Statistical Techniques in Robotics 16-831 - The Robotics Institute

Project 1: The Weighted Majority Algorithm

Deadline: September 19th, 2017 11:59 PM

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 Canvas. Remember to start **early** as this class has no late days! 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 Bayesian Halving: Predicting with a Prior (10 points)

In the class, we focused on binary classification problem using experts' advice, where we assume before the game starts, we have no knowledge about which expert is good or bad. In this problem, we assume that we are given initial information which describes how good or bad an expert is. This is in the form of prior probabilities over the experts, denoting their likelihood of being the best expert. Priors are commonplace in Bayesian statistics, expressing one's belief over a given uncertain quantity before any evidence is taken into account.

More formally, we have N experts $\{e_1, ..., e_N\}$ and there exists at least one expert is perfect (i.e., realizable setting). Every round each expert predicts 0 or 1. We are given a prior which is a probability distribution p over experts: $0 < p(e_i) < 1, \forall i$ and $\sum_i p(e_i) = 1$. Intuitively, $p(e_i)$ denotes the likelihood, in our knowledge, that expert e_i is the best expert.

In this case, a natural prediction strategy is to run the halving algorithm, but where the prediction of each expert e_i in the version space (i.e., all experts that have not yet made any mistakes) is weighted according to $p(e_i)$.

1. **(3 points) Question:** Convert the above high level strategy into pseudocode for an online expert algorithm.

2. **(4 points) Question:** Prove that the total number of mistakes is $O(\log(\frac{1}{p(e^*)}))$, where e^* is the perfect expert who never makes a mistake.

3. **(3 points) Question:** Compare this mistake bound to that of the Halving algorithm we saw in class. When will this Bayesian Halving algorithm perform better than the one we learned in class? When will it perform worse?

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 questions, please use either Matlab and Python.
- If you use Python, use standard scientific packages; nothing outside of this list: https://docs.continuum.io/anaconda/pkg-docs
- **The submitted code should be self-contained, except for the packages listed above.**
- Make sure you comment your code thoroughly to help the TAs understand your logic. Use good functional coding practices.

Here are the typical steps for an online learner in the PWEA setting:

1. The world/adversary presents an observation to the online learner.
2. The online learner asks each expert to make a prediction.

3. The world then determines the label. In the adversarial case, the adversary gets to see the predictions and weights of each expert, but **not** the final prediction made by the online learner.
4. The online learner makes a final prediction and the true label is revealed.
5. The online learner incurs a loss and updates the weights of the experts.
6. Repeat from step 1.

3.2 Programming the World (10 points)

Consider using PWEA algorithms to predict the results for the Tartans' sports games. Here, the prediction is binary: win or lose. You have three experts in $\mathcal{H}$: expert one is a die-hard, and rather foolishly optimistic, fan for Tartans' sports team and always predicts a win; expert two is a resigned pessimist who always predicts a loss; and expert three predicts that the Tartans will lose every odd-numbered game and win every even-numbered game.
Code: Design three world classes for the Tartans' sports team that generate true labels y_t as follows: (1) stochastic (2) deterministic and (3) adversarial, respectively. Be creative! **Note:** Only the adversarial world can see the learner's strategy or the experts' predictions and weights.

3.3 Weighted Majority Algorithm (15 points)

Code: Implement the Weighted Majority Algorithm. Assume the loss function is the {0,1} loss, i.e., 0 for a correct prediction and 1 for a mistake. Plot:

1. Average cumulative regret

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

   on the y-axis against time steps $t = 1, \ldots, T$ on the x-axis.

2. The learner's loss and all the experts' losses

$$\ell(\hat{y}^{(t)}, y^{(t)})$$
$$\ell(h(\mathbf{x}^{(t)}), y^{(t)}) \quad \forall h \in \mathcal{H}$$

   on the y-axis against time steps $t = 1, \ldots, T$ on the x-axis.

There should be 2 plots for each world, meaning 6 plots in total. Discuss your observations in each plot. Take the total number of rounds T as 100. Explore a few different values of $\eta \in [0, 1/2]$ and choose one you like. You are encouraged to explore different values of T and observe how it is coupled with η.

3.4 Randomized Weighted Majority Algorithm (15 points)

Code: Implement the Randomized Weighted Majority Algorithm. Again, assume the loss function is the {0,1} loss. Make the same 6 plots as in question 3.3. Discuss your observations in each plot. Take the total number of rounds T as 100. Explore a few different values of $\eta \in [0, 1/2]$ and choose one you like.

3.5 More Experts and Observations/Features (15 points)

At this point, you have probably have found that the PWEA algorithms can sometimes lead to large loss/errors, even though they might be no-regret (sometimes even negative regret). This is because we only have three dumb experts. The previous experts make no use of observations.

Now, say we have that the Tartans win more frequently when the weather is sunny. This information can be considered as a part of the world. An expert who observes the weather could predict a win if it's sunny and a loss if it's not. An even smarter expert could use past observations to learn this correlation (i.e. using an SVM or some other classifier).

Code: Your job now is to: (1) Add observations/contextual features for the PWEA algorithm. The features could be information like weather, home vs. away games, win streaks, etc. (2) Add more experts to the expert pool that utilize the features.

You should also modify the label generation procedure for the world classes to incorporate the new features; the features aren't useful if they don't tell us anything about the labels!

Question: Pick up to three additional experts which use the features. Plot the same six plots from the previous questions and show how regret and loss change. Make sure you describe your features and experts as well.

4 What to Submit

Your submission should consist of a single 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` with the answers to the theory questions and the results, explanations and images asked for in the coding questions.