, o_i)\}.$$

In order to guarantee that the truth value assignment for the set of variables $x_1, x_2, \ldots, x_n$ encodes a training set consistent subset S of T, it must be the avoided that there exist two objects o_i and o_j having different class labels, such that o_j belongs to S and o_j misclassifies o_i. The object o_i is not misclassified by the object o_j if there exists an object o_h in the set S, having the same class label of o_i and whose distance from o_i is less than the distance from o_i to o_j. As a whole the following property must be verified:

$$\begin{aligned}(\forall o_j)(\forall o_i)(\ell(o_j) \neq \ell(o_i) \wedge o_j \in S) \rightarrow \\ (\exists o_h \in S)(\ell(o_h) = \ell(o_i) \wedge \mathrm{d}(o_i, o_h) \leq \mathrm{d}(o_i, o_j)),\end{aligned}$$

that can be encoded through the following set of clauses

$$r_{i,j}^{(1)} \equiv x_j \rightarrow \left(\bigvee_{o_h \in c(o_i, o_j)} x_h \right) \equiv \neg x_j \vee \left(\bigvee_{o_h \in c(o_i, o_j)} x_h \right), \tag{1}$$

where i and j are such that $1 \leq i \leq n$, $1 \leq j \leq n$, and $\ell(o_i) \neq \ell(o_j)$. The number of clauses (1) is $\sum_{l=1}^{m} n_l(n - n_l) = \mathscr{O}(n^2)$ and each of them is composed of at most $1 + \max_{l=1}^{m} n_l = \mathscr{O}(n)$ literals. Thus, overall, clauses (1) are composed of at most $\mathscr{O}(n^3)$ literals.

Note that the truth value assignment which assigns false to every variable x_i satisfies clauses (1). Nonetheless, the empty set is not a valid training set consistent subset. Thus, the following set of clauses is needed in order to enforce nonemptyness of the solution set S:

$$r_l^{(2)} \equiv \bigvee_{o_i : \ell(o_i) = l} x_i, \tag{2}$$

where $l \in \{1, 2, \ldots, m\}$. The number of clauses (2) is m and, as a whole, they are composed of exactly n literals. In particular, clauses (2) require that for each class label at least one object of that class belongs to S. Clauses (1) and (2) form the set $\Psi_{cons}(T)$, which guarantees that S is a training set consistent subset of T.

Before concluding the description of the constraint clauses, it is important to point out that the truth value assignment which assigns true to every variable x_i, trivially satisfies all the clauses in $\Psi_{cons}(T)$. As a matter of fact, T is always a training set consistent subset of itself. Cardinality rules, described in the following, will take care of upper bounding the size of the subset S.

3.1.2 Cardinality clauses

The formula $\Psi^k_{size}(T)$ is defined on the n variables $x_1, x_2, \ldots, x_n$ and also on the nk auxiliary variables $e_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, k$. In particular, the variable $e_{i,j}$ being true (false, resp.) encodes the fact that the object o_i of T is (is not, resp.) the j-th element of the set S.

The clauses composing the set $\Psi^k_{size}(T)$ are detailed next.

First of all, it must be guaranteed that if o_i is the j-th element of S then x_i belongs to S (nk clauses of size 2):

$$r^{(3)}_{i,j} \equiv e_{i,j} \rightarrow x_i \equiv \neg e_{i,j} \vee x_i, \tag{3}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, k$.

Furthermore, if o_i belongs to S then it must exists a value $j_i \in \{1, 2, \ldots, k\}$ such that x_i is the j_i-th element of S (n clauses of size $k+1$):

$$r^{(4)}_i \equiv x_i \rightarrow \left(\bigvee_{j=1}^{k} e_{i,j} \right) \equiv \neg x_i \vee \left(\bigvee_{j=1}^{k} e_{i,j} \right), \tag{4}$$

where $i = 1, \ldots, n$.

Given Boolean variables $y_1, \ldots, y_n$, the *at-most-one constraint*

$$\textbf{at-most-one}(y_1, \ldots, y_n)$$

is a set of clauses which is satisfied if and only if at most one of the variables $y_1, \ldots, y_n$ is true.

The two following sets of clauses are needed to complete the cardinality clauses. The object o_i may occur at most one time in the subset S, that is, for each $i = 1, \ldots, n$,

$$r^{(5)}_i \equiv \textbf{at-most-one}\{e_{i,1}, \ldots, e_{i,k}\}, \tag{5}$$

and the j-th element of S may be at most one of the elements of T, that is, for each $j = 1, \ldots, k$,

$$r^{(6)}_j \equiv \textbf{at-most-one}\{e_{1,j}, \ldots, e_{n,j}\}. \tag{6}$$

Note that the formula $\Psi^k_{size}(T)$ enforces the set S to have at most k elements, hence S could be composed of less than k elements.

The at-most-one constraint can be formulated in different ways. Here we make use of the formulation known as *ladder encoding* [16, 4]. The ladder encoding of the at-most-one constraint $\textbf{at-most-one}(y_1, \ldots, y_n)$ is the Boolean formula, defined on the variables $y_1, \ldots, y_n$ and also on n novel variables $z_1, \ldots, z_n$, composed of the following $\mathcal{O}(n)$ clauses: the *ladder validity* clauses, for $i = 2, \ldots, n$,

$$c'_i \equiv z_i \rightarrow z_{i-1} \equiv \neg z_i \vee z_{i-1},$$

and the *channeling* clauses, for $i = 1, \ldots, n$,

Algorithm SAT–CNN
Input: a training set T and a timeout τ
Output: a training set consistent subset S_{opt} of T

1. Compute the *constraint clauses* $\Psi_{cons}(T)$
2. Optionally use a greedy method to find a seed cardinality training-set consistent subset S_{seed}, having size k_{seed}; otherwise set S_{seed} to T and k_{seed} to the size n of T
3. Set $k_{max} = k_{seed}$, $k_{min} = m$, $k_{opt} = k_{seed}$, $S_{opt} = S_{seed}$, and *approx* to false
4. If $k_{min} > k_{max}$ then goto 12
5. Set $k_{curr} = (k_{min} + k_{max})/2$
6. Compute the *cardinality clauses* $\Psi_{size}^{k_{curr}}(T)$
7. Solve the SAT problem $\Psi^{k_{curr}}(T) = \Psi_{cons}(T) \cup \Psi_{size}^{k_{curr}}(T)$
8. If the answer to $\Psi^{k_{curr}}(T)$ is "yes", then determine the size k_{sol} of the assignment $\sigma_{k_{curr}}$ found, that is the number of variables x_i which evaluate to true in $\sigma_{k_{curr}}$, and set $k_{max} = k_{sol}$, $S_{opt} = \{o_i \mid \sigma_{k_{curr}}(x_i) = true\}$, and $k_{opt} = k_{sol}$
9. If the answer to $\Psi^{k_{curr}}(T)$ is "no", then set $k_{min} = k_{curr} + 1$
10. If the answer to $\Psi^{k_{curr}}(T)$ is "unknown", then set $k_{min} = k_{curr} + 1$ and *approx* to true
11. Goto 4
12. Return the training set consistent subset S_{opt} and its size k_{opt}. If *approx* is set to true than the solution is approximate

Fig. 1 The Algorithm SAT–CNN.

$$c_i'' \equiv y_i \leftrightarrow (z_i \wedge \neg z_{i+1}) \equiv (y_i \vee \neg z_i \vee z_{i+1}) \wedge (\neg y_i \vee z_i) \wedge (\neg y_i \vee \neg z_{i+1}).$$

Intuitively, clauses c_i' impose that each truth value assignment for the variables $z_1, \ldots, z_n$ is of the form

$$(z_1, \ldots, z_t, z_{t+1}, \ldots, z_n) = (true, \ldots, true, false, \ldots, false),$$

where the number t of variables which evaluates to true can be zero, one, or more than one, while clauses c_i'' guarantee that y_t is true (if t is zero then no variable y_i is true).

3.2 SAT–CNN Algorithm

The algorithm SAT–CNN is a binary search based method enhanced with a greedy initialization step and exploiting the size of the current solution in order to accelerate convergence.

The algorithm is reported in Figure 1. Step 1 computes the constraint clauses $\Psi_{cons}(T)$. During the main cycle (steps 4-11) the minimum cardinality subset is searched for by adaptively adjusting the value of cardinality k_{curr} and then solving the SAT problem $P_{k_{curr}} = \Psi_{cons}(T) \cup \Psi_{size}^{k_{curr}}(T)$.

Other than the training set T, the algorithm receives in input a timeout τ, denoting the maximum amount of time allowed to the SAT solver to solve the current instance $\Psi^{k_{curr}}(T)$. If the solver does not return an answer to $\Psi^{k_{curr}}(T)$ within time τ, then it is stopped and its answer is assumed to be "unknown" (see step 10). Note that the solver may return the answer "unknown" also because either the memory is over or it is not able to answer to the given instance (the latter situation may occur only when the solver is not complete).

4 Experimental Results

We interfaced SAT-CNN with the RSat 2.0 SAT solver [22]. RSat is a DPLL-based [8] complete SAT solver that employs many modern techniques such as those used in MiniSat [11] and Chaff [21]. It won gold medals from the SAT'07 competition in the industrial category.

We compared the cardinality of the solution computed by the SAT–CNN algorithm with the cardinality of the solutions returned by well-known greedy algorithms, namely CNN, MCNN, NNSRM, and FCNN [18, 19, 9, 1, 3].

In the experiments, S_{seed} was always set to the whole training set (see step 2 in Figure 1), while the timeout τ was set to 500 seconds. We employed a Core 2 Duo based machine having 2GB of main memory.

The next table reports the data set employed in the experiments (data sets are from the UCI Machine Learning Repository[4]), together with the size of the solution computed by SAT–CNN compared with the best size returned by the greedy algorithms.

Data Set	*Size*	*Dims*	*Classes*	SAT–CNN	*Greedy*	*Ratio*
Bupa	345	6	10	145	168	86%
Colon Tumor	62	2,000	2	13	17	76%
Echocardiogram	61	11	2	2	5	40%
Iris	150	4	3	10	13	77%
Ionosphere	351	34	2	45	55	82%
Pima	768	8	2	300	316	95%
SPECT Heart	349	44	2	75	93	81%
Vehicle	846	18	4	348	382	91%
Wine	178	13	3	51	62	82%

The last column shows the ratio between the size of the SAT–CNN solution and the size of the best greedy solution. The SAT-CNN algorithm improved over greedy methods in all cases. Moreover, it reported that the solution is exact on the *Iris* and *Echocardiogram* data sets.

[4] See `http://mlearn.ics.uci.edu/MLRepository.html`.

Data Set	*Time 1*	*Time 2*	*Clauses*	*Max. Clauses*	*Max. Vars*
Bupa	3,021	64	3,811,018	593,440	178,882
Colon Tumor	1,140	7	49,984	19,304	5,920
Echocardiogram	6	3	20,432	18,210	5,642
Iris	665	31	220,825	116,849	34,124
Ionosphere	2,529	43	1,683,093	610,929	185,152
Pima	39,169	717	19,827,074	2,925,278	886,655
SPECT Heart	2,151	38	2,149,965	596,190	183,050
Vehicle	3,661	623	29,753,120	3,768,274	1,078,225
Wine	2,564	10	709,579	164,147	47,970

Finally, we report in the table above some statistics concerning SAT–CNN, that are the total execution time (column *Time 1*, in seconds), the rewriting time (column *Time 2*, in seconds), the total number of clauses evaluated (column *Clauses*), and the maximum number of clauses (column *Max. Clauses*) and variables (column *Max. Vars*) included in a single SAT instance.

5 Conclusions and Future Work

This work introduces the SAT–CNN algorithm, which exploits a suitable encoding of the CNN problem in a sequence of SAT problems in order to exactly solve it.

As future work we plan to extend experiments in order to study how the size of the solution varies with the timeout, to take into account other training sets, to investigate testing accuracy, and to compare with approximate optimization approaches. We also plan to run our method with other state of the art SAT solvers, and to provide encodings for other families of solvers, such as *pseudo-boolean solvers* and *stable models engines*. We will also investigate alternative rewritings for the cardinality clauses and methods to reduce the number of constraint clauses. Finally, we will extend the method here presented to other classification tasks that can be formalized as hard subset selection problems, as SNN [23], k-NN [13], k-center [17], CNNDD [2], and others.

References

1. Angiulli, F. (2005). Fast condensed nearest neighbor rule. In *22nd International Conference on Machine Learning (ICML)*, Bonn, Germany.
2. Angiulli, F. (2007). Condensed nearest neighbor data domain description. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1746–1758.
3. Angiulli, F. (2007). Fast nearest neighbor condensation for large data sets classification. *IEEE Trans. Knowl. Data Eng.*, 19(11):1450–1464.
4. Manyà, F., & Ansótegui, C (2004). Mapping problems with finite-domain variables into problems with boolean variables. In *Proc. of the Seventh Int. Conf. on Theory and Applications of Satisifiability Testing* (SAT), pages 111–119, Vancouver, BC, Canada.
5. Cook, S.A. (1971). The complexity of theorem-proving procedures. In *3rd ACM Symposium on Theory of Computing*, pages 151–158, Ohio, United States.
6. Hart P.E., & Cover, T.M. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

7. Dasarathy, B. (1994). Minimal consistent subset (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517.
8. Logemann, G., Loveland, D., & Davis, M. (1962). A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397.
9. Murty, M.N., & Devi, F.S. (2002). An incremental prototype set building technique. *Pattern Recognition*, 35(2):505–513.
10. Devroye, L. (1981). On the inequality of cover and hart in nearest neighbor discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:75–78.
11. Sörensson, N., & Eén, N. (2005). Minisat a sat solver with conflict-clause minimization. In *International Conference on Theory and Applications of Satisfiability Testing*.
12. Warmuth, M., & Floyd, S. (1995). Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21(3):269–304.
13. Hostetler, L.D., & Fukunaga, K. (1975). *k*-nearest-neighbor bayes-risk estimation. *IEEE Trans. on Information Theory*, 21:285–293.
14. Johnson, D.S., & Garey, M.R. (1979). *Computers and Intractability. A Guide to the Theory of NP-completeness*. Freeman and Comp., NY, USA.
15. Gates, W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433.
16. Prosser, P., & Gent, I.P. (2002). In *Proc. of the Fifth Int. Conf. on Theory and Applications of Satisifiability Testing* (SAT), Cincinnati, Ohio, USA.
17. Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.
18. Hart, P.E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516.
19. Krim, H., & Karaçali, B. (2003). Fast minimization of structural risk by nearest neighbor rule. *IEEE Transactions on Neural Networks*, 14(1):127–134.
20. Nakagawa, M., & Liu, C.L. (2001). Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pattern Recognition*, 34(3):601–615.
21. Madigan, C., Zhao, Y., Zhang, L., Malik, S., & Moskewicz, M. (2001). Engineering an efficient sat solver. In *39th Design Automation Conference (DAC)*.
22. Darwiche, A., & Pipatsrisawat, K. (2007). Rsat 2.0: Sat solver description. Technical Report D–153, Automated Reasoning Group, Computer Science Department, UCLA.
23. Woodruff, H.B., Lowry, S.R., Isenhour, T.L., & Ritter, G.L. (1975). An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21:665–669.
24. Stone, C. (1977). Consistent nonparametric regression. *Annals of Statistics*, 8:1348–1360.
25. Toussaint, G. (2002). Proximity graphs for nearest neighbor decision rules: Recent progress. In *Proceedings of the Symposium on Computing and Statistics*, Montreal, Canada, April 17–20.
26. Vapnik, V. (1995). *The Nature of the statistical learning theory*. Springer Verlag, New York.
27. Wilfong, G. (1992). Nearest neighbor problems. *International Journal of Computational Geometry & Applications*, 2(4):383–416.
28. Martinez, T.R., & Wilson, D.R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286.