Swarm Intelligence Seminar: Training Recurrent Neural Networks with Breeding Swarm

Xiwen Cheng & Barry van Veen

LIACS

November 16, 2010

Introduction: PSO

- inspired by behavior of bird flocking
- population based stochastic optimization technique
- proposed by Kennedy and Eberhart in 1995

- population with particles (birds)
- particle represents solution in search space
- particle consists of position and velocity
- particles position is updated

Basic algorithm

$t \leftarrow 0$;
randomly initialize $V(t)$;
randomly initialize $P(t)$;
evaluate $P(t)$;
update pBest;
update nBest;
repeat
    $V(t+1) \leftarrow updateVelocity(V(t))$;
    $P(t+1) \leftarrow P(t) + V(t+1)$;
    evaluate $P(t+1)$;
    update pBest;
    update nBest
    $t \leftarrow t+1$;
until *stop requirement*

Basic algorithm

$$\begin{aligned} V(t+1) = \quad & w \times V(t) \\ & + c_1 \times rand_1() \times (pBest - P(t)) \\ & + c_2 \times rand_2() \times (nBest - P(t)) \\ P(t+1) = \quad & P(t) + V(t+1) \end{aligned}$$

$pBest$	best solution found so far by a particle
$nBest$	best solution found by n neighbourhood
w	inertia weight (damping weight)
c_1, c_2	social parameters
$rand_1$, $rand_2$	normal distributed random values
$vMax$	maximum velocity

Update function

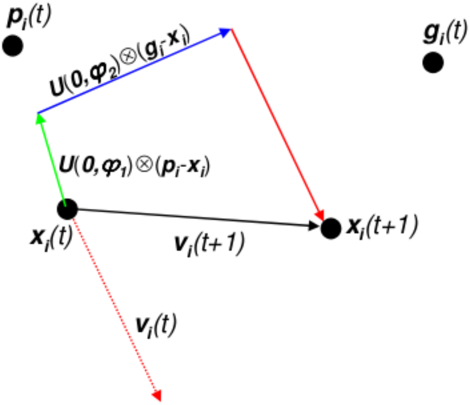


Figure: PSO update function

Modifications to the algorithm

- discrete version
- multiobjective optimization
- constraint optimization
- dynamic environments

Applications of the algorithm

- neural network training
- parameter optimization
- feature selection
- interaction with other algorithms

Introduction: GA

- inspired by evolution
- population based stochastic optimization technique
- proposed by Holland in 1970's

- population of bitstrings
- bitstrings encode solutions
- individual is changed by crossover and mutation
- population is changed by creating offspring and selecting individuals

Basic algorithm

$t \leftarrow 0;$
initialize $P(t);$
evaluate $P(t);$
repeat
  $P'(t) \leftarrow \textit{select-mates}(P(t));$
  $P''(t) \leftarrow \textit{crossover}(P'(t), p_c);$
  $P'''(t) \leftarrow \textit{mutation}(P''(t), p_m);$
  evaluate $P'''(t);$
  $P(t+1) \leftarrow P'''(t);$
  $t \leftarrow t + 1;$
until *stop requirement*

Operators

- Selection
  - proportional
  - tournament
- Crossover
  - single point
  - uniform
  - blended-α
- Mutation
- Elitism

```
selection:
0000000000
1111111111

crossover:
0000011111
1111100000

mutation:
1000011111
```

Blended-α crossover

select parents $x(t), y(t)$;
for $i = 1$ to n **do**
$\quad \delta_i \leftarrow |x_i(t) - y_i(t)|$;
$\quad min_i \leftarrow min(x_i, y_i)$
$\quad max_i \leftarrow max(x_i, y_i)$
$\quad u_x, u_y \leftarrow$ uniform random number on interval $[min_i - \alpha\delta_i, max_i + \alpha\delta_i]$;
$\quad x_i(t+1) = u_x$;
$\quad y_i(t+1) = u_y$;
end for

α was set to 0.1 and
Crossover rate was 0.6

Comparison PSO / GA

Similarities

- Population based
- Global search methods
- PSO update function performs actions similar to crossover, mutation
  - Update is depending on *pBest* and *nBest* and combines these two
  - Update function has random factors to increase exploration

Comparison PSO / GA

Differences

- PSO has no selection mechanism, GA has selection and elitism
- PSO has directional updates, GA has omnidirectional mutation
- PSO mixes neighbours, GA mixes "random" individuals
- PSO is more ergodic than a GA

Introduction: BS

Combining ideas of:

- PSO: velocity and position update rules
- GA: selection, crossover and mutation

Generic Hybrid Algorithm

Pseudo algorithm

$t \leftarrow 0;$
initialize $P(t), V(t);$
evaluate $P(t);$
repeat
$\quad P_{elite}(t) \leftarrow copyBest(P(t), N_{elite});$
$\quad P_{pso}(t) \leftarrow select_1(P(t), (N - N_{elite}) * \Psi);$
$\quad V'(t) \leftarrow updateVelocity(V(t), P_{pso}(t));$
$\quad P'_{pso}(t) \leftarrow updatePosition(P_{pso}(t), V'_{pso}(t));$
$\quad P_{ga}(t) \leftarrow select_2(P(t), (N - N_{elite}) * (1 - \Psi));$
$\quad P'_{ga}(t) \leftarrow crossover(P_{ga}(t), p_c);$
$\quad P''_{ga}(t) \leftarrow mutation(P'_{ga}(t), p_m);$
$\quad P(t+1) \leftarrow P_{elite} \cup P'_{pso} \cup P''_{ga};$
$\quad V(t+1) \leftarrow V'(t);$
$\quad$ *evaluate* $P(t+1);$
$\quad t \leftarrow t + 1;$
until *stop requirement*

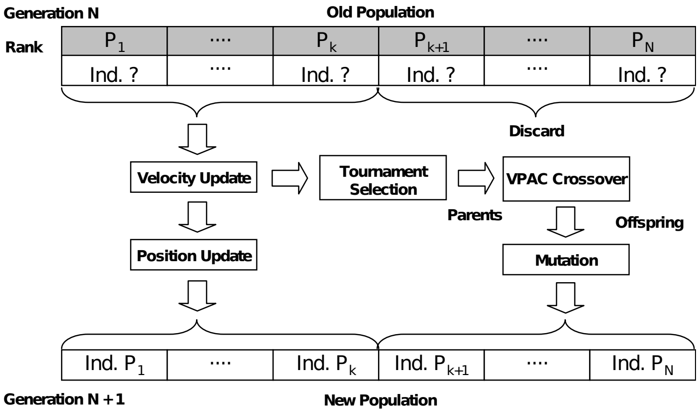
Generic Hybrid Algorithm

Parameters

N	Number of individuals
N_{elite}	Number of elites
Ψ	*Breeding ratio*, proportion that undergoes breeding; [0.0:1.0]
$select_1, select_2$	Two selection methods, may be different
p_c, p_m	Crossover rate and mutation rate respectively

Breeding Swarm Algorithm (Settles & Soule)

Flow



Breeding Swarm Algorithm (Settles & Soule)

Parameters

Designed such that:

- GA performs global search
- PSO performs local search

Standard velocity and position update rules are used. However:

N_{elite}	$= 0$
Ψ	$= 0.5$, even distribution between PSO and GA
$select_1 = select_2$	= tournament selection with size of 2
crossover	= *Velocity Propelled Averaged Crossover*
mutation	= Gaussian with mean 0 and variance reduced linearly each generation from 1.0 to 0.0
inertia weight	= reduced linearly each generation from 0.7 to 0.4
social parameter	$= 2$, $c1$ and $c2$ in position update
V_{max}	$= \pm 1$

Velocity Propelled Averaged Crossover (VPAC)

$$c_1(x_i) = \frac{p_1(x_i) + p_2(x_i)}{2.0} - \varphi_1 p_2(v_i) \quad (1)$$

$$c_2(x_i) = \frac{p_1(x_i) + p_2(x_i)}{2.0} - \varphi_2 p_1(v_i) \quad (2)$$

$c_1(x_i), c_2(x_i)$	Positions of childrens in dimension i
$p_1(x_i), p_2(x_i)$	Positions of parents in dimension i
$p_1(v_i), p_2(v_i)$	Velocities of parents in dimension i
φ_1, φ_2	Uniform random variable in [0.0:1.0]

Increases diversity: accelerate away from parent's current direction

Artificial Neural Network

- Inspired by Biological neurons
- Universal approximators for non-linear functions
- Consists of many layers: an input, multiple hidden and an output

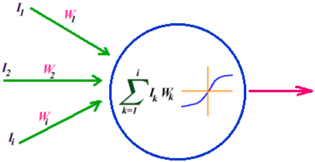


Figure: Single neuron

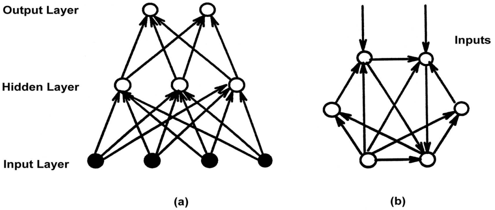


Figure: Feedforward and Hopfield (RNN)

Recurrent Neural Network

We're considering a *discrete-times, multi-layered and strongly connected* recurrent neural network

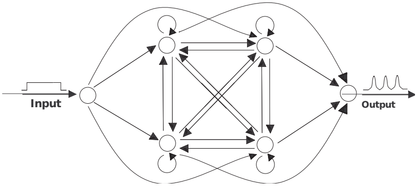


Figure: Strongly connected RNN

Each node uses a symmetric sigmoidal activation function:

$$f(x) = \frac{2}{1 + exp(-\beta x)} - 1 \tag{3}$$

Where $\beta = 1$ is the slope parameter

Test problem

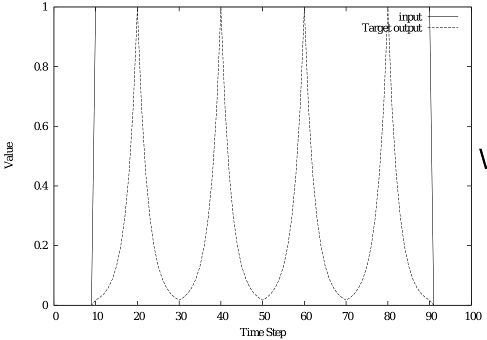


Figure: Target: pulsed output signal

Error function:

$$E = \sum_{i=1}^{100} [(t_i - o_i)^2 + 2 \cdot (ts_i - os_i)^2]$$

Where:

t_i = target at time step i

o_i = output at time step i

ts_i = $t_i - t_{i-1}$ (target slope)

os_i = $o_i - o_{i-1}$ (output slope)

Slope component helps steer the training towards periodic behavior

Test parameters

Breeding Swarm

- Training algorithms: GA, PSO and BS
- Network weight $w_i = \vec{x}_i$ where $\vec{x} \in P(t)$
- Dimensionality: $(LN)^2 + 2LN$ where L is #(layers) and N is #(nodes)/layer
- Networks with variety of hidden layers: 1-9
- Number of nodes per hidden layer: 1-9
- Per run: 2000 generations and 50 individuals
- 50 runs per network
- Initial weights are randomly selected from [-0.5, 0.5] range
- Velocity is restricted to [-1.0, 1.0], for PSO and BS

Results (1)

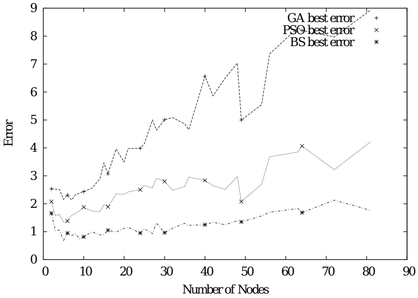


Figure 4: Average best error (averaged across all networks with the same number of nodes) values for each algorithm vs. total number of network nodes.

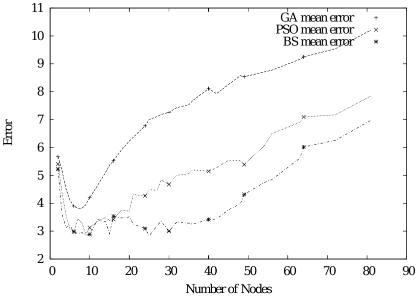
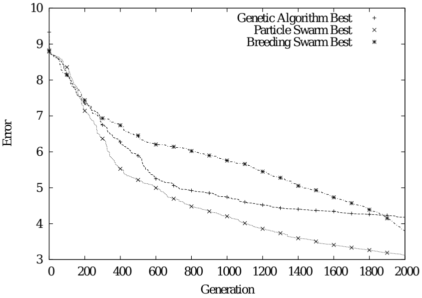
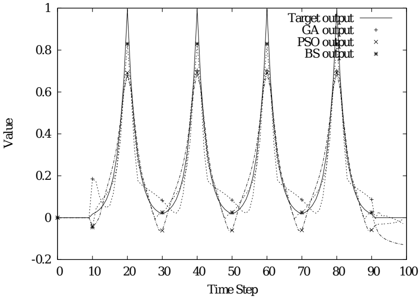


Figure 5: Average mean error (averaged across all networks with the same number of nodes) values for each algorithm vs. total number of network nodes.

Results (2)



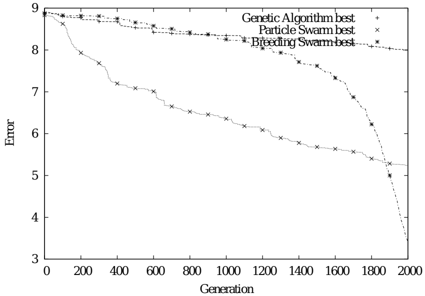
(a) Mean best error evaluation for each algorithm.



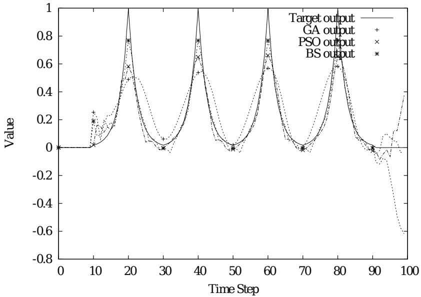
(b) Best output for each algorithm

Figure: 5 x 1 (35 weights)

Results (3)



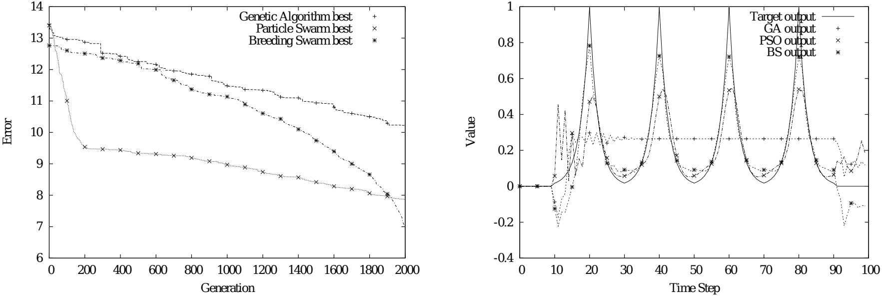
(a) Mean best error evaluation for each algorithm.



(b) Best output for each algorithm

Figure: 6 x 7 (1848 weights)

Results (4)



(a) Mean best error evaluation for each algorithm.

(b) Best output for each algorithm

Figure: 9 x 9 (6723 weights)

Conclusions

Breeding swarm:

- scales well on large networks
- produces as good as or better results than GA and PSO in most cases

Discussion

- In the BS articles non-standard versions of PSO and GA were used. Maybe these tests should be repeated with standard versions.
- The BS seems to perform good. What is still missing in the algorithm are self-adapting parameters.
- During the RNN training BS exhibits a remarkable behavior: Quickest improvements occur towards the end of the run.
- BS can evolve network topologies and weights simultaneously (VPAC).

References

This presentation is based on the research done by *Matthew Settles, Terrence Soule and Paul Nathan*.

The list of resources consulted are:

- Breeding Swarms: A GA/PSO hybrid – *M. Settles, T. Soule*
- Breeding Swarms: A New Approach to Recurrent Neural Network Training – *M. Settles, P. Nathan, T. Soule*
- Recent Advances in Particle Swarm – *X. Hu, Y. Shi, R. Eberhart*
- Comparison between Genetic Algorithms and Particle Swarm Optimization – *R. Eberhart, Y. Shi*
- Figures on NN were borrowed from the Neural Network Course, fall 2010.
- Figure on PSO update function was borrowed from the Natural Computing Course, fall 2010.