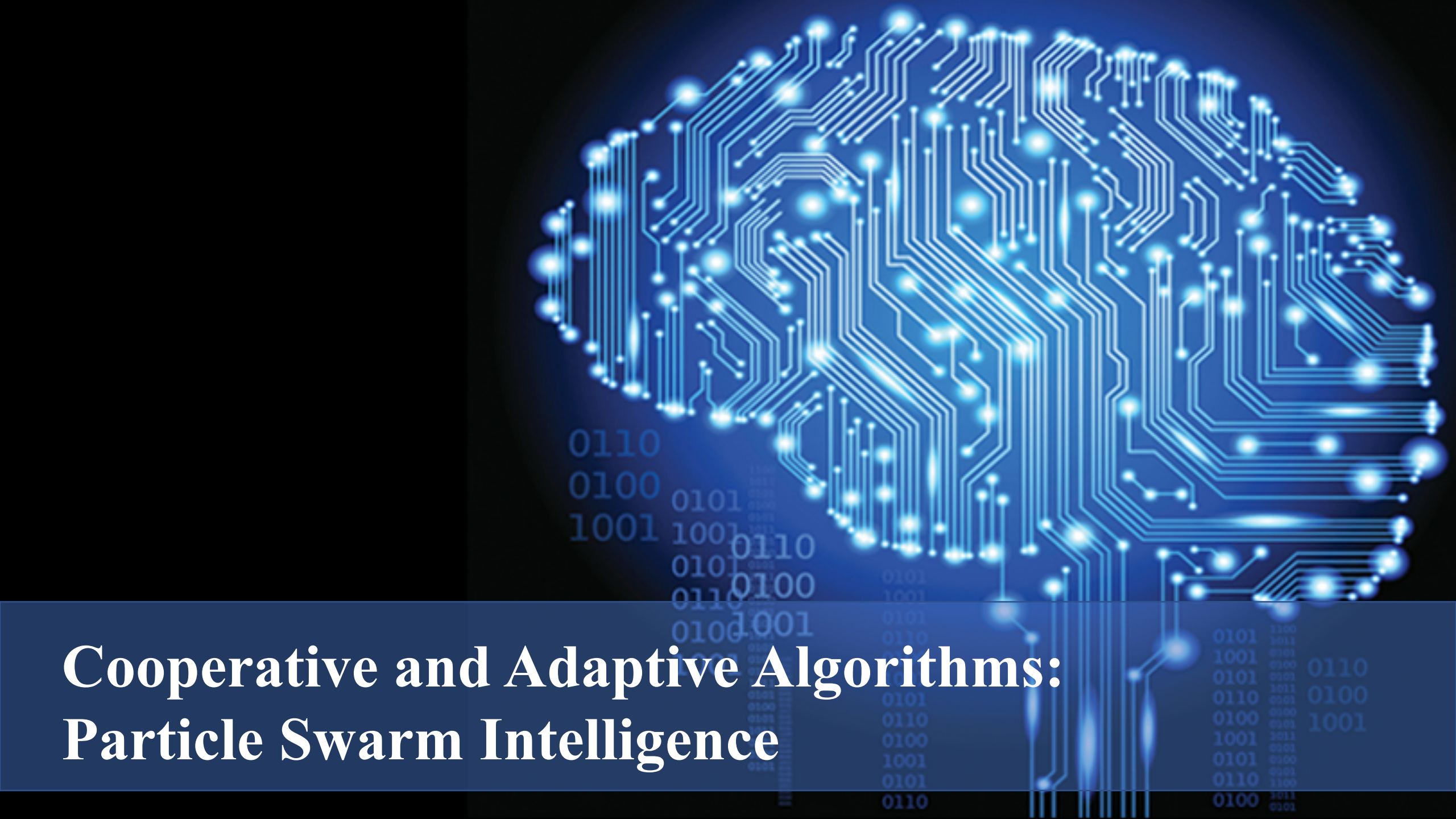


Cooperative and Adaptive Algorithms: Particle Swarm Intelligence

PSO Function Minimization Example

In this section, the PSO algorithm is explained in details to show how the particles are moved and also to show the influence of each parameter on the PSO algorithm. Assume the PSO algorithm has five particles (p^i, i = 1,…, 5), and the initial positions of the particles are presented in Table 1. The velocities of all particles are initialized to be zeros. Moreover, the initial best solutions of all particles are set to 1000 as shown in Table 1. In this example, De Jong function (see Equation 3) was used as a fitness function.

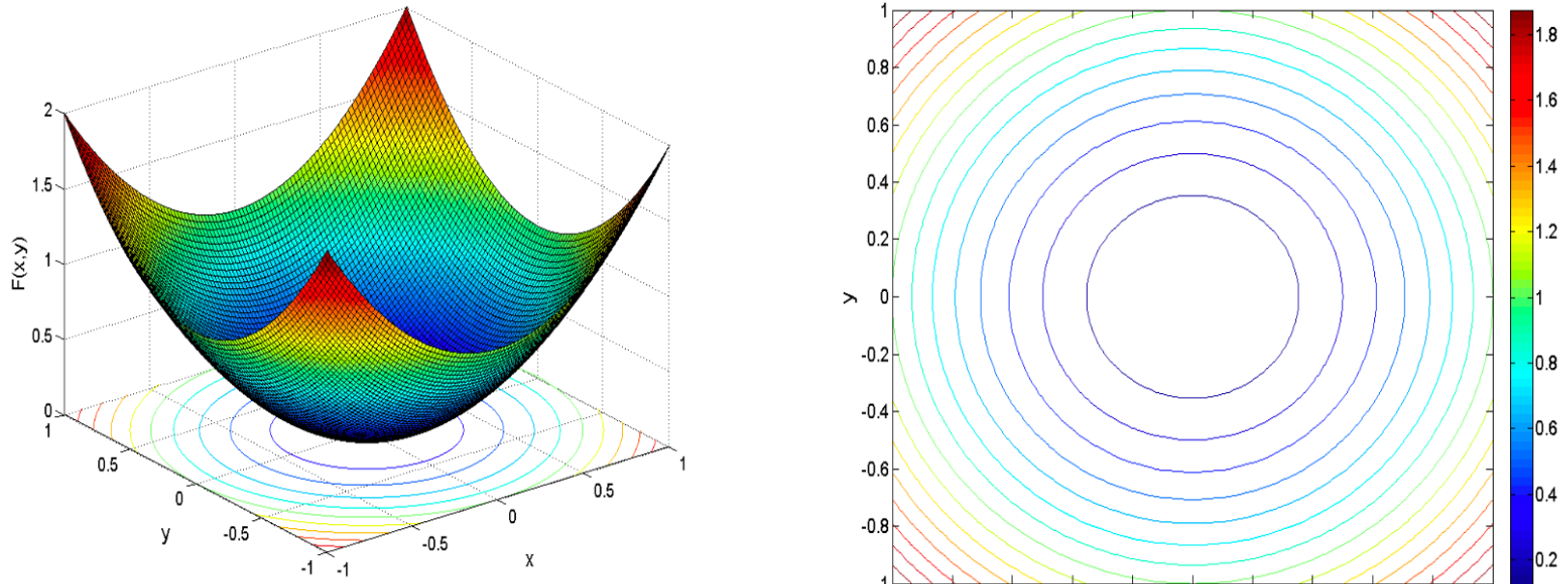
$$\min F(x,y) = x^2 + y^2 \quad (3)$$

where x and y are the dimensions of the problem. Figure 2 shows the surface and contour plot of the De Jong function. As shown, the function is a strictly convex function1. Moreover, the optimal solution is zero and it is found at the origin. Moreover, the lower and upper boundaries of both x and y dimensions were -1 and 1, respectively. Moreover, in PSO algorithm, the inertia (w) was 0.3, and the values of the cognitive and social constants were as follows, $C_1 = 2$ and $C_2 = 2$.

TABLE 1: Initial positions, velocity, and best positions of all particles.

Particle No.	Initial Positions		Velocity		Best Solution	Best Position		Fitness Value
	x	y	x	y		x	y	
P_1	1	1	0	0	1000	-	-	2
P_2	-1	1	0	0	1000	-	-	2
P_3	0.5	-0.5	0	0	1000	-	-	0.5
P_4	1	-1	0	0	1000	-	-	2
P_5	0.25	0.25	0	0	1000	-	-	0.125

In this example, PSO iteratively searches for the optimal solution, and in each iteration the movement for each particle was calculated including its position, velocity, and fitness function as follows:



(a)

(b)

Figure 2: The surface and contour plot of De Jong function in Equation 3, (a) Surface (b) Contour plot.

First Iteration: The first step in this iteration was to calculate the fitness value for all particles, and if the fitness value of any particle ($F(p^i)$) was lower than the corresponding previous best position (p^i), then save the position of this particle. As shown from Tables 1, the first particle was located at (1, 1); hence, the fitness value is 12 + 12 = 2. Similarly, the fitness values of all particles were calculated as in Table 1. As shown, the fitness values of all particles were lower than the current best solutions; hence, the values of p^i were then updated with the best positions as shown in Table 2, and the best solutions were also updated. Moreover, the fifth particle achieved the best solution, i.e. minimum fitness value; G = (0.25, 0.25). As shown in Table 1, the initial velocities of all particles were zero; thus, the particles will not move in this iteration.

TABLE 2: The positions, velocity and best positions of all particles after the first iteration.

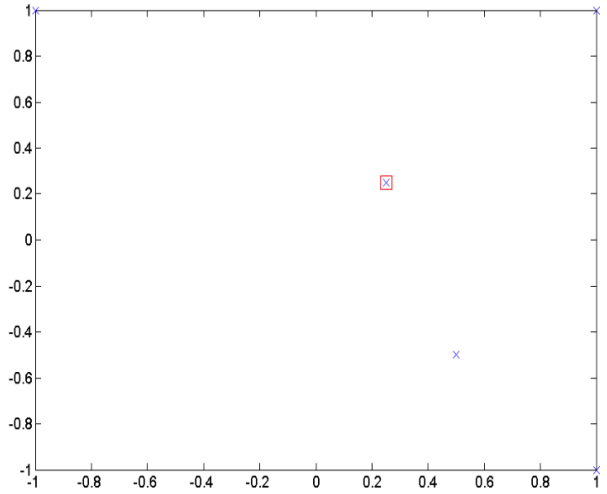
Particle No.	Initial Positions		Velocity		Best Solution	Best Position		Fitness Value
	x	y	x	y		x	y	
P_1	1	1	-0.75	-0.75	2	1	1	2
P_2	-1	1	1.25	-0.75	2	-1	1	2
P_3	0.5	-0.5	-0.25	0.75	0.5	0.5	-0.5	0.5
P_4	1	-1	-0.75	1.25	2	1	-1	2
P_5	0.25	0.25	0	0	0.125	0.25	0.25	0.125

TABLE 3: The positions, velocity and best positions of all particles after the second iteration.

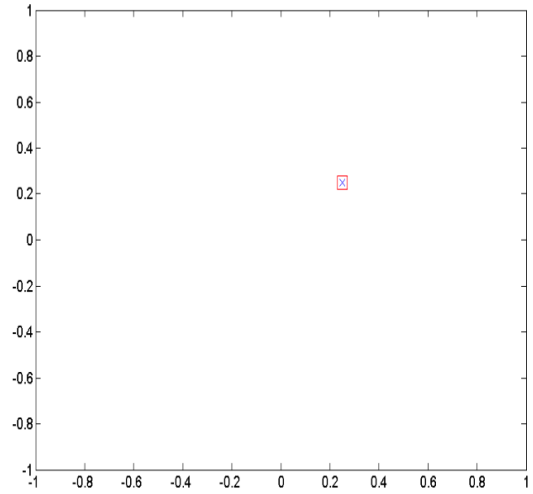
Particle No.	Initial Positions		Velocity		Best Solution	Best Position		Fitness Value
	x	y	x	y		x	y	
P_1	0.25	0.25	-0.3750	-0.3750	2	1	1	0.125
P_2	0.25	0.25	0.6250	-0.3750	2	-1	1	0.125
P_3	0.25	0.25	-0.1250	0.3750	0.5	0.5	-0.5	0.125
P_4	0.25	0.25	-0.3750	0.6250	2	1	-1	0.125
P_5	0.25	0.25	0	0	0.125	0.25	0.25	0.125

The velocity of each particle was calculated as in Equation (2). In this experiment, assume the two random numbers r_1 and r_2 were equal to 0.5. Since the initial velocity of all particles was zero as shown in Table 1 and the previous best positions and the current positions were equal; thus, the first two terms of

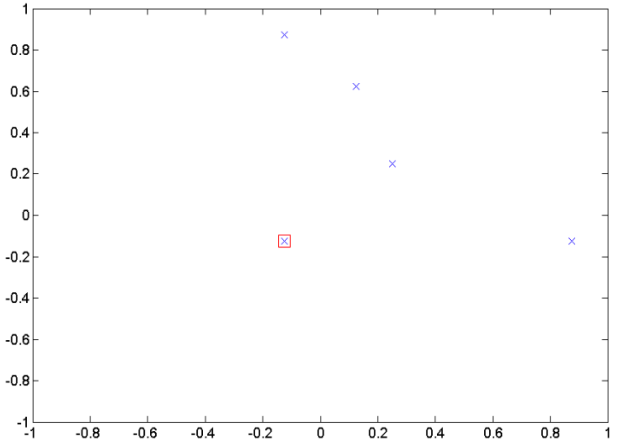
the velocity as shown in Equation (2) were equal to zero. Hence, the velocity in this iteration depends only on the global best position. The velocity of the first particle was calculated as follows, v^i (t+1) = $C_2 r_2$ (G - x^i (t)) = 2 * 0.5 * ((0.25 - 1), (0.25-1)) = (0.75, 0.75), and similarly the velocity of all particles were calculated and the values of all velocities are shown in Table 2. As shown in Table 2, the velocity of the fifth particle was (0, 0) because the fifth particle is the global best solution; hence, it remained at the same position at this iteration. Figure 3 shows the positions of all particles in this iteration. Moreover, Figure 4 shows how the best solution converged to the optimal solution during iterations.



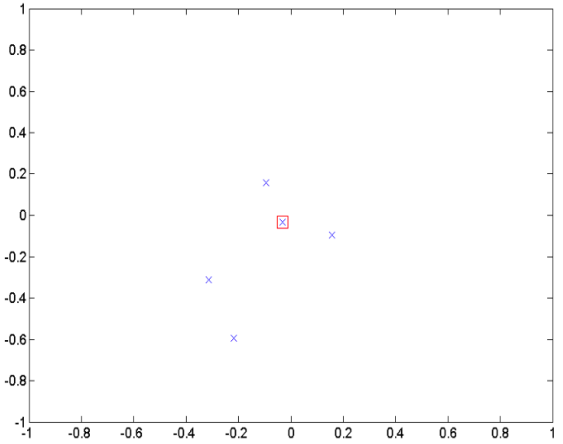
(a) First Iteration



(b) Second Iteration



(c) Third Iteration



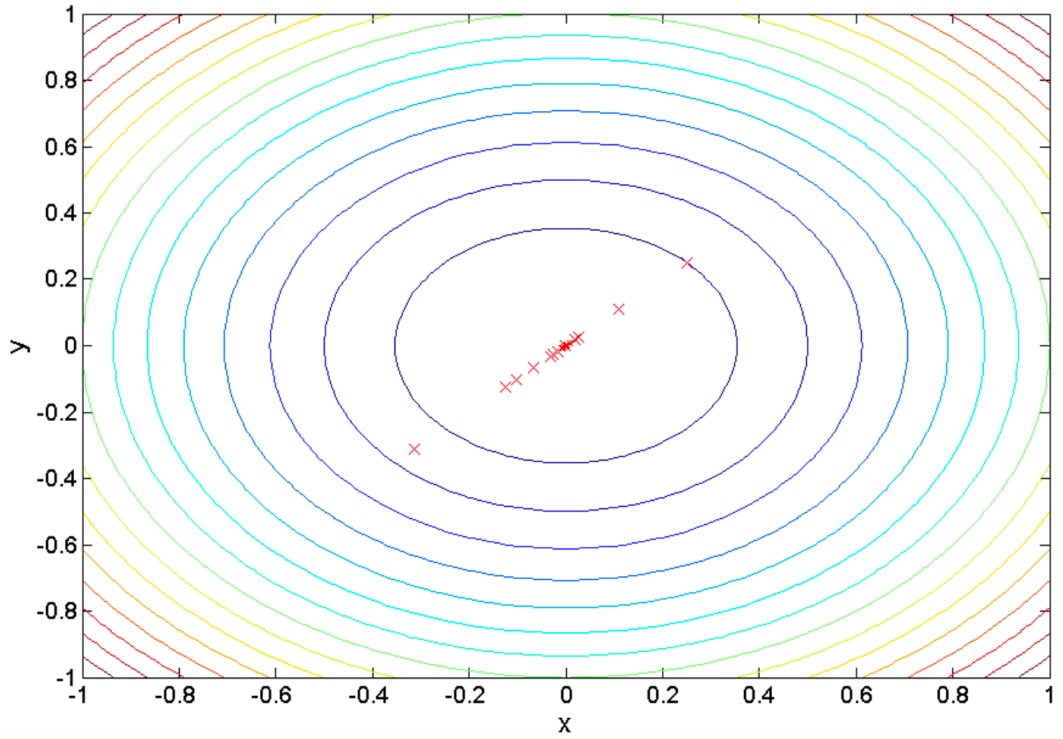
(d) Fourth Iteration

Second Iteration: In this iteration, the particles were moved to the new locations inside the search space using the positions and velocity that were calculated in the first iteration (see Section 4.1.1). The new positions of all particles are listed in Table 3. The velocity of all particles are then calculated (see Table 3) to move the particles in the next iteration. Moreover, the fitness values of all particles were calculated.

TABLE 4: The positions, velocity and best positions of all particles after the third iteration.

Particle No.	Initial Positions		Velocity		Best Solution	Best Position		Fitness Value
	x	y	x	y		x	y	
P_1	-0.1250	-0.1250	-0.1875	-0.1875	0.125	0.25	0.25	0.0313
P_2	0.8750	-0.1250	-1.3125	0.1875	0.125	0.25	0.25	0.7813
P_3	0.1250	0.6250	-0.1875	-0.9375	0.125	0.25	0.25	0.4063
P_4	-0.1250	0.8750	0.1875	-1.3125	0.125	0.25	0.25	0.7813
P_5	0.2500	0.2500	-0.3750	-0.3750	0.125	0.25	0.25	0.1250

Third Iteration: In this iteration, the particles continue moving towards the optimal solution. At the beginning, the particles were moved to the new positions and their fitness values were calculated as in Table 4. As shown, the first particle became much closer to the optimal solution than the other particles and its fitness value was 0.0313. As shown in this iteration, the velocities of all particles were not zero; in other words, the particles will be moved in the next iterations.

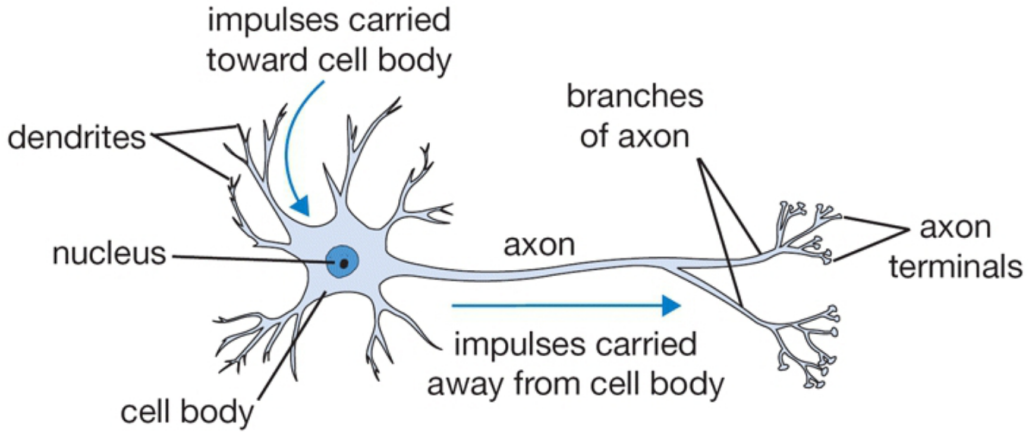


Particle Swarm Optimization

Neural Network Training

Biological motivation and connections

The basic computational unit of the brain is a **neuron**. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} — 10^{15} **synapses**. The diagram below shows a cartoon drawing of a biological neuron (left) and a common mathematical model (right).



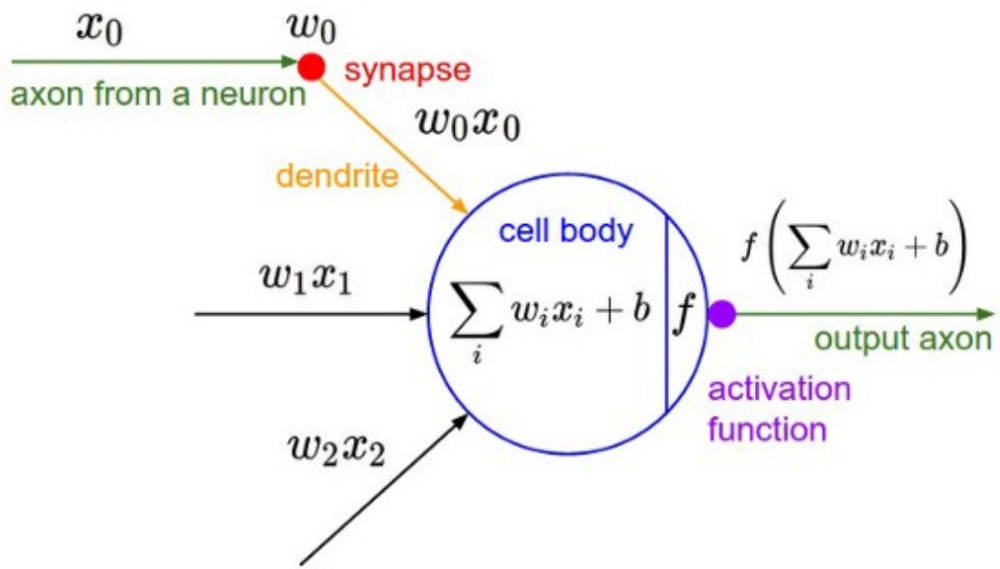
- The basic unit of computation in a neural network is the neuron , often called a node or unit.
- It receives input from some other nodes, or from an external source and computes an output.
- Each input has an associated weight (w), which is assigned on the basis of its relative importance to other inputs. The node applies a function to the weighted sum of its inputs.
- The idea is that the synaptic strengths (the weights *w*) are learnable and control the strength of influence and its direction: excitory (positive weight) or inhibitory (negative weight) of one neuron on another.
- In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can *fire*, sending a spike along its axon.

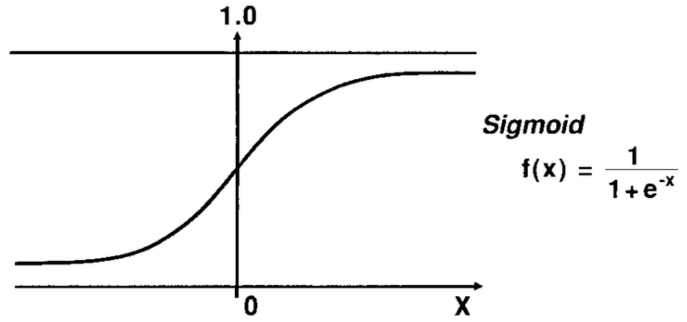
The Computational Model

- In the computational model:

The precise timings of the spikes are ignored,

- Only the frequency of the firing communicates information.
- We model the *firing rate* of the neuron with an **activation function** which represents the frequency of the spikes along the axon
- *The sigmoid function is an example.*





An illustration of the signal processing in a sigmoid function.

Neural Network Architecture

Neural Network Architecture

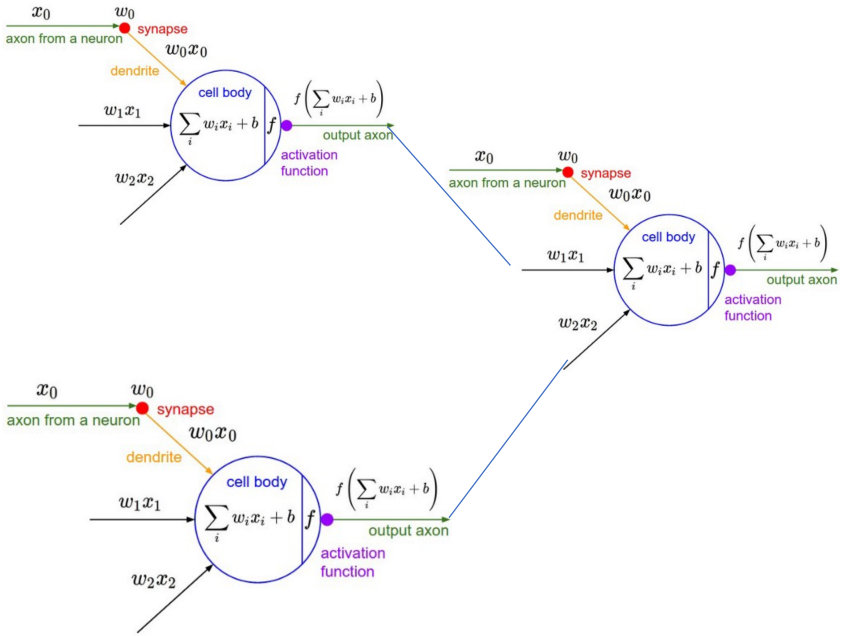
A neural network is made of neurons, biologically the neurons are connected through synapses where information flows (weights for out computational model), when we train a neural network we want the neurons to fire whenever they learn specific patterns from the data, and we model the fire rate using an activation function.

•**Input Nodes (input layer):** No computation is done here within this layer, they just pass the information to the next layer (hidden layer most of the time). A block of nodes is also called **layer**.

•**Hidden nodes (hidden layer):** In Hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or information) from the input layer to the following layer (another hidden layer or to the output layer). It is possible to have a neural network without a hidden layer and I'll come later to explain this.

•**Output Nodes (output layer):** Here we finally use an activation function that maps to the desired output format (e.g. class of the object in classification).

•**Connections and weights:** The *network* consists of connections, each connection transferring the output of a neuron i to the input of a neuron *j*. In this sense *i* is the predecessor of *j* and *j* is the successor of *i*, Each connection is assigned a weight *Wij.*

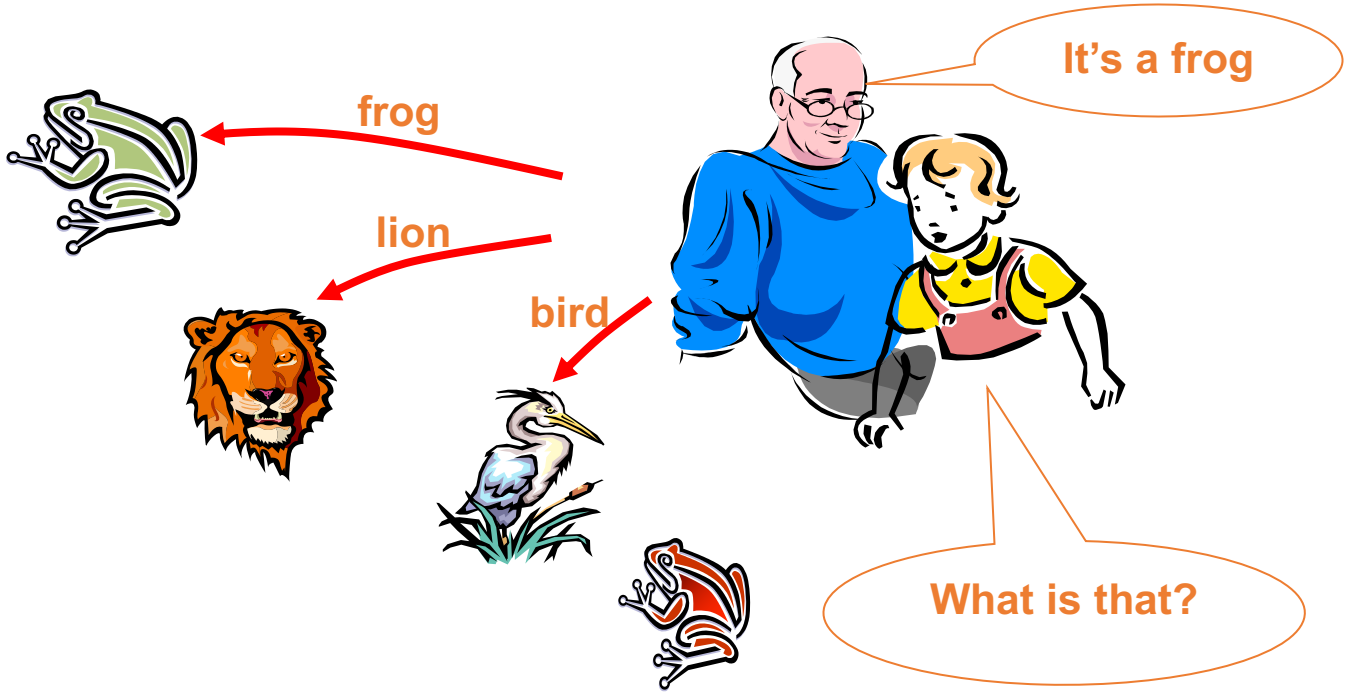


•**Activation function:** the **activation function** of a node defines the output of that node given an input or set of inputs. It is the *nonlinear* activation function that allows such networks to compute nontrivial problems using only a small number of nodes. In artificial neural networks this function is also called the transfer function.

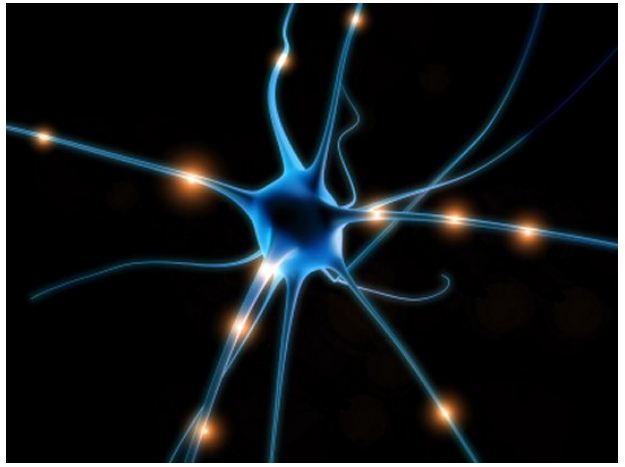
•**Learning rule:** The *learning rule* is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This *learning* process typically amounts to modifying the weights and thresholds.

Artificial Neural Network Learning

- Neural Networks (NN) learn the relationship between cause and effect or organize large volume of data into orderly and informative patterns.



Like people, they learn *from experience* (by example)



- **Neural network:** *information processing paradigm inspired by biological nervous systems, such as our brain*
  - Large number of highly interconnected processing elements (*neurons*) working together

Neural Network Training

- Neural Network (NN) performs a mapping from a set of input data to one or more output nodes,
- It's a structured set of nodes that are fully or partially connected,
- Each node is a processing unit that gets stimulated by:
  - An external input,
  - Another node.
- The node produces an output signal that reaches:
  - An output,
  - Another node.
- It's required to find the appropriate set of weights that will lead to a satisfactory network behaviour.

The Multi-Layer Perceptron Model

Artificial neural networks (NNs) are computational models based on the operation of biological neural networks, which constitute the information processing mechanism of the human brain. Their structure is based on the concept of the *artificial neuron*, which resembles biological neurons, as their main processing unit. The artificial neuron constitutes a nonlinear mapping between a set of input and a set of output data. Thus, if the input data are represented as a vector, $Q = (q_1, q_2, \ldots, q_m)^T$, the artificial neuron implements a function:

$$y = F\left(b_i + \sum_{i=1}^{m} w_i q_i \right),$$

where F is the *transfer function*; w_i, $i = 1, 2, \ldots, m$, are the *weights*; and b_i is a *bias*. In order to retain a compact notation, we will henceforth represent the bias, b_i, as a weight, w_0, with an auxiliary constant input, $q_0 = 1$.

The training of the neuron to learn an input-output pair, $\{Q, y\}$, is the procedure of detecting proper weights so that the output y is obtained if Q is presented to the neuron. Obviously, this procedure can be modeled as an error minimization problem:

$$\min_{w_i}\left(y - F\left(\sum_{i=0}^{m} w_i q_i \right) \right)^2 .$$

If more than one input vectors, $Q_1, Q_2, \ldots, Q_K$, are to be learned by the neuron, then the objective function is augmented with a separate square-error term for each input vector:

$$\min_{w_i} \sum_{k=1}^{K} \left(y_k - F\left(\sum_{i=0}^{m} w_i q_{ki} \right) \right)^2 ,$$

where q_{ki} is the i-th component of the k-th input vector, $Q_k = (q_{k1}, q_{k2}, \ldots, q_{km})^T$ (recall that w_0 is the bias of the neuron, with $q_{k0} = 1$ for all k). This kind of training procedure is also known as *batch training*, as all input vectors are presented to the neuron prior to any change of its weights.

Obviously, the dimension of the minimization problem is equal to the number, m, of the weights and biases, which is in direct correlation with the dimension of the input vector. The transfer function, $F(x)$, is selected based on the desired properties that shall be attributed to the model. Although linear transfer functions can be used, nonlinear functions equip the model with enhanced classification capabilities. Thus, nonlinear transfer functions are most commonly used, with *sigmoid* functions defined as:

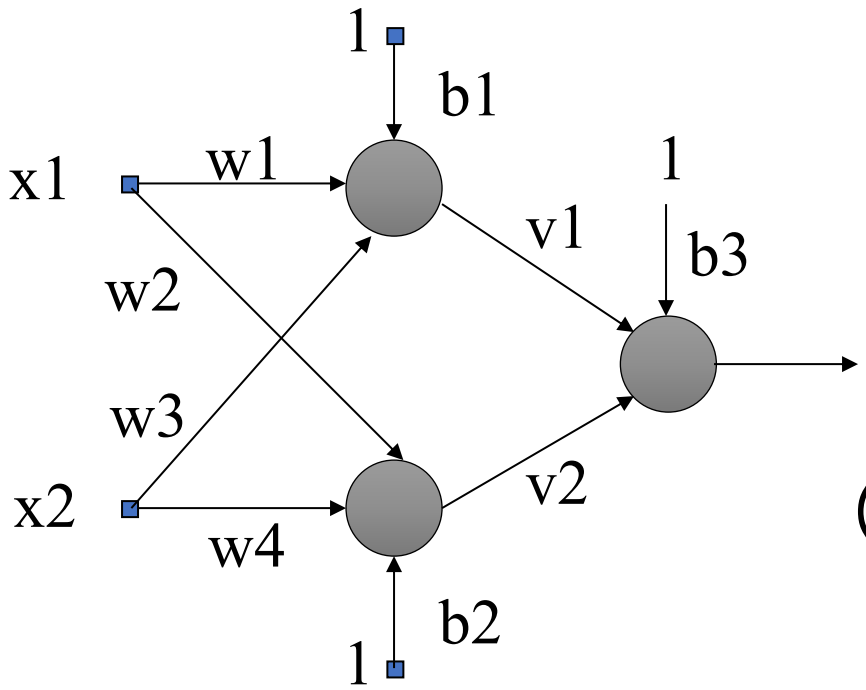
$$F(x,\lambda) = \frac{1}{1+\exp(-\lambda x)},$$

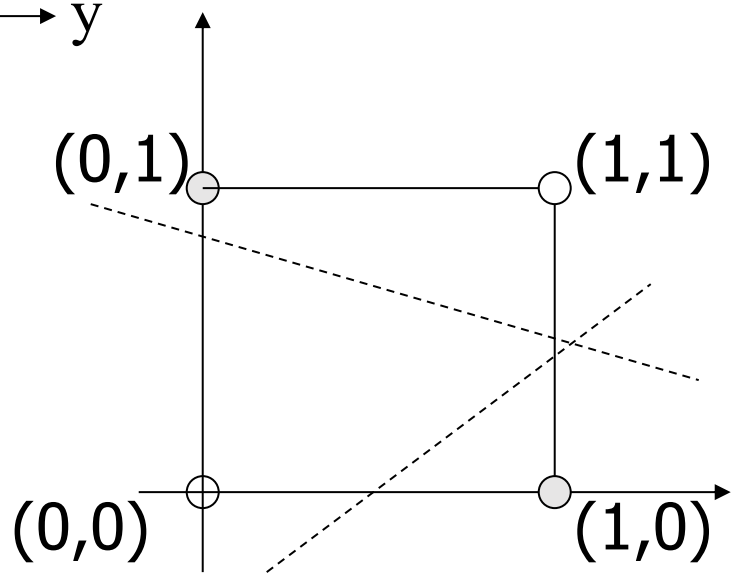
being the most common choice. In addition, alternative transfer functions have been proposed in the literature (Magoulas & Vrahatis, 2006).

Neural Network Training

- PSO proposed for optimizing the weights of an ANN performing the XOR operation.

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0





Neural Network Training

- How is the problem formulated?
  - PSO could be used to optimize the weights, each particle will be a 9-dimensional vector:

$$x = [w_1, w_2, w_3, w_4, b_1, b_2, b_3, v_1, v_2]$$

  - The velocity is a 9-dimensional vector of continuous values,
  - The objective is to minimize the error between the desired and the actual outputs of the network.

- It starts by randomly initializing a group of particles and let them move in the search space until an acceptable error is reached.

Neural Network Training

- PSO was effective as the back-propagation algorithm in training this NN,

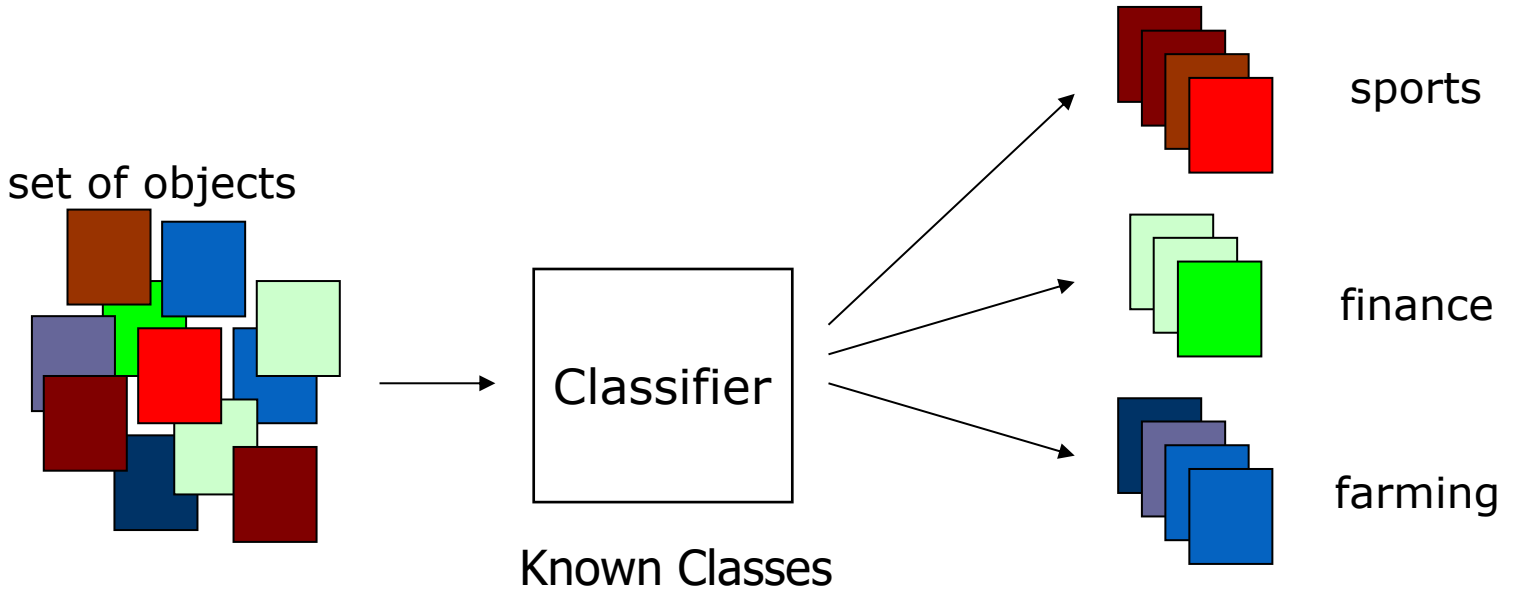
- A swarm of 20 particles was able to train network to an error of < 0.05 in an average of 30.7 iterations.

Particle Swarm Optimization

Clustering

Classification

- Function that assigns an object to a class
- Infer that “object X is about sports”
- Automatically learn the function from a set of examples
- Group similar objects into classes
- Similarity is high within the class and low between classes



Clustering vs. Classification

Clustering

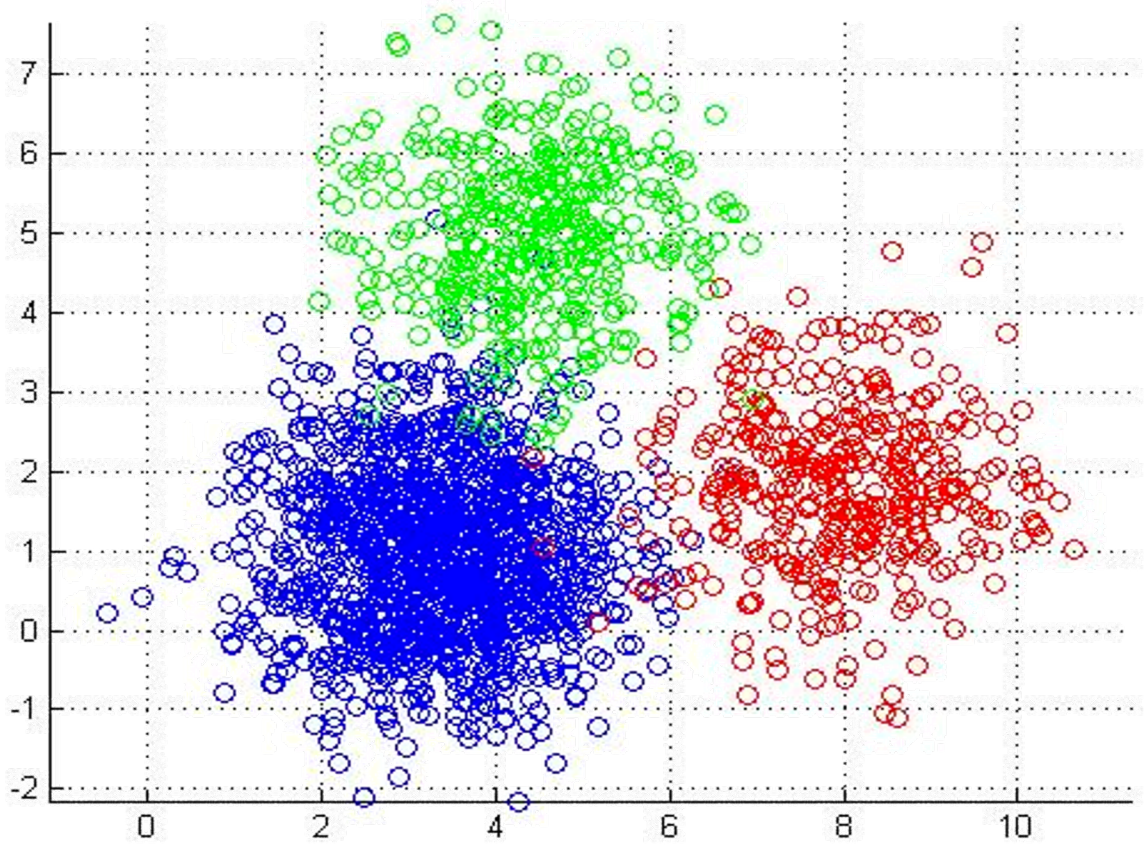
- Unsupervised
- Uses unlabeled data
- Organize patterns w.r.t. an optimization criteria
- Notion of similarity
- Hard to evaluate
- Example: K-means, Fuzzy C-means, Hierarchical

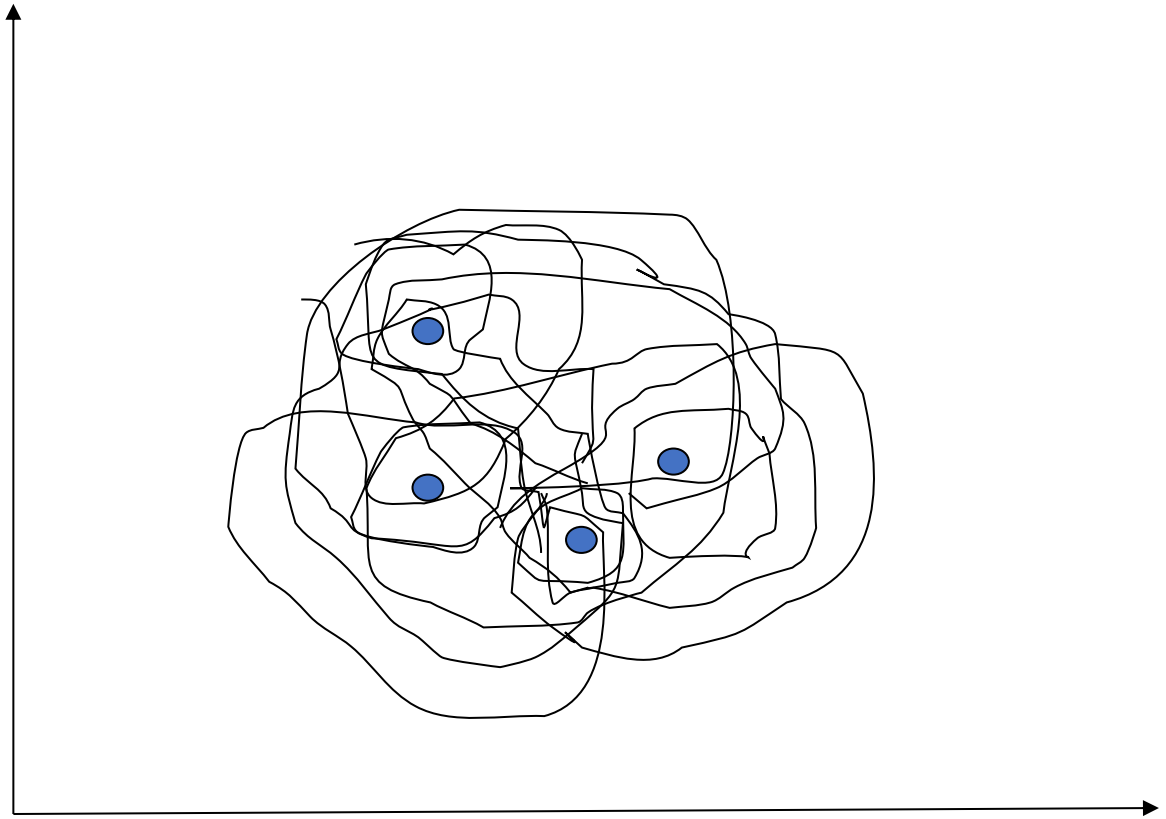
Classification

- Supervised
- Uses labeled data
- Requires training phase
- Domain sensitive
- Easy to evaluate
- Examples: Bayesian, k-NN, Decision Trees

Data Clustering Problem

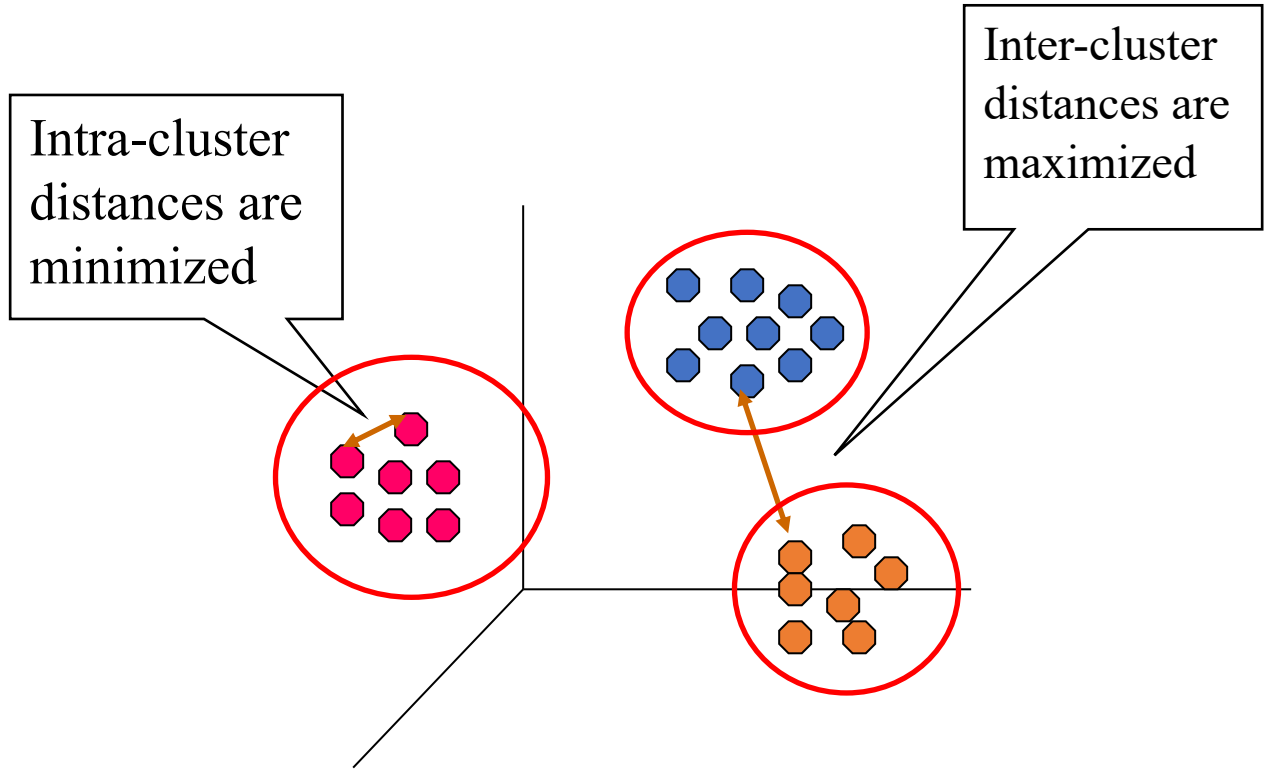
Given *n* objects (each could be a vector of d features), group them in *k* groups (Clusters) in such a way that all objects in a single group have a “natural" relation to one another, and objects not in the same group are somehow different





Applications of data clustering

- Data analysis
- Bioinformatics data
- Image segmentation
- GIS and spatial data
- Data and web mining
- Medical data
- Economic and financial data



K- means clustering Algorithm

Step 1: Initialize k Cluster centers

Repeat

Step 2: Distribute patterns among clusters using similarity measure and satisfying performance index.

Step 3: Compute new cluster centers

until no change in centers.

Apply the algorithm a number of times with different initial conditions

then choose the best solution.

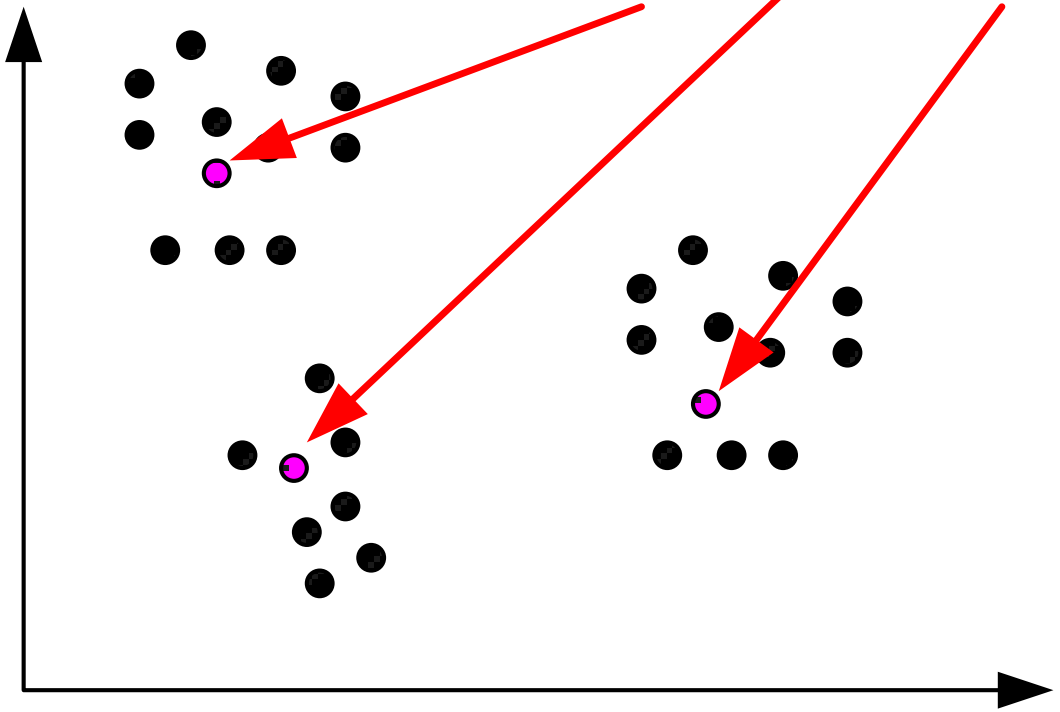
- The similarity measure could be the inverse of the Euclidean distance and a performance index can be the dispersion

$$f(C_1, C_2, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| x_j - z_i \right\|^2$$

Single Swarm Clustering

Particle representation

$$x^i = (m^{(1)}, m^{(2)}, m^{(3)})^i$$



PSO-Clustering

The PSO Clustering Algorithm

Initialize each particle with K random cluster centers.

for iteration count = 1 to maximum iterations do

for all particle i do

for all pattern X*p* in the dataset do

calculate Euclidean distance of X*p* with all cluster centroids

assign X*p* to the cluster that have nearest centroid to X*p*

end for

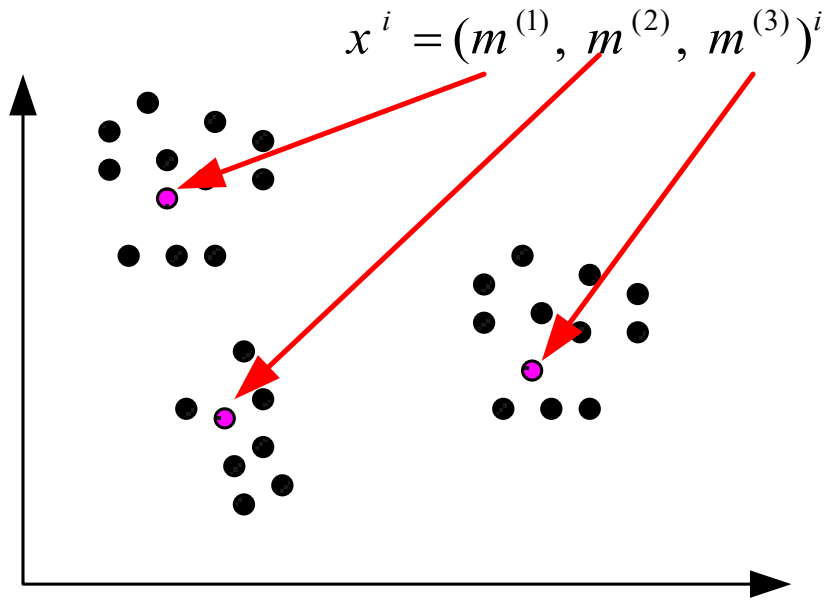
calculate the objective function for the current centers and assignment

end for

find the personal best and global best position of each particle.

Update the cluster centroids according to velocity updating and coordinate updating formula of PSO

end for.



Application to Document Clustering

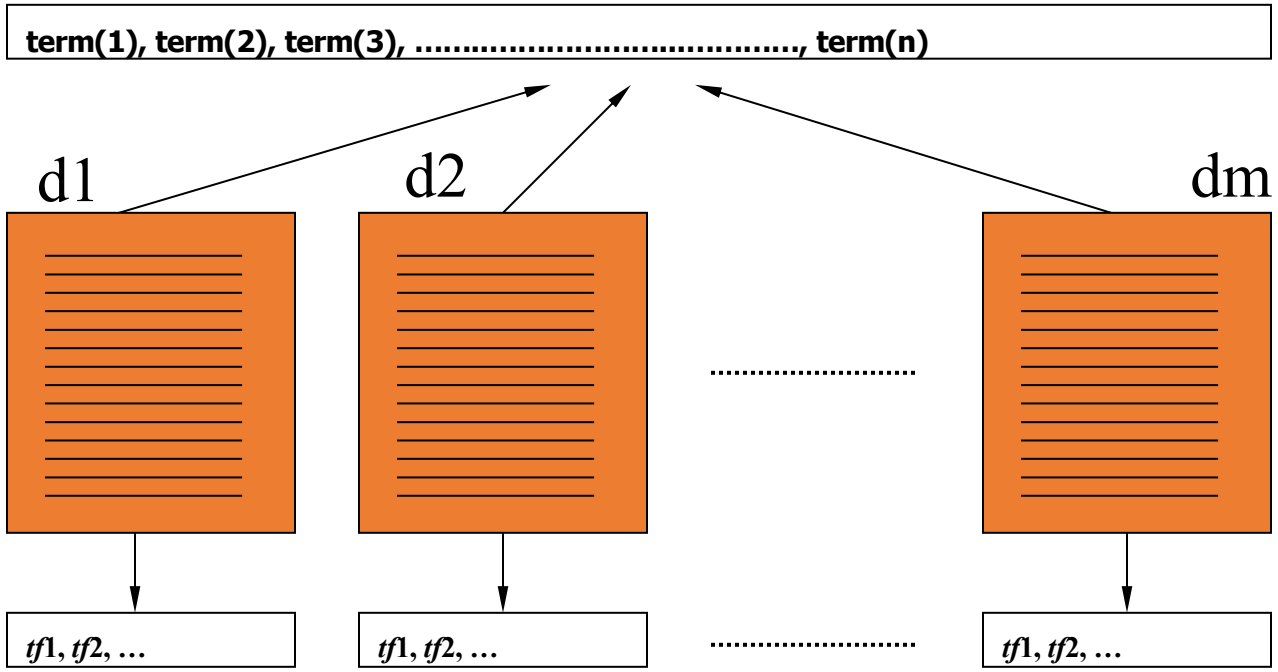
- Problems
  - Overwhelming amount of text information.
  - Search engines return linear ranked list of documents.
  - Documents are of various topics yet interleaved.

- Motivation
  - Uncovering inherent groupings in large document sets.
  - Easy navigation through large document sets.
  - Easy navigation of search engine results.

Document Representation Model

- Most Commonly used: Vector Space Model
- Document Representation: $\mathbf{d} = \{tf_1, tf_2, \ldots, tf_n\}$

  where $tf_i, i = 1, \ldots, n$ is the term frequency



Term Frequency Vectors

Document Representation Model

- Term-Document matrix

	t_1	t_2	t_3	.....	t_n
d_1	0.3	0	0.5	.....	0
d_2	0	0.1	0.4	.....	0.2
:	:	:	:		:
:	:	:	:	.....	:
:	:	:	:		:
d_m	0.1	0	0	.....	0

- **Term Frequency Disadvantage:** A term appearing frequently in all documents has less discriminating power.
- **Document Frequency (DF):** The number of documents in which a certain term appears.
  - The less the DF of a term, the more discriminating power it has.

Similarity Measure

- Cosine Measure (most commonly used) $\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$
- Euclidean Distance $\|x - y\|_2$
- The average distance of documents to the cluster centroid (ADDC)

$$(1/k)\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left\|d_{ij} - m_i\right\|_2 / n_i$$

- n_i is number of documents in cluster i

Hybrid PSO-K-means

1) Start the PSO clustering process until the maximum number of iterations is exceeded
2) Inherit clustering result from PSO as the initial centroid vectors of K-means module.
3) Start K-means process until maximum number of iterations is reached.

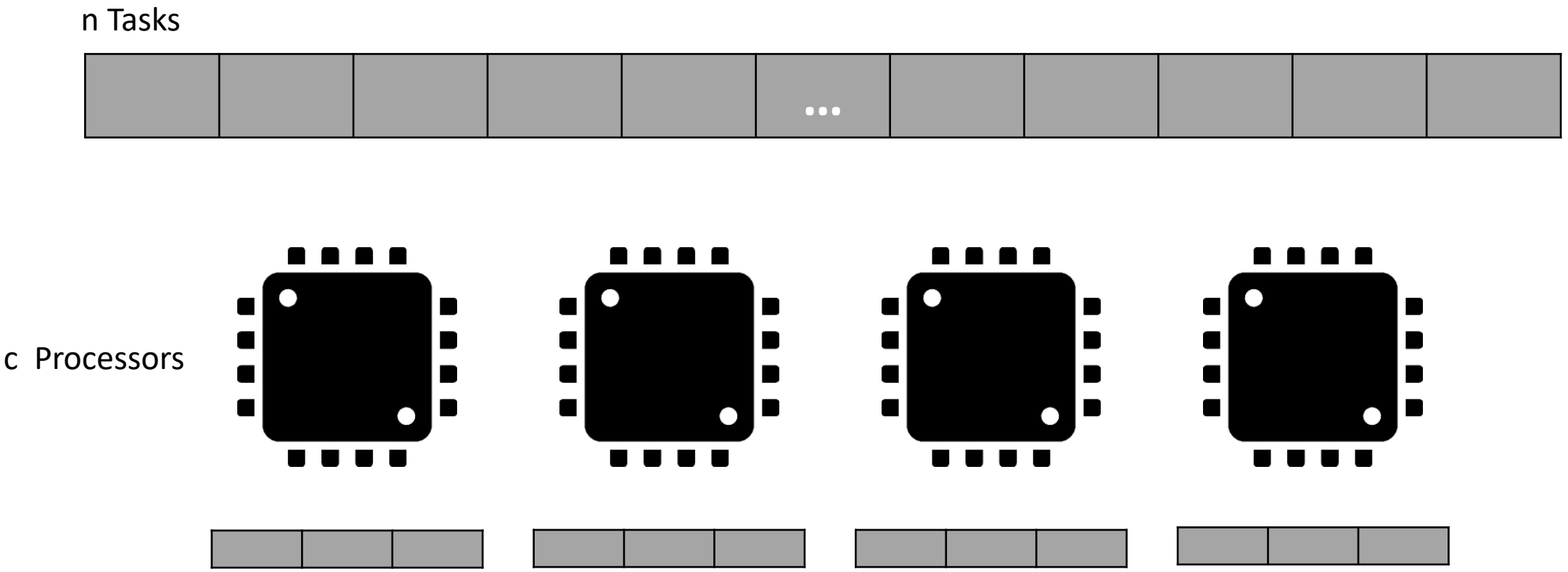
For PSO ***w*** is initially set as 0.72 and the
acceleration coefficient constants ***c1*** and ***c2*** are set as 1.49.
w is reduced by 1% at each generation to ensure good convergence.

Comparison between K-means, PSO, Hybrid PSO

Data Set	Similarity Measure	K-means	PSO-Kmeans	Hybrid PSO
Data Set 1	Euclidean	8.17817	8.11009	6.38039
	Cosine	8.96442	10.41271	8.14551
Data Set 2	Euclidean	7.26175	6.25172	4.51753
	Cosine	8.07653	9.57786	7.21153
Data Set 3	Euclidean	4.59539	4.14896	2.25961
	Cosine	4.97171	5.71146	4.00555
Data Set 4	Euclidean	9.08759	8.62794	6.37872
	Cosine	10.1739	12.8927	9.5379

Set Partitioning

- Set partitioning: a **partition** of a **set** is a grouping of the **set's** elements into non-empty subsets, in such a way that every element is included in one and only one of the subsets.



Set Partitioning

- Number of possible partitions is given by:

$$\mathrm{N}(n,c) = \left(\frac{1}{c!}\right)\sum_{m=1}^{c}(-1)^{c-m}\binom{c}{m}m^{n}$$

- If n = 50 and c = 4, the number is 5.3×10^{38}.

- If n is increased to 100, the number becomes 6.7×10^{58}.

- Enumerating all possible partitions for large problems is practically infeasible.

References

1. J. Kennedy and R. C. Eberhart. “Particle Swarm Optimization”. Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995.

2. K. Lei, Y. Qiu and Y. He. “A Novel Path Planning for Mobile Robots Using Modified Particle Swarm Optimizer”. Proc. of the 1st International Symposium on Systems and Control in Aerospace and Astronautics ISSCAA, pp. 981-984,2006.

3. Abraham, A., Das, S., & Roy, S. Swarm intelligence algorithms for data clustering. In Soft computing for knowledge discovery and data mining Part IV, pp. 279-313, 2007.

4. X. Cui; J. Gao and T. E. Potok. A flocking based algorithm for document clustering analysis. *Journal of Systems Architecture*, 52(8-9):505–515, August-September 2006

5. A. Ahmadi; F. Karray and M. Kamel. Multiple cooperating swarms for data clustering. In *IEEE Swarm Intelligence Symposium*, pages 206–212, 2007.