Uncertain<T>
A First-Order Type for Uncertain Data

James Bornholt  Australian National University
Todd Mytkowicz  Microsoft Research
Kathryn S. McKinley  Microsoft Research
Approximate computing

approximate edge detection
Machine learning

inputs

hidden units

outputs

\(x_D\)

\(x_1\)

\(x_0\)

\(z_0\)

\(z_1\)

\(z_M\)

\(w^{(1)}_{MD}\)

\(w^{(2)}_{KM}\)

\(w_{10}\)

\(y_K\)

\(y_1\)
Edge detection
Edge detection
Edge detection
Edge detection

\[ \text{Sobel}(\rho) \]
Edge detection

Sobel(p) → 0.4940
Edge detection

Sobel$(p)$ = 0.4940
Edge detection

Sobel(p) $\rightarrow$ 0.4940
Approximate edge detection

3.4% average error
Approximate edge detection

What is the gradient at pixel $p$?

$\text{Sobel}(p)$

3.4% average training error
What is the gradient at pixel $p$?

$Sobel(p)$

3.4% average training error

Is there an edge at pixel $p$?

```markdown
if (Sobel(p) > 0.1) 
  EdgeFound();
```
Approximate edge detection

What is the gradient at pixel $p$?

$\text{Sobel}(p)$

3.4% average training error

Is there an edge at pixel $p$?

$\text{if } (\text{Sobel}(p) > 0.1) \text{ EdgeFound}();$

36% false positives on the same data!
Approximate edge detection

What is the gradient at pixel $p$?

$\text{Sobel}(p)$

3.4% average training error

Is there an edge at pixel $p$?

\[ \text{if } (\text{Sobel}(p) > 0.1) \text{ EdgeFound}(); \]

36% false positives on the same data!

Computation compounds uncertainty!
GeoCoordinate Location;

double Grad = Sobel(p);
Uncertain<GeoCoordinate> Location;

Uncertain<double> Grad = Sobel(p);
Uncertain<T> is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.
Uncertain<GeoCoordinate> LastLoc =
    GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
    GPS.GetLocation();
Uncertain<GeoCoordinate> LastLoc =
    GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
    GPS.GetLocation();

Uncertain<double> Dist =
    GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;
Uncertain<GeoCoordinate> LastLoc =
    GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
    GPS.GetLocation();

Uncertain<double> Dist =
    GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");
Uncertain<GeoCoordinate> LastLoc =
    GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
    GPS.GetLocation();

Uncertain<double> Dist =
    GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");

print("Your speed: " + Speed.E());
Uncertain<GeoCoordinate> LastLoc =
            GPS.GetLocation();
Sleep(5);
Uncertain<GeoCoordinate> Loc =
            GPS.GetLocation();

Uncertain<double> Dist =
            GPS.Distance(Loc, LastLoc);
Uncertain<double> Speed = Dist / 5;

if (Speed > 4) print("Great job!");

print("Your speed: " + Speed.E());
Probabilistic programming

BUGS, Church, Infer.NET, …
Probabilistic programming

BUGS, Church, Infer.NET, …
Probabilistic programming

BUGS, Church, Infer.NET, …
Probabilistic programming

BUGS, Church, Infer.NET, ...

Uncertain<T> helps developers without statistics PhDs.
Uncertain<GeoCoordinate> LastLoc = GPS.GetLocation();

A variable of type Uncertain<T> is a random variable, represented by a distribution.
Uncertain<GeoCoordinate> LastLoc = GPS.GetLocation();

A variable of type Uncertain<T> is a random variable, represented by a distribution.

“We define accuracy as the radius of 68% confidence [of a] normal distribution.”
—Android
Sampling functions return random samples.
Sampling functions return random samples.

✓ Simple computations.
Sampling functions return random samples.

✓ Simple computations.

✓ Represent many distributions.
Sampling functions return random samples.

✓ Simple computations.

✓ Represent many distributions.

✗ Sampling is approximate.

(Later: how Uncertain<T> learned to love approximation, and you can too)
Uncertain<\texttt{double}> \texttt{Speed} = \texttt{Dist} / 5;
Uncertain<\texttt{double}> \texttt{Speed} = \texttt{Dist} / 5;

Or more generally, $Z = X + Y$, if $X$ and $Y$ are distributions.
Uncertain\texttt{<double> } Speed = \textit{Dist} / 5;

Or more generally, \( Z = X + Y \), if \( X \) and \( Y \) are distributions.
Uncertain<double> Speed = Dist / 5;

Or more generally, $Z = X + Y$, if $X$ and $Y$ are distributions.
Uncertain<
\texttt{double}> \texttt{Speed} = \texttt{Dist} / 5;

Or more generally, $Z = X + Y$, if $X$ and $Y$ are distributions.

If $x$ is a sample of $X$
and $y$ is a sample of $Y$
then $x+y$ is a sample of $X+Y$*

* if $X$ and $Y$ are independent
Bayesian network representation:

D = A / B
E = D – C
Bayesian network representation:

\[ D = A / B \]
\[ E = D - C \]

Sampling function for \( E \) recursively samples children.
If \( x \) is a sample of \( X \)
and \( y \) is a sample of \( Y \)
then \( x+y \) is a sample of \( X+Y \) *

* Only if \( X \) and \( Y \) are independent.
If \( x \) is a sample of \( X \) and \( y \) is a sample of \( Y \) then \( x+y \) is a sample of \( X+Y \) *

* Only if \( X \) and \( Y \) are independent.

\[
A = X + Y \quad (X,Y \text{ independent})
\]

\[
B = A + X
\]
If \( x \) is a sample of \( X \)
and \( y \) is a sample of \( Y \)
then \( x+y \) is a sample of \( X+Y \)^∗

^∗ Only if \( X \) and \( Y \) are independent.

\[
\begin{align*}
A &= X + Y \quad (X,Y \text{ independent}) \\
B &= A + X
\end{align*}
\]

\( A \) and \( B \) depend on \( X \) – not independent!
If \( x \) is a sample of \( X \) and \( y \) is a sample of \( Y \), then \( x+y \) is a sample of \( X+Y \) *

* Only if \( X \) and \( Y \) are independent.

\[
A = X + Y \quad (X,Y \text{ independent})
\]
\[
B = A + X
\]

\( A \) and \( B \) depend on \( X \) – not independent!
If \( x \) is a sample of \( X \) 
and \( y \) is a sample of \( Y \) 
then \( x+y \) is a sample of \( X+Y \) * 

* Only if \( X \) and \( Y \) are independent.

\[
A = X + Y \quad (X,Y \text{ independent}) \\
B = A + X
\]

\( A \) and \( B \) depend on \( X \) – not independent!
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");
if (Speed > 4) print("Great job!");

More likely than not that Speed > 4?
if (Speed > 4) print("Great job!");

More likely than not that Speed > 4?
if (Speed > 4).Pr(0.9) print("Great job!");
if (Speed > 4).Pr(0.9) print("Great job!");

At least 90\% likely that Speed > 4?
if (Speed > 4).Pr(0.9) print("Great job!");

Pr[Speed > 4] > 0.9
if (Speed > 4).Pr(0.9) print("Great job!");

Pr[Speed > 4] > 0.9

approximate!
if (Speed > 4).Pr(0.9) print("Great job!");

Pr[Speed > 4] > 0.9
approximate!
```python
if (Speed > 4).Pr(0.9) print("Great job!");
```

**null hypothesis**  \[ H_0: \Pr[\text{Speed} > 4] \leq 0.9 \]

\[
\Pr[\text{Speed} > 4] > 0.9 \quad \text{approximate!}
\]
if (Speed > 4).Pr(0.9) print("Great job!");

null hypothesis \( H_0: \Pr[\text{Speed} > 4] \leq 0.9 \)

alternate hypothesis \( H_A: \Pr[\text{Speed} > 4] > 0.9 \)
if (Speed > 4).Pr(0.9) print("Great job!");

null hypothesis  \( H_0: \Pr[\text{Speed} > 4] \leq 0.9 \)
alternate hypothesis  \( H_A: \Pr[\text{Speed} > 4] > 0.9 \)

How many samples?
if (Speed > 4).Pr(0.9) print("Great job!");

null hypothesis \( H_0: \operatorname{Pr}[\text{Speed} > 4] \leq 0.9 \)

alternate hypothesis \( H_A: \operatorname{Pr}[\text{Speed} > 4] > 0.9 \)

How many samples? Too many = too slow  
Too few = too noisy
if (Speed > 4).Pr(0.9) print("Great job!");

null hypothesis  $H_0$: $\Pr[\text{Speed} > 4] \leq 0.9$
alternate hypothesis  $H_A$: $\Pr[\text{Speed} > 4] > 0.9$

approximate!

How many samples?  Too many = too slow
Too few = too noisy

Sequential sampling: sample size depends on progress
Incorporate domain knowledge: “I’m on a road”
Incorporate domain knowledge: “I’m on a road”
Incorporate domain knowledge: “I’m on a road”

\[
\Pr[H|E] = \frac{\Pr[E|H]\Pr[H]}{\Pr[E]} \quad \text{posterior likelihood}
\]
Incorporate domain knowledge: “I’m on a road”

\[
Pr[H|E] = \frac{Pr[E|H] \cdot Pr[H]}{Pr[E]}
\]
Case studies

Smartphone GPS sensors

Noisy Game of Life (see the paper)

Neural networks/approximate computing
What is the gradient at pixel $p$?

$\text{Sobel}(p)$  

3.4% average error

Is there an edge at pixel $p$?

$\text{Sobel}(p) > 0.1$  

36% false positives!
single input
single input
single input → single output
single input  \rightarrow \text{approximate output}
Is there an edge at pixel \( p \)?

\[ \text{Sobel}(p) > 0.1 \] 36\% false positives!
Is there an edge at pixel \( p \)?

\[
\text{Sobel}(p) > 0.1 \quad \text{36\% false positives!}
\]
Is there an edge at pixel $p$?

$\text{Sobel}(p) > 0.1$  

36% false positives!
Is there an edge at pixel $p$?

$$\text{Sobel}(p) > 0.1$$

36% false positives!
Is there an edge at pixel $p$?

$\text{Sobel}(p) > 0.1$  

36% false positives!

$\Pr[\text{Sobel}(p) > 0.1] = 70\%$
Conditional threshold

\[ \text{Precision/Recall (\%)} \]

Pr[Sobel(p) > 0.1] > \(\alpha\)
Naive Precision: 60%
Naive Recall: 100%

Conditional threshold $\alpha$

$\Pr[\text{Sobel}(p) > 0.1] > \alpha$
Higher precision = fewer false positives

Pr[Sobel(p) > 0.1] > \( \alpha \)
Higher recall
= fewer false negatives

Higher precision
= fewer false positives
Uncertain $T$

Higher precision = fewer false positives

Higher recall = fewer false negatives

$\text{Pr}[\text{Sobel}(p) > 0.1] > \alpha$

Naive Precision

Naive Recall

Precision/Recall (%)
Uncertain\texttt{<T>} is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.
Uncertain\texttt{<T>} is an uncertain type abstraction.

It encourages non-expert developers to explicitly reason about uncertainty.

Thank you!