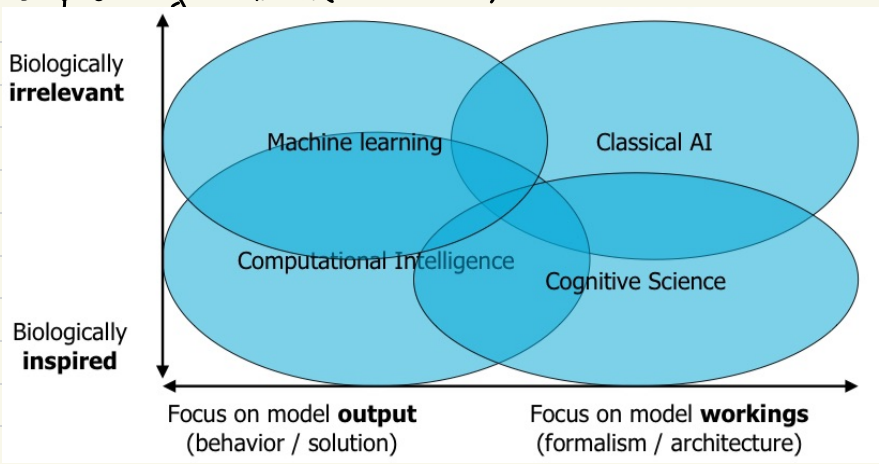
1. Introduction

AI History

- → Birth of AI (1943-56)
  - McCulloch and Pitts Neuron (1943)
  - Turing test (1950)
  - Darthmouth summer AI workshop (1956)
- → Great expectations (1956 - late 1960s)
  - LISP (McCarthy)
  - Rosenblatt's Perceptron
  - General problem solver (Newell & Simon)
- → Disillusionment (late 1960s - early 1970s)
- → Expert systems (early 1970s - mid 1980s)
- → Rebirth of artificial neural networks (1965 - onwards)
- → Evolutionary computation (early 1970s - onwards)
- → Deep learning revolution (2012 - onwards)



4 perspectives on CS

- → simulation of workings of brain (neural networks)
- → simulation of evolution (evolutionary computing)
- → simulation of (group) animal behavior (swarm intelligence)
- → simulation of animal learning (reinforcement learning)

2. Mathematical foundations and limitations

maximize $F(\theta)$ = minimize $-F(\theta)$

fitness of candidate - how "good" that state is in solving the "problem" (abstract view)
- SSE (sum of squared differences)
- cross-entropy (difference between two probability distributions)

Supervised learning

task T to link inputs x to outputs y in some domain $\mathcal{E}$ through f, $f: x \to y$

domain knowledge: $E = \{(x_i, y_i),\ i=1,\dots,p\} \subset \mathcal{E}$ (examples/experience); $\hat{y} = f(x;\theta)$

Binary classification: $y \in \{0,1\}$

Multi-class classification: $y \in \{1,2,\dots,m\}$

Regression: $y \in \mathbb{R}$

Unsupervised learning

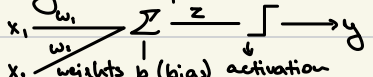
model f used for reasoning, decision making, predicting, communicating; y not given

Clustering: find groups of similar data based on distribution in input space

Error-correction: patterns giving incomplete or noisy ones (association)

optimization: gradient descent, random/local/exhaustive search, genetic algorithms

3. Multilayer Perceptron

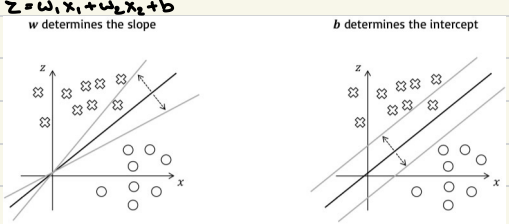


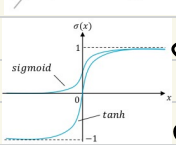
$z = w^T x + b$

$\text{step}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$

$y = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{if } w^T x + b < 0 \end{cases}$

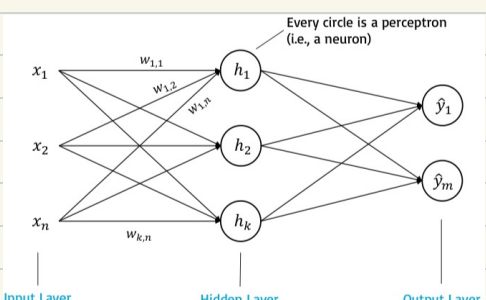
$z = w_1 x_1 + w_2 x_2 + b$





$\sigma_{\text{sigmoid}}(x) = \frac{1}{1+e^{-x}}$

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



loss: $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y - \hat{y})^2$

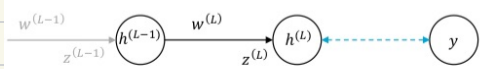
3.1. Gradient Descent

$$w^{t+1} = w^t - \alpha \frac{\partial \mathcal{L}}{\partial w}\bigg|_{w=w^t}$$ — Gradient Update

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}} = h_k^{(l-1)} \sigma'\left(z_j^{(l)}\right) \frac{\partial \mathcal{L}}{\partial h_j^{(l)}}$$ — Derivative of the weights

$$\frac{\partial \mathcal{L}}{\partial h_j^{(l)}} = \sum_{i=0}^{n_{l+1}-1} w_{ij}^{(l+1)} \sigma'\left(z_i^{(l+1)}\right) \frac{\partial \mathcal{L}}{\partial h_i^{(l+1)}}$$ — Derivative of $\mathcal{L}$ with respect to $h_j^{(l)}$ for a layer (l)

$$\frac{\partial \mathcal{L}}{\partial h_j^{(L)}} = 2(y_j - h_j^{(L)})$$ — Derivative of $\mathcal{L}$ with respect to $h_j^{(L)}$ for the final layer (L)



Loss for one training example

$$\mathcal{L}_0 = (y - h^{(L)})^2 \qquad h^{(L)} = \hat{y}$$

$$z^{(L)} = w^{(L)} h^{(L-1)} + b^{(L)} \qquad h^{(L)} = \sigma(z^{(L)})$$

$$\sigma'_{sigmoid}\left(z^{(L)}\right) = \sigma\left(z^{(L)}\right)\left(1 - \sigma\left(z^{(L)}\right)\right) = h^{(L)}(1 - h^{(L)})$$

$$\boldsymbol{\frac{\partial \mathcal{L}_0}{\partial w^{(L)}}} = \frac{\partial \mathcal{L}_0}{\partial h^{(L)}} \frac{\partial h^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}} = 2(y - h^{(L)}) \sigma'(z^{(L)}) h^{(L-1)}$$

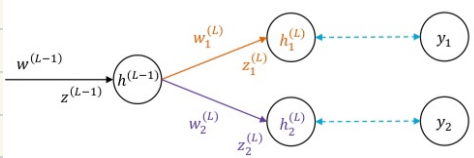
$$\frac{\partial \mathcal{L}_0}{\partial h^{(L)}} = 2(y - h^{(L)})$$

$$\frac{\partial h^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = h^{(L-1)}$$

$$\boldsymbol{\frac{\partial \mathcal{L}_0}{\partial b^{(L)}}} = \frac{\partial \mathcal{L}_0}{\partial h^{(L)}} \frac{\partial h^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b^{(L)}} = 2(y - h^{(L)}) \sigma'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1$$



$$\mathcal{L}_0 = \sum_{i=1}^{2} (y_i - h_i^{(L)})^2 = (y_1 - h_1^{(L)})^2 + (y_2 - h_2^{(L)})^2$$

$$z_1^{(L)} = w_1^{(L)} h^{(L-1)} + b_1^{(L)} \qquad h_1^{(L)} = \sigma(z_1^{(L)})$$

$$z_2^{(L)} = w_2^{(L)} h^{(L-1)} + b_2^{(L)} \qquad h_2^{(L)} = \sigma(z_2^{(L)})$$

$$z^{(L-1)} = w^{(L-1)} h^{(L-2)} + b^{(L-1)} \qquad h^{(L-1)} = \sigma(z^{(L-1)})$$

$$\frac{\partial \mathcal{L}_0}{\partial w_1^{(L)}} = \frac{\partial \mathcal{L}_0}{\partial h_1^{(L)}} \frac{\partial h_1^{(L)}}{\partial z_1^{(L)}} \frac{\partial z_1^{(L)}}{\partial w_1^{(L)}} = 2(y_1 - h_1^{(L)}) \sigma'(z_1^{(L)}) h^{(L-1)}$$

$$\frac{\partial \mathcal{L}_0}{\partial w_2^{(L)}} = \frac{\partial \mathcal{L}_0}{\partial h_2^{(L)}} \frac{\partial h_2^{(L)}}{\partial z_2^{(L)}} \frac{\partial z_2^{(L)}}{\partial w_2^{(L)}} = 2(y_2 - h_2^{(L)}) \sigma'(z_2^{(L)}) h^{(L-1)}$$

$$\boldsymbol{\frac{\partial \mathcal{L}_0}{\partial w^{(L-1)}}} = \frac{\partial \mathcal{L}_0}{\partial h^{(L-1)}} \frac{\partial h^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} =$$

$$= \left(\frac{\partial \mathcal{L}_0}{\partial h_1^{(L)}} \frac{\partial h_1^{(L)}}{\partial z_1^{(L)}} \frac{\partial z_1^{(L)}}{\partial h^{(L-1)}} + \frac{\partial \mathcal{L}_0}{\partial h_2^{(L)}} \frac{\partial h_2^{(L)}}{\partial z_2^{(L)}} \frac{\partial z_2^{(L)}}{\partial h^{(L-1)}} \right) \frac{\partial h^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} =$$

$$= \left(2\left(y_1 - h_1^{(L)}\right)\sigma'\left(z_1^{(L)}\right) w_1^{(L)} + 2\left(y_2 - h_2^{(L)}\right)\sigma'\left(z_2^{(L)}\right) w_2^{(L)}\right) \sigma'\left(z^{(L-1)}\right) h^{(L-2)}$$

4. Hyperparameters
   -> activation function
      ° output layer
         - Binary classification ($K=1$): $\hat{y} = \text{sigmoid}(z) = \frac{1}{1+e^{-z}}$
         - Multiclass classification ($K>1$):
            non-mutually exclusive labels (multilabel): $\hat{y}_i = \text{sigmoid}(z_i) = \frac{1}{1+e^{-z_i}}$
            mutually exclusive labels: $\hat{y}_i = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
         - Regression (any K): $\hat{y}_i = z_i$
      ° hidden layers
         - Sigmoid (sigmoid)
         - Hyperbolic Tangent (tanh)
         - Rectified Linear Unit (ReLU) -> $h(z) = \max(0, z)$
         - Leaky Rectified Linear Unit (LReLU) -> $h(z) = \max(\text{neg-slope} * x, x)$
   -> loss function
      - Binary classification: $\mathcal{L} = -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$ binary cross-entropy
      - Multiclass classification: $\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$ categorical cross-entropy
      - Multiclass multilabel classification: $\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$ binary on each output
      - Regression: $\mathcal{L} = \frac{1}{K}\sum_{i=1}^{K} (y_i - \hat{y}_i)^2$
   -> # layers
   -> # neurons
   -> regularization: $\mathcal{L} = \mathcal{L}_{error} + \lambda \mathcal{L}_{reg}$
      - L1: $\mathcal{L}_{reg} = \sum_i |w_i|$
      - L2: $\mathcal{L}_{reg} = \sum_i w_i^2$
   -> weights initialization
      - normal (Gaussian) distribution: $w_{jk}^{(l)} = N(0, 1)$
      - Xavier initialization (sigmoid/tanh): $w_{jk}^{(l)} = N(0, \frac{1}{n_{l-1}})$
      - He initialization (ReLU/LReLU): $w_{jk}^{(l)} = N(0, \frac{2}{n_{l-1}})$
   -> learning rate: speed in which weights are updated
   -> batch size

	update after	computation	variance
- Stochastic:	each example	fast	high
- Mini-batch:	some examples	fast	low
- Batch:	all examples	low	low

5. Convolutional & Recurrent Neural Networks

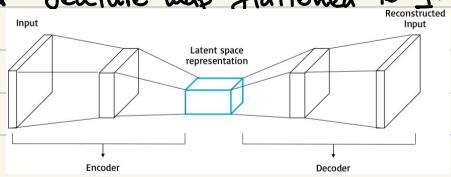
5.1. CNN

MLP on images - scales badly, ignores spatial structure, cannot handle translation

convolution filter - element-wise multiplication of filter and image intensity and sum

multidimensional filters - reduce width and height, increase depth of feature map

after last filter layer - feature map flattened to 1-dimensional array



Applications:

- image classification
- object detection
- object segmentation

Encoder-Decoder Architecture - symmetric structure with bottleneck middle

5.2. RNN

MLP on sequential data - requires fixed length, inputs treated equally

Self-supervised learning:

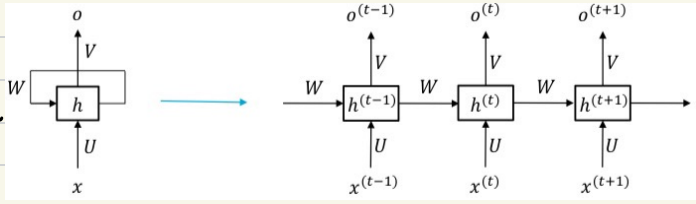
→ image reconstruction

→ next word prediction

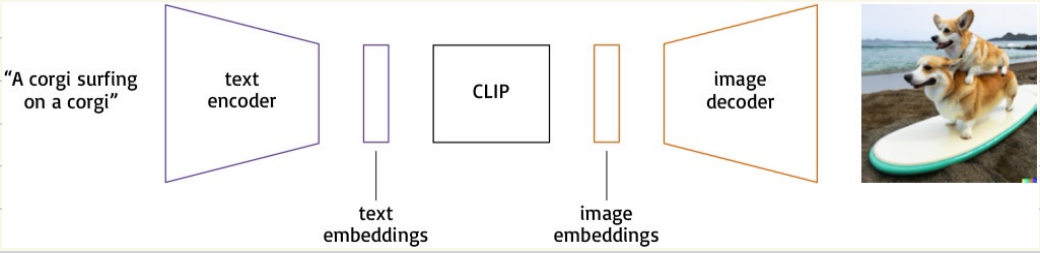
→ word embeddings generation

Applications:

- text classification
- machine translation
- chatbot



DALL·E



6. Genetic Algorithms (John Holland, 1970s)

Evolutionary Computing - used in optimization problems with a given objective function and a set of constraints

EC algorithms - flexible optimization:
- no constraints in type of variables
- works well with non-linear objective functions
- insensitive to discontinuities or shape
- adapted to multi-objective optimization problems
- few assumptions about the problem

intelligence - capability of a system to adapt its behaviour to an ever changing environment

heredity - process by which children receive properties of their parents

variation - variety of traits present in population

selection - mechanism by which some individuals have an opportunity to be parents and others do not ("survival of the fittest")

fitness - goodness of a candidate solution

Assumptions:
-> Every candidate in solution set has "fitness"
-> Similar solutions have similar fitness.

population - set of candidate solutions

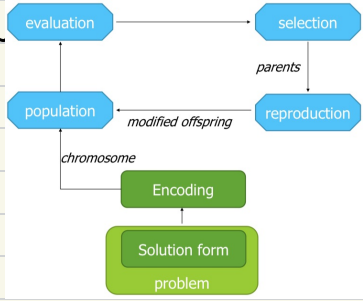
fitness - criterion for evaluating the goodness of candidates

selection - process through which solutions are picked for further refinement

reproduction - process through which new solutions are produced under assumptions
- exploitation - good solutions combined likely yield even better ones
- exploration - unexpected changes in solutions can yield better ones

chromosome - bit-string encoded candidate solution

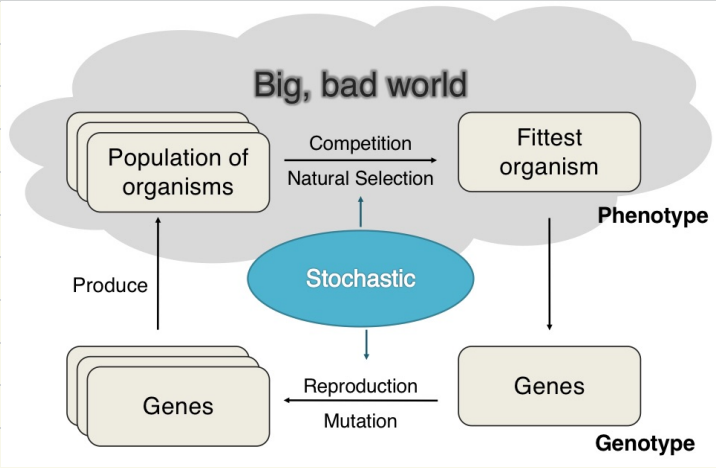
gene - each bit of the solution



Termination:
-> sufficiently good solution found
-> predefined number of iterations executed

Application Areas:
- Economics, Logistics, Engineering, Machine learning

Variations:
- → Coding
  - - Binary
  - - Gray
- → Selection
  - - Roulette-wheel
  - - Ranking
  - - Tournament
- → Cross-over
  - - Single-point
  - - Multi-point
  - - Uniform



Hyperparameters:
- → Length of chromosome, L (depends on required precision)
- → Population size, N (depends on search space)
- → Cross-over probability, P_c (generally high, 0.7)
- → Mutation probability, P_m (generally low, $\frac{1}{L}$ or $\frac{1}{10L}$)
- → Number of generations, G (depends on computational resources)

Holland's schema theorem:
The presence of
- short,
- low-order schema with
- above-average fitness

increases in successive generations

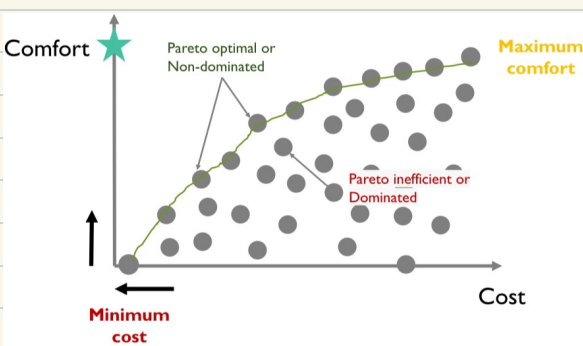
7. Multi-objective optimization

$\min/\max f_m(x), \quad m = 1, 2, \ldots, M$

subject to $g_j(x) \geq 0, \quad j = 1, 2, \ldots, J$

$h_k(x) = 0, \quad k = 1, 2, \ldots, K$

$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, n$



Pareto optimal solution – A solution that improves at least one objective without degrading the performance in any other objective

Advantages:

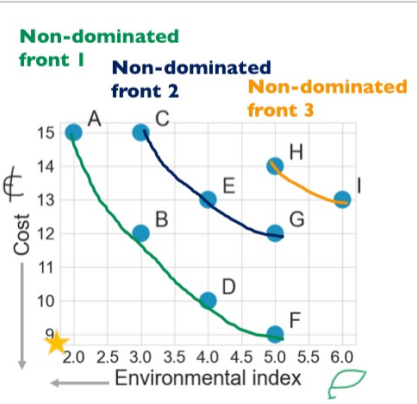
-> non-linear objective functions

-> insensitive to discontinuities or shape of Pareto front

-> flexible

-> Pareto set generated in single run

Car type	Env index (min)	Cost (min)	Points dominated	No. points that dominate	Pareto rank
A	2_{min}	$15K_{max}$	C	0	1
B	3	12K	C,E,G,H,I	0	1
C	3	15K	-	0	2
D	4	10K	E,G,H,I	0	1
E	4	13K	H,I	0	2
F	5	$9K_{min}$	G,H,I	0	1
G	5	12K	H,I	0	2
H	5	14K	-	0	3
I	6_{max}	13K	-	0	3



Crowding distance on front 1

$CD_{im} = \frac{f_m(x_{i+1}) - f_m(x_{i-1})}{f_m(x_{max}) - f_m(x_{min})}$

$CD_i = \sum_{m=1}^{M} CD_{im}$

env: 6 max, 2 min -> 4 diff

cost: 15 max, 9 min -> 6 diff

$CD_B = \frac{4-2}{4} + \frac{15-10}{6} = 1\frac{1}{3}$

$CD_D = \frac{5-3}{4} + \frac{12-9}{6} = 1$

Choose larger crowding distance to preserve diversity.

Car type	Env index (min)	Cost (min)	CD
A	2	15K	inf
B	3	12K	1.33
D	4	10K	1
F	5	9K	inf

selection pressure mechanism: non-dominated sorting

diversity preservation: crowding distance

8/9.0. Swarm Intelligence

intelligence in group; single parts not aware of bigger goal; no leader/centralization

interaction/coordination at microlevel { direct – through senses
indirect – through changes in environment

Ant Colony Optimization (ACO) -> combinatorial optimization (routing)

Particle Swarm Optimization (PSO) -> continuous optimization (control)

8. Ant Colony Optimization

Foraging behavior - act of gathering wild food resources

indirect communication mediated by pheromones

pheromone - secreted/excreted chemical factor enabling effective exploration

optimization problem - find shortest way to some goal

- initial: all links have the same ($\neq 0$) probability to be explored
- exploration: release pheromone τ, proportional to length of link
- choose path: depends on amount of pheromone deposited on it
- stochastic: Near-optimal solutions

pheromone on i-th link by ant k, $\Delta\tau_i^k = \frac{Q}{L_i}$ ← Q constant, L_i ← link length

update probability to take i-th link, $p_i = \frac{\sum_k \Delta\tau_i^k}{\sum_j \sum_k \Delta\tau_j^k}$ ← pheromones on link

assuming total pheromone on i-th link, $T_i = \sum_k \Delta\tau_i^k$ simplification

Assumptions:

→ ants have some memory

→ ants are not completely blind

→ discrete time

→ randomly assigned starting points

$\tau_{i,j} = (1-p)\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^k$, p - evaporation constant

$P_{i,j}^k(t) = \begin{cases} 0 \\ \frac{T_{i,j}(t)^\alpha \eta_{i,j}^\beta}{\sum_{m \in C_i^k} (\tau_{i,m}(t)^\alpha \eta_{i,m}^\beta)} \end{cases}$, otherwise $\eta_{i,j} = \frac{1}{L_{i,j}}$ - heuristic (can be another)

$j \in C_i^k$ C_i^k - set of all unvisited cities reachable from i

α, β - pre-specified parameters (trail x visibility)

Hyperparameters:

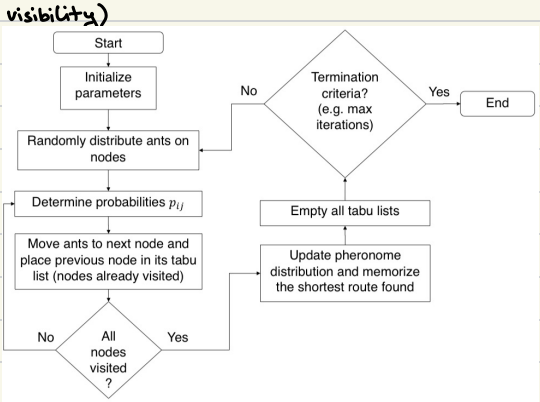
→ # ants

→ initial distribution of ants

→ Q value (the larger, the better precision)

→ α, β values

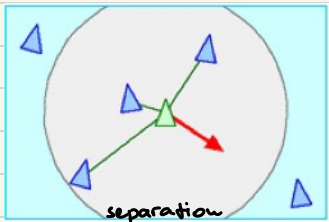
→ evaporation rate

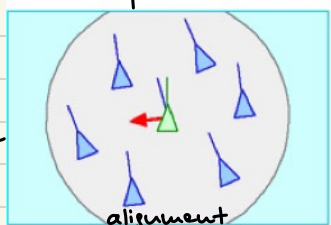


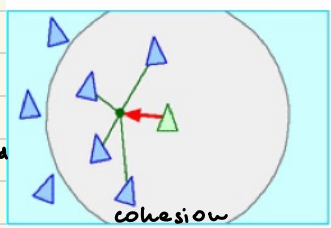
Applications:

- Scheduling, Routing, Network design, Image Processing, Biology

9. Particle Swarm Optimization







separation - steer to avoid crowding local flockmates

alignment - steer toward the average heading of locals

cohesion - steer to move toward the average position of locals

particle → possible solution
- → vector of numbers
- → assess "fitness" based on position

discrete synchronized movement of particles

change position according to results of particles and their companions

$x_i(t)$ - position (vector) of particle i at timestep t

$v_i(t+1)$ - velocity (vector) that drives particle from t to $t+1$

$x_i(t+1) = x_i(t) + v_i(t+1)$

$x_i(0)$ - randomly sampled from $U(x_{min}, x_{max})$

fitness function, $f(x_i(t))$

individual-best solution, $y_i(t)$ - best solution (position) found by particle i after t steps

global-best solution, $\hat{y}(t)$ - best position found by the whole swarm after t steps

$\hat{y}(t) = \max\{f(y_0(t)), f(y_1(t)), \dots, f(y_n(t))\}$ or $\max\{f(x_0(t)), f(x_1(t)), \dots, f(x_n(t))\}$ (best so far / best in this moment)

Velocity Update: $v_{ij}(t+1) = v_{ij}(t) + c_1 r_{ij}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{ij}(t)[\hat{y}_j(t) - x_{ij}(t)]$

velocity of particle i, dimension j — cognitive component — social component

$r_{ij}(t), r_{ij}(t)$ - random values in $[0, 1]$; c_1, c_2 - positive 'acceleration'-constants

Acceleration coefficients:

→ regulate self-confidence and trust in neighbors

→ typically, $c_1 \sim c_2$, $0 \le c_1, c_2 \le 4$

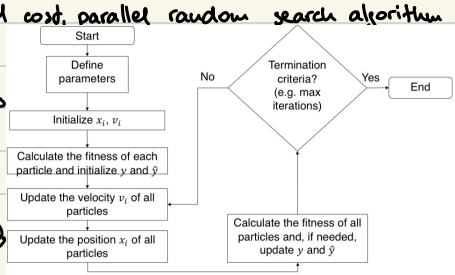
→ $c_1 = c_2 = 0$, fly in current speed until boundary of search space

→ $c_1 \gg c_2$ - local search, $c_2 \gg c_1$ - converge to local optima

↑#particles (10-30) = ↑ diversity, ↓ iterations, ↑ computational cost, parallel random search algorithm

Termination:

- max # iterations, good enough solution, no & positions velocities close to 0

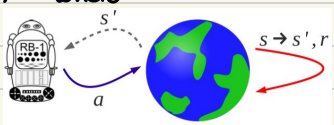


Variations:

- Maximum Velocity, Inertia ($v_{ij}(t+1) = \phi v_{ij}(t) + \dots$), Interaction with Neighbors, Binary PSO (inertia weight)

10. Reinforcement Learning

Markov Decision Process (MDP) $M = \langle S, A, T, R \rangle$ where:



- → S - set of world states s
- → A - set of actions a
- → T - transition function, specifying $p(s'|s,a)$

! Markov assumption: action effect depends only on current state

$p(s_t|s_{t-1}a_{t-1}s_{t-2}a_{t-2}\dots) = p(s_t|s_{t-1}a_{t-1})$; $T(s,a,s') = p(s'|a,s)$

- → R - reward function; $R(s',a,s) / R(s,a) / R(s') = r$

Methodology:

- optimize long-term rewards
- optimize sum of reward in an episode: $G_t = R_{t+1} + R_{t+2} + \dots + R_T$
- for continuous, discount factor $\gamma \in [0,1)$: $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$
- stochastic, optimize expected reward $v_\pi(s) = E_\pi[G_t | S_t = s]$, where π - policy
- satisfy Bellman equation: $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$
- update equation: $v_{k+1}(s) := r(s,\pi(s)) + \sum_{s'} p(s'|s,\pi(s)) \gamma v_k(s')$
- action-value function: $q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$
- policy improvement: $\pi'(s) \leftarrow \max_a q_\pi(s,a)$

11. Learning

$\hat{v}_\pi(s) := (1-\alpha)\hat{v}_\pi(s) + \alpha[r + \gamma \hat{v}_\pi(s')]$

(α: learning rate; $r + \gamma \hat{v}_\pi(s')$: update target)

$\hat{v}_\pi(s) := \hat{v}_\pi(s) + \alpha[R(s,a) + \gamma \hat{v}_\pi(s') - \hat{v}_\pi(s)]$ → Temporal Difference error

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ → SARSA

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma [\max_b Q(s',b)] - Q(s,a)]$ → Q-learning } off-policy

Action Selection / Exploration policy

- → uniform random: $p(a) = \frac{1}{|A|}$
- → greedy: $\arg\max_a Q(s,a)$
- → epsilon greedy: ε uniform random + $(1-\varepsilon)$ greedy
- → Boltzmann: $p(a) = \frac{\exp[\beta \times Q(s,a)]}{\sum_{i=1}^{|A|} \exp[\beta \times Q(s,a_i)]}$