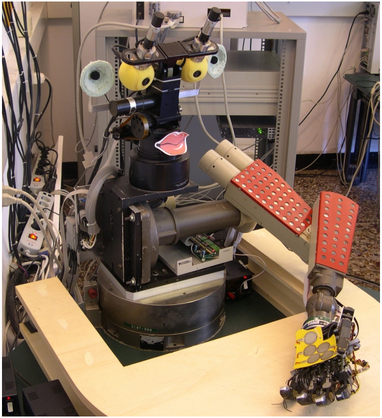
Simple and efficient online algorithms for real world applications

Francesco Orabona

Università degli Studi di Milano
Milano, Italy

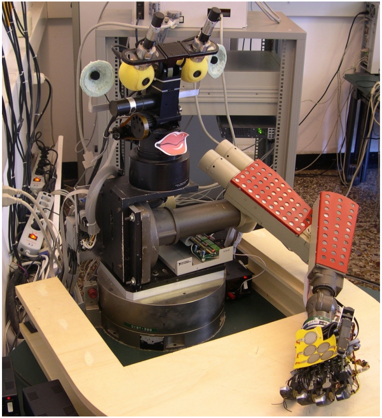
Talk @ Centro de Visión por Computador

Something about me



- PhD in Robotics at LIRA-Lab, University of Genova
- PostDoc in Machine Learning

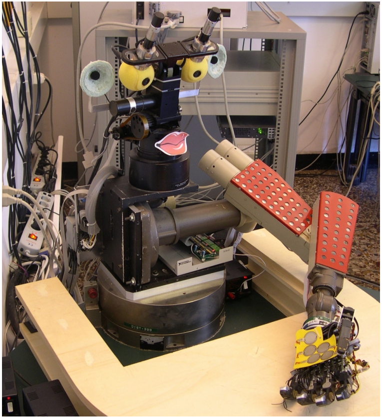
Something about me



- PhD in Robotics at LIRA-Lab, University of Genova
- PostDoc in Machine Learning

I like theoretical motivated algorithms

Something about me



- PhD in Robotics at LIRA-Lab, University of Genova
- PostDoc in Machine Learning

I like theoretical motivated algorithms
But they must work well too! ;-)

Outline

Outline

1. Online Learning
   - Motivation
   - The Perceptron
   - Beyond the Perceptron
2. Algorithms
   - OM-2
   - Projectron
   - BBQ

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier
- Learning with limited feedback: selecting publicity banners

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier
- Learning with limited feedback: selecting publicity banners
- Active learning: asking for specific samples to label

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier
- Learning with limited feedback: selecting publicity banners
- Active learning: asking for specific samples to label
- Interactive learning: the agent and the human are learning at the same time

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier
- Learning with limited feedback: selecting publicity banners
- Active learning: asking for specific samples to label
- Interactive learning: the agent and the human are learning at the same time

These problems cannot be solved with standard batch learning!

What do they have in common?

- Spam filtering: minimize the number of wrongly classified mails, “while training”
- Shifting distributions: the Learner must “track” the best classifier
- Learning with limited feedback: selecting publicity banners
- Active learning: asking for specific samples to label
- Interactive learning: the agent and the human are learning at the same time

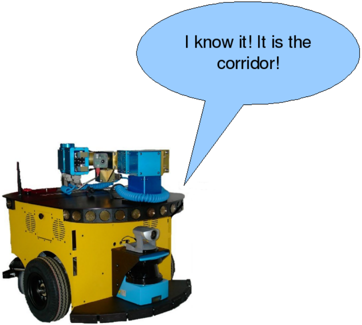
These problems cannot be solved with standard batch learning!
But they can in the online learning framework!

Learning in an artificial agent

- We have the Learner and the Teacher.
- The Learner observes examples in a sequence of rounds, and constructs the classification function incrementally.

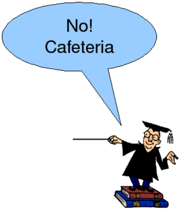
Learning in an artificial agent

- Given an input, the Learner predicts its label.



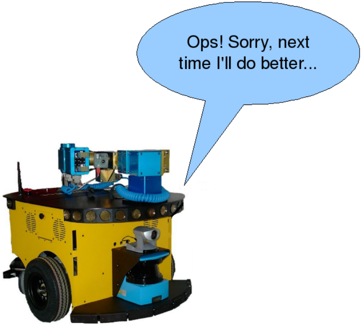
Learning in an artificial agent

- Then the Teacher reveals the true label.



Learning in an artificial agent

- The Learner compares its prediction with the true label and update its knowledge. The aim of the learner is to minimize the number of mistakes.



Machine learning point of view on online learning

- The Teacher is a black box
  - It can chose the examples arbitrarily - in the worst case the choice can be adversarial!
  - There is no IID assumption on the data!
  - Useful to model interaction between the user and the algorithm or non-stationary data
- Performance is measured “while training”: no separate testing set
- Update after each sample: efficiency of the update is important

See [Cesa-Bianchi & Lugosi, 2006] for a full introduction to the theory of online learning.

Online Learning - Formal setting

- Learning goes on in a sequence of T rounds
- Instances: $\mathbf{x}_t \in \mathcal{X}$
  - Images, sounds, etc.
- Labels: $y_t \in \mathcal{Y}$
  - Labels, numbers, structured output, etc.
- Prediction rule, $f_t(\mathbf{x}) = \hat{y}$
  - for binary classification $\hat{y} = sign(\langle \mathbf{w}, \mathbf{x} \rangle)$
- Loss, $\ell(\hat{y}, y) \in \mathbb{R}^+$

Regret bounds

The learner must minimize its loss on the T observed samples

$$\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$$

Regret bounds

The learner must minimize its loss on the T observed samples, compared to the loss of the best fixed predictor

$$\sum_{t=1}^{T} \ell(\hat{y}_t, y_t) \leq min_f \sum_{t=1}^{T} \ell'(f(\boldsymbol{x}_t), y_t) + \mathcal{R}_T$$

Regret bounds

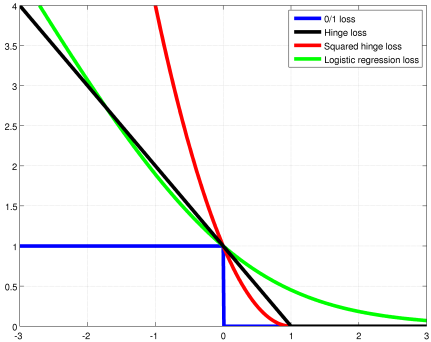
The learner must minimize its loss on the T observed samples, compared to the loss of the best fixed predictor

$$\sum_{t=1}^{T} \ell(\hat{y}_t, y_t) \leq min_f \sum_{t=1}^{T} \ell'(f(\boldsymbol{x}_t), y_t) + \mathcal{R}_T$$

We want the regret, $\mathcal{R}_T$, to be small: a small regret indicates that the performance of the learner is not too far from the one of the best fixed classifier

A loss for everyone

- Mistake, $\ell_{01}(\mathbf{w},\mathbf{x},y) := \mathbf{1}(y \neq sign(\langle \mathbf{w},\mathbf{x}\rangle))$
- Hinge, $\ell_{hinge}(\mathbf{w},\mathbf{x},y) := \max(0, 1 - y\langle \mathbf{w},\mathbf{x}\rangle)$
- Logistic regression, $\ell_{logreg}(\mathbf{w},\mathbf{x},y) := \log(1 + \exp(-y\langle \mathbf{w},\mathbf{x}\rangle))$
- exponential loss, etc.



Let's start from the Perceptron

for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x}_t \rangle)$
  Receive label y_t
  if $y_t \neq \hat{y}_t$ **then**
    $\mathbf{w} = \mathbf{w} + y_t \mathbf{x}_t$
  end if
end for

The mistake bound of the Perceptron

Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be any sequence of instance-label pairs, $y_t \in \{-1, +1\}$, and $\|\mathbf{x}_t\| \le R$.
The number of mistakes of the Perceptron is bounded by

$$\min_{\boldsymbol{u}} L + \underbrace{\|\boldsymbol{u}\|^2 R^2 + \|\boldsymbol{u}\| R \sqrt{L}}_{\mathcal{R}_T}$$

where $L = \sum_i^T \ell_{hinge}(\mathbf{u}, \mathbf{x}_i, y_i)$

The mistake bound of the Perceptron

Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be any sequence of instance-label pairs, $y_t \in \{-1, +1\}$, and $\|\mathbf{x}_t\| \le R$.
The number of mistakes of the Perceptron is bounded by

$$\min_{\boldsymbol{u}} L + \underbrace{\|\boldsymbol{u}\|^2 R^2 + \|\boldsymbol{u}\| R \sqrt{L}}_{\mathcal{R}_T}$$

where $L = \sum_i^T \ell_{hinge}(\mathbf{u}, \mathbf{x}_i, y_i)$

If the problem is linearly separable the maximum number of mistakes is $R^2 \|\boldsymbol{u}\|^2$, regardless of the ordering of the samples!

The mistake bound of the Perceptron

Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be any sequence of instance-label pairs, $y_t \in \{-1, +1\}$, and $\|\mathbf{x}_t\| \leq R$.
The number of mistakes of the Perceptron is bounded by

$$\min_{\boldsymbol{u}} L + \underbrace{\|\boldsymbol{u}\|^2 R^2 + \|\boldsymbol{u}\| R\sqrt{L}}_{\mathcal{R}_T}$$

where $L = \sum_i^T \ell_{hinge}(\mathbf{u}, \mathbf{x}_i, y_i)$

If the problem is linearly separable the maximum number of mistakes is $R^2\|\boldsymbol{u}\|^2$, regardless of the ordering of the samples!

Video

Pros and Cons of the Perceptron

Pros :-)

- Very efficient! It can be easily trained with huge datasets
- Theoretical guarantee on the maximum number of mistakes

Pros and Cons of the Perceptron

Pros :-)

- Very efficient! It can be easily trained with huge datasets
- Theoretical guarantee on the maximum number of mistakes

Cons :-(

- Linear hyperplane only
- Binary classification only

Pros and Cons of the Perceptron

Pros :-)

- Very efficient! It can be easily trained with huge datasets
- Theoretical guarantee on the maximum number of mistakes

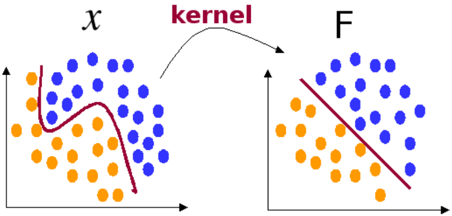
Cons :-(

- Linear hyperplane only
- Binary classification only

Let's generalize the Perceptron!

Non-linear classifiers using Kernels!

- Suppose to transform the inputs through a non-linear transform $\phi(\mathbf{x})$, to the *feature space*
- A linear classifier in the feature space will result in a non-linear classifier in the input space
- We can do even better: the algorithms just need to access to $\langle\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)\rangle$, so if we have such a function, $K(\mathbf{x}_1, \mathbf{x}_2)$, we do not need $\phi()$!



Kernel Perceptron

for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}\left(\sum_{\mathbf{x}_i \in \mathcal{S}} y_i K(\mathbf{x}_i, \mathbf{x}_t)\right)$
  Receive label y_t
  if $y_t \neq \hat{y}_t$ **then**
    Add $\mathbf{x}_t$ to the support set $\mathcal{S}$
  end if
end for

Follow The Regularized Leader

- The Perceptron algorithm is just a particular case of a more general algorithm: the “follow the regularized leader” (FTRL) algorithm
- In FTRL, at each time step we predict with the approximate batch solution using a linearization of the loss, instead of the true loss functions
- Regret bounds will come almost for free!

See [Kakade et al., 2009] for FTRL and [Orabona&Crammer, 2010] for an even more general algorithm

The general algorithm

for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\boldsymbol{x}_t$
  Predict $\hat{y}_t$
  Receive label y_t
  $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta_t \partial \ell(\boldsymbol{w}, \boldsymbol{x}_t, y_t)$
  $\boldsymbol{w} = \nabla g_t^*(\boldsymbol{\theta})$
end for

F. Orabona, and K. Crammer. New Adaptive Algorithms for Online Classification. Accepted in Neural Information Processing Systems (NIPS) 2010

The general algorithm

for $t = 1, 2, \ldots, T$ **do**
 Receive new instance $\boldsymbol{x}_t$
 Predict $\hat{y}_t$
 Receive label y_t
 $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta_t \partial \ell(\boldsymbol{w}, \boldsymbol{x}_t, y_t)$
 $\boldsymbol{w} = \nabla g_t^*(\boldsymbol{\theta})$
end for

- Suppose $g_t(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2 \Rightarrow \nabla g_t^*(\boldsymbol{\theta}) = \boldsymbol{\theta}$
- Let's use the hinge loss $\Rightarrow \partial \ell(\boldsymbol{w}, \boldsymbol{x}_t, y_t) = -y_t \boldsymbol{x}_t$
- $\eta_t = 1$ on mistakes, 0 otherwise
- We recovered the Perceptron algorithm!

F. Orabona, and K. Crammer. New Adaptive Algorithms for Online Classification. Accepted in Neural Information Processing Systems (NIPS) 2010

The Online Learning Recipe

Create the online algorithm that best suits your needs!

- Define a task $\Rightarrow$ this will define a (convex) loss function.
- Define which charateristic you would like your solution to have $\Rightarrow$ this will define a (strongly convex) regularizer.
- Compute the gradient of the loss.
- Compute the gradient of the fenchel dual of the regularizer.
- Just code it!

Some examples

- Multiclass-multilabel classification, M classes, M different classifiers $\boldsymbol{w}_i$.
  $\ell(\bar{\boldsymbol{w}}, \mathbf{x}, \mathcal{Y}) = \max(1 + \max_{y' \notin \mathcal{Y}} \langle \boldsymbol{w}_{y'}, \boldsymbol{x} \rangle - \min_{y \in \mathcal{Y}} \langle \boldsymbol{w}_y, \boldsymbol{x} \rangle, 0)$
- Do you prefer sparse classifiers?
  Use the regularizer $\|\boldsymbol{w}\|_p^2$ with $1 < p \leq 2$.
- K different kernels?
  Use the regularizer $\left\|[\|\boldsymbol{w}_1\|_2, \ldots, \|\boldsymbol{w}_K\|_2]\right\|_p^2$ with $1 < p \leq 2$.

Batch Solutions with Online Algorithms

- What most of the persons do when they want a batch solution: they stop the online algorithm and use the last solution found.
- This is wrong: the last solution can be arbitrarly bad!
- If the data are IID you can simply use the averaged solution [Cesa-Bianchi et al., 2004].
- Alternative way: several epochs of training, until convergence.

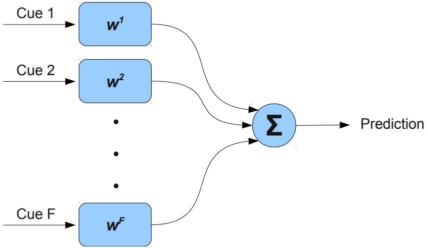
Batch Solutions with Online Algorithms

- What most of the persons do when they want a batch solution: they stop the online algorithm and use the last solution found.
- This is wrong: the last solution can be arbitrarily bad!
- If the data are IID you can simply use the averaged solution [Cesa-Bianchi et al., 2004].
- Alternative way: several epochs of training, until convergence.

Video

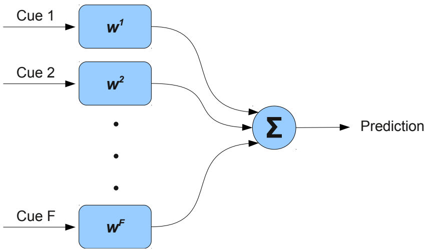
Outline

Multi Kernel Learning online?



We have *F* different features, e.g. color, shape, etc.

Multi Kernel Learning online?



We have *F* different features, e.g. color, shape, etc.
Solution: Online Multi-Class Multi-Kernel learning algorithm

L. Jie, F. Orabona, M. Fornoni, B. Caputo, and N. Cesa-Bianchi. OM-2: An Online Multi-class Multi-kernel Learning Algorithm. In Proc. of the 4th IEEE Online Learning for Computer Vision Workshop (in CVPR10)

OM-2: Pseudocode

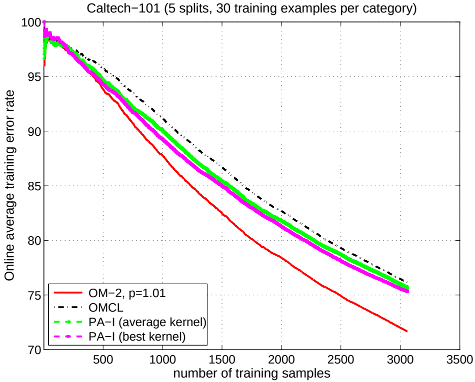
Input: q
Initialize: $\bar{\boldsymbol{\theta}}_1 = \mathbf{0}, \bar{\boldsymbol{w}}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\boldsymbol{x}_t$
  Predict $\hat{y}_t = \underset{y=1,\ldots,M}{\text{argmax}}\ \bar{\boldsymbol{w}}_t \cdot \bar{\phi}(\boldsymbol{x}_t, y)$
  Receive label y_t
  $\bar{\boldsymbol{z}}_t = \bar{\phi}(\boldsymbol{x}_t, y_t) - \bar{\phi}(\boldsymbol{x}_t, \hat{y}_t)$
  if $\ell(\bar{\boldsymbol{w}}_t, \boldsymbol{x}_t, y_t) > 0$ **then**
  $\eta_t = \min\left\{1 - \frac{2\bar{\boldsymbol{w}}_t \cdot \bar{\boldsymbol{z}}_t}{\|\bar{\boldsymbol{z}}_t\|_{2,q}^2}, 1\right\}$
  else $\eta_t = 0$
  $\bar{\boldsymbol{\theta}}_{t+1} = \bar{\boldsymbol{\theta}}_t + \eta_t \bar{\boldsymbol{z}}_t$
  $\boldsymbol{w}^j_{t+1} = \frac{1}{q}\left(\frac{\|\boldsymbol{\theta}^j_{t+1}\|_2}{\|\bar{\boldsymbol{\theta}}_{t+1}\|_{2,q}}\right)^{q-2} \boldsymbol{\theta}^j_{t+1}, \quad \forall j = 1, \cdots, F$
end for

Code available!

Experiments

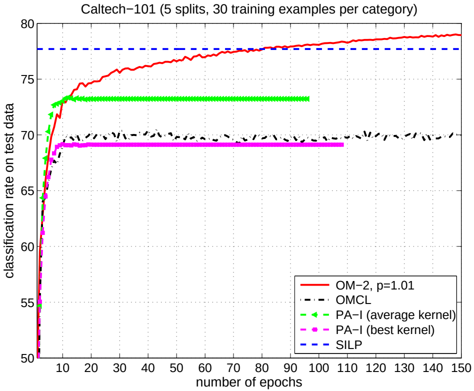
- We compared OM-2 to OMCL [Jie et al. ACCV09], and to PA-I [Crammer et al. JMLR06] using the best feature and the sum of the kernels.
- We also used SILP [Sonnenburg et al. JMLR06], a state-of-the-art MKL batch solver.
- We used the Caltech-101 with 39 different kernels, as in [Gehler and Nowozin ICCV09].

Caltech-101: online performance



- Best results with $p = 1.01$.
- OM-2 achieves the best performance among the online algorithms.

Caltech-101: batch performance



- Matlab implementation of OM-2 takes 45 mins, SILP more than 2 hours.
- The performance advantage of OM-2 over SILP is due the fact that OM-2 is based on a native multiclass formulation.

Learning with bounded complexity

- Perceptron-like algorithms will never stop updating the solution if the problem is not linearly separable.

Learning with bounded complexity

- Perceptron-like algorithms will never stop updating the solution if the problem is not linearly separable.
- On the other hand, if we use kernels, sooner or later the memory of the computer will finish...

Learning with bounded complexity

- Perceptron-like algorithms will never stop updating the solution if the problem is not linearly separable.
- On the other hand, if we use kernels, sooner or later the memory of the computer will finish...
- Is it possible to bound the complexity of the learner?

Learning with bounded complexity

- Perceptron-like algorithms will never stop updating the solution if the problem is not linearly separable.
- On the other hand, if we use kernels, sooner or later the memory of the computer will finish...
- Is it possible to bound the complexity of the learner?
- Yes! Just update with a “noisy” version of the gradient
- If the new vector can be well approximated with the old ones, just update the coefficients of the old ones.

F. Orabona, J. Keshet, and B. Caputo. Bounded Kernel-Based Online Learning. Journal of Machine Learning Research, 2009

The Projectron Algorithm

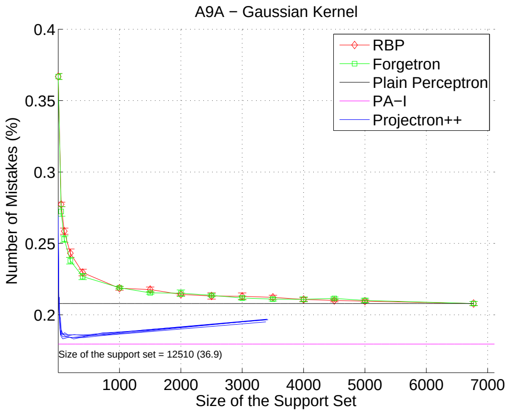
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x}_t \rangle)$
  Receive label y_t
  if $y_t \neq \hat{y}_t$ **then**
    $\mathbf{w}' = \mathbf{w} + y_t \mathbf{x}_t$
    $\mathbf{w}'' = \mathbf{w} + y_t P(\mathbf{x}_t)$
    if $\|\delta_t\| = \|\mathbf{w}'' - \mathbf{w}'\| \leq \eta$ **then**
      $\mathbf{w} = \mathbf{w}''$
    else
      $\mathbf{w} = \mathbf{w}'$
    end if
  end if
end for

- It is possible to calculate the projection even using Kernels.
- The algorithm has a mistake bound and a bounded memory growth.

Code available!

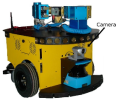
Average Online Error vs Budget Size

Adult9, 32561 samples, 123 features, Gaussian Kernel

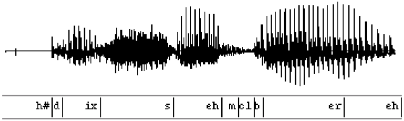


Robot Navigation & Phoneme Recognition

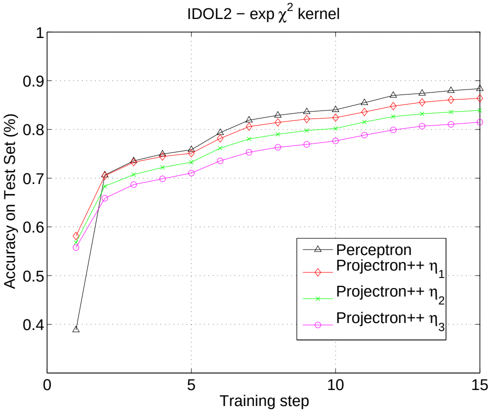
- Place Recognition
- IDOL2 database
- 5 rooms
- CRFH features



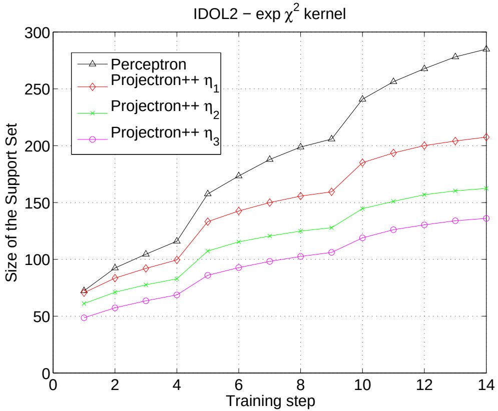
- Phoneme recognition
- Subset of the TIMIT corpus
- 55 phonemes
- MFCC $+ \Delta + \Delta\Delta$ features



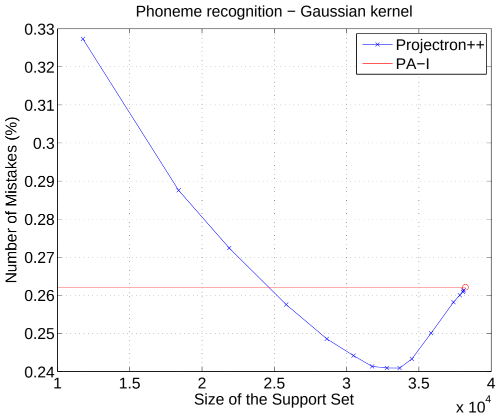
Place recognition



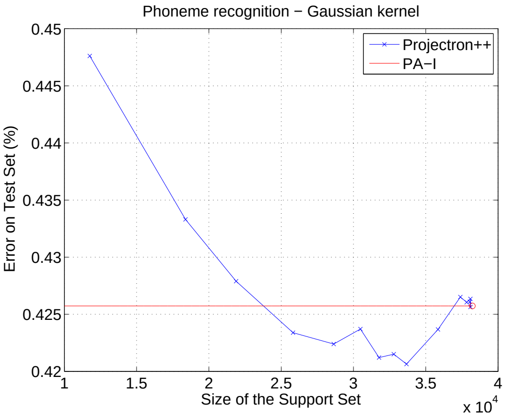
Place recognition



Phoneme recognition



Phoneme recognition



Semi-supervised online learning

- We are given N training samples and a regression problem, $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, $y_i \in [-1, 1]$
- Obtaining the output for a given sample can be expensive, so we want to *ask* for as few as possible labels.
- We assume the outputs y_t are realizations of random variables Y_t such that $\mathbb{E}\, Y_t = \boldsymbol{u}^\top \boldsymbol{x}_t$ for all t, where $\boldsymbol{u} \in \mathbb{R}^d$ is a fixed and unknown vector such that $\|\boldsymbol{u}\| = 1$.
  - The order of the data is still adversarial, but the outputs are generated by a stochastic source

Semi-supervised online learning

- We are given N training samples and a regression problem, $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, $y_i \in [-1, 1]$
- Obtaining the output for a given sample can be expensive, so we want to *ask* for as few as possible labels.
- We assume the outputs y_t are realizations of random variables Y_t such that $\mathbb{E}\, Y_t = \boldsymbol{u}^\top \boldsymbol{x}_t$ for all t, where $\boldsymbol{u} \in \mathbb{R}^d$ is a fixed and unknown vector such that $\|\boldsymbol{u}\| = 1$.
    - The order of the data is still adversarial, but the outputs are generated by a stochastic source

Solution: Bound on Bias Query algorithm (BBQ)

N. Cesa-Bianchi, C. Gentile and F. Orabona. Robust Bounds for Classification via Selective Sampling. In Proc. of the International Conference on Machine Learning (ICML), 2009

The Parametric BBQ Algorithm

Parameters: $0 < \varepsilon, \delta < 1$
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\boldsymbol{x}_t$
  Predict $\hat{y}_t = \boldsymbol{w}^\top \boldsymbol{x}_t$
  $r = \boldsymbol{x}_t^\top \left(I + S_{t-1} S_{t-1}^\top + \boldsymbol{x}_t \boldsymbol{x}_t^\top\right)^{-1} \boldsymbol{x}_t$
  $\boldsymbol{q} = S_{t-1}^\top \left(I + S_{t-1} S_{t-1}^\top + \boldsymbol{x}_t \boldsymbol{x}_t^\top\right)^{-1} \boldsymbol{x}_t$
  $s = \left\| \left(I + S_{t-1} S_{t-1}^\top + \boldsymbol{x}_t \boldsymbol{x}_t^\top\right)^{-1} \boldsymbol{x}_t \right\|$
  if $[\varepsilon - r - s]_+ < \|\boldsymbol{q}\| \sqrt{2 \ln \frac{t(t+1)}{2\delta}}$ **then**
    Query label y_t
    Update $\boldsymbol{w}$ with a Regularized Least Square
    $S_t = [S_t, \ \boldsymbol{x}_t]$
  end if
end for

- The algorithm follows the general framework
- Every time a query is not issued the predicted output is far from the correct one at most by ε

Regret Bound of Parametric BBQ

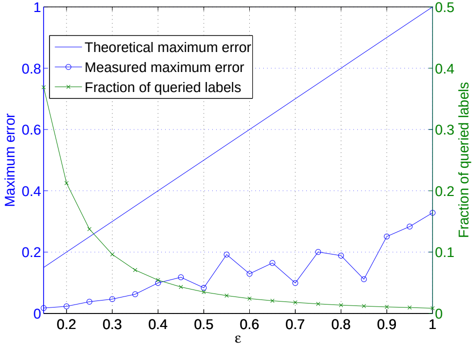
Theorem

If Parametric BBQ is run with input $\varepsilon, \delta \in (0, 1)$ then:

- *with probability at least $1 - \delta$, $\left|\widehat{\Delta}_t - \Delta_t\right| \leq \varepsilon$ holds on all time steps t when no query is issued;*
- *the number N_T of queries issued after any number T of steps is bounded as*

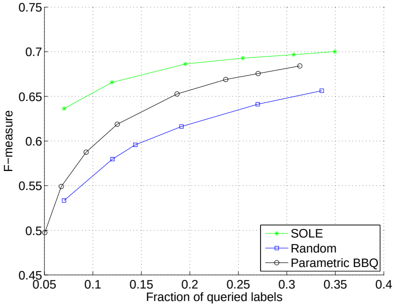
$$N_T = \mathcal{O}\left(\frac{d}{\varepsilon^2}\left(\ln\frac{T}{\delta}\right)\ln\frac{\ln(T/\delta)}{\varepsilon}\right) .$$

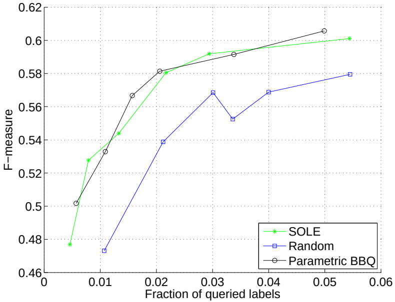
Synthetic Experiment



- We tested Parametric BBQ.
- 10,000 random examples on the unit circle in $\mathbb{R}^2$.
- The labels were generated according to our noise model using a randomly selected hyperplane $\boldsymbol{u}$ with unit norm.

Real World Experiments





F-measure and fraction of queried labels for different algorithms on Adult9 dataset (left)(Gaussian Kernel) and RCV1 (right)(linear kernel).

Summary

- Many real world problem cannot be solved using the standard batch framework
- The online learning framework offers a useful tool in these cases
- A general algorithm that covers many of the previous known online learning algorithms has been presented
- The framework allows to easily design algorithms for specific problems

Thanks for your attention

Code: `http://dogma.sourceforge.net`
My website: `http://francesco.orabona.com`