INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

SOKOLOVA, Marina L.

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE - DEGREE

Faculty of Administration

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Decision List Machines

Mario Marchand

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

N. Japkowicz

J.-M. Thizy

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Decision List Machines

by Marina L. Sokolova

Thesis

Submitted to the School of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Science

in Systems Science

Under the supervision of Prof. Mario Marchand

Faculty of Administration

University of Ottawa

Ontario, Canada

November 2001

© **Marina L. Sokolova, Ottawa, Canada, 2001**

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-67863-6

Canada

Contents

1 Introduction **7**

1.1 Overview of the problem 7

1.2 Motivation 9

1.3 Contributions 9

1.4 The thesis organization 10

2 The Background Review **11**

2.1 The classification problem 11

2.2 The feature space 13

2.3 The Set Covering Machine 14

2.4 Generalized balls 17

2.5 Experimental design 19

3 Decision List Machines **24**

3.1 Definition of a decision list 24

3.2 The feature space 26

3.3 Alternations and essential precedence 28

3.4 The learning algorithm 29
3.5 The DLM with generalized balls 31
3.6 Model selection 34

4 Empirical Results **36**
4.1 Results on artificial data 36
4.2 Results on "natural" data 40
4.3 Results on "natural" data in the case of asymmetric loss . . . 44

5 Theoretical Results **47**
5.1 The compression scheme for the DLM 48
5.2 Generalization error bounds 55

6 Conclusions **57**
6.1 Summary 57
6.2 Future work 58

A **63**

B **66**

List of Tables

2.1 Performance of the consistent SCM 21

2.2 Optimal performances of the SCM-conjunction, the SCM-disjunction and the SVM 23

4.1 Sizes of the DLM and the SCM on the artificial data 39

4.2 Sizes of the DLM for the case of zero empirical risk 40

4.3 Performance of the inconsistent DLM_i 42

4.4 Performances of different types of the DLM 42

4.5 Smallest errors obtained by the DLM, the SCM, NNC, and the SVM 43

4.6 The DLM performance for the asymmetric loss on Glass (87 positive and 76 negative examples) 46

B.1 DLM results for the asymmetric loss on Credit (296 positive and 357 negative examples) 66

B.2 DLM results for the asymmetric loss on Haberman (219 positive and 75 negative examples) 67

List of Figures

4.1 2D artificial data 37
4.2 3D artificial data 38
4.3 Results on the Glass data set for the asymmetric loss 46

ACKNOWLEDGMENTS

I am grateful to my supervisor, Dr. Mario Marchand for all the help he gave me during my years as a graduate student. Thanks go to Mario for arranging my financial support (through his NSERC research grant), and for being an enlightening source of supportive guidance.

Thanks go to Dr. Nathalie Japkowicz for her thoughtful comments on the thesis and useful suggestions for my work.

I wish to give special thanks to Dr. Jean-Michel Thizy, the Director of Systems Science Program, for his attention and encouragement during my studies.

Thanks go to my lovely husband, Guennadi V. Sokolov, for being the true master of the household, to our teenagers, Olga and Mikhail, for their patience and concern. Especially I would like to send my love and regards to my father, Dr. Leonid G. Safonov, for his life-long example of dedication and passion.

ABSTRACT

We learn decision lists over a space of features that are constructed from the data. A practical machine which we call the Decision List Machine comes as a result. We construct the Decision List Machine which uses generalized balls as data-dependent features.

We compare practical performance on some data sets with the performance of some other learning algorithms such as the Set Covering Machine and the Support Vector Machine.

This performance is evaluated for both symmetric and asymmetric loss coefficients.

We also provide a theoretical assessment of the performance of the DAM by computing upper bounds of the generalization error.

Chapter 1

Introduction

1.1 Overview of the problem

The problem of learning from data is one part of the general experimental procedure used in different fields of science and engineering. The need for understanding large, complex, information-rich data sets is common to virtually all fields of business, science, and engineering. Some examples include medical diagnosis, hand-written character recognition, and time series predictions. Methods for learning from data have been traditionally explored in such field as statistics (multivariate regression and classification)[1].

In general the problem encountered by the *learning machine* is to select a function (from the set of functions it supports) that best approximates the system's response.

Vapnik's theory of Structural Risk Minimization ([18], [19]) shows that the maximum margin hyperplane is a good solution for linearly separable

data sets. The maximum margin hyperplane is used to build the Support Vector Machine (SVM), which is a popular and effective method for constructing non-linear classifiers.

Support Vector Machines try to achieve good generalization by computing the maximum margin separating hyperplane in a high-dimensional feature space. This approach combines two good ideas. The first idea is to map the space of input vectors into a very high-dimensional feature space in such a way that nonlinear decision functions on the input space can be constructed by using only (linear) separating hyperplanes on the feature space. The second idea is to construct the separating hyperplane on the feature space, which has the largest possible margin ([1], [4]).

Marchand and Shawe-Taylor have designed the Set Covering Machine (SCM) [12]. The main idea of the SCM is the following: if the function to learn is a conjunction or a disjunction, then it is better to use a learning algorithm, which will construct this type of function. The SCM algorithm combines two powerful methods:

- map the input space in to the intermediate space,
- use the greedy set covering algorithm as an optimization method in the intermediate space(for definitions see Appendix A)

The algorithm can be used with discrete or continuous data, and does not require a quadratic optimization procedure such as the SVM. Therefore, the SCM is simpler to implement than the SVM, and, as it is shown further

in the thesis, the SCM better approximates the function to learn than the SVM for some data.

In this work we introduce the *Decision List Machine* (DLM). It

- maps the input space into the intermediate space
- constructs a decision list on the intermediate space

1.2 Motivation

The basic reason to build the Decision List Machine (DLM) and to use it as a learning machine is quite natural. The SCM has produced good results on many data sets. It was an open question how the same learning strategy would behave on the strictly richer class than conjunctions or disjunctions.

1.3 Contributions

The main contributions of the thesis are:

1. **A learning algorithm.** The learning algorithm is proposed for DLM using generalized balls for its set of features
2. **Experimental results.** The implementation of the learning algorithm has been done, and experimental results are obtained on artificial and "natural" data.
3. **Theoretical results.** Theoretical assessment of DLM performance is obtained and proved for different versions of the learning algorithm.

1.4 The thesis organization

The remainder of the thesis is organized as follows:

Chapter 2 introduces basic machine learning definitions and notions, used in the current research. It gives the description of the Set Covering Machine (SCM).

Chapter 3 presents the Decision List Machine (DLM). It provides the formulation of the learning algorithm and the implementation of the feature space of generalized balls in the DLM. It extends rules for model selection obtained for the SCM, to the case of the DLM.

Chapter 4 describes the testing performed by the DLM on both artificial and "natural" data sets. Test results are represented and compared with results of the SCM, the support vector machine and nearest-neighbor classifier. Results for loss functions with asymmetric loss coefficients are reported separately.

Chapter 5 presents a theoretical assessment of DLM performance. Upper bounds for the generalization error for different types of the DLM are determined.

Chapter 6 contains conclusions and propositions for future work.

Chapter 2

The Background Review

2.1 The classification problem

An *input* example $\vec{x} = (x_1, \ldots, x_n)$ needs to be classified into one (and only one) of j classes (or groups) $C_1, \ldots, C_j$. The existence of the classes is known a priori. *Classification* is concerned with the relationship between the class-membership *label* y and the input vector $\vec{x}$.

Examples of classification tasks are:

- The diagnosis of a medical condition from symptoms, in which the classes could be either the various disease states or the possible therapies
- The prediction of an election from the voter's information, in which the classes could be either an election's participation or a choice of a certain candidate

- The diagnosis of a person's creditability from a bank's information, in which the classes could be different levels of a mortgage repayment
- Deciding from atmospheric observations whether a severe thunderstorm is possible or unlikely.

In *supervised learning* the training input examples are given with their class labels or *outputs.*

In the *two-class* classification problem the output takes on only two symbolic values $y \in \{0, 1\}$, corresponding to two classes, called negative and positive.

A function $f(\vec{x})$ maps the input example $\vec{x}$ onto the class label y. This function can be expressed as a logical definition, a procedure, a decision tree, or a network [16].

Commonly the (0,1)-*loss function* $l(y, f(\vec{x}))$ measures the classification error

$$l(y, f(\vec{x})) = \begin{cases} 0 & \text{if } y = f(\vec{x}) \\ 1 & \text{otherwise} \end{cases}$$

.

For the (0,1)-loss function the *true risk* R is defined as

$$R = \int |y - f(\vec{x})| dP(\vec{x}),$$

where $P(\vec{x})$ is the (unknown) distribution of the input examples.

The *generalization error* of function $f(\vec{x})$ is

$$err(f) = Pr_{(\vec{x}) \in P(\vec{x})}(f(\vec{x}) \neq y)$$

for a randomly drawn example $(\vec{x}, y)$. In case of the (0,1)-loss function the generalization error is equal to the true risk.

A is called a *batch learning algorithm* if it constructs a function after examining all of the examples of a given finite training set. The goal of a learning algorithm is to produce a function $f(\vec{x})$ which minimizes the true risk (or the generalization error).

If the error is small and probability of failure is bounded by a small constant, then this is called *probably approximately correct learning* (PAC-learning)[13]. For the mathematically oriented framework on PAC-learning see [15].

Since the distribution $P(\vec{x})$ is unknown, we use the *empirical risk* R_{emp} as the estimation for the true risk:

$$R_{emp} = \frac{1}{m} \sum_{i=1}^{m} |y_i - f(\vec{x}_i)|.$$

For any loss function $l(y, f(\vec{x}))$

- true risk $R = \int l(y, f(\vec{x}))dP(\vec{x})$
- empirical risk $R_{emp} = \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(\vec{x}_i))$

2.2 The feature space

Consider an arbitrary n-dimensional input space $X \subset \mathcal{R}^n$.

For an input vector $\vec{x} \in X$ consider a mapping to a s-dimensional Boolean vector $\vec{h}(\vec{x}) = (h_1(\vec{x}), \ldots, h_s(\vec{x}))$, where each $h_i(\vec{x})$ is called a *feature*.

Simple functions on a feature space are built while original functions on the input space may be rather complex.

In this thesis we construct a learning algorithm which builds a decision list on a feature space. A learning algorithm which builds conjunctions and disjunctions on a feature space was designed by Marchand and Shawe-Taylor[12]. They call it the Set Covering Machine (SCM).

We introduce two important definitions.

A feature $h_i(\vec{x})$ is *consistent* with an example x if it makes zero error on it.

A feature $h_i(\vec{x})$ is *data-dependent* if it is constructed from the data.

2.3 The Set Covering Machine

Let us first review the work of Marchand and Shawe-Taylor[12].

The SCM classifies an input vector $\vec{x}$ with a *hypothesis* $f(\vec{x})$ - a conjunction (or disjunction) of some features $h_i(\vec{x})$ - that was found by the learning algorithm described below. Hence, for an input vector $\vec{x}$ the SCM outputs
$f(\vec{x}) = \bigwedge_{i \in \mathcal{R}} h_i(\vec{x})$ for a conjunction or
$f(\vec{x}) = \bigvee_{i \in \mathcal{R}} h_i(\vec{x})$ for a disjunction
where $\mathcal{R}$ indexes the set of features returned by the learning algorithm.

Given a set of m training examples, the learning algorithm for the SCM must also be provided with a set $\mathcal{S}$ of s (Boolean valued) features. $\mathcal{S}$ is required to satisfy the following condition: there exists a conjunction (or disjunction) of features in $\mathcal{S}$ which is consistent (makes zero error) with all

the training examples.

To construct a hypothesis, the SCM learning algorithm uses two well distinguishable steps.

Consider the case where we learn a conjunction. We assume that there exists a set $A \subset S$ over which a conjunction of features $\bigwedge_{i \in A} f_i(\vec{x})$ is consistent with all the training examples. The set A of features is included in the set B of features in S consistent with the positive training examples. $\bigwedge_{i \in B} f_i(\vec{x})$ is consistent with all the positive training examples by construction. It also never misclassifies any negative example of the training set since $A \subset B$.

Valiant step Find the set $B = \{f_1(\vec{x}), \ldots, f_n(\vec{x})\}$ of all features in S such that each $f_i(\vec{x}) \in B$ makes zero error with all the positive training examples.

Marchand and Shawe-Taylor showed in [12] that the generalization error of the SCM depends on the size of the conjunction. To reduce the generalization error we need to reduce the number of features in the output hypothesis.

Recall that $\bigwedge_{i \in B} f_i(\vec{x})$ is consistent with the set $\mathcal{N}$ of all the negative training examples. Therefore, each feature $f_i(\vec{x}) \in B$ is consistent with some the negative training examples. We denote the set of such examples as Q_i. Since $\bigcup_{i \in B} Q_i = \mathcal{N}$, we say that $\{Q_i\}_{i \in B}$ forms a cover for $\mathcal{N}$.

The problem to find the smallest subset of features in B whose conjunction is consistent with $\mathcal{N}$ becomes the problem to find the smallest collection Y of sets Q_i for which $\bigcup_{i \in Y} Q_i$ is equal to $\mathcal{N}$. This is the well known (and NP-complete) Set Cover Problem (for definition see Appendix A). It is hard to find the set cover of the minimum size, and we use the set cover greedy

algorithm instead. Recall that the set cover greedy algorithm has a good worst-case bound [7],[11].

Lemma 1 *Let r be the size of the minimal cover that covers m examples. Then the set cover greedy algorithm outputs a cover of size at most*

$$r \times ln(m) + 1.$$

Proof: The proof is in Appendix A.

Haussler step Apply the set cover greedy algorithm to find a subset $I \subseteq B$ of features that covers all the negative training examples.

Practically, first choose the set Q_i which covers the largest number of elements in $\mathcal{N}$, and remove these elements from $\mathcal{N}$ and each $Q_{j \neq i}$. Repeat the procedure on $\mathcal{N} \backslash Q_i$ with the set Q_k of the largest cardinality. Stop the repetition when there are no more uncovered examples in $\mathcal{N}$.

Let r be the size of the minimal cover that covers $\mathcal{N}$. According to the result of Lemma 1,the SCM learning algorithm outputs the conjunction which size is at most $r \times ln(|\mathcal{N}|) + 1$. Hence, the size of the output is linear with respect to the minimal number of features, covering all the negative examples, is logarithmic with respect to the cardinality of $\mathcal{N}$, and is independent of $|\mathcal{S}|$.

To learn a disjunction with the SCM we make the assumption that there exists a disjunction of features $\bigvee_{i \in A} f_i(\vec{x})$ which is consistent with all the training examples. In the Valiant step we build a set of features $C = \{f_1(\vec{x}), \ldots, f_n(\vec{x})\}$ which make zero error on all the negative training examples. Following the same logic as above, in the Haussler step we apply the set covering greedy algorithm to the Valiant step's outcome to find a

subset $I \subseteq C$ of features that covers all the positive training examples. Let $\mathcal{P}$ be the set of all the positive training examples. If r is the size of the optimal cover that covers $\mathcal{P}$, then the output disjunction has a size of at most $r \times ln(|\mathcal{P}|) + 1$ features.

2.4 Generalized balls

In this section we introduce a data-dependent set $\mathcal{S}$ of features with the required property that there exists a conjunction (or disjunction) of some features in $\mathcal{S}$ which is consistent with all the training examples.

Let $d(\vec{x}_i, \vec{x}_j)$ be the distance between two points in the original input space, ρ be any real-valued number, and $\overline{y_i}$ denotes the Boolean complement to $y_i \in \{0, 1\}$. For each training example $\vec{x}_i$ with label y_i consider a ball $h_{i,\rho}$ centered in $\vec{x}_i$ with radius ρ. On any training example $\vec{x}$ we define the value of the ball as follows:

$$h_{i,\rho} = h_{i,\rho}(\vec{x}) = \begin{cases} y_i & \text{if } d(\vec{x}_i, \vec{x}) \leq \rho \\ \overline{y_i} & \text{otherwise} \end{cases}$$

We can use any metric to calculate the distance, that is why we call $h_{i,\rho}$ a *generalized ball*. No initial restrictions are imposed on the radius ρ, hence, for a given set of m training examples the initial space $\mathcal{S}$ of features could be infinite.

For practical purpose we consider that each training example $\vec{x}_i$ defines $m - 1$ radii $\rho_{i,j} = d(\vec{x}_i, \vec{x}_j), i \neq j$. This consideration restricts the number of

generalized balls, built on m training examples, to $\mathcal{O}(m^2)$. Assume that the training set does not contain any contradictory examples , or if $\vec{x}_i = \vec{x}_j$ then $y_i = y_j$ is fulfilled. Then there always exists a conjunction (or a disjunction) of some generalized balls which is consistent with all the training examples.

We will now show that the SCM reduces the size of S to be equal to the number of training examples. Let us first consider the case of a conjunction.

For a positive training example $\vec{x}_i$, consider a set of generalized balls, centered on $\vec{x}_i$ where each of them includes all the positive training examples. Let $\rho_{i,min}$ denote the smallest distance from a positive training example $\vec{x}_i$, that includes all the positive training examples.

For a negative training example $\vec{x}_j$ consider a set of generalized balls, centered on $\vec{x}_j$ where each of them excludes all the positive training examples. Let $\rho_{j,max}$ denote the largest distance from a negative training example $\vec{x}_j$, that excludes all the positive training examples.

In the Haussler step the algorithm gives preference to the balls which misclassify as few negative examples as possible. For a positive example $\vec{x}_i$ a generalized ball, centered on $\vec{x}_i$, with a radius larger than $\rho_{i,min}$, will misclassify more negative examples that the generalized ball, centered on $\vec{x}_i$, with the radius $\rho_{i,min}$. Hence, for a positive example $\vec{x}_i$ we will consider only the generalized ball of radius $\rho_{i,min}$.

Similarly for a negative example $\vec{x}_j$, a generalized ball centered on $\vec{x}_j$, with a radius smaller than $\rho_{j,max}$ will misclassify more negative examples that the generalized ball, centered on $\vec{x}_j$, with the radius $\rho_{j,max}$. Hence, for a negative example $\vec{x}_j$ we will consider the generalized ball of radius $\rho_{j,max}$.

In the case of learning a disjunction, we also construct a set of features $\mathcal{S}$, such that $|\mathcal{S}|$ is equal to the number of training examples. For each positive example $\vec{x}_i$ we find the largest distance that excludes all the negative training examples, and build the corresponding generalized ball. For each negative example $\vec{x}_j$, we find the smallest distance that includes all the negative training examples, and build the corresponding generalized ball. Hence, the number of generalized balls is equal to the number of training examples.

2.5 Experimental design

To complete the description of the SCM we provide results of its performance on some "natural" data sets. The Glass data set was obtained from Robert Holte, now at the University of Alberta, and six other data sets were obtained from the machine learning repository at UCI [20]. All contradictory examples are eliminated, thus the size of the Haberman data set is reduced from 306 to 294 examples. Also we removed all examples with unknown attribute values.

For all data sets a *resampling method* is used for the estimate of the generalization error. Resampling methods make no assumptions on the statistics of the data or on the type of a function to construct.

The basic idea is first to build a hypothesis using a portion of the given data and then to use the remaining samples to estimate the true risk for this hypothesis. The first portion of the data is called a training set, and the second portion of the data is a testing set.

The more sophisticated approach used is known as *k-fold cross-validation.*

It consists of these steps:

1. Divide m given examples into k disjoint subsamples of roughly equal size m/k.

2. For each testing sample $S_i, i \leq k$ use the remaining data to construct a hypothesis h.

3. Calculate the empirical risk for S_i.

4. Compute the estimate for the true risk by averaging the empirical risks for $S_1, \ldots, S_k$.

Typical choices for k are 5 and 10.

The results of 10-fold cross-validation of [12] are summarized in table 2.1. The size of a conjunction (disjunction) is given as the average number of generalized balls, built by the SCM over the 10 different training sets of 10-fold cross-validation. The Euclidean (L_2) metric is used for distances. The error column represents the sum of misclassified examples over all testing sets of 10-fold cross-validation.

According to the structural risk minimization principle [19] the classification error can increase with the decrease of the empirical risk.

The attempt to improve the conjunction's (disjunction's) quality of fitting the training examples and receive zero training error, can prevent a better generalization with smaller conjunction (disjunction), which makes fewer errors on the training set.

Data		SCM - conjunc		SCM - disjunc	
data set	number of ex	size	error	size	error
Breast Wisc	683	13.1	26	16.8	34
Bupa	345	80.5	144	68.8	131
Credit	653	137.4	255	123.5	225
Glass	163	19	45	15.9	36
Haberman	294	45.5	104	46.6	128
Pima	768	147.3	260	123.9	230
Votes	52	2.9	7	3.1	6

Table 2.1: Performance of the consistent SCM

One possibility to build a smaller conjunction (disjunction) is to stop the set covering greedy algorithm when there remains a few more training examples to be covered. Recall that all these examples belong to the same class: negative for conjunctions, positive for disjunctions. To avoid a "one-side" error generalized balls are permitted to misclassify examples: positive for conjunction, negative for disjunction.

Consider a conjunction's case. A generalized ball is allowed to misclassify positive examples if and only if much more negative examples are classified correctly by the same generalized ball. To characterize generalized ball h, a generalized ball's score $V_h(p)$ is introduced.

Let $|Q_h|$ be the number of negative examples correctly classified by h and $|R_h|$ be the number of positive examples misclassified by the same h. Then $V_h(p) = |Q_h| - p \times |R_h|$, where p is a fixed penalty parameter. Notice that no

misclassification of positive examples is allowed if p tends to infinity. Hence, the initial SCM is buit as $p \to \infty$.

Some changes are done in the SCM algorithm to implement $V_h(p)$ when p is fixed and finite. The Valiant step is not needed now. The learning begins directly with the Haussler step, which is also modified. Now a generalized ball $h \in \mathcal{S}$ is chosen with the highest score V_h. Examples covered by Q_h are removed from the set $\mathcal{N}$ of negative training examples and from each $Q_{g \neq h}$, and examples covered by R_h are removed from each $R_{g \neq h}$. The procedure of finding the generalized ball h with the largest score V_h and updating $\mathcal{N}$ and each $Q_{g \neq h}$ and each $R_{g \neq h}$ is repeated until only a few number of elements remain in $\mathcal{N}$.

15.35% of accuracy improvement is obtained with the SCM modifications on average for seven data sets: 20.3% on average for the SCM - disjunction and 10.4% on average for the SCM - conjunction. For results, see Table 2.2 (on page 23). The error column represents the sum of examples misclassified by the consistent SCM over all testing sets of 10-fold cross-validation. The $error_i$ column represents the sum of examples misclassified by the inconsistent SCM with a fixed p over all testing sets of 10-fold cross-validation. We have reported the optimal value of p and the optimal size obtained by early stopping the set covering greedy algorithm.

In Table 2.2 (on page 23) we compare performances of the SCM and the SVM. All results are obtained from [12].

For the SVM, the size is calculated as the average number of support vectors, built by the SVM over the 10 different training sets. The listed

Data		SCM - conjunc			SCM - disjunc			SVM			
data set	ex	p	size	$error_i$	p	size	$error_i$	γ	C	size	error
Brest Wisc	683	1.8	2	15	0.5	1	17	0.005	2	57.7	19
Bupa	345	1.7	20	122	2.8	9	106	0.002	0.2	265.9	107
Credit	653	0.8	4	197	1.2	4	194	0.0006	32	423.2	190
Glass	163	0.85	4	33	3.8	16	36	0.8	2	91.8	34
Haberman	294	1.4	1	71	1.4	12	71	0.01	0.6	146.4	71
Pima	768	1.1	3	189	4.1	20	204	0.002	1	526.2	203
Votes	52	1.2	1	6	0.9	1	6	0.05	15	18.2	3

Table 2.2: Optimal performances of the SCM-conjunction, the SCM-disjunction and the SVM

values of the kernel parameter γ and the soft margin parameter C give the smallest 10-fold cross validation error.

Chapter 3

Decision List Machines

We are now in a position to extend the SCM ideas to a more general machine called the Decision List Machine.

3.1 Definition of a decision list

The DL algorithm will construct a *decision list.*

For an input vector $\vec{x} \in X \subset \mathcal{R}^n$ we construct a s-dimensional Boolean vector $\vec{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_s(\vec{x}))$, where each $f_i(\vec{x})$ is called a feature.

For an input vector $\vec{x}$ the DLM outputs a *hypothesis* $h(\vec{x})$ defined as:

"if $f_1(\vec{x})$ then b_1

else if $f_2(\vec{x})$ then b_2

...

else if $f_p(\vec{x})$ then b_p

else b_{p+1}".

where $f_i(\vec{x}), i = 1, \ldots, p$ is the set of features returned by the learning algorithm, $b_i \in \{0, 1\}$, $b_p \neq b_{p+1}$ and b_{p+1} is the *default* value corresponding to the default feature.

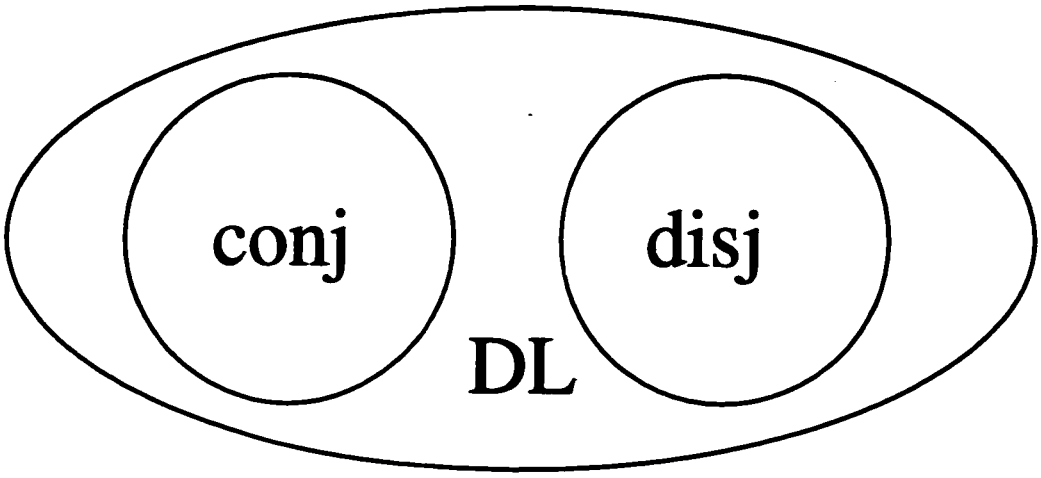
The output value of a decision list is equal to b_i, where b_i corresponds to the first satisfied condition $f_i(\vec{x}) = 1$.

Example 1 *We can represent the disjunction* $f_1(\vec{x}) \vee f_2(\vec{x}) \vee f_3(\vec{x})$ *as the decision list:*

$$\begin{aligned} &\text{"}\textit{if } f_1(\vec{x}) \textit{ then } 1 \\ &\textit{else if } f_2(\vec{x}) \textit{ then } 1 \\ &\textit{else if } f_3(\vec{x}) \textit{ then } 1 \\ &\qquad\qquad \textit{else } 0\text{"} \end{aligned}$$

Example 2 *We can represent the conjunction* $f_1(\vec{x}) \wedge f_2(\vec{x}) \wedge f_3(\vec{x})$ *as the decision list:*

$$\begin{aligned} &\text{"}\textit{if } \overline{f}_1(\vec{x}) \textit{ then } 0 \\ &\textit{else if } \overline{f}_2(\vec{x}) \textit{ then } 0 \\ &\textit{else if } \overline{f}_3(\vec{x}) \textit{ then } 0 \\ &\qquad\qquad \textit{else } 1\text{"} \end{aligned}$$



Let us construct an example of a decision list that cannot be represented either by a disjunction or by a conjunction.

Example 3 *An* alternating *decision list is:*

$$\begin{aligned} &\text{"}\textit{if } f_1(\vec{x}) \textit{ then } 1 \\ &\textit{else if } f_2(\vec{x}) \textit{ then } 0 \\ &\textit{else if } f_3(\vec{x}) \textit{ then } 1 \\ &\qquad\qquad \textit{else } 0\text{"} \end{aligned}$$

Hence, there exist functions that can be represented by decision list, but neither by a conjunction nor by a disjunction. Thus, decision lists strictly generalize these classes [8].

3.2 The feature space

Let $\mathcal{N}$ denote the set of all the negative training examples, and $\mathcal{P}$ denote the set of all the positive training examples. As before, $\mathcal{S}$ denotes a set of

features and f denotes an element of the feature space.

As for the SCM we want a practical machine which will work both with Boolean and non Boolean data. We assume the previous definitions about a feature. The decision function to learn is a decision list, and we call our machine *a decision list machine.*

Given a set of m training examples, the learning algorithm for the DLM must also be provided with a set $\mathcal{S}$ of s (Boolean valued) features. $\mathcal{S}$ must satisfy the following :

- there exists a conjunction of features in $\mathcal{S}$ which is consistent with all the training examples.
- there exists a disjunction of features in $\mathcal{S}$ which is consistent with all the training examples.

Since conjunctions and disjunctions are particular cases of decision lists, these two conditions imply that:

- there exists a decision list of features in $\mathcal{S}$ which is consistent with all the training examples.

A feature f_i is called *positive (negative)* if $b_i = 1$ ($b_i = 0$).

We say that features $f_i, f_{i+1} \ldots, f_{i+k}$ make up a *level* l if $b_{i-1} \neq b_i = \ldots = b_{i+k} \neq b_{i+k+1}$. The first level l_1 begins with f_1. A level l_i is called positive if it is constructed with positive features, and a level l_i is called negative if it is constructed with negative features. It is easy to see that a change of the order of features inside the same level does not affect the decision list's outcome.

3.3 Alternations and essential precedence

A decision list is said to be *k-alternating*, if there exist k pairs (b_i, b_{i+1}), such that $b_i \neq b_{i+1}$.

Consider a k-alternating decision list. Assume that level l_1 is constructed from positive features, and, therefore, level l_2 is constructed from negative features.

There is no negative example in $\mathcal{N}$ setting any $\forall f_1 \in l_1$ to 1, otherwise the decision list would be inconsistent with the example. Each $f_2 \in l_2$ has the property that any example $\vec{x} \in \mathcal{P}$ setting f_2 to 1 would also set a feature $f_1 \in l_1$ to 1. Hence, for any $f_2 \in l_2$ there is a set $l_{1p} \subseteq l_1$ of features f_1 such that each positive example in $\mathcal{P}$ setting f_2 to 1 sets one of the features in l_{1p} to 1.

Let us generalize the above analysis to an arbitrary negative level l_j and an arbitrary positive level l_i for i odd and j even.

For any feature $f_j \in l_j$, there exists a set l_{ip} of features from positive levels $l_i, i < j$, such that any positive example $x \in \mathcal{P}$ setting f_j to 1 sets one of features from l_{ip} to 1. Similarly, for an arbitrary positive level l_i it is true that any $f_i \in l_i$ has a corresponding set l_{jn} of features from negative levels $l_j, j < i$, such that any negative example $x \in \mathcal{N}$ setting f_i to 1 sets one of features from l_{jn} to 1.

We say that a feature f *covers* an example $\vec{x}$ if $\vec{x}$ sets f to 1.

We say that positive feature f_j *essentially precedes* negative feature $f_i, j < i$, iff there exists a positive training example $\vec{x}$ setting both features f_j and

f_i to 1.

We say that negative feature f_j *essentially precedes* positive feature $f_i, j < i$, iff there exists a negative training example $\vec{x}$ setting both features f_j and f_i to 1.

We say that feature f_i *essentially follows* feature $f_j, j < i$, iff f_j essentially precedes f_i.

We say that feature f_n is the *nearest* to feature f_j iff f_n essentially follows f_j and $n < i$ for all other f_i that essentially follows f_j.

3.4 The learning algorithm

We are going to generalize the greedy set covering algorithm for the SCM in a natural way.

Recall that, for the set $\mathcal{S}$ of features, the following is fulfilled:

- there exists a conjunction of features $f_i(\vec{x}), i \in A' \subseteq \mathcal{S}$, which is consistent with all the training examples .
- there exists a disjunction of features $f_i(\vec{x}), i \in A'' \subseteq \mathcal{S}$, which is consistent with all the training examples.

Let $\mathcal{P}$ be the set of all the positive training examples. Let $\mathcal{N}$ be the set of all the negative training examples.

Step 1: Find the set of features $\mathcal{S}_n$ which make zero error on the remaining positive training examples. For each feature $f_i \in \mathcal{S}_n$ define a set Q_i of the consistent negative training examples. We can define such a set because

of the assumption that there exists a conjunction which is consistent with all the training examples .

Step 2: Find the set of features $\mathcal{S}_p$ which make zero error on the remaining negative training examples. For each feature $f_i \in \mathcal{S}_p$ define a set R_i of the consistent positive training examples. We can define such a set because of the assumption that there exists a disjunction which is consistent with all the training examples.

Step 3: Compare the sizes of each Q_i and each R_i. Let T_i be the set of the largest size. If T_i contains positive examples, then begin building of a positive level (step 4), otherwise, begin building of a negative level (step 7).

Step 4 (a positive level): Place the positive feature f_i, corresponding to T_i in a positive level. Remove covered examples from $\mathcal{P}$ and every R_j.

Step 5: Test if f_i essentially follows some negative features f_j of previous negative levels. For each f_j that essentially precedes f_i assign a link to f_i.

Step 6: If there are some uncovered examples in $\mathcal{P}$ then go to step 1.Otherwise go to the pruning step (step 10).

Step 7 (a negative level): Place the negative feature f_i, corresponding to T_i in a negative level. Remove covered examples from $\mathcal{N}$ and every Q_j.

Step 8: Test if f_i essentially follows some positive features f_j of previous positive levels. For each f_j that essentially precedes f_i assign a link to f_i.

Step 9: If there are some uncovered examples in $\mathcal{N}$ then go to step 1.Otherwise go to the pruning step (step 10).

Step 10 (pruning step): Remove every feature f that has the same sign as the default feature and that does not essentially precede an other

feature.

It is easy to see that the DLM learning algorithm correctly classifies an arbitrary training example $\vec{x}_i$. Suppose that $\vec{x}_i$ is positive. Assume for the purpose of contradiction that the hypothesis classifies $\vec{x}_i$ as negative. Thus there exists a negative feature $f_i \in \mathcal{I}_n$ which is set to 1 by $\vec{x}_i$ but this is impossible according to the learning algorithm.

A proof for a negative example $\vec{x}_j$ follows by symmetry.

3.5 The DLM with generalized balls

Recall, that on any training example $\vec{x}$ we define the value of a generalized ball as following:

$$f_{i,\rho} = f_{i,\rho}(\vec{x}) = \begin{cases} y_i & \text{if } d(\vec{x}_i, \vec{x}) \leq \rho \\ \overline{y_i} & \text{otherwise} \end{cases}$$

where $\vec{x}_i$ is a centre of the ball and has a label y_i.

No initial restrictions are imposed on the radius ρ, hence, for a given set of m training examples the initial space $\mathcal{S}$ of features could be infinite.

For practical purpose we consider that each training example $\vec{x}_i$ defines $m-1$ radii $\rho_{i,j} = d(\vec{x}_i, \vec{x}_j), i \neq j$. This consideration restricts the number of generalized balls, built on m training examples, by $\mathcal{O}(m^2)$. Assume, that the training set does not contain any contradictory examples, or if $\vec{x}_i = \vec{x}_j$ then $y_i = y_j$ is fulfilled. $\mathcal{S}$ has the required property that there exists a decision list of some features in $\mathcal{S}$ which is consistent with all the training examples.

We now show that we only need to consider $2m$ balls in the initial set $\mathcal{S}$ of balls.

For a positive training example $\vec{x}_i$ consider two sets of generalized balls, centered on $\vec{x}_i$: $\mathcal{P}_i$ and $\mathcal{N}_e$. $\mathcal{P}_i$ is a set of generalized balls, each of them includes all the positive training examples. $\mathcal{N}_e$ is a set of generalized balls, each of them excludes all the negative training examples.

Let $\rho_{i,min}$ denote the smallest distance from a positive training example $\vec{x}_i$, that includes all the positive training examples. Let $\rho_{i,max}$ denote the largest distance from a positive training example $\vec{x}_i$, that excludes all the negative training examples.

For a negative training example $\vec{x}_j$ consider two sets of generalized balls, centered on $\vec{x}_j$: $\mathcal{N}_i$ and $\mathcal{P}_e$. $\mathcal{N}_i$ is a set of generalized balls, each of them includes all the negative training examples. $\mathcal{P}_e$ is a set of generalized balls, each of them excludes all the positive training examples.

Let $\rho_{j,min}$ denote the smallest distance from a negative training example $\vec{x}_j$, that includes all the negative training examples. Let $\rho_{j,max}$ denote the largest distance from a negative training example $\vec{x}_j$, that excludes all the positive training examples.

To build a positive level of a hypothesis, the learning algorithm chooses the balls which misclassify as few negative examples as possible. Consider a positive example $\vec{x}_i$. A generalized ball with the centre $\vec{x}_i$ and a radius larger than $\rho_{i,min}$ will misclassify more negative examples than the generalized ball with the centre $\vec{x}_i$ and the radius $\rho_{i,min}$.

Hence, for this centre $\vec{x}_i$, we will consider the radius $\rho_{i,min}$ in the initial

set $\mathcal{S}$ of features.

Similarly, for a negative example $\vec{x}_j$ a generalized ball with the centre $\vec{x}_j$ and a radius smaller than $\rho_{j,max}$ will misclassify more negative examples than the generalized ball with the centre $\vec{x}_j$ and the radius $\rho_{j,max}$.

Hence, for this centre $\vec{x}_j$, we will consider the radius $\rho_{j,max}$ in the initial set $\mathcal{S}$ of features.

To build a negative level of a hypothesis, the learning algorithm chooses the balls which misclassify as few positive examples as possible. Consider a negative example $\vec{x}_j$. A generalized ball with the centre $\vec{x}_j$ and a radius larger than $\rho_{j,min}$ will misclassify more positive examples than the generalized ball with the centre $\vec{x}_j$ and the radius $\rho_{j,min}$.

Hence, for this centre $\vec{x}_j$, we will consider the radius $\rho_{j,min}$ in the initial set $\mathcal{S}$ of features.

Similarly, for a positive example $\vec{x}_i$ a generalized ball with the centre $\vec{x}_i$ and a radius smaller than $\rho_{i,max}$ will misclassify more positive examples than the generalized ball with the centre $\vec{x}_i$ and the radius $\rho_{i,max}$.

Hence, for this centre $\vec{x}_i$, we will consider radius $\rho_{i,max}$ in the initial set $\mathcal{S}$ of features. .

Hence, the number of generalized balls in $\mathcal{S}$ is just twice of the number of training examples.

3.6 Model selection

For the same reasons as SCM, we have implemented the early stopping of the greedy algorithm and used penalty parameters in the DLM learning algorithm.

We allow a negative generalized ball f_n to misclassify some positive examples if and only if many more negative examples are covered. To characterize a negative generalized ball f_n we introduce a generalized ball's score $V_{f_n}(p_p)$. Let $|Q_{f_n}|$ be the number of negative examples correctly classified by f_n, and $|R_{f_n}|$ be the number of positive examples misclassified by the same f_n. Then $V_{f_n}(p_p) = |Q_{f_n}| - p_p \times |R_{f_n}|$, where p_p is a fixed positive penalty parameter.

We allow a positive generalized ball f_p to misclassify some negative examples if and only if many more positive examples are covered. To characterize a positive generalized ball f_p we introduce a generalized ball's score $V_{f_p}(p_n)$. Let $|R_{f_p}|$ be the number of positive examples correctly classified by f_p, and $|Q_{f_p}|$ be the number of negative examples misclassified by the same f_p. Then $V_{f_p}(p_n) = |R_{f_p}| - p_n \times |Q_{f_p}|$, where p_n is a fixed negative penalty parameter.

Again notice, that no misclassification of positive examples is allowed if p_p tends to infinity, and no misclassification of negative examples is allowed if p_n tends to infinity. Hence, we recover the initial DLM algorithm when $p_p \to \infty$ and $p_n \to \infty$.

We compare scores of generalized balls and choose a generalized ball f with the highest score V_f. Then we remove examples covered by Q_f from the set $\mathcal{N}$ of the negative training examples and from each $Q_{g \neq f}$. We remove

examples covered by R_f from the set $\mathcal{P}$ of the positive training examples and from each $R_{g \neq f}$. We repeat the procedure of finding the generalized ball f with the largest score V_f and updating sets $\mathcal{N}, \mathcal{P}, Q_{g \neq f}, R_{g \neq f}$ until only a few number of examples remain in either $\mathcal{N}$ or $\mathcal{P}$.

In the next chapter we show that this modified DLM algorithm improves the accuracy of the output hypothesis as well as it reduces its size.

Chapter 4

Empirical Results

To investigate the learning ability of the DLM we ran it on more than twenty artificial data sets and on seven "natural" data sets as well.

4.1 Results on artificial data

The goal of running the DLM on artificial data was to compare in a controllable way the performances of the DLM and the SCM. Both algorithms use spaces of generalized balls.

We generated more than twenty artificial data sets, mostly "wave-like" and "doughnut-like", in two-, three- and four-dimensions.

Each data set was used as a training set and then as a testing set. We ran versions of the DLM and the SCM algorithms with zero training errors, and searched for differences in sizes of the output hypothesis.

On the following data of 18 two-dimensional examples (see Figure 4.1) the

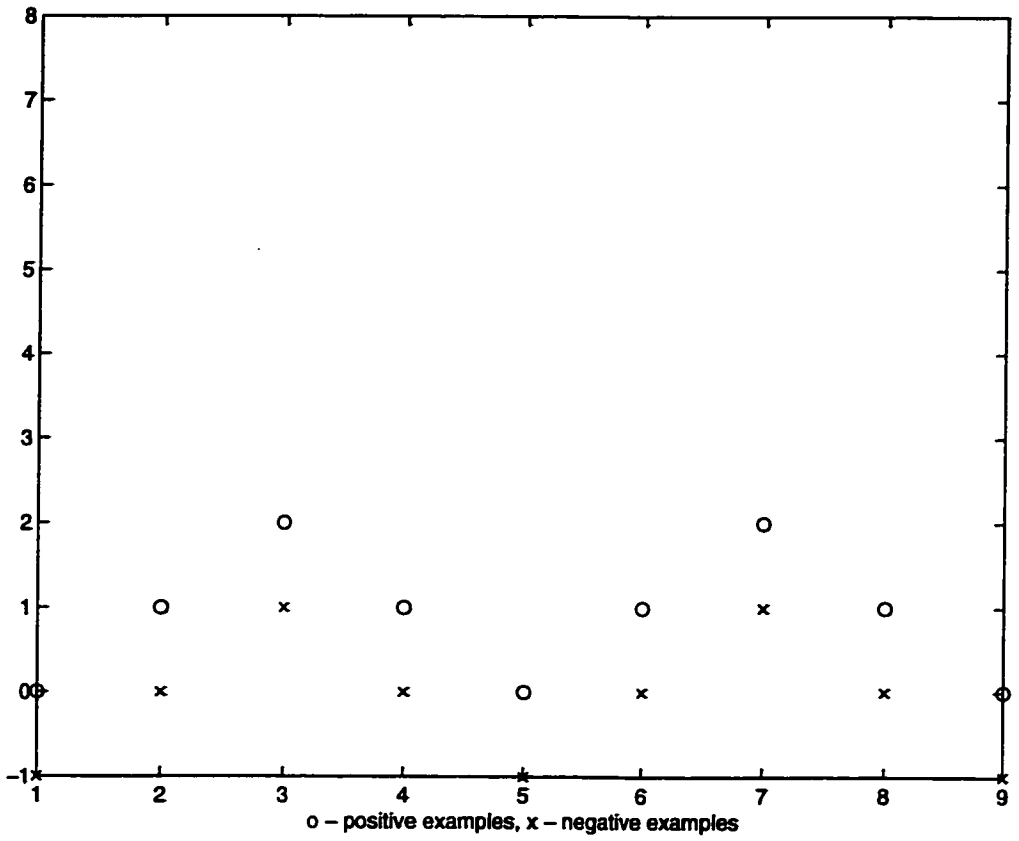


Figure 4.1: 2D artificial data

hypothesis output by the DLM contains four generalized balls. For this data set, the SCM-conjunction and the SCM-disjunction construct hypotheses which contain eight generalized balls each.

We expanded this data set to a three-dimensional data set of 54 examples (see Figure 4.2), and obtained even more impressive results. The DLM builds a hypothesis with nine generalized balls in order to classify all the training examples correctly. The SCM-conjunction builds a conjunction with twenty

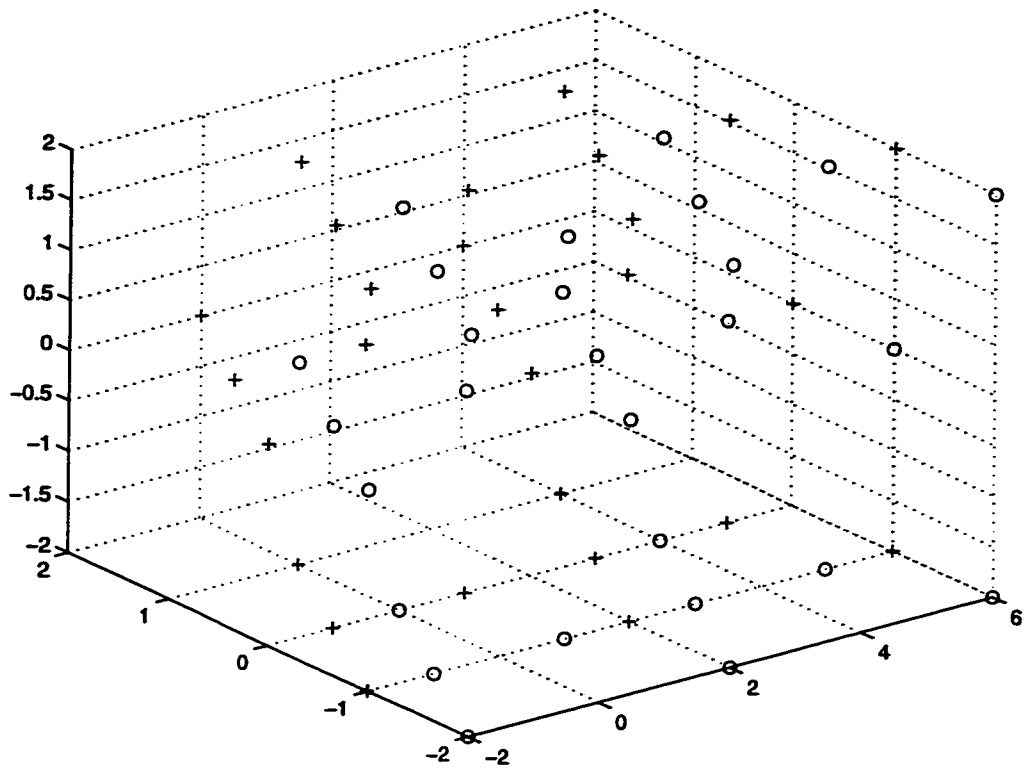


Figure 4.2: 3D artificial data

four generalized balls, and the SCM-disjunction builds a disjunction with twenty five generalized balls.

With a four-dimensional modification of the same data set, the DLM classifies correctly 108 examples of a "wave-like" data with the decision list of twenty one generalized ball, whereas the SCM-conjunction classifies the same data with forty eight generalized balls and the SCM-disjunction classifies it with fifty generalized balls.

On most of data sets the DLM builds a hypothesis which contains less

Data			DLM	SCM - c	SCM - d
pos ex	neg ex	dim	size	size	size
9	7	2	3	7	4
9	9	2	4	8	8
26	7	2	3	3	3
20	20	2	4	6	6
22	20	2	5	6	7
23	20	2	4	6	8
32	52	2	8	12	12
27	27	3	9	24	25
54	54	4	21	48	50
81	81	4	32	76	75

Table 4.1: Sizes of the DLM and the SCM on the artificial data

generalized balls than the SCM-conjunction and the SCM-disjunction. One exception where the DLM hypothesis has the same number of generalized balls as the SCM-conjunction hypothesis and the SCM-disjunction hypothesis is represented by a "doughnut-like" data set with a large number of concentrated positive examples among a few widely dispersed negative examples.

In Table 4.1 we compare sizes the DLM, the SCM-conjunction and the SCM-disjunction on some of the generated data sets.

Data		*DLM*	
data set	number of ex	size	N_{ep}
Breast Wisc	683	14	14
Bupa	345	66	66
Credit	653	145	144
Glass	163	21	21
Haberman	294	52	52
Pima	768	155	155
Votes	52	4	4

Table 4.2: Sizes of the DLM for the case of zero empirical risk

4.2 Results on "natural" data

On seven "natural" data sets we have used 10-fold cross-validation for the estimate of the generalization error. As it was written before, we obtained these data sets from the machine learning repository at UCI [20] except the Glass data set which was obtained from Robert Holte, now at the University of Alberta.

In the Table 4.2, in the "size" column, we give the number of generalized balls obtained by consistent machines (the case of zero empirical risk) when they are trained on the full data set. N_{ep} denotes the number of generalized balls with essential precedence. Notice that this number is the same as the size of a hypothesis for each data set (except for the Credit data set).

Hence, for most experiments, we ran the simplified DLM where the prun-

ning step was skipped. Hence the algorithm's steps 5, 8, 10 were omitted.

In table 4.3 the size s of a decision list is given as the average number of generalized balls, built by the DLM over the 10 different training sets of 10-fold cross validation. To calculate distances we have used the L_2 metric. The error column represents the sum of misclassified examples over all testing sets of 10-fold cross-validation.

The optimal values of penalty parameters are reported in column p_{pos} and p_{neg} respectively.

On the given data set $\mathcal{T}$ we consider a hypothesis h as a function of parameters s, p_p, p_n. Let $\mathcal{H} = \mathcal{H}_{\mathcal{T}}(s, p_p, p_n)$ denote a set of hypotheses built by the inconsistent DLM for the given data set $\mathcal{T}$. Our goal is to find such hypothesis $h' \in \mathcal{H}$ that $e(h') = min_{h \in \mathcal{H}} e(h)$. Thus we should find a such set of parameters s', p'_p, p'_n that for $h' = h(s', p'_p, p'_n)$
$e(h') = min_{h \in \mathcal{H}} e(h)$ is fulfilled.

The DLM which makes zero error on the training set is called the *consistent* DLM. The consistent DLM with implemented early stopping is called the *truncated* DLM. The DLM which generalized balls are allowed to make errors on the training set is called the *inconsistent* DLM.

We would like to compare the errors made by the consistent DLM, the truncated DLM_{tr} (with early stopping), and the inconsistent DLM_i. It is easy to see that the implementation of early stopping reduces errors (Table 4.4).

To compare the DLM performance with the performances of other batch learning algorithms (for definition see 2.1 on page 13) we compare the small-

Data		DLM_i			
data set	number of ex	s	error	p_{pos}	p_{neg}
Breast Wisc	683	2	14	2.1	1
Bupa	345	4	108	2	2
Credit	653	11	197	1	-
Glass	163	3	29	0.8	3.1
Haberman	294	8	70	-	1.7
Pima	768	6	189	1.5	1.5
Votes	52	1	6	-	-

Table 4.3: Performance of the inconsistent DLM_i

Data		DLM	DLM_{tr}	DLM_i
data set	number of ex	error	error	error
Breast Wisc	683	30	23	14
Bupa	345	126	123	108
Credit	653	242	228	197
Glass	163	36	34	29
Haberman	294	126	101	70
Pima	768	251	232	189
Votes	52	7	6	6

Table 4.4: Performances of different types of the DLM

est misclassification errors received by the DLM, the SCM-conjunction, the SCM-disjunction, the nearest neighbor classifier (NNC) and the SVM . Results presented for the NNC and the SVM performances are taken from [12].

For every data set the training set and the testing set, obtained in the 10-fold cross-validation process, were the same for each learning algorithm. Hence each machine was trained on the same training set and tested on the same testing set. We emphasize the smallest error obtained on each data set.

Data		DLM	SCM-c	SCM-d	NNC	SVM
data set	number of ex	error	error	error	error	error
Breast Wisc	683	*14*	15	17	29	19
Bupa	345	108	122	*106*	124	107
Credit	653	197	197	194	214	*190*
Glass	163	*29*	33	36	36	34
Haberman	294	*70*	71	71	107	71
Pima	768	*189*	*189*	209	247	203
Votes	52	6	6	6	7	*3*

Table 4.5: Smallest errors obtained by the DLM, the SCM, NNC, and the SVM

4.3 Results on "natural" data in the case of asymmetric loss

We assumed before that the cost assigned to misclassification of each class was equal to 1. We also know that some learning algorithms do not perform well for asymmetric loss coefficients (for example, C4.5). However, in many real-life applications the different types of misclassifications have unequal costs. For example, consider detection of a brain tumor. A false positive (detecting a tumor when there is none) is less costly than a false negative (notdetecting a tumor when there is one). Large asymmetry in costs can be handled by permitting *one-side* training errors only but, in general, we do need to be able to make training errors on both classes [14] .

Unequal costs of misclassification can be described by loss coefficients l_p and l_n, where l_p is the cost of misclassification of a positive example, and l_n is the cost of misclassification of a negative example.

Given a training example $\vec{x}$ labeled y, the loss function for a hypothesis h is defined as

$$l(\vec{x}, h(\vec{x})) = \begin{cases} l_p & \text{if } h(\vec{x}) = 0, y = 1 \\ l_n & \text{if } h(\vec{x}) = 1, y = 0 \end{cases}$$

Again, on the given data set $\mathcal{T}$ we consider a hypothesis h as a function of parameters s, p_p, p_n. Let $\mathcal{H} = \mathcal{H}_{\mathcal{T}}(s, p_p, p_n)$ denote a set of hypotheses built by the inconsistent DLM_i for the given data set $\mathcal{T}$. Our goal is to find such hypothesis $h' \in \mathcal{H}$ that for fixed l_p, l_n $L(h') = min_{h \in \mathcal{H}} L(h)$. Thus we should

find a such set of parameters s', p'_p, p'_n that for $h' = h(s', p'_p, p'_n)$ $L(h') = min_{h \in \mathcal{H}} L(h)$ is fulfilled.

We are especially interested to investigate cases when one of coefficients is substantially greater than the other one. It is easy to predict that the inconsisted DLM produces a hypothesis of a smaller size and with a lesser loss than a hypothesis constructed by the consistent DLM. For results of the DLM performance on the Glass data set see table 4.6 on page 46. Results on the Credit data set and the Haberman data set are given in Appendix B.

As before we apply the 10-fold cross-validation method, and use the metric L_2 to calculate distances. The size of a decision list is given as the average number of generalized balls built by the DLM over the 10 different training sets of 10-fold cross-validation. The L column represents the sum of loss functions over all testing sets of 10-fold cross-validation. For the DLM_i the err_p and err_n columns represent the sums of errors on the positive and the negative examples respectively over all testing sets of 10-fold cross-validation.

We also compared the values of the asymmetric loss obtained by the DLM, the SCM, and a trivial classifier (TCM). TCM-pos denotes a trivial classifier which classifies all the training examples as positive, and TCM-neg denotes a trivial classifier which classifies all the training examples as negative.

We plot the results of the DLM, the SCM-conjunction, the SCM-disjunction, for different loss coefficients on the Glass data set on figure 4.3 of page 46. It is easy to see that our algorithm is robust w.r.t. the asymmetry of loss coefficients.

Coef		DLM		DLM_{tr}		DLM_i					
l_p	l_n	L	size	L	size	p_p	p_n	err_p	err_n	L	size
1	100	1917	17.8	466	1	1	-	68	3	368	2
1	10	207	17.8	106	1	1	-	42	4	82	3
1	5	112	17.8	86	1	3	-	37	5	62	3
1	4	93	17.8	82	13	0.8	3.1	24	8	56	2
4	1	87	17.8	78	4	-	1	4	40	56	4
5	1	104	17.8	85	4	-	1	4	40	60	4
10	1	189	17.8	120	4	-	1	4	40	80	4
100	1	1719	17.8	750	4	-	1	4	40	440	4

Table 4.6: The DLM performance for the asymmetric loss on Glass (87 positive and 76 negative examples)

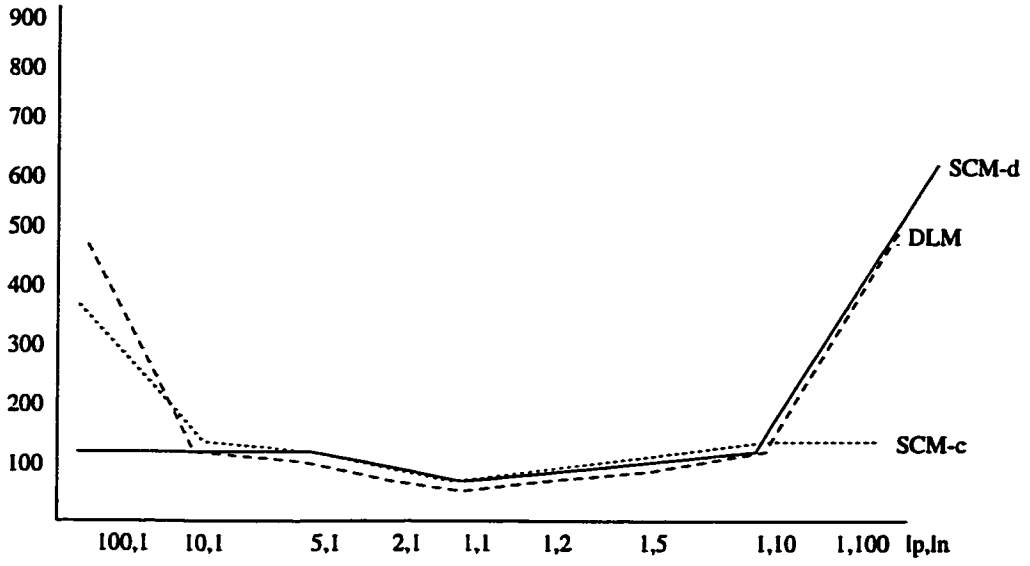


Figure 4.3: Results on the Glass data set for the asymmetric loss

Chapter 5

Theoretical Results

The empirical results of the previous chapter have shown that, in practice, the DLM performance is comparable and sometimes better than those of the SCM, and the SVM. In this chapter we provide a theoretical assessment of the DLM performance.

Two parameters characterize a hypothesis h output by the DLM. They are: the number of alternations k, and the number of features r involved in h. We try to bound the generalization error in terms of these parameters. To obtain such bounds, we apply a technique due to Littlestone and Warmuth [10]. According to Littlestone and Warmuth, if a hypothesis can be reconstructed from a small subset of training examples, then the generalization error can be expressed in terms of the size of this subset. In our case, the size of such subset depends on k and r.

We assume that each $\vec{x}_i$ (both in the training and the testing sets) is i.i.d. and originating from the same (unknown) statistical distribution [1].

5.1 The compression scheme for the DLM

The learning model is the following: consider an arbitrary n-dimensional input space X. For a fixed integer m, let $U = (X \times \{0,1\})^m$ be the set of training sets of size m. Let $\mathcal{T}$ be an arbitrary element of U.

Let A be the DLM learning algorithm using generalized balls. Given a training set $\mathcal{T}$, A constructs a hypothesis h with k alternations and r features.

To obtain bounds on the generalization error of A we extend the *sample compression and reconstruction* technique introduced by Littlestone and Warmuth [10]. See also Floyd and Warmuth [5], Marchand and Shawe-Taylor [12].

For the given learning algorithm A, an extended sample compression scheme Λ consists of a *compression function* Λ_c and a *reconstruction function* Λ_r. Given a training set $\mathcal{T}$, the compression function Λ_c maps it to an ordered sequence of small subsets which we call a *compression set* Λ_s.

Let us first recall some notions.

Alternations appear in a hypothesis where generalized balls change their signs. A set of generalized balls between two alternations is called a level. We call a level "positive" if its component generalized balls have a pure positive region. If its component generalized balls have a pure negative region, then the corresponding level is called "negative".

Consider the construction of a positive level feature. The decision list machine defines the radii of such features through negative border points. The border points remain in the data when constructing such features. Centres

of generalized balls that exclude positive examples are negative and remain in the data after constructing. We call such centers *remaining.*

On the opposite, centres of generalized balls that include positive examples are positive. They are removed from the data when constructing such features. We call such centers *removable.*

Construction of a negative feature follows by symmetry.

The compression set Λ_s represents an ordered sequence of sets

$\Lambda_s(\mathcal{T}) = (\Lambda'_1(\mathcal{T}), \Lambda''_1(\mathcal{T}), \Lambda'''_1(\mathcal{T}), \ldots, \Lambda'_k(\mathcal{T}), \Lambda''_k(\mathcal{T}), \Lambda'''_k(\mathcal{T}))$,

where sets $\Lambda'_i(\mathcal{T}), \Lambda''_i(\mathcal{T}), \Lambda'''_i(\mathcal{T})$ correspond to the ith level l_i of h:

$\Lambda'_i(\mathcal{T}) \subset \mathcal{T}$ is the set of all border points in l_i,

$\Lambda''_i(\mathcal{T}) \subset \mathcal{T}$ is the set of all removable centers in l_i

$\Lambda'''_i(\mathcal{T}) \subset \mathcal{T}$ is the set of all remaining centers in l_i.

$\sum_{i=1}^{k} |\Lambda'_i| = \sum_{i=1}^{k} (|\Lambda''_i| + |\Lambda'''_i|) = r$.

Let $\mathcal{P}$ be the set of all the positive training examples. Let $\mathcal{N}$ be the set of all the negative training examples.

If l_i is a positive level, then $\Lambda'_i \subset \mathcal{N}$, $\Lambda''_i \subset \mathcal{P}$, $\Lambda'''_i \subset \mathcal{N}$.

If l_i is a negative level, then $\Lambda'_i \subset \mathcal{P}$, $\Lambda''_i \subset \mathcal{N}$, $\Lambda'''_i \subset \mathcal{P}$.

The reconstruction function Λ_r maps every possible compression set Λ_s to a hypothesis h such that for any $\mathcal{T} \in U$

$$A(\mathcal{T}) = h = \Lambda_r(\Lambda_s(\mathcal{T}))$$

is fulfilled.

Lemma 2 *Let $\mathcal{T}$ be the training set. Let A be the DLM learning algorithm using generalized balls. Let Λ_c be the compression function defined above. Then there exists a reconstruction function Λ_r satisfying $A(\mathcal{T}) = \Lambda_r(\Lambda_s(\mathcal{T}))$.*

Proof: The reconstruction function Λ_r given the compression set Λ_s constructs the hypothesis h by reconstructing levels of h in a given order. The reconstruction begins from the first level and goes till the kth level.

For each level l_i Λ_r creates generalized balls in the following way:

1. for each $x_i \in \Lambda_i''$ it constructs a generalized ball centered in x_i with radius $d = max_{j \in \Lambda_i'} d(x_i, x_j)$

2. for each $x_i \in \Lambda_i'''$ it constructs a generalized ball centered in x_i with radius $d = min_{j \in \Lambda_i'} d(x_i, x_j)$

If Λ_i' is a set of negative examples, then Λ_r builds a positive level of these generalized balls. If Λ_i' is a set of positive examples, then Λ_r builds a negative level of these generalized balls

Levels are reconstructed according to the order they were output in the hypothesis. For each level of the hypothesis the same centers are used in $\Lambda_r(\Lambda_s(\mathcal{T}))$ as in $A(\mathcal{T})$, and radii corresponding to them are the same. Hence, $A(\mathcal{T}) = \Lambda_r(\Lambda_s(\mathcal{T}))$.

□

Recall that the error of a hypothesis h, built by a learning algorithm A, is the probability $err(h)$ that a randomly drawn example $\vec{x}$ is misclassified by

h (here we assume that the cost of misclassification of each class is equal to 1).

Denote $|\Lambda_1'| = d_1', |\Lambda_1''| = d_1'', |\Lambda_1'''| = d_1''', \ldots, |\Lambda_k'| = d_k', |\Lambda_k''| = d_k'', |\Lambda_k'''| = d_k''', d_0'' = 0$.

Let α be the set of indexes of odd levels of the hypothesis, β be the set of indexes of even levels of the hypothesis. Let F be the set of training examples of the same class as the border points of the first level of the hypothesis.

To simplify notations, we will use the following definitions:

$$A_i = \binom{|F| - \sum_{j=1}^{\frac{i-1}{2}} d_{2j}''}{d_i'} \binom{m - |F| - \sum_{j=1}^{\frac{i-1}{2}} d_{2j-1}''}{d_i''} \binom{|F| - \sum_{j=1}^{\frac{i-1}{2}} d_{2j}''}{d_i'''},$$

$$B_i = \binom{m - |F| - \sum_{j=1}^{\frac{i}{2}} d_{2j}''}{d_i'} \binom{|F| - \sum_{j=0}^{\frac{i-2}{2}} d_{2j}''}{d_i''} \binom{m - |F| - \sum_{j=1}^{\frac{i}{2}} d_{2j}''}{d_i'''}$$

Theorem 1 *Let Λ_r be the reconstruction function defined above. Then the probability that a training set of m examples contains a compression set Λ_s defined above with $2r \leq m$ examples and $|\alpha|$ odd levels and $|\beta|$ even levels that is reconstructed via Λ_r to a hypothesis h that is both consistent with all m examples and has an error larger than ϵ is at most*

$$(\prod_{i \in \alpha} A_i)(\prod_{i \in \beta} B_i)(1 - \epsilon)^{m-2r}.$$

Proof:

Let $\mathcal{A}$ be an event that a decision list hypothesis h is both consistent with all m training examples and has an error larger than ϵ.

For a particular compression set of size $2r \leq m$ we calculate the upper bound on the probability that with the given conditions, the hypothesis h with generalization error larger than ϵ, is consistent with the remaining $m-2r$

examples. Given that $err(h) = P(h(x) \neq c(x)) > \epsilon$, it is easy to see that the probability that the hypothesis is consistent with a random example x is less than $1 - \epsilon$. Hence the probability that the hypothesis h built via Λ_r is consistent with $m - 2r$ examples is less than $(1 - \epsilon)^{m-2r}$.

Let's calculate how many compression sets of size $2r$ we can build. Note that, while constructing each level, we actually delete only removable centres.

Let $\Upsilon_{d_1'}$ be the collection of d_1'-element subsets of a set of $|F|$ examples. Recall that these subsets contain the border points. The size of $\Upsilon_{d_1'}$ is equal to $\binom{|F|}{d_1'}$.

Let $\Upsilon_{d_2'}$ be the collection of d_2'-element subsets of a set of $m - |F| - d_1''$ examples. The size of $\Upsilon_{d_2'}$ is equal to $\binom{m-|F|-d_1''}{d_2'}$.

For an odd i let $\Upsilon_{d_i'}$ be the collection of d_i'-element subsets of $|F| - \sum_{j=1}^{\frac{i-1}{2}} d_{2j}''$ examples. Its size is equal to $\binom{|F|-\sum_{j=1}^{\frac{i-1}{2}} d_{2j}''}{d_i'}$. For an even i $\Upsilon_{d_i'}$ be the collection of d_i'-element subsets of $m - |F| - \sum_{j=1}^{\frac{i}{2}} d_{2j-1}''$ examples. There are exactly $\binom{m-|F|-\sum_{j=1}^{\frac{i}{2}} d_{2j-1}''}{d_i'}$ such subsets.

Let $\Upsilon_{d_1''}$ be the collection of d_1''-element subsets of $m-|F|$ examples. Recall that these subsets contain removable centers.There are exactly $\binom{m-|F|}{d_1''}$ such subsets.

Let $\Upsilon_{d_2''}$ be the collection of d_2''-element subsets of $|F|$ examples. There are exactly $\binom{|F|}{d_2''}$ such subsets.

For an odd i if $\Upsilon_{d_i''}$ is the collection of d_i''-element subsets of $m - |F| - \sum_{j=1}^{\frac{i-1}{2}} d_{2j-1}''$ examples, then its size is equal to $\binom{m-|F|-\sum_{j=1}^{\frac{i-1}{2}} d_{2j-1}''}{d_i''}$. For an even i if $\Upsilon_{d_i''}$ is the collection of d_i''-element subsets of $|F| - \sum_{j=0}^{\frac{i-2}{2}} d_{2j}''$ examples,

then its size is equal to $\binom{|F|-\sum_{j=0}^{\frac{i-2}{2}} d''_{2j}}{d''_i}$

Let $\Upsilon_{d''_1}$ be the collection of d'''_1-element subsets of $|F|$ examples. Recall that these subsets contain remaining centers. There are exactly $\binom{|F|}{d'''_1}$ such subsets.

Let $\Upsilon_{d''_2}$ be the collection of d'''_2-element subsets of $m-|F|-d''_1$ examples. There are exactly $\binom{m-|F|-d''_1}{d'''_2}$ such subsets.

For an odd i if $\Upsilon_{d''_i}$ is the collection of d'''_i-element subsets of $|F|-\sum_{j=1}^{\frac{i-1}{2}} d''_{2j}$ examples, then its size is equal to $\binom{|F|-\sum_{j=1}^{\frac{i-1}{2}} d''_{2j}}{d'''_i}$. For an even i if $\Upsilon_{d''_i}$ is the collection of d'''_i-element subsets of $m-|F|-\sum_{j=1}^{\frac{i}{2}} d''_{2j-1}$ examples, then its size is equal to $\binom{m-|F|-\sum_{j=1}^{\frac{i}{2}} d''_{2j-1}}{d'''_i}$.

Let Υ be the collection of $\Upsilon_{d'_i}, \Upsilon_{d''_i}, \Upsilon_{d''_i}$ collections, $i \in \alpha \cup \beta$. Its size is equal to

$$\prod_{i\in\alpha} \binom{|F|-\sum_{j=1}^{\frac{i-1}{2}} d''_{2j}}{d'_i} \binom{m-|F|-\sum_{j=1}^{\frac{i-1}{2}} d''_{2j-1}}{d''_i} \binom{|F|-\sum_{j=1}^{\frac{i-1}{2}} d''_{2j}}{d'''_i} \times$$
$$\prod_{i\in\beta} \binom{m-|F|-\sum_{j=1}^{\frac{i}{2}} d''_{2j}}{d'_i} \binom{|F|-\sum_{j=0}^{\frac{i-2}{2}} d''_{2j}}{d''_i} \binom{m-|F|-\sum_{j=1}^{\frac{i}{2}} d''_{2j}}{d'''_i}$$

or

$$(\prod_{i\in\alpha} A_i)(\prod_{i\in\beta} B_i)$$

Thus the probability of drawing m examples, such that there is a compression set of $2r$ examples and the corresponding hypothesis is both consistent with all m examples and has error larger than ϵ is at most

$$(\prod_{i\in\alpha|} A_i)(\prod_{i\in\beta} B_i)(1-\epsilon)^{m-2r}.$$

□

This theorem can be extended to the case where the hypothesis is allowed to misclassify some training examples.

Let k_i' denote the number of errors on examples of the class F on the ith level. Let k_i'' denote the number of errors on examples of the opposite class on the ith level. Let $\sum_{i\in\alpha\cup\beta}(k_i' + k_i'') = q$.

For ith level let M_i' be a set of borders and centres of type F of previous levels. For ith level let M_i'' be a set of borders and centres of opposite type of previous levels.

Denote the number of ways to choose errors from class F:

$$K_i' = \binom{|F|-|M_i'|}{k_i'}.$$

Denote the number of ways to choose errors from the opposite class:

$$K_i'' = \binom{m-|F|-|M_i''|}{k_i''}$$

Corollary 1 *Let Λ_r be the reconstruction function defined above. Let $\sum_{i\in\alpha\cup\beta}(k_i'+k_i'') = q$ be the number of examples of the training set $\mathcal{T}$ that are misclassified by h. Then the probability that a training set of m examples contains a compression set Λ_s defined above with $2r \leq m$ examples and $|\alpha|$ odd levels and $|\beta|$ even levels that is reconstructed via Λ_r to a hypothesis h that makes q errors on m examples and has error larger than ϵ is at most*

$$\prod_{i\in\alpha}(A_i(K_i' + K_i''))\prod_{i\in\beta}(B_i(K_i' + K_i''))(1-\epsilon)^{m-2r-q}.$$

Proof: Apply the proof of Theorem 1, and use the fact that all misclassified examples belong to remaining $m - 2r$ examples.

□

5.2 Generalization error bounds

To obtain bounds on the generalization error of the hypothesis we begin with a proof of the "basic" theorem when all training examples are correctly classified by the hypothesis.

Let h be a hypothesis built by the DLM using generalized balls that contains r features and k alternations.

Let $k = |\alpha| + |\beta|$, where $|\alpha|$ is the number of odd levels, and $|\beta|$ is the number of even levels.

Theorem 2 *Let h be the hypothesis defined above. Then with probability $1 - \delta$ over random training sets of size m, the generalization error $\epsilon(h)$ of the hypothesis h is at most*

$$\frac{1}{m-2r}\left(\sum_{i\in\alpha} lnA_i + \sum_{i\in\beta} lnB_i + ln\left(\frac{1}{\delta_D}\right)\right)$$

where $\delta_D = \left(\frac{\pi^2}{6}\right)^{-3(|\alpha|+|\beta|)}\left(\prod_{i\in\alpha\cup\beta}(d_i'+1)(d_i''+1)(d_i'''+1)\right)^{-2} \times \delta$.

Proof: For a fixed $D = (d_1', d_1'', d_1''', \ldots, d_k', d_k'', d_k''')$ we set the probability's bound from the Theorem 1 to be less than $\delta(D) = \delta_D$:

$$P_D(\epsilon(h) > \epsilon) \leq \left(\prod_{i\in\alpha} A_i\right)\left(\prod_{i\in\beta)} B_i\right)(1-\epsilon)^{m-2r} \leq \delta_D,$$

We consider realizations of D to be disjoint and equiprobable events. Hence for all possible realizations of D

$$P(\epsilon(h) > \epsilon) = \sum_D P_D$$

is fulfilled.

We use

$$\delta_D = (\frac{\pi^2}{6})^{-3(|\alpha|+|\beta|)}(\prod_{i\in\alpha\cup\beta}(d_i'+1)(d_i''+1)(d_i'''+1))^{-2} \times \delta$$

to satisfy $\sum P_D \leq \delta$ since $\sum_{i=1}^{\infty}(\frac{1}{i^2}) = \frac{\pi^2}{6}$.

To obtain the claimed bound substitute $(1-\epsilon)^{m-2r}$ by $e^{-\epsilon(m-2r)}$, and solve the inequality w.r.t. ϵ.

□

We extend the previous theorem to the case where misclassifications of training examples by a hypothesis are allowed.

Corollary 2 *Let h be the hypothesis defined above. Let $\sum_{i\in\alpha\cup\beta}(k_i'+k_i'') = q$ be the number of examples of the training set that are misclassified by h. Then with probability $1-\delta$ over random training sets of size m, the generalization error $\epsilon(h)$ of the hypothesis h is at most*

$$\frac{1}{m-2r-q}(\sum_{i\in\alpha} ln(A_i(K_i'+K_i'')) + \sum_{i\in\beta} ln(B_i(K_i'+K_i'')) + ln(\frac{1}{\delta_D'}))$$

where $\delta_D' = (\frac{\pi^2}{6})^{-5(|\alpha|+|\beta|)}(\prod_{i\in\alpha\cup\beta}(d_i'+1)(d_i''+1)(d_i'''+1)(k_i'+1)(k_i''+1))^{-2}\times\delta$.

Proof: To prove this result apply the proof of Theorem 2 for the probability's bound from Corollary 1.

□

The bounds of theorem 2 and corollary 2 show that the generalization error will be small if the size of the compression set is small. Evidently, the size of the compression set depends on the number of alternations and the number of generalized balls in the output hypothesis. Hence the generalization error will be small if the DLM outputs a hypothesis with a small number of alternations and a small number of generalized balls.

Chapter 6

Conclusions

This chapter summarizes the contributions of the thesis, and suggests some ideas for future research.

6.1 Summary

The goal of this thesis was to design and develop a practical learning algorithm for the decision list machine, and to assess both empirically and theoretically its learning ability. The work was done by

- designing the decision list learning algorithm with the use of generalized balls,
- programming this algorithm and testing it on different data sets and
- obtaining bounds on the generalization error of the learning algorithm.

Several key ideas that appear in various parts of the research were inspired by the Set Covering Machine (SCM): the feature space of generalized balls, applying early stopping and finite error tolerance on a testing set (so-called "penalty parameter").

The learning algorithm for the DLM was formulated in Chapter 3. The new notion of essential precedence for a decision list was defined. Rules of model selection for the DLM, such as early stopping and penalty parameters, were described as well.

The learning performance of the DLM on the artificial and some "natural" data sets was presented in Chapter 4. Empirical results are represented and compared with results of the SCM, the support vector machine, and nearest-neighbor classifier on the same data. The comparison indicates that **learners could benefit from the use of the DLM.**

We have also evaluated the performance of the DLM with asymmetric loss coefficients and found that the DLM was able to perform well for large asymmetry in the loss coefficients.

Finally, upper bounds for the generalization errors are established and proved for the consistent and the inconsistent DLM. Basically, these bounds show that we expect good generalization wherever a small DLM is found.

6.2 Future work

We propose the following directions of future research.

- We can use knowledge-basic metrics to calculate distances. How it will affect the performance of the DLM is an open problem.
- Another problem at hand is the design of features which are different from generalized balls.

Bibliography

[1] Cherkassky V., Mulier F.(1998) Learning from Data. New York: Wiley

[2] Chvatal V.(1979) A Greedy Heuristic for the Set Covering Problem. Mathematics of Operations Research, Vol.4, pp. 233-235

[3] Cormen T.H.,Leiserson C., Rivest R.(2000)Introduction to Algorithms. Cambridge: MIT Press

[4] Cristianini N., Shawe-Taylor J.(2000) An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press.

[5] Floyd S., Warmuth M.(1995)Sample compression, learnability, and the Vapnik-Chervonenkis dimension. Machine Learning, Vol.21, pp.269-304

[6] Haussler D.(1988) Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. Artificial Intelligence, Vol.36, pp.177-221

[7] Johnson D.S.(1974) Approximation Algorithms for Combinatorial Problems. Comput.System Sci. J., 9, p.p.256-298.

[8] Kearns M.J., Vazirani U.V.(1994) An Introduction to Computational Learning Theory. Cambridge: MIT Press.

[9] Langley P.(1996) Elements of Machine Learning. San Francisco: Morgan Kaufman Publ.

[10] Littlestone N., Warmuth M.(1986) Relating Data Compression and Learnability. Tech. report, University of California, Santa Cruz

[11] Lovasz L.(1975) On the Ration of Optimal Integral and Fractional Covers, Discrete Math, 13,p.p. 383-390

[12] Marchand M., Shawe-Taylor J.(2001) Learning with the Set Covering Machine, ICML'2001, pp.345-352. San Francisco: Morgan Kaufmann

[13] Mitchell T.,(1997) Machine Learning, McGraw Hill.

[14] Natarajan B.K. (1991) Machine Learning: a Theoretical Approach. San Mateo: Morgan Kaufman Publ.

[15] Schapire R.E. (1991) The Design and Analysis of Efficient Learning Algorithms. Cambridge:MIT Press.

[16] Reading in Machine Learning, edited by Shavlik J.W. and Dietterich T.G.(1990). San Mateo: Morgan Kaufman Publ.

[17] Valiant L.G.(1984) A Theory of the Learnable. Comm. ACM 27, pp.1134-1142

[18] Vapnik V.N.(1995) The Nature of Statistical Learning Theory.Springer Verlag

[19] Vapnik V.N.(1998) Statistical Learning Theory. New York: Wiley

[20] www.ics.uci.edu/~mlearn/MLRepository.html

Appendix A

The set-covering problem is an optimization problem that models many resource-selection problems (see [2], [3]). It is known to be NP-hard (for a brief and explicit definition see [3]).

An input $(\mathcal{X}, \mathcal{F})$ of the set-covering problem consists of a finite set $\mathcal{X}$ and a family $\mathcal{F}$ of subsets of $\mathcal{X}$, such that every element of $\mathcal{X}$ belongs to at least one subset in $\mathcal{F}$:

$\mathcal{X} = \bigcup_{S \in \mathcal{F}} S$.

We say that $\mathcal{F}$ covers $\mathcal{X}$. The problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ that cover all of $\mathcal{X}$:

$\mathcal{X} = \bigcup_{S \in \mathcal{C}} S$.

We say that any $\mathcal{C}$ satisfying the above equation covers $\mathcal{X}$.

Even though it may be difficult to find an optimal (minimal) set cover, it is not too hard to find a set cover that is near-optimal. An *approximation algorithm* for a problem A returns near-optimal solution of A. Now we present a *greedy* approximation algorithm known as the *greedy set covering* algorithm.

The greedy method works by picking, at each stage, the set S that covers the largest number of uncovered elements.

1. $U \leftarrow X$
2. $C \leftarrow \emptyset$
3. while $U \neq \emptyset$ do:
   (a) do select an $S \in F$ that maximizes $|S \cap U|$
   (b) $U \leftarrow U - S$
   (c) $C \leftarrow C \cup \{S\}$
4. return C

The algorithm works as follows. The set U contains, at each stage, the set of remaining uncovered elements. The set C contains the cover being constructed. Line 3.1 is the greedy decision-making step. A subset S is chosen that covers as many uncovered elements as possible. After S is selected, its elements are removed from U, and S is placed in C. When the algorithm terminates, the set C contains a subfamily of F that covers X.

The greedy algorithm returns a set cover that is not too much larger than an optimal set cover (see [2] for the proof). As the size of the input gets larger, the size of the approximation solution may grow, relative to the size of an optimal solution. If $|X| = m$ and $|S_{opt}|$ is the size of the optimal cover, then the greedy cover has a size of at most $|S_{opt}|ln(m) + 1$.

Lemma 1 *Let r be the size of the minimal cover that covers m examples. Then the set cover greedy algorithm outputs a cover of size at most $r \times ln(m) + 1$.*

Proof: Let S_l denote the set of examples not covered after l steps of the algorithm. Then $|S_{l+1}| \leq |S_l|(1 - \frac{1}{r})$ due to the fact that there is a set in the minimal cover that covers a $\frac{1}{r}$ fraction of examples on each step.
By induction on l: $|S_l| \leq (1 - \frac{1}{r})^l m$. If $l = r \times ln(m) + 1$ then all m examples are covered.
∇.

The greedy set cover algorithm can be implemented to run in time polynomial in $|\mathcal{X}|$ and $|\mathcal{F}|$. Since the number of iterations of the loop on lines 3.1-3.3 is at most $\min(|\mathcal{X}|, |\mathcal{F}|)$, and the loop body can be implemented to run in time $\mathcal{O}(|\mathcal{X}||\mathcal{F}|)$, there is an implementation that runs in time $\mathcal{O}(|\mathcal{X}||\mathcal{F}|min(|\mathcal{X}|, |\mathcal{F}|))$.

Appendix B

Coef		DLM		DLM_{tr}		DLM_i			
l_p	l_n	L	size	L	size	p_p	p_n	L	size
1	100	12716	139.1	3550	1	0.6	-	373	2
1	10	1376	139.1	580	1	1	-	283	2
1	5	746	139.1	415	1	1	-	278	2
1	2	368	139.1	314	77	1	-	230	14
2	1	358	139.1	332	18	2.5	-	277	1
5	1	706	139.1	384	3	3.9	-	371	1
10	1	1286	139.1	434	3	0.5	-	411	2
100	1	11726	139.1	1334	3	-	0.4	939	2

Table B.1: DLM results for the asymmetric loss on Credit (296 positive and 357 negative examples)

Coef		DLM		DLM_{tr}		DLM_i			
l_p	l_n	L	size	L	size	p_p	p_n	L	size
1	100	4581	43.8	499	2	0.2	-	497	2
1	10	531	43.8	229	2	0.3	-	225	2
1	5	306	43.8	214	2	0.4	-	191	2
1	2	171	43.8	155	4	-	3.5	111	1
2	1	207	43.8	138	10	-	2.9	75	6
5	1	450	43.8	249	10	-	0.4	79	2
10	1	855	43.8	433	10	-	0.4	84	2
100	1	8145	43.8	3764	10	-	0.4	174	2

Table B.2: DLM results for the asymmetric loss on Haberman (219 positive and 75 negative examples)