Statistical Techniques in Robotics 16-831 - The Robotics Institute

Project 1: The Weighted Majority Algorithm

Deadline: September 30th, 2015 11:59 PM (submit via Blackboard)

1 Introduction

One application of the Weighted Majority Algorithm is the Prediction with Expert Advice (PWEA) problem. In PWEA, a finite number of experts will each make a prediction. The online learning algorithm then chooses to listen to a particular expert or combine predictions from multiple experts. The goal of the online learning algorithm is to minimize the number of mistakes/loss in the long run.

The theoretical goal of this project is to help you understand and become comfortable deriving basic performance bounds for online algorithms for solving the PWEA problem.

The programming portion of the project is designed to help you understand the actual implementation of an online algorithm. We hope to show that once all of the theory has been laid out, the actual implementation of the online learning algorithm is quite simple and that most of the algorithms are very short.

2 Theory Questions (40 points)

2.1 Realizability (5 points)

What is realizability and why is it important for online learning?

2.2 Hypothesis class (5 points)

What is a hypothesis class? Is the hypothesis class finite or infinite? Why is this important?

2.3 Consistent (Greedy) Algorithm Regret Bound (10 points)

In class, we assumed realizability when we derived the mistake bounds for the consistent algorithm (i.e., keep only the experts which were consistently correct).

The mistake bound in the realizable case is:

$$M \leq N - 1 \tag{1}$$

where M is the number of mistakes the learner makes and N is the number of experts.

When there is no perfect expert (not realizable), we cannot simply eliminate the wrong experts. Now let's suppose we always listen to the expert that made the fewest mistakes so far, and if there are multiple such experts, pick the one who has the lowest index. Prove that the number of mistakes M the algorithm made is upper bounded as follows:

$$M \leq N \cdot m^* + (N - 1), \tag{2}$$

where N is the number of experts and m^* is the number of mistakes made by the best expert.

2.4 Understanding the η (penalty) parameter (10 points)

We have taught in class that the randomized weighted majority algorithm has a expected regret bound as follows:

$$E(R) \leq \eta m^* + \frac{\ln N}{\eta} \tag{3}$$

, where N is the number of experts and m^* is the number of mistakes the best expert made, $1 - \eta$ is the "shrinking factor" that applies to wrong experts' weights

(1) Derive the optimal value of η given N and m^*.

(2) Usually we won't be able to know m^* at the time we choose the value for η. However, suppose we know that the number of predictions rounds T is much smaller compared with the number of experts N, should we choose a bigger or a smaller value for η?

(3) Assume $m^* = O(T)$, where T is the number of prediction rounds. How should we pick η? What if m^* is sublinear in T (e.g., $O(\sqrt{T})$) ?

2.5 Understanding Adversarial Environments (10 points)

Consider a game setting where:

1. The world/adversary presents a observation x_t to the online learner (PWEA).
2. The online learner receives x_t and then asks each expert to make a {0,1} prediction.
3. The adversary sees the prediction and current weight of each expert, then decides the true {0,1} label y_t.

4. The online learner chooses a PWEA strategy (e.g., weighted majority or randomized weighted majority) to make a final prediction before y_t is revealed to the online learner.

(1) Show that a strategy exists for the adversary such that loss (the number of mistakes) is always maximized for deterministic weighted majority algorithm. You should be able to explain the strategy in a few sentences. Hint: consider a simple case of just 2 experts making a binary prediction.

(2) What's the largest expected loss an adversarial strategy can make for randomized weighted majority (assume perfectly random, i.e., the random seed is completely hidden from both the learner and adversary)? Why does randomization help improve the worst case performance of the learner?

3 Coding (60 Points)

3.1 General Instructions

- For this coding problem, you are free to choose your favorite programming language among Matlab, C++ and Python.
- Make sure to upload all the dependencies as well (if any). **The code should be compilable.**
- Make necessary code comments to help TA understand the code logic.

Here's a rough guideline for your code:

1. Nature sends observation.
2. Learner receives observation, asks each expert to make a prediction.
3. Nature determines the label. For adversarial case, the nature gets to see the weight vector and prediction for each expert (but not directly the final prediction made by the learner).
4. Learner makes a final prediction and the true label is revealed.
5. Learner suffers loss and updates expert weight.
6. Repeat.

3.2 Programming Nature (10 points)

Consider using PWEA algorithms to predict the results (win or lose binary prediction) for Tartan's sports games. You have three experts: expert one is a die-hard fan for Tartan's sports team and always says win; expert two is pessimistic and always says lose; and expert three predicts Tartan will lose for odd-number matches and will win for even-number matches.

Design three nature classes that generate true labels y_t in a fashion that is (1) stochastic (2) deterministic and (3) adversarial, respectively. (Be creative)

A reference value for the total rounds of prediction T is 100. You are encouraged to explore different values of T, particularly the coupling effect with η.

3.3 Implement the Weighted Majority Algorithm (15 points)

Assume the loss function is {0,1} loss, i.e, whether a mistake is made or not. Plot both the regret and learner's loss as a function of time/rounds t for all three nature classes. Make a few explanations for each plot.

3.4 Implement the Randomized Weighted Majority Algorithm (20 points)

Again assume the loss function is {0,1} loss. Experience a few different values of $\eta \in [0, 1/2]$ and plot both the regret and learner's loss as a function of time/rounds t for all three nature classes. Make a few explanations for each plot.

3.5 More Experts and Observations/Features (15 points)

At this point, you probably have found that the PWEA algorithms can sometimes lead to large loss/errors though being no-regret (even negative regret). This is because we only have three dumb experts.

Your job now is to 1) add more available observations/contextual features for the PWEA algorithm; and 2) add more experts that utilize the observations to the expert pool. Previously expert 3 makes predictions based on the prediction round id while experts 1 and 2 make no use of observations.

Show how regret and loss change after increasing experts that use more observations. You may also want to modify the label generation procedure for the nature classes to include new observation factors. Here's an example: Suppose now the weather is also revealed to the learner, we can introduce a new expert that predicts win if it's rainy and lose if it's sunny. A new deterministic nature class could generate match result that combines factors of weather, prediction round id, and last game result.

4 What to Submit

Your submission should consist of only 1 file, a zip file named `<AndrewId>.zip`. This zip file should contain

- a folder `code` containing all the code and data (if any) files you were asked to write and generate.

- a `pdf` named `writeup.pdf` containing the results, explanations and images asked for in the coding section and the answers to the theory questions.