Online Machine Learning

Simone Filice
filice.simone@gmail.com
University of Roma Tor Vergata

Web Mining e Retrieval 2013/2014

Motivations

- Common ML algorithms simultaneously exploit a whole dataset. This process, referred as *batch learning*, is not practical when:
  - New data naturally arise over the time: exploiting new data means building from scratch a new model → usually not feasible!
  - The dataset is too large to be efficiently exploited: memory and computational problems!
  - The concept we need to learn changes over the time: batch learning provide a static solution that will surely degrade as time goes by

Online Machine Learning

- Incremental Learning Paradigm:
  - Every time a new example is available, the learned hypothesis is updated
- Inherent Appealing Characteristics:
  - The model does not need to be re-generated from scratch when new data is available
  - Capability of tracking a Shifting Concept
  - Faster training process if compared to batch learners (e.g. SVM)

Overview

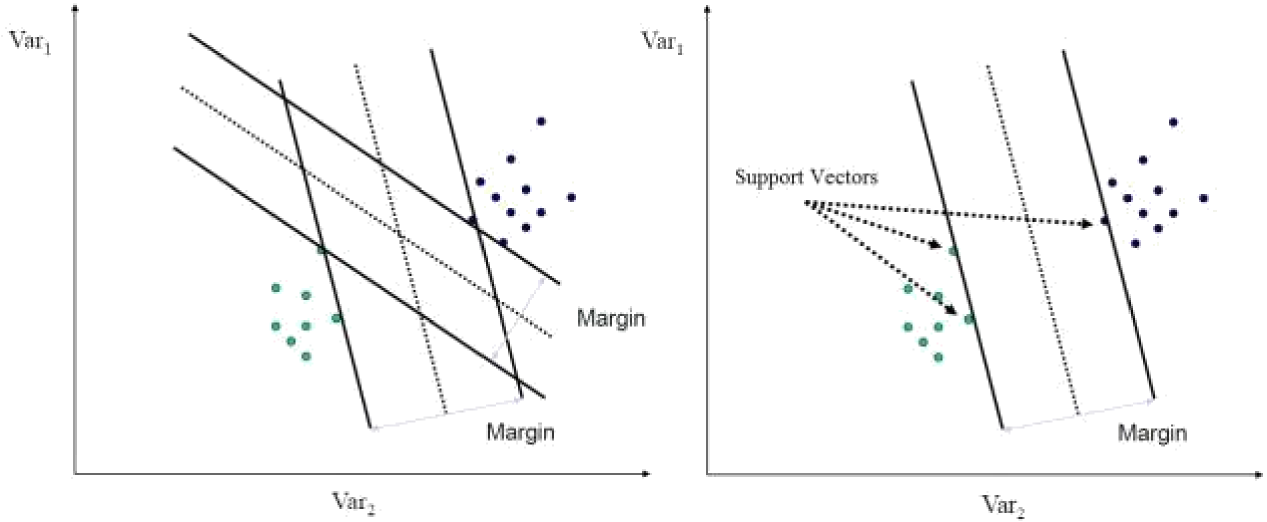
- Linear Online Learning Algorithms
- Kernelized Online Learning Algorithms
- Online Learning on a Budget

Overview

- **Linear Online Learning Algorithms**
- Kernelized Online Learning Algorithms
- Online Learning on a Budget

Perceptron

- Perceptron is a simple discriminative classifier
  - Instances are feature vectors $\boldsymbol{x}' \in \mathbb{R}^d$ with label $y \in [-1, +1]$
  - Classification function is an hyperplane in $\mathbb{R}^d$: $f(\boldsymbol{x}') = \boldsymbol{w}' \cdot \boldsymbol{x}' + b$



- Compact notation: $\boldsymbol{w} = \{b, w'_1, w'_2, \ldots, w'_d\}$, $\boldsymbol{x} = \{1, x'_1, x'_2, \ldots, x'_d\}$

Batch Perceptron

- IDEA : adjust the hyperplane until no training errors are done (input data must be linearly separable)
- Batch perceptron learning procedure:

Start with $\boldsymbol{w}_1 = 0$

do

errors=false

For all t=1…T

Receive a new sample $\boldsymbol{x_t}$

Compute $y = \boldsymbol{w}_t \cdot \boldsymbol{x_t}$

if $y \cdot y_t < \beta_t$ then $\boldsymbol{w}_{t+1} = \gamma_t \boldsymbol{w}_t + \alpha_t y_t \boldsymbol{x}_t$ with $\alpha_t > 0$
errors=true

else

$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$

while(errors)

return $\boldsymbol{w}_{T+1}$

Online Learning Perceptron

- IDEA : adjust the hyperplane after each classification ($\boldsymbol{w}_t$ = weight vector at time *t*) and never stop learning

- Online perceptron learning procedure:

```
Start with w_1 = 0
For all t=1…
      Receive a new sample x_t
      Compute y = w_t · x_t
      Receive a feedback y_t
      if y · y_t < β_t then w_{t+1} = γ_t w_t + α_t y_t x_t    with α_t > 0
      else   w_{t+1} = w_t
endfor
```

Start with $\boldsymbol{w}_1 = 0$

For all t=1…

Receive a new sample $\boldsymbol{x}_t$

Compute $y = \boldsymbol{w}_t \cdot \boldsymbol{x}_t$

Receive a feedback y_t

if $y \cdot y_t < \beta_t$ then $\boldsymbol{w}_{t+1} = \gamma_t \boldsymbol{w}_t + \alpha_t y_t \boldsymbol{x}_t$ with $\alpha_t > 0$

else $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$

endfor

Shifting Perceptron

- IDEA: weak dependance from the past in order to obtain a tracking ability

- Shifting Perceptron learning procedure (Cavallanti et al 2006):

Start with $\boldsymbol{w}_1 = 0$, k=0

For all t=1…

  Receive a new sample $\boldsymbol{x}_t$

  Compute $y = \text{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$

  Receive a feedback y_t

  if $y \neq y_t$ then

    $\lambda_k = \frac{\lambda}{\lambda+k}$ with $\lambda > 0$

    $\boldsymbol{w}_{t+1} = (1 - \lambda_k)\boldsymbol{w}_t + \lambda_k y_t \boldsymbol{x}_t$

    k=k+1

  else $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$

endfor

Online Linear Passive Aggressive (1/3)

- IDEA: Every time a new example $\langle x_t, y_t \rangle$ is available the current classification function is modified as less as possible to correctly classify the new example

- Passive Aggressive learning procedure (Crammer et al 2006):

```
Start with w_1 = 0 , k=0
For all t=1…
     Receive a new sample x_t
     Compute y = sign(w_t · x_t)
     Receive a feedback y_t
     Measure a classification loss (divergence between y_t and y)
     Modify the model to get zero loss, preserving what was
     learned from previous examples
```

Online Linear Passive Aggressive (2/3)

- Loss measure:

  Hinge loss: $l(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)) = \max(0; 1 - y_t(\boldsymbol{w} \cdot \boldsymbol{x}_t))$

- Model variation:

  $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2$

- Passive Aggressive Optimization Problem:

  $\boldsymbol{w}_{t+1} = argmin_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2$ such that $l(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)) = 0$

- Closed form solution:

  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$ where $\tau_t = \frac{l(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))}{\|\boldsymbol{x}_t\|^2}$

Online Linear Passive Aggressive (3/3)

- The previous formulation is a hard margin version that has a problem:
  - a single outlier could produce a high hyperplane shifting, making the model forget the previous learning
- Soft version solution:
  - control the algorithm aggressiveness through a parameter C

- PA-I formulation:

$$\boldsymbol{w}_{t+1} = argmin_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + C\xi \;\text{ s.t. } l\big(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)\big) \leq \xi \text{ with } \xi \geq 0$$

$$\Rightarrow \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t \;\text{ where } \tau_t = \min\left\{C; \frac{l(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))}{\|\boldsymbol{x}_t\|^2}\right\}$$

- PA-II model:

$$\boldsymbol{w}_{t+1} = argmin_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + C\xi^2 \text{ s.t. } l\big(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)\big) \leq \xi \text{ with } \xi \geq 0$$

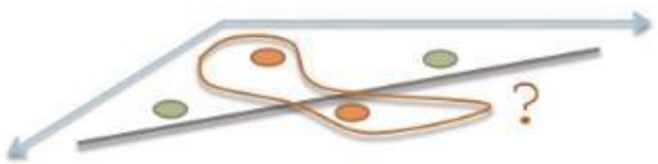
$$\Rightarrow \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t \;\text{ where } \tau_t = \frac{l(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))}{\|\boldsymbol{x}_t\|^2 + \frac{1}{2}C}$$

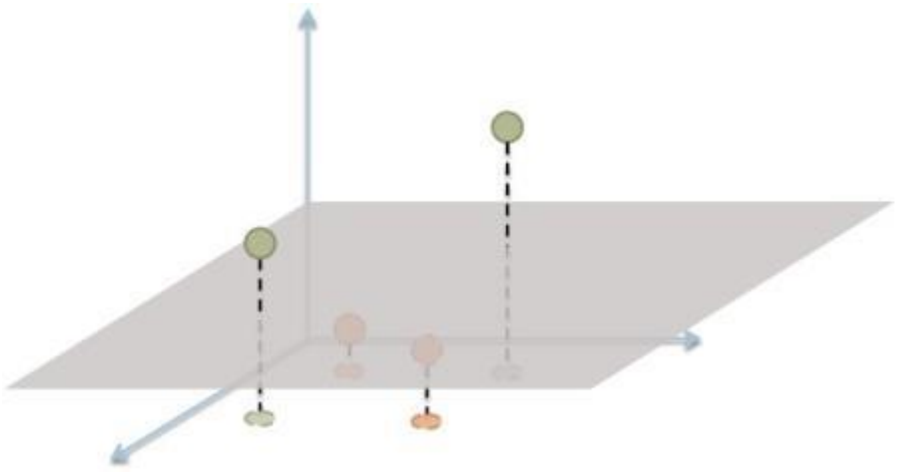
Overview

- Linear Online Learning Algorithms
- **Kernelized Online Learning Algorithms**
- Online Learning on a Budget

Data Separability

- Training data could not be separable
- Possible solutions:
  - Use a more complex classification function → Risk of overfitting!
  - Define a new set of feature that makes the problem linearly separable

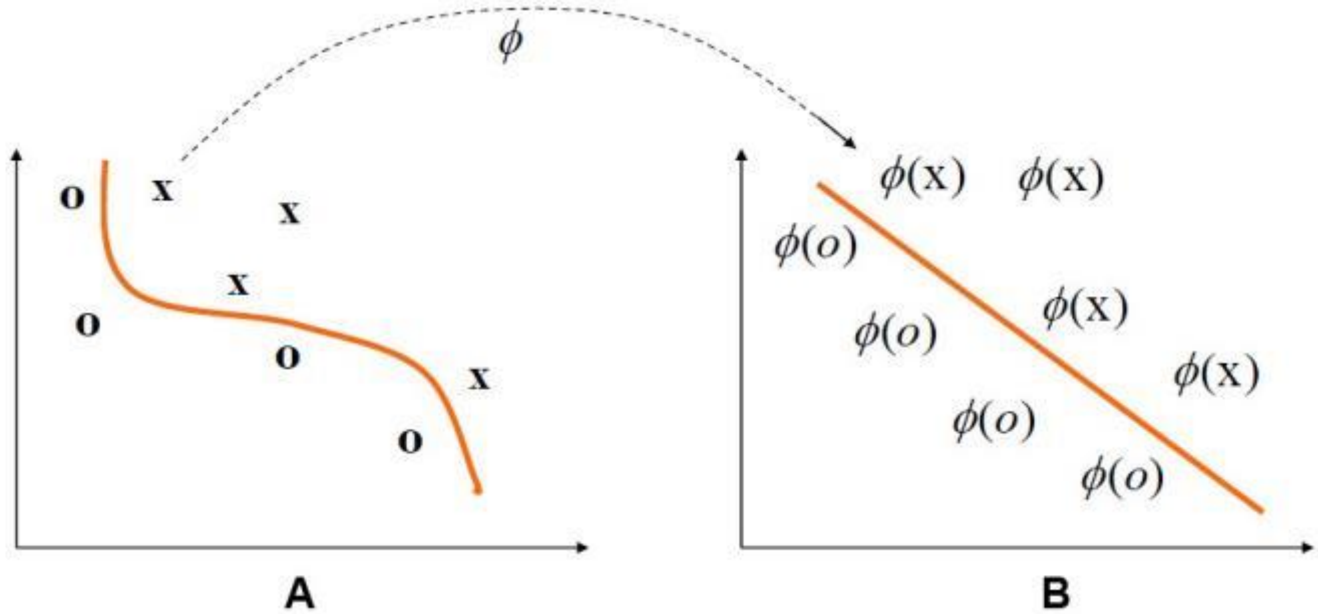




  - Project the current examples in a space in which they are separable…

Kernel Methods

- Training data can be projected in a space in which they are more easily separable



- Kernel Trick: any kernel function K performs the dot product in the kernel space without explicitly project the input vectors in that space
- Structured data (tree, graph, high order tensor…) can be exploited

Kernelized Passive Aggressive

- In kernelized Online Learning algorithms a new support vector is added every time a misclassification occurs

LINEAR VERSION	KERNELIZED VERSION
Classification function	
$f_t(\boldsymbol{x}) = \boldsymbol{w}_t{}^T \boldsymbol{x}$	$f_t(x) = \sum_{i \in S} \alpha_i k(x, x_i)$
Optimization Problem (PA-I)	
$\boldsymbol{w}_{t+1} = argmin_{\boldsymbol{w}} \frac{1}{2} \lVert \boldsymbol{w} - \boldsymbol{w}_t \rVert^2 + C\xi$ Such that $1 - y_t \mathrm{f}_t(\boldsymbol{x}_t) \leq \xi, \xi \geq 0$	$f_{t+1}(x) = \mathrm{argmin}_{\mathrm{f}} \frac{1}{2} \lVert f(x) - f_t(x) \rVert^2{}_{\mathcal{H}} + C\xi$ Such that $1 - y_t f_t(x_t) \leq \xi, \xi \geq 0$
Closed form solution	
$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$ where $\tau_t = \min\left\{C; \frac{\max(0, 1 - y_t f_t(\boldsymbol{x}_t))}{\lVert \boldsymbol{x}_t \rVert^2}\right\}$	$f_{t+1}(x) = \mathrm{f}_t(x) + \alpha_t k(x, x_t)$ where $\alpha_t = y_t \cdot \min\left\{C; \frac{\max(0, 1 - y_t f_t(x_t))}{\lVert x_t \rVert^2{}_{\mathcal{H}}}\right\}$

Linear Vs Kernel Based Learning

LINEAR VERSION	KERNELIZED VERSION
Classification function	
explicit hyperlplane in the original space ☹ Only linear functions can be learnt	implicit hyperplane in the RKHS ☺ Non linear functions can be learnt
Example form	
☹ Only feature vectors can be exploited	☺ Structured representations can be exploited
Computational complexity	
☺ A classification is a single dot product	☹ A classification involves $\|S\|$ kernel computations
Memory usage	
☺ Only a the explicit hyperplane must be stored	☹ All the support vectors and their weights must be stored

Overview

- Linear Online Learning Algorithms
- Kernelized Online Learning Algorithms
- **Online Learning on a Budget**

Learning on a Budget

- In kernelized online learning algorithm the set of support vectors can grow without limits
- Possible solution: Limit the number of support vector, defining a budget *B*
- This solution has the following advantages:
  - The memory occupation is upperbounded by *B* support vectors
  - Each classification needs at most *B* kernel computations
  - In shifting concept tasks, budget algorithms can outperform non-budget counterparts because they are faster in adapting

Limit the number of Support Vectors

- In order to respect the budget B, different policies can be formulated:
  - Stop learning when budget is exceeded: *Stoptron*
  - Delete a random support vector: *Randomized Perceptron*
  - Delete the more redundant support vector: *Fixed Budget Conscious Perceptron*
  - Delete the oldest support vector: *Least recent Budget Perceptron* and *Forgetron*
  - Modify the Support Vectors weights in order to adapt the classification hypothesis to the new sample: *Projectron*
  - *Online Passive-Aggressive on a Budget*

Stoptron

- Baseline of the online learning on a budget algorithms: Fix a budget B and stop learning when the number of support vectors is equal to B
- Stoptron algorithm (Orabona et al 2008):

```
Start with S = ∅
For all t=1…
      Receive a new sample x_t
      Compute  y = Σ_{i∈S} α_i y_i K(x_i, x_t)
      Receive a feedback y_t
      if  y y_t < β  and |S| < B  then
               S = S ∪ {t}
               α_t = 1
      endif
endfor
```

Randomized Perceptron

- <u>Simplest deleting policy:</u> when the budget B is exceeded remove a random support vector
- Randomized Perceptron algorithm (Cavallanti et al 2007):

```
Start with S = ∅
For all t=1…
    Receive a new sample x_t
    Compute   y = ∑_{i∈S} α_i y_i K(x_i, x_t)
    Receive a feedback y_t
    if  y y_t < β
        if |S| = B
            select randomly s ∈ S, S = S \ {s}
        endif
        S = S ∪ {t}  α_t = 1
    endif
endfor
```

Forgetron

- Deleting policy: Every time a new support vector is added, the weights of the others are reduced. Thus SVs lose weight with aging and removing the older SV should assure a minimum impact to the classification function.
- Forgetron algorithm (Dekel et al 2008):

Start with $S = \emptyset$

For all t=1…

    Receive a new sample $\boldsymbol{x}_t$

    Compute $y = \sum_{i \in S} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_t)$

    Receive a feedback y_t

    if $y y_t < \beta$

        if $|S| = B$

            S=$S \backslash min\{S\}$ //the oldest Support vector is removed

        endif

        $S = S \cup \{t\}$ $\alpha_t = 1$, $\alpha_i = \phi_t \alpha_i$ $\forall i \in S \setminus \{t\}$ //adding a new Sv and shrinking

    endif

endfor

Passive Aggressive Algorithms on a Budget (1/2)

- When $|S| = B$, to respect the budget B, the PA optimization problem is modified as follows (Wang et al 2010):

$$f_{t+1}(x) = \text{argmin}_{\text{f}} \frac{1}{2} \|f(x) - f_t(x)\|^2{}_{\mathcal{H}} + C\xi$$

Such that: $1 - y_t f_t(x_t) \le \xi, \xi \ge 0$ (old constraints)

$$f = f_t - \underbrace{\alpha_r k(x_r, \cdot)}_{SV\ elimination} + \underbrace{\textstyle\sum_{i \in V} \beta_i k(x_i, \cdot)}_{weight\ modification}$$ (new constraint)

Where V is the set of the indices of support vectors whose weights can be modified and r is the support vector to be removed.

Passive Aggressive Algorithms on a Budget (2/2)

- Given a *r* to be deleted, the optimization problem can be solved and the optimal weight modifications β_i for a given *r* can be computed
- A brute force approach is performed in order to chose r^* (the best *r* is the one that minimizes the objective function) and the corresponding ${\beta_i}^*$
  - *B* optimization problems must be solved every time a new SV must be added (when the budget is reached)
  - The computational complexity of a single optimization problem depends on $|V|$ (i.e. the number of SV whose weights can be modified)
  - Three proposal for V:
    - BPA-simple: $V=\{t\}$
    - BPA-projecting: $V=S\cup\{t\}\backslash\{r\}$
    - BPA-Nearest-Neighbor: $V=\{t\}\cup NN\{r\}$

Online Learning Algorithm Comparison

- DATASET USED:
  - Adult: determine whether a person makes over 50K a year using census attributes (2 classes, 21K samples, 123 features)
  - Banana: An artificial data set where instances belongs to several clusters with a banana shape (2 classes, 4.3K samples, 2 feature)
  - Checkerboard: An artificial dataset where instances of two classes are distributed like a checkerboard (2 classes, 10K samples, 2 features)
  - NCheckerboard: noisy version of checkerboard dataset (15% of the samples are bad classified)
  - Covertype Data Set: Predicting forest cover type from cartographic variables only (Elevation, Distance to hydrology…) (7 classes, 10K samples, 41 features)
  - Phoneme: phoneme recognition (11 classes, 10K samples, 41 features)
  - USPD: optical character recognition dataset. (10 classes, 7.3 K samples, 256 features)

Online Learning Algorithm Comparison

- DATASET USED:
  - Adult: determine whether a person makes over 50K a year using census attributes (2 classes, 21K samples, 123 features)
  - Banana: An artificial data set where instances belongs to several clusters with a banana shape (2 classes, 4.3K samples, 2 feature)
  - Checkerboard: An artificial dataset where instances of two classes are distributed like a checkerboard (2 classes, 10K samples, 2 features)
  - NCheckerboard: noisy version of checkerboard dataset (15% of the samples are bad classified)
  - Covertype Data Set: Predicting forest cover type from cartographic variables only (Elevation, Distance to hydrology…) (7 classes, 10K samples, 41 features)
  - Phoneme: phoneme recognition (11 classes, 10K samples, 41 features)
  - USPD: optical character recognition dataset. (10 classes, 7.3 K samples, 256 features)

Results using a RBF kernel

Time	Algs	Adult 21K×123 75%	Banana 4.3K×2 55%	Checkerb 10K×2 50%	NCheckerb 10K×2 50%	Cover 10K×54 51%	Phoneme 10K×41 50%	USPS 7.3K×256 52%	Avg
	Memory-unbounded online algorithms								
$O(N)$	Pcptrn	80.2±0.2	87.4±1.5	96.3±0.6	83.4±0.7	76.0±0.4	78.9±0.6	94.6±0.1	85.3
	(#SV)	(4.5K)	(0.6K)	(0.5K)	(2.8K)	(2.8K)	(2.4K)	(0.4K)	
	PA	83.6±0.2	89.1±0.7	97.2±0.1	95.8±1.0	81.6±0.2	82.6±0.9	96.7±0.1	89.5
	(#SV)	(15K)	(2K)	(2.6K)	(5.9K)	(9.9K)	(7.2K)	(4.5K)	
	PA^R	**84.1**±0.1	**89.3**±0.7	**97.5**±0.1	**96.2**±0.8	**82.7**±0.3	**83.7**±0.7	**96.7**±0.1	90.0
	(#SV)	(4.4K)	(1.5K)	(2.6K)	(3.3K)	(9.8K)	(6.5K)	(4.5K)	
	Budgeted online algorithms (B=100)								
$O(B)$	Stptrn	76.5±2.0	86.7±2.1	87.3±0.9	75.4±4.3	64.2±1.7	67.6±2.7	89.1±1.2	78.1
	Rand	76.2±3.6	84.1±2.6	85.6±1.2	69.4±2.9	61.3±3.2	65.0±4.4	87.1±0.9	75.5
	Fogtrn	72.8±6.1	82.8±2.4	86.1±1.0	68.2±3.5	60.8±2.7	65.6±1.2	86.2±2.1	74.6
	PA+Rnd	78.4±1.9	84.9±2.1	83.3±1.4	75.1±3.6	63.1±1.5	64.0±3.9	86.2±1.1	76.4
	BPA-S	82.4±0.1	89.4±1.3	90.0±0.8	87.4±0.7	68.6±1.9	67.4±3.0	89.6±1.3	82.1
	BPA^R-S	82.4±0.1	89.5±1.7	90.0±1.0	88.2±1.2	69.3±1.8	67.0±3.2	89.3±1.2	82.2
	BPA-NN	82.8±0.4	89.6±1.4	94.0±1.2	90.2±1.3	69.1±1.8	74.3±0.7	90.8±0.9	84.4
	BPA^R-NN	**83.1**±0.0	89.8±1.1	94.2±0.9	**92.3**±0.5	70.3±0.8	74.6±0.8	90.8±0.6	85.0
$O(B^2)$	Pjtrn++	80.1±0.1	89.5±1.1	**95.4**±0.7	88.1±0.7	68.7±1.0	74.6±0.7	89.2±0.7	83.7
$O(B^3)$	BPA-P	83.0±0.2	**89.6**±1.1	**95.4**±0.7	91.7±0.8	**74.3**±1.4	**75.2**±1.0	**92.8**±0.7	86.0
	BPA-P^R	**84.0**±0.0	**89.6**±0.8	95.2±0.8	**94.1**±0.9	**75.0**±1.0	**74.9**±0.6	**92.6**±0.7	86.5
	Budgeted online algorithms (B=200)								
$O(B)$	Stptrn	78.7±1.8	85.6±1.5	92.8±1.1	76.0±3.1	65.5±2.3	70.5±2.6	92.3±0.7	80.2
	Rand	76.4±2.8	83.6±2.0	90.3±1.3	74.5±2.1	62.4±2.4	67.3±2.5	89.8 ±1.1	77.8
	Fogtrn	72.9±6.8	85.0±1.3	90.9±1.7	72.2±4.4	62.1±2.8	68.0±2.3	90.3±0.9	77.3
	PA+Rnd	80.1±2.4	86.7±1.9	87.0±1.3	78.3±1.8	64.2±2.7	68.7±4.3	88.8±0.8	79.1
	BPA-S	82.7±0.2	89.5±0.7	93.4±0.5	89.7±0.9	71.7±1.7	71.3±2.3	92.6±0.9	84.4
	BPA^R-S	83.1±0.1	89.5±0.9	93.9±0.6	90.8±0.8	71.7±1.2	71.6±2.2	92.1±0.6	84.7
	BPA-NN	83.1±0.4	89.6±1.1	95.5±0.4	91.7±1.3	72.7±1.0	75.8±1.0	92.8±0.6	85.9
	BPA^R-NN	83.3±0.4	89.5±1.4	95.2±0.5	**93.3**±0.6	72.7±1.4	77.2±1.7	94.0±0.4	86.5
$O(B^2)$	Pjtrn++	82.9±0.1	89.5±1.2	**95.8**±0.5	92.5±1.0	75.1±2.0	75.2±0.6	93.2±0.6	86.3
$O(B^3)$	BPA-P	**83.8**±0.0	**89.7**±0.7	**95.9**±0.6	92.8±0.7	**76.0**±1.3	**78.0**±0.3	**94.8**±0.3	87.3
	BPA^R-P	**84.6**±0.0	**90.3**±1.5	95.6±1.2	**94.5**±1.1	**76.3**±1.0	**77.6**±0.6	**94.8**±0.3	87.7

Summary

- Online learning methods can:
  - Incrementally learn from new samples
  - Dinamically adapt to problem variations
  - Reduce the computational cost of building a new model
- Online learning methods can be used with kernels but they suffer from the "*curse of kernelization*":
  - The number of support vectors can grow without bounds
- Several number of budgeted solutions have been proposed