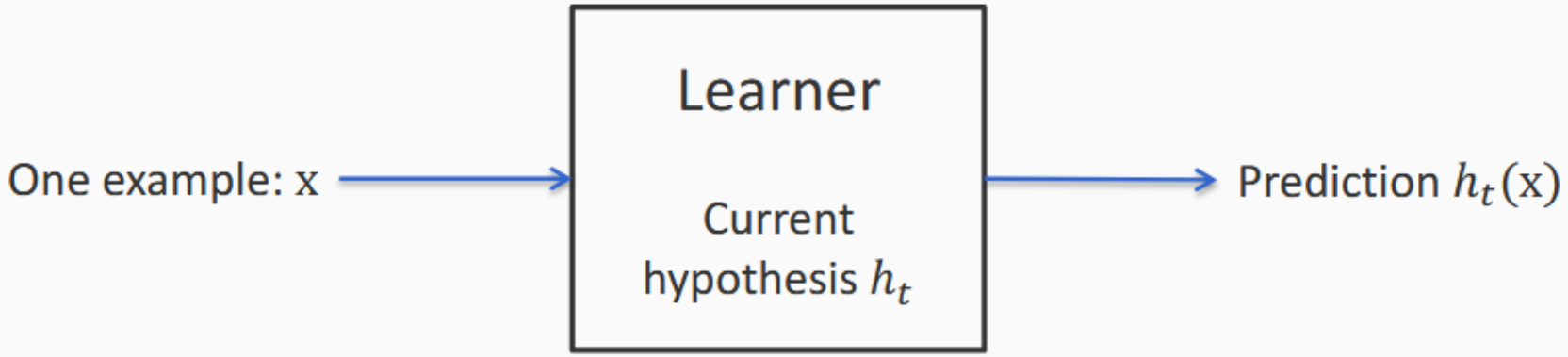
Online Learning

PAC Learning Setup

- PAC learning
  - Learner (A) receives a batch of examples S
  - Outputs a hypothesis $h \in H$
  - Sample complexity $m_H(A)$ determines the sample size of S required to achieve an error of (ε, δ)
  - We bound sample complexity in terms of VC-dimension

  - Target function may be realizable or not realizable

Online Learning



Loop forever:

1. Receive example x
2. Make a prediction using the current hypothesis $h_t(\mathrm{x})$
3. Receive the true label for x.
4. If $h_t(\mathrm{x})$ is not correct, then:
   - Update h_t to h_{t+1}

Mistake Bound Algorithns

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}, f \in C$ the concept class (parameterized by n)

Mistake Bound Algorithms

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}, f \in C$ the concept class (parameterized by n)

- Learning Protocol:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*

Mistake Bound Algorithms

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}, f \in C$ the concept class (parameterized by n)

- Learning Protocol:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*

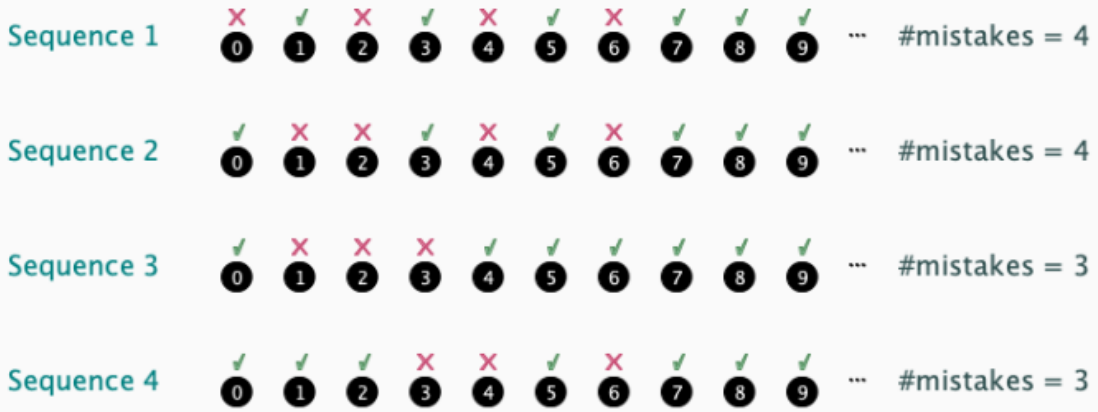
- Performance: learner makes a mistake when $h(\mathbf{x}) \neq f(x)$
  - $M_A(f, S)$: Number of mistakes algorithm A makes on sequence S of examples for the target function f

Mistake Bound Algorithms

- **Setting**:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}$, $f \in C$ the concept class (parameterized by n)
- **Learning Protocol**:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*
- **Performance**: learner makes a mistake when $h(\mathbf{x}) \neq f(x)$
  - $M_A(f, S)$: Number of mistakes algorithm A makes on sequence S of examples for the target function f

Mistake Bound Algorithms

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}$, $f \in C$ the concept class (parameterized by n)
- Learning Protocol:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*
- Performance: learner makes a mistake when $h(\mathbf{x}) \neq f(x)$
  - $M_A(f, S)$: Number of mistakes algorithm A makes on sequence S of examples for the target function f



Mistake Bound Algorithms

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}, f \in C$ the concept class (parameterized by n)
- Learning Protocol:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*
- Performance: learner makes a mistake when $h(\mathbf{x}) \neq f(x)$
  - $M_A(f, S)$: Number of mistakes algorithm A makes on sequence S of examples for the target function f
  - $M_A(C) = \max_{f \in S} M_A(f, S)$: The maximum possible number of mistakes made by A for any target function in C and any sequence S of examples

Mistake Bound Models

- Setting:
  - Instance space: $\mathcal{X}$ (dimensionality n)
  - Target f: $\mathcal{X} \rightarrow \{0,1\}, f \in C$ the concept class (parameterized by n)
- Learning Protocol:
  - Learner is given $\mathbf{x} \in \mathcal{X}$, randomly chosen
  - Learner predicts $h(\mathbf{x})$ and is then given $f(\mathbf{x})$ ⟵ *the feedback*
- Performance: learner makes a mistake when $h(\mathbf{x}) \neq f(x)$
  - $M_A(f, S)$: Number of mistakes algorithm A makes on sequence S of examples for the target function f
  - $M_A(C) = \max_{f \in S} M_A(f, S)$: The maximum possible number of mistakes made by A for any target function in C and any sequence S of examples
- Algorithm A is a ***mistake bound algorithm*** for the concept class C if $M_A(C)$ is a polynomial in the dimensionality n

Learnability: Mistake Bound Models

- Algorithm A is a ***mistake bound algorithm*** for the concept class C if $M_A(C)$ is a polynomial in the dimensionality n
  - That is, the maximum number of mistakes it makes for any sequence of inputs (perhaps even an adversarially chosen one) is polynomial in the dimensionality

Learnability of Mistake Bound Models

- Algorithm A is a ***mistake bound algorithm*** for the concept class C if $M_A(C)$ is a polynomial in the dimensionality n
  - That is, the maximum number of mistakes it makes for any sequence of inputs (perhaps even an adversarially chosen one) is polynomial in the dimensionality

- A concept class is *learnable* in the *mistake bound model* if there ***exists*** *an algorithm* that makes a polynomial number of mistakes for any sequence of examples
  - Polynomial in the dimensionality of the examples

Online Learning and Mistake Bound

- Not the most general setting for online learning
- Not the most general metric
- Other metrics: Regret, cumulative loss

Online Learning

- No assumptions about the distribution of examples
- Examples are presented to the learning algorithm in a sequence. *Could be adversarial!*

  For each example:

  1. Learner gets an unlabeled example
  2. Learner makes a prediction
  3. Then, the true label is revealed

- In the mistake bound model, we only count the number of mistakes

Mistake Bound Model of Online Learning

DEFINITION 21.1 (Mistake Bounds, Online Learnability) Let $\mathcal{H}$ be a hypothesis class and let A be an online learning algorithm. Given any sequence $S = (x_1, h^\star(y_1)), \ldots, (x_T, h^\star(y_T))$, where T is any integer and $h^\star \in \mathcal{H}$, let $M_A(S)$ be the number of mistakes A makes on the sequence S. We denote by $M_A(\mathcal{H})$ the supremum of $M_A(S)$ over all sequences of the above form. A bound of the form $M_A(\mathcal{H}) \leq B < \infty$ is called a *mistake bound*. We say that a hypothesis class $\mathcal{H}$ is online learnable if there exists an algorithm A for which $M_A(\mathcal{H}) \leq B < \infty$.

Online Learning

- Simple and intuitive model, widely applicable
- Important in the case of very large data sets, when the data cannot fit memory (streaming data)
- Evaluation: We will try to make the smallest number of mistakes in the long run.
  - Some things to think about:
    - What is the relation to the "real" goal? What is the real goal of learning?
    - Does online learning generate a hypothesis that does well on previously unseen data?

Online/Mistake Bound Learning

- No notion of data distribution; a worst case model
- No (or not much) memory: get example → update hypothesis → get rid of it
- Drawbacks:
  - Too simple
  - Global behavior: not clear when will the mistakes be made
- Advantages:
  - Simple
  - Many issues arise already in this setting
  - Generic conversion to other learning models (online-to-batch conversion)

Is counting mistakes enough?

- Under the mistake bound model, we are not concerned about the number of examples needed to learn a function
- We only care about not making mistakes
- Eg: Suppose the learner is presented the *same example* over and over
  - *Under the mistake bound model, it is okay*
  - *We won't be able to learn the function, but we won't make any mistakes either!*

CONSISTENT Online Learner

- Equivalent to ERM

CONSISTENT Online Learner

- Let C be a finite concept class
- Goal: Learn $f \in C$

- Algorithm CON (short for consistent):
  In the i^{th} stage of the algorithm:
  - C_i = all concepts in C consistent with all i – 1 previously seen examples
  - Choose randomly $f \in C_i$ and use it to predict the next example

- It is not hard to show that $C_{i+1} \subseteq C_i$

- If a mistake is made on the i^{th} example, then $|C_{i+1}| < |C_i|$
  progress is made

Version Space

- Subset of the hypothesis space consisting of hypothesis that are consistent with a set of training examples
- Realizable target concept will have a non-null version space
- CON algorithm randomly picks an element of the version space

Mistake Bound of the CON Learner

- The CON algorithm makes at most $|H|$ -1 mistakes

Obviously, whenever `Consistent` makes a prediction mistake, at least one hypothesis is removed from V_t. Therefore, after making M mistakes we have $|V_t| \leq |\mathcal{H}| - M$. Since V_t is always nonempty (by the realizability assumption it contains $h^\star$) we have $1 \leq |V_t| \leq |\mathcal{H}| - M$. Rearranging, we obtain the following:

COROLLARY 21.2 *Let $\mathcal{H}$ be a finite hypothesis class. The `Consistent` algorithm enjoys the mistake bound $M_{\texttt{Consistent}}(\mathcal{H}) \leq |\mathcal{H}| - 1$.*

HALVING Algorithm

- Let C be a finite concept class
- Goal: Learn $f \in C$

- Initialize $C_0 = C$, the set of all possible functions

We will construct a series of sets of functions C_i

- Learning ends when there is only one element in C_i

HALVING Algorithm

- Let C be a finite concept class
- Goal: Learn $f \in C$

- Initialize C_0 = C, the set of all possible functions
- When an example **x** arrives:
  - Predict the label for **x** as 1 if a majority of the functions in C_i predict 1. Otherwise 0. That is, output = 1 if

$$|\{h(\mathbf{x}) = 1 : h \in C_i\}| > |\{h(\mathbf{x}) = 0 : h \in C_i\}|$$

  - If prediction ≠ *f(**x**)*: (i.e error)
    - Update C_{i+1} = all elements of C_i that agree with f(**x**)
- Learning ends when there is only one element in C_i

HALVING Algorithm: Mistake Bound

THEOREM 21.3 *Let $\mathcal{H}$ be a finite hypothesis class. The* **Halving** *algorithm enjoys the mistake bound* $M_{\textbf{Halving}}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$.

Proof We simply note that whenever the algorithm errs we have $|V_{t+1}| \leq |V_t|/2$, (hence the name Halving). Therefore, if M is the total number of mistakes, we have

$$1 \leq |V_{T+1}| \leq |\mathcal{H}|\, 2^{-M}.$$

Rearranging this inequality we conclude our proof. □

HALVING Algorithm

- A simple algorithm for *finite* concept spaces
  - Stores a set of hypotheses that it iteratively refines
    - Receive an input
    - Prediction: the label of the majority of hypotheses still under consideration
    - Update: If incorrect, remove all inconsistent hypotheses

- Not always optimal because we care about minimizing the number of mistakes in the future!
  - What if, instead of eliminating functions that disagree with this example, we down-weight them
  - Perhaps via multiplicative or additive updates…

Example

Hidden function: conjunctions

- The learner is to learn functions like $f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

- Number of conjunctions with n variables = $|C| = 3^n$
  - $\log|C| = O(n)$

Learning Conjunctions: Elimination

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Teacher (Nature) provides the labels (f(x))

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,...0,1,1), 1>
- <(1,0,1,1,1,0,...0,1,1), 0>
- <(1,1,1,1,1,0,...0,0,1), 1>
- <(1,0,1,0,0,0,...0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,...0,1,1), 0>

Notation: <example, label>

Learning Conjunctions

Teacher (Nature) provides the labels (f(x))

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,…0,1,1), 1>
- <(1,0,1,1,1,0,…0,1,1), 0>
- <(1,1,1,1,1,0,…0,0,1), 1>
- <(1,0,1,0,0,0,…0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,…0,1,1), 0>

Look for the variables that are present in *every* positive example.

All other variables can be eliminated

Why?

Learning Conjunctions: Elimination

Teacher (Nature) provides the labels (f(x))

- <(1,1,1,1,1,1,…,1,1), 1>
- <(1,1,1,0,0,0,…,0,0), 0>
- <(1,1,1,1,1,0,…0,1,1), 1>
- <(1,0,1,1,1,0,…0,1,1), 0>
- <(1,1,1,1,1,0,…0,0,1), 1>
- <(1,0,1,0,0,0,…0,1,1), 0>
- <(1,1,1,1,1,1,…,0,1), 1>
- <(0,1,0,1,0,0,…0,1,1), 0>

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Whenever the output is 1, x_1 is present

Conservative Learner

Definition 2. *We say an online learning algorithm is* ***conservative*** *if it only updates its current hypothesis when making a mistake.*

Conservative Learner

Lemma 1. *Suppose that $\mathcal{A}$ learns a hypothesis class C with mistake bound M. Then there exists a conservative algorithm $\mathcal{A}'$ that also learns C with the same mistake bound, M.*

Proof. The algorithm $\mathcal{A}'$ will be the same as $\mathcal{A}$ except for when the latter recieves an example it correctly labels. In this case, since $\mathcal{A}$ is not necessarily conservative, it may update its current hypothesis. $\mathcal{A}'$ simply "undoes" this update (or rather, acts as if this example had not arrived), and maintains the previous current hypothesis. Thus on a given sequence of elements, $x_1, ..., x_t, ...$ suppose that the updates of $\mathcal{A}'$ are given by the indices $i1, ..., ik$, then $\mathcal{A}'$ performs as if $\mathcal{A}$ had seen the sequence $x_{i1}, ..., x_{ik}$, which are indeed all mistakes, but there are at most M of these as assumed. □

Relation between PAC and Online Learning

Theorem 1. *Suppose that a concept class C is learnable in the mistake-bound model. Then C is also PAC-learnable.*

Proof. We assume that C is learned in the mistake-bound model by the conservative algorithm $\mathcal{A}$ that makes at most M mistakes. Furthermore, since we are in the PAC-learning paradigm, we have access to an example oracle $EX(c, D)$. We simply sample from the oracle in an online manner, and if ever the current hypothesis, say h, of $\mathcal{A}$ survives for more than $m = \frac{1}{\varepsilon} \log\left(\frac{M}{\delta}\right)$ samples, then return h.

The algorithm makes a mistake if it returns a hypothesis, h such that $err(h) > \varepsilon$ For a given sequence of m samples this happens with probability at most $(1-\varepsilon)^m < \frac{\delta}{M}$. Since the algorithm is conservative, the working hypothesis of $\mathcal{A}$ can change at most M times, therefore the total probability of error is bounded by $M(1-\varepsilon)^m < M\frac{\delta}{M} = \delta$. Furthermore, the total number of samples needed is at most $\frac{M}{\varepsilon} \log\left(\frac{M}{\delta}\right)$ □

Lower Bounds

Theorem 2. *Suppose that C is a concepet class with $VC(C) = d$. Then for any deterministic online learning algorithm there is a sequence of inputs for which the algorithm makes at least d mistakes.*

Proof. Since $VC(C) = d$, let $S = x_1, , , .x_d$ be a shatterable set under C. This means an adversary can label these points in any dichotomy that is still consistent with some $c \in C$. Therefore no matter what guess an algorithm gives on those points, an adversary can still force an error from a valid hypothesis. □

Online Learning as a Game

Idea: view online learning as a 2-player game between learner and the environment.

At time t :

$\rightarrow$ environment picks x_t

$\rightarrow$ learner picks p_t

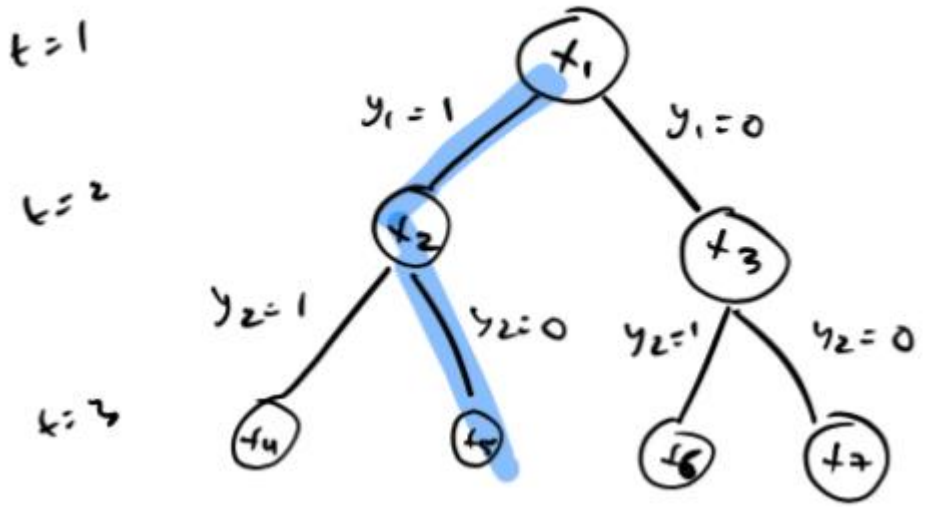
$\rightarrow$ environment picks $y_t = 1 - p_t$

Online Learning as a Game

Q) How to choose x_t to get the learner to make maximum number of mistakes, while ensuring realizability?

Strategy for the environment can be formally described as a binary tree. Each node of the tree is associated with an instance from X. If the learner predicts $p_t = 1$ the environment will declare that this is a wrong prediction (i.e., $y_t = 0$) and will traverse to the right child of the current node. If the learner predicts pt $= 0$ then the environment will set $y_t = 1$ and will traverse to the left child. This process will continue and at each round.

Online Learning as Game



Shattered Game Tree

Definition 4. *$\mathcal{H}$ shattered tree: A shattered tree of depth d is a sequence of instances $x_1, x_2, \ldots, x_{2^d-1} \in X$ such that for every path from the root to a leaf, $\exists\ h \in \mathcal{H}$ which realizes all the labels along this path.*

Littlestone Dimension

Definition 5. *Littestone dimension: Littlestone dimension of hypothesis class* $\mathcal{H}(Ldim(\mathcal{H}))$ *is the maximal* T *such that* $\exists$ *a tree of depth* T *shattered by* $\mathcal{H}$.

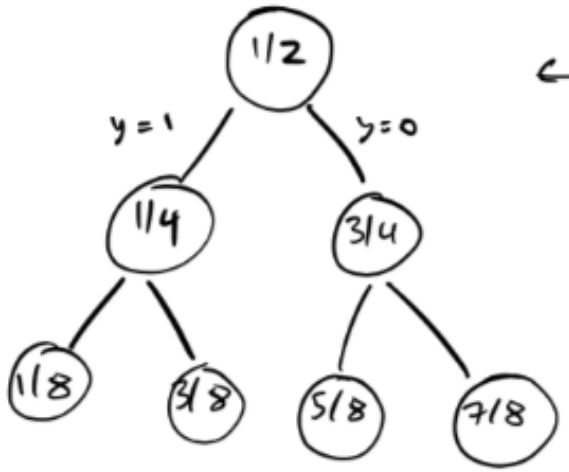
Lemma 6. *For any online learning algorithm* A, $M_A(\mathcal{H}) \geq Ldim(\mathcal{H})$.

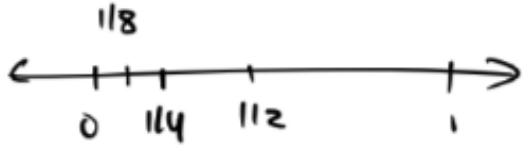
Example 1

1. Let $\mathcal{H}$ be a finite hypothesis class. Then $Ldim(\mathcal{H}) \leq \log_2(\mathcal{H})$.
   $\rightarrow$ Any tree which is shattered by $\mathcal{H}$ must have $|\mathcal{H}| \geq \#$ leaves in the tree.

Example 2

2. $x = [0, 1]$, $\mathcal{H} = \{x \mapsto \mathbb{1}(x < a), a \in [0, 1]\}$ (thresholds on $[0, 1]$). Then, $Ldim(\mathcal{H}) = \infty$





Standard Optimal Algorothm

Standard Optimal Algorithm (SOA)

input: A hypothesis class $\mathcal{H}$

initialize: $V_1 = \mathcal{H}$

for $t = 1, 2, \ldots$

receive $\mathbf{x}_t$

for $r \in \{0, 1\}$ let $V_t^{(r)} = \{h \in V_t : h(\mathbf{x}_t) = r\}$

predict $p_t = \text{argmax}_{r \in \{0,1\}} \text{Ldim}(V_t^{(r)})$

(in case of a tie predict $p_t = 1$)

receive true label y_t

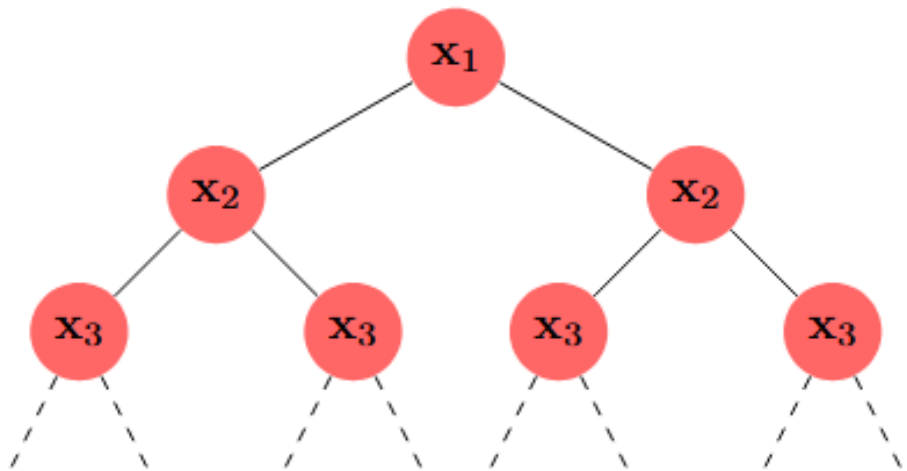
update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$

LEMMA 21.7 *SOA enjoys the mistake bound* $M_{SOA}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$.

Proof It suffices to prove that whenever the algorithm makes a prediction mistake we have $\text{Ldim}(V_{t+1}) \leq \text{Ldim}(V_t) - 1$. We prove this claim by assuming the contrary, that is, $\text{Ldim}(V_{t+1}) = \text{Ldim}(V_t)$. If this holds true, then the definition of p_t implies that $\text{Ldim}(V_t^{(r)}) = \text{Ldim}(V_t)$ for both $r = 1$ and $r = 0$. But, then we can construct a shaterred tree of depth $\text{Ldim}(V_t) + 1$ for the class V_t, which leads to the desired contradiction. □

THEOREM 21.9 *For any class $\mathcal{H}$, $\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H})$, and there are classes for which strict inequality holds. Furthermore, the gap can be arbitrarily larger.*

Proof We first prove that $\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H})$. Suppose $\mathrm{VCdim}(\mathcal{H}) = d$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_d$ be a shattered set. We now construct a complete binary tree of instances $\mathbf{v}_1, \ldots, \mathbf{v}_{2^d-1}$, where all nodes at depth i are set to be $\mathbf{x}_i$ – see the following illustration:



Now, the definition of a shattered set clearly implies that we got a valid shattered tree of depth d, and we conclude that $\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H})$. To show that the gap can be arbitrarily large simply note that the class given in Example 21.4 has VC dimension of 1 whereas its Littlestone dimension is infinite. □

Unrealizable Case: Regret

Formally, the regret of an algorithm A relative to h when running on a sequence of T examples is defined as

$$\text{Regret}_A(h, T) = \sup_{(x_1, y_1), \ldots, (x_T, y_T)} \left[\sum_{t=1}^{T} |p_t - y_t| - \sum_{t=1}^{T} |h(x_t) - y_t| \right], \qquad (21.1)$$

and the regret of the algorithm relative to a hypothesis class $\mathcal{H}$ is

$$\text{Regret}_A(\mathcal{H}, T) = \sup_{h \in \mathcal{H}} \text{Regret}_A(h, T). \qquad (21.2)$$

Weighted Majority Rule

- Mixture of Experts
- Weight vector over the experts denoting probability of choosing an expert in a round
- Weight vector evolves with rounds

Weighted-Majority

input: number of experts, d ; number of rounds, T

parameter: $\eta = \sqrt{2\log(d)/T}$

initialize: $\tilde{\mathbf{w}}^{(1)} = (1, \ldots, 1)$

for $t = 1, 2, \ldots$

set $\mathbf{w}^{(t)} = \tilde{\mathbf{w}}^{(t)}/Z_t$ where $Z_t = \sum_i \tilde{w}_i^{(t)}$

choose expert i at random according to $\mathbb{P}[i] = w_i^{(t)}$

receive costs of all experts $\mathbf{v}_t \in [0,1]^d$

pay cost $\langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle$

update rule $\forall i,\ \tilde{w}_i^{(t+1)} = \tilde{w}_i^{(t)} e^{-\eta v_{t,i}}$

Regret of Weighted Majority Algorithm

THEOREM 21.11 *Assuming that* $T > 2\log(d)$, *the* **Weighted-Majority** *algorithm enjoys the bound*

$$\sum_{t=1}^{T} \langle \mathbf{w}^{(t)}, \mathbf{v}_t \rangle - \min_{i \in [d]} \sum_{t=1}^{T} v_{t,i} \;\leq\; \sqrt{2\log(d)\,T}.$$

COROLLARY 21.14 *Let* $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)$ *be a sequence of examples and let* $\mathcal{H}$ *be a hypothesis class with* $\mathrm{Ldim}(\mathcal{H}) < \infty$. *There exists* $L \leq \mathrm{Ldim}(\mathcal{H})$ *and indices* $1 \leq i_1 < i_2 < \cdots < i_L \leq T$, *such that* $\mathrm{Expert}(i_1, i_2, \ldots, i_L)$ *makes at most as many mistakes as the best* $h \in \mathcal{H}$ *does, namely,*

$$\min_{h \in \mathcal{H}} \sum_{\iota=1}^{T} |h(\mathbf{x}_\iota) - y_\iota|$$

mistakes on the sequence of examples.