Hakaru Language: Standard Library Implementation and Language Validation Testing

Justin Staples, Mahmoud Khattab, Nevin Mahilal and Aryan Sohrabi, supervised by Dr. Christopher Anand & Dr. Jacques Carette

{staplejw, khattm, mahilank, sohraa3, anandc, carette}@mcmaster.ca

Department of Computing and Software, McMaster University

Introduction

Applications

Real world phenomena often behave according to a probability distribution. By letting us sample from distributions, Hakaru enables us to model real world phenomena that include uncertainty. Hakaru can then perform transformations on these models to turn them into conditional probabilities that let us make inferences based on a priori knowledge. This, for example, is useful in machine learning applications where the goal is often to predict or infer information about the model based on the past outcomes.

Mathematical Background

The mathematical functions that we use to describe these probabilities are called probability distributions and the quantity that denotes the outcome of the experiment is called a random variable. A random variable can take on continuous or discrete values depending on the application. Continuous random variables are formally defined by a probability density function (PDF), which maps the outcomes of the random variable to real numbers that represent a probability. Similarly, a discrete random variable is described by a probability mass function (PMF). As an example, let us consider one of the most ubiquitous and naturally occurring distributions, the normal distribution. We can say that a random variable, X, is normally distributed with the following notation.

$$X \sim Normal(\mu, \sigma^2)$$

Here, we can see that the normal distribution is parametrized by μ, the population mean, and σ, the standard deviation. A plot the PDF is shown below. Remember that the PDF describes the liklihood that a value sampled from this distribution will equal x.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

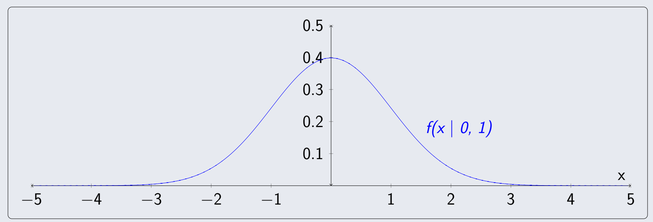

Figure 1: PDF of the normal distribution, with $\mu = 0$ and $\sigma = 1$

Language Guide

Hakaru is an experimental **probabilistic programming language**, which aims to simplify the way users implement statistical distributions.

- Niche application ⇒ small language with limited features.
- Running a hakaru program generates a stream of random numbers distributed according the statistical distribution modeled.
- Hakaru code can be compiled to C and Haskell so that models can be exported for use in larger applications.
- Terminology: implementation of statistical distribution == probabilistic model == model.

'hk-maple' is a provided inference algorithm that uses Maple to perform algebraic transformations on hakaru programs. In its default mode (Simplify), it returns an equivalent hakaru program with greater sampling efficiency. In applications requiring billions of samples (e.g. machine learning) this has the potential to save significant processing time!

Key Concepts

This section is dedicated to explaining some key ideas and terminology for our project.

Motivation

Objectives

- Increase language accessibility: add new language features such as primivite mathematical functions (e.g. log, choose), new distributions, error handling, etc.
- Test language validity: use relationships between distributions to test the validity of program transformations like 'hk-maple'.

Endeavours

- Standard library development: the UDR is used to branch out and implement new distributions.
- Syntax-highlighting-for-hakaru package for Sublime Text: this package makes developing Hakaru code much more convenient and streamlined (Mahmoud).
- Test case writing: we focus mainly on comprehensively testing only a select few distributions (e.g. chi-square).

Standard Library Development

In order to make Hakaru more accessible to users, the standard library should implement commonly used probabilistic models. To this this end, we have implemented the distributions described in the UDR, a portion of which can be seen below. In general, we have used the following principles to guide the development of the standard library:

- Whenever possible, implement distributions as transformations on pre-existing models.
- Reference the UDR chart for possible ways to implement each distribution.
- In the case of multiple possible implementations, take the shortest possible path from a primitive distribution on the UDR.

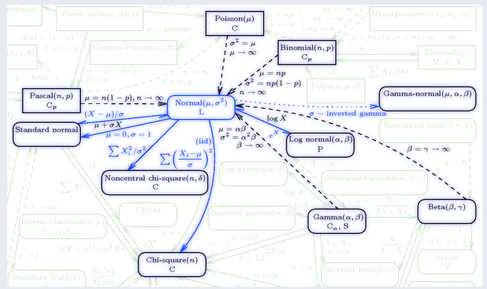

Figure 2: A snapshot of the UDR shows how the normal distribution can be transformed into a multitude of other distributions.

Hakaru functions representing probabilistic models (i.e. functions with a `measure(<type>)` return type) cannot be used as operands in normal mathematical operations. This is to say we cannot transform models directly (i.e. we cant do algebraic transformation on the PDF). The only operator we can use on a model is bind (<~), to pull a sample from it. However, we are able to transform samples however we like. Therefore, we are interested in implementing transformations of the following form.

$$R(p, q) \Rightarrow X \sim A(p) \Rightarrow f(X) \sim B(q)$$

In the equation above, p is a set of values parameterizing the distribution A, q is a set of values parameterizing the distribution B, $R(p, q)$ is a set of relationships between p and q that must be satisfied, and f is a function that applies a transformation to a sample. We can expand this definition to include transformations defined in terms of an aggregation of multiple independent samples. For example, the standard chi-square distribution is defined as the sum of the squares of n normal random variables (see Figure 3).

```
def chiSq(means array(real), stdevs array(prob) ):
    q <~ plate _ of size(means): normal(means[_],stdevs[_])
```

Hakaru also lends itself very well to Bayesian transformations where we want to pull a sample from a distribution which is parameterized with a value sampled from another distribution. These transformations take the following form. Here, we are saying that if X is parameterized by p, then the distribution C, which is parameterized by p and q, is equal to another distribution, B, which is parameterized by q and X. The gamma-poisson distribution can be described by such a transformation (see Figure 4).

$$X \sim A(p) \Rightarrow Y \sim B(q, X) = C(p, q)$$

```
def gammaPoisson(shape prob, scale prob) measure(nat):
    mu <~ gamma(shape, scale)
    X <~ poisson(mu)
    return X
```

Figure 4: Our implementation of the gamma-poisson distribution.

Some distributions on the UDR are unreachable from Hakaru's primitive distributions using transformations like the 2 described above. In these cases, we have to implement the model in terms of the PMF for a discrete distribution or in terms of the PDF for a continuous distributions.

Testing Relationships Between Distributions

Recall that when developing the standard library, the UDR chart often implied multiple possible implementations for a given distribution. Naturally, we would expect alternative implementations for the same distribution to result in equivalent probabilistic models. This line of thought is the basis for the kinds of test cases we have implemented. More generally, we have the following hypothesis.

Assume we know a proven relationship between 2 statistical distributions, A and B, which allows us to transform A and B into distributions that are equivalent to each other. Further assume this transformation takes on one of the forms discussed in the Standard Library Development section. We hypothesize that by applying the appropriate transformations to implementations of A and B, we can create two Hakaru programs whose 'hk-maple' outputs will be equivalent to each other. Test cases that prove our hypothesis true indicate the validity of the Hakaru language implementation. Test cases that prove our hypothesis false indicate an underlying bug in the language definition which is to be passed back to the language developers.

Conclusions & Future Work

Conclusions & Future Work

References

References