

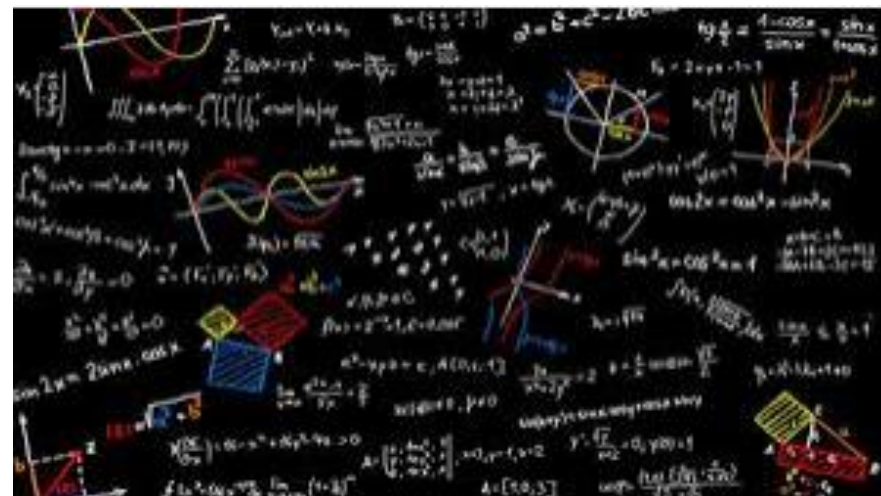


Matemáticas Aplicadas

**Ing. Wilson Mauro Rojas Reales,
Esp., Mg., Ph.D(c)
Profesor Asociado**

Correo: rojaswilson@unbosque.edu.co

**lunes, miércoles (16:00 – 18:00)
jueves (11:00 – 13:00)**



Contenido del curso

~~0.- Preeliminares~~

~~1.- Teoría de números.~~

~~2.1.- Números primos.~~

~~2.2.- Números aleatorios.~~

~~2.3.- Aritmética Modular~~

~~2.4.- Criptografía~~

3.- Introducción a la IA.

3.1- Redes Neuronales.

3.2.- Algoritmos Genéticos.

4.- Aplicabilidad en Software

-Autómatas celulares.

5.- Aplicabilidad en redes.

5.1.- Teoría de grafos y sus aplicaciones.

5.2.- Códigos detectores y correctores de errores.

Segundo parcial (50%):
Octubre 05 y 06.2022

Evaluación final (50%):
Noviembre 28,30 y diciembre 1 de 2022

Esquema de evaluación

Segundo corte

- ❑ Actividades de aprendizaje (talleres, exposiciones, otras) equivalen al **40%**.
- ❑ La participación **activa** equivale al **30%**. Un estudiante realiza un resumen de la clase anterior y a partir de la interacción con el docente se genera un debate durante los primeros quince (15) minutos de la sesión de clases. **La nota se obtiene con base al estudiante quien haya logrado la mayor cantidad de puntos de participación durante el corte.**
- ❑ La evaluación parcial del segundo corte está dividida en dos sesiones. Una evaluación **escrita** de mínimo 44 preguntas y máximo 60 preguntas para responder en un tiempo máximo de 50 minutos. Esta evaluación comprende los temas desde el primer día de clases a la fecha de evaluación. Y una evaluación práctica que constará de un ejercicio de programación bajo el lenguaje de programación **Python** o **Java**. Estas dos partes, equivalen al otro **30%** restante.

Sobre la entrega de talleres

- ☐ Todo a través del AULA VIRTUAL. No se permite la entrega por correo electrónico salvo el caso que el Aula Virtual presente problemas. En este caso, el estudiante primero deberá solicitar autorización al docente mediante correo electrónico enviando el soporte gráfico del inconveniente que presente el aula virtual en ese momento.
- ☐ **NO** están autorizadas las **grabaciones (en cualquier medio)** o imágenes obtenidas de las clases sin el previo consentimiento del docente.
- ☐ La utilización de la imagen de las personas: docentes o estudiantes sin su previa autorización está expresamente prohibida de acuerdo a la normatividad de Habeas Data.

Sobre la entrega de talleres

Siempre se solicitará el **código fuente** cuando se trate de hacer programas. No se debe entregar “carpetas de proyectos”. Los archivos “.txt” **NO** constituyen un código fuente. Incumplir esta regla “anula” por completo el ejercicio y la nota será de 0.0.

Sobre la evaluación teórica del corte

Fecha límite para el registro de las preguntas: **fecha límite septiembre 30.2022**

❑ **Artículo 68.** DE LAS FALTAS DISCIPLINARIAS GRAVES: Se consideran faltas disciplinarias graves, las siguientes conductas:

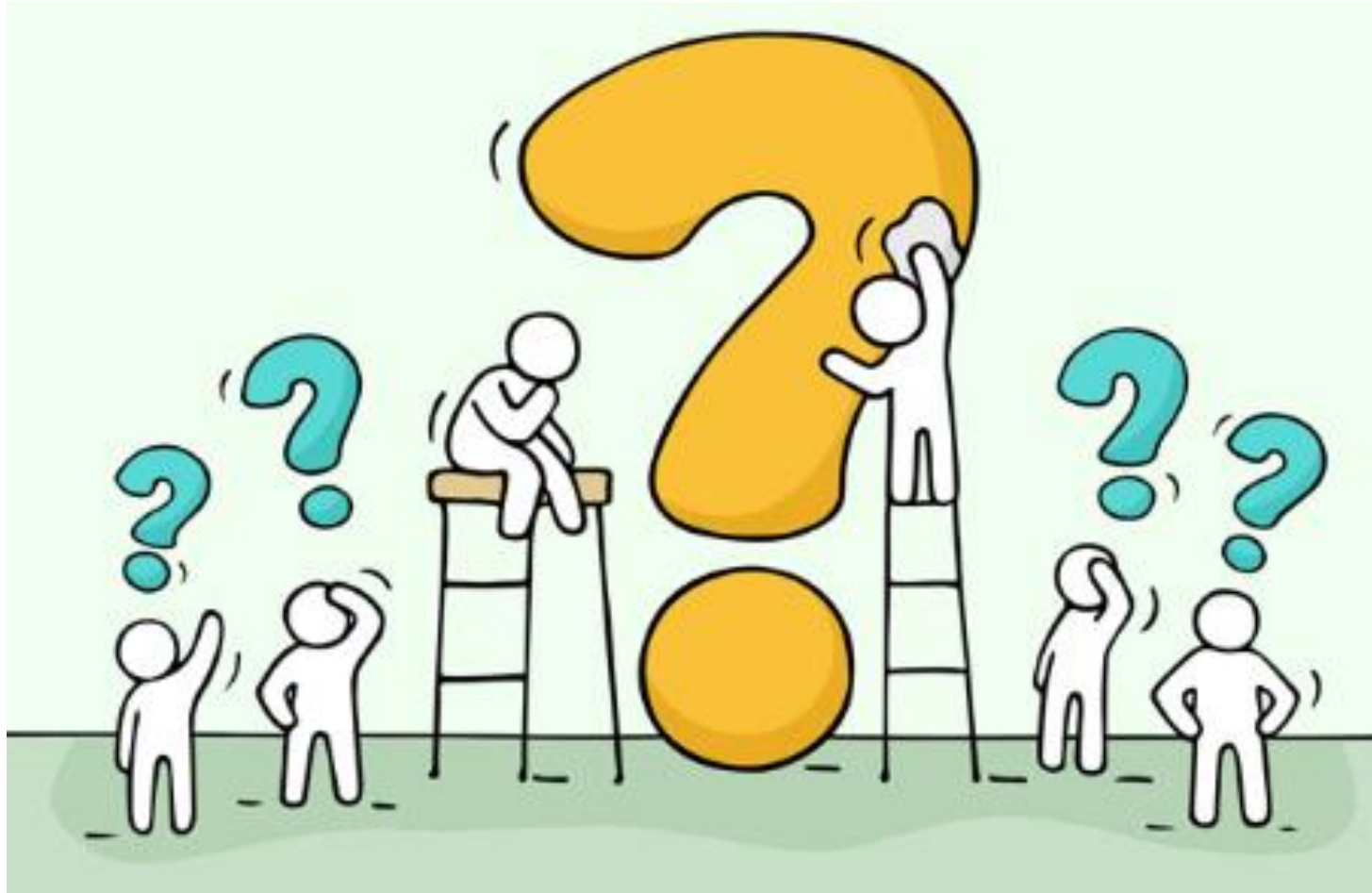
b.- Hacer o intentar fraude en los exámenes u otras pruebas académicas, o coadyuvar a ello, en cuyo caso se anulará el respectivo examen o prueba académica y recibirá una calificación de cero coma cero (0.0). La asignatura perdida como consecuencia de lo dispuesto en este literal, no podrá ser objeto de ninguna prueba de recuperación y por tanto deberá cursarse nuevamente.

El protagonista

- ☐ El estudiante es el **protagonista** de su propio aprendizaje y por tanto debe asumir su responsabilidad.
- ☐ Debe tener sus propios objetivos e intereses, proponerse estrategias para lograrlo. Debe ser disciplinado, organizado, comprometido.
- ☐ El docente es un guía de la metodología.
- ☐ La asistencia a las sesiones de clases es fundamental.



Hasta aquí cómo vamos...?



Temas:

1. Machine Learning.
2. Deep Learning.
3. Neural Networks.
4. Computer Vision.
5. Natural Language Processing.
6. Chatbot.
7. Recommender Systems.
8. Predictive Analysis.

Fecha de inicio: Septiembre 07.2022

Exposiciones

Un integrante del grupo realiza la presentación. El tiempo máximo es de cinco (5) minutos para vender la idea.

Cualquier integrante del grupo podrá “reforzar” la presentación.

El docente decide si el grupo continua realizando la presentación por más de cinco (5) minutos.

Cualquier miembro del auditorio podrá “cuestionar” la presentación.

Las notas a obtener son: 2.5 o 5.0.

Exposiciones – Grupos

INTEGRANTES

David Salamanca Sánchez Oscar Calderón Forero Jorge Bravo Villa Nicolás Ferreira Pérez Luis Pineda Poveda	Kevin Pinzón Castillo Hernán Alvarado Parra Jorge Yate Bocanegra Sergio Sanabria García Julián Cortés Gómez	Paula Anaya Ramírez Juan Castillo Fuenmayor Juan Bravo Guerrero Nelson Uribe Calderón Andrés Cuadros Bacca	Oscar Moreno Moreno Nicolás Rodríguez Vargas Nelson Fandiño Díaz José Salcedo Mercado Tomás Espitia Galindo
---	---	--	---

INTEGRANTES

Sebastián Castillo Chavés Alejandro Chitiva Castillo Gabriel Maldonado Saez Santiago Melo Cristancho Jeison Pedraza Sanabria	Nicolás Camacho Lesmes Juan Muñoz Peñuela Javier Villarreal Cruz Juan Piza Rodríguez Esteban Hernández Londoño	Felipe Galindo Melo Andrés Zambrano Rojas Johan Ayala Gaitan Yefferson Linares Ospina	María Angarita Barrantes Laura Valentina Aguirre Rasbel Castellano Correa María Enciso Mendoza David Moya Aldana
--	--	--	--

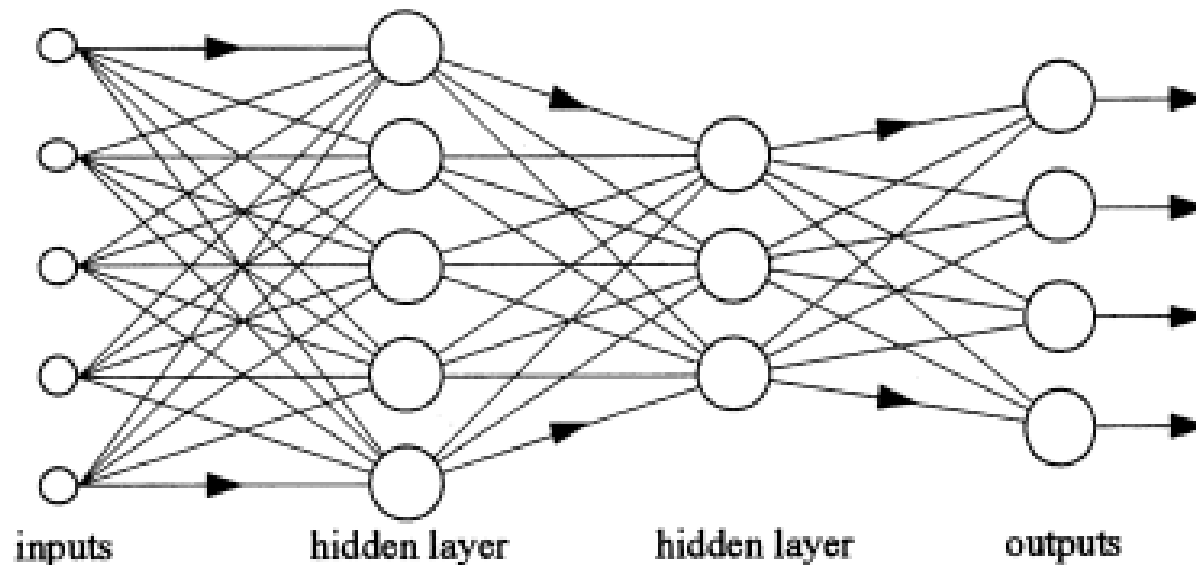
INTEGRANTES

Laura Mateus Ruíz
Kevin García Pérez
Jeanpierr Ramos Sandoval
Natalia Ardila Verano

Redes Neuronales Artificiales

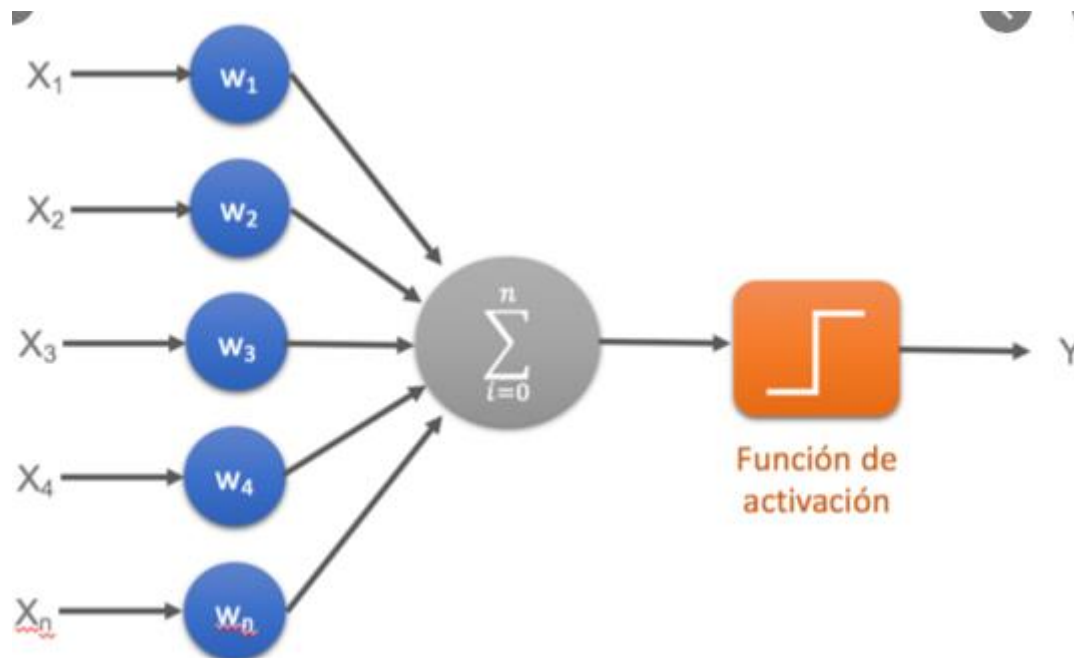
Topología hacia adelante:

- 1.- Propagación de señales en un solo sentido.
- 2.- El número de neuronas por capa puede ser distinto en cada capa.



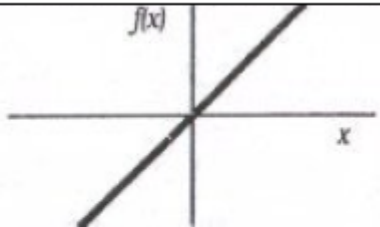
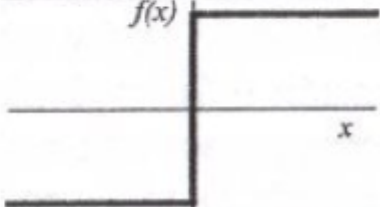
Funciones de “Activación”

Determinan el estado de activación actual de la neurona con el potencial resultante y el estado de activación de la neurona anterior. La función de **Salida** proporciona el valor de salida de la neurona, con base al estado de activación de la neurona.

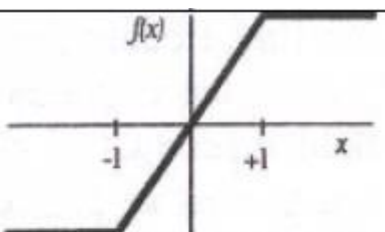
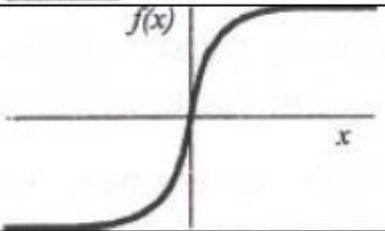
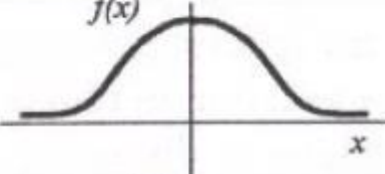
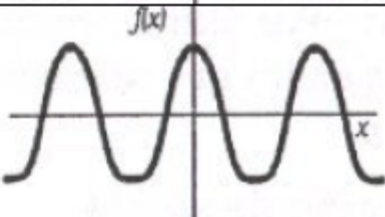


Funciones de “Activación”

Calcula el estado de actividad de una neurona transformando la entrada global (menos el umbral) en un valor (estado) de activación cuyo rango va de 0..1 o de (-1..1). Recuerde que la neurona puede estar inactiva (0 o -1) o activa (1).

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, \infty]$	
Escalón	$y = \begin{cases} 1, & \text{si } x \geq 0 \\ 0, & \text{si } x < 0 \end{cases}$ $y = \begin{cases} 1, & \text{si } x \geq 0 \\ -1, & \text{si } x < 0 \end{cases}$	$[0, 1]$ $[-1, 1]$	

Funciones de “Activación”

Lineal a tramos	$y = \begin{cases} 1, & \text{si } x > 1 \\ x, & \text{si } -1 \leq x \leq 1 \\ -1, & \text{si } x < -1 \end{cases}$	$[-1, 1]$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \tanh(x)$	$[0, 1]$ $[-1, 1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, 1]$	
Sinusoidal	$y = A \sin(wx + \varphi)$	$[-1, 1]$	

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Pesos iniciales:

$$W_1 = 1.6$$

$$W_2 = 2.5$$

$$W_0 = 4.3$$

Término de
tendencia W_0

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #1 (caso 1)(Comprende los 4 casos)

$$Y = 1.6 * 0 + 2.5 * 0 + 4.3 = 4.3$$

Se esperaba una salida “0” y se obtuvo un “1”

$$\text{error} = 0 - 1 = -1$$

(valor esperado menos el valor obtenido)

ajuste de pesos:

$$W_1 = 1.6 - 1 * 0 = 1.6$$

$$W_2 = 2.5 - 1 * 0 = 2.5$$

$$W_0 = 4.3 - 1 * 1 = 3.3$$

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Ajuste de pesos:

$$W_{k,i(nuevo)} = W_{k,i(anterior)} + Error \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 3.3$

Iteración #1 (Caso 2)

$$Y = 1.6 * 0 + 2.5 * 1 + 3.3 = 5.8$$

Se esperaba una salida “0” y se obtuvo un “1”

$$\text{error} = 0 - 1 = -1$$

(valor esperado menos el valor obtenido)

ajuste de pesos:

$$w_1 = 1.6 - 1 * 0 = 1.6$$

$$w_2 = 2.5 - 1 * 1 = 1.5$$

$$w_0 = 3.3 - 1 * 1 = 2.3$$

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Ajuste de pesos:

$$w_{k,i(\text{nuevo})} = w_{k,i(\text{anterior})} + \text{Error} \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

Iteración #1 (Caso 3)

$$Y = 1.6 * 1 + 1.5 * 0 + 2.3 = 3.9$$

Se esperaba una salida "0" y se obtuvo un "1"

$$\text{error} = 0 - 1 = -1$$

(valor esperado menos el valor obtenido)

ajuste de pesos:

$$w_1 = 1.6 - 1 * 1 = 0.6$$

$$w_2 = 1.5 - 1 * 0 = 1.5$$

$$w_0 = 2.3 - 1 * 1 = 1.3$$

ajuste de pesos:

$$w_{k,i(\text{nuevo})} = w_{k,i(\text{anterior})} + \text{Error} \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$\text{Salida}_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$

$w_2 = 2.5$

$w_0 = 4.3$

Iteración #1 (Caso 4)

$Y = 0.6 * 1 + 1.5 * 1 + 1.3 = 3.4$

Se obtuvo un “1” que era la salida esperada

error = $1 - 1 = 0$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #2 (Caso 1)

$$Y = 0.6 * 0 + 1.5 * 0 + 1.3 = 1.3$$

Se obtuvo un “1” y se esperaba un “0”

$$\text{error} = 0 - 1 = -1$$

(valor esperado menos el valor obtenido)

ajuste de pesos:

$$w_1 = 0.6 - 1 * 0 = 0.6$$

$$w_2 = 1.5 - 1 * 0 = 1.5$$

$$w_0 = 1.3 - 1 * 1 = 0.3$$

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

ajuste de pesos:

$$w_{k,i(\text{nuevo})} = w_{k,i(\text{anterior})} + \text{Error} \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #2 (Caso 2)

$$Y = 0.6 * 0 + 1.5 * 1 + 0.3 = 1.8$$

Se obtuvo un “1” y se esperaba un “0”

$$\text{error} = 0 - 1 = -1$$

(valor esperado menos el valor obtenido)

ajuste de pesos:

$$w_1 = 0.6 - 1 * 0 = 0.6$$

$$w_2 = 1.5 - 1 * 1 = 0.5$$

$$w_0 = 0.3 - 1 * 1 = -0.7$$

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

ajuste de pesos:

$$w_{k,i(\text{nuevo})} = w_{k,i(\text{anterior})} + \text{Error} \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$

$w_2 = 2.5$

$w_0 = 4.3$

Iteración #2 (Caso 3)

$$Y = 0.6 * 1 + 0.5 * 0 - 0.7 = -0.1$$

Se obtuvo un "0" y se esperaba un "0"

$$\text{error} = 0 - 0 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Pesos iniciales:

$w_1 = 1.6$

$w_2 = 2.5$

$w_0 = 4.3$

Término de
tendencia w_0

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #2 (Caso 4)

$$Y = 0.6 * 1 + 0.5 * 1 - 0.7 = 0.4$$

Se obtuvo un “1” y se esperaba un “1”

$$\text{error} = 1 - 1 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Término de
tendencia w_0

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

Iteración #3 (Caso 1)

$$Y = 0.6 * 0 + 0.5 * 0 - 0.7 = -0.7$$

Se obtuvo un “0” y se esperaba un “0”

$$\text{error} = 0 - 0 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

Término de
tendencia w_0

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #3 (Caso 2)

$$Y = 0.6 * 0 + 0.5 * 1 - 0.7 = -0.2$$

Se obtuvo un “0” y se esperaba un “0”

$$\text{error} = 0 - 0 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Pesos iniciales:

$w_1 = 1.6$

$w_2 = 2.5$

$w_0 = 4.3$

Término de
tendencia w_0

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #3 (Caso 3)

$$Y = 0.6 * 1 + 0.5 * 0 - 0.7 = -0.1$$

Se obtuvo un “0” y se esperaba un “0”

$$\text{error} = 0 - 0 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$

Perceptrón Simple - Ejemplo 01

Función lógica **AND**

Pesos iniciales:

$w_1 = 1.6$ $w_2 = 2.5$ $w_0 = 4.3$

Término de
tendencia w_0

x_1	x_2	y $x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Iteración #3 (Caso 4)

$$Y = 0.6 * 1 + 0.5 * 1 - 0.7 = 0.4$$

Se obtuvo un “1” y se esperaba un “1”

$$\text{error} = 1 - 1 = 0$$

(valor esperado menos el valor obtenido)

ajuste de pesos: No se ajustan.

**HASTA AQUÍ TERMINA EL PROCESO DE
ENTRENAMIENTO.**

$$Salida_k = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} \geq 0 \\ 0 & \text{si } \sum_{i=1}^n w_{k,i} x_i + w_{k,0} < 0 \end{cases}$$



Incluyendo un Factor de Aprendizaje

- ❑ Se utiliza para lograr mayor rapidez en la ejecución del algoritmo y también su exactitud. Cuanto mayor sea, se necesitaran menos iteraciones pero el aprendizaje no se logra con exactitud. En cambio, cuanto menor sea el proceso es más lento pero el aprendizaje se logra con mayor exactitud.
- ❑ En caso de agregarse un factor de aprendizaje, el nuevo peso deberá calcularse de la siguiente manera:

$$w_{k,i(nuevo)} = w_{k,i(anterior)} + \mu \cdot Error \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Perceptrón Simple – Ejemplo 02

Se desea entrenar un **Perceptrón** con las siguientes condiciones iniciales (incluiremos el factor de aprendizaje cuyo valor puede variar entre 0.0 y 1.0) : $W_1 = 0.5$, $W_2 = 1.5$, $\alpha = 1.0$, $W_0 = 1.5$, $X_0 = 1$ (Umbral). La salida deseada es la función **OR**. (**Nota:** Al umbral también se le asigna un peso)

Iteración #1 (Caso 1)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 0 + 1.5 * 1 = 1.5$$

La salida es “1” porque $f(y) \geq 0$

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Como no es la salida deseada, se deben

Ajustar los pesos:

$$(W_{\text{new}} = W_{\text{ant}} + \alpha * (\text{error}) * x_i).$$

$$\text{error} = 0 - 1 = -1$$

$$W_0 = 1.5 + 1 * (-1) * 1 = 0.5$$

$$W_1 = 0.5 + 1 * (-1) * 0 = 0.5$$

$$W_2 = 1.5 + 1 * (-1) * 0 = 1.5$$

Perceptrón Simple - Ejemplo 02

Iteración #1 (Caso 2)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 1 + 0.5 * 1 = 2.0$$

La salida es “1” porque $f(y) \geq \theta$

Como es la salida deseada, no se ajustan los pesos.

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Iteración #1 (Caso 3)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 0 + 0.5 * 1 = 1.0$$

La salida es “1” porque $f(y) \geq \theta$

Como es la salida deseada, no se ajustan los pesos.

Perceptrón Simple - Ejemplo 02

Iteración #1 (Caso 4)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 1 + 0.5 * 1 = 2.5$$

La salida es "1" porque $f(y) \geq 0$

Como es la salida deseada, no se ajustan los pesos.

Iteración #2 (Caso 1)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 0 + 0.5 * 1 = 0.5$$

La salida es "1" porque $f(y) \geq 0$

En este caso es necesario ajustar los pesos. Se esperaba un cero (0).

$$\text{error} = 0 - 1 = -1$$

$$W_0 = 0.5 + 1 * (-1) * 1 = -0.5$$

$$W_1 = 0.5 + 1 * (-1) * 0 = 0.5$$

$$W_2 = 1.5 + 1 * (-1) * 0 = 1.5$$

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrón Simple - Ejemplo 02

Iteración #2 (Caso 2)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 1 + (-0.5) * 1 = 1.0$$

La salida es “1” porque $f(y) \geq 0$

NO es necesario realizar el ajuste de los pesos.

Iteración #2 (Caso 3)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 0 + (-0.5) * 1 = 0.0$$

La salida es “1” porque $f(y) \geq 0$

NO es necesario realizar el ajuste de los pesos.

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrón Simple - Ejemplo 02

Iteración #2 (Caso 4)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 1 + (-0.5) * 1 = 1.5$$

La salida es “1” porque $f(y) \geq 0$

NO es necesario realizar el ajuste de los pesos.

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Iteración #3 (Caso 1)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 0 + (-0.5) * 1 = -0.5$$

La salida es “0” porque $f(y) < 0$

NO es necesario realizar el ajuste de los pesos.

Perceptrón Simple - Ejemplo 02

Iteración #3 (Caso 2)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 0 + 1.5 * 1 + (-0.5) * 1 = 1.0$$

La salida es “1” porque $f(y) \geq \theta$

NO es necesario realizar el ajuste de los pesos.

Iteración #3 (Caso 3)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 0 + (-0.5) * 1 = 0.0$$

La salida es “1” porque $f(y) \geq \theta$

NO es necesario realizar el ajuste de los pesos.

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrón Simple - Ejemplo 02

Iteración #3 (Caso 4)

$$Y = W_1 * X_1 + W_2 * X_2 + W_0 * X_0$$

$$Y = 0.5 * 1 + 1.5 * 1 + (-0.5) * 1 = 1.5$$

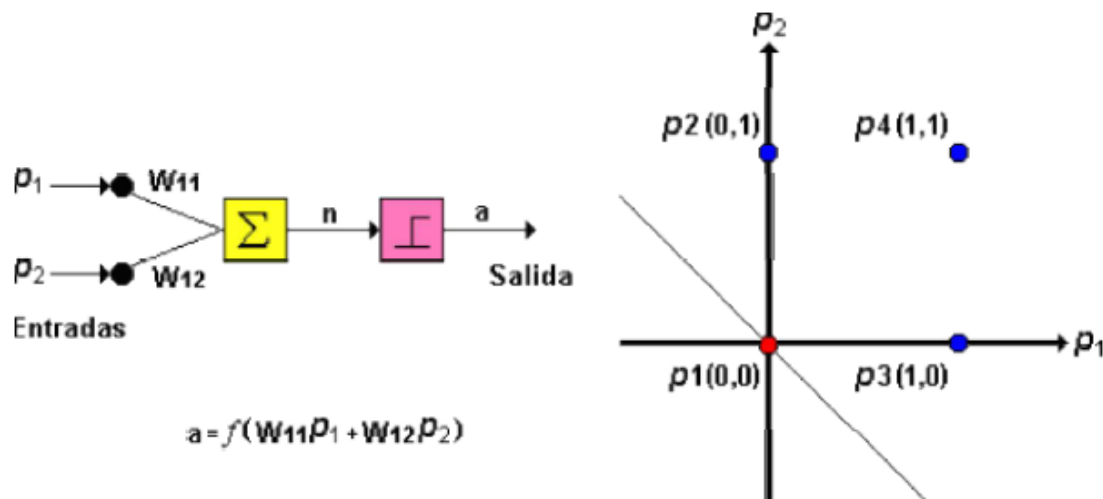
La salida es “1” porque $f(y) \geq 0$

NO es necesario realizar el ajuste de los pesos.

X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1

HASTA AQUÍ TERMINA EL PROCESO DE ENTRENAMIENTO. Los pesos ideales son: $W1 = 0.5$, $W2 = 1.5$ y $W0 = -0.5$

Perceptrón Simple - Ejemplo 02



X_1	X_2	$d(t) = X_1 \text{ or } X_2$
0	0	0
0	1	1
1	0	1
1	1	1



Ejercicio

- Utilizando un **Perceptrón Simple** implemente una solución para resolver la siguiente clasificación:

X	Y	Familia
-1	-1	A
-0.5	-0.5	A
-1	0	A
0	-1	A
1	1	B
0.5	0.5	B
1	0	B
0	1	B

Tenga en cuenta el factor de aprendizaje.
EL valor del umbral es 1.0 (variable “u”) y
 $W_0 = 1.5$

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x - u > 0 \\ 0 & \text{en otro caso} \end{cases}$$

Fórmula para calcular los nuevos pesos

$$w_{k,i(nuevo)} = w_{k,i(anterior)} + \mu \cdot Error \cdot x_i \quad \text{Para todas las } i, \text{ de } 1 \text{ a } n$$

Reconocimiento de patrones

- ❑ **Qué es un patrón?**. un patrón es una entidad a la que se le puede dar un nombre y que está representada por un conjunto de propiedades medidas y las relaciones entre ellas (vector de características). Por ejemplo un **Patrón** podría ser una imagen de una cara humana de las cuales se extrae el vector de características formado por un conjunto de valores numéricos calculados a partir de la misma.
- ❑ **El reconocimiento automático**, descripción, clasificación y agrupamiento de patrones son actividades importantes en una gran variedad de disciplinas científicas, como biología, sicología, medicina, visión por computador, inteligencia artificial, etc.

Pasos: Reconocimiento de patrones

- ❑ Establecer el número de neuronas en la capa de entrada y el número de neuronas en la capa de salida.
- ❑ Es necesario encontrar una estructura de datos para almacenar todos los elementos que se desean reconocer. Por ejemplo, utilizar una matriz si queremos representar números o letras.
- ❑ Representar el valor en cada una de las celdas dependiendo el carácter a representar.
- ❑ Convertir la matriz en un vector lineal.
- ❑ Por cada elemento que se desee reconocer por lo general se debe buscar por lo menos siete modelos.

Reconocimiento de patrones - ejemplo

- ❑ A continuación se explica de manera breve un ejemplo para el caso de reconocer números enteros.
- ❑ **Paso1:** Defina al menos tres patrones (Este es un ejemplo) y la estructura de datos a utilizar.

Matriz de 7 x 6

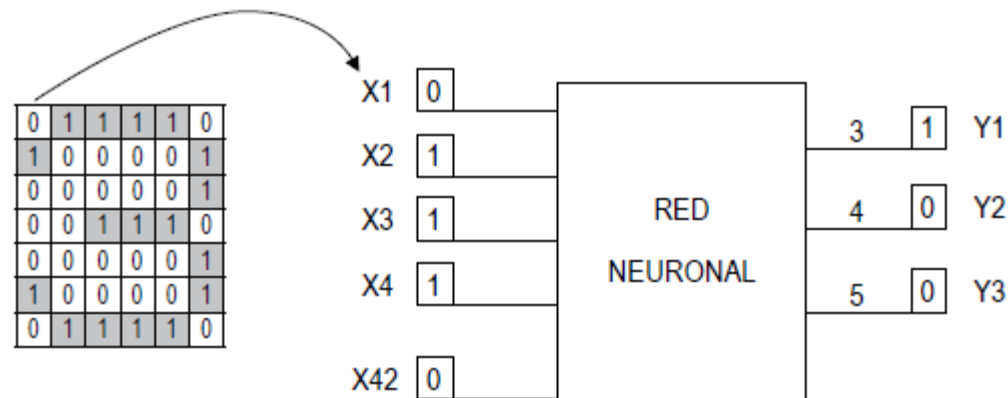
1	1	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	0	1	1	1	1
0	0	0	0	0	1
0	0	0	0	0	1
1	1	1	1	1	1

0	1	1	1	1	0
1	0	0	0	0	1
0	0	0	0	0	1
0	0	1	1	1	0
0	0	0	0	0	1
1	0	0	0	0	1
0	1	1	1	1	0

Reconocimiento de patrones - ejemplo

- ❑ **Paso2:** La entrada es un vector de 42 posiciones (para este ejemplo), en donde cada posición es una posición de la matriz.
- ❑ **Paso3:** Recuerde que los pesos son valores aleatorios entre 0 y 1.

Se utilizará como umbral = 0. La salida debe ser “1” si el patrón es correcto.

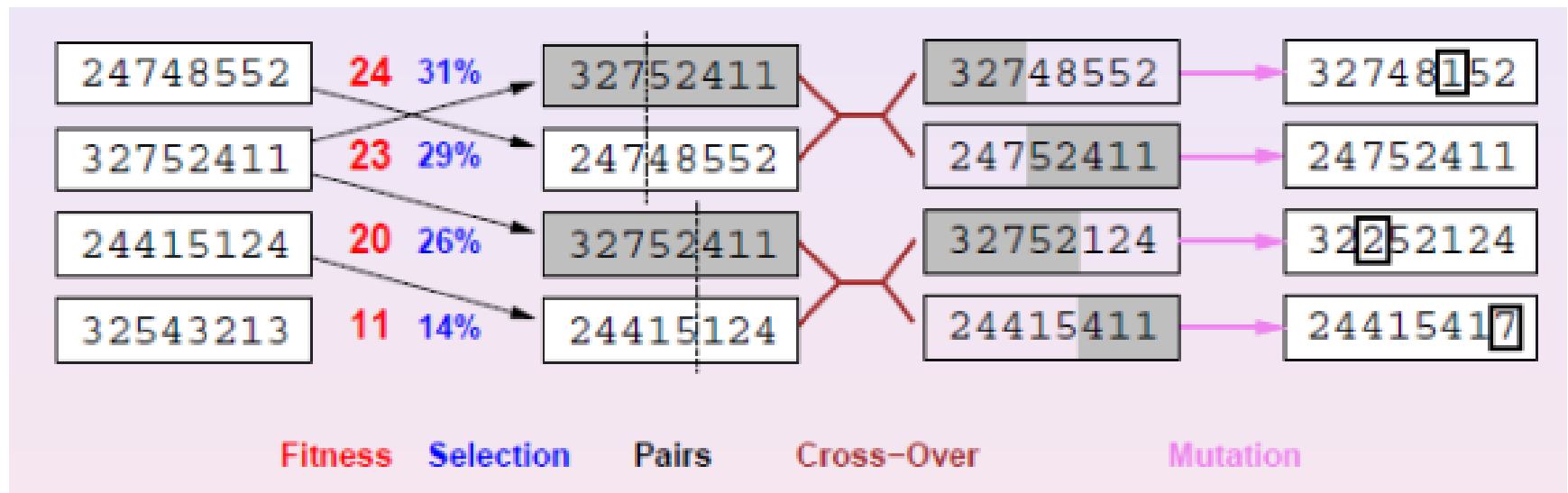




Algoritmos Genéticos

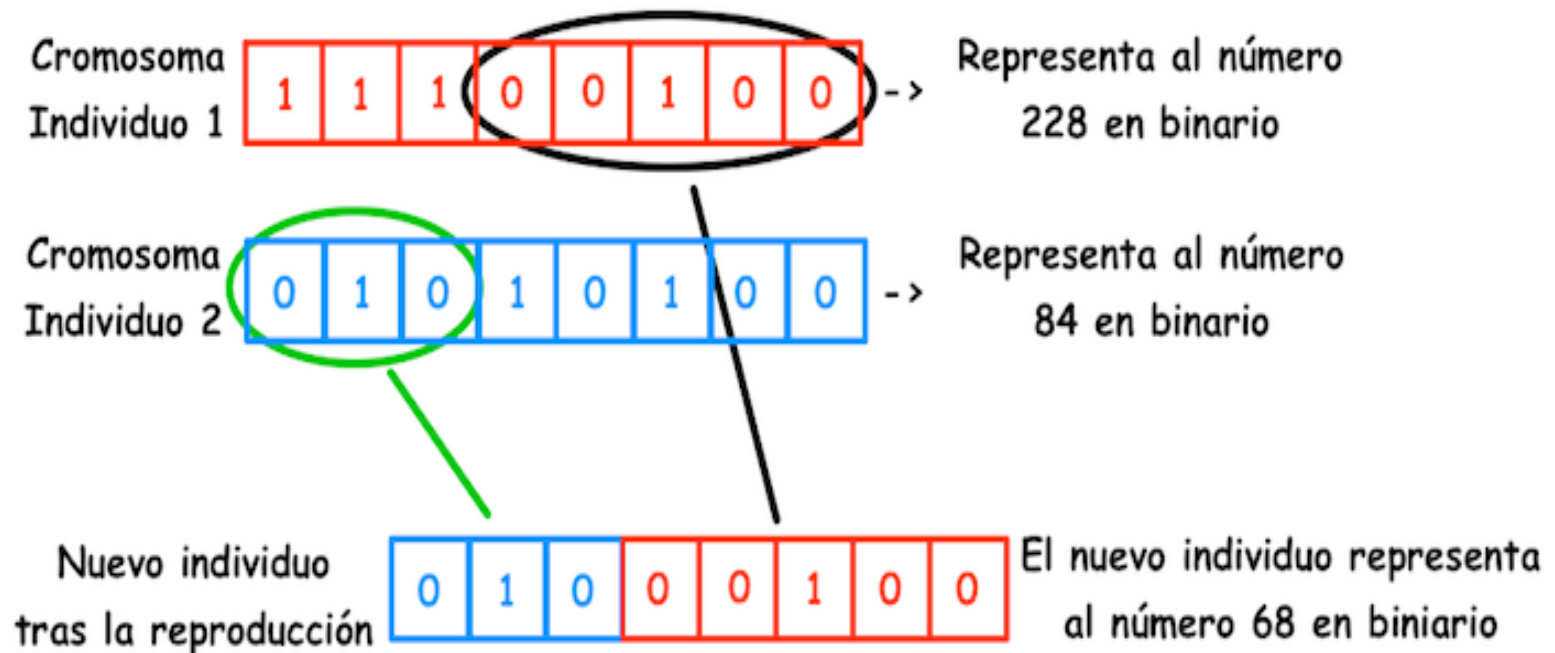
- Algoritmos basados en los principios de la evolución natural.
- Se utilizan en problemas donde no se pueden encontrar soluciones o éstas no son satisfactorias.
- Los AG trabajan encontrando individuos con el mejor desempeño en sus tareas y utilizan sus características para producir “**mejor adaptados**” en cada generación.
- Cada individuo se presenta como una secuencia de caracteres donde cada carácter se llama “**gen**” y el conjunto de caracteres se llama “**cromosoma**”.

Generalidades AG



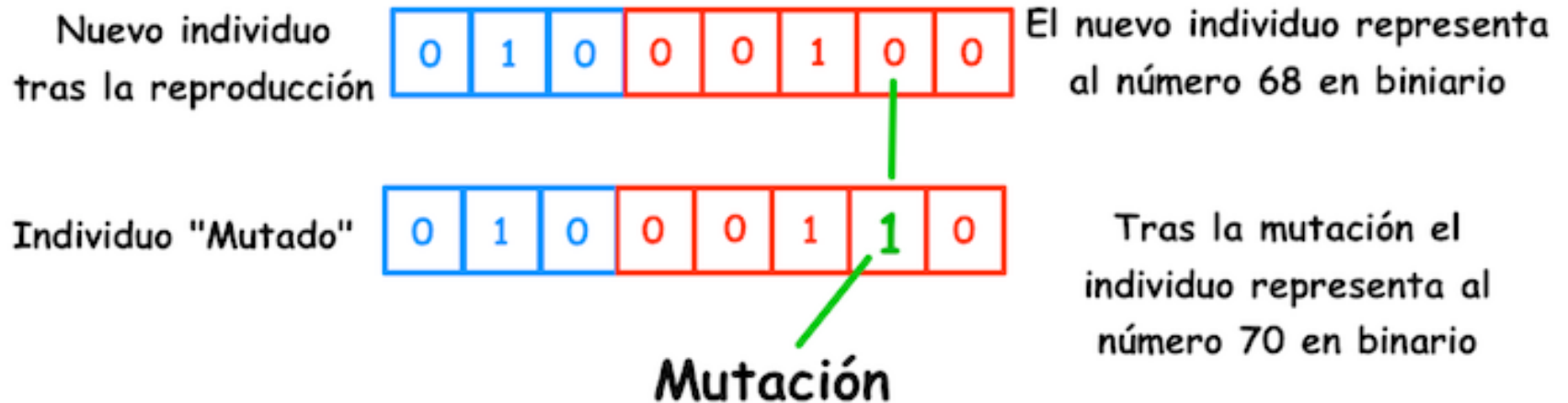
Algoritmos Genéticos

Proceso de reproducción



Algoritmos Genéticos

Mutación



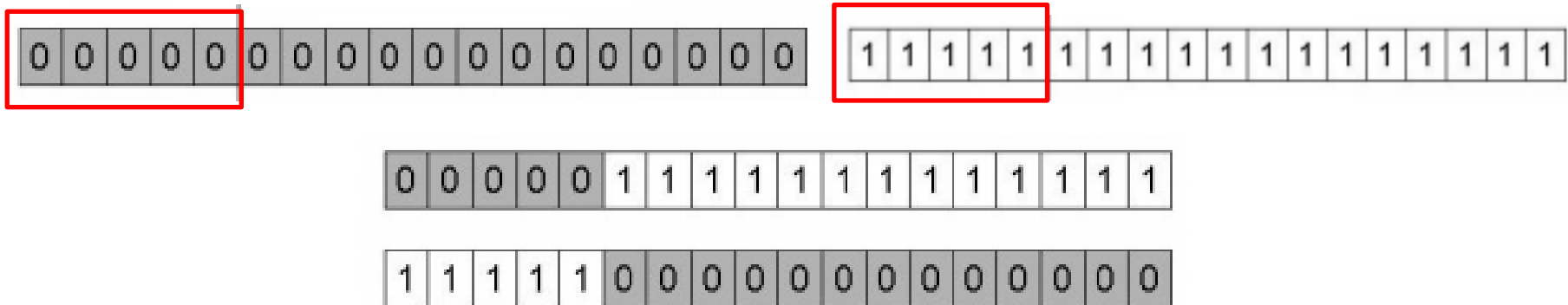
Operadores de Mutación

- *Inversión de un solo bit:*
 - Dado un cromosoma se altera el valor de cada gen con una probabilidad “ P_m ”
- *Inversión por fragmentos:*
 - Toda la cadena (cromosoma) es invertida.
- *Selección aleatoria:*
 - Una cadena elegida al azar es invertida.

Operadores de Cruce

- Cruce en un punto:*

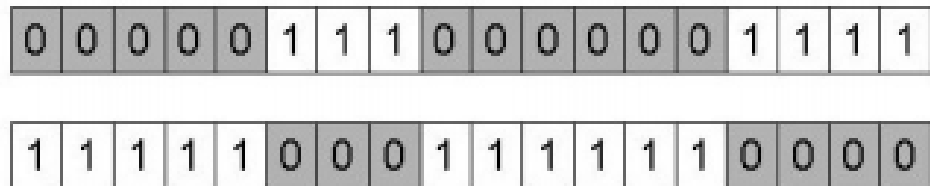
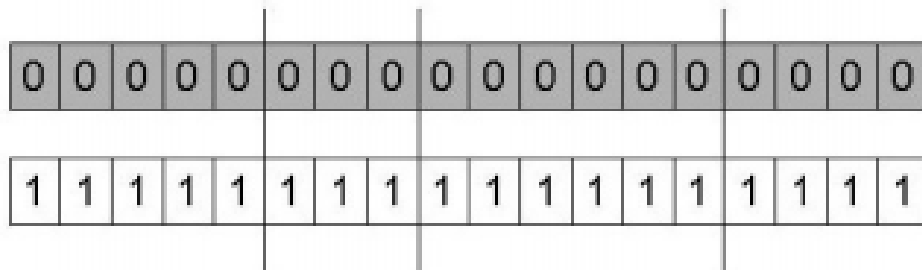
- Se selecciona un punto de cruce de manera aleatoria.
- Se dividen los padres en ese punto.
- Se crean hijos intercambiando partes de los cromosomas.



Operadores de Cruce

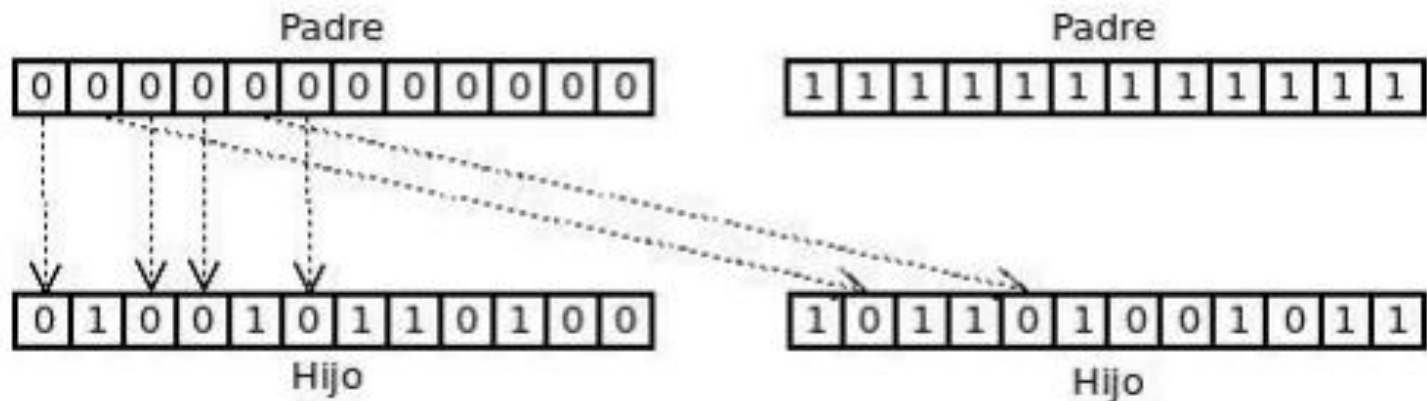
- Cruce en “n” puntos:**

- Se eligen “n” puntos de cruce aleatoriamente.
- Se fragmentan los cromosomas “padres” en esos puntos.
- Se juntan fragmentos, alternando los padres.



Operadores de Cruce

- *Cruce uniforme:*
 - Para cada uno de los genes del cromosoma existe una probabilidad en el gen de la cadena del descendiente de que pertenezca al padre y otra de que pertenezca a la madre.



Algoritmos Genéticos - Ejemplo # 1

- Hallar el máximo de la función $f(x) = x^2$ en el intervalo $[0, 31]$. (Denominada función de calidad o fitness).
- Determinar la gráfica para observar cuál sería el resultado de manera previa (antes de utilizar un AG).
- Para cada punto, se tendrá una codificación binaria de cinco (5) dígitos. Esta sería la población. **La población es de 32.** Porqué...??
- Ahora hay que escoger un **tamaño de la población.** Supongamos que se toman seis (6) individuos.
- Es necesario partir de una población inicial (se generará de manera aleatoria).

Algoritmos Genéticos - Ejemplo # 1

- **Proceso de selección** (competencia de individuos entre sí):

Tabla 1.- SELECCION				
(1)	(2)	(3)	(4)	(5)
1	(0,1,1,0,0)	12	144	6
2	(1,0,0,1,0)	18	324	3
3	(0,1,1,1,1)	15	225	2
4	(1,1,0,0,0)	24	576	5
5	(1,1,0,1,0)	26	676	4
6	(0,0,0,0,1)	1	1	1

$f(x)$ mejor individuo: 676

Mejor individuo: 5

Promedio: 324.3333

- **(1)**:Número del individuo. **(2)**:Codificación binaria
- **(3)**:Valor decimal (x). **(4)**:Valor de $f(x)$.
- A cada individuo de la población se le asigna una pareja. Entre ellas la mejor genera dos copias y el peor se desecha. (5) pareja asignada a cada individuo (**proceso aleatorio**).

Algoritmos Genéticos - Ejemplo # 1

- **Proceso de selección** (competencia de individuos entre sí):

(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	(0,1,1,0,0)	12	144	[6]	1	(0,1,1,0,0)
2	(1,0,0,1,0)	18	324	[3]	225	(1,0,0,1,0)
3	(0,1,1,1,1)	15	225	[2]	324	(1,0,0,1,0)
4	(1,1,0,0,0)	24	576	[5]	676	(1,1,0,1,0)
5	(1,1,0,1,0)	26	676	[4]	576	(1,1,0,1,0)
6	(0,0,0,0,1)	1	1	[1]	144	(0,1,1,0,0)

- **(1)**:Número del individuo.
- **(2)**:Codificación binaria.
- **(3)**:Valor decimal (x).
- **(4)**:Valor de $f(x)$.
- **(5)**:Individuo de cruce.
- **(6)**: $f(x)$ del cruce.
- **(7)**:mejor individuo entre la pareja.

Algoritmos Genéticos - Ejemplo # 1

- **Proceso de cruce** (cambio de un bit elegido de manera aleatoria):

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	(0,1,1,0,0)	12	144	[5]	1	(0 ,1,0,1,0)	10
2	(1,0,0,1,0)	18	324	[4]	1	(1 ,1,0,1,0)	26
3	(1,0,0,1,0)	18	324	[6]	3	(1,0, 0 ,0,0)	16
4	(1,1,0,1,0)	26	676	[2]	3	(1,1, 0 ,1,0)	26
5	(1,1,0,1,0)	26	676	[1]	4	(1,1,0, 1 ,0)	26
6	(0,1,1,0,0)	12	144	[3]	4	(0,1,1, 0 ,0)	12

- **(5)**: Se forman parejas entre los individuos de manera aleatoria.
- **(6)**: Se establece un punto de cruce aleatorio (número aleatorio entre 1 y 4 -la longitud del individuo menos 1-) (Esto se hace para evitar que los descendientes coincidan con los padres).
- **(7)**: resultado del cruce (nueva generación de individuos).
- **(8)**: valor decimal.

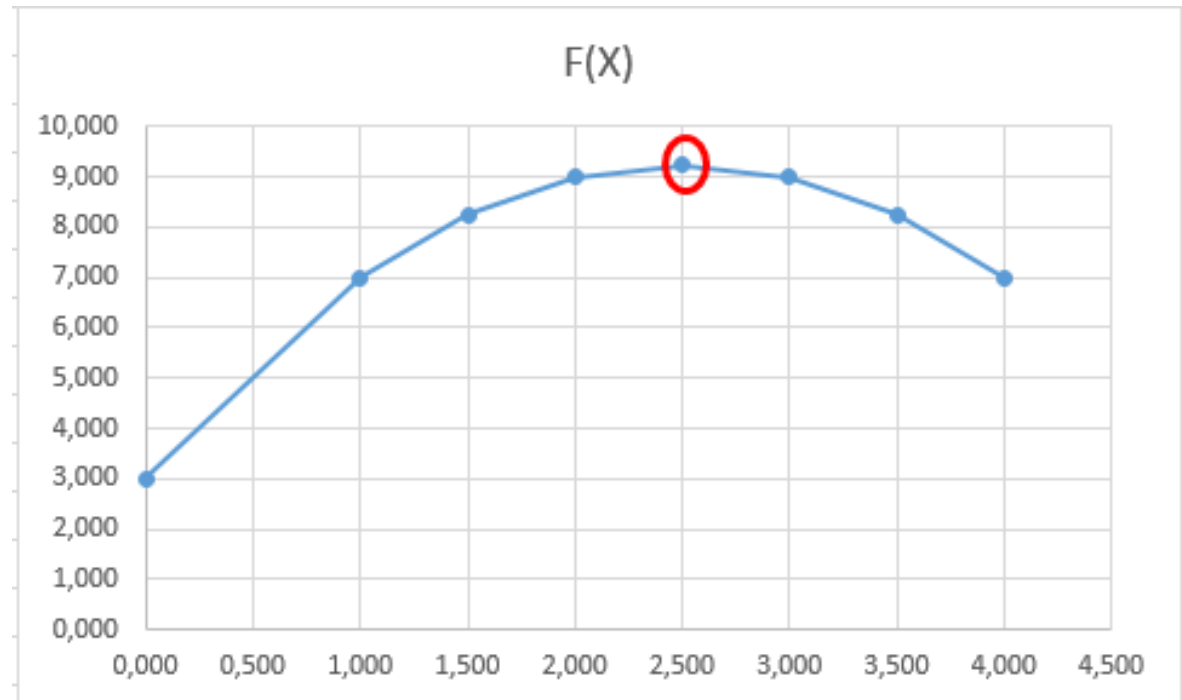
Algoritmos Genéticos - Ejemplo # 1

- La columna (7) es una nueva generación de individuos. Y el promedio de $f(x)$ será mayor (**421.33**). Esto significa que los individuos tras el proceso de selección y cruce son mejores.
- Se podría tomar esta nueva generación como una nueva población inicial. Las iteraciones nos aproximarán cada vez más al óptimo.
- Los algoritmos genéticos intentan encontrar soluciones óptimas a problemas.
- Un algoritmo genético realiza tres operaciones: **selección, cruce y mutación**. En el ejemplo presentado NO se realiza el proceso de **mutación**.

Algoritmos Genéticos - Ejemplo # 2

- Hallar el máximo de la función $f(x) = -x^2 + 5x + 3$ en el intervalo $(0,4)$ con una precisión de tres (3) dígitos decimales.

X	F(X)
0,000	3,000
1,000	7,000
1,500	8,250
2,000	9,000
2,500	9,250
3,000	9,000
3,500	8,250
4,000	7,000



Algoritmos Genéticos - Ejemplo # 2

- Hallar el máximo de la función $f(x) = -x^2 + 5x + 3$ en el intervalo $(0,4)$ con una precisión de tres (3) dígitos decimales.

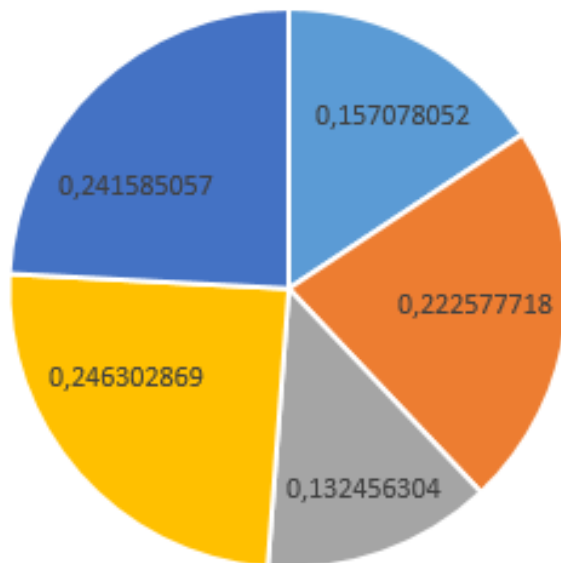
Población Inicial			
Valor de "X"	Cromosoma	Valor de f(x)	Probabilidad
3,791015625	[1 1,1 1 0 0 1 0 1 0 1 0]	5,8865	0,15707805
1,546640625	[0 1,1 0 0 0 1 1 1 0 1 0]	8,3411	0,22257772
0,119140600	[0 0,0 0 0 1 1 1 1 0 1 0]	4,9638	0,1324563
2,359375000	[1 0,0 1 0 1 1 1 0 0 0 0]	9,2302	0,24630287
2,056640600	[1 0,0 0 0 0 1 1 1 0 1 0]	9,0534	0,24158506
	Sumatoria	37,475000000	
	Promedio = 7,495		

Se establece una **probabilidad** para seleccionar las mejores soluciones para el cruce y eventual mutación.

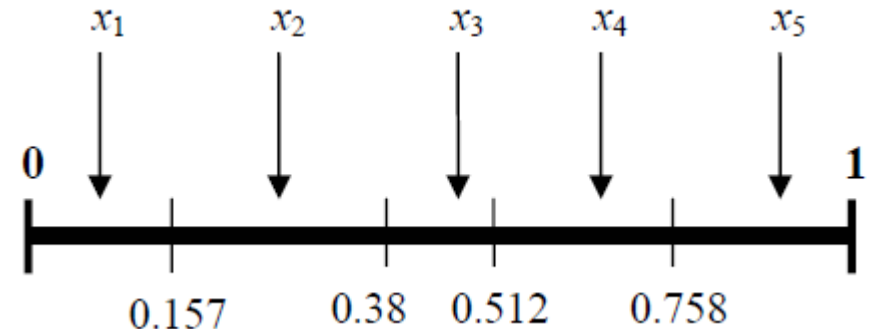
Algoritmos Genéticos - Ejemplo # 2

- Selección por ruleta:

Probabilidad de selección



■ 1 ■ 2 ■ 3 ■ 4 ■ 5



x_1 15.7%

x_2 22.3%

x_3 13.2%

x_4 24.6%

x_5 24.1%

Suponiendo que se obtienen los números aleatorios: **0.532**, **0.776**, **0.723**, **0.285**, **0.846**. De esta manera resultan seleccionados los individuos: **x_4** , **x_5** , **x_4** , **x_2** , **x_5**

Algoritmos Genéticos - Ejemplo # 2

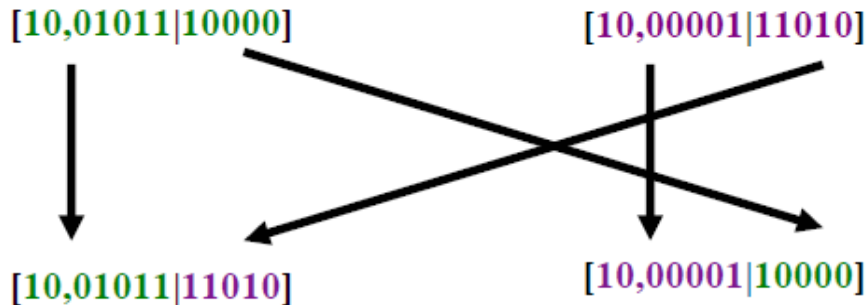
- Individuos seleccionados (el promedio es mayor=**8.58146**):

Cromosoma	Número real: x_i	Valor de la función: $f(x_i)$
$x_4 = [10,0101110000]$	$x_4 = 2,359375000$	$f(x_4) = 9,2302$
$x_5 = [10,0000111010]$	$x_5 = 2,056640600$	$f(x_5) = 9,0534$
$x_4 = [10,0101110000]$	$x_4 = 2,359375000$	$f(x_4) = 9,2302$
$x_2 = [01,1000111010]$	$x_2 = 1,546640625$	$f(x_2) = 8,3411$
$x_5 = [10,0000111010]$	$x_5 = 2,056640600$	$f(x_5) = 9,0534$

- Ahora, se generan números aleatorios para realizar el cruce de los individuos o cromosomas. Suponga que se realizará el cruce entre **x_4** y **x_5** . Se puede observar que el cromosoma **x_2** quedará descartado. Además, el gen de cruce de manera aleatoria suponga que fue siete (7).

Algoritmos Genéticos - Ejemplo # 2

- X_4 {10,0101110000} y X_5 {10,0000111010}



Población inicial con
mejor valor = 9.2302 y
promedio = 7.495

Cromosoma	Número real: x_i	Valor de la función: $f(x_i)$
$x_1 = [1\ 1,1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0]$	$x_1 = 3,791015625$	$f(x_1) = 5,8865$
$x_2 = [0\ 1,1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_2 = 1,546640625$	$f(x_2) = 8,3411$
$x_3 = [0\ 0,0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_3 = 0,119140600$	$f(x_3) = 4,9638$
$x_4 = [1\ 0,0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0]$	$x_4 = 2,359375000$	$f(x_4) = 9,2302$
$x_5 = [1\ 0,0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_5 = 2,056640600$	$f(x_5) = 9,0534$

**Población después del
cruzamiento** con mejor
valor = 9.2329 y promedio
= 7.4938

Cromosoma	Número real: x_i	Valor de la función: $f(x_i)$
$x_1 = [1\ 1,1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0]$	$x_1 = 3,791015625$	$f(x_1) = 5,8865$
$x_2 = [0\ 1,1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_2 = 1,546640625$	$f(x_2) = 8,3411$
$x_3 = [0\ 0,0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_3 = 0,119140600$	$f(x_3) = 4,9638$
$x'_4 = [1\ 0,0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0]$	$x'_4 = 2,369140625$	$f(x'_4) = 9,2329$
$x'_5 = [1\ 0,0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0]$	$x'_5 = 2,046875000$	$f(x'_5) = 9,0447$

Algoritmos Genéticos - Ejemplo # 2

- Pero recordemos que los nuevos individuos deben reemplazar a los dos peores de la población inicial. Entonces tenemos:

Cromosoma	Número real: x_i	Valor de la función: $f(x_i)$
$x'_4 = [1\ 0,0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0]$	$x'_4 = 2,369140625$	$f(x'_4) = 9,2329$
$x_2 = [0\ 1,1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_2 = 1,546640625$	$f(x_2) = 8,3411$
$x'_5 = [1\ 0,0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0]$	$x'_5 = 2,046875000$	$f(x'_5) = 9,0447$
$x_4 = [1\ 0,0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0]$	$x_4 = 2,359375000$	$f(x_4) = 9,2302$
$x_5 = [1\ 0,0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0]$	$x_5 = 2,056640600$	$f(x_5) = 9,0534$

Nueva población con mejores individuos con promedio de 8.98046.
 Mejor solución cuando el valor de “**X**” es **2.369140625**

Algoritmos Genéticos - Ejemplo # 3

- Hallar el máximo de la función $f(x) = -x^2 + 12x - 11$ en el intervalo $[0,15]$. **Paso 0:** Elaborar la gráfica.
- **Paso 1 (Selección):** Entre el conjunto de individuos pertenecientes al dominio del problema, se elige un número entre ellos de manera aleatoria. El dominio del problema comprende los números del 0 al 15 (codificación binaria desde $\{0,0,0,0\}$ a $\{1,1,1,1\}$. Tomaremos una población de 8 individuos. (ver siguiente diapositiva).
- **Paso 2 (Cruce):** Cada nuevo individuo es el resultado de la combinación de sus progenitores. En la siguiente diapositiva podemos ver la columna (5) con el cruce de individuos de la primera generación. Y en la columna (7) se tiene el mejor individuo.

Algoritmos Genéticos - Ejemplo # 3

- Proceso de selección y cruce:

(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	(0,0,1,1)	3	34	[2]	277	(1,1,0,0)
2	(1,1,0,0)	12	277	[1]	34	(1,1,0,0)
3	(0,1,1,0)	6	97	[4]	53	(0,1,1,0)
4	(0,1,0,0)	4	53	[3]	97	(0,1,1,0)
5	(0,1,1,0)	6	97	[6]	2	(0,1,1,0)
6	(0,0,0,1)	1	2	[5]	97	(0,1,1,0)
7	(0,0,1,0)	2	17	[8]	353	(1,1,1,0)
8	(1,1,1,0)	14	353	[7]	17	(1,1,1,0)

- (1):# del individuo. (2):Codificación binaria.
- (3):Valor decimal (x). (4):Valor de $f(x)$.
- (5):Individuo de cruce. (6): $f(x)$ del cruce.
- (7):mejor individuo.

Algoritmos Genéticos - Ejemplo # 3

- Proceso de Cruce:

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	(1,1,0,0)	12	277	[3]	1	(1 ,1,1,0)	14
2	(1,1,0,0)	12	277	[4]	2	(1, 1 ,1,0)	14
3	(0,1,1,0)	6	97	[1]	3	(0,1, 1 ,0)	6
4	(0,1,1,0)	6	97	[2]	1	(0 ,1,0,0)	4
5	(0,1,1,0)	6	97	[7]	2	(0, 1 ,1,0)	6
6	(0,1,1,0)	6	97	[8]	3	(0,1, 1 ,0)	6
7	(1,1,1,0)	14	353	[5]	1	(1 ,1,1,0)	14
8	(1,1,1,0)	14	353	[6]	2	(1, 1 ,1,0)	14

- (1):# del individuo. (2):Codificación binaria.
- (3):Valor decimal (x). (4):Valor de f(x).
- (5):Individuo de cruce. (6):Punto de corte
- (7):Resultado del cruce (nueva generación).
- (8):Valor decimal.

Buscando una frase – Ejemplo #4

- Se requiere un algoritmo genético para adivinar una frase. La población de frases consiste en series de cadenas de caracteres al azar pero que tienen la misma longitud que la frase objetivo.
- El algoritmo genético recibirá información acerca de qué tan parecida es a la frase objetivo. Para determinar qué tan cerca se encuentra a la frase objetivo podemos indicar el número de letras correctas que posee la frase.
- La función fitness (función de calidad) se tomará como $f(x) = 2^x$ en donde “x” es el número de letras (en su posición) correctas.
- Por ejemplo, si la frase objetivo fuese: **UN ALUMNO**, entonces la frase **MU ALUNNO** tendría un fitness de 64. Porqué...???? (El espacio se tiene en cuenta).

Buscando una frase – Ejemplo #4

- Si la frase objetivo seleccionada será: **EL SEGUNDO CORTE ES NICE** (solo se tendrán en cuenta letras mayúsculas).
- La frase tiene una longitud de 23 (incluidos los espacios en blanco). Se utilizará un espacio de 27 caracteres (alfabeto) más el espacio en blanco. Es decir se podrían generar 27^{23} frases distintas (cromosomas) de longitud 23.
- En este tipo de ejercicio se puede realizar un análisis comparativo entre un algoritmo de fuerza bruta y un algoritmo genético. ¿En condiciones ideales, cuál de los dos algoritmos terminaría el proceso más rápidamente?

Algoritmos Genéticos - Ejemplo # 5

- La frase objetivo seleccionada será: **MUNDO** (solo se tendrán en cuenta letras mayúsculas).
- La frase tiene una longitud de 5 caracteres. Se utilizará un espacio de 27 letras. Es decir se podrían generar 27^5 palabras distintas (cromosomas) de longitud 5 (14.348.907 posibilidades). Se tomará como población inicial 68.750 palabras (ver archivo **CROMOSOMAS.CSV** en el Aula Virtual).
- Después de algunas iteraciones se seleccionaron las palabras: **MNNMO**, **UUDDO**, **MNDDO**, **MUNDD**, **MUUDO**, **MUNOO**.

Algoritmos Genéticos - Ejemplo # 5

- MNNMO, UUUDDO, MNDDO, MUNDD, MUUDO, MUNOO.

(1)	(2)	(3)	(4)	(5)	(6)
1	MNNMO	3	8	[4]	MUNDD
2	UUDDO	3	8	[3]	UUDDO
3	MNDDO	3	8	[2]	MNDDO
4	MUNDD	4	16	[1]	MUNDD
5	MUUDO	4	16	[6]	MUUDO
6	MUNOO	4	16	[5]	MUNOO

- (3): Cantidad de letras correctas.
- (4): Función fitness (Sumatoria=72. Promedio=12).
- (5): Individuo seleccionado para cruce.
- (6): Mejor cromosoma(individuo).

Algoritmos Genéticos - Ejemplo # 5

- MNNMO, UUDDO, MNDDO, MUNDD, MUUDO, MUNOO.

(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	MUNDD	[2]	1	M UDDO	4	16
2	UUDDO	[1]	2	U UDDO	3	8
3	MNDDO	[5]	3	MN DDO	3	8
4	MUNDD	[6]	4	MUN DO	5	32
5	MUUDO	[3]	1	M NDDO	3	8
6	MUNOO	[4]	2	MU NDD	4	16

- (3)**:Emparejamiento (números aleatorios).
- (4)**:Punto de cruce(bit) [Aleatorio].
- (5)**:Resultado del emparejamiento.
- (6)**:Letras correctas.
- (7)**:Resultado de la función.

Fitness: 88.
Promedio: 14.6

<http://robologs.net/2015/09/01/como-programar-un-algoritmo-genetico-parte-ii-implementacion-en-python/>

```
Modelo: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
Poblacion Inicial:
```

```
[[2, 5, 2, 6, 7, 1, 2, 3, 1, 5], [8, 9, 4, 5, 4, 6, 2, 6, 8, 1],  
[5, 6, 9, 4, 9, 1, 3, 9, 6, 8], [2, 2, 5, 2, 9, 7, 7, 6, 5, 2],  
[7, 7, 6, 9, 4, 3, 5, 6, 2, 6], [7, 4, 6, 8, 3, 6, 4, 6, 3, 5],  
[9, 1, 5, 7, 9, 6, 1, 7, 5, 2], [3, 1, 8, 3, 7, 3, 5, 2, 3, 3],  
[7, 5, 8, 2, 9, 4, 8, 9, 4, 2], [3, 6, 2, 1, 1, 7, 2, 4, 2, 3]]
```

```
Poblacion Final:
```

```
[[9, 1, 1, 1, 1, 9, 1, 9, 1, 1], [9, 1, 1, 1, 1, 9, 1, 9, 1, 1],  
[9, 1, 1, 1, 1, 9, 1, 9, 1, 1], [9, 1, 1, 1, 6, 9, 1, 9, 1, 1],  
[9, 1, 1, 1, 1, 9, 1, 9, 1, 1], [9, 1, 1, 1, 1, 9, 1, 9, 1, 1],  
[9, 1, 1, 1, 1, 9, 1, 9, 1, 1], [9, 1, 1, 1, 1, 9, 1, 9, 1, 1]]
```


Algoritmos Genéticos - Ejemplo # 7

- Maximizar la función $f(x) = \text{ABS}[(x - 5) / (2 + \text{Sen}(x))]$ en el intervalo $[0, 15]$
- Graficar la función (responsabilidad del estudiante) para comprobar que el máximo de la función se encuentra cuando el valor de $X = 11$.
- Se tienen en cuenta las siguientes consideraciones para resolver el ejercicio:
 - Longitud del cromosoma : 4
 - Conjunto de elementos del cromosoma : $\{0,1\}$
 - Número de individuos de la población: 2
 - Probabilidad de emparejamiento: 0.7
 - Probabilidad de mutación: 0.3

Algoritmos Genéticos - Ejemplo # 7

- **Paso1:** Para generar los cromosomas (4) de cada individuo desde la primera generación, vamos a aplicar la siguiente regla: Se genera un número aleatorio (entre 0 y 1) y si es menor que 0.5 se colocará un “0”. En caso contrario se colocará un “1”. Si el número aleatorio es 0.5, se volverá a generar un nuevo número aleatorio.
- Al aplicar la regla, los cromosomas fueron: $c1 = \{0,1,1,1\}$ y $c2 = \{0,1,0,0\}$ que representan al **7** y al **4** respectivamente.
- Se aplica la función de fitness para cada cromosoma(individuo) y se obtiene como resultado **$f(7) = 0.75$** y **$f(4) = 0.8$** . (el valor de “**x**” está dado en radianes).

$$\text{Individuo 1 (x=7):} \quad f(7) = ABS \left| \frac{7-5}{2+\text{Sen}(7)} \right| = 0.75$$

$$\text{Individuo 2 (x=4):} \quad f(4) = ABS \left| \frac{4-5}{2+\text{Sen}(4)} \right| = 0.8$$

Ahora calculamos la probabilidad de cada individuo para el emparejamiento con base en la siguiente regla de tres simple:

$$\text{Individuo 1} = \frac{0.75}{0.75 + 0.8} = 0.48$$

$$\text{Individuo 2} = \frac{0.8}{0.75 + 0.8} = 0.52$$

Algoritmos Genéticos - Ejemplo # 7

- Con estos resultados se busca determinar qué individuos son **seleccionados** para el emparejamiento. Se generan números aleatorios para cada caso. (En caso de que haya emparejamiento es esa generación).
- Rango del Individuo 1 para emparejamiento $[0 - 0.48]$. Si el número aleatorio fuese 0.41 significa que este individuo es apto para el emparejamiento. Rango del individuo 2 para el emparejamiento $[0.48 - 1]$. Si fuese 0.74 el número aleatorio generado significa que este individuo también es apto para el emparejamiento. Ambos individuos son aptos para emparejamiento.

Rango del individuo 1 $[0-0.48]$

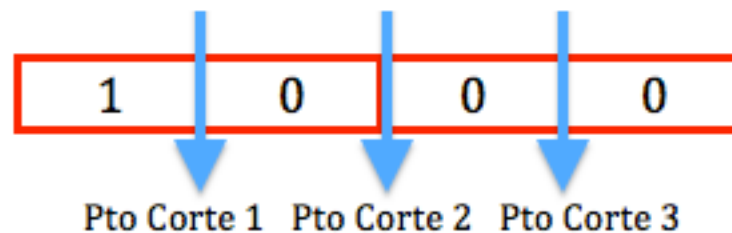
0.41 esta dentro del rango =>
El individuo 1 es apto para el emparejamiento

Rango del individuo 2 $[0.48-1]$

0.74 esta dentro del rango =>
El individuo 2 es apto para el emparejamiento

Algoritmos Genéticos - Ejemplo # 7

- **Aplicando la probabilidad de emparejamiento (0.7).**
Para determinar si se puede o no realizar el emparejamiento, se genera un nuevo número aleatorio. El rango de emparejamiento será $[0 - 0.7]$ y el rango de NO emparejamiento será de $[0.7 - 1]$. Si el nuevo número aleatorio es 0.15 significa que hay emparejamiento en esta generación.
- **Punto de cruce (corte):** Si se selecciona el punto de cruce en el tercer cromosoma el emparejamiento dará como resultado lo siguiente:



Rango Pto Corte 1 $[0-0.33]$

Rango Pto Corte 2 $[0.33-0.66]$

Rango Pto Corte 3 $[0.66-1]$

Algoritmos Genéticos - Ejemplo # 7

- Si el número aleatorio es 0.85, significa que se selecciona al punto 3 como punto de corte. En este caso tendremos:
- Nuevo individuo C1 = {0,1,1,0}. Nuevo individuo C2={0,1,0,1}.
- **Aplicando la probabilidad de mutación (0.3) [Nuevo]:** por último verificamos si alguno de estos nuevos individuos sufrirá alguna mutación teniendo en cuenta cada elemento del cromosoma (gen) con respecto a la probabilidad de mutación.
- Se generan ocho (8) números aleatorios (4 para cada cromosoma) y si el número aleatorio generado se encuentra en el rango entre [0 - 0.3] habrá mutación en caso contrario no se realiza la mutación.

Algoritmos Genéticos - Ejemplo # 7

- Para el individuo C1 se tienen los números: 0.51, 0.44, 0.89, 0.85. Significa que no hay mutación. $C1=\{0,1,1,0\}$. Representa el número 6.
- Para el individuo C2 se tienen los números: 0.43, **0.07**, 0.97, 0.93. Se muta el elemento 2 y queda como resultado $C2=\{0,\mathbf{0},0,1\}$

Cromosoma del individuo 1

0.51	0.44	0.89	0.85
↓	↓	↓	↓
0	1	1	0

Representa a $x=6$

Cromosoma del individuo 2

0.43	0.07	0.97	0.93
↓	↓	↓	↓
0	0	0	1

Representa a $x=1$

Muta el 2º elemento del cromosoma

Algoritmos Genéticos - Ejemplo # 7

- Al llegar a este punto se tiene una nueva generación de individuos. Si aplicamos la función fitness tendremos:
- Individuo 1 ($x = 6$): $f(6) = 0.58$
- Individuo 2 ($x = 1$): $f(1) = 1.41$
- Para llegar a mejores individuos será necesario utilizar más individuos y realizar este mismo proceso con más generaciones. Es decir, repetir el proceso varias veces. Con esta sola generación no nos hemos acercado al mejor individuo que sería el individuo con valor $x = 11$.

$$\text{Individuo 1 (x=6):} \quad f(6) = \text{ABS} \left| \frac{6-5}{2+\text{Sen}(6)} \right| = 0.58$$

$$\text{Individuo 2 (x=1):} \quad f(1) = \text{ABS} \left| \frac{1-5}{2+\text{Sen}(1)} \right| = 1.41$$

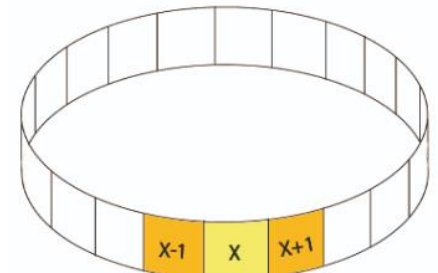


- **Autómata:** Es un modelo matemático de una máquina abstracta definida formalmente sobre un grupo de estados, una serie de transiciones entre esos estados y los datos que rigen dichas transiciones (regla de transición: componente dinámico del modelo).

- **Autómata celular:** pequeñas entidades independientes pero que interaccionan entre sí. Celulares porque son la unidad elemental del universo donde van a existir y autómatas porque deciden por ellas mismas, basadas en un conjunto de reglas predefinido, cuando el tiempo avanza de forma discreta (es decir, a *pasos*).

Autómatas Celulares

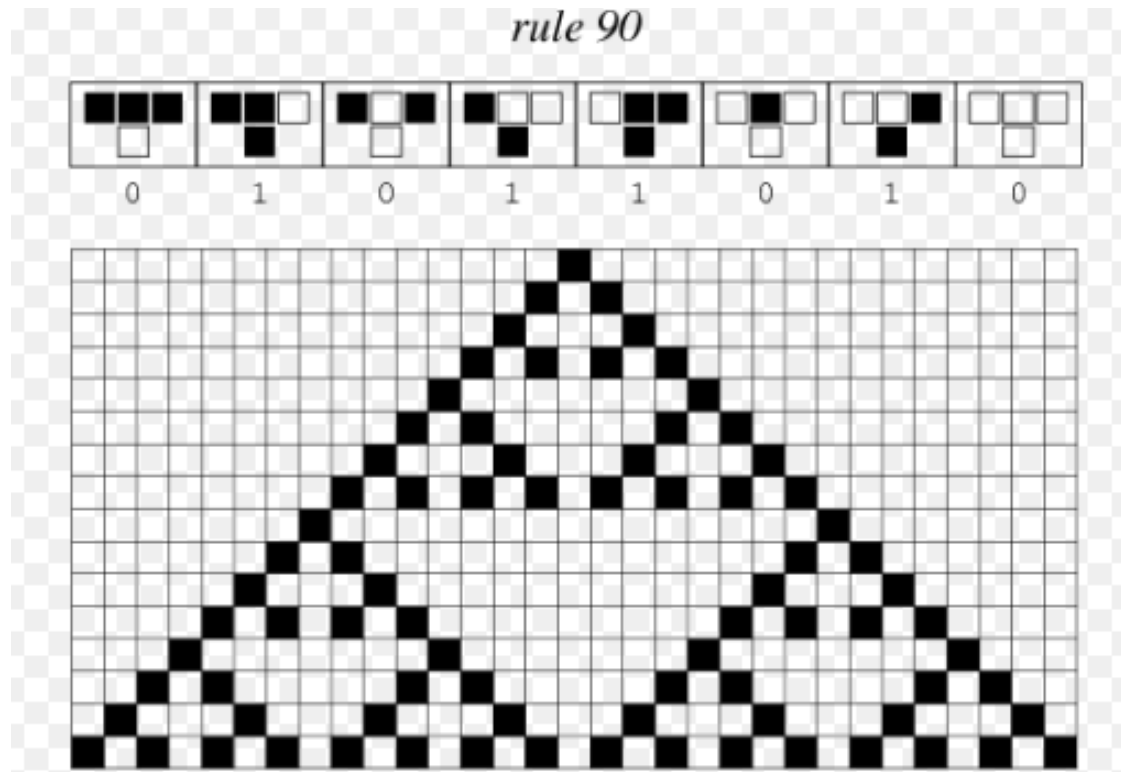
- ❑ Cada célula tiene una vecina a su izquierda y a su derecha.
- ❑ Es adecuado para modelar sistemas naturales o artificiales.
- ❑ Entre sus usos se encuentra la IA.
- ❑ Se puede utilizar en Criptografía, etc.
- ❑ Poseen una gran complejidad.



Autómata celular unidimensional con un radio de vecindad $r=1$ y una condición de frontera periódica.

Autómatas Celulares

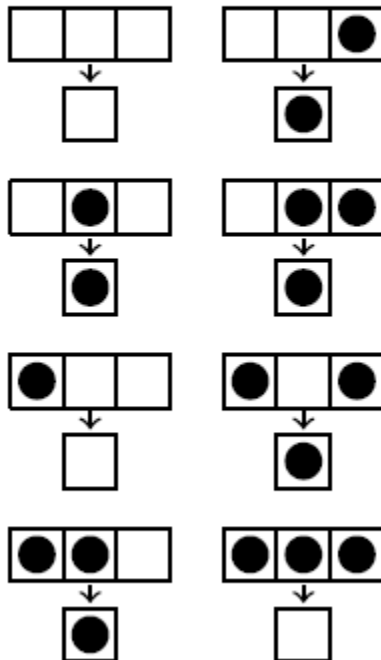
- Las **reglas de evolución** definen como se darán los cambios de estado de cada celda, dependiendo del estado anterior, propio y de su vecindad.



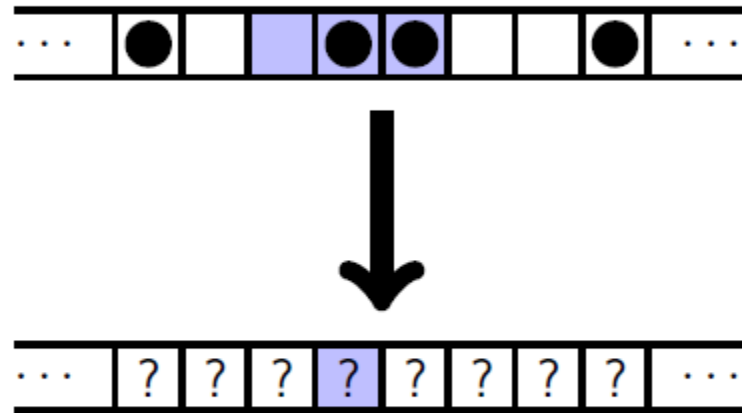
Autómatas Celulares

Las Reglas

Las reglas



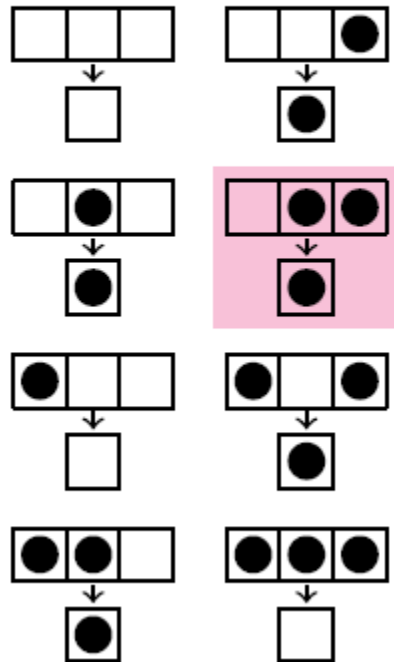
¿Cómo se usan?



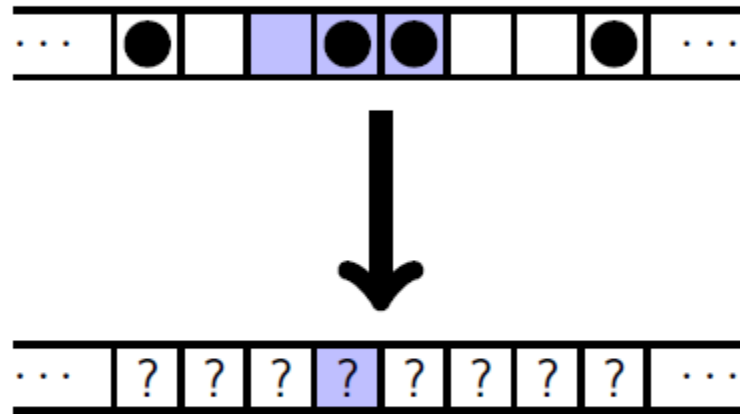
Autómatas Celulares

Las Reglas

Las reglas



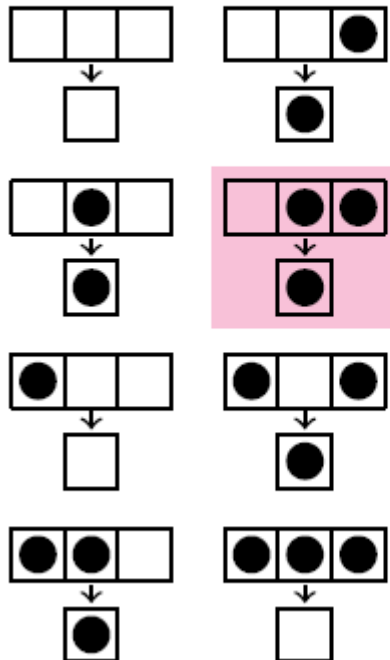
¿Cómo se usan?



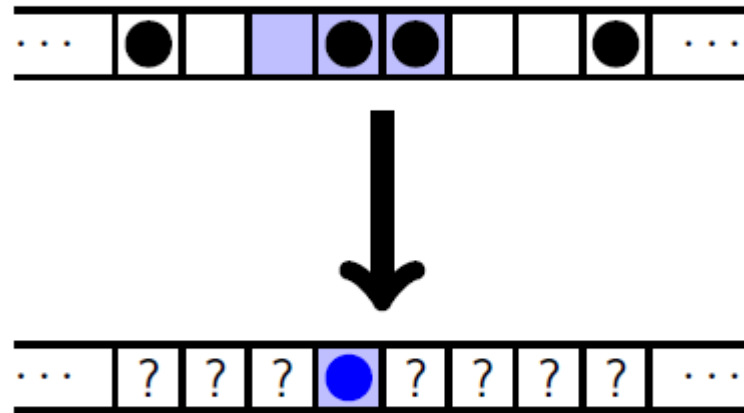
Autómatas Celulares

Las Reglas

Las reglas



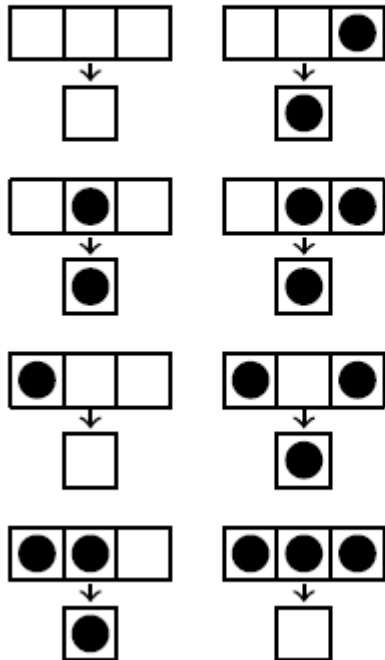
¿Cómo se usan?



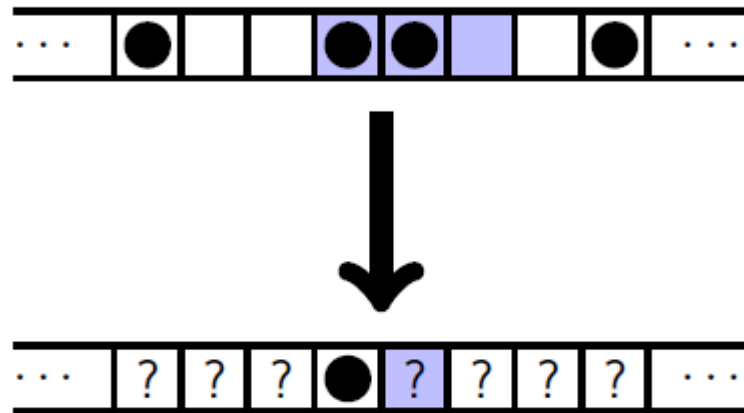
Autómatas Celulares

Las Reglas

Las reglas



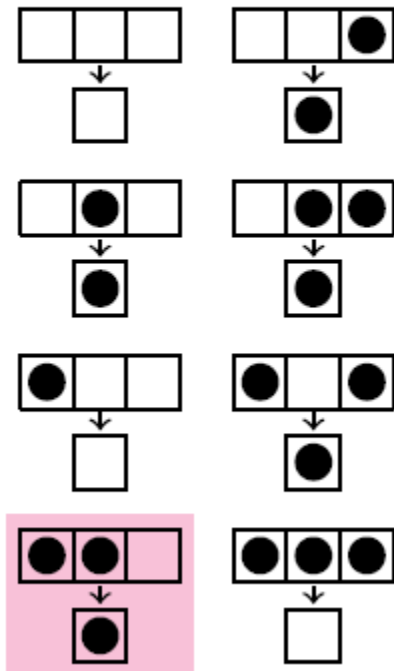
¿Cómo se usan?



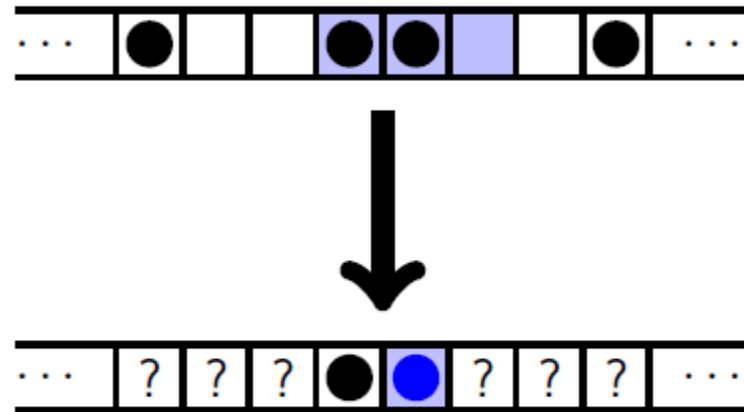
Autómatas Celulares

Las Reglas

Las reglas



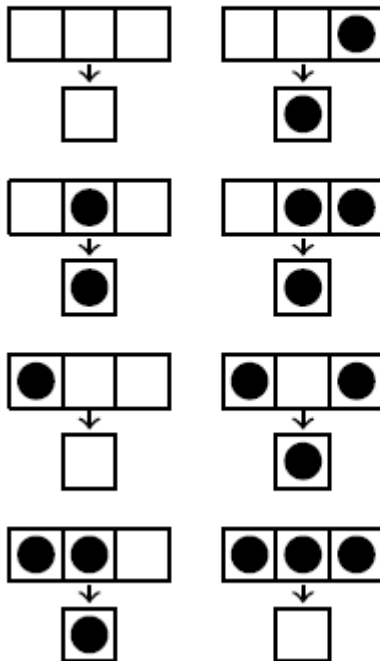
¿Cómo se usan?



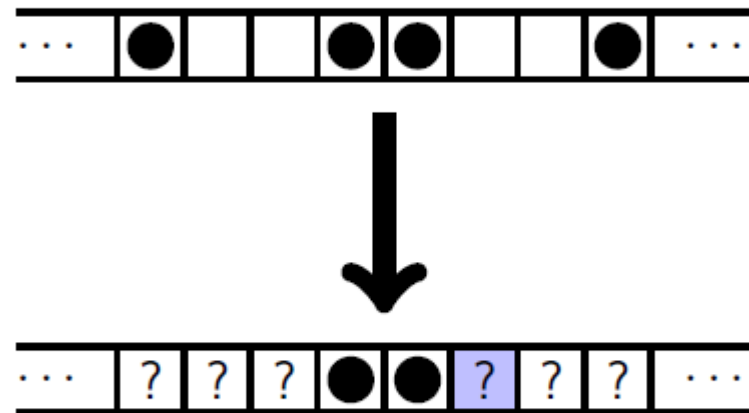
Autómatas Celulares

Las Reglas

Las reglas



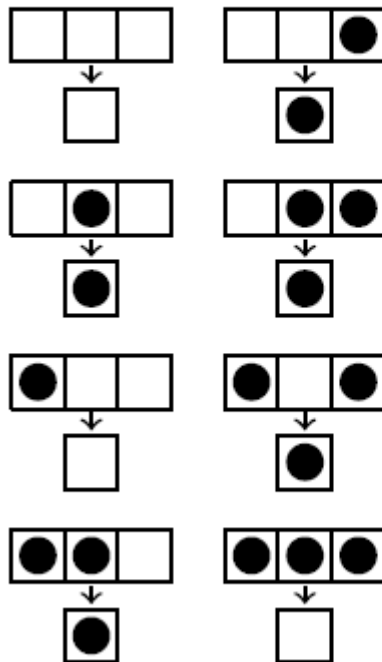
¿Cómo se usan?



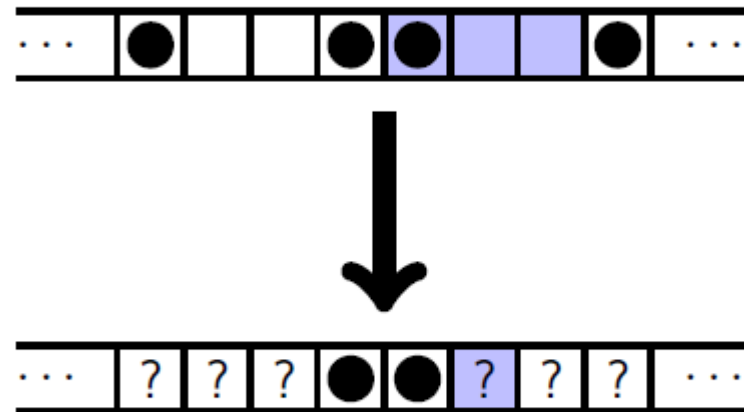
Autómatas Celulares

Las Reglas

Las reglas



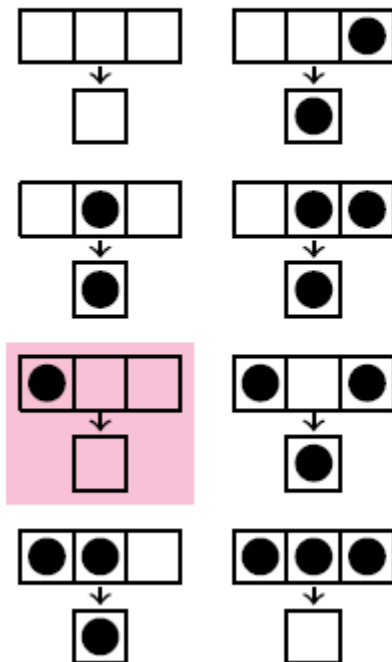
¿Cómo se usan?



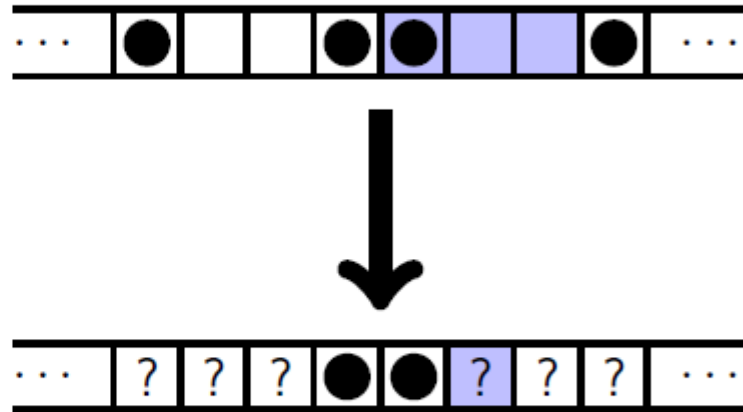
Autómatas Celulares

Las Reglas

Las reglas



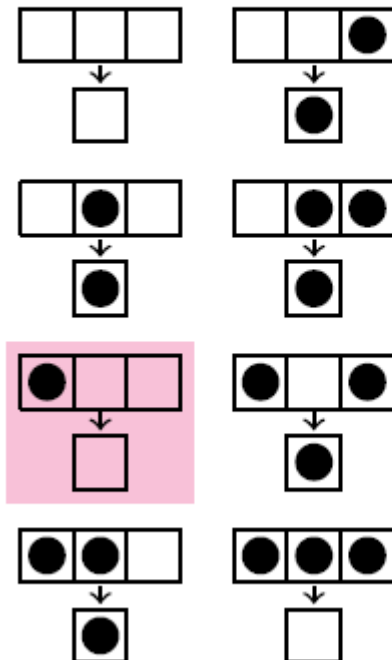
¿Cómo se usan?



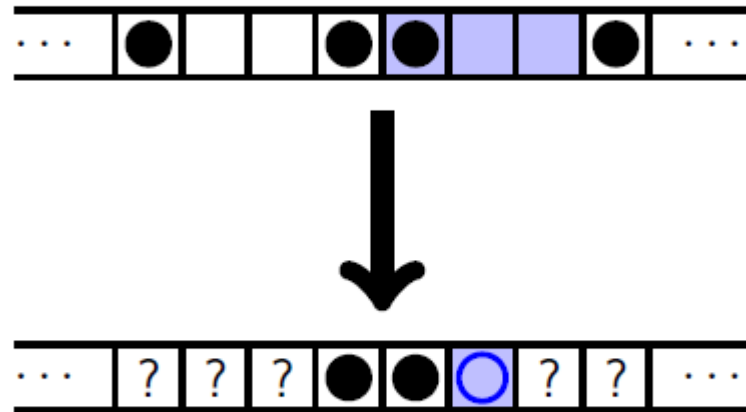
Autómatas Celulares

Las Reglas

Las reglas



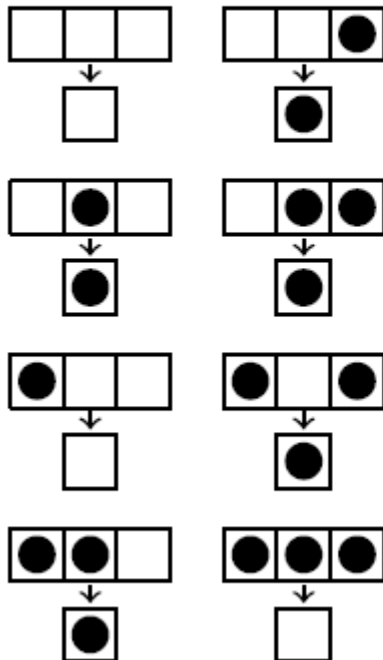
¿Cómo se usan?



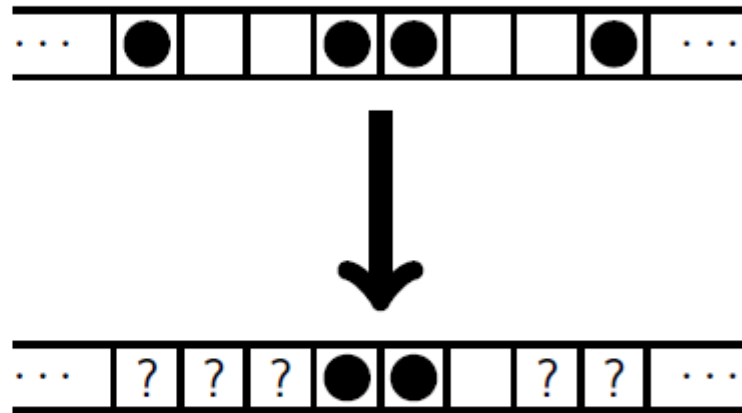
Autómatas Celulares

Las Reglas

Las reglas



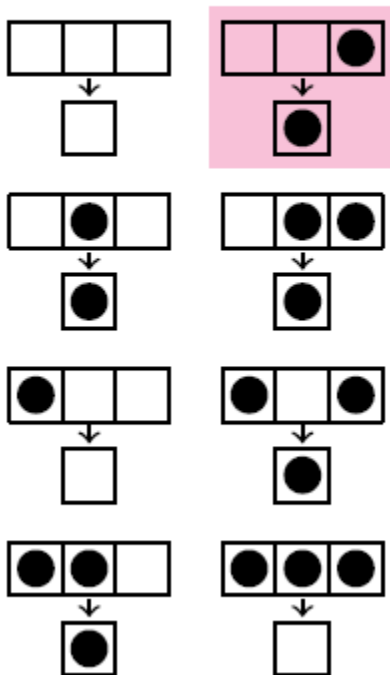
¿Cómo se usan?



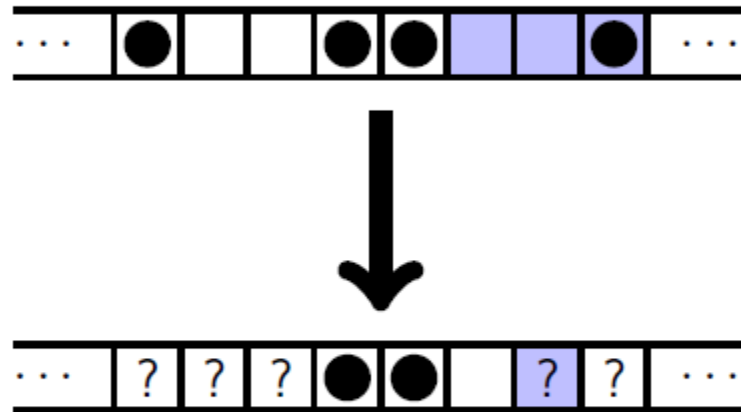
Autómatas Celulares

Las Reglas

Las reglas



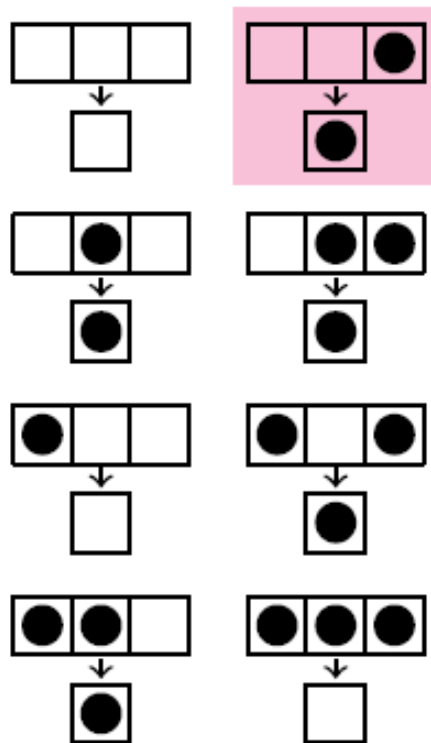
¿Cómo se usan?



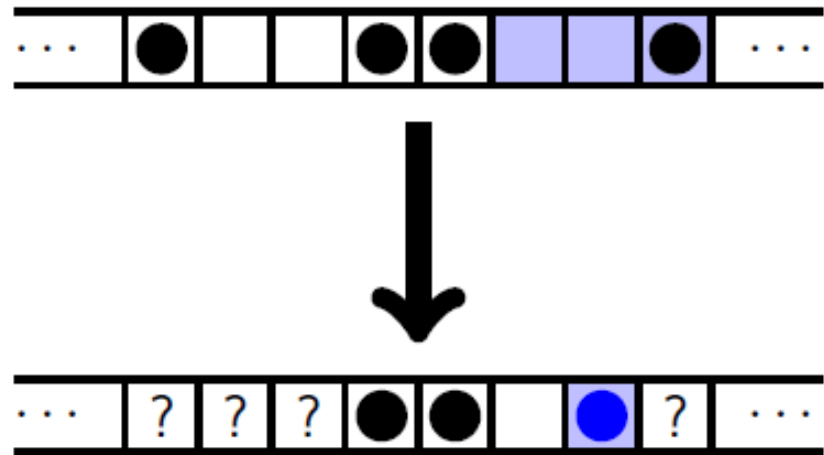
Autómatas Celulares

Las Reglas

Las reglas



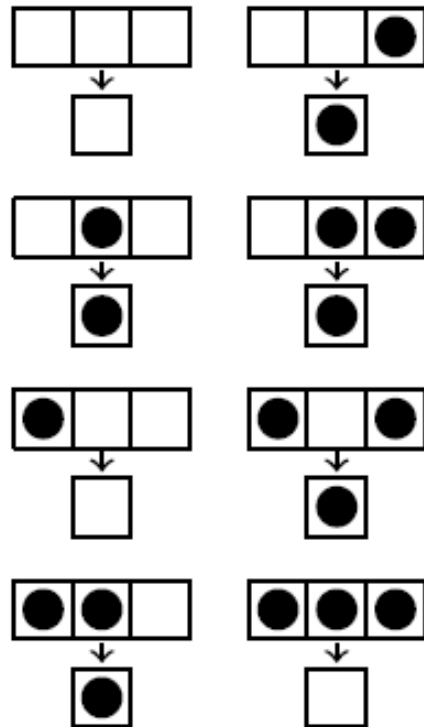
¿Cómo se usan?



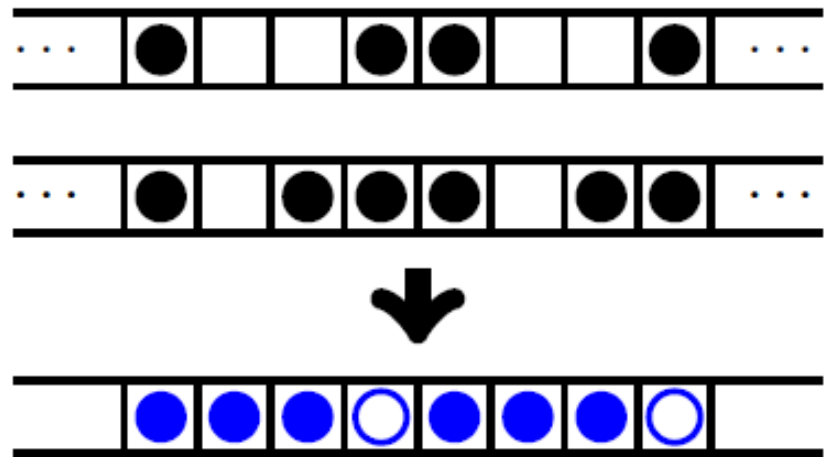
Autómatas Celulares

Las Reglas

Las reglas



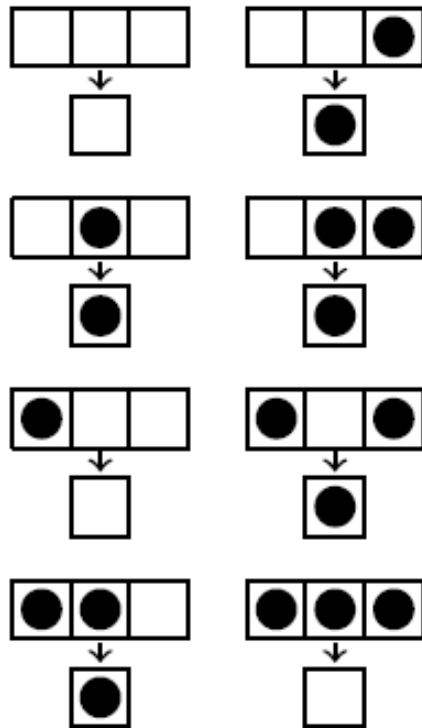
¿Cómo se usan?



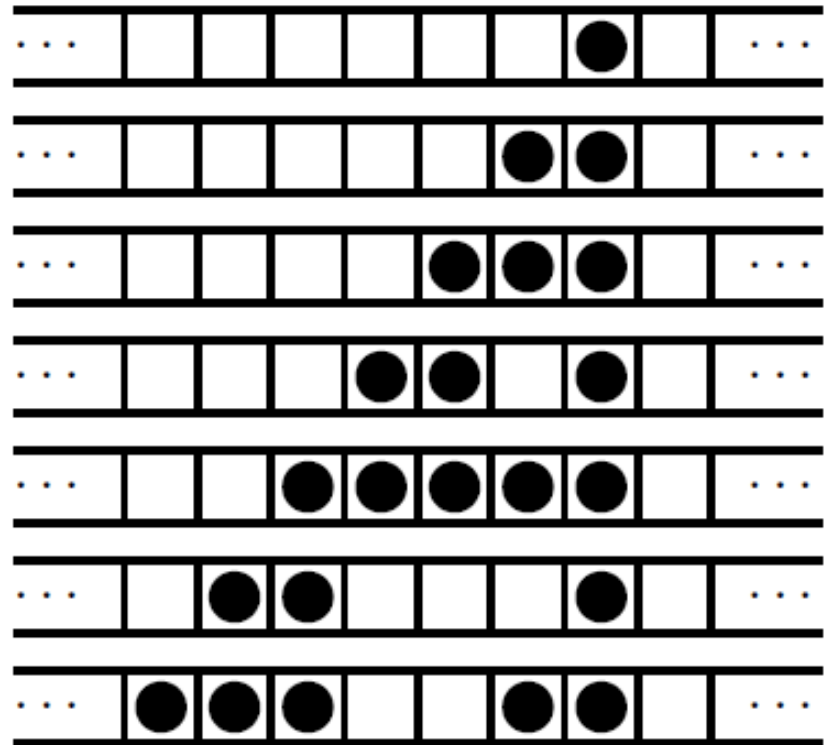
Autómatas Celulares

Las Reglas

Las reglas



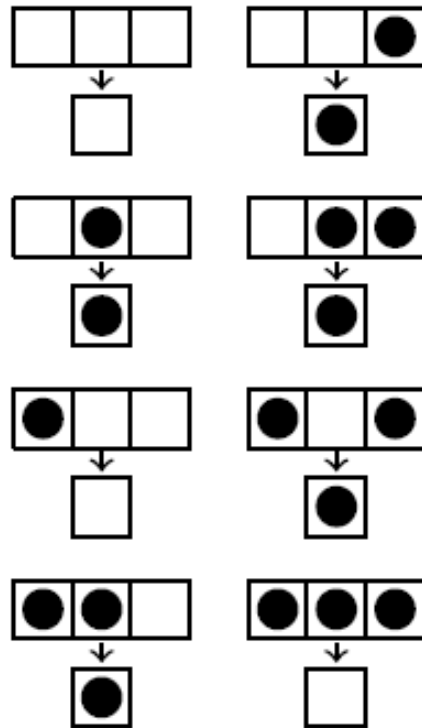
Otro ejemplo



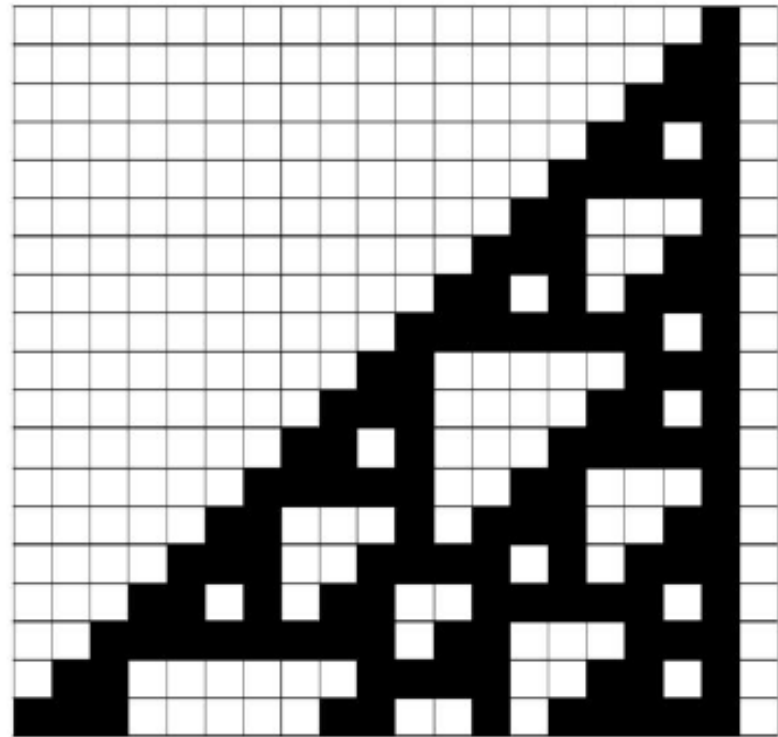
Autómatas Celulares

Las Reglas

Las reglas



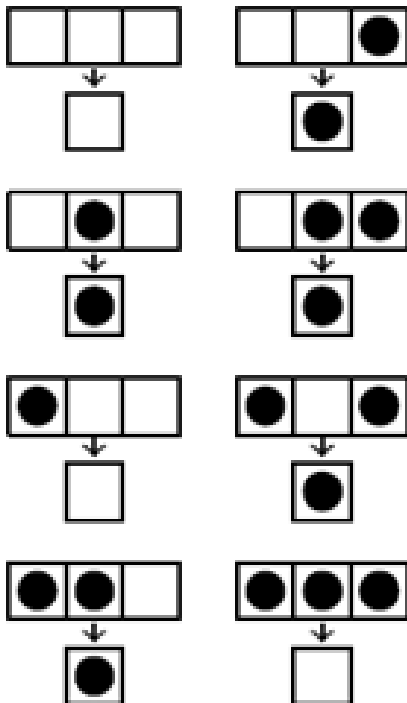
Otro ejemplo



Autómatas Celulares

Ejercicio #1

Las reglas

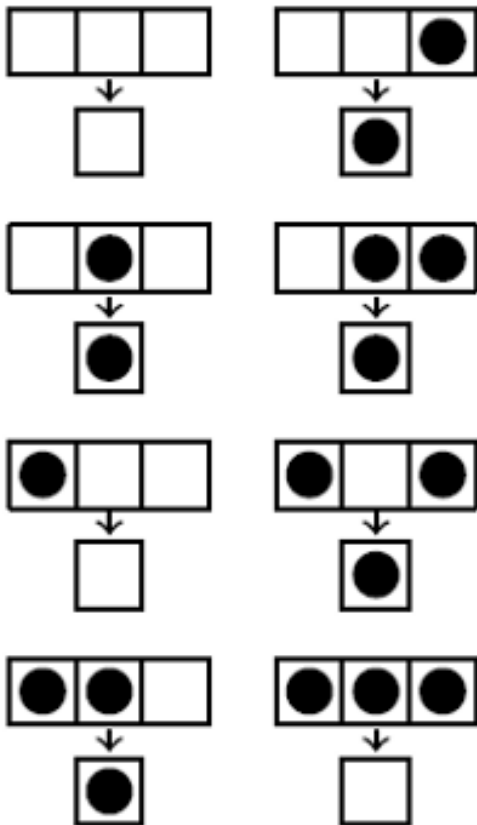


	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
IT0													
IT1													
IT2													
IT3													
IT4													
IT5													
IT6													
IT7													
IT8													
IT9													
IT10													

¿Cuál es el resultado?

Autómatas Celulares

Ejercicio #2



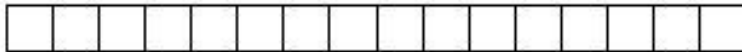
0	1	1	0	1	1	0	0	1	1

¿Cuál es el resultado?

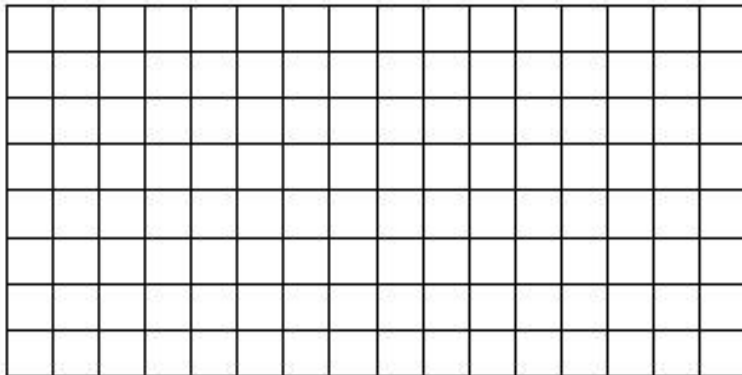
Autómatas Celulares

Evoluciones

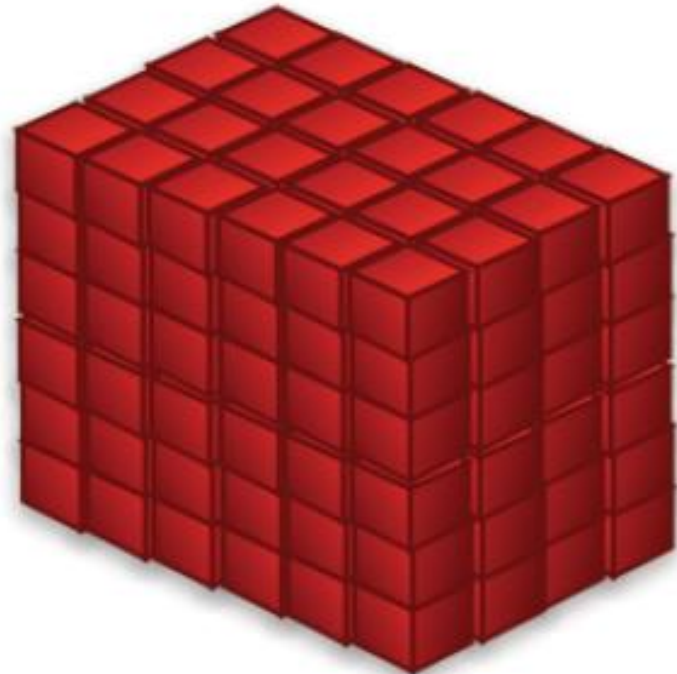
1D



2D



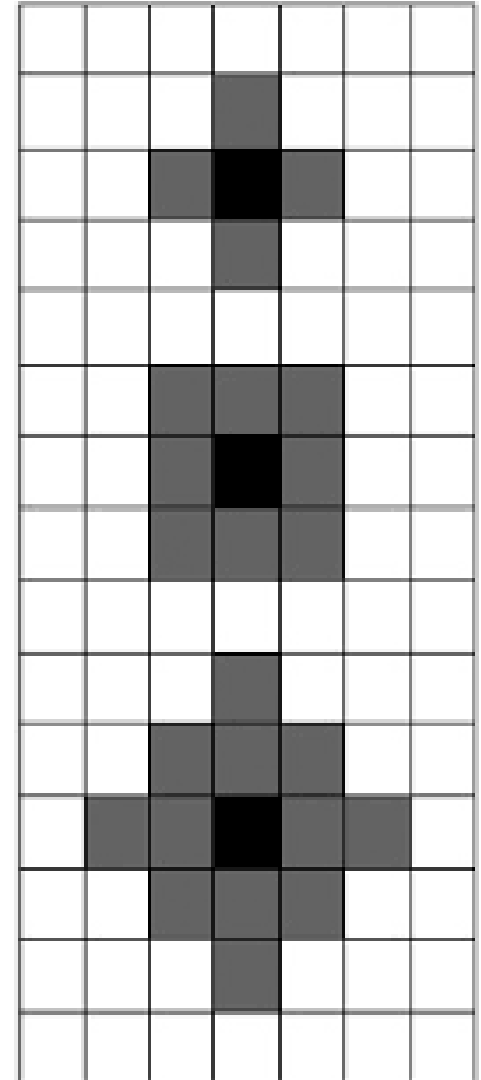
3D



Autómatas Celulares

Configuración inicial

- Un espacio n-dimensional (usualmente 2D) dividido en un número de sub-espacios conocidos como **células**, denominada **mall**.
- Cada célula puede estar en un **estado**, perteneciente a un **conjunto finito** (o numerable) o de estados.
- Una **configuración inicial**, consistente en la distribución de estados de cada celda del autómata en T_0 .
- Un **criterio de vecindad** determinado por las posiciones relativas de las células consideradas vecinas a una célula dada.



Autómatas Celulares

Esquema – Regla 30

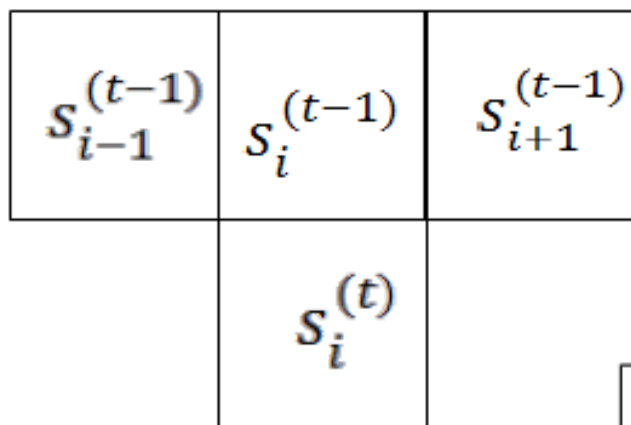


Figura 2. Regla de evolución del autómata

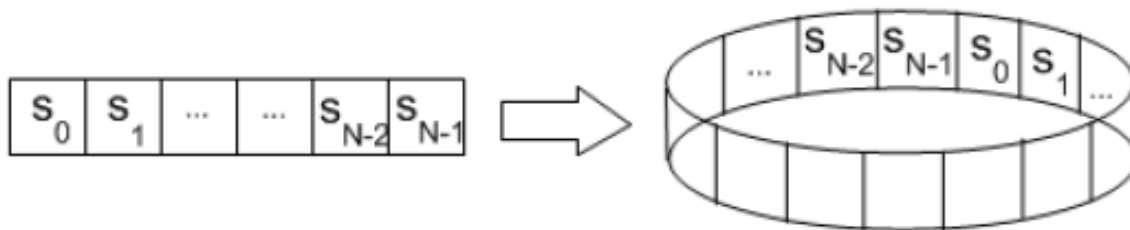


Figura 1: Esquema

Figura 3: Condiciones periódicas en la frontera

Condición inicial:

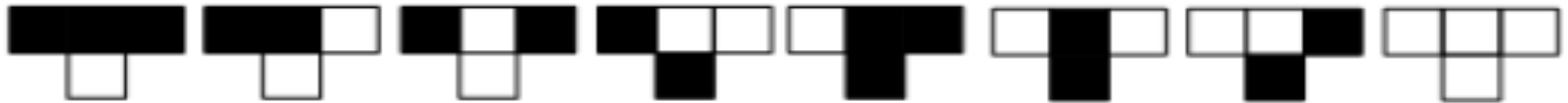
$C^0 = \{1, 1, 1, 0\}$

Autómata con 4 células

Etapa	$s_0^{(t)}$	$s_1^{(t)}$	$s_2^{(t)}$	$s_3^{(t)}$
0	1	1	1	0
1	1	0	0	0
2	1	1	0	1
...				

Autómatas Celulares

Ejercicio #3 – regla 30

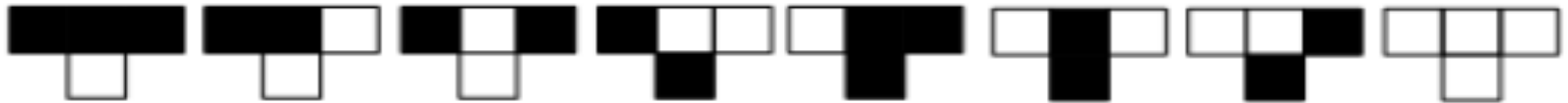


1	1	1	0	0	1

¿Cuál es el resultado al realizar cinco iteraciones?

Autómatas Celulares

Ejercicio #4 – regla 40

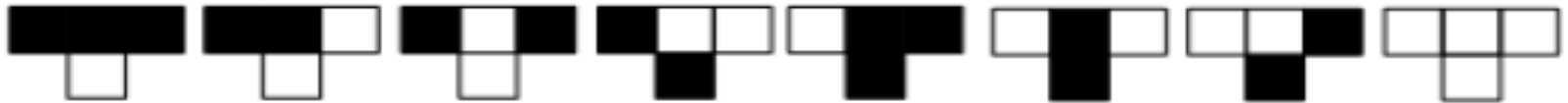


0	0	1	1	0	1

¿Cuál es el resultado al realizar cinco iteraciones?

Autómatas Celulares

Ejercicio #5 – regla 199



1	1	0	0	1	0

¿Cuál es el resultado al realizar cinco iteraciones?

Autómatas Celulares

Análisis de las reglas

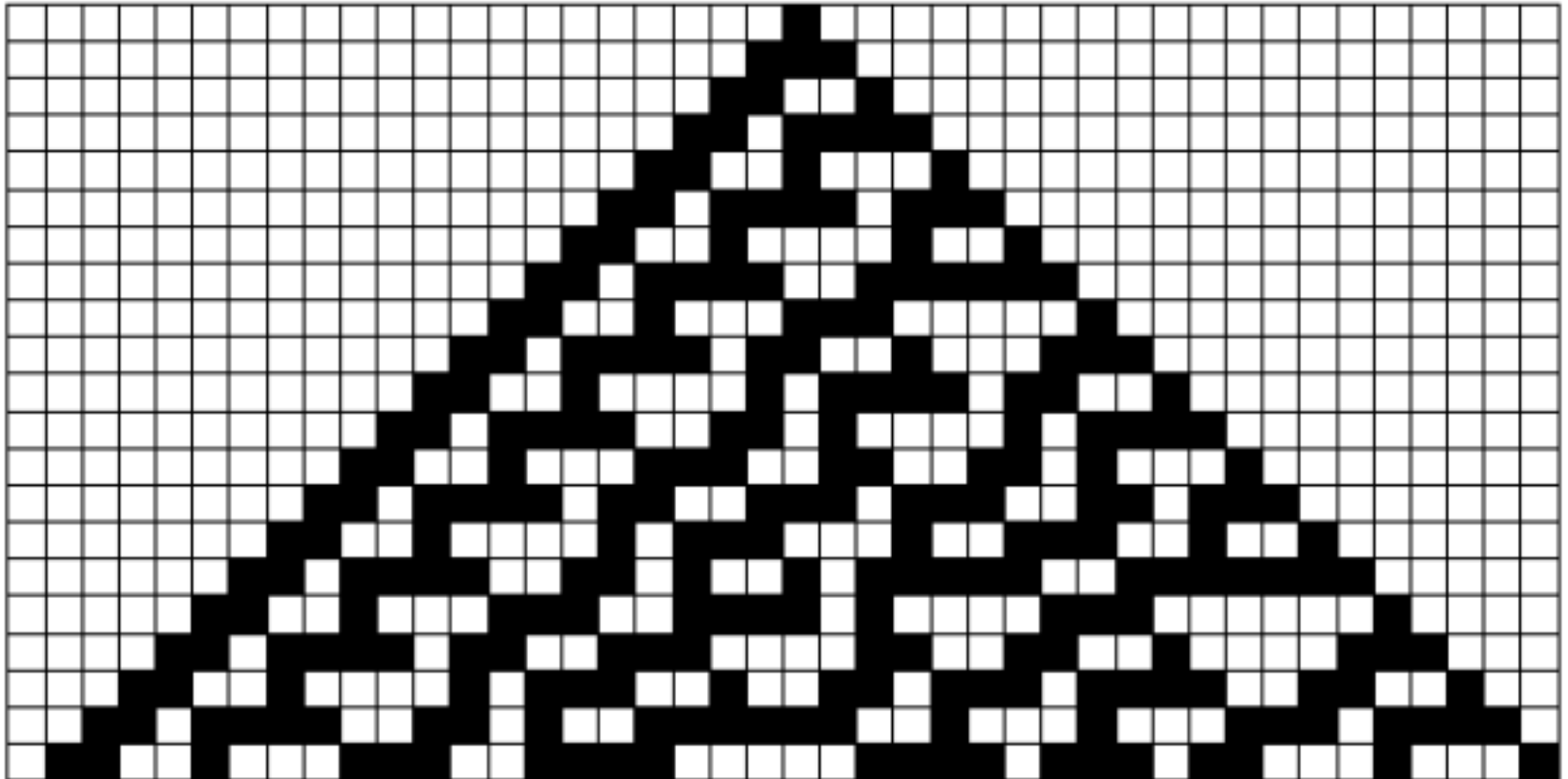
- ❑ En la regla anterior se tienen 256 posibles combinaciones dado de que hay 8 estados posibles de las tres células.
- ❑ En el ejemplo anterior la **regla de evolución es la regla 30.**



- ❑ En la siguiente gráfica observamos la evolución de un autómata con la regla 30 y 43 células. En la configuración inicial todas las células están a cero exceptuando la número 22 que posee un “1”. El número de etapas total es de 21.

Autómatas Celulares

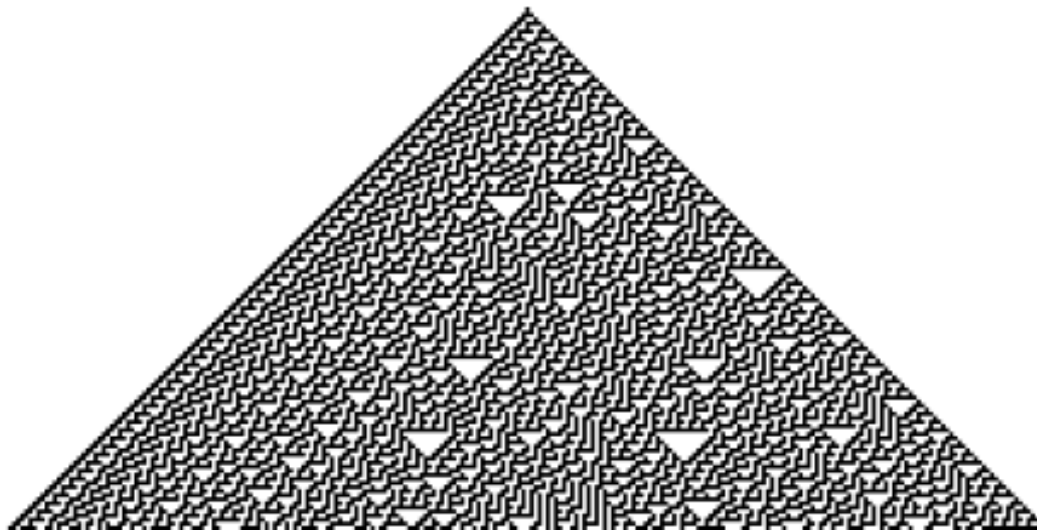
Análisis de las reglas



Autómatas Celulares

Análisis de las reglas

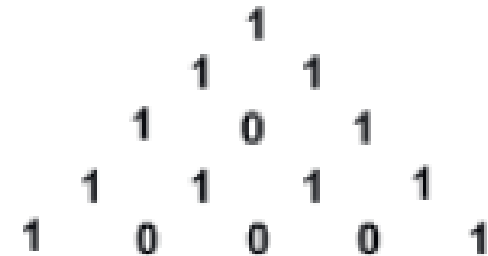
- ❑ Evolución de un autómatata celular con 257 células. Mostramos las primeras 128 etapas, aplicando la regla 30. Se sigue el mismo estado inicial del ejemplo anterior:



- ❑ Compare la caparazón de un caracol marino y un fragmento de gráfico obtenido con la regla 30.

Triángulo de Pascal – Módulo 2

- Los números impares se colocan a “1” (negro) y los pares se colocan en “0” (blanco).

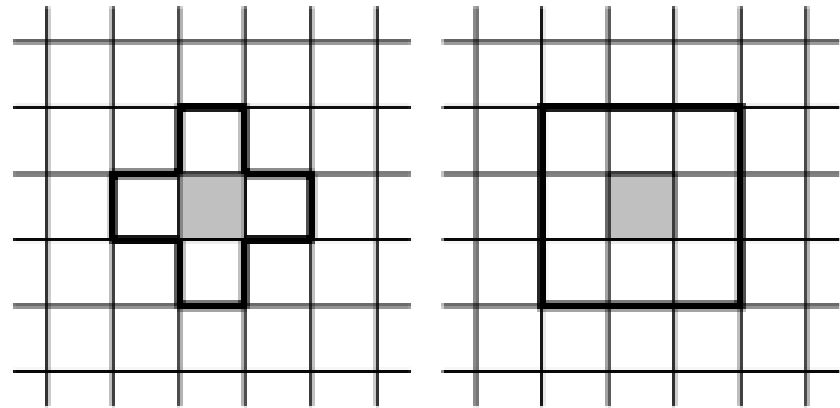


- Para ello, el estado de la célula “ i ” en la etapa $t + 1$ consiste en sumar (módulo 2) los estados de las células $i - 1$ e $i + 1$ en la etapa anterior, es decir en la etapa t .

$$s_i^t = (s_{i-1}^{t-1} + s_{i+1}^{t-1}) \bmod 2$$

Autómatas Celulares – Un resumen

- ❑ Se trata de un *modelo matemático* para un *sistema dinámico* que evoluciona en pasos *discretos*.
- ❑ Cada celda de la cuadrícula se conoce como *célula*. Cada célula se caracteriza por su vecindad y puede tomar un valor a partir de un *conjunto finito de estados*.
- ❑ Para el caso de Autómatas Celulares 2D hay dos tipos de vecindades. La Vecindad de *Von Neumann* y la Vecindad de *Moore*.



Von Neumann

Moore



Cifrado de Mensajes

❑ Utilizando el siguiente espacio de caracteres:

#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

❑ Un número entre 0 y 31 corresponde a un binario con 5 bits, siendo el 0 = {00000} y el 31 = {11111}.

❑ Si por ejemplo queremos cifrar la palabra “**SOPA**”, en binario tendríamos el binario con bits cada uno: **M** = {**10100,10000,10001,00001**}. Es decir, se trata de una secuencia de 20 bits.

❑ Ahora tomamos como clave secreta **CAMINO** que equivale a {3,1,13,9,14,16}. O sea: {**00011,00001,01101,01001,01110,10000**}. Esta será la configuración inicial.

Cifrado de Mensajes

❑ Como configuración inicial tenemos entonces:

$$C^{(0)} = \{s_0^{(0)}, s_1^{(0)}, s_2^{(0)}, \dots, s_{N-1}^{(0)}\} = \{000110000101101010010111010000\}$$

❑ **Al aplicar la regla 30** a un autómata que tiene la configuración inicial anterior, cuando se haya terminado el proceso de cierto número de iteraciones se podría elegir una célula determinada y la secuencia de bits:

$$s_i^{(t)}, s_i^{(t+1)}, s_i^{(t+2)}, \dots$$

❑ Por ejemplo, si se selecciona la célula 15 (**$i = 15$**) y la etapa 25 (**$t = 25$**) con el mensaje a cifrar de 20 bits se obtendría la siguiente secuencia:

$$K = \{s_{15}^{(25)}, s_{15}^{(26)}, \dots, s_{15}^{(44)}\} = \{10010011101010000100\}$$

Cifrado de Mensajes

- ❑ Hasta ahora se ha logrado conseguir una lista “**K**” de 20 bits a partir de una clave secreta que era **CAMINO**. Recuerde que eran 30 células numeradas desde la **0** a la **29**.
- ❑ Utilizando el cifrado de **Vernam** (se trata de un método sencillo que utiliza la operación **XOR**) tendremos las siguientes secuencias:

- ❑ $M = \{10100, 10000, 10001, 00001\}$
- ❑ $K = \{10010, 01110, 10100, 00100\}$
- ❑ $C = \{00110, 11110, 00101, 00101\}$

+	0	1
0	0	1
1	1	0

- ❑ El cifrado corresponde a: {6, 30, 5, 5}
- ❑ Finalmente: **F3EE**

En resumen:

Para cifrar: $C = M \oplus K$

Para descifrar: $M = C \oplus K$

Cifrado de Imágenes

- ❑ El proceso es diferente: Peso, configuración, esquema de umbrales, manejo de pixeles por autómata celular.
- ❑ En este caso, el estado de una célula en la etapa “**t**” depende del estado del autómata en las dos etapas anteriores (**t – 1 y t – 2**).
- ❑ Para actualizar en cada nueva etapa el estado de la célula, se utiliza la siguiente fórmula:

$$S_i^t = \left(\sum_{p=-1}^1 \lambda_p S_{i+p}^{(t-1)} + S_i^{(t-2)} \right) \bmod 256$$

El valor del parámetro
puede ser 1 o 0

λ_p

Cifrado de Imágenes

- ❑ Será necesario conocer la configuración inicial del AC en las dos primeras etapas. $C^{(0)}$ y $C^{(1)}$.
- ❑ Si se desea cifrar una imagen a escala de grises de 256 x 256 píxeles, significa que cada fila tendrá 256 píxeles. El total de bits serían $256 \times 8 = 2.048$ bits.
- ❑ Los 2.048 bits de la primera fila serán el estado inicial **C(0)** y los 2.048 bits de la segunda fila será el estado **C(1)**.
- ❑ Al realizar una serie de iteraciones cada etapa es un vector con 2.048 bits cada uno. Se pasa a decimal cada grupo de 8 bits y se obtendrán dos vectores (si se toman cada dos filas) de 256 componentes que serán las dos primeras filas de la imagen cifrada.

Cifrado de Imágenes

- ☐ El proceso se realiza para las siguientes filas de la imagen original tomando cada dos filas.
- ☐ Este mismo proceso se puede realizar para una imagen a color.

Cifrado de Imágenes

Imagen Original							Imagen Cifrada					
165	140	18	161	63	71	Cada celda corresponde a un pixel de 8 bits	240					
138	231	153	28	143	144	Dimensión: 13 filas x 6 columnas						
213	72	199	171	248	23	Total de pixeles : 78						
9	11	144	38	80	121	Cada fila tiene 6 pixeles, 48 bits.						
153	59	36	243	13	144							
139	177	232	2	32	99	$(-1 * 167 + 0 * 98 + 1 * 230) + 177$						
171	250	82	75	190	25	$(-167 + 0 + 230 + 177) \bmod 256 = 240$						
138	224	14	58	159	247							
75	69	104	122	207	95	$(-1 * 143 + 0 * 144 + 1 * 138) + 71$						
53	69	152	92	220	82	$(-143 + 0 + 138 + 71) \bmod 256 =$						
246	89	183	241	66	114							
177	225	168	217	74	73							
98	230	41	190	200	167							

$$S_i^t = \left(\sum_{p=-1}^1 \lambda_p S_{i+p}^{(t-1)} + S_i^{(t-2)} \right) \bmod 256$$

Cifrado de Imágenes

	Imagen Cifrada					Imagen Cifrada				
Cada celda corresponde a un pixel de 8 bits	240					252	155			66
Dimensión: 13 filas x 6 columnas										
Total de pixeles : 78										
Cada fila tiene 6 pixeles, 48 bits.										
$(-1 * 167 + 0 * 98 + 1 * 230) + 177$										
$(-167 + 0 + 230 + 177) \bmod 256 = 240$										
$(-1 * 143 + 0 * 144 + 1 * 138) + 71$										
$(-143 + 0 + 138 + 71) \bmod 256 =$										

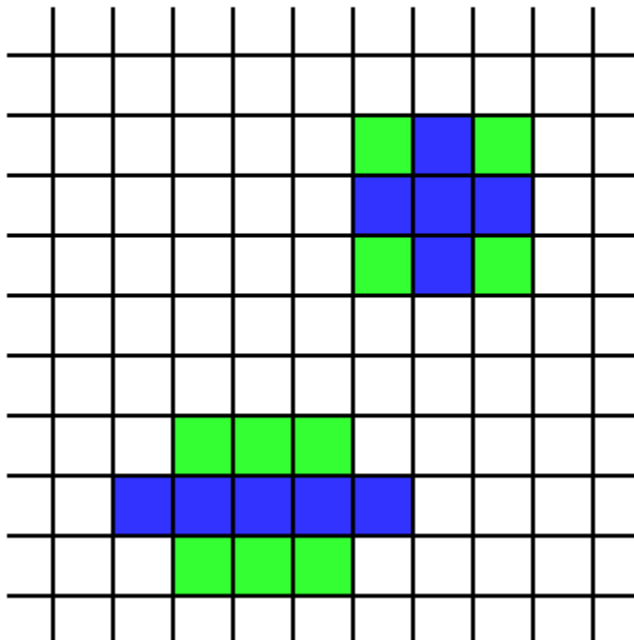
$$S_i^t = \left(\sum_{p=-1}^1 \lambda_p S_{i+p}^{(t-1)} + S_i^{(t-2)} \right) \bmod 256$$



Autómatas Celulares

El Juego de la Vida

Reglas



Cada casilla del tablero
tiene 8 casillas vecinas

- ▶ Una célula con menos de dos vecinos se muere de aburrimiento
- ▶ Una célula con dos o tres vecinos sobrevive
- ▶ Una célula con más de tres vecinos se muere ahogada
- ▶ En un lugar vacío que tiene *tres* células vecinas nace una célula nueva

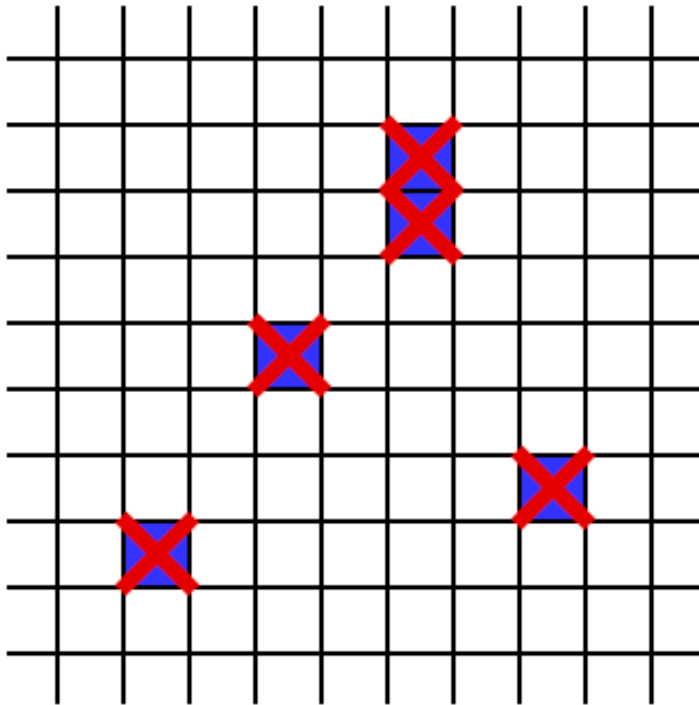
Autómatas Celulares

El Juego de la Vida

Reglas

Cada casilla del tablero
tiene 8 casillas vecinas

- Una célula con menos de dos vecinos se muere de aburrimiento

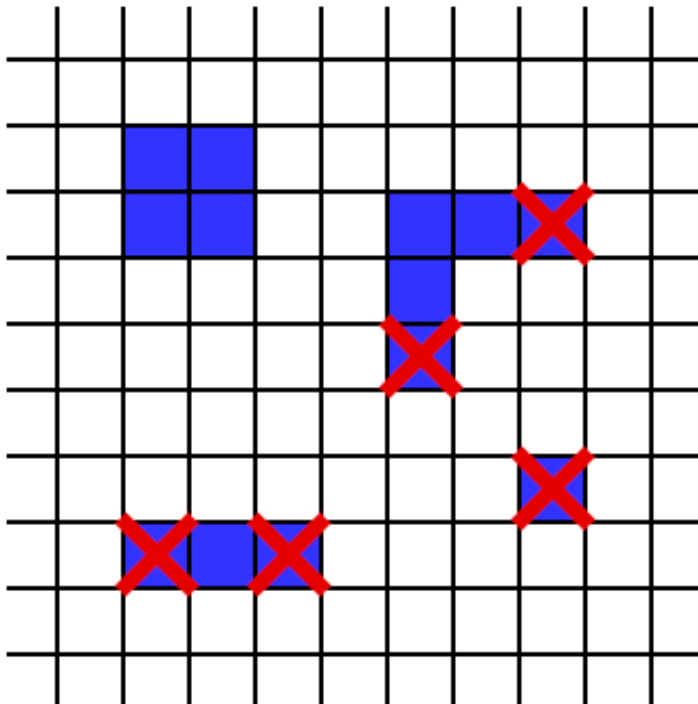


Autómatas Celulares

El Juego de la Vida

Reglas

Cada casilla del tablero
tiene 8 casillas vecinas

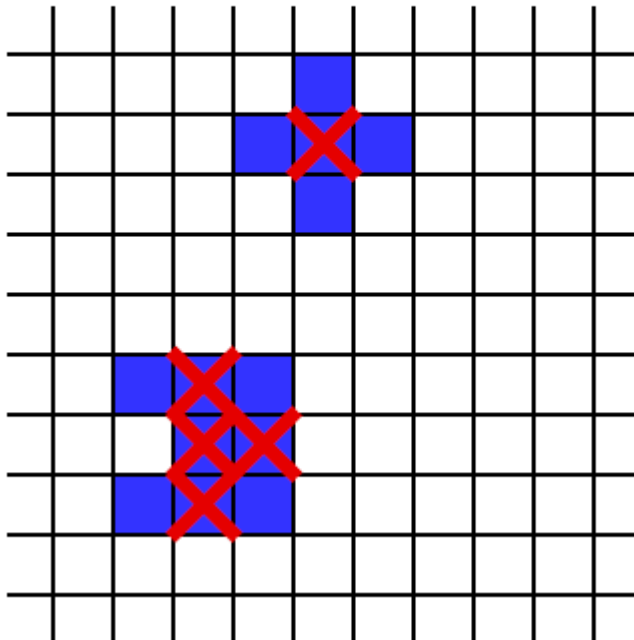


- ▶ Una célula con menos de dos vecinos se muere de aburrimiento
- ▶ Una célula con dos o tres vecinos sobrevive

Autómatas Celulares

El Juego de la Vida

Reglas



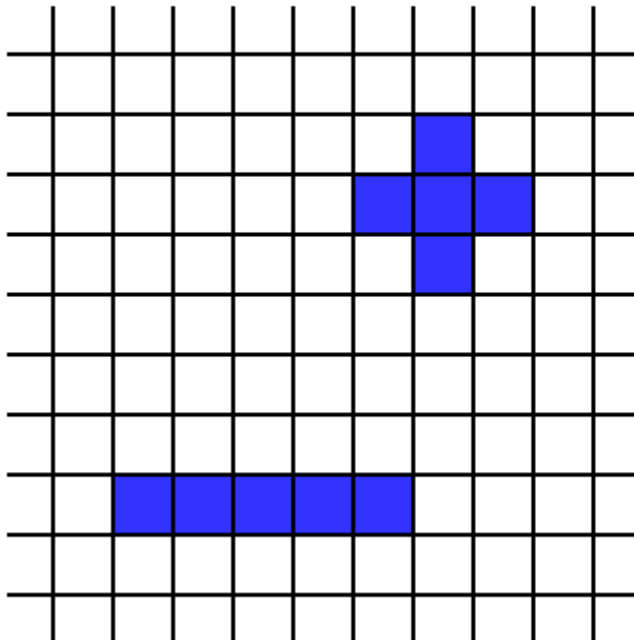
Cada casilla del tablero
tiene 8 casillas vecinas

- ▶ Una célula con menos de dos vecinos se muere de aburrimiento
- ▶ Una célula con dos o tres vecinos sobrevive
- ▶ Una célula con más de tres vecinos se muere ahogada

Autómatas Celulares

El Juego de la Vida

Reglas



Cada casilla del tablero
tiene 8 casillas vecinas

- ▶ Una célula con menos de dos vecinos se muere de aburrimiento
- ▶ Una célula con dos o tres vecinos sobrevive
- ▶ Una célula con más de tres vecinos se muere ahogada
- ▶ En un lugar vacío que tiene *tres* células vecinas nace una célula nueva