Universal source coding and the Online Bayes algorithm

Yoav Freund

January 20, 2010

Outline

The source compression problem

- **Example:** "There are no people like show people"
  $\stackrel{\text{encode}}{\longrightarrow} x \in \{0,1\}^n$
  $\stackrel{\text{decode}}{\longrightarrow}$ "there are no people like show people"
- **Lossless:** Message reconstructed perfectly.
- **Goal:** minimize expected length $E(n)$ of coded message.
- Can we do better than $\lceil \log_2(26) \rceil = 5$ bits per character?
- **Basic idea:** Use short codes for common messages.
- **Stream compression:**
  - Message revealed one character at a time.
  - Code generated as message is revealed.
  - Decoded message is constructed gradually.
- Easier than block codes when processing long messages.
- A natural way for describing a distribution.

The Guessing game

- Message reveraled one character at a time
- An algorithm predicts the next character from the revealed part of the message.
- If algorithm wrong - as for next guess.
- **Example**

t	h	e	r	e		a	r	e		n	o		p	e
6	2	1	2	1	1	5	2	1	1	4	1	1	5	3

- Code = sequence of number of mistakes.
- To decode use the same prediction algorithm

Arithmetic Coding (background)

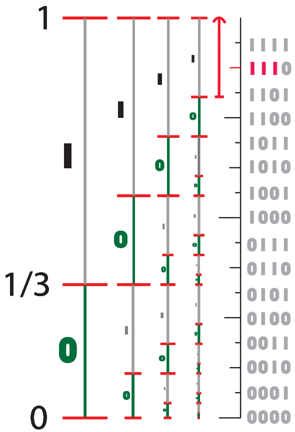
- Refines the guessing game:
  - In guessing game the predictor chooses order over alphabet.
  - In arithmetic coding the predictor chooses a Distribution over alphabet.
- First discovered by Elias (MIT).
- Invented independently by Rissanen and Pasco in 1976.
- Widely used in practice.

Arithmetic Coding (basic idea)

- Easier notation: represent characters by numbers $1 \leq c_t \leq |\Sigma|$. (English: $|\Sigma| = 26$)
- message-prefix $c_1, c_2, \ldots, c_{t-1}$ represented by line segment $[l_{t-1}, u_{t-1})$
- Initial segment $[l_0, u_0) = [0, 1)$
- After observing $c_1, c_2, \ldots, c_{t-1}$, predictor outputs $p(c_t = 1 \,|c_1, c_2, \ldots, c_{t-1}), \ldots, p(c_t = |\Sigma| \;|c_1, c_2, \ldots, c_{t-1})$,
- Distribution is used to partition $[l_{t-1}, u_{t-1})$ into $|\Sigma|$ sub-segments.
- next character c_t determines $[l_t, u_t)$
- Code = discriminating binary expansion of a point in $[l_t, u_t)$.

Arithmetic Coding (Example)

- Simplest case.
- $\Sigma = \{0, 1\}$
- $\forall t$,
  $p(c_t = 0) = 1/3$
  $p_t(c_t = 1) = 2/3$
- Message = 1111
- Code = 111
- (Technical: Assume decoder knows message length)



The code length for arithmetic coding

- Given m bits of binary expansion we assume the rest are all zero.
- Distance between two m bit expansions is 2^{-m}
- If $l_T - u_T \geq 2^{-m}$ then there must be a point x described by m expansion bits such that $l_T \leq x < u_T$
- Required number of bits is $\lceil -\log_2(u_T - l_T) \rceil$.
- $u_T - l_T = \prod_{t=1}^{T} p\,(c_t\,|c_1, c_2, \ldots, c_{t-1}\,) \doteq p(c_1, \ldots c_T)$
- Number of bits required to code $c_1, c_2, \ldots, c_T$ is $\lceil -\sum_{t=1}^{T} \log_2 p_t(c_t) \rceil$.
- We call $-\sum_{t=1}^{T} \log_2 p_t(c_t) = -log_2 p(c_1, \ldots c_T)$ the Cumulative log loss
- Holds for all sequences.

Expectation of code length

- Fix the messsage length T
- Suppose the message is generated at random according to the distribution $p(c_1, \ldots c_T)$
- Then the expected code length is

$$
\begin{aligned}
& \sum_{c_1,\ldots c_T} p(c_1, \ldots c_T) \lceil -\log_2 p(c_1, \ldots c_T) \rceil \\
\leq \quad & 1 - \sum_{c_1,\ldots c_T} p(c_1, \ldots c_T) \log_2 p(c_1, \ldots c_T) \\
\doteq \quad & 1 + H(p_T)
\end{aligned}
$$

- $H(p)$ is the entropy of the distribution p.

Shannon's lower bound

- Assume p_T is "well behaved". For example, IID.
- Let $T \to \infty$
- $H(p) \doteq \lim_{T\to\infty} \frac{H(p_T)}{T}$ exists and is called the per character entropy of the source p
- The expected code length for any coding scheme is at least
$$(1 - o(1))H(p_T) = (1 - o(1)) \; T \; H(p)$$
- The proof of Shannon's lower bound is not trivial!

log loss encourages unbiased prediction

- Suppose the source is random and the probability of the next outcome is $p(c_t | c_1, c_2, \ldots, c_{t-1})$
- Then the prediction that minimizes the log loss is $p(c_t | c_1, c_2, \ldots, c_{t-1})$.
- Note that when minimizing expected number of mistakes, the best prediction in this situation is to put all of the probability on the most likely outcome.
- There are other losses with this property, for example, square loss.

Monthly bonuses for a weather forecaster

- Before the first of the month assign one dollar to the forecaster's bonus. $b_0 = 1$
- Forecaster assigns probability p_t to rain on day t.
- If it rains on day t then $b_t = 2b_{t-1}p_t$
- If it does not rain on day t then $b_t = 2b_{t-1}(1 - p_t)$
- At the end of the month, give forecaster b_T
- Risk averse strategy: Setting $p_t = 1/2$ for all days, guarantees $b_T = 1$
- High risk prediction: Setting $p_t \in \{0, 1\}$ results in Bonus $b_T = 2^T$ if always correct, zero otherwise.
- If forecaster predicts with the true probabilities then

$$E(\log b_T) = T - H(p_T)$$

and that is the maximal expected value for $E(\log b_T)$

“Universal” coding

- Suppose there are N alternative prediction algorithms.
- We would like to code almost as well as the best one.

Two part codes

- Send the index of the coding algorithm before the message.
- Requires $\log_2 N$ additional bits.
- Requires the encoder to make **two** passes over the data.
- Is the key idea of MDL (Minimal Description Length) modeling.
  - Good prediction model = model that minimizes the total code length
- Often inappropriate because based on lossless coding. **Lossy** coding often more appropriate.

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 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

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

- 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:

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

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

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

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

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

EQUALITY not bound!

Simple Bound

- Use uniform initial weights $w_i^1 = 1/N$
- 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*

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

Tuning η as a function of T

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

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

- 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
- 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.
- 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.

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. 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.