Project 1: The Weighted Majority Algorithm

Dawei Wang (daweiwan@andrew.cmu.edu)

1. **Realizability**: there exists a hypothesis $h^* \in \mathcal{H}$ such that for any $t = 1, 2, \ldots,$ $h^*(\mathbf{x}^{(t)}) = y^{(t)}$. This assumption provides performance guarantees for the online learner. ■

2. **Hypothesis class**: the set of all mappings from the set of observations $\mathcal{X}$ to the set of outcomes $\mathcal{Y}$. It is finite. It provides an upper bound on the number of mistakes that a learner can make. ■

3. Since we only listen to the *expert* with the fewest mistakes, if the *best* expert has made m^* mistakes, 1) any expert must have made at least m^* mistakes, 2) any expert that has made at least $m^* + 1$ mistakes will never be selected for making another prediction, and it is therefore impossible for them to make more mistakes. Hence, an upper bound for the number of mistakes M is

$$m^* + (N-1)(m^*+1) = N \cdot m^* + (N-1), \tag{1}$$

assuming the worst case, where all experts except for the best expert has made $m^* + 1$ mistakes. ■

4. (1) The optimal μ provides the tightest upper bound for the expected regret. By Jensen's inequality,

$$\mu m^* + \frac{\ln N}{\mu} \geq 2\left(\mu m^* \cdot \frac{\ln N}{\mu}\right)^{-1} \tag{2}$$

the equality holds if and only if $\mu m^* = \mu^{-1} \ln N$, therefore the optimal value $\mu = \sqrt{m^{-*} \ln N}$.

(2) In this case we can eliminate (or devalue the opinions of) experts more frequently without having to worry about running out of experts. Therefore, a large μ should be selected.

(3) If $m^* = O(T)$, for any finite μ,

$$\lim_{T\to\infty} \frac{E(R)}{T} = \lim_{T\to\infty}\left[\mu \cdot \frac{O(T)}{T} + \frac{1}{T}\frac{\ln N}{\mu}\right] = \mu \cdot O(1) \tag{3}$$

To approach the criterion for a *no-regret* learner as much as possible, μ should be small. However, if m^* is sub-linear in T, the first term would most likely to converge to zero quickly, regardless of the choice of μ; for the second term, since μ is in the denominator, selecting a larger μ would be helpful to enable the average regret to converge to zero quicker. ■

5. (1) Since the adversary knows the experts' predictions and their weights, the prediction made by the deterministic learner can be derived; the adversary simply selects the label opposite to that prediction, such that the learner is guaranteed to make one mistake every round.

(2) For randomized weighted majority, the largest expected loss is 0.5 mistakes per round. Basically, the adversary sets $y_t = 1$ if the expected label $\tilde{y}_t = \sum_i \hat{y}_i w_i / \sum_i w_i$ is less than a half, or $y_t = 0$ otherwise. In this way, the expected loss per round is, for $\tilde{y}_t < 0.5$ and $y_t = 1$,

$$\frac{1 \cdot \sum_{i,\hat{y}_i=0} w_i + 0 \cdot \sum_{i,\hat{y}_i=1} w_i}{\sum_i w_i} = \frac{\sum_i (1-\hat{y}_i) w_i}{\sum_i w_i} = 1 - \frac{\sum_i \hat{y}_i w_i}{\sum_i w_i} > 0.5 \tag{4}$$

and symmetrically, in the case that $\tilde{y}_t \geq 0.5$, $y_t = 0$:

$$\frac{0 \cdot \sum_{i,\hat{y}_i=0} w_i + 1 \cdot \sum_{i,\hat{y}_i=1} w_i}{\sum_i w_i} = \frac{\sum_i \hat{y}_i w_i}{\sum_i w_i} \geq 0.5 \tag{5}$$

therefore, with this strategy, the learner is only statistically guaranteed to make 0.5 mistakes per round, half as many as that made by a deterministic learner. Essentially randomization creates a probability that the label y_t deterministically computed by the adversary may be the same as the prediction $\hat{y}_t$, therefore lowering the expected loss by being less likely to make mistakes. ■

The programming part revolves around three classes of elements: the experts (`expert.py`), the natures (`nature.py`), and the learners (`learner.py`). Instances of the same class share a common interface but may have different traits in decision making. Here is a brief introduction:

Natures There are four natures. All of them generate observations that consist of

- `match_number`: $1, 2, 3, \ldots$, incremented each time the nature produces an observation;
- `weather`: either `sunny` or `rainy`, selected uniformly at random;
- `time`: either `afternoon` or `evening`, selected uniformly at random;
- `location`: either `home` or `away`, selected uniformly at random.

However, different natures have different policies for generating the label:

- `NaiveNature`: either `true` or `false`, selected uniformly at random (stochastic);
- `TrialNature`: `true` if `match_number` is not a multiple of three, `false` otherwise (deterministic);
- `ChaoticNature`: `true` if the expected outcome derived by applying the *weighted majority algorithm* on the weights and the predictions obtained from the experts associated with the learner leans towards `false`, and `false` otherwise (adversarial);
- `LawfulNature`: `true` if the *score* obtained with a weighted vote from `location`, `home`, and `time`, with weights 0.45, 0.35, and 0.20 for being `home`, `evening`, `sunny` respectively, is greater than or equal to 0.5, or `false` otherwise.

Experts There are seven experts. They have different policies for making predictions:

- `NaiveExpert`: either `true` or `false`, selected uniformly at random;
- `OptimisticExpert`: always `true`;
- `PessimisticExpert`: always `false`;
- `DualExpert`: `true` if `round_number` is not a multiple of two, `false` otherwise;
- `WeatherExpert`: `true` if `weather` is `sunny`, `false` otherwise;
- `TimeExpert`: `true` if `time` is `evening`, `false` otherwise;
- `HomeAdvantageExpert`: `true` if `location` is `home`, `false` otherwise.

Learners There are three learners. They treat the experts differently:

- `NaiveLearner`: chooses from `true` and `false` uniformly at random as its final prediction;
- `WeightedMajorityLearner`: uses the *weighted majority* algorithm to combine experts' predictions;
- `RandomizedWeightedMajorityLearner`: uses the *randomized weighted majority* algorithm.

`>>>` `main.py` is the script that connects all these elements, which essentially,

1. Instantiates the nature and the learner of our choice,
2. Lets the learner hire the experts of our choice,
3. Iterates for `round_number`-many times,
   (a) Lets the nature produce an observation;
   (b) Feeds the observation to the learner;
   (c) Lets the nature produce the true label, which may peek at the learner if adversarial;
   (d) Feeds the true label back to the learner, which then updates the weights;
   (e) Records the regrets and the loss for this round;
4. Computes the cumulative regrets and losses,
5. Plots the cumulative regrets and losses. ■

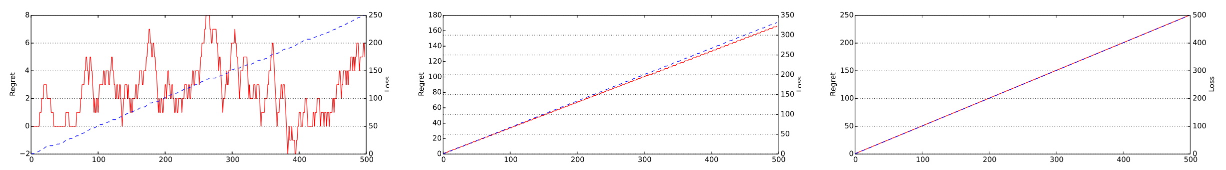


Figure 1: Weighted Majority versus `NaiveNature`, `TrialNature`, and `ChaoticNature`
- - - losses, — regrets, penalty = 0.50

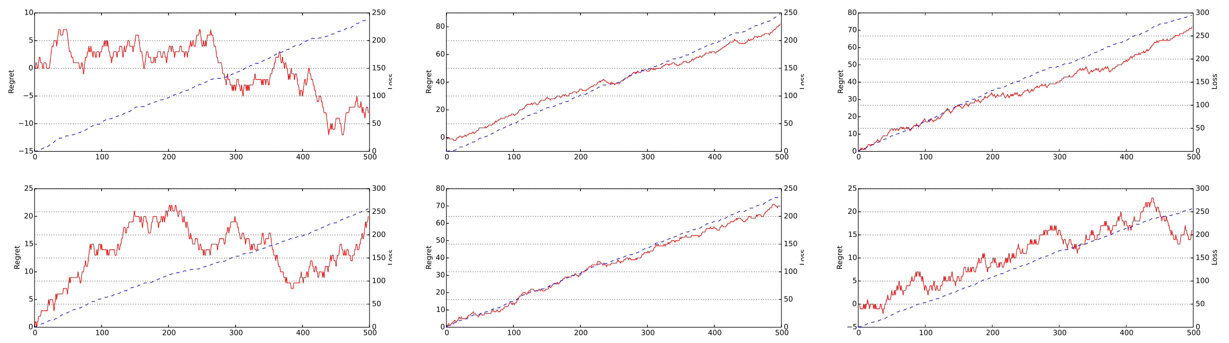


Figure 2: Randomized Weighted Majority versus `NaiveNature`, `TrialNature`, and `ChaoticNature`
- - - losses, — regrets, penalty = 0.50 (top) and 0.20 (bottom)

Section 3.3 See Figure 1. The cumulative losses keep increasing since the learner does not have the expert resources to fully represent the policy adopted by the nature to generate its labels.

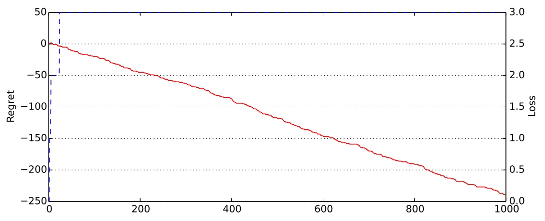
- `NaiveNature`: the expected loss is just half of the round number, since the nature generates the label uniformly at random. This seems to be the case regardless of the experts hired by the learner. The regret walks randomly around zero since neither the learner or the experts have any idea of how the nature is generating the labels, let alone being able to outperform each other.
- `TrialNature`: the nature is producing a deterministic sequence of `true`, `true`, `false`, `true`, `true`, `false`, . . . , and the learner will make one mistake when the outcome flips, since it *incorrectly* adjusts the weights to accommodate the previous mistake while the situation is actually alternating, which accounts for two thirds of the time. The regret grows one third as fast as the round number because the `OptimisticExpert` is correct for two thirds of the time.
- `ChaoticNature`: the nature in this case *correctly* assumes that the learner has adopted the *weighted majority* algorithm and is consequently capable of producing exactly the opposite label each single time, guaranteeing that the learner will make one mistake each round. However, if the learner has instead been listening to either the `PessimisticExpert` or the `OptimisticExpert`, neglecting the weights and other predictions, it will be correct for half of the time, explaining the observation that the regret grows half as fast as the loss. This latter case is true because the nature, although adversarial in nature, fails to see through the policy that the learner has adopted. ■

Section 3.4 See Figure 2. The situation for `NaiveNature` has not changed substantially because the nature has *already* been making decisions uniformly at random. The performance against `TrialNature` has improved: it seems that the expected loss is half of the round number; however, the regret is still positive since the learner is still outperformed by the `OptimisticExpert`. As for the `ChaoticNature`, the learner has been able to perform much better, since the label deterministically computed by the nature cannot always be the exact opposite of a stochastic prediction anymore. The penalty does not affect the results that much, except for the third nature, where a smaller penalty is preferred, presumably because the learner is less likely to be misled to penalize its experts by the adversarial nature. ■

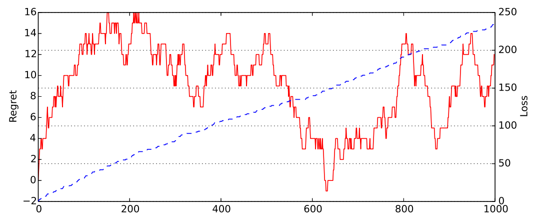
Section 3.5 The following experts have been added to the pool:

- `WeatherExpert`, and
- `TimeExpert`, and
- `HomeAdvantageExpert`;

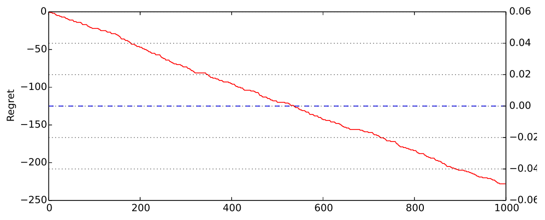
additional features include `weather`, `time`, and `location`, as described earlier. A new deterministic nature, `LawfulNature` has also been introduced, which computes the label *lawfully* with a majority vote from the aforementioned three factors. The idea is that although each expert may not be able to predict the outcome prefectly, but with the *weighted majority* algorithm, the learner should be able to combine the opinions of the experts with their associated weights converging to the true weights. See Figure 3.



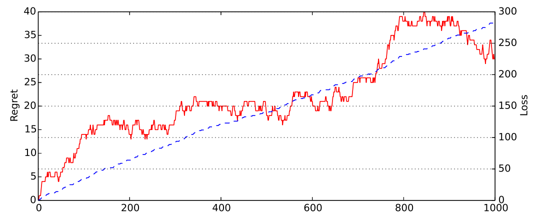
(a) 6 Experts + Weighted Majority



(b) 6 Experts + Randomized Weighted Majority



(c) 3 Experts + Weighted Majority



(d) 3 Experts + Randomized Weighted Majority

Figure 3: Learners versus `LawfulNature`, with All Experts or New Experts Only
- - - losses, — regrets, penalty = 0.50

The result is very interesting. The learner that uses the *weighted majority* algorithm hardly makes any mistakes, largely thanks to the fact that the initial weights are more or less close to the true weights, and also that both the outcome and the loss are binary. This actually makes perfect sense. However, the stochastic learner seems to be performing much worse, presumably due to the fact that the learner randomly deviates from the initial weights and then gets lost, since it only receives one feedback for all the experts it has hired, and it is therefore hard to discriminate between them. Nonetheless, generally speaking, since the number of rounds has been increased to 1000, a cumulative regret of roughly two digits is quite impressive, in comparison to the dumb experts and shrewd natures we had earlier. ■