Tracking the best Expert

Yoav Freund

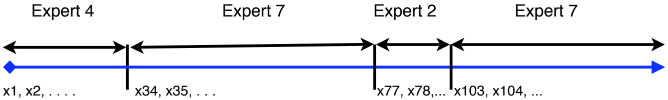
February 7, 2025

Based on “Tracking the best linear predictor” and “Tracking the best expert” by Herbster and Warmuth. Also, section 11.5 in Prediction learning and Games.

Tracking Linear Experts

- Usually: compare algorithm's total loss to total loss of the best expert.
- drifting experts: Compare with a sequence of experts that change over time.
- The amount of change is measured using total bregman divergence.
- Regret depends on $\sum_t \Delta_F(\mathbf{u}_{t-1}, \mathbf{u}_t)$
- The Projection Update After computing the unconstrained update, project the $\mathbf{w}_{t+1}$ onto a convex set.
- Does not allow the algorithm to over-commit to an extreme vector from which it is hard to recover.

Switching experts setup

- Usually: compare algorithm's total loss to total loss of the best expert.
- Switching experts: compare algorithm's total loss to total loss of best expert sequence with k switches.
- 

An inefficient algorithm

- Fix:
  - l - sequence length
  - k - number of switches
  - n - number of experts
- Consider one partition-expert per sequence of switching experts.
- No. of partition-experts : $\binom{l}{k-1} n(n-1)^k = O\left(n^{k+1}\left(\frac{el}{k}\right)^k\right)$
- The log-loss regret is at most $(k+1)\log n + k \log \frac{l}{k} + k$
- Requires maintaining $O\left(n^{k+1}\left(\frac{el}{k}\right)^k\right)$ weights.

generalization to mixable losses

- In this lecture we assume loss function is mixable.
- There is an exponential weights algorithm with learning rate η that achieves (in the non-switching case) a bound

$$L_A \leq \min_i L_i + \frac{1}{\eta} \log n$$

- Then using the partition-expert algorithm for the switching-experts case we get a bound on the regret $\frac{1}{\eta}\left((k+1)\log n + k \log \frac{l}{k} + k\right)$

Weight sharing algorithms

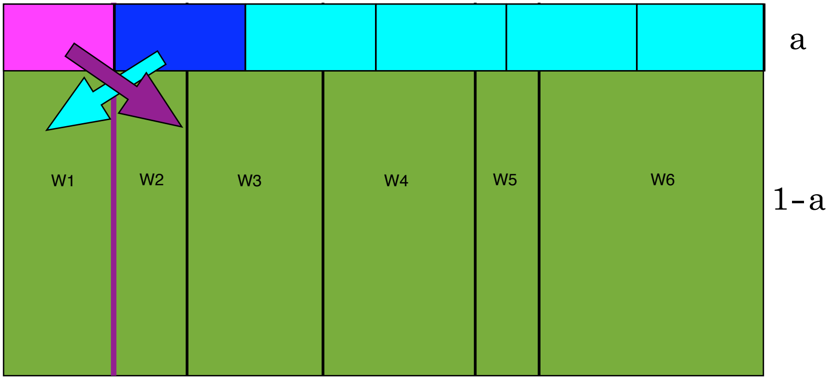
- Update weights in two stages: loss update then share update.
- Prediction uses the normalized s weights $w^s_{t,i} / \sum_j w^s_{t,j}$
- Loss update is the same as always, but defines intermediate m weights:

$$w^m_{t,i} = w^s_{t,i} e^{-\eta L(y_t, x_{t,i})}$$

- Share update: redistribute the weights
- Fixed-share:

$$\begin{aligned} pool &= \alpha \sum_{i=1}^{n} w^m_{t,i} \\ w^s_{t+1,i} &= (1-\alpha) w^m_{t,i} + \frac{1}{n-1} (pool - \alpha w^m_{t,i}) \end{aligned}$$

The fixed-share algorithm



Proving a bound on the fixed-share

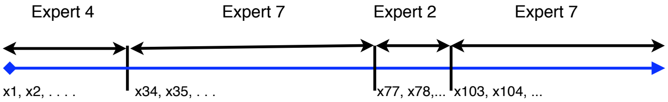
- The relation between algorithm loss and total weight does not change because share update does not change the total weight.
- Thus we still have

$$L_A \leq \frac{1}{\eta} \sum_{i=1}^{n} w^s_{l+1,i}$$

- The harder question is how to lower bound $\sum_{i=1}^{n} w^s_{l+1,i}$

Lower bounding the final total weight

- Fix some switching experts sequence:



- "follow" the weight of the chosen expert i_t.
- The loss update reduces the weight by a factor of $e^{-\eta \ell_{t,i_t}}$.
- The share update reduces the weight by a factor larger than:
  - $1 - \alpha$ on iterations with no switch.
  - $\frac{\alpha}{n-1}$ on iterations where a switch occurs.

Bound for arbitrary α

- Combining we lower bound the final weight of the last expert in the sequence

$$w^{s}_{l+1,e_k} \geq \frac{1}{n} e^{-\eta L_*} (1-\alpha)^{l-k-1} \left(\frac{\alpha}{n-1}\right)^{k}$$

Where L_* is the cumulative loss of the switching sequence of experts.

- Combining the upper and lower bounds we get that for any sequence

$$L_A \leq L_* + \frac{1}{\eta}\left(\ln n + (l-k-1)\ln\frac{1}{1-\alpha} + k\left(\ln\frac{1}{\alpha} + \ln(n-1)\right)\right)$$

Tuning α

- let k^* be the best number of switches (in hind sight) and $\alpha^* = k^*/l$
- Suppose we use $\alpha \approx \alpha^*$ then the bound that we get is

$$L_A \leq L_* + \frac{1}{\eta}((k+1)\ln n + (l-1)(H(\alpha^*) + D_{\mathrm{KL}}(\alpha^*||\alpha)))$$

Where

$$H(\alpha^*) = -\alpha^* \ln \alpha^* - (1-\alpha^*)\ln(1-\alpha^*)$$

$$D_{\mathrm{KL}}(\alpha^*||\alpha) = \alpha^* \ln \frac{\alpha^*}{\alpha}(1-\alpha^*)\ln \frac{1-\alpha^*}{1-\alpha}$$

- This is very close to the loss of the computationally inefficient algorithm.
- For the log loss case this is essentially optimal.
- Not so for square loss!

What can we hope to improve?

- In the fixed-share algorithm, the weight of a suboptimal expert never decreases below α/n.
- The algorithm does not concentrate only on the best expert, even if the last switch is in the distant past.
- The regret depends on the length of the sequence.

The idea of variable-share

- Let the fraction of the total weight given to the best expert get arbitrarily close to 1.
- we can get a regret bound that depends only on the number of switches, not on the lenght of the sequence.
- Requires that the loss be bounded.
- Works for square loss, but not for log loss!

Variable-share

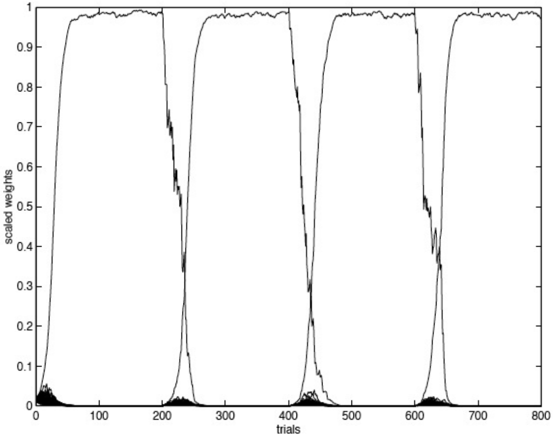
$$\begin{aligned} pool &= \sum_{i=1}^{n} \left(1 - (1-\alpha)^{\ell_{t,i}}\right) w^{m}_{t,i} \\ w^{s}_{t+1,i} &= (1-\alpha)^{\ell_{t,i}} w^{m}_{t,i} + \frac{1}{n-1}\left(pool - \left(1 - (1-\alpha)^{\ell_{t,i}}\right) w^{m}_{t,i}\right) \end{aligned}$$

If $\ell_{t,i} = 0$, then expert i does not contribute to the pool.
Expert can get fraction of the total weight arbitrarily close to 1.
Shares the weight quickly if $\ell_{t,i} > 0$

Bound for variable share

- $$\frac{1}{\eta}\ln n+\Bigg(1+\frac{1}{(1-\alpha)\eta}\Bigg)L_*+k\Bigg(1+\frac{1}{\eta}\Bigg(\ln n-1+\ln\frac{1}{\alpha}+\ln\frac{1}{1-\alpha}\Bigg)$$
- α should be tuned so that it is (close to) $\frac{k}{2k+L_*}$

An experiment using variable share



Switching within a small subset

- Suppose the best switching sequence is repeatedly switching among a small subset of the experts $n' \ll n$
- In the context of speech recognition - the speaker repeatedly uses a small number of phonemes.
- If we know the subset, we can pay $\ln n'$ per switch rather than $\ln n$
- Can track switches much more closely.
- Easy to describe an inefficient algorithm (consider all $\binom{n}{n'}$ subsets.)
- Switching to Slides from Manfred Warmuth.