Introduction to machine learning

exam topics

Bálint Boda, Péter Szabó, Dominik Bocsi, Dániel Fülep

Fall 2023

Contents

1 Tests for Artificial General Intelligence

1.1 Turing test

A machine and a human both converse with a second human. The second human must evaluate which of the two is the machine. The test is passed if the evaluator is fooled a significant fraction of the time.

The AI Eugene Goostman achieved Turing's estimate of convincing 30% of judges.

1.2 Robot College Student test

A machine enrolls in a university, taking and passing the same classes that humans would, obtaining a degree.

Some LLMs can now pass university level exams without even attending classes.

1.3 Employment test

A machine performs an economically important job at least as well as a human would.

AIs are now replacing humans in many roles like fast food and marketing.

1.4 IKEA test (flatpack furniture test)

An AI views the parts and instructions of an IKEA flat-pack product, then controls a robot to assemble the furniture correctly.

1.5 Coffee test

A machine enters an average home and figures out how to make coffee:

- finds the coffee machine
- finds coffee
- adds water
- finds a mug
- brews the coffee by using the machine properly

This has not yet been completed.

2 Techniques for generative AI

2.1 Autoencoder

Neural networks trained to reproduce their input data at the output layer. By using a bottleneck layer in the middle, autoencoders can learn a compressed representation of the input data, which can be used for generating new samples.

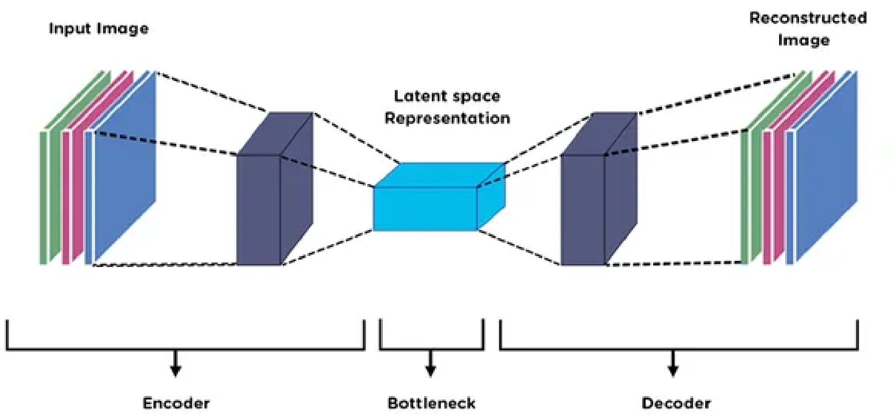

Figure 1: autoencoder

2.2 Generative Adversarial Network

A generator and a discriminator, trained simultaneously through a competitive process. The generator aims to create realistic data, while the discriminator tries to differentiate between real and generated data.

3 Text to image models

- DALL-E
- Stable Diffusion
- Midjourney

4 Foraging Ants

4.1 Model

- ants wander around on 2D grid:
  - starting from nest
  - avoid obstacles (if any)
  - follow pheromone gradient (probabilistically)
- at current location if not carrying food: pick up a piece of food
- if at nest and carrying food: put down food
- deposite a unit of pheromone at current location
  - "A" if searching for food
  - "B" if carrying food
- Pheromone diffuses and evaporates by constant rate, uniformly across space

4.2 Ant Colony Optimization (ACO)

- instead of a grid a graph is used, pheromone is placed on edges
- a random starting node is selected
- next node is selected probabilistically following edge pheromone gradient
- when a solution is found pheromone amounts on path are adjusted: proportionally to quality
- the simulation ends when most ants select the same solution

5 The Schelling model

An agent-based model of segregation.

- plays on a 2D grid
- agents are split into two groups
- each agent occupies exactly one tile
- each agent has a personal tolerance level in $[0, 1]$
- an agent is happy if:

$$\text{tolerance level} \geq \frac{\text{number of neighbours from the other group}}{\text{number of neighbours}}$$

- if an agent is unhappy it moves to an empty tile

6 Basic ethical frameworks for technology

When you invent a new technology, you uncover a new class of responsibilities. If your invention confers power it starts a race, that without coordination/regulation could result in tragedy.

7 Different approaches to machine learning

7.1 Supervised learning

In supervised learning we have access to input data and its desired outputs (labels). Our objective is to train a program to generalize the knowledge from our data, so the labels of new data can be predicted.

7.2 Unsupervised learning

In unsupervised learning the desired output of our input data is unknown. The goal is to learn the structure of the data, so that it can be clustered/categorized.

7.3 Reinforcement learning

In reinforcement learning an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on the actions it takes, allowing it to adjust its behaviour in order to optimize strategies over time.

7.4 Deep learning

A subset of machine learning composed of algorithms that permit software to train itself to perform tasks by exposing multilayered neural networks to vast amount of data.

8 Basic concept of supervised learning

Given a training set of n example input/output pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (called labelled data), the goal is to approximate the (unknown) function f that maps input vectors to outputs, so that the output of new unlabelled data can be predicted.

Figure 2: Supervised learning

9 Supervised learning by decision trees

A decision tree is used to model the mapping between input vectors and decisions (output labels). It is a sequence of tests done starting from the root until a leaf node is reached.

The simplest kind of decision tree is a boolean decision tree, where each test has a single boolean outcome. This means the entire process can be represented as:

$$OUTPUT \iff (Path_1 \lor Path_2 \lor \ldots)$$

where each path is the conjunction of attribute value tests representing a path from the root to a leaf.

Most of the time decision tress represent more complex criteria including both discrete and continuous comparisons.

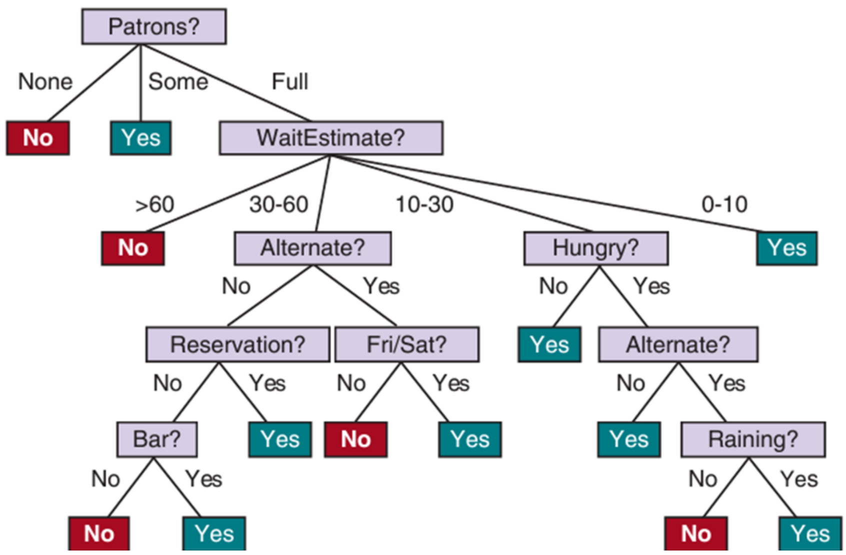

Figure 3: Decision tree

9.1 Tree building

The decision tree algorithm recursively splits the data into subsets based on the values of different features. At each node of the tree, a decision is made by choosing the feature that best separates the data into distinct classes or reduces the variance in the target variable. The process continues until a stopping criterion is met.

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
     A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
     tree ← a new decision tree with root test A
     for each value v of A do
        exs ← {e : e ∈ examples and e.A = v}
        subtree ← LEARN-DECISION-TREE(exs, attributes − A, examples)
        add a branch to tree with label (A = v) and subtree subtree
     return tree
```

Figure 4: Decision tree building

where PLURALITY_VALUE returns the most common label from a set of data.

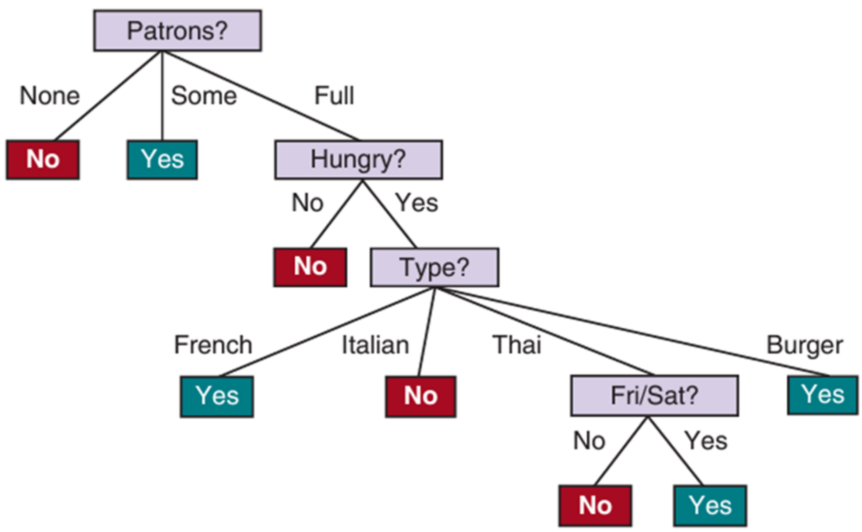

Figure 5: Decision tree

It is important to note this decision tree has shorter paths and is much simpler than our original tree. The algorithm does not have any knowledge about actual function it merely looks at the examples.

This means the tree is fitted to our training data. This results in decision tress not generalizing well:

- If we increase the number of attributes, overfitting is more likely
- If we increase the number of training samples, overfitting is less likely

To improve generalization a technique called pruning is used, during which we examine potentially nodes that only has leaf descendents. If the node appears to be irrelevant it is replaced with a leaf node.

9.2 Pros

- Easy to understand
- scales well to large data sets
- can handle both discrete and continuous inputs
- can performing classification and regression

9.3 Cons

- Suboptimal accuracy (largely due to the greedy search)
- If trees are very deep, making a prediction can be expensive
- decision trees are unstable – adding just one new example can change the entire tree

10 Basic concept of unsupervised learning

Sometimes we are presented with data without any labels. In some of these cases data may even lock distinguishing characteristics, resulting in manual labelling becoming impossible.

Unsupervised learning takes a given set of data, and produces output data (labels) and a function, which maps input to output.

A possible solution to this problem is partitioning data into cluster, groups of highly similar data.

Figure 6: Unsupervised learning

11 k-means algorithm

An unsupervised, iterative method for clustering data.

11.1 Algorithm

1. define k, the number of clusters
2. initialize k cluster centers at random
3. repeat:
   (a) assign each point based on the nearest cluster center
   (b) move the center point of each cluster to mean of its members
   (c) if no points changed ownership exit

11.2 Example

In a wrestling competition, wrestlers are divided into leagues based on their height and weight. Divide the competitors into two leagues ($k = 2$) based on the data obtained using the k-means algorithm. Perform the calculation over two iterations, taking the values of the first and second competitors as the initial center points.

ID	height (cm)	weight(kg)
1	185	76
2	170	66
3	168	68
4	179	74
5	182	73
6	188	75

First iteration:

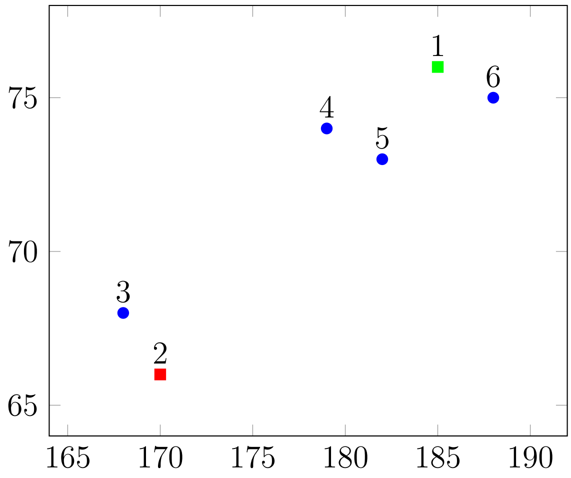

Figure 7: Initialize center points

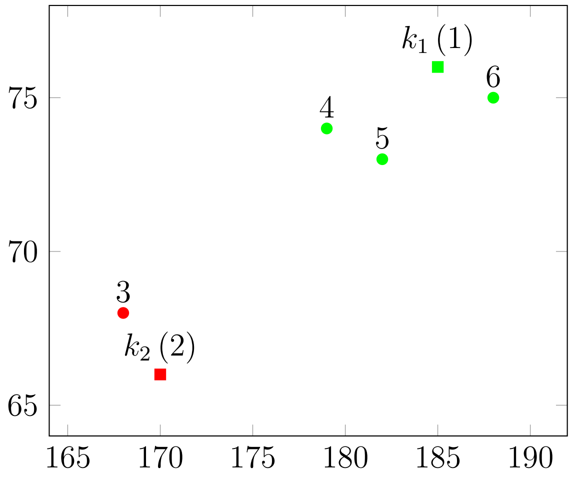

Figure 8: Assign points to nearest center

Recalculate center points:

$$k_1 = \left(\frac{185+179+182+188}{4}, \frac{76+74+73+75}{4}\right) = (183.5, 74.5)$$
$$k_2 = \left(\frac{170+168}{2}, \frac{66+68}{2}\right) = (169, 67)$$

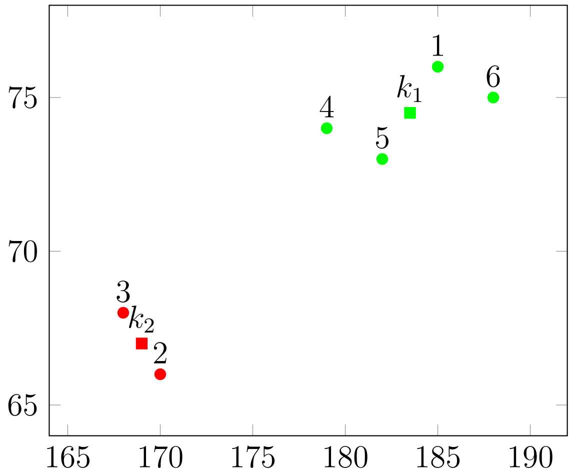

Figure 9: Move cluster centers

Second iteration: The points need to be reassigned, based on the new center points. In our case no points change ownership resulting in the termination of the algorithm.

12 Mechanism of reinforcement learning

Reinforcement learning is a machine learning paradigm where an agent learns to make decisions by interacting with an environment in order to maximize some notion of cumulative reward over time.

The Markov Decision Process (MDP) which is a mathematical framework that provides a formal way to model decision-making in situations where outcomes are partially random and partially under the control of a decision maker. The model contains:

- State Space (S): a set of states, representing the possible configurations or situations of the environment
- Action Space (A): a set of possible actions that the agent can take in each state
- Transition Model ($T(s, a, s') \sim \Pr(s' \mid s, a)$): the likelihood of transitioning from state s to s' given a particular a action
- Reward Function (R): assigns a numerical value, indicating the immediate reward associated with:
  - being in a state ($R(s)$)
  - taking a specific action in a particular state ($R(s, a)$)
  - taking an action and ending up in a different state ($R(s, a, s')$)
- Policy ($\pi : S \to A$): a strategy that specifies the agent's behavior, determining which action to take in each state

The goal of reinforcement learning is for the agent to learn an optimal policy that maximizes the reward function. Both MDPs and reinforcement learning involve the concept of value functions. The state-value function (V) estimates the expected cumulative reward from a given state under a specific policy, while the action-value function (Q) estimates the expected cumulative reward from taking a specific action in a particular state under a specific policy.

13 Q-learning method

Q-learning is a model-free reinforcement learning algorithm that is used to find the optimal action-selection policy for a given finite Markov decision process. Q-learning is particularly well-suited for problems where the environment is not fully known in advance, and the agent needs to learn by interacting with the environment.

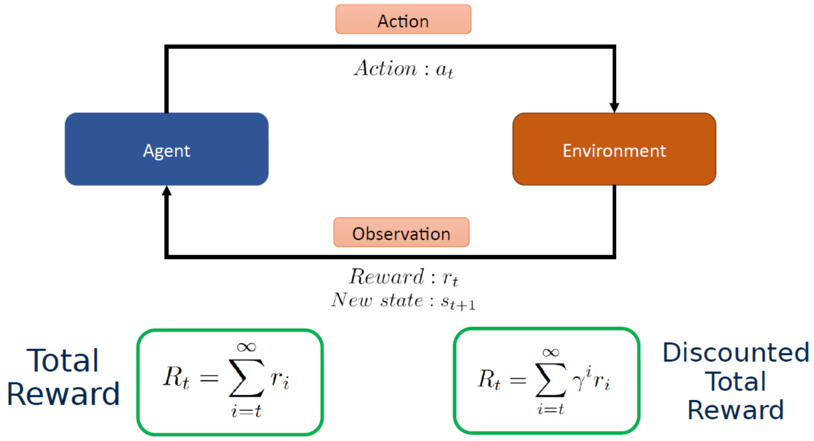

Figure 10: Q-learning

Key concepts and components of Q-learning:

- State (S) and Action (A): The environment is represented as a set of states (S) and a set of possible actions (A). The agent interacts with the environment by taking actions at different states

- Q-Values (Q(s, a)): Q-values are used to represent the expected cumulative reward of taking a particular action in a specific state and following the optimal policy thereafter.

- Q-Table: In Q-learning, a Q-table is used to store Q-values for all state-action pairs. Initially, the Q-table is filled with arbitrary values, and the agent updates these values as it interacts with the environment.

- Learning Iteration: The agent iteratively interacts with the environment, updates the Q-values using the Bellman equation, and refines its policy over time.

Downsides of Q-learning:

- Cannot handle continuous action spaces

- Policy is deterministically computed from the Q function by maximizing the reward, so the model cannot learn stochastic policies

14 Deep Learning methods, value learning and policy learning

Deep learning algorithms can be classified based on their strategy for finding the optimal policy (π^*).

14.1 Value learning

The optimal policy is derived from a learned function Q called the value function. The optimal policy is always choosing the best possible action in every state:

$$\pi^*(s) = \arg\max_a Q(s, a)$$

14.2 Policy learning

In policy learning, the goal is to directly learn the optimal policy without explicitly estimating the value function. It estimates the policy using a parameterized function, which is fine tuned through a learning process.

$$\pi^*(s) \sim \pi(s)$$

15 Policy gradient algorithm

The training algorithm of a policy is the following:

1. Initialize the agent
2. Run the policy until termination:
   (a) Record all states, actions, rewards
   (b) Decrease probability of actions that resulted in low reward
   (c) Increase probability of actions that resulted in high reward

16 Basic concept of evolutionary algorithms

Evolutionary algorithms are stochastic search methods that computationally simulate the natural evolutionary process using the concept of the survival of the fittest.

- Gene: functional entity that encodes a specific feature of the individual (e.g. hair color)
- Allele: value of gene (e.g. blonde)
- Genotype: the specific combination of alleles carried by an individual
- Phenotype: the physical makeup of an organism
- Locus: position of the gene within the chromosome
- Individual (chromosome): represents an encoded (binary or real) candidate solution for the problem
- Population: collection of individuals currently alive

17 Optimization by genetic algorithm

A genetic algorithm is a population-based stochastic optimization method inspired by natural selection. It utilizes three operators inspired by biology: selection, crossover and mutation.

The individuals are evaluated according to some criterion called the fitness function on how good of a solution they can provide to the given problem. Better individuals have a higher fitness value, thus they have a higher chance to survive.

17.1 Selection

There are many ways to select chromosomes to survive to the next generation. One such method is roulette wheel selection (also known as fitness proportionate selection), where the chance of selecting an individual is proportionate to its fitness value.

$$\text{expected count in crossover} = \frac{\text{fitness of individual}}{\text{total fitness of population}} \cdot \text{size of population}$$

17.2 Crossover

A random locus is selected. The tails after the locus are exchanged.

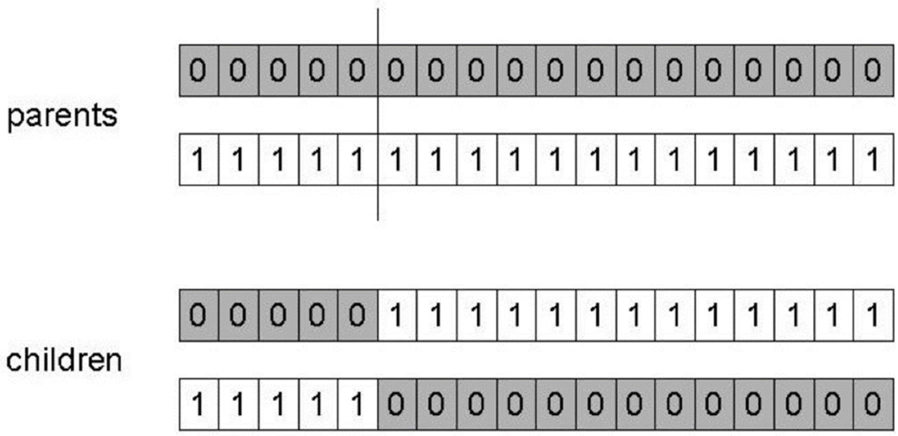

Figure 11: Crossover

17.2.1 Alternative crossover operators

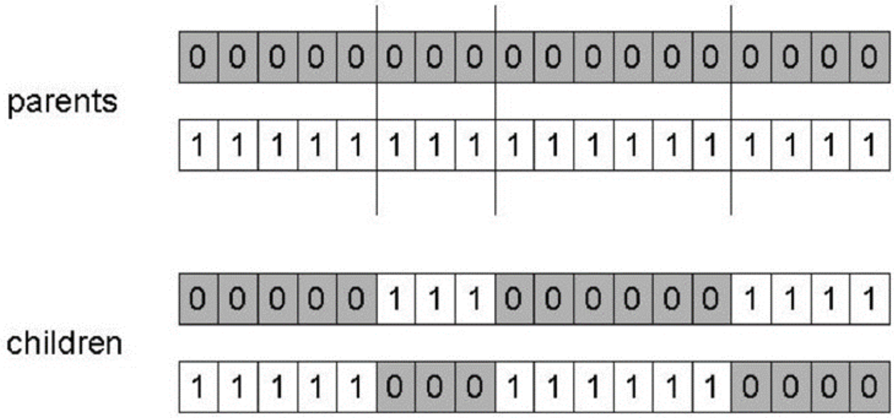

Figure 12: n point crossover

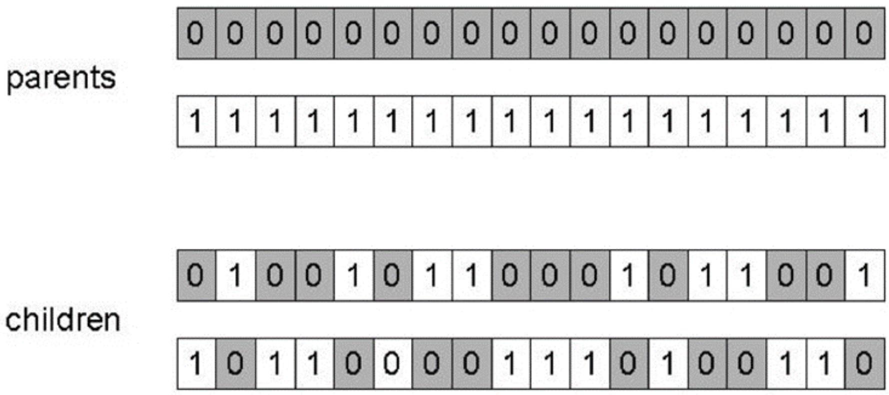

Figure 13: uniform crossover

17.3 Mutation

Each gene has a chance to change with a p_m probability called the mutation rate.

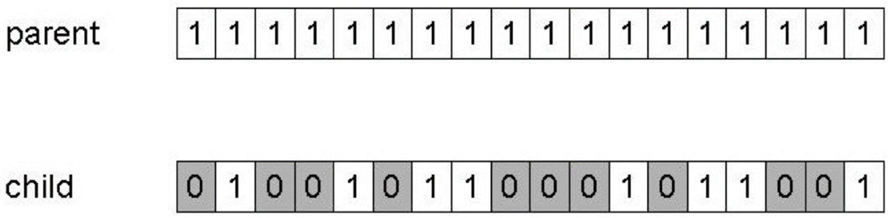

Figure 14: Mutation

17.4 Example

Find x^2 over $\{0, 1, \ldots, 31\}$!

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

Figure 15: Selection

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Figure 16: Crossover

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	28	784
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	20	400
Sum				2538
Average				634.5
Max				784

Figure 17: Mutation

18 Basic concept of genetic programming, differences with genetic algorithms

Genetic programming applies the approach of genetic algorithm to the space of possible computer programs, allowing the generation of syntactically valid and executable programs.

This way each individual becomes an expression tree representing a syntactically valid executable program.

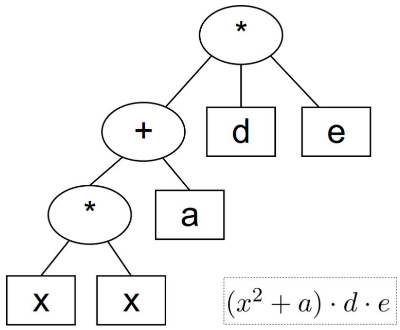

Figure 18: Expression tree

The approach requires the definition of the following:

- set of terminals (foo, bar, x, y, i)
- set of functions (+, -, IF, %)
- the fitness measure
- the parameters for the run
- the criterion for terminating a run

18.1 Selection (reproduction)

- select parent based on fitness
- copy it without any change into the next generation of the population

18.2 Mutation

- select 1 parent (based on fitness)
- pick a point in the tree
- replace the subtree at the picked point with a new subtree generated the same way as trees in the initial population
- put the offspring into the next generation of the population

18.3 Crossover

- select 2 parents (based on fitness)
- randomly pick a node in the tree for first parent
- independently randomly pick a node for second parent
- exchange the subtrees at the two picked points
- put the offspring into the next generation of the population

19 The basic concept of swarm intelligence

A collective system capable of performing complex tasks in a dynamic and changing environment without any external control or central coordination. Capable of achieving a collective performance that cannot normally be achieved by the organism alone.

- Distributed, massively parallel (many agents)
  - Individual agents are simple and disposable (cheap)
  - Typically partially stochastic (i.e., non-deterministic)
- Intelligence is optimising or performing a task
  - Stochastic: approximations, different runs may give slightly different results
  - Typically continuous optimisation / performance of a series of tasks
- Robust:
  - Adapts to changing environments (and performs well)
  - Graceful degradation: withstands removal of agents (potentially many of them)

20 Optimization by Particle Swarm Optimization

PSO applies the concept of social interaction to problem solving. Particles move in swarms in search of the best solution. Each particle is a spatial point that adjusts its flight based on experience gained by itself and by its peers. This way particles will converge towards the optimal solution.

- pbest: best solution achieved by the particle
- gbest: best solution achieved by the swarm

Each particle changes its position based on the following information:

- current position
- current velocity
- distance between current position and pbest
- distance between current position and gbest

21 Recent swarm intelligence techniques

21.1 Firefly

- similar to Particle Swarm Optimization
- particles are fireflies who emit light
- light intensity reduces over distance and respects absorption (γ)
- brightness depends on how good of a solution they provide:

  $$(I_0, d, \gamma) = I_0 \cdot e^{-\gamma d^2}$$

  where d is a distance metric and I_0 is the individuals light intensity at 0 distance
- each individual moves towards brighter fireflies and also do some random movement

21.1.1 Special cases

- $\gamma = 0$ clear air, all fireflies can see each other
- $\gamma = \infty$ foggy air, fireflies can't see each other result in random walk

21.2 Grey wolf

GWO algorithm

22 Basics of neural networks

The fundamental cellular unit of nervous system is called neuron. A neuron is a simple processing unit connected to approximately 1000 neurons. The function of a neuron is receiving and combining signals from other neurons through input paths called dendrites. If the combined input signal is strong enough, the neuron fires and produces an output signal and sends the output along the axon to other neurons.

An artificial neuron is a mimic of a biological neuron.

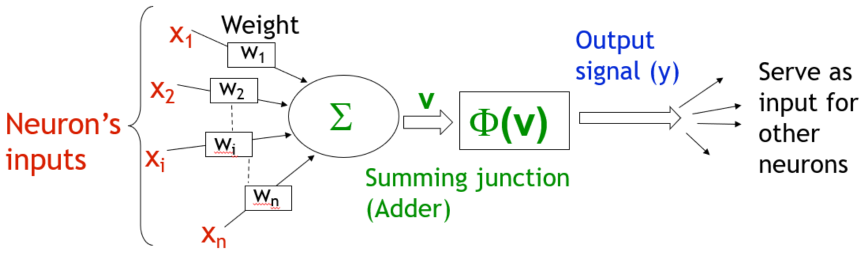

Figure 19: Artificial neuron

A single neuron is unable to solve complex problem, however an interconnected system of neurons can deliver complex behaviour. Φ is the activation function responsible for keeping the values in given range.

23 Perceptron, Perceptron training

23.1 Definition

The perceptron is an supervised algorithm for learning a binary classifier called a threshold function. A (single-layer) perceptron is only capable of learning linear threshold functions. In the context of neural networks, a perceptron is an artificial neuron using the Heaviside step function as the activation function.

Multilayer perceptron also exist which are capable of separating linearly inseparable variables as well.

23.2 Training

1. Initialization
   - set the initial weights
   - set the threshold θ to a random number in $[-0.5, 0.5]$
   - set the learning rate μ to a positive value less than 1
2. Activation (calculate actual output of at iteration)
3. Weight learning (update weights based on output)
4. Iteration (repeat steps 2-4 until convergence)

24 Basic concept of CRISP-DM

An open standard process model that describes common approaches used by data mining experts, that breaks the process of data mining into six major phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment