

# Tracking the best Expert

Yoav Freund

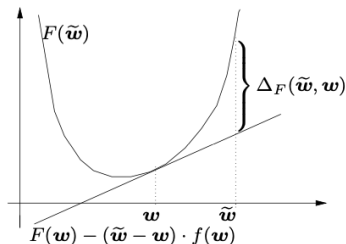
February 3, 2020

## Bregman Divergences [Br,CL,Cs]

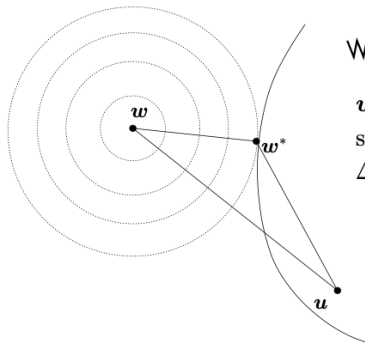
For **any** differentiable convex function  $F$

$$\Delta_F(\tilde{w}, w) = F(\tilde{w}) - F(w) - (\tilde{w} - w) \cdot \underbrace{\nabla_w F(w)}_{f(w)}$$

$$= F(\tilde{w}) - \begin{array}{l} \text{supporting hyperplane} \\ \text{through } (w, F(w)) \end{array}$$



## A Pythagorean Theorem [Br,Cs,A,HW]

 $\mathcal{W}$ 

$w^*$  is **projection** of  $w$  onto convex set  $\mathcal{W}$  w.r.t. Bregman divergence  $\Delta_F$ :

$$w^* = \operatorname{argmin}_{u \in \mathcal{W}} \Delta_F(u, w)$$

**Theorem:**

$$\Delta_F(u, w) \geq \Delta_F(u, w^*) + \Delta_F(w^*, w)$$

## Unnormalized Relative entropy

- ▶ prediction, outcome  $\mathbf{p}, \mathbf{q}$  are  $n$  dimensional vectors with non-negative coordinates.
- ▶ Loss is RE extended to non-negative vectors:

$$\text{RE}(\mathbf{p} \parallel \mathbf{q}) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} - \sum_{i=1}^n (q_i - p_i)$$

Coincides with RE when  $\sum_{i=1}^n p_i = \sum_{i=1}^n q_i = 1$

- ▶ Unnormalized RE is the Bregman divergence corresponding to the unnormalized entropy:

$$F(\mathbf{p}) = \sum_{i=1}^n p_i \log p_i - \sum_{i=1}^n p_i$$

# Inequalities for Unnormalized Relative entropy

► No triangle inequality

$$\exists \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \quad \text{RE}(\mathbf{p}_1 \parallel \mathbf{p}_3) > \text{RE}(\mathbf{p}_1 \parallel \mathbf{p}_2) + \text{RE}(\mathbf{p}_2 \parallel \mathbf{p}_3)$$

► Generalized Pythagorean inequality For any closed convex set  $S$  and any point  $\mathbf{p}_1 \notin S$ , define the projection of  $\mathbf{p}_1$  on  $S$  to be  $\mathbf{p}_2 = \operatorname{argmin}_{\mathbf{u} \in S} \text{RE}(\mathbf{p}_1 \parallel \mathbf{u})$ , then:

$$\forall \mathbf{p}_3 \in S; \quad \text{RE}(\mathbf{p}_1 \parallel \mathbf{p}_3) \geq \text{RE}(\mathbf{p}_1 \parallel \mathbf{p}_2) + \text{RE}(\mathbf{p}_2 \parallel \mathbf{p}_3)$$

## half squared euclidean distance

- ▶ prediction, outcome  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$



$$\lambda_{\text{sq}}(\mathbf{u}, \mathbf{v}) = \frac{1}{2} \|\mathbf{u} - \mathbf{v}\|^2 = \frac{1}{2} \sum_{i=1}^n (u_i - v_i)^2$$

- ▶ Bregman divergence with respect to the square euclidean norm

$$\|\mathbf{v}\|_2$$

- ▶ Triangle inequality does not hold.
- ▶ **Pythagoras inequality** : For any closed convex set  $S$  and any point  $\mathbf{v}_1 \notin S$ , define the projection of  $\mathbf{v}_1$  on  $S$  to be  $\mathbf{v}_2 = \operatorname{argmin}_{\mathbf{u} \in S} \|\mathbf{v}_1 - \mathbf{u}\|^2$ , then:

$$\forall \mathbf{v}_3 \in S; \quad \|\mathbf{v}_1 - \mathbf{v}_3\|^2 \geq \|\mathbf{v}_1 - \mathbf{v}_2\|^2 + \|\mathbf{v}_2 - \mathbf{v}_3\|^2$$

## Bregman divergence regularization

- ▶ Idea: Set  $\mathbf{w}_{t+1}$  to be  $\mathbf{u}$  that minimizes:

$$\Delta_F(\mathbf{w}_t, \mathbf{u}) + \alpha \ell_t(\mathbf{u})$$

- ▶ In general, hard to compute the minimum.
- ▶ Efficient approximation **Mirror Descent**. Will be covered later.

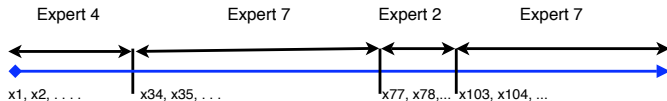
## Tracking Linear Experts

- ▶ **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  $\sum_t \Delta_F(\mathbf{u}_t - \mathbf{1}, \mathbf{u}_t)$
- ▶ **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  $\eta$  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  $\frac{1}{\eta} ((k+1) \log n + k \log \frac{l}{k} + k)$

## Weight sharing algorithms

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

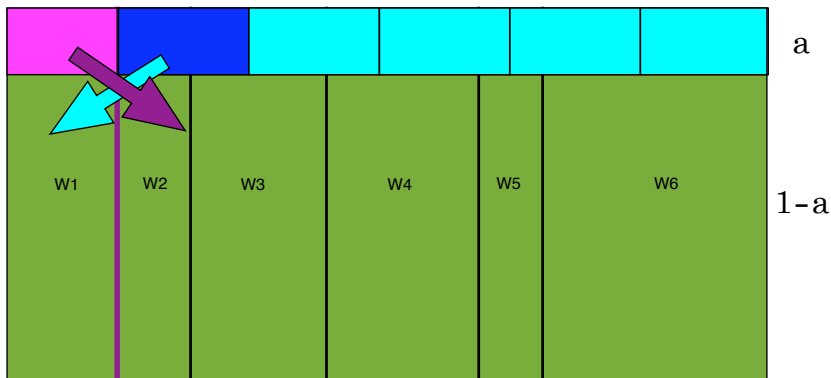
$$w_{t,i}^m = w_{t,i}^s e^{-\eta L(y_t, x_{t,i})}$$

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

$$pool = \alpha \sum_{i=1}^n w_{t,i}^m$$

$$w_{t+1,i}^s = (1 - \alpha) w_{t,i}^m + \frac{1}{n - 1} (pool - \alpha w_{t,i}^m)$$

## The fixed-share algorithm



## Proving a bound on the fixed-share

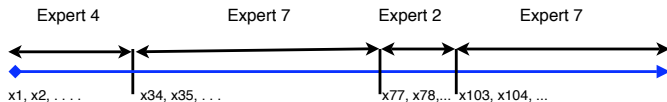
- ▶ The relation between algorithm loss and total weight does not change because share update does not change the total weight.
- ▶ Thus we still have

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

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

## 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 $\alpha$

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

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

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 $\alpha$

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

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

Where

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

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

- ▶ 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  $\alpha/n$ .
- ▶ 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 length of the sequence.
- ▶ Requires that the loss be bounded.
- ▶ Works for **square** loss, but not for **log** loss!

## Variable-share

$$pool = \sum_{i=1}^n \left(1 - (1 - \alpha)^{\ell_{t,i}}\right) w_{t,i}^m$$

$$w_{t+1,i}^s = (1 - \alpha)^{\ell_{t,i}} w_{t,i}^m + \frac{1}{n-1} \left( pool - (1 - (1 - \alpha)^{\ell_{t,i}}) w_{t,i}^m \right)$$

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  $\frac{1}{n}$ .  
Shares the weight quickly if  $\ell_{t,i} > 0$

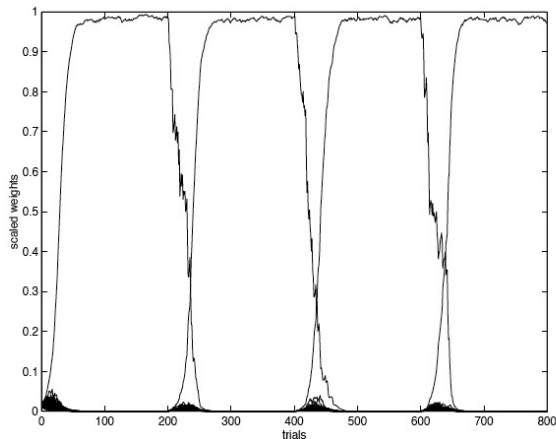
## Bound for variable share



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

- $\alpha$  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.