Evolving Artificial Neural Networks

Cole Rottweiler

11/11/03

CS152

Evolution in ANNs

- EA’s => stochastic search algorithms
  - Evolution Strategies (ES)
  - Evolutionary Programming (EP)
  - Genetic Algorithms (GA)

- Connection weight training
- Architecture design
  - overall connectivity
  - transfer functions
- Learning rule adaptation

Population Based Search Strategy

1. Create initial random population
2. While termination criterion not satisfied
   a) evaluate each member of the population
   b) choose parents based on fitness function
   c) apply search operators to parents and create offspring for next generation

EA’s versus GDA’s

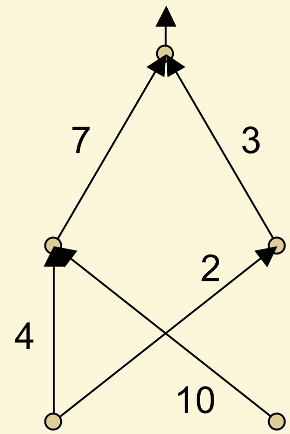
- EA => global search, avoids gradient descent problems
  - work well when many local minima exist
  - still work when GD info is difficult to compute
- Still… problem dependent

Weight Training

- Weight training for GDA’s:
  - minimization of an error function (MSE)
- Weight training for EA’s
  - define any error function, possibly undifferentiable
  - choose a weight representation
  - choose an evolutionary process

Connection Weight Encodings (binary)

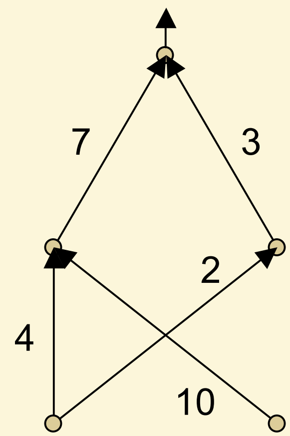
- Binary encoding is canonical GA example
  - each connection weight = length *n* bit string
  - ANN = concatenation of all bit string



= 0100 1010 0010 0000 0111 0011

- Maintain feature detectors
  - keep hidden node weights close to each other
    - be careful with crossover
  - Pros: ease of use, very simple search operators
  - Cons: representation precision vs. efficiency

Connection Weight Encodings (reals)

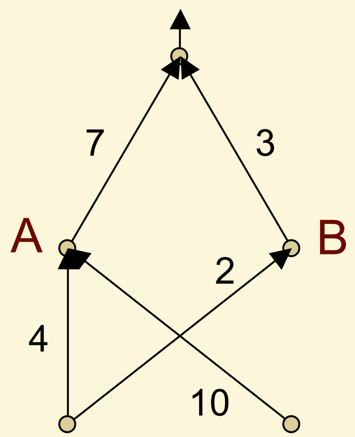


= (4.0, 10.0, 2.0, 0.0, 7.0, 3.0)

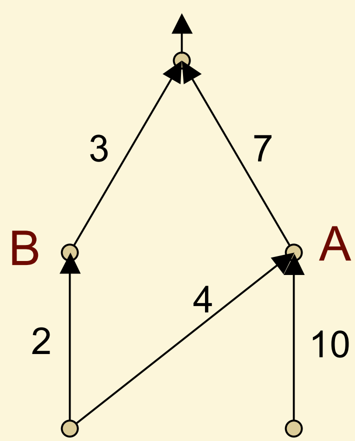
- More difficult to design
  - must create new search operators
- Often better performance
  - several results show strong competition with BP

Permutation Problem

- Different genotypes can encode the same phenotype



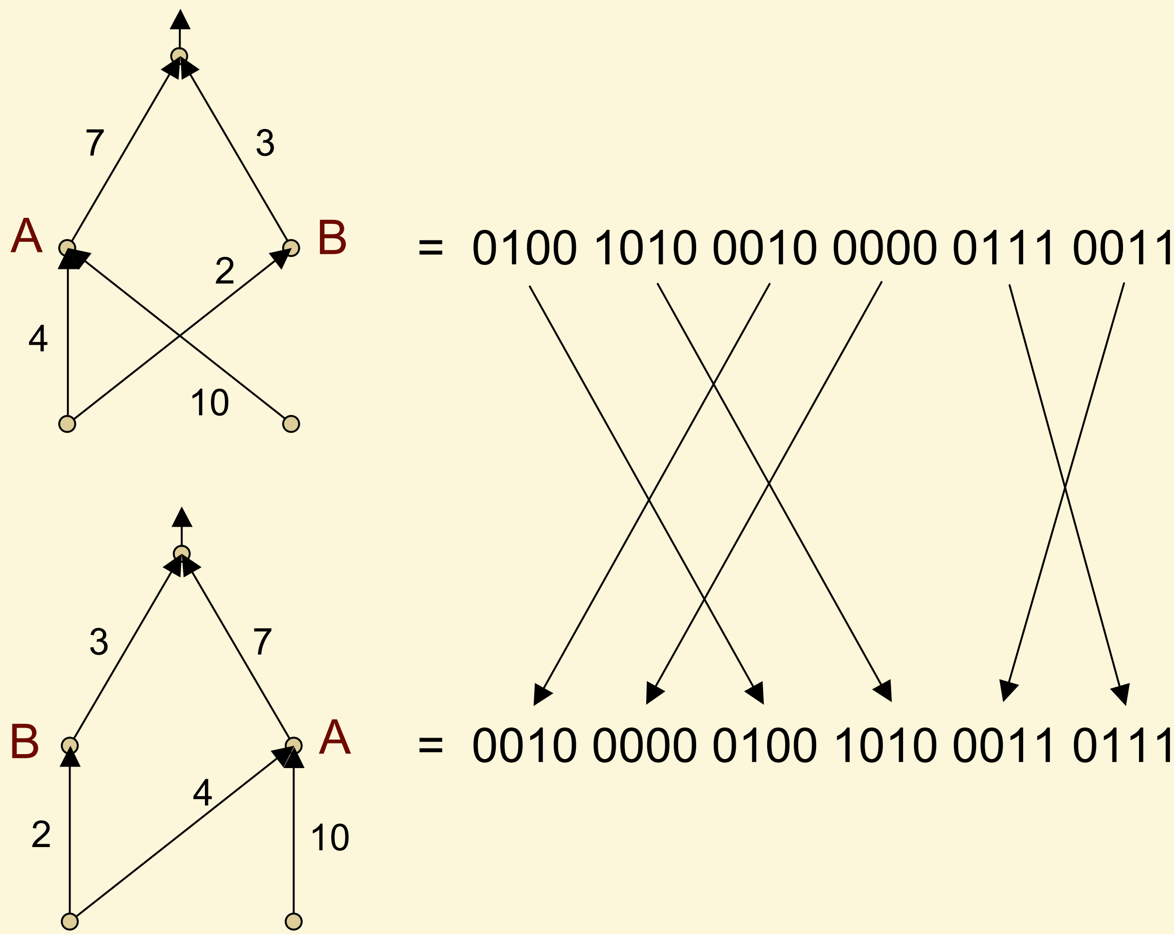
= 0100 1010 0010 0000 0111 0011



= 0010 0000 0100 1010 0011 0111

Permutation Problem

- Different genotypes can encode the same phenotype



EA Training vs. GD Training (again)

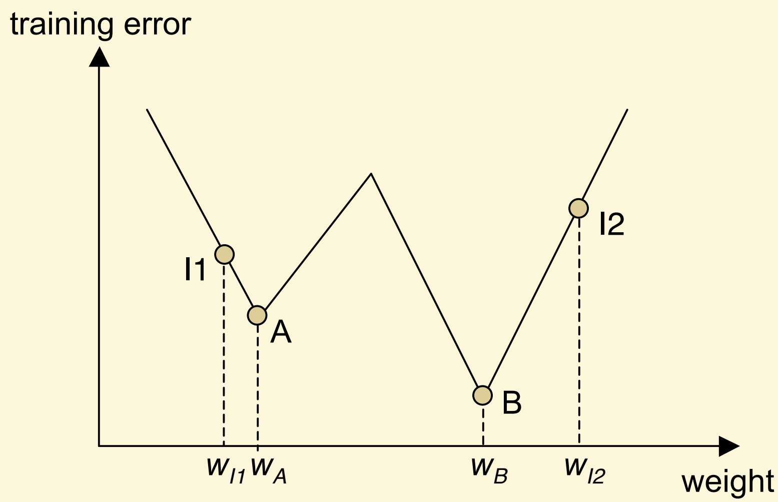
- ANN's => evolutionary applications with recurrent ANN's, higher order ANN's, and fuzzy ANN's
- General applicability
  - less human effort required
  - can easily include parameters to decrease complexity, apply weight decay, etc.

Speed and Reliability

- Optimized BP algorithms can be faster than EA's
- EA's much less sensitive to initial conditions
- BP algorithms often require several runs to avoid local minima
- BP fast for simple problems, often slower for larger problems
- Wide variety of results reported
    - depends on specific algorithms, problem, etc.

Hybrid Training

- EA’s => very good at global searches, less so at fine tuning
- Hybrid with a local search (BP) to fine tune
    - avoids local minima better
    - finds near optimal solutions



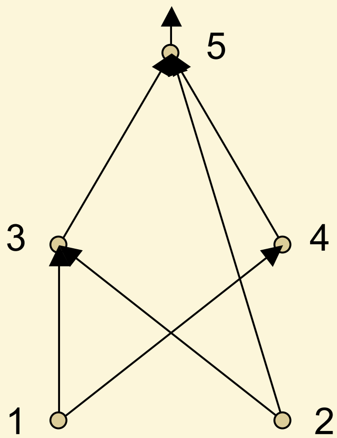
Evolving Architectures

- Architectures => connection weights and transfer functions
  - architecture defines much of processing ability
  - too small, limited transfer functions => inability to compute complex problems
  - too large, complex transfer functions => overfit data
- Trial and error often involved in human decisions
- Construction/destruction techniques still can get trapped in local minima

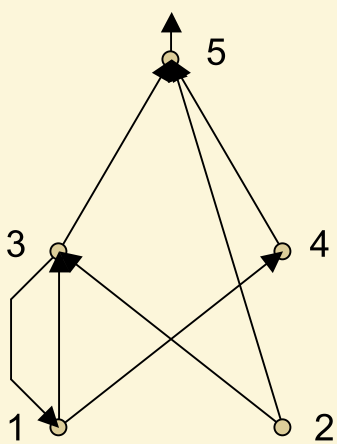
Evolving Architectures

- Search architecture space
  - infinitely large
  - nondifferentiable
  - complex, noisy, and deceptive surface
  - multimodal
- Choose performance criteria
  - lowest training error
  - lowest complexity
  - etc.
- How much architecture information should be encoded?

Direct Encoding



= = 0110 101 01 1



= = 00110 00100 10001 00001 01000

Population Based Search Strategy

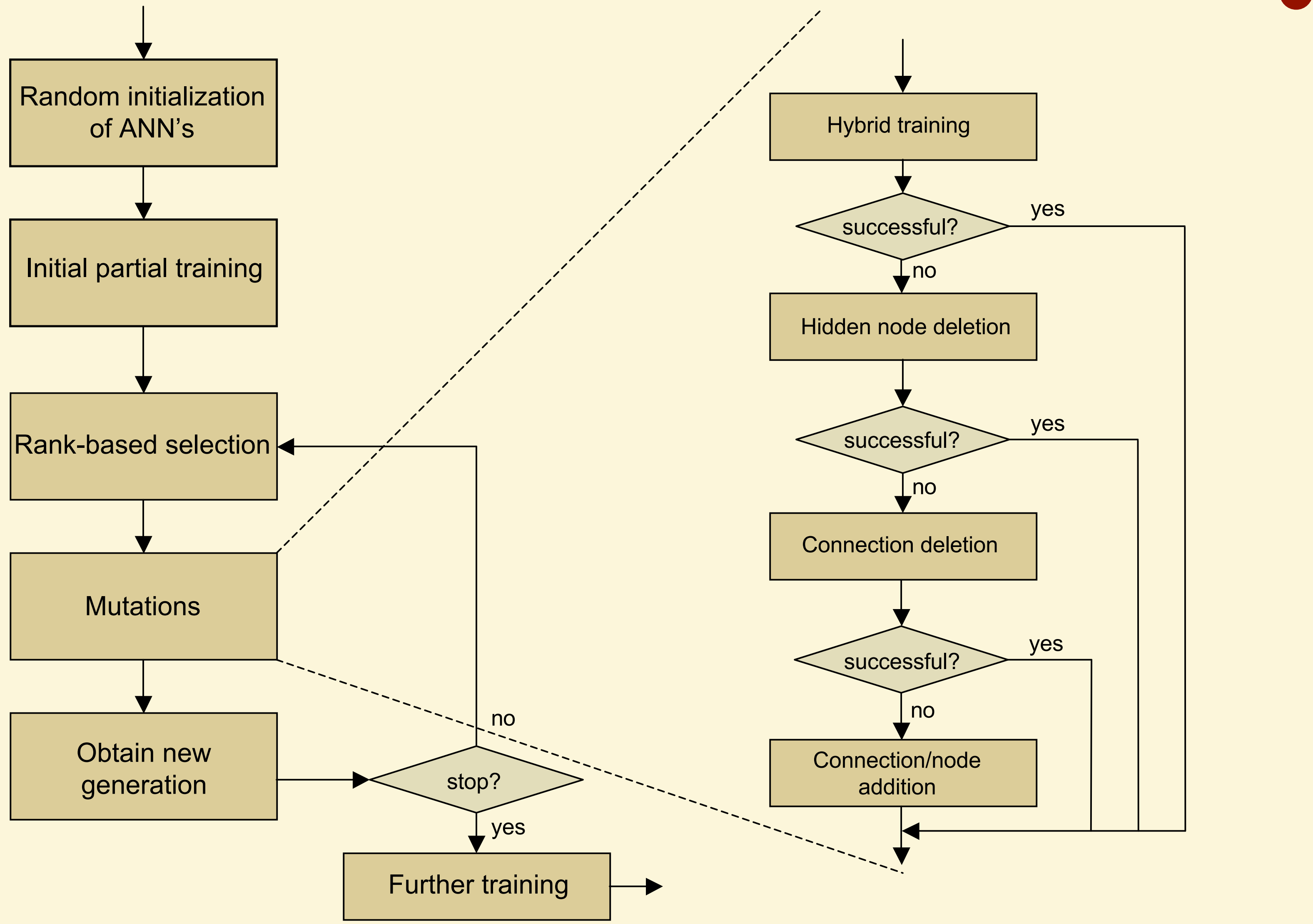
1. Decode each individual in this generation
2. Train each decoded ANN
3. Compute fitness of each individual
   a) use predetermined learning rules
   b) randomize initial weights
4. Choose most fit parents
5. Apply search operators to parents to create new generation

- Pros: choice of any fitness function; can increase generalization
- Cons: size considerations; noisy; permutation problem

Architecture/Weight Coevolution

- Evolving architecture alone can be problematic
  - noisy fitness evaluation
  - often requires many training runs to limit noise
- Coevolving => genotype → phenotype one-to-one mapping
- EPNet
  - automatic system
  - coevolves network architecture and connection weights

EPNet



Random initialization of ANN's
Initial partial training
Rank-based selection
Mutations
Obtain new generation
stop?
no
yes
Further training
Hybrid training
successful?
yes
no
Hidden node deletion
successful?
yes
no
Connection deletion
successful?
yes
no
Connection/node addition

Conclusions

- Three levels of ANN evolution:
    1. Connection weight evolution – lowest level, fastest
        - predetermined architecture, learning rules
    2. Architectural evolution – noisy if evolved alone
        - often coevolved with weights
    3. Learning rule evolution – noisy if evolved alone
        - less defined
- Order of last two depends on problem knowledge

Conclusions

- Connection weight evolution
  - global search
  - especially useful when gradient info is difficult to find
- Architectural evolution
  - automatically generate near-optimal architectures
  - direct encoding => fine tuning
  - indirect encoding => quickly find specific type of ANN
- Coevolution of weights and architecture
  - limits noise, improves overall results
- EA's global search => expensive, but can be beneficial

Literature

- X. Yao, “Evolving Artificial Neural Networks,” *Proceedings of the IEEE,* vol. 8, no. 9, pp. 1423-1447, Sept. 1999.

GA Applet

- http://www.rennard.org/alife/english/gavgb.html