UNIVERSITÉ LAVAL

LABORATOIRE DE VISION ET SYSTÈMES NUMÉRIQUES

Online Learning Algorithms in Python 3

Luis Enrique Güitrón Leal

Professor
Ph.D. Christian Gagné

Supervisors
M.S. Changjian Shui
Ph.D. Ihsen Hedhli

Introduction

The development of this project took place in the Laboratory of Vision and Numerical Systems in University Laval in Quebec City Canada, and was made during an internship in the summer of 2018.

The project took place in a research environment, and it involved both a theoretical and practical approach for its completion. The theoretical phase involved comprehension of Online Learning algorithms by reading research papers with information on the subject, during this phase I was able to interact with my professor and with my supervisors in order to get a better understanding on the content of the papers.

For the practical part of the project I was assigned to program an implementation for the Online Learning algorithms that I had read about, the goal of this implementation was that it could be reliable, simple to use, and applicable on large scale data sets.

The library developed for my project is programmed in python 3, and is based on LIBOL (Library for Online Learning) [1], which is an implementation for Online Learning algorithms in MATLAB. The newly developed library is called LIBOL-python and is available on GitHub.

Background

Online learning is a sub field of machine learning in which models are learned incrementally by receiving data in a sequential manner. In this setting the training of the model is a continuous process as we do not know in advance the number of samples that will be fed to the model.
Our main objective when training this kind of classifiers is to minimize the cumulative error rate of our models, this is the total number of errors divided by the amount of samples that have been observed by our classifier. Most of the approaches used in the online learning setting can be categorized into first order, and second order methods.

First Order Methods: Methods that use the first order gradient information to minimize a loss function. The main advantage of this methods is cheap cost of computation.

Second Order Methods: Methods that take geometrical properties of the data into account. This methods are more robust than first order methods as they don’t suffer in performance from data transformations. These methods are also more computationally expensive than first order methods.

Objective

The main objective for LIBOL-python is to compare new Online Learning methods with already existing methods in a way that the time to integrate new code is minimal and the results displayed by the program are precise enough to be able to draw conclusions of the new methods.
In order to achieve this goal the library offers options to the user so that it is easy to:

- Add new algorithms
- Modify hyper-parameter values
- Select range of hyper-parameter testing (for automatic selection)
- Add new kernel methods
- Add new regularization methods
- Compare algorithms in terms of performance and computational cost

Algorithms

The algorithms included in this library are focused in solving both binary classification problems, and multi-class classification problems.
This library has a total of 24 algorithms that can be grouped into 5 different categories.
Algorithms from the same category are similar between each other, and differ mainly in the update rule, and the loss function.
The way in which these algorithms are considered to be part of a specific category is shown in the following table:

Perceptron	Passive-Aggressive	Online Gradient Descent	Confidence-weighted Learning	Adaptive Regularization of Weight Vectors
Perceptron: First order, binary classification	PA / PA1 / PA2: First Order, binary classification	OGD: First order, binary classification	CW/ SCW/SCW2: Second order, binary classification	AROW / NAROW: Second order, binary classification
Gaussian Kernel Perceptron: First order, binary classification	M_PA / M_PA1 / M_PA2: First Order, multi-class classification	Gaussian Kernel OGD: First order, binary classification	M_CW / M_SCW / M_SCW2: Second order, multi-class classification	M_AROW: Second order, multi-class classification
Second Order Perceptron (SOP): Second order, binary classification		M_OGD: Fisrt order, multi-class classification		
M_PerceptronM / M_PerceptronS / M_PerceptronU: First order, multi-class classification				

Perceptron [2]: First order algorithm in which weights are updated whenever a sample is incorrectly classified during the training session, these updates modify the weights in a way that the sample seen is more likely to be classified correctly by the new classifier.
A second order version of the perceptron algorithm keeps track of a matrix which approximates correlations between seen samples, and transforms the new samples with this matrix in order to make more accurate classifications.

Passive-Aggressive [3]: First order family of algorithms that attempts to modify the weights of the classifier in a way in which there is a trade-off between keeping the new classifier similar to the classifier in the previous step (Passiveness), and having a low loss value for the newest received data sample (Aggressiveness).
The original formulation of the passive aggressive algorithm forces new samples to be classified correctly and with some minimum distance away from the decision boundary, other versions of this

algorithms relax this constraint and introduce a hyper-parameter that represents trade-off between passiveness and aggressiveness.

Online Gradient Descent[4]: Receive a classification loss according to a loss function (0-1 loss, hinge loss, logarithmic loss, square loss), and attempt to minimize the loss for future samples by moving the model's parameters in the direction of the negative gradient. (same as Stochastic Gradient Descent, except that i.i.d data is not assumed)

Confidence Weighted Learning [5]: Second order family of algorithms that attempts to give a prediction as well as a prediction confidence for new training samples, in a similar manner to the Passive-Aggressive algorithm, this algorithm seeks to trade-off between Passiveness (Modify the model's parameters as little as possible) Aggressiveness (Correctly classify new training samples with a minimum confidence threshold).

Adaptive Regularization of Weight Vectors [6]: A specific variant of confidence weighted learning algorithms in which three desires are introduced in the loss function instead of two (in contrast to other CW algorithms). The loss function in this methods attempt to make the model's parameters change as little as possible, predict the newer samples with low loss (commonly use hinge loss), and to gradually increase the confidence on the model's parameters as it receives more samples.

Kernel Methods

When the data is not separable in the original feature space it is mapped into higher dimensional feature spaces with the use of kernel methods.
Mapping into higher dimensional features allows the model to learn non-linear decision boundaries in the original feature space.
The kernel methods implemented in this library are:

Gaussian Kernel: This kernel stores a fixed amount of support vectors, and computes the similarity between the new data instance and all stored support vectors in terms of the original features

Polynomial Kernel: Compute new features as the result of polynomial combination between the original features.

Regularization Methods

The main purpose of regularization in machine learning is to train a model that is able to correctly generalize for previously unseen data.
To achieve good generalization, we use a sparsity inducing regularizer which penalizes the model for becoming too complex. By doing this, the search space of the problem is drastically reduced and in some cases adding regularization also allows for the computation of a closed form solution.

In the online learning setting a common regularization method is truncated gradient descent [7], this method regularizes the model by gradually decaying weights with a small absolute value towards 0, this makes the model more simple by learning to ignore features that may not be decisive enough, and allows for better generalization.

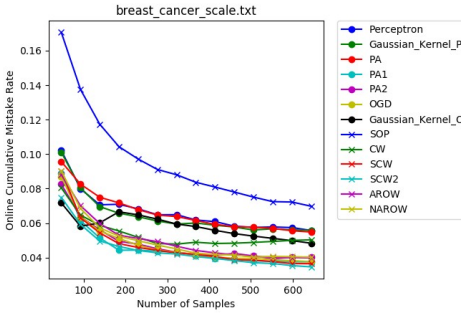
Results

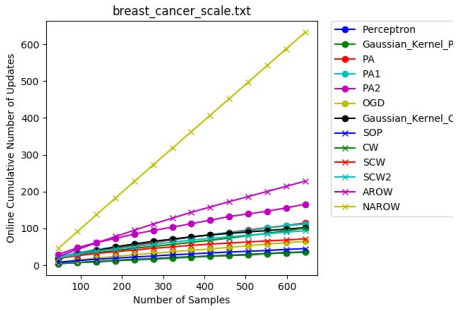
The library allows making comparisons between learning algorithms in terms of cumulative error rate, time of computation, and number of updates. Executing a comparison runs all of the algorithms of the selected task (binary or multi-class classification) with the specified data set, displays running statistics on console, and plots the resulting values for all of the algorithms.

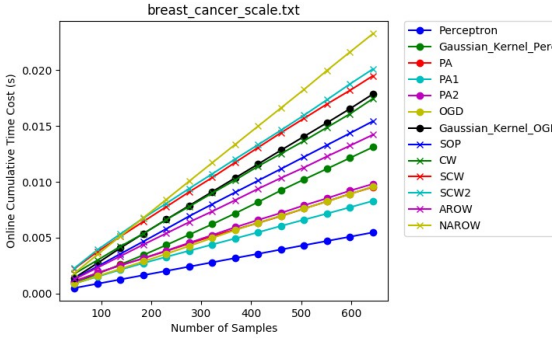
By default, the execution of algorithms in the library includes a process of hyper-parameter tuning in which all parameter values are kept fixed with the exception of one that takes different values from a specified range. After doing this process with all hyper-parameters the algorithm will be executed once more with the best values found.

The following plots that are shown with hyper-parameter tuning option active for all of the algorithms:

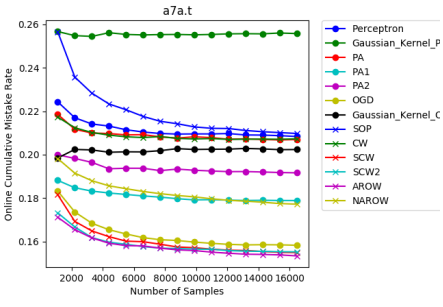
<u>Breast Cancer Data set</u>

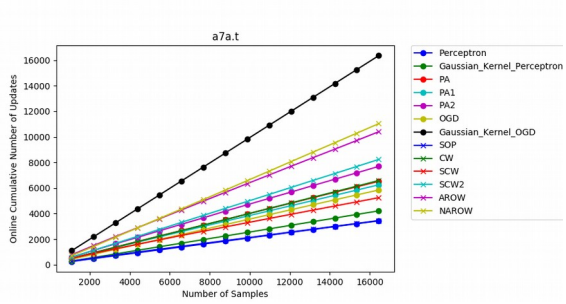


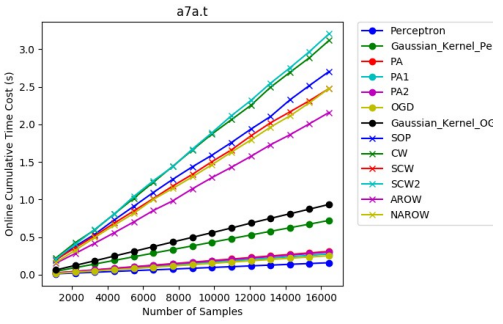




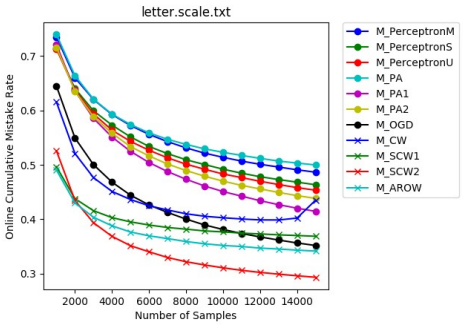
<u>a7a Data set</u>

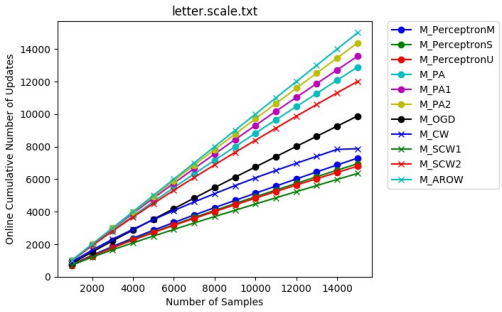


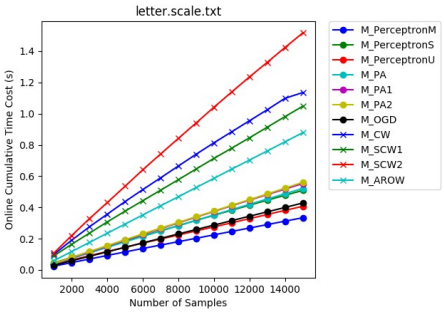




Letter (Scaled) Data Set







By analyzing these plots it is possible to draw conclusions about the performance of the learning algorithms for a particular data set, and for the options selected by the user. For instance, by comparing the results shown by the plots for different data sets it can be seen consistently that confidence weighted learning algorithms tend to have the best performance in terms of cumulative error rate, the simple perceptron algorithm tends to have the cheapest computation cost, but with not a very good performance (cumulative error rate), and passive-aggressive algorithms, AROW, and OGD usually fit in the middle in these categories.

Conclusion

LIBOL python is a library with the purpose of aiding in the testing of new approaches with respect to commonly known approaches in the online learning setting. This library is intended to be useful in research purposes due to its simplicity and the ease of source code modifications for external users with the intention of testing new ideas.

Github Link

https://github.com/LGuitron/LIBOL-python

References

[1] S.C.H. Hoi, J. Wang, P. Zhao. LIBOL : A Library for Online Learning Algorithms. Journal of Machine Learning Research 15 (2014) 495-499. pp 8 – 17

[2] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review 65 (6) (1958) 386.

[3] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz,Y. Singer, Online passive-aggressive algorithms, The Journalof Machine Learning Research 7 (2006) 551–585.

[4] E. Hazan, A. Rakhlin, P. L. Bartlett, Adaptive online gradient descent, in: Advances in Neural Information Processing Systems, 2007, pp. 65–72.

[5] M. Dredze, K. Crammer, F. Pereira, Confidence-weighted linear classification, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 264–271.

[6] K. Crammer, A. Kulesza, M. Dredze, Adaptive regularization of weight vectors, Machine Learning (2009) 1–33.

[7] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, The Journal of Machine Learning Research 10 (2009) 777–801.