Overview

- SOM demo
- Recurrent networks
- Genetic algorithms
- Course evaluation

SOM Demo: Traveling Salesman Problem

Using Fröhlich’s SOM applet:

- 1D SOM map ($1 \times n$, where $n$ is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 8.
- Stop when radius < 0.001.
- Try 50 nodes, 20 input points.

Click on [Parameters] to bring up the config panel. After the parameters are set, click on [Reset] in the main applet, and then [Start learning].

SOM Demo: Space Filling in 2D

Using Fröhlich’s SOM applet:

- 1D SOM map ($1 \times n$, where $n$ is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 100.
- Stop when radius < 0.001.
- Try 1000 nodes, and 1000 input points.

Jochen Fröhlich’s Neural Networks with JAVA page:

http://rfhs8012.fh-regensburg.de/saj39122/jfroehl/diplom/e-index.html

Check out the Sample Applet link.
SOM Demo: Space Filling in 3D

Using Fröhlich’s SOM applet:

- 2D SOM map \((n \times n\), where \(n\) is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 10.
- Stop when radius < 0.001.
- Try \(30 \times 30\) nodes, and 500 input points. Limit the \(y\) range to 15.

Also try \(50 \times 50\), 1000 input points, and 16 initial radius.

Recurrent Networks

Connection graph can contain cycles, e.g. reciprocal connections: i.e. not strictly feed-forward.

- Statistical mechanics based models (associative or content-addressable memory): Hopfield network, Boltzman machines, etc.
- Sequence encoding: Simple Recurrent Network, etc.
- Other biologically motivated networks: laterally connected self-organizing maps, etc.

Simple Recurrent Network (Elman Network)

- Sequence encoding.
- Hidden layer activation from previous time step used as input.
- Use standard back-propagation learning.

SRN Example

Example input and target sequence: output 1 when two 1s appear in a row.

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Target</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Evolution as a problem solving strategy:

- population of solutions, where each gene represent an individual
- selection based on fitness function: survival of the fittest
- mating (cross-over) and reproduction
- random mutation

Properties of Genetic Algorithms

- each gene encodes a solution
- similar to hill-climbing search
- parallel search
- works for both immediate or delayed reward

Designing a GA Solution to a Problem

There are many different issues:

- What is the fitness function?
- How is an individual represented (how to encode)?
- How are individuals selected?
- How do individuals reproduce?

Issues in GA

- Diversity: letting only very successful ones reproduce can seriously reduce the gene pool and an epidemic can wipe out the whole population.
- Cross-over strategy: success depends on how genes are encoded (or represented).
- Not too much theoretical understanding about why it works so well.
GA Demo

- Generation of melodies
- Gaming AI: harvesters and predators

Key Points

- Try out the effects of different parameters, network size, 1D or 2D map, neighborhood radius, etc.
- Simple recurrent networks: how can it encode sequences, how is it different from standard backprop and who similar is it?
- Genetic algorithms basics.

Next Time