Advanced Machine Learning
Online learning, Bandits, Reinforcement learning

Master MLDM

Assignment: Due date 15 December 2018 22:00 on claroline
The work can be done by groups of 3 students maximum

Note: all the material is available on the Claroline page!

Objectives of this practical session:

- Program an online learning method and compare it to the result of libSVM on UCI datasets.
- Program some bandit algorithms on a simple synthetic problem.
- Study a reinforcement learning project for playing PACMAN

You should write a report explaining your work and your results with relevant discussions. For the parts that need an implementation, you should provide the code.

1 Online Passive-Aggressive Algorithms

Passive Agressive Online algorithm is a special case of online algorithms for binary classification that can be seen as a kind of extension of SVM to the context of online learning. The principle followed here is for each iteration to update the classifier in order to remain as close as possible to the current one while achieving at least a unit margin on the most recent example. However, forcing a unit margin might turn out to be too aggressive in the presence of noise. In this last context, 2 variations can be considered casting a tradeoff between the desired margin and the proximity to the current classifiers. The pseudo-code is provided in Algorithm 1.

```
begin
    Initialize $\boldsymbol{w}_1 \leftarrow (0, \dots, 0)$;
    for $t = 1, 2, \dots$ do
        receive instance: $\boldsymbol{x}_t \in \mathbb{R}^n$;
        predict $\hat{y}_t = \text{sign}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)$;
        receive correct label: $y_t \in \{-1, 1\}$;
        compute loss $l_t = \max\{0, 1 - y_t(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)\}$;
        compute $\tau_t$ (cf text in the document);
        compute update: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$;
```

Algorithm 1: Passive Aggressive Online algorithm. the 3 possible updates for τ are described in the text of the document

Three possible updates can be considered for the τ value considered in Algorithm 1:

1. $\tau_t = \frac{l_t}{\|x_t\|^2}$ - the classic update,
2. $\tau_t = \min\{C, \frac{l_t}{\|x_t\|^2}\}$ - a first relaxation,
3. $\tau_t \frac{l_t}{\|x_t\|^2 + \frac{1}{2C}}$ - a second relaxation.

Work to do:

1. Implement the 3 versions of the algorithm in the context of linear binary classification.
2. Apply this algorithm on binary classification datasets available from the LibSVM software page and compare it to LibSVM.
   For the comparison, you should consider a learning set that are used by both algorithms for learning and test set for evaluation (think of a relevant setup). You can consider both running time and accuracy as comparison measures.
   http://www.csie.ntu.edu.tw/~cjlin/libsvm/
3. Try to introduce some noise (by flipping randomly some labels) and see the influence of the different update strategies.
4. Present your results in a report where you should first define clearly the experimental setup and discuss the results obtained.

Bonus Try to implement a kernel-based version of these methods by considering that any classifier can be defined as a weighted sum of seen examples. For this last case, you can fix a kernel and and consider a fixed size window corresponding to the considered landmarks on which the decision is taken (*i.e.* they correspond ot support vectors - nice update of the support vectors is out of the scope of this work but you could try to think about it).

2 Bandit Algorithm

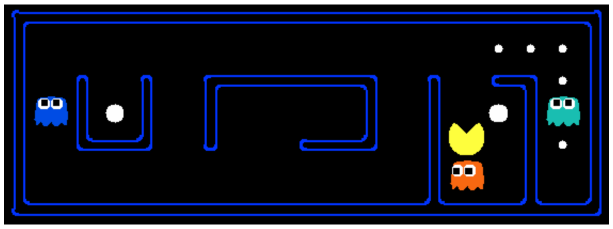
We will consider a set of 10 armed bandits such that each of them has a reward in the range $[0, 1]$ drawn according to a fixed probability distribution (Gaussian, Uniform, ...) that you have to fix. The goal is to compare 3 bandit algorithms and check their ability to find the best arm:

- **Incremental Uniform.** This algorithm repeatedly loops through the arms pulling each arm once each time through the loop. The average rewards for each arm are tracked and for the simple regret objective, the arm with the best average reward is returned as the best arm.
- **UCB.** The UCB algorithm. At the end of the exploration, the algorithm returns the arm that has accumulated the largest average reward.
- **ϵ-Greedy**. ϵ is a parameter of the algorithm such that the arm that appears to be the best is selected with a probability of ϵ otherwise a random arm is selected from among the other arms. At the end, it returns the best arm.

To compare the algorithms, you should run m trials and for each trial you will use the algorithm n times correspond to a sequence of n arm pulls. From the sequence of n arms you can compute the cumulative regret versus the number of pulls. Averaging over the trials will give an estimation of the expected cumulative regret. Your objective is to provide an experimental study of the behavior of the 3 strategies: Uniform, UCB, ϵ-Greedy (you free to experiment with other ϵ values but this is not required). You can also measure the simple regret by considering at each the best arm the algorithm thinks it is the best to pull.

3 Reinforcement Learning and PACMAN

We will use a project from UC Berkeley[1] implementing reinforcement learning approaches. The project can be found here http://ai.berkeley.edu/reinforcement.html, and you can download the code on the page threw this link: https://s3-us-west-2.amazonaws.com/cs188websitecontent/projects/release/reinforcement/v1/001/reinforcement.zip. This is a python 2.7 implementation.



You are not asked to complete the implementation of this project, you can actually find some solutions on the Web, *e.g.*:

- https://github.com/shiro873/pacman-projects/tree/master/p3_reinforcement_learning
- https://github.com/worldofnick/pacman-AI/tree/master/reinforcement
- https://github.com/TuringKi/PacMan-AI/tree/master/Reinforcement%20Learning

You can of course probably find other implementations. The idea is rather to understand what it is done to solve this project.

Work to do:

- You are asked to provide a report containing the answers to the questions asked in the presentation of the project. Answers can be brief but you are encouraged to provide some pseudo-code/algorithm when this is relevant. It is important to show that you understood each part.
- Provide an experimental study on the behavior of the different strategies presented in the code. You can limit yourself to small grids.
- Propose your own improvements. In this case you should provide the full code with instructions to use your implementation easily.

[1]http://ai.berkeley.edu