Online Learning

with emphasis on the Winnow Algorithm

Online Learning Overview

Given:

- A set of functions
- Assurances that the target concept lies within this set
- Individual adversary-specified queries followed by feedback
- No training data!

Find:

- The target concept

Criteria:

- Competitive ratio not applicable in general
- Minimize worst-case errors

Online Learning Overview

Additional Constraints:

- Finite Boolean inputs
- Single Boolean output
- Consistent data

Applications:

- Attribute-Rich Environments
- Concept Construction

The Trial

Instances and instance space:

Each *instance* is an *n*-bit Boolean vector in $\{0,1\}^n$, the *instance space*. There is exactly one instance per trial.

Predictions:

The learner must immediately respond with a *prediction* of either 0 or 1.

Reinforcement:

The learner is told immediately, with 100% accuracy, whether the prediction was right or wrong.

The trial may or may not be followed with another; there is no guarantee as to how many there will be.

Mistake Bounds

$M_A(f)$, *f* a function:

The worst-case number of errors learning algorithm *A* makes when learning target function *f*.

$M_A(C)$, *C* a function class:

$\max_{f \in C} M_A(f)$ if *C* is non-empty, -1 otherwise

Mistake bound:

X is a *mistake bound* for learning algorithm A and class *C* if it is an uppper bound for $M_A(C)$

Optimal mistake bound:

$opt(C) = \min_A M_A(C)$, regardless of space/time efficiency

The Obvious

Given target class C and a point x in instance space X, define $\xi_0(C,x)$ to be the subset of C that predicts 0 for x, and likewise for $\xi_1(C,x)$.

Then the Halving Algorithm is as follows.

- Initialize $CONSIST \Leftarrow C$.
- Predicts x as 1 if and only if $|\xi_0(C,x)| < |\xi_1(C,x)|$.
- If the correct response is 0, $CONSIST \Leftarrow \xi_0(C,x)$, else $CONSIST \Leftarrow |\xi_1(C,x)|$.

It's good.

$M_{HALVING}(C) \leq \log_2(C)$ for non-empty C.

Proof:

- There are only two possible answers, 0 and 1.
- *HALVING* therefore chooses an response agreed to by at least half the remaining consistent functions.
- If *HALVING* is wrong, *CONSIST* is replaced by a set that is no larger than half its prior size, and still containing the correct concept.

It's not *that* good.

Define functions f_1 through f_9 on instances a_1 through a_8 as follows.

- f_x, for $x=1..6$, is 1 for a_x and 0 elsewhere.
- f_7 is 1 only for a_6 and a_8.
- f_8 is 1 only for a_6 and a_7.
- f_9 is 1 only for a_6, a_7 and a_8.

Then, using $C=\{f_1..f_9\}$, and a_6, a_7 and a_8 as the first three instances, concept f_9 will result in 3 mistakes. There exists an algorithm such that no sequence of these instances paired with any target function in this C will make more than 2 mistakes.

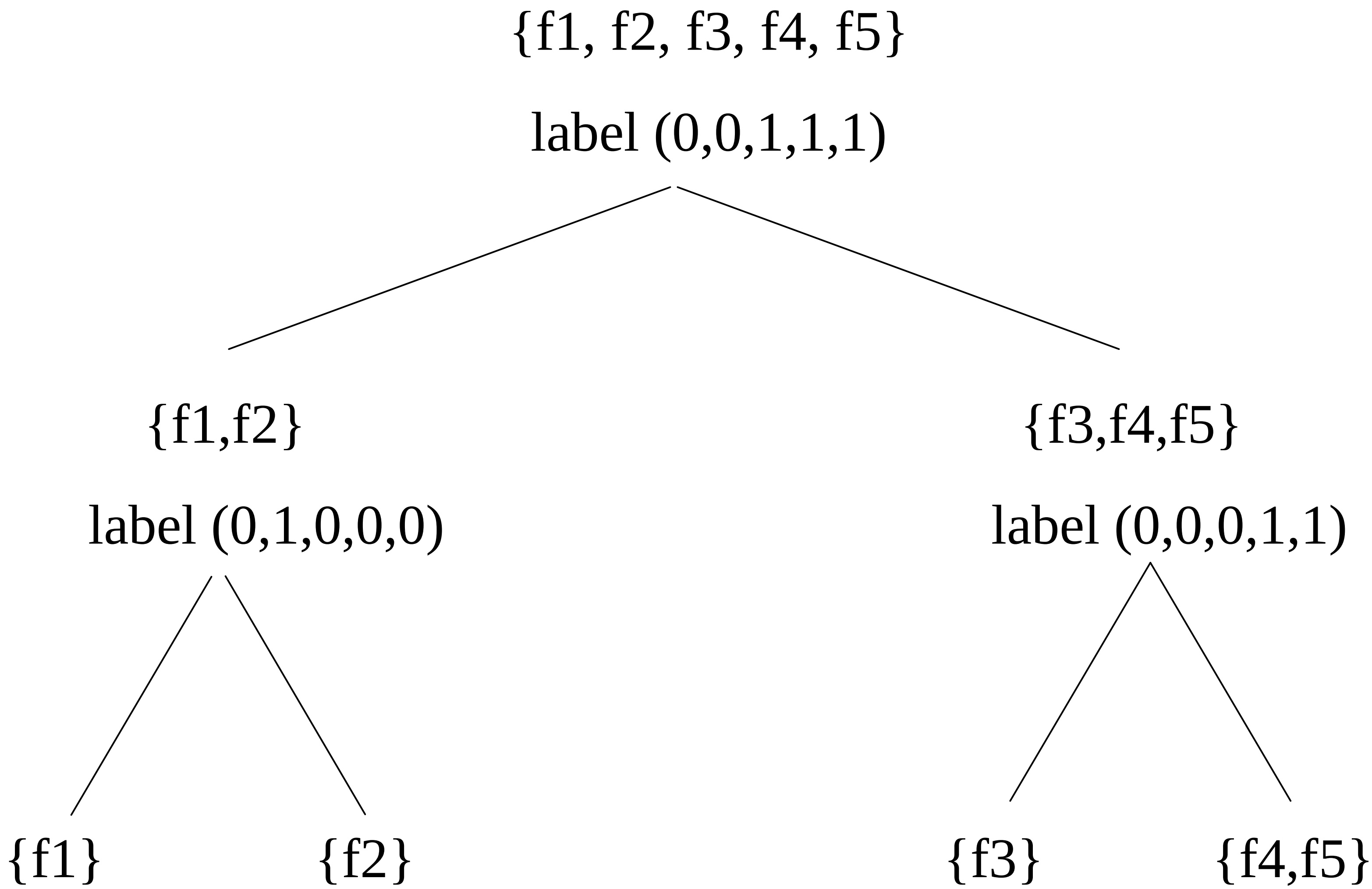
Concept: Mistake trees.

A *mistake tree* for a target class C over an instance space X is a binary tree with the following properties.

- Every node corresponds to a non-empty subset of C. The root corresponds to C itself.
- Every internal node is labeled with an example within X.
- If internal node corresponding to C' is labeled with instance x, the left child if present corresponds to $\xi_0(C',x)$ and the right child to $\xi_1(C',x)$.

A *complete k-mistake tree* is a mistake tree that is also a complete binary tree of height k, with the height equal to the maximum length in edges of any path from the root.

Sample complete 2-mistake tree



$K(C)$

For non-empty C, define $K(C)$ to be the largest integer k such that there exists a complete k-mistake tree for C.

$K(\emptyset) = -1$.

Intuitively, we should be able to make at most $K(C)$ mistakes when predicting the values of any concept in C, since we should be able to identify the concept within that many predictions.

How?

Standard Optimal Algorithm (SOA)

- Initialize *CONSIST* $\Leftarrow$ *C*.
- For instance *x*, predict 1 if and only if

  $K(\xi_1(CONSIST, x)) > K(\xi_0(CONSIST, x))$
- Then replace *CONSIST* with $\xi_1(CONSIST, x)$ or $\xi_0(CONSIST, x)$ according to the feedback.

Any mistakes therefore result in *CONSIST* being replaced with the subset with the *smaller* maximal complete mistake tree.

SOA Example

Recall the previous example with nine functions, eight variables, and a_6, a_7, and a_8 as the examples. The first five functions all predict *0* for a_6, which would make *HALVING* predict *0*. However, SOA predicts *1*, as it is easier to distinguish between these five versus the latter four; if the feedback indicates that the target IS one of these functions, it then predicts *0* until its second mistake, which will identify the target concept.

If the prediction of *1* is correct, clearly SOA can identify the correct function out of four with at most two mistakes.

Proof that, for any C, $opt(C) \geq K(C)$

Let $k=K(C)$; $k < 1$ is trivial. Otherwise, the adversary tracks the current mistake tree. Initially, this tree is a complete k-mistake tree for C.

The adversary chooses as instance the label of the root of the current mistake tree, and regardless of prediction informs the algorithm that it is incorrect.

The remaining consistent functions consist of either $\xi_0(C, x)$ or $\xi_1(C, x)$ -- and the corresponding $(k-1)$-mistake tree is one of the two children of the root of the original k-mistake tree.

Repeat for a total of k instances -- all mistakes -- after which the remaining concepts are indistinguishable.

And in the other direction...

With C be finite and non-empty, let the examples be x_1 through x_t. Using SOA, let $CONSIST_i$ be the value of $CONSIST$ before trial i. The following proves that, for any $k \geq 0$, i in $\{1..t\}$, $K(CONSIST_i)=k$ implies SOA will make at most k mistakes during trials $i..t$.

$k=0$ serves as a base case. By definition of K this implies that all remaining consistent functions are completely indistinguishable. As the target concept is in C, no further mistakes can be made. As $K(\emptyset)=-1$, SOA will use the correct prediction in this case.

Inductively, assume this holds for up to k-1. On the first mistake (if any) it makes in trials $i..t$, it replaces $CONSIST$ with a set C' such that $K(C') \leq k$-1 by the definition of SOA and K. Then invoke the induction hypothesis.

Vapnik-Chervonenkis Dimension

Target class C *shatters* $S \subseteq X$ if for every $U \subseteq S$ there exists a function $f \in C$ with $f(x)=1$ if and only if $x \in U$.

The *Vapnik-Chervonenkis dimension* of a non-empty target class C, $VCdim(C)$, is the cardinality of the largest set shattered by C. $VCdim(\emptyset)=-1$.

$opt(C) \geq VCdim(C)$ because for $k=VCdim(C)$ and a matching subset of X $\{x_1..x_k\}$, a complete k-mistake tree can be constructed with nodes at depth $j=0,1,\ldots,j-1$ labelled with x_{j+1}. Function subsets are guaranteed to be non-empty due to definition of shattering.

SOA and *HALVING* Summary

- For finite C, SOA is optimal, and $M_A(C)=opt(C)=K(C)$.
- $opt(C) \geq VCdim(C)$
- SOA predicts according to the heights of the complete k-mistake trees, favoring the deeper tree.
- *HALVING* is non-optimal, as it uses cardinalities (majority rule) instead.
- Both require tracking all consistent functions.

Reality check

SOA is too demanding. With 64 Boolean variables, there are $2^{2^{64}}$ possible Boolean functions. Without adding initial constraints on *C*, both memory and time requirements are extreme.

If we constrain *C* to a simpler class, such as *linearly separable* Boolean functions, or an even smaller class like *monotone disjunctions* -- disjunctions without using negation -- simpler algorithms can do quite well.

One of these algorithms is Winnow.