Lecture 12

Hybrid intelligent systems:
Evolutionary neural networks and fuzzy evolutionary systems

- Introduction
- Evolutionary neural networks
- Fuzzy evolutionary systems
- Summary

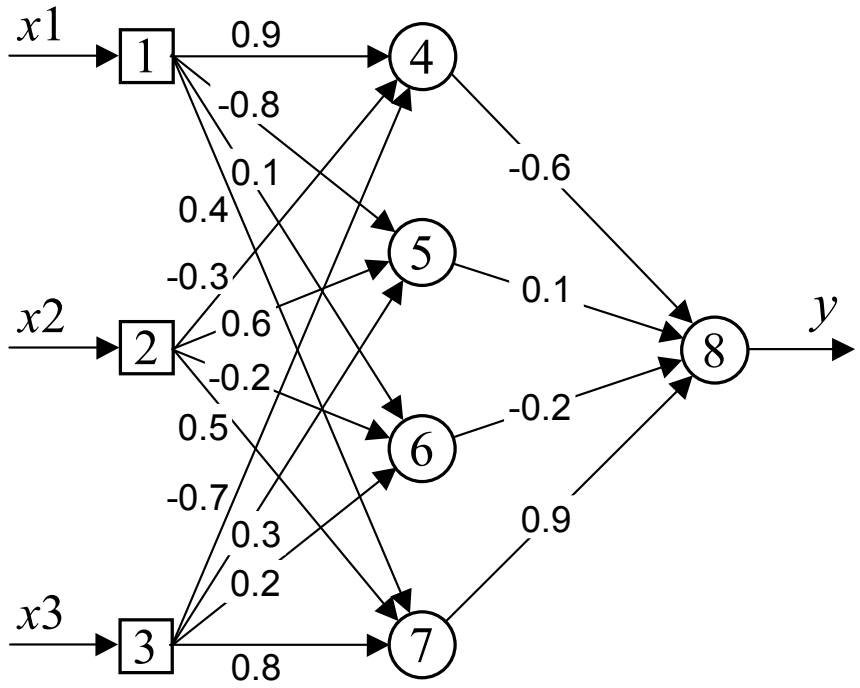
Evolutionary neural networks

- Although neural networks are used for solving a variety of problems, they still have some limitations.
- One of the most common is associated with neural network training. The back-propagation learning algorithm cannot guarantee an optimal solution. In real-world applications, the back-propagation algorithm might converge to a set of sub-optimal weights from which it cannot escape. As a result, the neural network is often unable to find a desirable solution to a problem at hand.

- Another difficulty is related to selecting an optimal topology for the neural network. The "right" network architecture for a particular problem is often chosen by means of heuristics, and designing a neural network topology is still more art than engineering.
- Genetic algorithms are an effective optimisation technique that can guide both weight optimisation and topology selection.

Encoding a set of weights in a chromosome

From neuron:	1	2	3	4	5	6	7	8
To neuron: 1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0.9	-0.3	-0.7	0	0	0	0	0
5	-0.8	0.6	0.3	0	0	0	0	0
6	0.1	-0.2	0.2	0	0	0	0	0
7	0.4	0.5	0.8	0	0	0	0	0
8	0	0	0	-0.6	0.1	-0.2	0.9	0



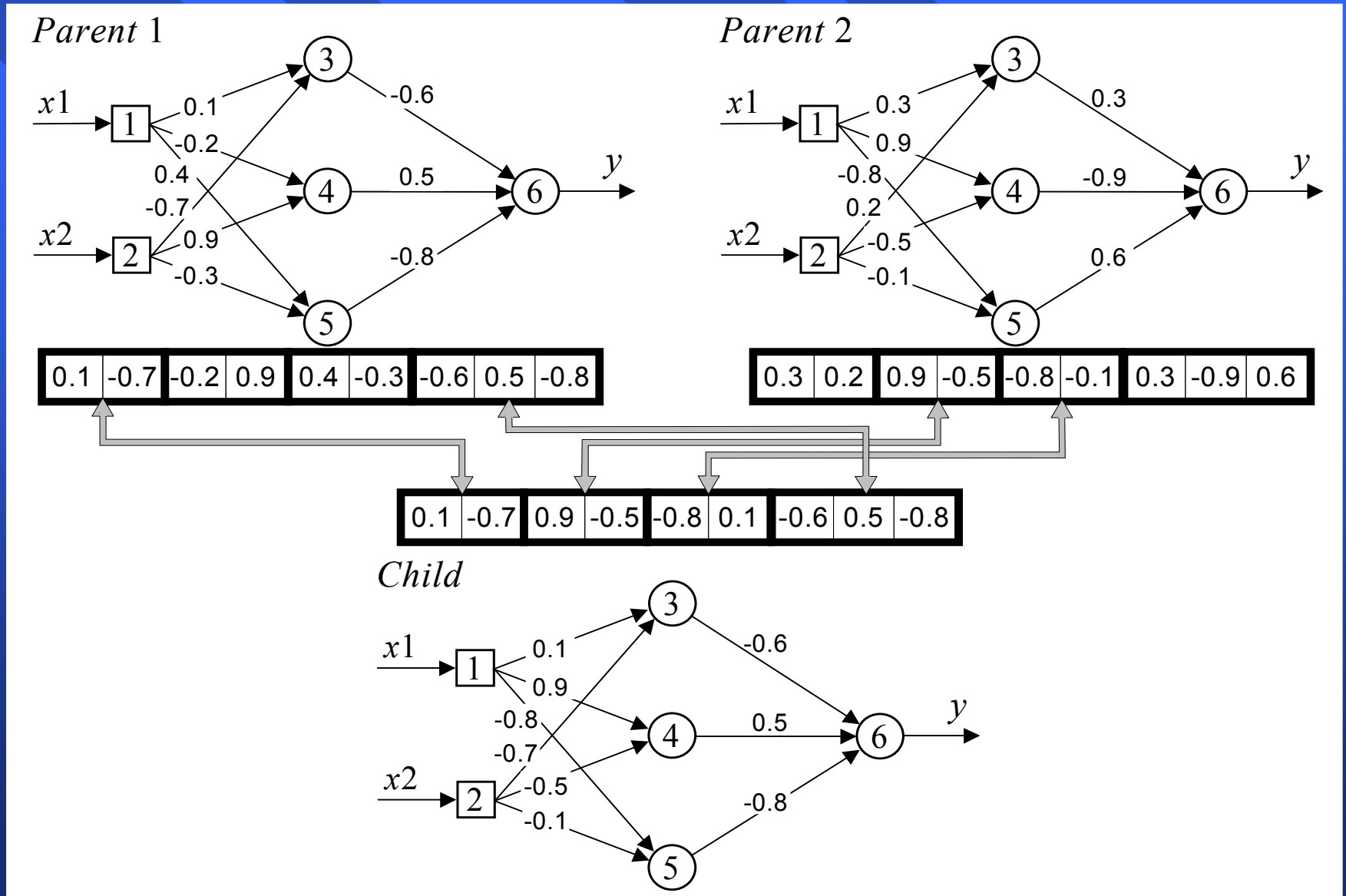
Chromosome:

0.9	-0.3	-0.7	-0.8	0.6	0.3	0.1	-0.2	0.2	0.4	0.5	0.8	-0.6	0.1	-0.2	0.9

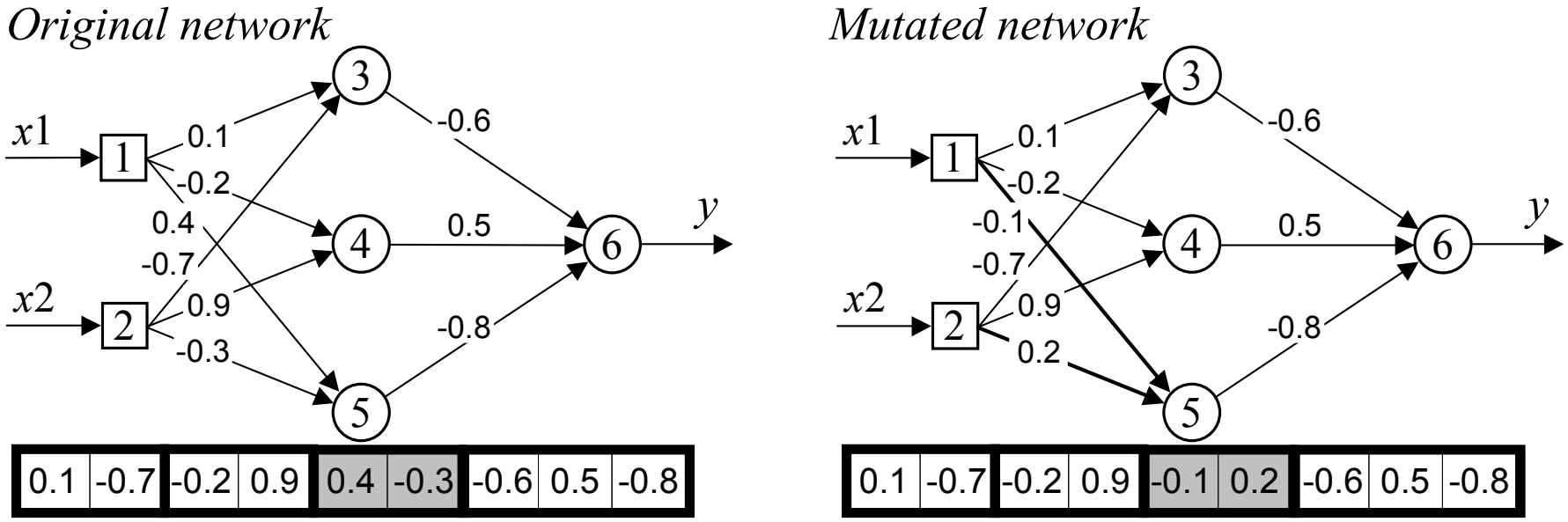
- The second step is to define a fitness function for evaluating the chromosome's performance. This function must estimate the performance of a given neural network. We can apply here a simple function defined by the sum of squared errors.
- The training set of examples is presented to the network, and the sum of squared errors is calculated. The smaller the sum, the fitter the chromosome. **The genetic algorithm attempts to find a set of weights that minimises the sum of squared errors.**

- The third step is to choose the genetic operators – crossover and mutation. A crossover operator takes two parent chromosomes and creates a single child with genetic material from both parents. Each gene in the child's chromosome is represented by the corresponding gene of the randomly selected parent.
- A mutation operator selects a gene in a chromosome and adds a small random value between –1 and 1 to each weight in this gene.

Crossover in weight optimisation



Mutation in weight optimisation



Can genetic algorithms help us in selecting the network architecture?

The architecture of the network (i.e. the number of neurons and their interconnections) often determines the success or failure of the application. Usually the network architecture is decided by trial and error; there is a great need for a method of automatically designing the architecture for a particular application. Genetic algorithms may well be suited for this task.

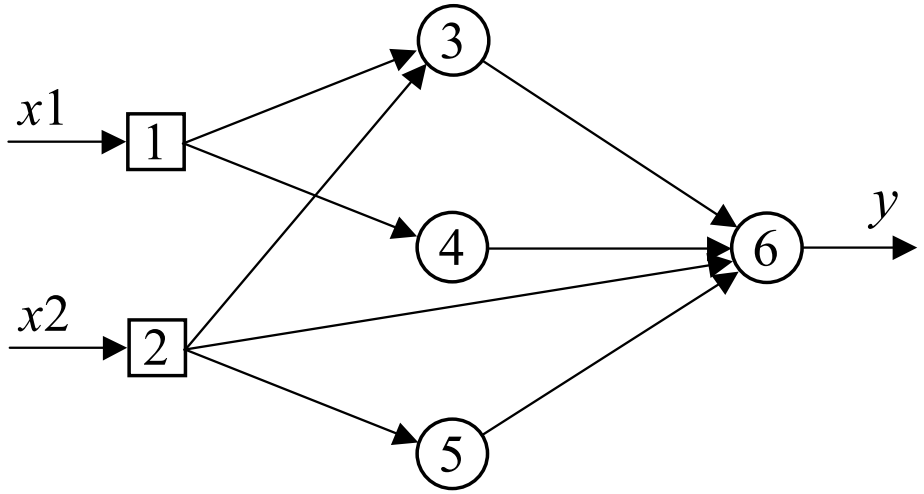
- The basic idea behind evolving a suitable network architecture is to conduct a genetic search in a population of possible architectures.
- We must first choose a method of encoding a network's architecture into a chromosome.

Encoding the network architecture

- The connection topology of a neural network can be represented by a square connectivity matrix.
- Each entry in the matrix defines the type of connection from one neuron (column) to another (row), where 0 means no connection and 1 denotes connection for which the weight can be changed through learning.
- To transform the connectivity matrix into a chromosome, we need only to string the rows of the matrix together.

Encoding of the network topology

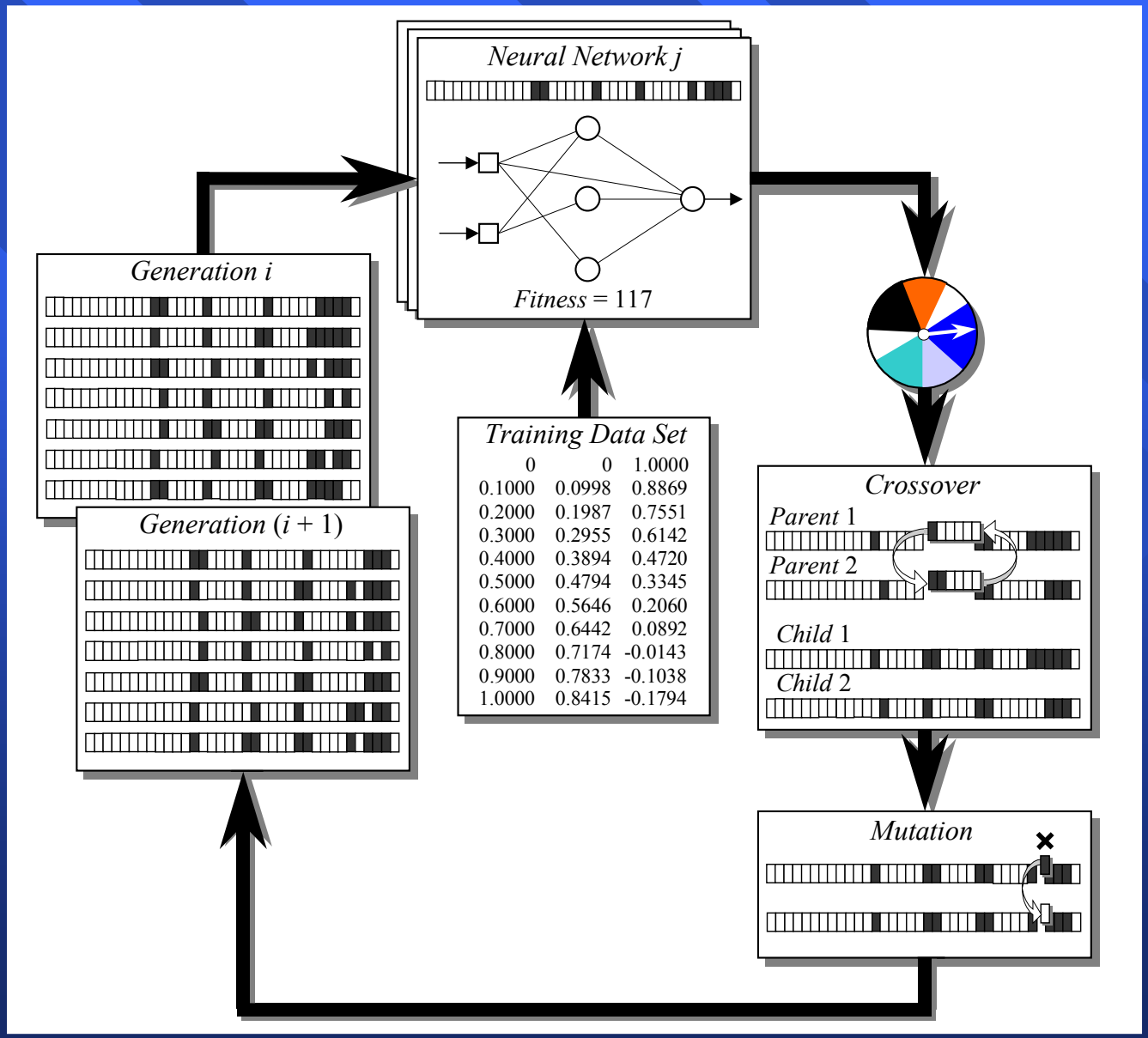
From neuron:	1	2	3	4	5	6
To neuron: 1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	1	1	1	1	0



Chromosome:

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0

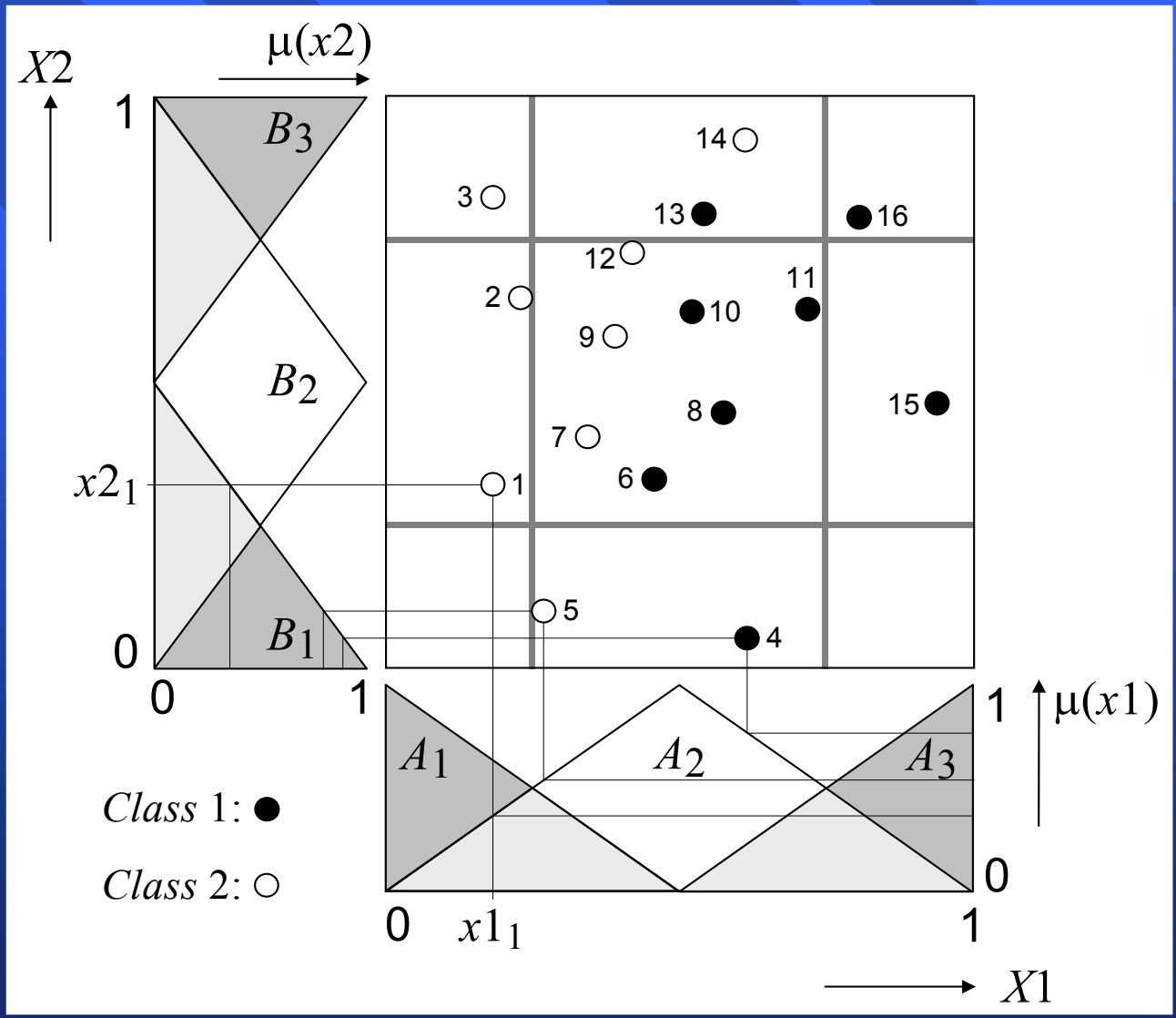
The cycle of evolving a neural network topology



Fuzzy evolutionary systems

- Evolutionary computation is also used in the design of fuzzy systems, particularly for generating fuzzy rules and adjusting membership functions of fuzzy sets.
- In this section, we introduce an application of genetic algorithms to select an appropriate set of fuzzy IF-THEN rules for a classification problem.
- For a classification problem, a set of fuzzy IF-THEN rules is generated from numerical data.
- First, we use a grid-type fuzzy partition of an input space.

Fuzzy partition by a 3×3 fuzzy grid



Fuzzy partition

- Black and white dots denote the training patterns of *Class* 1 and *Class* 2, respectively.
- The grid-type fuzzy partition can be seen as a rule table.
- The linguistic values of input $x1$ (A_1, A_2 and A_3) form the horizontal axis, and the linguistic values of input $x2$ (B_1, B_2 and B_3) form the vertical axis.
- At the intersection of a row and a column lies the rule consequent.

In the rule table, each fuzzy subspace can have only one fuzzy IF-THEN rule, and thus the total number of rules that can be generated in a $K \times K$ grid is equal to $K \times K$.

Fuzzy rules that correspond to the $K \times K$ fuzzy partition can be represented in a general form as:

Rule R_{ij}:

IF $x1_p$ is A_i $\quad i = 1, 2, \ldots, K$

AND $x2_p$ is B_j $\quad j = 1, 2, \ldots, K$

THEN $\mathbf{x}_p \in C_n \left\{ CF_{A_i B_j}^{C_n} \right\}$ $\quad \mathbf{x}_p = (x1_p, x2_p),\ p = 1, 2, \ldots, P$

where $\mathbf{x}_p$ is a training pattern on input space $X1 \times X2$, P is the total number of training patterns, C_n is the rule consequent (either *Class* 1 or *Class* 2), and $CF_{A_i B_j}^{C_n}$ is the certainty factor that a pattern in fuzzy subspace $A_i B_j$ belongs to class C_n.

To determine the rule consequent and the certainty factor, we use the following procedure:

Step 1: Partition an input space into $K \times K$ fuzzy subspaces, and calculate the strength of each class of training patterns in every fuzzy subspace.

Each class in a given fuzzy subspace is represented by its training patterns. The more training patterns, the stronger the class – in a given fuzzy subspace, the rule consequent becomes more certain when patterns of one particular class appear more often than patterns of any other class.

Step 2: Determine the rule consequent and the certainty factor in each fuzzy subspace.

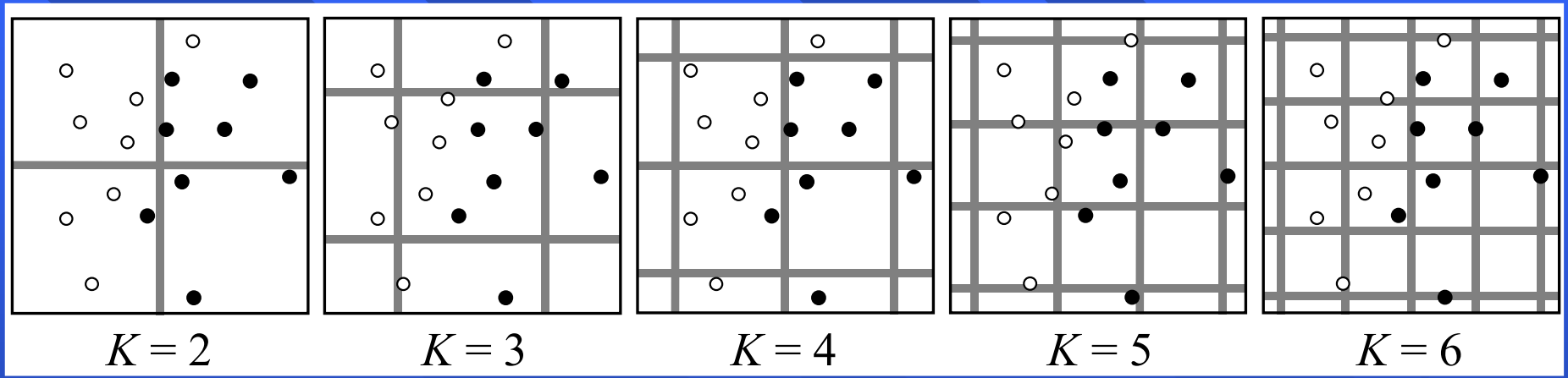
The certainty factor can be interpreted as follows:

- If all the training patterns in fuzzy subspace $A_i B_j$ belong to the same class, then the certainty factor is maximum and it is certain that any new pattern in this subspace will belong to this class.
- If, however, training patterns belong to different classes and these classes have similar strengths, then the certainty factor is minimum and it is uncertain that a new pattern will belong to any particular class.

- This means that patterns in a fuzzy subspace can be misclassified. Moreover, if a fuzzy subspace does not have any training patterns, we cannot determine the rule consequent at all.
- If a fuzzy partition is too coarse, many patterns may be misclassified. On the other hand, if a fuzzy partition is too fine, many fuzzy rules cannot be obtained, because of the lack of training patterns in the corresponding fuzzy subspaces.

Training patterns are not necessarily distributed evenly in the input space. As a result, it is often difficult to choose an appropriate density for the fuzzy grid. To overcome this difficulty, we use **multiple fuzzy rule tables**.

Multiple fuzzy rule tables



Fuzzy IF-THEN rules are generated for each fuzzy subspace of multiple fuzzy rule tables, and thus a complete set of rules for our case can be specified as:

$$2^2 + 3^2 + 4^2 + 5^2 + 6^2 = 90 \text{ rules.}$$

Once the set of rules S_{ALL} is generated, a new pattern, $\mathbf{x} = (x1, x2)$, can be classified by the following procedure:

Step 1: In every fuzzy subspace of the multiple fuzzy rule tables, calculate the degree of compatibility of a new pattern with each class.

Step 2: Determine the maximum degree of compatibility of the new pattern with each class.

Step 3: Determine the class with which the new pattern has the highest degree of compatibility, and assign the pattern to this class.

The number of multiple fuzzy rule tables required for an accurate pattern classification may be large. Consequently, a complete set of rules can be enormous. Meanwhile, these rules have different classification abilities, and thus by selecting only rules with high potential for accurate classification, we reduce the number of rules.

Can we use genetic algorithms for selecting fuzzy IF-THEN rules ?

- The problem of selecting fuzzy IF-THEN rules can be seen as a combinatorial optimisation problem with two objectives.
- The first, more important, objective is to maximise the number of correctly classified patterns.
- The second objective is to minimise the number of rules.
- Genetic algorithms can be applied to this problem.

A basic genetic algorithm for selecting fuzzy IF-THEN rules includes the following steps:

Step 1: Randomly generate an initial population of chromosomes. The population size may be relatively small, say 10 or 20 chromosomes. Each gene in a chromosome corresponds to a particular fuzzy IF-THEN rule in the rule set defined by S_{ALL}.

Step 2: Calculate the performance, or fitness, of each individual chromosome in the current population.

The problem of selecting fuzzy rules has two objectives: to maximise the accuracy of the pattern classification and to minimise the size of a rule set. The fitness function has to accommodate both these objectives. This can be achieved by introducing two respective weights, w_P and w_N, in the fitness function:

$$f(S) = w_P \frac{P_s}{P_{ALL}} - w_N \frac{N_S}{N_{ALL}}$$

where P_s is the number of patterns classified successfully, P_{ALL} is the total number of patterns presented to the classification system, N_S and N_{ALL} are the numbers of fuzzy IF-THEN rules in set S and set S_{ALL}, respectively.

The classification accuracy is more important than the size of a rule set. That is,

$$f(S) = 10\frac{P_s}{P_{ALL}} - \frac{N_S}{N_{ALL}}$$

Step 3: Select a pair of chromosomes for mating. Parent chromosomes are selected with a probability associated with their fitness; a better fit chromosome has a higher probability of being selected.

Step 4: Create a pair of offspring chromosomes by applying a standard crossover operator. Parent chromosomes are crossed at the randomly selected crossover point.

Step 5: Perform mutation on each gene of the created offspring. The mutation probability is normally kept quite low, say 0.01. The mutation is done by multiplying the gene value by –1.

Step 6: Place the created offspring chromosomes in the new population.

Step 7: Repeat *Step 3* until the size of the new population becomes equal to the size of the initial population, and then replace the initial (parent) population with the new (offspring) population.

Step 9: Go to *Step 2*, and repeat the process until a specified number of generations (typically several hundreds) is considered.

The number of rules can be cut down to less than 2% of the initially generated set of rules.