Lecture 9

Instructor: Haipeng Luo

1 Interval Regret

In the following lectures, we will discuss how to evaluate online learning algorithms using measures that are more challenging than the classic notion of regret and also make more sense when dealing with non-stationary environments. We first focus on one of these measures: *interval regret*, in the general OCO setting for this lecture.

Recall that the classic regret compares the loss of the algorithm to the loss of the best fixed point. One natural question that we have deferred discussing until now is: is the loss of the best fixed point necessarily a good benchmark to compare to? The answer is no, especially not in some non-stationary environments. To see this, consider a simple expert problem with two experts where the first one always suffers loss 0 for the first $T/2$ rounds and loss 1 for the other $T/2$ rounds, and the situation for the second expert is exactly reversed. Then overall the best fixed expert (which is any one of the two experts) has loss $T/2$, and thus even if the regret to the best fixed expert is zero, all we can say is that the loss of the algorithm is bounded by $T/2$, not a very impressive guarantee.

Moreover, this is not just due to lazy analysis that does not give tight enough bounds. One can show that some algorithms with sublinear regret guarantees indeed suffer linear loss $\Omega(T)$ is this case. Take Hedge as an example. Observe that by the algorithm the weight for the first expert is always not smaller than the second one (since the cumulative loss of the first expert is always not larger). This means that for the second $T/2$ rounds, the loss of the algorithm is at least $1/2$, and thus the cumulative loss is at least $T/4$.

Therefore, classic regret is not the always the right objective to minimize, especially in a non-stationary environments where there is no single fixed point that does well overall. To address this issue, we introduce the notion of interval regret. Specifically, we use the notation $\mathcal{I} = [s, e]$ to denote the rounds $\{s, s+1, \ldots, e-1, e\}$ and call it an interval. Then the interval regret with respect to an interval $\mathcal{I}$ is simply (and literally) the regret on this interval

$$\mathcal{R}_{\mathcal{I}} = \sum_{t \in \mathcal{I}} f_t(w_t) - \min_{w \in \Omega} \sum_{t \in \mathcal{I}} f_t(w).$$

In other words, $\mathcal{R}_{\mathcal{I}}$ is comparing the loss of the algorithm on interval $\mathcal{I}$ to the loss of the best fixed point in terms of the cumulative loss on interval $\mathcal{I}$. Imagine we know where the starting point s of the interval $\mathcal{I}$ is, then we would simply run an online learning algorithm with sublinear (regular) regret starting from round s and obtain $\mathcal{R}_{\mathcal{I}} = \mathcal{O}(\sqrt{|\mathcal{I}|})$ (omitting other terms) where we use $|\mathcal{I}|$ to denote the length of interval $\mathcal{I}$. Of course, the challenge is that we do not know what $\mathcal{I}$ is beforehand, or in other words, we want to design an algorithm with interval regret $\mathcal{R}_{\mathcal{I}} = \mathcal{O}(\sqrt{|\mathcal{I}|})$ *simultaneously* for all $\mathcal{I}$. In the literature, such an algorithm is also sometimes called a *strongly adaptive* algorithm.

Going back to the illustrating example discussed above. If we have a strongly adaptive algorithm, then we can conclude that the interval regret for the first $T/2$ rounds and the second $T/2$ rounds are both of order $\mathcal{O}(\sqrt{T})$. More importantly, since the best expert on these two intervals (expert 1 and 2 respectively) both have zero cumulative loss on their respective interval, it means that the cumulative loss of such strongly adaptive algorithm over T rounds is only $\mathcal{O}(\sqrt{T})$, much better than Hedge for example.

2 Sleeping Experts

How should we design strongly adaptive algorithms? It turns out that there is a general mechanism that allows one to turn any algorithm with low (regular) regret into a strongly adaptive algorithm. To introduce this approach, we will need to take a detour and discuss the *sleeping experts* [Freund et al., 1997] problem first.

The sleeping expert problem is a generalization of the expert problem where experts with different expertise can choose to abstain from providing advice for a given round. Formally, at round $t = 1, \dots, T$,

1. the environment first decides (possibly adversarially) which of the N experts are awake and which are asleep: $a_t(i) = 1$ means expert i is awake and $a_t(i) = 0$ means it is asleep;
2. a_t is then revealed to the learner who needs to decide a distribution $p_t \in \Delta(N)$ with the restriction that no weights are put on asleep experts, that is, $p_t(i) = 0$ if $a_t(i) = 0$;
3. the environment reveals the losses for the awake experts, that is, $\ell_t(i)$ for i s.t. $a_t(i) = 1$.

The regular expert problem is clearly a special case with $a_t(i) = 1$ for all t and i. Because an expert is now not necessarily involved in every round of the game, the regret against this expert is naturally defined only in terms of those rounds when the expert is awake:

$$R_T(i) = \sum_{t:a_t(i)=1} \left(\langle p_t, \ell_t \rangle - \ell_t(i)\right).$$

Note that while we use the notation $\ell_t \in [0,1]^N$, some of it coordinates might not be defined since the corresponding experts could be asleep on round t. However, this is not really an issue because p_t is required to put zero weight on those coordinates anyway and thus they make no difference to the inner product $\langle p_t, \ell_t \rangle$.

It is natural to ask for a sleeping experts algorithm with $R_T(i) = \mathcal{O}(\sqrt{|\{t : a_t(i) = 1\}| \ln N})$ for all i. Indeed, this is achievable by reducing the sleeping expert problem to the regular expert problem. Specifically, suppose we are given an regular expert algorithm $\mathcal{E}$ with prediction $\widehat{p}_t$ on round t. To come up with a prediction p_t for the sleeping expert problem so that $p_t(i) = 0$ for those asleep experts, a natural idea is to simply ignore the weights in $\widehat{p}_t$ for asleep experts and renormalize the others, that is, $p_t(i) \propto a_t(i)\widehat{p}_t(i)$.

Next we need to come up with a loss vector as the feedback to $\mathcal{E}$. For the awake experts, it is natural to just use the same loss we observe in the sleeping expert problem. What about the asleep experts? Suppose we assign the same value x to all these asleep experts, with the goal of forcing the loss of $\mathcal{E}$ for this round to be the same as the loss of the sleeping expert algorithm we arrive at an equation

$$\sum_{i:a_t(i)=1} \widehat{p}_t(i)\ell_t(i) + \left(\sum_{i:a_t(i)=0} \widehat{p}_t(i)\right) x = \sum_{i:a_t(i)=1} p_t(i)\ell_t(i).$$

Using the definition of p_t and solving for x give

$$\begin{aligned} x &= \frac{\sum_{i:a_t(i)=1} \left(p_t(i) - \widehat{p}_t(i)\right) \ell_t(i)}{\sum_{i:a_t(i)=0} \widehat{p}_t(i)} \\ &= \frac{\left(\frac{1}{\sum_{i:a_t(i)=1} \widehat{p}_t(i)} - 1\right) \sum_{i:a_t(i)=1} \widehat{p}_t(i)\ell_t(i)}{1 - \sum_{i:a_t(i)=1} \widehat{p}_t(i)} \\ &= \sum_{i:a_t(i)=1} p_t(i)\ell_t(i), \end{aligned}$$

which means the "fake" loss of asleep experts should be exactly the loss of the sleeping expert algorithm! As discussed earlier, the value of $\ell_t(i)$ for $a_t(i) = 0$ does not matter in the sleeping expert problem. We will therefore overload the notation ℓ_t to denote the loss vector for both the regular expert problem and the sleeping expert problem, where $\ell_t(i) = \sum_{i:a_t(i)=1} p_t(i)\ell_t(i) = \langle p_t, \ell_t \rangle$ if $a_t(i) = 0$ and otherwise is the loss revealed by the environment. The complete reduction is shown in Algorithm 1.

Algorithm 1: Reduction from Sleeping Expert to Regular Expert

Input: a regular expert algorithm $\mathcal{E}$
for $t = 1, \dots, T$ **do**
  let $\widehat{p}_t$ be the prediction of $\mathcal{E}$ on round t
  observe a_t from the environment
  play p_t such that $p_t(i) \propto a_t(i)\widehat{p}_t(i)$
  observe $\ell_t(i)$ for i such that $a_t(i) = 1$
  set $\ell_t(i) = \langle p_t, \ell_t \rangle$ for i such that $a_t(i) = 0$
  pass ℓ_t to $\mathcal{E}$

By the reduction, we have

$$R_T(i) = \sum_{t:a_t(i)=1} (\langle p_t, \ell_t \rangle - \ell_t(i)) = \sum_{t=1}^{T} (\langle p_t, \ell_t \rangle - \ell_t(i)) = \sum_{t=1}^{T} (\langle \widehat{p}_t, \ell_t \rangle - \ell_t(i))$$

which is exactly the regret of $\mathcal{E}$ against expert i in the regular expert problem (and therefore we overload the notation $R_T(i)$ too). However, if $\mathcal{E}$ only has regret bound $\mathcal{O}(\sqrt{T \ln N})$ (such as Hedge), then we only obtain $R_T(i) = \mathcal{O}(\sqrt{T \ln N})$ instead of the desired bound $\mathcal{O}(\sqrt{|\{t : a_t(i) = 1\}| \ln N})$. In fact, what we need here is an expert algorithm with adaptive regret bounds such as $R_T(i) = \mathcal{O}(\sqrt{(\sum_t r_t^2(i)) \ln N})$ where $r_t(i) = \langle \widehat{p}_t, \ell_t \rangle - \ell_t(i)$ is the instantaneous regret. Indeed, with such an adaptive regret bound we arrive at

$$R_T(i) = \mathcal{O}\left(\sqrt{\left(\sum_{t=1}^{T} (\langle \widehat{p}_t, \ell_t \rangle - \ell_t(i))^2\right) \ln N}\right) = \mathcal{O}\left(\sqrt{\left(\sum_{t:a_t(i)=1} (\langle p_t, \ell_t \rangle - \ell_t(i))^2\right) \ln N}\right)$$

which is of order $\mathcal{O}(\sqrt{|\{t : a_t(i) = 1\}| \ln N})$. Such an adaptive regret bound is not unfamiliar to us – we have shown that Squint enjoys exactly such bound (and have in fact discussed several interesting consequences of having such adaptive bounds). Plugging the general Squint regret bound, we have for any T and any competitor $q \in \Delta(N)$ (omitting small terms),

$$\mathbb{E}_{i\sim q}[R_T(i)] = \mathcal{O}\left(\sqrt{(\mathbb{E}_{i\sim q}[|\{t : a_t(i) = 1\}|]) \operatorname{KL}(q, \widehat{p}_1)}\right) \tag{1}$$

where $\widehat{p}_1$ is a prior distribution. Note that the term $|\{t : a_t(i) = 1\}|$ can be even improved to the loss of expert i: $\sum_{t:a_t(i)=1} \ell_t(i)$ by the same argument we have used to show "small-loss" bounds for Squint.

As another remark, recall that Squint is completely parameter-free – it does not even need to know the total number of experts N in advance. This sounds strange in the regular expert problem since the algorithm needs to compute a distribution in $\Delta(N)$ and thus of course needs to and will know N. However, it does make more sense in the sleeping expert setting if the total number of experts is not given ahead of time. We will see such an example immediately.

3 Strongly Adaptive Algorithms via Sleeping Expert

We are now ready to introduce a general mechanism to derive strongly adaptive algorithms in the OCO setting given any OCO algorithm $\mathcal{A}$ with regular regret $\mathcal{O}(\sqrt{T})$ for all T.[1] As mentioned earlier, if the interval $\mathcal{I} = [s, e]$ was known, one could simply run $\mathcal{A}$ starting at time s. Now since we want to consider all intervals $\mathcal{I}$, a natural idea is to start a new instance of $\mathcal{A}$ at the beginning of every round and to combine the predictions from different instances to come up with the final prediction.

This can be exactly captured by the sleeping expert problem: each instance of $\mathcal{A}$ is an expert, and the instance that starts at time t (denoted by $\mathcal{A}_t$) is asleep for the first $t-1$ rounds and awake for the

[1] If an algorithm requires knowing T, then a simple doubling trick can make it agnostic to T.

Algorithm 2: Strongly Adaptive Algorithm via Sleeping Expert

Input: a regular OCO algorithm $\mathcal{A}$, a sleeping expert algorithm $\mathcal{S}$
for $t = 1, \dots, T$ **do**
  start a new instance of $\mathcal{A}$, called $\mathcal{A}_t$
  obtain predictions from $\mathcal{A}_1, \dots, \mathcal{A}_t$, denoted by $w_t^1, \dots, w_t^t$
  pass a_t to $\mathcal{S}$ where $a_t(i) = 1$ for $i \le t$ and $a_t(i) = 0$ for $i > t$
  obtain distribution p_t from $\mathcal{S}$
  predict $w_t = \sum_{i=1}^t p_t(i) w_t^i$
  observe loss function f_t, suffer loss $f_t(w_t)$
  pass f_t to $\mathcal{A}_1, \dots, \mathcal{A}_t$
  pass ℓ_t to $\mathcal{S}$ where $\ell_t(i) = f_t(w_t^i)$ for $i \le t$.

rest of the game. The final prediction at round t will be the convex combination of predictions from $\mathcal{A}_1, \dots, \mathcal{A}_t$ according to the distribution decided by a sleeping expert algorithm. See Algorithm 2 for details. By the construction, we have for any $w \in \Omega$ and interval $\mathcal{I} = [s, e]$,

$$\begin{aligned}
\sum_{t \in \mathcal{I}} (f_t(w_t) - f_t(w)) &\le \sum_{t \in \mathcal{I}} \left(\sum_{i=1}^t p_t(i) f_t(w_t^i) - f_t(w) \right) && \text{(Jensen's inequality)} \\
&= \sum_{t \in \mathcal{I}} (\langle p_t, \ell_t \rangle - \ell_t(s)) + \sum_{t \in \mathcal{I}} (f_t(w_t^s) - f_t(w)) \\
&= R_e(s) + \mathcal{O}(\sqrt{|\mathcal{I}|}) && \text{(by the guarantee of } \mathcal{A}) \\
&= \mathcal{O}(\sqrt{|\mathcal{I}| \ln T}) + \mathcal{O}(\sqrt{|\mathcal{I}|}). && \text{(by the guarantee of } \mathcal{S})
\end{aligned}$$

In fact, by using Squint as the sleeping expert algorithm and a special prior $\widehat{p}_1(i) \propto 1/i^2$, we can improve the first term to $\mathcal{O}(\sqrt{|\mathcal{I}| \ln s})$ since

$$\mathrm{KL}(q, \widehat{p}_1) = \ln \left(\frac{1}{\widehat{p}_1(s)} \right) = \ln \left(s^2 \sum_{i=1}^\infty \frac{1}{i^2} \right) = \mathcal{O}(\ln s).$$

Note that this is a concrete example where the total number of the experts is unknown ahead of time and keeps increasing, but it is clear that Squint can be run without any trouble.

Finally we discuss computational efficiency of Algorithm 2. Since we need to maintain t instances of $\mathcal{A}$ and thus t experts at time t, the running time per round is $\mathcal{O}(t)$, which is not very efficient and keeps increasing. Fortunately, it turns out that one can significantly improve the running time to $\mathcal{O}(\ln t)$ without sacrificing any regret guarantee. The idea is to kill some instances when they have lived long enough in some sense so that at each time there are only $\mathcal{O}(\ln t)$ instances alive. Killing an instance can be easily incorporated in the sleeping expert model by just putting the corresponding expert to sleep forever. The key is to do this in a careful way so that the regret is almost not affected.

There are in fact many different ways to do this. One simple approach taken from [Hazan and Seshadhri, 2007] is to let $\mathcal{A}_t$ live for $2^{d(t)}$ rounds where $d(t)$ is the largest integer such that $t = b(t) \times 2^{d(t)}$ for some (odd) integer $b(t)$. In other words, $d(t)$ is the number of 2's in t's prime factorization. (Try drawing a picture to see what the awake intervals are like for the first few (say 20) instances.)

To see why this leads to a more efficient algorithm, first note that at any time t and any integer d, there is at most one expert with lifetime 2^d awake (think about why). On the other hand, at time t the longest lifetime of any awake experts is clearly bounded by $2^{\lfloor \log_2 t \rfloor}$. Therefore, at any time t the total number of awake experts is at most $\lfloor \log_2 t \rfloor + 1$, meaning that the running time of the algorithm is only $\mathcal{O}(\ln t)$ per iteration.

Next we argue that the regret is almost not affected. For an interval $\mathcal{I} = [s, e]$, since $\mathcal{A}_s$ does not necessarily live until the end of this interval and we thus cannot just compare to $\mathcal{A}_s$ as before, it is natural to divide the interval into several disjoint and consecutive subintervals and compare to different instances of $\mathcal{A}$ on these subintervals. Formally, let $\mathcal{I}_m = [s_m, e_m]$ for $m = 1, \dots, M$ be

these subintervals where $s_1 = s, s_m = e_{m-1} + 1$ for $1 < m \leq M$, $e_m = s_m + 2^{d(s_m)} - 1$ for $1 \leq m < M$ and $e_M = e$. Clearly for each $\mathcal{I}_m$ $(m < M)$, there is an instance $(\mathcal{A}_{s_m})$ that is run solely on this interval. More importantly, the length of these intervals is doubling in the sense that $2|\mathcal{I}_m| \leq |\mathcal{I}_{m+1}|$ for $1 \leq m < M - 2$. This is because

$$s_{m+1} = e_m + 1 = s_m + 2^{d(s_m)} = (b(s_m) + 1) \times 2^{d(s_m)} = \frac{b(s_m) + 1}{2} \times 2^{d(s_m)+1}$$

where $\frac{b(s_m)+1}{2}$ has to be an integer since $b(s_m)$ is odd. This implies that $d(s_{m+1}) \geq d(s_m) + 1$ and thus $2|\mathcal{I}_m| \leq |\mathcal{I}_{m+1}|$. As a result, we have

$$\begin{aligned}
\sum_{t\in\mathcal{I}}(f_t(w_t) - f_t(w)) &\leq \sum_{t\in\mathcal{I}}\left(\sum_{i=1}^{t} p_t(i) f_t(w_t^i) - f_t(w)\right) && \text{(Jensen's inequality)} \\
&= \sum_{m=1}^{M}\sum_{t\in\mathcal{I}_m}(\langle p_t, \ell_t\rangle - \ell_t(s_m)) + \sum_{m=1}^{M}\sum_{t\in\mathcal{I}_m}(f_t(w_t^{s_m}) - f_t(w)) \\
&= \sum_{m=1}^{M} R_{e_m}(s_m) + \sum_{i=1}^{M}\mathcal{O}(\sqrt{|\mathcal{I}_m|}) && \text{(by the guarantee of } \mathcal{A}) \\
&= \sum_{m=1}^{M}\mathcal{O}(\sqrt{|\mathcal{I}_m|\ln T}) && \text{(by the guarantee of } \mathcal{S}) \\
&\leq \sum_{i=0}^{\infty}\mathcal{O}(\sqrt{2^{-i}|\mathcal{I}|\ln T}) \\
&= \mathcal{O}(\sqrt{|\mathcal{I}|\ln T}),
\end{aligned}$$

giving the same result (up to constants) as before.

References

Yoav Freund, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. Using and combining predictors that specialize. In *29th Annual ACM Symposium on the Theory of Computing*, pages 334–343. ACM, 1997.

Elad Hazan and C. Seshadhri. Adaptive algorithms for online decision problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14, 2007.