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?
// Create two strings
string a = "Hello uncertain";
string b = "world";
// Format strings together into a new string
string c = a + " " + b + "!!";
// Write it out
Console.WriteLine(c);
In probabilistic C# (Csoft)

```csharp
// 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

50%: "Hello uncertain" 50%: "Hello uncertain world"

Observe result

Observe (c == "Hello uncertain world");
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’.
We can define observations on random variables:
- \text{Observe}(\text{visible}==\text{true})
- \text{Observe}(\text{length}==4)
- \text{Observe}(\text{length}>0)
- \text{Observe}(\text{i}==\text{j})

\textbf{Observe (b)} means ‘we observe b to be true’.
Inference

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

- **Example:**
  ```java
  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
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.
// 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
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.

<table>
<thead>
<tr>
<th>Gamertag</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sully</td>
<td>25</td>
</tr>
<tr>
<td>SniperEye</td>
<td>22</td>
</tr>
<tr>
<td>DrSlowPlay</td>
<td>17</td>
</tr>
</tbody>
</table>

New Estimates of Players’ Skills

Old Estimates of Players’ Skills
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.
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

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/unpub1/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_relatio...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 ...
Let's look at the next result… and see if it's worth clicking on…

Let's look at the page… and see if it's useful.

Aaargh!

It's relevant! Done!

That looks promising… let's click…

Programming a user

```javascript
var doNext = Random(Bernoulli(appeal));
var next = Random(Bernoulli(0.2));
var isRel = Random(Bernoulli(relevance));
```
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]));
for (int d = 0; d < nRanks; d++)
    Observe(click[d] == user.clicks[d]);
Clickthrough Demo

1. Probabilistic Programming
   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.
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

```csharp
// 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
Observe(text1, "Hello! My name is John.");
Observe(text2, "Hello! My name is Andy.");

Now template is 100%:"Hello! My name is {0}."
Example 4: non-string values

```csharp
// 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.StartEndNonWord() + "{1}" + Strings.StartNonWord();
// Format names into strings using the template
string text = String.Format(template, name, dateString);
// 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
TCGAACACAC
ACAAACACGT
TAGCACAATA
Motif finding: data (synthetic)

CGTGACGGTTACCGCTTCTATTT
CGTAGAGGCGCGATGAACATGAGAC
GGGAATAACTATGACTTACTTGCGG
CGTCGGTGAAAATATTAGCATGATT
CTTTACTTCGTCGTAACCTTGGAC
CCCTAAACAACGAAAGAAACCGTTTA
AGGACTGTGACCCCATCGTTGACGCT
CCCCGGACGTTGACTATCTAATGCA
GAGTGCTCTAGCACTATGAATACGT
CTGGGATCTAGACTATTACAGACGA
GGCACGGGGAGTAAGTCTGAAATGC
CTTAGGCCACGGTTAAGCAACAGGC
CACAACGGTGCCTAACGTATCTTAA
TATCTACGATGAATCACCCAAATAT
GGTCAGGAATTATGACATGCATTGT
AATTTGCTCCTAGCAACTGTGAAGA
AGTGCCGCGGACTATGTCTGTTACA
CGACTGCAACGAACTGTGACATTAA
ACGGTGAAATAAGGTCGATAATGGTT
TTGTGCGCAATTTCGCTAACTCTGAA
CACGGTGAAATTTCGTCGCTCTCAGT
GACCACCTGGGAAGGAAACAGCCC
CACGTGCCCATCCGATGAACGTATC
GCGACTTTGTCTCTATATGATTATG
GTCACCTGTGACGCGTGCAGTTTA
CAAGATGCTGGTTCAACGTCGAAAA
GGGAAACAGTACGAGTAGGACTCAA
TACTGTAACGATGACTGCCGCCG
CGGAAACAGTACGAGTAGGACTCAA
CACGTGCCCATCCGATGAACGTATC
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

CGTGACGGTTACCGCTTCTTATTTG AATTTCCTCATGCAACTGTGAAGA
CATAGAGGCGCGATGAACATGAGAC AGTGCCCGGGACTATGTCTGTAA
GGGAATAAACTATGACCTTACTTGCACG ACTGCAACGAACGTGACATTAA
CGTGACGGTTAAAATATTAGCATGATT ACGGTGAATAAGGTCGATAATGGTT
CTTAAACTTCTCGTGTCGCTTACGTTGC AAGTTCTAACTCTGGA
CCCTAAAACAACGAAAAGCATGAAAT TCCGTGCTGCTCTCAGT
AGGACCTTGACCCCATCTGGTTCTCAAG AAGGAAACAGCCCAGC
CCCCCGACGTGGACTTACTTGTGGTT CAAGGTTCTTCGATGAAACTGTGAAG
GAGTGCTCTAGCACTATGAAATACGT TCTCTATATGATTATG
CTGGGATCTAGCTACTTTCACGTTA TACGCCGTGCCTTTTA
GGCACCGGGGAGTAAAGTCTGAAATGC CAAGATGCTGGTTCAACCTGCGAAA
CTTAGGGCCAACGTTAAGCAACAGGC GGGAAACAGTAGGACTGACTCCAA
CACAACGGTGCCTAAGCTATCTTAA TACTGTAACGATGACTCCGGCCGC
TATCTACGATGAATCCCAATTATGC GCACATTACGATGACTCCGGCCGC
GGTCAGGAATTATGACATGCATTGT CGAAACCGCAGTTACGTTGACC

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

```csharp
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

```clike
(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

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

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:

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

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

[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

\[ m_{v \rightarrow \psi}, m_{s \rightarrow \psi}, m_{h \rightarrow \psi} \]

\[ v_k \sim \text{Gam}(\alpha_v, \beta_v) \]

\[ s_k \sim \text{N}(\mu_s, \lambda_s^{-1}) \]

\[ h_k \sim \text{N}(\mu_h, \lambda_h^{-1}) \]

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 \( w_k = \text{Gam}(d_k | \alpha_d, \beta_d) \)

\[ m_{d \rightarrow \psi} \]

4. Moments of weighted samples, e.g. for \( v \)

\[ \hat{\mu} = \frac{\sum_k w_k v_k}{\sum_k w_k} \]

\[ \hat{\sigma}^2 = \frac{\sum_k w_k (v_k - \hat{\mu})^2}{\sum_k w_k} \]
Logistic Factor  \( f(x) = \frac{1}{1+\exp(-z)} \)

**Logistic regression experiments (artificial data)**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(D)</th>
<th>(D_{rel})</th>
<th>(N)</th>
<th>Prior var.</th>
<th>EP Test acc.</th>
<th>NN Test acc.</th>
<th>(KL(EP|NN))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>500</td>
<td>.1</td>
<td>0.7750</td>
<td>0.7745</td>
<td>1.9770</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>500</td>
<td>1</td>
<td>0.7750</td>
<td>0.7750</td>
<td>1.3247</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>10</td>
<td>500</td>
<td>.1</td>
<td>0.7635</td>
<td>0.7635</td>
<td>0.4649</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>10</td>
<td>500</td>
<td>1</td>
<td>0.7660</td>
<td>0.7660</td>
<td>0.5641</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>40</td>
<td>500</td>
<td>.1</td>
<td>0.8825</td>
<td>0.8820</td>
<td>0.6022</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>40</td>
<td>100</td>
<td>1</td>
<td>0.8065</td>
<td>0.8050</td>
<td>0.0847</td>
</tr>
</tbody>
</table>

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

![Graph showing Log time (ms) vs Problems seen for different methods: Infer.NET, Infer.NET + KNN, Infer.NET + JIT, Sampling, Sampling + KNN, Sampling + JIT.](image)
\[ f() \sim \int \text{Gam}(\cdot | \alpha_B, \beta) \text{Gam}(\beta | \alpha_A, \beta_A) d\beta \]

**IS comparison**

- **Naïve q**
  - log(W) vs. rate in

- **Robust q**
  - log(W) vs. rate in

**Message surfaces for robust q**

- **shape (robust IS)**
  - rate vs. shape in

- **shape (NN)**
  - rate vs. shape in

- **rate (robust IS)**
  - rate vs. shape in

- **rate (NN)**
  - rate vs. shape in
Speed up over sampling
Ball Throwing Simulation

public static double ThrowABall(double v, double s, double h)
{
    double c = 9.80665;
    v = v>0 ? v : 0;
    h = h>0 ? h : 0;

    double theta = Math.Atan(Math.Exp(s));
    double vy = Math.Sin(theta)*v;
    double vx = Math.Cos(theta)*v;
    double vc = (vy/(2*c));
    double t0 = vy/(2*c) +
        Math.Sqrt( vc*vc + h / c);
    return vx*t0;
}
Ball Throwing Simulation

\[ \beta \cdot v \cdot \beta \cdot r \cdot v \cdot \alpha \cdot r \cdot v \cdot \mu_s \cdot \lambda_s \cdot \mu_h \cdot \lambda_h \]

\[ v \cdot s \cdot h \]

\[ \epsilon \]

\[ d \]

\[ T \]

\[ N \]

Velocity rate

- 1 observation
- 2 observations
- 5 observations
- 10 observations

Height mean

- IS (full model)
- learned factor
- learned (damped)
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!

http://research.microsoft.com/infernet