BDUOL: Double Updating Online Learning on a Fixed Budget

Peilin Zhao[†] Steven C.H. Hoi[†]

[†]Nanyang Technological University, Singapore

Sep 01, 2012

Outline

Online Learning

Online Classification

The learner firstly receives an instance, then predicts the label, and then receives the true label, finally updates the prediction function.

Why Online Learning

- In many cases, data arrives sequentially while predictions are required immediately.

Examples: spam filtering, stock market prediction

- For some large scale problem, it is time consuming to use off-line learning.

Examples: large scale support vector machine training

Kernel-based online learning

Online Prediction Function

Sequentially update a nonlinear kernel-based classfier :

$$f_t(\cdot) = \sum_{i \in \mathcal{SV}} \alpha_i y_i \kappa(\mathbf{x}_i, \cdot) \tag{1}$$

Can increase the modeling capacity over linear online learning and usually improve the prediction performance.

Limitation

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 Online Learning

SV removal

- Random Budget Perceptron(RBP, Cavallanti et al, Machine Learning 2007): randomly remove support vector
- Forgetron(Dekel et al, NIPS 2005): remove the support vector with least weight

SV projection(Orabona et al, ICML 2008, JMLR 2009)

- cannot guarantee the exact number of support vectors
- high computational cost of the projection step

Problem Setting

Kernel-based Online Classification

- Receive Example $\mathbf{x}_t \in \mathbb{R}^d$
- Make Prediction $\widehat{y}_t = \text{sgn}(f_{t-1}(\mathbf{x}_t))$, the predicted label. Where $f_{t-1} \in \mathcal{H}_\kappa$, $\mathcal{H}_\kappa$ is an RKHS endowed with a kernel function κ 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 $y_t \in \{-1, +1\}$
- 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 (DUOL)

Prediction

the t-th step, $\widehat{y}_t = \text{sgn}(f_{t-1}(\mathbf{x}_t))$ using:

$$f_{t-1}(\cdot) = \sum_{i \in S_{t-1}} \widehat{\gamma}_i y_i \kappa(\mathbf{x}_i, \cdot),$$

where S_{t-1} is the index set of the SVs for the $(t-1)$-th step, and $\widehat{\gamma}_i$ is the weight of the i-th existing support vector.

Double Updating Online Learning (DUOL)

Motivation

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

In order to alleviate this problem, we propose to update the weight for the existing support vector whose classification confidence is significantly affected by the new misclassified example.

Conflict

$(\mathbf{x}_t, y_t)$ conflicts with $(\mathbf{x}_b, y_b)$, $b \in S_{t-1}$, when the following conditions are satisfied:

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

Double Updating Online Learning (DUOL)

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 and d_{γ_b} are computed in the following equations:

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

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 $C > 0$;

Double Updating Online Learning (DUOL)

Single Updating: if no conflict exists

$$f_t(\cdot) = f_{t-1}(\cdot) + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot), \tag{3}$$

where $\gamma_t = \min(C, \ell_t / k_t^2)$.
Remark: It is not difficult to see that the single update strategy is reduced to the Passive-Aggressive updating strategy (Crammer et al, JMLR 2006).

Budget Double Updating Online Learning

Key Issue

- to ensure $|SV_t|$ less than a pre-fixed budget B

Solution

- 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.

Algorithm

Initialize $S_0 = \emptyset$, $f_0 = 0$;
for $t = 1, 2, \ldots, T$ **do**
  Receive a new instance $\mathbf{x}_t$;
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$;
  Receive its label y_t;
  $\ell_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$;
  if $\ell_t > 0$ **then**
    if $(|S_t| == B)$ **then**
      $f_{t-1} = f_{t-1} - \Delta f_{t-1}$; (Budget Maintenance)
    end if
    $\ell_t = max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$;
    if $\ell_t > 0$ **then**
      Regular DUOL update
    end if
  end if
end for
Return f_T, S_T

Theoretical Analysis

Theorem

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:*

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

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

To minimize the bound, maximize the following value

$$\max_{\Delta\gamma_1, \ldots, \Delta\gamma_t} DA_t = -\sum_{i=1}^{t} \Delta\gamma_i + \sum_{i=1}^{t} \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2} \|\Delta f_t\|^2_{\mathcal{H}_\kappa}. \qquad (4)$$

Budget Maintenance Strategies

BDUOL Algorithm by Removal Strategy

assume the j-th SV is removed

$$\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot). \tag{5}$$

the optimal solution is to discard the SV which can maximize

$$DA_{t,j} = -\gamma_j(1 - y_j f_t(\mathbf{x}_j)) - \frac{1}{2}(\gamma_j)^2 \kappa(\mathbf{x}_j, \mathbf{x}_j). \tag{6}$$

Budget Maintenance Strategies

BDUOL Algorithm by Projection Strategy (exact)

the j-th SV for removal will be projected to the space spanned by the rest SVs, formally:

$$\min_{\beta_i \in [-\gamma_i, C-\gamma_i], i \neq j} \|\gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot)\|^2_{\mathcal{H}_\kappa}. \tag{7}$$

which is essentially a Quadratic Programming (QP) problem.

Budget Maintenance Strategies

BDUOL Algorithm by Projection Strategy (approximate)

firstly solve the unconstrained optimization problem and then project the solution into the feasible region of the constraints

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

$$\overline{\beta} = \gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y}, \tag{8}$$

  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 $\overline{y} = [y_i]^\top, i \neq j$.

- project each β_i as follows:

$$\overline{\beta} = \Pi_{[-\overline{\gamma}, C-\overline{\gamma}]}(\gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j ./ \overline{y}), \tag{9}$$

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

Budget Maintenance Strategies

BDUOL Algorithm by 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:

$$\beta_{N_j} = \Pi_{[-\gamma_{N_j}, C-\gamma_{N_j}]}(\gamma_j y_j \kappa(\mathbf{x}_{N_j}, \mathbf{x}_{N_j})^{-1} \kappa(\mathbf{x}_{N_j}, \mathbf{x}_j)/y_{N_j}), \quad (10)$$

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

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

Experimental Datasets

Table: Details of the datasets in our experiments.

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

All the datasets are downloaded from LIBSVM website [1] and UCI machine learning repository [2].

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html

Algorithms

Compared Algorithms

- “RBP": the Random Budget Perceptron algorithm,
- “Forgetron": the Forgetron algorithm,
- “Projectron": the Projectron algorithm, and
- “Projectron++": the aggressive version of Projectron algorithm.

The Proposed Algorithms

- “BDUOL$_{remo}$": the BDUOL algorithm by the **remo**val strategy,
- “BDUOL$_{proj}$": the BDUOL algorithm by the exact **proj**ection strategy,
- “BDUOL$_{appr}$": the BDUOL algorithm by the **appr**oximate projection strategy ,
- “BDUOL$_{near}$": the BDUOL algorithm by the **near**est neighbor strategy.

Parameter Setting

- A Gaussian kernel with kernel width 8
- C is selected with cross validation from $2^{[-10:10]}$
- ρ is set as 0
- B for different datasets are set as proper fractions of the support vector size of Perceptron

- All the experiments were conducted over 20 random permutations for each dataset.
- The results were reported by averaging over these 20 runs.

Budget Size	B=500		B=1000		B=2000	
Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
RBP	31.826 %± 0.924	0.065	29.220 %± 0.550	0.069	27.417 %± 0.598	0.059
Fogetron	32.461 %± 0.971	0.077	29.641 %± 0.742	0.084	27.424 %± 0.644	0.082
Projectron	29.237 %± 0.750	0.238	27.480 %± 0.484	0.929	27.750 %± 2.134	2.776
Projectron++	28.822 %± 0.725	0.819	28.693 %± 6.781	3.874	27.559 %± 1.572	7.277
BDUOL$_{remo}$	34.559 %± 1.308	0.522	28.989 %± 0.927	2.013	25.661 %± 0.709	4.541
BDUOL$_{near}$	28.180 %± 1.084	0.756	25.607 %± 0.748	3.874	24.187 %± 0.584	8.570
BDUOL$_{appr}$	28.950 %± 2.001	5.555	26.843 %± 0.948	16.396	24.846 %± 0.695	30.186
BDUOL$_{proj}$	**27.448** %± 0.961	8.837	**25.307** %± 0.793	53.987	**23.780** %± 0.640	197.199

Table: Evaluation of several budgeted algorithms on spambase.

- 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$_{appr}$

Budget Size	B=200		B=400		B=600	
Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
RBP	6.551 %± 1.056	0.054	17.841 %± 1.539	0.073	3.488 %± 0.713	0.049
Fogetron	11.273 %± 1.750	0.085	14.154 %± 2.748	0.094	3.895 %± 1.141	0.079
Projectron	4.264 %± 0.613	0.095	3.703 %± 0.731	0.099	3.207 %± 0.473	0.107
Projectron++	3.986 %± 0.297	0.161	3.557 %± 0.174	0.178	3.484 %± 0.117	0.197
BDUOL$_{remo}$	8.754 %± 2.438	0.210	2.547 %± 0.865	0.181	0.891 %± 0.152	0.162
BDUOL$_{near}$	1.329 %± 0.198	0.141	0.710 %± 0.095	0.132	0.618 %± 0.081	0.131
BDUOL$_{appr}$	1.065 %± 0.218	0.193	0.667 %± 0.092	0.194	0.623 %± 0.080	0.248
BDUOL$_{proj}$	**0.729** %± 0.060	0.230	**0.604** %± 0.080	0.266	**0.577** %± 0.071	0.290
Budget Size	B=500		B=1000		B=1500	
Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
RBP	4.681 %± 0.257	0.291	4.597 %± 0.227	0.316	4.434 %± 0.108	0.332
Fogetron	4.697 %± 0.102	0.367	4.584 %± 0.180	0.390	4.466 %± 0.137	0.398
Projectron	4.680 %± 0.290	0.727	4.625 %± 0.456	1.174	4.406 %± 0.130	1.929
Projectron++	3.880 %± 0.526	4.366	3.672 %± 0.214	7.763	3.666 %± 0.152	8.796
BDUOL$_{remo}$	3.832 %± 0.170	0.995	3.361 %± 0.145	1.824	**3.175** %± 0.133	2.401
BDUOL$_{near}$	3.572 %± 0.114	1.241	3.269 %± 0.133	3.324	3.183 %± 0.087	4.859
BDUOL$_{appr}$	4.126 %± 0.502	4.553	4.001 %± 0.153	6.794	3.775 %± 0.121	9.646
BDUOL$_{proj}$	**3.367** %± 0.096	11.306	**3.227** %± 0.126	18.965	3.254 %± 0.332	29.807

Table: Evaluation on mushrooms and w7a.

Conclusion

- We presented a new framework of budget double updating online learning for kernel-based online learning on a fixed budget
- We theoretically analyzed the bounds of the proposed algorithms.
- We proposed three budget maintenance strategies: removal, projection, and nearest neighbor
- Encouraging results show the proposed algorithms are effective.

Thank you!

Any question?