THE CASCADE-CORRELATION LEARNING ARCHITECTURE
S. FAHLMAN & C. LEBIERE
AUGUST 1991

Presentation by Jeremy Wurbs
CSCE 636
2.22.2010
PRESENTATION OVERVIEW

- CC Learning Architecture
  - Basic Architecture
  - Adding Hidden Units
- Advantages of CCLA
- Benchmark Tests
- Closing Remarks
CCLA – Basic Architecture
CCLA – Basic Architecture

Inputs

\( x_1 \)
\( x_2 \)
\( x_3 \)

Weights

\( \sum \)

Outputs

\( o_1 \)
\( o_2 \)

Learning Rule:
- Delta
- Perceptron
- Quickprop
- Etc.
CCLA – Adding Hidden Units

Sample Input Patterns

<table>
<thead>
<tr>
<th>a1</th>
<th>b1</th>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>a3</td>
<td>b3</td>
<td>c3</td>
</tr>
</tbody>
</table>

Inputs

- a₁
- a₂
- a₃

Hidden Units

\[ \sum \int \rightarrow V_a \]

Outputs

\[ \sum \int \rightarrow a_{o,1} \]
\[ \sum \int \rightarrow a_{o,2} \]

Desired

\[ d_{a,o1} \]
\[ d_{a,o2} \]

Error

\[ E_{a,1} \]
\[ E_{a,2} \]

Output

<table>
<thead>
<tr>
<th>Input</th>
<th>V_a</th>
<th>E_{a,1}</th>
<th>E_{a,2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>V_a</td>
<td>E_{a,1}</td>
<td>E_{a,2}</td>
</tr>
<tr>
<td>b</td>
<td>V_b</td>
<td>E_{b,1}</td>
<td>E_{b,2}</td>
</tr>
<tr>
<td>c</td>
<td>V_c</td>
<td>E_{c,1}</td>
<td>E_{c,2}</td>
</tr>
</tbody>
</table>

Outputs - Desired = Error

\[ a_{o,1} - d_{a,o1} = E_{a,1} \]
\[ a_{o,2} - d_{a,o2} = E_{a,2} \]
## CCLA – Adding Hidden Units

### Input - Output

<table>
<thead>
<tr>
<th>Input</th>
<th>(V_p)</th>
<th>(E_{b,1})</th>
<th>(E_{b,2})</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>(V_a)</td>
<td>(E_{a,1})</td>
<td>(E_{a,2})</td>
</tr>
<tr>
<td>b</td>
<td>(V_b)</td>
<td>(E_{b,1})</td>
<td>(E_{b,2})</td>
</tr>
<tr>
<td>c</td>
<td>(V_c)</td>
<td>(E_{c,1})</td>
<td>(E_{c,2})</td>
</tr>
</tbody>
</table>

### Equations

\[
S = \sum_o \left| \sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \right|
\]

\[
\frac{\partial S}{\partial w_i} = \sum_{p,o} \sigma_o (E_{p,o} - \overline{E_o}) f'_p I_{i,p}
\]

### Notes

- \(\sigma_o = \text{sign}(V_p - E_{p,o})\)
- \(I_{i,p} = \text{input to the CU from unit } i, \text{ pattern } p\)
- \(f'_p = \text{derivative of the CU’s activation function wrt the sum of its inputs}\)

*CU denotes ‘candidate unit’*
CCLA – Adding Hidden Units

Inputs:
- $x_1$
- $x_2$
- $x_3$

Hidden Units

Outputs:
- $o_1$
- $o_2$

$\sum x \rightarrow V_x \rightarrow o_1 \rightarrow o_2$
WHY USE CASCADE-CORRELATION LA? CITED PROBLEMS

- The Step-Size Problem
  - How large should each gradient descent step be?
  - Momentum Terms
  - Quickprop

- The Moving Target Problem
  - Lack of communication b/w neurons
  - Herd Effect
  - Similar to adjusting spokes on a bicycle wheel
Why Use Cascade-Correlation LA?
General Advantages

- Each hidden trained one at a time, limiting the moving target problem
- Network dimensions not needed in advance
- Easily builds higher order features
  - Complex learning structure that builds many layers quickly
  - Hidden units may use different activation functions
- Feature detectors aren’t cannibalized
- Candidate pools can be used to assure unit utility
Benchmark Tests: 2-Spiral Problem

Figure 2: Training points for the two-spirals problem, and output pattern for one network trained with Cascade-Correlation.
Benchmark Tests: 2-Spiral Problem
BENCHMARK TESTS: 2-SPIRAL PROBLEM
**Benchmark Tests: N-Parity Problem**

- **N-Parity Problem:**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b₃ = 0:</td>
<td>b₃ = 1:</td>
<td></td>
</tr>
<tr>
<td>N=2:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N=3:</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

N=4: ...
Benchmark Tests: N-Parity Problem

Benchmark Results:

<table>
<thead>
<tr>
<th>N</th>
<th>Cases</th>
<th>Hidden Units</th>
<th>Average Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>2–3</td>
<td>142</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>3</td>
<td>161</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>4–5</td>
<td>292</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>4–5</td>
<td>357</td>
</tr>
</tbody>
</table>

N = 10:

<table>
<thead>
<tr>
<th>Train Cases</th>
<th>Test Cases</th>
<th>Hidden Units</th>
<th>Train Epochs</th>
<th>Test Errs</th>
<th>% Errs</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>512</td>
<td>4</td>
<td>282</td>
<td>9</td>
<td>1.8%</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
<td>7</td>
<td>551</td>
<td>30</td>
<td>5.8%</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
<td>7</td>
<td>491</td>
<td>32</td>
<td>6.2%</td>
</tr>
<tr>
<td>512</td>
<td>512</td>
<td>5</td>
<td>409</td>
<td>14</td>
<td>2.7%</td>
</tr>
<tr>
<td>256</td>
<td>768</td>
<td>4</td>
<td>382</td>
<td>111</td>
<td>14.4%</td>
</tr>
<tr>
<td>256</td>
<td>768</td>
<td>4</td>
<td>362</td>
<td>90</td>
<td>11.7%</td>
</tr>
<tr>
<td>256</td>
<td>768</td>
<td>4</td>
<td>276</td>
<td>55</td>
<td>7.2%</td>
</tr>
<tr>
<td>256</td>
<td>768</td>
<td>4</td>
<td>311</td>
<td>49</td>
<td>6.4%</td>
</tr>
</tbody>
</table>
CLOSING REMARKS

- First ‘complex’ network architecture we’ve seen
- First network to dynamically add new hidden units & layers
- Paper was published nearly 2 decades ago; progress?