Uncertain<T>: Uncertain data as a first-order type

James Bornholt *
Australian National University
u4842199@anu.edu.au

1. Introduction

Developers are increasingly working with estimated data to sense and reason about the physical world. For example, GPS sensors estimate a user's location in space, and software benchmarking estimates the performance of different configurations. Uncertainty, in the form of random error, is the difference between estimation and fact.

To program correctly with estimated data, developers must account for this uncertainty. Failing to do so introduces *uncertainty bugs*: unexpected or incorrect output due to random error. For example, an application using GPS estimates to calculate speed may report the user walking at 59 mph.

Current programming language abstractions for uncertainty are insufficient. On one hand, simplistic abstractions such as GPS APIs are too high-level and use deterministic types (floats, integers, etc.) to represent stochastic data. This impedance mismatch leads to incorrect computation on estimated data. On the other hand, probabilistic programming languages require expertise in statistics to use, forcing developers to rephrase even simple problems in complex ways.

We introduce the *uncertain type*, a programming language abstraction that captures probability distributions for estimated values. By overloading familiar operators and offering tools to work with distributions, the uncertain type allows programmers to work correctly and efficiently with estimated data. The uncertain type propagates uncertainty through calculations, forces developers to ask the right questions of their data, and allows advanced developers to improve estimates by using background knowledge.

2. Motivation

We consider applications that compute with GPS location estimates as a motivating example. These applications often experience uncertainty bugs, which fall into three categories.

Using an estimate as fact. GPS sensors estimate location. Treating these estimates as facts produces incorrect results, like locating the user in an ocean while driving.

Computation compounds errors. Because GPS sensors estimate location, subsequent calculations like calculating speed from two locations are also estimates. These computations compound uncertainty, resulting in absurd speeds even if the location estimates are very accurate.

Inference. Estimated data cannot answer deterministic inferences such as "are you speeding?". Trying to do so leads to false positives – answering yes based on random error when the true answer is no.

Fixing these bugs requires accounting for uncertainty. Since general purpose programming languages are deterministic, most developers simply ignore uncertainty. We find only one of the top 100 Windows Phone applications reasons about uncertainty in GPS readings.

3. Background

Random variables model estimation processes using probability distributions, which assign likelihoods to each possible value of the variable. Jaroszewicz and Korzeń [1] present the PaCAL software library for computing with probability distributions, and Infer.NET [2] is a framework for performing Bayesian inference using probability distributions in a general programming language. Experts can use tools like these to correctly account for uncertainty. But PaCAL requires developers to explicitly specify the distributions and operations they are performing, and can only handle distributions that have closed forms. Infer.NET requires a strong grasp of Bayesian statistics to phrase problems in the language of graphical models. Our uncertain type achieves the same goal of correctly addressing uncertainty but without the complex background knowledge.

4. Uncertainty as a first-order type

We propose a new generic data type, $\mathit{Uncertain}\langle T \rangle$, to capture and manipulate uncertainty. This *uncertain type* is an abstraction over distributions. For example, the uncertain type encapsulates GPS coordinates as type $\mathit{Uncertain}\langle \mathit{GeoCoordinate} \rangle$, a distribution over coordinates.

4.1 Defining distributions

The distribution to use for a particular estimate is necessarily domain-specific. In many cases, library developers will provide distributions. Third party developers consume these distributions. For example, we have developed a GPS library

* This work was performed as a Microsoft Research intern, supervised by Kathryn McKinley and Todd Mytkowicz.

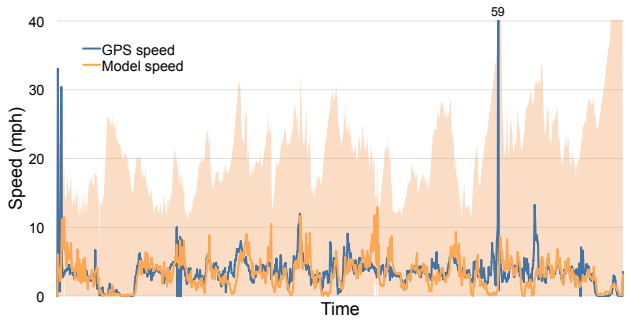

Figure 1: The uncertain type easily incorporates additional models and data into estimation processes. Above a physics model improves the GPS estimation of speed. The shaded area is the 95% confidence interval for the modelled speed.

with distributions based on models of GPS error and ported applications to use it. For more advanced problems, developers will define their own distributions, based on theory or empirical results.

4.2 Computing with random variables

The uncertain type lifts the normal arithmetic operations on type T to work on distributions over T, allowing developers to work with distributions in a natural way. But if two random variables are dependent, the developer must define the joint probability distribution of the two variables for computations to be correct.

4.3 Inference using random variables

Developers commonly use variables in conditionals to make decisions. Distributions complicate this style of inference, because there is more than one way to interpret the question $A < B$ for distributions A and B.

Confidence intervals. One approach is to use confidence intervals. For example, given a random variable *Speed* we may ask "are you speeding?", that is, if $Speed > 60$. But we cannot answer this deterministic question with stochastic data. What we can ask is "are you speeding with 99% confidence?". We only answer yes if the evidence is very strong, thus controlling false positives. But this creates a ternary logic, because the evidence may not be sufficient to say either $Speed > 60$ or $Speed \leq 60$.

Expected values. We can also interpret conditionals with expected values. If we have two random variables *DistanceToA* and *DistanceToB*, we may ask if $DistanceToA > DistanceToB$. We can interpret this as $\mathrm{E}[DistanceToA] > \mathrm{E}[DistanceToB]$. Since the expected value is a real number, this defines a binary logic, avoiding the unexpected ternary logic of confidence intervals.

By providing both interpretations, developers can choose the right approach for their particular problem.

5. Using the uncertain type

We demonstrate the uncertain type on GPS data. The first step is to define the distribution for a GPS sample, which we put in a library. Based on prior work, we model GPS error based on the Rayleigh distribution [3].

We build a simple application that estimates walking speed based on GPS samples. The application obtains two GPS samples, one second apart, and calculates speed based on the formula $Speed = \frac{Distance}{Time}$. We use data from a 15 min walk around the local neighborhood.

5.1 Identifying absurd values

Without considering uncertainty, this application produces absurd results as shown by the blue line in Figure 1. At one point the user walks at 59 mph, and walks for 30 seconds at over 7 mph (a running pace). With the uncertain type, uncertainty propagates through the calculation of speed. The uncertain type shows that the 95% confidence interval for speed reaches a peak of $\pm$300 mph. The absurd values are therefore simply a result of random noise.

5.2 Producing improved estimates

A key advantage of the uncertain type is that it makes it easy to produce better estimates by incorporating prior data. In the walking application, the developer may provide a prior distribution over speed that reflects normal walking speeds. The developer then uses Bayesian inference to incorporate this prior with GPS data.

The dark orange line in Figure 1 shows that adding such a physics model removes the absurd spikes from the data, and the 95% confidence interval (the shaded area) is smaller.

6. Conclusion

Applications are increasingly using uncertain data. Other areas of science require proper statistical techniques when working with estimates. We argue that programmers should too. The uncertain type provides an abstraction that allows developers to work correctly with uncertain data. By capturing uncertainty as a first-order type, we can manage and hide many of the advanced details of statistics from most developers, and still expose the details to those who want them. Programs using the uncertain type are therefore more powerful and more correct.

References

[1] S. Jaroszewicz and M. Korzeń. Arithmetic operations on independent random variables: A numerical approach. *SIAM Journal on Scientific Computing*, 34:A1241–A1265, 2012.

[2] T. Minka, J. Winn, J. Guiver, and D. Knowles. Infer.NET 2.5, 2012. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

[3] F. van Diggelen. GNSS Accuracy: Lies, Damn Lies, and Statistics. *GPS World*, 18(1):26–32, 2007.