Universal source coding and the Online Bayes algorithm

Yoav Freund

January 19, 2006

Outline

Combining experts in the log loss framework

Outline

Outline

Combining experts in the log loss framework

The online Bayes Algorithm
- Comparison to **Hedge**(η)

The performance bound
- Comparison to **Hedge**(η)
- Advantage over two part codes

Outline

Outline

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^tth is a distribution over Σ: $\mathbf{p}_A^t = \langle p_A^t(1), p_A^t(2), \ldots, p_A^t(k) \rangle$

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^tth is a distribution over Σ: $\mathbf{p}_A^t = \langle p_A^t(1), p_A^t(2), \ldots, p_A^t(k)\rangle$
- When c^t is revealed, the loss we suffer is $-\log p_A^t(c^t)$

The 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^tth is a distribution over Σ: $\mathbf{p}_A^t = \langle p_A^t(1), p_A^t(2), \ldots, p_A^t(k) \rangle$
- 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 $L_A{}^T = -\sum_{t=1}^T \log p_A^t(c^t)$

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^tth is a distribution over Σ: $\mathbf{p}_A^t = \langle p_A^t(1), p_A^t(2), \ldots, p_A^t(k) \rangle$
- 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 $L_A{}^T = -\sum_{t=1}^T \log p_A^t(c^t)$
- $\lceil L_A{}^T \rceil$ is the code length if A is combined with arithmetic coding.

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: $\mathbf{p}_1^t, \ldots, \mathbf{p}_N^t$

The 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 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 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

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

The **Hedge**(η)Algorithm

Consider action i at time t

- Total loss:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

The **Hedge**(η)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**(η)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**(η)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:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t},$$

The **Hedge**(η)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:

$$L_i^t = \sum_{s=1}^{t-1} \ell_i^s$$

- 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:

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^{N} w_i^t}, \quad \mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{j=1}^{N} w_i^t}$$

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!

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.

$$
\begin{aligned}
L_A^T &= -\log W^{T+1} = -\log \sum_{i=1}^{N} w_i^{T+1} \\
&= -\log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T}
\end{aligned}
$$

Simple Bound

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

$$
\begin{aligned}
L_A^T &= -\log W^{T+1} = -\log \sum_{i=1}^{N} w_i^{T+1} \\
&= -\log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T}
\end{aligned}
$$

Simple Bound

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

$$
\begin{aligned}
L_A^T &= -\log W^{T+1} = -\log \sum_{i=1}^{N} w_i^{T+1} \\
&= -\log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T} \\
&\leq \log N - \log \max_i e^{-L_i^T}
\end{aligned}
$$

Simple Bound

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

$$\begin{aligned} L_A^T &= -\log W^{T+1} = -\log \sum_{i=1}^{N} w_i^{T+1} \\ &= -\log \sum_{i=1}^{N} \frac{1}{N} e^{-L_i^T} = \log N - \log \sum_{i=1}^{N} e^{-L_i^T} \\ &\leq \log N - \log \max_i e^{-L_i^T} = \log N + \min_i L_i^T \end{aligned}$$

- 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 **Hedge**(η)

Lemma (upper bound)

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

$$\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right) \leq -(1 - e^{-\eta}) L_{\textbf{Hedge}(\eta)}.$$

Tuning η as a function of T

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

$$L_{\textbf{Hedge}(\eta)} \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$

Tuning η as a function of T

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

$$L_{\textbf{Hedge}(\eta)} \leq \min_i L_i + \sqrt{2T \ln N} + \ln N$$

- per iteration we get:

$$\frac{L_{\textbf{Hedge}(\eta)}}{T} \leq \min_i \frac{L_i}{T} + \sqrt{\frac{2 \ln N}{T}} + \frac{\ln N}{T}$$

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
  $L_A \leq \log NK + \min_i L_i^T = \log NK + \min_i L_i^T$

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:

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

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 = **posterior probability distribution.**

Comparison with standard Bayesian statistics

- The weight update rule is the same.
- Normalized weights $=$ **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 $=$ **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.

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”.

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 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 familty 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 famiIty 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.

Next class

- How to deal with an uncountably infinite class of models.

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:
  - Register on TWiki and update your information.

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:
  - Register on TWiki and update your information.
  - If you are taking the class for 4 points, start a project page.

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:
  - Register on TWiki and update your information.
  - If you are taking the class for 4 points, start a project page.
- For EXTRA credit

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:
  - Register on TWiki and update your information.
  - If you are taking the class for 4 points, start a project page.
- For EXTRA credit
  - Post (and answer) questions.

Next class

- How to deal with an uncountably infinite class of models.
- To maintain Satisfactory status in class:
  - Register on TWiki and update your information.
  - If you are taking the class for 4 points, start a project page.
- For EXTRA credit
  - Post (and answer) questions.
  - Read the background material I put on the class twiki page (for class no. 5)