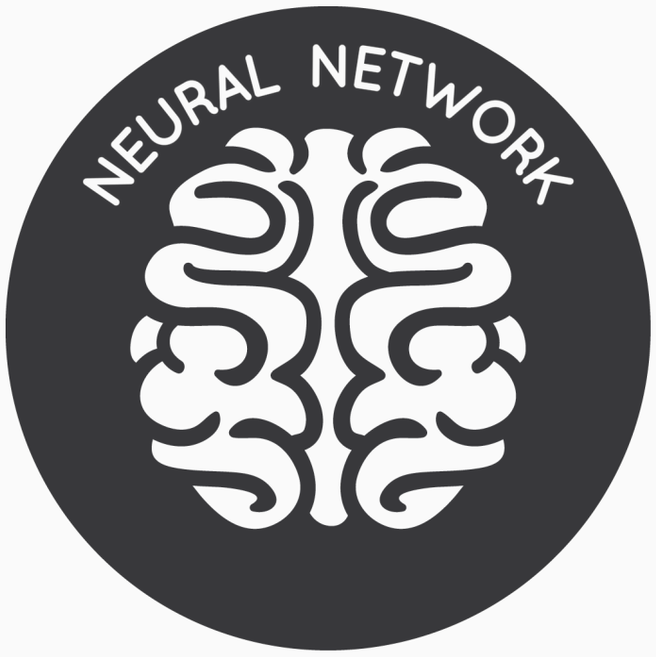
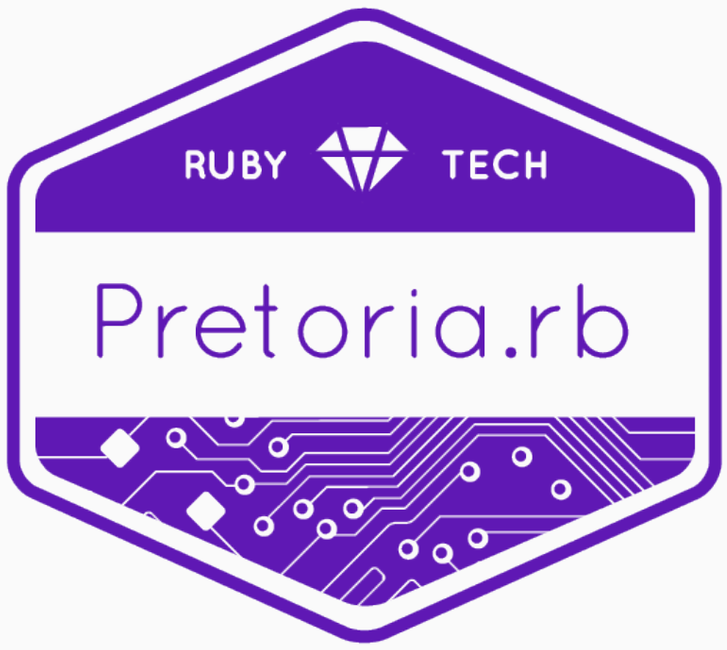

WELCOME TO JOZI RUBY

Platform 45

(Building for the web)



(AI Fanatic)



(Community support)

This talk is about **CI**, namely, Computational Intelligence – the study of adaptive mechanisms to enable or facilitate intelligent behaviour in complex and changing environments. These mechanisms include those **CI** paradigms that exhibit an ability to learn or adapt to new situations, to generalise, abstract, discover and associate.

COMPUTATIONAL

INTELLIGENCE

AI

Weak AI

"specific"

Strong AI

"general"

New era AI

"CI"

Old era AI

"brute force"

Stochastic

Deterministic

Evolutionary Computation

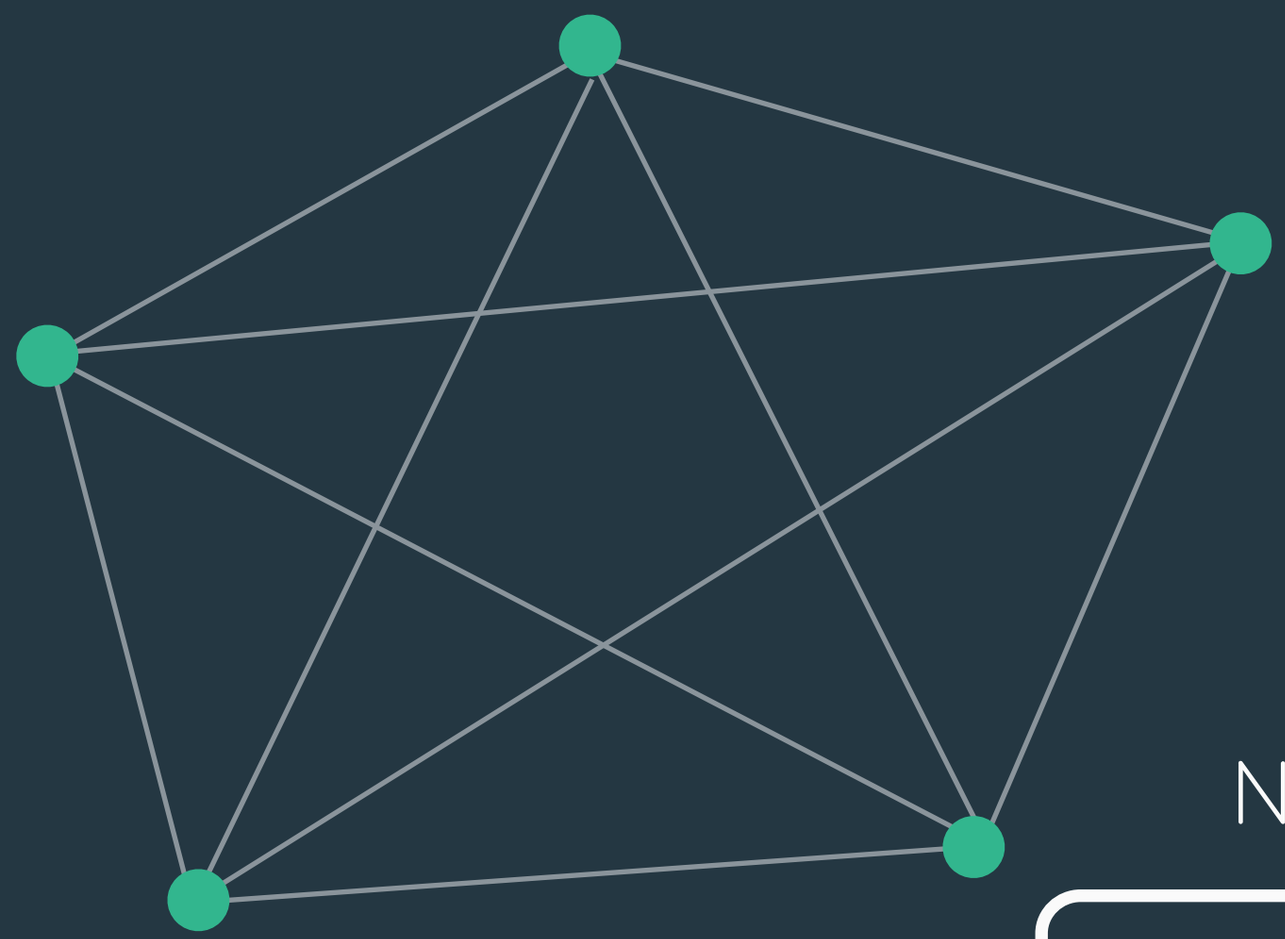
EC

Swarm Intelligence

SI

FS

Fuzzy Systems



AIS

Artificial Immune Systems

NN

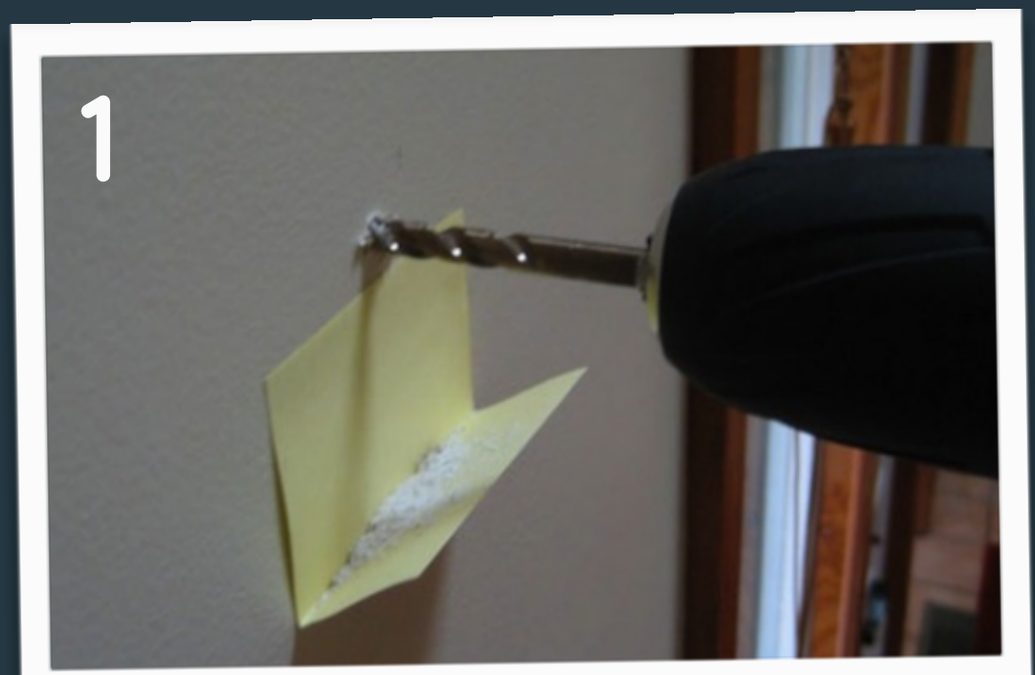
Neural Networks

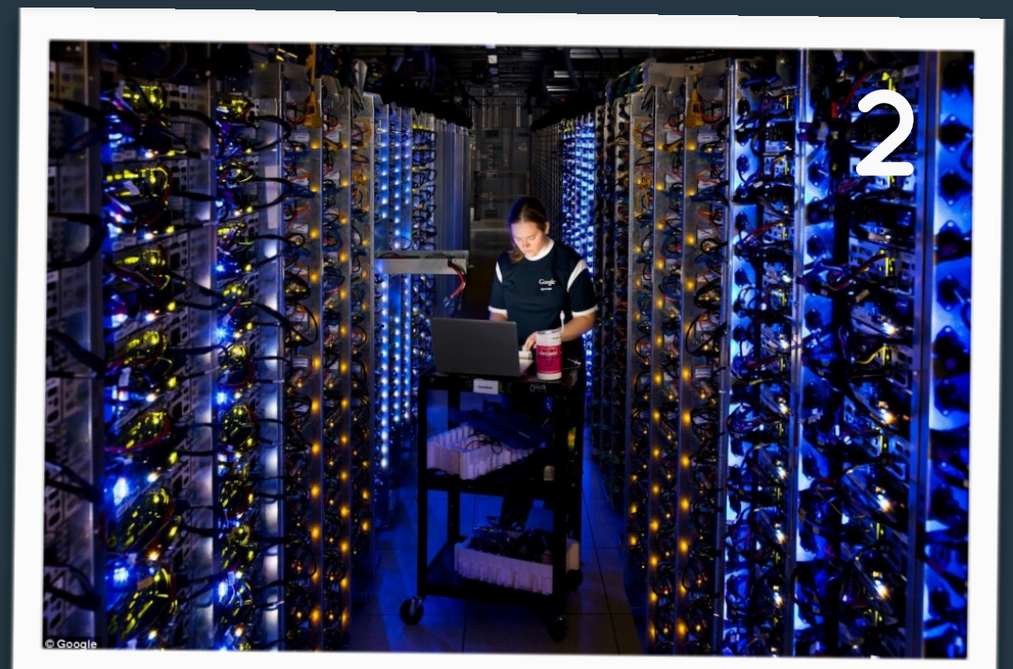
Deterministic

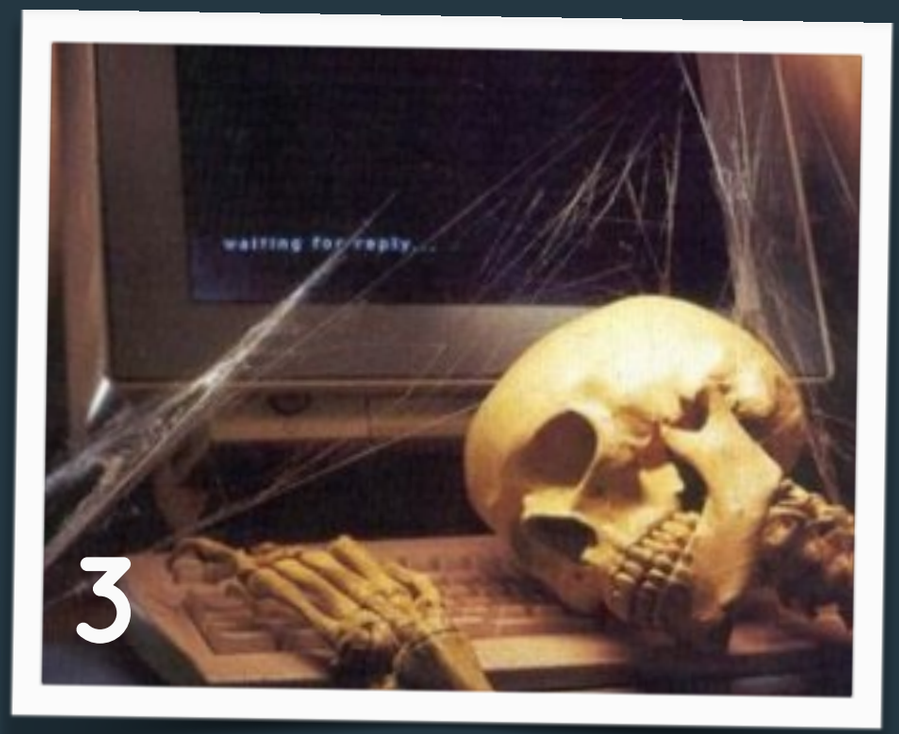
Stochastic

Knowledge

No Knowledge







concepts

0. Global Search

1. Search Space ‘fitness landscape’

2. Candidate Solution

3. Fitness Function (heuristic)

HELLO WORLD

FOR AI

THE PROBLEM

Write a program that will print “hello world” to standard out in these four cases:

1. Deterministic algorithm with knowledge (informed)
2. Stochastic algorithm with knowledge (informed)
3. Deterministic algorithm with no knowledge (uninformed)
4. Stochastic algorithm with no knowledge (uninformed)

1 DETERMINISTIC & KNOWLEDGE

```
puts "hello world"
```

```
if rand() < 0.95      # 5% failure rate
  puts "hello world"
else
  fail "a statistically unlikely death"
end
```

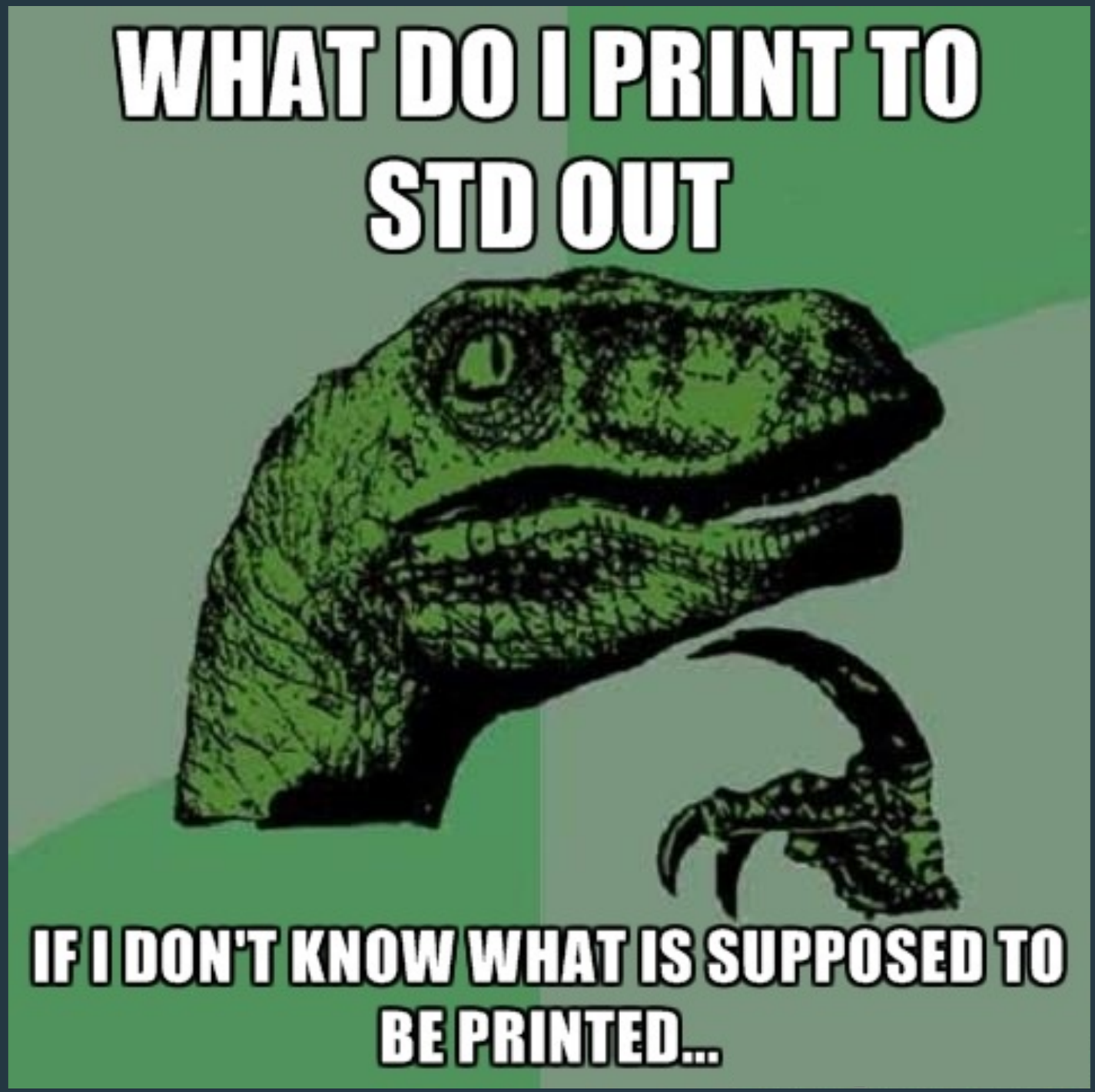
NO KNOWLEDGE

(uninformed)

Write a program that will print *something* to standard out, *without knowing* exactly what it is that should be outputted...

NO KNOWLEDGE

(uninformed)

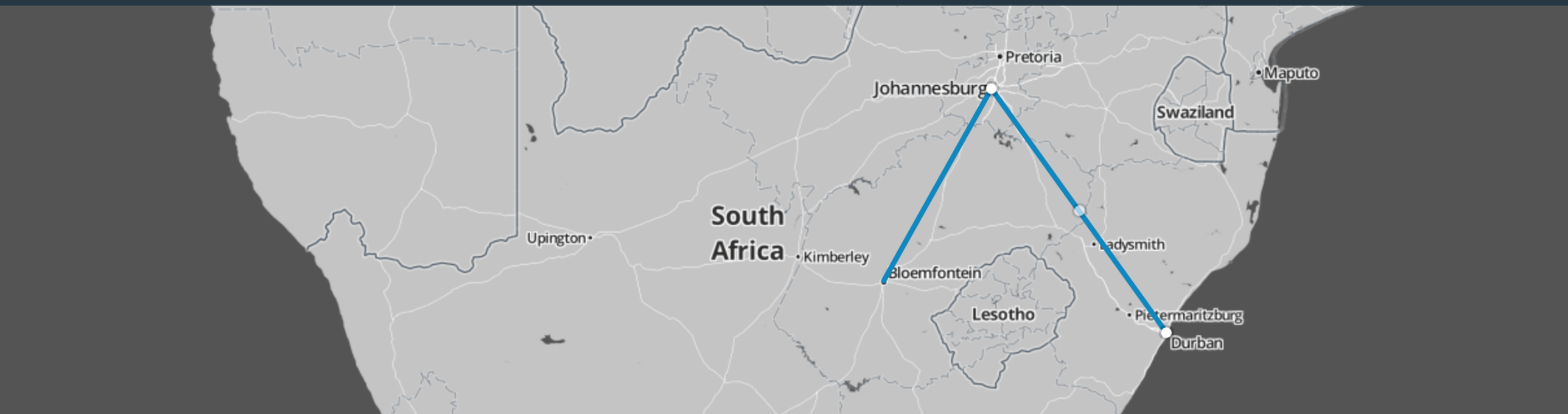


NO KNOWLEDGE

(uninformed)

Define: **(3) heuristic**

Computing proceeding to a solution by trial and error or by rules that are only loosely defined.



NO KNOWLEDGE

(uninformed)

Define: **(3) fitness function**

A single real value that reflects to some accuracy how close a solution is from being correct

Sample solutions

```
"hello steve" #=> half way there
"hello Mr. T" #=> half way there
" ello world" #=> almost there!
```

NO KNOWLEDGE

(uninformed)

Lexical Distance

```
"hello steve" #=> 21
"hello mr  t" #=> 34
" ello world" #=> 6
"iello world" #=> 1
"hello world" #=> 0
```

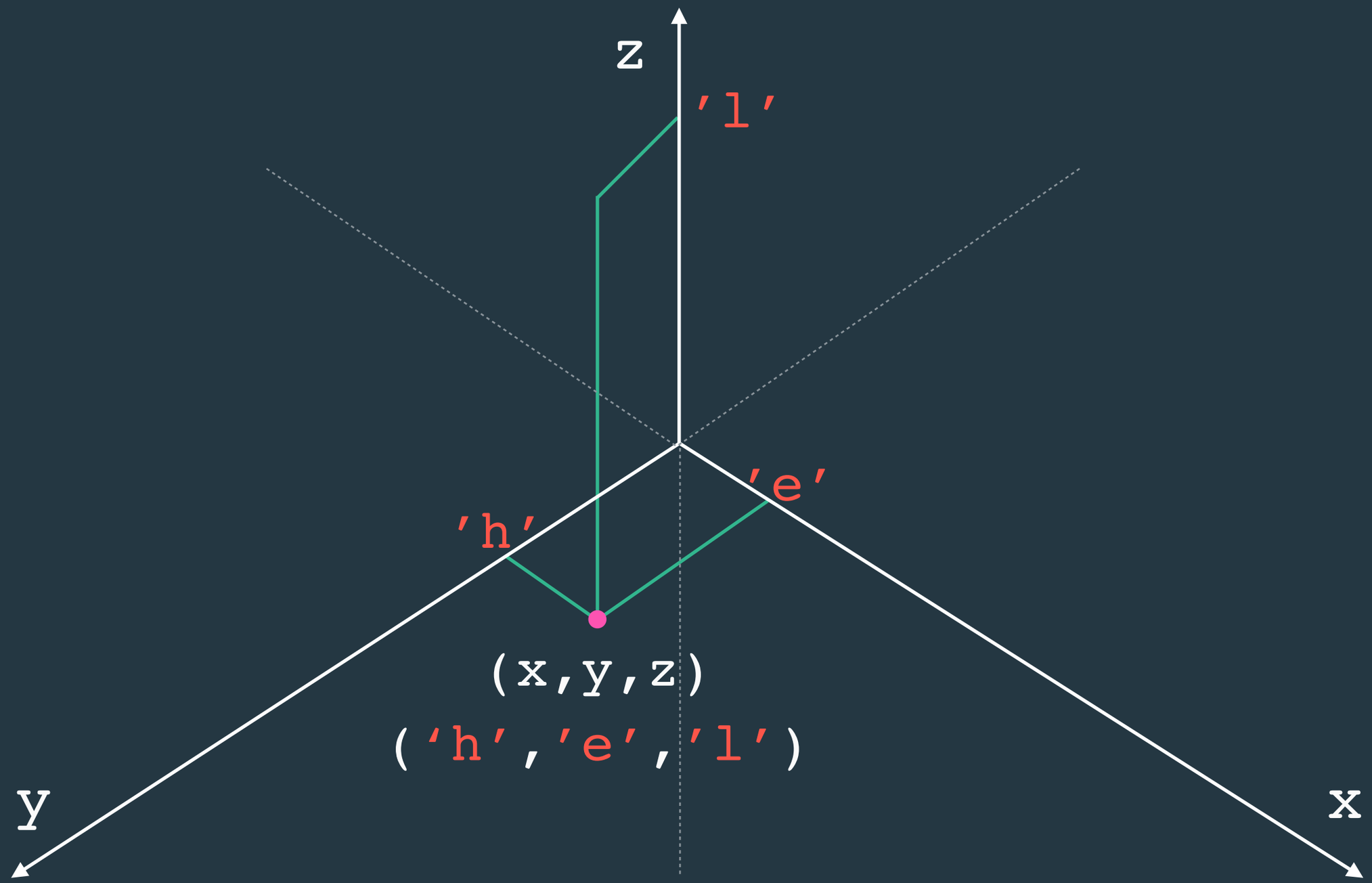
(2) Candidate solution

```
ALPHABET = ('a'..'z').to_a + [" "]

def distance a, b
  return nil unless a.length == b.length
  a.chars.each_index.inject(0) do |sum, i|
    sum + char_distance(a[i], b[i])
  end
end

def char_distance ai, bi
  ia = ALPHABET.index(ai)
  ib = ALPHABET.index(bi)
  ia = (ia - ALPHABET.length).abs if ia > (ALPHABET.length/2)
  ib = (ib - ALPHABET.length).abs if ib > (ALPHABET.length/2)
  (ia - ib).abs
end
```

3 DETERMINISTIC & NO KNOWLEDGE



(1) Search space

3 DETERMINISTIC & NO KNOWLEDGE

```
require_relative 'distance'
# D = domain (all possible values)
D = ALPHA
#       "hello world".length = 11 chars
best = "aaaaaaaaaaa"
D.each do |a|
  D.each do |b|
    D.each do |c|
      D.each do |d|
        D.each do |e|
          D.each do |f|
            D.each do |g|
              D.each do |h|
                D.each do |i|
                  D.each do |j|                                    ##
                    D.each do |k|                                   ###
                      candidate = "#{a}#{b}#{c}#{d}#{e}#{f}#{g}#{h}#{i}#{j}#{k}"
                      best = distance(candidate) < distance(best) ? candidate : best
                      puts "solution = #{best}" and exit 0 if distance(best) == 0
                    end                                             ###
                  end                                               ##
                end
              end
            end
          end
        end
      end
    end
  end
end
```

“Arrow solution”

3 DETERMINISTIC & NO KNOWLEDGE

Combinations = 27^11 = 5.5590606e+15 for brute force

Combinations = 27*11 = 297 for optimising each dimension independently (since the problem is ‘separable’)

Problem: takes far too long for a simple hello world, imagine something that required actual computing power...

4 STOCHASTIC & NO KNOWLEDGE

Clearly, we need an intelligent solution...

(0) Global Search!!!!!!!!!!!!!!!!

concepts

RECAP

0. Global Search

1. Search Space ‘fitness landscape’

2. Candidate Solution

3. Fitness Function (heuristic)

EVOLVING A SOLUTION TO HELLO WORLD

WITH NATURAL SELECTION

"The sequencing is just enormously complex"

~ Kevin

"But, didn't you write it?"

~ Sam

"Ha, some of it. The rest is just, beyond me"

~ Kevin

Evolutionary Computation

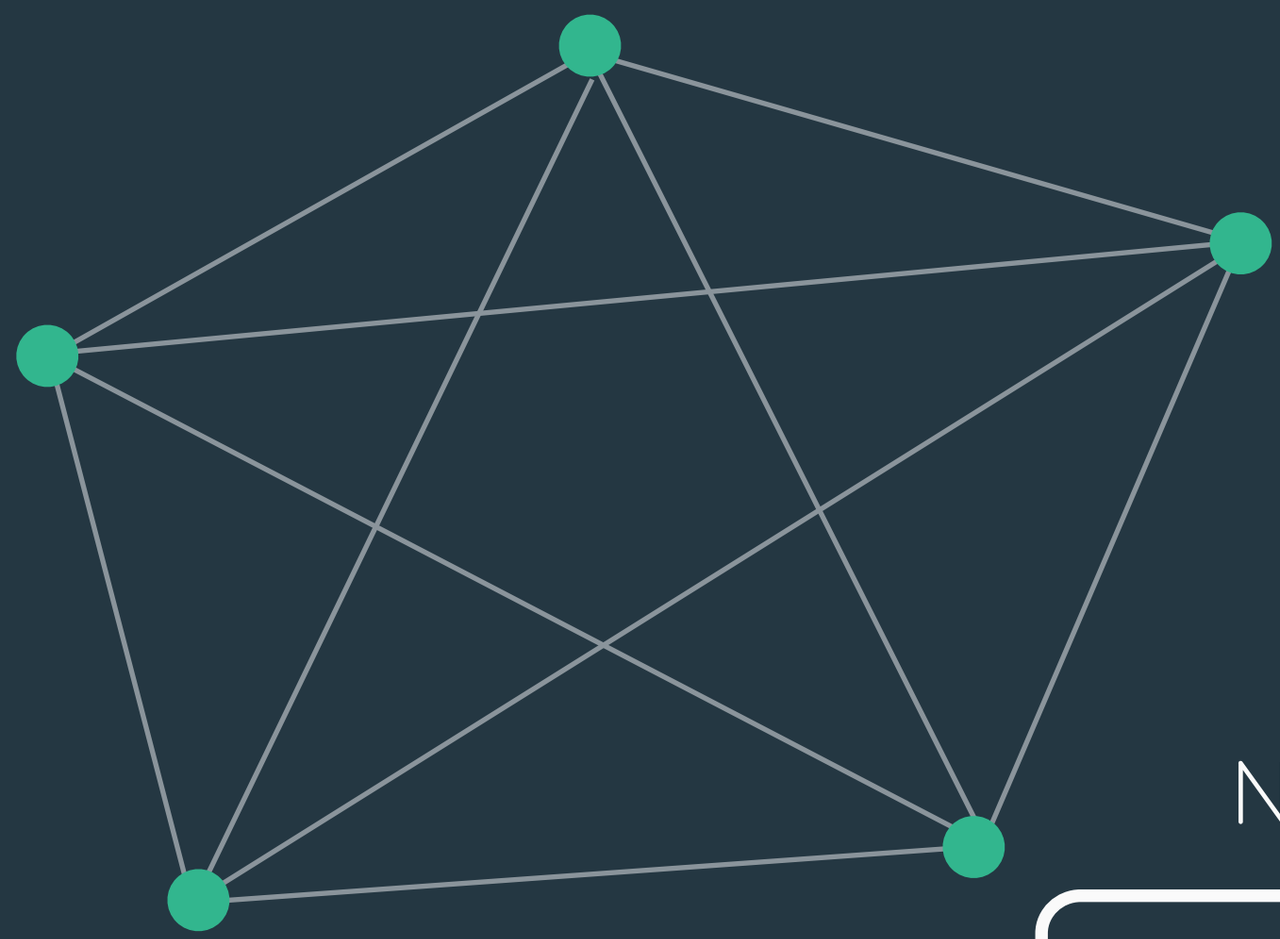
EC

Swarm Intelligence

SI

FS

Fuzzy Systems



NN

Neural Networks

AIS

Artificial Immune Systems

EC

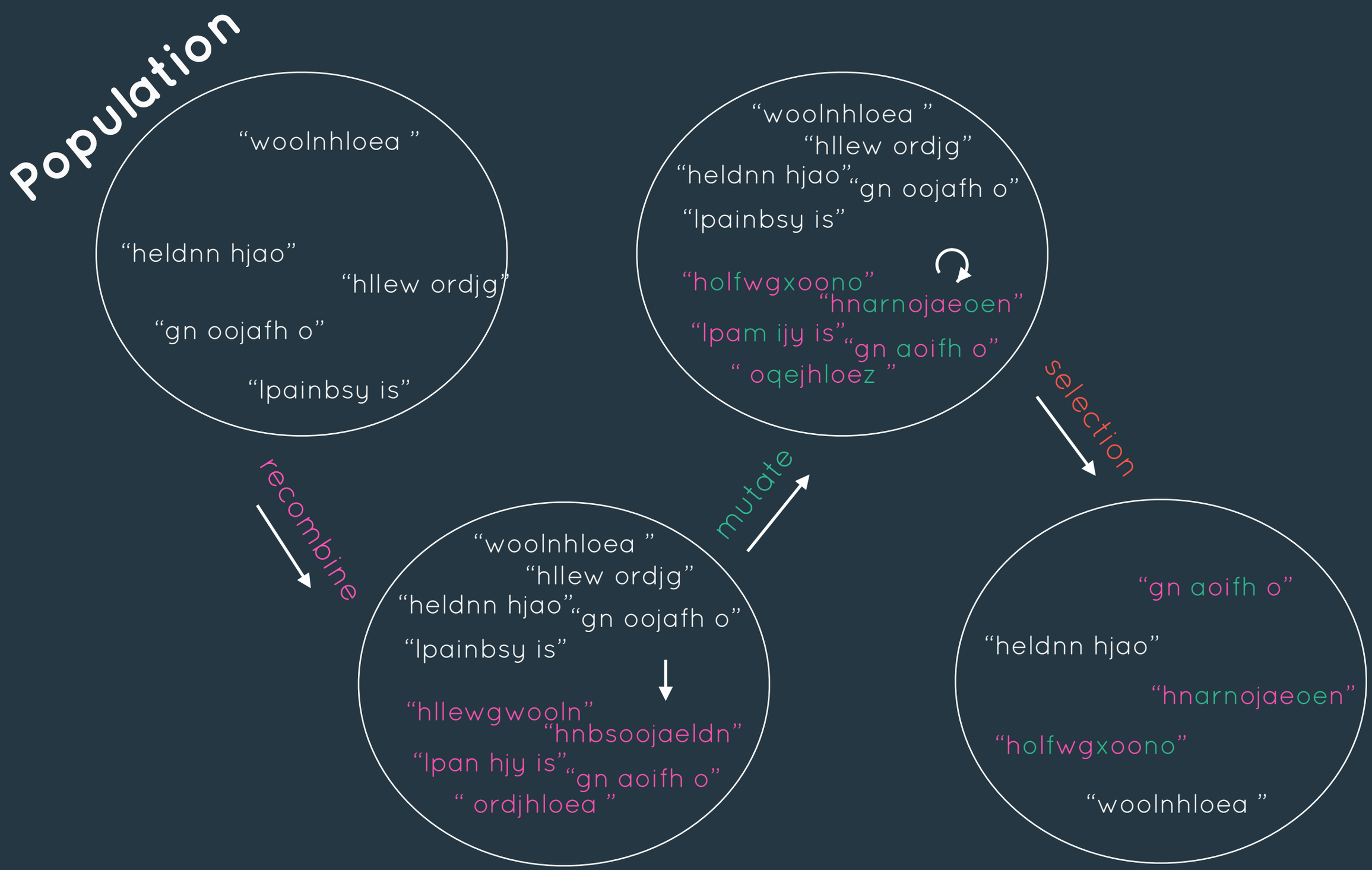
GA: “Genetic Algorithm”

Algorithmic model
developed to simulate
biological evolution.

Jean-Baptiste Lamarck's theory of evolution was that of heredity, i.e. the inheritance of acquired traits. The main idea is that individuals adapt during their lifetimes, and transmit their traits to their offspring

Charles Darwin ~ Individuals with the "best" characteristics (traits/genes) are more likely to survive and to reproduce, and those characteristics will be passed on to their offspring. These desirable characteristics are inherited by the following generations, and (over time) become dominant among the population.

Genetic Algorithm Overview



Genetic Algorithm Overview

```
@individuals = array_of_individuals
generations.times do |generation|
  # Calculate fitness
  @individuals.each{ |individual| individual.calc_fitness }

  # Recombine (copulate)
  offspring = @crossover_strategy.crossover(@individuals)

  # Mutate offspring
  mutated_offspring = @mutation_strategy.mutate(offspring, problem)

  # Select next generation
  generation_pool = (@individuals + mutated_offspring)
  new_population = @selection_strategy.select(generation_pool, @population)

  # Ensure elitism
  @individuals = elitism(new_population, generation_pool)
end

return best_solution_found
```

"Heredity"

Crossover / Recombination

h	e	n	n	o		x	o	r	d	l

g	n	l	b	y	b	w	r	o	l	d

+

0	1	1	0	1	0	0	0	1	1	0

=

h	n	l	n	y		x	o	o	l	l

+

=

"Diversity"

Mutation

h	e	n	n	o		x	o	r	d	l

+

0	1	-1	0	1	0	0	0	-1	-1	0

=

h	f	m	n	p		x	o	q	c	l

Selection

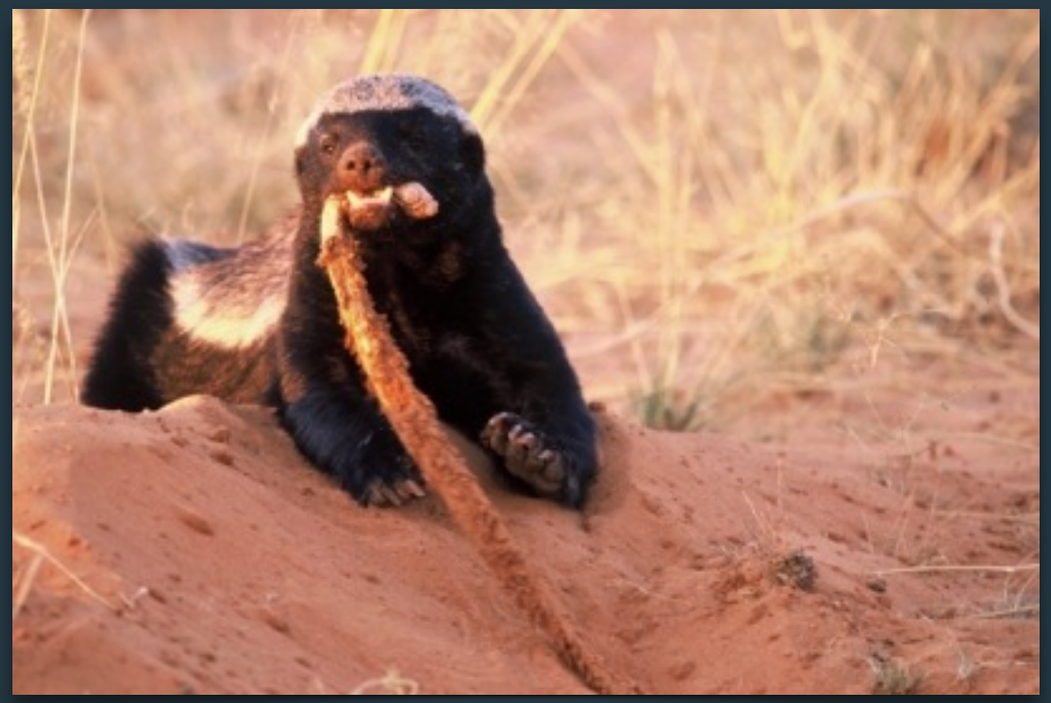
```
def select entities, population
  # Rank by fitness
  fitnesses = entities.map{ |e| e.fitness }
  sum = fitnesses.reduce(:+).to_f
  normalized_ranks = ranks.map{ |r| r.to_f/sum }

  # Calculate, form array of tuples to keep a reference to rank & entity
  tuples = []
  entities.each_with_index{ |e, i| tuples << [e, normalized_ranks[i]] }
  tuples.sort!{ |a,b| a[1] <=> b[1] }

  # Select probabilistically based on rank
  size = population
  selected = []
  while selected.length < size
    tuples.each_with_index do |tuple, index|
      if rand() < tuple[1]
        selected << tuples[index][0]
      end
    end
  end
  selected
end
```

Elitism

Make sure the best
individual(s) survive to the
next generation



badassoftheweek.com

DEMO

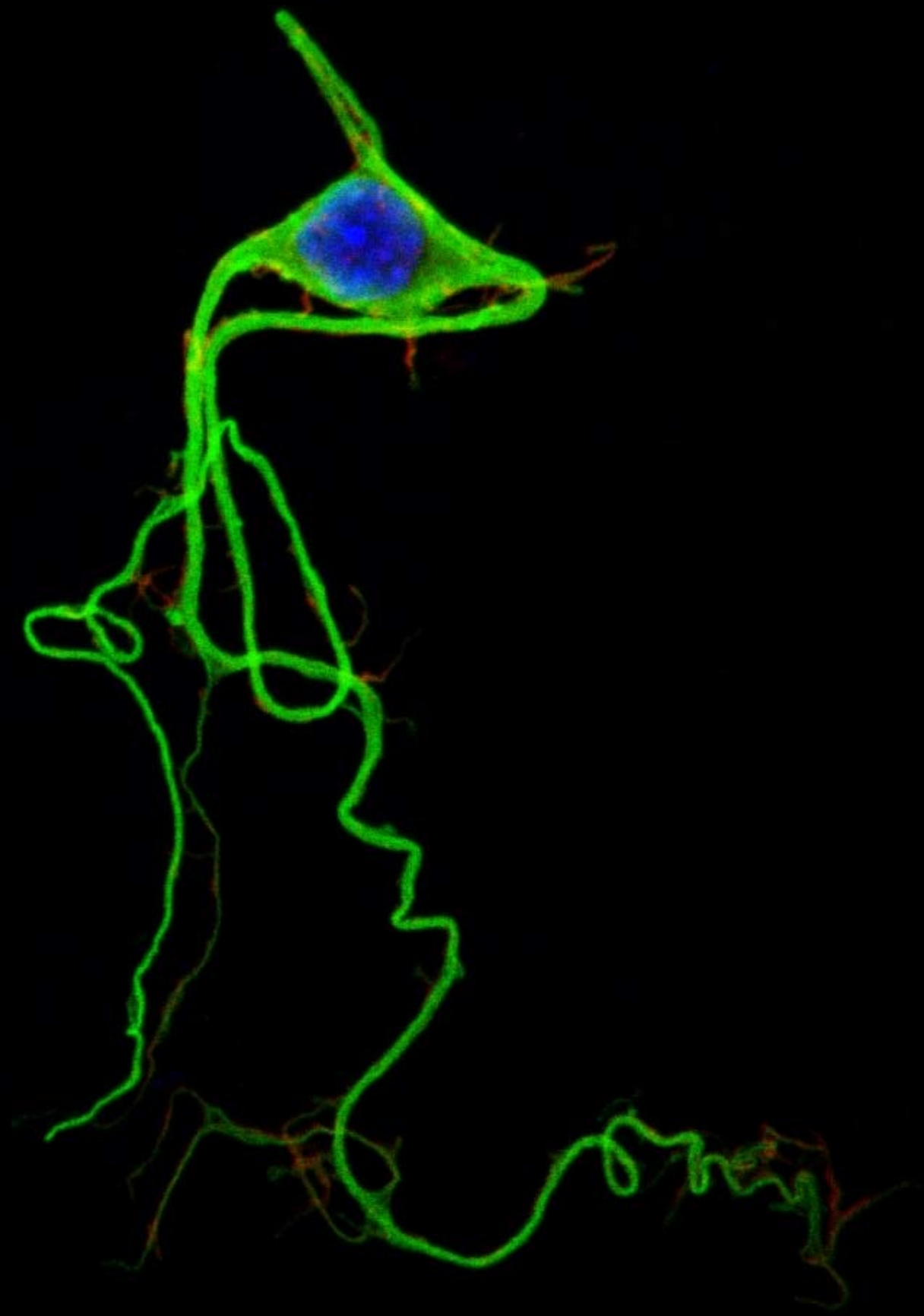
QUESTIONS

BREAK

DIAGNOSING CANCER

WITH NN & PSO

Artificial Neural Networks



~10b neurons
each connected
to thousands
others
via synapse

Learn by thickening

Synapse are elastic

Excited /
Inhibited
States

Evolutionary Computation

EC

Swarm Intelligence

SI

FS

Fuzzy Systems

Neural Networks

NN

AIS

Artificial Immune Systems

Artificial model developed to approximate the generalization of knowledge & discovery.

FFNN: “Feed Forward Neural Network”

Neural Networks

NN

Classification Problem

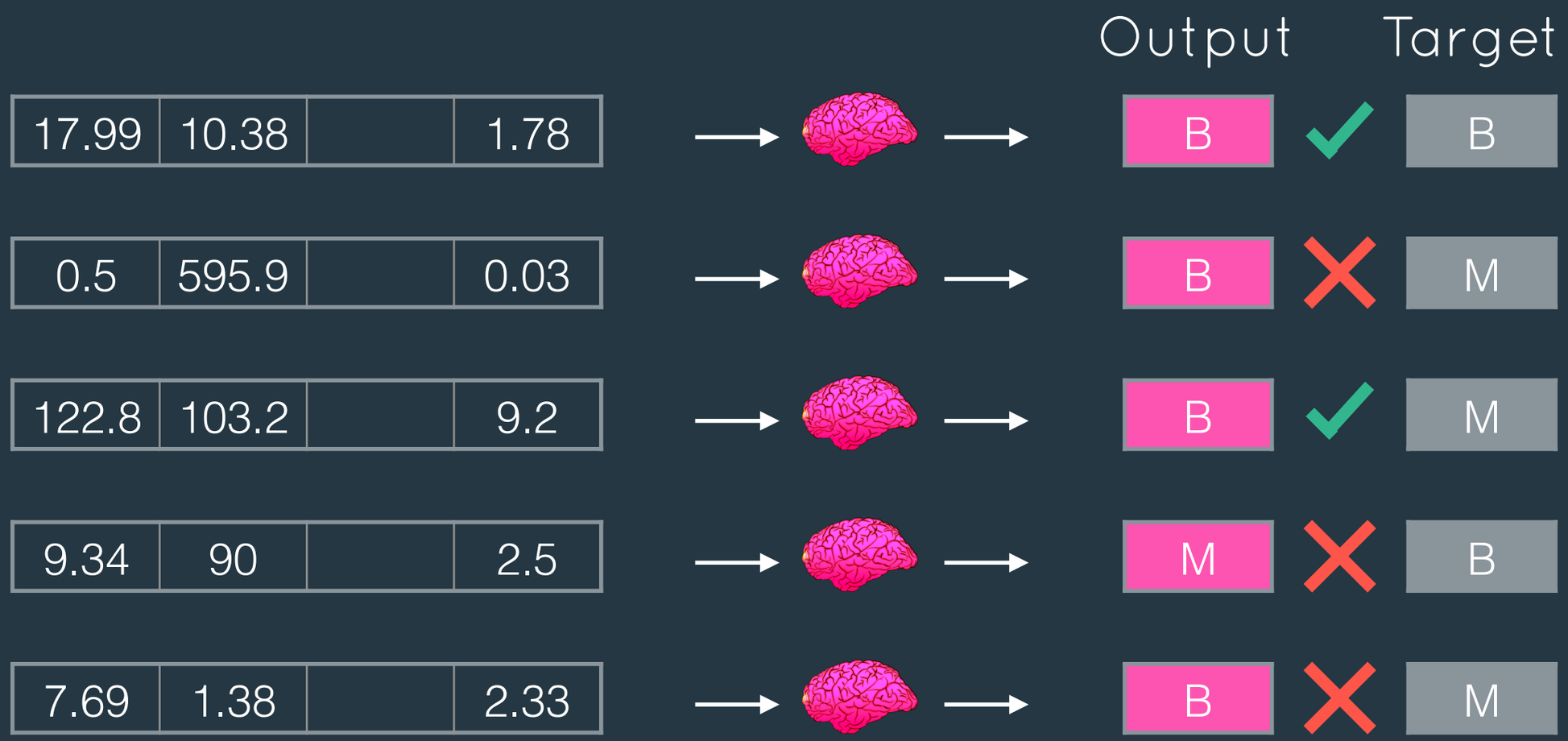
Each row is 1 persons tissue measurements

Attr1	Attr2	…	AttrN	Class
17.99	10.38		1.78	B
0.5	595.9		0.03	M
122.8	103.2		9.2	M
9.34	90		2.5	B
				…

Input vector
“Relevant data”

Classfication
“Target vector”

Classification Problem Training



Classification Problem Evaluation

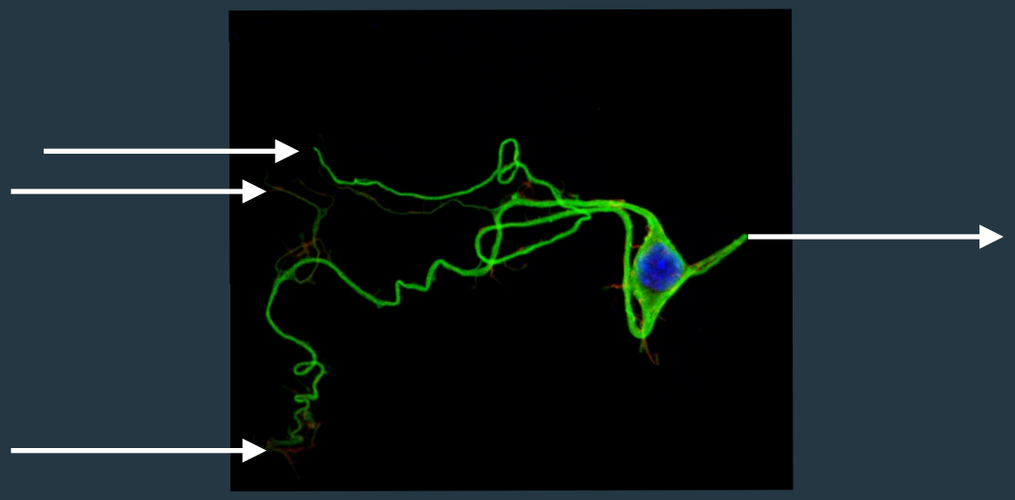
Attr1	Attr2	…	AttrN
17.99	10.38		1.78

Class
B

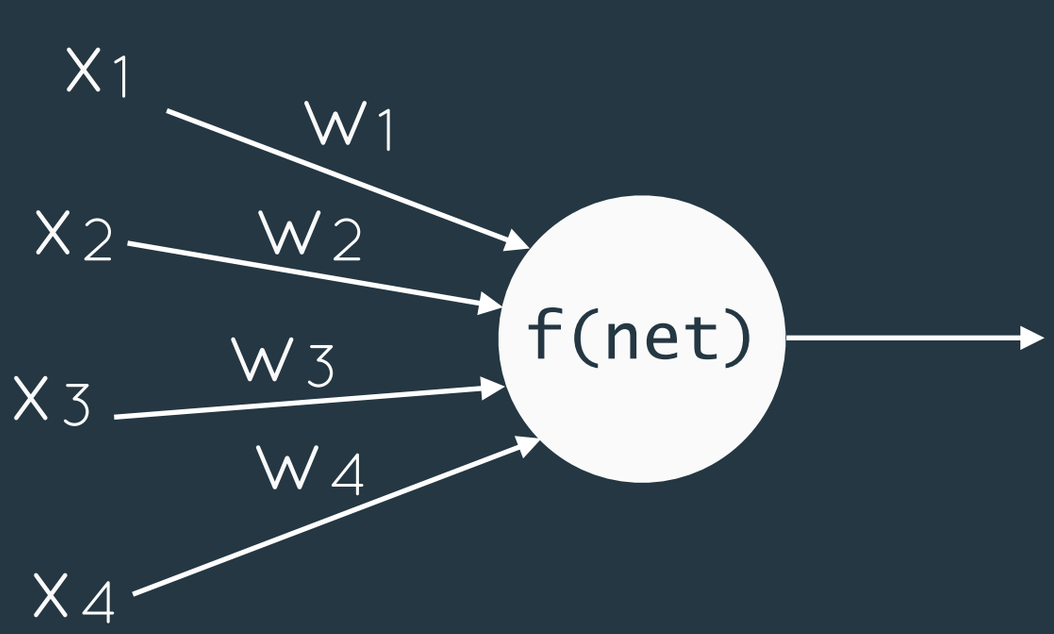
$F_{NN}: R^n \rightarrow \{M,B\}$

(Malignant or Benign)

The Perceptron



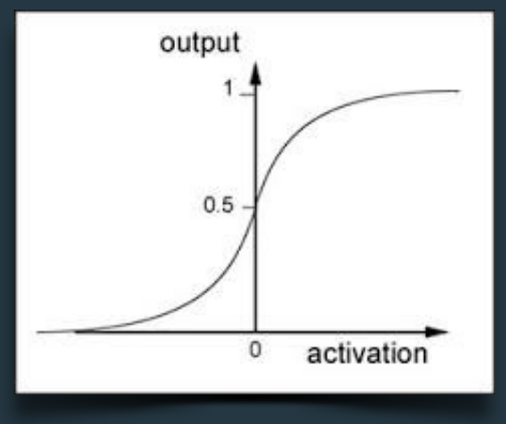
Activation strength



$f(net) = f(x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4)$

f is an activation function:
[step, sigmoid, h tan, linear]

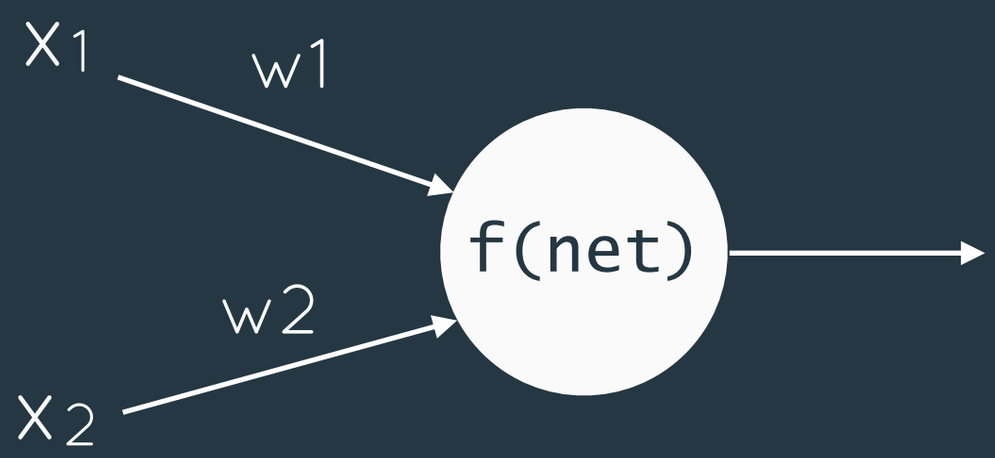
sigmoid:



The Perceptron: OR

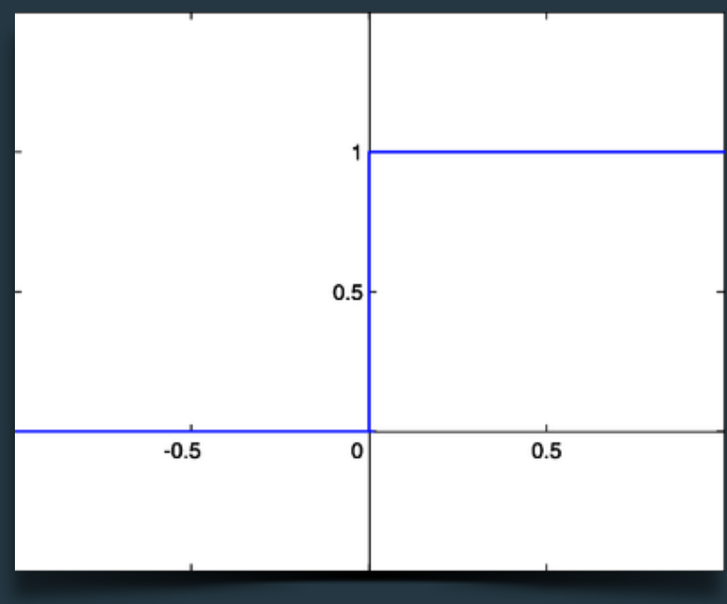
x	x	f(net)
0	0	0
0	1	1
1	0	1
1	1	1

Guess values for x_1 and x_2...



(Step activation)

$$f(net) = f(x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4)$$



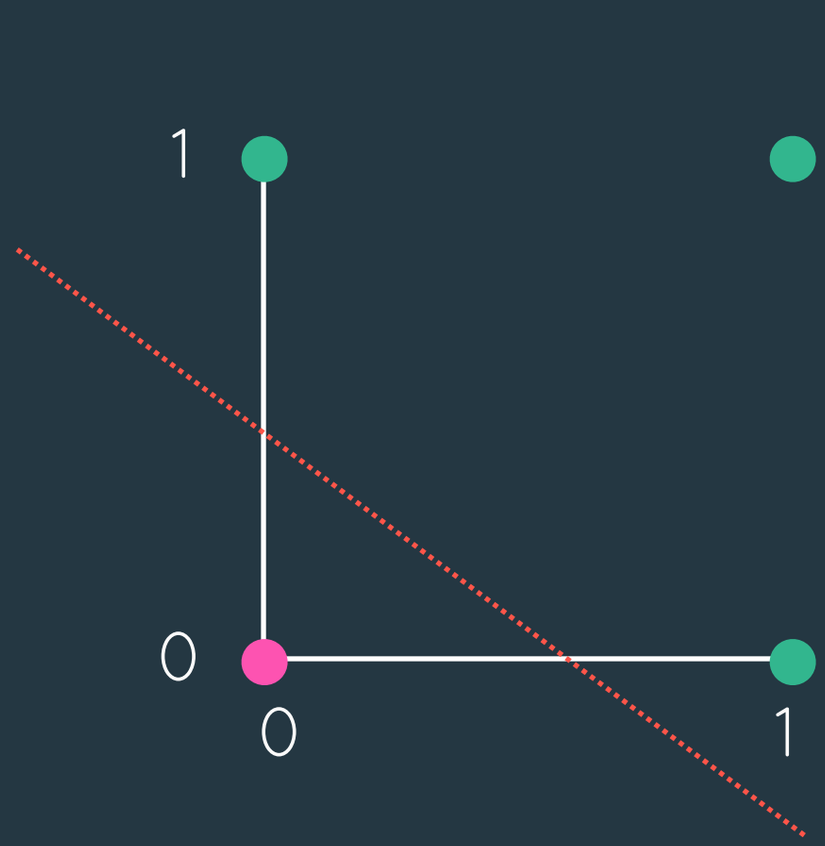
The Perceptron: OR & XOR

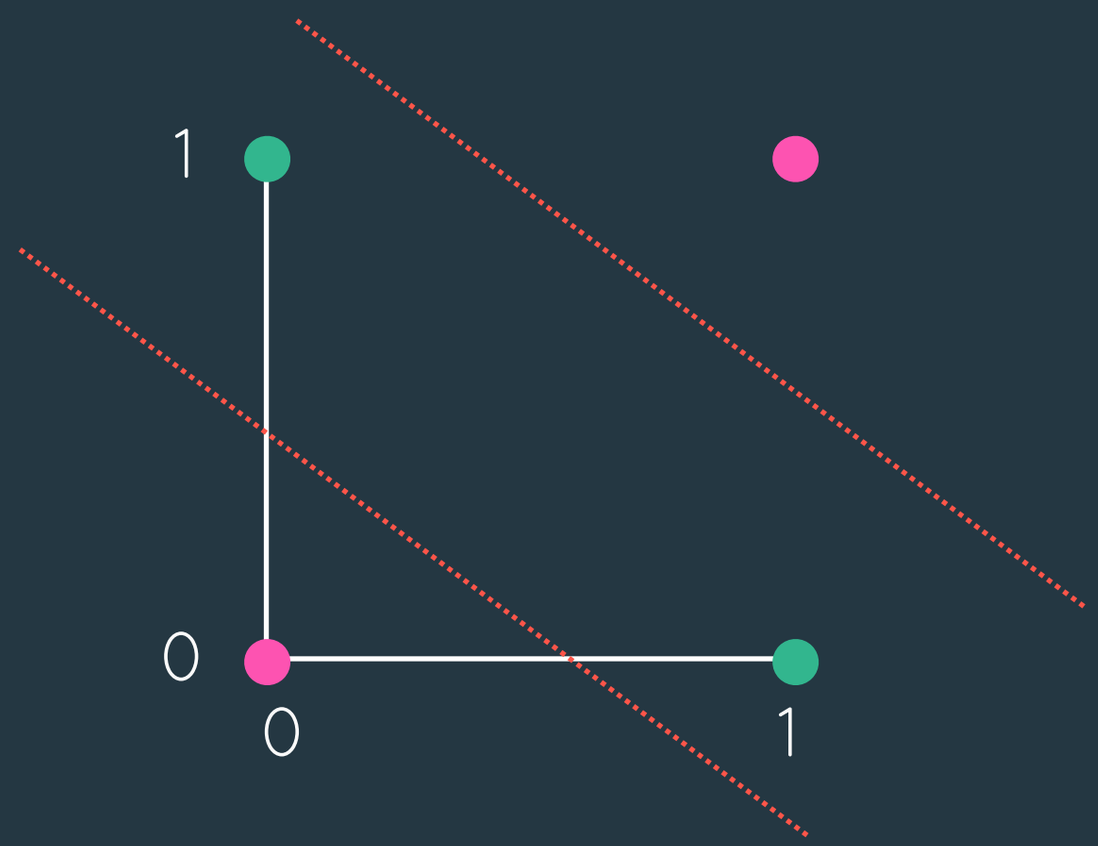
OR: ✓

x	x	f(net)
0	0	0
0	1	1
1	0	1
1	1	1

XOR: ✗

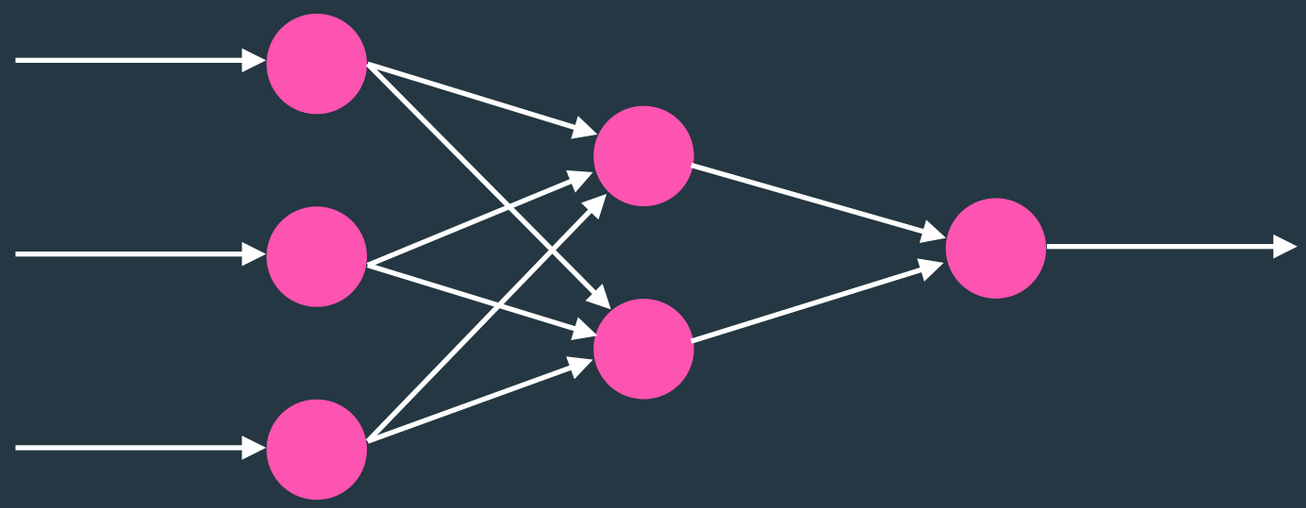
x	x	f(net)
0	0	0
0	1	1
1	0	1
1	1	0





The 3 Layer FFNN

We can compose Perceptrons (which have activation functions) to create a higher order function from them that has more "information capacity".



The 3 Layer FFNN Output

Output

F_{NN}: R^n -> {M,B}

meaning…

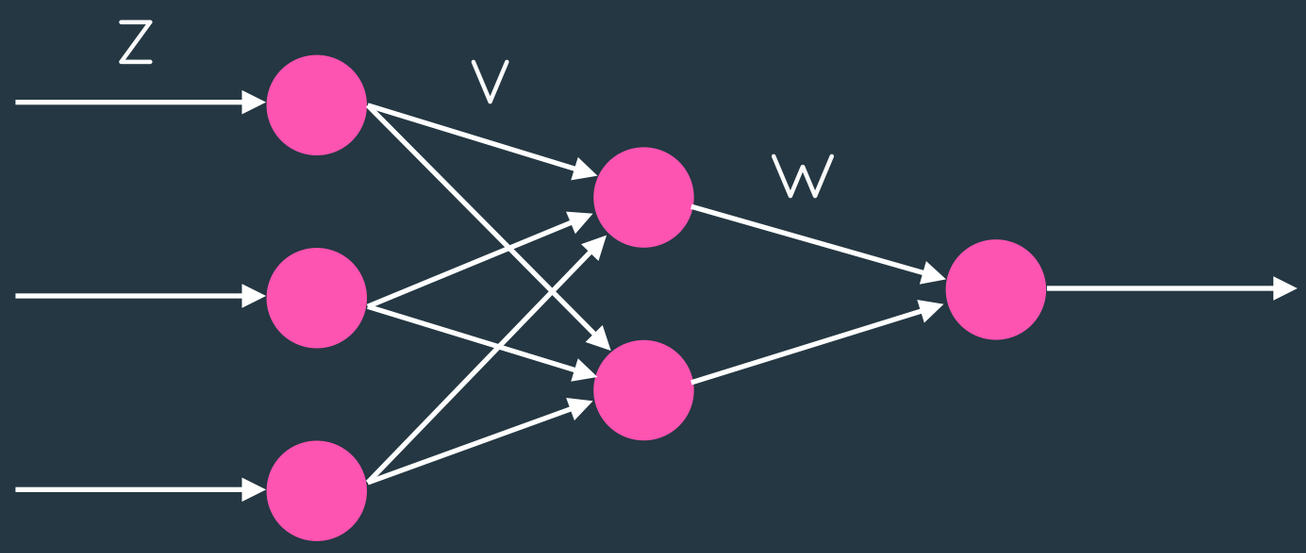
$output = f_{out}(net)$

$= f_{out}(sum(wf_{middle}(net)))$

$= f_{out}(sum(wf_{middle}(sum(vz))))$

Fitness
(accuracy)

```
error = patterns.each do |pattern|
  sum(difference(target,output))
end / patterns.length
```



How do we train it?

1. Gradient descent
2. Simulated annealing
3. RPROP
4. Global search? PSO!

TRAINING OUR NN

WITH A PSO

Particle Swarm Optimization

Simple individuals,
working socially to
exhibit complex,
emergent,
behaviour

Evolutionary Computation

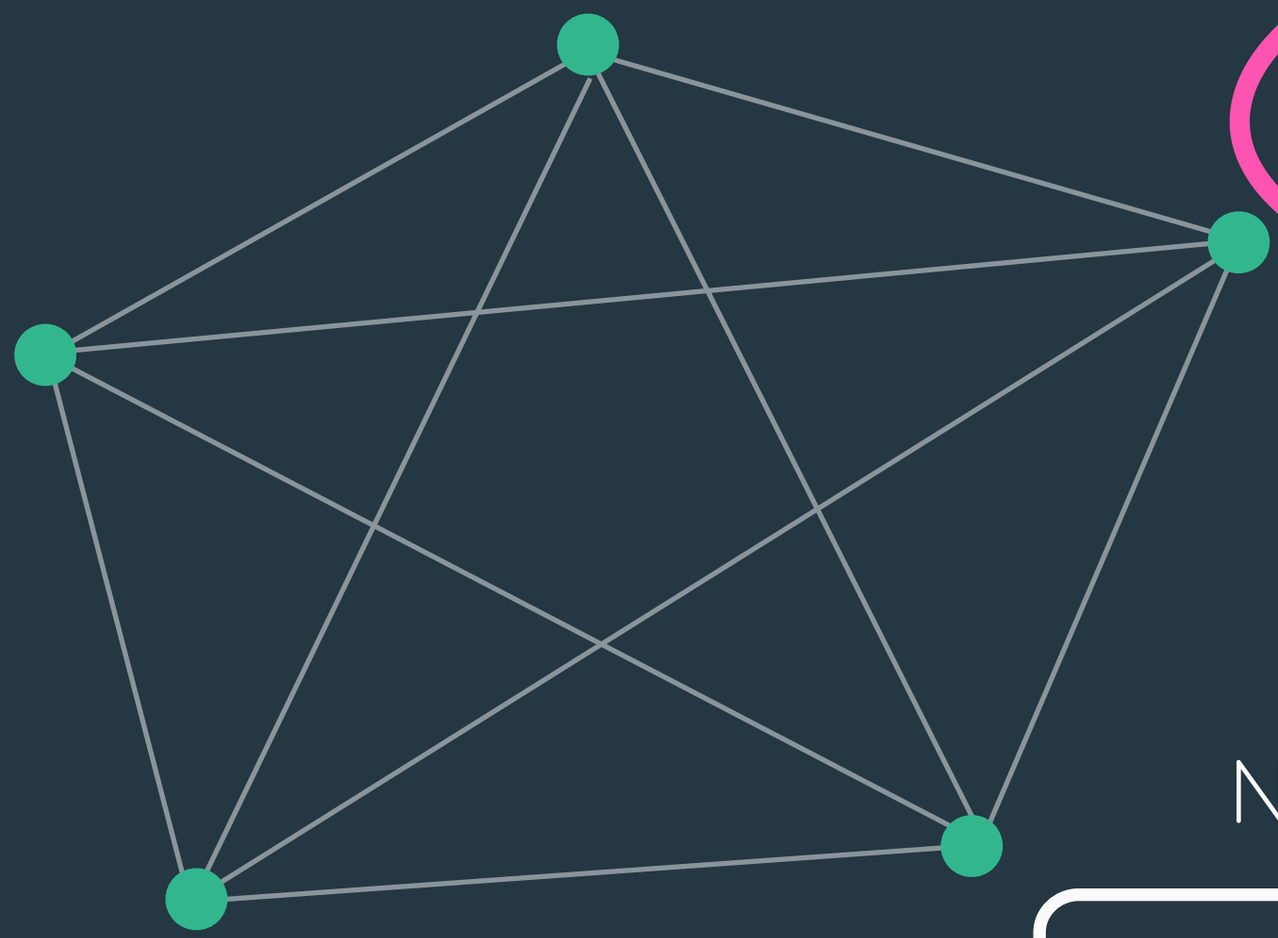
EC

Swarm Intelligence

SI

FS

Fuzzy Systems



Neural Networks

NN

AIS

Artificial Immune Systems

Swarm
Intelligence

SI

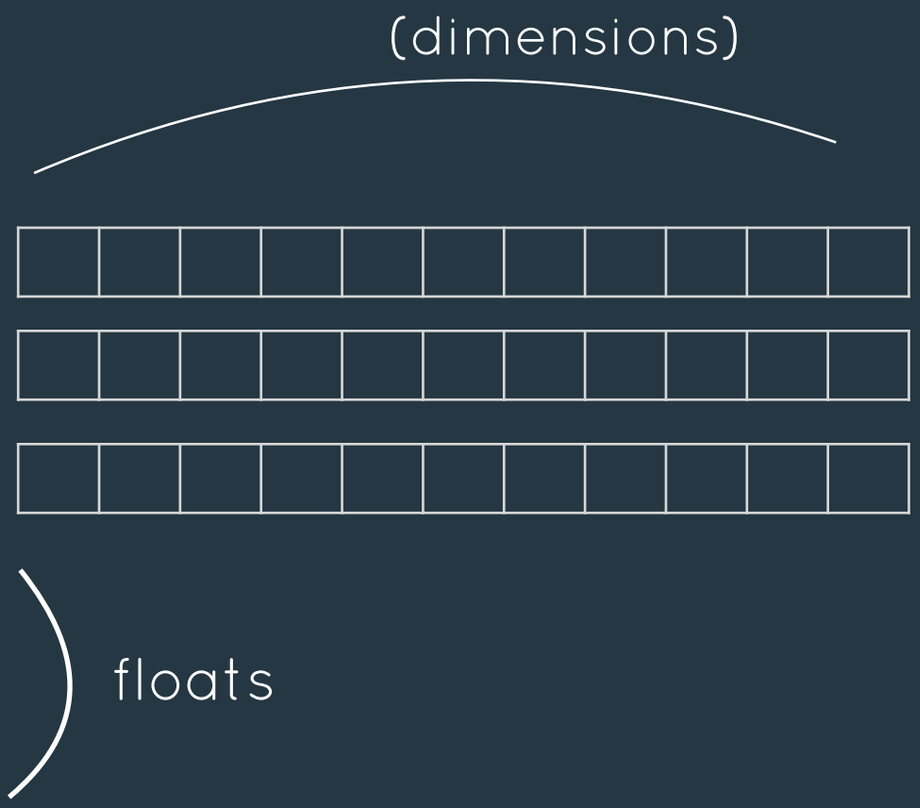
PSO: “Particle Swarm Optimizer”

Algorithmic model
developed to simulate
complex emergent
behaviour of swarms.

PSO Objects

Particle:

velocity
position
personal best position
inertia
social_weight
cognitive_weight



Swarm:

[particle]
global best position

PSO Algorithm

```
initialize_swarm
@iterations.times do |i|
  @swarm.each do |particle|
    particle.update_velocity
    particle.update_position
    particle.update_pbest
    update_gbest particle
  end
end
```

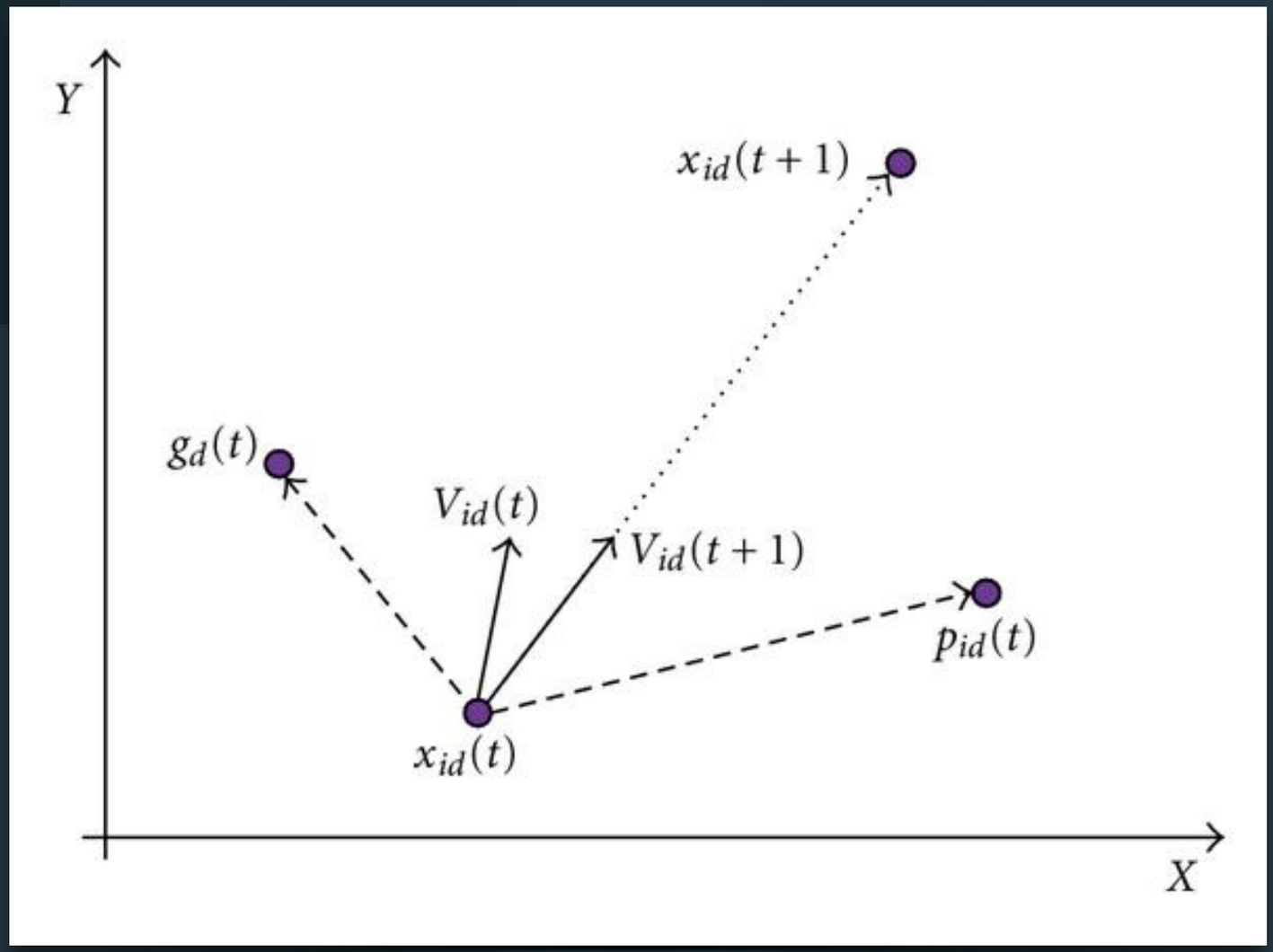
PSO Algorithm: Velocity & Position Updates

Position update

```
position = position + v
```

Velocity update

```
v = wv + c₁r₁(pbest - position)
       + c₂r₂(gbest - position)
```



Ones Problem

Use a PSO to generate a
vector of float values where
each value equals exactly 1

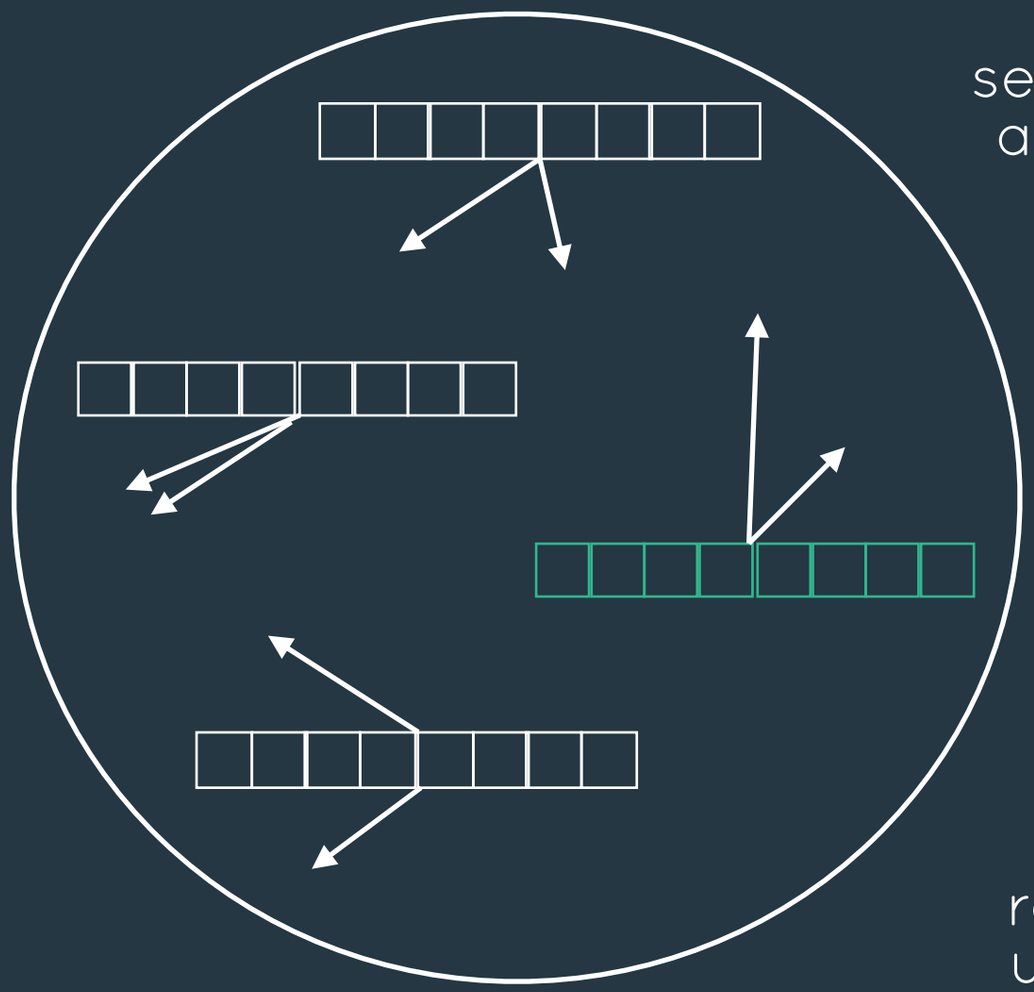
The solution: [1.0, 1.0, 1.0, 1.0, 1.0]

DEMO

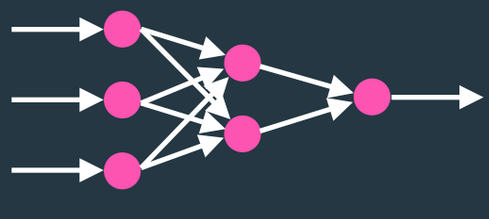
QUESTIONS

Bringing it together

Swarm

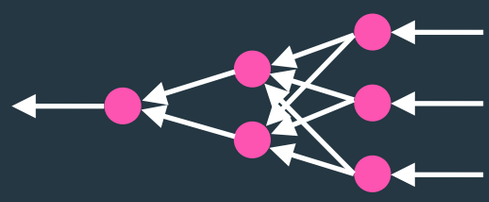


set solution vector
as weights in NN



How well does
it classify?

Update
personal best



report fitness to
update gbest in
the swarm

DEMO

QUESTIONS

fin.