From automatic differentiation to message passing

Tom Minka
Microsoft Research
EuroAD 2019
Message-passing

• Unified perspective on algorithms in machine learning, signal processing
  • Constraint satisfaction
  • Optimization
  • Integration / summation / probabilistic inference
Examples of message-passing

- Viterbi, Dijkstra algorithms
- Markov chain, Kalman filter recursions
- Modern error-correcting codes
- Belief propagation in Bayesian networks
Message-passing algorithms

• Decompose function into elementary operations
• Approximate each operation
• Re-assemble the function
• To answer specific questions
Why “message passing”?  

- Each operation talks to its neighboring operations, perhaps asynchronously, to reach a fixed point  
- Passing messages in the computation graph
• Originally done from scratch for each problem

• Some libraries take explicit computation graph as input

• Now a program transformation
## Summary of AutoDiff

<table>
<thead>
<tr>
<th>Feature</th>
<th>AD</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs not formulas</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Graph structure / sparsity</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Source-to-source</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Only one execution path</td>
<td>Yes</td>
<td>Not always</td>
</tr>
<tr>
<td>Single forward-backward sweep</td>
<td>Yes</td>
<td>Not always</td>
</tr>
<tr>
<td>Exact</td>
<td>Yes</td>
<td>Not always</td>
</tr>
</tbody>
</table>
Automatic differentiation

c = f(x,y)

dc = df1(x,y) * dx + df2(x,y) * dy

xB = cB * df1(x,y)
yB = cB * df2(x,y)
Interval constraint propagation

• What is largest and smallest value each variable could have?

• Each operation in program is interpreted as a constraint between inputs and output

• Propagates information forward and backward until convergence
Find \((x, y)\) that satisfies \(x^2 + y^2 = 1\) and \(y = x^2\)
Input program

\[ y = x^2 \]
\[ yy = y^2 \]
\[ z = y + yy \]
\[ \text{assert}(z == 1) \]
Interval propagation program

Input program

\[
\begin{align*}
y &= x^2 \\
yy &= y^2 \\
z &= y + yy \\
\text{assert}(z == 1)
\end{align*}
\]

Edge program

\[
\begin{align*}
y &= x^2 \\
(y_1, y_2) &= \text{dup}(y) \\
yy &= y_1^2 \\
z &= y_2 + yy \\
\text{assert}(z == 1)
\end{align*}
\]
Interval propagation program

Edge program

\[
y = x^2
\]

\[
(y1, y2) = \text{dup}(y)
\]

\[
\text{yy} = y1^2
\]

\[
z = y2 + \text{yy}
\]

assert(z == 1)

Message program

Until convergence:

\[
yF = xF^2
\]

\[
y1F = yF \cap y2B
\]

\[
y2F = yF \cap y1B
\]

\[
\text{yy}F = y1F^2
\]

\[
y1B = \sqrt{y1F, \text{yy}B}
\]

\[
y2B = zB - \text{yy}F
\]

\[
\text{yy}B = zB - y2F
\]

\[
zB = [1,1]
\]
Running \(^2\) backwards

\[ yy = y1^2 \quad \rightarrow \quad y1B = \sqrt{y1F, yyB} = \text{project}[y1F \cap \sqrt{yyB}] \]

\[ yyB = [1, 4] \]
\[ \sqrt{yyB} = [-2, -1] \cup [1, 2] \]
\[ y1F = [0, 10] \]
\[ y1F \cap \sqrt{yyB} = [] \cup [1, 2] \]
\[ \text{project}[y1F \cap \sqrt{yyB}] = [1, 2] \]
\[ y1F \cap \text{project}[\sqrt{yyB}] = [0, 2] \]
• If all intervals start \((-\infty, \infty)\) then \(x \to (-1,1)\) (overestimate)

• Apply subdivision

• Starting at \(x = (0.1,1)\) gives \(x \to (0.786, 0.786)\)
Simplifications of message-passing

• Message dependencies dictate execution

• If forward messages do not depend on backward, becomes non-iterative

• If forward messages only include single state, only one execution path is explored

• AutoDiff has both properties
Probabilistic Programming

- Probabilistic programs replace Bayesian networks
- Using a program to specify a probabilistic model
- Program is not a black box: undergoes analysis and transformation to help inference
Loopy belief propagation

• Loopy belief propagation has same structure as interval propagation, but using distributions

• Expectation propagation adds projection steps

• Subsumes AD: If forward messages are point-mass distributions, backward messages are derivatives
Thanks!

Infer.NET is open source: http://dotnet.github.io/infer