On Regrets: From External Regret to Average Reward Regret

Alireza Masoumian AMASOUMI@UALBERTA.CA
Department of Computing Science, University of Alberta / Amii

In online decision-making, the performance of the algorithm or the agent is usually measured by a notion of regret. We see regret in Online learning, Bandits, and (online) Reinforcement learning as they are the frameworks for modeling online decision-making. There are different notions, although they usually capture the same logic which is *measuring the difference between what is done and a benchmark for acting in hindsight*. Assuming T rounds of taking actions, we denote the regret of an algorithm (let's call it Alg) by $\mathsf{Reg}(T)$. Usually having zero regret is impossible (unless you know the future!), so with a little bit of compromise, we define no-regret algorithms to be the ones that have sub-linear regret in time, i.e. $\mathsf{Reg}(T) \in o(T)$. In this post, we go over different notions of regret.

1. External Regret

This is the most common notion and at the same time one of the simplest ones. External regret appears in Bandit an Online learning literature. In words, it compares the algorithm performance to taking the best fixed action in hindsight. Suppose at each iteration $t \in [1 : T]$, the algorithm takes an action $a_t \in \mathcal{A}$ and the environment, stochastically or adversarially, chooses $u_t : \mathcal{A} \to \mathbb{R}$ as a function that determines the reward of taking each action. The external regret is defined as,

$$\mathsf{Reg}_E(T) = \max_{a \in \mathcal{A}} \sum_{t=1}^{T} u_t(a) - \sum_{t} u_t(a_t)$$

There are many algorithms developed for this notion of regret, like Explore then Commit (ETC), or mirror descent-based algorithm (FTRL, EXP3, MWU). I will frequently use *competitors set* in this post, and it refers to the domain on which we take the maximization. For instance in external regret, the competitors' set is the fixed actions Alg could have taken. In fact, the algorithm competes with the alternatives of this set to have low regret. External regret is weak enough for having sublinear regret even if u_ts are taken by a non-oblivious adversary (the choice of u_t is based on the history up to t). See (Lattimore and Szepesvári, 2020) for the algorithm.

Another thing is that "Feedback is important, but for the algorithm!" You might have a full information (knowing u_t after t) or Bandit information (knowing only the entry of $u_t(a_t)$) feedback and it affects your regret bound, but in this post, we want to introduce different notions of regret, which is not that relevant to the feedback.

1.1. Action Modifiers

One can imagine $|\mathcal{A}| = \boldsymbol{A}$ many functions $\phi_1, \ldots, \phi_{|\mathcal{A}|}$, where each $\phi_i : \mathcal{A} \to \mathcal{A}$ is an action modifier, and ϕ_i maps all the actions in $\mathcal{A}$ to the single action of $i \in \mathcal{A}$, i.e. $\phi_i(a) = i$ for all $a \in \mathcal{A}$. Put all these functions in a set $\mathcal{E} = \{\phi_1, \ldots, \phi_{\boldsymbol{A}}\}$. Now recall the set of competitors, $\mathcal{E}$ actually

represents the set of competitors for the external regret. We can rewrite the external regret definition as follows,

$$\mathsf{Reg}_E(T) = \max_{\phi_i \in \mathcal{E}} \sum_{t=1}^{T} u_t(\phi_i(a_t)) - \sum_t u_t(a_t).$$

You might think "why making that complicated!?", but we need this to generalize external regret to some stronger notions.

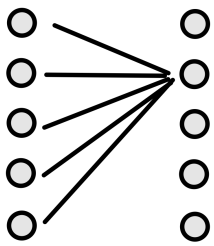

Figure 1: Action modifiers for external regret

2. Internal Regret

Consider another set of action modifiers $\mathcal{I} = \{\phi_{ij} : i, j \in \mathcal{A}\}$ such that

$$\begin{cases} \phi_{ij}(i) = j \\ \phi_{ij}(k) = k \quad \forall k \neq i. \end{cases}$$

In other words, $\mathcal{I}$ contains $\boldsymbol{A}^2 - \boldsymbol{A} + 1$ functions like $\phi_{ij} : \mathcal{A} \to \mathcal{A}$ that maps i to j and it is identical anywhere else. Using $\mathcal{I}$ as the set of competitors we obtain the internal regret,

$$\mathsf{Reg}_I(T) = \max_{\phi_i \in \mathcal{I}} \sum_{t=1}^{T} u_t(\phi_i(a_t)) - \sum_t u_t(a_t).$$

Note that $\mathcal{E} \not\subset \mathcal{I}$, which means for a fixed trajectory of actions, $a_{1:T}$, and utilities, $u_{1:T}$, Reg_I is not necessarily greater than Reg_E. However, a no-internal regret algorithm is necessarily no-external regret too. We see why this is the case by introducing swap regret.

3. Swap Regret

By setting the competitors set to be the set of all functions from $\mathcal{A}$ to itself, i.e. $\mathcal{S} = \mathcal{A}^{\mathcal{A}}$, we have swap regret,

$$\mathsf{Reg}_S(T) = \max_{\phi_i \in \mathcal{S}} \sum_{t=1}^{T} u_t(\phi_i(a_t)) - \sum_t u_t(a_t).$$

Obviously $\mathcal{E} \subset \mathcal{S}$ and $\mathcal{I} \subset \mathcal{S}$, meaning swap regret is a stronger notion than both external and internal. But there is an interesting connection between internal and swap regret.

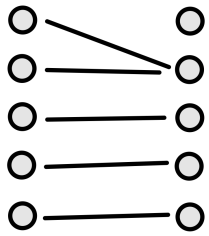

Figure 2: Action modifiers for internal regret

Statement 3.1 *Every no-internal regret algorithm is also a no-swap regret algorithm (and thus no-external regret).*

This result shows that the internal action modifiers in $\mathcal{I}$ are sufficient representatives for all action modifiers, in the sense that if Alg can compete with $\mathcal{I}$, then it also enjoys a sub-linear regret in the competition with $\mathcal{S}$. Also, note that the size of $\mathcal{I}$ is exponentially smaller than $|\mathcal{S}| = \boldsymbol{A}^{\boldsymbol{A}}$ in $\boldsymbol{A}$. Now you might be looking forward to seeing a no-internal regret algorithm! Another surprising result shows that they are not that out of reach! According to Blum-Mansour reduction, we know

Statement 3.2 *Every no-external regret algorithm can be efficiently converted to a no-internal regret algorithm.*

Blum and Mansour (2007) obtained a swap regret in $O(\sqrt{T\boldsymbol{A}\log(\boldsymbol{A})})$ for full-information feed back. This well-taught lecture by Tim Roughgarden (link) for proof. After tasting the beauty of this idea, you might think "Is it possible to drop that linear dependency on $\boldsymbol{A}$!?". Another important recent work (Dagan et al., 2024) shows that this can be done, at the cost of having exponential dependence on the error ϵ.

When generally a set of actions modifiers denote by of Φ is considered, the induced regret is called Φ regret(Greenwald and Jafari, 2003).

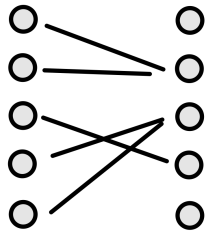

Figure 3: Action modifiers for swap regret

4. Dynamic Regret

In this section, we introduce dynamic regret which is stronger than all the previous notions presented above,

$$\mathsf{Reg}_D(T) = \sum_{t=1}^{T} \max_{a \in \mathcal{A}} u_t(a) - \sum_t u_t(a_t).$$

In dynamic regret, for each iteration t, we (dynamically!) take maximum over the actions. This means the value of the dynamic regret can be *incrementally* computed, while this is not the case for previous notions for which a high-level view over the whole T iteration is needed. Sub-linear dynamic regret is achievable in stochastic settings but, adversarial choices of u_t might guarantee regret in $\Omega(T)$. The value of all previous notions of regret, even the swap regret, may be negative for some trajectories of $a_{1:T}$ and $u_{1:T}$ (think why!). However, dynamic regret is always positive.

5. Adaptive Notions of Regret

So far we have not assumed that taking action a_t will affect its following utility functions. On the other hand, when there is a planning effect for taking actions, the learner might take actions with low immediate rewards, with the hope of getting better utilities in the future. Reinforcement learning captures this point by bringing states and transition kernel into the picture. The effect of previous actions is encoded in the current state s_t which is (usually) observable for the learner at time t. Then the reward function with the inputs of s_t and a_t determines the utility at time t, i.e. $u_t = r(s_t, a_t)$. The adaptivity is what the transition P makes between the states, $\mathbb{P}(S_{t+1} = s_{t+1} | A_t = a_t, S_t = s_t) = P_{a_t}(s_t, s_{t+1})$. Now we can define our competitor in this setting (the horizon is denoted by H instead of T in this part for some reason!),

$$V_H^*(s_1) = \max_{a_1^*, \dots, a_H^*} \sum_{t=1}^{H} r(s_t, a_t^*). \tag{1}$$

$V_H^*(s_1)$ is the maximum cumulative utility that one can achieve in this adaptive setting starting from state s_1. Note that in this horizon, you might have ups and downs, sweet and bitter days, high and low immediate rewards, but the whole sequence of actions $a_1^*, \dots, a_T^*$, is designed to give you the highest cumulative reward. Now the question is "How can I get this well-tuned sequence of actions?". The fundamental theorem of RL states that,

Statement 5.1 *(Fundamental theorem) In a* Markov *Decision Process (when the random variables of the current reward and the next state only depend on the current state and action, and not further in the past), the current state is all you need to know to take the optimal actions.*

To say it a bit more fancy, there exists a memory-less deterministic optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ such that $V_H^*(s_1) = \sum_{t=1}^{H} r(s_t, \pi^*(s_t)) := V_H^{\pi^*}(s_1)$. This means that the domain of maximization in the equation 1, i.e. $\mathcal{A}^H$ can be replaced with the set of policies $\mathcal{S}^A$, which does not grow in H. Note that, defining proper states is crucial. On one hand, it should be rich enough to capture needed information in history to make Markov rewards and transitions, and on the other hand, The bigger the state space is the harder it is to the optimal policy. This is studied in a research thread called *model selection*. Maybe I should have mentioned this earlier, but we do not have any discounting in rewards and this does not make any trouble as we assume a finite horizon. Now we can move on and use this competitor in different notions of regrets.

5.1. Episodic Regret

Episodic regret is a common notion in online reinforcement learning. Suppose there are K episodes, each with the length of H iterations. The episodic regret is defined as

$$\mathsf{Reg}_{Ep}(T) = \sum_{k=1}^{K} [V_H^{\pi^*}(s_1) - V_H^{\pi_k}(s_1)],$$

where π_k is the fixed policy employed by the algorithm in episode $k \in [K]$. This can be a natural evaluation when there is an episodic structure in the environment. This notion still does not fully capture the adaptivity point as it is like adding the sub-optimality of policies used in each episode. At the level of episodes, it is still like external regret (even π^* is optimal for all of the episodes which is not necessarily the case for external regret.) The whole number of iterations in this interaction is $T = HK$. The other point is that the algorithm can not change the policy within an episode. Check out (Zhang et al., 2024) which achieves the optimal episodic regret in $\tilde{O}(\sqrt{SAH^3K})$ without any burn-in iterations.

5.2. Adaptive Dynamic Regret

Suppose an episodic regret with a planning horizon of $H = 1$ (i.e., $K = T$), where the initial state of the episode $k > 1$ is the state that the transition kernel suggests based on the last state and action in the previous one, i.e. s_{k-1} and a_{k-1}. In this case, you can change the policy at every iteration. Furthermore, $V_1^{\pi^*}(s_k)$ coincides with the highest immediate reward. This gives us the notion of *adaptive dynamic regret* (I guess this is not a common name in literature though!),

$$\mathsf{Reg}_{AD}(T) = \mathbb{E}\big[\sum_{k=1}^{T} V_1^{\pi^*}(S_k) - V_1^{\pi_k}(S_k)\big] \tag{2}$$

$$= \mathbb{E}\big[\sum_{t=1}^{T} \max_{a^*} r(a^*, S_t) - r(A_t, S_t)\big] \tag{3}$$

We emphasize *adaptive* dynamic regret since the current action a_t not only determines the current reward, but also affects the future utility functions through the transition kernel.

5.3. Average Reward Regret

Note that the competitor in adaptive dynamic regret is not equal to the optimal value, exactly because of the the planning effect of the actions,

$$\mathbb{E}\big[\sum_{t=1}^{T} \max_{a^* \in \mathcal{A}} r(a^*, S_t)\big] \leq \max_{a^*_{1:T} \in \mathcal{A}^T} \mathbb{E}\big[\sum_{t=1}^{T} r(a^*_t, S_t)\big] = V_T^*(s_1).$$

So if we use $V_T^*(s_1)$ as the competitor, we will have,

$$\mathsf{Reg}_V(T) = V_T^*(s_1) - \mathbb{E}\big[\sum_{t=1}^{T} r(A_t, S_t)\big],$$

which is stronger than adaptive dynamic regret due to the above inequality. In case of having no planning effect for the actions, like when the states suggested by transition kernel are independent of the actions (contextual bandit), Reg_V coincides with Reg_{AD}. The regret Reg_V can be interpreted as an episodic regret for a single episode plus the ability to change the policy in each interaction. But Reg_V is not actually the average reward regret. There is a quantity called *gain* in the average reward reinforcement learning, which is the asymptotic average reward that a policy gains, i.e. $g^\pi(s_1) := \lim_{T\to\infty} \frac{1}{T} V_T^\pi(s_1)$. The optimal policy then is defined as $g^*(s_1) := \max_\pi g^\pi(s_1)$. In weakly communicating MDPs (Puterman, 2014), the impact of the initial state fades away in optimal policy (not for all policies), so $g^*(s_1)$ all have the same value for different s_1. So we can use a scalar g^* instead. The Average reward regret is defined as,

$$\mathsf{Reg}_{AR}(T) := Tg^* - \mathbb{E}\big[\sum_{t=1}^{T} r(A_t, S_t)\big].$$

The gap between Reg_{AR} and Reg_V, i.e. $|Tg^* - V_T^*(s_1)|$, is actually upper bounded by a constant (for curious readers the constant is $2sp(h^*)$ where h^* is the optimal bias). Thus these two notions of regret are very close to each other. From a This is the notion that is used in the celebration work by Auer et al. (2008), which introduces UCRL2 algorithm and the *doubling trick* to make some inner epochs for this case that we do not have explicit episodes. Recently, another nice paper has proposed an efficient optimal algorithm guaranteeing an average reward regret in the order of $\tilde{O}(\sqrt{sph(h^*)SAT})$ (Boone and Zhang, 2024). The ticket in their work is modified Extended Value Iteration (EVI) by cascading a projection to the Bellman operator. Another observation is that the values of Reg_V, Reg_{AD} and Reg_{Ep} are always non-negative, while Reg_{AR} might be negative (but not that much!, $\mathsf{Reg}(AR) \geq -2sp(h^*)$).

5.4. Policy Regret

In the previous parts, the effect of current action on the future utilities was put on the shoulders of the states and transition kernel. But this is not the only way of modeling adaptivity. One can directly assume that the utility function maps the joint previous and current actions to a real number, i.e. $u_t : \mathcal{A}^t \to \mathbb{R}$. This modification leads us to the policy regret,

$$\mathsf{Reg}_P(T) = \max_{(a_1^*,\dots,a_T^*)\in\mathcal{C}_T} \sum_{t=1}^{T} u_t(a_{1:t}^*) - \sum_{t=1}^{T} u_t(A_{1:t}),$$

where $a_1, \dots, a_t$ is abbreviated by $a_{1:t}$. Also, It is assumed a competitor set $\mathcal{C}_T$ which the counterfactual sequences of actions belong to. Even for a simple $\mathcal{C}_T$, having no-policy regret algorithms is not possible (Arora R, 2012), i.e.

Statement 5.2 *For any Algorithm, and for $\mathcal{C}_T$ being the set of constant sequences of form $(a, \dots, a)$ for all $a \in \mathcal{A}$, there exists a sequence of utilities $u_{1:T}$ such that $\mathsf{Reg}_P \in \Omega(T)$.*

However, the policy regret would not be that hopeless if we added some limitations on utility functions. Assume that there is a limited memory for the utility functions, i.e. only the last m actions determine the current utility. In other words, $u_t : \mathcal{A}^m \to \mathbb{R}$. In this case, achieving sub-linear regret is feasible, usually with an exponent larger than $\frac{1}{2}$ for T (See (Arora R, 2012)).

6. Game Theoretic Concerns

There are very nice connections between these notions of regret, and different notions of equilibria. Just in brief, suppose the underlying procedure is a repeated game, meaning the actions are pure strategies of a stage game, and the utilities for the learner come from joint actions chosen by all players. In this case, averaging over the actions that a no-external regret algorithm takes over time, converges to the course correlated, and the if you do the same for a no-swap regret algorithm it converges to correlated equilibrium. Adding some constraints also leads us to the well-known Nash equilibrium(Roughgarden, 2016). But one might point out that the agents are usually *utility maximize* and not *regret minimizer*! These two might be really different depending on the notion of regret (Brown et al., 2024). In adaptive notions of regret, the average reward regret is the most fitted regret notion for *utility maximization* as it compares the performance of the algorithm to the best utility-collecting procedure, even with considering the planning effect of the actions. This planning is common in repeated games especially if we want to capture that the other players (also) take actions in response to the actions that the learner takes. Finally, based on policy regret, policy equilibrium is also developed (Arora et al., 2018).

There are many interesting problems to work on in this area, such as ”How is having some information about the opponents' algorithm exploitable for the learner?” or ”How can we have game-dependent regret bounds?” and many others.

REFERENCES

Arora, R., Dinitz, M., Marinov, T. V., and Mohri, M. (2018). Policy regret in repeated games. *Advances in Neural Information Processing Systems*, 31.

Arora R, Dekel O, T. A. (2012). Online bandit learning against an adaptive adversary: from regret to policy regret. *arXiv preprint arXiv:1206.6400*.

Auer, P., Jaksch, T., and Ortner, R. (2008). Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21.

Blum, A. and Mansour, Y. (2007). From external to internal regret. *Journal of Machine Learning Research*, 8(6).

Boone, V. and Zhang, Z. (2024). Achieving tractable minimax optimal regret in average reward mdps. *arXiv preprint arXiv:2406.01234*.

Brown, W., Schneider, J., and Vodrahalli, K. (2024). Is learning in games good for the learners? *Advances in Neural Information Processing Systems*, 36.

Dagan, Y., Daskalakis, C., Fishelson, M., and Golowich, N. (2024). From external to swap regret 2.0: An efficient reduction for large action spaces. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1216–1222.

Greenwald, A. and Jafari, A. (2003). A general class of no-regret learning algorithms and game-theoretic equilibria. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, pages 2–12. Springer.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Roughgarden, T. (2016). *Twenty lectures on algorithmic game theory*. Cambridge University Press.

Zhang, Z., Chen, Y., Lee, J. D., and Du, S. S. (2024). Settling the sample complexity of online reinforcement learning. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 5213–5219. PMLR.