Lecture 15

Knowledge engineering:
Genetic algorithms and hybrid systems

- Will genetic algorithms work for my problem?
  - The travelling salesman problem
- Will a hybrid system work for my problem?
  - Neuro-fuzzy decision-support systems
  - Time-series prediction
- Summary

Will genetic algorithms work for my problem?

- ***Optimisation*** is the process of finding a better solution to a problem.
- A genetic algorithm generates a population of competing candidate solutions and then causes them to evolve through the process of natural selection – poor solutions tend to die out, while better solutions survive and reproduce. By repeating this process over and over again, the genetic algorithm breeds an optimal solution.

Case study 8
The travelling salesman problem

Suppose, we are going to travel by car in Western and Central Europe. We want to produce an optimal itinerary for visiting all major cities and returning home.

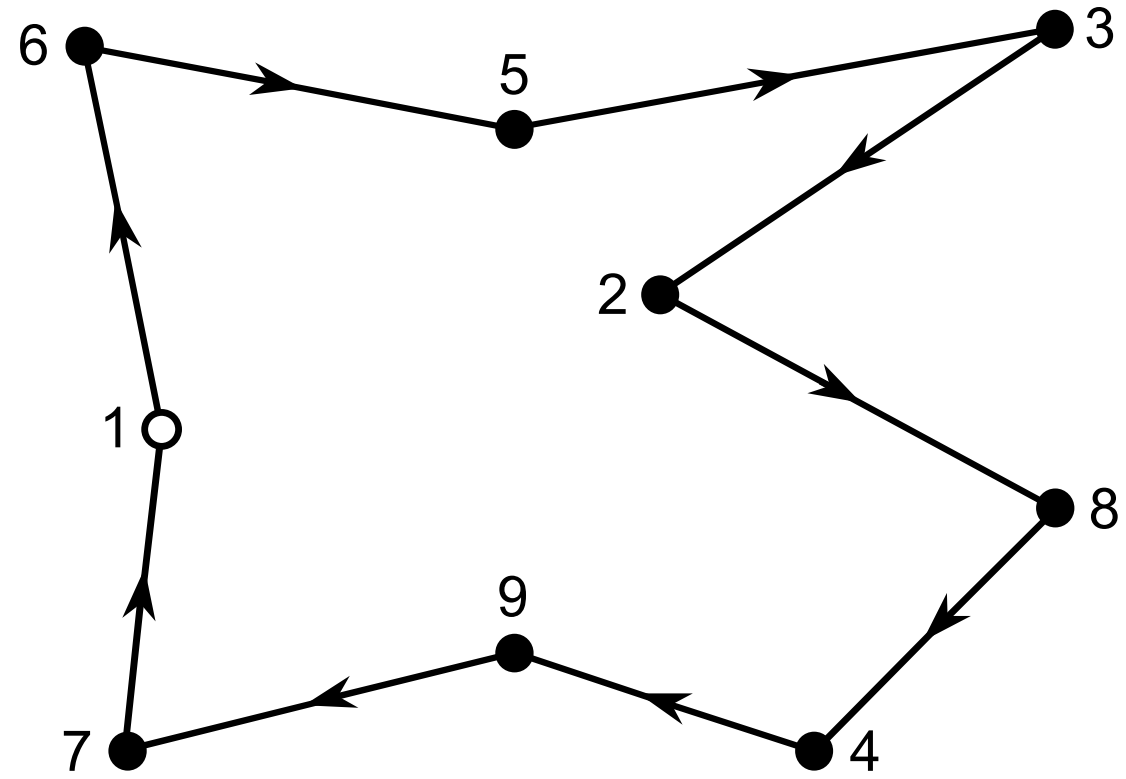
This problem is well known as the ***travelling salesman problem***. Given a finite number of cities, *N*, and the cost of travel (or the distance) between each pair of cities, we need to find the cheapest way (or the shortest route) for visiting each city exactly once and returning to the starting point.

- The TSP is represented in numerous transportation and logistics applications such as
  - arranging routes for school buses to pick up children in a school district,
  - delivering meals to home-bound people,
  - scheduling stacker cranes in a warehouse,
  - planning truck routes to pick up parcel post and many others.
- A classic example of the TSP is the scheduling of a machine to drill holes in a circuit board.

How does a genetic algorithm solve the travelling salesman problem?

- First, we need to decide how to represent a route of the salesman. The most natural way of representing a route is the ***path representation***. Each city is given an alphabetic or numerical name, the route through the cities is represented as a chromosome, and appropriate genetic operators are used to create new routes.
- Suppose we have nine cities named from 1 to 9. In a chromosome, the order of the integers represents the order in which the cities will be visited by the salesman.

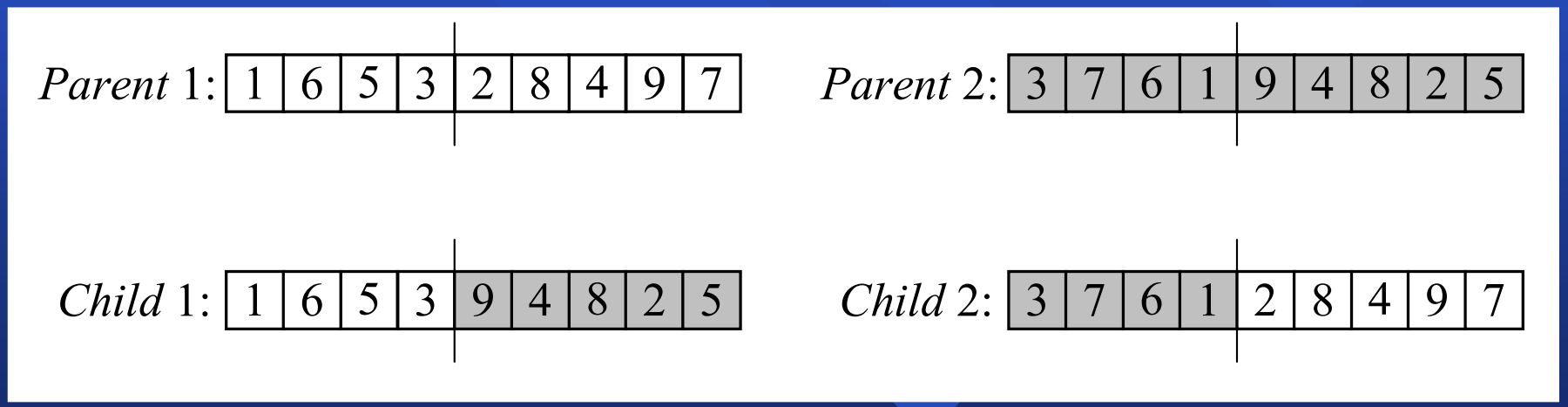
An example of the salesman's route



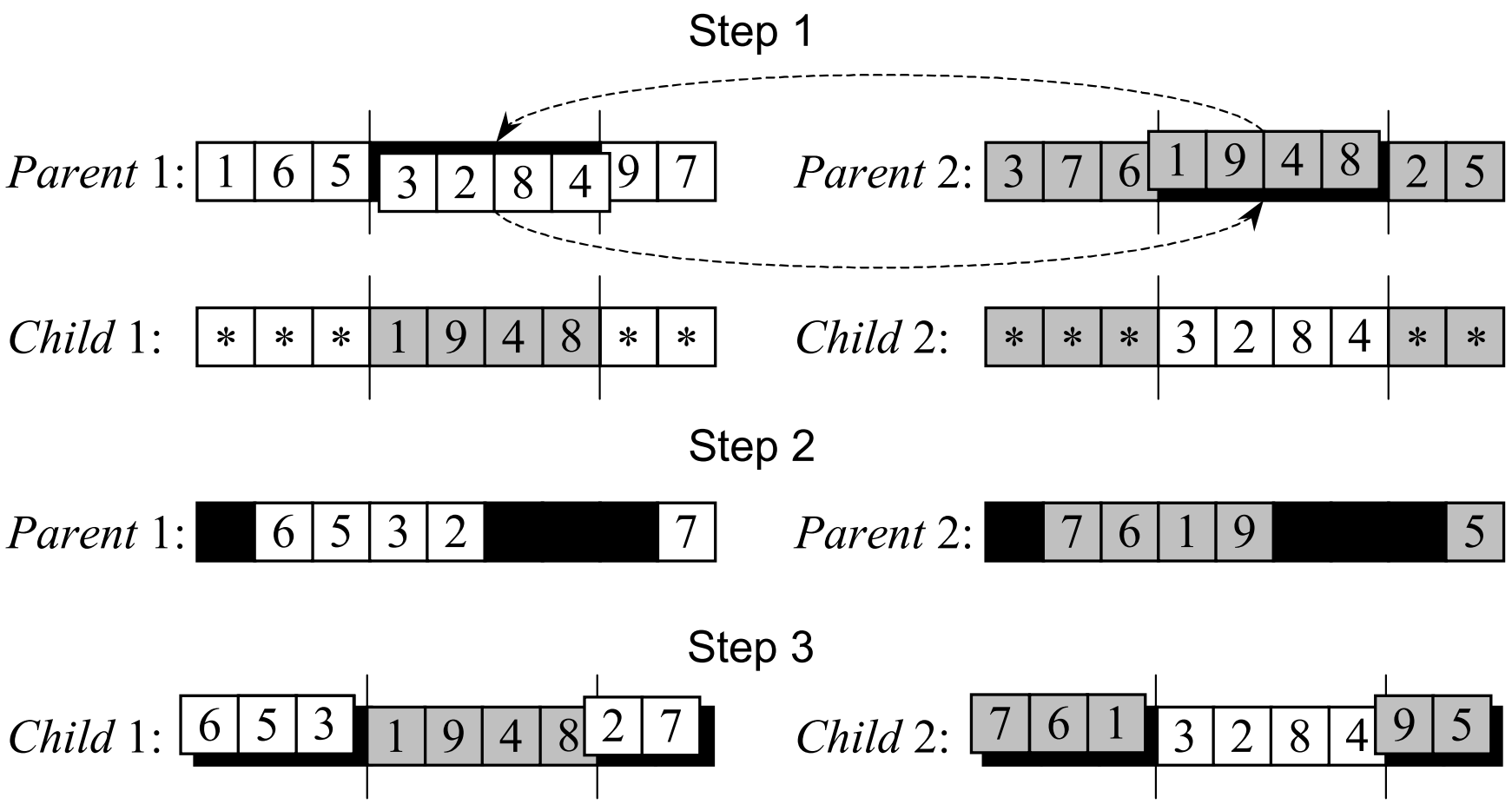
1	6	5	3	2	8	4	9	7

How does the crossover operator works?

The crossover operator in its classical form cannot be directly applied to the TSP. A simple exchange of parts between two parents would produce illegal routes containing duplicates and omissions – some cities would be visited twice while some others would not be visited at all:



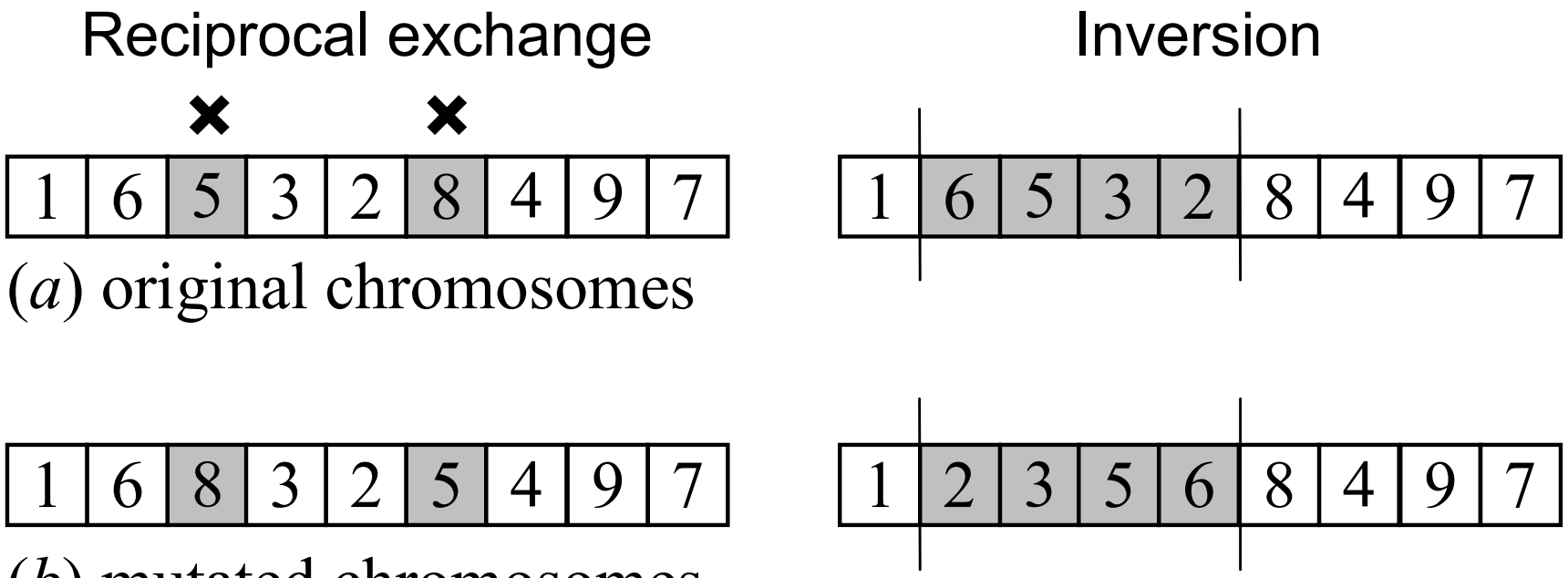
Crossover operators for the TSP



How does the mutation operator works?

- There are two types of mutation operators: ***reciprocal exchange*** and ***inversion***.
- The *reciprocal exchange operator* simply swaps two randomly selected cities in the chromosome.
- The *inversion operator* selects two random points along the chromosome string and reverses the order of the cities between these points.

Mutation operators for the TSP

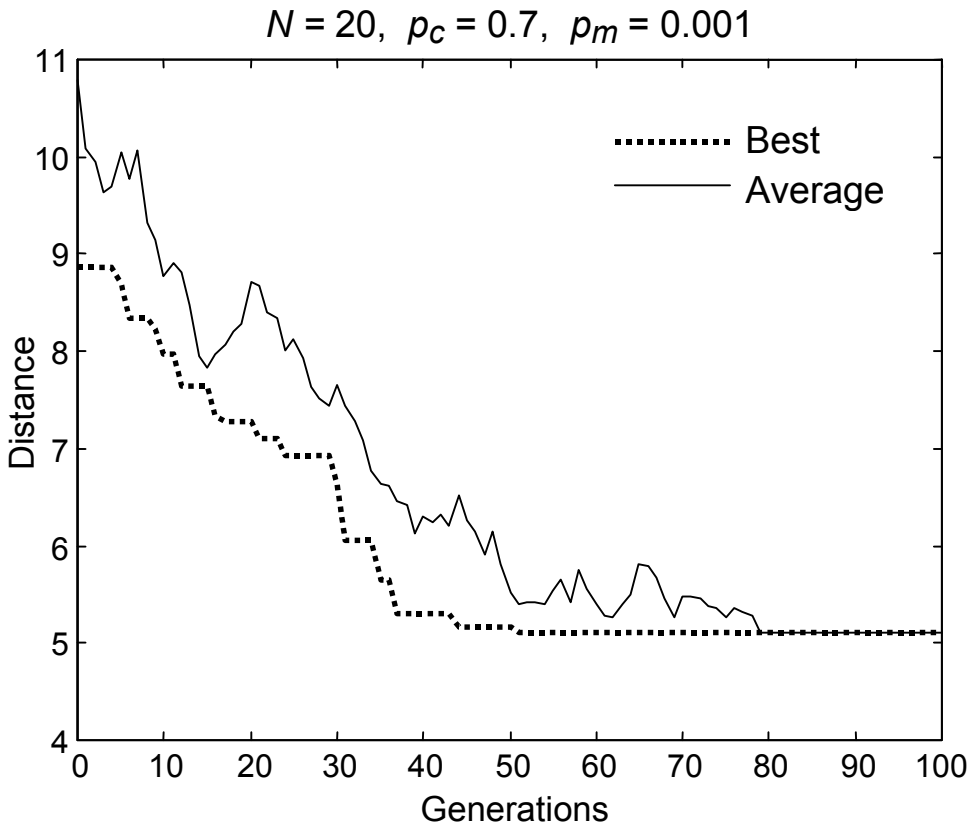


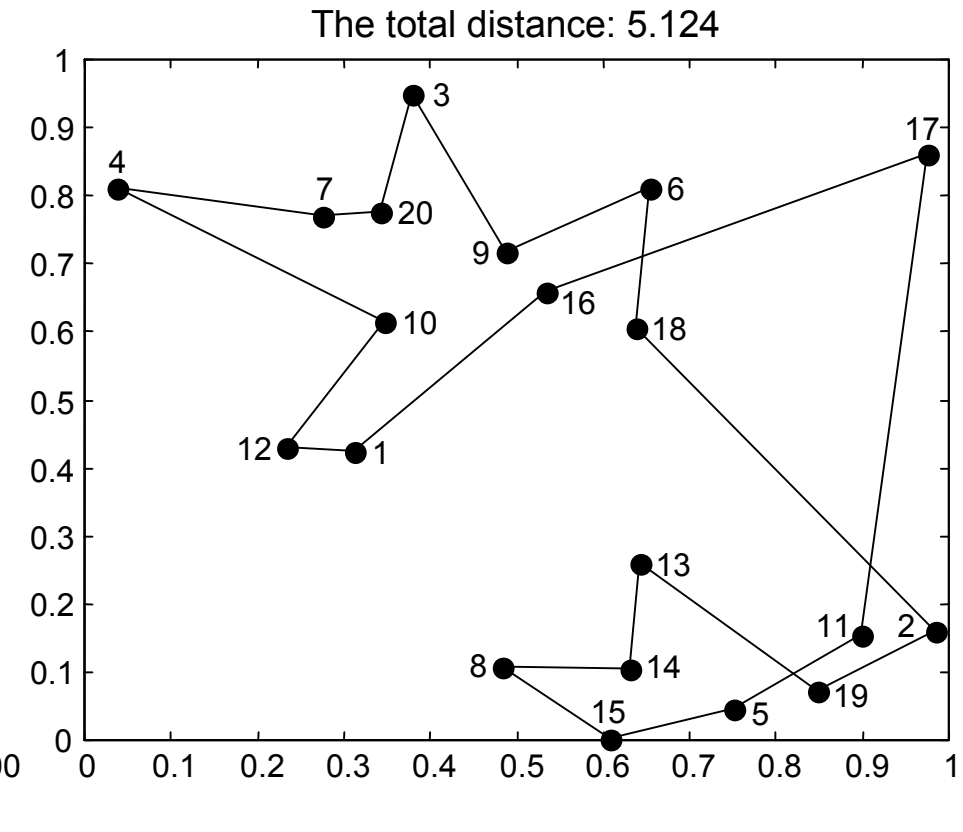
How do we define a fitness function in the TSP?

The fitness of each individual chromosome is determined as the reciprocal of the route length.

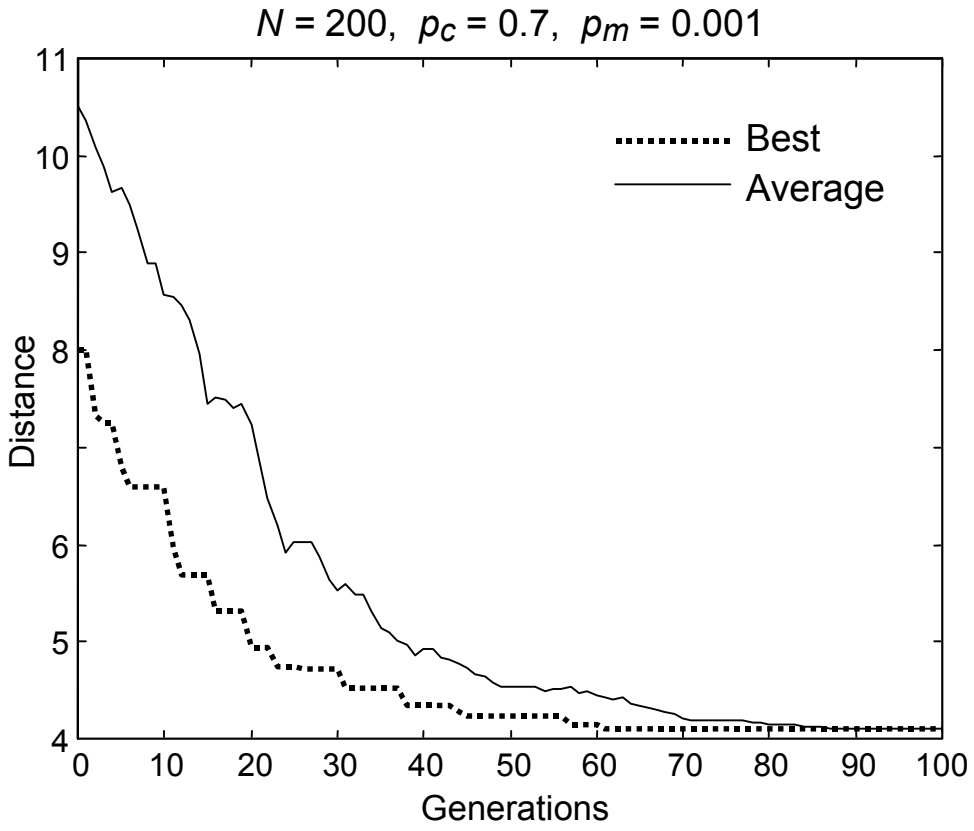
In other words, the shorter the route, the fitter the chromosome.

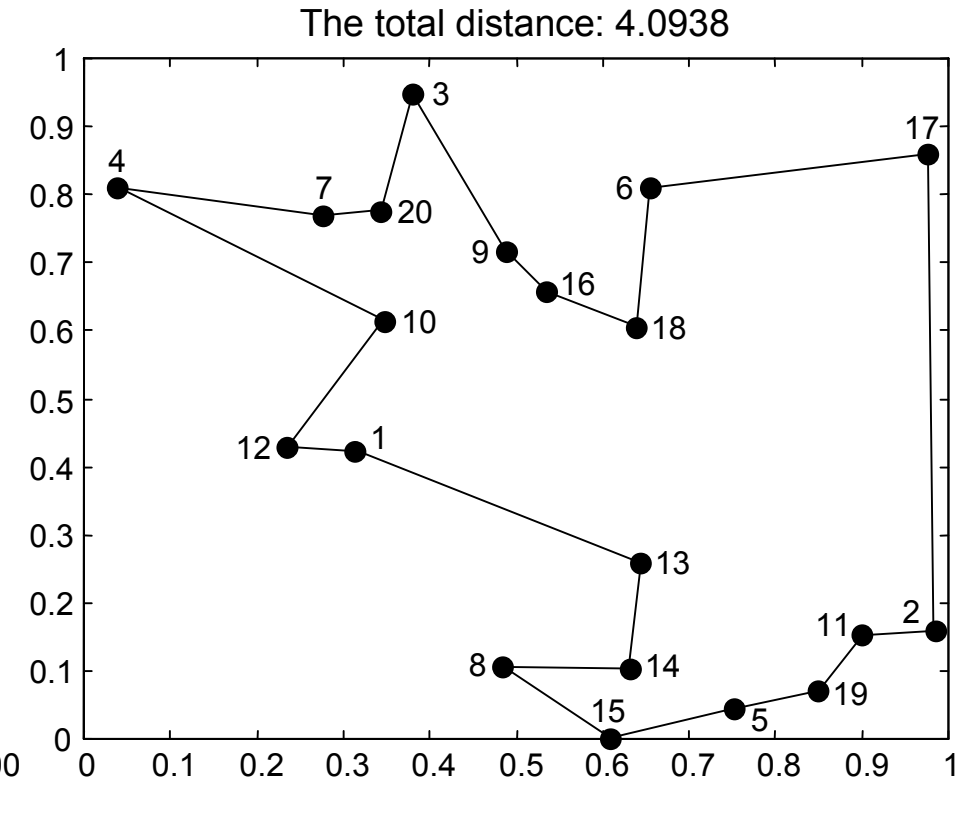
Performance graph and the best salesman's route created in a population of 20 chromosomes after 100 generations



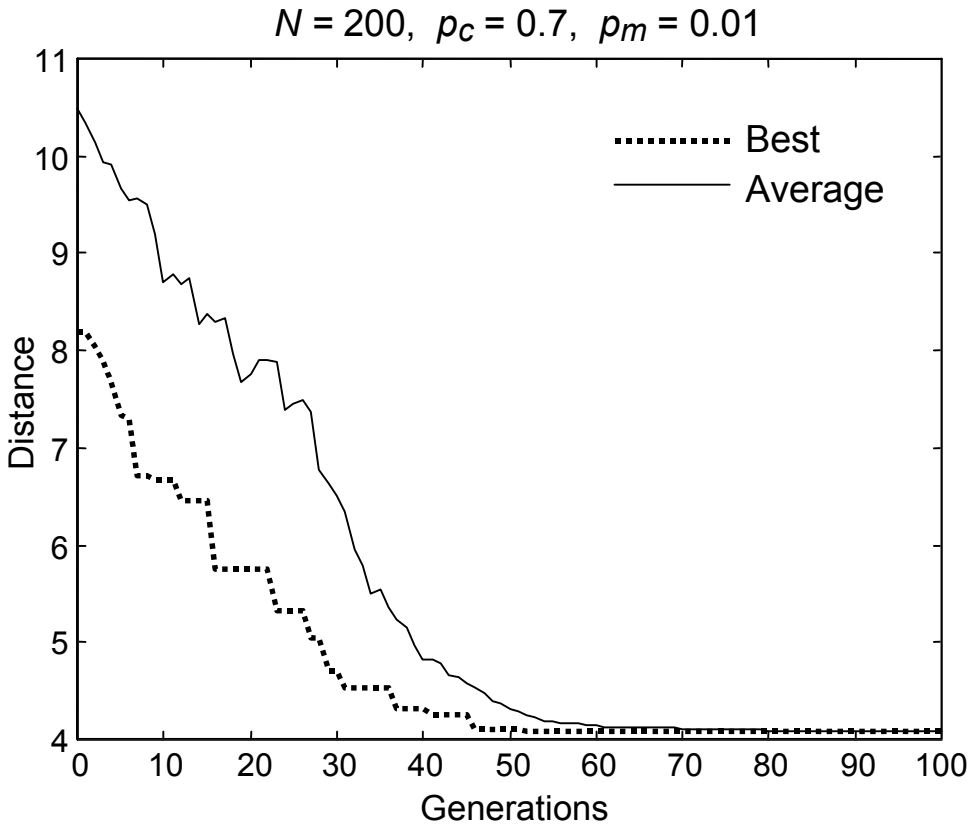


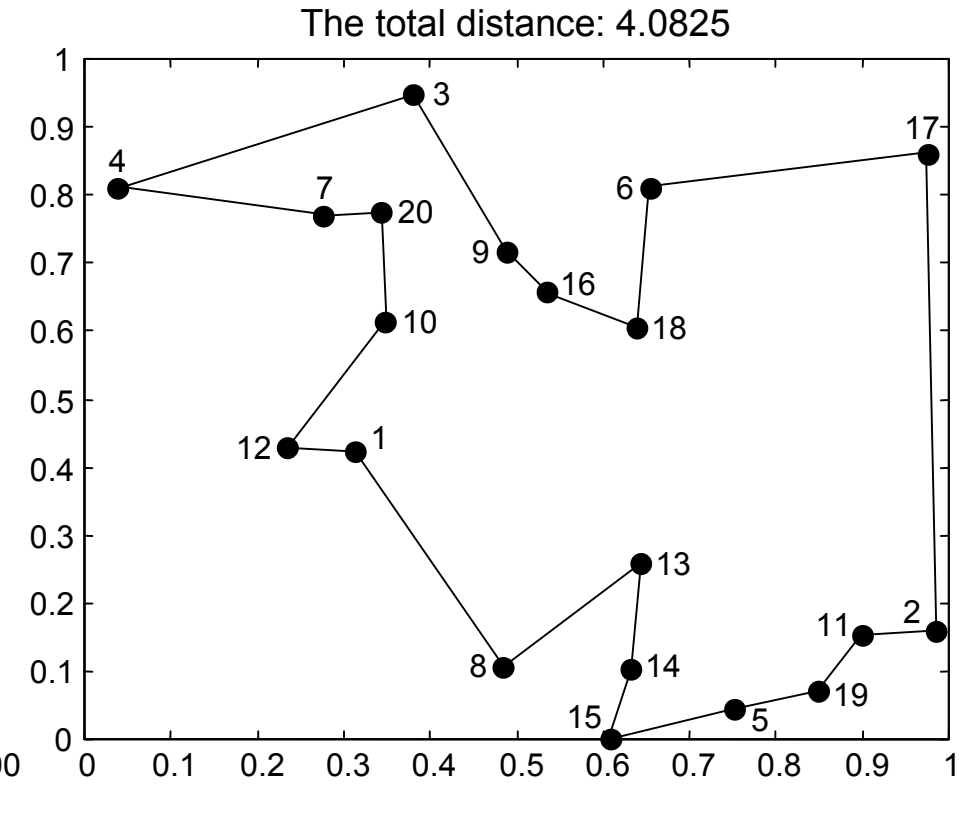
Performance graphs and the best routes created in a population of 200 chromosomes: mutation rate is 0.001





Performance graphs and the best routes created in a population of 200 chromosomes: mutation rate is 0.01





Will a hybrid system work for my problem?

Solving complex real-world problems requires an application of complex intelligent systems that combine the advantages of expert systems, fuzzy logic, neural networks and evolutionary computation. Such systems can integrate human-like expertise in a specific domain with abilities to learn and adapt to a rapidly changing environment.

Case study 9
Neuro-fuzzy decision-support systems

As an example, we will develop an intelligent system for diagnosing myocardial perfusion from cardiac images. Suppose, we have a set of cardiac images as well as the clinical notes and physician's interpretation.

What are SPECT images?

- Diagnosis in modern cardiac medicine is based on the analysis of SPECT (Single Proton Emission Computed Tomography) images.
- By injecting a patient with radioactive tracer, two sets of SPECT images are obtained: one is taken 10 – 15 minutes after the injection when the stress is greatest (stress images), and the other is taken 2 – 5 hours after the injection (rest images). Distribution of the radioactive tracer in the cardiac muscle is proportional to the muscle's perfusion.
- A cardiologist detects abnormalities in the heart function by comparing stress and rest images.

- The SPECT images are usually represented by high resolution two-dimensional black-and-white pictures with up to 256 shades of grey. Brighter patches on the image correspond to well-perfused areas of the myocardium, while darker patches may indicate the presence of an ischemia.
- Unfortunately a visual inspection of the SPECT images is highly subjective; physicians' interpretations are therefore often inconsistent and susceptible to errors.

- For this study, we use 267 cardiac diagnostic cases. Each case is accompanied by two SPECT images (the stress image and the rest image), and each image is divided into 22 regions.
- The region's brightness, which in turn reflects perfusion inside this region, is expressed by an integer number between 0 and 100.
- Thus, each cardiac diagnostic case is represented by 44 continuous features and one binary feature that assigns an overall diagnosis – normal or abnormal.

The SPECT data set

- The entire SPECT data set consists of 55 cases classified as normal (positive examples) and 212 cases classified as abnormal (negative examples).
- The entire set is divided into training and test sets.
- The training set has 40 positive and 40 negative examples. The test set is represented by 15 positive and 172 negative examples.

Can we train a back-propagation neural network to classify the SPECT images into normal and abnormal?

- The number of neurons in the input layer is determined by the total number of regions in the stress and rest images. In this example, each image is divided into 22 regions, so we need 44 input neurons.
- Since SPECT images are to be classified as either normal or abnormal, we use two output neurons.
- A good generalisation is obtained with as little as 5 to 7 neurons in the hidden layer.

- However, when we test the network on the test set, we find that the network's performance is rather poor – about 25 percent of normal cardiac diagnostic cases are misclassified as abnormal and over 35 percent of abnormal cases are misclassified as normal; the overall diagnostic error exceeds 33 percent.
- This indicates that the training set may lack some important examples (a neural network is only as good as the examples used to train it).

Can we improve the accuracy of the diagnosis?

First, we need to redefine the problem. To train the network, we used the same number of positive and negative examples. Although in real clinical trials, the ratio between normal and abnormal SPECT images is very different, the misclassification of an abnormal cardiac case could lead to infinitely more serious consequences than the misclassification of a normal case. Therefore, in order to achieve a small classification error for abnormal SPECT images, we might agree to have a relatively large error for normal images.

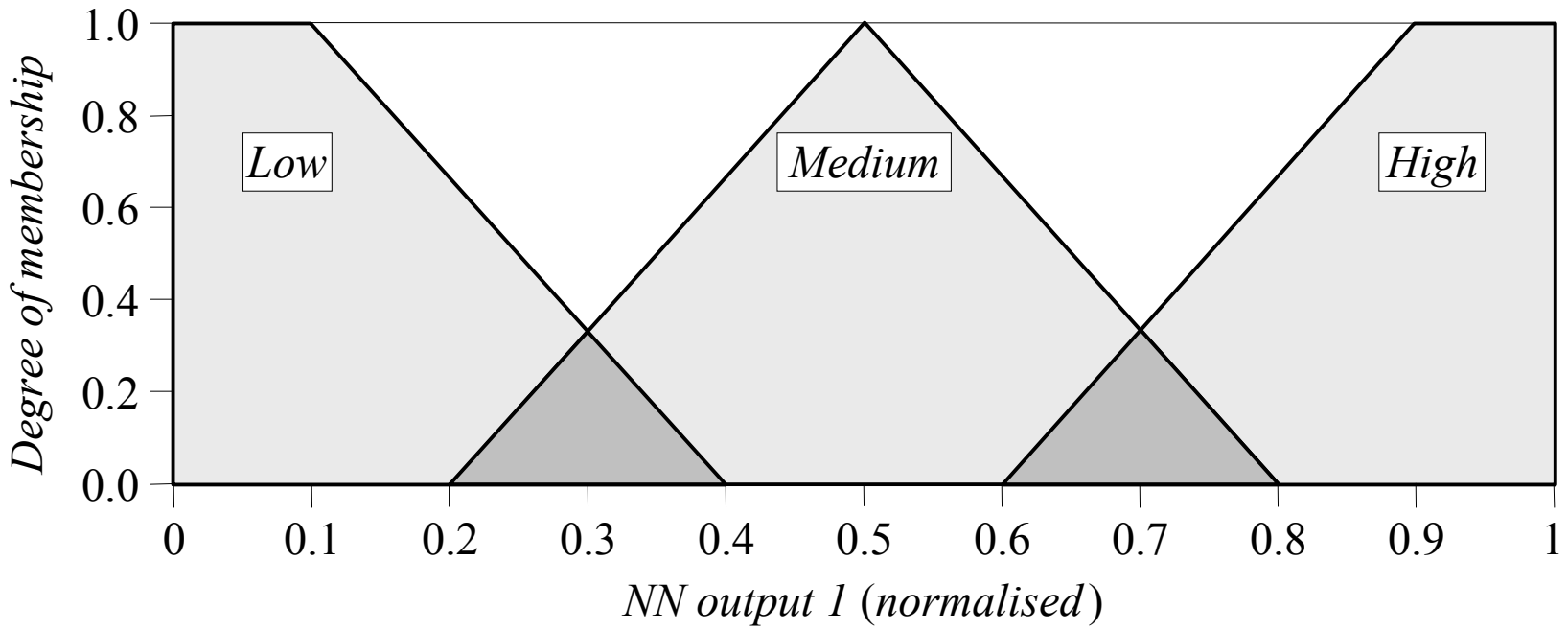
- The neural network produces two outputs. The first output corresponds to the possibility that the SPECT image belongs to the class *normal*, and the second to the possibility that the image belongs to the class *abnormal*. If, for example, the first (*normal*) output is 0.92 and the second (*abnormal*) is 0.16, the SPECT image is classified as normal, and we can conclude that the risk of a heart attack for this case is low.

- On the other hand, if the *normal* output is low, say 0.17, and the *abnormal* output is much higher, say 0.51, the SPECT image is classified as abnormal, and we can infer that the risk of a heart attack in this case is rather high. However, if the two outputs are close – say the *normal* output is 0.51 and the *abnormal* 0.49 – we cannot confidently classify the image.

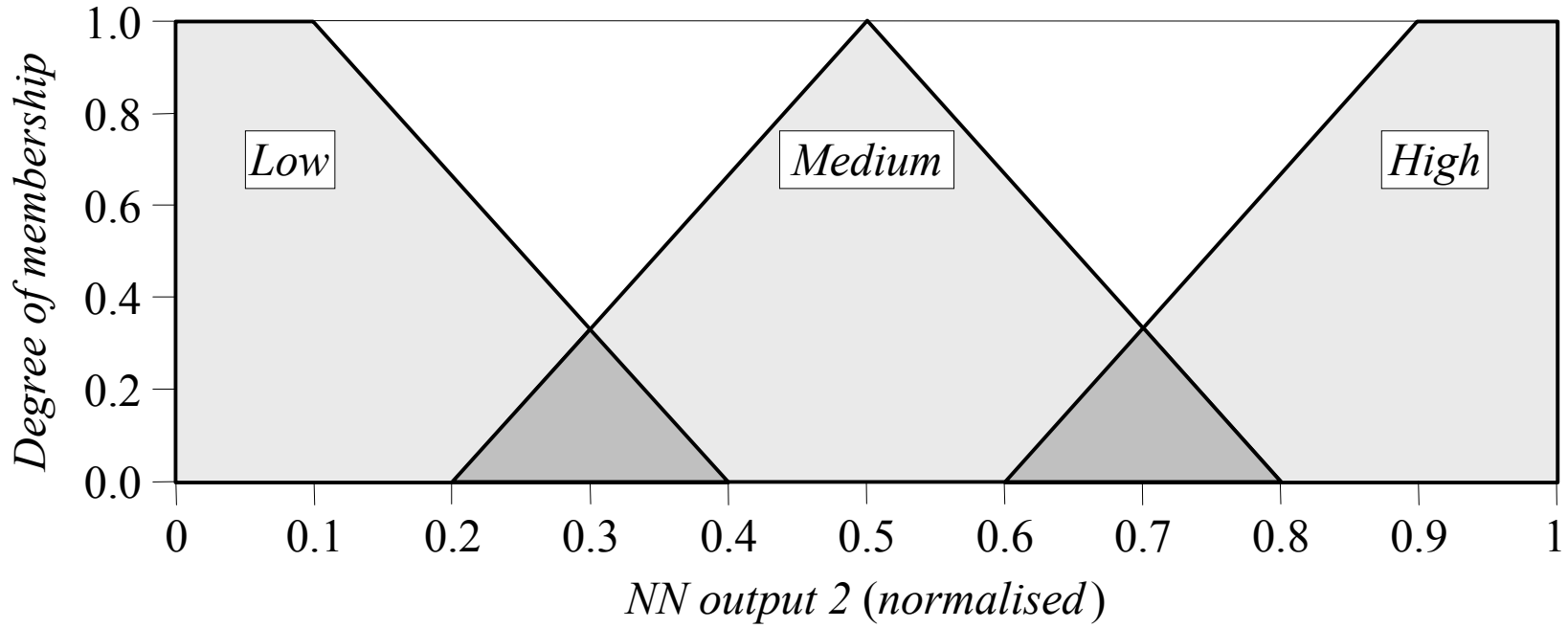
Can we use fuzzy logic for decision-making in medical diagnosis?

- To build a fuzzy system, we first need to determine input and output variables, define fuzzy sets and construct fuzzy rules.
- For our problem, there are two inputs (*NN output 1* and *NN output 2*) and one output (the *risk* of a heart attack).
- The inputs are normalised to be within the range of [0, 1], and the output can vary between 0 and 100 percent.

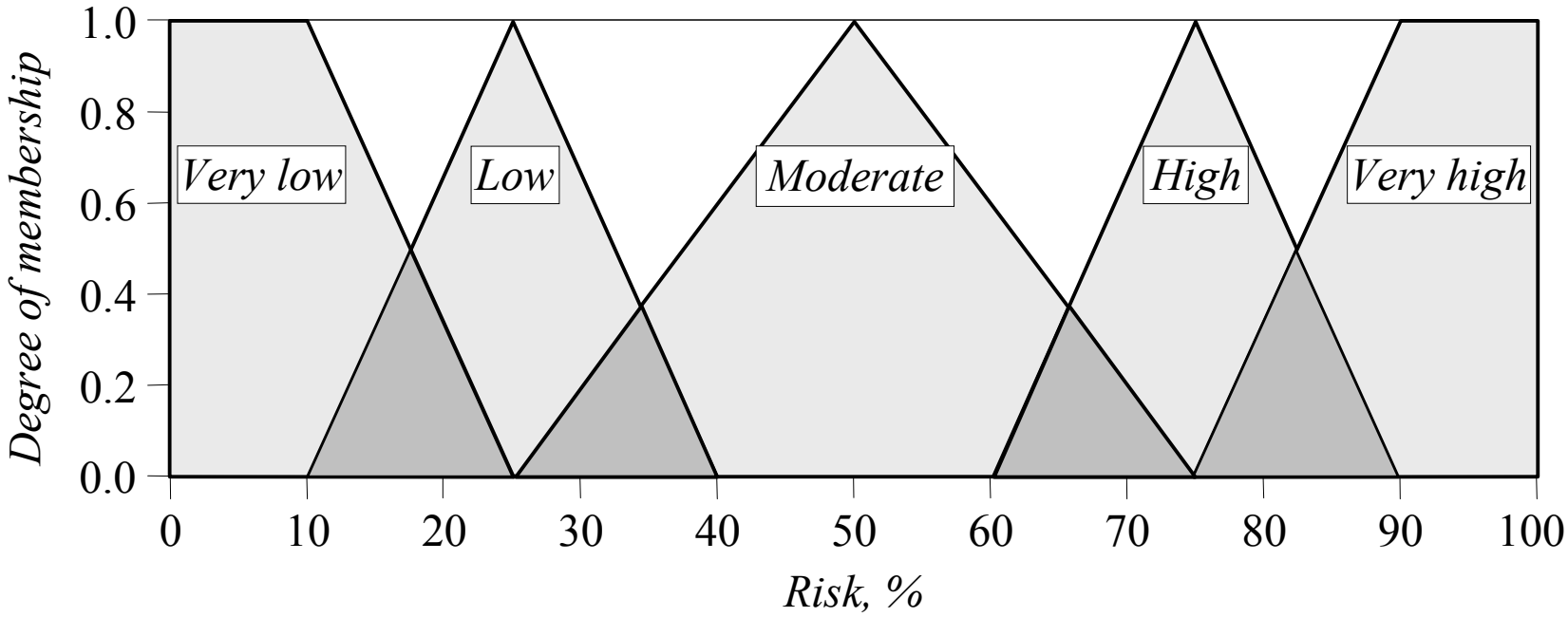
Fuzzy sets of the neural network output *normal*



Fuzzy sets of the neural network output *abnormal*



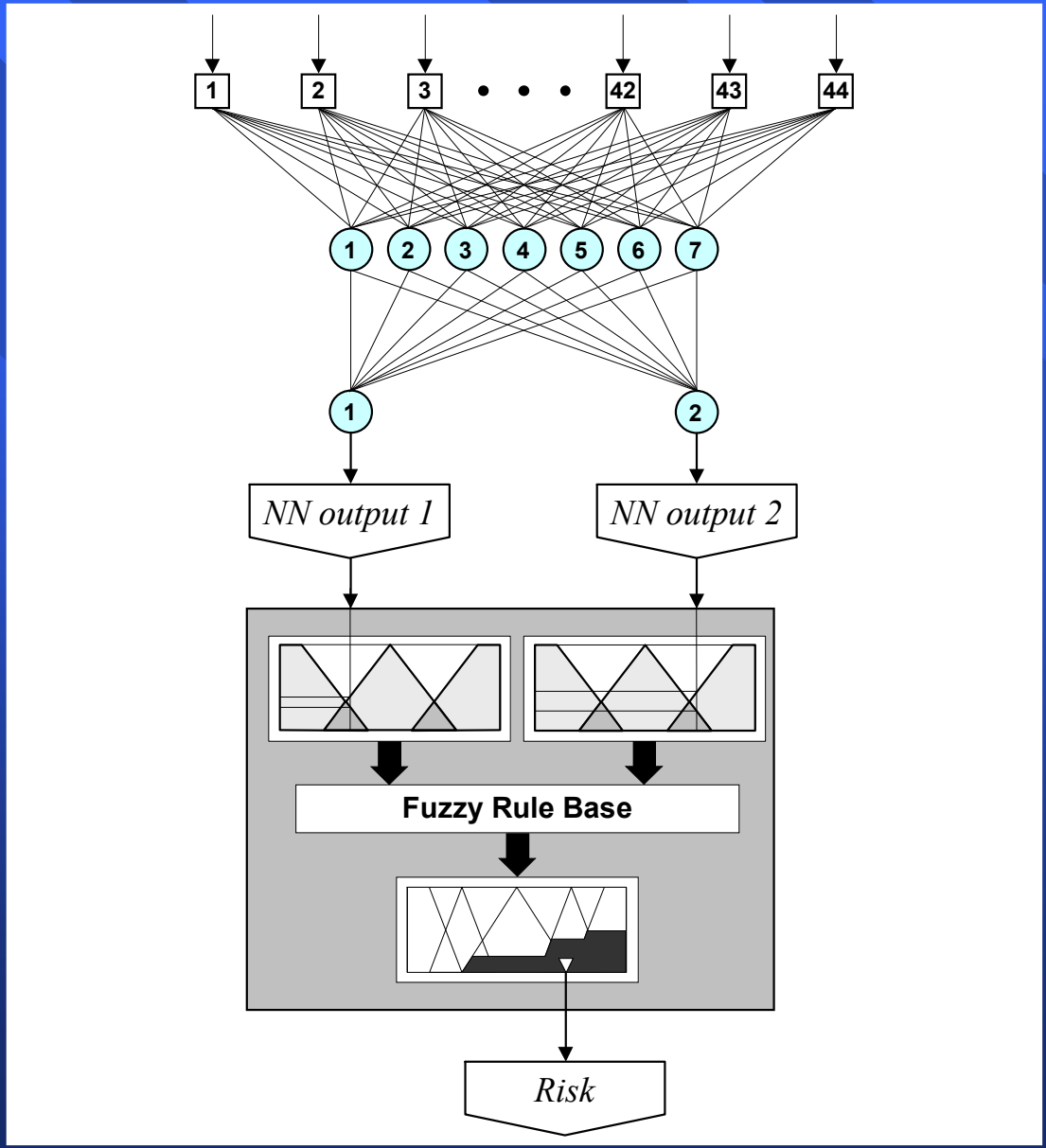
Fuzzy sets of the linguistic variable *Risk*



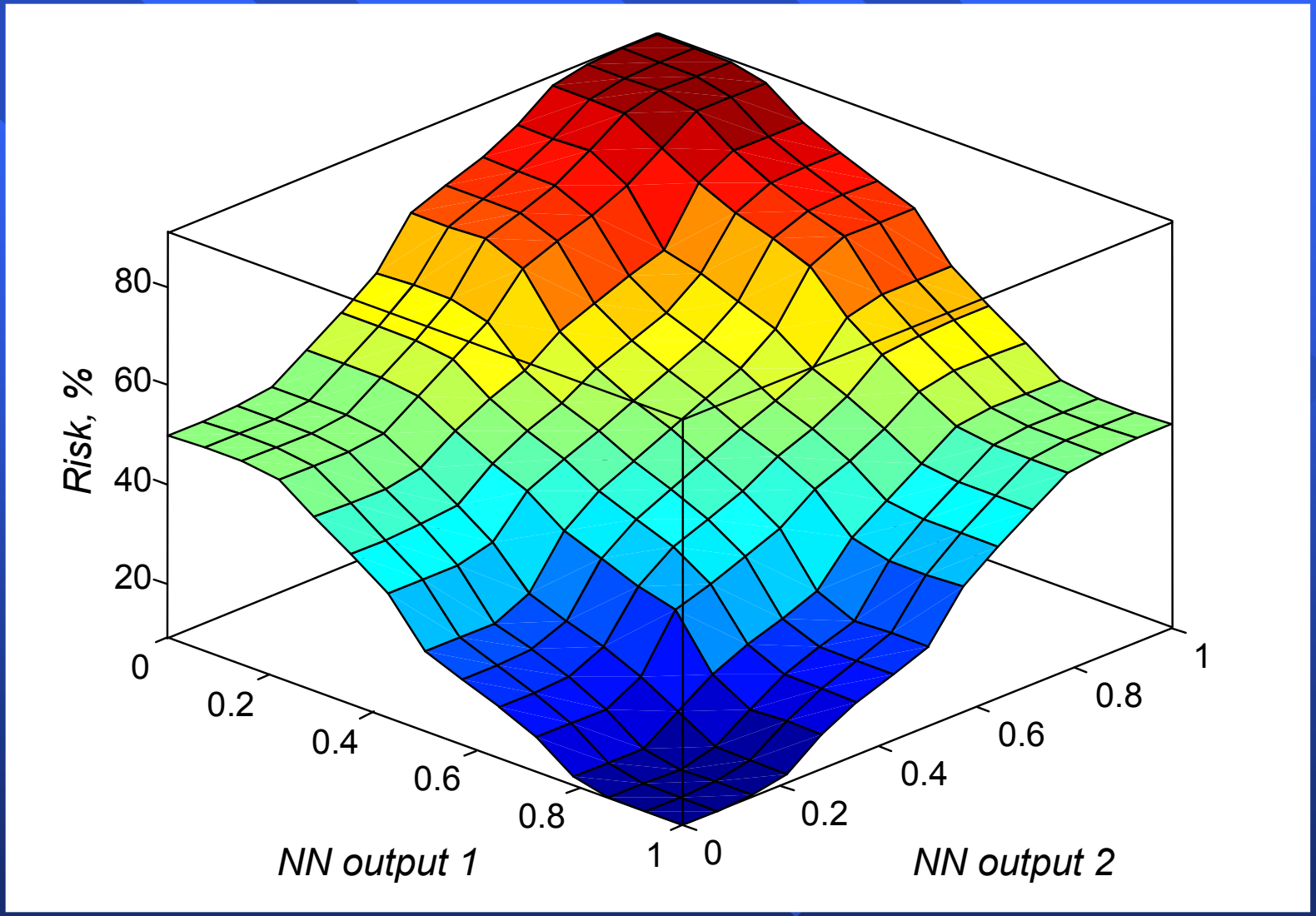
Fuzzy rules for assessing the risk of a heart decease

1. If (NN_output1 is Low) and (NN_output2 is Low) then (Risk is Moderate)
2. If (NN_output1 is Low) and (NN_output2 is Medium) then (Risk is High)
3. If (NN_output1 is Low) and (NN_output2 is High) then (Risk is Very_high)
4. If (NN_output1 is Medium) and (NN_output2 is Low) then (Risk is Low)
5. If (NN_output1 is Medium) and (NN_output2 is Medium) then (Risk is Moderate)
6. If (NN_output1 is Medium) and (NN_output2 is High) then (Risk is High)
7. If (NN_output1 is High) and (NN_output2 is Low) then (Risk is Very_low)
8. If (NN_output1 is High) and (NN_output2 is Medium) then (Risk is Low)
9. If (NN_output1 is High) and (NN_output2 is High) then (Risk is Moderate)

Structure of the neuro-fuzzy system



Three-dimensional plot for the fuzzy rule base



- The system's output is a crisp number that represents a patient's risk of a heart attack.
- Based on this number, a cardiologist can now classify cardiac cases with greater certainty – when the risk is quantified, a decision-maker has a much better chance of making the right decision. For instance, if the risk is low, say, smaller than 30 percent, the cardiac case can be classified as ***normal***, but if the risk is high, say, greater than 50 percent, the case is classified as ***abnormal***.

However, cardiac cases with the risk between 30 and 50 percent cannot be classified as either ***normal*** or ***abormal*** – rather, such cases are ***uncertain***.

Can we classify some of the uncertain cases using the knowledge of a cardiologist?

A cardiologist knows that, in normal heart muscle, perfusion at maximum stress is usually higher than perfusion at rest:

- If perfusion inside region *i* at stress is higher than perfusion inside the same region at rest, then the risk of a heart attack should be decreased.
- If perfusion inside region *i* at stress is not higher than perfusion inside the same region at rest, then the risk of a heart attack should be increased.

These heuristics can be implemented in the diagnostic system as follows:

Step 1: Present the neuro-fuzzy system with a cardiac case.

Step 2: If the system's output is less than 30, classify the presented case as ***normal*** and then stop. If the output is greater than 50, classify the case as ***abnormal*** and stop. Otherwise, go to Step 3.

Step 3: For region 1, subtract perfusion at rest from perfusion at stress. If the result is positive, decrease the current risk by multiplying its value by 0.99. Otherwise, increase the risk by multiplying its value by 1.01. Repeat this procedure for all 22 regions and then go to Step 4.

Step 4: If the new risk value is less than 30, classify the case as *normal*; if the risk is greater than 50, classify the case as ***abnormal***; otherwise – classify the case as ***uncertain***.

- The accuracy of diagnosis has dramatically improved – the overall diagnostic error does not exceed 5 percent, while only 3 percent of abnormal cardiac cases are misclassified as normal.
- Although we have not improved the system's performance on normal cases (over 30 percent of normal cases are still misclassified as abnormal), and up to 20 percent of the total number of cases are classified as uncertain, the neuro-fuzzy system can actually achieve even better results in classifying SPECT images than a cardiologist can.

- In this example, the neuro-fuzzy system has a ***heterogeneous*** structure – the neural network and fuzzy system work as independent components (although they cooperate in solving the problem). When a new case is presented to the diagnostic system, the trained neural network determines inputs to the fuzzy system. Then the fuzzy system using predefined fuzzy sets and fuzzy rules, maps the given inputs to an output, and thereby obtains the risk of a heart attack.
- Adaptive Neuro-Fuzzy Inference System (ANFIS) is a typical example of a neuro-fuzzy system with a ***homogeneous*** structure.

Case study 10
Time-series prediction

As an example, we will develop a tool to predict an aircraft's trajectory during its landing aboard an aircraft carrier.

Suppose, we have a database of landing trajectories of various aircraft flown by different pilots, and we also can use RADAR numerical data, which provide real-time trajectories of landing aircraft. Our goal is to predict an aircraft's trajectory at least two seconds in advance, based on the aircraft's current position.

Prediction of the aircraft's position

- The landing of an aircraft, particularly aboard aircraft carriers, is an extremely complex process.
- It is affected by such variables as the flight deck's space constraints and its motions (both pitch and roll), the aircraft's ordinance and fuel load, continuous mechanical preparations, and the most critical of all – time constraints.
- The ship may be heaving 10 feet up and 10 feet down, making a 20-foot displacement from a level deck. In addition, it is difficult to see approaching aircraft at night or during stormy conditions.

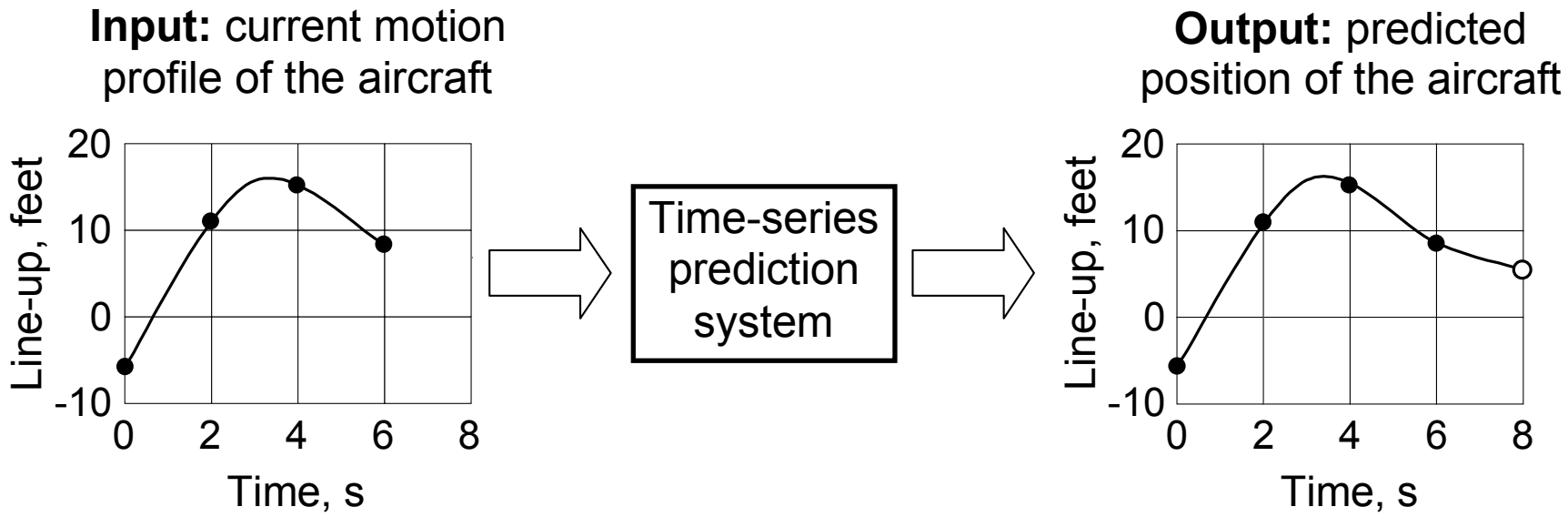
- Responsibility for the aircraft's final approach and landing lies with the Landing Signal Officer (LSO).
- When an aircraft is within one nautical mile of the landing deck, which roughly corresponds to 60 seconds in real time, the aircraft's flight is carefully observed and guided. During this critical time, the LSO needs to predict the aircraft's position at least two seconds ahead.
- Such problems are known in mathematics as ***time-series prediction*** problems.

What is a time series?

- A time series can be defined as a set of observations, each one being recorded at a specific time. For instance, a time series can be obtained by recording the aircraft's positions over a time interval of, say, 60 seconds before landing.
- Real-world time-series problems are non-linear and often exhibit chaotic behaviour, which make them hard to model.

- Prediction of the aircraft's landing trajectory is mainly based on the experience of a LSO (all LSOs are trained pilots).
- An automatic prediction system can use aircraft-position data given by the ship's RADAR, and also data records of previous landings executed by pilots flying different types of aircraft.
- The system is trained off-line with the past data. Then it is presented on-line with the current motion profile, and required to predict the aircraft's motion in the next few seconds.

On-line time-series prediction of an aircraft's trajectory



To predict an aircraft's position on-line we will use an ANFIS.

What do we use as ANFIS inputs?

To predict a future value for a time series, we use values that are already known. For example, if we want to predict an aircraft's position two seconds ahead, we may use its current position data as well as data recorded, say, 2, 4 and 6 seconds before the current position. These four known values represent an input pattern – a four-dimensional vector of the following form:

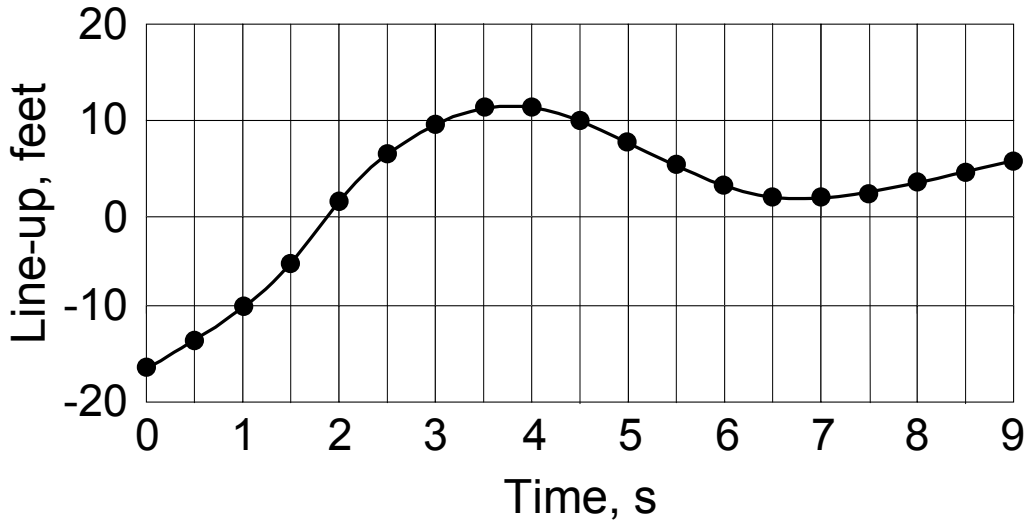
$$\mathbf{x} = \left[x(t-6) \;\; x(t-4) \;\; x(t-2) \;\; x(t) \right]$$

What is the ANFIS output?

The ANFIS output corresponds to the trajectory prediction: the aircraft's position two seconds ahead, $x(t + 2)$.

For this case study, we will use 10 landing trajectories – five for training and five for testing. Each trajectory is a time series of the aircraft's position data points recorded every half a second over a time interval of 60 seconds before landing. Thus, a data set for each trajectory contains 121 values.

An aircraft trajectory and a data set built to train the ANFIS



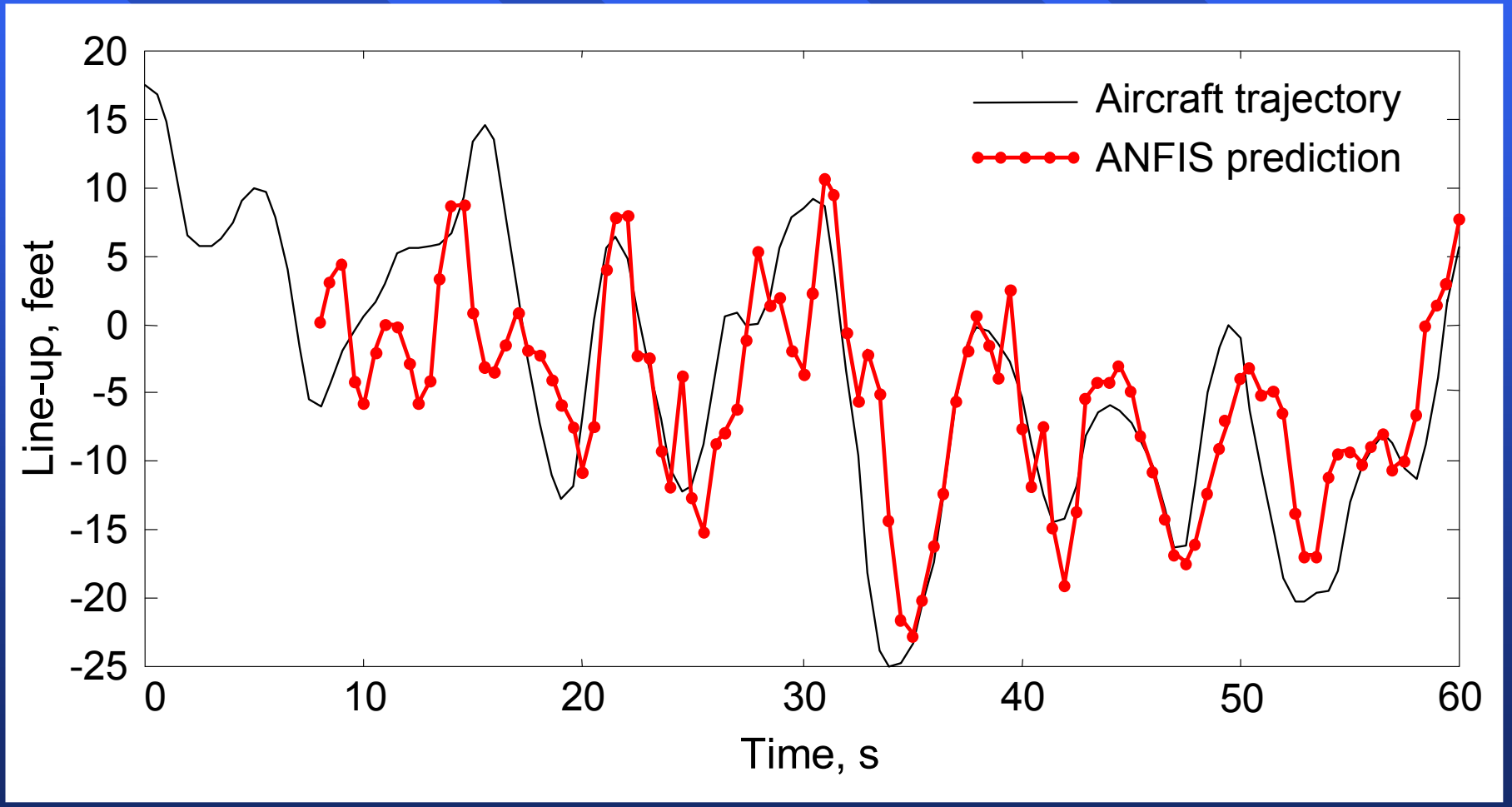
Inputs				Desired output
x_1	x_2	x_3	x_4	
–17.4	2.1	11.0	3.9	4.2
–12.9	7.5	10.1	2.1	4.9
–10.0	9.8	8.2	2.0	5.3

For a landing trajectory recorded over a time interval of 60 seconds, we obtain 105 training samples represented by a 105 × 5 matrix. Thus, the entire data set, which we use for training the ANFIS, is represented by a 525 × 5 matrix.

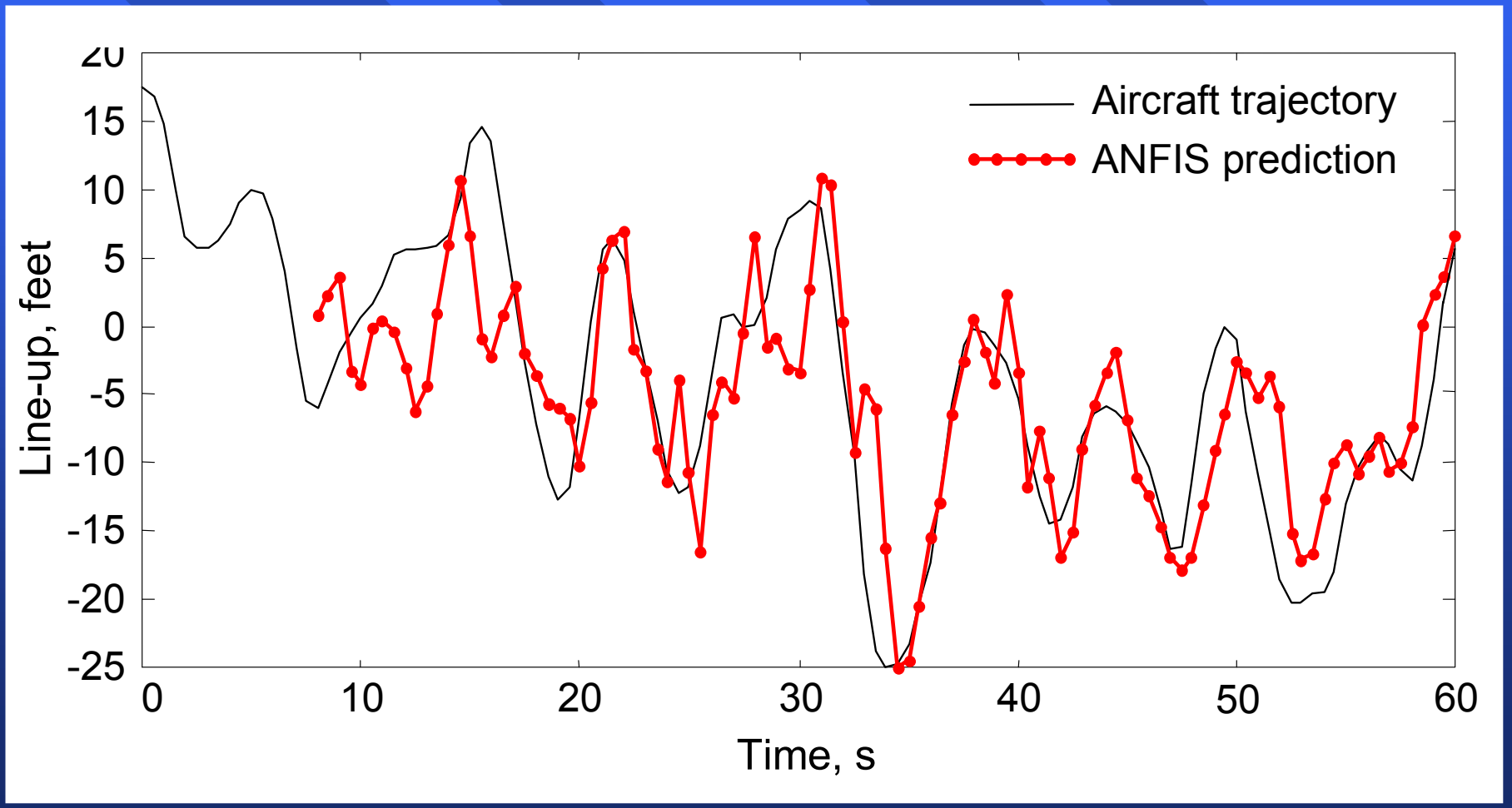
How many membership functions should we assign to each input variable?

A practical approach is to choose the smallest number of membership functions. Thus, we may begin with two membership functions assigned to each input variable.

Performance of the ANFIS with four inputs and two membership functions assigned to each input: one epoch



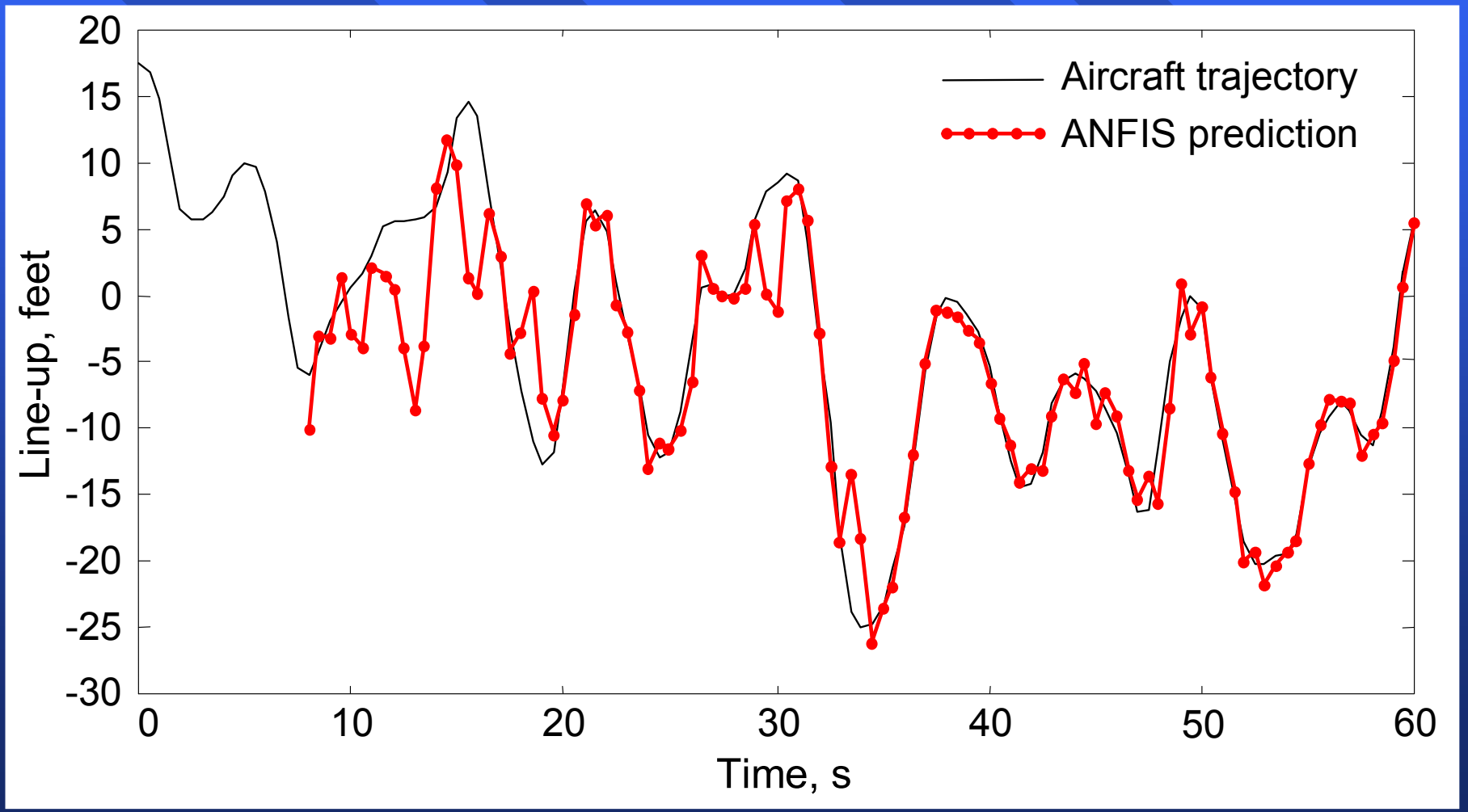
Performance of the ANFIS with four inputs and two membership functions assigned to each input: 100 epochs



How can we improve the ANFIS's performance?

The ANFIS's performance can be significantly improved by assigning ***three membership functions to each input variable***.

Performance of the ANFIS with four inputs and three membership functions assigned to each input after one epoch of training

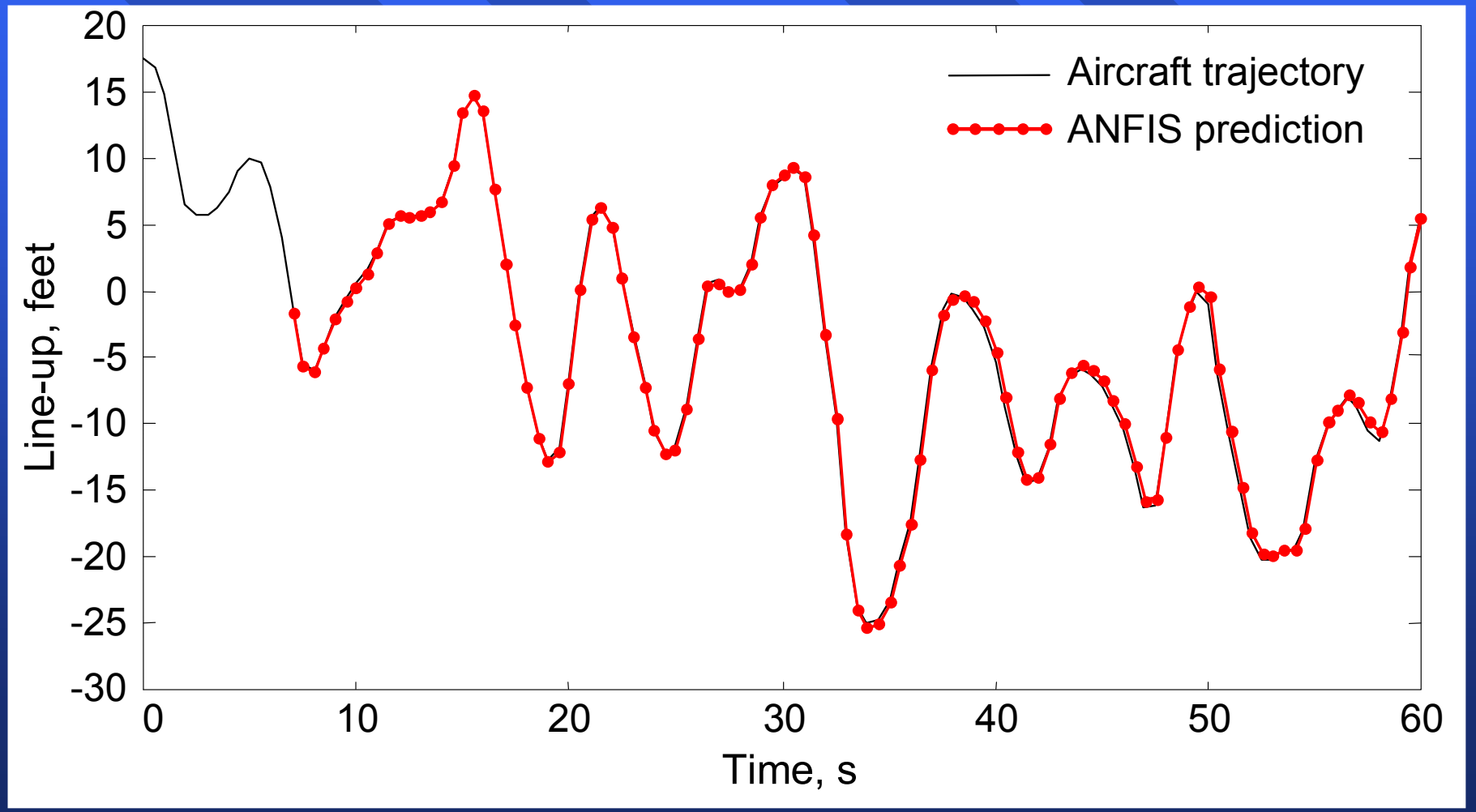


Another way of improving time-series prediction is to ***increase the number of input variables***.

Let us, for example, examine an ANFIS with six inputs that correspond to the aircraft's flight positions at $(t-5)$, $(t-4)$, $(t-3)$, $(t-2)$, $(t-1)$, and t, respectively. The ANFIS output still remains the two-second prediction.

The training data set is now represented by a 535×7 matrix.

Performance of the ANFIS with six inputs and two membership functions assigned to each input: prediction after one epoch



Performance of the ANFIS with six inputs and two membership functions assigned to each input: prediction errors