3. A Short Detour: Drifting Games

A framework that generalizes boosting and some online learning problems. We study a special case:

- Formal setup: for \( t = 1, 2, \ldots, T \)
- player needs to choose a distribution \( p_t \) over \( N \) experts.
- adversary reveals the loss \( L_t(s) \) for each expert
- player suffers loss \( l_t = \sum_i p(t) L_t(s) \) for this round

\( \text{instantaneous regret} \) (to expert \( i \)) at time \( t \): \( r_t(i) = l_t - l_{\text{opt}}(i) \)

\( \text{cumulative regret} \) (to expert \( i \)) for \( T \) rounds: \( R_T(i) = \sum_{t=1}^{T} r_t(i) \)

Goal: minimize the final loss \( \sum_{t=1}^{T} L_t(s(t)) \), where \( L \) is a known non-increasing function.

4. Minimax algorithm and Relaxation

Studying minimal loss against worst-case adversary:

\[
\min_{\Phi} \max_{\{s \mid \Phi(s) \leq \beta\}} \sum_{t=1}^{T} \Phi(s(t))
\]

gives minimax optimal algorithm:

\[
p_t(i) \propto \arg\max_{s} \max_{\Phi} \Phi(s(i)+1) + w(s(i) - 1)
\]

where \( \Phi(s(i)) = L(s(i)) \; \Phi(s) = \min_{s'} \Phi(s_i+s' + w(s' - s)) \)

Relaxing the equality and using convex potential \( \Phi(s) \): gives

\[
p_t(i) \propto \frac{1}{2} \max_{s} \Phi(s)+(s(i) + 1) + \frac{1}{2} \beta - \Phi(s(i)-1)
\]

where \( \Phi(s(i)+1) > L(s(i)) \; \Phi(s(i)-1) < \frac{1}{2} \beta - \Phi(s(i)) \)

Guarantee:

\[
\sum_{t=1}^{T} L_t(s(t)) \leq \sum_{t=1}^{T} \Phi(s(t)) \leq \sum_{t=1}^{T} \Phi(s(t) - 1) \leq \cdots \leq N \Phi(0)
\]

5. Hedge as a Drifting Game

By construction, we have

\[
\sum_{t=1}^{T} L(s_t) = 0 \implies \min_{s_t} L_t(s_t) > -R \implies \max_{R_T(s)} R_T(s) > R
\]

Question: how large should \( R \) be to ensure \( \sum_{t=1}^{T} L(s_t) = 0 \)?

More generally, if \( k = \min \{R_T(s)\} \) is the best expert, then

\[
\sum_{t=1}^{T} L(s_t) < N \implies R_T(s) < R
\]

For the potential based algorithm: pick \( R \) s.t. \( \Phi(0) < c \).

6. Concrete Drifting Algorithms

By using different potential functions, we not only recover existing algorithms, but also give a new parameter-free algorithm (NormalHedge.DT).

\[
\Phi(s(t)) = \Phi_2(s(t)) = \beta \sum_{t=1}^{T} \Phi(s(t))
\]

Advantages over NormalHedge:

- better bound (no infN term)
- faster (no numerical search for c)
- can deal with infinitely many experts
- arguably simpler analysis.

7. Applications to Boosting

Converting NormalHedge.DT into a boosting algorithm:

Input: \( T \) training examples \( (x_t, y_t) \) \( t = 1, \ldots, T \)

Output: A predictor \( h(x) \)

Set: \( s_0 = 0 \)

for \( i = 1 \) to \( T \) do

Set: \( p_t(i) \propto \exp(-\beta \Phi_2(s_t(i))) \)

Get: \( h_t = \text{WL}(h_t, (x_t, y_t)) \)

Set: \( s_t = s_{t-1} + \sqrt{2 \Phi_2(s_t)/\beta} \)

end

Output: \( H(x) = \text{sign}(\sum_{t=1}^{T} h_t(x)) \)

- completely ignore some examples per round.
- adaptivity of NormalHedge.DT implies the fraction of training examples with margin at most \( \beta \) is of order

\[
O(\Phi(1/2) - \beta^2)
\]

8. Experiments

\[
\begin{array}{c|c|c|c|c|c}
\text{ADL} & \text{BD} & \text{MD} & \text{ADL} & \text{BD} & \text{MD} \\
\hline
0.5 & 2.1 & 2.6 & 0.5 & 2.1 & 2.6 \\
1.0 & 3.6 & 3.1 & 1.0 & 3.6 & 3.1 \\
1.5 & 5.1 & 4.6 & 1.5 & 5.1 & 4.6 \\
\end{array}
\]

Figure: Comparison of cumulative margins

(a) census
(b) splice
(c) wiba