Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

Modernizing Agricultural Practice using Internet of Things

MAPIoT Summer School in Norway

24.07.2022 – 07.08.2022

Melsom High School, Sandefjord

organized by University of South-Eastern Norway

AI (neural networks), GA (genetic algorithms) & Fuzzy Rules

– applied in Modelling, Control, Predicting and Managing of processes from Agriculture and Food Engineering domains –

Professor Adrian FLOREA, PhD

Lucian Blaga University of Sibiu

adrian.florea@ulbsibiu.ro

HPI Knowledge Transfer Institute at ULBS

http://centers.ulbsibiu.ro/itchpiulbs/en/

MAPIoT Motivation

Societal Challenges

- **Food waste** (1/3 of produced food is lost/wasted every year)
- **Population growth, industrialization and transition to cities => limitation of resources (water, energy), weather uncertainty and climate change**
- **Inefficiency** in planting, harvesting, feeding, monitoring, water use
- In the agricultural sector, there are **significant disparities between EU countries**

Technical Challenges

- **Agriculture** is now **moving into the information age**, where data are collected and analysed to improve both production and quality
- **Technological gap, lack of digitalization** (digital technologies and Internet connectivity)
- **Need for Knowledge Transfer of IT in Agriculture and Food processing / security** domains

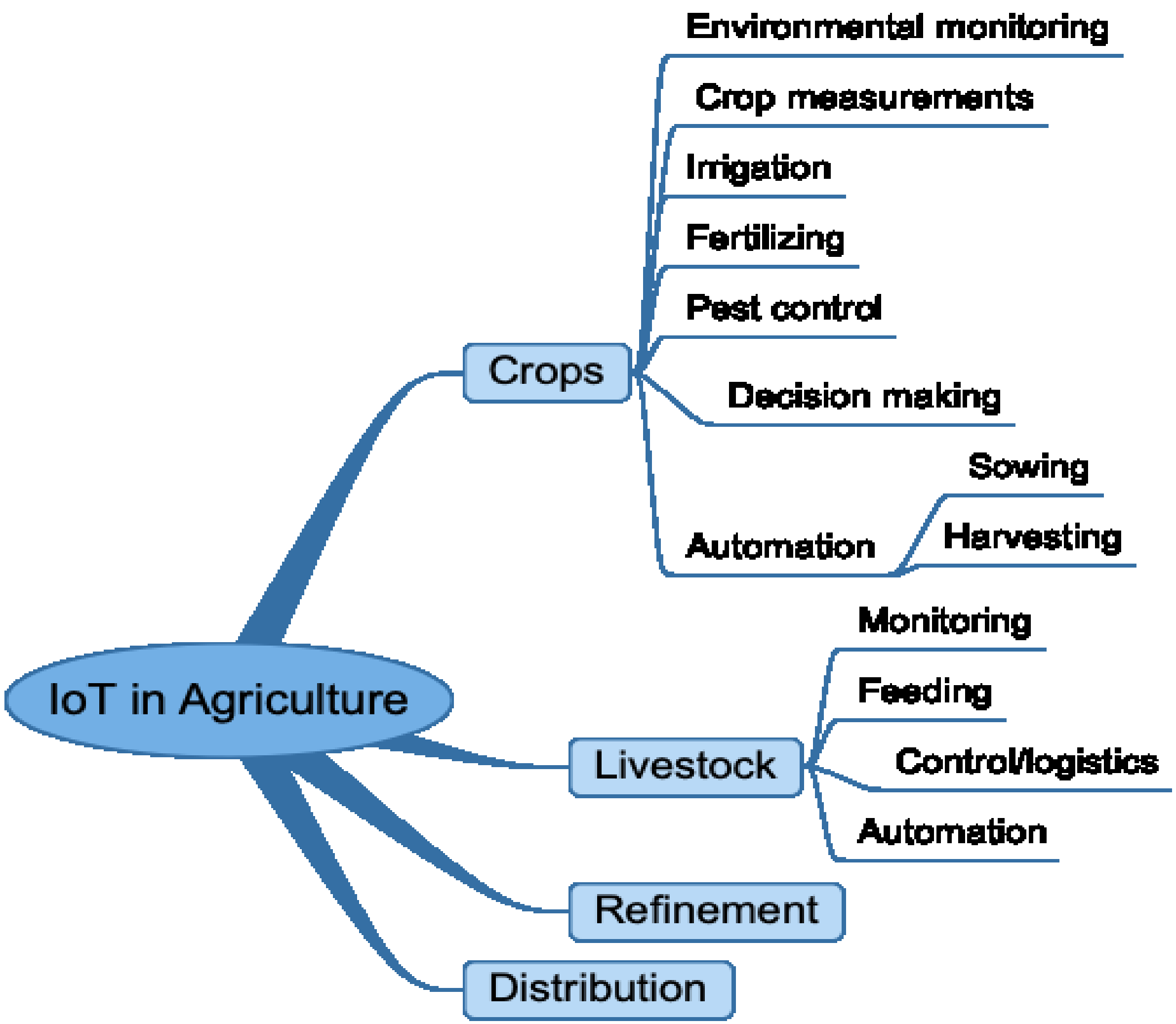
MAPIoT Motivation

Proposed Solutions

- April 2019, EU member states signed the declaration of cooperation on "**A smart and sustainable digital future for European agriculture and rural areas**"
- Investment in **education**, scientific **research, innovation,** and **infrastructure**
- Waste recycling, more responsible consumption
- Internet of Things (**IoT**) plays an **essential role in innovative developments**
- Introducing in Agriculture / Food Engineering domains **automation and robotics**, **smart drones**, IoT systems, or new methods for processes monitoring, control or prediction using algorithms of computer vision, **Artificial Intelligence** (AI), **Genetic Algorithms** (GA) and knowledge-based systems (**Fuzzy Rules**) as sustainable software applications

Iceland Liechtenstein Norway grants
UNIVERSITATEA LUCIAN BLAGA DIN SIBIU
USN University of South-Eastern Norway
MAPIoT

IoT in agriculture: examples



Applying AI in Modelling, Control, Predicting and Managing of processes from Agriculture and Food Engineering domains

Solutions

1. Using **neural networks (NN)**:
   - to obtain indirect information about the state variables in an **alcoholic fermentation process**
   - to **detect fruit / plant diseases & nutrient deficiency** in very initial stage based on image processing and pattern matching
   - to **identify ripe vegetables and fruits** by analysing shape and colour --> **crop growth and harvesting optimization**
   - **forecasting of production in agriculture** on the basis of a wide range of independent variables, caused by unexpected crisis (war, climate changes), lack of water, population growth
   - if we can **predict the probability of a specific product for food recall**, this will help food producers to improve food safety & reduce food waste

Iceland Liechtenstein Norway grants

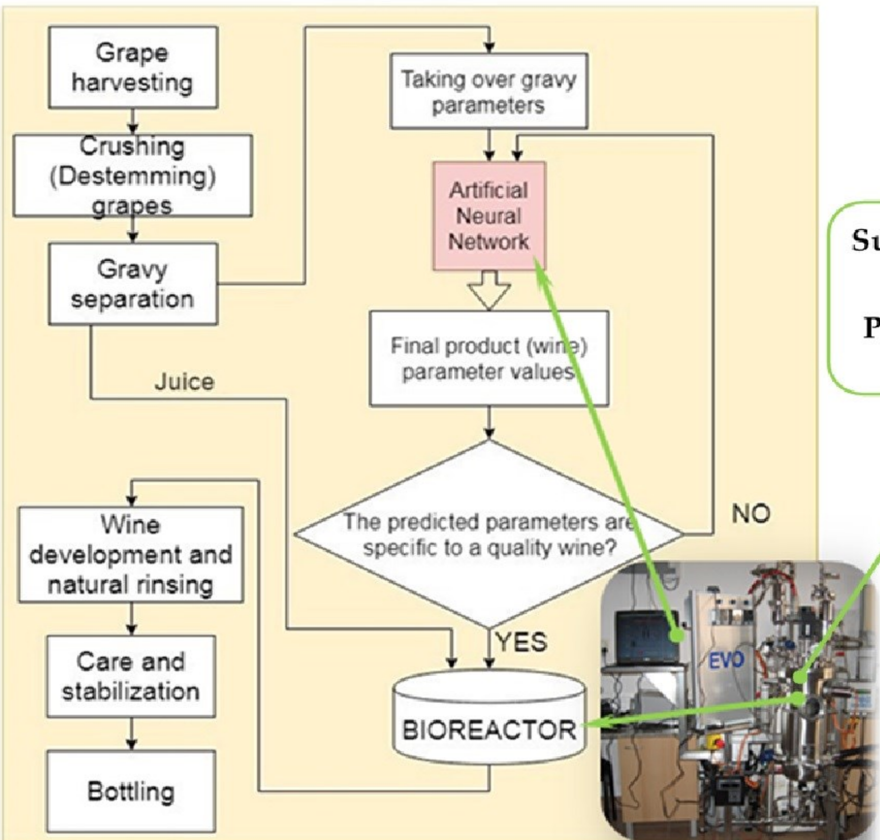
UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

MAPIoT

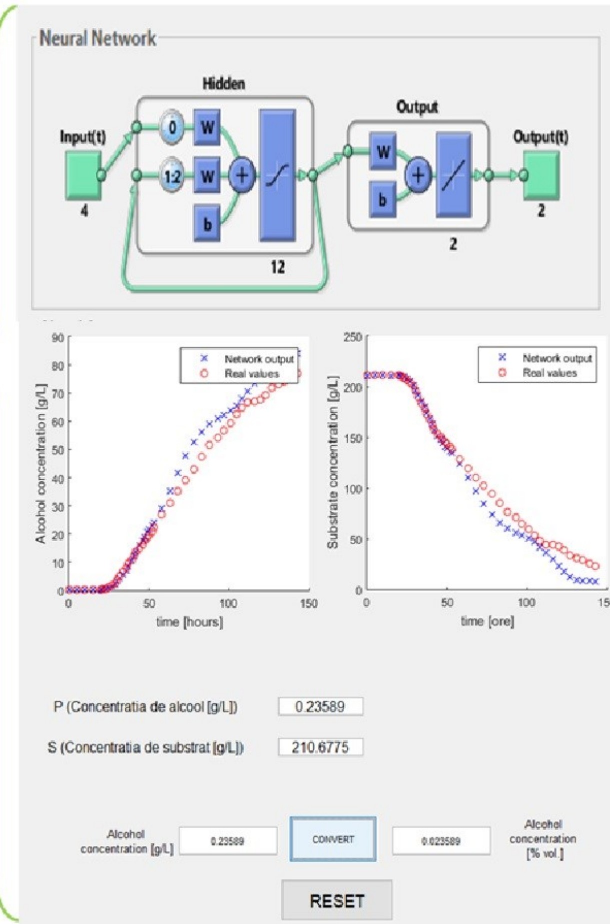
Using neural networks to obtain indirect information about the state variables in an alcoholic fermentation process

Anca Sipos, Adrian Florea, Maria Arsin, Ugo Fiore



The process of making wine using a bioreactor and an artificial neural network

Substrate equipment analyser
Product equipment analyser



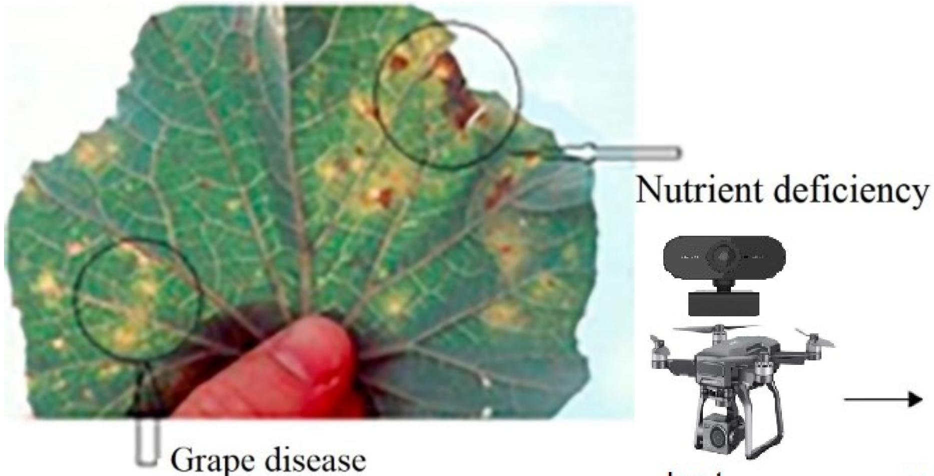
Iceland Liechtenstein Norway grants

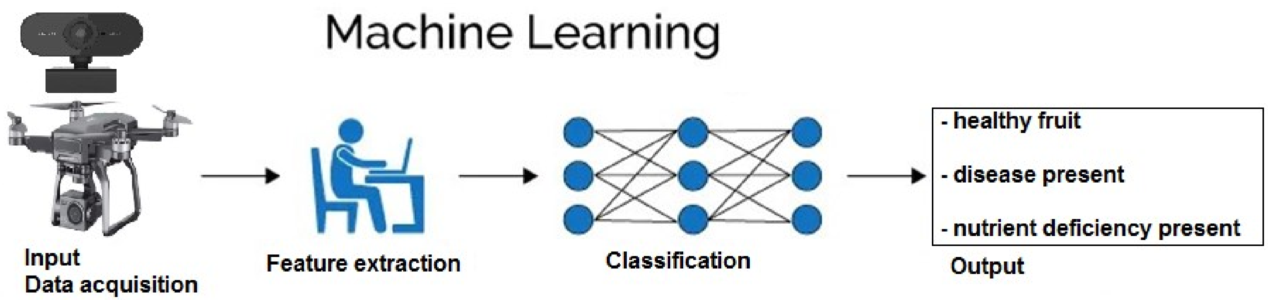
UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

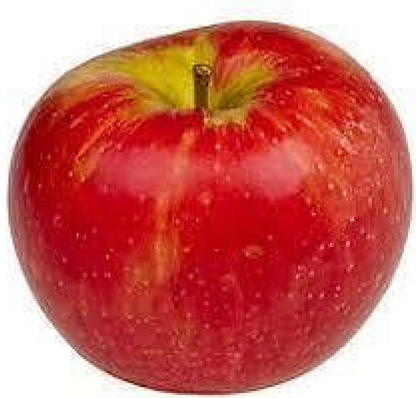
MAPIoT

Using NN to detect fruit / plant diseases & nutrient deficiency in very initial stage based on image processing and pattern matching



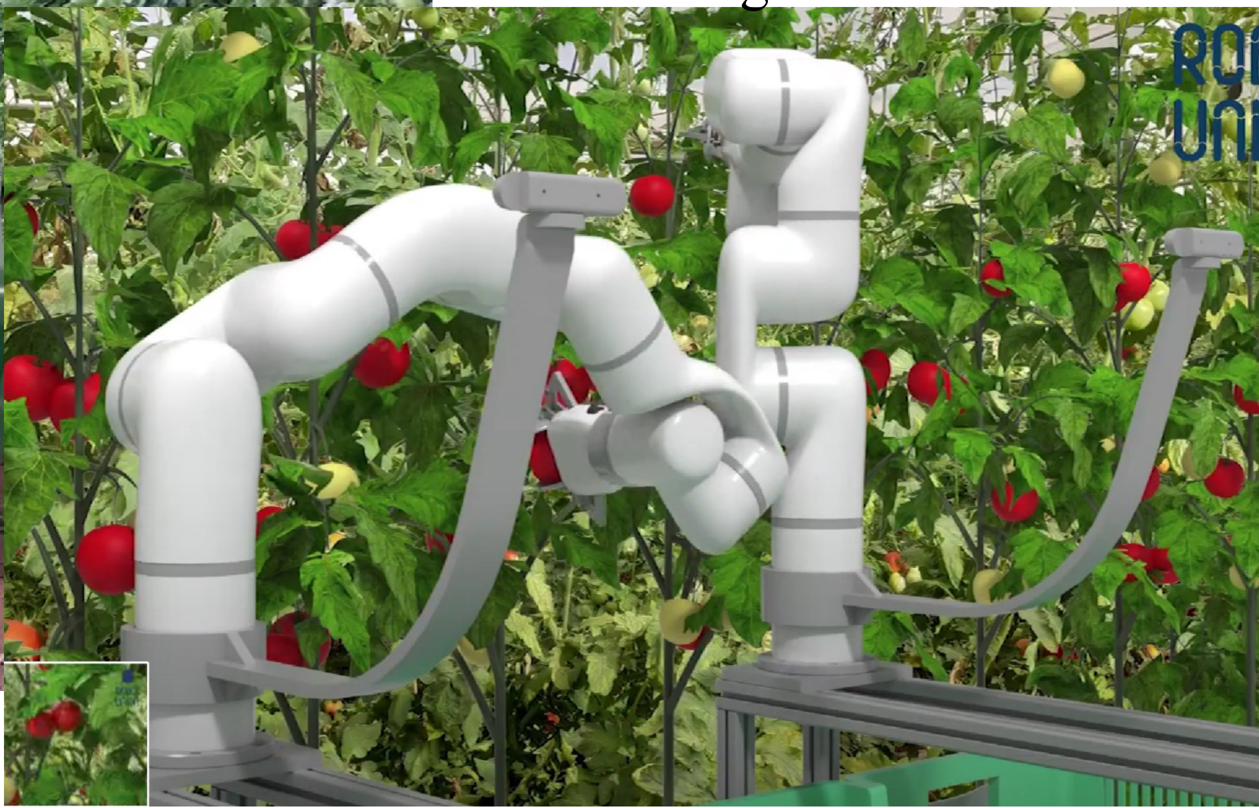


Normal (health) / Blotch / Scab / Rotten Apple



Robot able to harvest tomatoes

- Combine **Computer Vision** with **Neural Networks** for shape and colour prediction to identify the right moment of harvesting



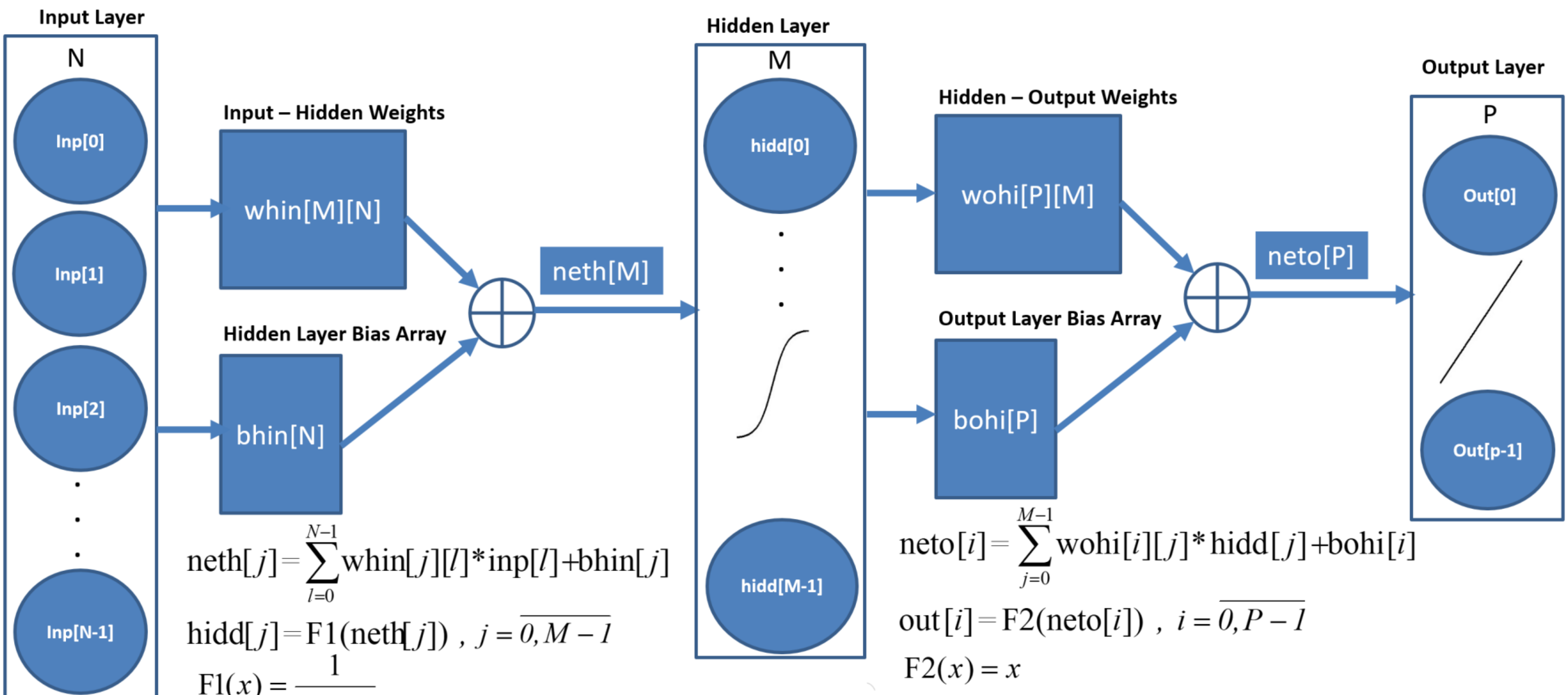
Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

MAPIoT

Forecasting of production in agriculture on the basis of a wide range of independent **variables**, caused by **unexpected crisis** (war, climate changes), **lack of water, population growth**: **customized Artificial Neural Network for Crop Yield Prediction**



$$\mathrm{neth}[j]=\sum_{l=0}^{N-1}\mathrm{whin}[j][l]*\mathrm{inp}[l]+\mathrm{bhin}[j]$$

$$\mathrm{hidd}[j]=\mathrm{F1}(\mathrm{neth}[j])\ ,\ j=\overline{0,M-1}$$

$$\mathrm{F1}(x)=\frac{1}{1+e^{-x}}$$

$$\mathrm{neto}[i]=\sum_{j=0}^{M-1}\mathrm{wohi}[i][j]*\mathrm{hidd}[j]+\mathrm{bohi}[i]$$

$$\mathrm{out}[i]=\mathrm{F2}(\mathrm{neto}[i])\ ,\ i=\overline{0,P-1}$$

$$\mathrm{F2}(x)=x$$

Customized Artificial Neural Network for Crop Yield Prediction

Table 1: Sample wheat data set[2]

Bio-mass	ESW	NO_3	Rain	Trans-piration	Soil-Evapo ration	Wheat yield
5253.3	105.3	28.3	33	140.148	65.963	2006.1
6905.3	113.5	25.5	106	171.259	88.736	2908.5
5911.9	94.2	23.8	39	145.226	75.363	2135.4
5163.9	113.2	30.9	34	139.884	58.763	1819.2
5739.7	98.5	25.8	8	132.416	56.592	2087.2
5822.7	93.5	24.1	35	148.363	71.335	2572.1
5163.3	98.4	27.3	32	133.102	63.143	1952.5

ESW: Extractable soil water

The **parameters** considered **for wheat yield prediction**:

- **Biomass** ($kg \cdot ha^{-1}$) is the accumulated energy in plants
- ESW (mm) is the **extractable soil water**
- NO_3 ($kg \cdot ha^{-1}$) is **nitrogen content present in soil**
- Rain (mm) is amount of **rainfall since sowing**
- **Transpiration** (mm) is the amount of **water evaporated from the leaf**
- **Soil evaporation** (mm) is the amount of water evaporated from soil
- **Historic wheat yield** ($kg \cdot ha^{-1}$) from previous period

Applying AI in Modelling, Control, Predicting and Managing of processes from Agriculture and Food Engineering domains

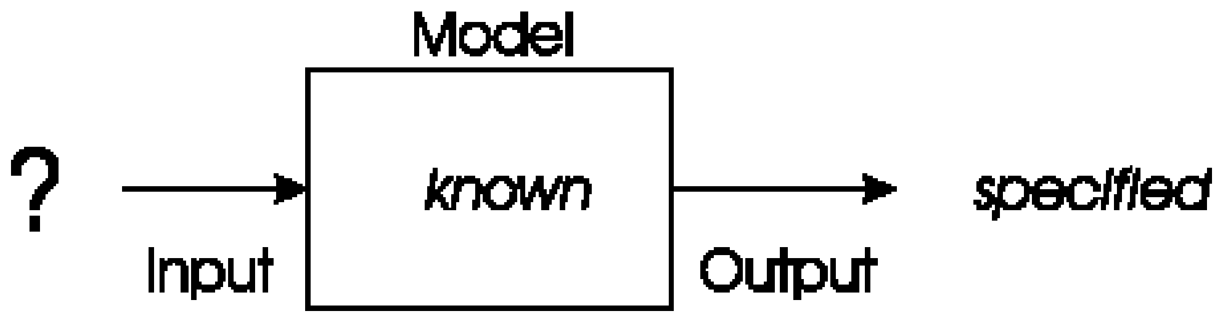
Solutions

2. Using **Genetic Algorithms (GA)**:

- ***Optimization*:**
  - **of planning the harvesting**
  - **Automatic design space exploration**
- ***Simulation*:**
  - **Identification of natural resource sets for maximizing regional diversity and maintaining long-term biodiversity**
  - **Simulation based optimization in computer architectures, maintenance processes**
- ***Modeling*:**
  - **Global climate modelling.**
  - **Train the Neural Networks**. Generating initial weights

Problem type 1: Optimization

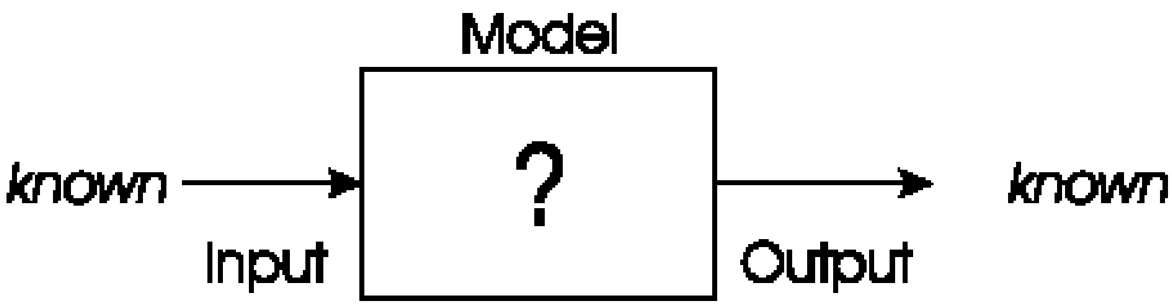
We have a model of our system and **seek inputs** that give us a specified goal!



- **Optimization of planning the harvesting** according to the **area and the number of harvesting fields**, the number of **harvesting machines** (combines), the number of **drivers**, the number of **warehouses** and the **quantity that fits in the warehouses to reduce both the working time and the fuel (multi-objective optimization problem)**
- *Optimization*: **Automatic design space exploration**

Problem types 2: Modelling

We have corresponding sets of inputs & outputs and **seek model** that delivers correct output for every known input!

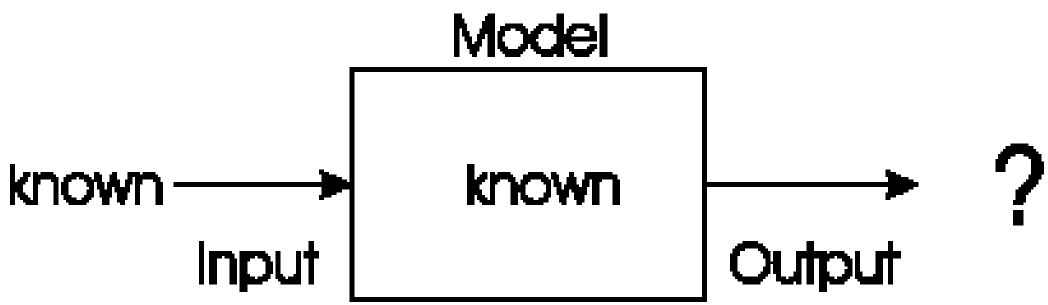


***Modeling*:**

- **Global climate modelling, results are more precise if not only the atmosphere and the oceans, but also the rainforests, deserts and cities are modelled.**
- **Train the Neural Networks. Generating initial weights**

Problem type 3: Simulation

We have a given model and **wish to know the outputs** that arise under different input conditions!



Simulation:

- **Identification of natural resource sets for maximizing regional diversity and maintaining long-term biodiversity**
- **Simulation based optimization in computer architectures, maintenance processes**

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

Applying AI in Modelling, Control, Predicting and Managing of processes from Agriculture and Food Engineering domains

Training the Neural Networks initial weights

http://193.226.29.27/WineFermentation/

Application for supervising and controlling

White Wine Fermentation Parameter Evolution

| Home | | Neural Network | | Genetic Algorithm | | Normalize Data | | Train and Test | | Results | | Predict Parameters |

This Web application presents an user-friendly interface to a software tool meant to predict the process variables within a biochemical process, namely the white fermentation process, by implementing the following main features:

- User-customized Multi-Layer Perceptron neural network NN
- Genetic Algorithm designed to enhance the neural network performance
- Raw data-processing to fit the NN input requirements
- Train NN and check the predicted outcomes
- Compare results with the expected values
- Graphical visualization and comparison for the acquired findings
- Parameters prediction by using the previously designed NN-based tool

Applying AI in Modelling, Control, Predicting and Managing of processes from Agriculture and Food Engineering domains

Solutions

3. Using **Fuzzy Rules (FR)** for **modeling of imprecise concepts**:
    - To automate and control the irrigation process
    - To detect the fermentation phase
    - To reduce the design space of parameters and improve the quality of solutions

    Fuzzy logic are based on fuzzy set theory developed by Lotfi Zadeh since 1965.

Irrigation system
FUZZY LOGIC SYSTEM: Variables

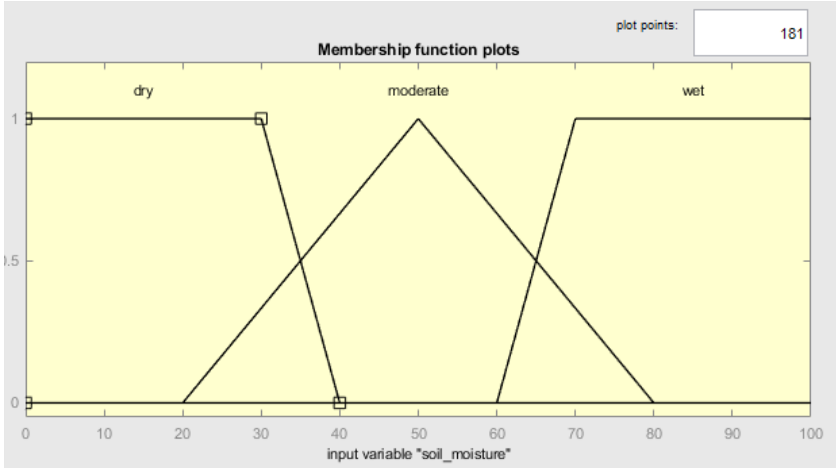
Four inputs / Each has defined 3 membership functions

Soil Moisture {DRY, MODERATE, WET}

Air Humidity {LOW, MODERATE, HIGH}

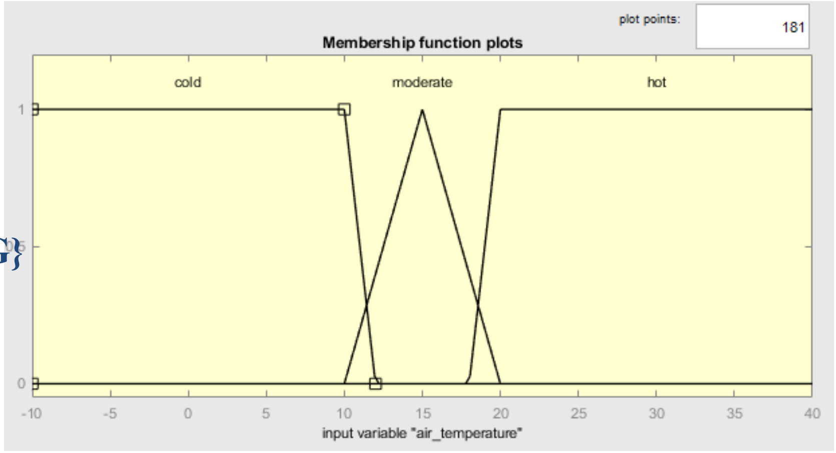
Light Intensity {DARK, MODERATE, BRIGHT}

- **are represented in percentage with values in [0, 100] interval**



Air Temperature {COLD, MODERATE, HOT }

- **is represented in Celsius degrees in [-30, 40] interval.**



One output:

Irrigation Time {NONE, SHORT, MEDIUM, LONG, VERY LONG}

- **The output is calculated with the centroid defuzzification method.**

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

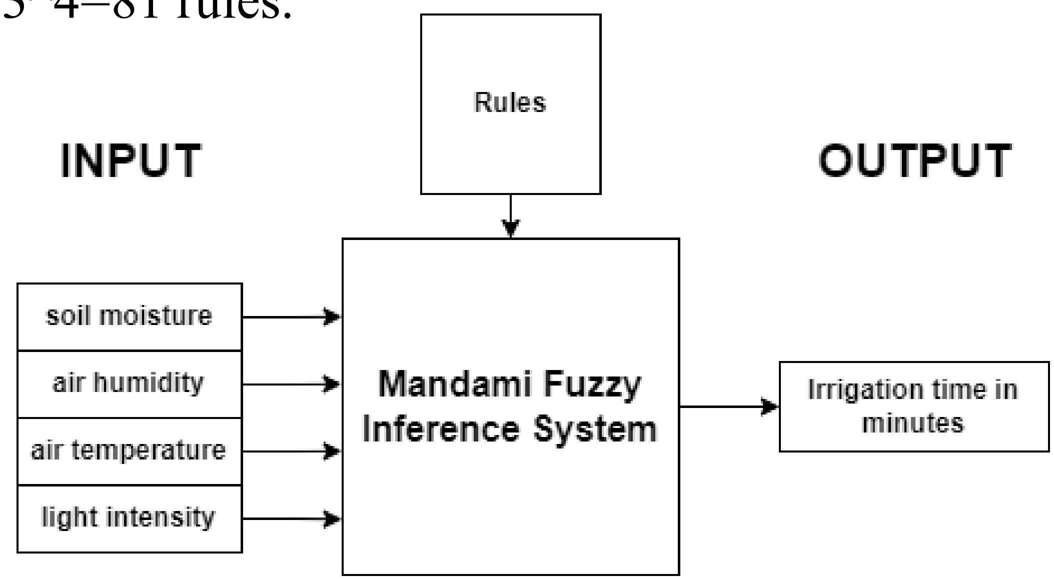
MAPIoT

Irrigation system SINK:
MANDAMI INFERENCE FUZZY LOGIC SYSTEM (1)

R1: *IF (**soil_moisture** IS **wet**) AND (**air_temperature** IS **cold**) AND (**air_humidity** IS **high**) AND (**light_intensity** IS **bright**) THEN (**irrigation_time** IS **none**)*

R2: *IF (**soil_moisture** IS **dry**) AND (**air_temperature** IS **moderate**) AND (**air_humidity** IS **low**) AND (**light_intensity** IS **moderate**) THEN (**irrigation_time** IS **very_long**)*

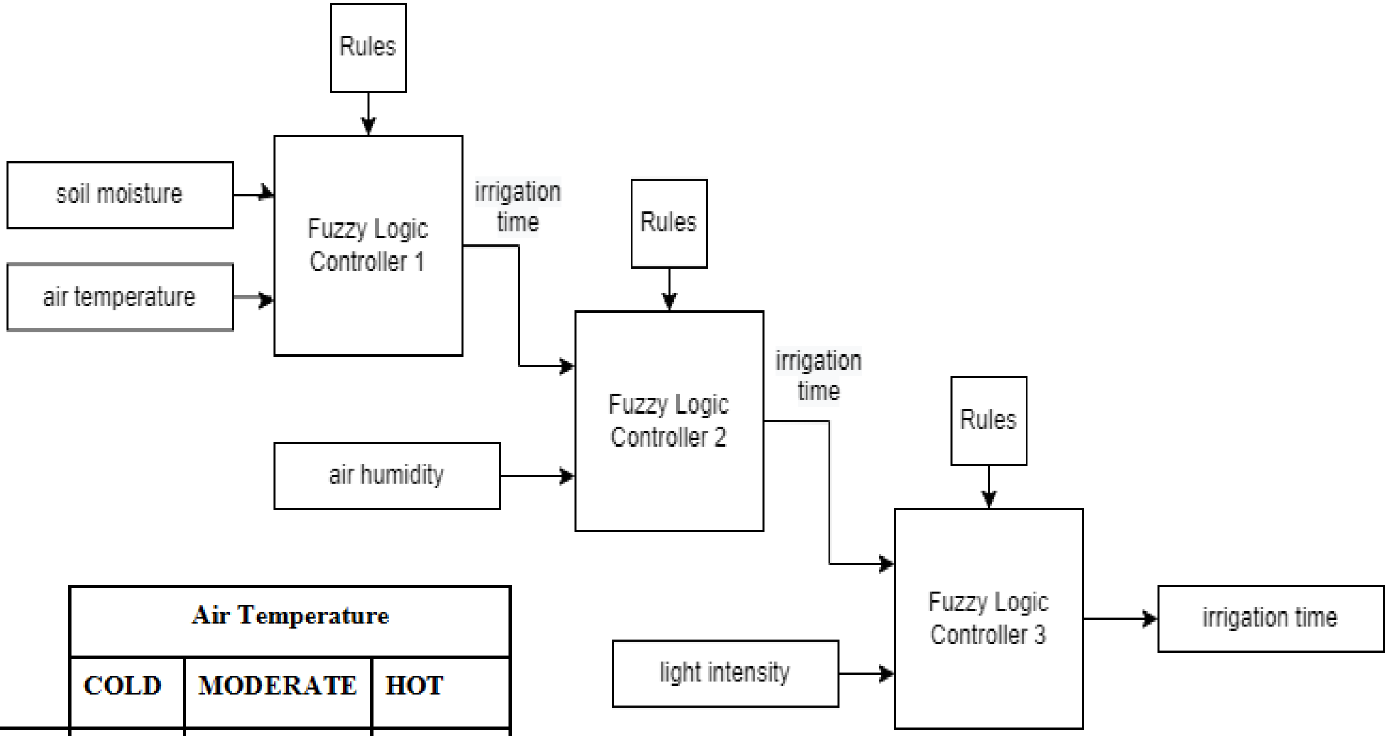
The rules were chosen on nursery managers' experience.
Disadvantage: We need to define 3^4=81 rules.



Irrigation system SINK: MANDAMI INFERENCE FUZZY LOGIC SYSTEM (2)

Solution:

Cascaded rules => the number of rules decreases, the performance increases and it is easier to define them



		Air Temperature		
		COLD	**MODERATE**	**HOT**
Soil Moisture	**DRY**	LONG	VERY LONG	VERY LONG
	MODERATE	SHORT	MEDIUM	MEDIUM
	WET	NONE	NONE	NONE

Rules of FLC 1

Neural Networks (NN)

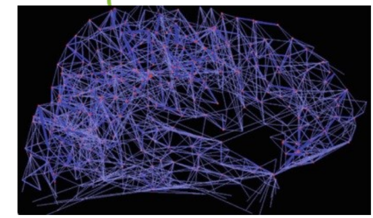
- Short history
- AI: domain affiliation
- What are NN? Advantages. Challenges
- Types of Neural Networks
- Structure of an artificial neuron
- Activation functions
- Simple perceptron and Multi-Layer Perceptron. Constraints
- Learning Mechanism: Backpropagation
- Source code examples: MATLAB and C#
  - Creating the network
  - Selection of activation function
  - Metrics
  - Training
- Application & Results

History of Neural Networks (I)

- The idea of "**a machine that thinks**" can be **traced to the Ancient Greeks.**
- Next, focus on the key events that led to the **evolution of thinking around neural networks**:
  - **1943: Warren S. McCulloch** and **Walter Pitts** published "A logical calculus of the ideas immanent in nervous activity". This research sought to **understand how the human brain could produce complex patterns through connected brain cells**, or neurons. One of the main resulting ideas was the **comparison of neurons with a binary threshold to Boolean logic** (0/1 or true/false statements).
  - **1958: Frank Rosenblatt** is credited with the development of the perceptron - "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". He takes McCulloch and Pitt's work a step further by **introducing weights to the equation**. Leveraging an IBM 704, Rosenblatt was able to **get a computer to learn how to distinguish cards marked on the left vs. cards marked on the right**.

History of Neural Networks (II)

- **1969**: **Marvin Minsky** and **Seymour Papert** have analysed the **learning possibilities of the perceptron** and reached rather sceptical conclusions, proving the **impossibility for the single-layer perceptron to solve simple problems** such as learning the XOR function (a function that is **not linearly separable**).
- **1974**: While numerous researchers contributed to the idea of **backpropagation**, **Paul Werbos** was the first person in the US to note **its application within neural networks** within his PhD thesis.
- **1989**: **Yann LeCun** published a paper illustrating how the use of **constraints in backpropagation and its integration into the neural network architecture can be used to train algorithms**. This research successfully leveraged a neural network to recognize hand-written zip code digits provided by the U.S. Postal Service.



Neural Networks: What are?

- Neural networks try to simulate the neurophysiological structure of the human brain.
- Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.
    - The cortex is composed of a large number of interconnected biological cells called neurons. **Each neuron receives signals from the neurons connected to it through the dendrites and conveys a signal using the axon**.
- NNs allow computer programs to recognize patterns and solve common problems.
- Also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a **subset of machine learning** and are at the heart of deep learning algorithms.

Types of Neural Networks

There are several types of artificial NNs classified according to different factors:

1. From a **purpose** point of view, NNs can be viewed as part of the larger domain of **pattern recognition and Artificial Intelligence** by the **necessity of learning (supervised vs. not-supervised)**.

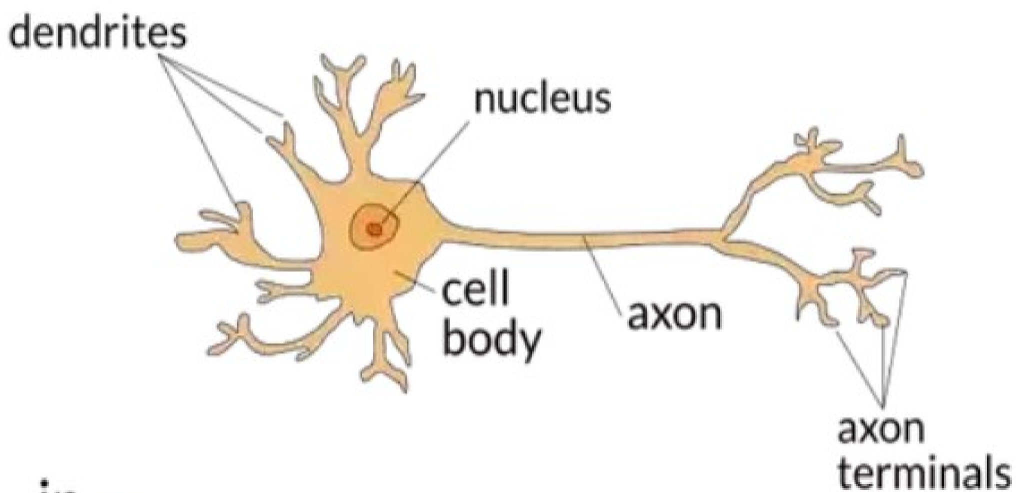
2. From the point of view of the **method applied**, NNs fall within the **parallel distributed processing domain**.

3. The **topological structure of the neurons**:
   - **single-layer** networks
   - **multilayer** networks

4. The **direction** in which the signals flow:
   - **feed-forward** networks
   - **feedback** networks

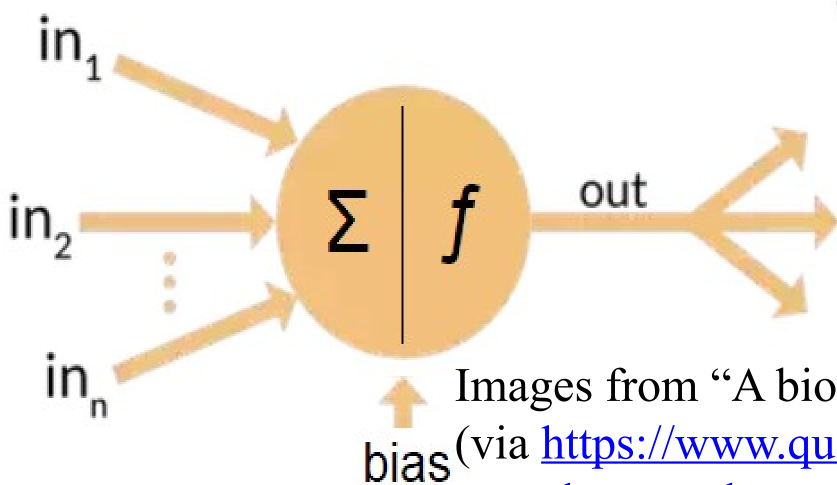
The biological and the artificial neuron

There are between 86 to 100 billion interconnected **neurons** by synapses.



Active – if the information entering the neuron exceeds a certain stimulation threshold
Pasive – otherwise

Biological NN	Artificial NN
Cell body	Network node
Dendrites	Inputs in NN
Axon	Output of NN
Activation	Processing
Synapses	Weighted links



Images from “A biological and an artificial neuron” (via https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network)

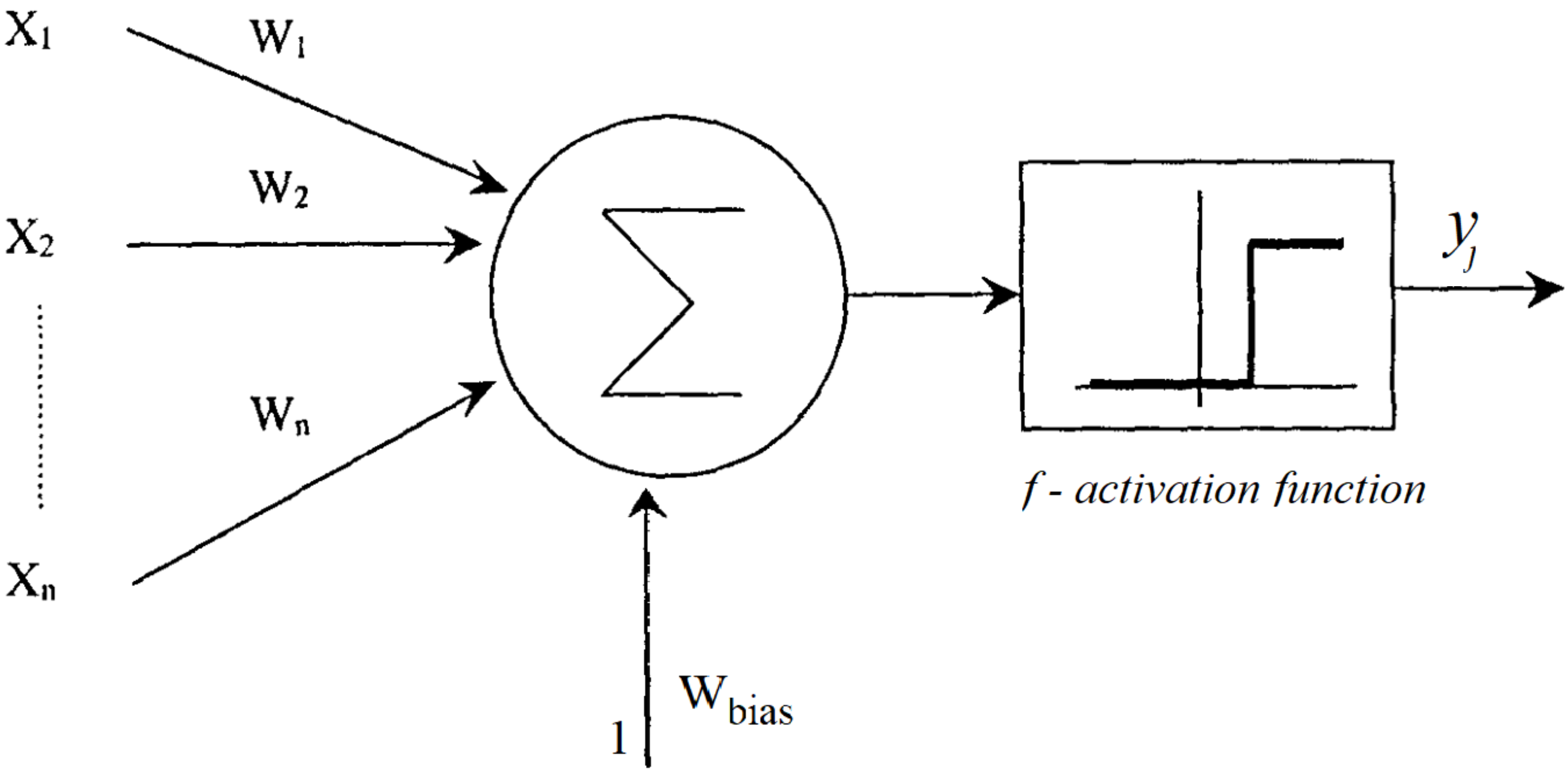
Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

MAPIoT

Mathematical model of an artificial neuron



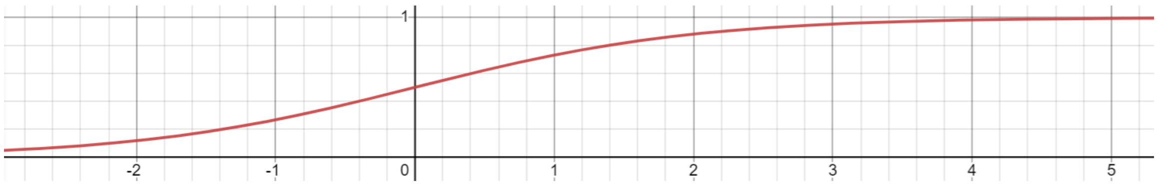
f - activation function

$$y_j = f\left(\sum_i x_i * w_{i,j} + w_{bias}\right)$$

Types of Activation Functions

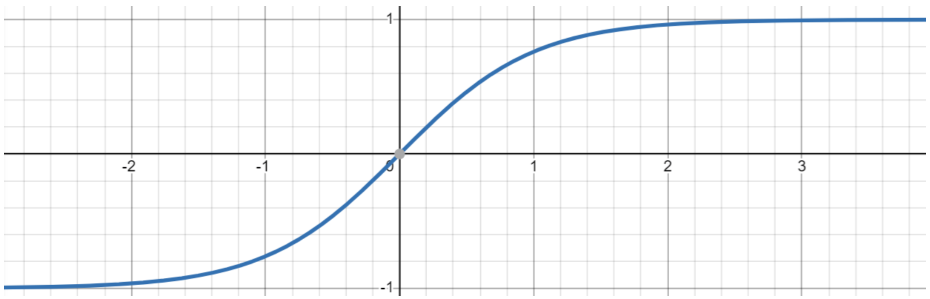
- **Sigmoid** function

$$f(x) = \frac{1}{1 + e^{-x}}$$



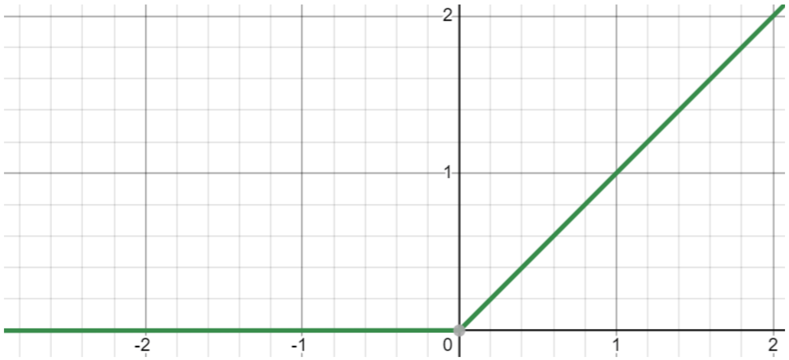
- **Hyperbolic tangent** function

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



- **ReLU (rectified linear unit)** function

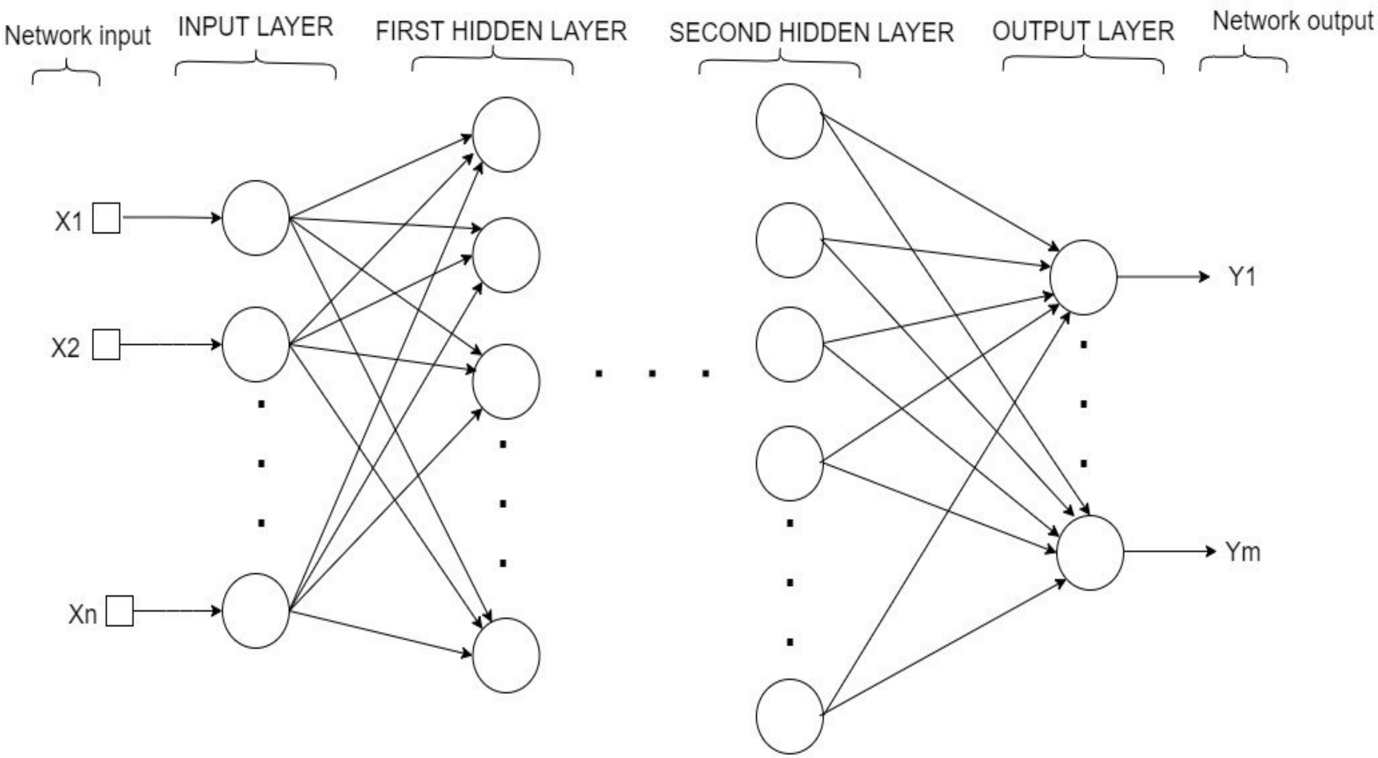
$$f(x) = \max(0, x)$$



- Softmax function

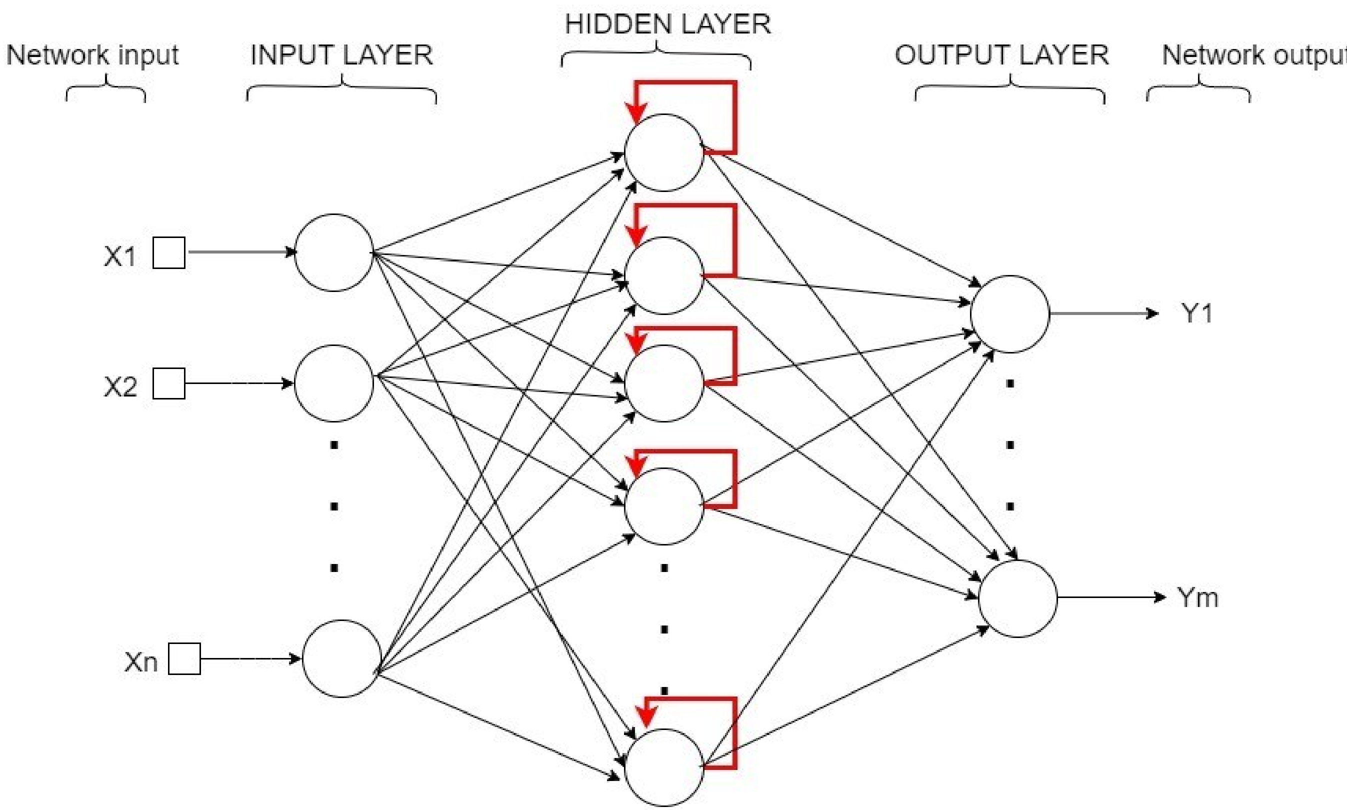
$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j z_j}$$

The architecture of a generic multilayer neural network with two hidden layers – feed forward



X1, X2, ..., Xn - inputs of the neural network
Y1, Y2, ..., Ym - outputs of the neural network

The architecture of a generic multilayer neural network with two hidden layers – feedback (recurrent)



X1, X2, ..., Xn - inputs of the neural network
Y1, Y2, ..., Ym - outputs of the neural network

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

MAPIoT

Neural Networks (NN)

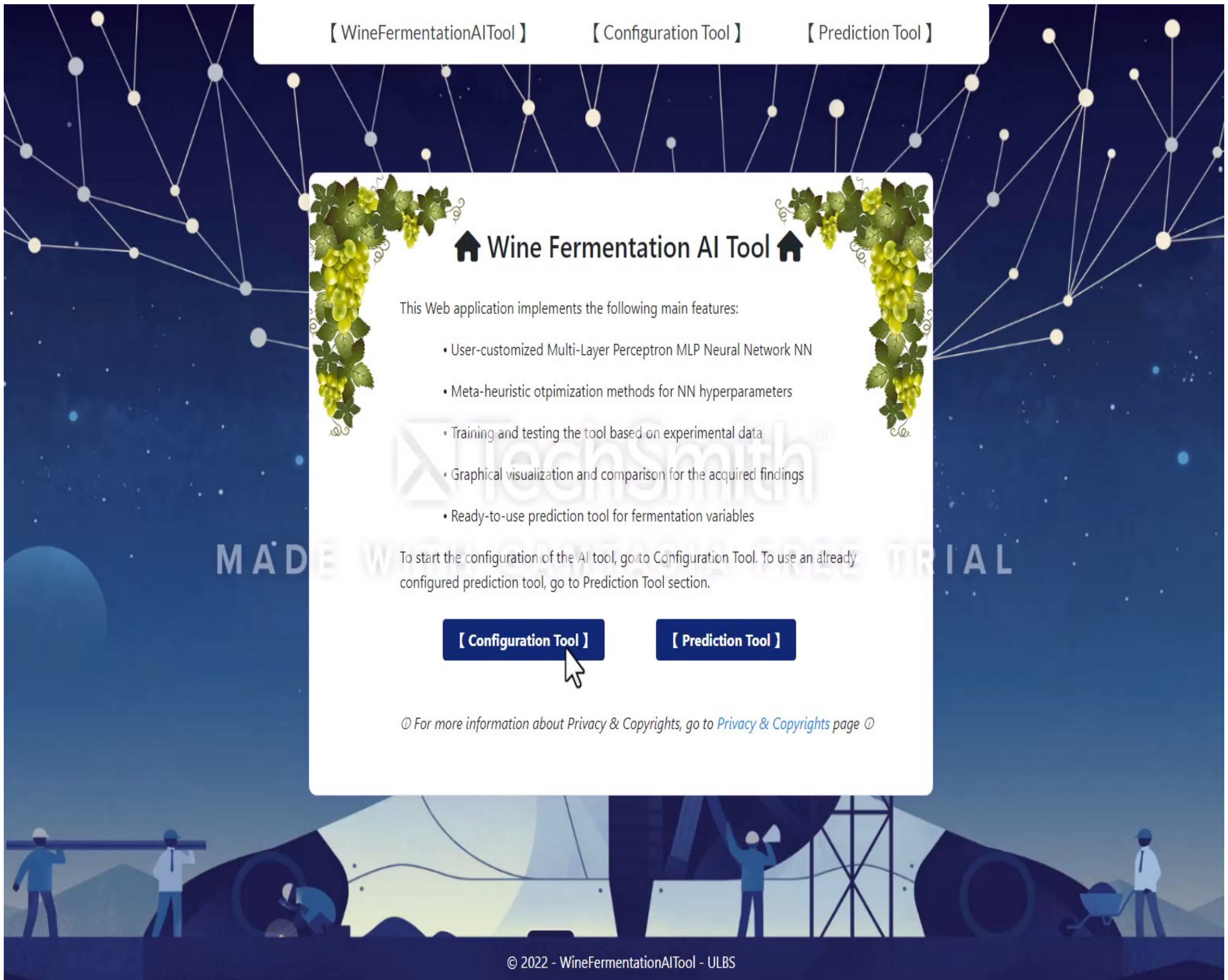
- Simple perceptron and Multi-Layer Perceptron. Constraints
- Learning Mechanism: Backpropagation
- Source code examples: Matlab and C#
  - Creating the network
  - Selection of activation function
  - Metrics
  - Training
- Application & Results
  - Florea, A., Sipos, A., & Stoisor, M. C. (2022). *Applying AI Tools for Modeling, Predicting and Managing the White Wine Fermentation Process*. Fermentation, 8(4), 137.

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

USER INTERFACE



1st approach: NN implementation in MATLAB

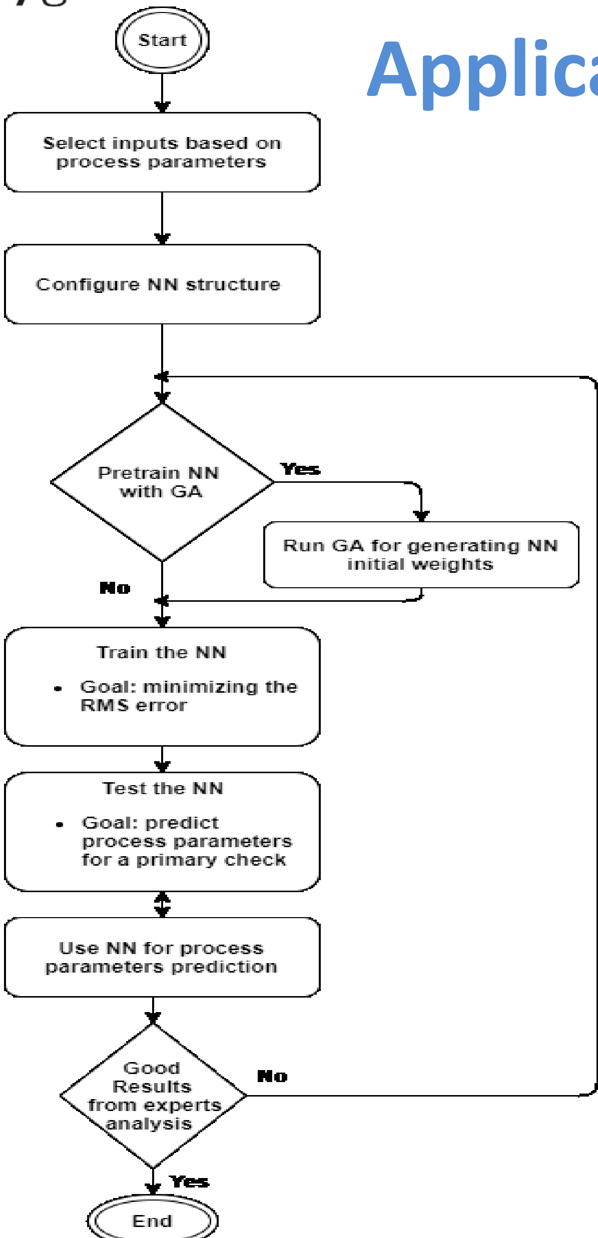
```
% ******* CREATE NETWORK **********************
% create vector dimensiune retea
   dimensiune_retea = [];
   for i = 1:nr_straturiAscunse
           dimensiune_retea = [dimensiune_retea nr_neuroni];
   end
   network = newff(input_antrenareRetea,target_antrenareRetea, dimensiune_retea);
%******* TRAIN NETWORK CONFIG ***********************
% network.trainFcn = 'traingdx'; Gradient descent with momentum and adaptive learning rate BP
%SET THE TRAINING PARAMETERS
    network.trainparam.epochs = nr_iteratii;
% TRAIN THE NETWORK
   trained_network = train(network,input_antrenareRetea,target_antrenareRetea);
%****COMPUTING AND PRINTING THE ERROR *****
  output_antrenareRetea = trained_network(input_antrenareRetea);
  eroare_antrenare = 0;
  for i = 1:length(output_antrenareRetea)
      eroare_antrenare = eroare_antrenare + ((output_antrenareRetea(i) - target_antrenareRetea(i))^2)
  end
  set(handles.edit_EroareAntrenare, 'String', num2str(eroare_antrenare/length(output_antrenareRetea)));
```

2nd approach: NN class members and methods in C#

```
public class NeuralNetwork {
        int N1, N2, N3;
        double[] Expected;
        double[,] TrainingSet, TestingSet;
        public Layer InLayer, HidLayer, OutLayer;
        double LearningRate, TrainingError;
        int MaxEpochs;
        bool IsType1;
        bool[] IsTrainingSet;
        string InitializationType;

        List<double[,]> DataSets;
        GA MyGA;
        PSO MyPSO;
        bool IsSigmoidFunction;
        public NeuralNetwork(bool isType1, int N2, bool isSigmoidFunction,
        double[,] W12, double[,] W23, double[] hBias, double[] oBias);
        private void SetTrainingAndTestingSet();
        private void Forward();
        public double[] Predict(double[] inParams);
        public double TestNetwork(string path);
        …
        }
```

Application & Experiment organization



The application development stages consisted of:

- The **back-end** component implements the functionality of the application the configuration of the NN and its prediction.
- The **front-end** component contains all the design and presentation features of the application (ASP.NET, HTML 5.0, CSS and JavaScript).

Through this implementation mode, the **application is available remotely**, **facilitating its operation without the need to be near the bioreactor**.

Fermentation process parameters

Data sets used for training and testing the NN (in and out parameters).

	DATASET 1	DATASET 2	SIGNIFICANCE
INPUT	T	T	*Temperature (°C)*
	t	t	*time (h)*
	S_0	S_0	*Initial substrate concentration (g/L)*
	X	X	*Biomass concentration (g/L)*
		pH	*pH*
		CO_2	*CO_2 concentration released (percentage volume)*
OUTPUT	P	P	*Alcohol concentration (g/L)*
	S	S	*Substrate concentration (g/L)*

Dataset organization spreadsheet for a fermentation temperature of 24 °C.

INPUT				OUTPUT	
***Time* [h]**	***X* [g/L]**	***T* [°C]**	**S_0 [g/L]**	***P* [g/L]**	***S* [g/L]**
0	0.1	24	210	0.2	210
5	0.1	24	210	0.2	210
...	...	...	...	...	...
127	2.4259	24	210	19.1123	158.7665
...	...	...	...	...	...
197	1.107	24	210	52.0241	24.135

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

MAPIoT

Application for supervising and controlling White Wines Fermentation Parameters Evolution

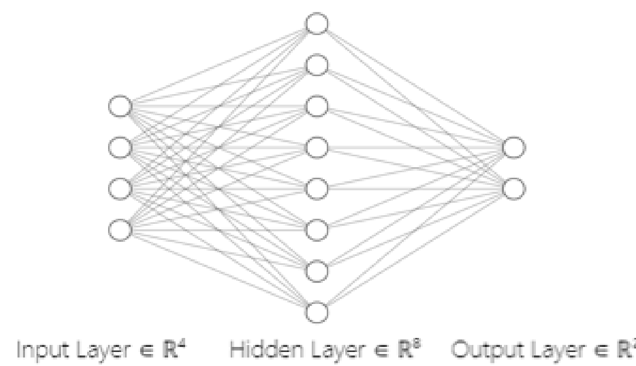
* Configure Neural Network *

Choose Neural Network architecture:

[x] Simulate without pH and CO2 parameters [ ] Simulate with pH and CO2 parameters

Hidden Layer Size:

[ ] N1+2 [x] N1+4 [ ] N1+6 [ ] N1+8



Learning Rate

[ ] 0.1 [x] 0.3 [ ] 0.5 [ ] 0.7 [ ] 0.9

Activation Function

[x] Use Sigmoid function [ ] Use Tanh function [ ] Use ReLU function [ ] Use Softmax function

Weight initialization

[ ] Use random Xavier initilization [x] Use weights pretrained with GA

Max epochs

10000 epochs

NN training and test

- Metrics
  - At the end of the training phase, the errors obtained from training data are calculated with the following **mean square error** formula (MSE):

$$E = \frac{\sum_{i=1}^{M}(NetworkOutput - ScopeValue)^2}{M}$$

where M is the pairs number of the input–output used in training.

- Training (75% of the data) & Testing phases (rest of 25% of the data)

*** Train and Test ***

Choose sets used for training (recommended min. 2):

☑ Data Set 1 (T = 26 °C) ☑ Data Set 2 (T = 24 °C) ☐ Data Set 3 (T = 22 °C) ☑ Data Set 4 (T = 20 °C)

File loaded succesfully! [Choose File] DataSetNormalized.xlsx [Train & Test]

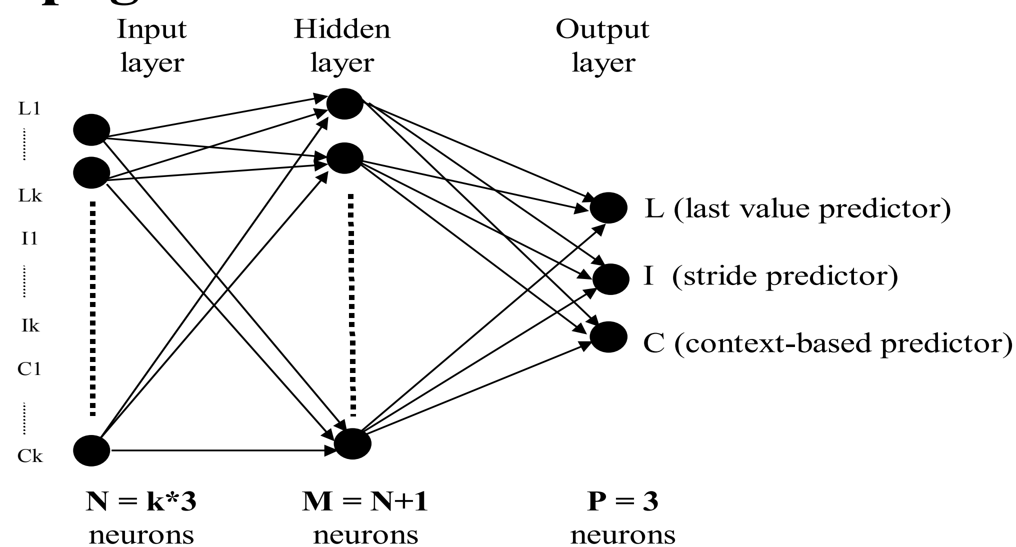
Multi-Layer Perceptron (MLP)

Learning Mechanism: **Backpropagation (BP)**

- Is used in **feed-forward networks**.
- BP comprises **two steps**:
  - The **first step** (**forward**) is where information is passed from input to output, followed by a step from output to input. The forward step **propagates the input vector to the first level of the network**; the outputs of this level produce a new vector that will be the input for the next level until it reaches the last level, where the outputs are the network outputs.
  - The **second step** (**backward**) is similar to the forward step, except that **errors are propagated backward through the network to cause the weights to adjust**. Based on gradient descent for weight adjustment, the BP algorithm uses the chain rule to compute the gradient of the error for each unit with respect to its weights.

Multi-Layer Perceptron (MLP)

- Learning Mechanism: **Backpropagation**
- Feasibility issues. Constraints
  - Neurons from the same layer are not connected



1. Create a feed-forward network with N inputs, one single hidden layer with $M = N+1$ neurons and with P neurons on output layer.
2. Initialize all network weights $W^1_{i,j};\ i=\overline{1,N};\ j=\overline{1,M}$ and $W^2_{i,j};\ i=\overline{1,M};\ j=\overline{1,P}$ to small random numbers belonging to the [0.3, 0.7] interval. In our example the activation function used is sigmoid: $F(x) = \dfrac{1}{1+e^{-x}}$.

   In the following t_k represents the value of k's neuron from the output layer and O_k is the desired value of the same neuron.

Multi-Layer Perceptron (MLP)

- Learning Mechanism: **Backpropagation**

3. *Until* $E(\overline{W}) = \frac{1}{2} \sum_{k \in Outputs(P)} (t_k - O_k)^2 \leq T$ (*threshold*), *do*:

   3.1. Input the instance $\overline{X}$ to the network and propagates forward through the network and compute the output $\overline{O}$ (matrix product).

   $$\overline{O} = \overline{X} \cdot \overline{W^1} \cdot \overline{W^2}$$

   3.2. For each network output unit k, $k \in \overline{1,P}$ calculate its error term δ_k:

   $$\delta_k = O_k(1 - O_k)(t_k - O_k)$$

   3.3. For each hidden unit h, $h \in \overline{1,M}$ calculate its error term δ_h:

   $$\delta_h = O_h(1 - O_h) \sum_{k \in Outputs(P)} W_{k,h}^2 \cdot \delta_k$$

   3.4. Update each network weight $W_{i,j}$:

   $$W_{i,j} = W_{i,j} + \Delta W_{i,j}$$

   $$\Delta W_{i,j} = \alpha \cdot \delta_i \cdot X_{i,j}$$

where α is the learning step and $X_{i,j}$ is the input layer if you want to adjust the weights between the input level and the hidden layer, δ_i being δ_h, or is the hidden layer if the weights between the hidden layer and the output layer are to be updated, respectively, δ_i being δ_k in this case.

Multi-Layer Perceptron (MLP)

- Learning Mechanism: **Levenberg-Marquardt (LM)**

The Levenberg-Marquardt algorithm is an iterative optimizational technique that uses aspects of the **gradient descent** and Gauss-Newton method and is fast in practice, as we demonstrated in our experiments.

MATLAB implementation: Simulation results obtained with the original data using the Levenberg-Marquardt and the Backpropagation algorithms respectively.

No. of Iterations	No. of Hidden Layers	No. of Neurons on Hidden Layers	Average Error	
			Levenberg-Marquardt Algorithm	Backpropagation Algorithm
1500	1	5	3.5498	5.6236
1500	1	6	2.2282	5.4599
1500	1	7	1.6872	4.5100
1500	1	8	1.7170	3.9243
1500	1	9	1.3018	4.6077
1500	1	10	1.9558	5.9983
1500	1	11	2.0424	3.8234
1500	1	12	**0.9954**	**4.0962**

net.trainFcn = '**trainlm**' – in MATLAB
[trainedNet,tr] = train(net,...)

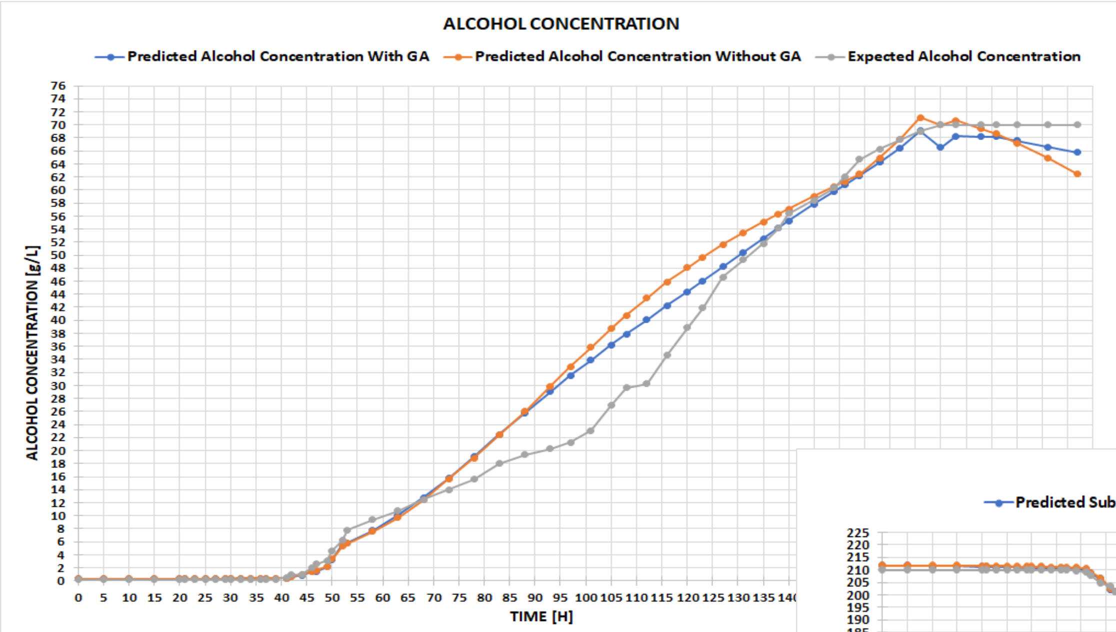
Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

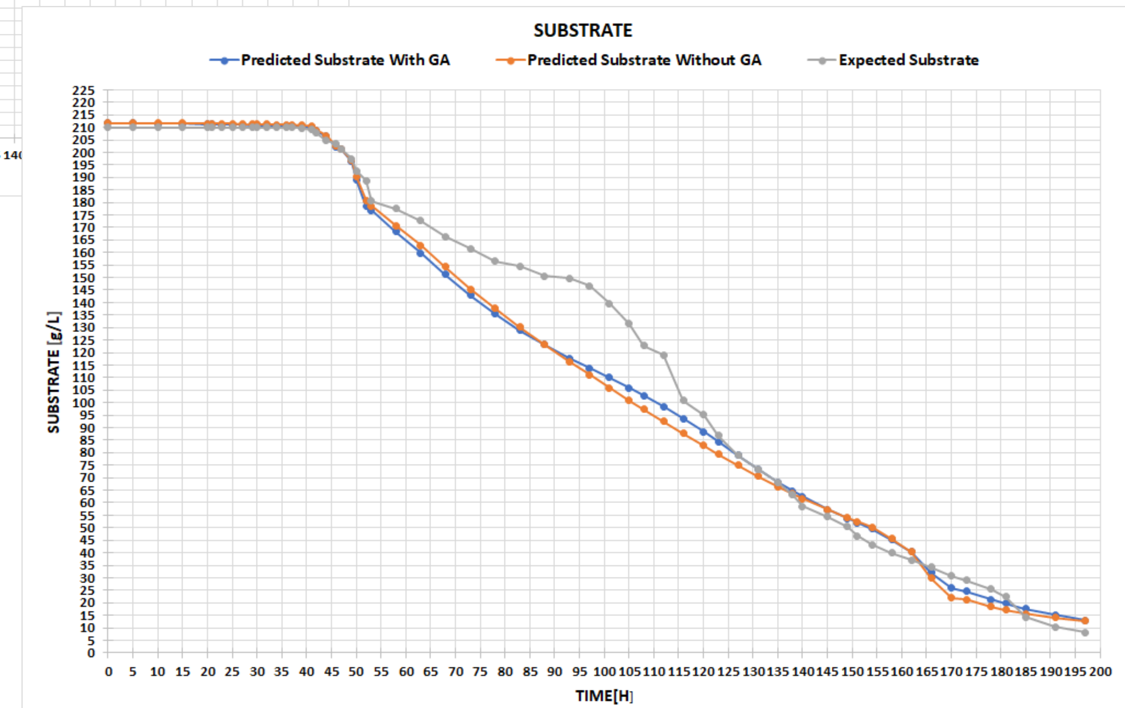
University of South-Eastern Norway

Neural Networks Results (I)

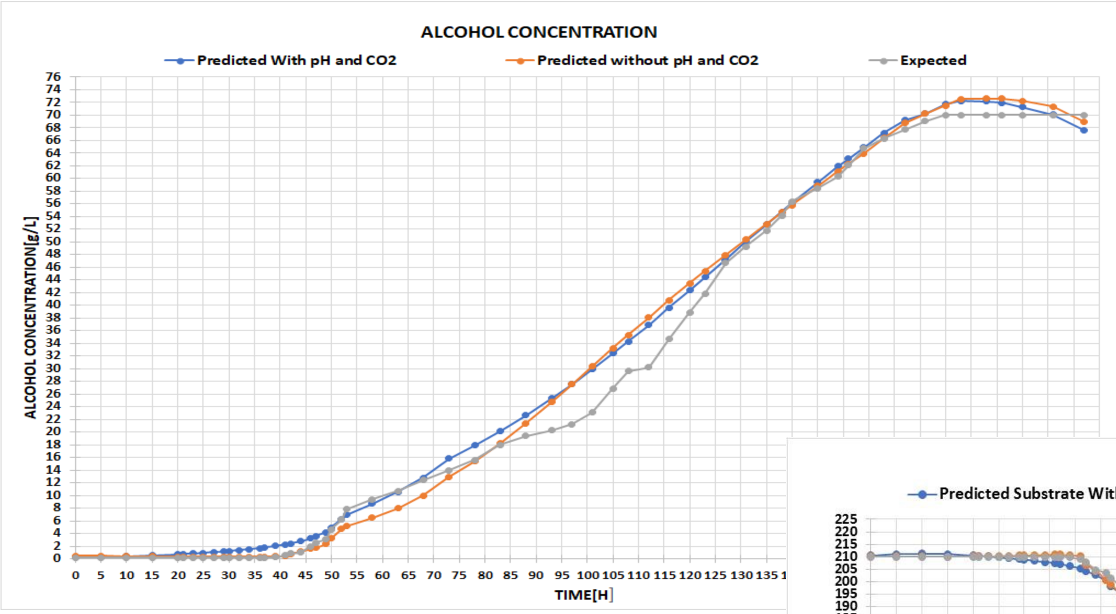
Random Initialization vs. GA Initialization of NN weights



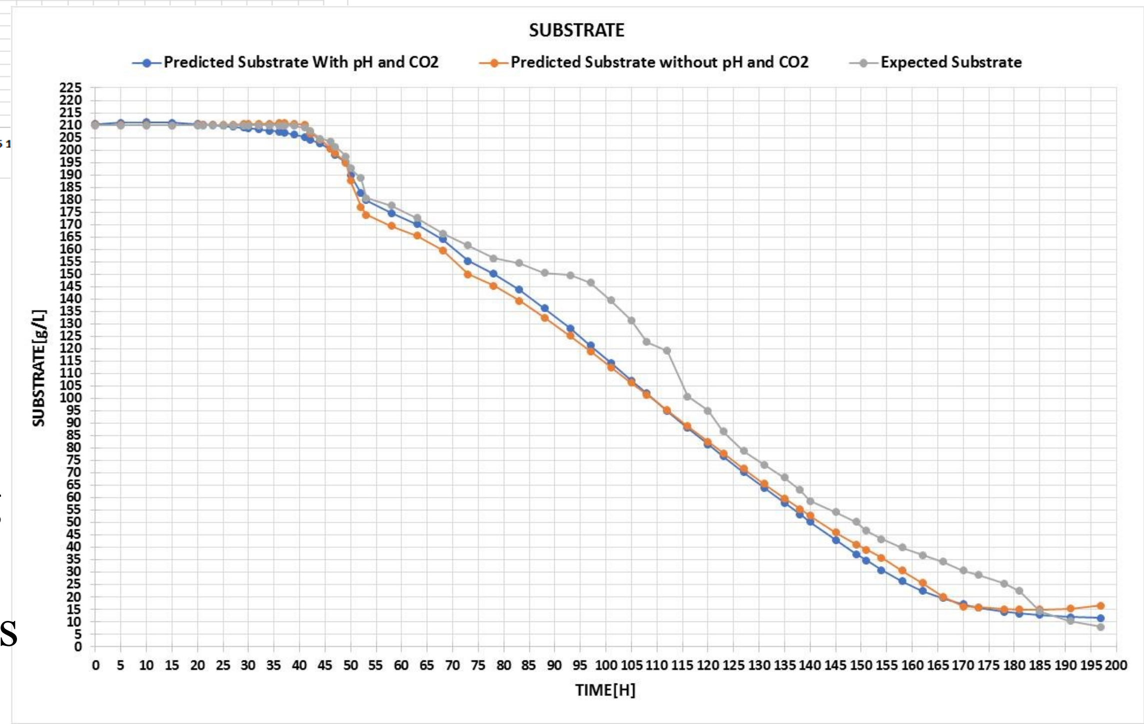
We achieved **better results with the neural network pre-trained with the genetic algorithm**, with a testing error of 0.03 compared to 0.045 in the case of the pseudo-random Xavier initialization.



Neural Networks Results (II)



Incorporating the pH and the released CO_2 in the prediction process



There was an increase in the prediction accuracy once the additional information about the pH and the released carbon dioxide was incorporated, leading to the conclusion that a larger amount of data positively influences the performance of the NN.

Neural Networks: Challenges

- The choice of alpha (**learning rate**) greatly influences the backpropagation learning algorithm based on minimizing the mean square error. However, its choice depends on the specifics of the problem.
- Although **there is no universal method for choosing in a given problem**, it is recommended that it be subunit or possibly decreasing with increasing iteration number. Usually the most convenient value is chosen after **laborious simulations**.
- **Difficult to customize Machine Learning models** for specific food types or recall types
- Are necessary **complex Machine Learning models** and **large volumes of data to ensure accurate predictions**
- **Time-consuming simulations** and AI models training requiring days/weeks to be performed

Genetic Algorithms (GA)

The most widely known type of Evolutionary Algorithms

- Quick overview
- Advantages of GA. Limitations of GA.
- Structure of a Genetic Algorithm (pseudocod)
- GA: Individual/Chromosome Representation
- GA: Fitness & parents selection
- Genetic operators: Recombination & Mutation
- GA: Survivor selection
- Application & Results

https://webspace.ulbsibiu.ro/adrian.florea/html/Planificari/EvolutionaryComputing/Planif_EvolutionaryComputing_ACS_2.pdf

Iceland Liechtenstein Norway grants

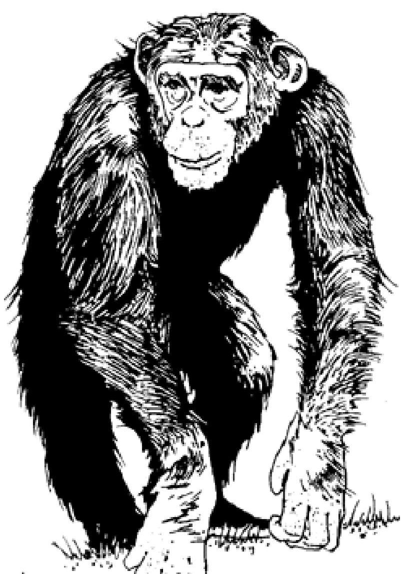
UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

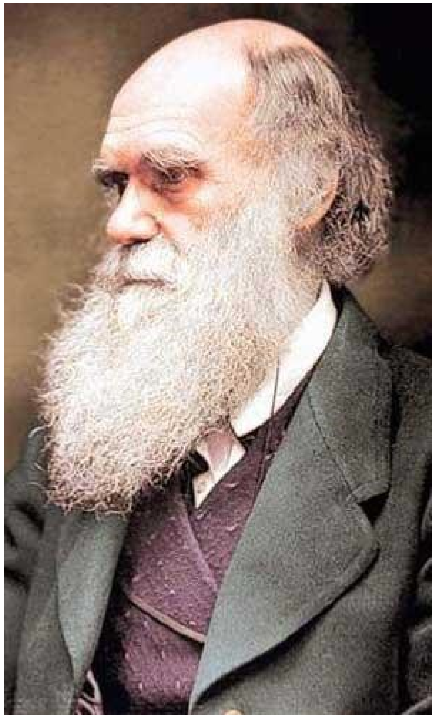
USN University of South-Eastern Norway

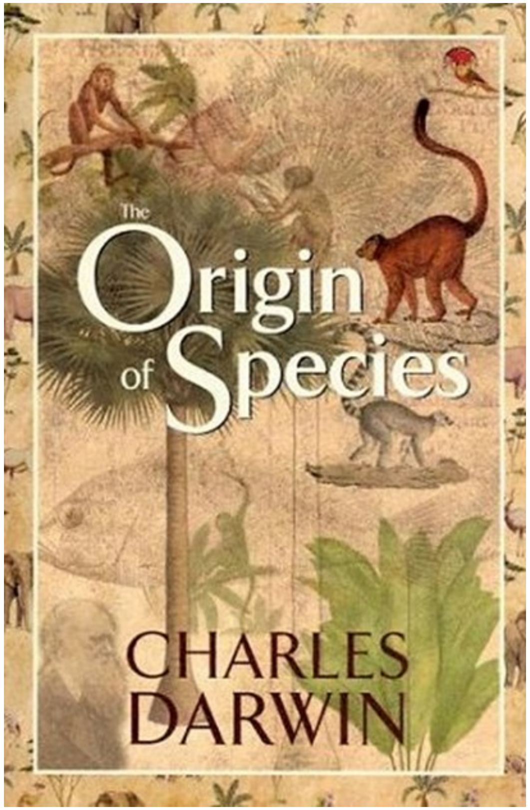
MAPIoT

Evolutionary Algorithms Inspiration: Mother Nature

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."







1859

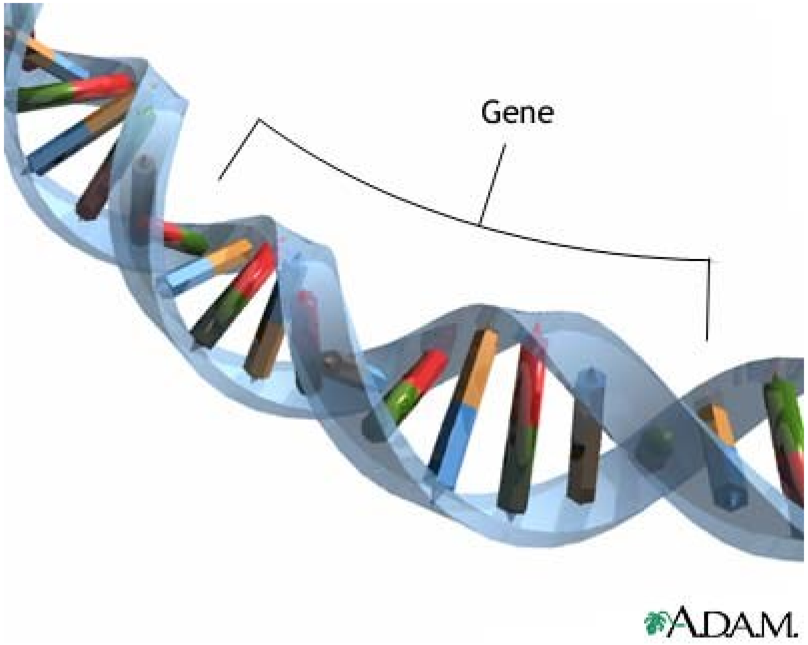
How survive ?

In Nature:

- Competition,
  survival of the fittest
- Variation introduced into a population
  - **Parental recombination**
  - **Mutation**
- Variations that provide a selective advantage stick around:
  - **elitist / generational**

How survive ?

- It's all about the **DNA**
- **Heritable traits**: genes



At the Molecular Level:

Variation introduced through **parental recombination** and **mutation**

Variation Through Mutation

In nature:

- Environmental factors
- Radiation
- Oxidation
- Mistakes in replication or repair

What is Evolutionary Computing / Algorithms ?

- Evolutionary computing began by lifting **ideas from biological evolutionary theory into computer science**, and continues to look toward new biological research findings for inspiration.
- Darwin’s principle “**Survival of the fittest**” and „**Natural selection** and **genetic inheritance**!” can be used as a **starting point** in introducing evolutionary algorithms / computing.
- Although the history of evolutionary computing dates back to the **1950s and 1960s**, only within **the last two decades** have evolutionary algorithms became practicable for solving real-world problems on desktop computers.

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

MAPIoT

The Main Evolutionary Computing Metaphor

EVOLUTION		PROBLEM SOLVING
Environment	⟷	Problem
Individual	⟷	Candidate Solution
Fitness	⟷	Quality
Population	⟷	Set of potential solutions
Chromosome	⟷	Encoding of potential solutions
Gene	⟷	Part of encoding

Fitness $\rightarrow$ chances for survival and reproduction

Quality $\rightarrow$ chance for seeding new solutions

Advantages of GA

- Parallelism
- Solution space is wider
- The problem has multi objective function
- Easily modified for different problems
- They require no knowledge or gradient information about the response surface and only uses function evaluations
- They are resistant to becoming trapped in local optima
- They perform very well for large-scale optimization problems
- Can be employed for a wide variety of optimization problems

Limitations of GA

- The problem of identifying fitness function
- Definition of representation for the problem
- Premature convergence occurs
- Cannot easily incorporate problem specific information
- Not good at identifying local optima
- Needs to be coupled with a local search technique (*memetic algorithms*)
- Have trouble finding the exact global optimum
- Require large number of response (fitness) function evaluations
- Configuration is not straightforward

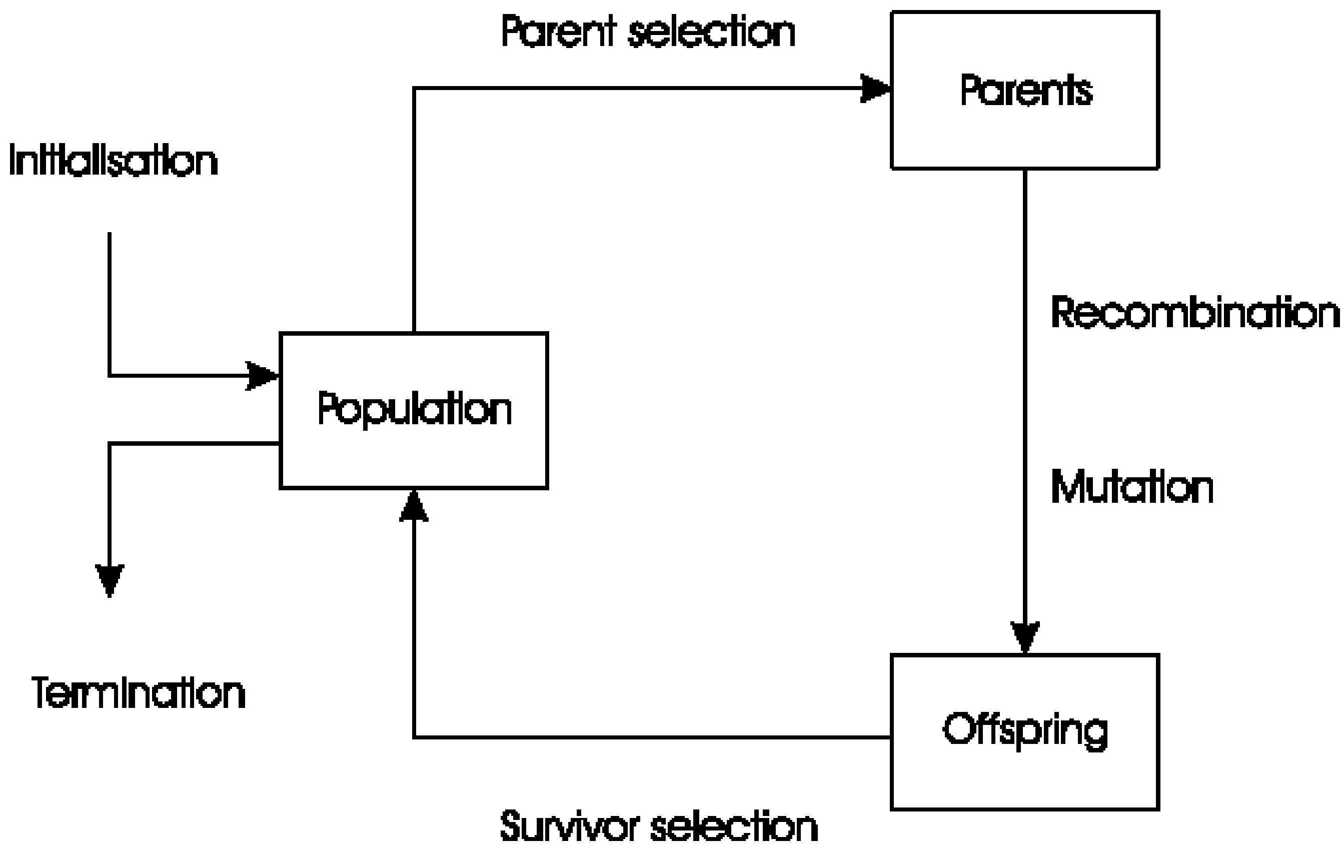
Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

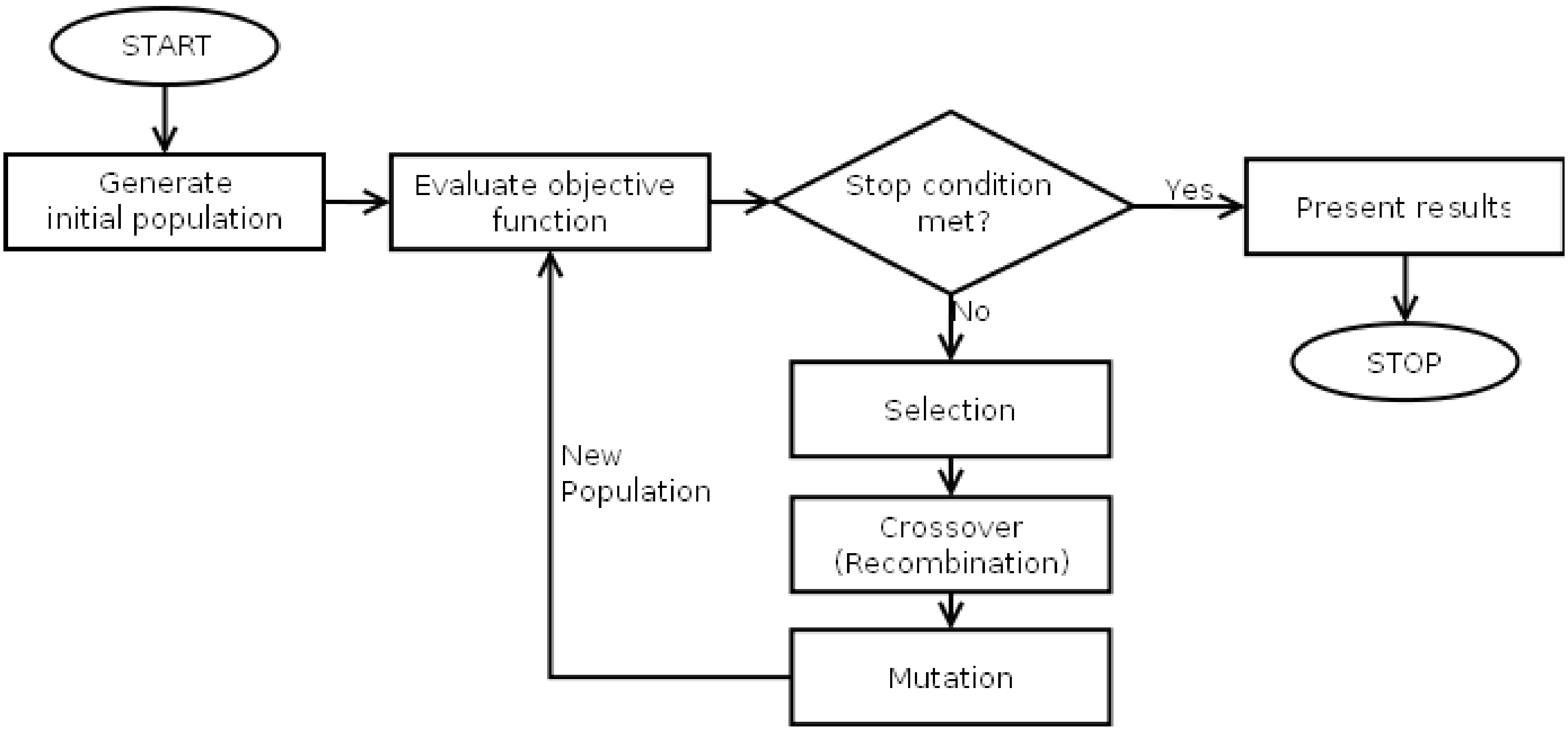
University of South-Eastern Norway

MAPIoT

General Scheme of EAs



The generic structure of EAs



Pseudo-code for typical EA

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

1 – Select parents – **stochastic**

5 – Survival selection – **deterministic**

Genetic Algorithms (pseudocod)

```
Procedure GA{
   t = 0;
   Initialize P(t);
   Evaluate P(t);
   While (Not Done)
   {
      Parents(t) = Select_Parents(P(t));
         Offspring(t) = Procreate(Parents(t));
         Evaluate(Offspring(t));
         P(t+1)= Select_Survivors(P(t),Offspring(t));
         t = t + 1;
   }
```

To solve a problem using a genetic algorithm is necessary to define a **fitness function** (F) to evaluate the performance of each chromosome.

GA: Individual / Chromosome Representation

Representation - The most critical decision in any application, namely that of deciding how best to represent a candidate solution of the algorithm

- **Binary encoding**
- **Permutation**
- **Integer encoding**
- **Real valued problems**

Based on the representation – it depends the setting of genetic operators and the computing of fitness function!

GA: Binary Representation

Binary Representation (*knapsack problem* object selection, *loading trucks*, combinatorial problems, maximum/minimum, *FPGA chip design*)

The **knapsack problem** is a problem in *combinatorial optimization*: **Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible**.

It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most useful items.

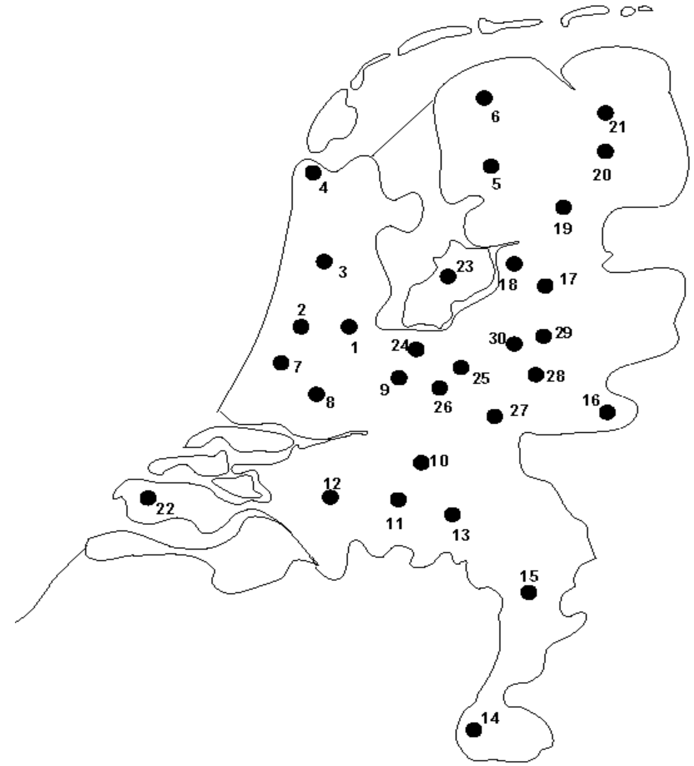
A bitvector with an entry for each potential item, a 1 if in the sack, a 0 if not

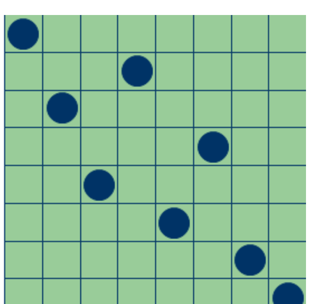
0	0	1	0	1	1	0	1	0	0

GA: Permutation representation

Permutation Representation (Traveling Salesman Problem / Vehicle Routing Problem)

- Problem:
  - Given *n* cities
  - Find a complete tour with minimal length
- Encoding:
  - Label the cities 1, 2, … , *n*
  - One complete tour is one **permutation** (e.g. for n =4 [**1,2,3,4**], [3,4,2,1] are OK)
- Search space is BIG:
  for 30 cities there are 30! ≈ 10^{32} possible tours





The 8 queens problem

Obvious mapping

1	3	5	2	6	4	7	8

GA: Integer Representation

Integer Representation (image processing parameters, microarchitectural configuration)

```
public class PSATSimSolutionType: SolutionType {
      public override Variable[] CreateVariables(Problem problem)
       {
        var variables = new PSATSimVariable[problem.NumberOfVariables];
        variables[0] = new PSATSimVariable(1, 16, "Super-Scalar Factor");
        variables[2] = new PSATSimVariable(1, 512, "Reorder entries", MutationType.Exponential);
        variables[6] = new PSATSimVariable(1, 8, "Integer Execution Units");
        …
        }}
```

Extend bit-flipping mutation from binary representation to make random choice (esp. categorical variables)

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

MAPIoT

GA: Real values Representation

Real valued problems (continuous parameter optimisation, using a GA for Neural Network training in order to increase prediction accuracy)

$w^{i}_{0,0}$	$w^{i}_{0,1}$	...	$w^{i}_{0,7}$	$w^{i}_{1,0}$	...	$w^{i}_{1,7}$	...	...	...	$w^{i}_{3,7}$	$w'^{i}_{0,0}$	...	...	$w'^{i}_{7,1}$

Each individual has a **finite number of genes**, which in this case are represented by the **linear matrix of weights**, where one weight of NN corresponds to one gene.

```
#define NR_CRO 50
class CGeneticLearn : public CLearningAlg {
private:
...
    long unsigned HRG;
    long unsigned HRLTable[1024];
    int nrno_l1;        int nrno_l2; // number of neurons on Level 1 and Level 2
    int nrp_l1;         int nrp_l2;  // number of wights on Level 1 and Level 2
    double gene[NR_CRO][500];
    int nrGen;
    void Simulate(CString file);
    void Reproduce(int i1,int i2,int j1, int j2);
    void Cross();
    void Mutate();
    void calcEvals();
    void Init();
public:
    CString str;
    CGeneticLearn(CNetw* m_nntw,CProjView* view,CString file,bool* done);
    virtual ~CGeneticLearn();
    void Run();
};
```

GA: Mapping real values on bit strings

$z \in [x,y] \subseteq \Re$ represented by $\{a_1,\ldots,a_L\} \in \{0,1\}^L$

- $[x,y] \rightarrow \{0,1\}^L$ must be invertible (one phenotype per genotype)
- Γ: $\{0,1\}^L \rightarrow [x,y]$ defines the representation

$$\Gamma(a_1,\ldots,a_L) = x + \frac{y-x}{2^L-1} \cdot \left(\sum_{j=0}^{L-1} a_{L-j} \cdot 2^j\right) \in [x,y]$$

- Only 2^L values out of infinite are represented
- L determines possible maximum precision of solution
- High precision → long chromosomes (slow evolution)

GA: Parent Selection Methods

GA researchers have used a number of parent selection methods. Some of the more popular methods are:

– **Proportionate Selection (FPS or *roulette wheel*)**

  - individuals are assigned a probability of being selected based on their fitness

    $$p_i = f_i / \Sigma f_j$$

  - where p_i is the probability that individual *i* will be selected, f_i is the fitness of individual *i*, and Σf_j represents the sum of all the fitnesses of the individuals with the population.

– **Rank based Selection (Linear, Exponential)**

  - Attempt to remove problems of FPS by **basing selection probabilities on relative rather than absolute fitness**
  - This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time

– **Tournament Selection**

  - Two members are selected at random to compete against each other with only the winner of the competition progressing to the next level of the tournament.

GA: Genetic Procreation Operators

- Genetic Algorithms typically use two types of operators:
  - **Crossover** (Sexual Recombination), and
  - **Mutation** (Asexual)
- **Crossover** is usually the **primary operator** with **mutation serving** only as a mechanism **to introduce diversity** in the population.
- However, when designing a GA to solve a problem it is not uncommon that one will have to **develop unique crossover and mutation operators that take advantage of the structure of the chromosomes** comprising the search space.

GA: Types of Crossover

- However, there are a number of **crossover operators** that have been used on **binary and real-coded GAs**:
  - **Single-point** Crossover
  - **n-point** Crossover
  - **Uniform** Crossover
  - **Half-uniform** Crossover
- **Crossover operators** used on **permutation representation of GAs**:
  - **Order 1 crossover** Crossover
  - **Partially matched** Crossover
  - **Cycle** Crossover
  - **Edge** Crossover

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

University of South-Eastern Norway

GA: Alternative Mutation Operators

- Through mutation are introduced in the population individuals which could not be generated by other mechanisms.
- Mutation operators used on **binary and real-coded GAs**:
  - **Strong** Mutation
  - **Weak** Mutation
  - **Single chromosome (individual)** mutation
  - **Not uniform** mutation
  - **Adaptive not-uniform** mutation
- Mutation operators used on **permutation representation of GAs**:
  - **Insert** Mutation
  - **Swap** Mutation
  - **Inversion** mutation
  - **Scramble**mutation

GA: Crossover OR mutation?

- Decade long debate: **which one is better / necessary** / main-background
- Answer (at least, rather wide agreement):
  - it **depends on the problem**, but
  - in general, **it is good to have both**
  - both have another role
  - **mutation-only-EA is possible**, **xover-only-EA would not work**
- Achieving a balance between information **exploitation** and by the state-space **exploration** to obtain new better solutions, is typical of all powerful optimization methods.
- If the solutions obtained are exploited too much, then reaches a **premature convergence**.
- On the other hand, if too much emphasis on exploration, it is possible that the **information already obtained is not used properly**. Search time grows and approaches that of random search methods.

GA: Crossover OR mutation? (cont'd)

Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem

Exploitation: Optimising within a promising area, i.e. using information

There is **co-operation AND competition** between them

- **Crossover is explorative**, it makes a *big* jump to an area somewhere "in between" two (parent) areas
- **Mutation is exploitative**, it creates random *small* diversions, thereby staying near (in the area of) the parent

An Example Run (*Steady-State* GA)

- This process of:
  - Selecting two parents,
  - Allowing them to create two offspring, and
  - Immediately replacing the two worst individuals in the population with the offspring
- Is repeated until a stopping criterion is reached
- Notice that on each **cycle the steady-state GA will make two function evaluations while a generational GA will make P** (where P is the population size) **function evaluations**.
- Therefore, you must be careful to count only function evaluations when comparing generational GAs with steady-state GAs.

GA: Additional Properties

- **Generation Gap**: The fraction of the population that is replaced each cycle. A generation gap of 1.0 means that the whole population is replaced by the offspring. A generation gap of 0.02 (given a population size of 100) means two offsprings replace two parents.

- **Elitism**: The fraction of the population that is guaranteed to survive to the next cycle. An elitism rate of 0.98 (given a population size of 100) means 98 parents survive and an elitism rate of 0.02 means that only 2 parents survive.

Applying GA for training weights of NN

**** Configure Genetic Algorithm ****

Structural parameters:

Population size

50 [individuals]

Max Epochs for fitness function

1000 [epochs]

Max Generations for genetic algorithm

100 [generations]

Genetic operators customization:

Crossover rate (%)

85 [%]

Mutation rate (%)

35 [%]

Beta coefficient

0.75

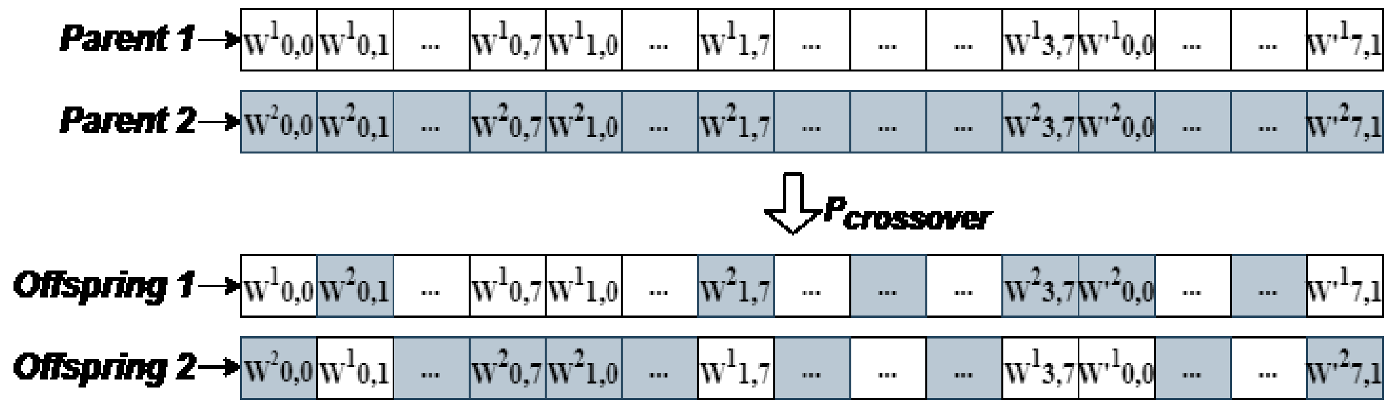
Selection Method

☑ Elitist ☐ Tournament

The **fitness function** used to evaluate each individual is the **backpropagation algorithm itself**, while the **fitness score** is the output of this algorithm, **the network error that resulted after training** the network for the specific number of epochs selected by the user.

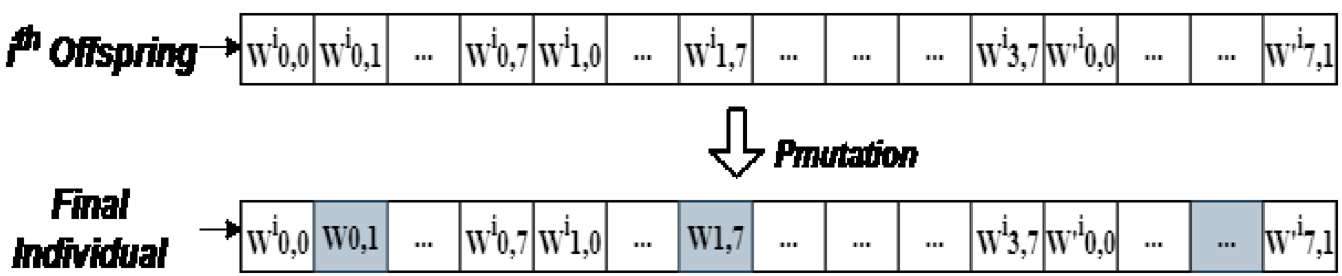
Bring diversity to population in GA

Uniform crossover, with $P_{crossover}$ probability



$$P_{mutation\_new} = P_{mutation\_old} * e^{(1-generation_{no})*Beta}$$

Uniform mutation with $P_{mutation}$ probability



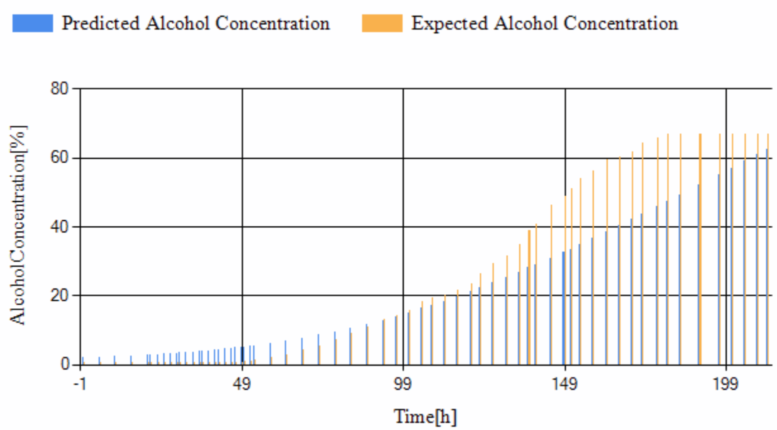
Running WineFermentation and training NN with GA

Network results

TrainingError: 1.15767014387585 **TestingError:** 0.364959310036762

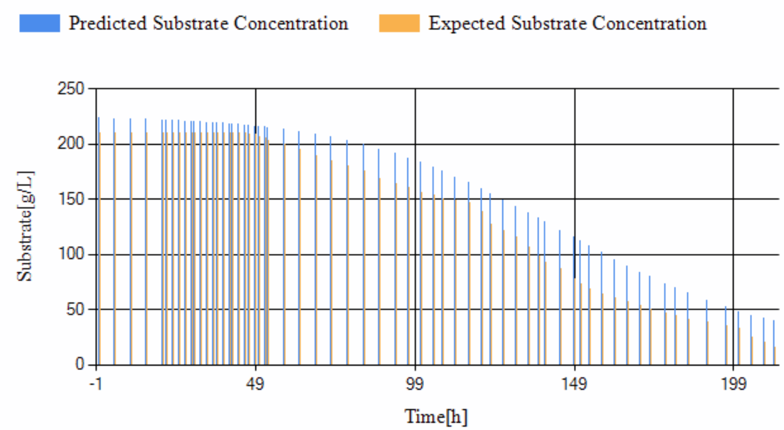
Select Chart Type: Point

Alcohol Concentration



Select Chart Type: Point

Substrate



Predict Fermentation Process Parameters

Input params:

Biomass [g/L]: 0.519

Time [h]: 53

Temperature [C]: 22

Initial substrate [g/L]: 210

CO2 [g/L]:

Ph:

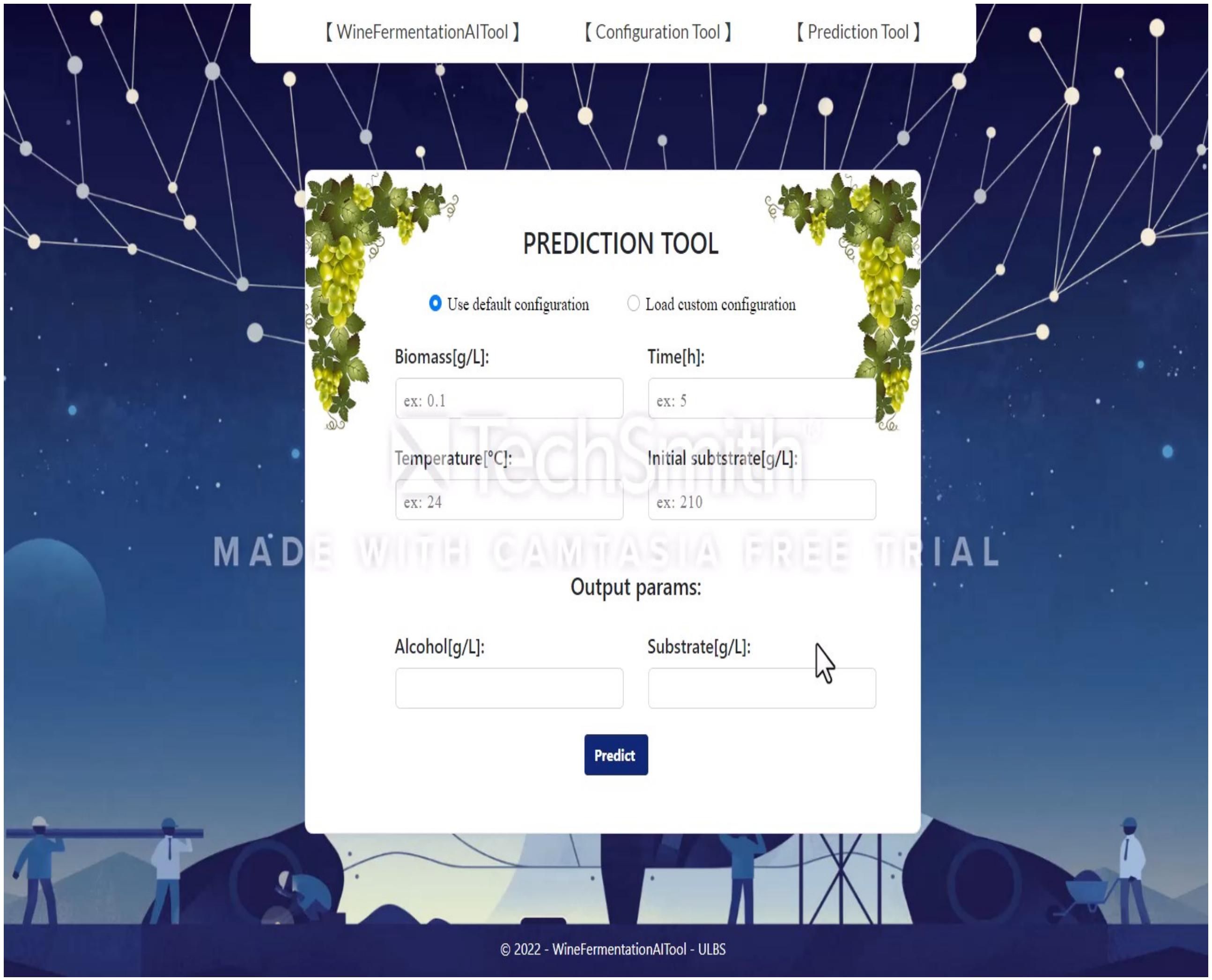
Output params:

Alcohol concentration [g/L]: 3.678504418

Substrate [g/L]: 219.0712878

Predict

PREDICTION TOOL



Fuzzy Systems (FS) & Fuzzy Rules (FR)

- Quick overview. Terminology: Input, output, membership functions
- Advantages of FS
- Fuzzy Rules: Fuzzification, Inference and Defuzzification
- Application: Irrigation system automate control / Wine fermentation

Advantages of Fuzzy Logic Systems (FLS)

Fuzzy Logic is a **method of reasoning (a *precise problem-solving methodology*)** that resembles human reasoning applied in **industrial process control systems.**

- The **mathematical foundations** of fuzzy reasoning are extremely straightforward
- The **flexibility** of fuzzy logic allows you to change a FLS by **simply adding or deleting rules**
- **Robust setup -** FLS help in dealing engineering uncertainties: are capable of **accepting distorted, noisy, imprecise** input data
- FLS **are simple to build and comprehend**
- Fuzzy logic systems **can be programmed** in a situation **when feedback sensors stop working**
- **Large applicability:** cement factories, steam turbines, trains and subways, water purification, irrigation process, wine fermentation, power plants, prediction of energy production / consumption, airplanes, autonomous driving and car equipments (automatic transmision, braking system), household appliances, etc.

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

MAPIoT

FR: Quick overview. Terminology

- **Fuzzy logic** are based on fuzzy set theory **developed by** Lotfi Zadeh since **1965**. **Classical sets** can be described by a **characteristic function**. In the classical theory of sets, the characteristic function associated of a set A, for a given element *x* is 1 or 0 depending if *x* belongs or not to A, respectively.
- **Crisp values** – physical quantities that take **real, precise, well-determined values in a range**
- Crisp rules have problems with **uncertainty**
- **Fuzzy sets** are supersets of **crisp sets** aimed for:
  - Modeling of **imprecise concepts** like

    Age, Weight, Height, …
  - Modeling of **imprecise dependencies (rules)**:

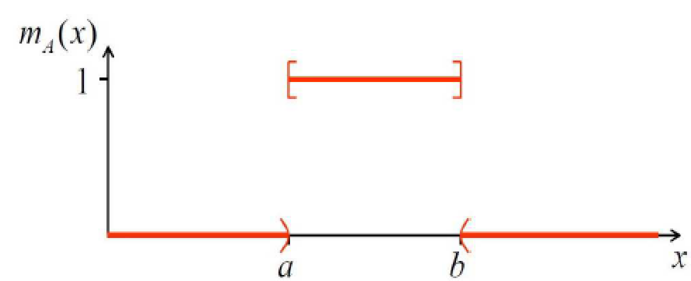
    IF age IS young AND car-power IS high THEN risk IS high
  - **Representation of information extracted from inherently imprecise data**

Crisp rules

- Consist of **antecedents** and **consequents**
- Each part of an **antecedent** is a logical expression
  - e.g. **A > 0.5**, **light is on**
- **Consequent will be asserted if antecedent is true**
  - IF (Presentation is Dull) AND (Voice is Monotone)
  - THEN Lecture is boring
- **Difficulties:**
  - Only **one rule at a time** allowed to fire
  - A rule will either fire or not fire; Sequential firing of rules also is a problem (**ordering the rules** to fire!)
  - Crisp rules **have problems with uncertainty** - representing concepts like *small, large, thin, wide*
  - Classical Crisp Sets can be described by a characteristic function ⇒

$$m_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases} \qquad m_A(x) \in \{0,1\}$$

Example: A = {x | a ≤ x ≤ b}

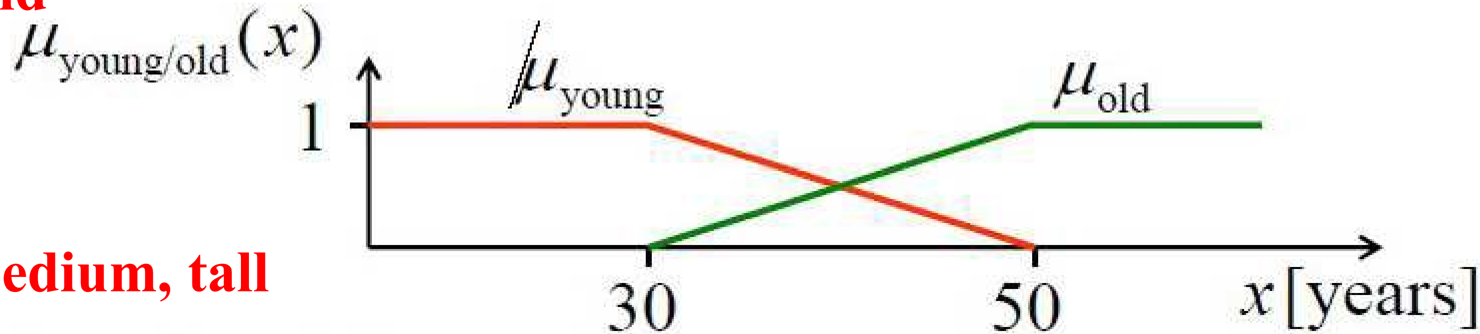


FR: Terminology. Linguistic variables and values

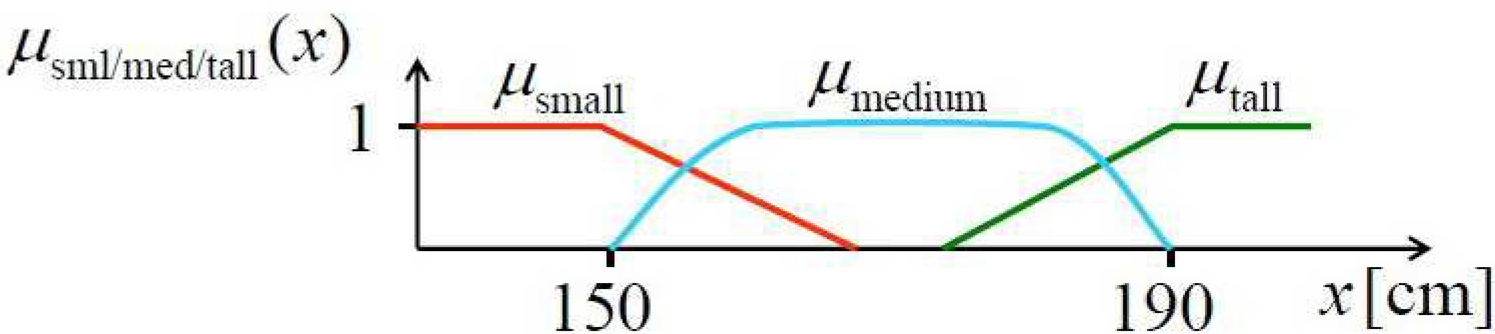
- Associating meaning (semantic) with fuzzy sets results in:
  - ➢ **Linguistic Variables**: the (labeled!) **domain of the fuzzy sets**
  - ➢ **Linguistic Values**: a (labeled!) **collection of fuzzy sets on this domain**
  - ➢ Linguistic values are **inherently context dependent!**

- Examples:

  Age: **young, old**



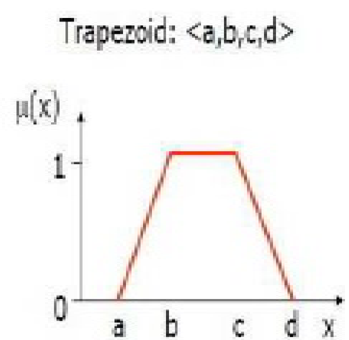
  Size: **small, medium, tall**

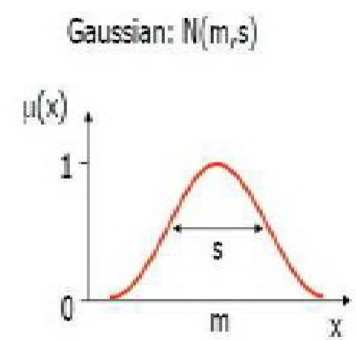


Fuzzy Sets

- Supersets of crisp sets
- Items can belong to varying degrees:
  - **degrees of membership**
  - **usually in [0,1]**
- Fuzzy sets are defined in two ways:
  - **membership functions (MF, μ_A) -** return the degree of membership in a fuzzy set (*A* in this case)
    - Many different types in existence:

      Gaussian, Triangular, Trapezoid, Singleton

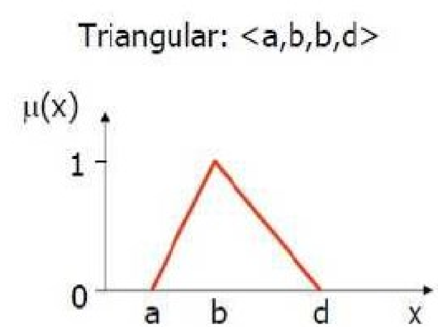




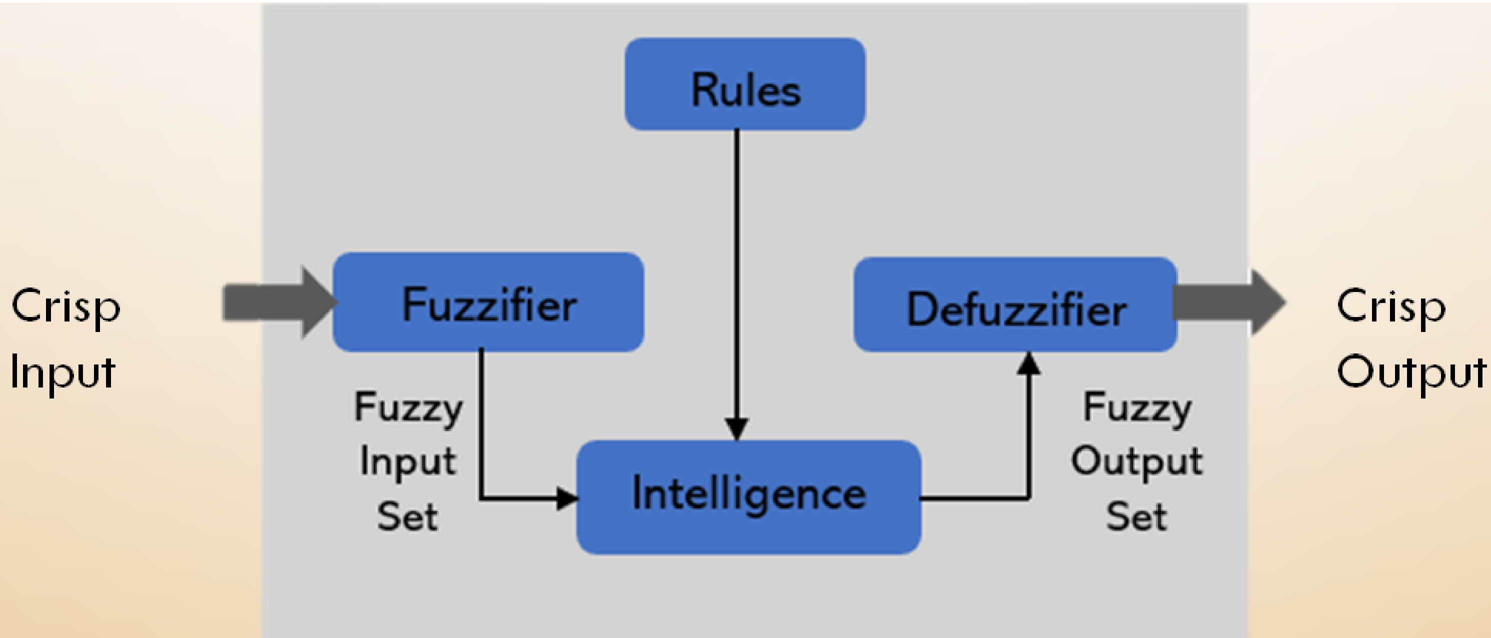
  - **sets of ordered pairs (Crisp, Fuzzy) values**

Membership functions.

Parameter	Range	Small	Big
SLVP associativity	1	1	0
	(1, 8)	Linearly decreases to 0	Linearly increases to 1
	8	0	1
DL1 cache	[16, 2048]	1	0
	(2048, 32768)	Linearly decreases to 0	Linearly increases to 1
	[32768, 8388608]	0	1



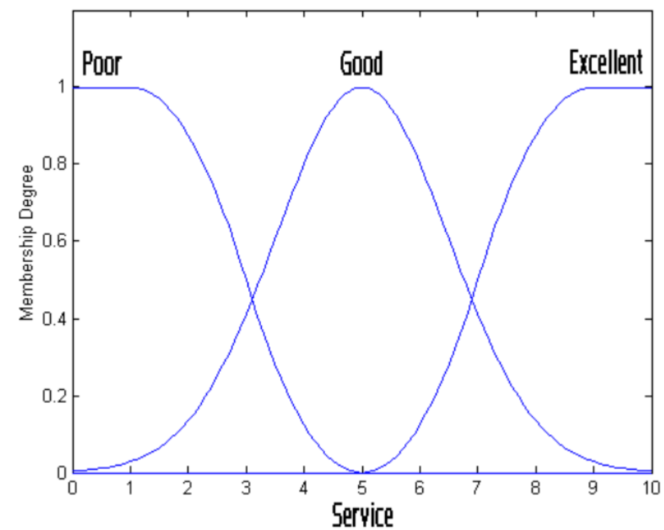
Fuzzy process flow (fuzzy inference process)



- **Fuzzifier** converts a clear input (crisp) to a fuzzy value based on gradual membership function.
- **Intelligence**
  - applies the **fuzzy rules** to infer fuzzy conclusions from fuzzy facts
  - assigns fuzzy sets to **outputs**, determined by degree of support for rules
  - **aggregates** the outputs of fuzzy rules based on classical **fuzzy operators**
- **Defuzzifier** converts the membership function of fuzzy output values to crisp output values using Centre of Gravity or Mean of Maxima methods

Fuzzy Rules (FR): Construction & Examples

- FR - also have **antecedents** and **consequents**
- Both deal with partial truths
  - Antecedents **match** fuzzy sets
  - E.g. **Restaurant tipping example**, antecedent **variables** are (quality of service, quality of food)
  - Consequents **assign** fuzzy sets
    - consequent **variable** is Tip
      - Service (range 0-10) can be:
        - *Poor*
        - *Good*
        - *Excellent*

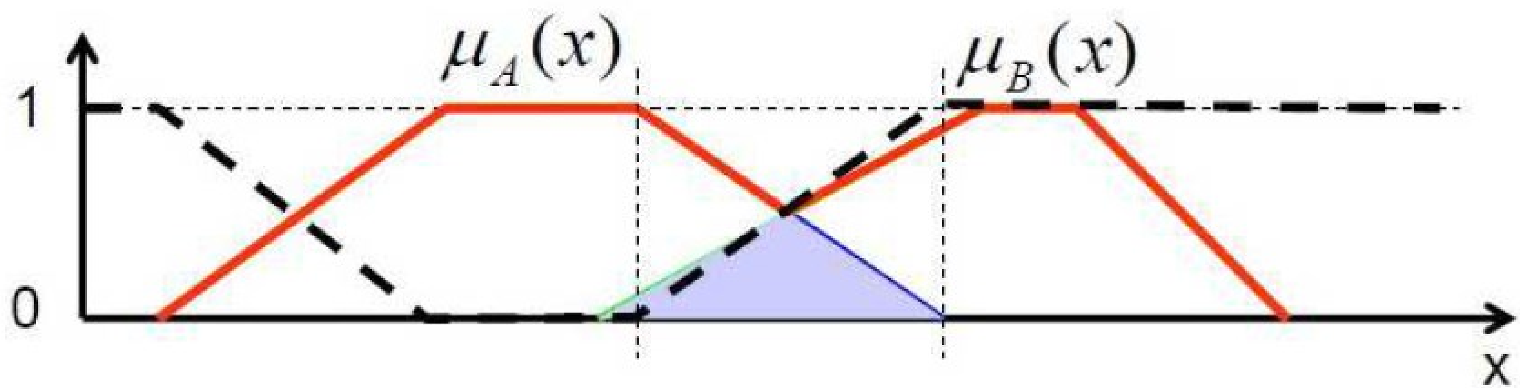


- Fuzzy rules can have **weightings**
  - usually in [0, 1]
  - based on importance of each rule

Fuzzy Rules: Operations

- **Classical Fuzzy Operators: Min/Max Norm**

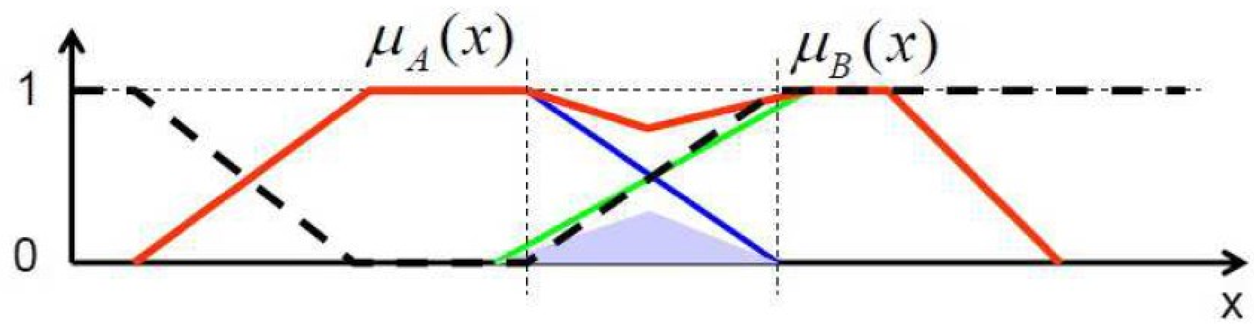
  - Conjunction: $\mu_{A \wedge B}(x) := \min\{\mu_A(x), \mu_B(x)\}$
  - Disjunction: $\mu_{A \vee B}(x) := \max\{\mu_A(x), \mu_B(x)\}$
  - Negation: $\mu_{\neg A}(x) := 1 - \mu_A(x)$



Fuzzy Rules: Operations

- **Classical Fuzzy Operators: Product / Bounded-Sum**

  - Conjunction: $\mu_{A\wedge B}(x) := \mu_A(x)\cdot\mu_B(x)$
  - Disjunction: $\mu_{A\vee B}(x) := \mu_A(x) + \mu_B(x) - \mu_A(x)\cdot\mu_B(x)$
  - Negation: $\mu_{\neg A}(x) := 1 - \mu_A(x)$



$$\mu_{A\to B}(x,y) = \max\big((1-\mu_A(x)), \mu_B(y)\big)$$

Fuzzy Rules: 2 main categories of fuzzy intelligence

- **Mandami** inference systems (**1975**)
  - It is logical
  - It has wide dissemination
  - It works well with human input

  Rule: IF <Antecedent> THEN <Consequent>

  Antecedent: **Conjunction** of **fuzzy memberships**

  Consequent: **Fuzzy Set**

- **Sugeno** inference systems (Takagi, Sugeno & Kang, **1985**)
  - It has good computational efficiency
  - It is compatible with linear techniques
  - It functions well with adaptive and optimization techniques
  - It has guaranteed the consistency of the output volume
  - It lends itself well to mathematical analysis

  Rule: IF x is A and y is B THEN z = f(x, y) => Crisp Function

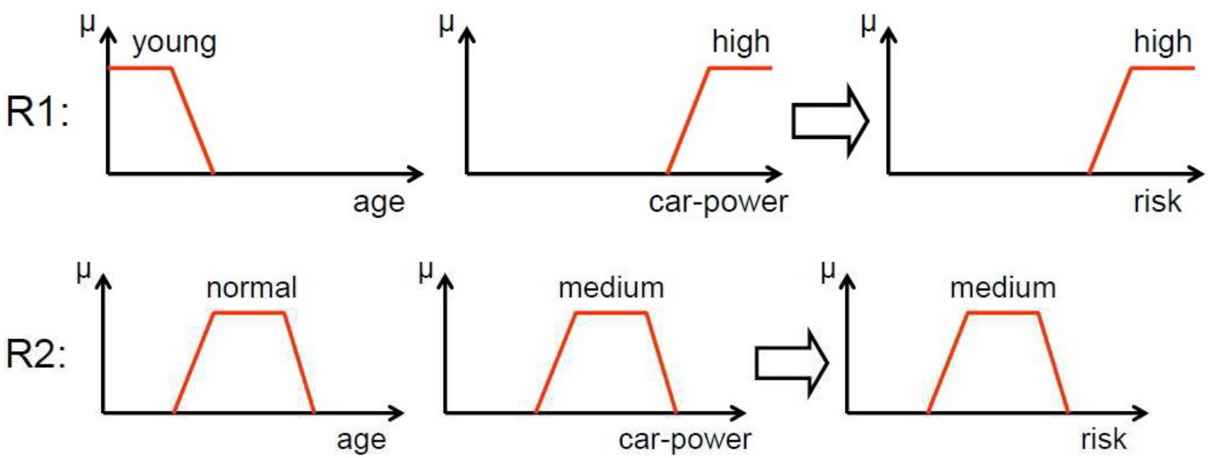
Fuzzy Rules: Inference Example (1)

- **Fuzzification of crisp inputs ⇒ Logical inference (via Min/Max - Norm) ⇒ Defuzzification**

Let's consider **two rules** expressed in fuzzy logic:

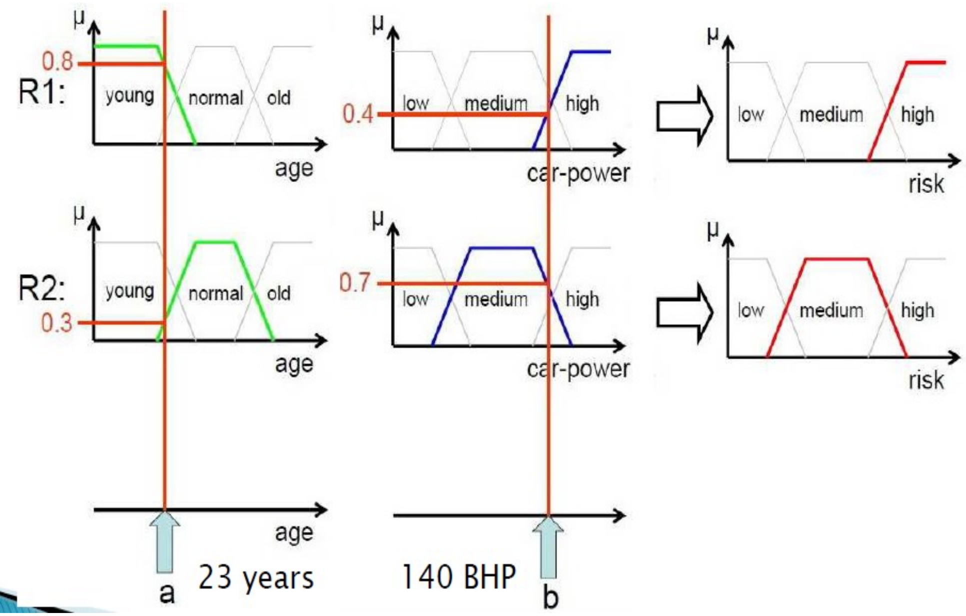
R1: IF age IS young AND car-power IS high THEN risk IS high

R2: IF age IS normal AND car-power IS medium THEN risk IS medium

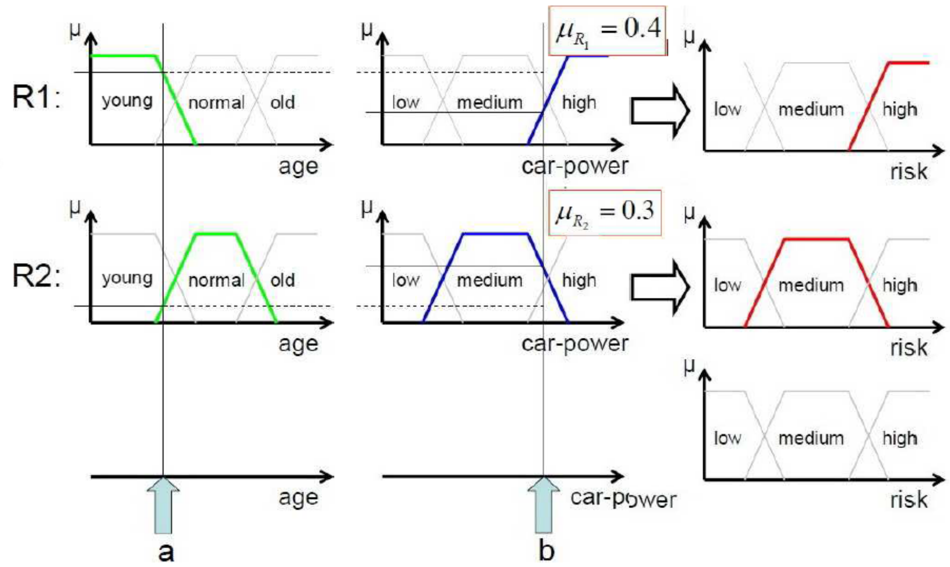


Fuzzy Rules: Inference Example (2)

- Step 1: **Fuzzification** of crisp inputs



- Step 2: **Inference** (Min/Max-Norm)



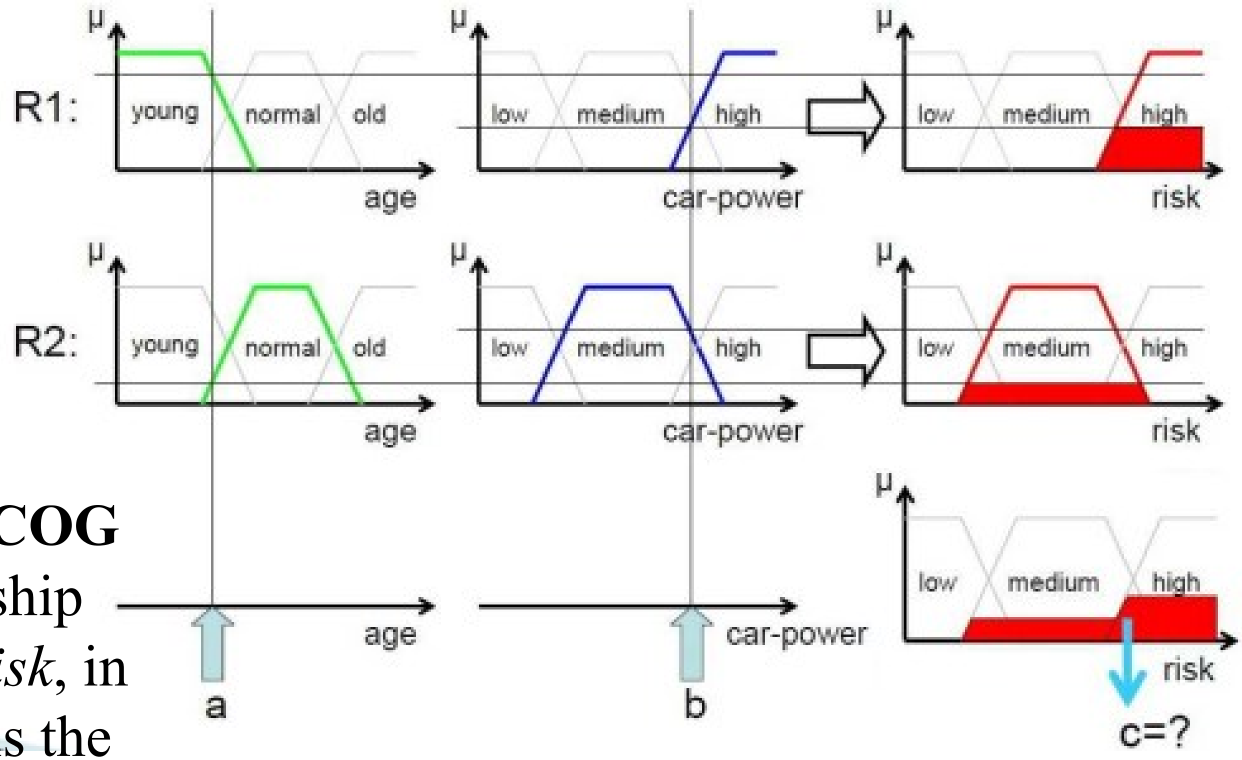
According the **R1** rule results the membership function related to **output** *risk=high*, according AND rule's, namely $\mu_{risk/high/R1} = \min\{\mu_{age/young=a}, \mu_{car-power/high=b}\}$

According the **R2** rule results the membership function related to output *risk=medium*, according to the rule $\mu_{risk/medium/R2} = \min\{\mu_{age/normal=a}, \mu_{car-power/medium=b}\}$

Consequently, we are **superpositioning** the $\mu_{risk/high/R1}$ with $\mu_{risk/medium/R2}$ Then, **calculate** the **center of gravity for the membership function resulted by superposition**.

Fuzzy Rules: Inference Example (3)

- Step 3: **Defuzzification**



Center of Gravity - COG of **objective** membership function (objective *risk*, in this case) where $\mu(x)$ is the **membership function** resulted after the processes of fuzzy logical inferences.

$$COG = \frac{\int x \cdot \mu(x)dx}{\int \mu(x)dx} \qquad COG_{approx} = \frac{\sum_{j=0}^{\max\_x} x_j \cdot \mu_j(x_j)}{\sum_{j=0}^{\max\_x} \mu_j(x_j)}$$

Fuzzy Inference System: Summary

1. Determining a set of fuzzy rules
2. Fuzzifying the inputs using the input membership functions
3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength
4. Finding the consequence of the rule by combining the rule strength and the output membership function (Mamdani FLS)
5. Combining the consequences to get an output distribution
6. Defuzzifying the output distribution

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

MAPIoT

Automation rules for first level automation type

RULE: rule_number

CF: certain factor

PRIO: priority

IF: premise

THEN: consequence

DESCRIPTION:

where

<premise>::=<logical _expression>

<logical_expression>::=(<variable_identifier> <relational_operator> <value>) AND/OR/NOT … AND/OR/NOT (<variable_identifier> <relational_operator> <value>),

and

<consequence>::=(<variable_identifier>=<value>)AND/OR … AND/OR <variable_identifier>=<value>).

1_Rule:

IF (Biomass = small) **AND** ((alcohol=const.) **OR** (alcohol=small)) **AND** (substrate concentration=const.) **AND** (last_phase=latent_phase)

THEN (new_phase=latent_phase)

DESCRIPTION: identify the latent phase in biomass developing

Automation rules – C# implementation

```
if (biomass[i] < 0.2 && alcohol[i] < (minAlcohol + 1) && ((i
   > 0) && (isConstant(substrate[i], substrate[i-1], 0.5)) ||
   substrate[i] > 205) && lastPhase == "latent")
{
    predictedAlcoholList[i].tooltip = "Latent phase: \n";
    lastPhase = "latent";
}
else if (biomass[i] > 0.2 &&  isIncreasing(alcohol[i],
    alcohol[i-1], 0.5)
    && substrate[i] > 20 && (lastPhase == "latent" ||
    lastPhase =="exponential"))
{
    predictedAlcoholList[i].tooltip = "Exponential growth
    phase";
    lastPhase = "exponential";
}
```

- https://de.mathworks.com/products/fuzzy-logic.html
- http://jfuzzylogic.sourceforge.net/html/example_fcl.html

Iceland Liechtenstein Norway grants

UNIVERSITATEA LUCIAN BLAGA DIN SIBIU

USN University of South-Eastern Norway

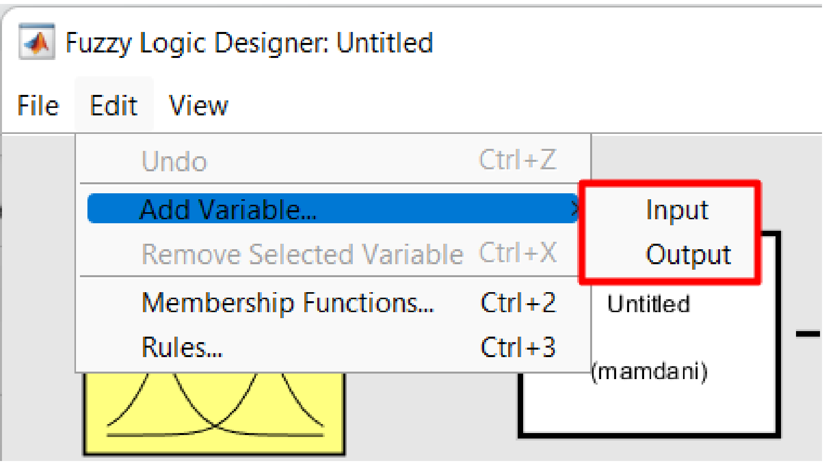
MAPIoT

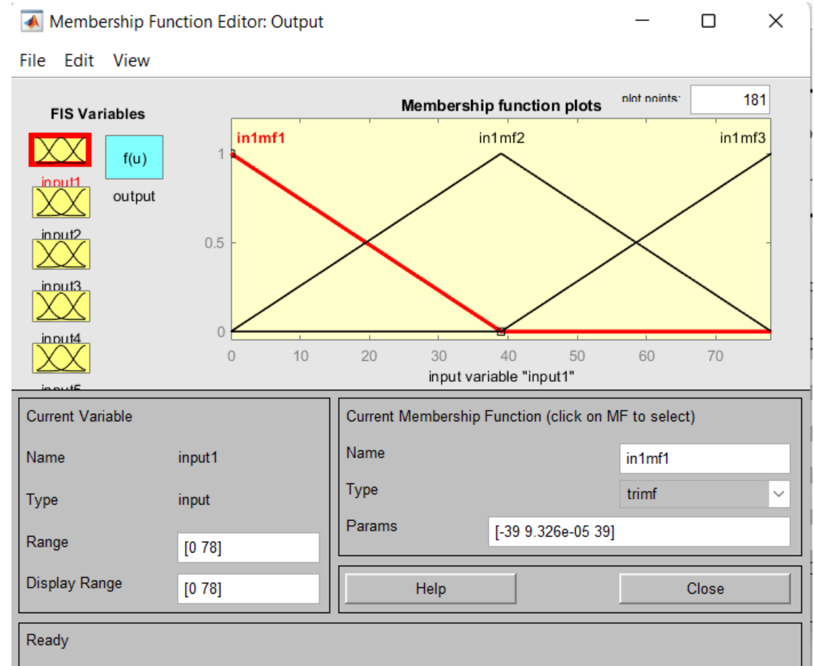
Automation Rules

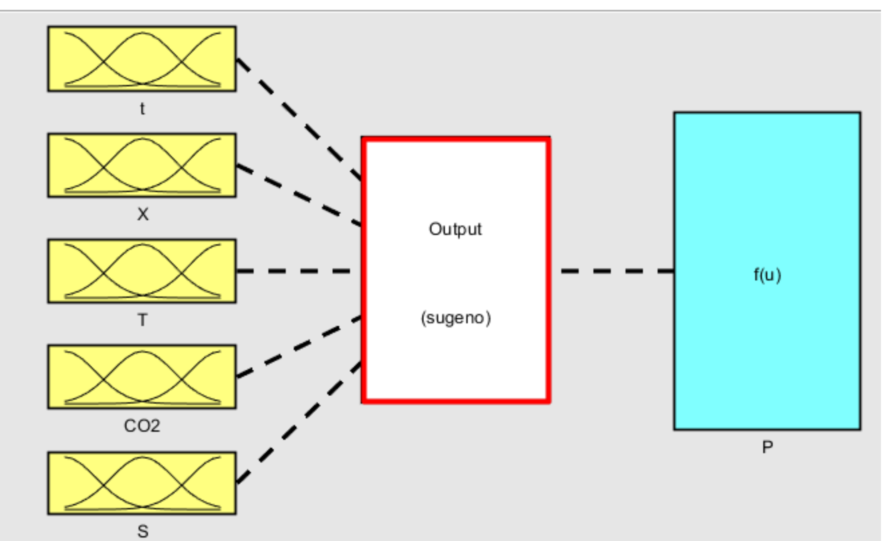
Fermentation phase	Latent	Exponential growth	Decay
Number of examples	17	36	7

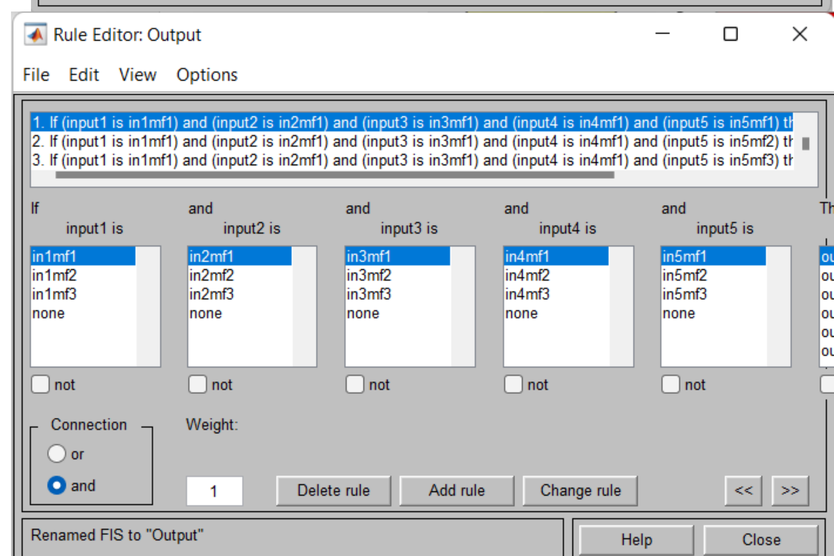
Time	Predicted Alcohol	Expected Alcohol	Alcohol Phase	Predicted Substrate	Expected Alcohol	Alcohol Phase
0	0.5924	0.2	latent	210.9708	210	latent
5	0.6736	0.2	latent	210.8072	210	latent
…						
49	3.4505	3.1095	exp. growth	196.1738	197.5316	exp. growth
50	4.2833	4.6204	exp. growth	182.2209	188.6707	exp. growth
52	5.7213	6.2040	exp. growth	178.8094	180.5961	exp. growth
…						
191	79.2037	70.016	decay	15.7378	10.3751	decay
212	79.1773	70.016	decay	13.1312	8.1352	decay

MATLAB Neuro-fuzzy designer









Implementation of irrigation system fuzzy logic (1)

Used Simpful python library for implementation

```
from simpful import *
```

Implemented a class with one method to calculate the irrigation time based on the environment variables.

FLC
- fs1: FuzzySystem - fs2: FuzzySystem - fs3: FuzzySystem
+ solve(double, double, double, double): double

1. Define fuzzy variables (name, membership functions, universe of discourse) and rules as constants.

```
IRRIGATION_TIME_SET = {
    "irrigation_time": [
        {
            "none": [0, 0, 2]
        }, {
            "short": [0, 2, 4]
        } ...
    ],
    "uod": [0, 10]
}
```

```
FLC1_RULES = [
    "IF (soil_moisture IS dry) AND (air_temperature IS cold) THEN (irrigation_time IS long)",
    "IF (soil_moisture IS dry) AND (air_temperature IS moderate) THEN (irrigation_time IS very_long)",
    "IF (soil_moisture IS dry) AND (air_temperature IS hot) THEN (irrigation_time IS very_long)",
    "IF (soil_moisture IS moderate) AND (air_temperature IS cold) THEN (irrigation_time IS short)",
    "IF (soil_moisture IS moderate) AND (air_temperature IS moderate) THEN (irrigation_time IS medium)",
    "IF (soil_moisture IS moderate) AND (air_temperature IS hot) THEN (irrigation_time IS medium)",
    "IF (soil_moisture IS wet) THEN (irrigation_time IS none)"
]
```

2. Initialize the fuzzy systems

```
self.fs1 = FuzzySystem()
self.fs2 = FuzzySystem()
self.fs3 = FuzzySystem()
```

Implementation of irrigation system fuzzy logic (2)

3.1. Create fuzzy sets with terms and membership functions of each variable

```
set = FuzzySet(function = Triangular_MF(a, b, c), term = name)
set = FuzzySet(function = Trapezoidal_MF(a, b, c, d), term = name)
…
fuzzy_sets.append(set)
```

! Where *a, b, c, d* are membership function points of a term from a configuration set. (E.g. *a*=0, *b*=0, *c*=2, *name*="none"; for Irrigation Time Set)

3.2. Add a variable into the fuzzy system

```
fs.add_linguistic_variable(set_name, LinguisticVariable(fuzzy_sets, universe_of_discourse = uod))
```

! Where *set_name* is the name of the variable, *fuzzy_sets* is defined in step 2.2 and *uod* is values interval.

3.3. Repeat with the other variables

4. Add the rules

```
self.fs1.add_rules(FLC1_RULES)
```

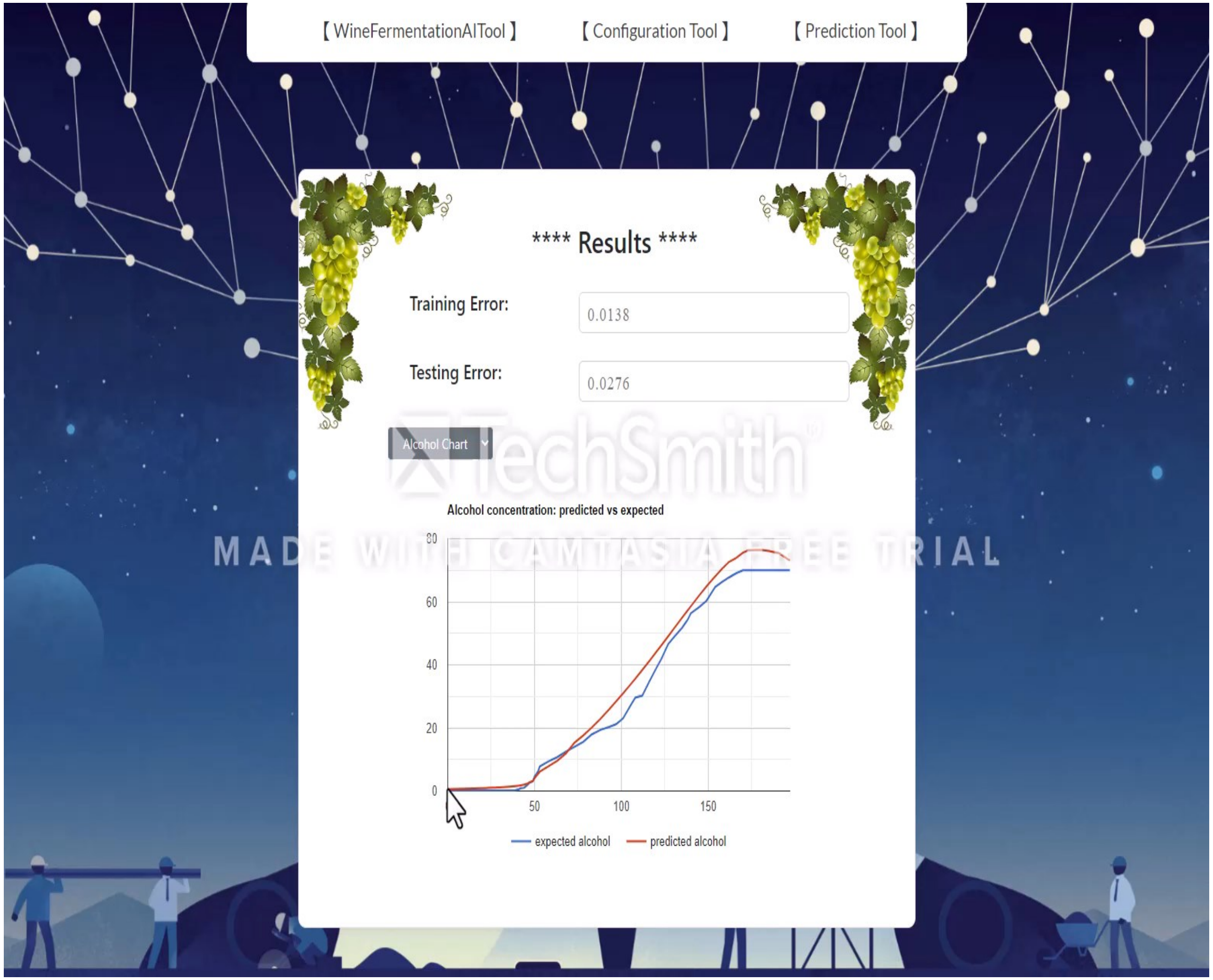
5. Calculate the output variable (irrigation time)

```
self.fs1.set_variable("soil_moisture", soil_moisture)
self.fs1.set_variable("air_temperature", air_temperature)
result = self.fs1.Mamdani_inference(["irrigation_time"])
irrigation_time = result["irrigation_time"]
```

! Where *soil_moisture* and *air_temperature* are numerical values which need to be fuzzified before calculating the irrigation time.

Full source code: https://github.com/pdany1116/is-iot-sink

AUTOMATION RULES



References

- Florea, A., Sipos, A., & Stoisor, M. C. (2022). Applying AI Tools for Modeling, Predicting and Managing the White Wine Fermentation Process. Fermentation 2022, 8, 137.
- Sipos, A., Florea, A., Arsin, M., & Fiore, U. (2020). Using Neural Networks to Obtain Indirect Information about the State Variables in an Alcoholic Fermentation Process. Processes 2021, 9, 74.
- Gellert, A., Florea, A., Fiore, U., Zanetti, P., & Vintan, L. (2019). Performance and energy optimisation in CPUs through fuzzy knowledge representation. Information Sciences, 476, 375-391.
- Berntzen, L., Florea, A., Molder, C., & Bouhmala, N. (2019). A strategy for drone traffic planning dynamic flight-paths for drones in smart cities. In SMART 2019, The Eighth International Conference on Smart Cities, Systems, Devices and Technologies.
- Spolaor S., Fuchs C., Cazzaniga P., Kaymak U., Besozzi D., Nobile M.S.: Simpful: a user-friendly Python library for fuzzy logic, International Journal of Computational Intelligence Systems, 13(1):1687–1698, 2020 DOI:10.2991/ijcis.d.201012.002