Statistical Techniques in Robotics 16-831 - The Robotics Institute

Homework 1: The Weighted Majority Algorithm

Deadline: February 6th, 2019 11:59 PM

TA: Rawal Khirodkar Submission: Gradescope

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.

1.1 Instructions

Submit this homework on Gradescope. Make sure your Gradescope student account is made using your **Andrew email ID** and your official name in the CMU system - this is essential to correctly record grades. Join the course using the code: **97EDJG**. Remember, you can only use a maximum of 2 late days (out of 3 for the semester) for any assignment. Beyond that, you will be penalized by a $1/3^{\text{rd}}$ deduction in points per day. Detailed instructions on what to submit are given in Section 4.

2 Theory Questions (45 points)

2.1 Regret (5 points)

Recall the definition of regret:

$$R_t(\mathcal{H}) = \sum_{i=1}^{t} \ell(\hat{y}^{(i)}, y^{(i)}) - \min_{h \in \mathcal{H}} \sum_{i=1}^{t} \ell(h(\mathbf{x}^{(i)}), y^{(i)})$$

The regret at time t is defined as the difference in the loss incurred by the learner and the loss of the best possible expert in hindsight up until time t. Here, assume each h in the hypothesis class $\mathcal{H}$ is just an indicator of which expert's prediction to select from $x^{(t)}$. In other words, the hypotheses are the experts themselves.

Question: Can regret be negative? If yes, provide an example. If no, provide a mathematical explanation. **Note:** Your answer depends on the assumptions you make. Make sure to state them with your answer.

2.2 Constructing Experts (5 points)

In the specific case of the PWEA problem, we defined the hypothesis class as a set of indicator functions over experts. What if we don't have explicit experts available to us?

We now define our input space as a finite set of N feature vectors: $\mathcal{X} = \{x_1, x_2, ..., x_N\}$. Previously, these were vectors where each element was the prediction of a single expert. Now, you can think of x_i as some feature representation of an image of a given size (e.g., pixel values of image). We define our output space the finite set: $\mathcal{Y} = \{1, 2, ..., K\}$. Here, $y \in \mathcal{Y}$ could be the label of an image (e.g., car, dog, building).

In general, given an input space $\mathcal{X}$ and an output space $\mathcal{Y}$, a hypothesis h is just a mapping: $h : \mathcal{X} \to \mathcal{Y}$. There is a true label for each input vector $x_i \in \mathcal{X}$ which we do not know.

Question: Construct a hypothesis class $\mathcal{H}$ to make this a realizable scenario. In other words, show that in the case where we have a finite input space and a finite output space, we can always construct a hypothesis class (a set of experts) which contains the perfect expert, no matter the true labeling for the feature vectors. What is the size of such a hypothesis class?

2.3 Understanding penalty parameter η (10 points)

In this question, we use the shorthand R^* for the regret of the best hypothesis, i.e., $R(h^*)$. Recall the regret and expected regret bounds for the weighted majority (subscript W) and randomized weighted majority (subscript RW) algorithms respectively:

$$R_W^* \leq (1 + 2\eta)m^* + \frac{2 \ln N}{\eta}$$

$$E(R_{RW}^*) \leq \eta m^* + \frac{\ln N}{\eta}$$

Here N is the number of experts, m^* is the number of mistakes the best expert made, and $(1 - \eta)$ is the "shrinking factor" for incorrect experts' weights. An-

swer the following questions in the context of **both** weighted and randomized weighted majority algorithms. **Note:** Clearly state the assumptions you make.

1. **(3 points) Question:** In each case, derive the optimal value of η given N and m^*. Consider if there are any constraints you can put on η.

2. **(2 points) Question:** Suppose we don't know m^* at the time we choose η (this is usually the case). However, suppose we know that the number of prediction rounds T is much smaller than N. In each case, should we choose a bigger or smaller value for η?

3. **(5 points) Question:** In each case, can we pick η to make the algorithm no-regret for the following cases of m^*? If yes, provide η such that the algorithm is no-regret.

   (a) $m^* = O(T)$.

   (b) $m^* = O(T^x),\ x < 1$, i.e., m^* is sublinear in T.

2.4 Multi-Class Classficiation (15 points)

In class, we focused on the binary classification problem (e.g., experts only bet on one of two horses) and studied the Halving algorithm. Now let us consider the general k-class ($k \geq 3$) prediction problem (e.g., experts need to bet on one of k different horses). More formally, let us denote k classes as $\{1, 2, ..., k\}$. As usual, we have N experts and every round, each expert will choose a label from the set $\{1, 2., ..., k\}$. Here, we assume the realizable setting, where at least one expert never makes a mistake.

Let us consider two scenarios: (1) A fully observable scenario where the learner is told the true label after it makes a prediction, and (2) a partially observable scenario where the learner is only told if its prediction is correct or not but, it is **not** given the true label. The second scenario is partially observable because we cannot compute the loss for all of the experts, whereas in the first scenario we can compute the loss for each expert.

1. **(5 points) Question:** Show that for the *fully observable scenario*, the Halving algorithm has the same mistake bound as the binary label case. Here, the prediction at every round t is the label $\hat{i} \in \{1, 2, ..., k\}$ that has the most expert votes (we can break ties arbitrarily). On being given the true label, we eliminate every incorrect expert and move to round $t + 1$, just as usual.

2. For the *partially observable scenario*, consider the following high-level strategy. The prediction at each step is the same as above. But now, we do not get to see the true label. We only are told if our prediction is correct or not. If label $\hat{i}$ is correct, we do nothing. On the other hand, if label $\hat{i}$ is wrong (i.e., we just made a mistake), we eliminate all experts who chose label $\hat{i}$ at round t. We then move to round $t + 1$.

(10 points) Question: Convert the above high-level strategy into pseudocode for an algorithm and derive its mistake bound.

Note: The mistake bound has to be logarithmic with respect to the number of experts N. It is not acceptable for the bound to be polynomial with respect to N.

Hint: The pigeonhole principle may be useful.

2.5 Understanding Adversarial Environments (10 points)

Consider a non-realizable PWEA setting where:

- The world/adversary presents a observation x_t to the online learner.
- The online learner then asks each expert to make a {0,1} prediction.
- The adversary sees all the individual expert predictions and their current weights, then decides the true {0,1} label y_t.
- The online learner now 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. **Note:** The adversary has access to the exact strategy being used to make this prediction. But it does not have access to the outputs of the randomizer which the learner might use.

Now, answer the following questions:

1. **(5 points) Question:** Show that a strategy exists for the adversary such the loss (the number of mistakes) is always maximized for the deterministic Weighted Majority Algorithm. You should be able to explain the strategy in a few sentences. Hint: Consider a simple case with just 2 experts.
2. **(5 points) Question:** Assume that at least one expert is correct at least once. Prove that the expected loss of the Randomized Weighted Majority Algorithm against the worst adversary (as designed above) is strictly less than the loss of WMA. Why does randomization help improve the worst case performance of the learner?

3 Coding (55 Points)

3.1 General Instructions

- For this coding problem, you are free to choose your favorite programming language among Matlab and Python.
- If using Python, use standard scientific packages (nothing outside of this list).

- **The submitted code should be self-contained, except for the packages listed above.**
- Make necessary code comments to help TA understand the code logic. Use good functional coding practices.

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

- Average cumulative regret (y-axis)

$$\frac{R_t(h^*)}{t} = \frac{1}{t} \max_{h \in \mathcal{H}} \Big(\sum_{i=1}^{t} \ell(\hat{y}^{(i)}, y^{(i)}) - \sum_{i=1}^{t} \ell(h(\mathbf{x}^{(i)}), y^{(i)}) \Big)$$

  for $t = 1, \ldots, T$ (x-axis). Set the axes for this plot to $[0, 1]$ in the y-axis and $[0, 100]$ in the x-axis.

- The learner's and all the experts' total losses (y-axis)

$$\sum_{i=1}^{t} \ell(\hat{y}^{(t)}, y^{(t)})$$

$$\sum_{i=1}^{t} \ell(h(\mathbf{x}^{(t)}), y^{(t)}) \quad \forall h \in \mathcal{H}$$

  for $t = 1, \ldots, T$ (x-axis). Set the axes for this plot to $[0, 100]$ in the y-axis and $[0, 100]$ in the x-axis.

These two plots, for all 3 nature classes = 6 plots in total.

3.4 Implement the Randomized Weighted Majority Algorithm (15 points)

Again assume the loss function is {0,1} loss. Try a few different values of $\eta \in [0, 1/2]$ and make the same 6 plots as question 3.3. 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. For example, suppose the Tartans win more frequently when the weather is sunny (this is part of nature). An expert (which observes the weather) could predict the Tartans win if it's sunny and lose if it's rainy. An even smarter expert could use past observations to learn this correlation (i.e. using an SVM or some other classifier).

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.

4 What to Submit

Your submission should consist of:

1. a zip file named `<AndrewId>.zip` consisting of a folder `code` containing all the code and data (if any) files you were asked to write and generate. Submit this to `Homework 1 - Programming` on Gradescope.

2. a `pdf` named `<AndrewId>.pdf` with the answers to the theory questions and the results, explanations and images asked for in the coding questions. **It is compulsory to type-set your answers; images of handwritten documents will not be accepted**. Submit this to `Homework 1 - Theory` on Gradescope.

It is mandatory to include in the write-up the names of all people you have discussed this assignment with!