Jeffreys' prior. An application: Context tree weighting

Contents

0 Context

0.1 Bayesian approximation for universal probability distributions

We recall that we are interested in the following probability distribution:

$$P_4(x) = \sum_{\text{probability distributions } \mu} 2^{-|\mu|}\mu(x), \tag{1}$$

where $|\mu|$ is the length of the shortest program computing μ (with the convention that $|\mu| = \infty$ if μ is not computable).

Since P_4 itself is not computable, it has to be approximated, for example by replacing the set of all computable probability distributions by a parametric family

$$\mathcal{F} = \{\mu_\theta, \theta \in \Theta\}, \tag{2}$$

and $|\mu|$ is replaced by $K(\theta)$. The distribution we are interested in is therefore:

$$P_{\mathcal{F}} = \sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta. \tag{3}$$

We mentioned four possibilities to approximate $P_{\mathcal{F}}$. Among them:

- Two-part codes: encode some θ first, then use P_θ to encode the data.
- Replace $2^{-K(\theta)}$ in $P_{\mathcal{F}}$ by a prior q on Θ, which yields

$$P_{\mathcal{F}}^{\text{Bayes}}(x) = \int_\Theta q(\theta) \mu_\theta(x) \mathrm{d}\theta. \tag{4}$$

We are going to show that there exists a canonical prior on Θ (Jeffreys' prior), but firstly, we recall the optimal precision for the encoding of θ in the case of two-part codes.

0.2 Reminder from Talk 2: Optimal precision in the i.i.d. case

Consider we are trying to encode a message $(x_1, ..., x_n)$ with $(x_i)_{i \in [1,n]}$ i.i.d. using a two-part code. We need to encode some $\theta^* \in \Theta$.

Suppose for example that $\Theta = [0, 1]$. In general, it is not possible to encode the full binary expansion of θ^*, since we would need an infinite number of bits. Consequently, we will encode some θ close to θ^*, by simply writing the first k bits. In that case, we can guarantee that

$$|\theta - \theta^*| \leqslant 2^{-k} =: \varepsilon. \tag{5}$$

We have seen in Talk 2 that the optimal[1] epsilon ε^* was given by:

$$\varepsilon^* \approx \frac{1}{\sqrt{n}} \frac{1}{\sqrt{I(\theta^*)}}, \tag{6}$$

where $I(\theta^*)$ is the Fisher information matrix at θ^*.[2]

Now, let us construct Jeffreys' prior.

1 Jeffreys' prior

1.1 Motivation

We recall that we needed a reasonable default prior q for the probability distribution (4):

$$P_{\mathcal{F}}^{\text{Bayes}}(x) = \int_\Theta q(\theta) \mu_\theta(x) \mathrm{d}\theta. \tag{7}$$

A naive choice for a default prior is "the uniform prior" (i.e. $q(\theta) = \text{cst}$).

[1] In terms of expected codelength.

[2] In dimension larger than 1, $I(\theta^*)$ is replaced by $\det I(\theta^*)$.

Sadly, *the* uniform prior on a family of probability distributions can be ill-defined.

Consider for example $\mathcal{B}$ the family of Bernoulli distributions.

If P_θ is the Bernoulli distribtion of parameter θ, and Q_θ is the Bernoulli distribtion of parameter θ^{100}, then $\{P_\theta, \theta \in [0,1]\} = \{Q_\theta, \theta \in [0,1]\} = \mathcal{B}$, but most of the time, the uniform prior on the family (Q_θ) will select a Bernoulli distribution with a parameter close to 0 (θ is picked uniformly in $[0,1]$, so $\theta^{100} < 0.1$ with probability $0.1^{\frac{1}{100}} \gtrsim .97$).

This shows that a uniform prior depends not only on our family of probability distributions, *but also on its parametrization*, which is an arbitrary choice of the user.

Any reasonable default prior should be invariant by reparametrization of the family $\{P_\theta, \theta \in \Theta\}$ (else, there would be as many possible priors as there are parametrizations), and Jeffreys' prior, constructed below, does have this property.

1.2 Construction

Consider we are sending a message of length n.

Equations (5) and (6) (about the optimal coding precision) indicate that when coding a given message with a two-part code, only a finite number of elements of Θ will actually be used in the first part of the coding, each θ_k being used for all $\theta \in I_k$; and equation (6) shows that the length of I_k is $\frac{1}{\sqrt{n}} \frac{1}{\sqrt{I(\theta_k)}}$.

The fact that all elements of I_k are encoded the same way means that we will not distinguish different θ in the same I_k, and in practice, we will only use the m different θ_k corresponding to each interval.

Consequently, a reasonable procedure to pick a θ would be to start by picking some $k \in [1, m]$, and then, pick $\theta \in I_k$ uniformly.

It is easy to see that the probability distribution q_n corresponding to this procedure is given by the density:

$$q_n(\theta) = K_n \sqrt{I(\theta_{k(\theta)})}, \tag{8}$$

where K_n is a normalization constant, and $k(\theta)$ is defined by the relation $\theta \in I_{k(\theta)}$.

Now, if $n \to \infty$, it can be proved[3] that (q_n) converges to the probability distribution q given by the density:

$$q(\theta) = K \sqrt{I(\theta)}, \tag{9}$$

where K is a normalization constant.[4] Also notice that sometimes (for example with the family of Gaussian distributions on $\mathbb{R}$), Jeffreys' prior cannot be normalized, and consequently, cannot be used.

We now have a reasonable default prior, and we are going to use it to predict the next element of a sequence of zeros and ones.

[3] The important idea is that $\theta_{k(\theta)} \to \theta$ when $n \to \infty$, since the length of all the intervals I_k tends to 0.

[4] As in footnote 2, in dimension larger than 1, (9) becomes $q(\theta) = K\sqrt{\det I(\theta)}$.

1.3 Example: the Krichevsky–Trofimov estimator

We introduce the following notation:

Notation 1. *We denote by P_θ the Bernoulli distribution of parameter θ, and we define $\mathcal{B} := \{P_\theta, \theta \in [0,1]\}$.*

Suppose we are trying to learn a frenquency on the alphabet $X = \{c, d\}$. For example, given the message *ccc*, we want $P(x_4 = c)$.

A first approach is using the maximum likelihood (ML) estimator:

$$P^{\mathrm{ML}}(x_{n+1} = c | x_1, ..., x_n) = \frac{\sum_{i=1}^n \mathbb{1}_{x_i = c}}{n} = \frac{\text{number of c in the past}}{\text{number of observations}}. \tag{10}$$

This has two drawbacks: ML assigns the probability 0 to any letter that has not been seen yet (which is a major problem when designing codes, since probability 0 corresponds to infinite codelength), and is undefined for the first letter.

Let us now consider the Bayesian approach with Jeffreys' prior.

Lemma 2. *The Fisher metric on $\mathcal{B}$ is given by*

$$I(\theta) = \frac{1}{\theta(1-\theta)}, \tag{11}$$

and Jeffreys' prior is given by

$$q(\theta) = \frac{1}{\pi} \frac{1}{\sqrt{\theta(1-\theta)}} \tag{12}$$

Proof. It is a straightforward calculation. We have $P_\theta(0) = 1 - \theta$, $P_\theta(1) = \theta$, so $\frac{\partial^2 \ln P_\theta(0)}{\partial \theta^2} = \frac{-1}{(1-\theta)^2}$ and $\frac{\partial^2 \ln P_\theta(1)}{\partial \theta^2} = \frac{-1}{\theta^2}$. Finally:

$$\begin{aligned} I(\theta) &= -\frac{\partial^2 \ln P_\theta(0)}{\partial \theta^2} P_\theta(0) - \frac{\partial^2 \ln P_\theta(1)}{\partial \theta^2} P_\theta(1) && (13)\\ &= \frac{1}{(1-\theta)^2}(1-\theta) + \frac{1}{\theta^2}\theta && (14)\\ &= \frac{1}{1-\theta} + \frac{1}{\theta} && (15)\\ &= \frac{1}{\theta(1-\theta)}, && (16) \end{aligned}$$

and we have the Fisher information. The only difficulty to compute Jeffreys' prior is the normalization constant, which is

$$K := \int_0^1 \frac{\mathrm{d}\theta}{\sqrt{\theta(1-\theta)}}, \tag{17}$$

and the substition $\theta = \sin^2 u$ yields

$$K = \int_0^{\pi/2} \frac{2 \sin u \cos u \mathrm{d}u}{\sqrt{\sin^2 u \cos^2 u}} = \pi. \tag{18}$$

□

We have therefore, by using Jeffreys' prior in (4):

$$P^{\text{Jeffreys}}(x_{n+1}|x_1,...,x_n) = \frac{1}{\pi}\int_0^1 \frac{1}{\sqrt{\theta(1-\theta)}} P_\theta(x_1,...,x_n) P_\theta(x_{n+1}|x_1,...,x_n)\mathrm{d}\theta. \tag{19}$$

This is actually easy to compute, thanks to the following proposition, which will not be proved here:

Proposition 3. *Let $\alpha, \beta > 0$. If $q(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$, then*

$$P^{\text{Bayes}}(x_{n+1} = c|x_1,...,x_n) = \frac{\alpha + \sum_{i=1}^n \mathbb{1}_{x_i=c}}{\alpha+\beta+n}, \tag{20}$$

where

$$P^{\text{Bayes}}(x) = \int_\Theta q(\theta)\mathcal{B}_\theta(x)\mathrm{d}\theta \tag{21}$$

is the Bayesian prediction with Bernoulli distributions and the prior q on the parameter θ of the distributions.

In other words, using such a prior is equivalent to using ML, with fictional observations (c occuring α times and d occuring β times).[5]

In particular, for Jeffreys' prior, we have $\alpha = \beta = \frac{1}{2}$, so

$$P^{\text{Jeffreys}}(x_{n+1} = c|x_1,...,x_n) = \frac{\frac{1}{2} + \sum_{i=1}^n \mathbb{1}_{x_i=c}}{n+1} = \frac{\frac{1}{2} + \text{number of c in the past}}{1 + \text{number of observations}}. \tag{22}$$

Consequently, we introduce the following notation: $\mathrm{KT}(x,y) := \dfrac{\frac{1}{2}+x}{1+x+y}$.

This estimator is called "Krichevsky–Trofimov estimator" (KT), and it can be proved ([7],[5]) that, when using it instead of knowing the "real" θ, no more than $1 + \frac{1}{2}\log(n)$ bits are wasted.

We can also define the generative model corresponding to Jeffreys' prior. It is easy to prove (by induction on the total number of symbols) that the probability assigned to any given sequence $(x_1,...,x_{a+b})$ with a zeros and b ones is equal to

$$P_J(a,b) := \frac{\left(\prod_{0\leqslant i<a}(i+\frac{1}{2})\right)\left(\prod_{0\leqslant i<b}(i+\frac{1}{2})\right)}{\prod_{0\leqslant i<a+b}(i+1)} = \prod_{0\leqslant i<a}\mathrm{KT}(i,0)\prod_{0\leqslant j<b}\mathrm{KT}(a,j), \tag{23}$$

with the convention that the empty product is equal to 1 (i.e. $P_J(0,0) = 1$).

In particular, we can see that the probability of a sequence $(x_1,...,x_n)$ depends only on the *number* of c and d in $(x_1,...,x_n)$, and not on the order, and P_J satisfies the relations, $P_J(a+1,b) = P_J(a,b)\mathrm{KT}(a,b)$ and $P_J(a,b+1) = P_J(a,b)(1-\mathrm{KT}(a,b))$.

It is also useful to remark that $P_J(0,0) = 1$ and $P_J(1,0) = P_J(0,1) = \frac{1}{2}$.

In practice, Jeffreys' prior is not efficient when the optimal θ is on the boundary: for instance, when encoding text, most punctuation symbols are always

[5]Distributions satisfying $q(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$ are called *beta* distributions. The simplification in this proposition is due to the fact that if we have a beta prior for a Bernoulli distribution, then the posterior will also be a beta distribution. The beta distributions are called *conjugate priors* to Bernoulli distributions .

followed by a space. The corresponding new prior for Bernoulli distributions is given by

$$\frac{1}{2}q + \frac{1}{4}\delta_0 + \frac{1}{4}\delta_1, \tag{24}$$

where q is Jeffreys' prior on $\mathcal{B}$, and δ_i is the Dirac distribution at i. This new prior is called a "zero-redundancy estimator".

We are now going to apply all we have done until now with a text prediction algorithm.

2 Context tree weighting

In this section, we quickly present context tree weighting, a Bayesian text prediction model combining *all* visible Markov models of any finite order, see [7] and [6] for more information.

We start by showing a simple way to describe *one* visible Markov model.

2.1 Markov models and full binary trees

Learning a frequency on the alphabet $X = \{c, d\}$ corresponds to restricting ourselves to the family of Bernoulli distributions, which can be thought of as the family of Markov models of order 0.

Still working on the alphabet $\{c, d\}$, we are going to present the context tree weighting algorithm, which uses the family of all visible Markov models.

Notice in particular that we can use Markov models of *variable* order, for example: "if the last symbol was c, predict c with probability 0.7, if the last symbol was d, then look at the next-to-last symbol: if it was a c then predict c with probability 0.5, else predict c with probability 0.3."

A Markov model can be described by a full[6] binary tree the following way:

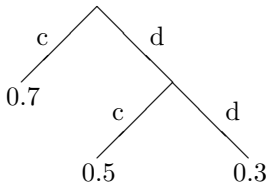

Figure 1: Tree corresponding to the example above

For a given tree, the contexts we are interested in are the leaves of the tree: with the example above, the leaf labeled 0.7 corresponds to the context "c?", the leaf 0.5 corresponds to the context "cd?" and the leaf 0.3 corresponds to "dd?" (notice in particular that contexts are read *backwards* in the tree).

Suppose for now that we have such a tree. It is easy to see that there is exactly one context appearing in the tree and corresponding to a suffix of the past observations: we output c with probability equal to the value of the leaf corresponding to this context.

[6] All nodes have either 0 or 2 children.

Our model can therefore be decomposed as a finite full binary tree T (the structure of the tree), and the ordered set $\theta_T := (\theta_s)$ of the labels on the leaves, and we will try to estimate it online.

Now that we are able to describe efficiently a Markov model, we can go back to our main problem: prediction using universal probability distributions on all visible Markov models.

2.2 Prediction for the family of visible Markov models

The distribution we would like to use for prediction is (3), with $\Theta = \{(T, \theta_T), T \text{ finite binary tree}, \theta_T \in [0,1]^{\text{number of leaves of } T}\}$:

$$P_{\mathcal{F}} = \sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta,$$

where μ_θ is given by the previous subsection. $K(\theta)$ remains problematic, but we can decompose $\sum_\theta 2^{-K(\theta)}$ into $\sum_T 2^{-K(T)} \sum_{\theta_T} 2^{-K(\theta_T)}$, and then:

- A full binary tree T can be described by as many bits as it has nodes, by labelling internal nodes by 1 and leaves by 0, and read breadth-first (our example would give the code 10100).
- We can use Jeffreys' prior for θ_T.

With these new approximations, we have:

$$P_{\mathcal{F}} = \sum_{T \text{ full binary trees}} 2^{-\#\text{nodes}(T)} \int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} \cdots \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}, \tag{25}$$

where θ_i denotes the parameter of the i-th leaf. (25) can be interpreted as a double mixture over the full binary trees and over the parameter values.

Now, we have to compute (25). This sum has an infinite number of terms (and even if we restrict ourselves to a finite depth D, the number is still exponential in D). However, it is possible to compute it efficiently and *exactly*.

2.3 Computing the prediction

The general idea is to maintain the tree of all observed contexts, and each node s will weight the choices "using s as a context", and "splitting s into the subonctexts cs and ds" by using information from deeper nodes (more precisely, the number of times a c or a d has been written in a given context).

2.3.1 Bounded depth

Here, we restrict ourselves to Markov models of depths at most D. For simplicity, we will not try to predict the D first symbols.

Notice that in that case, the cost to describe a tree can be modified: since a node at depth D is automatically a leaf, it is no longer necessary to use bits to describe them. If we denote by T_D the complete binary tree of depth D,

and by $n_D(T)$ the number of nodes at depths less than D of a tree T, then the probability distribution corresponding to this model is

$$P_{\mathcal{F}} = \sum_{T \text{ subtree of } T_D} 2^{-n_D(T)} \int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} \cdots \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}, \tag{26}$$

Notation 4. *We will denote each node by the suffix corresponding to it (for example, the root is ε, the node labeled 0.7 in Figure 1 is c, etc...), and at each node s, we will maintain two numbers c_s and d_s, respectively the number of times a c and a d have been observed after the context s (starting the count at x_1). Notice that we have the relation $c_s = c_{1s} + c_{0s}$.*[7]

Now, let us look at equation (26) more closely. The goal of the following lemma and its corollary is to write equation (26) in a way that will allow to compute it efficiently.

Firstly, we can see that

Lemma 5. *If $c_s(k)$ and $d_s(k)$ are respectively the number of times a c and a d have been observed after the context s from x_1 to x_k, we have:*

$$P_{(T,(\theta_{s_i}))}(x_1, ..., x_n) = \prod_{s \text{ leaves of T}} \theta_s^{c_s(n)} (1-\theta_s)^{d_s(n)}, \tag{27}$$

See the appendix for the proof.

Consequently, the integral in (26) can be computed with Proposition 3:

Corollary 6. *We have:*

$$\int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} \cdots \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}(x_1, ..., x_n) = \prod_{u \text{ leaves of } T} P_J(c_u, d_u) \tag{28}$$

Proof. It is a consequence of Lemma 5 and of the fact that $P_J(a, b)$ is the probability of seeing a sequence containing respectively a and b times the letters c and d with Jeffreys' prior. □

Equation 26 can therefore be rewritten:

$$P_{\mathcal{F}}(x_1, ..., x_n) = \sum_{T \text{ subtree of } T_D} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u). \tag{29}$$

Under this form, $P_{\mathcal{F}}$ can be computed recursively, starting by its leaves, by the following lemma:

Lemma 7. *If we assign to each node s the following probability:*

$$P^s := \begin{cases} P_J(c_s, d_s) & \text{if} \quad |s| = D \\ \frac{1}{2} P_J(c_s, d_s) + \frac{1}{2} P^{cs} P^{ds} & \text{if} \quad |s| < D \end{cases} \tag{30}$$

[7] Unless s is a prefix of the word we are reading. In practice, as we will see in the next section, this problem can be solved by adding a new character § indicating the beginning of the message: $c_s = c_{0s} + c_{1s} + c_{\S s}$ is *always* true.

then, for any node s, P^s satsifies:

$$P^s = \sum_{T \text{ subtree of } T_D \text{ with root } s} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u). \tag{31}$$

Notice that by definition, s is automatically a suffix of all the u in the equation above.

See the appendix for the proof.

As an immediate corollary, we have:

$$P^\varepsilon = \sum_{T \text{ subtree of } T_D \text{ with root } \varepsilon} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u), \tag{32}$$

which is exactly equation (29).

Let us now consider the general case.

2.3.2 Generalization

The method presented above has two shortcomings: the fact that we cannot predict the first D symbols, and the bounded depth.

The first one can be solved by adding a "beginning of the message" character §. The corresponding trees are then ternary, and since undefined symbols can only occur at the beginning of a word, any suffix starting with § is a leaf.

Now, to work with unbounded depth, we simply maintain the tree T_n of all suffixes seen up to time n (i.e: all factors of the word $\S, x_1, ..., x_{n-1}, x_n$. T_n is therefore exactly of depth $n+1$), and, at each node s of T_n, the counts c_s, d_s, and $\S_s$.

To compute the corresponding distribution, we replace the P^s in (30) by:

$$P^s := \begin{cases} \frac{1}{2} & \text{if} \quad s \text{ is a leaf of } T_n \\ \frac{1}{2} P_J(c_s, d_s) + \frac{1}{2} P^{cs} P^{ds} P^{\S s} & \text{else,} \end{cases} \tag{33}$$

with the convention that $P^{\lambda s} = 1$ if λs has never appeared yet (for $\lambda = c, d, \S$), and we use P^ε again as our probability distribution.

It can be proved that this algorithm achieves entropy (i.e. optimal coding length) for arbitrary-depth tree sources (Theorem 3 in [6]).

We can now describe the actual algorithm.

2.4 Algorithm

Suppose that we have $P_{\mathcal{F},n}(x_n | x_1, ..., x_{n-1})$, with the corresponding tree T_{n-1} (the tree containing all suffixes in $x_1, ..., x_{n-1}$).

When we read x_n, we update T_{n-1} by adding to it the suffix corresponding to the branch $(x_1, ..., x_n)$. We add at most n nodes. Now, we compute the new P^s. Since the only modified P^s are those corresponding to a suffix of $(x_1, ..., x_n)$, the complexity is $O(n)$.

We can therefore read $P(x_1, ..., x_n)$ at the root. Now, add the node $x_1, ..., x_n, c$ to compute $P(x_1, ..., x_n, c)$ at the root (the computation is again $O(n)$), and the prediction is $P_{\mathcal{F},n+1}(x_{n+1} | x_1, ..., x_n) = \frac{P_{\mathcal{F},n+1}(x_1,...,x_{n+1})}{P_{\mathcal{F},n}(x_1,...,x_n)}$.

The complexity of this algorithm is $O(n^2)$, whereas the bounded version has complexity $O(nD)$, where, n is the length of the data we are compressing and D is the depth bound. However, if the data is "sufficiently random", the complexity of the unbounded version is only $O(n \log_2 n)$ when coded properly[8], which makes it usable in general.

[8]The algorithm described here is always $O(n^2)$, but it can be improved by using the fact that some points are "equivalent" (for example, we can stop going deeper in the tree once we find a suffix that has appeared only once).

Appendix

Proof of Lemma 5. By induction. $P_{(T,(\theta_{s_i}))}(x_1) = \theta_{s_T(Lx_1)}^{\mathbb{1}_{x_1=c}}(1-\theta_{s_T(Lx_1)})^{\mathbb{1}_{x_1=d}}$, where $s_T(x)$ is the only leaf of T corresponding to a suffix of x, which is what we want.

Now,

$$\begin{aligned}P_{(T,(\theta_{s_i}))}(x_1,...,x_{k+1}) =& P_{(T,(\theta_{s_i}))}(x_{k+1}|x_1,...,x_k)P_{(T,(\theta_{s_i}))}(x_1,...,x_k)\\ =& \theta_{s_T(Lx_1...x_k)}^{\mathbb{1}_{x_{k+1}=c}}(1-\theta_{s_T(Lx_1...x_k)})^{\mathbb{1}_{x_{k+1}=d}}\prod_{s \text{ leaves of T}}\theta_s^{c_s(k)}(1-\theta_s)^{d_s(k)},\end{aligned}$$

and since for all s for $\lambda \in \{c,d\}$, $\lambda_s(k+1) = \lambda_s(k) + \mathbb{1}_{x_{k+1}=\lambda}\mathbb{1}_{s=s_T(Lx_1...x_k)}$, we find

$$P_{(T,(\theta_{s_i}))}(x_1,...,x_{k+1}) = \prod_{s \text{ leaves of T}}\theta_s^{c_s(k+1)}(1-\theta_s)^{d_s(k+1)}, \tag{34}$$

which is what we wanted. □

Proof of Lemma 7. By induction. If $|s| = D$, it is clearly true. Suppose now that (31) holds for all nodes deeper than s. We have, by definition:

$$\begin{aligned}P^s =& \frac{1}{2}P_J(c_s,d_s) + \frac{1}{2}P^{cs}P^{ds}\\ =& \frac{1}{2}P_J(c_s,d_s) + \frac{1}{2}\left(\sum_{T \text{ subtree of } T_D \text{ with root } cs} 2^{-n_D(T)}\prod_{u \text{ leaves of } T}P_J(c_u,d_u)\right)\\ &* \left(\sum_{T \text{ subtree of } T_D \text{ with root } ds} 2^{-n_D(T)}\prod_{u \text{ leaves of } T}P_J(c_u,d_u)\right)\end{aligned}$$

The first term corresponds to the subtree composed only of the root, and the second term corresponds to all other subtrees, described by the part on the left of the root and the part of the right of the root. In other words:

$$P^s = 2^{-1}P_J(c_s,d_s) + \sum_{\substack{T \text{ subtree of } T_D \text{ with root } cs\\ T' \text{ subtree of } T_D \text{ with root } ds}} 2^{-1-n_D(T)-n_D(T')}\prod_{u \text{ leaves of } T \text{ or } T'}P_J(c_u,d_u), \tag{35}$$

which is what we want.

□

References

[1] Peter D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[2] Marcus Hutter. On universal prediction and Bayesian confirmation. *Theoretical Computer Science*, 384(1):33–48, 2007.

[3] Ming Li and Paul M.B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008.

[4] Ray J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7, 1964.

[5] Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context tree switching. *CoRR*, abs/1111.3182, 2011.

[6] Frans M. J. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44:792–798, 1994.

[7] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.