Information Storage in Finite Memory

Santosh S. Venkatesh

University of Pennsylvania

A Finite Memory (One Bit!) Problem

Setup:

- Data: $\{u^t \in \mathbb{B}, t \geq 1\}$ is a sequence of symmetric Bernoulli trials.

- Memory Updates: $\{w^t \in \mathbb{B}, t \geq 1\}$ is a sequence of on-line updates of a single bit of memory via a sequence of Boolean functions of two Boolean variables $\{f_t, t \geq 1\}$, which are possibly randomized and time-varying:

$$w^{t+1} = f_t(w^t, u^t) \qquad (t \geq 1).$$

Komlós' Problem: Determine

$$I_n = \sup_{f_1,\ldots,f_n} \min_{1 \leq t \leq n} \mathbf{E}\left(w^{n+1} u^t\right).$$

Theorem . (Venkatesh, Franklin)

$$\frac{1}{n} \leq I_n < \frac{2}{n}.$$

Hierarchical Construction of Boolean Functions

Setup:

- Sequence of (deterministic) Boolean functions:
  $g_k : \mathbb{B}^2 \to \mathbb{B} \quad (k \geq 2)$.

- Recursive specification of Boolean functions:
  $f_k : \mathbb{B}^k \to \mathbb{B} \quad (k \geq 3)$.

$$f_2(X_1, X_2) = g_2(X_1, X_2),$$
$$f_k(X_1, \ldots, X_{k-1}, X_k) = g_k\Big(f_{k-1}(X_1, \ldots, X_{k-1}), X_k\Big).$$

- Hierarchical family $\mathcal{F}_n$: all Boolean functions f_n constructed as above.

Enumerate: What is $|\mathcal{F}_n|$?

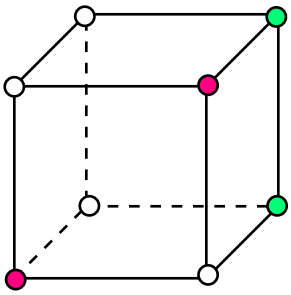
Theorem . (Venkatesh, Franklin)

$$|\mathcal{F}_n| = \frac{2}{5}\,6^n + \frac{8}{5} \qquad (n \geq 2).$$

Learning from Examples

Given: Points (exemplars) $\mathbb{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\} \subseteq \mathbb{B}^n$

Labels $\mathbb{V} = \{\nu^1, \ldots, \nu^m\} \in \mathbb{B}$



Labeled m-sample

$$\mathbb{S} = (\mathbb{U}, \mathbb{V}) = \{(\mathbf{u}^1, \nu^1), \ldots, (\mathbf{u}^m, \nu^m)\}$$

Goal: Find a solution vector (vertex) $\mathbf{w} \in \mathbb{B}^n$ for the labeled sample $\mathbb{S}$ satisfying

$$\operatorname{sgn}\langle \mathbf{w}, \mathbf{u}^t \rangle = \nu^t \qquad (1 \leq t \leq m).$$

Equivalent System : An equivalent set of positive examples may be obtained by replacing $\mathbf{u}^t$ by $-\mathbf{u}^t$ in $\mathbb{U}$ whenever $\nu^t = -1$. Thus, without loss of generality, we may take $\nu^t = +1$ for each $t \geq 1$.

Binary Integer Programming

The Basic Problem : Find $\boldsymbol{w} \in \mathbb{B}^n$ such that

$$\begin{aligned}
\langle \boldsymbol{w}, \mathbf{u}^1 \rangle &= w_1 u_1^1 + w_2 u_2^1 + \cdots + w_n u_n^1 > 0 \\
\langle \boldsymbol{w}, \mathbf{u}^2 \rangle &= w_1 u_1^2 + w_2 u_2^2 + \cdots + w_n u_n^2 > 0 \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\langle \boldsymbol{w}, \mathbf{u}^m \rangle &= w_1 u_1^m + w_2 u_2^m + \cdots + w_n u_n^m > 0
\end{aligned}$$

Algorithm : For every labeled sample $\mathbb{S}$, generate a weight vector $\boldsymbol{w} = \boldsymbol{w}(\mathbb{S}) \in \mathbb{B}^n$.

The Basic Roadblock: The learning problem is NP-complete.

Random Classification Problems

Probability Distribution:

➠ $\mathbb{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ i.i.d., $\mathbf{u}^t \sim \text{Uniform}(\mathbb{B}^n)$.

➠ $\mathbb{V} = \{\nu^1, \ldots, \nu^m\}$ arbitrary and independent of $\mathbb{U}$.

Equivalently: a set of positive examples

$$\mathbb{S} = \{(\mathbf{u}^1, 1), \ldots, (\mathbf{u}^m, 1)\}$$

is generated by independent sampling from the uniform distribution on the vertices $\mathbb{B}^n$ of the cube.

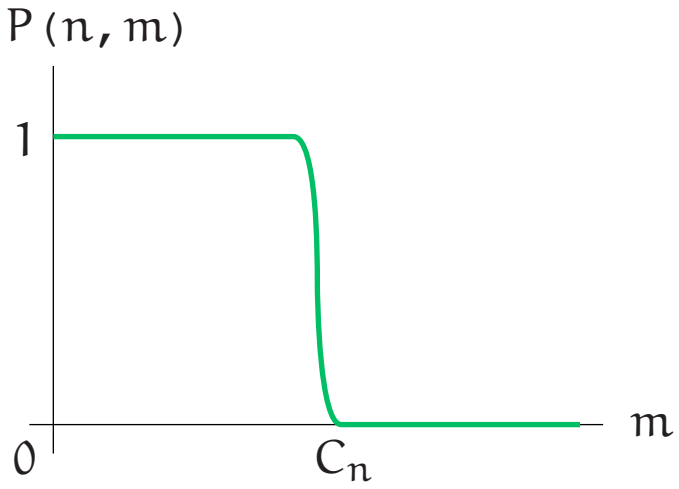
Algorithm : Generates $\boldsymbol{w} = \boldsymbol{w}(\mathbb{S}) \in \mathbb{B}^n$. The algorithm has memorized the data if $\boldsymbol{w}(\mathbb{S})$ is a solution vector:

$$P(n, m) = \Pr\{\boldsymbol{w}(\mathbb{S}) \text{ is a solution vector for } \mathbb{S}\}.$$

Definition: A sequence C_n is a capacity function for the algorithm if, for every $0 < \epsilon < 1$, as $n \to \infty$:

① $P(n, m_n) \to 1$ if $m_n \lesssim (1 - \epsilon) C_n$;

② $P(n, m_n) \to 0$ if $m_n \gtrsim (1 + \epsilon) C_n$.



Concept Learning Problems

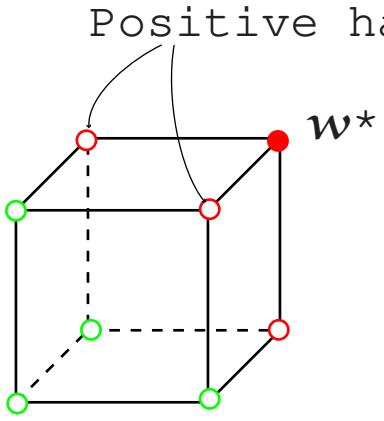
Boolean Concepts: Two equivalent characterizations of a linear threshold Boolean function parameterized by a vertex $w^* \in \mathbb{B}^n$.

- As a look-up table for the Boolean function $f_{w^*}: \mathbb{B}^n \to \mathbb{B}$ defined by

$$f_{w^*}(u) = \operatorname{sgn}\langle w^*, u\rangle \qquad (u \in \mathbb{B}^n).$$

- Via a specification of the positive half-space $\mathbb{B}_+^n(w^*)$ of positive examples of the concept f_{w^*}:

$$\mathbb{B}_n^+(w^*) = \{u \in \mathbb{B}^n : \langle w^*, u\rangle > 0\}.$$



Probability Distribution:

- $\mathbb{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ i.i.d., $\mathbf{u}^t \sim \text{Uniform}\,(\mathbb{B}^n)$.

- For each $t \geq 1$, $v^t = \text{sgn}\langle \boldsymbol{w}^*, \mathbf{u}^t \rangle$ where $\boldsymbol{w}^* \in \mathbb{B}^n$ is a fixed concept.

Equivalently (for odd n): a set of positive examples

$$\mathbb{S} = \{(\mathbf{u}^1, 1), \ldots, (\mathbf{u}^m, 1)\}$$

is generated by independent sampling from the uniform distribution on the positive half-space of vertices

$$\mathbb{B}_n^+(\boldsymbol{w}^*) = \{\mathbf{u} \in \mathbb{B}^n : \langle \boldsymbol{w}^*, \mathbf{u} \rangle > 0\}$$

induced by the concept vector $\boldsymbol{w}^*$.

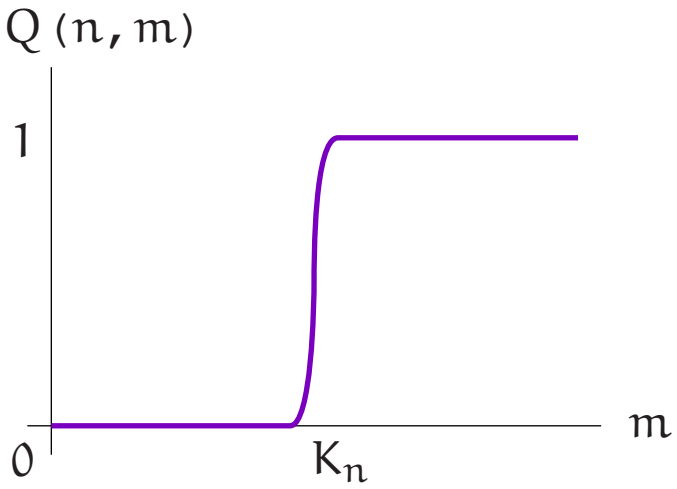
Algorithm : Generates $\boldsymbol{w} = \boldsymbol{w}(\mathbb{S}) \in \mathbb{B}^n$. The algorithm has learnt the concept if $w(\mathbb{S}) = \boldsymbol{w}^*$:

$$Q(n, m) = \Pr\{\boldsymbol{w}(\mathbb{S}) = \boldsymbol{w}^*\}.$$

Definition: A sequence K_n is a sample complexity function for the algorithm if, for every $0 < \epsilon < 1$, as $n \to \infty$:

① $Q(n, m_n) \to 0$ if $m_n \lesssim (1 - \epsilon)K_n$;

② $Q(n, m_n) \to 1$ if $m_n \gtrsim (1 + \epsilon)K_n$.



On-Line Algorithms

- **Data:** Sequence $\{u^t, t \geq 1\}$ of (positive) examples of the concept generated by cycling through the set of exemplars $\mathbb{U}$.

- **Recurrence:** Weight vector updates $\{w^t, t \geq 1\}$ are recursively generated via a sequence of possibly randomized, time-varying rules f_t:

$$w^{t+1} = f_t(w^t, u^t) \qquad (t \geq 1).$$

- **Time Complexity:** Expected mistake bound

$$\mathbf{E}\Big(\#\{t : \langle w^t, u^t \rangle < 0\}\Big).$$

Harmonic Update

- Tranining sequence: $\{u^t, 1 \leq t \leq m\}$.
- Initialization: $w^1 \in \mathbb{B}^n$ arbitrary.
- Recurrence: For $i = 1, \ldots, n$,
  - If $w_i^t = u_i^t$, then set $w_i^{t+1} = w_i^t$;
  - If $w_i^t \neq u_i^t$, then set

$$w_i^{t+1} = \begin{cases} -w_i^t & \text{with probability } 1/t, \\ w_i^t & \text{with probability } 1 - 1/t. \end{cases}$$

- Output Hypothesis: $w = w(\mathbb{S}) = w^{m+1}$.

Theorem . (Fang, Venkatesh) $\sqrt{n}/\sqrt{\log n}$ is a capacity function for Harmonic update.

Directed Drift

- Tranining sequence: $\{\mathbf{u}^t, t \geq 1\}$.
- Initialization: $\mathbf{w}^1 \in \mathbb{B}^n$ arbitrary.
- Positive check: If $\langle \mathbf{w}^t, \mathbf{u}^t \rangle \geq 0$, set $\mathbf{w}^{t+1} = \mathbf{w}^t$.
- Negative check: If $\langle \mathbf{w}^t, \mathbf{u}^t \rangle < 0$, pick a random index j uniformly from the set of mismatched indices $\{i : w_i^t \neq u_i^t\}$ and set

$$w_i^{t+1} = \begin{cases} w_i^t & \text{if } i \neq j, \\ -w_i^t & \text{if } i = j. \end{cases}$$

Theorems.

- (Kim, Roche) Directed Drift has capacity function $c_1 n$ for a constant $c_1 < 1$.

- (Baum) Directed Drift has sample complexity function $c_2 n$ for a constant $1 < c_2 < 2$.

- (Fang, Venkatesh) Directed Drift converges to w^* in finite time with expected mistake bound bounded between $\Omega(e^{0.139n})$ and $O(e^{n \log n})$.

Majority Rule

Off-Line Algorithm. For each $i = 1, \ldots, n$, set

$$w_i = \operatorname{sgn}\left(\sum_{t=1}^{m} u_i^t\right).$$

This is the vertex closest to the centroid of $\mathbb{U}$, the set of positive examples.

Theorems.

- (Fang, Venkatesh) Majority Rule has capacity function $n/\pi \log n$.

- (Fang, Venkatesh) Majority Rule has sample complexity function $\pi n \log n$.