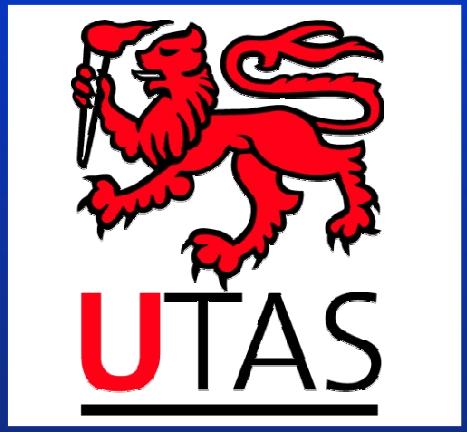
Hybrid Neuro-Fuzzy Systems or How to Combine German Mechanics with Italian Love

by

Professor Michael Negnevitsky



University of Tasmania

Contents

- Introduction
- Heterogeneous Hybrid Systems
- Diagnosis of myocardial perfusion
- System evaluation
- Homogeneous neuro-fuzzy system
- Landing of an aircraft as a time-series prediction problem
- Conclusions

Introduction

- A **hybrid intelligent system** is one that combines at least two intelligent technologies. For example, combining a neural network with a fuzzy system results in a hybrid neuro-fuzzy system.
- Lotfi Zadeh is reputed to have said that a good hybrid would be "British Police, German Mechanics, French Cuisine, Swiss Banking and Italian Love".
- But…

- … “British Cuisine, German Police, French Mechanics, Italian Banking and Swiss Love” would be a bad one.
- Likewise, a hybrid intelligent system can be good or bad – it depends on which components constitute the hybrid. So our goal is to select the right components for building a good hybrid system.

Comparison of Fuzzy Systems and Neural Networks

	FS	*NN*
Knowledge representation	☑	☐
Uncertainty tolerance	☑	☑
Imprecision tolerance	☑	☑
Adaptability	☑	☑
Learning ability	☐	☑
Explanation ability	☑	☐

* The terms used for grading are:

☐ - bad ☑ - good

Heterogeneous Hybrid Systems

A heterogeneous neuro-fuzzy system is hybrid system that consists of a neural network and a fuzzy system working as independent components.

As an example of the application of such a system, we will consider a problem of diagnosing myocardial perfusion from cardiac images.

As an example of the application of such a system, we will consider a problem of 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.

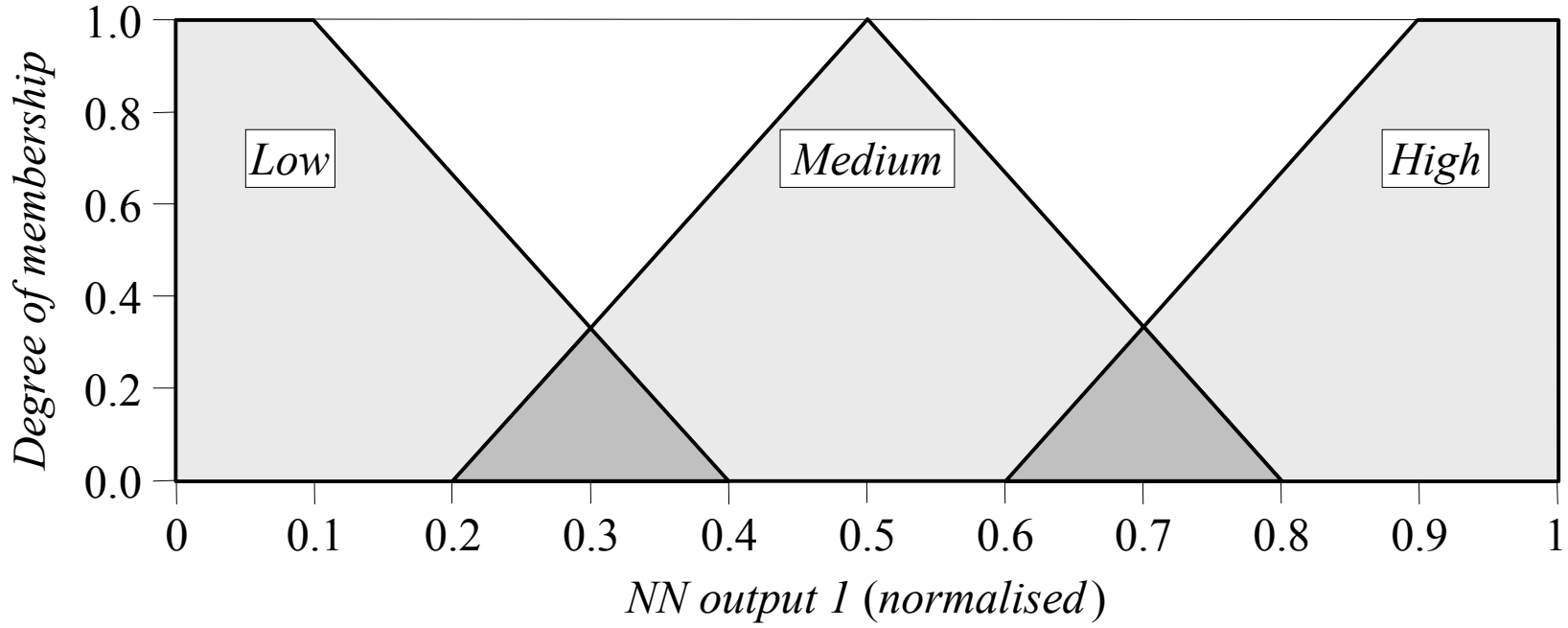
- The neural network produces two outputs. The first output corresponds to the possibility that the SPECT image belongs to the class *normal*, and the second to the possibility that the image belongs to the class *abnormal*. If, for example, the first (*normal*) output is 0.92 and the second (*abnormal*) 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.

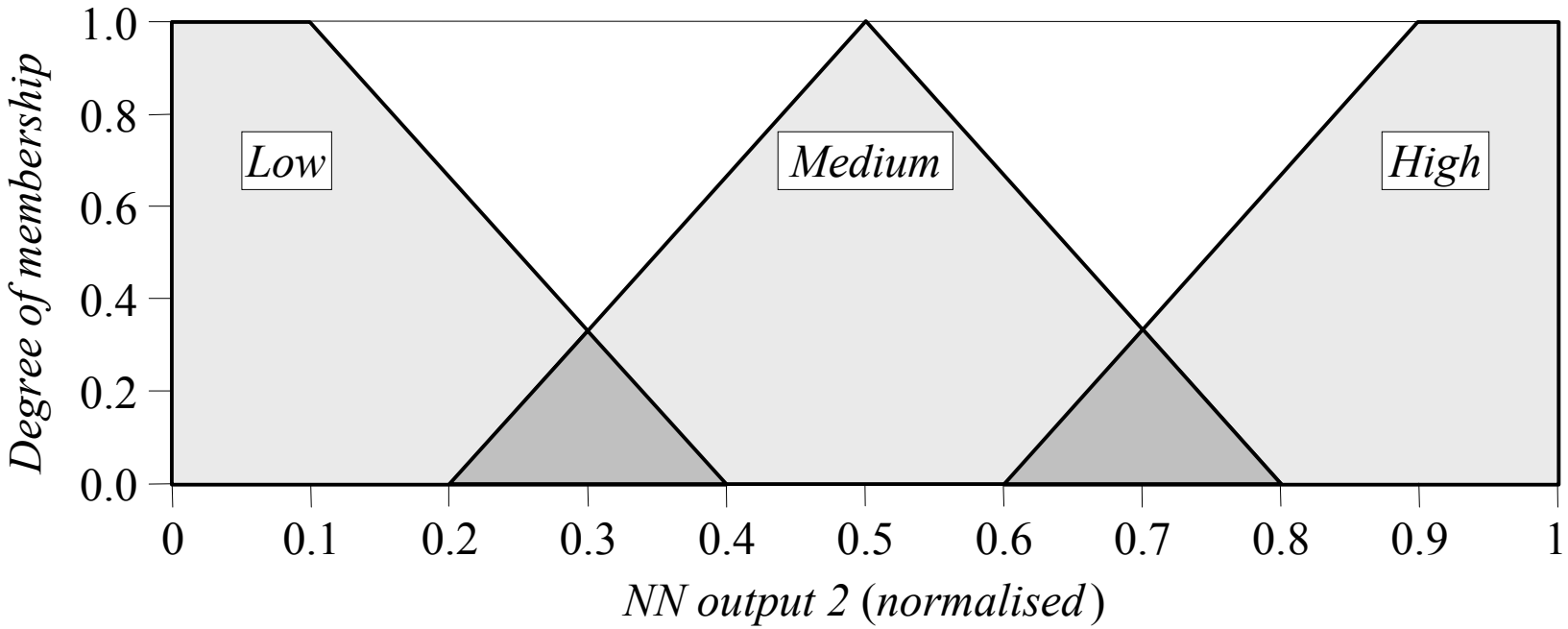
Can we use fuzzy logic for decision-making in medical diagnosis?

- 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.

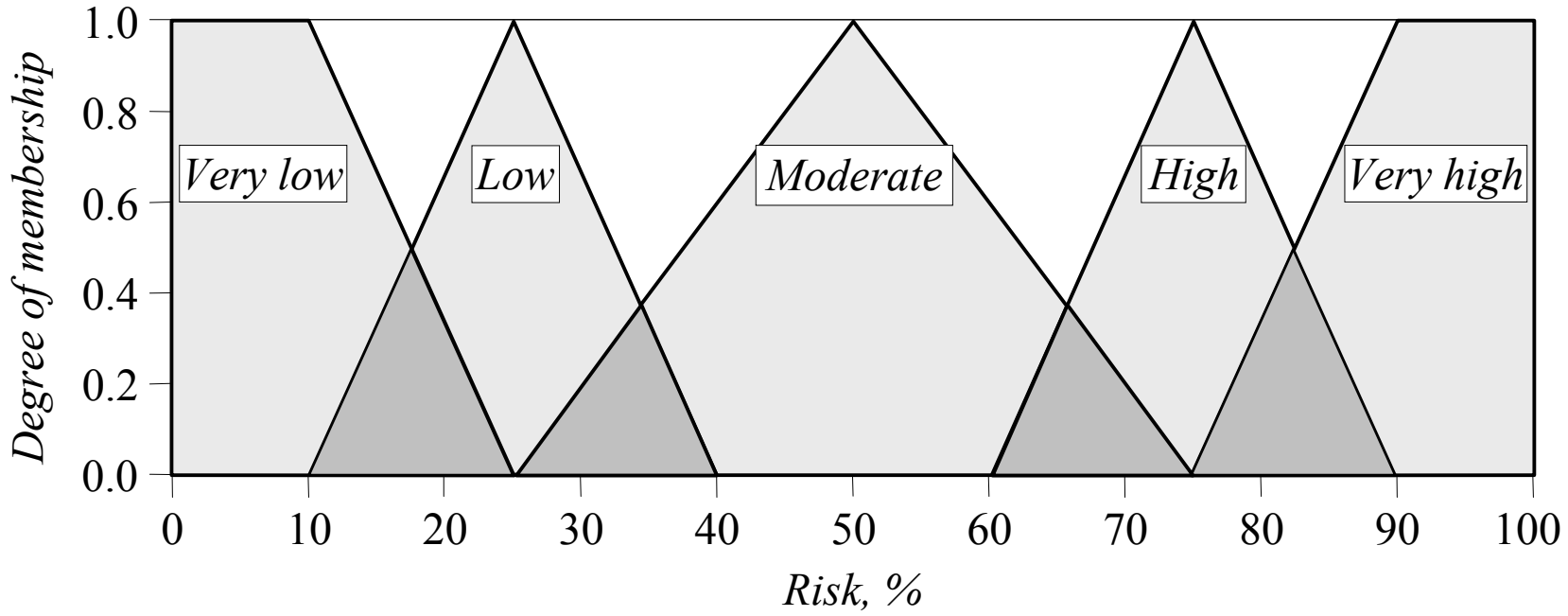
Fuzzy sets of the neural network output *normal*



Fuzzy sets of the neural network output *abnormal*



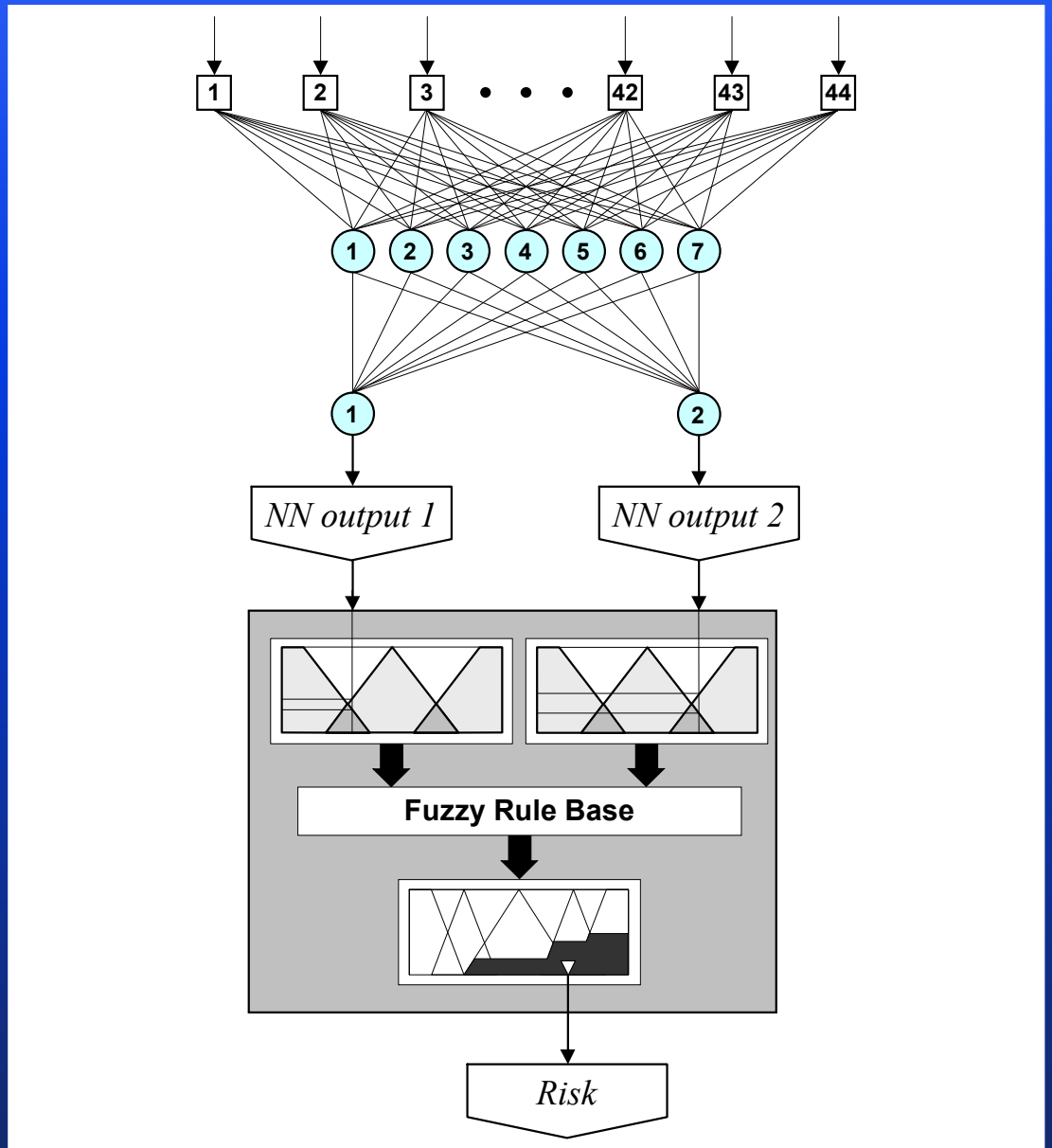
Fuzzy sets of the linguistic variable *Risk*



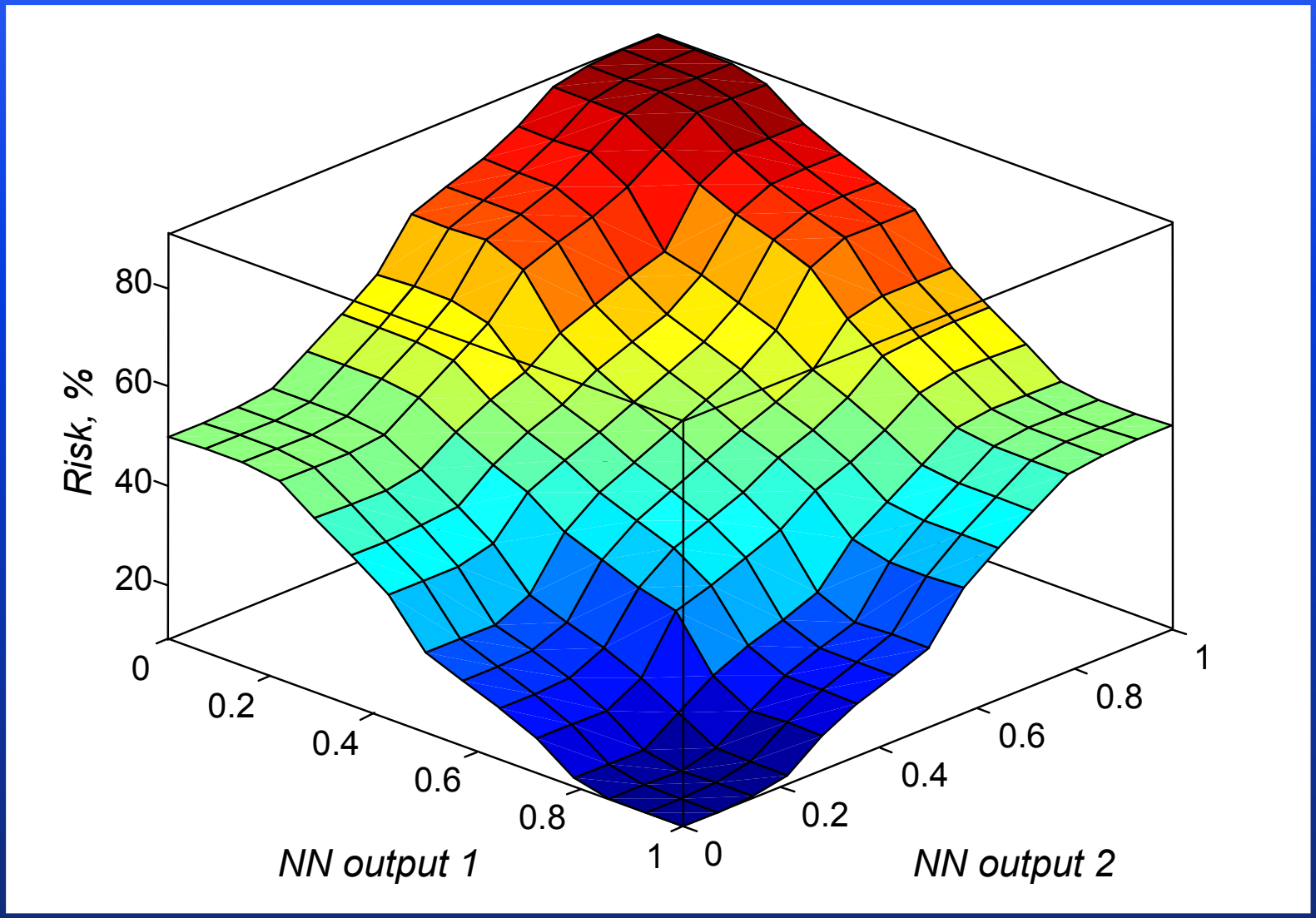
Fuzzy rules for assessing the risk of a heart decease

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***.

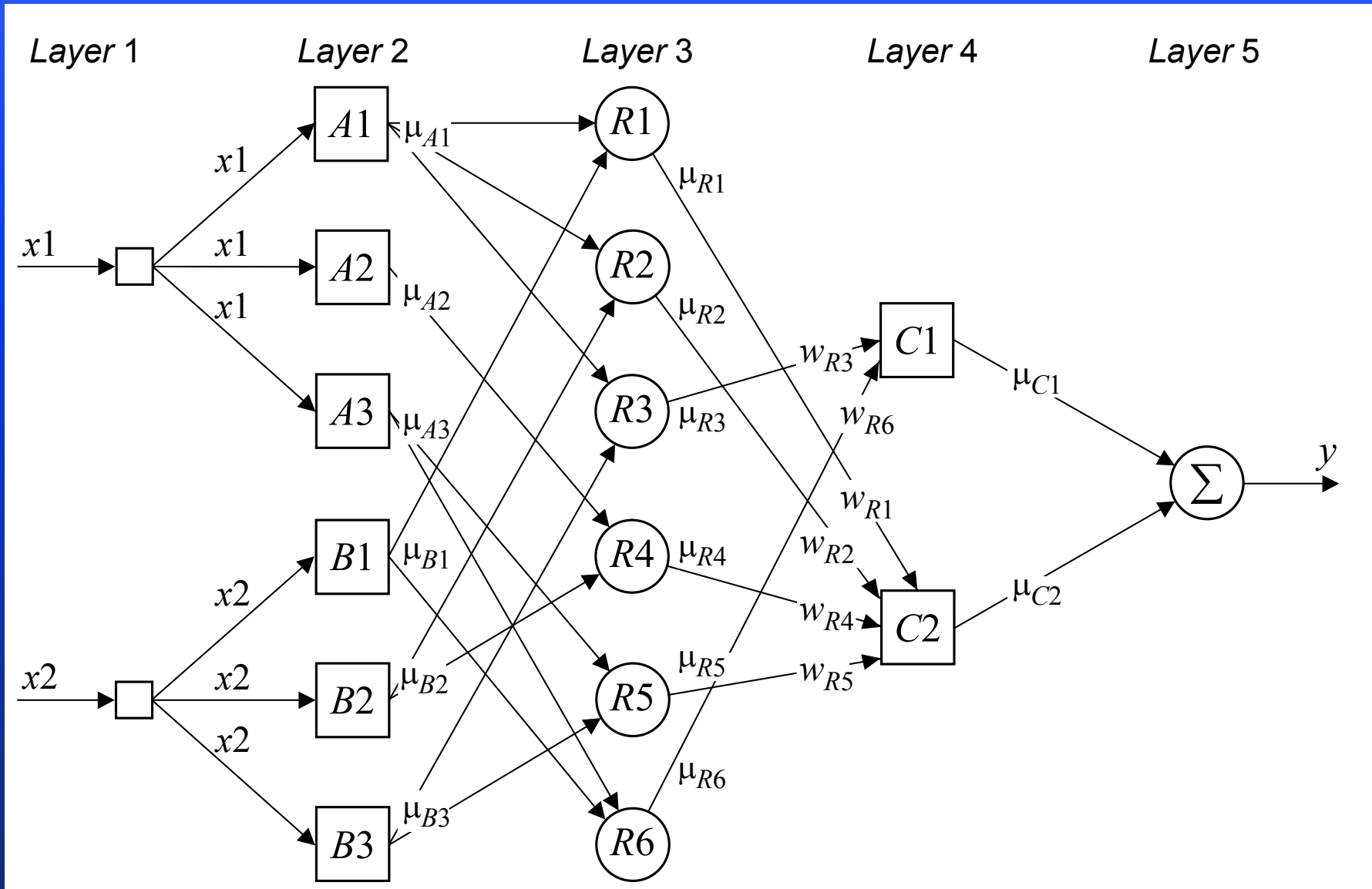
- The accuracy of diagnosis has dramatically improved – the overall diagnostic error does not exceed 5 percent, while only 3 percent of abnormal cardiac cases are misclassified as normal.
- Although we have not improved the system's performance on normal cases (over 30 percent of normal cases are still 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.

Homogeneous Hybrid Systems

- A homogeneous neuro-fuzzy system is a neural network that is functionally equivalent to a fuzzy inference model.
- It can be trained to develop IF-THEN fuzzy rules and determine membership functions for input and output variables of the system.
- Expert knowledge can be easily incorporated into the structure of the neuro-fuzzy system. At the same time, the connectionist structure avoids fuzzy inference, which entails a substantial computational burden.

- The structure of a homogeneous neuro-fuzzy system is similar to a multi-layer neural network.
- In general, a neuro-fuzzy system has input and output layers, and three hidden layers that represent membership functions and fuzzy rules.

Neuro-fuzzy system



Each layer in the neuro-fuzzy system is associated with a particular step in the fuzzy inference process.

Layer 1 is the **input layer**. Each neuron in this layer transmits external crisp signals directly to the next layer. That is,

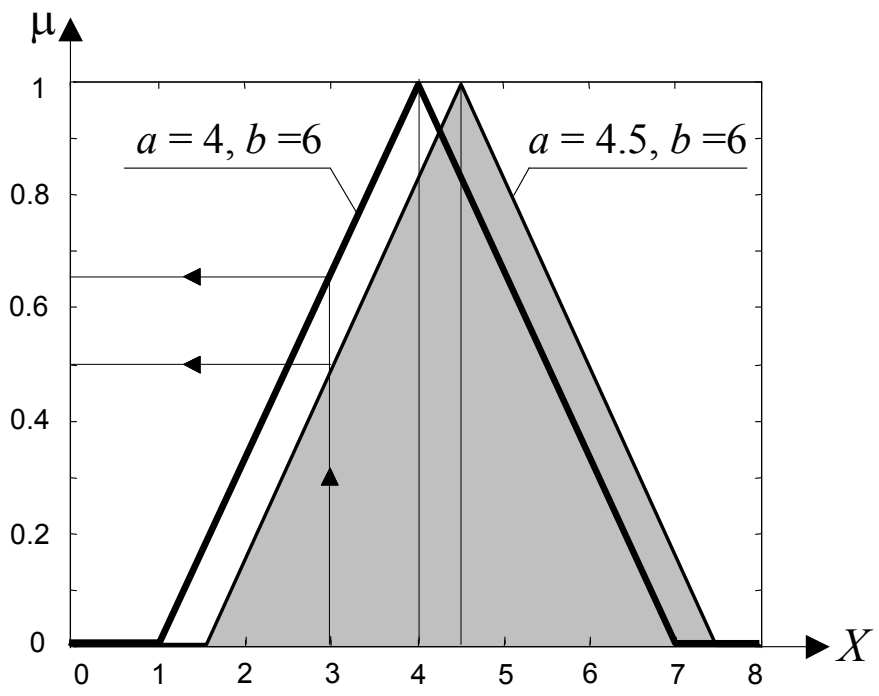
$$y_i^{(1)} = x_i^{(1)}$$

Layer 2 is the **fuzzification layer**. Neurons in this layer represent fuzzy sets used in the antecedents of fuzzy rules. A fuzzification neuron receives a crisp input and determines the degree to which this input belongs to the neuron's fuzzy set.

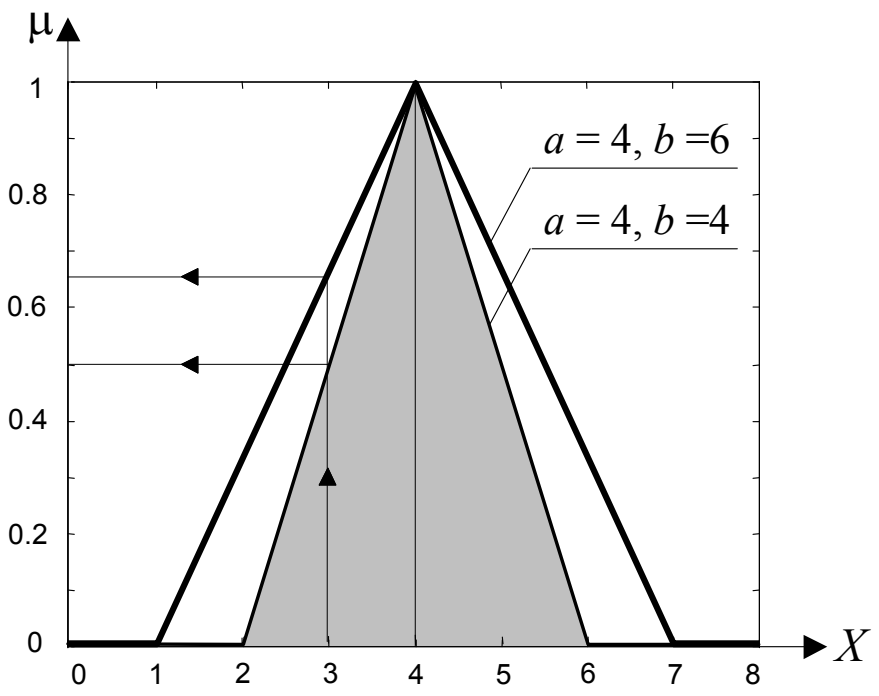
The activation function of a membership neuron is set to the function that specifies the neuron's fuzzy set. We use triangular sets, and therefore, the activation functions for the neurons in *Layer* 2 are set to the **triangular membership functions**. A triangular membership function can be specified by two parameters $\{a, b\}$ as follows:

$$y_i^{(2)} = \begin{cases} 0, & \text{if } x_i^{(2)} \le a - \dfrac{b}{2} \\ 1 - \dfrac{2\left|x_i^{(2)} - a\right|}{b}, & \text{if } a - \dfrac{b}{2} < x_i^{(2)} < a + \dfrac{b}{2} \\ 0, & \text{if } x_i^{(2)} \ge a + \dfrac{b}{2} \end{cases}$$

Triangular activation functions



(*a*) Effect of parameter *a*.



(*b*) Effect of parameter *b*.

Layer 3 is the **fuzzy rule layer**. Each neuron in this layer corresponds to a single fuzzy rule. A fuzzy rule neuron receives inputs from the fuzzification neurons that represent fuzzy sets in the rule antecedents. For instance, neuron $R1$, which corresponds to *Rule* 1, receives inputs from neurons $A1$ and $B1$.

In a neuro-fuzzy system, intersection can be implemented by the **product operator**. Thus, the output of neuron i in *Layer* 3 is obtained as:

$$y_i^{(3)} = x_{1i}^{(3)} \times x_{2i}^{(3)} \times \ldots \times x_{ki}^{(3)}$$

$$y_{R1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_{R1}$$

Layer 4 is the **output membership layer**. Neurons in this layer represent fuzzy sets used in the consequent of fuzzy rules.

An output membership neuron combines all its inputs by using the fuzzy operation **union**. This operation can be implemented by the **probabilistic OR**. That is,

$$y_i^{(4)} = x_{1i}^{(4)} \oplus x_{2i}^{(4)} \oplus \ldots \oplus x_{li}^{(4)}$$

$$y_{C1}^{(4)} = \mu_{R3} \oplus \mu_{R6} = \mu_{C1}$$

The value of μ_{C1} represents the integrated firing strength of fuzzy rule neurons *R*3 and *R*6.

Layer 5 is the **defuzzification layer**. Each neuron in this layer represents a single output of the neuro-fuzzy system. It takes the output fuzzy sets clipped by the respective integrated firing strengths and combines them into a single fuzzy set.

Neuro-fuzzy systems can apply standard defuzzification methods, including the centroid technique.

We will use the **sum-product composition** method.

The sum-product composition calculates the crisp output as the weighted average of the centroids of all output membership functions. For example, the weighted average of the centroids of the clipped fuzzy sets $C1$ and $C2$ is calculated as,

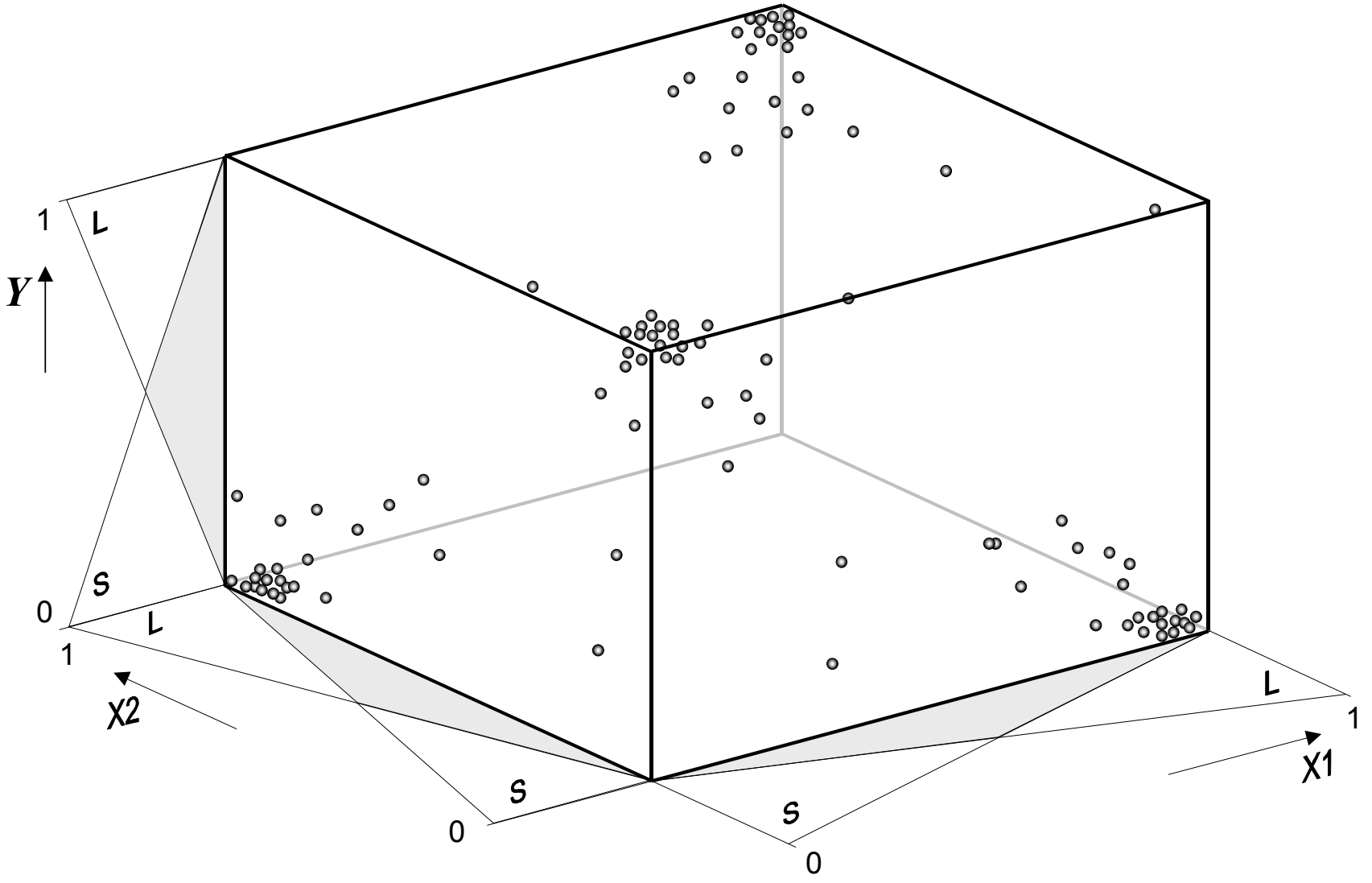
$$y = \frac{\mu_{C1} \times a_{C1} \times b_{C1} + \mu_{C2} \times a_{C2} \times b_{C2}}{\mu_{C1} \times b_{C1} + \mu_{C2} \times b_{C2}}$$

How does a neuro-fuzzy system learn?

A neuro-fuzzy system is essentially a multi-layer neural network, and thus it can apply standard learning algorithms developed for neural networks, including the back-propagation algorithm.

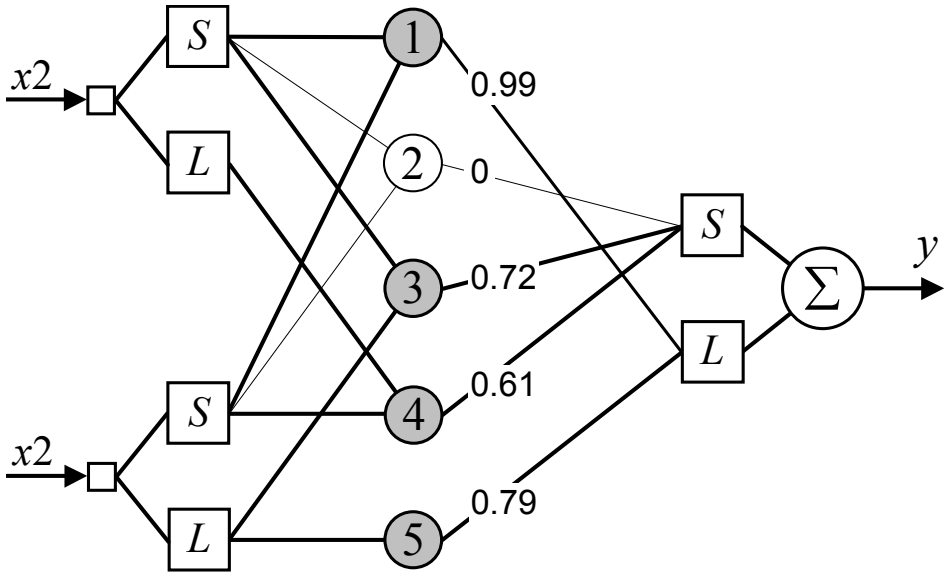
- When a training input-output example is presented to the system, the back-propagation algorithm computes the system output and compares it with the desired output of the training example. The error is propagated backwards through the network from the output layer to the input layer. The neuron activation functions are modified as the error is propagated. To determine the necessary modifications, the back-propagation algorithm differentiates the activation functions of the neurons.
- Let us demonstrate how a neuro-fuzzy system works on a simple example.

Training patterns

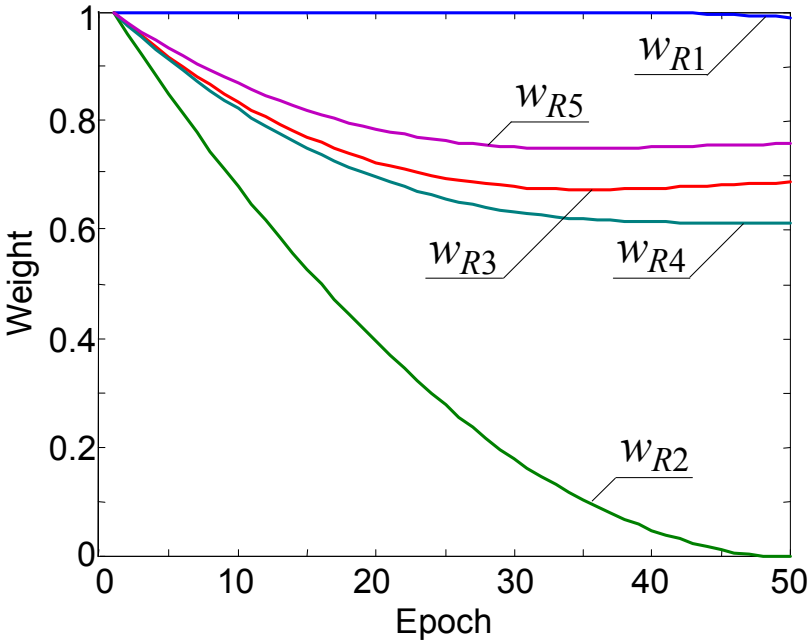


The data set is used for training the five-rule neuro-fuzzy system shown below.

Five-rule neuro-fuzzy system



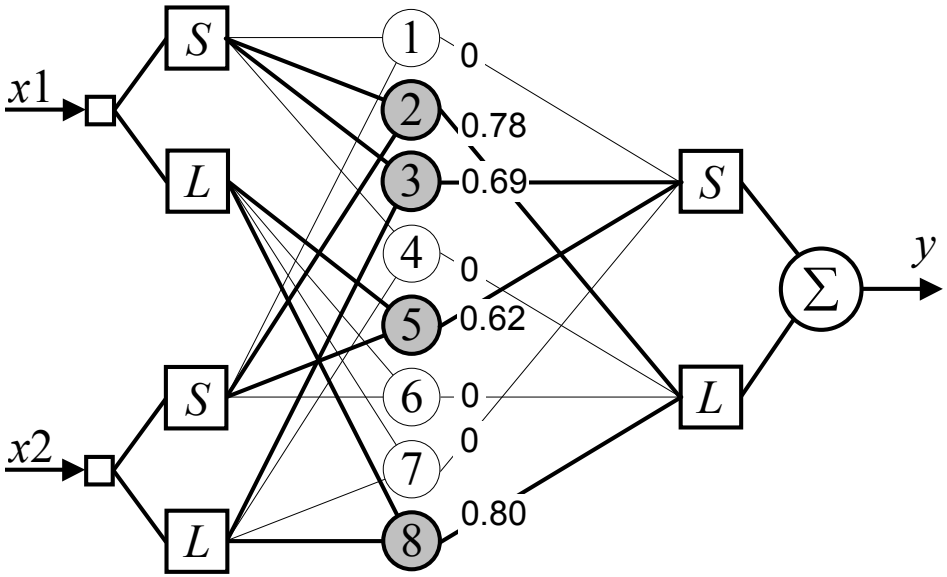
(*a*) Five-rule system.



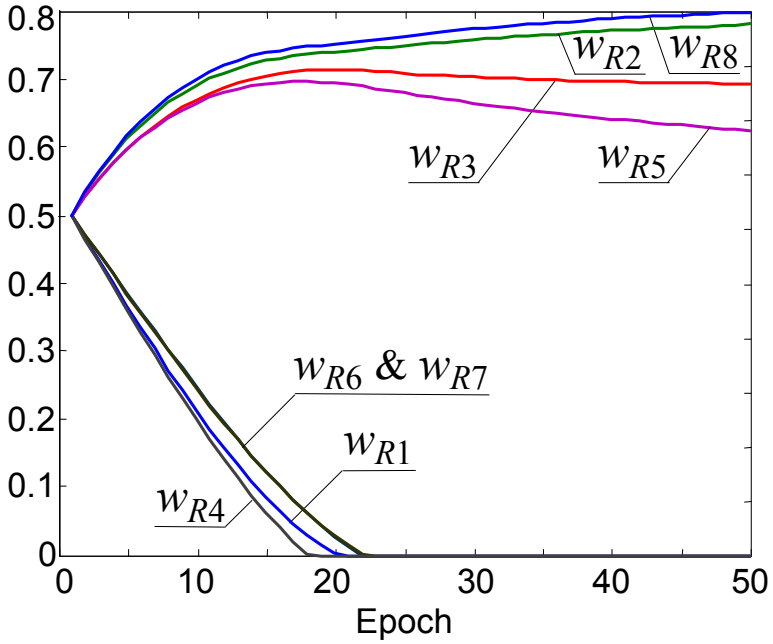
(*b*) Training for 50 epochs.

- Given input and output linguistic values, a neuro-fuzzy system can automatically generate a complete set of fuzzy IF-THEN rules.
- Let us create the system for the XOR example. This system consists of $2^2 \times 2 = 8$ rules. Because expert knowledge is not embodied in the system this time, we set all initial weights between *Layer* 3 and *Layer* 4 to 0.5.
- After training we can eliminate all rules whose certainty factors are less than some sufficiently small number, say 0.1. As a result, we obtain the set of four fuzzy IF-THEN rules that represents the inverse of the XOR operation.

Eight-rule neuro-fuzzy system



(*a*) Eight-rule system.



(*b*) Training for 50 epochs.

Neuro-fuzzy systems: summary

- The combination of fuzzy logic and neural networks constitutes a powerful means for designing intelligent systems.
- Domain knowledge can be put into a neuro-fuzzy system by human experts in the form of linguistic variables and fuzzy rules.
- When a representative set of examples is available, a neuro-fuzzy system can automatically transform it into a robust set of fuzzy IF-THEN rules, and thereby reduce our dependence on expert knowledge when building intelligent systems.

Landing of an aircraft as a time-series prediction problem

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

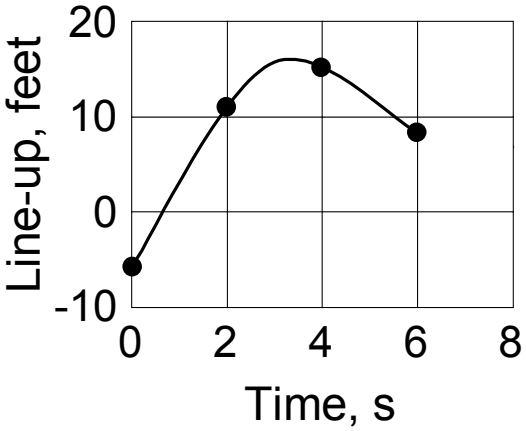
- The landing of an aircraft, particularly aboard aircraft carriers, is an extremely complex process.
- It is affected by such variables as the flight deck's space constraints and its motions (both pitch and roll), the aircraft's ordinance and fuel load, continuous mechanical preparations, and the most 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.

- 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.

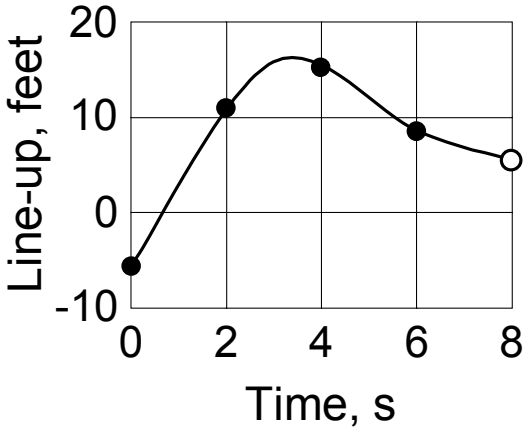
On-line time-series prediction of an aircraft's trajectory

Input: current motion profile of the aircraft



Time-series prediction system

Output: predicted position of the aircraft



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:

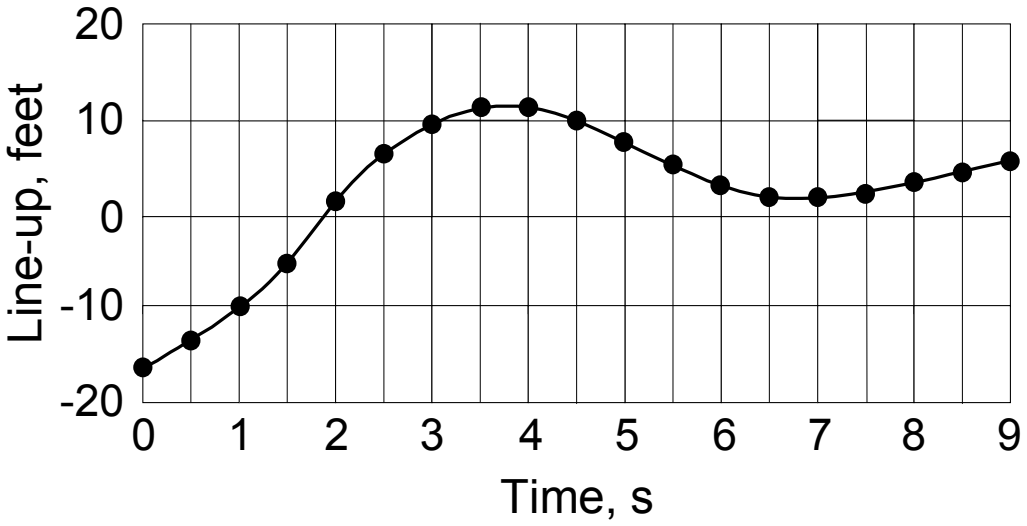
$$\mathbf{x} = \left[x(t-6) \;\; x(t-4) \;\; x(t-2) \;\; x(t) \right]$$

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



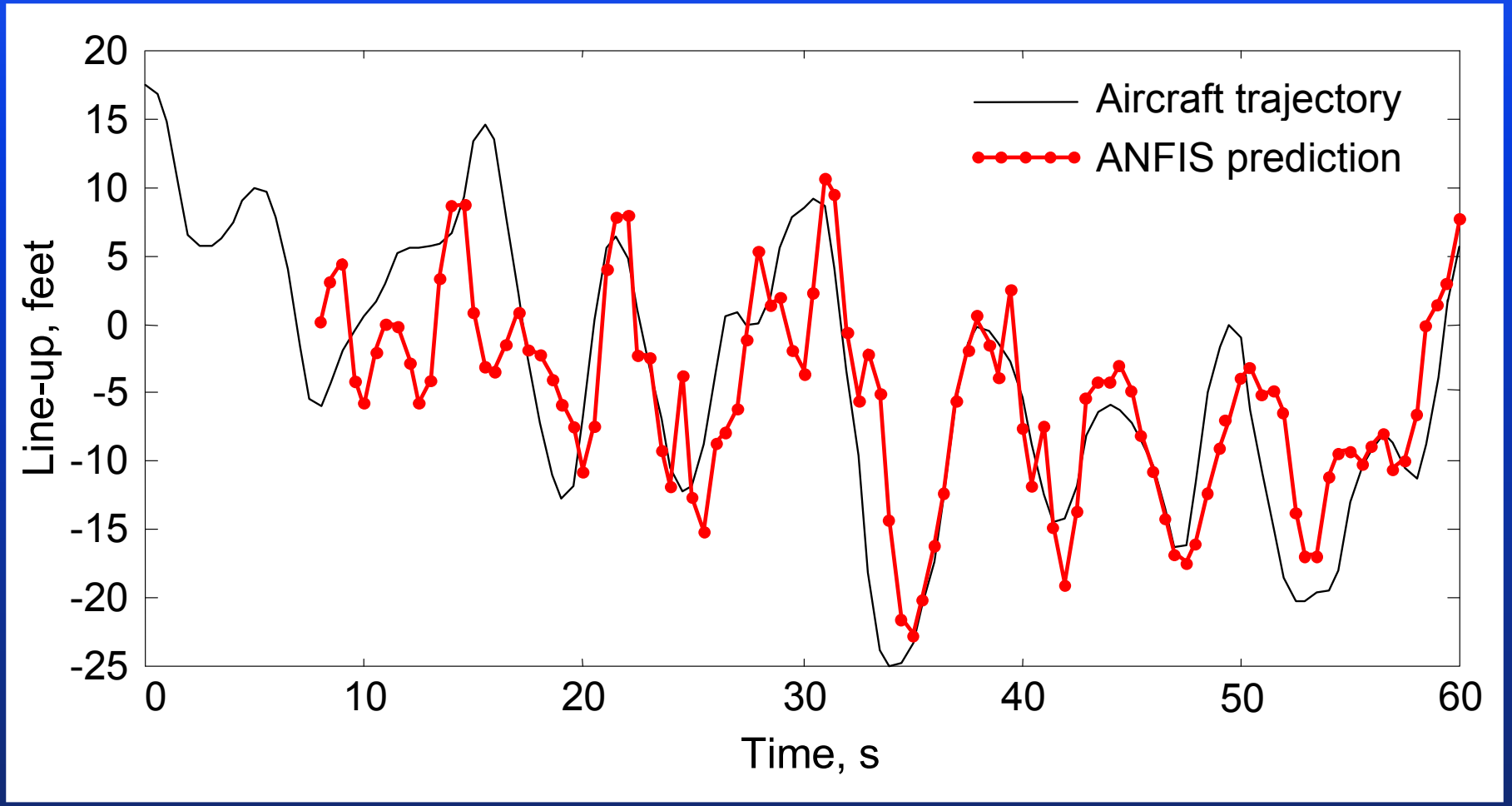
Inputs				Desired output
x_1	x_2	x_3	x_4	
–17.4	2.1	11.0	3.9	4.2
–12.9	7.5	10.1	2.1	4.9
–10.0	9.8	8.2	2.0	5.3

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.

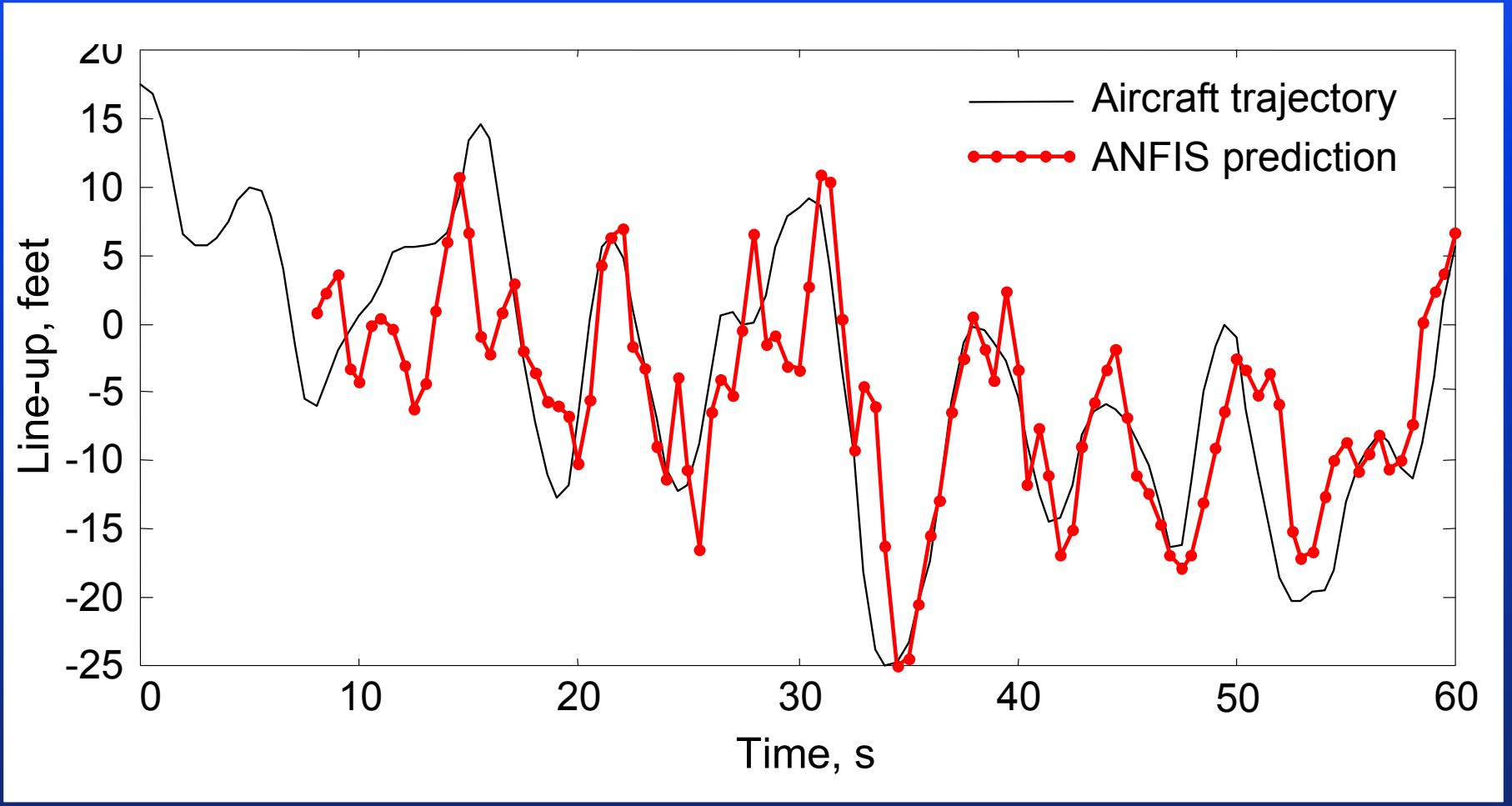
How many membership functions should we assign to each input variable?

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



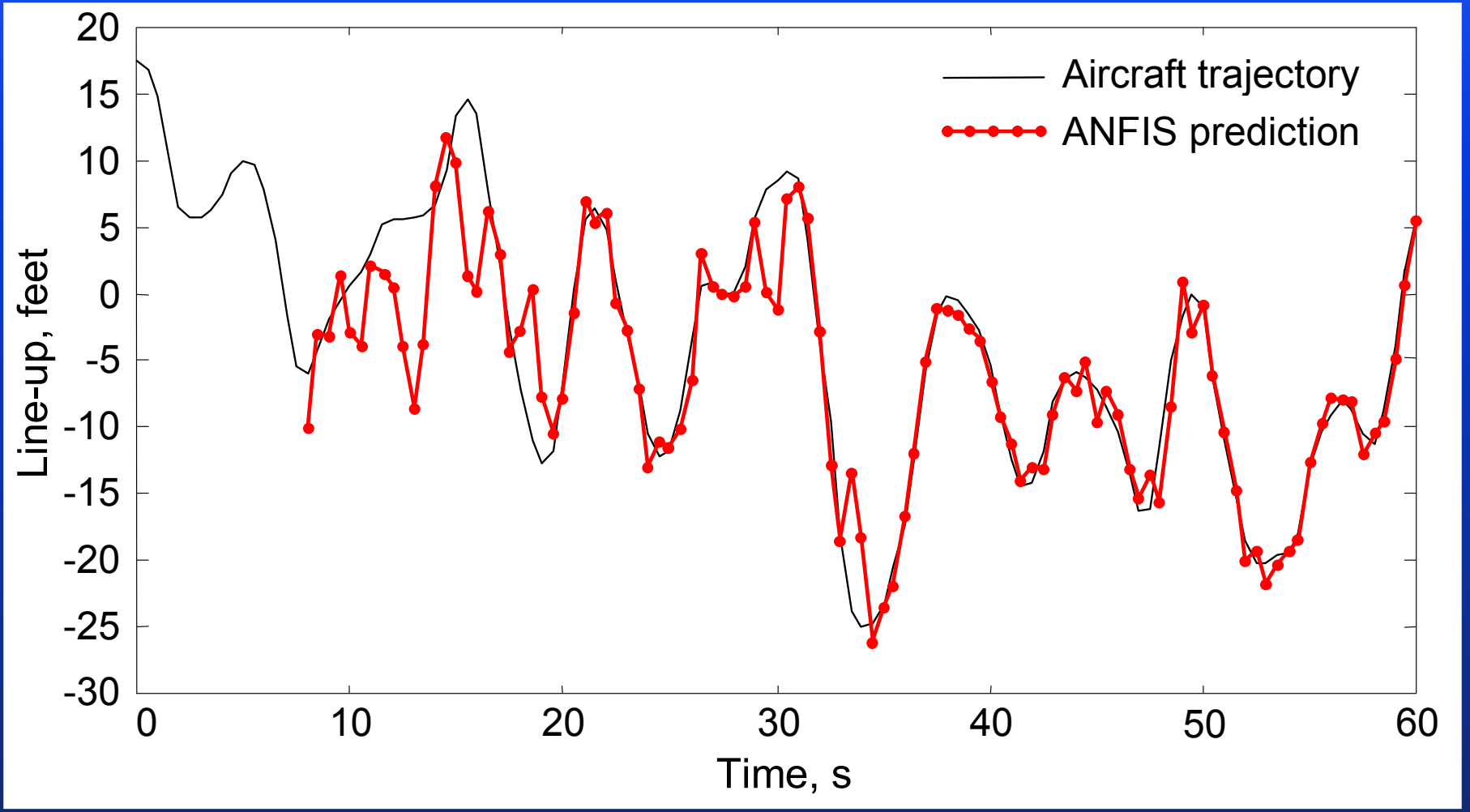
Performance of the ANFIS with four inputs and two membership functions assigned to each input: 100 epochs



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

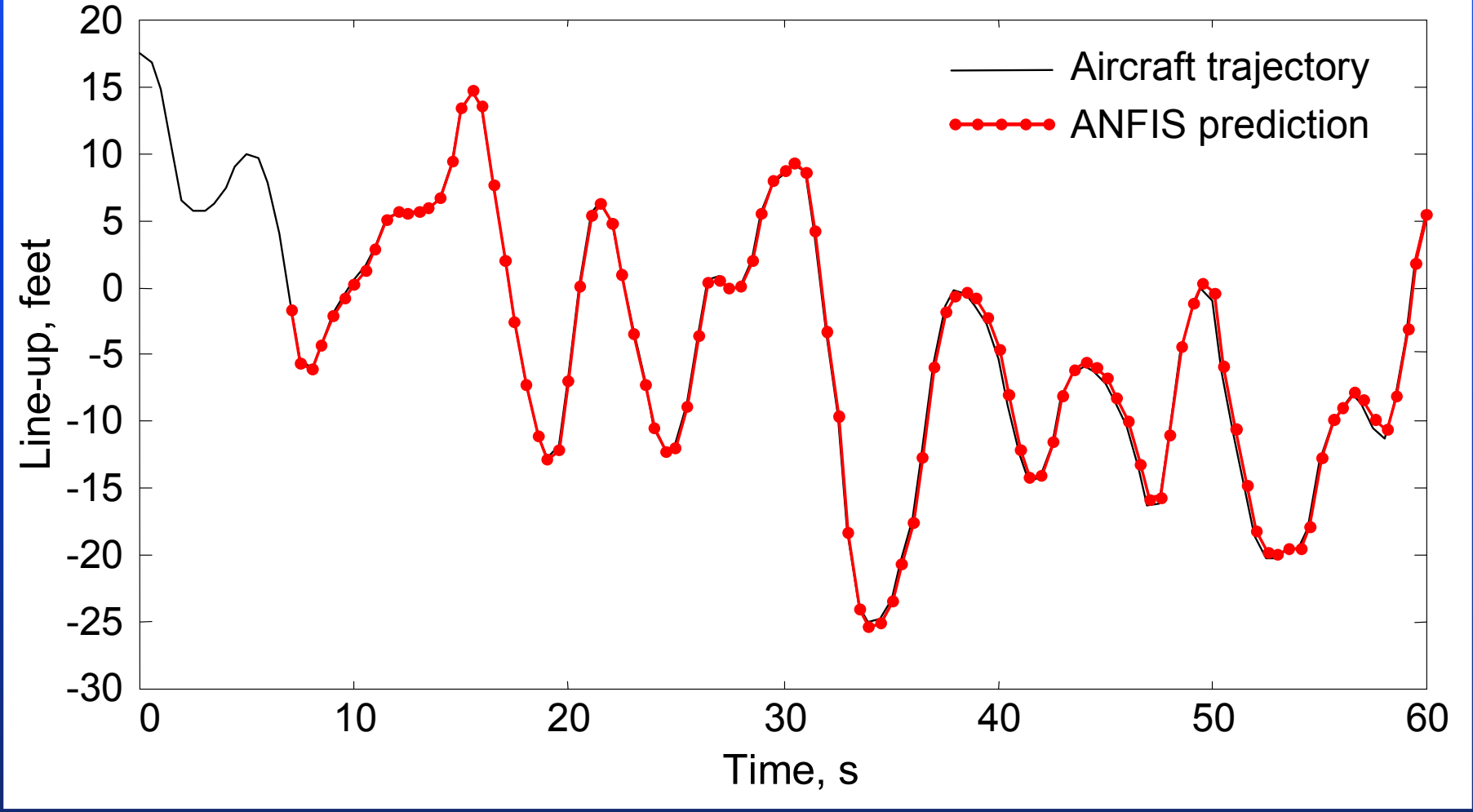


Another way of improving time-series prediction is to ***increase the number of input variables***.

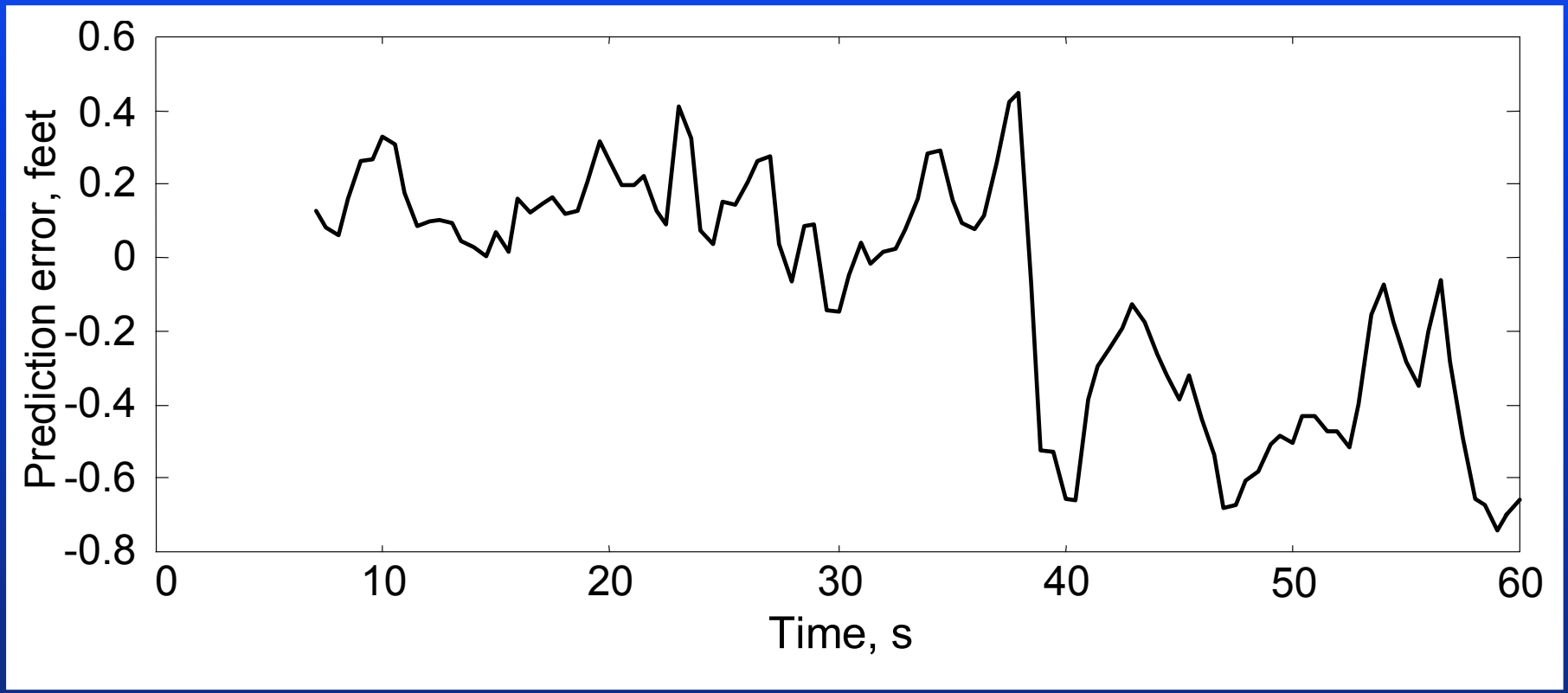
Let us, for example, examine an ANFIS with six inputs that correspond to the aircraft's flight positions at $(t-5)$, $(t-4)$, $(t-3)$, $(t-2)$, $(t-1)$, and t, respectively. The ANFIS output still remains the two-second prediction.

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

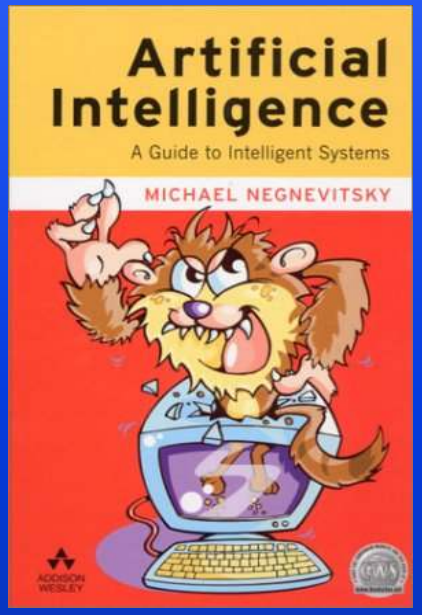


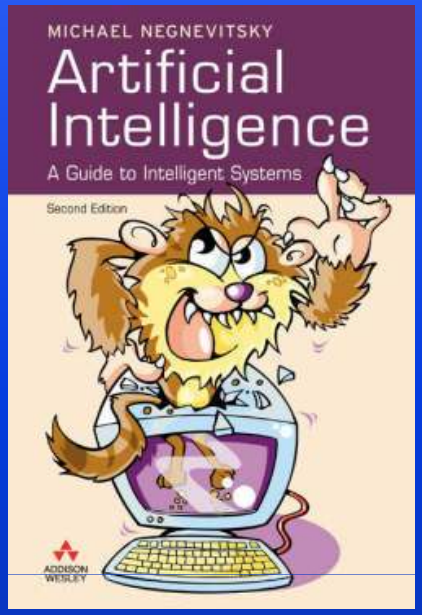
Conclusion

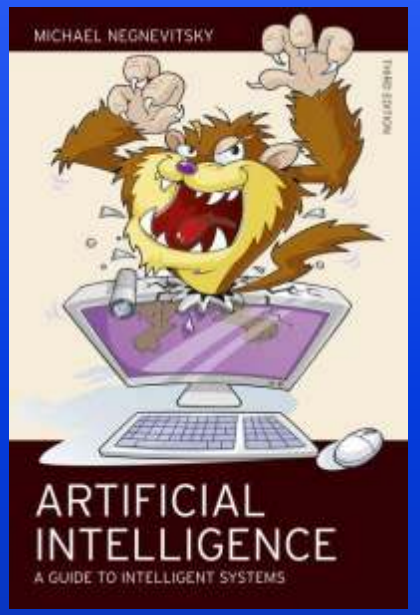
- This paper presented two different structures of hybrid neuro-fuzzy systems - heterogeneous and homogeneous.
- In a *heterogeneous* structure, the neural network and fuzzy system work as independent components (although they cooperate in solving the problem).
- An example was demonstrated through the development of a diagnostic system for myocardial perfusion from cardiac images. 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.

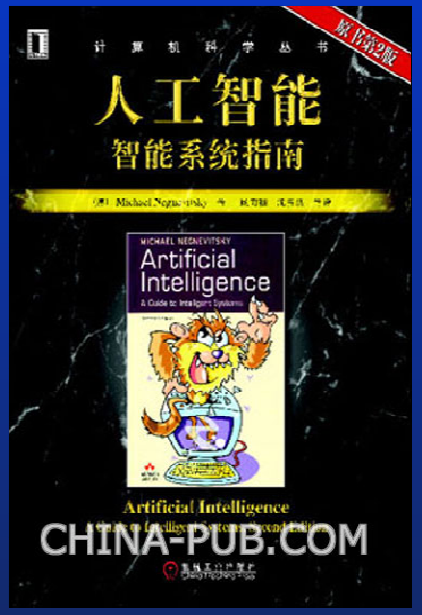
- Unlike a heterogeneous system, a *homogeneous* one cannot be divided into two independent and distinct parts. It is represented by a multi-layer neural network that performs fuzzy inferencing.
- The paper presented a basic structure of the heterogeneous neuro-fuzzy system, and demonstrated its application on the problem of predicting an aircraft's trajectory during its landing aboard an aircraft carrier.

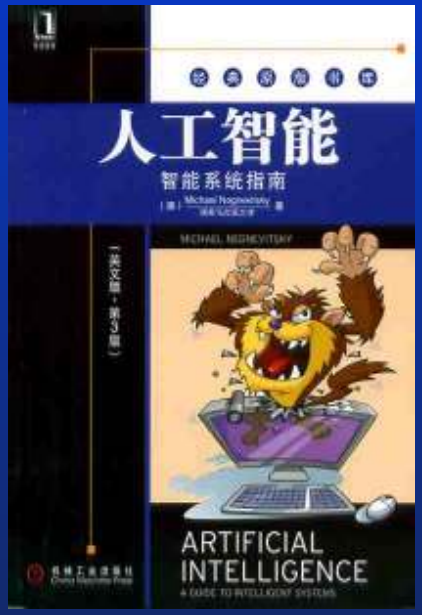
http:/www.booksites.net/negnevitsky

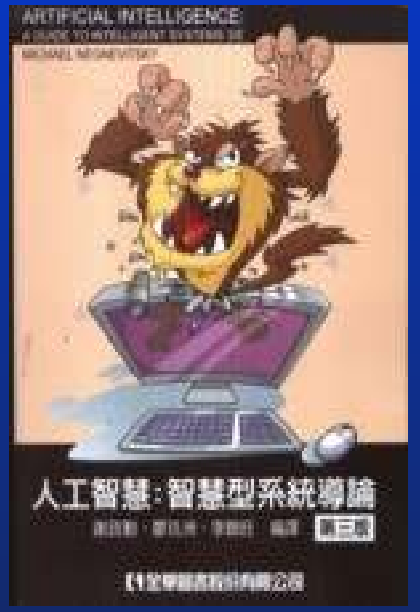












Questions ?

… if you dare …