Universal probability distributions, two-part codes, and their optimal precision

Contents

0 An important reminder

Firstly, notice that a probability distribution P on X^n induces a probability distribution P_k on X^k with $k \leqslant n$:

$$P_k(x_1, ..., x_k) = \sum_{y_1,...,y_{n-k}} P(x_1, ..., x_k, y_1, ..., y_{n-k}). \tag{1}$$

We will systematically drop the k in the notation.

Now, if we work in the set X, we recall that a generative model at horizon n (i.e. a probability distributions on sequences of length n, $P(x_1, ..., x_n)$), a predictor (A probability distribution on X given observations, $Q(x_k|x_1, ..., x_{k-1})$) and an encoding scheme (assiging to each x its codelength) are all the same thing:

The predictor Q corresponding to the generative model P is given by

$$Q(x_k|x_1, ..., x_{k-1}) = \frac{P(x_1, ..., x_k)}{P(x_1, ..., x_{k-1})}, \tag{2}$$

and conversely, the generative model P corresponding to the predictor Q is given by

$$P(x_1, ..., x_k) = Q(x_1)Q(x_2|x_1)...Q(x_k|x_1, ..., x_{k-1}). \tag{3}$$

Moreover, since any probability distribution P corresponds to an encoding scheme (encoding x with $-\log_2 P(x)$ bits), and conversely (assign to x the probability $2^{-l(x)}$), we have the equivalence between encoding schemes and generative models.

As an example, if we have several generative models P_i, then $\sum w_i P_i$ will also be a generative model. However, it should be noted that the associated predictor is a mixture with Bayesian weights, which *depend on the data* (in particular, they are *not* the w_i: the w_i are only the *prior* weights). See equation (11) for an illustration.

1 Universal probability distributions in theory

This equivalence between encoding and probability distribution, combined with Occam's razor is probably the main reason for defining Kolmogorov complexity: Kolmogorov complexity gives the length of the "best" encoding (by Occam's razor), so the probability distribution it defines must be the "best" predictor or generative model.

More precisely, we can define the following probability distributions on X^* (All finite sequences of elements of X. These distributions are defined up to a normalization constant for programs that do not end), that should be usable for any problem (see [4]).

$$P_1(x) = 2^{-K(x)}, \tag{4}$$

$$P_2(x) = \sum_{\text{all deterministic programs}} 2^{-|p|} \mathbb{1}_{p \text{ outputs } x}, \tag{5}$$

$$P_3(x) = \sum_{\text{all random programs}} 2^{-|p|} P(p \text{ outputs } x), \tag{6}$$

$$P_4(x) = \sum_{\text{probability distributions}} 2^{-|\mu|} \mu(x), \tag{7}$$

where a program is any string of bits to be read by a universal Turing machine, and a random program is a program that has access to a stream of random bits (in particular, a deterministic program is a random program), and $|\mu|$ is the Kolmogorov complexity of μ, i.e., the length of the shortest program computing μ (if μ is not computable, then its Kolmogorov complexity is infinite, and μ does not contribute to P_4: the sum is in practice restricted to computable probability distributions). For example, P_2 is the output of a program written at random (the bits composing the program are random, but the program is deterministic), and P_3 is the output of a random program written at random.

Notice that $P_1(x)$ can be rewritten as $\sum_{\text{Diracs}} 2^{-|\delta|} \delta(x)$: P_4 is to P_1 what P_3 is to P_2 (the Diracs are the "deterministic probability distributions").

Since Kolmogorov complexity is defined only up to an additive constant, the probability distributions above are only defined up to a multiplicative constant. This leads us to the following definition:

Definition 1. *Let P and P' be two probability distributions on a set X. P and P' are said to be* equivalent *if there exists two constants m and M such that:*

$$mP \leqslant P' \leqslant MP \tag{8}$$

Proposition 2. *P_1, P_2, P_3 and P_4 are equivalent.*

Consequently, we can pick any of these probability distributions (which are called Solomonoff universal prior) as a predictor. We choose P_4:

$$P_4(x_{k+1}|x_k,...x_1) := \frac{\sum_\mu 2^{-|\mu|}\mu(x_1,...,x_{k+1})}{\sum_{\mu,x'} 2^{-|\mu|}\mu(x_1,...,x_k,x')} \tag{9}$$

$$= \frac{\sum_\mu 2^{-|\mu|}\mu(x_1,...,x_k)\mu(x_{k+1}|x_1,...,x_k)}{\sum_\mu 2^{-|\mu|}\mu(x_1,...,x_k)} \tag{10}$$

$$= \frac{\sum_\mu w_\mu \mu(x_{k+1}|x_1,...,x_k)}{\sum_\mu w_\mu}, \tag{11}$$

where $w_\mu = 2^{-|\mu|}\mu(x_1,...,x_k)$ can be seen as a Bayseian posterior on the probability distributions (we had the prior $2^{-|\mu|}$). In particular, *the posterior weights depend on the data.*

However, as we have seen, the Kolmogorov complexity and therefore P_4 are not computable.

2 Universal probability distributions in practice

We have to replace Kolmogorov complexity by something simpler: we choose a family of probability distibutions $\mathcal{F}$, and restrict ourselves to this family. The distribution we obtain is therefore $P_\mathcal{F} := \sum_{\mu \in \mathcal{F}} 2^{-|\mu|}\mu$.

We now give some possible families:

- $\mathcal{F} = \{\mu\}$. This is often the case with real data: people are already happy to have one simple model that explains past data.
- $\mathcal{F} = \{\mu_1, ..., \mu_n\}$: as seen in equation (11), we obtain a Bayesian combination.
- $\mathcal{F} = \{\mu_\theta, \theta \in \Theta\}$: this case will be studied below. As a simple example, we can take $\Theta = [0,1]$, and μ_θ is Bernoulli, with parameter θ.

In that case, we have

$$P_\mathcal{F} = \sum_{\theta \in \Theta} 2^{-K(\theta)}\mu_\theta. \tag{12}$$

Notice that the right side of equation (12) is only a countable sum: if θ is not computable, then $K(\theta) = +\infty$, and the corresponding term does not contribute to the sum.

There exist different techniques for approximating $P_\mathcal{F}$ in the parametric case (see for example [1]):

1. Encoding the best θ_0 to encode the data. In that case, $\sum_{\theta \in \Theta} 2^{-K(\theta)}\mu_\theta$ is approximated by the largest term of the sum. Since we encode first a $\theta \in \Theta$, and then the data, with the probability distribution P_θ, these codes are called *two-part codes.*

2. We can replace the penalization for a complex θ by a continuous Bayesian prior q on Θ. In that case, $\sum_{\theta\in\Theta} 2^{-K(\theta)}\mu_\theta$ is approximated by $\displaystyle\int_\Theta q(\theta)\mu_\theta$. As we will see later, there exists a prior (called Jeffreys' prior) with good properties with respect to this construction.

3. Normalized maximum likelihood techniques (will be discussed later)

4. We can make online prediction in the following way: use a default prior for x_1, and to predict (or encode) x_k, we use past data to choose the best μ_θ.[1] With this method, the parameter θ is defined implicitly in the data: there is no need to use bits to describe it.

We also introduce the following definition (although it will not be used in the remainder of this talk):

Definition 3. *A generative model P is said to be* prequential *(contraction of predictive-sequential, see [1]) if* $\displaystyle\sum_x P(x_1, ..., x_k, x) = P(x_1, ..., x_k)$ *(i.e. different time horizons are compatible).*

A predictive model Q is said to be prequential if the equivalent generative model is prequential, or equivalently, if $Q(x_{k+1}, x_k|x_1, ..., x_{k-1}) = Q(x_{k+1}|x_1, ..., x_k)Q(x_k|x_1, ..., x_{k-1})$ *(i.e. predicting two symbols simultaneously and predicting them one after the other are the same thing, hence the name "predictive-sequential")*

It can be checked that the Bayesian model and the online model are prequential, whereas the two others are not.

Notice that being prequential is equivalent to being compatible with the point of view in Section 0, in the sense that prequential models really correspond to one probability distributions on X^n. The reason why two-part codes and NML codes are not prequential is that for any fixed k, they correspond to a probability distriution P_k on the set of sequences of symbols of length k, but for $k' \neq k$, P_k and $P_{k'}$ are not necessarily related: they do not satisfy equation (1) (and consequently, NML codes and two-part codes themselves do not satisfy equations (2) and (3)).

We now study the first of these techniques: two-part codes.

2.1 Two-part codes

Our strategy is to use the best code θ_0 in our family (P_θ). Since the decoder cannot know which P_θ we are using to encode our data, we need to send θ_0, first.

Since θ_0 is a real parameter, we would almost surely need an infinite number of bits to encode it exactly: we will encode some θ "close" to θ_0 instead.

Suppose for example that $\theta_0 \in [0, 1]$. If we use only its k first binary digits for θ, then we have $|\theta - \theta_0| \leqslant 2^{-k}$.

[1] Here, "the best" does not mean the maximum likelihood estimator: if a symbol does not appear, then it will be predicted with probability 0, and we do not want this. A possible solution is to add fictional points of data before the message, which will also yield the prior on x_0. For example, when encoding the results of a game of heads or tails, it is possible to add before the first point a head, and a tail, each with weight 1/2.

Consequently, let us define the precision of the encoding of some $\theta \in [0, 1]$ with $\epsilon := 2^{-\text{number of bits to encode }\theta}$ (so we immediately have the bound $|\theta - \theta_0| \leqslant \epsilon$).

We recall the bound given in the first talk for any probability distribution μ:

$$K(x) \leqslant K(\mu) - \log_2(\mu(x)), \tag{13}$$

which corresponds to coding μ, and then, using an optimal code with respect to μ to encode the data.

Here, increasing the precision (or equivalently, reducing ϵ) increases the likelihood of the data (and consequently $-\log_2(\mu(x))$ decreases), but $K(\mu)$ increases: we can suppose that there exists some optimal precision ϵ^*. Let us compute it.

2.1.1 Optimal precision

In the ideal case (infinite precision), we encode the data $x_1, ..., x_n$ by sending $\theta^* := \text{argmax}_\theta \mu_\theta(x_1, ..., x_k)$, and then, the data encoded with the probability distribution μ_{θ^*}.

The codelength corresponding to a given ϵ is:

$$l(\epsilon) := -\log_2 \epsilon - \log_2(\mu_\theta(x)), \tag{14}$$

where $|\theta - \theta^*| \leqslant \epsilon$ (i.e. we encode θ, which takes $-\log_2 \epsilon$ bits, and then, we encode x using μ_θ, which takes $-\log_2(\mu_\theta(x))$ bits).

With a second order Taylor expansion around θ^*, we find:

$$l(\epsilon) = -\log_2 \epsilon - \log_2 \mu_{\theta^*}(x) + \frac{\partial(-\log_2 \mu_\theta(x))}{\partial\theta}(\theta - \theta^*) + \frac{\partial^2}{\partial\theta^2}(-\log_2 \mu_\theta(x))\frac{(\theta - \theta^*)^2}{2} + o((\theta - \theta^*)^2), \tag{15}$$

where the derivatives are taken at $\theta = \theta^*$.

The first order term is equal to zero, since by definition, μ_{θ^*} is the probability distribution minimizing $-\log_2 \mu_\theta(x)$. If we approximate $\theta - \theta^*$ by ϵ and write $J(\theta^*) := \frac{\partial^2}{\partial\theta^2}(-\ln \mu_\theta(x))$ (which is positive), we find:

$$l(\epsilon) \approx -\log_2(\mu_{\theta^*}(x)) - \log_2 \epsilon + \frac{J(\theta^*)}{\ln 2}\frac{\epsilon^2}{2}. \tag{16}$$

Differentiating with respect to ϵ, we find $\frac{\mathrm{d}l}{\mathrm{d}\epsilon} = \frac{1}{\ln 2}\left(-\frac{1}{\epsilon} + \epsilon J(\theta^*)\right)$. If we denote by ϵ^* the optimal precision, we must have $\frac{\mathrm{d}l}{\mathrm{d}\epsilon}|_{\epsilon=\epsilon^*} = 0$, i.e.

$$\epsilon^* \approx \sqrt{\frac{1}{J(\theta^*)}}, \tag{17}$$

which, by plugging (17) into (16), yields the following codelength:

$$l(\epsilon^*) \approx -\log_2 \mu_{\theta^*}(x) + \frac{1}{2}\log_2 J(\theta^*) + \text{cst}. \tag{18}$$

Essentially, the idea is that if $\theta - \theta^* < \epsilon^*$, and if we denote by x the data, the difference between $P_\theta(x)$ and $P_{\theta^*}(x)$ is not significant enough to justify using more bits to improve the precision of θ. Another possible way to look at this is that we cannot distinguins θ from θ^*, in the sense that it is hard to tell if the data have been sampled from one distribution or the other.

2.1.2 The i.i.d. case: confidence intervals and Fisher information

Let us now consider the i.i.d. case: all the x_i are sampled from the same probability distribution, and we have $-\log_2 \mu_\theta(x) = \sum_i -\log_2 \alpha_\theta(x_i)$. In that case, $J(\theta) = \sum_{i=1}^{n} \frac{\partial^2}{\partial \theta^2} \ln \alpha_\theta(x_i)$, so it is roughly proportional to n, and ϵ^* is therefore proportional to $\frac{1}{\sqrt{n}}$: we find the classical confidence interval.

Moreover, if the x_i really follow α_θ, then

$$\mathbb{E}_{x\sim\alpha}(J(\theta)) = n\mathbb{E}_{x\sim\alpha}\left(\frac{\partial^2}{\partial\theta^2}\left(-\ln \alpha_\theta(x)\right)\right) =: nI(\theta), \tag{19}$$

where $I(\theta)$ is the so-called Fisher information. We will often use the following approximation

$$J(\theta) \approx nI(\theta). \tag{20}$$

With this, we can rewrite equation (17) for the i.i.d. case, and we find that the optimal ϵ is approximately equal to:

$$\epsilon^* \approx \frac{1}{\sqrt{n}} \frac{1}{\sqrt{I(\theta^*)}}. \tag{21}$$

By simply studying the optimal precision to use for a parameter from a coding perspective, we managed to recover confidence intervals and the Fisher information.

2.1.3 Link with model selection

Now that we have the optimal precision and the corresponding codelength, we can also solve certain model selections problems.

Consider for example you are playing heads or tails n times, but at the middle of the game, the coin (i.e. the parameter of Bernoulli's law) is changed. You are given the choice to encode a single θ, or θ_1 which will be used for the first half of the data, and θ_2 which will be used for the second half.

Let us compute the codelengths corresponding to these two models. If we denote by x all the data, by x_1 the first half of the data, and by x_2 the second half of the data, we find, by combining equations (18) and (20):

$$l_1 = -\log_2 \mu_\theta(x) + \frac{1}{2}\log_2 I(\theta) + \frac{1}{2}\log_2(n) + \text{cst}, \tag{22}$$

$$l_2 = -\log_2 \mu_{\theta_1}(x_1) - \log_2 \mu_{\theta_2}(x_2) + \frac{1}{2}\log_2 I(\theta_1) + \frac{1}{2}\log_2 I(\theta_2) + \frac{1}{2}\log_2(n/2) + \frac{1}{2}\log_2(n/2) + \text{cst} \tag{23}$$

$$= -\log_2 \mu_{\theta_1,\theta_2}(x) + \frac{2}{2}\log_2(n) + O(1). \tag{24}$$

It is easy to see that for a model with k parameters, we would have:

$$l_k = -\log_2 \mu_{\theta_1,\ldots,\theta_k}(x) + \frac{k}{2}\log_2(n) + O(1). \tag{25}$$

Asymptotically, we obtain the Bayesian information criterion, which is often used. It could be interesting to use the non-asymptotic equation (23) instead, but the Fisher information is usually hard to compute.

It is also interesting to notice that using two-part codes automatically makes the corresponding coding suboptimal, since it reserves several codewords for the same symbol: coding θ_1 followed by x coded with P_{θ_1} yields a different code than θ_2 followed by x coded with P_{θ_2}, but these two codes are codes for x. A solution to this problem is to set $P_\theta(x) = 0$ if there exists θ' such that $P_{\theta'}(x) > P_\theta(x)$, and renormalize. Then, for a given x, only the best estimator can be used to encode x. This yields the normalized maximum likelihood distribution (if we denote by $\hat{\theta}(x)$ the maximum likelihood estimator for x, $\mathrm{NML}(x) = \dfrac{P_{\hat{\theta}(x)}(x)}{\sum_x P_{\hat{\theta}(x)}(x)}$).

Another way to look at this problem is the following: consider as an example the simple case $\Theta = \{1, 2\}$. The encoding of θ corresponds to a prior q on Θ (for example, using one bit to distinguish P_1 from P_2 corresponds to the uniform prior $q(1) = q(2) = 0.5$).

The two-part code corresponds to using $\max(q(1)P_1, q(2)P_2)$ as our "probability distribution" to encode the data, but its integral is not equal to 1: we lose $\int_X \min(q(1)P_1(x), q(2)P_2(x))dx$, which makes the codelengths longer. Consequently, it is more interesting to directly use the mixture $q(1)P_1 + q(2)P_2$ to encode the data when it is possible, because *all* codewords will then be shorter.

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.