Mixable losses
and
Tracking the best Expert

Yoav Freund

January 22, 2014

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Outline

The log-loss game

- Prediction algorithm A has access to N experts.

The log-loss game

- Prediction algorithm A has access to N experts.
- The following is repeated for $t = 1, \ldots, T$

The log-loss game

- Prediction algorithm A has access to N experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $\mathbf{p}_1^t, \ldots, \mathbf{p}_N^t$

The log-loss game

- Prediction algorithm A has access to N experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $\mathbf{p}_1^t, \ldots, \mathbf{p}_N^t$
  - Algorithm generates its own prediction $\mathbf{p}_A^t$

The log-loss game

- Prediction algorithm A has access to N experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $\mathbf{p}_1^t, \ldots, \mathbf{p}_N^t$
  - Algorithm generates its own prediction $\mathbf{p}_A^t$
  - c^t is revealed.

The log-loss game

- Prediction algorithm A has access to N experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $\mathbf{p}_1^t, \ldots, \mathbf{p}_N^t$
  - Algorithm generates its own prediction $\mathbf{p}_A^t$
  - c^t is revealed.
- **Goal:** minimize regret:

$$-\sum_{t=1}^{T} \log p_A^t(c^t) + \min_{i=1,\ldots,N} \left(-\sum_{t=1}^{T} \log p_i^t(c^t) \right)$$

The online Bayes Algorithm

- Total loss of expert i

$$L_i^t = -\sum_{s=1}^{t} \log p_i^s(c^s); \quad L_i^0 = 0$$

The online Bayes Algorithm

- Total loss of expert i

$$L_i^t = -\sum_{s=1}^{t} \log p_i^s(c^s); \quad L_i^0 = 0$$

- Weight of expert i

$$w_i^t = w_i^1 e^{-L_i^{t-1}} = w_i^1 \prod_{s=1}^{t-1} p_i^s(c^s)$$

The online Bayes Algorithm

- Total loss of expert i

$$L_i^t = -\sum_{s=1}^{t} \log p_i^s(c^s); \quad L_i^0 = 0$$

- Weight of expert i

$$w_i^t = w_i^1 e^{-L_i^{t-1}} = w_i^1 \prod_{s=1}^{t-1} p_i^s(c^s)$$

- Freedom to choose initial weights.
  $w_t^1 \geq 0, \sum_{i=1}^{n} w_i^1 = 1$

The online Bayes Algorithm

- Total loss of expert i

$$L_i^t = -\sum_{s=1}^{t} \log p_i^s(c^s); \quad L_i^0 = 0$$

- Weight of expert i

$$w_i^t = w_i^1 e^{-L_i^{t-1}} = w_i^1 \prod_{s=1}^{t-1} p_i^s(c^s)$$

- Freedom to choose initial weights.
  $w_t^1 \geq 0, \sum_{i=1}^{n} w_i^1 = 1$
- Prediction of algorithm A

$$\mathbf{p}_A^t = \frac{\sum_{i=1}^{N} w_i^t \mathbf{p}_i^t}{\sum_{i=1}^{N} w_i^t}$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^{N} w_i^t$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^{N} w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^{N} w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^{N} w_i^t}$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t}$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

$$-\log \frac{W^{t+1}}{W^t} = -\log p_A^t(c^t)$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

$$-\log \frac{W^{t+1}}{W^t} = -\log p_A^t(c^t)$$

$$-\log \frac{W^{T+1}}{W^1} = -\sum_{t=1}^T \log p_A^t(c^t)$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

$$-\log \frac{W^{t+1}}{W^t} = -\log p_A^t(c^t)$$

$$-\log \frac{W^{T+1}}{W^1} = -\sum_{t=1}^T \log p_A^t(c^t) = L_A^T$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

$$-\log\frac{W^{t+1}}{W^t} = -\log p_A^t(c^t)$$

$$-\log W^{T+1} = -\log\frac{W^{T+1}}{W^1} = -\sum_{t=1}^T \log p_A^t(c^t) = L_A^T$$

Cumulative loss vs. Final total weight

Total weight: $W^t \doteq \sum_{i=1}^N w_i^t$

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^N w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^N w_i^t} = \frac{\sum_{i=1}^N w_i^t p_i^t(c^t)}{\sum_{i=1}^N w_i^t} = p_A^t(c^t)$$

$$-\log \frac{W^{t+1}}{W^t} = -\log p_A^t(c^t)$$

$$-\log W^{T+1} = -\log \frac{W^{T+1}}{W^1} = -\sum_{t=1}^T \log p_A^t(c^t) = L_A^T$$

EQUALITY not bound!

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.
On each trial $t = 1, 2, \ldots$

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.
On each trial $t = 1, 2, \ldots$

1. Each expert $i \in \{1 \ldots N\}$ makes a prediction $\gamma_i^t \in \Gamma$

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.
On each trial $t = 1, 2, \ldots$

1. Each expert $i \in \{1 \ldots N\}$ makes a prediction $\gamma_i^t \in \Gamma$
2. The learner, after observing $\langle \gamma_1^t \ldots \gamma_N^t \rangle$,
   makes its own prediction γ^t

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.
On each trial $t = 1, 2, \ldots$

1. Each expert $i \in \{1 \ldots N\}$ makes a prediction $\gamma_i^t \in \Gamma$
2. The learner, after observing $\langle \gamma_1^t \ldots \gamma_N^t \rangle$, makes its own prediction γ^t
3. Nature chooses an outcome $\omega^t \in \Omega$

Vovk's general prediction game

Γ - prediction space. Ω - outcome space.
On each trial $t = 1, 2, \ldots$

1. Each expert $i \in \{1 \ldots N\}$ makes a prediction $\gamma_i^t \in \Gamma$
2. The learner, after observing $\langle \gamma_1^t \ldots \gamma_N^t \rangle$, makes its own prediction γ^t
3. Nature chooses an outcome $\omega^t \in \Omega$
4. Each expert incurs loss $\ell_i^t = \lambda(\omega^t, \gamma_i^t)$
   The learner incurs loss $\ell_A^t = \lambda(\omega^t, \gamma^t)$

Achievable loss bounds

- $L_A \doteq \sum_{t=1}^{T} \ell_A^t$ - total loss of algorithm

Achievable loss bounds

- $L_A \doteq \sum_{t=1}^{T} \ell_A^t$ - total loss of algorithm
- $L_i \doteq \sum_{t=1}^{T} \ell_i^t$ - total loss of expert i

Achievable loss bounds

- $L_A \doteq \sum_{t=1}^{T} \ell_A^t$ - total loss of algorithm
- $L_i \doteq \sum_{t=1}^{T} \ell_i^t$ - total loss of expert i
- **Goal:** find an algorithm which guarantees that

$$(a, c) \in [0, \infty), \quad L_A \leq aL_{\min} + c \ln N$$

For any sequence of events.

Achievable loss bounds

- $L_A \doteq \sum_{t=1}^{T} \ell_A^t$ - total loss of algorithm
- $L_i \doteq \sum_{t=1}^{T} \ell_i^t$ - total loss of expert i
- **Goal:** find an algorithm which guarantees that

$$(a, c) \in [0, \infty), \quad L_A \leq aL_{\min} + c \ln N$$

  For any sequence of events.
- We say that the pair (a, c) is achievable.

The set of achievable bounds

- Fix loss function $\lambda : \Omega \times \Gamma \to [0, \infty)$

The set of achievable bounds

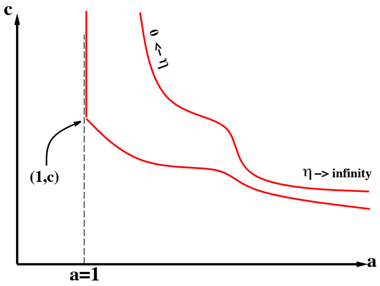
- Fix loss function $\lambda : \Omega \times \Gamma \to [0, \infty)$
- The pair (a, c) is *achievable* if there exists *some* prediction algorithm such that for *any* $N > 0$, *any* set of N prediction sequences and *any* sequence of outcomes

$$L_A \leq aL_{\min} + c \ln N$$

The set of achievable bounds

- Fix loss function $\lambda : \Omega \times \Gamma \to [0, \infty)$
- The pair (a, c) is *achievable* if there exists *some* prediction algorithm such that for *any* $N > 0$, *any* set of N prediction sequences and *any* sequence of outcomes

$$L_A \leq aL_{\min} + c \ln N$$



Some useful loss functions

- Outcomes: $\omega^1, \omega_2, \ldots \omega^t \in [0, 1]$

Some useful loss functions

- Outcomes: $\omega^1, \omega_2, \ldots \omega^t \in [0, 1]$
- Predictions: $\gamma^1, \gamma^2, \ldots \gamma^t \in [0, 1]$

Log loss (Entropy loss)

- $$\lambda_{\mathrm{ent}}(\omega, \gamma) = \omega \ln \frac{\omega}{\gamma} + (1 - \omega) \ln \frac{1 - \omega}{1 - \gamma}$$

Log loss (Entropy loss)

- $$\lambda_{\text{ent}}(\omega, \gamma) = \omega \ln \frac{\omega}{\gamma} + (1 - \omega) \ln \frac{1 - \omega}{1 - \gamma}$$
- When $q_t \in \{0, 1\}$ Cumulative log loss $=$ coding length ± 1

Log loss (Entropy loss)

- $$\lambda_{\text{ent}}(\omega,\gamma) = \omega \ln \frac{\omega}{\gamma} + (1-\omega) \ln \frac{1-\omega}{1-\gamma}$$
- When $q_t \in \{0,1\}$ Cumulative log loss $=$ coding length ± 1
- If $P[\omega_t = 1] = q$, optimal prediction $\gamma^t = q$

Log loss (Entropy loss)

- $$\lambda_{\text{ent}}(\omega, \gamma) = \omega \ln \frac{\omega}{\gamma} + (1 - \omega) \ln \frac{1 - \omega}{1 - \gamma}$$
- When $q_t \in \{0, 1\}$ Cumulative log loss $=$ coding length ± 1
- If $P[\omega_t = 1] = q$, optimal prediction $\gamma^t = q$
- Unbounded loss.

Log loss (Entropy loss)

- $$\lambda_{\text{ent}}(\omega, \gamma) = \omega \ln \frac{\omega}{\gamma} + (1 - \omega) \ln \frac{1 - \omega}{1 - \gamma}$$
- When $q_t \in \{0, 1\}$ Cumulative log loss $=$ coding length ± 1
- If $P[\omega_t = 1] = q$, optimal prediction $\gamma^t = q$
- Unbounded loss.
- Not symmetric $\exists p, q \;\; \lambda(p, q) \neq \lambda(q, p)$.

Log loss (Entropy loss)

- $$\lambda_{\text{ent}}(\omega, \gamma) = \omega \ln \frac{\omega}{\gamma} + (1 - \omega) \ln \frac{1 - \omega}{1 - \gamma}$$
- When $q_t \in \{0, 1\}$ Cumulative log loss $=$ coding length ± 1
- If $P[\omega_t = 1] = q$, optimal prediction $\gamma^t = q$
- Unbounded loss.
- Not symmetric $\exists p, q \;\; \lambda(p, q) \neq \lambda(q, p)$.
- No triangle inequality
  $\exists p_1, p_2, p_3 \;\; \lambda(p_1, p_3) > \lambda(p_1, p_2) + \lambda(p_2, p_3)$

Square loss (Breier Loss)

- $$\lambda_{\mathsf{sq}}(\omega,\gamma) = (\omega-\gamma)^2$$

Square loss (Breier Loss)

- $$\lambda_{\text{sq}}(\omega, \gamma) = (\omega - \gamma)^2$$
- $P[\omega^t = 1] = q, \;\; P[\omega^t = 0] = 1 - q$, optimal prediction $\gamma^t = q$

Square loss (Breier Loss)

- $$\lambda_{\mathrm{sq}}(\omega, \gamma) = (\omega - \gamma)^2$$
- $P[\omega^t = 1] = q, \quad P[\omega^t = 0] = 1 - q$, optimal prediction $\gamma^t = q$
- Bounded loss.

Square loss (Breier Loss)

- $$\lambda_{\mathrm{sq}}(\omega, \gamma) = (\omega - \gamma)^2$$
- $P[\omega^t = 1] = q, \;\; P[\omega^t = 0] = 1 - q$,
  optimal prediction $\gamma^t = q$
- Bounded loss.
- Defines a metric (symmetric and triangle ineq.)

Square loss (Breier Loss)

- $$\lambda_{\mathrm{sq}}(\omega, \gamma) = (\omega - \gamma)^2$$
- $P[\omega^t = 1] = q, \;\; P[\omega^t = 0] = 1 - q$,
  optimal prediction $\gamma^t = q$
- Bounded loss.
- Defines a metric (symmetric and triangle ineq.)
- Corresponds to regression.

Hellinger Loss

- $$\lambda_{\mathrm{hel}}(\omega,\gamma)=\frac{1}{2}\left(\left(\sqrt{\omega}+\sqrt{\gamma}\right)^2+\left(\sqrt{1-\omega}+\sqrt{1-\gamma}\right)^2\right)$$

Hellinger Loss

- $$\lambda_{\text{hel}}(\omega, \gamma) = \frac{1}{2}\left((\sqrt{\omega} + \sqrt{\gamma})^2 + \left(\sqrt{1-\omega} + \sqrt{1-\gamma}\right)^2 \right)$$

- If $P[\omega^t = 1] = q, \;\; P[\omega^t = 0] = 1 - q$,
  optimal prediction $\gamma^t = q$

Hellinger Loss

- $$\lambda_{\text{hel}}(\omega,\gamma) = \frac{1}{2}\left(\left(\sqrt{\omega}+\sqrt{\gamma}\right)^2 + \left(\sqrt{1-\omega}+\sqrt{1-\gamma}\right)^2\right)$$
- If $P[\omega^t = 1] = q$, $P[\omega^t = 0] = 1 - q$, optimal prediction $\gamma^t = q$
- Loss is bounded.

Hellinger Loss

- $$\lambda_{\text{hel}}(\omega, \gamma) = \frac{1}{2}\left(\left(\sqrt{\omega} + \sqrt{\gamma}\right)^2 + \left(\sqrt{1-\omega} + \sqrt{1-\gamma}\right)^2\right)$$
- If $P[\omega^t = 1] = q, \; P[\omega^t = 0] = 1 - q$, optimal prediction $\gamma^t = q$
- Loss is bounded.
- Defines a metric.

Hellinger Loss

- $$\lambda_{\text{hel}}(\omega, \gamma) = \frac{1}{2}\left(\left(\sqrt{\omega} + \sqrt{\gamma}\right)^2 + \left(\sqrt{1-\omega} + \sqrt{1-\gamma}\right)^2\right)$$
- If $P[\omega^t = 1] = q$, $P[\omega^t = 0] = 1 - q$, optimal prediction $\gamma^t = q$
- Loss is bounded.
- Defines a metric.
- $\lambda_{\text{hel}}(p, q) \approx \lambda_{\text{ent}}(p, q)$ when $p \approx q$ and $p, q \in (0, 1)$

Absolute loss

- $$\lambda(\omega, \gamma) = |\omega - \gamma|$$

Absolute loss

- $$\lambda(\omega, \gamma) = |\omega - \gamma|$$

- Probability of making a mistake if predicting 0 or 1 using a biased coin

Absolute loss

- $$\lambda(\omega, \gamma) = |\omega - \gamma|$$
- Probability of making a mistake if predicting 0 or 1 using a biased coin
- If $P[\omega^t = 1] = q, \quad P[\omega^t = 0] = 1 - q$, then the optimal prediction is
$$\gamma^t = \begin{cases} 1 & \text{if } q > 1/2, \\ 0 & \text{otherwise} \end{cases}$$

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$
- Loss is the dot product: $\lambda_{\text{dot}}(\omega, \gamma) = \gamma \cdot \omega$

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$
- Loss is the dot product: $\lambda_{\text{dot}}(\omega, \gamma) = \gamma \cdot \omega$
- Corresponds to the hedging game.

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$
- Loss is the dot product: $\lambda_{\text{dot}}(\omega, \gamma) = \gamma \cdot \omega$
- Corresponds to the hedging game.
- For hedge loss the regret is $\Omega(\sqrt{T \log N})$.

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$
- Loss is the dot product: $\lambda_{\text{dot}}(\omega, \gamma) = \gamma \cdot \omega$
- Corresponds to the hedging game.
- For hedge loss the regret is $\Omega(\sqrt{T \log N})$.
- For the log loss the regret is $O(\log N)$

Structureless bounded loss

- Prediction is a distribution $\gamma = \langle p_1, \ldots, p_N \rangle$, $p_i \geq 0$, $\sum_{i=1}^{N} p_i = 1$
- Outcome is a loss vector $\omega = \langle \omega_1, \ldots, \omega_N \rangle$, $0 \leq \omega_i \leq 1$
- Loss is the dot product: $\lambda_{\mathrm{dot}}(\omega, \gamma) = \gamma \cdot \omega$
- Corresponds to the hedging game.
- For hedge loss the regret is $\Omega(\sqrt{T \log N})$.
- For the log loss the regret is $O(\log N)$
- **Which losses behave like entropy loss and which behave like hedge loss?**

Some technical requirements

- There should be a topology on the prediction set Γ such that

Some technical requirements

- There should be a topology on the prediction set Γ such that
- Γ is compact.

Some technical requirements

- There should be a topology on the prediction set Γ such that
- Γ is compact.
- $\forall \omega \in \Omega$, the function $\gamma \to \lambda(\omega, \gamma)$ is continuous

Some technical requirements

- There should be a topology on the prediction set Γ such that
- Γ is compact.
- $\forall \omega \in \Omega$, the function $\gamma \rightarrow \lambda(\omega, \gamma)$ is continuous
- There is a universally reasonable prediction
  $\exists \gamma \in \Gamma, \forall \omega \in \Omega, \lambda(\omega, \gamma) < \infty$

Some technical requirements

- There should be a topology on the prediction set Γ such that
- Γ is compact.
- $\forall \omega \in \Omega$, the function $\gamma \to \lambda(\omega, \gamma)$ is continuous
- There is a universally reasonable prediction $\exists \gamma \in \Gamma, \forall \omega \in \Omega, \lambda(\omega, \gamma) < \infty$
- There is no universally optimal prediction $\neg \exists \gamma \in \Gamma, \forall \omega \in \Omega, \lambda(\omega, \gamma) = 0$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$
- 1.

$$W_i^t = \frac{1}{N} e^{-\eta L_i^t}$$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$
- 1.
$$W_i^t = \frac{1}{N} e^{-\eta L_i^t}$$

Choose γ_t so that, for all $\omega^t \in \Omega$:

$$\lambda(\omega^t, \gamma^t) - c \ln \sum_i W_i^t \leq -c \ln \left(\sum_i W_i^t e^{-\eta \lambda(\omega^t, \gamma_i^t)} \right)$$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$
- 1.

$$W_i^t = \frac{1}{N} e^{-\eta L_i^t}$$

Choose γ_t so that, for all $\omega^t \in \Omega$:

$$\lambda(\omega^t, \gamma^t) - c \ln \sum_i W_i^t \le -c \ln \left(\sum_i W_i^t e^{-\eta\lambda(\omega^t, \gamma_i^t)} \right)$$

- 2. If choice of γ^t always exists, then the total loss satisfies:

$$\sum_t \lambda(\omega^t, \gamma^t) \le -c \ln \sum_i W_i^{T+1} \le aL_{\min} + c \ln N$$

Vovk's meta-algorithm

- Fix an achievable pair (a, c) and set $\eta = a/c$
- 1.

$$W_i^t = \frac{1}{N}\, e^{-\eta L_i^t}$$

Choose γ_t so that, for all $\omega^t \in \Omega$:

$$\lambda(\omega^t, \gamma^t) - c \ln \sum_i W_i^t \leq -c \ln \left(\sum_i W_i^t e^{-\eta\lambda(\omega^t, \gamma_i^t)} \right)$$

2. If choice of γ^t always exists, then the total loss satisfies:

$$\sum_t \lambda(\omega^t, \gamma^t) \leq -c \ln \sum_i W_i^{T+1} \leq aL_{\min} + c \ln N$$

- Vovk's result: *yes!* a good choice for γ_t always exists!

Vovk's algorithm is the the highest achiever [Vovk95]

The pair (a, c) is achieved by some algorithm if and only if it is achieved by Vovk's algorithm.

Vovk's algorithm is the the highest achiever [Vovk95]

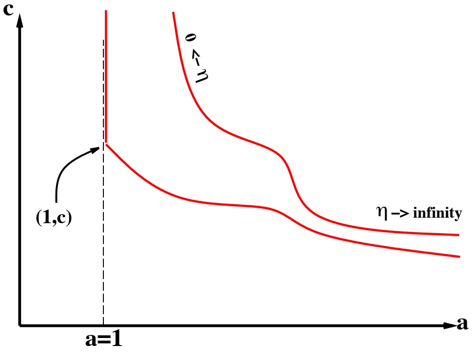
The pair (a, c) is achieved by some algorithm if and only if it is achieved by Vovk's algorithm.

The separation curve is $\left\{ \left(a(\eta), \frac{a(\eta)}{\eta} \right) \middle| \eta \in [0, \infty] \right\}$

Vovk's algorithm is the the highest achiever [Vovk95]

The pair (a, c) is achieved by some algorithm if and only if it is achieved by Vovk's algorithm.

The separation curve is $\left\{ \left(a(\eta), \frac{a(\eta)}{\eta} \right) \middle| \eta \in [0, \infty] \right\}$



Mixable Loss Functions

- A Loss function is mixable if a pair of the form $(1, c), \; c < \infty$ is achievable.

$$L_A \leq L_{\min} + c \ln N$$

Mixable Loss Functions

- A Loss function is mixable if a pair of the form $(1, c),\ c < \infty$ is achievable.

$$L_A \leq L_{\min} + c \ln N$$

- Vovk's algorithm with $\eta = 1/c$ achieves this bound.

Mixable Loss Functions

- A Loss function is mixable if a pair of the form $(1, c),\ c < \infty$ is achievable.

$$L_A \leq L_{\min} + c \ln N$$

- Vovk's algorithm with $\eta = 1/c$ achieves this bound.
- $\lambda_{\text{ent}}, \lambda_{\text{sq}}, \lambda_{\text{hel}}$ are mixable

Mixable Loss Functions

- A Loss function is mixable if a pair of the form $(1, c),\ c < \infty$ is achievable.

$$L_A \leq L_{\min} + c \ln N$$

- Vovk's algorithm with $\eta = 1/c$ achieves this bound.
- $\lambda_{\text{ent}}, \lambda_{\text{sq}}, \lambda_{\text{hel}}$ are mixable
- $\lambda_{\text{abs}}, \lambda_{\text{dot}}$ are not mixable

The convexity condition

- requirement for loss to be $(1, 1/\eta)$ mixable

The convexity condition

- requirement for loss to be $(1, 1/\eta)$ mixable
- $\forall \langle (\gamma_1, W_1), \ldots, (\gamma_N, W_N) \rangle$
  $\exists \gamma \in \Gamma$
  $\forall \omega \in \Omega$:

$$\lambda(\omega, \gamma) - \frac{1}{\eta} \ln \sum_i W_i \leq -\frac{1}{\eta} \ln \left(\sum_i W_i e^{-\eta \lambda(\omega, \gamma_i)} \right)$$

The convexity condition

- requirement for loss to be $(1, 1/\eta)$ mixable
- $\forall\langle(\gamma_1, W_1), \ldots, (\gamma_N, W_N)\rangle$
  $\exists\gamma \in \Gamma$
  $\forall\omega \in \Omega$:

$$\lambda(\omega, \gamma) - \frac{1}{\eta} \ln \sum_i W_i \leq -\frac{1}{\eta} \ln \left(\sum_i W_i e^{-\eta\lambda(\omega,\gamma_i)} \right)$$

- Can be re-written as:

$$e^{-\eta\lambda(\omega,\gamma)} \geq \sum_i \left(\frac{W_i}{\sum_j W_j} \right) e^{-\eta\lambda(\omega,\gamma_i)}$$

The convexity condition

- requirement for loss to be $(1, 1/\eta)$ mixable
- $\forall \langle (\gamma_1, W_1), \ldots, (\gamma_N, W_N) \rangle$
  $\exists \gamma \in \Gamma$
  $\forall \omega \in \Omega$:

$$\lambda(\omega, \gamma) - \frac{1}{\eta} \ln \sum_i W_i \leq -\frac{1}{\eta} \ln \left(\sum_i W_i e^{-\eta \lambda(\omega, \gamma_i)} \right)$$

- Can be re-written as:

$$e^{-\eta \lambda(\omega, \gamma)} \geq \sum_i \left(\frac{W_i}{\sum_j W_j} \right) e^{-\eta \lambda(\omega, \gamma_i)}$$

- Equivalently - the image of the set Γ under the mapping $F(\gamma) = \left\langle e^{-\eta \lambda(\omega, \gamma)} \right\rangle_{\omega \in \Omega}$ is concave.

convexity condition: Pictorially

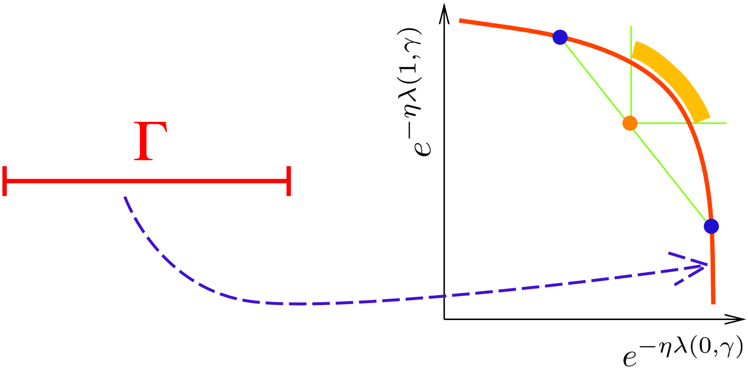
- **Example:** Suppose $\Omega = \{0, 1\}$, $\Gamma = [0, 1]$. then

$$F(\gamma) = \left\langle e^{-\eta\lambda(0,\gamma)}, e^{-\eta\lambda(1,\gamma)} \right\rangle$$

convexity condition: Pictorially

- **Example:** Suppose $\Omega = \{0, 1\}$, $\Gamma = [0, 1]$. then

$$F(\gamma) = \left\langle e^{-\eta\lambda(0,\gamma)}, e^{-\eta\lambda(1,\gamma)} \right\rangle$$



-

Vovk Algorithm for log loss

- The log loss is mixable with $\eta = 1$

Vovk Algorithm for log loss

- The log loss is mixable with $\eta = 1$
- The image of $[0, 1]$ through $F(\gamma) = \left\langle e^{-\eta\lambda(0,\gamma)}, e^{-\eta\lambda(1,\gamma)} \right\rangle$ is a straight line segment.

Vovk Algorithm for log loss

- The log loss is mixable with $\eta = 1$
- The image of $[0, 1]$ through $F(\gamma) = \left\langle e^{-\eta\lambda(0,\gamma)}, e^{-\eta\lambda(1,\gamma)} \right\rangle$ is a straight line segment.
- The only satisfactory prediction is

$$\gamma = \frac{\sum_i W_i \gamma_i}{\sum_i W_i}$$

Vovk Algorithm for log loss

- The log loss is mixable with $\eta = 1$
- The image of $[0, 1]$ through $F(\gamma) = \left\langle e^{-\eta\lambda(0,\gamma)}, e^{-\eta\lambda(1,\gamma)} \right\rangle$ is a straight line segment.
- The only satisfactory prediction is

$$\gamma = \frac{\sum_i W_i \gamma_i}{\sum_i W_i}$$

- We are back to the online Bayes algorithm.

Vovk algorithm for square loss

- The square loss is mixable with $\eta = 2$.

Vovk algorithm for square loss

- The square loss is mixable with $\eta = 2$.
- Prediction must satisfy

$$1 - \sqrt{-\frac{1}{2} \ln \sum_i V_i^t e^{-2(1-p_i^t)^2}} \leq p^t \leq \sqrt{-\frac{1}{2} \ln \sum_i V_i^t e^{-2(p_i^t)^2}}$$

where $V_i^t = \frac{W_i^t}{\sum_s W_i^s}$.

Vovk algorithm for square loss

- The square loss is mixable with $\eta = 2$.
- Prediction must satisfy

$$1 - \sqrt{-\frac{1}{2}\ln \sum_i V_i^t e^{-2(1-p_i^t)^2}} \leq p^t \leq \sqrt{-\frac{1}{2}\ln \sum_i V_i^t e^{-2(p_i^t)^2}}$$

  where $V_i^t = \frac{W_i^t}{\sum_s W_i^s}$.

- $$L_A \leq L_{\min} + \frac{1}{2}\ln N$$

Simple prediction for square loss

- We can use the prediction

$$\gamma = \frac{\sum_i W_i \gamma_i}{\sum_i W_i}$$

Simple prediction for square loss

- We can use the prediction

$$\gamma = \frac{\sum_i W_i \gamma_i}{\sum_i W_i}$$

- But in that case we must use $\eta = 1/2$ when updating the weights.

Simple prediction for square loss

- We can use the prediction

$$\gamma = \frac{\sum_i W_i \gamma_i}{\sum_i W_i}$$

- But in that case we must use $\eta = 1/2$ when updating the weights.
- Which yields the bound

$$L_A \leq L_{\min} + 2 \ln N$$

Summary of bounds for mixable losses

TRACKING THE BEST EXPERT

Loss Functions:	c values: $(\eta = 1/c)$	
	$\mathbf{pred}_{\text{wmean}}(v, x)$	$\mathbf{pred}_{\text{Vovk}}(v, x)$
$L_{\text{sq}}(p, q)$	2	1/2
$L_{\text{ent}}(p, q)$	1	1
$L_{\text{hel}}(p, q)$	1	$1/\sqrt{2}$

Figure 2. $(c, 1/c)$-realizability: c values for loss and prediction function pairin

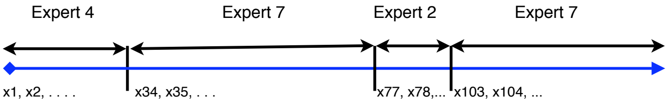
Switching experts setup

- 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

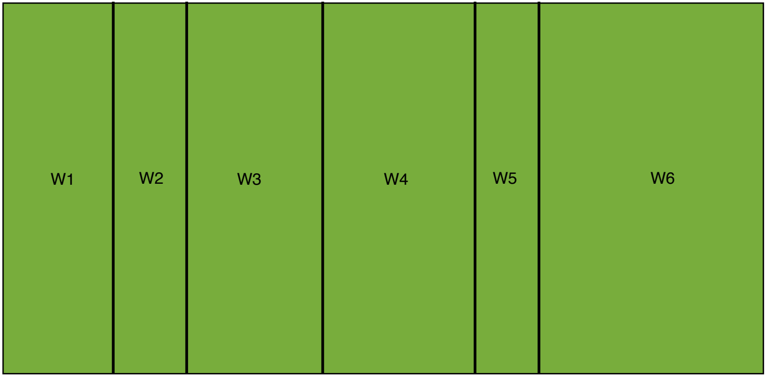
- Update weights in two stages: loss update then share update.
- Prediction uses the normalized s weights $w^s_{t,i} / \sum_j w^s_{t,j}$
- Loss update is the same as always, but defines intermediate m weights:

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

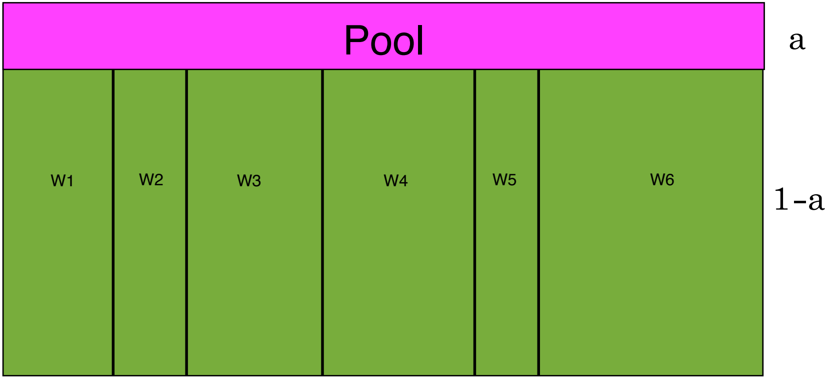
- Share update: redistribute the weights
- Fixed-share:

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

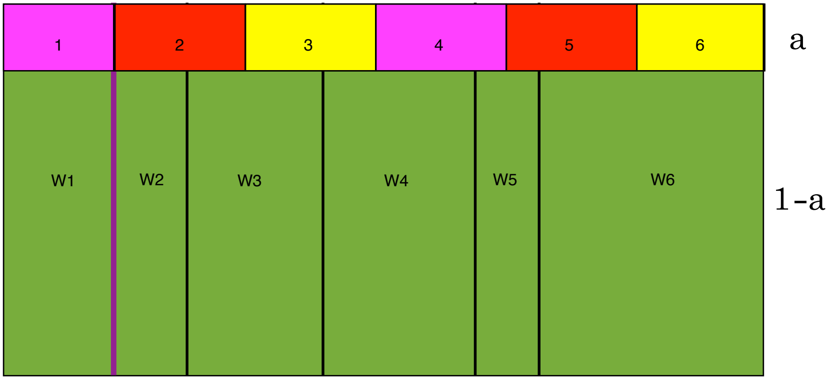
The fixed-share algorithm



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

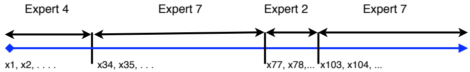
- The relation between algorithm loss and total weight does not change because share update does not change the total weight.
- Thus we still have

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

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

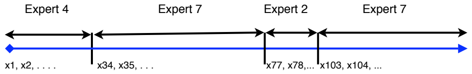
Lower bounding the final total weight

- Fix some switching experts sequence:



Lower bounding the final total weight

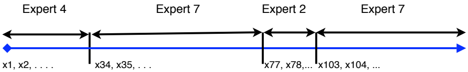
- Fix some switching experts sequence:



- "follow" the weight of the chosen expert i_t.

Lower bounding the final total weight

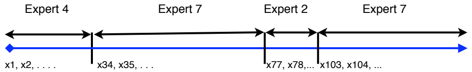
- Fix some switching experts sequence:



- "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

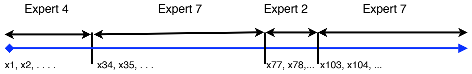
- Fix some switching experts sequence:



- "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

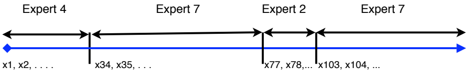
- Fix some switching experts sequence:



- "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 algorithm does not concentrate only on the best expert, even if the last switch is in the distant past.

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.

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!

Variable-share

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

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.

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.
Expert can get fraction of the total weight arbitrarily close to 1.

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 $\ell_{t,i} > 0$

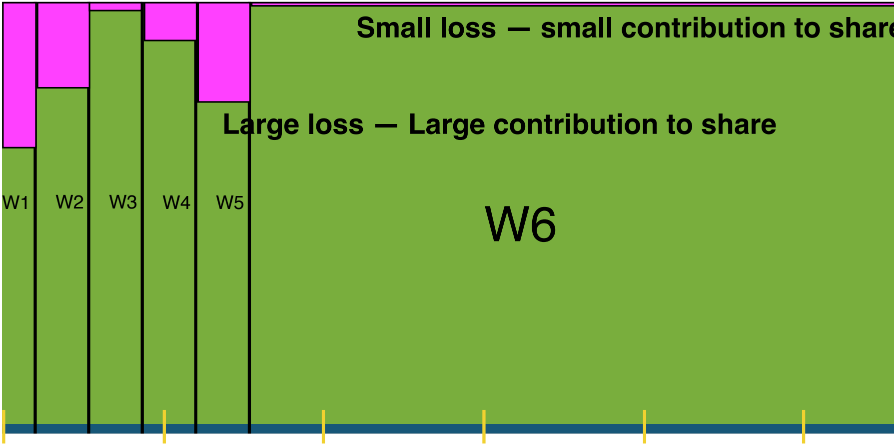
Bound for variable share

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

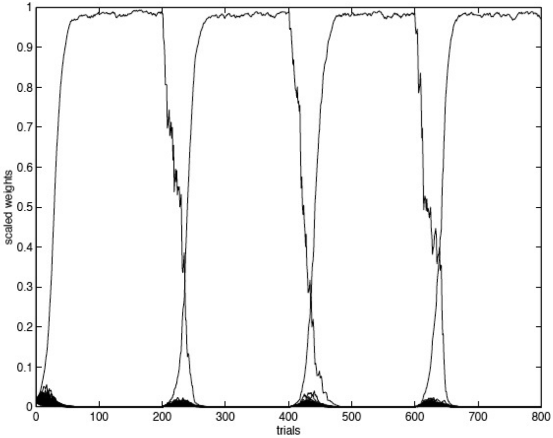
Bound for variable share

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

Variable share figure



An experiment using variable share



Next Class

- Suppose the best switching sequence is repeatedly switching among a small subset of the experts $n' \ll n$

Next Class

- 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.

Next Class

- 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 $\ln n$

Next Class

- 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.

Next Class

- 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.)

Next Class

- 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.)
- Next class - how to do as well with just one weight per expert.