Statistical Techniques in Robotics (16-831, S22) Lecture #02 (Monday, January 24)

Online Learning and Prediction with Expert Advice

Lecturer: Kris Kitani *Scribes: Grace Brueggman, Kevin Gmelin*

1 Review

In the last lectures, on the topic of Robot Learning Problems, we discovered there are a vast number of learning algorithms. We also learned there are three major categories of feedback from learning algorithms. These categories are (1) Exhaustive versus Sampled Feedback, (2) Instructive versus Evaluative Feedback, and (3) One-shot versus Sequence Feedback. In the next three subsections, we will define each of these categories and provide examples of algorithms that are classified as such. Understanding these categories helps to frame the context of this course, which focuses on learning from evaluative feedback, and where the upcoming content falls in the bigger picture of algorithms that are outside of the scope for the semester. On a smaller scale, it grounds us for this topic on PWEA, or Prediction with Expert Advice.

1.1 Exhaustive & Sampled Feedback

This distinction of feedback deals with the question: *Does your learner have access to all of the data?* Examples of algorithms that are classified as exhaustive or sampled can be viewed in the Appendix at Table 1.

Definition 1. Exhaustive Feedback is the situation where the learner is exposed to all possible situations.

Definition 2. Sampled Feedback is the situation where the learner only has access to a sub-set of the possible situations.

1.2 Instructive & Evaluative Feedback

This distinction of feedback deals with the question: *What kind of feedback do we get from the environment?* Examples of algorithms that are classified as instructive or evaluative can be viewed in the Appendix at Table 2.

Definition 3. Instructive Feedback provides the learner feedback on all possible actions.

Definition 4. Evaluative Feedback provides the learner feedback for the chosen action with no mention how the alternative actions would have resulted.

1.3 One-shot & Sequence Feedback

This distinction of feedback deals with the question: *Is the data generation affected by the previous action?* Examples of algorithms that are classified as one-shot or sequence can be viewed in the Appendix at Table 3.

Definition 5. One-shot Feedback is an isolated event where the outcome of the current action does not affect the next one.

Definition 6. Sequential Feedback is the situation where the outcome of the current action affects future actions and outcomes.

2 Summary

2.1 Online Learning

In regular supervised learning, there is a training stage where the agent learns the parameters of a model, followed by the test stage where the agent uses its learned model to make predictions. Thus, in supervised learning, the training and test stages are separated. When the training and test stages can't be separated, online learning is used. This is a different type of learning algorithm where the learning is interleaved with execution.

At each time step, the agent receives some observations. From these observations and the agent's learned models, the agent has to make a prediction. After making a prediction, the agent receives the true outcome. The goal of online learning is to minimize the error between the prediction and true outcome, which is formally encapsulated as regret. This process of online learning is shown below in Figure 1.

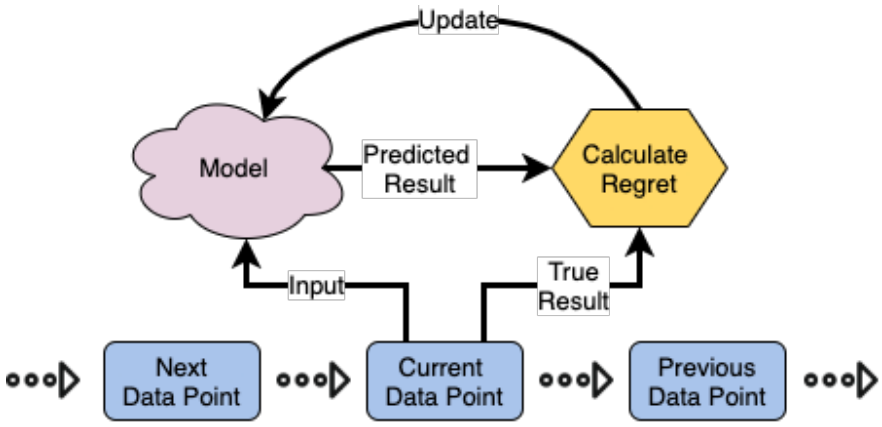

Figure 1: Online-Learning Flowchart

Unlike supervised learning algorithms, where separate training and test stages require an assumption of a stationary data distribution, online learning algorithms can adapt on-the-fly and can thus work with stochastic, deterministic, and adversarial distributions.

Definition 7. Stochastic Distribution is when the data has randomness infused and values have a probabilistic nature. It is the opposite of deterministic.

An example of stochastic distributed data is a sequence of dice rolls.

Definition 8. Deterministic Distribution is when there is enough information to predict a result with absolute certainty. It is the opposite of stochastic.

An example of deterministic distributed data is a look-up table.

Definition 9. Adversarial Distribution is when the data is generated to specifically break the model.

An example of adversarial distributed data is fast gradient sign [2].

Online learning can also work with stationary data distributions, but supervised learning algorithms are likely a more appropriate choice for such a data set. Because the online learning model incrementally adapts as it collects more inputs and attempts to minimize regret, duplicate data points previously seen can appear again at later time steps and achieve different results.

In online learning, the feedback may be either exhaustive or sampled. Furthermore, the feedback may be either instructive or evaluative. However, typically the feedback is one-shot since it is assumed to be not correlated with past feedback and actions.

Online learning algorithms are useful in cases where:

- The feedback is incremental such that learning needs to be interleaved with execution
- The training data is too large to process at once
- The data distribution changes and the agent needs to adapt

Types of online learning algorithms include:

- Prediction with Expert Advice
- Bandit
- Contextual Bandit
- Online Convex Optimization

2.2 Prediction with Expert Advice

One subset of online learning algorithms is prediction with expert advice (PWEA). The feedback for this set of algorithms is:

- One-Shot: We assume the sequence of data is not correlated
- Instructive: We receive feedback not only on the chosen action but also on all alternative actions
- Exhaustive: Both the action and state spaces are finite and so the learner can be exposed to all possible situations

To introduce the topic, let us propose an example modified from How to Use Expert Advice [1]:

Say a learner is trying to predict the weather today, at time $t = 1$, where either it will snow, indicated as a 1, or it will not snow, indicated as a 0. Prior to the learner making its prediction, it can consider the predictions of a finite group of local weathermen $\mathcal{E} = \{\mathcal{E}_1, ..., \mathcal{E}_E\}$, here acting as the E experts, where each expert has made a binary prediction $\mathbf{x}^{(1)} \in \{0, 1\}$ about whether it will snow or not. The learner can then make a prediction $\hat{y}^{(1)} \in \{0, 1\}$, where the method of coming to that choice is dependent on the specific algorithm within PWEA utilized.

By the end of the day, it will have either snowed or not snowed, and the learner will have feedback on not only if the learner predicted correctly, but also feedback for if each of the experts predicted correctly. The next day, where $t = 2$, rather than considering all of the experts $\in \mathcal{E}$, the learner will only consider some of the experts' predictions, choosing the subset of those experts who predicted correctly the day before. This continues day after day, $t = 3, 4, ..., N$, with the learner only considering the advice from the experts with a track history of correctly predicting the weather, based on the assumption that this will lead to a better chance of correct weather predictions by the learner.

The essential goal of prediction with expert advice problems is to perform as well as the best prediction strategy, or expert, resulting in a minimized regret [3].

Definition 10. Regret is the difference between the losses of the learner and the losses of the best expert.

Definition 11. Loss is the difference between the predicted result and the true result.

In PWEA problems with E experts, the expert advice at trial t can be represented as an E-dimensional vector: $\mathbf{x}^{(t)}$. The advice of the e-th expert is $\mathbf{x}_e^{(t)} \in \{0, 1\}$. The set of possible observations is the instance domain, which is represented by $\mathcal{X}$.

The prediction made by the learner at trial t is specified by $\hat{y}^{(t)} \in \{0, 1\}$. The outcome of trial t is specified by $y^{(t)} \in \{0, 1\}$. The set of all possible outcomes is the target domain, which is represented by $\mathcal{Y}$.

The goal of a prediction with expert advice algorithm such as the greedy algorithm is to find a "good" hypothesis. Here, the notion of good is formalized by the concept of minimizing regret. A hypothesis h is a mapping from the instance domain to the target domain. The set of all possible hypotheses is called the hypothesis class, and it is represented by $\mathcal{H}$.

2.3 Greedy (Consistent) Algorithm

One prediction with expert advice algorithm is the greedy algorithm, the pseudocode for which is shown below in Algorithm 1.

Algorithm 1 Greedy Algorithm

$\mathbf{V}^{(1)} = \mathcal{H}$ ▷ Version space initially stores all hypotheses
for $t = 1, \cdots, T$ **do**
  RECEIVE $(\mathbf{x}^{(t)})$ ▷ Receive expert advice
  $h = \text{FIRSTELEMENT}(\mathbf{V}^{(t)})$ ▷ Greedy selection
  $\hat{y}^{(t)} = h(\mathbf{x}^{(t)})$ ▷ Prediction
  RECEIVE $(y^{(t)})$ ▷ Receive true outcome
  $\mathbf{V}^{(t+1)}) = \{h \in \mathbf{V}^{(t)} : h(\mathbf{x}^{(t)}) = y^{(t)}\}$ ▷ Update by keeping only consistent hypotheses
end for

In the greedy algorithm, we maintain a set, called the version space, of hypotheses (maps from the instance domain to the target domain) that have been consistent with every outcome received so far. A hypothesis is consistent if, for every time step, it correctly maps the expert advice at that time step to the true outcome. At each time step, the greedy algorithm first receives expert advice. In order to make a prediction, it selects the first hypothesis from its version space and uses this hypothesis to map the expert advice to a prediction. It then receives the true outcome and it removes all hypotheses from the version space that are inconsistent with this true outcome.

To expand the previous example of general PWEA to predict the weather, consider the following, a more concrete example in regard to predicting the weather with the Greedy Algorithm:

> There are 5 local weathermen $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}$. On the first day at $t = 1$, they make the following predictions about whether it will snow, 1, or not snow, 0, such that $\mathbf{x}^{(1)} = \{1, 1, 0, 0, 1\}$. The Greedy approach forces the learner to choose to follow the advice of the first expert, $\mathcal{E}_1$, which is $x_1^{(1)} = 1$. Therefore $\hat{y}^{(1)} = 1$. By the end of the day, there is no snow. The learner, $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_5$ predicted incorrectly, and therefore have a loss $l^{(1)} = 1$. Since two of the experts, $\mathcal{E}_3$ and $\mathcal{E}_4$, predicted correctly and have a loss $l^{(1)} = 0$, the cumulative difference between the learner's losses, also known as the regret is $r = 1$. Because $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_5$ are not consistent, they will be removed from the set of experts whose advice is considered in the future.
>
> The next day, at $t = 2$, the learner only considers the advice of the consistent experts $\mathcal{E}_{consistent} = \{\mathcal{E}_3, \mathcal{E}_4\}$, where both predict it will snow, $\mathbf{x}^{(2)} = \{1, 1\}$. The Greedy approach forces the learner to choose to follow the advice of the first expert, $\mathcal{E}_3$, which is $x_1^{(2)} = 1$. Therefore $\hat{y}^{(2)} = 1$. Both experts and the learner were correct and it did snow that day, so the advice of $\mathcal{E}_3$ and $\mathcal{E}_4$ will be considered the next day. The learner's loss this day is $l^{(2)} = 0$ and its regret is $r = 1$.
>
> The next day, at $t = 3$, the learner considers the advice of the consistent experts $\mathcal{E}_{consistent} = \{\mathcal{E}_3, \mathcal{E}_4\}$, where $\mathbf{x}^{(3)} = \{0, 1\}$. The Greedy approach forces the learner to choose to follow the advice of the first expert, $\mathcal{E}_3$, which is $x_1^{(3)} = 0$. Therefore $\hat{y}^{(3)} = 0$. It snows that day, so the learner and $\mathcal{E}_3$ are incorrect, suffering a loss of $l^{(1)} = 1$ The learner's loss this day is $l^{(2)} = 1$, making the learner's cumulative loss $l = 2$. There is only one consistent (perfect) expert remaining, $\mathcal{E}_{consistent} = \{\mathcal{E}_4\}$, and their cumulative loss is $l = 0$. Therefore the difference of the learner's and $\mathcal{E}_4$'s cumulative losses is our regret, $r = 2$.

The greedy algorithm works best when at least one of the experts is perfect. An expert is perfect if their advice is always the same as the true outcome. If the greedy algorithm is applied in an

instance where there are no perfect experts, it is possible that none of the original hypotheses in the version space are consistent and the version space becomes empty and the greedy algorithm breaks. The idea of a perfect expert can be more generally described using the concept of realizability.

For a problem to be realizable, it must satisfy two requirements:

1. There must exist some mapping from the instance domain to the target domain that generates the true outcomes:
$$h^* : \mathcal{X} \to \mathcal{Y}$$
2. Such a mapping must be a member of the hypothesis class:
$$h^* \in \mathcal{H}$$

If a prediction with expert advice problem is realizable, than we can show that the greedy algorithm will make at most $|\mathcal{H}| - 1$ mistakes, where $|\mathcal{H}|$ is the size of the hypothesis class and a mistake is a prediction that differs from the true outcome. Intuitively, this comes from the fact that the version space initially has size $|\mathcal{H}|$ and each mistake removes at least one hypothesis from the version space. If more than $|\mathcal{H}| - 1$ mistakes are made, that would result in the version space becoming empty. However, since the problem is realizable, the version space can not become empty, and thus the greedy algorithm will make at most $|\mathcal{H}| - 1$ mistakes. A more formal proof is derived below.

Theorem 12. The greedy/consistent algorithm will make at most $|\mathcal{H}| - 1$ mistakes given realizability.

To prove this, we first note that each mistake removes at least one hypothesis from the version space. This is due to the fact that the greedy algorithm makes a prediction using a hypothesis from the version space. If a mistake is made, that means the chosen hypothesis did not map the instance to the true outcome. Thus, the chosen hypothesis is inconsistent and since the chosen hypothesis belongs to the version space, it will be removed. Thus, after one mistake, we have that:

$$|\mathbf{V}^{(t+1)}| \leq |\mathbf{V}^{(t)}| - 1$$

Likewise, after M mistakes since the version space was initiated, at least M hypotheses will have been removed from the version space:

$$|\mathbf{V}^{(t)}| \leq |\mathbf{V}^{(0)}| - M = |\mathcal{H}| - M$$

where the equality comes from the fact that the version space is initiated to be the same as the hypothesis class.

Next, we place a lower bound on the size of the version space by noting that the assumption of realizability states there is some hypothesis in the hypothesis class that will always correctly map from the instance domain to the target domain. Since the version space is initialized to be the hypothesis class, than the version space will contain such a perfect hypothesis, and such a perfect hypothesis will by definition be consistent. Thus, the version space will contain a consistent hypothesis which will not be removed by the greedy algorithm, and we find that:

$$|\mathbf{V}^{(t)}| \geq 1$$

Combining these lower and upper bound equalities:

$$1 \leq |\mathbf{V}^{(t)}| \leq |\mathcal{H}| - M$$

Removing the middle element:

$$1 \leq |\mathcal{H}| - M$$

This can be rearranged to find:

$$M \leq |\mathcal{H}| - 1$$

Hence, the number of mistakes for the greedy algorithm, given realizability, is bounded by one less than the size of the hypothesis class.

The overall strategy used for finding this performance domain can be applied in other scenarios, and is summarized as follows:

1. Find a potential function (size of version space)
2. Find lower bound for potential
3. Find upper bound for potential
4. Combine bounds
5. Simplify using algebra to get a performance bound

This will be useful in future lectures for finding performance bounds on other algorithms.

References

[1] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *conference paper at ICLR*, 2015.

[3] S. Rakhlin. Online learning lecture. Technical report, Massachusetts Institute of Technology MA, 2008.

3 Appendix

3.1 Examples of Algorithms for Various Feedback Methods

Exhaustive	Sampled
Look-up-table	Supervised Learning
Expert Systems	Reinforcement Learning
PWEA	Imitation Learning
Multi-Armed Bandit	Online Linear Classification
Planning	Online-Convex Optimization
Tabular Reinforcement Learning	Contextual Bandit
	Max-Margin & Boosting

Table 1: Exhaustive Feedback and Sampled Feedback Algorithm Examples

Instructive	Evaluative
Supervised Learning	Reinforcement Learning
Cost Sensitive Learning	Cost Sensitive Learning
PWEA	Imitation Learning
Online Linear Classification	Bandit Learning
Max-Margin & Boosting	Online-Convex Optimization
	Planning

Table 2: Instructive Feedback and Evaluative Feedback Algorithm Examples

One-shot	Sequence
Online Expert	Reinforcement Learning
Bandit	Tabular Reinforcement Learning
Supervised Learning	Imitation Learning
Online Convex Optimization	Planning
Online Linear Classification	
PWEA	
Max-Margin & Boosting	

Table 3: One-Shot Feedback and Sequence Feedback Algorithm Examples