

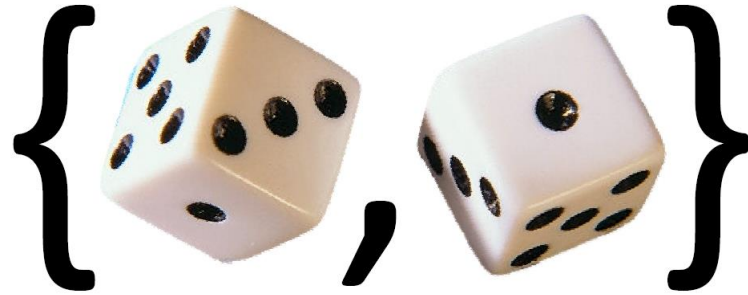
Probabilistic programming the future of Machine Learning?

John Winn

Ph.D. Summer School, July 2014

Outline

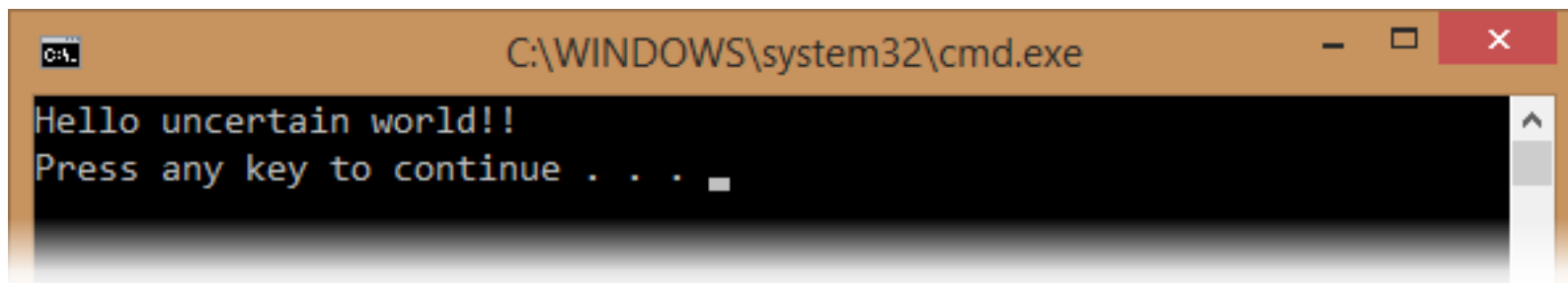
- ▶ What is probabilistic programming?
- ▶ Why is it important for machine learning?
- ▶ Example: interpreting user behaviour
- ▶ Example: parsing text
- ▶ Getting probabilistic programs to run quickly



What is probabilistic programming?

Hello uncertain world (C#)

```
// Create two strings
string a = 50%:"Hello uncertain" 50%:"Hello"
string b = 50%:"world" 50%:"uncertain world"
// Format strings together into a new string
string c = a + " " + b + "!!";
// Write it out
Console.WriteLine(c);
```



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window content displays the output of the C# code: 'Hello uncertain world!!' followed by 'Press any key to continue . . .'. A small cursor is visible at the end of the second line.

In probabilistic C# (Csoft)

```
// Create two strings
string a = Random(Strings.Uniform());
string b = Random(Strings.Uniform());
// Format strings together into a new string
string c = a + " " + b + "!!";
```

Uniform over strings

Random variables

- Normal variables have a fixed single value:

```
int length=6,  
bool visible=true.
```

- Random variables have uncertain value specified by a probability distribution:

```
int length = Random(Uniform(0,10));  
bool visible = Random(Bernoulli(0.8));
```

- **Random** means 'is distributed as'.

Observed data (constraints)

- We can define observations on random variables:
 - `Observe (visible==true)`
 - `Observe (length==4)`
 - `Observe (length>0)`
 - `Observe (i==j)`
- `Observe (b)` means 'we observe b to be true'.

Inference

- **Infer** gives the posterior distribution of one or more random variables.
- Example:

```
int i = Random(Uniform(1,10));  
bool b = (i*i>50);  
var bdist = Infer(b); //Bernoulli(0.3)
```
- Output of **Infer** is always *deterministic* even when input is *random*.

Example: linear regression (x to y)

```
// Unknown line parameters
double a = Random(Gaussian(0,1000));
double b = Random(Gaussian(0,1000));
// Loop over points
for (int i=0; i < x.Length; i++) {
    // Equation of line
    double clean_y = a * x[i] + b;
    // To match the data, we must add some noise.
    Observe(y[i] == Random(Gaussian(clean_y,1)));
}

var adist = Infer(a); // Learn slope
var bdist = Infer(b); // Learn intercept
```

Semantics: sampling interpretation

Imagine running the program (very) many times:

- ▶ **Random** (d) *samples* from the distribution d .
- ▶ **Observe** (b) *discards* the run if b is false.
- ▶ **Infer** (x) *stores* the value of x .
 - ▶ If enough x 's have been stored, returns their distribution.
 - ▶ Otherwise stops the run and starts a new run.

Example: linear regression (x to y)

```
// Unknown line parameters
double a = Random(Gaussian(0,1000));
double b = Random(Gaussian(0,1000));
// Loop over points
for (int i=0; i < x.Length; i++) {
    // Equation of line
    double clean_y = a * x[i] + b;
    // To match the data, we must add some noise.
    Observe(y[i] == Random(Gaussian(clean_y,1)));
}
var adist = Infer(a); // Learn slope
var bdist = Infer(b); // Learn intercept
```

Running probabilistic programs

Probabilistic programs can be:

- ▶ Interpreted on-the-fly e.g. Church, Dimple
- ▶ Compiled to generate algorithm code e.g. Infer.NET, Stan, OpenBUGS

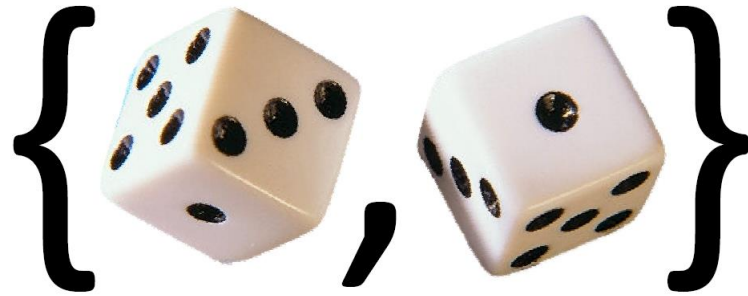
Our Infer.NET compiler:

- ▶ Runs deterministic message passing
e.g. Expectation Propagation, Variational Message Passing
- ▶ Supports a wide range of probabilistic programs
- ▶ Scales up to very large data sets (GB, TB or more)
- ▶ Used in hundreds of applications (internal and external)
- ▶ Now 10 years old!

<http://research.microsoft.com/infernet>



infer.net



Why is probabilistic programming important?

Case study: How Good Are You at Halo?

Xbox Live

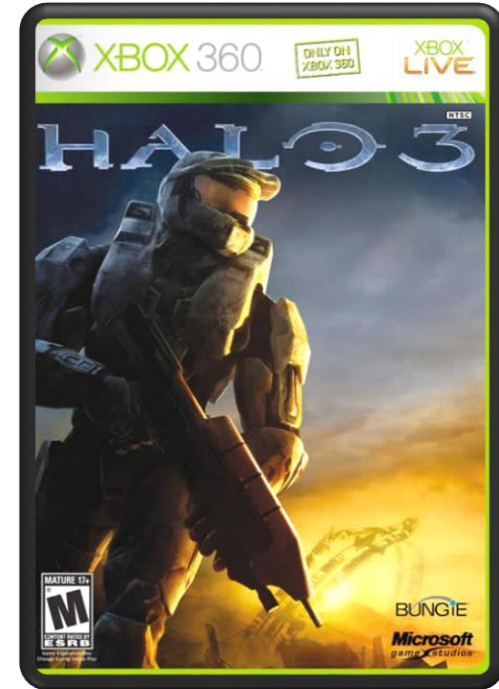
- > 12 million players
- > 2 million matches per day
- > 2 billion hours of gameplay

The Challenge

- > Tracking how good each player is to match players of similar skill.

TrueSkill™

- > Months of work, algorithm was 100s of lines of code.

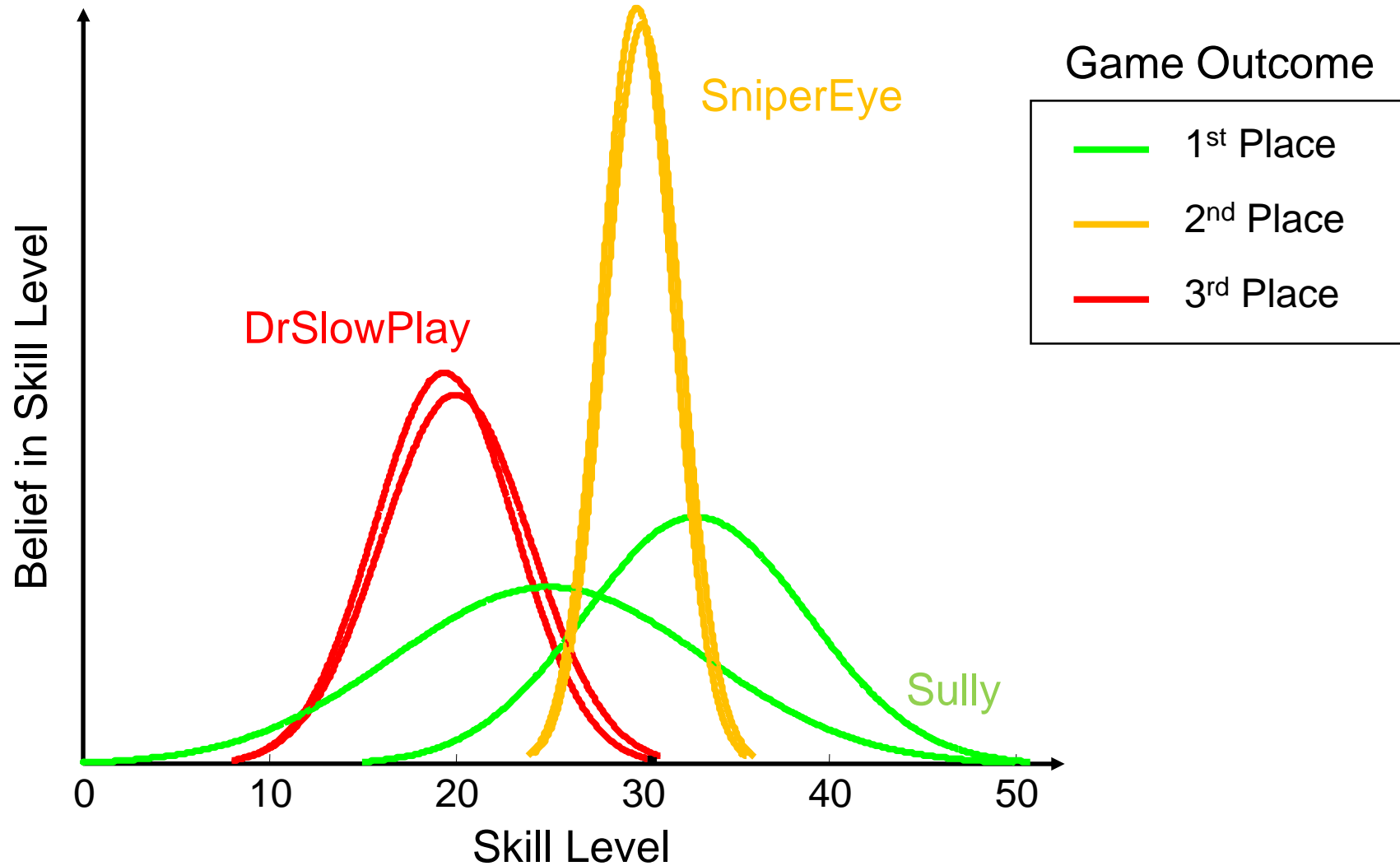


Old Estimates
of Players' Skills

Gamertag	Score
Sully	25
SniperEye	22
DrSlowPlay	17

New Estimates
of Players' Skills

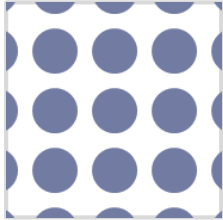
Inferring Skills



TrueSkill™ in Probabilistic C#

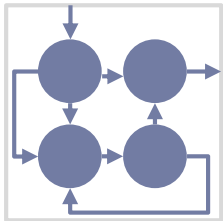
```
double[] skill=new double[nPlayers];
double[] performance=new double[nPlayers];
for (int j = 0; j<nPlayers; j++) {
    skill[j] = Random(Gaussian(means[j],vars[j]));
    double noise = Random(Gaussian(0, beta));
    performance[j] = skill[j] + noise;
    if (j>0) Observe(performance[j-1] > performance[j]);
}
return Infer(skill);
```


Machine learning is becoming...



More **common**

Ever more places where machine learning plays a key role.



More **complex**

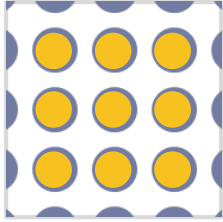
Used to address increasingly challenging problems.



More **large scale**

Larger datasets, coming from more diverse sources.

...more common



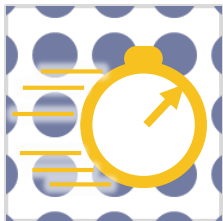
Probabilistic programming requires (much) less expertise.

» more people can use machine learning



Probabilistic programs are more transparent.

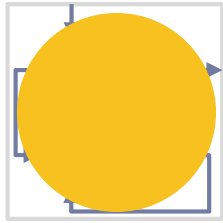
» experts can provide support more easily



Probabilistic programs are short.

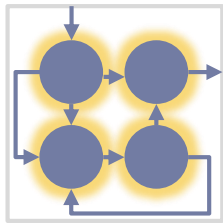
» quicker to write and experiment (play) with

...more complex



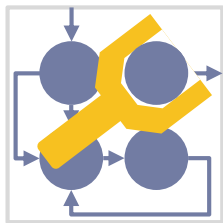
Probabilistic programming hides complexity of Bayesian methods.

» helps break the 'complexity barrier'



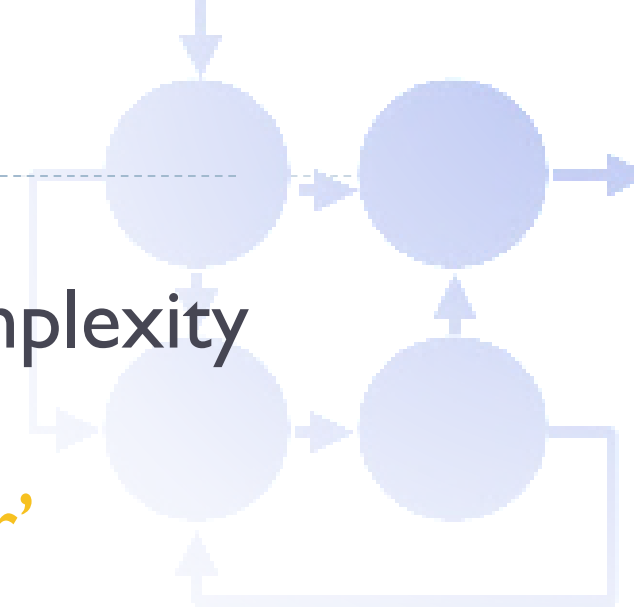
Re-usable inference engines.

» thoroughly tested, fewer bugs

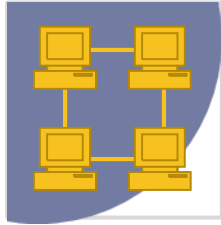


Probabilistic development tools.

» help to manage complexity

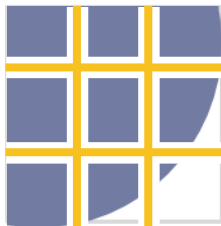


...more large scale



Probabilistic programs say *what* problem to solve, not *how* to solve it.

» multiple possible execution back ends
(Mobile, CPU, GPU, Cluster, Cloud)



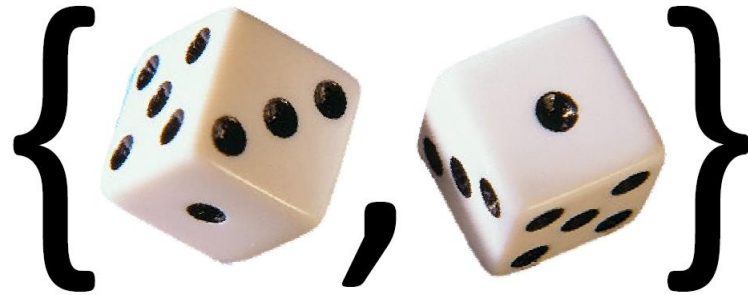
Probabilistic programs allow auto-parallelisation.

» Exploits the structure of the program



Probabilistic programs allow online inference.

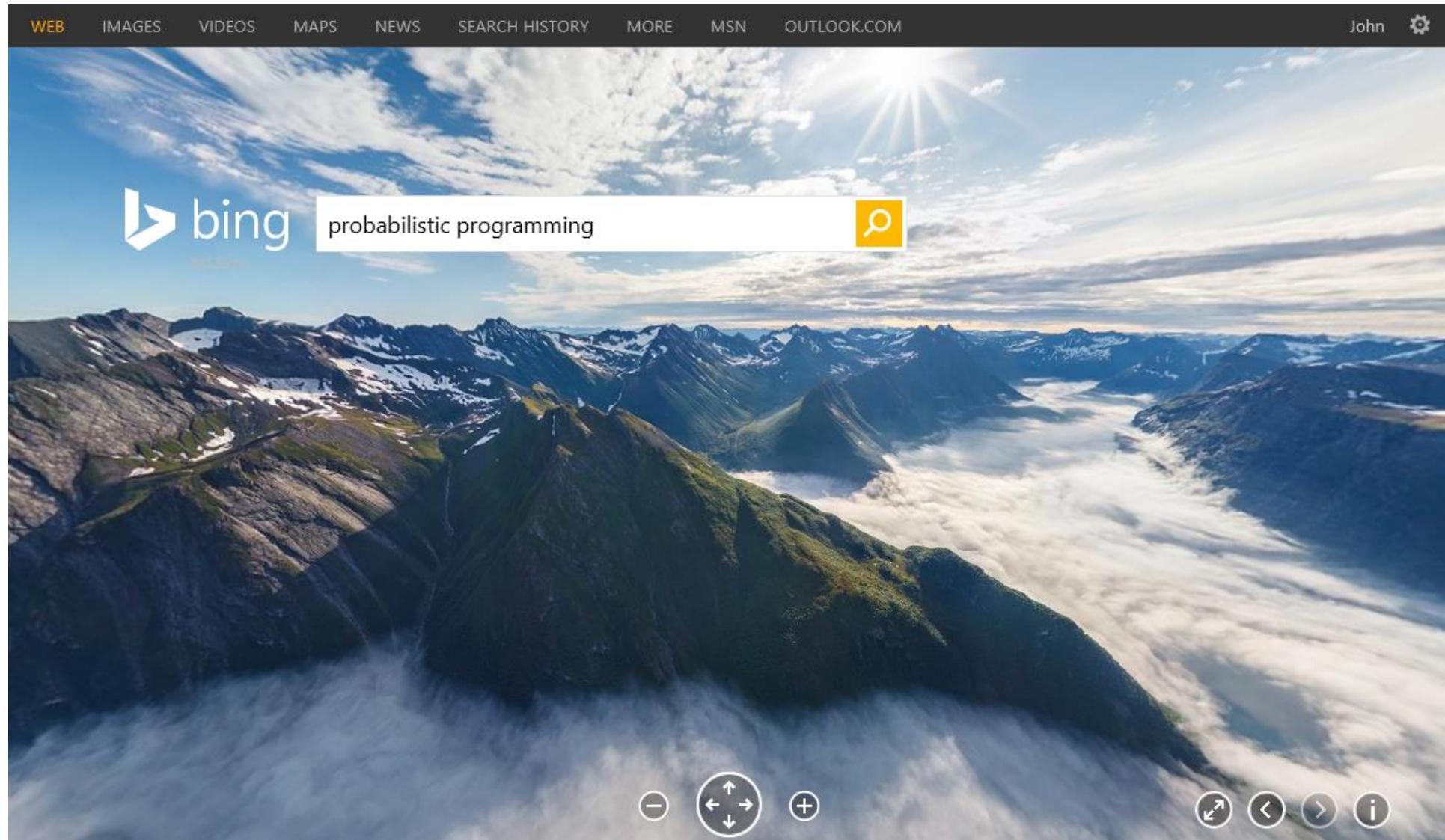
» Batch & online versions from one program



Probabilistic programming for interpreting user behaviour

With Tom Minka, John Guiver

Example : Search Log Analysis



The Click Log

WEB IMAGES VIDEOS MAPS NEWS MORE

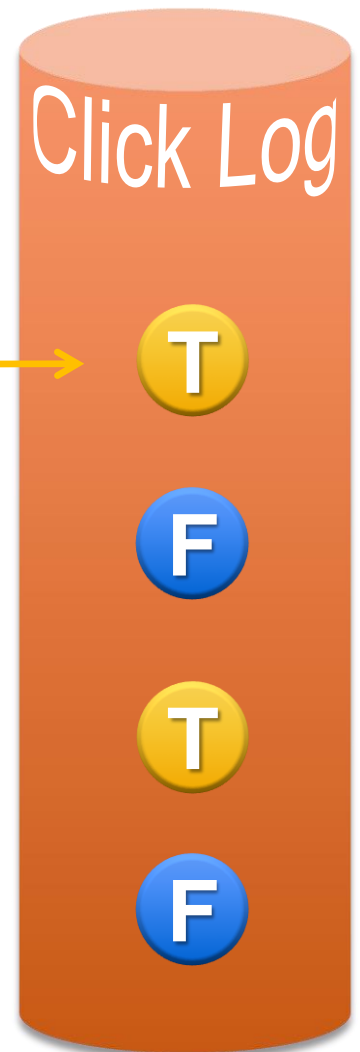
bing probabilistic programming

6,540,000 RESULTS Narrow by language Narrow by region

- Probabilistic Programming**
[probabilistic-programming.org](#)
PROBABILISTIC PROGRAMMING.org. This website serves as a repository of links and information about probabilistic programming languages, including both academic ...
- What is probabilistic programming? - O'Reilly Radar**
[radar.oreilly.com/2013/04/probabilistic-programming.html](#)
16/04/2013 · Probabilistic programming languages are in the spotlight. This is due to the announcement of a new DARPA program to support their fundamental research.
- programming Probabilistic**
[research.microsoft.com/en-us/um/...probabilisticprogramming-slides.pdf](#) · PDF file
Goals of Probabilistic Programming Make it easier to do probabilistic inference in custom models If you can write the model as a program, you can do inference on it
- Probabilistic programming language - Wikipedia, the free ...**
[en.wikipedia.org/wiki/Probabilistic_relational_programming_language](#)
A probabilistic programming language (PPL) is a programming language specially designed to describe and infer with probabilistic models. PPLs often extend from a ...
Probabilistic ... · List of ...

Related searches

- Church Wikipedia
- Wiki Churches
- Edu Church



Programming a user (code snippets)

```
appeal[d] = Random(Beta(1,1)); // Unknown appeal
relevance[d] = Random(Beta(1,1)); // Unknown relevance
    :
// For each user/document
// Should user examine the next search result?
examine[d] = examine[d - 1] &
    (((!click[d - 1]) & nextIfNotClick) |
    (click[d - 1] & nextIfClick));
// User clicks if they examined result and it appealed to them
click[d] = examine[d] & Random(Bernoulli(appeal[d]));
// User finds relevant page if they clicked and was relevant
isRelevant[d] = click[d] & Random(Bernoulli(relevance[d]));
    :
```

Attaching data to the program



6,540,000 RESULTS Narrow by language ▾ Narrow by region ▾

1 [Probabilistic Programming](#)

[probabilistic-programming.org](#) ▾
PROBABILISTIC-PROGRAMMING.org. This website serves as a repository of links and information about probabilistic programming languages, including both academic ...

2 [What is probabilistic programming? - O'Reilly Radar](#)

[radar.oreilly.com/2013/04/probabilistic-programming.html](#) ▾
16/04/2013 · Probabilistic programming languages are in the spotlight. This is due to the announcement of a new DARPA program to support their fundamental research.

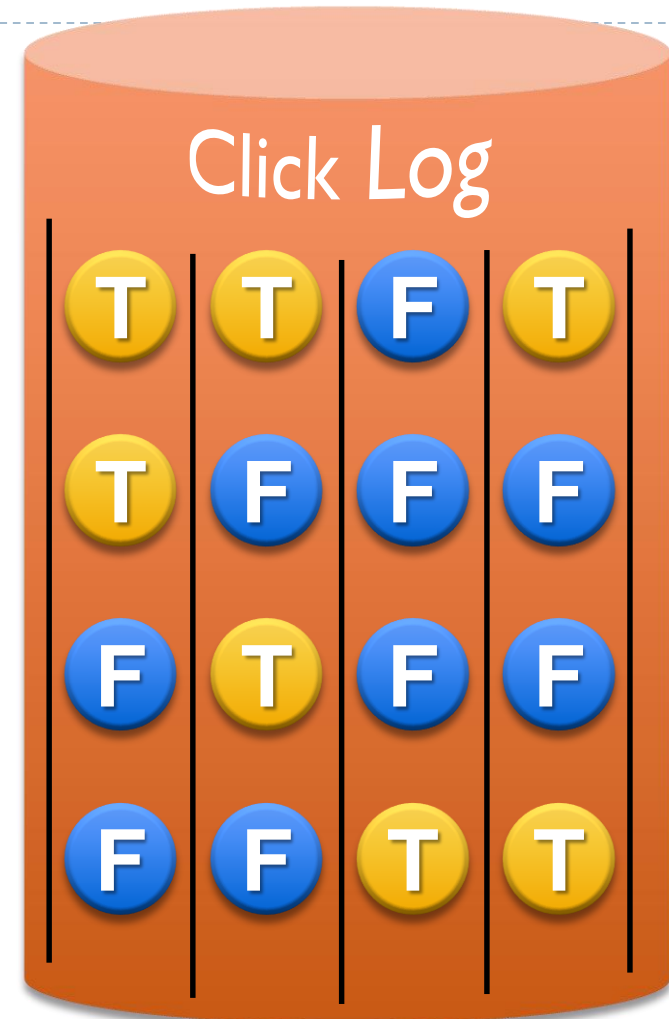
3 [programming Probabilistic](#)

[research.microsoft.com/en-us/um/...probabilisticprogramming-slides.pdf](#) · PDF file
Goals of Probabilistic Programming Make it easier to do probabilistic inference in custom models If you can write the model as a program, you can do inference on it

4 [Probabilistic programming language - Wikipedia, the free ...](#)

[en.wikipedia.org/wiki/Probabilistic_relational_programming_language](#) ▾
A probabilistic programming language (PPL) is a programming language specially designed to describe and infer with probabilistic models. PPLs often extend from a ...
[Probabilistic ...](#) · [List of ...](#)

```
for (int d = 0; d < nRanks; d++)  
    Observe(click[d] == user.clicks[d]);
```



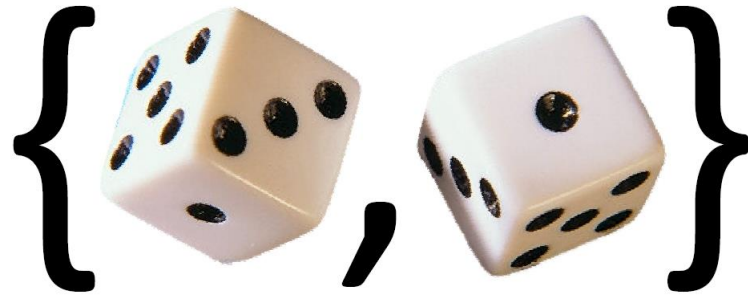
Clickthrough Demo

The image shows a Bing search results page for the query "probabilistic programming". The search bar shows the query and a magnifying glass icon. Below the search bar, it indicates "6,540,000 RESULTS" and provides options to "Narrow by language" and "Narrow by region".

The search results are listed as follows:

- 1** [Probabilistic Programming](#)
[probabilistic-programming.org](#) ▾
PROBABILISTIC-PROGRAMMING.org. This website serves as a repository of links and information about probabilistic programming languages, including both academic ...
- 2** [What is probabilistic programming? - O'Reilly Radar](#)
[radar.oreilly.com/2013/04/probabilistic-programming.html](#) ▾
16/04/2013 · Probabilistic programming languages are in the spotlight. This is due to the announcement of a new DARPA program to support their fundamental research.
- 3** [programming Probabilistic](#)
[research.microsoft.com/en-us/um/...probabilisticprogramming-slides.pdf](#) · PDF file
Goals of Probabilistic Programming Make it easier to do probabilistic inference in custom models If you can write the model as a program, you can do inference on it

Overlaid on the search results is a clickthrough demo. It features two columns of colored boxes: a purple column labeled "Appeal" and a green column labeled "Relevance". To the right of the results are four vertical sliders, each labeled "0 users". Below each slider are four circular buttons: two yellow buttons with a "T" (True) and two blue buttons with an "F" (False). The buttons are arranged in a 2x2 grid for each result. For the first result, the buttons are T, T, F, F. For the second result, the buttons are T, F, T, F. For the third result, the buttons are T, F, T, F.



Probabilistic programming for parsing text

With Tom Minka, Boris Yangel

Example 1: learning a name

```
// Pick a name
string name = Strings.Capitalized(); ←A random capitalized string
// Format it into a text string
string text = String.Format("My name is {0}.", name);

// Observe result
Observe(text, "My name is John.");
// Infer name
var namedist = Infer(name); ←Returns 100%:"John"
```

Example 2: learning a template

```
// Pick a name
string name = Strings.Capitalized();
// Pick a template
string template = Strings.Any() + Chars.Nonword() + "{0}"
                + Chars.Nonword() + Strings.Any();
// Format name into a string using the template
string text = String.Format(template , name);
// Observe result
Observe(text, "My name is John.");
// Infer template
var tempdist = Infer(template); ←Returns 100%:"My name is {0}."
```

Example 2: learning a template

```
// Pick a name
string name = Strings.Capitalized();
// Pick a template
string template = Strings.Any() + Chars.Nonword() + "{0}"
                + Chars.Nonword() + Strings.Any();
// Format name into a string using the template
string text = String.Format(template , name);
// Observe result
Observe(text, "Hello! My name is John.");
// Infer template
var tempdist = Infer(template);
```

50%: "Hello! My name is {0}."
50%: "Hello! {0} name is John."

Example 3: shared template

```
// Pick two names
string name1 = Strings.Capitalized();
string name2 = Strings.Capitalized();
// Pick a template
string template = Strings.Any() + Chars.Nonword() + "{0}"
                + Chars.Nonword() + Strings.Any();
// Format names into strings using the template
string text1 = String.Format(template , name1);
string text2 = String.Format(template , name2);

// Observe texts
// Now template is 100%:"Hello! My name is {0}."
Observe(text1, "Hello! My name is John.");
Observe(text2, "Hello! My name is Andy.");
```


Example 4: non-string values

```
// Pick a name and date
string name = Strings.Capitalized();
DateTime date = Dates.Any(); ←A random DateTime
// Convert date to string
string dateFormat = Strings.OneOf("d MMM, yyyy", "d/M/YY");
string dateString = date.ToString(dateFormat);
// Pick a template
string template = "{0}" + Strings.StartEndN
                  "{1}" + Strings.StartNon
// Format names into strings using the template
string text = String.Format(template, name, dateStr);
// Observe texts
Observe(text, "Fred was born on 6 May, 1963.");
```

name = "Fred"
date = 6/5/1963
dateFormat = "d MMM, yyyy"
dateString = "6 May, 1963"
template = "{0} was born on {1}."

Motif finding



TAGAAAGT
TCGAACAC
ACAAACGT
TAGCACAA

Motif finding: data (synthetic)

CGTGACGGTTACCGCTTCCTATTTG
CATAGAGGCGCGATGAACATGAGAC
GGGAATAACTATGACTTACTTGCGG
CGTCGGTGAAAATATTAGCATGATT
CTTAACTTCGTCGTGAAACTTTGAC
CCCTAAACAACGAAGAAACCGTTTA
AGGACTGTGACCCATCGTTGACGCT
CCCCGGACGTTGACTATCTAATGCA
GAGTGCTCTAGCACTATGAATACGT
CTGGGATCTAGACTATTACAGACGA
GGCACGGGGAGTAAGTCTGAAATGC
CTTAGGCCACGGTTAAGCAACAGGC
CACAACGGTGCCTAACGTATCTTAA
TATCTACGATGAATCACCCAAATAT
GGTCAGGAATTATGACATGCATTGT

AATTTGCTCCTAGCAACTGTGAAGA
AGTGCCGCGGACTATGTCTGTTACA
CGACTGCAACGAACTGTGACATTAA
ACGGTGAATAAGGTCGATAATGGTT
TTGTGCGCAATTCGCTAACTCTGAA
CACGGTGAAATTCGTCGCTCTCAGT
GACCACTGGGAAGGAAACAGCCCGC
CACGTGCCCATCCGATGAACTGATC
GCGACTTTGTCTCTATATGATTATG
GTCACCTGTGACGCCGTGCGGTTTA
CAAGATGCTGGTTCAACGTCGAAAA
GGGAAACAGTACGAGTAGGACTCAA
TACTGTAACGATGACTGCCGGCCGC
GCACATTACGATGAACAATGCCATT
CGAAACCGCGACGTTACGTTGACCC

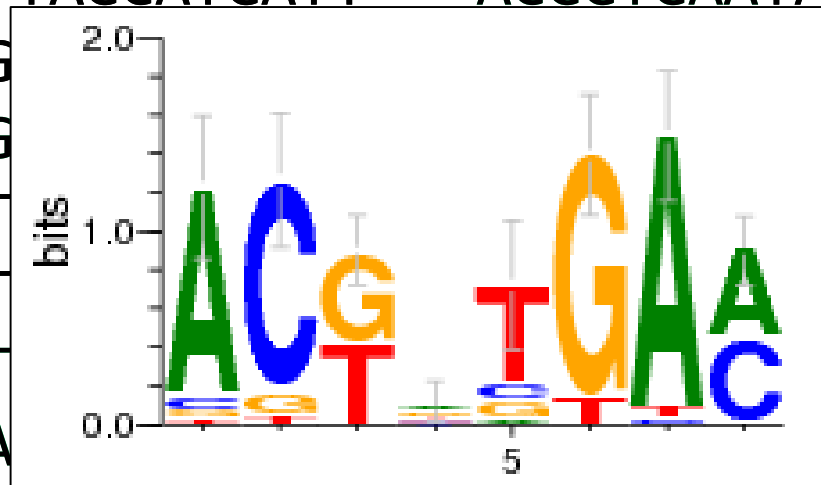
Motif finding: probabilistic program

```
// Generate motif i.e. base probabilities
Vector[] motifProbs = new Vector[motifLen];
for (int i=0; i < motifLen; i++) motifProbs[i] = Random(dirichletPrior);
// Loop over sequences
for (int j=0; j < seqs.Length; j++) {
    // Generate motif instance
    for (int i=0; i < motifLen; i++) motif[i] = (char)Random(motifProbs[i]);
    // Choose motif position
    int motifPosition = Random(Discrete(0, seqs[j].Length-motifLen));
    // Generate left and right background strings
    string left = Strings.OfLength(motifPosition, backgroundDist);
    int rightLength = seqs[j].Length - motifPosition - motifLen;
    string right = Strings.OfLength(rightLength, backgroundDist);
    // Observe result
    Observe(seqs[j] == left + motif + right);
}
```

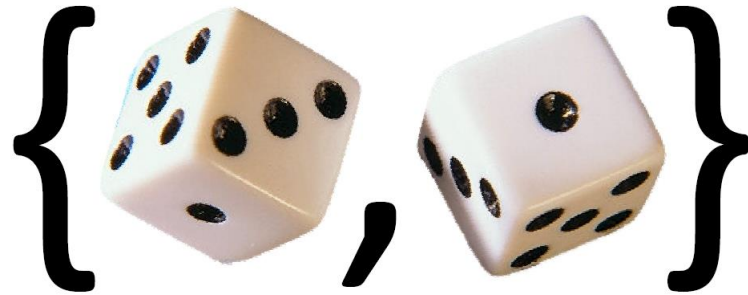
Motif finding: results

CGTGACGGTTACCGCTTCCTATTTG
CATAGAGGCGCGATGAACATGAGAC
GGGAATAACTATGACTTACTTTGCGG
CGTCGGTGAAAATATTAGCATGATT
CTTAACTTCGTCGTG
CCCTAAACAACGAAG
AGGACTGTGACCCAT
CCCCGGACGTTGACT
GAGTGCTCTAGCACT
CTGGGATCTAGACTA
GGCACGGGGAGTAAGTCTGAAATGC
CTTAGGCCACGGTTAAGCAACAGGC
CACAAACGGTGCCTAACGTATCTTAA
TATCTACGATGAATCACCAAATAT
GGTCAGGAATTATGACATGCATTGT

AATTTGCTCCTAGCAACTGTGAAGA
AGTGCCGCGGACTATGTCTGTTACA
CGACTGCAACGAACTGTGACATTAA
ACGGTGAATAAGGTCGATAATGGTT
TTCGCTAACTCTGAA
TTCGTCGCTCTCAGT
AAGGAAACAGCCCGC
TCCGATGAACTGATC
CTCTATATGATTATG
ACGCCGTGCGGTTTA
CAAGATGCTGGTTCAACGTCGAAAA
GGGAAACAGTACGAGTAGGACTCAA
TACTGTAACGATGACTGCCGGCCGC
GCACATTACGATGAACAATGCCATT
CGAAACCGCGACGTTACGTTGACCC



PREDICTION OVERLAP GROUND TRUTH



Making probabilistic programs run faster (or at all!)

With Nicholas Heess, Danny Tarlow and Ali Eslami

Two probabilistic programming systems

Probabilistic C# & Infer.NET

```
bool[] ProbitRegression(VectorGaussian weightsPrior,  
                        Vector[] features)  
{  
    Vector weights = Random(weightsPrior);  
    bool[] labels = new bool[features.Length];  
    for (int i = 0; i < features.Length; i++)  
    {  
        double score = Vector.InnerProduct(weights, features[i]);  
        double noise = Random(Gaussian(0,0.1));  
        labels[i] = (score + noise) > 0;  
    }  
    return labels;  
}
```

Probit Regression Classifier

Church

```
(define (DP alpha proc)  
  (let ((sticks (mem (lambda x (beta 1.0 alpha))))  
        (atoms (mem (lambda x (proc)))))  
    (lambda () (atoms (pick-a-stick sticks 1)))))  
  
(define (pick-a-stick sticks J)  
  (if (< (random) (sticks J))  
      J  
      (pick-a-stick sticks (+ J 1))))  
  
(define (DPmem alpha proc)  
  (let ((dps (mem (lambda args  
                  (DP alpha  
                    (lambda () (apply proc args))  
                    )))))  
    (lambda argsin ((apply dps argsin))) ))
```

Dirichlet Process

How inference works

Probabilistic C# & Infer.NET

- Deterministic message passing for large set of built-in factors e.g. EP, VMP.
- Detailed understanding of the program structure and functions used.
- Messages can be passed in either direction.

+ Very fast inference/large data

– Less flexible modelling language

Church

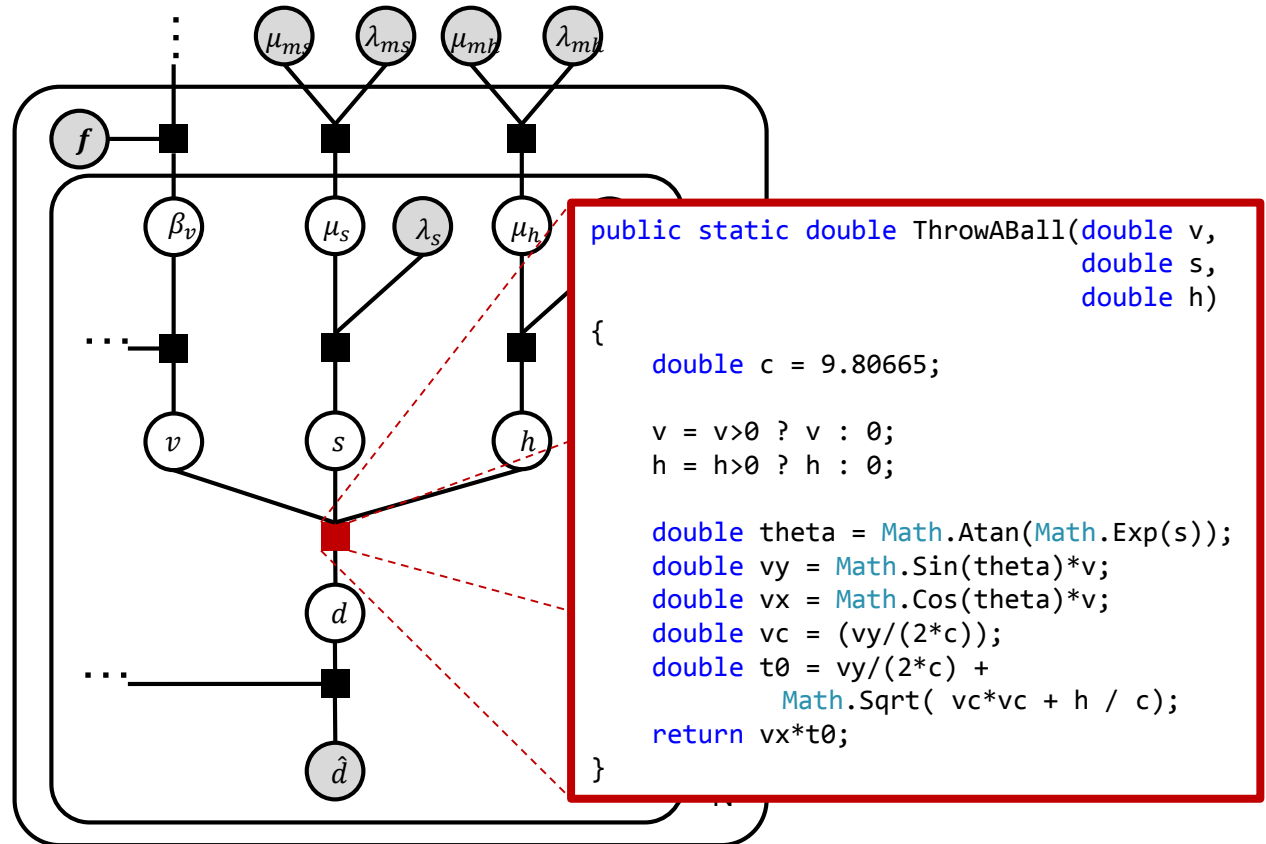
- Various sampling approaches e.g. MH, MCMC
- Program treated as a ‘black box’ (more recent work uses dependencies)
- Program only run forwards.

+ Very flexible modelling language

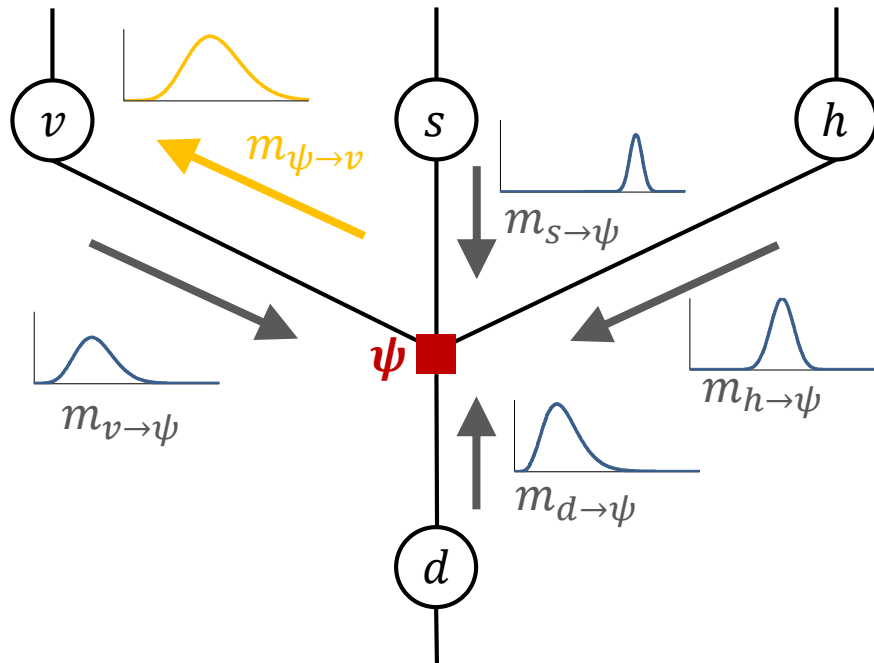
– Slower inference/smaller data

Best of both worlds?

1. Define an *arbitrary* factor using its forward program.
2. Use sampling methods to (very slowly) compute the messages coming from the factor.
3. Train a fast regression model to predict outward messages from input messages.
e.g. random forest/neural net
4. Use regression model to do fast, scalable inference in the entire graph.



Expectation Propagation



Variable-to-factor message

$$m_{\psi i}(x_i) = \frac{\text{proj} \left[\int \psi(x_{\text{out}} | x_{\text{in}}) \left(\prod_{i' \in \text{Scope}(\psi)} m_{i' \psi}(x_{i'}) \right) dx_{\psi-i} \right]}{m_{i \psi}(x_i)}$$

Hard to compute – Infer.NET mainly uses hand-crafted numerical approximations.

Instead – *learn* to compute these messages.

Training data via sampling

Using some proposal distribution q over input variables, compute variable-to-factor message:

$$\int \psi(x_{out} | x_{in}) \left(\prod_{i' \in Scope(\psi)} m_{i'\psi}(x_{i'}) \right) dx_{\psi} = \mathbb{E}_r \left[\frac{\prod_{i' \in Scope(\psi)} m_{i'\psi}(x_{i'})}{q(x_{in})} \right]$$

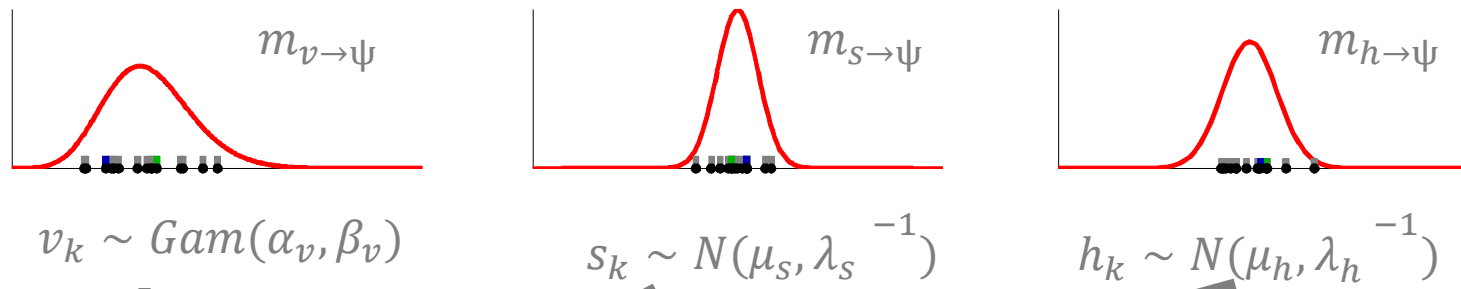
where $r(x) = q(x_{in})\psi(x_{out} | x_{in})$. In the simplest case: $q(x_{in}) = \prod_{i' \in in} m_{i'\psi}(x_{i'})$.

[See also ABC-EP, S. Barthelme and N. Chopin.]

Need also to specify a distribution over input messages or use messages encountered as the inference runs.

Train neural net/random forest to predict output from input.

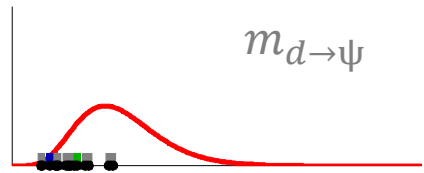
① Sample from incoming messages



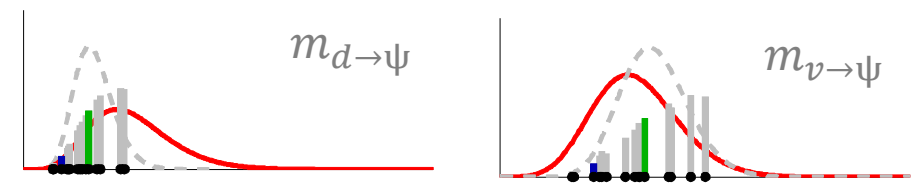
ψ

② Sample factor function

$$d_k = f(v_k, s_k, h_k) \sim \psi(\cdot | v_k, s_k, h_k)$$



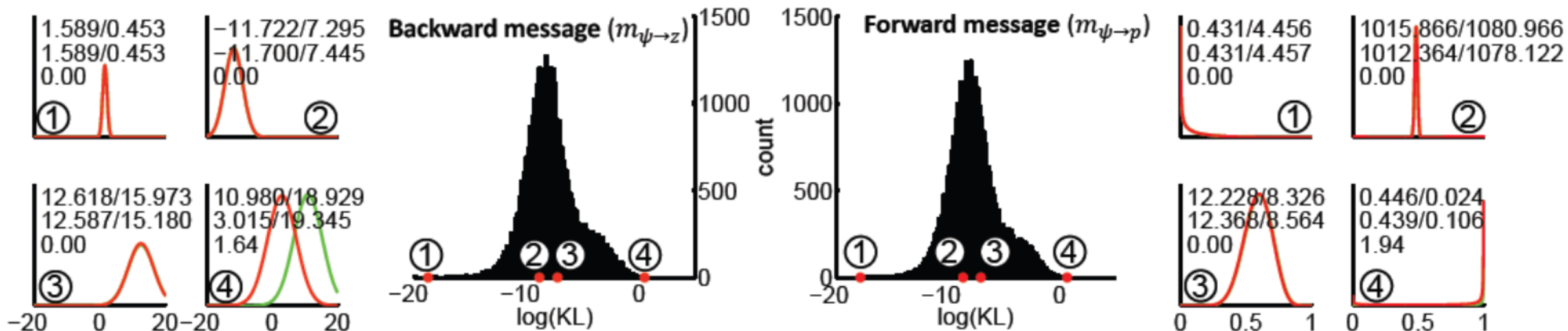
③ Compute IS weights $w_k = \text{Gam}(d_k | \alpha_d, \beta_d)$ e.g.



④ Moments of weighted samples, e.g. for v

$$\hat{\mu} = \frac{\sum_k w_k v_k}{\sum_k w_k} \quad \hat{\sigma}^2 = \frac{\sum_k w_k (v_k - \hat{\mu})^2}{\sum_k w_k}$$

Logistic Factor $f(x)=1/(1+\exp(-z))$

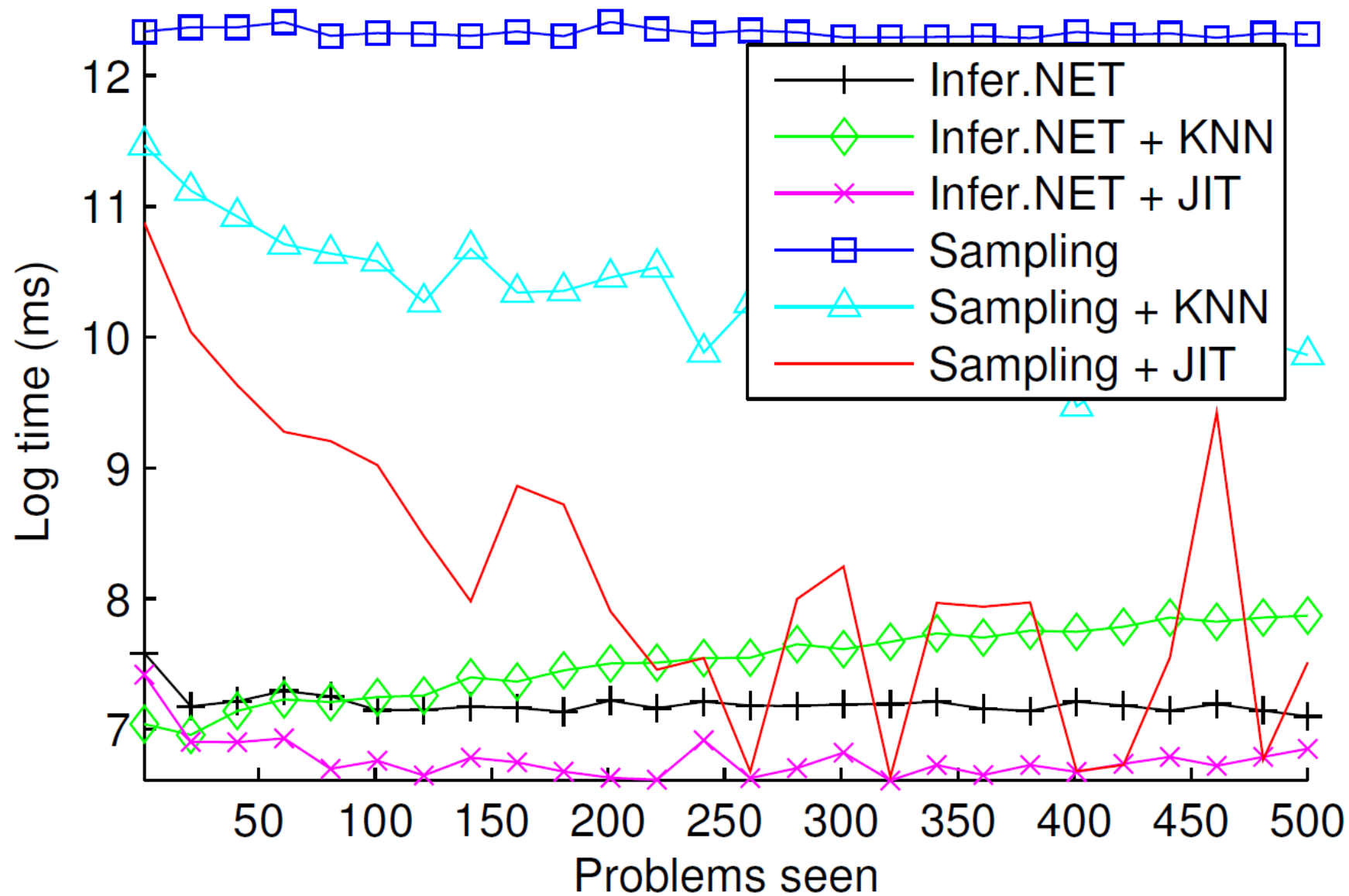


Logistic regression experiments (artificial data)

Dataset	D	D_{rel}	N	Prior var.	EP Test acc.	NN Test acc.	$KL(EP NN)$
1	10	4	500	.1	0.7750	0.7745	1.9770
1	10	4	500	1	0.7750	0.7750	1.3247
2	50	10	500	.1	0.7635	0.7635	0.4649
2	50	10	500	1	0.7660	0.7660	0.5641
4	50	40	500	.1	0.8825	0.8820	0.6022
5	50	40	100	1	0.8065	0.8050	0.0847

Test accuracy on UCI ionosphere dataset: 0.8839 (default factor) vs. 0.8800 (learned NN factor).

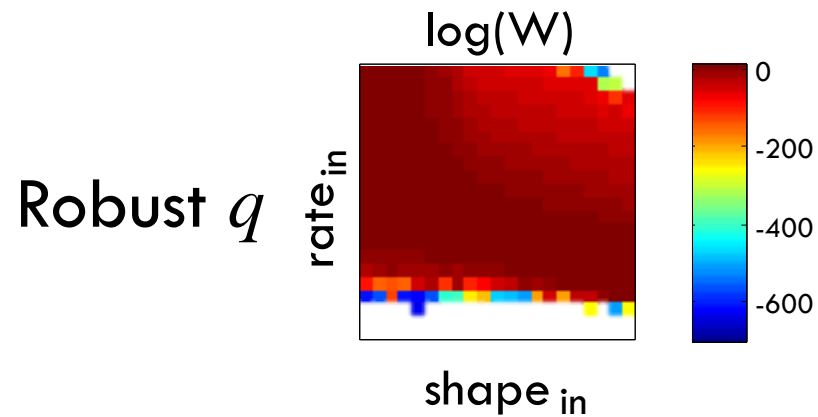
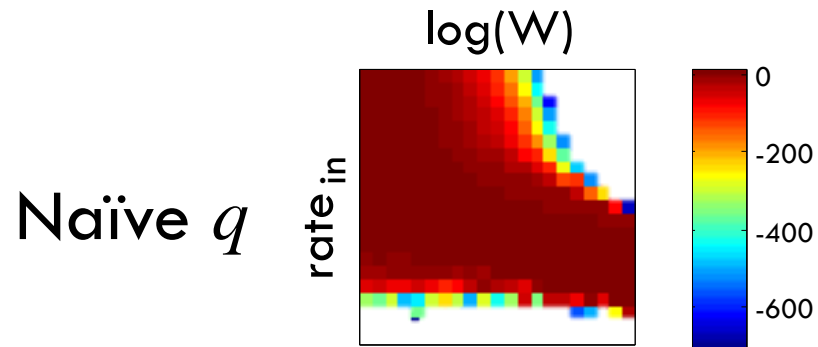
Speed up over sampling



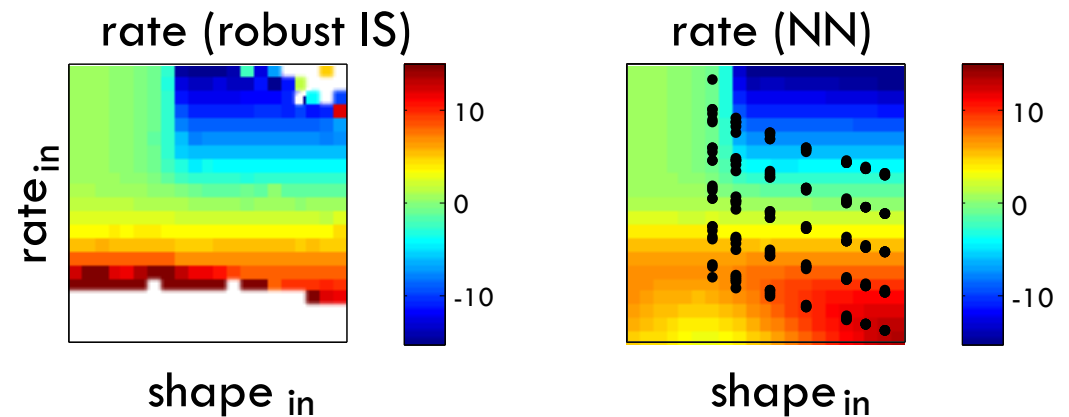
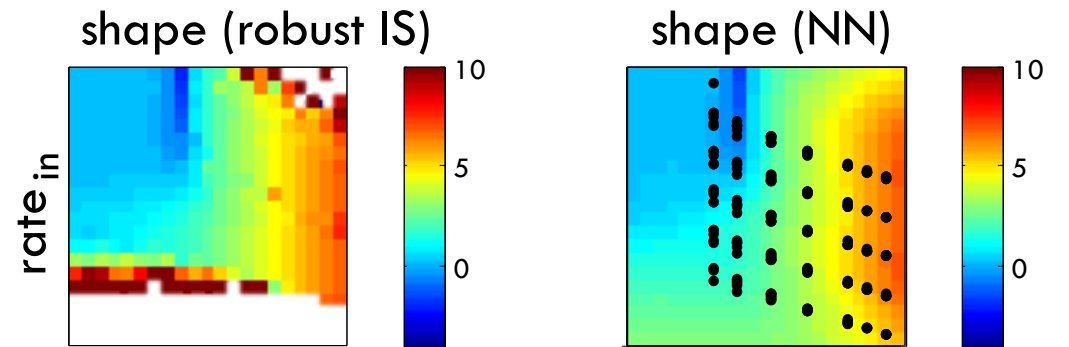
Compound Gamma

$$f() \sim \int \text{Gam}(\cdot | \alpha_B, \beta) \text{Gam}(\beta | \alpha_A, \beta_A) d\beta$$

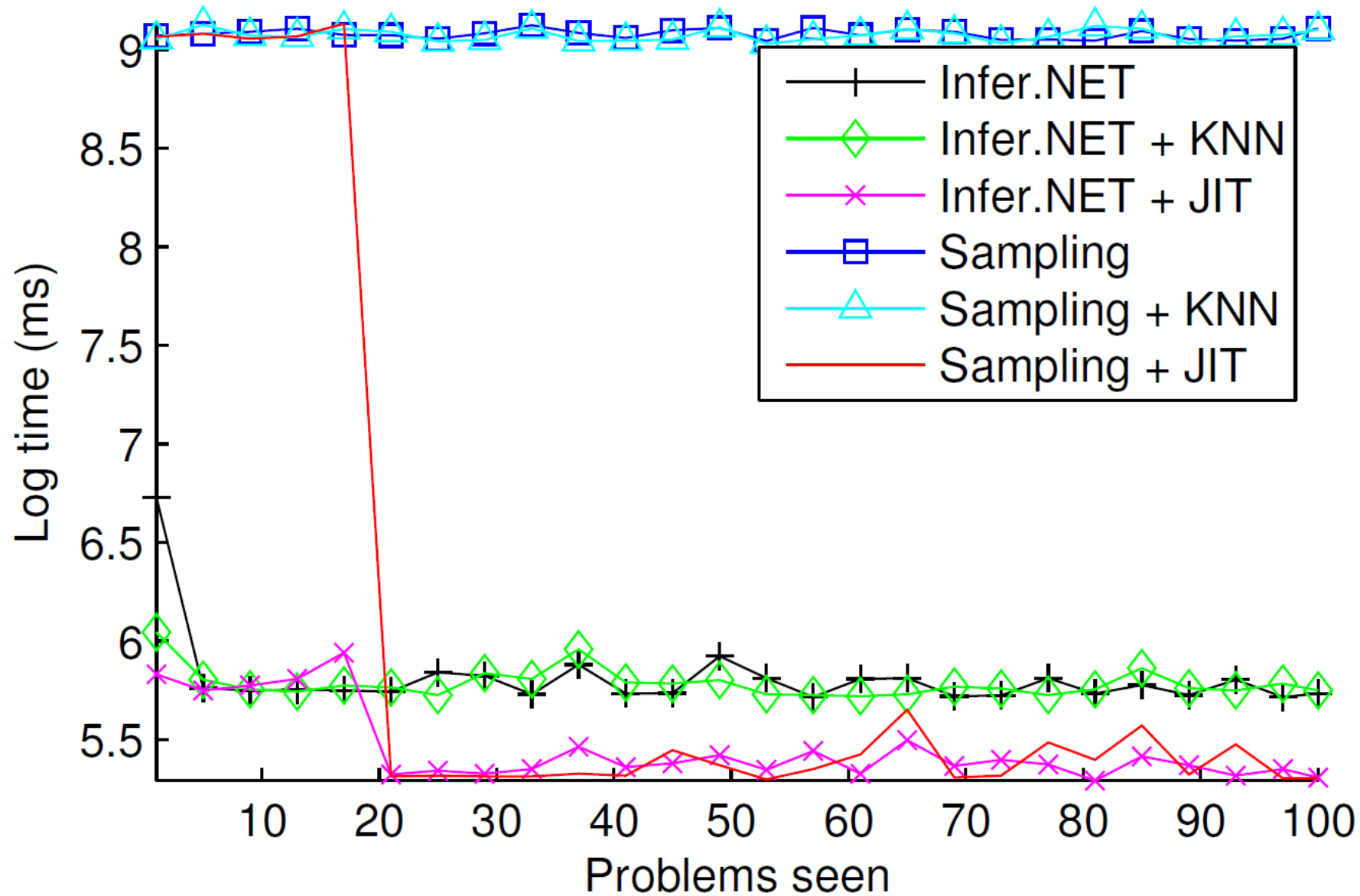
IS comparison



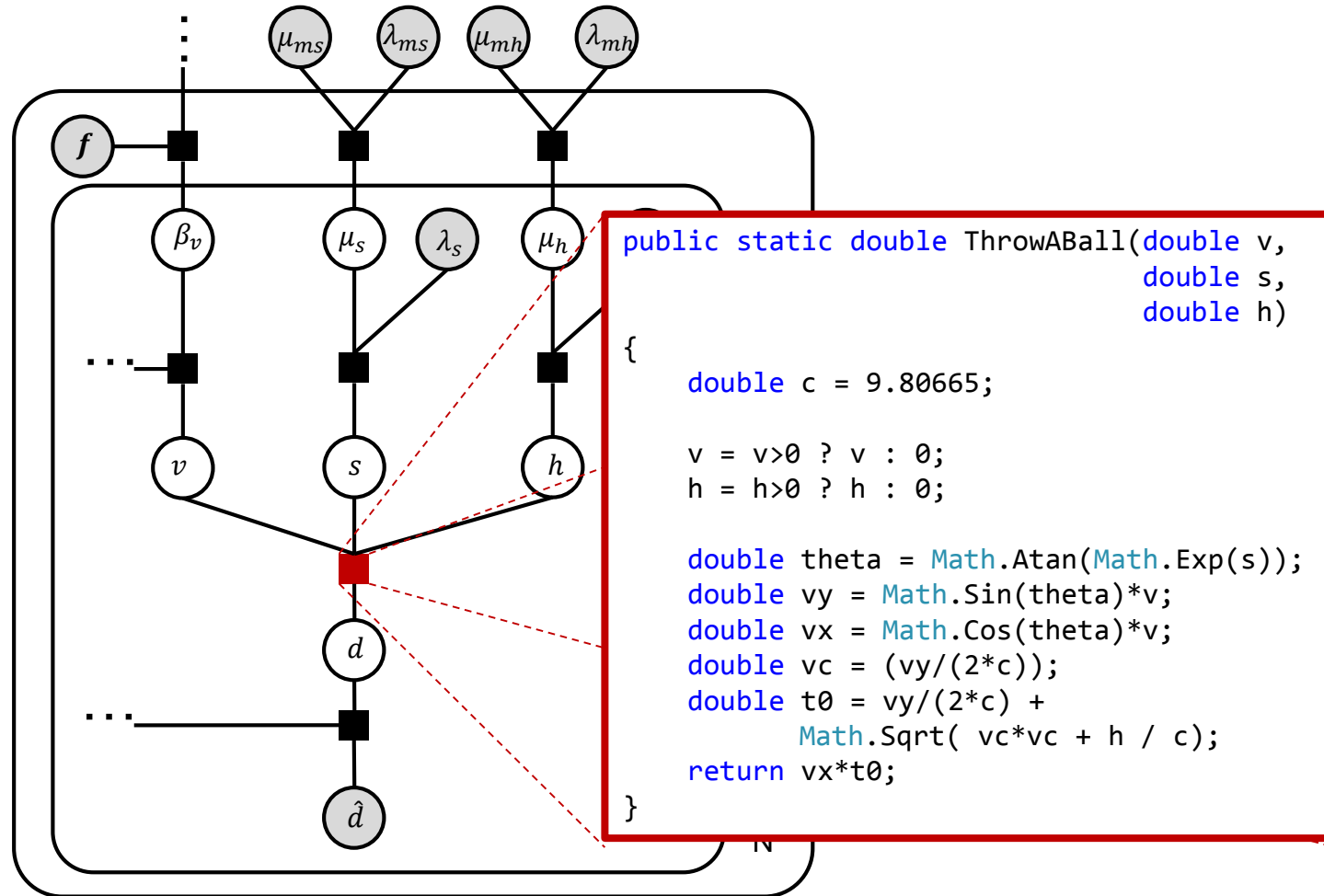
Message surfaces for robust q



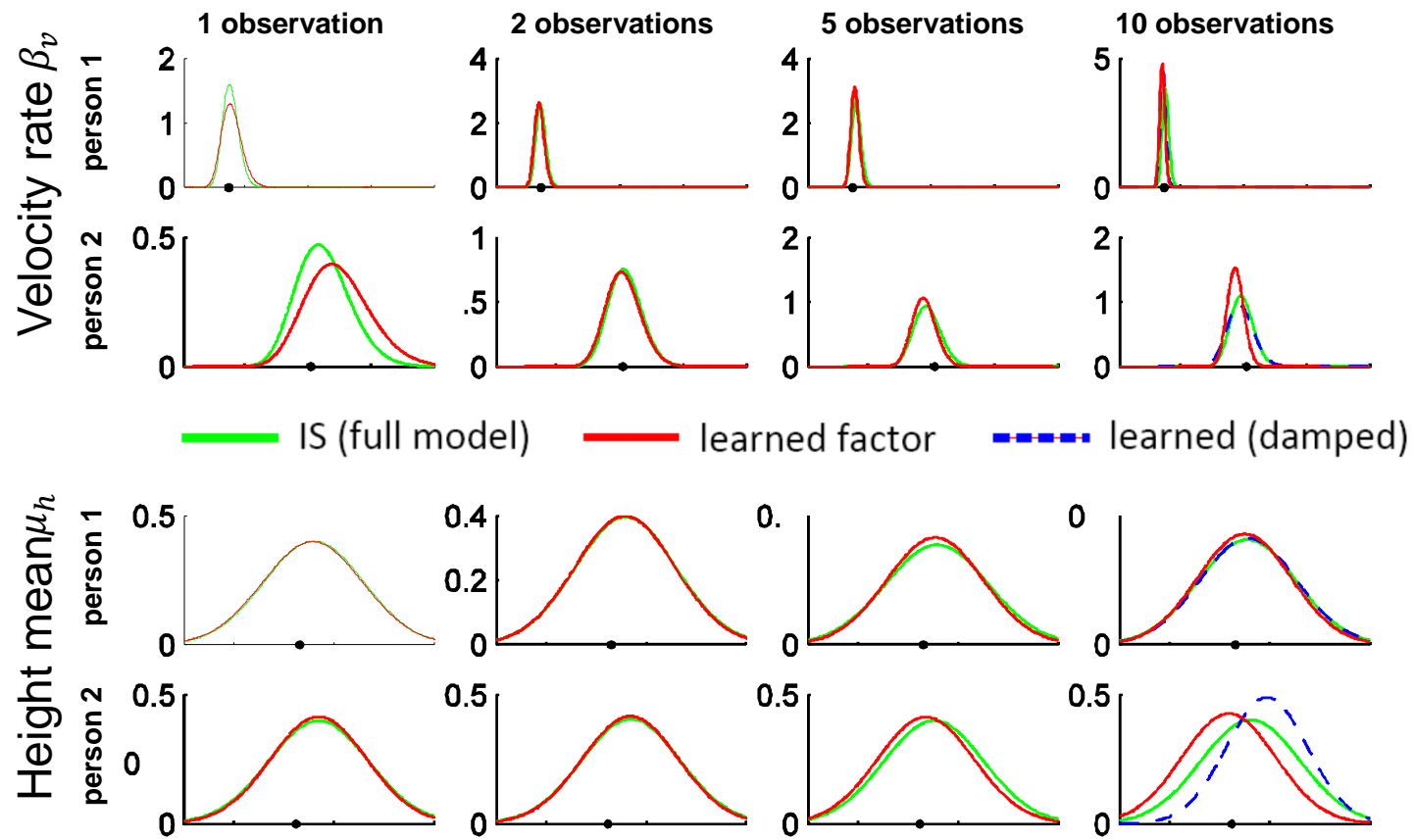
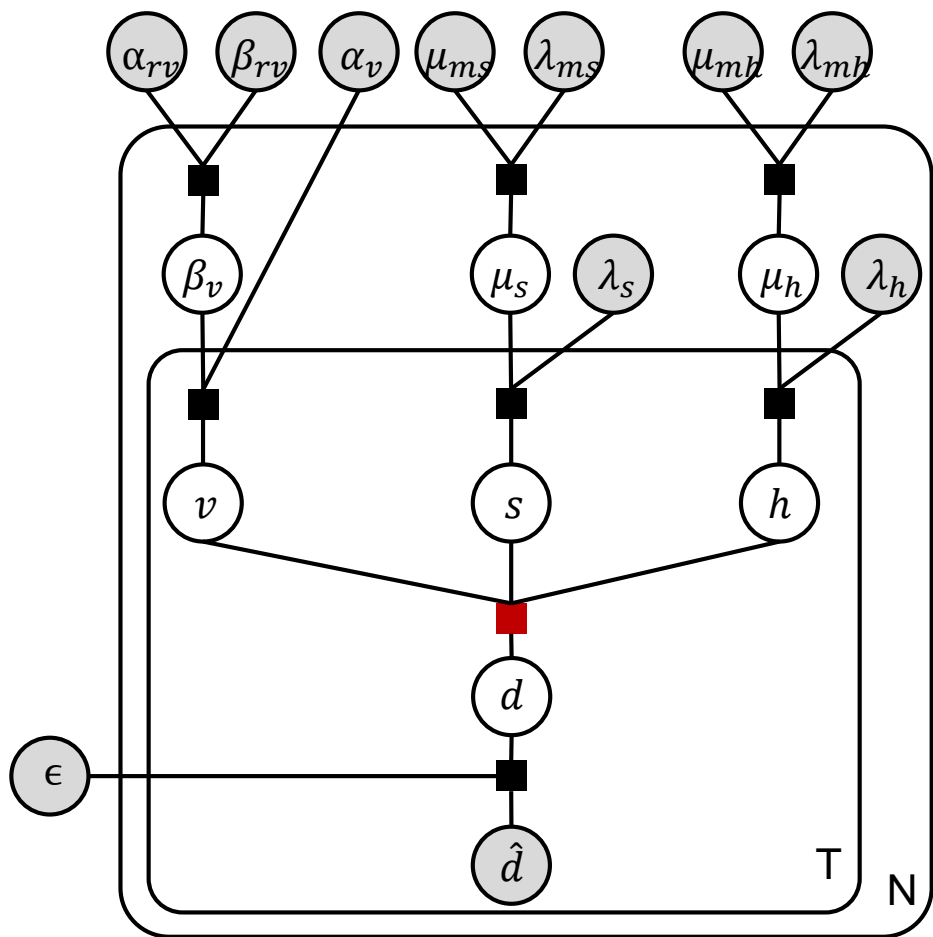
Speed up over sampling



Ball Throwing Simulation



Ball Throwing Simulation



Conclusions

Probabilistic programs can...

- ▶ Solve complex problems,
- ▶ Be quick to write,
- ▶ Be fast to execute.

But more work is needed on...

- ▶ Speed
- ▶ Speed
- ▶ Speed.
- ▶ (& ease of use/reliability/debugging/training/examples...)

In the future, every programmer will be doing machine learning!

Thanks!



infer.net

<http://research.microsoft.com/infernet>