

1.1 Introduction

Many tasks from real application domain can be described as a process of “learning”. For example, the problem of recognizing if there is a human face in the given set of images can be described as the following:

- represent each image by a *feature vector* (depending on specific application), say (e_1, e_2, \dots, e_n) , and map it into the n -dimensional space \mathbb{R}^n ;
- label each node as *positive* if it contains a human face or *negative* otherwise;
- form the training set using the labelled nodes;
- apply any state-of-the-art machine learning algorithm on the training set to build a learning model;
- use the learning model to test any given new image.

In the above example, we are trying to learn something “new”. It remains unknown about the performance of the learning model if we can only empirically evaluate it. Specifically, we would like to ask the questions such as whether the learning algorithm will finally converge, how long does it take for its convergence, and how many instances in the training set are needed. In order to answer such questions, we need to build a mathematical framework and foundation for machine learning. Thus the study of *Computational Learning Theory* fills this crucial need.

In this lecture, we are going to introduce the *Mistake Bound* (MB) model, our first mathematical model in this course. Then we will discuss how to learn disjunctive concepts and decision lists under MB model.

Preliminary For most of this course, we will focus on the problem of *concept learning*. In the above example, we are given data such as a group of images with/without human faces in them and we want to learn from this data to classify future instances correctly. In the following, we will describe the definitions and notations related to *concept learning*, which will be used in this and the following lectures.

- The *instance* x is a single observation/measurement, typically represented by its values on some set of *attributes* or *variables* (We will use these two terms interchangeable in this class). Examples:

$$x = (0, 1, 1, 0, 1), x = (0.6, -3.4, 2.8, 7.2)$$

- The *instance space* X is a space from which all instances x are drawn, e.g., $x \in X$. Examples:

$$X = \{0, 1\}^n, X = \mathbb{R}^n$$

- A *labelled example* is an instance together with a labeling (e.g. *positive* or *negative*). Examples:

$$(0, 1, 1, 0, 1, +), (0.6, -3.4, 2.8, 7.2, -)$$

- A *concept* c is a boolean function over the instance space X . For instance, the concept $x_1 \wedge \bar{x}_3$ over the set $\{0, 1\}^n$ is the boolean function that outputs 1 on any n -dimensional instance from the set $\{0, 1\}^n$ whose first feature is set to 1 and third feature is set to 0.
- A *concept class* C is a set of *concepts* with the same representation. In this case, the *concept class* C is a subset of all boolean functions. For example, the class of *conjunctions* consists of all concepts that can be expressed as a conjunction of literals (variable and its negation).

1.2 Mistake Bound Model

In the mistake bound model, the learning process is performed in multiple trials. In each trial:

1. The learning algorithm is presented with an unlabeled instance x to update its current hypothesis $h(x)$,
2. the learner evaluates $h(x)$ to predict the label of x ,
3. the learner receives the correct label $f(x)$ from an oracle and increases its mistake bound by one if $h(x) \neq f(x)$.

The performance of a learning algorithms in mistake bound model is measured by the number of mistakes it can make. Note that the mistake bound model explicitly models the learning as an online process, i.e., the learning algorithm incrementally update its hypothesis every time when a new example is presented.

Definition 1 Let C be a concept class over the instance space X , we say algorithm A has mistake bound $M(c)$ for learning some concept $c \in C$ if A makes at most M mistakes on any sequence that is consistent with concept c . And we say algorithm A has mistake bound $M(C) = \max_{c \in C} M(c)$ over the concept class C , which is the worst number of mistakes made by A for all concept $c \in C$.

Let n be the number of examples and $size(c)$ be the description length of concept c under some encoding scheme (i.e., for any concept c in the class of conjunctions, $size(c)$ equals to the number of *literals* (variable or its negation) in c), we have the following definition of learnability in MB model:

Definition 2 We say a concept class C over instance space X is *efficiently learnable* in MB model if there exists an algorithm A for any concept $c \in C$, which has mistake bound $M(c) = poly(n, size(c))$ and its running time per trial is also $poly(n, size(c))$.

In fact, if we do not consider the efficiency, there exists an algorithm (non-efficient) for learning class C with mistake bound $O(\log(|C|))$.

Algorithm (Halving Algorithm):

step 1 enumerate all concept $c \in C$ in the hypothesis h ;

step 2 for each example, predict its label using majority vote over all concepts c in hypothesis h ;

step 3 if a mistake occurs, drop the concepts that are inconsistent with the example.

Analysis:

Since for each mistake the above algorithm cut down at least half number of concepts c , it makes at most $\log(|C|)$ mistakes.

1.2.1 Learning Disjunctive Concepts

The class of disjunctive concepts C over n Boolean variables is the set of all disjunctions with length $\leq n$, i.e., $\{x_1 \vee x_2, x_3 \vee \bar{x}_5 \vee x_7, \dots\}$. It is easy to show that $|C| = 3^n$, where $|C|$ is the number of concepts in class C .

Theorem 1 *The class of disjunction concepts is efficiently learnable under MB model.*

We can devise an algorithm which has mistake bound $O(n)$ for learning any concept in disjunction class C .

Algorithm:

step 1 The learning algorithm starts with hypothesis $h = x_1 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_2 \vee \dots \vee x_n \vee \bar{x}_n$;

step 2 for each mistake on a negative example, drop all unsatisfied literals from h ;

step 3 if h make a mistake on a positive example, output “no consistent concept”, otherwise output h as the final hypothesis.

Analysis:

Since initial hypothesis h contains $2n$ variables, the above algorithm will stop by making $2n$ mistakes maximally. (Actually the algorithm can only make $n+1$ mistakes maximally since it will drop exactly n variables in the first mistake and at least one variable on each subsequent mistake.) Therefore, the learning algorithm has mistake bound $O(n)$. Note that the Halving algorithm introduced in section 1.2 has mistake bound $O(\log(n))$, however enumerating all concepts has running time $O(3^n)$.

Proof: (Theorem 1) As illustrated in the above algorithm and its analysis, given n examples the algorithm makes $n + 1$ mistakes maximally in learning any disjunction concept. It shows that the class of disjunction concepts are efficiently learnable under MB model. ■

1.2.2 Learning Decision Lists

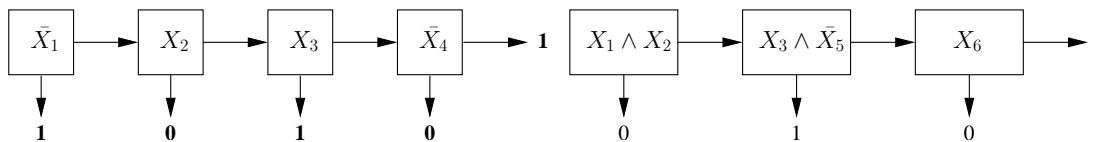
Definition 3 A decision list with length k is a list L of pairs

$$(t_1, v_1), (t_2, v_2), \dots, (t_i, v_i), \dots, (t_k, v_k)$$

where each t_i is a term (conjunction of literals) and each v_i is a Boolean value, and the last term t_k is the null conjunction (constant function **true**). Specifically, if any term in L has at most t literals, we denote it as t -decision list (t -DL).

We also call the pair (t_i, v_i) in L a basic block in the decision list. For a t -decision list over n variables, we have totally $(2n)^t \cdot 2$ basic blocks. Any basic block in the target concept is called true basic block for that concept.

A decision list defines a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as follows: for any $x \in \{0, 1\}^n$, $L(x) = v_i$ where i is the least index such that $t_i(x) = 1$. (Such term always exists, since the last term in L is the constant function **true**.)



(a) An example of 1-decision list

(b) An example of 2-decision list

Figure 1.1: Decision lists with different length of term.

Figure 1.1 gives the example of decision lists with different length of term. We can think of each component in the decision list as a block representing a *if-then-else* statement. From this point of view, the decision list L can be regarded as an extended *if-then-else* statement.

Theorem 2 Decision list are efficiently learnable under MB model.

Without loss of generality, let's consider the decision list in which each term contains at most one literal, e.g., 1-DL. Figure 1.1(a) gives an example of 1-DL. Obviously 1-DL over n variables has $4n$ basic blocks.

Sketch of Proof First we will present an algorithm for learning a decision list of length k over n variables. Then we show that the given algorithm has a mistake bound of $O(nk)$. ■

Algorithm:

- step 1:** Create k bags correspond to k levels of the decision lists. In our initial hypothesis, put all $4n$ basic blocks into the first bag and the rest are empty;
- step 2:** given an example x , start from the first bag, find a basic block such that its term is satisfied by x . If there are multiple satisfied blocks, we choose arbitrary one. If there is no block satisfied

by example x in current bag, we continue this procedure on the next bag until we find the satisfied block;

step 3: use the output of the basic block we found as the output of the decision list. If a mistake is made, demote the basic block into the next bag.

Note that, the above algorithm may not produce only one decision list but a set of decision lists which are all consistent with the given set of examples X since there could be multiple basic blocks in each level of the k bags.

Claim 1 *The true basic blocks in position i will never be demoted past position i .*

Proof: We prove the claim by induction on level k .

- Base: ($k = 1$) Since the algorithm only demote a block when it makes a mistake, the true block corresponding to the head of the target decision list will never be demoted.
- Induction: Assume that the claim is hold for all true blocks from 1 to $k - 1$. Consider a new example x that cannot be satisfied by any blocks in bags from 1 to $k - 1$, by inductive hypothesis, any true blocks in bags from 1 to $k - 1$ will not be demoted. Therefore to predict the label of x , we need to choose a block in $k + 1$ th bag. If the chosen block outputs the correct label, this block is a true block. We will not demote this block to the next bag.

From above we know that any true block in position i will never be demoted past position i . ■

Claim 2 *Every time a mistake is made, at least one block will be demoted.*

Proof: This claim is obvious in step 3 of the algorithm: the chosen block will be demoted when a mistake is made. ■

Proof: (Theorem 2) From Claim 1 and Claim 2, we know that in each level the algorithm makes at most n mistakes and there are totally k levels. Therefore the mistake bound of the algorithm is $O(nk)$, which shows that the decision lists are efficiently learnable. ■

Note that there are at most $(4n)^k$ 1-DLs of length k over n variables, the Halving algorithm will give a mistake bound $O(k \log(n))$.

Furthermore, for the general case of t -DL, we have:

Theorem 3 *Any t -decision list with arbitrary length can be learnt with mistake bound $n^{O(t)}$ under the MB model.*

Proof: Note that there are at most $(2n)^t$ conjunctions with length t over n variables. We can create a new variable y_i to replace each of such conjunctions. Therefore the total number of basic blocks is $(2n)^{(2t)}$. Also without loss of generality, we assume that no conjunction will be repeated in the

decision list, therefore the length of the decision list is at most $(2n)^t$. Now we can use the same algorithm to learn the decision list over variables y_i 's. The mistake bound of that algorithm would be $(2n)^{(2t)} \cdot (2n)^t = n^{O(t)}$. ■