Universal source coding
and
the Online Bayes algorithm

Yoav Freund

January 11, 2011
Outline

Combining experts in the log loss framework
Outline

Combining experts in the log loss framework
The online Bayes Algorithm
Outline

Combining experts in the log loss framework

The online Bayes Algorithm

The performance bound
Outline

Combining experts in the log loss framework

The online Bayes Algorithm

The performance bound

Comparison with Bayesian Statistics
Outline

Combining experts in the log loss framework
The online Bayes Algorithm
The performance bound
Comparison with Bayesian Statistics
Computational issues
Outline

Combining experts in the log loss framework
The online Bayes Algorithm
The performance bound
Comparison with Bayesian Statistics
Computational issues
The Universal prediction machine
Outline

Combining experts in the log loss framework
The online Bayes Algorithm
The performance bound
Comparison with Bayesian Statistics
Computational issues
The Universal prediction machine
The biased coins set of experts
Outline

Combining experts in the log loss framework

The online Bayes Algorithm

The performance bound

Comparison with Bayesian Statistics

Computational issues

The Universal prediction machine

The biased coins set of experts

Generalization to larger sets of distributions
The log-loss framework

- Algorithm $A$ predicts a sequence $c^1, c^2, \ldots, c^T$ over alphabet $\Sigma = \{1, 2, \ldots, k\}$
The log-loss framework

- Algorithm $A$ predicts a sequence $c^1, c^2, \ldots, c^T$ over alphabet $\Sigma = \{1, 2, \ldots, k\}$
- The prediction for the $c^t$th is a distribution over $\Sigma$: $p^t_A = \langle p^t_A(1), p^t_A(2), \ldots, p^t_A(k) \rangle$
The log-loss framework

- Algorithm $A$ predicts a sequence $c^1, c^2, \ldots, c^T$ over alphabet $\Sigma = \{1, 2, \ldots, k\}$
- The prediction for the $c^t$th is a distribution over $\Sigma$:
  \[ p^t_A = \langle p^t_A(1), p^t_A(2), \ldots, p^t_A(k) \rangle \]
- When $c^t$ is revealed, the loss we suffer is $- \log p^t_A(c^t)$
The log-loss framework

- Algorithm $A$ predicts a sequence $c^1, c^2, \ldots, c^T$ over alphabet $\Sigma = \{1, 2, \ldots, k\}$
- The prediction for the $c^t$th is a distribution over $\Sigma$: $p_A^t = \langle p_A^t(1), p_A^t(2), \ldots, p_A^t(k) \rangle$
- When $c^t$ is revealed, the loss we suffer is $-\log p_A^t(c^t)$
- The cumulative log loss, which we wish to minimize, is $L_A^T = -\sum_{t=1}^T \log p_A^t(c^t)$
Online Bayes Alg.
Combining experts in the log loss framework

### The log-loss framework

- Algorithm $A$ predicts a sequence $c^1, c^2, \ldots, c^T$ over alphabet $\Sigma = \{1, 2, \ldots, k\}$
- The prediction for the $c^t$th is a distribution over $\Sigma$:
  \[ p^t_A = \langle p^t_A(1), p^t_A(2), \ldots, p^t_A(k) \rangle \]
- When $c^t$ is revealed, the loss we suffer is $-\log p^t_A(c^t)$
- The cumulative log loss, which we wish to minimize, is
  \[ L_A^T = - \sum_{t=1}^{T} \log p^t_A(c^t) \]
- $\lceil L_A^T \rceil$ is the code length if $A$ is combined with arithmetic coding.
Online Bayes Alg.

Combining experts in the log loss framework

The game

- Prediction algorithm $A$ has access to $N$ experts.
The game

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

- Prediction algorithm $A$ has access to $N$ experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $p_1^t, \ldots, p_N^t$
Online Bayes Alg.
Combining experts in the log loss framework

The game

- Prediction algorithm $A$ has access to $N$ experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $p^t_1, \ldots, p^t_N$
  - Algorithm generates its own prediction $p^t_A$
The game

- Prediction algorithm $A$ has access to $N$ experts.
- The following is repeated for $t = 1, \ldots, T$
  - Experts generate predictive distributions: $p_1^t, \ldots, p_N^t$
  - Algorithm generates its own prediction $p_A^t$
  - $c^t$ is revealed.
Online Bayes Alg.
Combining experts in the log loss framework

The game

- Prediction algorithm \( A \) has access to \( N \) experts.
- The following is repeated for \( t = 1, \ldots, T \):
  - Experts generate predictive distributions: \( p^t_1, \ldots, p^t_N \)
  - Algorithm generates its own prediction \( p^t_A \)
  - \( c^t \) is revealed.
- **Goal:** minimize regret:

\[
- \sum_{t=1}^{T} \log p^t_A(c^t) + \min_{i=1,\ldots,N} \left( -\sum_{t=1}^{T} \log p^t_i(c^t) \right)
\]
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 $i$

  $$L^t_i = - \sum_{s=1}^{t} \log p^s_i(c^s); \quad L^0_i = 0$$

- **Weight** of expert $i$

  $$w^t_i = w^1_i e^{-L^{t-1}_i} = w^1_i \prod_{s=1}^{t-1} p^s_i(c^s)$$

- Freedom to choose initial weights.

  $$w^1_i \geq 0, \quad \sum_{i=1}^{n} w^1_i = 1$$
The online Bayes Algorithm

- **Total loss** of expert $i$

  \[
  \hat{L}_i^t = -\sum_{s=1}^{t} \log p_i^s(c^s); \quad L_i^0 = 0
  \]

- **Weight** of expert $i$

  \[
  w_t^i = w_1^i e^{-\hat{L}_i^{t-1}} = w_1^i \prod_{s=1}^{t-1} p_i^s(c^s)
  \]

- Freedom to choose initial weights.

  $w_1^i \geq 0, \sum_{i=1}^{N} w_1^i = 1$

- **Prediction** of algorithm $A$

  \[
  p_A^t = \frac{\sum_{i=1}^{N} w_t^i p_i^t}{\sum_{i=1}^{N} w_t^i}
  \]
The **Hedge**(\(\eta\)) Algorithm

Consider action \(i\) at time \(t\)

- Total loss:

\[
L_i^t = \sum_{s=1}^{t-1} \ell_i^s
\]
The **Hedge**($\eta$)Algorithm

Consider action $i$ at time $t$

- **Total loss:**
  \[
  L_i^t = \sum_{s=1}^{t-1} \ell_i^s
  \]

- **Weight:**
  \[
  w_i^t = w_i^1 e^{-\eta L_i^t}
  \]

Note freedom to choose initial weight ($w_i^1$) \(\sum_{i=1}^{n} w_i^1 = 1\).
The **Hedge**$(\eta)$**Algorithm**

Consider action $i$ at time $t$

- **Total loss:**
  \[ L_i^t = \sum_{s=1}^{t-1} \ell_{i}^{s} \]

- **Weight:**
  \[ w_i^t = w_i^1 e^{-\eta L_i^t} \]

Note freedom to choose initial weight $(w_i^1)$  $\sum_{i=1}^{n} w_i^1 = 1$.

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$
The **Hedge**($\eta$) Algorithm

Consider action $i$ at time $t$

- **Total loss:**
  \[
  L_i^t = \sum_{s=1}^{t-1} \ell_i^s
  \]

- **Weight:**
  \[
  w_i^t = w_i^1 \cdot e^{-\eta L_i^t}
  \]
  Note freedom to choose initial weight ($w_i^1$) \( \sum_{i=1}^{n} w_i^1 = 1 \).

- $\eta > 0$ is the learning rate parameter. Halving: $\eta \to \infty$

- **Probability:**
  \[
  p_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_j^t},
  \]
The Hedge\((\eta)\) Algorithm

Consider action \(i\) at time \(t\)

▶ Total loss:
\[
L_i^t = \sum_{s=1}^{t-1} \ell_i^s
\]

▶ Weight:
\[
w_i^t = w_i^1 e^{-\eta L_i^t}
\]

Note freedom to choose initial weight \((w_i^1)\) \(\sum_{i=1}^{n} w_i^1 = 1\).

▶ \(\eta > 0\) is the learning rate parameter. Halving: \(\eta \to \infty\)

▶ Probability:
\[
p_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_i^t},
\]
The **Hedge**(η) Algorithm

Consider action \(i\) at time \(t\)

- **Total loss:**
  \[
  L^t_i = \sum_{s=1}^{t-1} \ell^s_i
  \]

- **Weight:**
  \[
  w^t_i = w^1_i e^{-\eta L^t_i}
  \]

Note freedom to choose initial weight \((w^1_i)\) \(\sum_{i=1}^{n} w^1_i = 1\).

- \(\eta > 0\) is the learning rate parameter. Halving: \(\eta \to \infty\)

- **Probability:**
  \[
  p^t_i = \frac{w^t_i}{\sum_{j=1}^{N} w^t_j}, \quad p^t = \frac{w^t}{\sum_{j=1}^{N} w^t_j}
  \]
Cumulative loss vs. Final total weight

Total weight: $W^t = \sum_{i=1}^{N} w_i^t$
Cumulative loss vs. Final total weight

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

$$\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^{N} w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^{N} w_i^t}$$
Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w^t_i \)

\[
\frac{W^{t+1}}{W^t} = \frac{\sum_{i=1}^{N} w^t_i e^{\log p_i^t(c^t)}}{\sum_{i=1}^{N} w^t_i} = \frac{\sum_{i=1}^{N} w^t_i p_i^t(c^t)}{\sum_{i=1}^{N} w^t_i}
\]
Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w_i^t \)

\[
\frac{W^{t+1}}{W^{t}} = \frac{\sum_{i=1}^{N} w_i^t e^{\log p_i^t(c^t)}}{\sum_{i=1}^{N} w_i^t} = \frac{\sum_{i=1}^{N} w_i^t p_i^t(c^t)}{\sum_{i=1}^{N} w_i^t} = p_A^t(c^t)
\]
Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w_i^t \)

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

\[- \log \frac{W^{t+1}}{W^t} = - \log p_A^t(c^t)\]
Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w_i^t \)

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

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

\[-\log \frac{W^{T+1}}{W^1} = -\sum_{t=1}^{T} \log p_A^t(c^t)\]
Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w_i^t \)

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

\[ - \log \frac{W^{t+1}}{W^t} = - \log p_A^t(c_t) \]

\[ - \log \frac{W^{T+1}}{W^1} = - \sum_{t=1}^{T} \log p_A^t(c_t) = L_A^T \]
Online Bayes Alg.

The performance bound

Cumulative loss vs. Final total weight

Total weight: \( W^t = \sum_{i=1}^{N} w_i^t \)

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

\[-\log \frac{W^{t+1}}{W^t} = -\log p_A^t(c_t)\]

\[-\log W^{T+1} = -\log \frac{W^{T+1}}{W^1} = -\sum_{t=1}^{T} \log p_A^t(c_t) = L_A^T\]
Cumulative loss vs. Final total weight

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

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

$$- \log \frac{W_{t+1}}{W_t} = - \log p_A^t(c_t)$$

$$- \log W_{T+1} = - \log \frac{W_{T+1}}{W_1} = - \sum_{t=1}^{T} \log p_A^t(c_t) = L_A^T$$

**EQUALITY** not bound!
Simple Bound

- Use uniform initial weights $w_i^{1} = 1/N$
Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

\[ L_A^T = - \log W^{T+1} \]
Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

$$L_A^T = - \log W^{T+1}$$
Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

$$L_A^T = - \log W^{T+1} = - \log \sum_{i=1}^{N} w_i^{T+1}$$
Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

$$L_A^T = - \log W^{T+1} = - \log \sum_{i=1}^{N} w_i^{T+1}$$

$$= - \log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T}$$
Simple Bound

- Use uniform initial weights \( w_i^1 = 1/N \)
- Total Weight is at least the weight of the best expert.

\[
L_A^T = - \log W^{T+1} = - \log \sum_{i=1}^{N} w_i^{T+1} \\
= - \log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T}
\]
Online Bayes Alg.

The performance bound

Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

$$L_A^T = -\log W^{T+1} = -\log \sum_{i=1}^{N} w_i^{T+1}$$

$$= -\log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T}$$

$$\leq \log N - \log \max_i e^{-L_i^T}$$
Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- Total Weight is at least the weight of the best expert.

$$L_A^T = - \log W^{T+1} = - \log \sum_{i=1}^{N} w_i^{T+1}$$

$$= - \log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T}$$

$$\leq \log N - \log \max_i e^{-L_i^T} = \log N + \min_i L_i^T$$

- Dividing by $T$ we get $\frac{L_A^T}{T} = \min_i \frac{L_i^T}{T} + \frac{\log N}{T}$
Upper bound on $\sum_{i=1}^{N} w_i^{T+1}$ for $\text{Hedge}(\eta)$

Lemma (upper bound)

For any sequence of loss vectors $\ell^1, \ldots, \ell^T$ we have

$$\ln \left( \sum_{i=1}^{N} w_i^{T+1} \right) \leq -(1 - e^{-\eta}) L_{\text{Hedge}(\eta)}.$$
Tuning $\eta$ as a function of $T$

- trivially $\min_i L_i \leq T$, yielding

$$L_{\text{Hedge}}(\eta) \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$
Tuning $\eta$ as a function of $T$

- Trivially $\min_i L_i \leq T$, yielding
  $$L_{\text{Hedge}}(\eta) \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$

- Per iteration we get:
  $$\frac{L_{\text{Hedge}}(\eta)}{T} \leq \min_i \frac{L_i}{T} + \sqrt{\frac{2 \ln N}{T}} + \frac{\ln N}{T}$$
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have \( K \) copies of each expert.
- Two part code has to point to one of the \( KN \) experts
  \[
  L_A \leq \log NK + \min_i L^T_i = \log NK + \min_i L^T_i
  \]
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts
  \[ L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T \]
- If we use Bayes predictor + arithmetic coding we get:

  \[ L_A = - \log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L_i^T} = \log N + \min_i L_i^T \]
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts
  \[ L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T \]
- If we use Bayes predictor + arithmetic coding we get:
  \[ L_A = - \log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L_i^T} = \log N + \min_i L_i^T \]
- We don’t pay a penalty for copies.
Bound better than for two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts
  \[ L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T \]
- If we use Bayes predictor + arithmetic coding we get:
  \[ L_A = -\log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L_i^T} = \log N + \min_i L_i^T \]
- We don’t pay a penalty for copies.
- More generally, the regret is smaller if many of the experts perform well.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights $= \text{posterior probability distribution}$. 
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights $= \text{posterior probability distribution}$.
- Bayesian analysis interested in the final posterior.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights = posterior probability distribution.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights = posterior probability distribution.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
- Goal of Bayesian is to estimate true distribution, goal of online learning is to minimize regret.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights $= \text{posterior probability distribution}$.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
- Goal of Bayesian is to estimate true distribution, goal of online learning is to minimize regret.
- Optimality of algorithm is axiom of Bayesian statistics.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights = posterior probability distribution.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
- Goal of Bayesian is to estimate true distribution, goal of online learning is to minimize regret.
- Optimality of algorithm is axiom of Bayesian statistics.
- Bayesian methods perform poorly when the loss is not log loss and the data not generated by a distribution in the support.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights = posterior probability distribution.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
- Goal of Bayesian is to estimate true distribution, goal of online learning is to minimize regret.
- Optimality of algorithm is axiom of Bayesian statistics.
- Bayesian methods perform poorly when the loss is not log loss and the data not generated by a distribution in the support.
  - Loss can sometimes be defined through the noise distribution: square loss is equivalent to assuming guassian noise.
Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights = posterior probability distribution.
- Bayesian analysis interested in the final posterior.
- Bayesian analysis assumes the data is generated by a distribution in the support of the prior.
- Goal of Bayesian is to estimate true distribution, goal of online learning is to minimize regret.
- Optimality of algorithm is axiom of Bayesian statistics.
- Bayesian methods perform poorly when the loss is not log loss and the data not generated by a distribution in the support.
  - Loss can sometimes be defined through the noise distribution: square loss is equivalent to assuming guassian noise.
  - For number of mistakes - Bayesian method cannot be “fixed”. Requires variable learning rate.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.

- Bayesian tricks:
  - Conjugate priors: A prior over a continuous domain whose functional form does not change when updated. The number of parameters defining the posterior is constant. The update rule translates into an update of parameters, which correspond to "sufficient statistics".
  - Markov Chain Monte Carlo: Sample the posterior. Sometimes can be done efficiently. Efficient sampling relates to the mixing rate of the Markov chain whose limiting distribution is the posterior.
Online Bayes Alg.

Computational issues

Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters. Parameters correspond to “sufficient statistics”.

Markov Chain Monte Carlo
Sample the posterior. Can sometimes be done efficiently. Efficient sampling relates to mixing rate of markov chain whose limit dist is the posterior dist.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters. parameters correspond to “sufficient statistics”. exists for the familty of exponential distributions.
  - **Markov Chain Monte Carlo**: Sample the posterior.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters. parameters correspond to “sufficient statistics”. exists for the family of exponential distributions.
  - **Markov Chain Monte Carlo** Sample the posterior.
Online Bayes Alg.

Computational issues

Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters. Parameters correspond to “sufficient statistics”. Exists for the family of exponential distributions.
  - **Markov Chain Monte Carlo** Sample the posterior. Can sometimes be done efficiently.
Computational Issues

- Naive implementation: calculate the prediction of each of the $N$ experts.
- Puts severe limit on number of experts.
- What if set of experts is uncountably infinite.
- Bayesian tricks:
  - **Conjugate priors**: A prior over a continuous domain whose functional form does not change with when updated. Number of parameters defining posterior is constant. Update rule translates into update of parameters. Parameters correspond to “sufficient statistics”. Exists for the family of exponential distributions.
  - **Markov Chain Monte Carlo**: Sample the posterior. Can sometimes be done efficiently. Efficient sampling relates to mixing rate of markov chain whose limit dist is the posterior dist.
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$. 

\[ V(\vec{b}, \vec{X}, t) = 1 \text{ if the program } \vec{b} \text{, given } \vec{X} \text{ as input, halts within } t \text{ steps and outputs a well-formed prediction.} \]

\[ V(\vec{b}, \vec{X}, t) = 0 \] is computable (recursively enumerable).
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that...
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\tilde{X} \in \{0, 1\}^t$
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\vec{X} \in \{0, 1\}^t$
  - runs finite time and outputs
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\vec{X} \in \{0, 1\}^t$
  - runs finite time and outputs
  - A prediction for the next bit $p(\vec{X}) \in [0, 1]$.
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\vec{X} \in \{0, 1\}^t$
  - runs finite time and outputs
  - A prediction for the next bit $p(\vec{X}) \in [0, 1]$.
  - To ensure $p$ has a finite description. Restrict to rational numbers $n/m$
Standardizing online prediction algorithms

- Fix a universal Turing machine \( U \).
- An online prediction algorithm \( E \) is a program that
  - given as input The past \( \vec{X} \in \{0, 1\}^t \)
  - runs finite time and outputs
  - A prediction for the next bit \( p(\vec{X}) \in [0, 1] \).
  - To ensure \( p \) has a finite description. Restrict to rational numbers \( n/m \)
- Any online prediction algorithm can be represented as code \( \vec{b}(E) \) for \( U \). The code length is \( |\vec{b}(E)| \).
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\vec{X} \in \{0, 1\}^t$
  - runs finite time and outputs
  - A prediction for the next bit $p(\vec{X}) \in [0, 1]$.
  - To ensure $p$ has a finite description. Restrict to rational numbers $n/m$
- Any online prediction algorithm can be represented as code $\vec{b}(E)$ for $U$. The code length is $|\vec{b}(E)|$.
- Most sequences do not correspond to valid prediction algorithms.
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\vec{X} \in \{0, 1\}^t$
  - runs finite time and outputs
  - A prediction for the next bit $p(\vec{X}) \in [0, 1]$.
  - To ensure $p$ has a finite description. Restrict to rational numbers $n/m$
- Any online prediction algorithm can be represented as code $\vec{b}(E)$ for $U$. The code length is $|\vec{b}(E)|$.
- Most sequences do not correspond to valid prediction algorithms.
- $V(\vec{b}, \vec{X}, t) = 1$ if the program $\vec{b}$, given $\vec{X}$ as input, halts within $t$ steps and outputs a well-formed prediction. Otherwise $V(\vec{b}, \vec{X}, t) = 0$
Standardizing online prediction algorithms

- Fix a universal Turing machine $U$.
- An online prediction algorithm $E$ is a program that
  - given as input The past $\tilde{X} \in \{0, 1\}^t$
  - runs finite time and outputs
  - A prediction for the next bit $p(\tilde{X}) \in [0, 1]$.
  - To ensure $p$ has a finite description. Restrict to rational numbers $n/m$
- Any online prediction algorithm can be represented as code $\vec{b}(E)$ for $U$. The code length is $|\vec{b}(E)|$.
- Most sequences do not correspond to valid prediction algorithms.
- $V(\vec{b}, \tilde{X}, t) = 1$ if the program $\vec{b}$, given $\tilde{X}$ as input, halts within $t$ steps and outputs a well-formed prediction. Otherwise $V(\vec{b}, \tilde{X}, t) = 0$
- $V(\vec{b}, \tilde{X}, t)$ is computable (recursively enumerable).
Assign to the code $\vec{b}$ the initial weight $w^1_{\vec{b}} = 2^{-|\vec{b}| - \log_2 |\vec{b}|}$. 
A universal prediction machine

- Assign to the code $\vec{b}$ the initial weight $w^1_{\vec{b}} = 2^{-|\vec{b}| - \log_2 |\vec{b}|}$.
- The total initial weight over all finite binary sequences is one.
A universal prediction machine

- Assign to the code $\vec{b}$ the initial weight $w_{\vec{b}}^1 = 2^{-|\vec{b}| - \log_2 |\vec{b}|}$.
- The total initial weight over all finite binary sequences is one.
- Run the Bayes algorithm over “all” prediction algorithms.
A universal prediction machine

- Assign to the code $\vec{b}$ the initial weight $w_{\vec{b}}^1 = 2^{-|\vec{b}| - \log_2 |\vec{b}|}$.
- The total initial weight over all finite binary sequences is one.
- Run the Bayes algorithm over “all” prediction algorithms.

**technical details:** On iteration $t$, $|\vec{X}| = t$. Use the predictions of programs $\vec{b}$ such that $|\vec{b}| \leq t$ and for which $V(\vec{b}, \vec{X}, 2^t) = 1$. Assign the remaining mass the prediction $1/2$ (insuring a loss of 1)
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i^1)$
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i^1)$
- Assume $E$ is a prediction algorithm which generates the $t$th prediction in time smaller than $2^t$
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i^1)$
- Assume $E$ is a prediction algorithm which generates the $t$th prediction in time smaller than $2^t$
- When $t \leq |\bar{b}(E)|$ the algorithm is not used and thus it’s loss is 1
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i)$
- Assume $E$ is a prediction algorithm which generates the $t$th prediction in time smaller than $2^t$
- When $t \leq |\tilde{b}(E)|$ the algorithm is not used and thus it’s loss is 1
- We get that the loss of the Universal algorithm is at most $2|\tilde{b}(E)| + \log_2 |\tilde{b}(E)| + L_E$
Performance of the universal prediction algorithm

- Using \( L_A \leq \min_i (L_i - \log w_i^1) \)
- Assume \( E \) is a prediction algorithm which generates the \( t \)th prediction in time smaller than \( 2^t \)
- When \( t \leq |\tilde{b}(E)| \) the algorithm is not used and thus it’s loss is 1
- We get that the loss of the Universal algorithm is at most \( 2|\tilde{b}(E)| + \log_2 |\tilde{b}(E)| + L_E \)
- More careful analysis can reduce \( 2|\tilde{b}(E)| + \log_2 |\tilde{b}(E)| \) to \( |\tilde{b}(E)| \)
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i^1)$
- Assume $E$ is a prediction algorithm which generates the $t$th prediction in time smaller than $2^t$
- When $t \leq |\vec{b}(E)|$ the algorithm is not used and thus it’s loss is 1
- We get that the loss of the Universal algorithm is at most $2|\vec{b}(E)| + \log_2 |\vec{b}(E)| + L_E$
- More careful analysis can reduce $2|\vec{b}(E)| + \log_2 |\vec{b}(E)|$ to $|\vec{b}(E)|$
- Code length is arbitrarily close to the Kolmogorov Complexity of the sequence.
Performance of the universal prediction algorithm

- Using $L_A \leq \min_i (L_i - \log w_i^1)$
- Assume $E$ is a prediction algorithm which generates the $t$th prediction in time smaller than $2^t$
- When $t \leq |\vec{b}(E)|$ the algorithm is not used and thus it’s loss is 1
- We get that the loss of the Universal algorithm is at most $2|\vec{b}(E)| + \log_2 |\vec{b}(E)| + L_E$
- More careful analysis can reduce $2|\vec{b}(E)| + \log_2 |\vec{b}(E)|$ to $|\vec{b}(E)|$
- Code length is arbitrarily close to the Kolmogorov Complexity of the sequence.
- Ridiculously bad running time.
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts
  \[ L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T \]
  If we use Bayes predictor + arithmetic coding we get:
  \[ L_A = -\log W_T + 1 \leq \log K \max_i 1 \frac{1}{NK e} - L_i^T = \log N + \min_i L_i^T \]
- We don't pay a penalty for copies.
- More generally, the regret is smaller if many of the experts perform well.
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts:
  $$L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T$$
- If we use Bayes predictor + arithmetic coding we get:
  $$L_A = -\log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L_i^T} = \log N + \min_i L_i^T$$
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts
  
  \[ L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T \]

- If we use Bayes predictor + arithmetic coding we get:

  \[ L_A = - \log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L_i^T} = \log N + \min_i L_i^T \]

- We don’t pay a penalty for copies.
Bayes coding is better than two part codes

- Simple bound as good as bound for two part codes (MDL) but enables online compression
- Suppose we have $K$ copies of each expert.
- Two part code has to point to one of the $KN$ experts:
  $$L_A \leq \log NK + \min_i L^T_i = \log NK + \min_i L^T_i$$
- If we use Bayes predictor + arithmetic coding we get:
  $$L_A = -\log W^{T+1} \leq \log K \max_i \frac{1}{NK} e^{-L^T_i} = \log N + \min_i L^T_i$$
- We don’t pay a penalty for copies.
- More generally, the regret is smaller if many of the experts perform well.
The biased coins set of experts

► Each expert corresponds to a biased coin, predicts with a fixed $\theta \in [0, 1]$. 

Online Bayes Alg.

The biased coins set of experts
The biased coins set of experts

- Each expert corresponds to a biased coin, predicts with a fixed \( \theta \in [0, 1] \).
- Set of experts is uncountably infinite.
The biased coins set of experts

- Each expert corresponds to a biased coin, predicts with a fixed $\theta \in [0, 1]$.
- Set of experts is uncountably infinite.
- Only countably many experts can be assigned non-zero weight.
The biased coins set of experts

- Each expert corresponds to a biased coin, predicts with a fixed $\theta \in [0, 1]$.
- Set of experts is uncountably infinite.
- Only countably many experts can be assigned non-zero weight.
- Instead, we assign the experts a Density Measure.
The biased coins set of experts

- Each expert corresponds to a biased coin, predicts with a fixed $\theta \in [0, 1]$.
- Set of experts is uncountably infinite.
- Only countably many experts can be assigned non-zero weight.
- Instead, we assign the experts a Density Measure.
- $L_A \leq \min_i (L_i - \log w^1_i)$ is meaningless.
The biased coins set of experts

- Each expert corresponds to a biased coin, predicts with a fixed $\theta \in [0, 1]$.
- Set of experts is uncountably infinite.
- Only countably many experts can be assigned non-zero weight.
- Instead, we assign the experts a Density Measure.
- $L_A \leq \min_i (L_i - \log w_i^1)$ is meaningless.
- Can we still get a meaningful bound?
Bayes Algorithm for biased coins

- Replace the initial weight by a density measure
  \[ w(\theta) = w^1(\theta), \int_0^1 w(\theta) d\theta = 1 \]
Bayes Algorithm for biased coins

- Replace the initial weight by a density measure
  \[ w(\theta) = w^1(\theta), \int_0^1 w(\theta) d\theta = 1 \]

- Relationship between final total weight and total log loss remains unchanged:
  \[ L_A = \ln \int_0^1 w(\theta) e^{-L_{T+1}} d\theta \]
Bayes Algorithm for biased coins

- Replace the initial weight by a density measure
  \[ w(\theta) = w^1(\theta), \int_0^1 w(\theta) d\theta = 1 \]

- Relationship between final total weight and total log loss remains unchanged:
  \[ L_A = \ln \int_0^1 w(\theta) e^{-L^T_{\theta} + 1} d\theta \]

- We need a new lower bound on the final total weight
Main Idea

If $w^t(\theta)$ is large then $w^t(\theta + \epsilon)$ is also large.
Main Idea

If $w_t(\theta)$ is large then $w_t(\theta + \epsilon)$ is also large.
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.
  \[
  \hat{\theta} = \frac{\#\{x^t = 1; \ 1 \leq t \leq T\}}{T}
  \]
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.
  $$\hat{\theta} = \frac{\# \{ x^t = 1; 1 \leq t \leq T \}}{T}$$

- The total loss scales with $T$
  $$L_\theta = T \cdot (\hat{\theta} \ell(\theta, 1) + (1 - \hat{\theta}) \ell(\theta, 0)) \equiv T \cdot g(\hat{\theta}, \theta)$$
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.
  
  $$\hat{\theta} = \frac{\# \{ x^t = 1; 1 \leq t \leq T \}}{T}$$

- The total loss scales with $T$
  
  $$L_\theta = T \cdot (\hat{\theta} \ell(\theta, 1) + (1 - \hat{\theta}) \ell(\theta, 0)) \doteq T \cdot g(\hat{\theta}, \theta)$$
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.
  \[ \hat{\theta} = \frac{\#\{x^t = 1; 1 \leq t \leq T\}}{T} \]

- The total loss scales with $T$
  \[ L_{\theta} = T \cdot (\hat{\theta} \ell(\theta, 1) + (1 - \hat{\theta}) \ell(\theta, 0)) \approx T \cdot g(\hat{\theta}, \theta) \]

\[ L_A - L_{\min} \leq \ln \int_{0}^{1} w(\theta) e^{-L_{\theta}} d\theta - \ln e^{L_{\min}} \]
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.

$$\hat{\theta} = \frac{\#\{x^t = 1; 1 \leq t \leq T\}}{T}$$

- The total loss scales with $T$

$$L_{\theta} = T \cdot (\hat{\theta} \ell(\theta, 1) + (1 - \hat{\theta}) \ell(\theta, 0)) \equiv T \cdot g(\hat{\theta}, \theta)$$

$$L_A - L_{\text{min}} \leq \ln \int_0^1 w(\theta) e^{-L_{\theta}} d\theta - \ln e^{L_{\text{min}}}$$

$$= \ln \int_0^1 w(\theta) e^{-(L_{\theta} - L_{\text{min}})} d\theta$$
Expanding the exponent around the peak

- For log loss the best $\theta$ is empirical distribution of the seq.

$$\hat{\theta} = \frac{\#\{x^t = 1; \ 1 \leq t \leq T\}}{T}$$

- The total loss scales with $T$

$$L_\theta = T \cdot (\hat{\theta}\ell(\theta, 1) + (1 - \hat{\theta})\ell(\theta, 0)) \doteq T \cdot g(\hat{\theta}, \theta)$$

$$L_A - L_{\min} \leq \ln \int_0^1 w(\theta) e^{-L_\theta} d\theta - \ln e^{L_{\min}}$$

$$= \ln \int_0^1 w(\theta) e^{-(L_\theta - L_{\min})} d\theta$$

$$= \ln \int_0^1 w(\theta) e^{T(g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))} d\theta$$
Laplace approximation (idea)

- Taylor expansion of \( g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}) \) around \( \theta = \hat{\theta} \).
Laplace approximation (idea)

- Taylor expansion of $g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})$ around $\theta = \hat{\theta}$.
- First and second terms in the expansion are zero.
Laplace approximation (idea)

- Taylor expansion of $g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})$ around $\theta = \hat{\theta}$.
- First and second terms in the expansion are zero.
- Third term gives a quadratic expression in the exponent
Laplace approximation (idea)

- Taylor expansion of $g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})$ around $\theta = \hat{\theta}$.
- First and second terms in the expansion are zero.
- Third term gives a quadratic expression in the exponent.
- $\Rightarrow$ a gaussian approximation of the posterior.
Laplace approximation (idea)

- Taylor expansion of \( g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}) \) around \( \theta = \hat{\theta} \).
- First and second terms in the expansion are zero.
- Third term gives a quadratic expression in the exponent.
- \( \Rightarrow \) a gaussian approximation of the posterior.
Laplace approximation (idea)

- Taylor expansion of $g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})$ around $\theta = \hat{\theta}$.
- First and second terms in the expansion are zero.
- Third term gives a quadratic expression in the exponent.
- $\Rightarrow$ a gaussian approximation of the posterior.
Laplace Approximation (details)

\[
\int_{0}^{1} w(\theta) e^{T(g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))} d\theta
\]
Laplace Approximation (details)

\[
\int_0^1 w(\theta) e^{T(g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))} d\theta \\
= w(\hat{\theta}) \sqrt{\frac{-2\pi}{T \left. \frac{d^2}{d\theta^2} \right|_{\theta=\hat{\theta}} (g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))}} + O(T^{-3/2})
\]
Choosing the optimal prior

- Choose $w(\theta)$ to maximize the worst-case final total weight

$$\min_{\hat{\theta}} w(\hat{\theta}) \sqrt{T \frac{d^2}{d\theta^2} \bigg|_{\theta=\hat{\theta}} (g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))}$$
Choosing the optimal prior

Choose $w(\theta)$ to maximize the worst-case final total weight

$$\min_{\hat{\theta}} w(\hat{\theta}) \sqrt{T \left. \frac{d^2}{d\theta^2} \right|_{\theta=\hat{\theta}} (g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})) ^{-2\pi}}$$

Make bound equal for all $\hat{\theta} \in [0, 1]$ by choosing

$$w^*(\hat{\theta}) = \frac{1}{Z} \sqrt{\left. \frac{d^2}{d\theta^2} \right|_{\theta=\hat{\theta}} (g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta})) ^{-2\pi}} ,$$

where $Z$ is the normalization factor:

$$Z = \sqrt{\frac{1}{2\pi}} \int_0^1 \sqrt{\left. \frac{d^2}{d\theta^2} \right|_{\theta=\hat{\theta}} (g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))} \ d\hat{\theta}$$
The bound for the optimal prior

Plugging in we get

\[ L_A - L_{\text{min}} \leq \ln \int_0^1 w^*(\theta) e^{T(g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}))} d\theta \]

\[ = \ln \left( \sqrt{\frac{2\pi Z}{T}} + O(T^{-3/2}) \right) \]

\[ = \frac{1}{2} \ln \frac{T}{2\pi} - \frac{1}{2} \ln Z + O(1/T). \]
Online Bayes Alg.
— The biased coins set of experts

Solving for log-loss

- The exponent in the integral is

$$g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}) = \hat{\theta} \ln \frac{\hat{\theta}}{\theta} + (1 - \hat{\theta}) \ln \frac{1 - \hat{\theta}}{1 - \theta} = D_{KL}(\hat{\theta}||\theta)$$
Solving for log-loss

- The exponent in the integral is

\[ g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}) = \hat{\theta} \ln \frac{\hat{\theta}}{\theta} + (1 - \hat{\theta}) \ln \frac{1 - \hat{\theta}}{1 - \theta} = D_{KL}(\hat{\theta}||\theta) \]

- The second derivative

\[ \frac{d^2}{d\theta^2} D_{KL}(\hat{\theta}||\theta) \bigg|_{\theta = \hat{\theta}} = \hat{\theta}(1 - \hat{\theta}) \]

Is called the empirical Fisher information
Solving for log-loss

- The exponent in the integral is

\[
g(\hat{\theta}, \theta) - g(\hat{\theta}, \hat{\theta}) = \hat{\theta} \ln \frac{\hat{\theta}}{\theta} + (1 - \hat{\theta}) \ln \frac{1 - \hat{\theta}}{1 - \theta} = D_{KL}(\hat{\theta} || \theta)
\]

- The second derivative

\[
\frac{d^2}{d\theta^2} \bigg|_{\theta=\hat{\theta}} D_{KL}(\hat{\theta} || \theta) = \hat{\theta}(1 - \hat{\theta})
\]

is called the empirical Fisher information.

- The optimal prior:

\[
w^*(\hat{\theta}) = \frac{1}{\pi \sqrt{\hat{\theta}(1 - \hat{\theta})}}
\]

Known in general as Jeffrey’s prior. And, in this case, the Dirichlet-(1/2, 1/2) prior.
The cumulative log loss of Bayes using Jeffrey’s prior

\[ L_A - L_{\text{min}} \leq \frac{1}{2} \ln(T + 1) + \frac{1}{2} \ln \frac{\pi}{2} + O(1/T) \]
Online Bayes Alg.

The biased coins set of experts

But what is the prediction rule?

- As luck would have it the Dirichlet prior is the **conjugate prior** for the Binomial distribution.
But what is the prediction rule?

- As luck would have it the Dirichlet prior is the conjugate prior for the Binomial distribution.
- Observed \( t \) bits, \( n \) of which were 1. The posterior is:

\[
\frac{1}{Z \sqrt{\theta (1-\theta)}} \theta^n (1-\theta)^{t-n} = \frac{1}{Z} \theta^{n-1/2} (1-\theta)^{t-n-1/2}
\]
But what is the prediction rule?

- As luck would have it the Dirichlet prior is the **conjugate prior** for the Binomial distribution.
- Observed $t$ bits, $n$ of which were 1. The posterior is:

$$
\frac{1}{Z} \frac{\theta^n (1 - \theta)^{t-n}}{\sqrt{\theta(1 - \theta)}} = \frac{1}{Z} \theta^{n-1/2} (1 - \theta)^{t-n-1/2}
$$

- The posterior average is:

$$
\frac{\int_0^1 \theta^{n+1/2} (1 - \theta)^{t-n-1/2} d\theta}{\int_0^1 \theta^{n-1/2} (1 - \theta)^{t-n-1/2} d\theta} = \frac{n + 1/2}{t + 1}
$$
But what is the prediction rule?

- As luck would have it the Dirichlet prior is the conjugate prior for the Binomial distribution.
- Observed $t$ bits, $n$ of which were 1. The posterior is:
  $$\frac{1}{Z \sqrt{\theta(1 - \theta)}} \theta^n (1 - \theta)^{t-n} = \frac{1}{Z} \theta^{n-1/2} (1 - \theta)^{t-n-1/2}$$
- The posterior average is:
  $$\int_0^1 \frac{\theta^{n+1/2} (1 - \theta)^{t-n-1/2}}{\int_0^1 \theta^{n-1/2} (1 - \theta)^{t-n-1/2} d\theta} d\theta = \frac{n + 1/2}{t + 1}$$
- This is called the Trichevsky Trofimov prediction rule.
Laplace Rule of Succession

- Laplace suggested using the uniform prior, which is also a conjugate prior.
Laplace Rule of Succession

- Laplace suggested using the uniform prior, which is also a conjugate prior.
- In this case the posterior average is:

\[
\frac{\int_0^1 \theta^{n+1}(1 - \theta)^{t-n} d\theta}{\int_0^1 \theta^n(1 - \theta)^{t-n} d\theta} = \frac{n + 1}{t + 2}
\]
Laplace Rule of Succession

- Laplace suggested using the uniform prior, which is also a conjugate prior.
- In this case the posterior average is:

  \[
  \frac{\int_{0}^{1} \theta^{n+1}(1 - \theta)^{t-n} d\theta}{\int_{0}^{1} \theta^{n}(1 - \theta)^{t-n} d\theta} = \frac{n + 1}{t + 2}
  \]

- The bound on the cumulative log loss is worse:

  \[
  L_A - L_{\text{min}} = \ln T + O(1)
  \]
Laplace Rule of Succession

- Laplace suggested using the uniform prior, which is also a conjugate prior.
- In this case the posterior average is:

\[
\int_0^1 \frac{\theta^{n+1}(1 - \theta)^{t-n}}{\int_0^1 \theta^n(1 - \theta)^{t-n} d\theta}\ d\theta = \frac{n + 1}{t + 2}
\]

- The bound on the cumulative log loss is worse:

\[
L_A - L_{\text{min}} = \ln T + O(1)
\]

- Suffers larger regret when \( \hat{\theta} \) is far from \( 1/2 \)
What is the optimal prediction when $T$ is known in advance?
What is the **optimal** prediction when $T$ is known in advance?

\[
L^T_\star - \min_{\theta} L^T_\theta \geq \frac{1}{2} \ln (T + 1) + \frac{1}{2} \ln \frac{\pi}{2} - O\left(\frac{1}{\sqrt{T}}\right)
\]
Multinomial Distributions

- For a distribution over $k$ elements (Multinomial) [Xie and Barron]
Online Bayes Alg.

Generalization to larger sets of distributions

Multinomial Distributions

- For a distribution over $k$ elements (Multinomial) [Xie and Barron]
- Use the add 1/2 rule (KT).

$$p(i) = \frac{n_i + 1/2}{t + k/2}$$
Multinomial Distributions

- For a distribution over \( k \) elements (Multinomial) [Xie and Barron]
- Use the add 1/2 rule (KT).

\[
p(i) = \frac{n_i + 1/2}{t + k/2}
\]

- Bound is

\[
L_A - L_{\text{min}} \leq \frac{k - 1}{2} \ln T + C + o(1)
\]

The constant \( C \) is optimal.
Multinomial Distributions

- For a distribution over $k$ elements (Multinomial) [Xie and Barron]
- Use the add 1/2 rule (KT).

$$p(i) = \frac{n_i + 1/2}{t + k/2}$$

- Bound is

$$L_A - L_{\text{min}} \leq \frac{k - 1}{2} \ln T + C + o(1)$$

- The constant $C$ is optimal.
Expontential Distributions

- For any set of distributions from the exponential family defined by $k$ parameters (constituting an open set) [Rissanen96]
Exponential Distributions

- For any set of distributions from the exponential family defined by $k$ parameters (constituting an open set) [Rissanen96]
- Use Bayes Algorithm with Jeffrey’s prior:

$$w^*(\hat{\theta}) = \frac{1}{Z} \frac{1}{\sqrt{|H(D_{KL}(\hat{\theta}||\theta))|_{\theta=\hat{\theta}}}}$$

$H$ denotes the Hessian.
Exponential Distributions

- For any set of distributions from the exponential family defined by $k$ parameters (constituting an open set) [Rissanen96]

- Use Bayes Algorithm with Jeffrey’s prior:

$$w^*(\hat{\theta}) = \frac{1}{Z} \frac{1}{\sqrt{H(D_{KL}(\hat{\theta}||\theta))|_{\theta=\hat{\theta}}}}$$

$H$ denotes the Hessian.

- $L_A - L_{min} \leq \frac{k - 1}{2} \ln T - \ln Z + o(1)$
General Distributions

- Characterize distribution family by metric entropy.
General Distributions

- Characterize distribution family by metric entropy.
- Fixed parameter set usually corresponds to polynomial metric entropy

\[ N(1/\epsilon) = O\left(\frac{1}{\epsilon^d}\right) \]

\(d\) is the number of parameters.

[Haussler and Opper] show that the coefficient in front of \(\ln T\) is optimal for distribution families where the metric entropy is up to \(N(1/\epsilon) = O(e^{\epsilon - \alpha})\) for all \(\alpha \leq 5/2\).
General Distributions

- Characterize distribution family by metric entropy.
- Fixed parameter set usually corresponds to polynomial metric entropy

\[ N(1/\varepsilon) = O\left( \frac{1}{\varepsilon^d} \right) \]

\( d \) is the number of parameters.

- [Haussler and Opper] show that the coefficient in front of \( \ln T \) is optimal for distribution families where the metric entropy is up to

\[ N(1/\varepsilon) = O\left( e^{\varepsilon^{-\alpha}} \right) \]

For all \( \alpha \leq 5/2 \).