

# Pearl's and Jeffrey's Update as Modes of Learning in Probabilistic Programming

Bart Jacobs & Dario Stein

*Institute for Computing and Information Sciences (iCIS)  
Radboud University  
Nijmegen, The Netherlands  
[bart@cs.ru.nl](mailto:bart@cs.ru.nl)    [dario.stein@ru.nl](mailto:dario.stein@ru.nl)*

---

## Abstract

The concept of updating a probability distribution in the light of new evidence lies at the heart of statistics and machine learning. Pearl's and Jeffrey's rule are two natural update mechanisms which lead to different outcomes, yet the similarities and differences remain mysterious. This paper clarifies their relationship in several ways: via separate descriptions of the two update mechanisms in terms of probabilistic programs and sampling semantics, and via different notions of likelihood (for Pearl and for Jeffrey). Moreover, it is shown that Jeffrey's update rule arises via variational inference. In terms of categorical probability theory, this amounts to an analysis of the situation in terms of the behaviour of the multiset functor, extended to the Kleisli category of the distribution monad.

*Keywords:* probabilistic reasoning, probabilistic programming, category theory, machine learning, statistical inference, variational inference, denotational semantics, Pearl, Jeffrey.

---

## 1 Introduction

Suppose you test for a certain disease, say Covid. You take three consecutive tests, because you wish to be sure – two of them come out positive but one is negative. How do you compute the subsequent (posterior) probability that you actually have the disease? In a medical setting one starts from a prevalence, that is, an *a priori* disease probability, which is assumed to hold for the whole population. Medical tests are typically not perfect: one has to take their sensitivity and specificity into account. They tell, respectively, if someone has the disease, the probability that the test is positive, and if someone does not have the disease, the probability that the test is negative.

When all these probabilities (prevalence, sensitivity, specificity) are known, one can apply Bayes' rule and obtain the posterior probability after a single test. But what if we do three tests? And what if we do a thousand tests?

It turns out that things become fuzzy when tests are repeated multiple times. One can distinguish two approaches, associated with Pearl and Jeffrey. They agree on single tests. But they may disagree wildly on multiple tests, see the example in Section 2 below. This is disconcerting, certainly in the current age of machine learning, in which so many decisions are based on statistical learning and decision making.

Earlier work (of one of the authors) [6,8] analysed the approaches of Pearl and Jeffrey. The difference there was formulated in terms of learning from 'what is right' and from 'what is wrong'. As will be

recalled below, Pearl’s update rule involves increasing validity (expected value), whereas Jeffrey’s rule involves decreasing (Kullback-Leibler) divergence. The contributions of this paper are threefold.

- It adds the perspective of probabilistic programming. Pearl’s and Jeffrey’s approaches to updating are formulated, for the medical test example, in a standard probabilistic programming language, namely WebPPL [4,5], see Section 2. Pearl’s update is straightforwardly expressible using built-in conditioning constructs, while Jeffrey’s update involves nested inference, a simple form of reasoning about reasoning [13]. We further explore the different dynamics behind the two update techniques are operationally using rejection samplers in Section 6.
- The paper also offers a new perspective on the Pearl/Jeffrey distinction in terms of different underlying generative models and their associated likelihoods: with Pearl’s update rule one increases one form of ‘Pearl’ likelihood, whereas with Jeffrey’s update rule one increases another form of ‘Jeffrey’ likelihood. These two likelihoods are described in terms of different forms of evaluating data (as a multiset of data points) with respect to a multinomial distribution. These two forms of likelihood are directly related to the respective update mechanisms, see Section 7. Pearl likelihood occurs in practice, for example as the basis of the multinomial naive Bayes classifier [12], while Jeffrey likelihood — and its difference to Pearl’s — is new, as far as we know.
- Pearl’s likelihood directly leads to the associated update rule, see Theorem 7.3. For Jeffrey’s likelihood the connection is more subtle and involves variational inference [10,11]: it is shown that Jeffrey’s update is least divergent from the update rule for Jeffrey likelihood, in a suitable sense, see Theorem 8.5. This likelihood update rule is described categorically in terms of the extension of the multiset functor to the Kleisli category of the (discrete) distribution monad, see [3,7]. This analysis clarifies the mathematical situation, for instance in Equation 11, where it is shown that this extended multiset functor commutes with the ‘dagger’ reversal of channels. This is a new result, with a certain esthetic value.

This paper develops the idea that Pearl’s and Jeffrey’s rule involve a difference in perspective: are we trying to learn something about an individual or about a population?

## 2 A Motivating Example

Consider some disease with an *a priori* probability (or ‘prevalence’) of 5%. There is a test for the disease with the following characteristics:

- (‘sensitivity’) If someone has the disease, then the test is positive with probability of 90%.
- (‘specificity’) If someone does not have the disease, there is a 95% chance that the test is negative.

We are told that someone takes three consecutive tests and sees two positive and one negative outcome. These test outcomes are our observed data that we wish to learn from.

The question is: what is the posterior probability that this person has the disease, in the light of this test data? You may wish to stop reading here and calculate this probability yourself. Outcomes, using Pearl’s and Jeffrey’s rule, will be provided in Examples 4.3 and 5.2 below.

Below we present several possible implementations of the medical test situation in the probabilistic programming language WebPPL [4,5], giving three different solutions to the above question. The code starts by defining a function `test` which models the test outcome, incorporating the above sensitivity and specificity. Here, `flip(p)` tosses a biased coin with bias `p`.

---

```
var test = function(dis) {
  return dis ? (flip(0.9) ? 'pos' : 'neg') : (flip(0.95) ? 'neg' : 'pos');
}
```

---

We then define three inference functions which we simply label as `prog1`, `prog2`, `prog3`. At this stage we do not wish to connect them to Pearl/Jeffrey. We invite the reader to form a judgement about what is

the ‘right’ way to model the above situation with three test outcomes (‘pos’, ‘pos’, ‘neg’).

---

```

var prog1 = function() {
  var dis = flip(0.05);
  condition(test(dis) == 'pos');
  condition(test(dis) == 'pos');
  condition(test(dis) == 'neg');
  return dis;
}

var prog2 = function() {
  var target = uniformDraw(['pos', 'pos', 'neg']);
  var dis = flip(0.05);
  condition(test(dis) == target);
  return dis;
}

var prog3 = function() {
  var target = uniformDraw(['pos', 'pos', 'neg']);
  return sample(Infer(function() {
    var dis = flip(0.05);
    condition(test(dis) == target);
    return dis;
  })))
}

```

---

All functions make use of the `condition` command to instruct WebPPL to compute a conditional probability distribution. `prog1` uses three successive conditions, while the other two use a single condition on a randomly chosen target. `prog3` additionally makes use of *nested inference*, that is, it wraps the `Infer` function around part of its code. Nested inference is a form of reasoning about reasoning [13] and has been applied for example to the study of social cognition, linguistics and theory of mind [5, Ch. 6]. We give a short overview of WebPPL’s semantics and usage in Section 10. All programs can be run using exhaustive enumeration or rejection sampling as inference algorithms, which we elaborate further in Section 4.

The three functions can be executed in WebPPL and the posteriors visualized using the command `viz(Infer(prog1))`. The posterior disease probabilities of each of the programs are respectively:

- `prog1`: 64%
- `prog2`: 9%
- `prog3`: 33%

The same probabilities appear in the mathematical analysis in Examples 4.3 and 5.2 below.

An interesting question to ask is: suppose we do not have 3 tests (2 positive, 1 negative), but 3000 tests (2000 positive, 1000 negative). Does that change the outcome of the above computations? Not so for the second and third program, which only require a statistical sample of the data. The first program however, quickly converges to 100% disease probability when the number of tests increases (still assuming the same ratio of 2 positive and 1 negative). But this first program becomes increasingly difficult to compute, because each test result emits further conditioning instructions that the inference engine needs to take into account. The two other programs on the other hand scale almost trivially. We return to this scaling issue at the end of Section 7.

The three implementations will be reiterated throughout the paper and related to Pearl’s and Jeffrey’s update. In Section 6, where we also make their semantics explicit using rejection samplers.

### 3 Multisets, Distributions, and Channels

Sections 3 – 5 introduce the mathematics underlying the update situations that we are looking at. This material is in essence a recap from [6,8]. We write  $\mathcal{M}$  and  $\mathcal{D}$  for the multiset and distribution monads on the category **Sets** of sets and functions. For a set  $X$ , multisets  $\varphi \in \mathcal{M}(X)$  can equivalently be written as a function  $\varphi: X \rightarrow \mathbb{N}$  with finite support, or as a finite formal sum  $\sum_i n_i |x_i\rangle$ , where  $n_i \in \mathbb{N}$  is the multiplicity of element  $x_i \in X$ . Similarly, a distribution  $\omega \in \mathcal{D}(X)$  is written either as a function  $\omega: X \rightarrow [0, 1]$  with finite support and  $\sum_x \omega(x) = 1$ , or as a finite formal convex combination  $\sum_i r_i |x_i\rangle$  with  $r_i \in [0, 1]$  satisfying  $\sum_i r_i = 1$ .

Functoriality of  $\mathcal{M}$  (and  $\mathcal{D}$ ) works in the following manner. For a function  $f: X \rightarrow Y$  we have  $\mathcal{M}(f): \mathcal{M}(X) \rightarrow \mathcal{M}(Y)$ , given as  $\mathcal{M}(f)(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x)$ .

For a multiset  $\varphi \in \mathcal{M}(X)$  we write  $\|\varphi\| \in \mathbb{N}$  for its size, defined as sum of its multiplicities:  $\|\varphi\| := \sum_x \varphi(x)$ . When this size is not zero, we can define an associated distribution  $\text{flrn}(\varphi) \in \mathcal{D}(X)$ , via frequentist learning (normalisation), as:

$$\text{flrn}(\varphi) := \sum_{x \in X} \frac{\varphi(x)}{\|\varphi\|} |x\rangle.$$

For  $K \in \mathbb{N}$  we write  $\mathcal{M}[K](X) = \{\varphi \in \mathcal{M}(X) \mid \|\varphi\| = K\}$  for the set of multiset of size  $K$ . There is an accumulation function  $\text{acc}: X^K \rightarrow \mathcal{M}[K](X)$ , given by  $\text{acc}(x_1, \dots, x_K) = 1|x_1\rangle + \dots + 1|x_K\rangle$ . For instance  $\text{acc}(a, b, a, c, a, b) = 3|a\rangle + 2|b\rangle + 1|c\rangle$ , using  $X = \{a, b, c\}$  and  $K = 6$ .

For two distributions  $\omega \in \mathcal{D}(X)$ ,  $\rho \in \mathcal{D}(Y)$  one can form the (parallel) product distribution  $\omega \otimes \rho \in \mathcal{D}(X \times Y)$ , with  $(\omega \otimes \rho)(x, y) = \omega(x) \cdot \rho(y)$ . We often use the  $K$ -fold product  $\omega^K = \omega \otimes \dots \otimes \omega \in \mathcal{D}(X^K)$ .

A distribution  $\omega \in \mathcal{D}(X)$  may be seen as an urn with coloured balls, where  $X$  is the set of colours. The number  $\omega(x) \in [0, 1]$  is the probability of drawing a ball of colour  $x$ . We are interested in  $K$ -sized draws, formalised as multiset  $\varphi \in \mathcal{M}[K](X)$ . The multinomial distribution  $\text{mn}[K](\omega) \in \mathcal{D}(\mathcal{M}[K](X))$  assigns probabilities to such draws:

$$\text{mn}[K](\omega) := \mathcal{D}(\text{acc})(\omega^K) = \sum_{\varphi \in \mathcal{M}[K](X)} (\varphi) \cdot \prod_{x \in X} \omega(x)^{\varphi(x)} | \varphi \rangle \quad \text{where} \quad (\varphi) := \frac{\|\varphi\|!}{\prod_x \varphi(x)!}. \quad (1)$$

A Kleisli map  $c: X \rightarrow \mathcal{D}(Y)$  for the distribution monad  $\mathcal{D}$  is often called a *channel*, and written as  $c: X \rightarrow Y$ . For instance, the above accumulation map  $\text{acc}: X^K \rightarrow \mathcal{M}[K](X)$  has a probabilistic inverse  $\text{arr}: \mathcal{M}[K](X) \rightarrow \mathcal{D}(X^K)$ , where  $\text{arr}$  stands for arrangement, see [7] for details. This arrangement is defined as:

$$\text{arr}(\varphi) := \sum_{\mathbf{x} \in \text{acc}^{-1}(\varphi)} \frac{1}{(\varphi)} | \mathbf{x} \rangle \quad \text{with } (\varphi) \text{ as defined in (1)}. \quad (2)$$

Kleisli extension gives a pushforward operation along a channel: a distribution  $\omega \in \mathcal{D}(X)$  can be turned into a distribution  $c \gg \omega \in \mathcal{D}(Y)$  via the formula:

$$c \gg \omega := \sum_{y \in Y} \left( \sum_{x \in X} \omega(x) \cdot c(x)(y) \right) | y \rangle.$$

This new distribution  $c \gg \omega$  is often called the prediction. One can prove:  $\text{flrn} \gg \text{mn}[K](\omega) = \omega$  and  $\text{arr} \gg \text{mn}[K](\omega) = \omega^K$ , see [7].

The following two programs are equivalent ways of sampling from a prediction  $c \gg \omega$ :

$$\frac{\begin{array}{l} \mathbf{x} \leftarrow \omega \\ \mathbf{y} \leftarrow c(\mathbf{x}) \\ \text{samples.add}(\mathbf{y}) \end{array}}{\quad} \qquad \frac{\begin{array}{l} \mathbf{y} \leftarrow c \gg \omega \\ \text{samples.add}(\mathbf{y}) \end{array}}{\quad} \quad (3)$$

It shows that such sampling can be done in two steps: The notation  $x \leftarrow \omega$  is used for sampling a random element  $x \in X$  from a distribution  $\omega \in \mathcal{D}(X)$ , where the randomness takes the probabilities in  $\omega$  into account. This is a standard construct in probabilistic programming. If multiple samples  $x_i \leftarrow \omega$  are taken, and accumulated in a multiset  $\varphi \in \mathcal{M}(X)$ , then the normalisation  $\text{flrn}(\varphi)$  of  $\varphi$  approaches the original distribution  $\omega$ .

Lastly, the tensor product  $\otimes$  extends pointwise to channels:  $(c \otimes d)(x, y) = c(x) \otimes d(y)$ . Then one can prove, for instance,  $(c \otimes d) \gg (\omega \otimes \rho) = (c \gg \omega) \otimes (d \gg \rho)$ .

#### 4 Validity, Conditioning, and Pearl’s Update Rule

A (fuzzy) predicate on a set  $X$  is a function  $p: X \rightarrow [0, 1]$ . Each element  $x \in X$  gives rise to a *point predicate*  $\mathbf{1}_x: X \rightarrow [0, 1]$ , with  $\mathbf{1}_x(y) = 1$  if  $x = y$  and  $\mathbf{1}_x(y) = 0$  if  $x \neq y$ . For two predicates  $p_1, p_2: X \rightarrow [0, 1]$  we can form a conjunction  $p_1 \& p_2: X \rightarrow [0, 1]$  via pointwise multiplication:  $(p_1 \& p_2)(x) = p_1(x) \cdot p_2(x)$ .

The validity (or expected value) of a predicate  $p: X \rightarrow [0, 1]$  in a distribution  $\omega \in \mathcal{D}(X)$  is written as  $\omega \models p$  and defined as:

$$\omega \models p := \sum_{x \in X} \omega(x) \cdot p(x).$$

When this validity is non-zero we can define the updated distribution  $\omega|_p \in \mathcal{D}(X)$  as:

$$\omega|_p := \sum_{x \in X} \frac{\omega(x) \cdot p(x)}{\omega \models p} |x\rangle. \quad (4)$$

For a channel  $c: X \rightarrow Y$  and a predicate  $q: Y \rightarrow [0, 1]$  on its codomain, we can define a pullback predicate  $c \ll q$  on  $X$  via the formula:

$$(c \ll q)(x) := \sum_{y \in Y} c(x)(y) \cdot q(y).$$

The following result contains the basic facts that we need here. Proofs can be found for instance in [6,8].

**Lemma 4.1** *For a channel  $c: X \rightarrow Y$ , a distribution  $\omega \in \mathcal{D}(X)$ , predicates  $p, p_1, p_2$  on  $X$  and  $q$  on  $Y$ ,*

- (i)  $c \gg \omega \models q = \omega \models c \ll q$ ;
- (ii)  $\omega|_{p_1}|_{p_2} = \omega|_{p_1 \& p_2}$ ;
- (iii)  $\omega|_p \models p \geq \omega \models p$ . □

The last result shows that a predicate  $p$  is ‘more true’ in an updated distribution  $\omega|_p$  than in the original  $\omega$ . The next result from [6,8] contains both the formulation of Pearl’s update, and the associated validity increase.

**Theorem 4.2** *Let  $c: X \rightarrow Y$  be a channel with a prior distribution  $\omega \in \mathcal{D}(X)$  on its domain and a predicate  $q: Y \rightarrow [0, 1]$  on its codomain. The posterior distribution  $\omega_P \in \mathcal{D}(X)$  of  $\omega$ , via Pearl’s update rule, with the evidence predicate  $q$ , is defined as:*

$$\omega_P := \omega|_{c \ll q} \quad \text{and satisfies} \quad c \gg \omega_P \models q \geq c \gg \omega \models q. \quad \square$$

The proof follows from an easy combination of points (i) and (iii) of Lemma 4.1. The increase in validity that is achieved via Pearl's rule means that the validity of predicate  $q$  is higher in the predicted distribution obtained from the posterior distribution  $\omega_P$ , than in the prediction obtained from original, prior distribution  $\omega$ .

The following are two rejection samplers that allow sampling from a posterior distribution: On the left below we show how to obtain an updated distribution  $\omega|_p$  via sampling, and on the right how to get a Pearl update  $\omega|_{c \ll q}$ .

$$\begin{array}{c}
 \hline
 \mathbf{x} \leftarrow \omega \\
 \mathbf{y} \leftarrow \mathbf{flip}(p(\mathbf{x})) \\
 \mathbf{if} \ \mathbf{y} == 1: \\
 \quad \mathbf{samples.add}(\mathbf{x}) \\
 \hline
 \end{array}
 \qquad
 \begin{array}{c}
 \hline
 \mathbf{x} \leftarrow \omega \\
 \mathbf{y} \leftarrow c(\mathbf{x}) \\
 \mathbf{z} \leftarrow \mathbf{flip}(q(\mathbf{y})) \\
 \mathbf{if} \ \mathbf{z} == 1: \\
 \quad \mathbf{samples.add}(\mathbf{x}) \\
 \hline
 \end{array}
 \tag{5}$$

The probabilistic program `prog1` at the end of Section 2 computes the Pearl update. How this update works in detail will be described next.

**Example 4.3** We are now in a situation to explain the 64% posterior disease probability claimed in Section 2. It is obtained via repeated Pearl updates. We first translate the information given there into mathematical structure.

We use  $X = \{d, d^\perp\}$  for the set with elements  $d$  for disease and  $d^\perp$  for no-disease. The given prevalence of 5% for the disease corresponds to a prior distribution  $\omega \in \mathcal{D}(X)$  given by  $\omega = \frac{1}{20}|d\rangle + \frac{19}{20}|d^\perp\rangle$ .

The test is formalised as a channel  $c: X \rightarrow \mathcal{D}(Y)$  where  $Y = \{p, n\}$  the set of positive and negative test outcomes. The sensitivity and specificity of the test translate into, respectively:

$$c(d) := \frac{9}{10}|p\rangle + \frac{1}{10}|n\rangle \quad \text{and} \quad c(d^\perp) := \frac{1}{20}|p\rangle + \frac{19}{20}|n\rangle.$$

There are two obvious point predicates  $\mathbf{1}_p: Y \rightarrow [0, 1]$  and  $\mathbf{1}_n: Y \rightarrow [0, 1]$  on the set  $Y = \{p, n\}$  of test outcomes. We are told that there are two positive and one negative test. This translates in the conjunction  $(c \ll \mathbf{1}_p) \& (c \ll \mathbf{1}_p) \& (c \ll \mathbf{1}_n)$ . Since conjunction is commutative, the order does not matter. Updating with this conjunction is equivalent to three successive update, see Lemma 4.1 (ii), and gives the claimed outcome:

$$\begin{aligned}
 \omega_P = \omega|_{(c \ll \mathbf{1}_p) \& (c \ll \mathbf{1}_p) \& (c \ll \mathbf{1}_n)} &= \omega|_{c \ll \mathbf{1}_p}|_{c \ll \mathbf{1}_p}|_{c \ll \mathbf{1}_n} = \frac{648}{1009}|d\rangle + \frac{361}{1009}|d^\perp\rangle \\
 &\approx 0.642|d\rangle + 0.358|d^\perp\rangle.
 \end{aligned}$$

This is the probability computed in `prog1` in Section 2.

The validity increase associated with Pearl's update rule takes the following form.

$$c \gg \omega_P \models (c \ll \mathbf{1}_p)^2 \& (c \ll \mathbf{1}_n) \approx 0.049 \geq 0.0096 \approx c \gg \omega \models (c \ll \mathbf{1}_p)^2 \& (c \ll \mathbf{1}_n).$$

## 5 Dagger channels and Jeffrey's update rule

First we recall that the difference (divergence) between two distributions  $\omega, \rho \in \mathcal{D}(X)$  is commonly expressed as *Kullback-Leibler divergence*, defined as:

$$D_{KL}(\omega, \rho) := \sum_{x \in X} \omega(x) \cdot \ln \left( \frac{\omega(x)}{\rho(x)} \right), \quad \text{where } \ln \text{ is the natural logarithm.} \tag{6}$$

The main ingredient that we need for Jeffrey’s rule is the dagger of a channel  $c: X \rightarrow Y$  with respect to a prior distribution  $\omega \in \mathcal{D}(X)$ . This dagger is a channel  $c_\omega^\dagger: Y \rightarrow X$  in the opposite direction. It is also called Bayesian inversion, see [2,1], and it is defined on  $y \in Y$  as:

$$c_\omega^\dagger(y) := \omega|_{c \ll \mathbf{1}_y} \stackrel{(4)}{=} \sum_{x \in X} \frac{\omega(x) \cdot c(x)(y)}{(c \gg \omega)(y)} |x\rangle. \quad (7)$$

We again combine Jeffrey’s rule with its main divergence reduction property, from [8]. The set-up is very much as for Pearl’s rule, in Theorem 4.2, but with evidence now in the form of distribution instead of a predicate.

**Theorem 5.1** *Let  $c: X \rightarrow Y$  be a channel with a prior distribution  $\omega \in \mathcal{D}(X)$  and an evidence distribution  $\tau \in \mathcal{D}(Y)$ . The posterior distribution  $\omega_J \in \mathcal{D}(X)$  of  $\omega$ , obtained via Jeffrey’s update rule, with the evidence distribution  $\tau$ , is defined as:*

$$\omega_J := c_\omega^\dagger \gg \tau \quad \text{and satisfies} \quad D_{KL}(\tau, c \gg \omega_J) \leq D_{KL}(\tau, c \gg \omega). \quad \square$$

The proof of this divergence decrease is remarkably hard, see [8] for details. The result says that the prediction from  $\omega_J$  is less wrong than from  $\omega$ , when compared to the ‘target’ distribution  $\tau$ .

**Example 5.2** We build on the test channel  $c: X \rightarrow Y$  and prevalence distribution  $\omega \in \mathcal{D}(X)$  from Example 4.3. The first task is to compute the dagger channel  $f := c_\omega^\dagger: Y \rightarrow X$ . It yields:

$$f(p) = \frac{18}{37}|d\rangle + \frac{19}{37}|d^\perp\rangle \quad \text{and} \quad f(n) = \frac{2}{363}|d\rangle + \frac{361}{363}|d^\perp\rangle.$$

The fact that there are two positive and one negative test translates into the ‘empirical’ evidence distribution  $\tau = \frac{2}{3}|p\rangle + \frac{1}{3}|n\rangle \in \mathcal{D}(Y)$ . The posterior, updated disease distribution, obtained from this evidence, gives the 33% probability mentioned in Section 2:

$$\omega_J = f \gg \tau = \frac{13142}{40293}|d\rangle + \frac{27151}{40293}|d^\perp\rangle \approx 0.326|d\rangle + 0.674|d^\perp\rangle.$$

This probability is computed by `prog3` in Section 2.

The divergence decrease from Theorem 5.1 takes the following form:

$$D_{KL}(\tau, c \gg \omega_J) \approx 0.24 \leq 0.98 \approx D_{KL}(\tau, c \gg \omega).$$

Having seen this, we may ask: why not use the evidence distribution  $\tau = \frac{2}{3}|p\rangle + \frac{1}{3}|n\rangle$  not as a predicate  $q = \frac{2}{3}\mathbf{1}_p + \frac{1}{3}\mathbf{1}_n$ , and then do a single Pearl update:

$$\omega|_{c \ll q} = \frac{2}{23}|d\rangle + \frac{21}{23}|d^\perp\rangle \approx 0.087|d\rangle + 0.913|d^\perp\rangle. \quad (8)$$

This is the distribution computed by program `prog2` in Section 2.

For future use we record the following standard properties of the dagger of a channel (7).

**Lemma 5.3** (i) *Daggers preserve sequential composition: for two successive channels  $X \xrightarrow{c} Y \xrightarrow{d} Z$  and a distribution  $\omega \in \mathcal{D}(X)$ ,*

$$(d \circ c)_\omega^\dagger = c_\omega^\dagger \circ d_{c \gg \omega}^\dagger.$$

(ii) *Daggers preserve parallel composition: for two channels  $c: X \rightarrow A$ ,  $d: Y \rightarrow B$  with distributions  $\omega \in \mathcal{D}(X)$ ,  $\rho \in \mathcal{D}(Y)$ ,*

$$(c \otimes d)_{\omega \otimes \rho}^\dagger = c_\omega^\dagger \otimes d_\rho^\dagger.$$



## 6 An Operational Understanding of Jeffrey’s Update

We return to the probabilistic programs of Section 2. As discussed in Section 4, `prog1` expresses repeated Pearl updates. It remains to understand the difference between `prog2` and `prog3`. As shown in (8), `prog2` corresponds to a single Pearl’s update with the target distribution, as predicate. Further, `prog3` is Jeffrey’s update, with the nested inference corresponding to the computation of the dagger channel  $c_{\omega}^{\dagger}$ . The difference between the two programs `prog2` and `prog3` is surprisingly subtle, so we begin by illustrating it using a different kind of metaphor, and derive a rejection sampler for each case in turn.

Consider a large queue of people waiting in front of a club. Each person prefers either rock or pop. The club’s management wants to achieve a target ratio of 75% rock fans on the inside. To that end, they equip their doorman with a special ticker device, see Figure 6. The ticker displays a current target (either ‘Rock’ or ‘Pop’), and the doorman admits the next person if and only if they prefer the targeted style. The doorman can click the device to obtain a new target (either by cycling sequentially through the targets, or picking one randomly), but there remains a choice when to click.

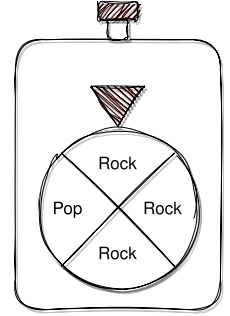


Fig. 1: Ticker device

- (i) Single Pearl Policy: pick a new target after every person:

---

```

for person in queue:
  if person.preference == ticker.target:
    club.admit(person)
  ticker.click()

```

---

- (ii) Jeffrey Policy: pick a new target only after admitting a person:

---

```

for person in queue:
  if person.preference == ticker.target:
    club.admit(person)
    ticker.click()

```

---

It may be clear that only the Jeffrey Policy is suitable to achieve the management’s goal. Approximately 75% of the people which are admitted are rock fans. This is in line with the key property of Jeffrey’s update rule: reducing the divergence with the target distribution  $\tau$ , see Theorem 5.1. It is unclear what the single Pearl policy achieves in this context.

We may also wonder how the door policy influences other statistical properties of the audience (such as age or gender) which may correlate with music preference: If the prior distribution in the queue is  $\omega$ , what will the resulting distribution be inside the club? For the Jeffrey Policy, this update is precisely described by Jeffrey’s update. We summarize this section with a concrete description of rejection samplers for Pearl’s update with a random target (left) and Jeffrey’s update (right), corresponding to the semantics of the probabilistic programs `prog2` and `prog3`:

---

```

while True:
  x ←  $\omega$ 
  y ← c(x)
  target ←  $\tau$ 
  if y == target:
    samples.add(x)

```

---



---

```

while True:
  x ←  $\omega$ 
  y ← c(x)
  if y == target:
    samples.add(x)
    target ←  $\tau$ 

```

---

(9)



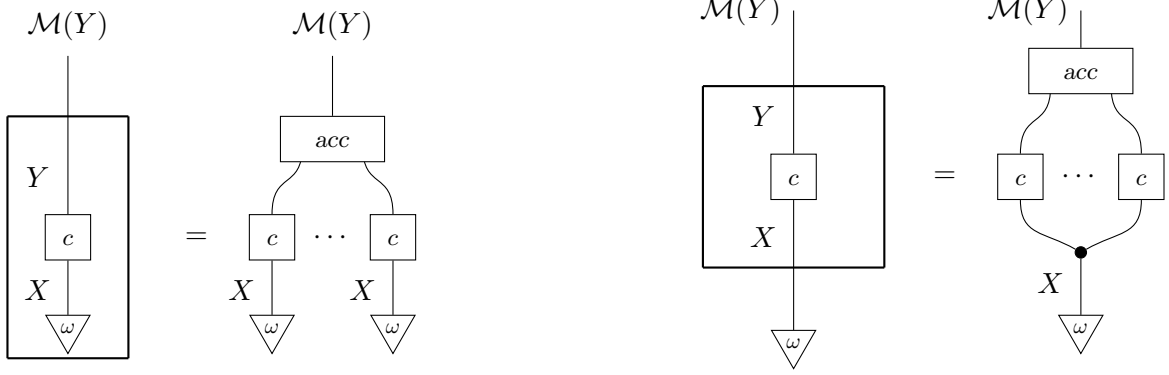


Fig. 2. Graphical representation of Jeffrey likelihood on the left, and Pearl likelihood on the right, see Definition 7.1.

## 7 Likelihoods and Generative Models for Pearl and Jeffrey

This section first identifies two forms of likelihood of data in the situation with a statistical model given by a channel  $X \rightarrow Y$  and a distribution on  $X$ . It then relates these two forms of likelihood to the two update rules of repeated-Pearl and Jeffrey — in Theorems 4.2 and 5.1.

**Definition 7.1** Let  $\psi \in \mathcal{M}[K](Y)$  be a multiset of data, of size  $K = \|\psi\| \in \mathbb{N}$ . Let  $c: X \rightarrow Y$  be a channel with a distribution  $\omega \in \mathcal{D}(X)$  on its domain.

- (i) The *Jeffrey likelihood* of the multiset  $\psi$  is given by the number:

$$mn[K](c \gg \omega)(\psi).$$

- (ii) The *Pearl likelihood* of  $\psi$  in the same model is the first expression below, which has several alternative formulations. It uses the abbreviation  $mn[K](c) := mn[K] \circ c$ .

$$\begin{aligned} (mn[K](c) \gg \omega)(\psi) &= mn[K](c) \gg \omega \models \mathbf{1}_\psi \\ &= \omega \models mn[K](c) \ll \mathbf{1}_\psi \quad \text{by Lemma 4.1 (i)}. \end{aligned}$$

Associated to these two likelihoods are different generative models, *i.e.* distributions over multisets, in  $\mathcal{D}(\mathcal{M}[K](Y))$ , which we evaluate on the dataset  $\psi$ . For Jeffrey likelihood in item (i) we first do the Kleisli extension  $c \gg (\cdot)$  of  $c$  and then take the multinomial, as in the composite:

$$\mathcal{D}(X) \xrightarrow{c \gg (\cdot)} \mathcal{D}(Y) \xrightarrow{mn[K]} \mathcal{D}(\mathcal{M}[K](Y)).$$

We can concisely illustrate this with string diagrams using an informal ‘plate’ notation to copy parts of the string diagram (inspired by the use of plates in graphical models), see Figure 2 on the left. In contrast, for the Pearl likelihood in item (ii) we use the composite  $mn[K](c) := mn[K] \circ c$  in the pushforward:

$$\mathcal{D}(X) \xrightarrow{mn[K](c) \gg (\cdot)} \mathcal{D}(\mathcal{M}[K](Y)).$$

Here, the plate does not extend over the distribution  $\omega$ , whose output is copied instead of resampled, see Figure 2 on the right.

The Pearl likelihood is used in the multinomial naive Bayes classifier [12]. For the likelihood of Jeffrey we shall see alternative formulations in Section 8 below.

Our first result says that minimising the Kullback-Leibler divergence that occurs in Theorem 5.1 — and that is actually reduced by Jeffrey's update rule — corresponds to maximising the Jeffrey likelihood of Definition 7.1 (i).

**Theorem 7.2** (i) *For distributions  $\omega, \omega' \in \mathcal{D}(X)$  and channels  $c, c': X \multimap Y$ , with data  $\psi \in \mathcal{M}(Y)$ , we have that Jeffrey likelihood is oppositely ordered to Kullback-Leibler divergence in:*

$$\text{mn}[K](c \gg \omega)(\psi) \leq \text{mn}[K](c' \gg \omega')(\psi) \iff D_{\text{KL}}(\text{flrn}(\psi), c \gg \omega) \geq D_{\text{KL}}(\text{flrn}(\psi), c' \gg \omega').$$

(ii) *Fix a channel  $c: X \multimap Y$ . Then:*

$$\operatorname{argmax}_{\omega \in \mathcal{D}(X)} \text{mn}[K](c \gg \omega)(\psi) = \operatorname{argmin}_{\omega \in \mathcal{D}(X)} D_{\text{KL}}(\text{flrn}(\psi), c \gg \omega).$$

The above expression on the right is the divergence between the data distribution and the prediction  $c \gg \omega$ . This divergence can be reduced via Jeffrey's rule. The above result says that Jeffrey's rule thus increases the Jeffrey likelihood, see Theorem 5.1.

**Proof.** We only prove the first item, since the second one is a direct consequence. We use that the natural logarithm  $\ln: \mathbb{R}_{>0} \rightarrow \mathbb{R}$  preserves and reflects the order:  $a \leq b$  iff  $\ln(a) \leq \ln(b)$ . This is used in the first step below. We additionally use that the logarithm sends multiplications to sums.

$$\begin{aligned} \text{mn}[K](c \gg \omega)(\psi) &\leq \text{mn}[K](c' \gg \omega')(\psi) \\ \iff \ln \left( \text{mn}[K](c \gg \omega)(\psi) \right) &\leq \ln \left( \text{mn}[K](c' \gg \omega')(\psi) \right) \\ \iff \ln \left( (\psi) \cdot \prod_{y \in Y} (c \gg \omega)(y)^{\psi(y)} \right) &\leq \ln \left( (\psi) \cdot \prod_{y \in Y} (c' \gg \omega')(y)^{\psi(y)} \right) \\ \iff \ln \left( (\psi) \right) + \sum_{y \in Y} \psi(y) \cdot \ln \left( (c \gg \omega)(y) \right) &\leq \ln \left( (\psi) \right) + \sum_{y \in Y} \psi(y) \cdot \ln \left( (c' \gg \omega')(y) \right) \\ \iff - \sum_{y \in Y} \frac{\psi(y)}{\|\psi\|} \cdot \ln \left( (c \gg \omega)(y) \right) &\geq - \sum_{y \in Y} \frac{\psi(y)}{\|\psi\|} \cdot \ln \left( (c' \gg \omega')(y) \right) \\ \iff \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( \text{flrn}(\psi)(y) \right) - \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( (c \gg \omega)(y) \right) & \\ &\geq \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( \text{flrn}(\psi)(y) \right) - \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( (c' \gg \omega')(y) \right) \\ \iff \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( \frac{\text{flrn}(\psi)(y)}{(c \gg \omega)(y)} \right) &\geq \sum_{y \in Y} \text{flrn}(\psi)(y) \cdot \ln \left( \frac{\text{flrn}(\psi)(y)}{(c' \gg \omega')(y)} \right) \\ \iff D_{\text{KL}}(\text{flrn}(\psi), c \gg \omega) &\geq D_{\text{KL}}(\text{flrn}(\psi), c' \gg \omega'). \quad \square \end{aligned}$$

We also relate Pearl likelihood to Pearl's update rule.

**Theorem 7.3** *Consider a channel  $c: X \multimap Y$  with distribution  $\omega \in \mathcal{D}(X)$  and data  $\psi \in \mathcal{M}[K](Y)$ . The validity increase of Theorem 4.2, applied to the last formulation of Pearl likelihood in Definition 7.1 (ii), gives an increase of Pearl likelihood via a repetition of Pearl's rule:*

$$\begin{aligned} (\text{mn}[K](c) \gg \omega)(\psi) &= \omega \models \text{mn}[K](c) \ll \mathbf{1}_\psi \\ &\leq \omega|_{\text{mn}[K](c) \ll \mathbf{1}_\psi} \models \text{mn}[K](c) \ll \mathbf{1}_\psi = (\text{mn}[K](c) \gg \omega|_{\text{mn}[K](c) \ll \mathbf{1}_\psi})(\psi). \end{aligned}$$

This updated distribution  $\omega|_{mn[K](c) \ll \mathbf{1}_\psi} = mn[K](c)^\dagger_\omega(\psi) \in \mathcal{D}(Y)$  can be described via repeated Pearl updates as:

$$\begin{aligned}
\omega|_{mn[K](c) \ll \mathbf{1}_\psi} &= \omega|_{\&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)}} \\
&= \omega|_{(c \ll \mathbf{1}_{y_1})^{\psi(y_1)} \& \dots \& (c \ll \mathbf{1}_{y_n})^{\psi(y_n)}} \quad \text{if } \text{supp}(\psi) = \{y_1, \dots, y_n\} \\
&= \omega|_{\underbrace{(c \ll \mathbf{1}_{y_1}) \& \dots \& (c \ll \mathbf{1}_{y_1})}_{\psi(y_1) \text{ times}} \& \dots \& \underbrace{(c \ll \mathbf{1}_{y_n}) \& \dots \& (c \ll \mathbf{1}_{y_n})}_{\psi(y_n) \text{ times}}} \\
&= \omega|_{c \ll \mathbf{1}_{y_1} \cdots |_{c \ll \mathbf{1}_{y_1}} \cdots |_{c \ll \mathbf{1}_{y_n}} \cdots |_{c \ll \mathbf{1}_{y_n}}}.
\end{aligned}$$

We have used such successive updates in the calculation of the disease probabilities according to Pearl in Example 4.3.

**Proof.** We first note that we can write Pearl’s likelihood as:

$$\begin{aligned}
\omega \models mn[K](c) \ll \mathbf{1}_\psi &= \sum_{x \in X} \omega(x) \cdot mn[K](c(x))(\psi) = \sum_{x \in X} \omega(x) \cdot (\psi) \cdot \prod_{y \in Y} c(x)(y)^{\psi(y)} \\
&= (\psi) \cdot \sum_{x \in X} \omega(x) \cdot \prod_{y \in Y} (c \ll \mathbf{1}_y)(x)^{\psi(y)} \\
&= (\psi) \cdot \sum_{x \in X} \omega(x) \cdot \left( \&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)} \right) (x) \\
&= (\psi) \cdot \left( \omega \models \&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)} \right).
\end{aligned}$$

Now, for  $x \in X$ ,

$$\begin{aligned}
\omega|_{mn[K](c) \ll \mathbf{1}_\psi}(x) &\stackrel{(4)}{=} \frac{\omega(x) \cdot mn[K](c(x))(\psi)}{\omega \models mn[K](c) \ll \mathbf{1}_\psi} = \frac{\omega(x) \cdot (\psi) \cdot (\&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)})(x)}{(\psi) \cdot (\omega \models \&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)})} \\
&= \frac{\omega(x) \cdot (\&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)})(x)}{\omega \models \&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)}} = \omega|_{\&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)}}(x). \quad \square
\end{aligned}$$

The conjunction predicate  $\&_{y \in Y} (c \ll \mathbf{1}_y)^{\psi(y)}$  used in the above Theorem 7.3 loses its value in practice as soon as we have much data, that is, when the multiset  $\psi$  is big. The conjunction involves multiplication of probabilities and thus quickly becomes unmanageably small. Thus, Pearl update works only (in practice) for small amounts of data.

There is an exception however, which is beyond the scope of the current paper. When there is a conjugate prior situation, Pearl updates may happen via updates of the hyperparameters. This does scale to big multisets of data.

## 8 Jeffrey’s Update Rule via Variational Inference

In this section we like to make the idea precise that Jeffrey’s update rule involves a ‘population’ perspective, in contrast to the individual perspective in Pearl’s rule. We show how Jeffrey’s rule emerges from updating a multinomial distribution  $mn[K](\omega)$ . There are two challenges.

- A multinomial distribution  $mn[K](\omega)$  is a distribution on multisets  $\mathcal{M}[K](X)$  of size  $K$ , when  $\omega \in \mathcal{D}(X)$ . When we wish to update along a channel  $c: X \rightarrow Y$  we first have to extend  $c$  to a channel  $\mathcal{M}[K](c): \mathcal{M}[K](X) \rightarrow \mathcal{M}[K](Y)$ . This can be done via an extension of the multiset functor to the Kleisli category  $\mathcal{Kl}(\mathcal{D})$  of the distribution monad  $\mathcal{D}$ . This will occupy us first in this section.

Once we have this channel extension  $\mathcal{M}[K](c)$ , for a multiset of data  $\psi \in \mathcal{M}[K](Y)$  we can form the following update of the multinomial distribution, abbreviated as  $\sigma \in \mathcal{D}(\mathcal{M}[K](X))$ .

$$\sigma := mn[K](\omega) \Big|_{\mathcal{M}[K](c) \ll \mathbf{1}_\psi} \quad (10)$$

We like to think of this  $\sigma$  as a distribution of the form  $mn[K](\omega')$ . The obvious way to obtain this distribution  $\omega'$  is via frequentist learning, as  $flrn \gg \sigma$ . Indeed, as we have seen before (3),  $flrn \gg mn[K](\rho) = \rho$ . The first of our two main results in this section is Theorem 8.3; it says that  $flrn \gg \sigma$  is the Jeffrey update  $c_\omega^\dagger \gg flrn(\psi)$ . This is a technically non-trivial result.

- Next we use techniques from variational inference [10,11]: we like to determine the ‘best’ distribution  $\omega'$  such that  $mn[K](\omega')$  approximates the above distribution  $\sigma$  in (10). We thus look for the distribution with minimal Kullback-Leibler divergence. There again we find Jeffrey's update:

$$\operatorname{argmin}_{\omega' \in \mathcal{D}(X)} D_{KL}(mn[K](\omega'), \sigma) = c_\omega^\dagger \gg flrn(\psi).$$

This is the content of our second main result below, Theorem 8.5.

### 8.1 Jeffrey's rule via Frequentist Learning

Taking multisets of a particular size  $K \in \mathbb{N}$  forms a functor  $\mathcal{M}[K]: \mathbf{Sets} \rightarrow \mathbf{Sets}$ . This functor can be extended to the Kleisli category  $\mathcal{Kl}(\mathcal{D})$  of the distribution monad  $\mathcal{D}$ . This works via a distributive law  $\mathcal{M}[K]\mathcal{D} \Rightarrow \mathcal{D}\mathcal{M}[K]$ , see [3,7]. The extension can also be written via accumulation and arrangement, see Lemma 8.1 (i) below. We shall use it in that form.

The resulting extension is still written as  $\mathcal{M}[K]: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$ . It sends a set/object  $X$  in  $\mathcal{Kl}(\mathcal{D})$  to the set  $\mathcal{M}[K](X)$  of multisets of size  $K$ . On a channel/morphism  $c: X \rightarrow Y$  one defines a channel  $\mathcal{M}[K](c): \mathcal{M}[K](X) \rightarrow \mathcal{M}[K](Y)$  via the distributive law as:

$$\mathcal{M}[K](c) := \left( \mathcal{M}[K](X) \xrightarrow{\mathcal{M}(c)} \mathcal{M}[K](\mathcal{D}(Y)) \xrightarrow{\text{law}} \mathcal{D}(\mathcal{M}[K](Y)) \right).$$

Notice that we have written  $\mathcal{M}(c)$  for the application of the multiset functor  $\mathcal{M}: \mathbf{Sets} \rightarrow \mathbf{Sets}$ , in order to distinguish it from the extension  $\mathcal{M}[K]: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$ .

**Lemma 8.1** (i) For a channel  $c: X \rightarrow Y$  and a number  $K \in \mathbb{N}$  the following diagram commutes.

$$\begin{array}{ccc} X^K & \xrightarrow{c^K} & Y^K \\ \uparrow \text{arr} & & \downarrow \text{acc} \\ \mathcal{M}[K](X) & \xrightarrow{\mathcal{M}[K](c)} & \mathcal{M}[K](Y) \end{array}$$

- (ii) Accumulation  $\text{acc}$  and frequentist learning  $flrn$  are natural transformations between functors extended to Kleisli categories:

$$\begin{array}{ccc} \mathcal{Kl}(\mathcal{D}) & \begin{array}{c} \xrightarrow{(-)^K} \\ \Downarrow \text{acc} \\ \xrightarrow{\mathcal{M}[K]} \end{array} & \mathcal{Kl}(\mathcal{D}) \\ \mathcal{Kl}(\mathcal{D}) & \begin{array}{c} \xrightarrow{\bar{\mathcal{D}}} \\ \Downarrow mn[K] \\ \xrightarrow{\mathcal{M}[K]} \end{array} & \mathcal{Kl}(\mathcal{D}) \\ \mathcal{Kl}(\mathcal{D}) & \begin{array}{c} \xrightarrow{\mathcal{M}[K+1]} \\ \Downarrow flrn \\ \xrightarrow{\text{id}} \end{array} & \mathcal{Kl}(\mathcal{D}) \end{array}$$

The functor  $(-)^K: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$  is the  $K$ -fold tensor product, and  $\bar{\mathcal{D}}: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$  is the standard extension of a monad to its Kleisli category, given on  $c: X \rightarrow Y$  by  $\bar{\mathcal{D}}(c) = \eta \circ c: \mathcal{D}(c): \mathcal{D}(X) \rightarrow \mathcal{D}(Y)$ , where  $\eta$  is the unit of the monad  $\mathcal{D}$ .

**Proof.** This follows from the results in [7].  $\square$

A crucial observation is that the formulation of the extension  $\mathcal{M}[K](c)$  in Lemma 8.1 (i) also works for daggers. It demonstrates that ‘multisets’ and ‘daggers’ commute, see (11) below.

**Proposition 8.2** *Consider a channel  $c: X \rightarrow Y$  with a distribution  $\omega \in \mathcal{D}(X)$  and a number  $K \in \mathbb{N}$ . Then the following diagram of daggers commutes.*

$$\begin{array}{ccc} X^K & \xleftarrow{(c^\dagger_\omega)^K} & Y^K \\ \text{acc} \downarrow & & \uparrow \text{arr} \\ \mathcal{M}[K](X) & \xleftarrow{\mathcal{M}[K](c)^\dagger_{mn[K](\omega)}} & \mathcal{M}[K](Y) \end{array}$$

This means that the extended multiset functor  $\mathcal{M}[K]: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$  commutes with daggers, where the original prior distribution  $\omega$  is replaced by the multinomial distribution  $mn[K](\omega)$ , that is:

$$\mathcal{M}[K](c^\dagger_\omega) = \mathcal{M}[K](c)^\dagger_{mn[K](\omega)}. \quad (11)$$

**Proof.** We concentrate on proving commutation of the diagram, since it implies (11) via Lemma 8.1 (i). We use Lemma 5.3 (i) as first step in:

$$\begin{aligned} \mathcal{M}[K](c)^\dagger_{mn[K](\omega)} &= \left( \text{acc} \circ c^K \circ \text{arr} \right)^\dagger_{mn[K](\omega)} \\ &= \text{arr}^\dagger_{mn[K](\omega)} \circ (c^K)^\dagger_{\text{arr} \gg mn[K](\omega)} \circ \text{acc}^\dagger_{c^K \gg (\text{arr} \gg mn[K](\omega))} \\ &= \text{acc} \circ (c^\dagger_\omega)^K \circ \text{arr}. \end{aligned}$$

This last equation is justified by the three following steps.

- The dagger channel  $\text{arr}^\dagger_{mn[K](\omega)}: X^K \rightarrow \mathcal{M}[K](X)$  is determined on  $\mathbf{x} \in X^K$  as:

$$\begin{aligned} \text{arr}^\dagger_{mn[K](\omega)}(\mathbf{x}) &\stackrel{(7)}{=} \sum_{\varphi \in \mathcal{M}[K](X)} \frac{mn[K](\omega)(\varphi) \cdot \text{arr}(\varphi)(\mathbf{x})}{(\text{arr} \gg mn[K](\omega))(\mathbf{x})} |\varphi\rangle \\ &= \frac{mn[K](\omega)(\text{acc}(\mathbf{x})) \cdot \frac{1}{|\varphi\rangle}}{\omega^K(\mathbf{x})} |\text{acc}(\mathbf{x})\rangle = \frac{\prod_y \omega(y)^{\text{acc}(\mathbf{x})(y)}}{\prod_i \omega(x_i)} |\text{acc}(\mathbf{x})\rangle = 1 |\text{acc}(\mathbf{x})\rangle. \end{aligned}$$

- We again use  $\text{arr} \gg mn[K](\omega) = \omega^K$ , so that we can apply Lemma 5.3 (ii):

$$(c^K)^\dagger_{\text{arr} \gg mn[K](\omega)} = (c^K)^\dagger_{\omega^K} = (c^\dagger_\omega)^K.$$

- For the channel  $\text{acc}^\dagger_{c^K \gg (\text{arr} \gg mn[K](\omega))}: \mathcal{M}[K](Y) \rightarrow Y^K$  we observe that  $c^K \gg (\text{arr} \gg mn[K](\omega)) = c^K \gg \omega^K = (c \gg \omega)^K$  so that:

$$\begin{aligned} \text{acc}^\dagger_{c^K \gg (\text{arr} \gg mn[K](\omega))}(\psi) &= \text{acc}^\dagger_{(c \gg \omega)^K}(\psi) \stackrel{(7)}{=} \sum_{\mathbf{y} \in X^K} \frac{(c \gg \omega)^K(\mathbf{y}) \cdot \text{acc}(\mathbf{y})(\psi)}{(\text{acc} \gg (c \gg \omega)^K)(\psi)} |\mathbf{y}\rangle \\ &= \sum_{\mathbf{y} \in \text{acc}^{-1}(\psi)} \frac{(c \gg \omega)^K(\mathbf{y})}{mn[K](c \gg \omega)(\psi)} |\mathbf{y}\rangle \\ &= \sum_{\mathbf{y} \in \text{acc}^{-1}(\psi)} \frac{1}{|\psi\rangle} |\mathbf{y}\rangle = \text{arr}(\psi). \quad \square \end{aligned}$$

At this stage we return to Jeffrey likelihood  $mn[K](c \gg \omega)(\psi)$ , as described in Definition 7.1 (i). Using the extended functor  $\mathcal{M}[K]: \mathcal{Kl}(\mathcal{D}) \rightarrow \mathcal{Kl}(\mathcal{D})$  and the fact that multinomial is a natural transformation  $mn[K]: \overline{\mathcal{D}} \Rightarrow \mathcal{M}[K]$ , see Lemma 8.1 (ii), we get:

$$\begin{aligned} mn[K](c \gg \omega)(\psi) &= (\mathcal{M}[K](c) \gg mn[K](\omega))(\psi) \\ &= \mathcal{M}[K](c) \gg mn[K](\omega) \models \mathbf{1}_\psi \\ &= mn[K](\omega) \models \mathcal{M}[K](c) \ll \mathbf{1}_\psi. \end{aligned}$$

Lemma 4.1 (iii) tells us that in order to increase the latter validity we have to form the updated distribution  $mn[K](\omega)|_{\mathcal{M}[K](c) \ll \mathbf{1}_\psi}$ , that we abbreviated as  $\sigma$  in (10). The next two results show that this  $\sigma$  is ‘close’ to Jeffrey’s update.

**Theorem 8.3** *Let  $c: X \rightarrow Y$  be a channel with distribution  $\omega \in \mathcal{D}(X)$  and data  $\psi \in \mathcal{M}(Y)$ . Then:*

$$flrn \gg (mn[K](\omega)|_{\mathcal{M}[K](c) \ll \mathbf{1}_\psi}) = c_\omega^\dagger \gg flrn(\psi).$$

**Proof.** By the following argument.

$$\begin{aligned} flrn \gg (mn[K](\omega)|_{\mathcal{M}[K](c) \ll \mathbf{1}_\psi}) &\stackrel{(7)}{=} (flrn \circ \mathcal{M}[K](c)_{mn[K](\omega)}^\dagger)(\psi) \\ &= (flrn \circ acc \circ (c_\omega^\dagger)^K \circ arr)(\psi) && \text{by Proposition 8.2} \\ &= (flrn \circ \mathcal{M}[K](c_\omega^\dagger) \circ acc \circ arr)(\psi) && \text{by Lemma 8.1 (ii)} \\ &= (c_\omega^\dagger \circ flrn)(\psi) && \text{again by Lemma 8.1 (ii)} \\ &= c_\omega^\dagger \gg flrn(\psi). \end{aligned} \quad \square$$

## 8.2 Jeffrey's Rule as Variational Inference

Variational inference [10] is a well-known technique in probability theory for finding approximations  $\tau$  of ‘difficult’ distributions  $\sigma$ . One then determines another distribution  $\tau$  as the distribution (from a certain class) that diverges minimally from  $\sigma$ .

**Lemma 8.4** *Let an arbitrary distribution  $\sigma \in \mathcal{D}(\mathcal{M}[K](X))$  be given. The distribution  $\omega \in \mathcal{D}(X)$  with minimal Kullback-Leibler divergence*

$$D_{KL}(\sigma, mn[K](\omega))$$

*is  $flrn \gg \sigma \in \mathcal{D}(X)$ .*

**Proof.** We first note that  $flrn \gg \sigma \in \mathcal{D}(X)$  is given by:

$$(flrn \gg \sigma)(x) = \sum_{\varphi \in \mathcal{M}[K](X)} \sigma(\varphi) \cdot flrn(\varphi)(x) = \frac{1}{K} \cdot \sum_{\varphi \in \mathcal{M}[K](X)} \sigma(\varphi) \cdot \varphi(x). \quad (*)$$

Then, for an arbitrary  $\omega \in \mathcal{D}(X)$ , we unravel the divergence in the following manner, where *Const* is an

irrelevant constant that depends only on  $\sigma$ , not on  $\omega$ .

$$\begin{aligned}
D_{KL}(\sigma, mn[K](\omega)) &\stackrel{(6)}{=} \sum_{\varphi \in \mathcal{M}[K](X)} \sigma(\varphi) \cdot \ln \left( \frac{\sigma(\varphi)}{mn[K](\omega)(\varphi)} \right) \\
&\stackrel{(1)}{=} \sum_{\varphi \in \mathcal{M}[K](X)} \sigma(\varphi) \cdot \ln \left( \sigma(\varphi) \right) - \sigma(\varphi) \cdot \ln \left( (\varphi) \right) - \sigma(\varphi) \cdot \sum_{x \in X} \varphi(x) \cdot \ln \left( \omega(x) \right) \\
&= Const - \sum_{x \in X} \left( \sum_{\varphi \in \mathcal{M}[K](X)} \sigma(\varphi) \cdot \varphi(x) \right) \cdot \ln \left( \omega(x) \right) \\
&\stackrel{(*)}{=} Const - K \cdot \sum_{x \in X} (flrn \gg= \sigma)(x) \cdot \ln \left( \omega(x) \right) \\
&= Const - K \cdot \ln \left( \prod_{x \in X} \omega(x)^{(flrn \gg= \sigma)(x)} \right).
\end{aligned}$$

Thus, in order to minimise the original divergence  $D_{KL}(\sigma, mn[K](\omega))$  we have to maximise the latter log-expression  $\ln(\dots)$ . This is a familiar maximal likelihood estimation (MLE) problem, see *e.g.* [9, Ex. 17.5]. The log expression is maximal for  $\omega = flrn \gg= \sigma$ .  $\square$

With this lemma we can get our ‘variational’ characterisation of Jeffrey’s theorem.

**Theorem 8.5** Consider a channel  $c: X \rightarrow Y$  with distribution  $\omega \in \mathcal{D}(X)$  and data  $\psi \in \mathcal{M}(Y)$ . Jeffrey’s update  $c_{\omega}^{\dagger} \gg= flrn(\psi)$  is the distribution  $\omega' \in \mathcal{D}(X)$  such that  $mn[K](\omega')$  diverges minimally from multinomial update  $mn[K](\omega)|_{\mathcal{M}[K](c) \approx \mathbf{1}_{\psi}}$ , that is:

$$\operatorname{argmin}_{\omega' \in \mathcal{D}(X)} D_{KL} \left( mn[K](\omega)|_{\mathcal{M}[K](c) \approx \mathbf{1}_{\psi}}, mn[K](\omega') \right) = c_{\omega}^{\dagger} \gg= flrn(\psi).$$

**Proof.** By Lemma 8.4 this minimal distribution is

$$flrn \gg= \left( mn[K](\omega)|_{\mathcal{M}[K](c) \approx \mathbf{1}_{\psi}} \right).$$

By Theorem 8.3 this equals Jeffrey’s update  $c_{\omega}^{\dagger} \gg= flrn(\psi)$ .  $\square$

## 9 Conclusions

The difference in outcomes of Pearl’s and Jeffrey’s update rules remains an intriguing topic. The paper does not offer the definitive story about when to use which rule, but it does enrich the field with several new ingredients (such as the different likelihoods and variational inference) and offers a wider perspective (including probabilistic programming). The main points that we have made explicit are that, when we learn from data,

- repeated application of Pearl’s rule, for each data point, corresponds to an update of the prior distribution, along a multinomial channel, see Theorem 7.3;
- Jeffrey’s rule is best understood as an update of all the multinomial draws from the prior and the formulation in Jeffrey’s rule is a best approximation of this update, see Theorem 8.5.

In these two update mechanisms there seems to be different perspectives at stake: the Pearlian posterior disease probability for an *individual* can be computed from a couple of tests, whereas the Jeffreyan posterior probability for a *population* requires many tests.



## References

- [1] K. Cho and B. Jacobs. Disintegration and Bayesian inversion via string diagrams. *Math. Struct. in Comp. Sci.*, 29(7):938–971, 2019.  
[doi:10.1017/s0960129518000488](https://doi.org/10.1017/s0960129518000488).
- [2] F. Clerc, F. Dahlqvist, V. Danos, and I. Garnier. Pointless learning. In J. Esparza and A. Murawski, editors, *Foundations of Software Science and Computation Structures*, number 10203 in Lect. Notes Comp. Sci., pages 355–369. Springer, Berlin, 2017.  
[doi:10.1007/978-3-662-54458-7\\_21](https://doi.org/10.1007/978-3-662-54458-7_21).
- [3] S. Dash and S. Staton. A monad for probabilistic point processes. In D. Spivak and J. Vicary, editors, *Applied Category Theory Conference*, Elect. Proc. in Theor. Comp. Sci., 2020.  
[doi:10.4204/EPTCS.333.2](https://doi.org/10.4204/EPTCS.333.2).
- [4] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. <http://dippl.org>, 2014. Accessed: 2021-8-3.
- [5] Noah D Goodman and Joshua B. Tenenbaum. Probabilistic Models of Cognition. <http://probmods.org>, 2016. Accessed: 2021-3-26.
- [6] B. Jacobs. The mathematics of changing one’s mind, via Jeffrey’s or via Pearl’s update rule. *Journ. of Artif. Intelligence Research*, 65:783–806, 2019.  
[doi:10.1613/jair.1.11349](https://doi.org/10.1613/jair.1.11349).
- [7] B. Jacobs. From multisets over distributions to distributions over multisets. In *Logic in Computer Science*. IEEE, Computer Science Press, 2021.  
[doi:10.1109/lics52264.2021.9470678](https://doi.org/10.1109/lics52264.2021.9470678).
- [8] B. Jacobs. Learning from what’s right and learning from what’s wrong. In A. Sokolova, editor, *Math. Found. of Programming Semantics*, number 351 in Elect. Proc. in Theor. Comp. Sci., pages 116–133, 2021.  
[doi:10.4204/EPTCS.351.8](https://doi.org/10.4204/EPTCS.351.8).
- [9] D. Koller and N. Friedman. *Probabilistic Graphical Models. Principles and Techniques*. MIT Press, Cambridge, MA, 2009.
- [10] D. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.
- [11] K. Murphy. *Machine Learning. A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.
- [12] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.  
[doi:10.1023/A:1007692713085](https://doi.org/10.1023/A:1007692713085).
- [13] Y. Zhang and N. Amin. Reasoning about “reasoning about reasoning”: Semantics and contextual equivalence for probabilistic programs with nested queries and recursion. *Proc. ACM Program. Lang.*, 6(POPL), jan 2022.  
[doi:10.1145/3498677](https://doi.org/10.1145/3498677).

## 10 Appendix: Overview of WebPPL

We give a brief overview of the WebPPL probabilistic programming language: WebPPL is based on a purely functional subset of Javascript, that is then extended with probabilistic primitives for sampling, conditioning and inferring posterior probability distributions. Its implementation is described in detail [4], and the language can be tried out a browser under [webppl.org](http://webppl.org).

In WebPPL, probability can be manipulated in the form of distribution objects (such as `Bernoulli({p: 0.3})`) and as samplers, that is functions which return random draws from a distribution, such as `bernoulli({p: 0.3})`. A distribution object `dist` can be sampled from using the command `sample(dist)`. Thus, the programs `bernoulli({p: 0.3})` and `sample(Bernoulli({p: 0.3}))` are equivalent. A shorthand for a biased coin flip is `flip(p)`. WebPPL comes with a library of common probability distributions, both discrete and continuous.

The command `condition(p)` expresses a boolean condition which must be met. The precise semantics of `sample` and `condition` will depend on the chosen inference algorithm, described below. Soft conditioning is

available using the syntax `observe(dist, observation)` but we don't need this in the current paper.

The command `Infer(fn)` takes a sampler `fn`, i.e. a higher-order function which represents a probabilistic experiment including conditions, and turns it into a distribution object which represents the exact or approximate posterior. The inference algorithm can be customized using a `method` argument. The default algorithm for discrete problems such those as in this paper is exact enumeration. That is `Infer` exhaustively tracks all random calls made within `fn`, discards those that violate the conditions, and computes the exact posterior. This strategy is only feasible for small problem instances. A typical call to `Infer` looks like

---

```
var posterior = Infer({method: 'enumerate'}, function() {
  var x = bernoulli({p: 0.3})
  var y = bernoulli({p: 0.9})
  condition(x==y)
  return x
})
```

---

The result is a distribution object `posterior`, which we can for example visualize using the command `viz(posterior)`. We can also sample from the posterior using `sample(posterior)`. Because `Infer` and `sample` are first-class operations in WebPPL, inference code can be nested without issue, expressing inference about inference. We use this pattern in our explanation of Jeffrey's update.

If the inference problems are no longer tractable using exact enumeration, approximate or sampling-based inference techniques can be used. The simplest is Monte Carlo simulation using rejection sampling, which will simply generate many execution traces of `fn()`, discard those whose conditions haven't been satisfied, and aggregate the results. More sophisticated algorithms are importance sampling, particle filters, variational inference and Markov Chain Monte Carlo. Internally, WebPPL is compiled into continuation-passing style which allows the `Infer` method a large amount of control over what happens at individual `sample` and `condition` commands [4].