Outline

- Introduction
- Bayesian Probability Theory
- Sequence Prediction and Data Compression
  - The Context Tree Weighting algorithm
- Bayesian Networks

Sequence Prediction

Given we have seen

1001110100

so far, what is likely to be the next bit?

Converting to a Learning Problem

Given we have seen

1001110100

so far, what is likely to be the next bit?

Construct examples $(x, y) \in \{0,1\}^* \times \{0,1\}$:

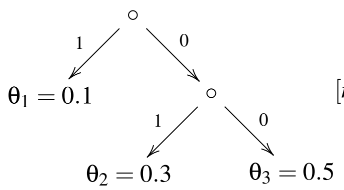
$$[(\varepsilon, 1), (1, 0), (10, 0), (100, 1), (1001, 1), \ldots, (100111010, 0)].$$

From that, learn a function from $\{0,1\}^* \to \mathit{Density}\ \{0,1\}$ to predict the next bit after 1001110100.

To proceed, we need a model class.

Model Class - Prediction Suffix Trees

Suppose we know the underlying source generating the data is a prediction suffix tree (PST).



$[model, parameters]$

E.g. after 10010, the next generated symbol is 1 with probability $PST(10010) = \theta_2$.
After 100100, the next symbol is 1 with probability θ_3.

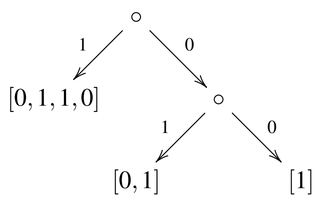
A Simpler Problem: Parameter Learning

- The overall problem is to identify the underlying prediction suffix tree generating the binary sequence. (Hard)
- Suppose we are given the model of the prediction suffix tree, but not its parameters.
- What do we do?

A Simpler Problem: Parameter Learning

- Idea: Just push all the examples down the model and then use the counts to estimate the Bernoulli distributions.

Suppose sequence seen so far is 10011101, then data is
$[(1,0),(10,0),(100,1),(1001,1),(10011,1),(100111,0),(1001110,1)]$.



The Krichevsky-Trofimov Estimator

- Let $x_{1:t} = x_1 x_2 \ldots x_t$ be the input string and let $x_{1:t|l}$ be the (non-contiguous) subsequence of $x_{1:t}$ with a zeros and b ones that end up at a leaf node l. How do we estimate the Bernoulli distribution at l?

The Krichevsky-Trofimov Estimator

- Let $x_{1:t} = x_1 x_2 \ldots x_t$ be the input string and let $x_{1:t|l}$ be the (non-contiguous) subsequence of $x_{1:t}$ with a zeros and b ones that end up at a leaf node l. How do we estimate the Bernoulli distribution at l?
- A solution is

$$\Pr(X = 1 \,|\, x_{1:t|l}) = \frac{b}{a+b}.$$

The Krichevsky-Trofimov Estimator

- Let $x_{1:t} = x_1 x_2 \dots x_t$ be the input string and let $x_{1:t|l}$ be the (non-contiguous) subsequence of $x_{1:t}$ with a zeros and b ones that end up at a leaf node l. How do we estimate the Bernoulli distribution at l?
- A solution is
$$\Pr(X = 1 \,|\, x_{1:t|l}) = \frac{b}{a+b}.$$
- Problematic when a and b are small.
- A way to correct for that is to use
$$\Pr_{kt}(X = 1 \,|\, x_{1:t|l}) = \frac{b+1/2}{a+b+1}.$$
- The KT-estimator converges to the true Bernoulli distribution quickly.

Sequence Prediction

- Now let's make the whole process incremental:

  Loop {
      Predict next bit using current prediction suffix tree;
      Observe next bit and update prediction suffix tree;
  }

Unknown Model

- Let $\mathcal{M}_d$ denote the set of all models of prediction suffix trees of depth at most d.
- Suppose we know the underlying source is in $\mathcal{M}_d$, but we don't know which one.
- What do we do?

Unknown Model

Bayesian optimal predictor:

$$\Pr(x_t \mid x_{1:t-1}) = \frac{1}{\Pr(x_{1:t-1})} \Pr(x_{1:t})$$

Unknown Model

Bayesian optimal predictor:

$$\begin{aligned}\Pr(x_t \,|\, x_{1:t-1}) &= \frac{1}{\Pr(x_{1:t-1})} \Pr(x_{1:t}) \\ &= K \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:t} \,|\, M)\end{aligned}$$

Unknown Model

Bayesian optimal predictor:

$$\begin{aligned}
\Pr(x_t \,|\, x_{1:t-1}) &= \frac{1}{\Pr(x_{1:t-1})} \Pr(x_{1:t}) \\
&= K \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:t} \,|\, M) \\
&= K \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:t-1} \,|\, M) \Pr(x_t \,|\, x_{1:t-1}, M)
\end{aligned}$$

$\Pr(M)$ is the prior on M;
$\Pr(x_{1:t-1} \,|\, M)$ is the evidence for M given past data $x_{1:t-1}$;
$\Pr(x_t \,|\, x_{1:t-1}, M)$ is the prediction made by M after its parameters are estimated using $x_{1:t-1}$.

Unknown Model

Sequence of Bayesian optimal predictors:

$$\Pr(x_1) = \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_1 \,|\, M)$$

$$\Pr(x_2 \,|\, x_1) = \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_1 \,|\, M) \Pr(x_2 \,|\, x_1, M)$$

$$\Pr(x_3 \,|\, x_{1:2}) = \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:2} \,|\, M) \Pr(x_3 \,|\, x_{1:2}, M)$$

$$\vdots$$

$$\Pr(x_t \,|\, x_{1:t-1}) = \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:t-1} \,|\, M) \Pr(x_t \,|\, x_{1:t-1}, M)$$

The Context Tree Weighting Algorithm

- Can we compute Bayesian optimal predictors efficiently in this case?

$$\Pr(x_t \,|\, x_{1:t-1}) = \sum_{M \in \mathcal{M}_d} \Pr(M) \Pr(x_{1:t-1} \,|\, M) \Pr(x_t \,|\, x_{1:t-1}, M)$$

- The summation over $\mathcal{M}_d$ looks ominous at first sight...
- It turns out that the structure in the class $\mathcal{M}_d$ can be exploited to produce an efficient algorithm.

A Prior for Prediction Suffix Trees

- Need a prior $\Pr(M)$ for $M \in \mathcal{M}_d$.
- Occam's razor: smaller trees are more probable.
- How do we proceed?

A Prior for Prediction Suffix Trees

- Need a prior $\Pr(M)$ for $M \in \mathcal{M}_d$.
- Occam's razor: smaller trees are more probable.
- How do we proceed?
- Idea: Find an encoding scheme for trees and then use the length of the encoding as an estimate of a tree's probability.

A Coding Scheme for Trees

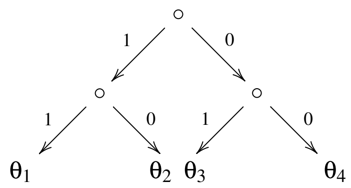
- Walk the tree in preorder. Everytime we see an internal node, we write down 1. Everytime we see a leaf node, we write a 0 if the depth of the leaf node is less than d; otherwise we write nothing.
- The cost $\Gamma_d(t)$ of a tree $t \in \mathcal{M}_d$ is the length of its code.
- Define $\Pr(M) = 2^{-\Gamma_d(M)}$.
- One can show that

$$\sum_{M \in \mathcal{M}_d} 2^{-\Gamma_d(M)} = 1.$$

Context Tree

We define the depth-d context tree as the fully balanced depth-d prediction suffix tree.

E.g. $d = 2$



Fact: Every member in $\mathcal{M}_d$ is a submodel of the depth-d context tree.

CTW Algorithm

Suppose $x_{1:t}$ is the sequence seen so far.

Let $x_{1:t|n}$ be the (non-continguous) subsequence of $x_{1:t}$ that pass through or end up in node n of the context tree.

Define $\Pr_{kt}(y_{1:k}) = \Pr_{kt}(y_1)\Pr_{kt}(y_2|y_1)\Pr_{kt}(y_3|y_{1:2})\cdots\Pr_{kt}(y_k|y_{1:k-1})$.

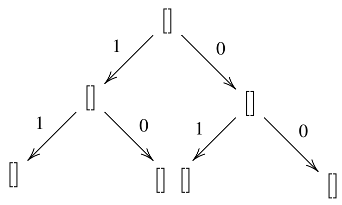
The weighted probability P_w^n of each node n in the context tree is defined inductively as follows:

$$P_w^n(x_{1:t|n}) = \begin{cases} \Pr_{kt}(x_{1:t|n}) & \text{if } n \text{ is a leaf node} \\ \frac{1}{2}\Pr_{kt}(x_{1:t|n}) + \frac{1}{2}P_w^{n_l}(x_{1:t|n_l})P_w^{n_r}(x_{1:t|n_r}) & \text{otherwise,} \end{cases}$$

where n_l and n_r are the left and right childs of n.

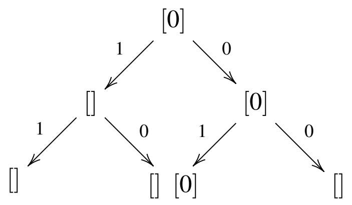
CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is $[(10,0),(100,1),(1001,1),(10011,1),(100111,0),(1001110,1)]$.



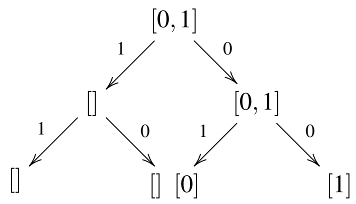
CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is
$[\underline{(10,0)},(100,1),(1001,1),(10011,1),(100111,0),(1001110,1)]$.



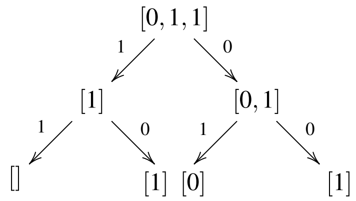
CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is
$[\underline{(10,0),(100,1)},(1001,1),(10011,1),(100111,0),(1001110,1)]$.



CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is
$[\underline{(10,0),(100,1),(1001,1)},(10011,1),(100111,0),(1001110,1)]$.



CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is
$[\underline{(10,0),(100,1),(1001,1),(10011,1),(100111,0),(1001110,1)}]$.

$$
\begin{tikzcd}[column sep=small]
 & & & {[0,1,1,1,0,1]} \arrow[dll, "1"'] \arrow[drr, "0"] & & & \\
 & {[1,1,0]} \arrow[dl, "1"'] \arrow[dr, "0"] & & & & {[0,1,1]} \arrow[dl, "1"'] \arrow[dr, "0"] & \\
{[1,0]} & & {[1]} & & {[0,1]} & & {[1]}
\end{tikzcd}
$$

CTW Algorithm Illustration

Suppose sequence seen so far is 10011101, then data is
$[(10,0),(100,1),(1001,1),(10011,1),(100111,0),(1001110,1)]$.

$$
\begin{tikzcd}[column sep=small]
 & & & \mathrm{Pr}_{kt}(011101) \arrow[dll, "1"'] \arrow[drr, "0"] & & & \\
 & \mathrm{Pr}_{kt}(110) \arrow[dl, "1"'] \arrow[dr, "0"] & & & & \mathrm{Pr}_{kt}(011) \arrow[dl, "1"'] \arrow[dr, "0"] & \\
\mathrm{Pr}_{kt}(10) & & \mathrm{Pr}_{kt}(1) & & \mathrm{Pr}_{kt}(01) & & \mathrm{Pr}_{kt}(1)
\end{tikzcd}
$$

CTW Algorithm Illustration

$$
\begin{tikzcd}[column sep=small]
& & & \mathrm{Pr}_{kt}(011101) \arrow[dll, "1"'] \arrow[drr, "0"] & & & \\
& \mathrm{Pr}_{kt}(110) \arrow[dl, "1"'] \arrow[dr, "0"] & & & & \mathrm{Pr}_{kt}(011) \arrow[dl, "1"'] \arrow[dr, "0"] & \\
\mathrm{Pr}_{kt}(10) & & \mathrm{Pr}_{kt}(1) & & \mathrm{Pr}_{kt}(01) & & \mathrm{Pr}_{kt}(1)
\end{tikzcd}
$$

$$P_w^l(110) = \frac{1}{2}\mathrm{Pr}_{kt}(110) + \frac{1}{2}\mathrm{Pr}_{kt}(10)\mathrm{Pr}_{kt}(1)$$

$$P_w^r(011) = \frac{1}{2}\mathrm{Pr}_{kt}(011) + \frac{1}{2}\mathrm{Pr}_{kt}(01)\mathrm{Pr}_{kt}(1)$$

$$P_w^\lambda(011101) = \frac{1}{2}\mathrm{Pr}_{kt}(011101) + \frac{1}{2}P_w^l(110)P_w^r(011)$$

Analysis of the CTW Algorithm

Theorem: For each node n in the context tree at depth d, we have

$$P_w^n(x_{1:t|n}) = \sum_{M \in \mathcal{M}_d} 2^{-T_d(M)} \Pr(x_{1:t|n}|M).$$

Corollary: The weighted probability $P_w(x_{1:t})$ computed at the root node of the context tree is Bayesian optimal under the prior $\Pr(M) = 2^{-T(M)}$.

Proof Sketch

- Key fact: $\Pr(x_{1:t}|M) = \prod_{l \in L(M)} \Pr_{kt}(x_{1:t|l}|M)$.
- Proof by induction

Proof Sketch

$$
\begin{aligned}
P_w^{\lambda}(011101) &= \frac{1}{2}\mathrm{Pr}_{kt}(011101) + \frac{1}{2}P_w^{l}(110)P_w^{r}(011) \\
&= \frac{1}{2}\mathrm{Pr}_{kt}(011101) + \frac{1}{2}\left(\frac{1}{2}\mathrm{Pr}_{kt}(110) + \frac{1}{2}\mathrm{Pr}_{kt}(10)\mathrm{Pr}_{kt}(1)\right) \\
&\qquad \left(\frac{1}{2}\mathrm{Pr}_{kt}(011) + \frac{1}{2}\mathrm{Pr}_{kt}(01)\mathrm{Pr}_{kt}(1)\right) \\
&= \frac{1}{2}\mathrm{Pr}_{kt}(011101) + \frac{1}{8}\mathrm{Pr}_{kt}(110)\mathrm{Pr}_{kt}(011) \\
&\quad + \frac{1}{8}\mathrm{Pr}_{kt}(110)\mathrm{Pr}_{kt}(01)\mathrm{Pr}_{kt}(1) \\
&\quad + \frac{1}{8}\mathrm{Pr}_{kt}(10)\mathrm{Pr}_{kt}(1)\mathrm{Pr}_{kt}(011) \\
&\quad + \frac{1}{8}\mathrm{Pr}_{kt}(10)\mathrm{Pr}_{kt}(1)\mathrm{Pr}_{kt}(01)\mathrm{Pr}_{kt}(1)
\end{aligned}
$$

CTW Algorithm: Prediction

Given $x_{1:t}$, how do we predict x_{t+1}.

The CTW algorithm gives us only block probabilities $\Pr(x_{1:j})$.

Since

$$\Pr(x_{t+1}|x_{1:t}) = \frac{\Pr(x_{1:t+1})}{\Pr(x_{1:t})},$$

we simply calculate $\Pr(x_{1:t}0)$ and $\Pr(x_{1:t}1)$ and predict the value with the higher probability.

Relationship with Compression

- Claim: If we can predict well, then we can compress well.
- Suppose we know the characters M, P, N, E occur with 60%, 20%, 10%, 10% probability, respectively, in a message of size n.
- Using a naive encoding of 2 bits per symbol requires 2n bits.
- From information theory, we know if we can encode
  - M with $-\log_2 0.6 = 0.74$ bits
  - P with $-\log_2 0.2 = 2.32$ bits
  - N and E with $-\log_2 0.1 = 3.32$ bits each

  then, on average, we need only $\sum_c -\Pr(c) \log_2 \Pr(c) = 1.57$ bits/symbol. Further, this is theoretically the best rate achievable.

Relationship with Compression

- A compression method called arithmetic coding takes as input an estimate $\hat{\Pr}(c|h)$ of $\Pr(c|h)$, where h is the message seen so far and c the next symbol, and produces an encoding that is very close to the optimal rate determined by $\hat{\Pr}(c|h)$.
- CTW can be used to construct $\hat{\Pr}(c|h)$ that gets very close to $\Pr(c|h)$.
- Therefore good predictive power implies good compression.
- Conversely, good compression usually implies good predictive power (in the sense that a Bayesian mixture is almost always dominated by the simple models in the class).

Session Review

- Bayesian optimal estimators are hard to compute in general, but there are exceptions.
- In the context of sequence prediction,
  - good predictive power implies good compression
  - good compression (usually) implies good predictive power