6.858 Machine Learning	Lecture 2 : September 12, 1994
Lecturer: Ron Rivest	*Scribe: Deniz Yuret*

2.1 Outline

- On-line learning with mistake bounds
- Optimal and Halving algorithms
- Winnow1 algorithm for $\{x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}\}$

2.2 On-line learning with mistake bounds

2.2.1 Review of the learning model

$x^{(1)}, x^{(2)}, \ldots \to c_*$	The goal of concept learning is to discover an unknown target concept from labeled instances. On-line learning makes a prediction for each instance. In the mistake bound model the goal is to minimize the number of prediction mistakes.
$\mathcal{X} = \{0,1\}^n$	Instance space $\mathcal{X}$ consists of instances described by bit vectors of length n.
$c \in \mathcal{C}, c : \mathcal{X} \to \{0,1\}$	Class of concepts $\mathcal{C}$ consists of concepts which classify each instance as negative or positive.
$c_* \in \mathcal{C}$: $c_*(x) = 1 \Rightarrow$ *positive example* $c_*(x) = 0 \Rightarrow$ *negative example*	There is an unknown target concept the learner is trying to discover.
for $i = 1, 2, \ldots$ *learner given* $x^{(i)} \in \mathcal{X}$ *learner predicts* $c_*(x^{(i)})$ *learner is told* $c_*(x^{(i)})$	The learning process keeps predicting and learning in a loop.

2.2.2 Minimum number of mistakes

With the mistake bound approach, we define the optimal algorithm as one that has the best worst-case bound:

$$Opt(\mathcal{C}) = \min_{learners\ L} \max_{c_* \in \mathcal{C}} \max_{examples\ x^{(i)}} \begin{pmatrix} \text{number of mistakes} \\ \text{made by } L \text{ when} \\ \text{learning } c_* \text{ from ex-} \\ \text{amples } x^{(i)} \end{pmatrix}$$

Some obvious bounds we can show on $Opt(\mathcal{C})$ are:

$Opt(\mathcal{C}) \leq 2^n$	There are only 2^n possible instances.
$Opt(\mathcal{C}) \leq \lvert\mathcal{C}\rvert - 1$	Learner predicts according to the first remaining c.
$Opt(\mathcal{C}) \leq log(\lvert\mathcal{C}\rvert)$	We will achieve this bound by using the halving algorithm.

2.3 Optimal and Halving algorithms

2.3.1 Version space

Let $V_0 = \mathcal{C}$,
$V_i = \{c \in \mathcal{C} : c \text{ is consistent with the first } i \text{ labeled examples}\}$

V_i is called the i'th "version space" in AI terminology due to Mitchell. V_i represents the remaining possible concepts.

Let $\xi_0(\mathcal{C}, x) = \{c \in \mathcal{C} : c(x) = 0\}$
$\xi_1(\mathcal{C}, x) = \{c \in \mathcal{C} : c(x) = 1\}$

Then for $i > 1$, $V_i = \begin{cases} \xi_0(V_{i-1}, x^{(i)}) \text{ if } c_*(x^{(i)}) = 0 \\ \xi_1(V_{i-1}, x^{(i)}) \text{ if } c_*(x^{(i)}) = 1 \end{cases}$

2.3.2 Halving algorithm

The halving algorithm is an on-line learning algorithm which always predicts by the majority of concepts in the current version space.

On input $x^{(i)}$, predict 1 if $\lvert\xi_1(V_{i-1}, x^{(i)})\rvert \geq \lvert\xi_0(V_{i-1}, x^{(i)})\rvert$, 0 otherwise.

Theorem: $Opt(\mathcal{C}) \leq \lg \lvert\mathcal{C}\rvert$

Proof: At each step the halving algorithm predicts according to the majority of concepts that are still in the version space. Thus when a mistake is made, more than half of the candidate concepts is eliminated. ■

Note that the halving algorithm may not be efficient, since determining and representing V_i's and their sizes may be hard.

2.3.3 Optimal algorithm

On input $x^{(i)}$ predict 1 if $Opt(\xi_1(V_{i-1}, x^{(i)})) \geq Opt(\xi_0(V_{i-1}, x^{(i)}))$, 0 otherwise.

The halving algorithm is not optimal. Instead of guessing in accordance with the majority of the valid concepts, we should guess according to the concept group that gives us the least number of expected mistakes. However this optimal algorithm is computationally even less efficient.

There are cases where the bound $\lg \lvert\mathcal{C}\rvert$ is not tight. Take the example instance space of a large number of integers. The target concept is being one particular integer. By always guessing false, the optimal algorithm can only make one possible mistake.

2.3.4 Randomized halving algorithm

Predict according to $c^{(i)} \in V_i$ selected at random.

On any step, the halving algorithm may not predict according to any concept. The prediction is based on a majority vote, and there may not be a single concept consistent with the majority votes. The randomized halving algorithm (due to Maass [2]), on the other hand, bases its predictions on a randomly selected $c^{(i)} \in V_i$.

Theorem: The expected number of mistakes randomized halving algorithm makes $\leq ln\lvert\mathcal{C}\rvert + O(1)$.

Proof: Assume that the sequence of examples are chosen independent of the learner's behavior. Thus $x^{(1)}, x^{(2)}, \ldots$ can be seen as a fixed sequence of examples. At step i,

V_i is the set of remaining concepts. These concepts can be ordered by when they are going to be eliminated by the examples. Let $c^{(1)}, c^{(2)}, \ldots, c^{(r)}$ be this order, where $r = |V_i|$. Let M_r be the expected number of mistakes to be made. To derive a recurrence, note that the algorithm is equally likely to pick one of $c^{(1)}, c^{(2)}, \ldots, c^{(r)}$ as the concept which it will predict according to at this step. With probability $1/r$, it will pick $c^{(r)}$, and make no more mistakes. With probability $(r-1)/r$ it will pick one of $c^{(i)}$, where $i < r$. It will then make at least one mistake, plus the expected number of mistakes for the remaining $r - i$ concepts, i.e. M_{r-i}.

$$\begin{aligned}
M_1 &= 0 \\
M_r &= \frac{r-1}{r}\left[1 + \frac{M_1 + M_2 + \cdots + M_{r-1}}{r-1}\right] \\
&= \frac{r-1}{r} + \frac{M_1 + M_2 + \cdots + M_{r-1}}{r} \\
rM_r &= (r-1) + M_1 + M_2 + \cdots + M_{r-1} \\
(r-1)M_{r-1} &= (r-2) + M_1 + M_2 + \cdots + M_{r-2} \\
r(M_r - M_{r-1}) + M_{r-1} &= 1 + M_{r-1} \\
M_r &= M_{r-1} + 1/r \\
M_r &= \sum_{i=2}^{r} 1/i = ln(r) + O(1)
\end{aligned}$$

2.4 Winnow algorithm

So far, we have seen algorithms for general concept classes. In this section we will consider a restricted concept class: monotone disjunctions of at most k variables. We will analyze the Winnow algorithm (due to Littlestone [1]) which makes at most $O(k \lg n)$ mistakes. This bound is particularly useful when n is large and k is restricted (i.e., when there are many irrelevant attributes).

$$\begin{aligned}
\mathcal{C} &= \{x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}\} \\
|\mathcal{C}| &= \binom{n}{k} + \binom{n}{k-1} + \cdots + \binom{n}{0} \\
\lg|\mathcal{C}| &= \Theta(k \lg n)
\end{aligned}$$

We know that this is the bound for the optimal algorithm. However this does not tell us how to implement it efficiently.

We could try to use the on-line conjunction learning algorithm from the last lecture, based on the following duality principle:

$$c(x) = x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k} \Rightarrow \overline{c}(x) = \overline{x}_{i_1}\overline{x}_{i_2}\ldots\overline{x}_{i_k}$$

However, this algorithm will give us the bound of $n + 1 - k$, which is linear in the total number of attributes. It would not be useful when there is a large number of irrelevant attributes compared to relevant ones.

Instead, we will learn disjunctions by an algorithm for learning linearly separable Boolean functions. A concept c is *linearly separable* if $\exists w \in \Re^n, \Theta \in \Re$ such that

$$\forall x \; c(x) = 1 \Leftrightarrow w \cdot x \geq \Theta$$

In other words, a function from $\{0,1\}^n$ to $\{0,1\}$ is linearly separable if there is a hyperplane in $\Re^n$ separating the points on which the function is 1 from the points on which it is 0. Monotone disjunctions are linearly separable. That is, for any concept $x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$, the separating hyperplane is given by $x_{i_1} + x_{i_2} + \cdots + x_{i_k} \geq 1/2$.

We now give the Winnow1 algorithm for learning monotone disjunctions. We present it as a linear-threshold algorithm, where the algorithm learns appropriate weights w.

Initialize $\Theta = n/2$, $w = (1, 1, \ldots, 1)$
Predict $c(x) = 1$ iff $w \cdot x > \Theta$
If prediction OK, do nothing
If $c(x) = 1$ but $w \cdot x \leq \Theta$, double all the weights w_i where $x_i = 1$ (promotion)
If $c(x) = 0$ but $w \cdot x > \Theta$, zero all the weights w_i where $x_i = 1$ (elimination)

Theorem: Winnow1 makes $O(k \lg n)$ mistakes.

Proof: Let u be the number of false negatives (promotions), and v the number of false positives (eliminations). The total number of mistakes is $u + v$.

Lemma 1: $v \leq n/\Theta + u$
In the case of a promotion, the increase in the weights is at most Θ. The algorithm doubles a subset of the weights which did not add up to Θ. Conversely, in the case of an elimination, the decrease in the weights is at least Θ. The algorithm zeroes a subset of the weights that added up to more than Θ. The sum of the weights can never go below zero, since they are either doubled or zeroed, and they start positive. Thus $0 \leq \sum w_i \leq n + u.\Theta - v.\Theta \Rightarrow v \leq n/\Theta + u$.

Lemma 2: $w_i \leq 2\Theta$
A promotion occurs only if the sum of the relevant weights is less than Θ. Thus a single weight which is above Θ can never be doubled.

Lemma 3: After u promotions, $\exists w_i$ such that $\lg w_i \geq u/k$.
Each promotion must double one of $w_{i_1}, w_{i_2}, \ldots, w_{i_k}$. Each doubling adds 1 to the logarithm.

Claim: $u + v \leq 2k(\lg \Theta + 1) + n/\Theta$

$$\begin{array}{rcll} u/k & \leq & \lg w_i \leq \lg \Theta + 1 & \text{by Lemmas 2 and 3} \\ u & \leq & k(\lg \Theta + 1) & \\ v & \leq & n/\Theta + k(\lg \Theta + 1) & \text{by Lemma 1} \\ \Rightarrow u + v & \leq & n/\Theta + 2k(\lg \Theta + 1) & \end{array}$$

With $\Theta = n/2$, number of mistakes $\leq 2 + 2k \lg n$. ■

References

[1] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285-318, 1988.

[2] Wolfgang Maass. On-line learning with an oblivious environment and the power of randomization. *Proceedings of the Fourth Annual ACM Workshop on Computational Learning Theory*, pp. 167-175, August 1991.