Weight Initialization for Backpropagation with Genetic Algorithms

Anoop Kunchukuttan
Sandeep Limaye
Ashish Lahane
Seminar Outline

- Weight Initialization Problem (Part – I)
- Introduction to Genetic Algorithms (Part – II)
- GA-NN based solution (Part – III)
Problem Definition

Part - I
Backpropagation

- Feed-forward neural network training method
- Minimizes Mean Square Error
- Gradient descent
- Greedy Method
Weight Initialization Problem

- Backpropagation is sensitive to initial conditions [KOL-1990]
- Initial conditions such as
  - initial weights
  - learning factor
  - momentum factor
Effect of Initial Weights

- Can get stuck in local minima
- May converge too slowly
- Achieved generalization can be poor
Solution

- Use of global search methods to get the final weights, such as: Genetic algorithms, Simulated Annealing etc.
- Use global search methods to get closer to global minimum & then run local search (BP) to get exact solution

We will concentrate on the latter method using Genetic algorithms
Motivation

- Inspired by evolution in living organisms

- “Survival of the fittest”

- “Fitter” individuals in the population preferred to reproduce to create new generation
Other algorithms inspired by nature

- Neural networks – neurons in brain
- Simulated annealing – physical properties of metals
- GA, NN have partly overlapped application areas – pattern recognition, ML, image processing, expert systems
Basic Steps

- Coding the problem
- Determining “fitness”
- Selection of parents for reproduction
- Recombination to produce new generation
### Coding

**Genetic Algorithms**

<table>
<thead>
<tr>
<th>GA in nature</th>
<th>GA in CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genes</td>
<td>Parameters of solution</td>
</tr>
<tr>
<td><strong>Chromosome = collection of genes</strong></td>
<td><strong>Chromosome = concatenated parameter values</strong></td>
</tr>
<tr>
<td>Genotype = set of parameters represented in the chromosome</td>
<td></td>
</tr>
<tr>
<td>Phenotype = Finished “construction” of the individual (values assigned to parameters)</td>
<td></td>
</tr>
</tbody>
</table>
Generic Model for Genetic Algorithm (1)
BEGIN /* genetic algorithm */
    generate initial population
    WHILE NOT finished DO
    BEGIN
        compute fitness of each individual
        IF population has converged THEN
            finished := TRUE
        ELSE
            /* reproductive cycle */
            apply genetic operators to produce children
        END
    END
END
END
Determining fitness

- Fitness function
- Varies from problem to problem
- Usually returns a value that we want to optimize
- Example: Strength / Weight ratio in a civil engineering problem
- Unimodal / Multimodal fitness functions – Multimodal: several peaks
Selection

- Individuals selected to recombine and produce offspring
- Selection of parents based on “Roulette Wheel” algorithm
- Fitter parents may get selected multiple times, not-so-fit may not get selected at all
- Child can be less fit than parents, but such a child will probably “die out” without reproducing in the next generation.
Roulette Wheel selection

Individual 1
Individual 2
Individual 3
Individual 4
Individual 5
Individual 6
Individual 7
Individual 8

Genetic Algorithms
Selection - variations

- Fitness-based v/s rank-based
Recombination - Crossover

- Crossover probability (typically $0.6 < CP < 1$)
- If no crossover, child is identical to parent
Crossover – variations (1)

- One-point crossover

| Parent 1: 001010011 | 010100101010101110 |
| Parent 2: 010101110 | 1010101101110101 |
| Child: 001010011 | 1010101101110101 |
Crossover – variations (2)

- Two-point crossover

Parent 1: 001010011 | 01010010 | 10101110

Parent 2: 010101110 | 10101011 | 01110101

Child: 001010011 10101011 01110101
Crossover – variations (3)

- Uniform crossover

Parent 1: 00000000000000000000000000000000
Parent 2: 11111111111111111111111111
Child: 1001001101110100100100111001
Recombination - Mutation

- Crossover – rapidly searches a large problem space
- Mutation – allows more fine-grained search
Convergence

- Gene convergence: when 95% of the population has the same value for that gene
- Population convergence: when all the genes reach convergence
Design Issues

- Goldberg’s principles of coding (building block hypothesis):
  - Related genes should be close together
  - Little interaction between genes
- Is it possible, (and if yes, how) to find coding schemes that obey the principles?
- If not, can the GA be modified to improve its performance in these circumstances?
GA - Plus points

- Robust: Wide range of problems can be tackled
- “Acceptably good” solutions in “acceptably quick” time
- Explore wide range of solution space reasonably quickly
- Combination of Exploration and Exploitation
- Hybridizing existing algorithms with GAs often proves beneficial
GA - shortcomings

- Need not always give a globally optimum solution!

- Probabilistic behavior
Weight Initialization using GA

Part - III
Applying GA to Neural Nets

- Optimizing the NN using GA as the optimization algorithm [MON-1989]
- Weight initialization
- Learning the network topology
- Learning network parameters
Why use genetic algorithms?

- Global heuristic search (Global Sampling)
- Get close to the optimal solution faster
- Genetic operators create a diversity in the population, due to which a larger solution space can be explored.
The Intuition

- Optimizing using only GA is not efficient.
  - Might move away after getting close to the solution.
- Gradient Descent can zoom in to a solution in a local neighbourhood.
- Exploit the global search of GA with local search of backpropagation.
- This hybrid approach has been observed to be efficient.

Weight Initialization using GA
The Hybrid (GA-NN) Approach

GA provides ‘seeds’ for the BP to run.
Basic Architecture

Random encoded weights

Next generation candidates

Evaluate

Fitness Function

Fittest Candidates from GA for BP

Final weights

Weight Initialization using GA
Considerations

- Initially high learning rate
  - To reduce required number of BP iterations
- Learning rate and number of BP iterations as a function of fitness criteria:
  - To speed up convergence
  - To exploit local search capability of BP
- Tradeoff between number of GA generations and BP iterations.

Weight Initialization using GA
Q. What to encode?
Ans. Weights.

Q. How?
Ans. Binary Encoding

Weight Initialization using GA
Binary Weight Encoding

Weight Initialization using GA
Issues

- Order of the weights
- Code length
- Weights are real valued

Weight Initialization using GA
Code Length \( \ell \)

- Code length determines
  - resolution
  - precision
  - size of solution space to be searched

- Minimal precision requirement \( \rightarrow \) minimum \( \ell \)
  \( \rightarrow \) limits the efforts to improve GA search by reducing gene length

Weight Initialization using GA
Real Value Encoding Methods

- Gray-Scale
- DPE (Dynamic Parameter Encoding)
Gray-Code Encoding

- Gray-code: Hamming distance between any two consecutive numbers is 1
- Better than or at least same as binary encoding
- Example

<table>
<thead>
<tr>
<th>Number</th>
<th>Gray Codes</th>
<th>Binary Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>111</td>
</tr>
</tbody>
</table>

Weight Initialization using GA
Real-Value Encoding Using Gray-code

Weight Initialization using GA
$T(i) = a + (b - a) \frac{i + X}{2^\ell}$

[a,b) – the interval
i – parameter value read from chromosome
\ell – code length
X – random variable with values from [0,1)
Drawbacks

- Too large search space → large ‘\( \ell \)’ → higher resolution → high precision → long time for GA to converge

- Instead: search can proceed from coarse to finer precision: i.e. DPE
DPE (Dynamic Parameter Encoding) [SCH-1992]

- use small ‘l’ → coarse precision → search for most favored area → refine mapping → again search → iterate till convergence with desired precision achieved

- Zooming operation

Weight Initialization using GA
Weight Initialization using GA
Fitness Function

- Mean Square Error
- Rate of change of Mean Square Error
Genetic operators – problem-specific variations (1)

- **UNBIASED-MUTATE-WEIGHTS**
  - With fixed p, replace weight by new randomly chosen value

- **BIASED-MUTATE-WEIGHTS**
  - With fixed p, add randomly chosen value to existing weight value – biased towards existing weight value – exploitation
Genetic operators – problem-specific variations (2)

- **MUTATE-NODES**
  - Mutate genes for incoming weights to \(n\) non-input neurons
  - Intuition: such genes form a logical subgroup, changing them together may result in better evaluation

- **MUTATE-WEAKEST-NODES**
  - Strength of a node =
    - \((n/w\) evaluation) – \((n/w\) evaluation with this node “disabled” i.e. its outgoing weights set to 0)
  - Select \(m\) weakest nodes and mutate their incoming / outgoing weights
  - Does not improve nodes that are already “doing well”, so should not be used as a single recombination operator

**Weight Initialization using GA**
Genetic operators – problem-specific variations (3)

- CROSSLEROVER-WEIGHTS
  - Similar to uniform crossover

- CROSSLEROVER-NODES
  - Crossover the incoming weights of a parent node together
  - Maintains “logical subgroups” across generations
Genetic operators – problem-specific variations (4)

- OPERATOR-PROBABILITIES
  - Decides which operator(s) from the operator pool get applied for a particular recombination
  - Initially, all operators have equal probability
  - An adaptation mechanism tunes the probabilities as the generations evolve
Computational Complexity

- Training time could be large due to running multiple generations of GA and multiple runs of BP.
- \[ \text{Total Time} = \text{Generations} \times \text{Population Size} \times \text{Training Time} \]
- There is a tradeoff between the number of generations and number of trials to each individual

Weight Initialization using GA
CONCLUSION & FUTURE WORK

- Hybrid approach promising
- Correct choice of encoding and recombination operators are important.

- As part of course project
  - Implement the hybrid approach
  - Experiment with different learning rates and number of iterations and adapting them
REFERENCES (1)


REFERENCES (2)

  http://citeseer.ist.psu.edu/belew90evolving.html

  http://citeseer.ist.psu.edu/kolen90back.html

  http://vishnu.bbn.com/papers/ijcai89.pdf
REFERENCES (3)

