BDUOL: Double Updating Online Learning on a Fixed Budget

Peilin Zhao and Steven C.H. Hoi {zhao0106@ntu.edu.sg, chhoi@ntu.edu.sg}
Nanyang Technological University, Singapore

ECML-PKDD 2012

NANYANG TECHNOLOGICAL UNIVERSITY

Settings

- Receive example $\mathbf{x}_t \in \mathbb{R}^d$
- Make prediction $\widehat{y}_t = \text{sgn}(f_{t-1}(\mathbf{x}_t))$, where $f_{t-1}(\cdot) = \sum_{i \in S_{t-1}} \widehat{\gamma}_i y_i \kappa(\mathbf{x}_i, \cdot)$ and $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any $\mathbf{x} \in \mathbb{R}^d$
- Reveal true label $y_t \in \{-1, +1\}$ and suffer loss $\ell(f_{t-1}(\mathbf{x}_t), y_t) = \max(0, 1 - y_t f_{t-1}(\mathbf{x}_t)) : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$.

Double Updating Online Learning

Motivation

- if $y_t f_{t-1}(\mathbf{x}_t) \leq -\gamma$, then $\exists\ (\mathbf{x}_b, y_b)$, $b \in \mathcal{S}_{t-1}$ s.t. $\beta y_t y_b k(\mathbf{x}_t, \mathbf{x}_b) \leq -\beta\gamma/|S_{t-1}|$.
- if $f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) + \beta y_t \kappa(\mathbf{x}_t, \mathbf{x})$ and $y_b f_{t-1}(\mathbf{x}_b) \leq \beta\gamma/n$, then $y_b f_t(\mathbf{x}_b) \leq 0$.

to alleviate this problem, we propose to update the weights for $(\mathbf{x}_b, y_b)$, $b \in S_{t-1}$ and $(\mathbf{x}_t, y_t)$, when they suffer **conflict**, i.e.,

- $\ell_t = 1 - y_t f_{t-1}(\mathbf{x}_t) > 0$;
- $\ell_b = 1 - y_b f_{t-1}(\mathbf{x}_b) > 0$;
- $y_t y_b \kappa(\mathbf{x}_t, \mathbf{x}_b) \leq \min(-\rho, y_t y_a \kappa(\mathbf{x}_t, \mathbf{x}_a))$, $a \in S_{t-1}$, and $a \neq b$, where $\rho \in [0, 1)$ is a threshold,

Double Updating: if conflict exists

$$f_t(\cdot) = f_{t-1}(\cdot) + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot),$$

where (γ_t, d_{γ_b}) are in the following equations:

$$\begin{cases} (C,\ C - \hat{\gamma}_b) & \text{if } (k_t C + w_{ab}(C - \hat{\gamma}_b) - \ell_t) < 0 \text{ and } (k_b(C - \hat{\gamma}_b) + w_{ab} C - \ell_b) < 0 \\ (C,\ \frac{\ell_b - w_{ab} C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_t k_b C + k_b \ell_t}{k_b} > 0 \text{ and } \frac{\ell_b - w_{ab} C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t},\ C - \hat{\gamma}_b) & \text{if } \frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t} \in [0, C] \text{ and } \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_t - w_{ab}(C - \hat{\gamma}_b)}{k_t} > 0 \\ (\frac{k_b \ell_t - w_{ab}\ell_b}{k_t k_b - w_{ab}^2},\ \frac{k_t \ell_b - w_{ab}\ell_t}{k_t k_b - w_{ab}^2}) & \text{if } \frac{k_b \ell_t - w_{ab}\ell_b}{k_t k_b - w_{ab}^2} \in [0, C] \text{ and } \frac{k_t \ell_b - w_{ab}\ell_t}{k_t k_b - w_{ab}^2} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}$$

where $k_t = \kappa(\mathbf{x}_t, \mathbf{x}_t), k_b = \kappa(\mathbf{x}_b, \mathbf{x}_b)$, $w_{ab} = y_t y_b \kappa(\mathbf{x}_t, \mathbf{x}_b)$ and $C > 0$;

Single Updating: if no conflict exists

$$f_t(\cdot) = f_{t-1}(\cdot) + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot),$$

where $\gamma_t = \min(C, \ell_t / k_t^2)$.

Motivation

Potentially unbounded number of support vectors

- Large amount of memory for storing SVs
- high computational cost per iteration
- not suitable for large scale applications

Budget DUOL

when $|SV_{t-1}| = B$, BDUOL performs: $f_{t-1} \leftarrow f_{t-1} - \Delta f_{t-1}$ such that the support vector size of the updated f_{t-1} is smaller than B.

BDUOL : Algorithm

- Receive example $\mathbf{x}_t \in \mathbb{R}^d$
- Make prediction $\widehat{y}_t = \text{sgn}(f_{t-1}(\mathbf{x}_t))$
- Reveal true label $y_t \in \{-1, +1\}$
- Suffer loss $\ell(f_{t-1}(\mathbf{x}_t), y_t)$
- Budget Maintenance: $f_{t-1} \leftarrow f_{t-1} - \Delta f_{t-1}$
- Regular DUOL Update

BDUOL : Analysis

Theorem 1 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t, and assume $C \geq 1$. Then for any function f in $\mathcal{H}_\kappa$, the number of prediction mistakes M made by BDUOL on this sequence of examples is bounded by:*

$$2 \min_{f \in \mathcal{H}_\kappa} \left\{ \frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^{T} \ell(f(x_i), y_i) \right\} - 2 \sum_{i=1}^{T} DA_i - \frac{\rho^2}{2} M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$.

To minimize the bound, maximize the following value

$$\max_{\Delta\gamma_1, \ldots, \Delta\gamma_t} DA_t = -\sum_{i=1}^{t} \Delta\gamma_i + \sum_{i=1}^{t} \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2}\|\Delta f_t\|_{\mathcal{H}_\kappa}^2.$$

Budget Maintenance Strategies

Removal Strategy: assume the j-th SV is removed

$$\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot).$$

the solution is to discard the SV maximizing

$$DA_{t,j} = -\gamma_j (1 - y_j f_t(\mathbf{x}_j)) - \frac{1}{2}(\gamma_j)^2 \kappa(\mathbf{x}_j, \mathbf{x}_j).$$

Projection Strategy (exact): the j-th SV for removal will be projected to the space spanned by the rest SVs, formally:

$$\min_{\beta_i \in [-\gamma_i, C - \gamma_i], i \neq j} \|\gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot)\|_{\mathcal{H}_\kappa}^2.$$

which is essentially a QP problem.

Projection Strategy (approximate): firstly solve the unconstrained optimization and then project the solution into the feasible domain

1 set the gradient with respect to $\overline{\beta} = [\beta_i]^\top, i \neq j$ as zero:

$$\overline{\beta} = \gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y},$$

where $\mathbf{K}$ is the kernel matrix for $\mathbf{x}_i, i \neq j$, $\mathbf{k}_j = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]^\top$, $i \neq j$, $./$ is element-wise division and $\overline{y} = [y_i]^\top, i \neq j$.

2 project each β_i as follows:

$$\overline{\beta} = \Pi_{[-\overline{\gamma}, C - \overline{\gamma}]}(\gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y}),$$

where $\Pi_{[a,b]}(x) = \max(a, \min(b, x))$ and $\overline{\gamma} = [\gamma_i]^\top, i \neq j$.

Nearest Neighbor Strategy: project the removed SV to its nearest neighbor SV, based on the distance in the mapped feature space

- the corresponding solution can be expressed as:

$$\beta_{N_j} = \Pi_{[-\gamma_{N_j}, C - \gamma_{N_j}]}(\gamma_j y_j \kappa(\mathbf{x}_{N_j}, \mathbf{x}_{N_j})^{-1} \kappa(\mathbf{x}_{N_j}, \mathbf{x}_j) / y_{N_j}),$$

where $\kappa(\mathbf{x}_{N_j}, \cdot)$ is the nearest neighbor of $\kappa(\mathbf{x}_j, \cdot)$.

- As a result, the corresponding $\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \beta_{N_j} y_{N_j} \kappa(\mathbf{x}_{N_j}, \cdot)$.

Experiments : Testbed and Setup

Dataset	# instances	# features
german	1000	24
MITface	6977	361
mushrooms	8124	112
spambase	4601	57
splice	3175	60
w7a	24692	300

Compared algorithms

1. "RBP": the Random Budget Perceptron,
2. "Forgetron",
3. "Projectron",
4. "Projectron++": the aggressive Projectron.

Proposed algorithms

1. "BDUOL$_{remo}$": **remo**val,
2. "BDUOL$_{proj}$": exact **proj**ection,
3. "BDUOL$_{appr}$": **appr**oximate projection,
4. "BDUOL$_{near}$": **near**est neighbor.

Parameter setting:

- Gaussian kernel with kernel width 8
- C is from $2^{[-10:10]}$ by CV
- ρ is set as 0
- B as proper fractions of the SV size

The results were by averaging over runs on 20 random permutations for each dataset.

Experiments : Main Results

Budget Size	B=500		B=1000	
Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
RBP	31.826 ± 0.924	0.065	29.220 ± 0.550	0.069
Fogetron	32.461 ± 0.971	0.077	29.641 ± 0.742	0.084
Projectron	29.237 ± 0.750	0.238	27.480 ± 0.484	0.929
Projectron++	28.822 ± 0.725	0.819	28.693 ± 6.781	3.874
BDUOL$_{remo}$	34.559 ± 1.308	0.522	28.989 ± 0.927	2.013
BDUOL$_{near}$	28.180 ± 1.084	0.756	25.607 ± 0.748	3.874
BDUOL$_{appr}$	28.950 ± 2.001	5.555	26.843 ± 0.948	16.396
BDUOL$_{proj}$	**27.448** ± 0.961	8.837	**25.307** ± 0.793	53.987

Observations:

- RBP and Forgetron achieve similar performance
- Projectron++ achieves a lower mistake rate than Projectron
- BDUOL$_{remo}$ algorithm achieves comparable or better mistake rate when the budget size is large, while fails to improve otherwise
- BDUOL$_{proj}$ always achieves the lowest mistake rates, while suffers the highest time consumption
- BDUOL$_{near}$ achieves better trade off between mistake rates and time complexity than BDUOL$_{appr}$