On learning k-parities and disjunctions

Presented by,
Monika Bansal

SPC Lab
Department of ECE
Indian Institute of Science

September 19, 2015

Overview

Introduction

On-line learning setup:

hidden vector $f \in \{0,1\}^n$ with $|f| = k$

learning examples $x_1, x_2, \ldots \in \{0,1\}^n$

true labels $y_1, y_2, \ldots \in \{0,1\}$

predicted labels $\hat{y}_1, \hat{y}_2, \ldots \in \{0,1\}$

Parity label

$$y_i = \sum_j x_j f_j \pmod{2}$$

Disjunction label

$$y_i = \begin{cases} 1 \text{ if } & \sum_j x_j f_j \geq 1 \,, \\ 0 \text{ if } & \sum_j x_j f_j = 0 \end{cases}$$

Introduction

Concept: A function $f : X \to Y$
where, $X = \{0,1\}^n$ instance space
$Y = \{0,1\}$ label set

Concept class: Set of concepts $\mathcal{C}$

Examples:

- Concept class of k-parities: $x_{i_1} \otimes x_{i_2} \ldots \otimes x_{i_k}$
- Concept class of disjunctions: $x_{i_1} \vee x_{i_2} \ldots \vee x_{i_k}$

Learning Model

- Mistake Bound Model:
  Given a fixed concept class $\mathcal{C}$, come up with an algorithm A such that, $\forall f \in \mathcal{C}, \forall$ sequence of examples$(x_1, x_2, ...)$, A makes at most m mistakes.

- PAC Learning Model:
  A learning algorithm is said to be PAC-learn $\mathcal{C}$ with *approximation parameter* ϵ *and confidence parameter* δ if $\forall$ distributions $\mathcal{D}$ and all target functions $f \in \mathcal{C}$, the algorithm draws at most s samples, runs for time at most t and outputs a function f^* such that, w.p. $1 - \delta$

$$\mathbf{Pr}_{x \leftarrow \mathcal{D}}[f(x) \neq f^*(x)] < \epsilon$$

Transformation from mistake bound model to PAC learning

- **Mistake bound model:** Mistake bound m,
  Running time per round t

- **PAC learning model:** Sample complexity $O(\frac{1}{\epsilon}m + \frac{1}{\epsilon}log\frac{1}{\delta})$,
  Running time $O(\frac{1}{\epsilon}mt + \frac{t}{\epsilon}log\frac{1}{\delta})$

Learning Algorithms for k-parities

Algorithm 1 (halving algorithm)

Given a target class $\mathcal{C}$, $x \in X$ and $N_{CONSIST} = \mathcal{C}$

$\hat{y} = 1 \qquad \text{if} \quad |N_{CONSIST}(x,1)| > |N_{CONSIST}(x,0)|$

where, $N_{\mathcal{C}}(x,0)$: set of those functions that are **0 at** x **in** $\mathcal{C}$
$N_{\mathcal{C}}(x,1)$: set of those functions that are **1 at** x **in** $\mathcal{C}$

Update:

$$N_{CONSIST} := N_{CONSIST}(x,y)$$

Mistake occurs if $\hat{y} \neq y$

$$M_{HALVING}(\mathcal{C}) \leq log_2|\mathcal{C}|$$

Algorithm 2

"Learning Parities in the Mistake-Bound Model"
Authors: H.Buhrman, David, A.matsliah

Let $\{e_1, e_2..., e_n\}$ standard basis for $\{0,1\}^n$

$\pi = C_1, ..., C_t$

Define $\mathcal{S} = \{s \subseteq [t] : |s| = k\}$, hence $|\mathcal{S}| = \binom{t}{k}$

subspace $M_s = span(U_s)$ where, $U_s \triangleq \bigcup_{i \in s} C_i$

$|M_s| < 2^{k\lceil n/t \rceil}$

- Every $f \in \{0,1\}^n$ with $|f| \leq k$ is in some M_s
- $|\bigcup_{s \in \mathcal{S}} M_s| \leq \sum_{s \in \mathcal{S}} |M_s| \leq \binom{t}{k} 2^{k\lceil n/t \rceil}$

Algorithm 2

Initialize: $N_s = M_s$ for all $s \in \mathcal{S}$

Compute:

$$n_0 = \sum_{s \in \mathcal{S}} |N_s(x, 0)|$$

$$n_1 = \sum_{s \in \mathcal{S}} |N_s(x, 1)|$$

$\hat{y} = 1$ if $n_1 > n_0$

Update: $N_s := N_s(x, y)$ for each $s \in \mathcal{S}$,

Mistake Bound:

$$M_{Algo2} \leq log\Big(\sum_{s \in \mathcal{S}} |M_s|\Big) \leq k\lceil n/t \rceil + \lceil log \binom{t}{k} \rceil$$

Running time: $O\big(\binom{t}{k}(kn/t)^2\big)$

Algorithm 3

"On learning k-parities with and without noise"
Authors: A.Bhattacharyya, A.Gadekar, N.Rajgopal

$$\pi = C_1, ..., C_T, \qquad T = \alpha t$$

$\mathcal{S}_1, ..., \mathcal{S}_m \subset [T]$, Random subsests with $|\mathcal{S}_i| = \alpha k$

$$M_i = span(U_{j \in \mathcal{S}_i} C_j)$$

$$|M_i| \leq 2^{\alpha k \lceil n/T \rceil} \leq 2^{kn/t + \alpha k} = 2^{(1+o(1))kn/t}$$

Initialise: $N_i = M_i$ for all $i \in [m]$

$$\hat{y} = 1 \qquad \text{if} \qquad \sum_{i \in [m]} |N_i(x, 1)| \geq \sum_{i \in [m]} |N_i(x, 0)|$$

Update: $N_i := N_i(x, y)$ for each $i \in [m]$

Algorithm 3

Choose m such that *for every set* $A \subset [T]$ *of size* k,
$A \subset \mathcal{S}_i$ *for some* $i \in [m]$ *with nonzero probability.*

$$\mathbf{Pr}(A \subset \mathcal{S}_i) = \binom{T-k}{\alpha k - k} / \binom{T}{\alpha k}$$

$$\mathbf{Pr}[\forall i \in [m], A \not\subset \mathcal{S}_i] = \left(1 - \binom{T-k}{\alpha k - k} / \binom{T}{\alpha k}\right)^m \leq e^{-m\binom{T-k}{\alpha k-k}/\binom{T}{\alpha k}}$$

$$m = 2\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}} \underbrace{\log\binom{T}{k}}_{\text{Due to Union bound}} = \tilde{O}\left(\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}}\right)$$

Algorithm 3

If α is a large enough constant,

$$\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}} \leq e^{-k/4.01}\binom{t}{k}$$

Mistake bound:

$$log\big(\sum_i |N_i|\big) \leq log\Big[\tilde{O}\Big(\frac{\binom{T}{\alpha k}}{\binom{T-k}{\alpha k-k}}\Big)2^{(1+o(1))kn/t}\Big]$$

$$\leq (1+o(1))kn/t + log\binom{t}{k} - \Omega(k) + logO\Big(log\binom{t}{k}\Big)$$

Running time:

$$O(ml^2) \leq e^{-k/4.01}.\binom{t}{k}.\tilde{O}\big((kn/t)^2\big)$$

Learning Algorithm for disjunctions

Algorithm 1 (halving algorithm): Same as of k-parities

Algorithm 2 (WINNOW)

w_i is non-negative real-valued weight and $(x_1, x_2, .., x_n) \in X$

$$If \sum_{i=1}^{n} w_i x_i > \theta \ \ then \ \ \hat{y} = 1$$

$$If \sum_{i=1}^{n} w_i x_i \leq \theta \ \ then \ \ \hat{y} = 0$$

learner's prediction	*correct response*	*update action*	*update name*
1	0	$w_i := 0$ if $x_i = 1$ w_i unchanged if $x_i = 0$	elimination step
0	1	$w_i := \alpha \cdot w_i$ if $x_i = 1$ w_i unchanged if $x_i = 0$	promotion step

Mistake bound: $\alpha k(log_\alpha \theta + 1) + \frac{n}{\theta}, \alpha > 1$ and $\theta \geq 1$

Goal

- To come up with the connection between Group Testing algorithms and learning algorithms.
- Comment on the performances.