Algorithmes Évolutionnaires (M2 MIAGE IA²)

Andrea G. B. Tettamanzi
Laboratoire I3S – Équipe SPARKS
andrea.tettamanzi@univ-cotedazur.fr

univ-cotedazur.fr

Séance 7
Neuroévolution

Introduction

- Artificial neural networks (ANNs) attempt to simulate by mathematical functions the processing units of the brain.
- The successful design of ANNs depends on several critical issues, regarding architecture and weight definition, training, data selecting, handling local minima, generalization, ecc.
- No easy answers to ANN definition and no standard design recipes exist for designing neural network for a given problem.
- Evolutionary Algorithms (EAs) are useful for complex optimization problems, where analytical solutions are difficult to obtain, and they find a global solution, vs local solution, over a domain.
- EANNs: biologically inspired computational model that uses EAs in conjunction with NNs. Adaptability to a dynamic environment is one distinct feature of EANNs.
- EANNs do not require any expert knowledge of the problem, and they can change network features appropriately without human intervention.

Soft Computing

- Tolerant of imprecision, uncertainty, and partial truth
- Adaptive
- Methodologies:
  - Evolutionary Algorithms
  - Neural Networks
  - Bayesian and Probabilistic Networks
  - Fuzzy Logic
  - Rough Sets

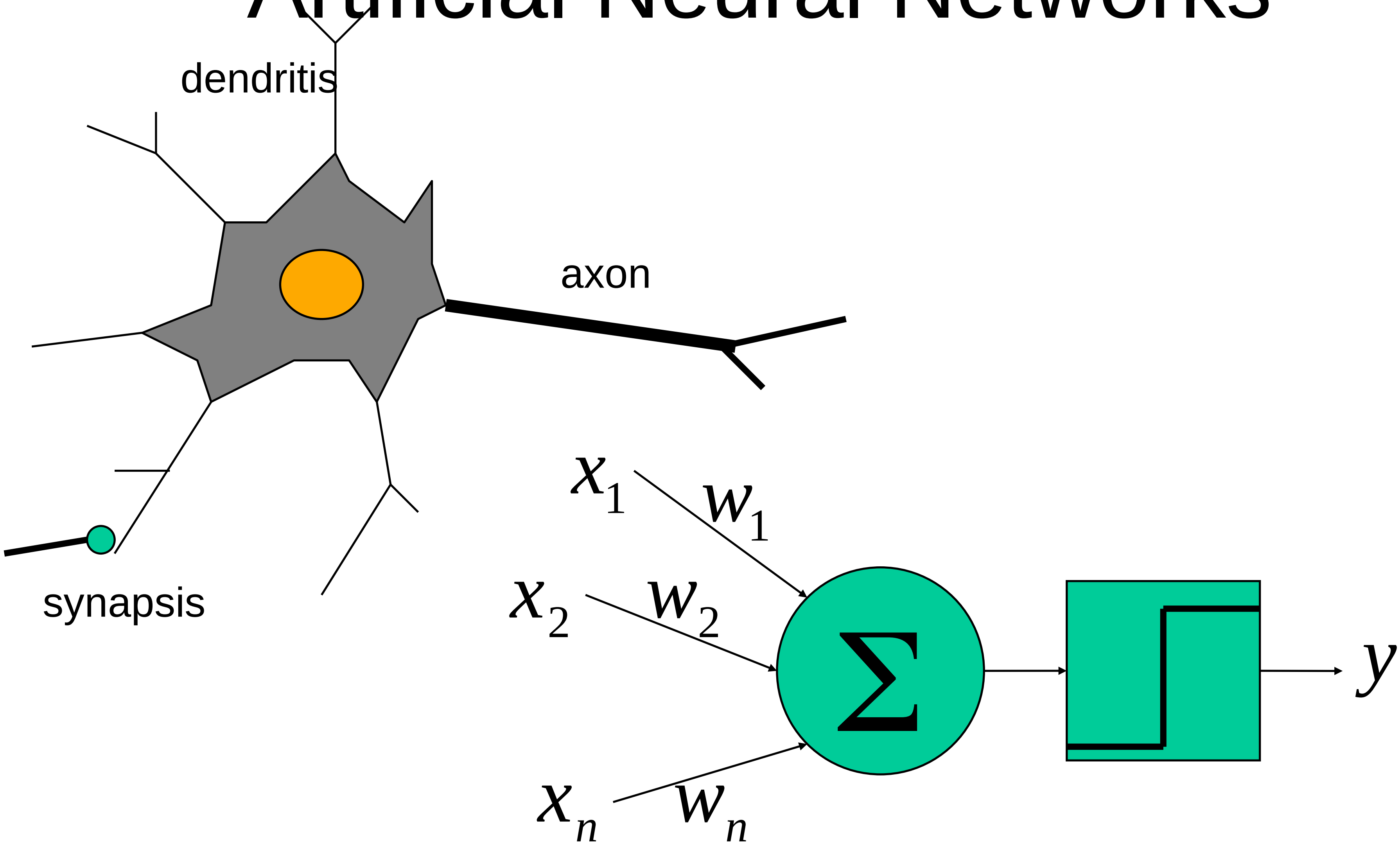
“Natural” Computing

- Bio-inspired - at different levels
- Evolutionary Computation:
  - lowest level: evolutionary mechanisms
  - adaptation at the population level
- Neural Computation:
  - intermediate level: learning and memory mechanisms
  - adaptation at the individual level
- Fuzzy Computation:
  - high level: approximate reasoning mechanisms
  - adaptation at the knowledge level

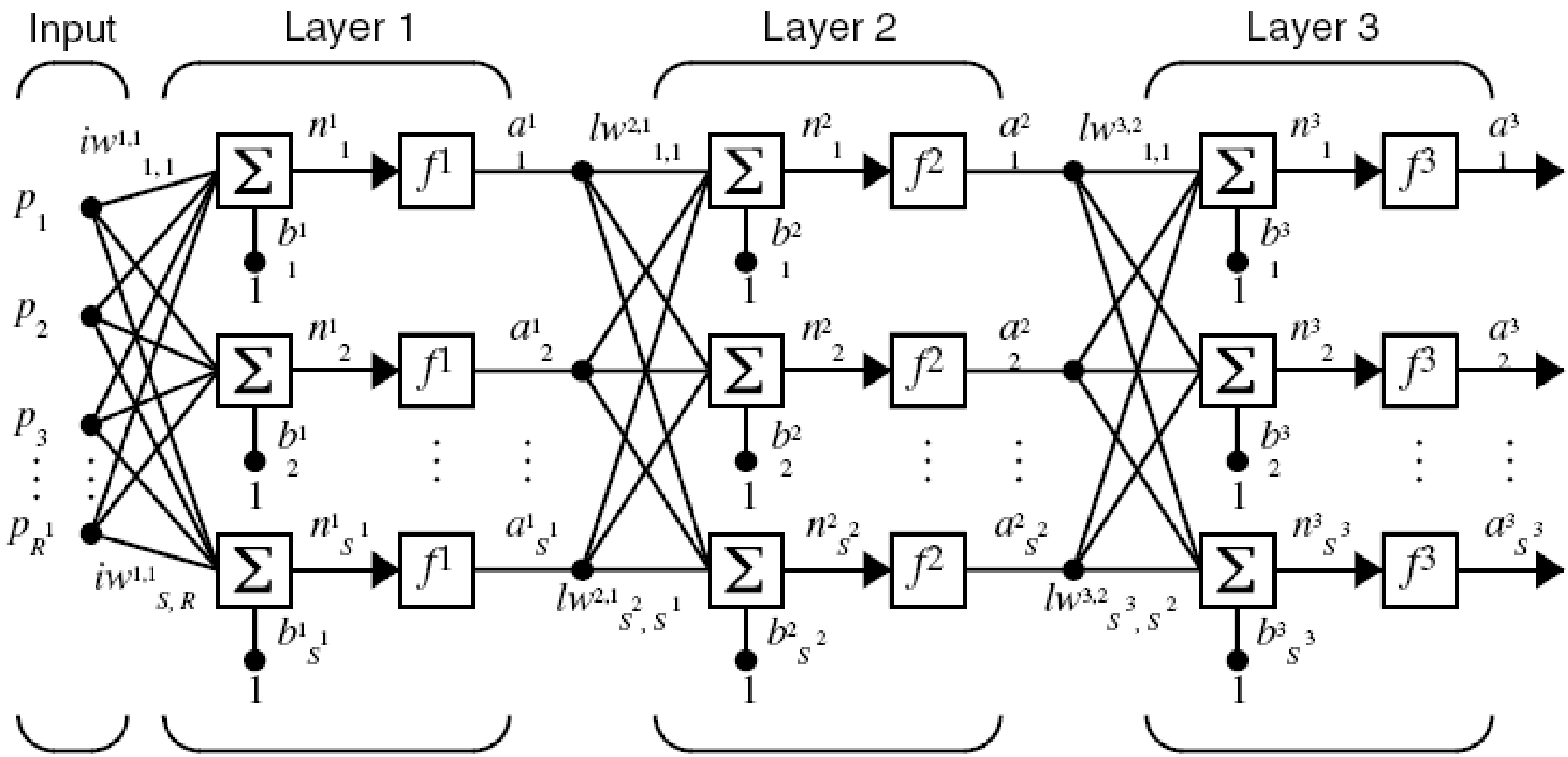
Artificial Neural Networks

- Highly simplified models of the brain.
- Combination of neurons and synaptic connections, capable of passing data through multiple layers.
- Pattern recognition and classification models.
- Improvement of the performance with “learning”.
- Adaptive system that self-organizes in order to approximate the solution.

Artificial Neural Networks



Feed-Forward Neural Network



$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\,\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

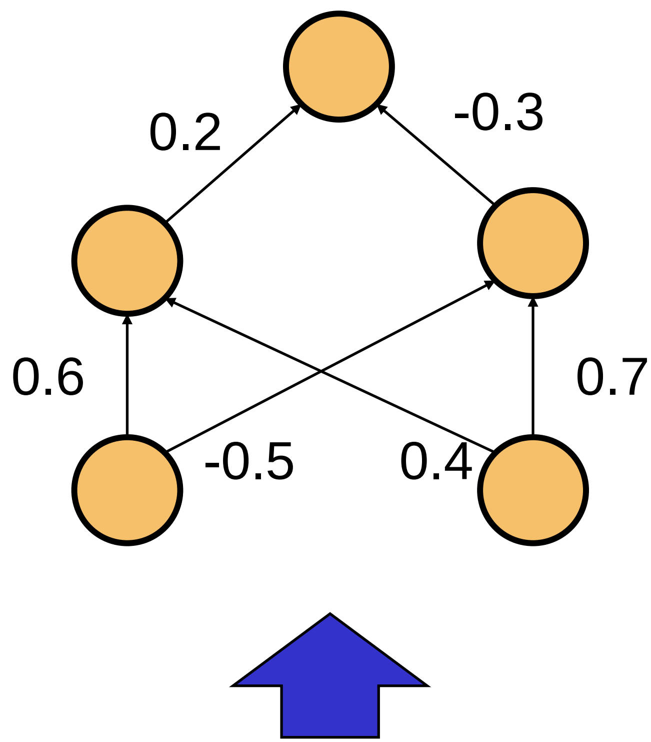
ANN Design and Optimization

- The successful design of artificial neural networks (ANNs) depends on several critical issues.
- No easy answers to ANN definition and no standard design recipes exist for designing neural network.
- Parameters can affect how easy a solution is to find.
- ANN design have to consider:
  - Architecture design (efficient number of nodes and connections)
  - Selecting data (availability and integrity of data)
  - Handling local minima (avoid to get stuck in local minima)
  - Generalization (ability to correctly map new inputs, not used in training phase)
  - Training process (adjusting the connection weights iteratively, so that learned ANNs can perform the desired task)

Possible Approaches

- Evolving weights for a network of predefined topology
- Evolving network structure
  - direct encoding
  - indirect encoding
- Evolving learning rules
- Transfer function optimization
- Input data selection
- Evolving topology and weights

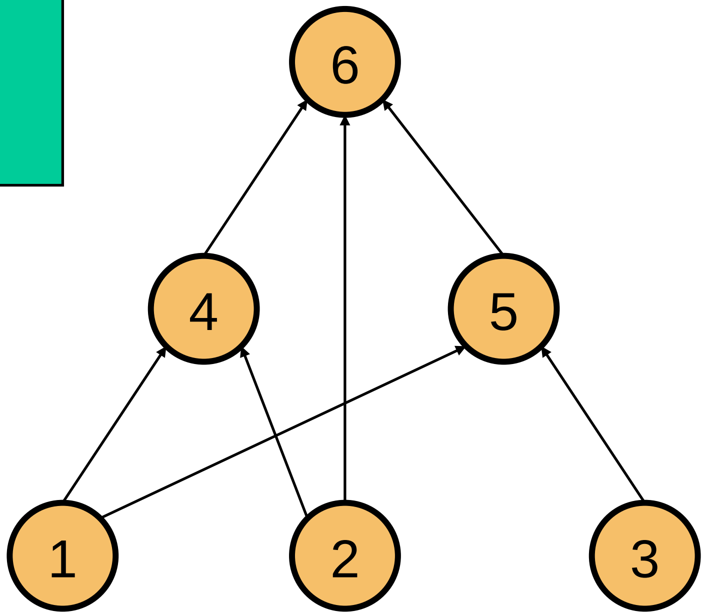
Evolving Weights (within a Pre-Defined Topology)



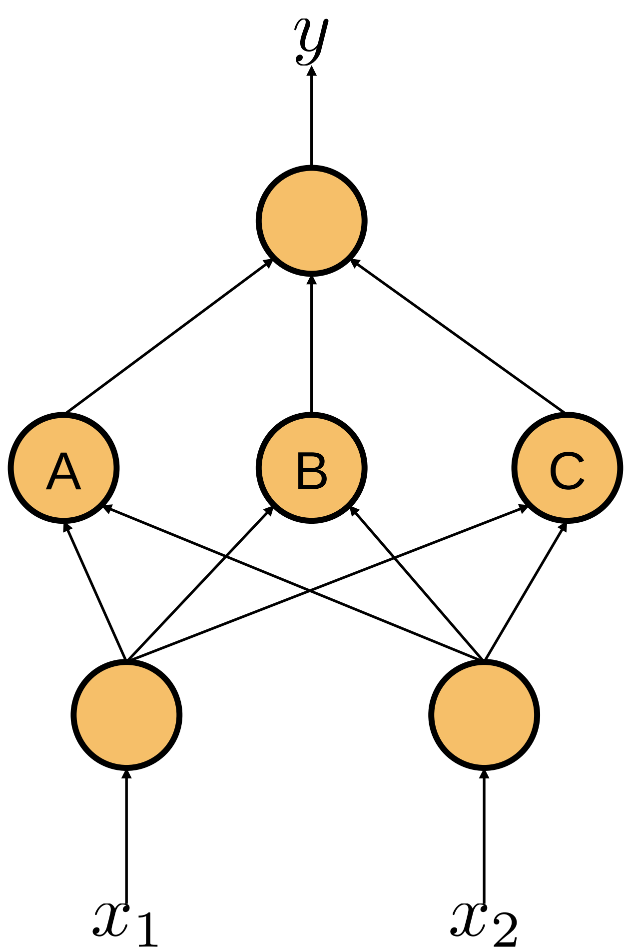
(0.2, -0.3, 0.6, -0.5, 0.4, 0.7)

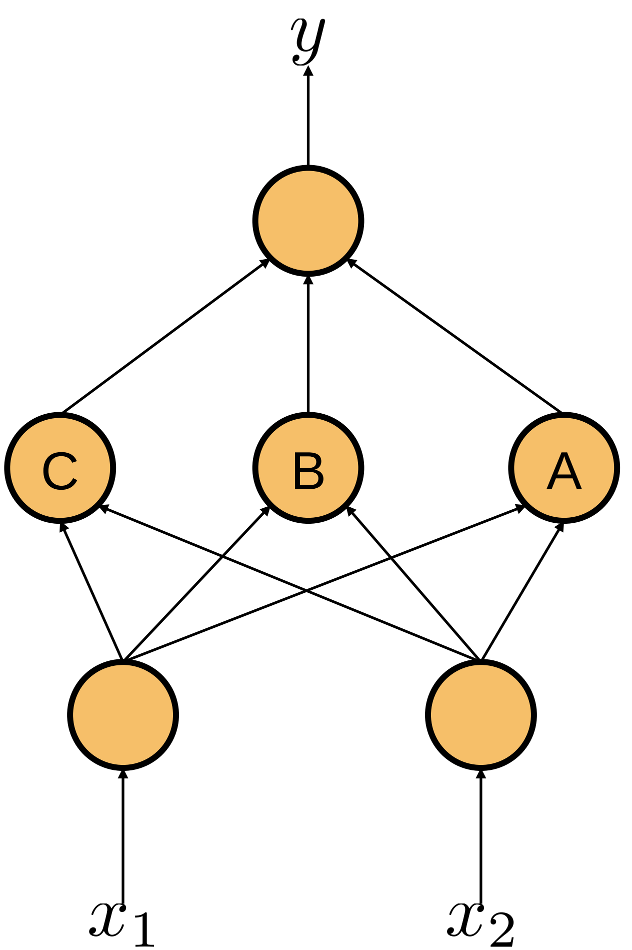
Topology Evolution: Direct Encoding

	1	2	3	4	5	6
1	0	0	0	1	1	0
2	0	0	0	1	0	1
3	0	0	0	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

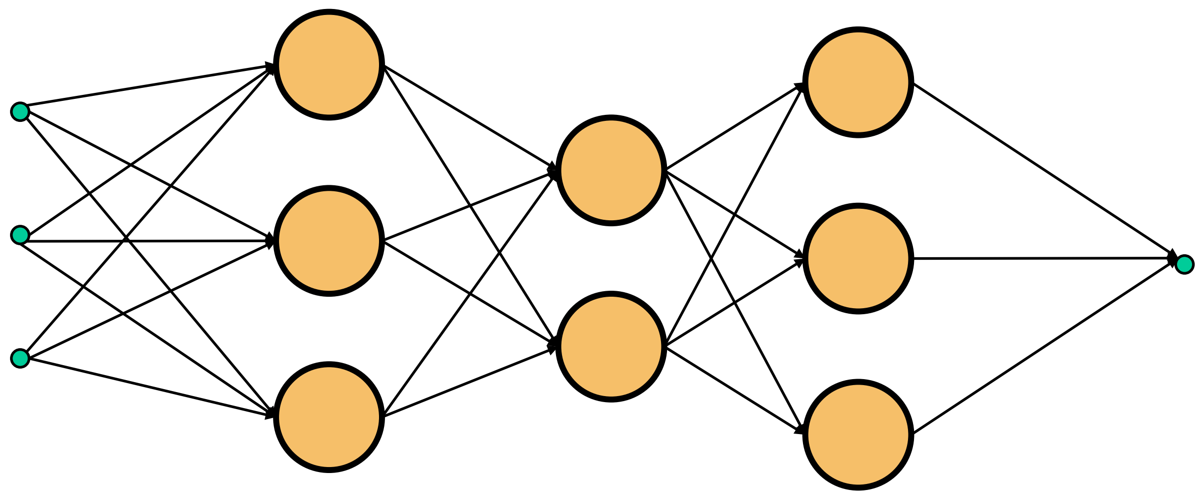


Competing Conventions





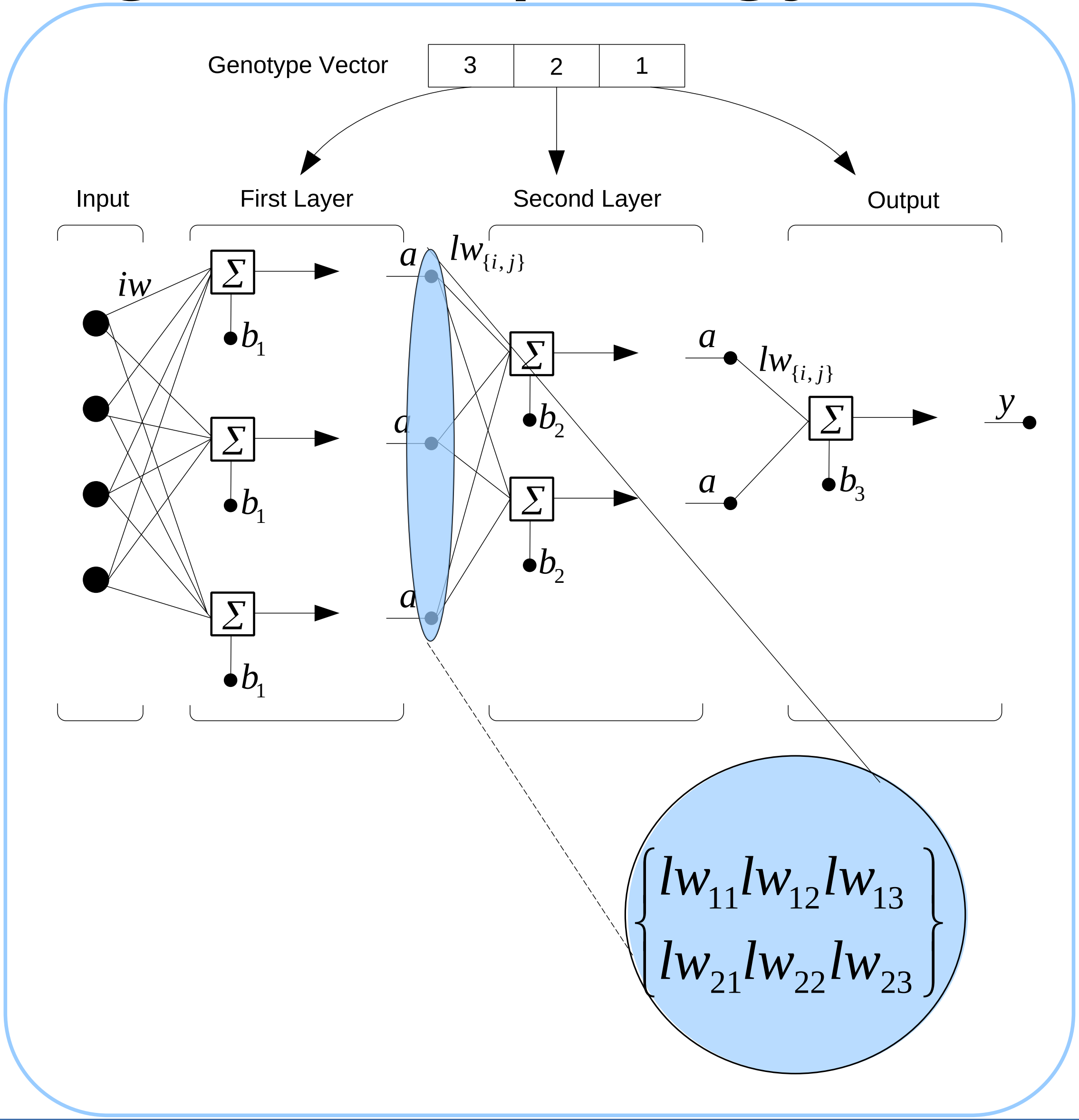
Feed-Forward Weight and Structure Evolution: Direct Encoding



3x3 3x2 2x3 3x1

(3, 2, 3) W_0 W_1 W_2 W_3

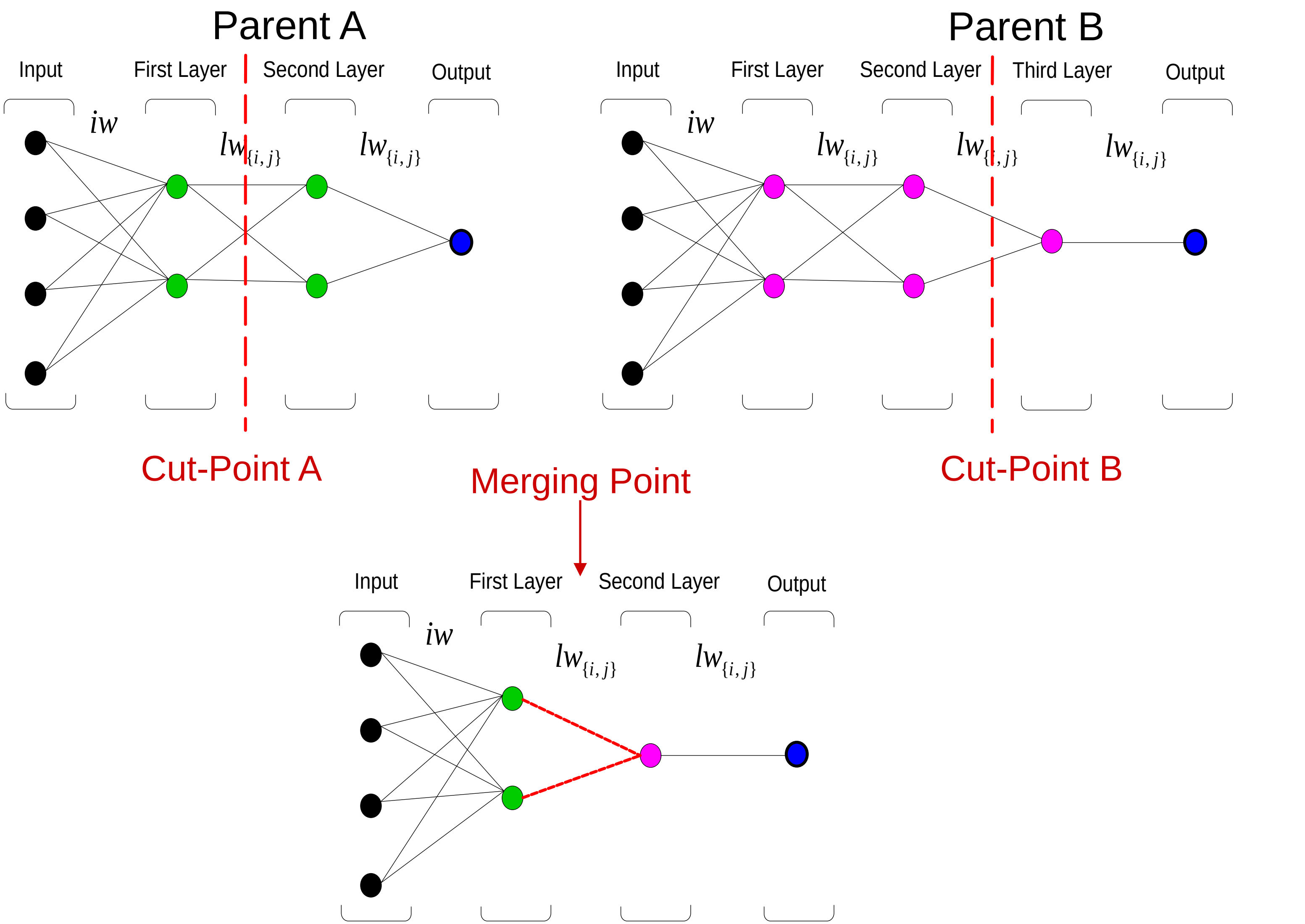
FF Weight & Topology Encoding



FF Weight and Topology Evolution

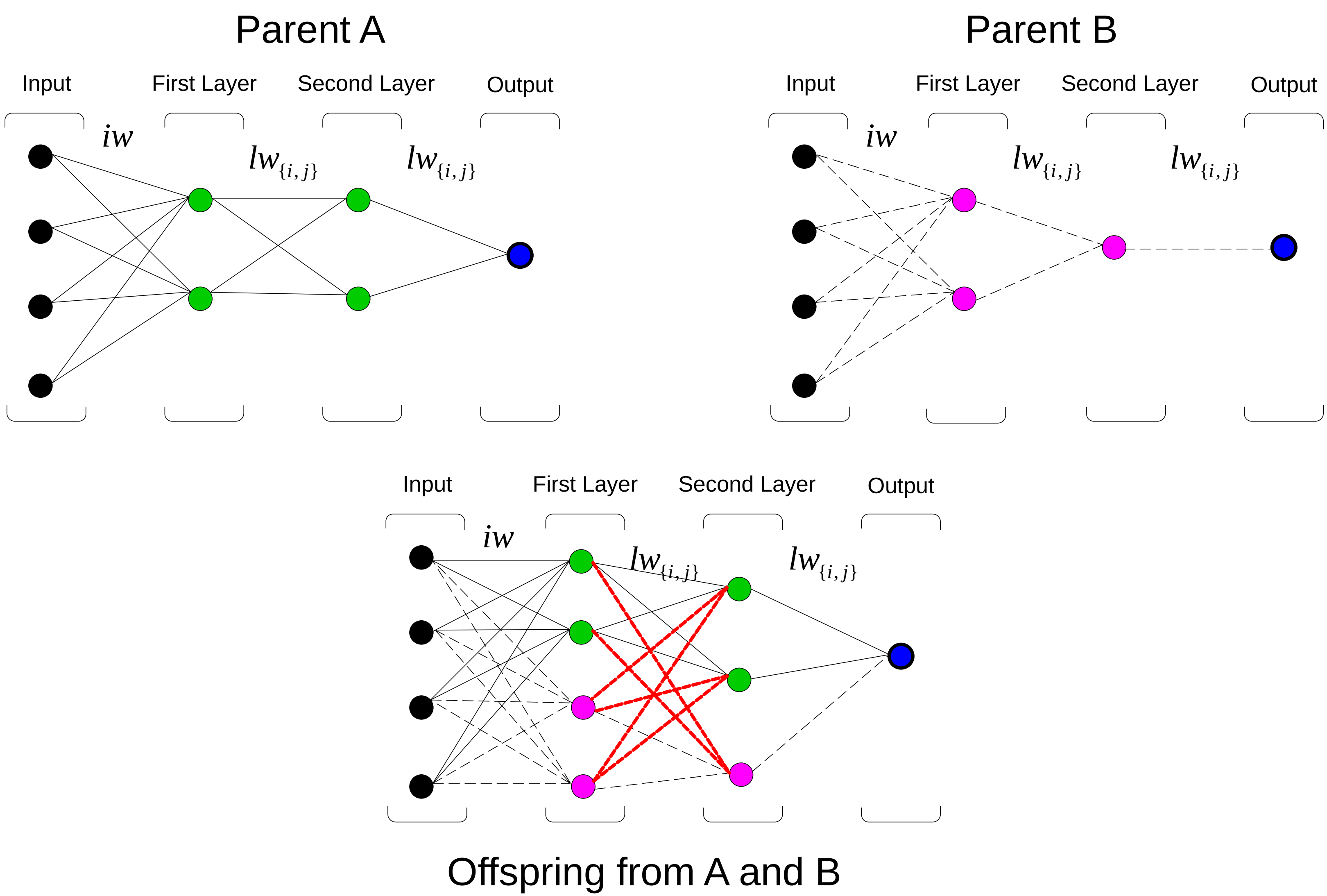
- Mutation Operator:
  - neuron removal: eliminate column in W_{i-1}, row in W_i;
  - neuron duplication: copy column in W_{i-1}, row in W_i;
  - one-neuron layer removal: $W^T_{i-1} W_i$;
  - layer duplication: insert identity matrix;
- Simplification Operator:
  - remove neurons whose row in W_i has norm < ε;
- Crossover Operator:
  - select two crossover points in parents between layers;
  - exchange “tails”;
  - connect layers from different parents with random weight matrix

Single-Point Crossover



Offspring from A and B

Merge or Vertical Crossover



Mutation

- **Mutation:**

- ***Weights and biases Mutation:*** mutate weights and biases by using variance matrices and evolution strategies to NN.

- ***Topology Mutation:*** affect the network structure
  - Neuron Mutation:
    - Neuron insertion/elimination
  - Layer Mutation:
    - Layer insertion/elimination

Weight Perturbation

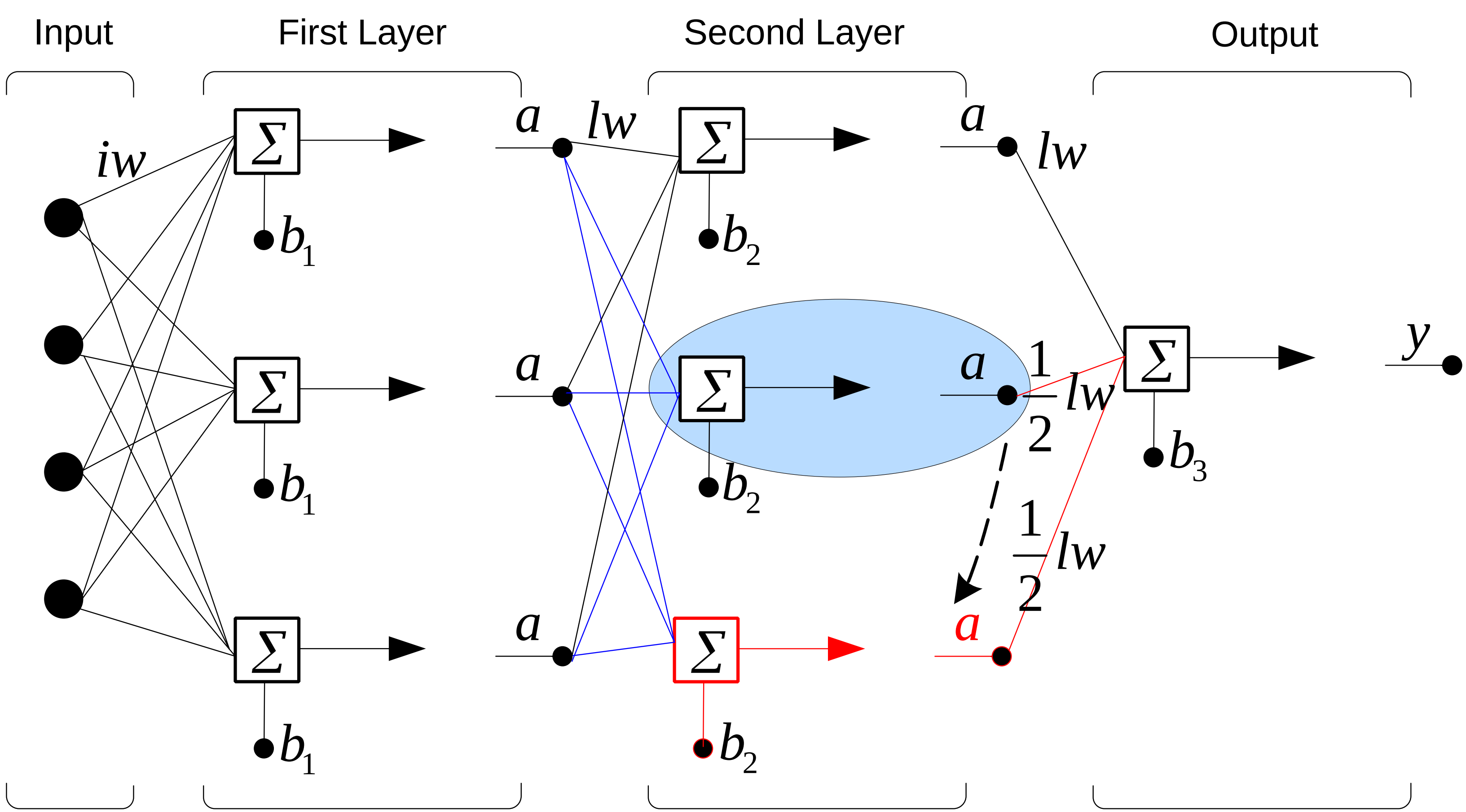
$$lw_{\{i,j\}} = lw_{\{i,j\}} + N(0,1) * Var_{\{i,j\}}$$

$$Var_{\{i,j\}} = Var_{\{i,j\}} * e^{(\tau' * N(0,1)) + (\tau * N(0,1))}$$

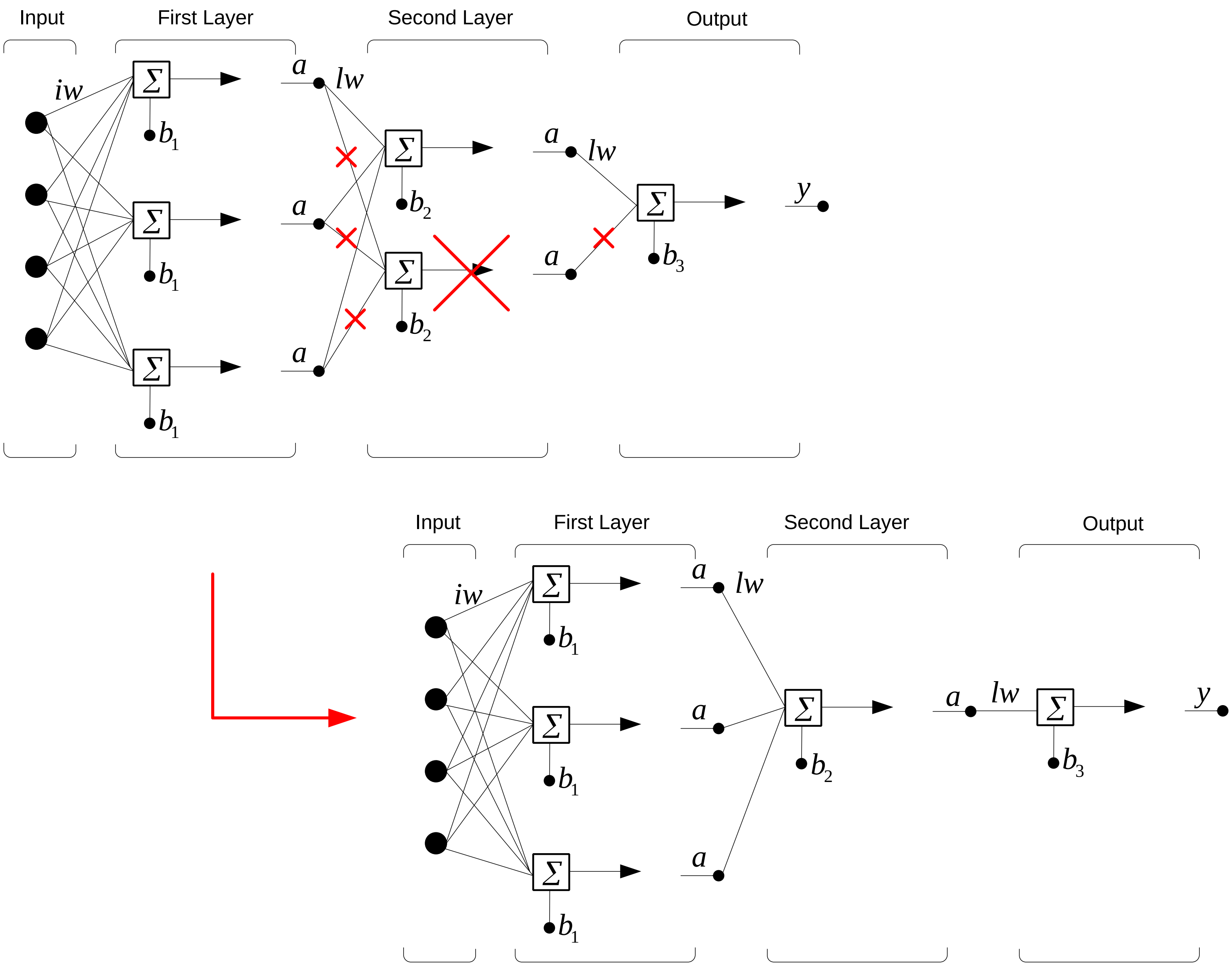
$$\tau = \frac{1}{\sqrt{2 * \sqrt{N_{syn}}}} \qquad \tau' = \frac{1}{\sqrt{2 * N_{syn}}}$$

Evolution Strategy

Topology Mutation: Neuron Insertion



Topology Mutation: Neuron Deletion



Neuron Deletion Conditions

1) Fixed Threshold

$$if \ \| W_j^{(i)} \| < \varepsilon$$

2) Variable Threshold

$$if \ \| W_j^{(i)} \| < (avg_k(\| W_j^{(i)} \|) - r \cdot stdev_j(\| W_j^{(i)} \|))$$

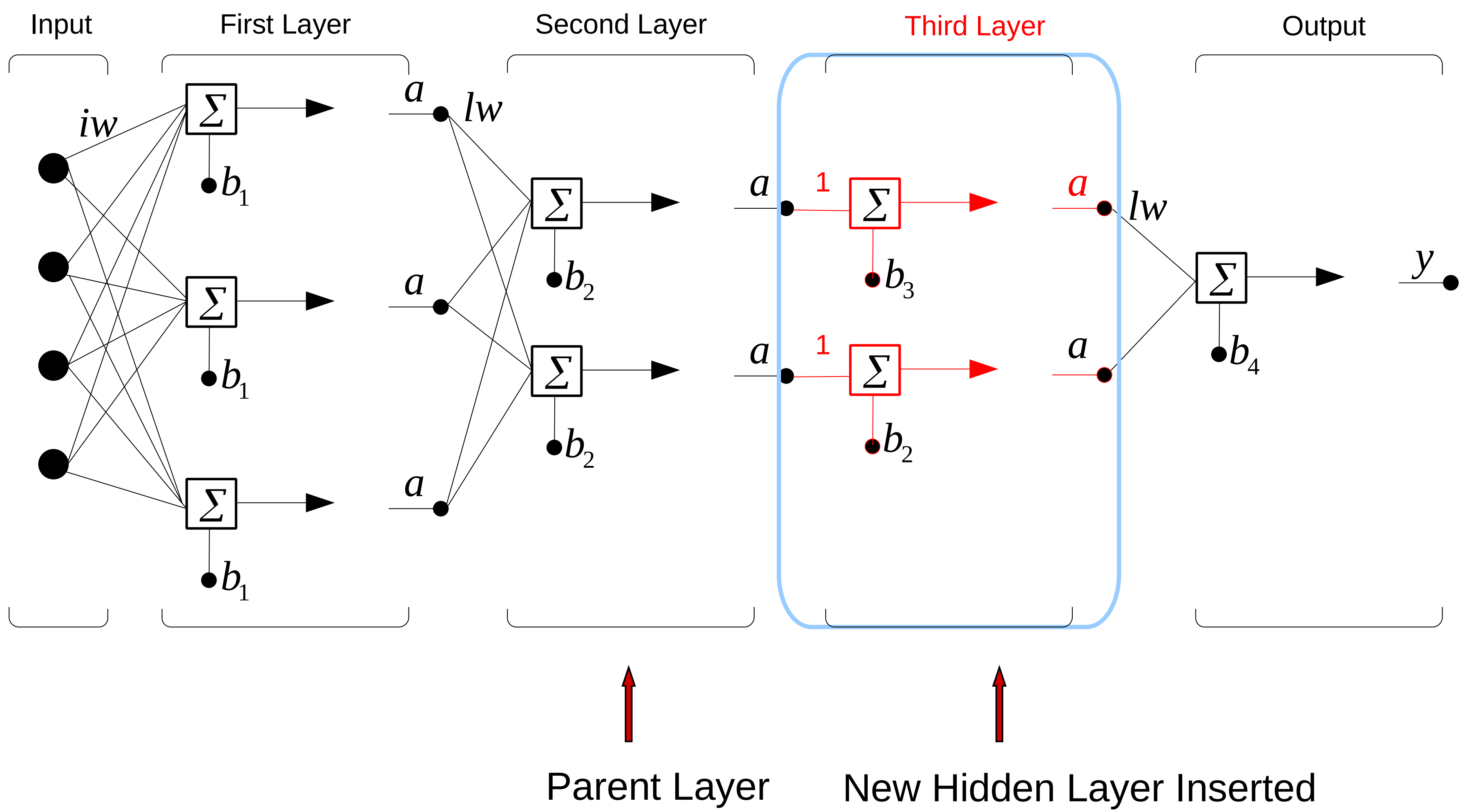
With:

$W_j^{(i)}$ *j*th column of the matrix

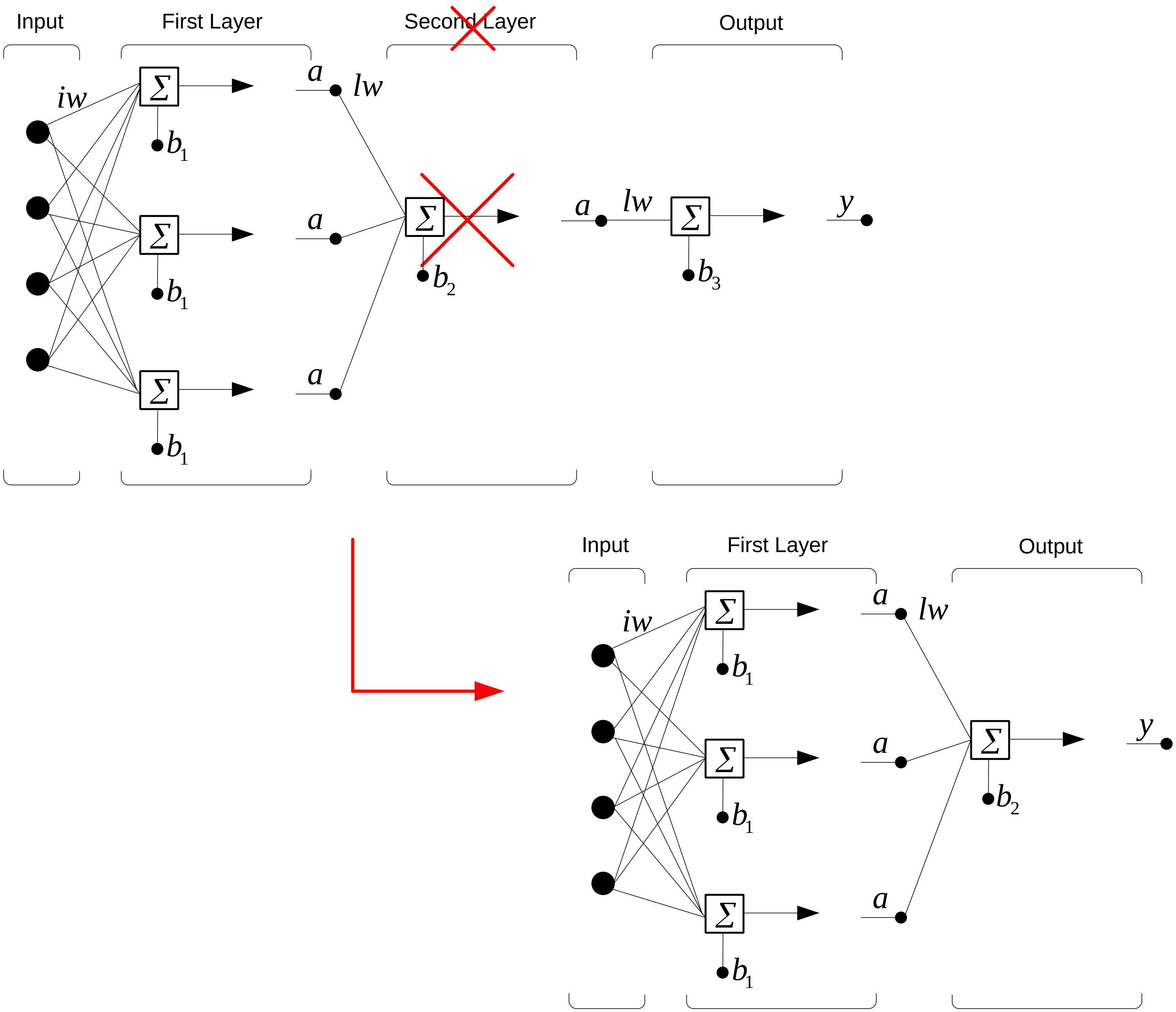
ε Value of fixed threshold

r Standard deviation tuning parameter

Topology Mutation: Layer Insertion



Topology Mutation: Layer Deletion



Structure Evolution: Indirect Encoding

Graph-generating Grammar

$$S \rightarrow \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$A \rightarrow \begin{pmatrix} c & d \\ a & c \end{pmatrix}, B \rightarrow \begin{pmatrix} a & a \\ a & e \end{pmatrix}, C \rightarrow \begin{pmatrix} a & a \\ a & a \end{pmatrix}, D \rightarrow \begin{pmatrix} a & a \\ a & b \end{pmatrix}$$

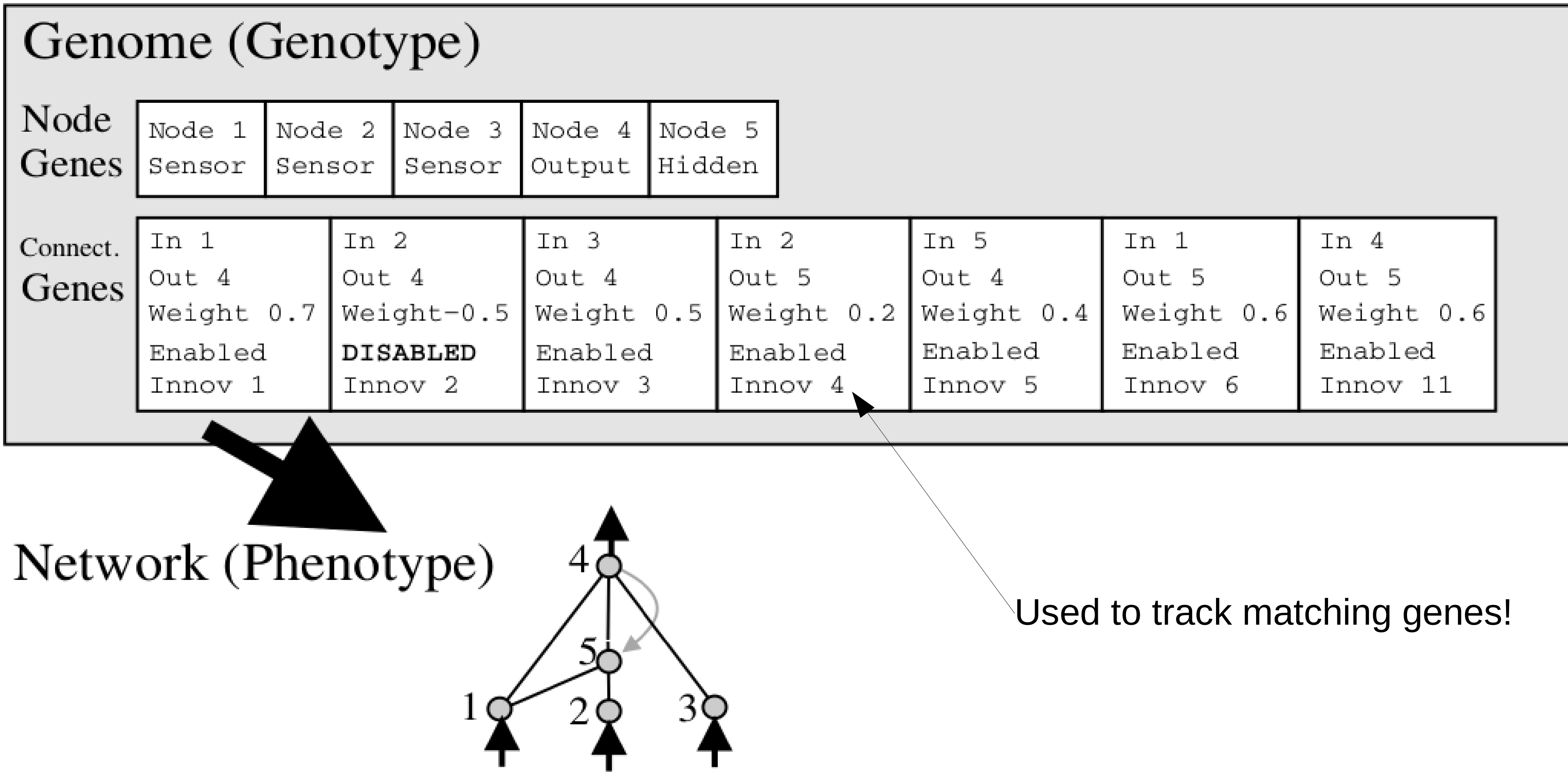
$$a \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, b \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, c \rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, d \rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, e \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

(S: A, B, C, D || A: c, d, a, c || B: a, a, a, e || C: a, a, a, a || ...)

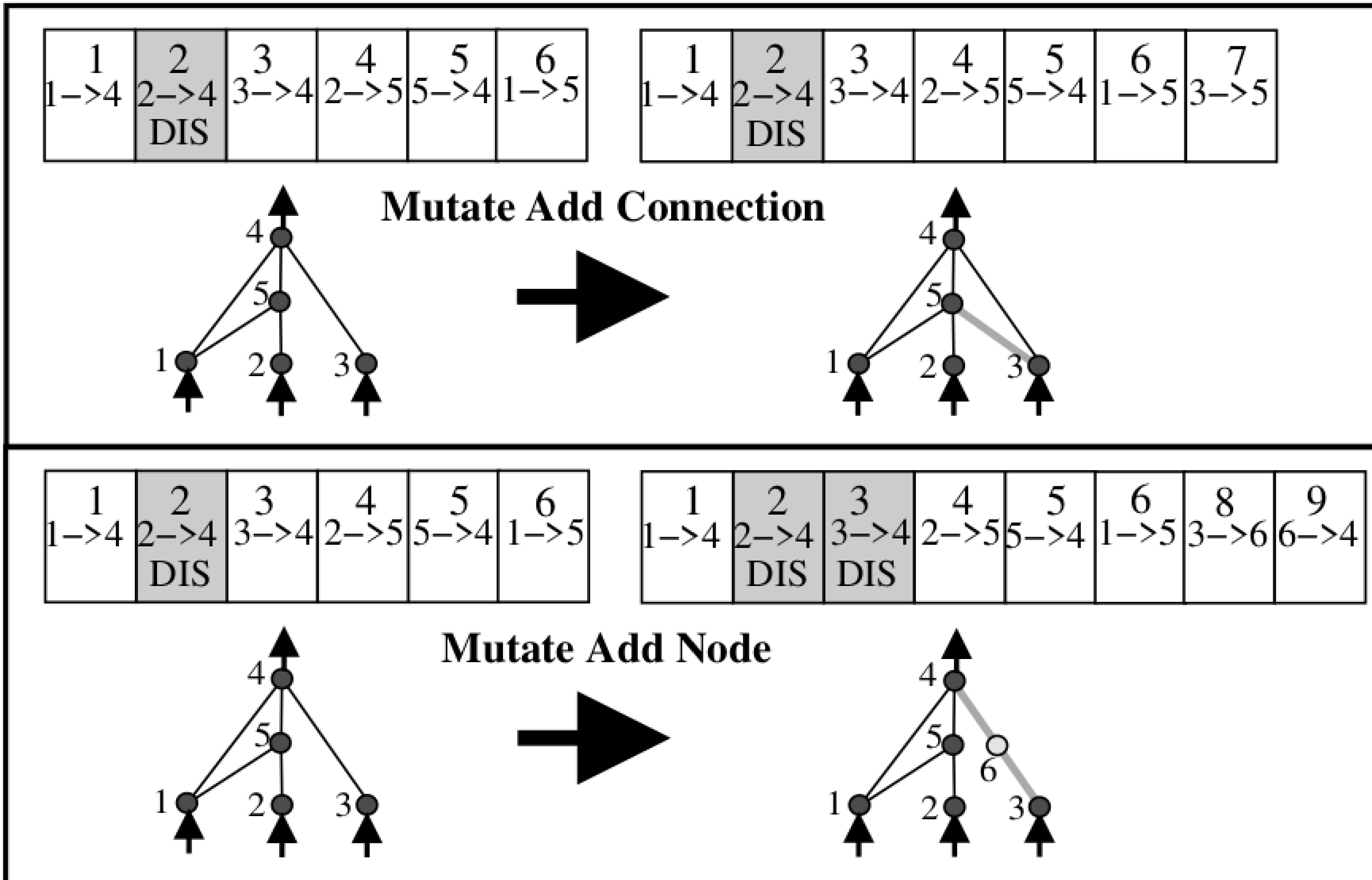
NeuroEvolution of Augmenting Topologies (NEAT)

- Proposed by Stanley and Miikkulainen
- Idea
  - start with **minimal** random topologies
  - Augment topologies as you go if required
  - Track mathcing genes to alleviate the competing convention problem
  - Protect innovations through speciation

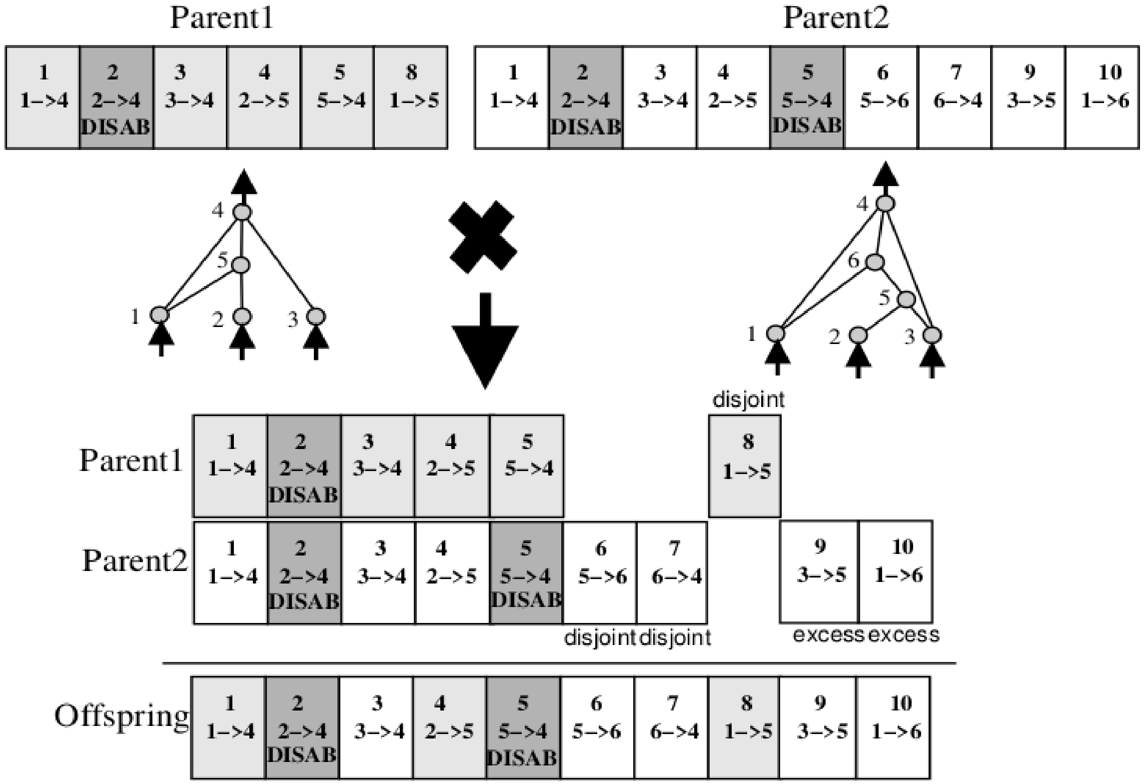
NEAT Encoding



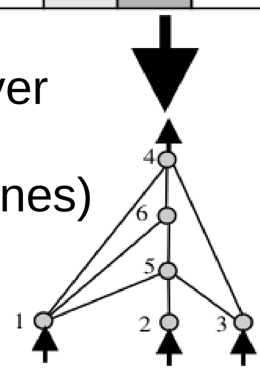
NEAT Structural Mutation



NEAT Crossover



Speciation: do not allow crossover
if parents are too diverse
(too many disjoint and excess genes)



Non-matching genes are called:
Disjoint if innov within range of other parent
Excess if innov outside range of other parent

EAs for NN Input Selection

- Given a detaset,
- Select a subset of columns for NN training

- Representation: straightforward bit string
- Operators: standard mutation and crossover

- Fitness: MSE of trained NN