Lecture#0

Course Introduction

CENG 632- Computational Intelligence, 2024-2025, Spring
Assist. Prof. Dr. Osman GÖKALP

Course Instructor

Assist. Prof. Dr. Osman GÖKALP
Department of Computer Engineering, IZTECH
Room 127

E-mail: osmangokalp@iyte.edu.tr

- Class: Thursday, 13:30-16:15
- Office Hours*: Tuesday, 10:00-11:00
  Wednesday, 14:00-15:00

* If you would like to meet at a time other than the office hours, please send me an e-mail.

Course Website

TEAMS: **Lecture notes, Assignments, Announcements**

Enrollment code: zcq5ca0

Teams enrollment codes have been announced at https://ceng.iyte.edu.tr/2024-2025-spring-term-microsoft-teams-course-codes/

The students are expected to **regularly check** the course's Teams class for announcements.

Textbooks

There is no single textbook for the course. Interested students may consider using the following books.

- ***"Computational Intelligence: A Methodological Introduction"**, R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, M. Steinbrecher, 3rd edition, Springer.*
- ***"Computational Intelligence: An Introduction"**, A.P. Engelbrecht, 2nd edition, Wiley.*
- ***"Introduction to Evolutionary Computing"**, A.E. Eiben and J.E. Smith, 2nd edition, Springer.*
- ***"Essentials of Metaheuristics"**, S. Luke, 2nd edition, Lulu.*

Course Outline (tentative)

Lecture#0: Introduction to Computational Intelligence

Lecture#1: Intro. to Artificial Neural Networks, Threshold Logic Units (Perceptrons)

Lecture#2: General Neural Networks

Lecture#3: Multi-layer Perceptrons-I

Lecture#4: Multi-layer Perceptrons-II

Lecture#5: Some variants of ANNs

Lecture#6: Introduction to Metaheuristics, Evolutionary Computation, and Swarm Intelligence

Lecture#7: Genetic Algorithms

Midterm Exam

Lecture#8: Differential Evolution

Lecture#9: Particle Swarm Optimization

Lecture#10: Ant Colony Optimization

Lecture#11: Student presentations

Lecture#12: Student presentations

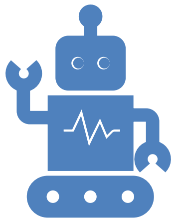
Final Exam

Grading Criteria (tentative)

• Midterm Exam	50%
• Term Project *	50%
• Project proposal	10%
• Project report in given format	20%
• Project presentation	20%

* You can work on a project in a group of **up to 3** people or individually.
Each group will **choose** its **own topic** based on their interests.

Introduction to Computational Intelligence



The Word 'Intelligence'

- Etymology
  - The word "**intelligence**" comes from the Latin "**intelligentia**", which derives from "**intelligere**", meaning "to **understand**" or "to **discern**".
  - Breaking it down:
    - "**inter-**" → "**between**"
    - "**legere**" → "to **choose**" or "to **read**"
  - Implies the ability to choose between options wisely.
  - In AI, "intelligence" retains this core idea—machines making decisions by "choosing between" different possibilities based on data and algorithms.

Artificial Intelligence

- Artificial intelligence (AI), in its broadest sense, is **intelligence** exhibited by **machines**, particularly **computer systems**.
- Field of **computer science**
- Methods and software that enable machines to **perceive** their **environment** and use learning and intelligence to **take actions** that **maximize** their chances of achieving defined **goals**.

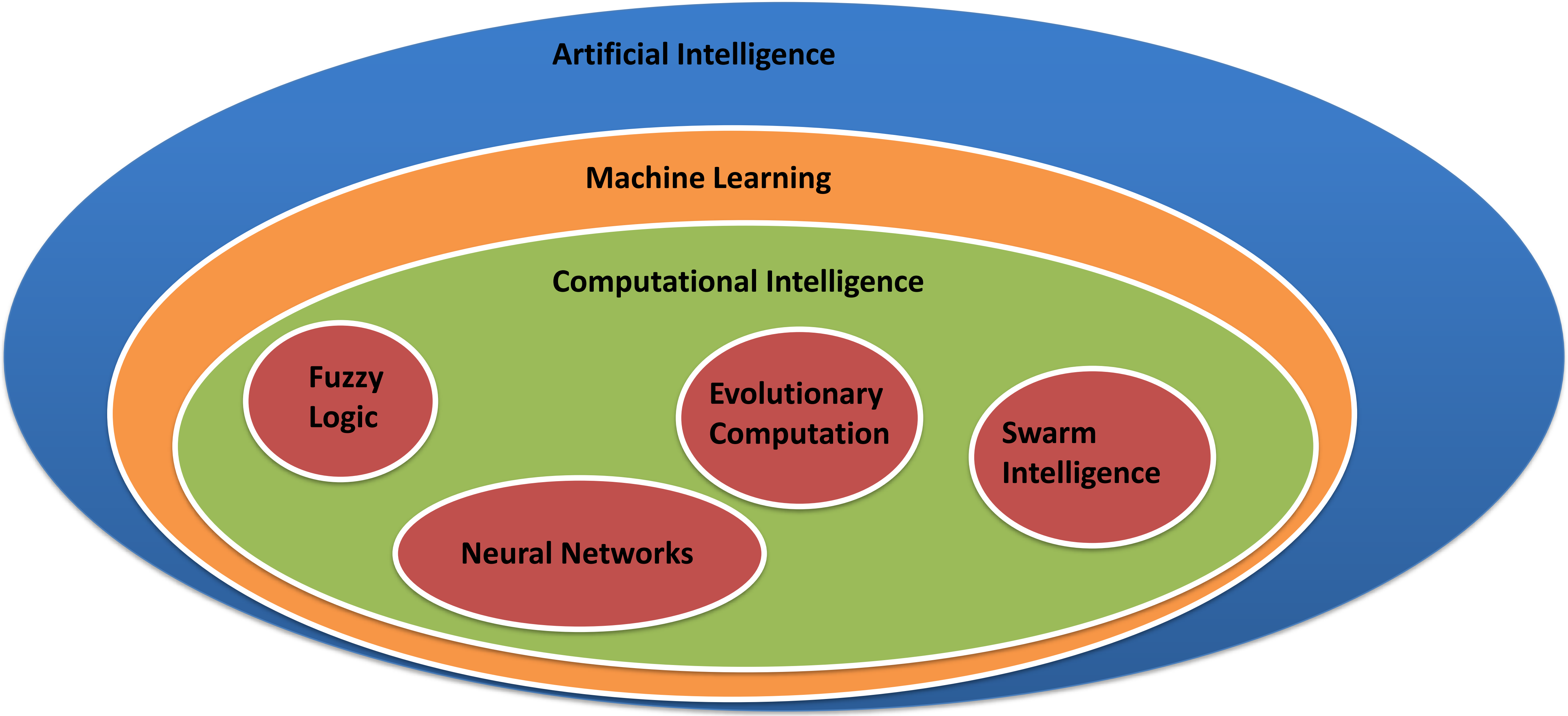
Source: https://en.wikipedia.org/wiki/Artificial_intelligence

What is Computational Intelligence?

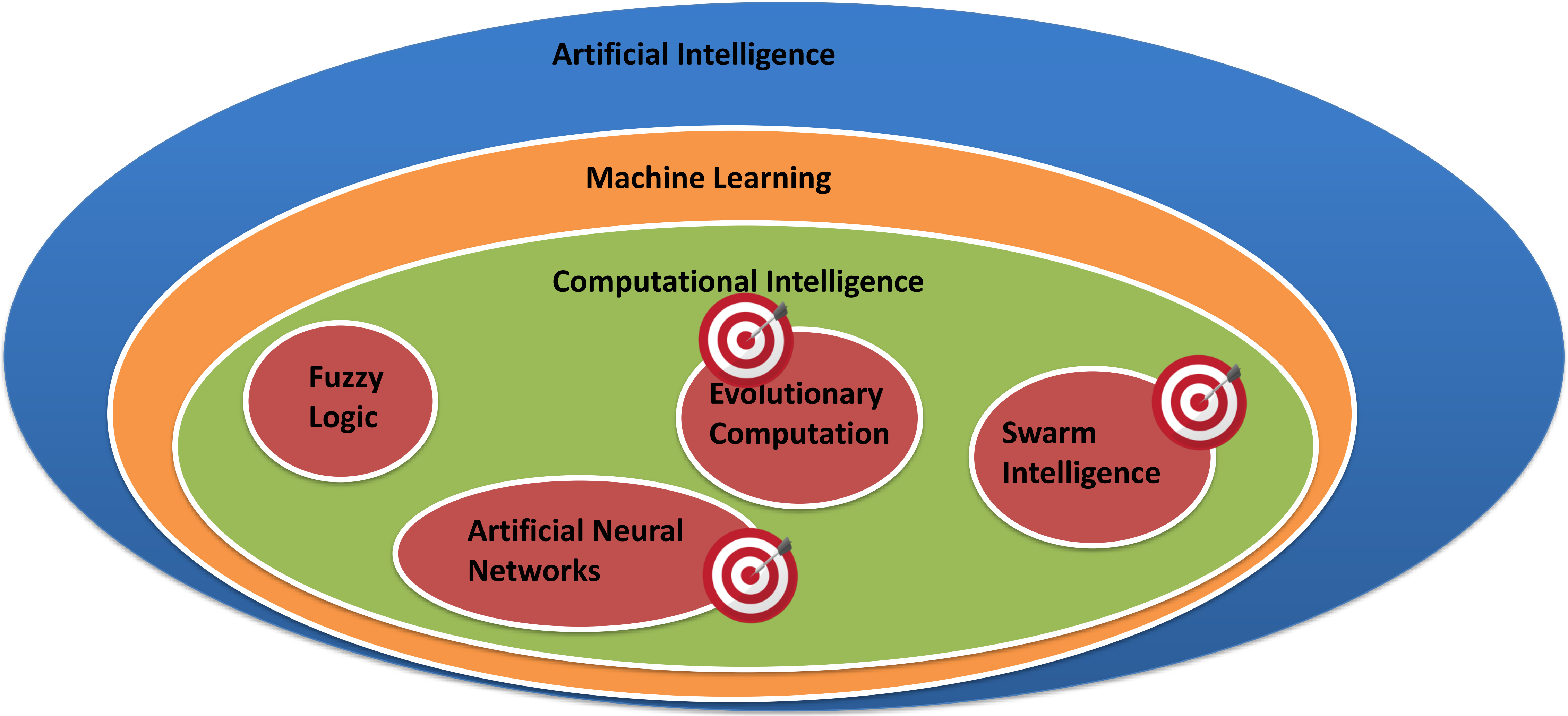
- <u>IEEE Computational Intelligence Society</u>:
  - Computational Intelligence (CI) is the theory, design, application and development of **biologically** and **linguistically motivated** computational paradigms.
  - Traditionally the three main pillars of CI have been **Neural Networks**, **Fuzzy Systems** and **Evolutionary Computation**.
- <u>Wikipedia</u>:
  - Computational intelligence (CI) refers to concepts, paradigms, algorithms and implementations of systems that are designed to **show "intelligent" behavior** in complex and changing **environments**.
  - Nature-analog or at least **nature-inspired** methods play a **key role** in this.

Source: https://cis.ieee.org/about/what-is-ci
https://en.wikipedia.org/wiki/Computational_intelligence

AI Methods and CI



CI Subjects We Will Focus on During Class



Introduction to Artificial Neural Networks

Artificial Neural Networks (ANNs)

- Computational models inspired by the human brain:
  - Algorithms that try to mimic the brain.
  - Massively parallel, distributed system, made up of simple processing units (neurons)
  - Synaptic connection strengths among neurons are used to store the acquired knowledge.
  - Knowledge is acquired by the network from its environment through a learning process

Source: http://web.cs.hacettepe.edu.tr/~nazli/courses/ai/ai7.ppt

History

- late-1800's - Neural Networks appear as an analogy to biological systems
- 1960's and 70's – Simple neural networks appear
  - Fall out of favor because the perceptron is not effective by itself, and there were no good algorithms for multilayer nets
- 1986 – Backpropagation algorithm appears
  - Neural Networks have a resurgence in popularity
  - More computationally expensive

Applications of ANNs

- ANNs have been widely used in various domains for:
  - Pattern recognition
  - Function approximation
  - Associative memory

Properties

- Inputs are flexible
  - any real values
  - Highly correlated or independent
- Target function may be discrete-valued, real-valued, or vectors of discrete or real values
  - Outputs are real numbers between 0 and 1
- Resistant to errors in the training data
- Long training time
- Fast evaluation
- The function produced can be difficult for humans to interpret

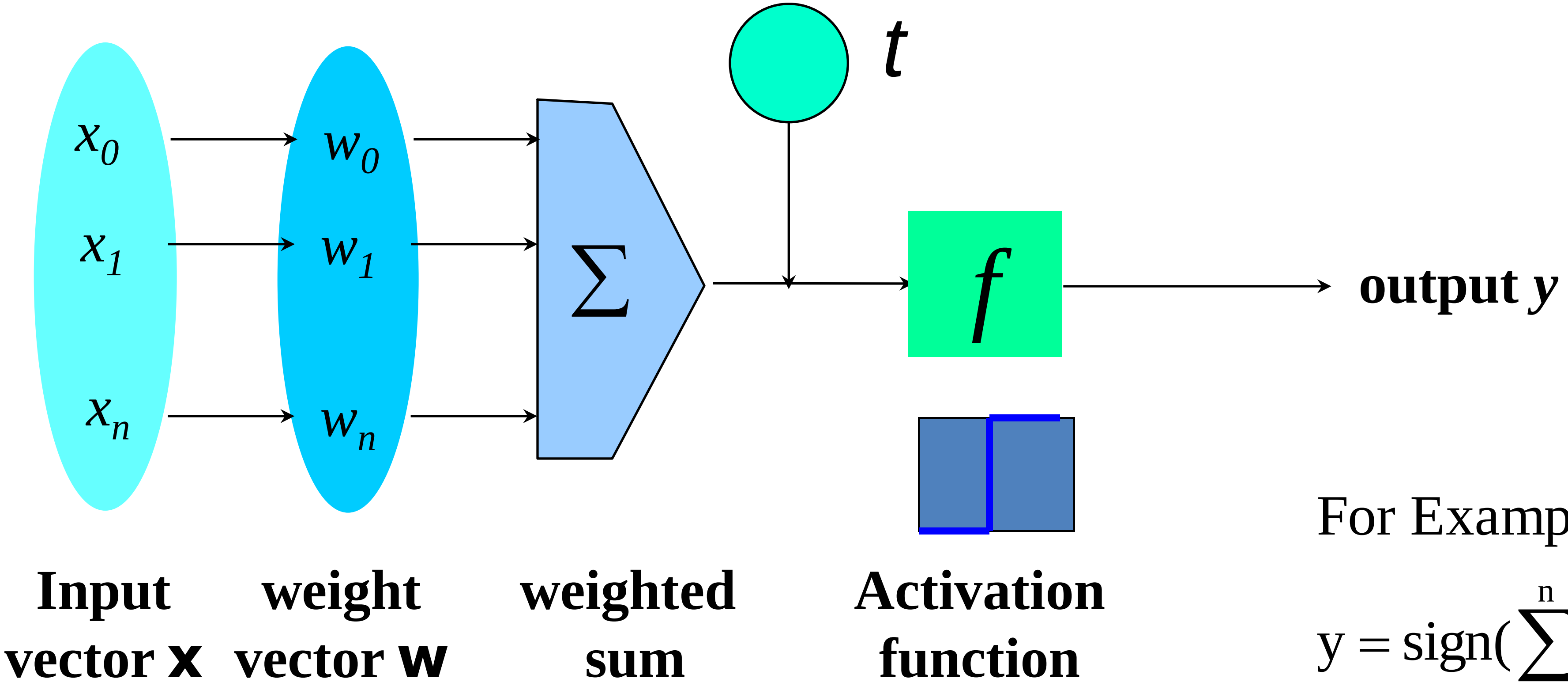
When to consider neural networks

- Input is high-dimensional discrete or raw-valued
- Output is discrete or real-valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of the result is not important

Examples:

- Speech recognition
- Image classification
- Financial prediction

A Neuron (= a perceptron)



For Example

$$y = \mathrm{sign}\left(\sum_{i=0}^{n} w_i x_i - t\right)$$

- The n-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Perceptron

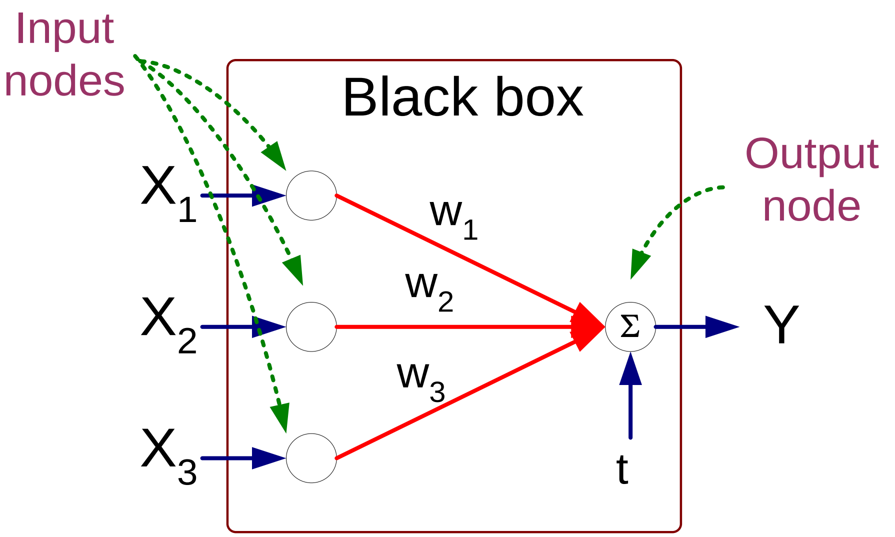
- **Basic unit** in a neural network
- **Linear** separator
- Parts
  - N inputs, $x_1 \dots x_n$
  - Weights for each input, $w_1 \dots w_n$
  - A bias input x_0 (constant) and associated weight w_0
  - Weighted sum of inputs, $y = w_0x_0 + w_1x_1 + \dots + w_nx_n$
  - A threshold function or activation function,
    - i.e 1 if $y > t$, -1 if $y <= t$

Artificial Neural Networks (ANN)

- Model is an a**ssembly** of inter-connected nodes and weighted links

- Output node sums up each of its input value according to the weights of its links

- Compare output node against some **threshold** t



Perceptron Model

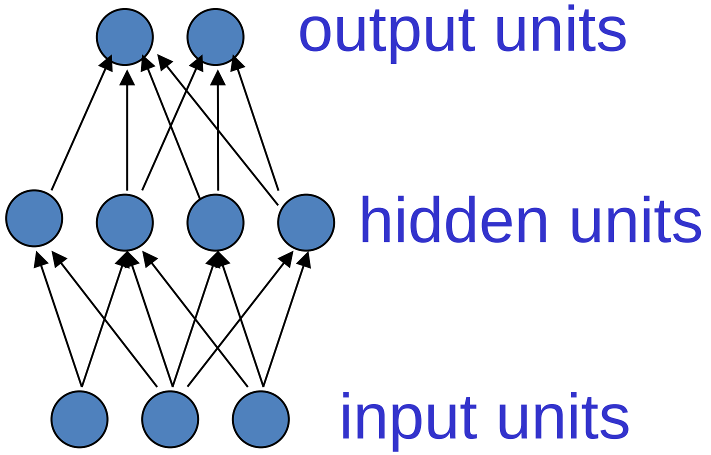
$$Y = I(\sum_i w_i x_i - t)$$

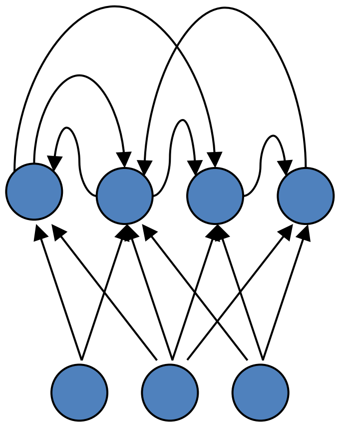
or

$$Y = sign(\sum_i w_i x_i - t)$$

Types of connectivity

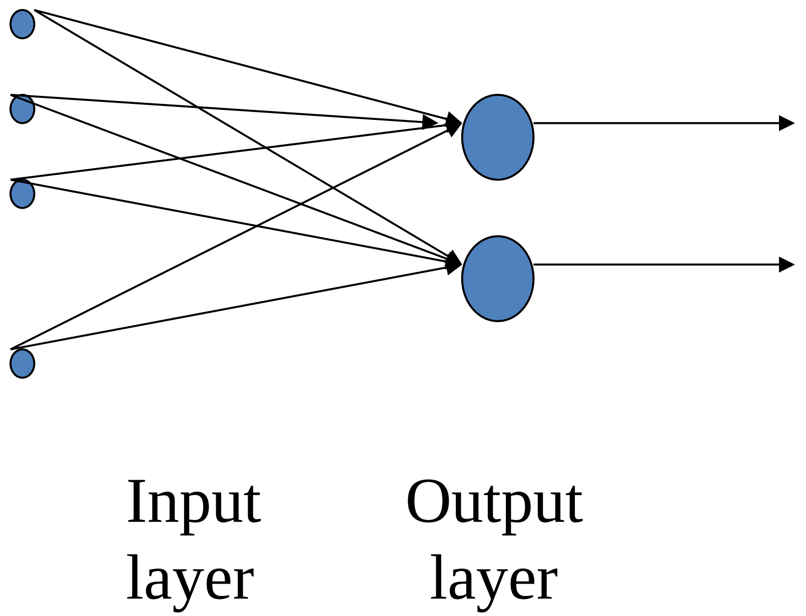
- Feedforward networks
  - These compute a series of transformations
  - Typically, the first layer is the input and the last layer is the output.
- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.





Different Network Topologies

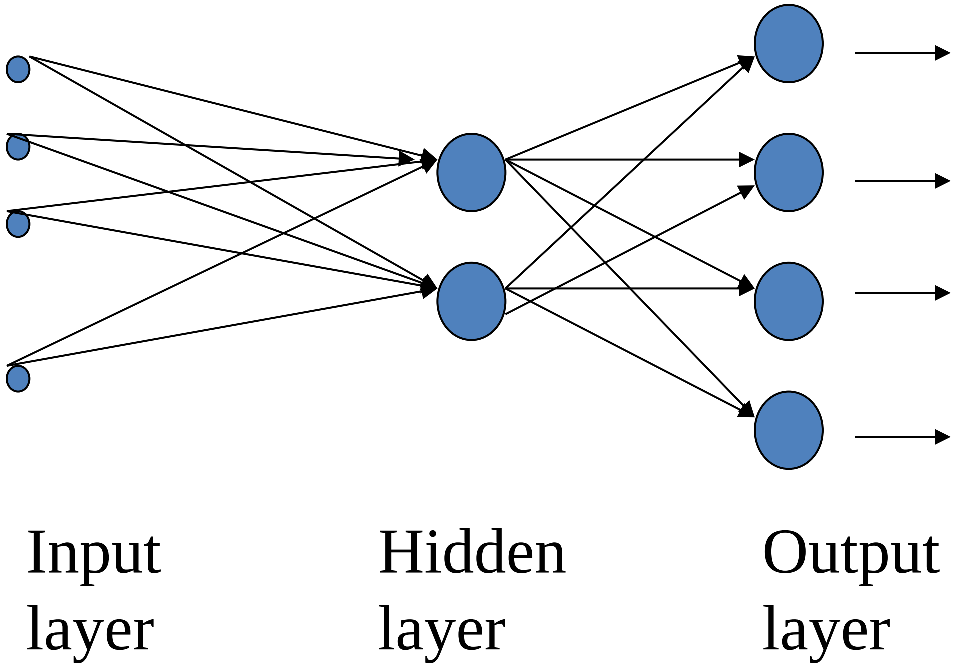
- Single layer feed-forward networks
  - Input layer projecting into the output layer



Single layer network

Different Network Topologies

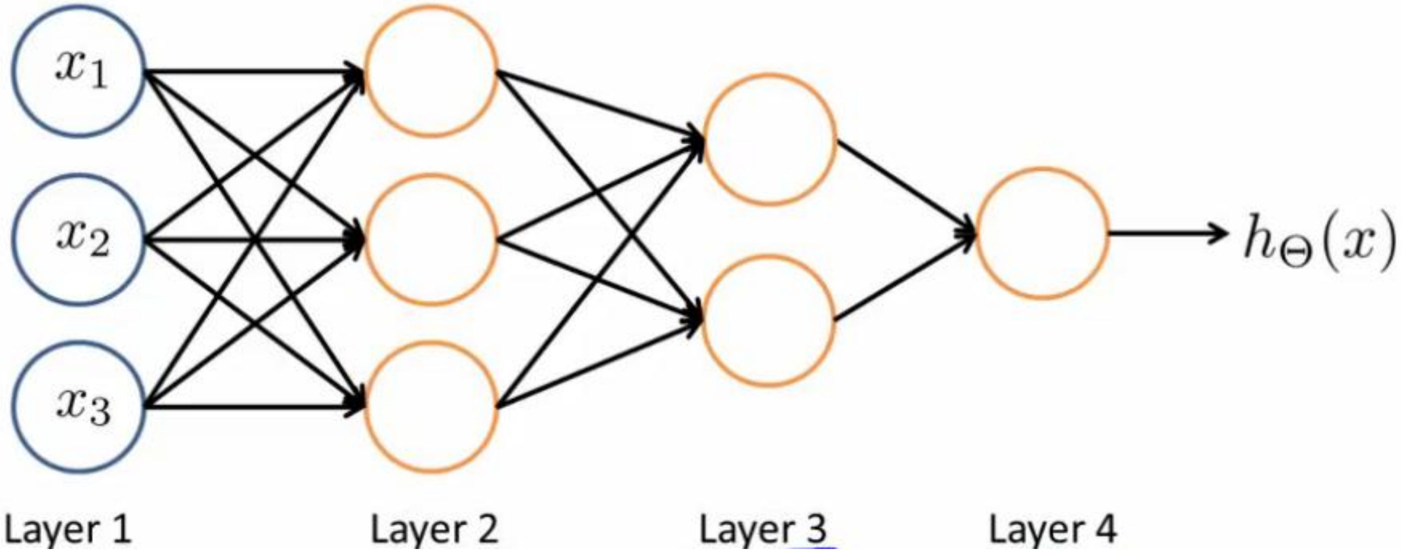
- Multi-layer feed-forward networks
  - One or more hidden layers. Input projects only from previous layers onto a layer.



2-layer or
1-hidden layer
fully connected
network

Different Network Topologies

- Multi-layer feed-forward networks



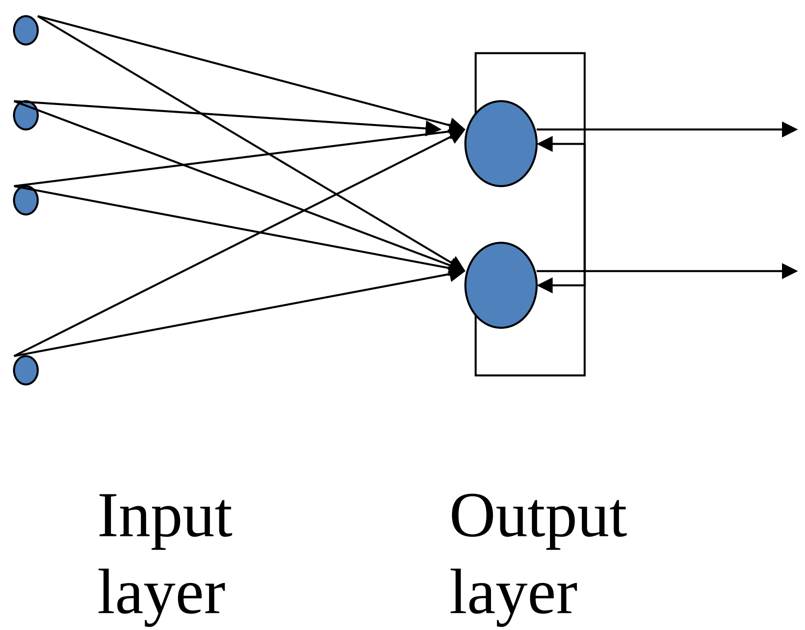
Input layer

Hidden layers

Output layer

Different Network Topologies

- Recurrent networks
  - A network with feedback, where some of its inputs are connected to some of its outputs (discrete time).



Recurrent network

Algorithm for learning ANN

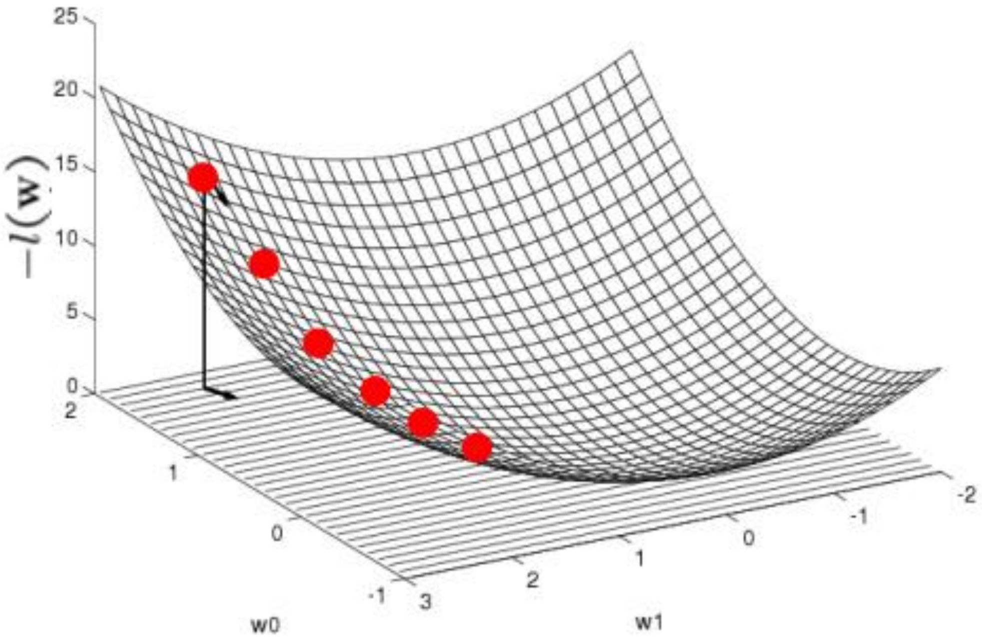
- Initialize the weights (w_0, w_1, …, w_k)

- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Error function: $$E = \sum_i \left[Y_i - f(w_i, X_i)\right]^2$$
  - Find the weights w_i's that minimize the above error function
    - e.g., gradient descent, backpropagation algorithm

Optimizing concave/convex function

- Maximum of a concave function = minimum of a convex function

Gradient ascent (concave) / Gradient descent (convex)



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_d}]'$$

Update rule: Learning rate, η>0

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

Gradient ascent rule

How a Multi-Layer Neural Network Learn? Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

How A Multi-Layer Neural Network Predict? Feed Forward

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel

Introduction to Evolutionary Computation

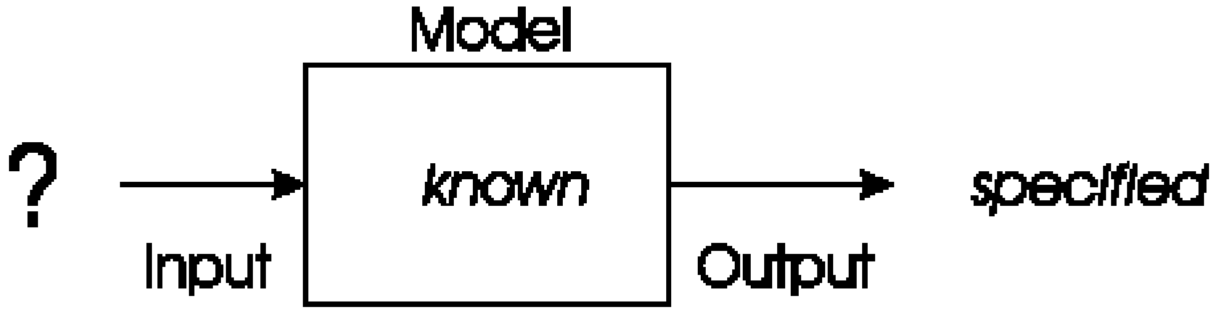
Evolutionary Computation

- Evolutionary computation is a family of algorithms for **global optimization inspired by** biological **evolution**, and the **subfield of artificial intelligence** and soft computing studying these algorithms.

Source: https://en.wikipedia.org/wiki/Evolutionary_computation

“Black box” model: Optimisation

- Model and desired output is known, task is to find inputs



- Examples:
  - Time tables for university, call center, or hospital
  - Design specifications
  - Traveling salesman problem (TSP)
  - Eight-queens problem, etc.

Historical perspective

- 1948, Turing:
  proposes "genetical or evolutionary search"
- 1962, Bremermann:
  optimization through evolution and recombination
- 1964, Rechenberg:
  introduces evolution strategies
- 1965, L. Fogel, Owens and Walsh:
  introduce evolutionary programming
- 1975, Holland:
  introduces genetic algorithms
- 1992, Koza:
  introduces genetic programming

… and many modern variants.

Biological Origins: Darwinian Evolution: Survival of the fittest

- All environments have **finite resources**

  (i.e., can only support a limited number of individuals)
- Life forms have basic instinct/ lifecycles geared **towards reproduction**
- Therefore some kind of **selection** is inevitable
- Those individuals that **compete** for the resources most effectively have increased chance of reproduction

Darwinian Evolution: Diversity drives change

- Phenotypic traits:
  - **Behaviour / physical** differences that affect response to environment
  - Partly determined by inheritance, partly by factors during development
  - Unique to each individual, partly as a result of random changes
- If phenotypic traits:
  - Lead to higher chances of reproduction
  - Can be inherited

then they will tend to increase in subsequent generations, leading to new combinations of traits …

Darwinian Evolution: Summary

- Population consists of diverse set of individuals
- Combinations of traits that are better adapted tend to increase representation in population

Individuals are "units of selection"

- Variations occur through random changes yielding constant source of diversity, coupled with selection means that:

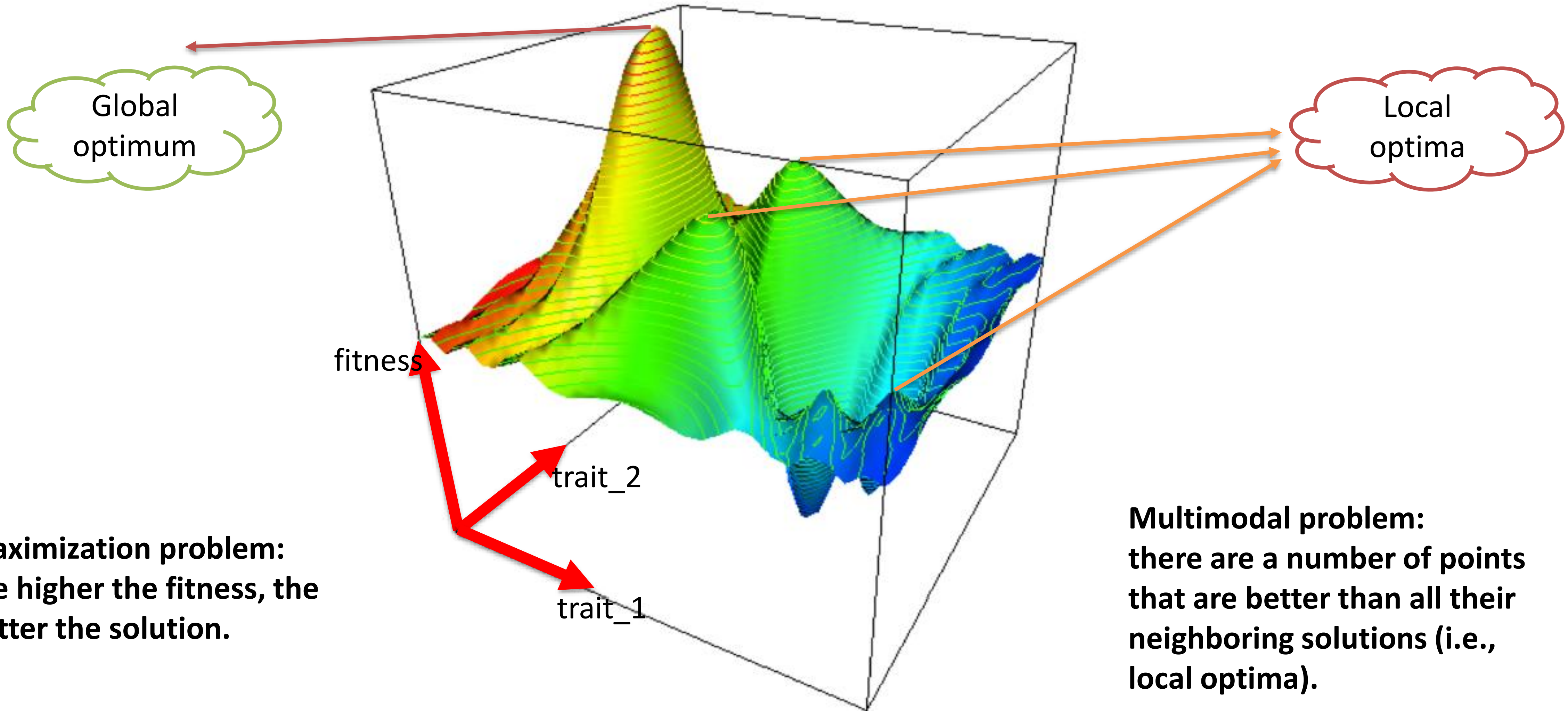
Population is the "unit of evolution"

- Note the absence of "guiding force"

Adaptive landscape metaphor (Wright, 1932)

- Can envisage population with ***n*** **traits** as existing in a ***n+1*****-dimensional space** (landscape) with height corresponding to **fitness**
- Each different individual (phenotype) represents a **single point** on the landscape
- Population is therefore a “cloud” of points, moving on the landscape over time as it evolves – adaptation

Adaptive landscape metaphor (Wright, 1932)



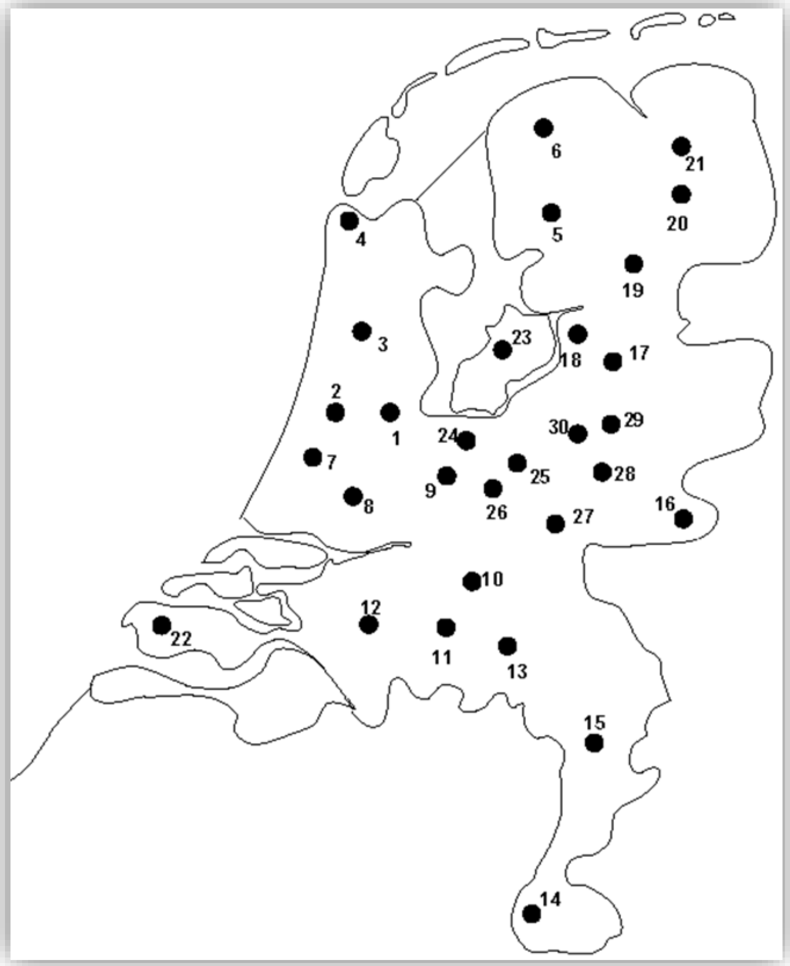
Maximization problem:
the higher the fitness, the better the solution.

Multimodal problem:
there are a number of points that are better than all their neighboring solutions (i.e., local optima).

An example optimization problem: Travelling Salesman Problem (TSP)

- 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:

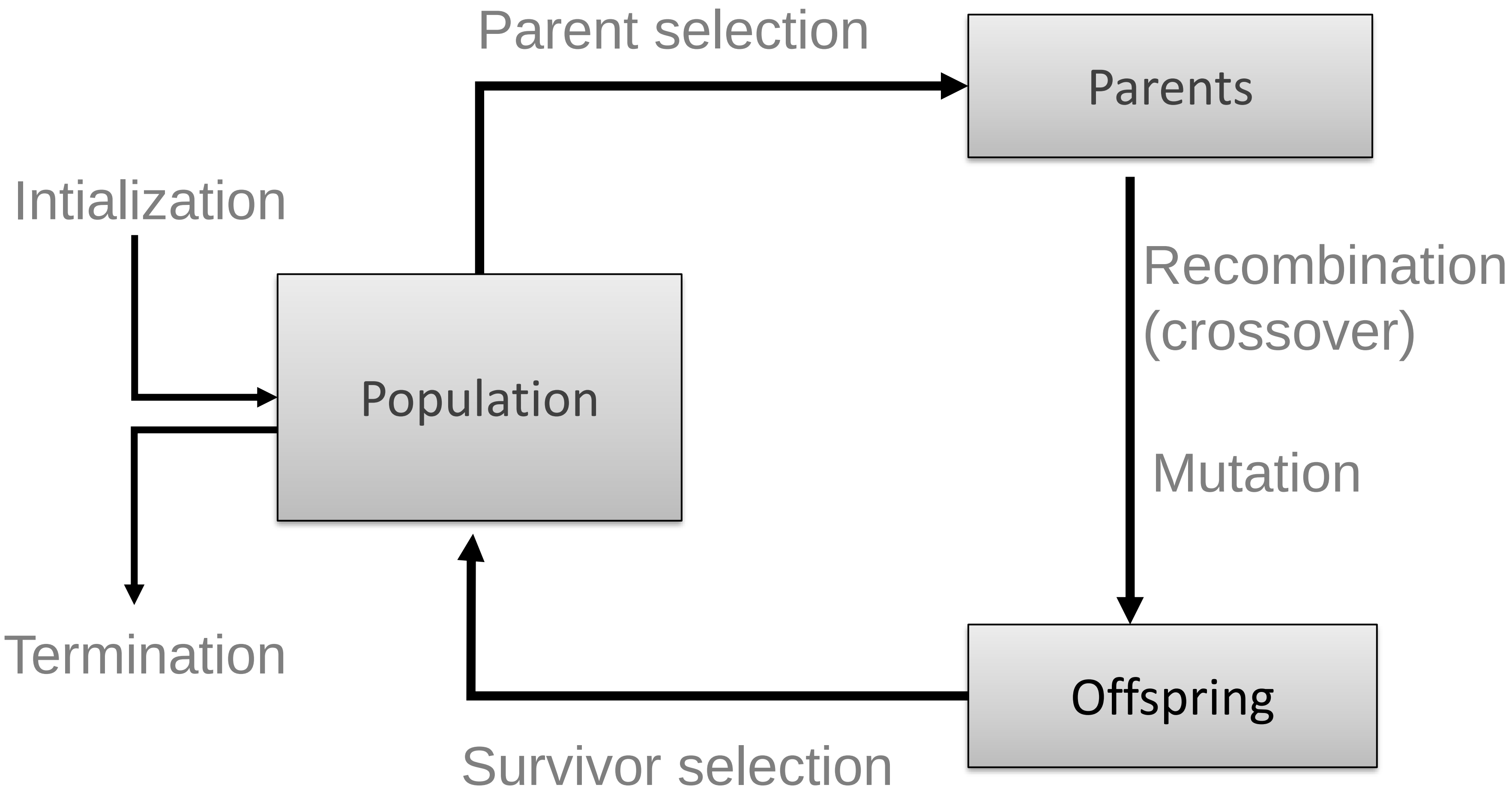
for 30 cities there are 30! ≈ 10^{32} possible tours



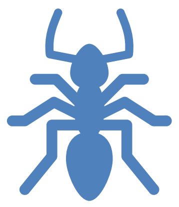
Methapors and Problem Solving Components

- Individual is a **solution** candidate (e.g., A TSP tour)
  - May be initialized randomly or by a heuristic rule
- Population is a **set** of solutions (e.g., array of TSP tours)
- Fitness is the quality of a given solution (e.g., tour length of a TSP tour)
- Parent is a **solution** that is used for **producing** new solutions.
- Recombination (crossover) is a **method** that combines parents' features into new solution(s).
- Offspring (Child) is a **solution** that is produced after recombination.
- Selection is a **method** that selects '**good**' solutions according to their **fitness**.
- Mutation is a **random** change on solutions.

General scheme of Evolutionary Agorithms (EAs)



Swarm Intelligence



Swarm Intelligence

Swarms, flocks, etc… often exhibit the following rather interesting properties:

- Individuals of the swarm are incapable of X*, or could do X with only low probability.
- However, the *swarm* as a unit is able to do X, with high probability.

The ability to do X is an *emergent property* of the swarm.

*X: an intelligent behaviour (e.g., find the shortest path)

Source: https://empslocal.ex.ac.uk/people/staff/reverson/sr/oldECM3412/NICLecture9.ppt

Swarm Intelligence II

- Each element of the swarm has its own simple behaviour, and a set of rules for interacting with its fellows, and with the environment.
- Every element is the same – there is no central controller.
- However, X emerges as a result of these local interactions.
- E.g. ants finding food, termites building mounds, jellyfish.

Swarm Algorithms

Inspiration from swarm intelligence has led to some highly successful optimisation algorithms. We will look at:

- Ant Colony (-based) Optimisation – a way to solve optimisation problems based on behavior of ants.
- Particle Swarm Optimisation — a different way to solve optimisation problems, based on the swarming behaviour of several kinds of organisms.

Emergent Problem Solving in *Lasius Niger* ants,

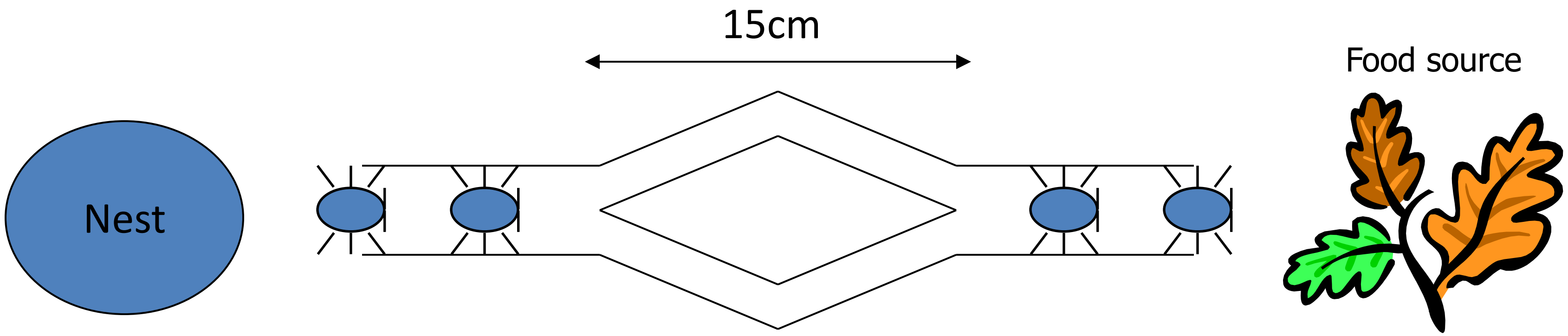
For *Lasius Niger* ants, [Franks, 89] observed:

- regulation of nest temperature within 1 degree celsius range;
- forming bridges;
- raiding specific areas for food;
- building and protecting nest;
- sorting brood and food items;
- cooperating in carrying large items;
- emigration of a colony;
- **finding shortest route from nest to food source**;
- preferentially **exploiting the richest food source available**.

These are swarm behaviours – beyond what any individual can do.

Real Ant Experiments

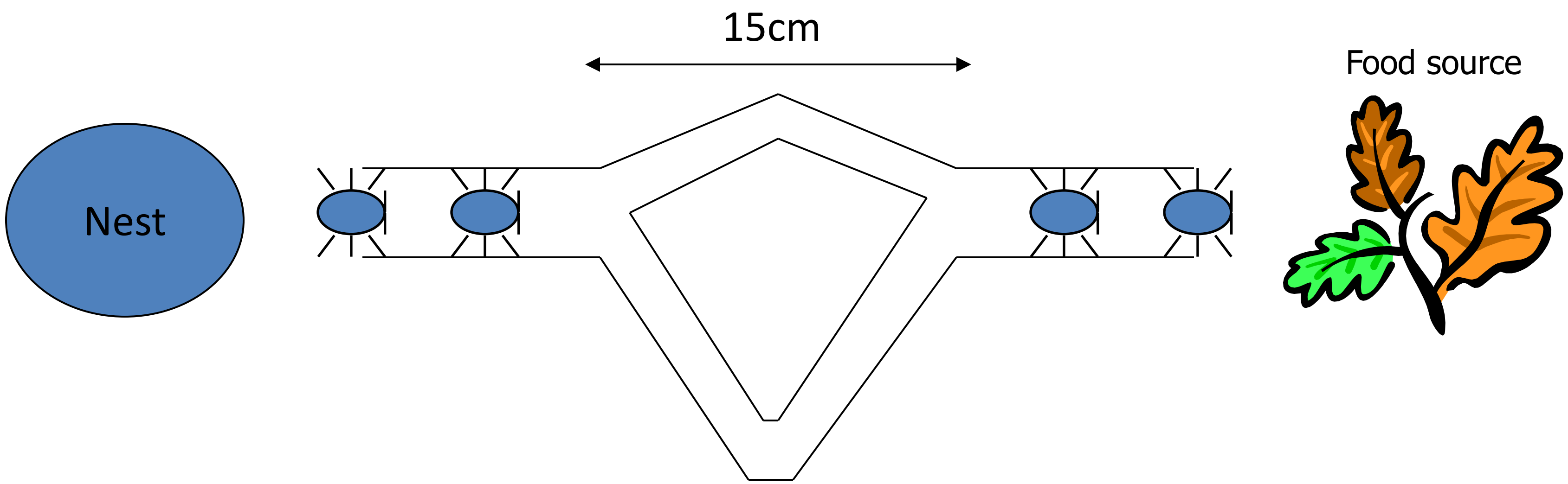
- Experiments conducted on real ants and found very interesting results.
- Deneubourg et al (1989) Double Bridge Experiment



- **Ants observed over time**
- **To begin with - random choices of path**
- **Later, one path taken by most ants (Why?)**

Real Ant Experiments

- Deneubourg et al (1989) Double Bridge Experiment – 2nd Experiment



- To begin with - random choices of path
- Soon, shortest path selected by most ants
- How?

A key player: Stigmergy

- **Stigmergy** is a mechanism of **indirect coordination**, through the **environment**, **between agents** or actions.
- The principle is that the **trace left** in the environment by an individual action **stimulates** the performance of a **succeeding action** by the same or different agent.
- Agents that respond to traces in the environment receive **positive fitness** benefits, **reinforcing** the likelihood of these **behaviors** becoming fixed within a population **over time**.
- Stigmergy is a form of self-organization.

Source: https://en.wikipedia.org/wiki/Stigmergy

Summary

- Computational intelligence (CI) refers to concepts, paradigms, algorithms and implementations of systems that are designed to **show "intelligent" behavior** in complex and changing **environments**.
- Nature-analog or at least **nature-inspired** methods play a **key role** in this.
- Main CI research areas: **Artificial Neural Networks, Evolutionary Computation, Swarm Intelligence**, Fuzzy Logic
  - Machine learning problems: **Artificial Neural Networks**
  - Optimization problems**: Evolutionary Computation, Swarm Intelligence**