Tracking the best Expert

Yoav Freund

February 25, 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.

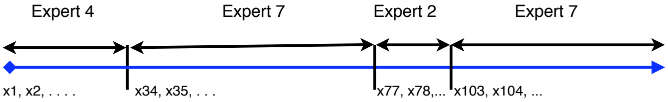
Switching experts setup

- Usually: compare algorithm's total loss to total loss of the best expert.

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.

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:

An inefficient algorithm

- Fix:
  - l - sequence length

An inefficient algorithm

- Fix:
  - l - sequence length
  - k - number of switches

An inefficient algorithm

- Fix:
  - l - sequence length
  - k - number of switches
  - n - number of experts

An inefficient algorithm

- Fix:
  - l - sequence length
  - k - number of switches
  - n - number of experts
- Consider one partition-expert per sequence of switching experts.

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 : $\binom{l}{k-1} n(n-1)^k = O\left(n^{k+1}\left(\frac{el}{k}\right)^k\right)$

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 $(k+1)\log n + k \log \frac{l}{k} + k$

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.

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

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

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.

Weight sharing algorithms

- Update weights in two stages: loss update then share update.
- Prediction uses the normalized s weights $w^s_{t,i} / \sum_j w^s_{t,j}$

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:

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

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

Weight sharing algorithms

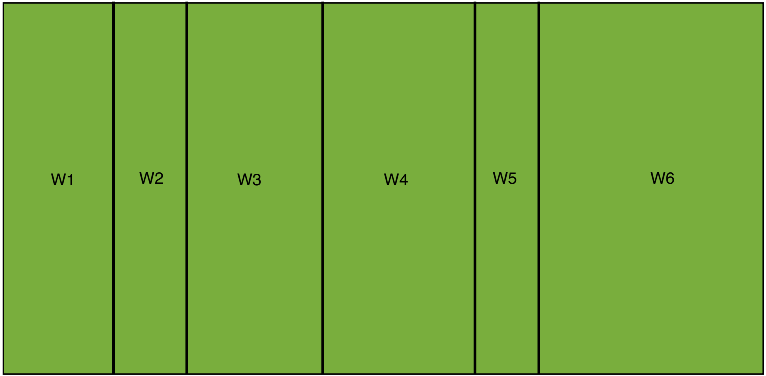
- Update weights in two stages: loss update then share update.
- Prediction uses the normalized s weights $w^s_{t,i} / \sum_j w^s_{t,j}$
- Loss update is the same as always, but defines intermediate m weights:

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

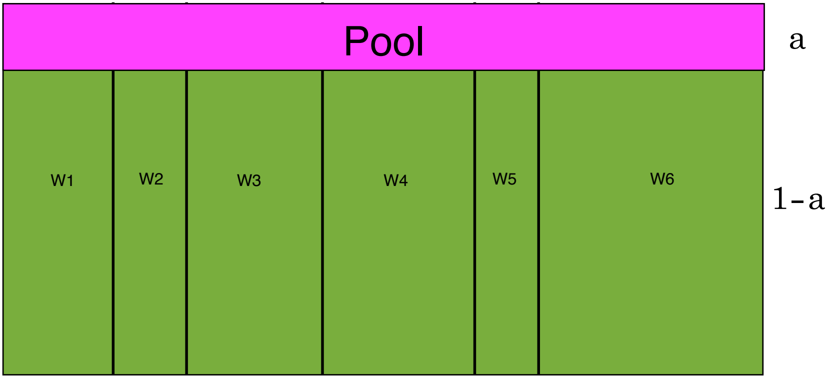
- Share update: redistribute the weights
- Fixed-share:

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

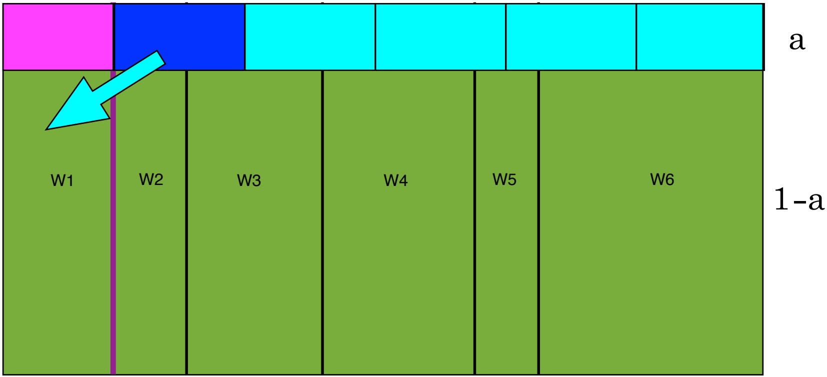
The fixed-share algorithm



The fixed-share algorithm



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.

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

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

Proving a bound on the fixed-share

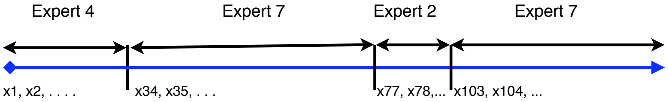
- The relation between algorithm loss and total weight does not change because share update does not change the total weight.
- Thus we still have

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

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

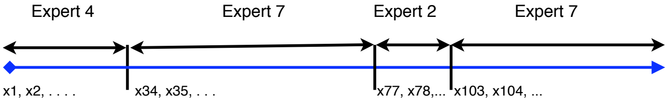
Lower bounding the final total weight

- Fix some switching experts sequence:



Lower bounding the final total weight

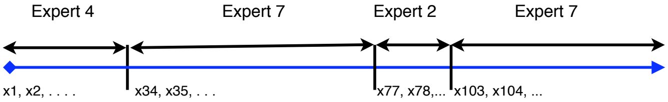
- Fix some switching experts sequence:



- “follow” the weight of the chosen expert i_t.

Lower bounding the final total weight

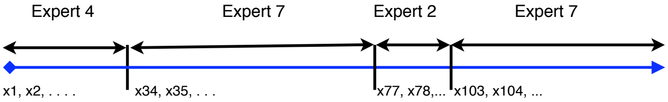
- Fix some switching experts sequence:



- "follow" the weight of the chosen expert i_t.
- The loss update reduces the weight by a factor of $e^{-\eta \ell_{t,i_t}}$.

Lower bounding the final total weight

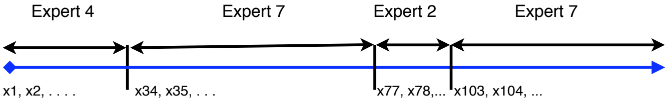
- Fix some switching experts sequence:



- "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:

Lower bounding the final total weight

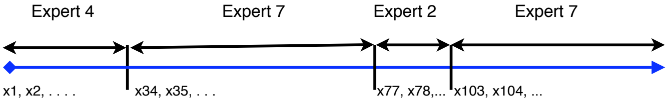
- Fix some switching experts sequence:



- "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.

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.

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 $\alpha^* = k^*/l$

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

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

Where

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

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

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.

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.

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.

What can we hope to improve?

- In the fixed-share algorithm, the weight of a suboptimal expert never decreases below α/n.
- The regret depends on the length of the sequence.

What can we hope to improve?

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

The idea of variable-share

- Let the fraction of the total weight given to the best expert get arbitrarily close to 1.

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.

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.

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!

Fixed-share:

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

Variable-share

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

Fixed-share:

$$
\begin{aligned}
pool &= \alpha \sum_{j=1}^{n} w_{t,j}^{m} \\
w_{t+1,i}^{s} &= (1-\alpha) w_{t,i}^{m} + \frac{1}{n-1} (pool - \alpha w_{t,i}^{m})
\end{aligned}
$$

Variable-share

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

If $\ell_{t,i} = 0$, then expert i does not contribute to the pool.

Fixed-share:

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

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.
If $\ell_{t,i} = 1$, then expert i contributes like fixed share.

Fixed-share:

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

Variable-share

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

If $\ell_{t,i} = 0$, then expert i does not contribute to the pool.
If $\ell_{t,i} = 1$, then expert i contributes like fixed share.
Expert can get fraction of the total weight arbitrarily close to 1.

Fixed-share:

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

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.
If $\ell_{t,i} = 1$, then expert i contributes like fixed share.
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

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

Bound for variable share

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

- α should be tuned so that it is (close to) $\frac{k}{2k+L_*}$

Bound for variable share

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

- α should be tuned so that it is (close to) $\frac{k}{2k+L_*}$
- there is no dependence on l the length of the sequence.

Some Experiments

Setup

- 2-3 expeerts

Some Experiments

Setup

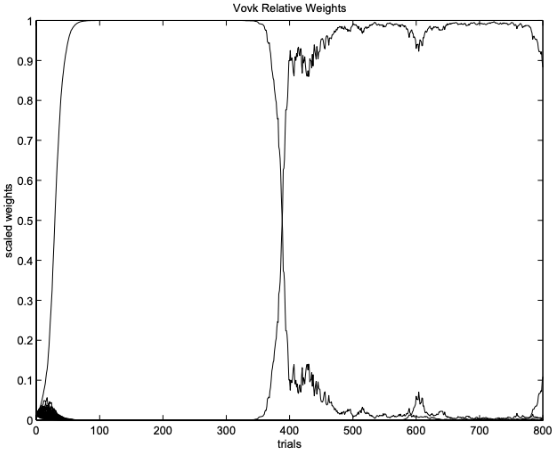
- 2-3 expeerts
- time is divided into equal length segments

Some Experiments

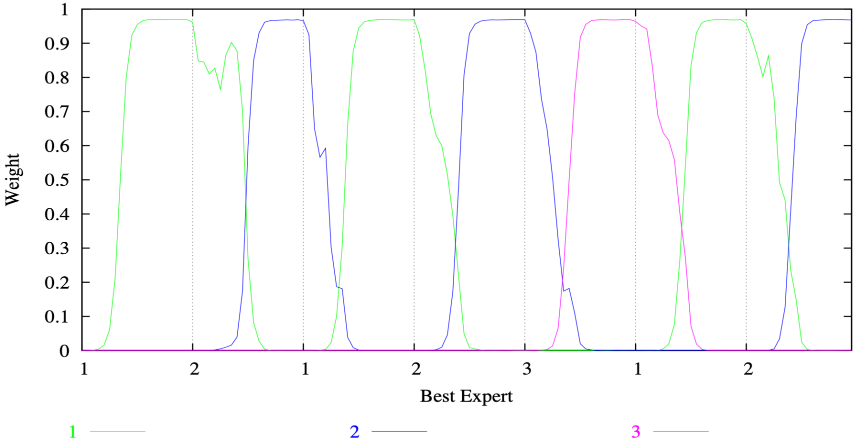
Setup

- 2-3 expeerts
- time is divided into equal length segments
- In each segment a different expert is good.

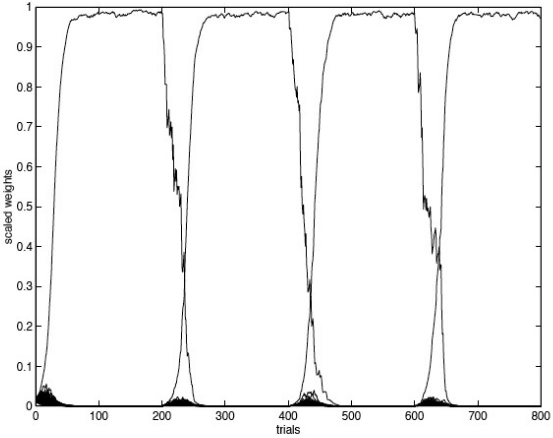
An experiment using static experts



An experiment using fixed share



An experiment using variable share



Analysis using Bregman divergences

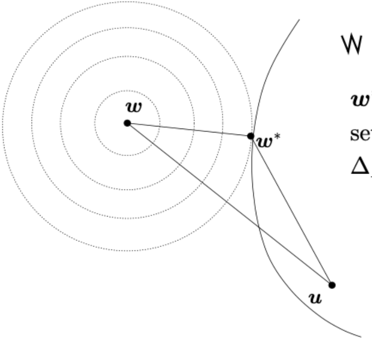
Mirror Descent (non-switching)

$$w_{t+1} = \arg\min_w \left[\eta \sum_{s=1}^{t} \langle w, z_s \rangle + D_R(w \| w_1) \right]$$

Regret Bound:

$$\sum_{t=1}^{T} \langle w_t - w^*, z_t \rangle \leq D_R(w^* \| w_1) + \sum_{t=1}^{T} D_R(w_t \| w_{t+1}).$$

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



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

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

Theorem:

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

Bounding Regret Using the Pythagorean Inequality

Pythagorean Inequality:

$$D_R\left(w^* \,\|\, w_t\right) \;\geq\; D_R\left(w^* \,\|\, w_{t+1}\right) \;+\; D_R\left(w_{t+1} \,\|\, w_t\right).$$

Bounding Regret Using the Pythagorean Inequality

Pythagorean Inequality:

$$D_R\big(w^* \,\|\, w_t\big) \;\geq\; D_R\big(w^* \,\|\, w_{t+1}\big) \;+\; D_R\big(w_{t+1} \,\|\, w_t\big).$$

plus three-point identity:

$$D_R(a \,\|\, b) \;+\; D_R(b \,\|\, c) \;-\; D_R(a \,\|\, c) \;=\; \langle a - b,\; \nabla R(c) - \nabla R(b)\rangle.$$

Bounding Regret Using the Pythagorean Inequality

Pythagorean Inequality:

$$D_R\big(w^* \,\|\, w_t\big) \;\geq\; D_R\big(w^* \,\|\, w_{t+1}\big) \;+\; D_R\big(w_{t+1} \,\|\, w_t\big).$$

plus three-point identity:

$$D_R(a \,\|\, b) \;+\; D_R(b \,\|\, c) \;-\; D_R(a \,\|\, c) \;=\; \langle a - b,\, \nabla R(c) - \nabla R(b)\rangle.$$

yields regret bound:

$$\sum_{t=1}^{T} \langle w_t - w^*,\, z_t\rangle \;\leq\; D_R\big(w^* \,\|\, w_1\big) \;+\; \sum_{t=1}^{T} D_R\big(w_{t+1} \,\|\, w_t\big).$$

Bounding Regret Using the Pythagorean Inequality

Pythagorean Inequality:

$$D_R\big(w^* \,\|\, w_t\big) \;\geq\; D_R\big(w^* \,\|\, w_{t+1}\big) \;+\; D_R\big(w_{t+1} \,\|\, w_t\big).$$

plus three-point identity:

$$D_R(a \,\|\, b) \;+\; D_R(b \,\|\, c) \;-\; D_R(a \,\|\, c) \;=\; \langle a - b,\, \nabla R(c) - \nabla R(b) \rangle.$$

yields regret bound:

$$\sum_{t=1}^{T} \langle w_t - w^*,\, z_t \rangle \;\leq\; D_R\big(w^* \,\|\, w_1\big) \;+\; \sum_{t=1}^{T} D_R\big(w_{t+1} \,\|\, w_t\big).$$

Incorporating Switching:

- Switching is controlled by $D_R\big(w_{t+1} \,\|\, w_t\big)$.

Bounding Regret Using the Pythagorean Inequality

Pythagorean Inequality:

$$D_R\big(w^* \,\|\, w_t\big) \;\geq\; D_R\big(w^* \,\|\, w_{t+1}\big) \;+\; D_R\big(w_{t+1} \,\|\, w_t\big).$$

plus three-point identity:

$$D_R(a \,\|\, b) \;+\; D_R(b \,\|\, c) \;-\; D_R(a \,\|\, c) \;=\; \langle a-b,\, \nabla R(c) - \nabla R(b)\rangle.$$

yields regret bound:

$$\sum_{t=1}^{T} \langle w_t - w^*,\, z_t\rangle \;\leq\; D_R\big(w^* \,\|\, w_1\big) \;+\; \sum_{t=1}^{T} D_R\big(w_{t+1} \,\|\, w_t\big).$$

Incorporating Switching:

- Switching is controlled by $D_R\big(w_{t+1} \,\|\, w_t\big)$.
- The total regret depends on the regularizer $R(w)$.

Fixed Share Algorithm

Fixed Share Update:

$$w_{t+1}^i = (1-\alpha)\frac{w_t^i e^{-\eta z_t^i}}{\sum_j w_t^j e^{-\eta z_t^j}} + \frac{\alpha}{N}.$$

Impact on Pythagorean Inequality:

$$D_R(w^* \| w_t) \geq D_R(w^* \| w_{t+1}) + D_R(w_{t+1} \| w_t).$$

Modification: The divergence $D_R(w_{t+1} \| w_t)$ increases due to the uniform mixing factor α.

Regret Bound:

$$\sum_{t=1}^{T} \langle w_t - w^*, z_t \rangle \leq D_R(w^* \| w_1) + \sum_{t=1}^{T} \left[D_R(w_t \| w_{t+1}) + \alpha D_{KL}(w_t \| u) \right].$$

Variable Share Algorithm

Variable Share Update:

$$w_{t+1}^i = (1 - \alpha_t)\frac{w_t^i e^{-\eta z_t^i}}{\sum_j w_t^j e^{-\eta z_t^j}} + \alpha_t S_t^i.$$

Impact on Pythagorean Inequality:

$$D_R(w^* \| w_t) \geq D_R(w^* \| w_{t+1}) + D_R(w_{t+1} \| w_t) + \alpha_t D_{KL}(w_t \| u).$$

Modification: The divergence term now depends on α_t, making it adaptive rather than constant.

Regret Bound:

$$\sum_{t=1}^{T} \langle w_t - w^*, z_t \rangle \leq D_R(w^* \| w_1) + \sum_{t=1}^{T} \left[D_R(w_t \| w_{t+1}) + \alpha_t D_{KL}(w_t \| u) \right].$$