



PROGRAMACIÓN II - UNIDAD 5

Ing. Gastón Weingand (gaston.weingand@uai.edu.ar)

Agenda

1. Descripción y ejemplos de algunas de las librerías más utilizadas: Pygame, RE, Collections, NumPy, SQLAlchemy, Request, Pillow, Matplotlib
2. Introducción al machine learning.
 - A. Introducción al Deep learning.
 - i. Algoritmos de redes neuronales.
 - ii. Práctica de generación de red neuronal sobre placa microbit y/o simulación.
 - B. Framework Anaconda y miniconda

Agenda

1. Descripción y ejemplos de algunas de las librerías más utilizadas: Pygame, RE, Collections, NumPy, SQLAlchemy, Request, Pillow, Matplotlib
2. Introducción al machine learning.
 - A. Introducción al Deep learning.
 - i. Algoritmos de redes neuronales.
 - ii. Práctica de generación de red neuronal sobre placa microbit y/o simulación.
 - B. Framework Anaconda y miniconda, descripción

1.Librería: Pygame

Pygame es un conjunto de módulos del lenguaje Python que permiten la creación de videojuegos en dos dimensiones de una manera sencilla. Está orientado al manejo de sprites.

Gracias al lenguaje, se puede prototipar y desarrollar rápidamente. Esto se puede comprobar en las competiciones que se disputan online, donde es cada vez más usado. Los resultados pueden llegar a ser profesionales.

1.Librería: Pygame

También puede utilizarse para crear otros programas multimedia o interfaces gráficas de usuario. Pygame está basado en la librería SDL 1.2, una alternativa más actual de SDL en Python podría ser Py-SDL2, que implementa varias mejoras respecto a Pygame.

1.Librería: Pygame

```
# Módulos
import pygame, sys
from pygame.locals import * #Para tener QUIT

# Constantes
WIDTH = 640
HEIGHT = 480

# Clases
# -----
# -----

# Funciones
# -----
# -----
```

1.Librería: Pygame

```
def main():
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Test Pygame")

    while True:
        for eventos in pygame.event.get():
            if eventos.type == QUIT:
                sys.exit(0)
        return 0

if __name__ == '__main__':
    pygame.init()
    main()
```

1.Librería: Re



**Módulo que se utiliza para el análisis de
Expresiones regulares.**

Documentación y ejemplo en la unidad nro. 4

1.Librería: Collections

Con la ayuda de este módulo se puede aumentar las estructuras de datos típicas disponibles en Python (Listas, tuplas, diccionarios,...). Veamos algunas utilidades disponibles:

```
#Collections

from collections import ChainMap

dict_a = {'a': 1, 'b': 10}
dict_b = {'b': 100, 'c': 1000}

cm = ChainMap(dict_a, dict_b)
for key, value in cm.items():
    print(key, value)

print(cm.maps)
```

Hemos añadido el valor de la clave c de dict_b sin necesidad de modificar nuestro diccionario original de configuración dict_a, es decir, hemos hecho un 'cambio' reversible

Los diccionarios originales están disponibles haciendo uso del atributo maps

1.Librería: Collections

Counter

Permite contar ocurrencias de forma simple.

```
#Counter

from io import StringIO
from collections import Counter

virtual_file = StringIO("""2010/01/01 2.7
2010/01/02 2.2
2010/01/03 2.1
2010/01/04 2.3
2010/01/05 2.4
2010/01/06 2.2
2010/01/02 2.2
2010/01/03 2.1
2010/01/04 2.3
""")

print(Counter(virtual_file.readlines()).most_common(1))
```

1.Librería: Collections

namedtuple

A veces se necesita crear algún tipo de estructura que guarda datos y algunos metadatos. Una forma simple sin crear una clase ad-hoc sería usar un diccionario:

```
#namedtuple

import numpy as np
import datetime as dt
from pprint import pprint

datos = {
    'valores': np.random.randn(100),
    'frecuencia': dt.timedelta(minutes = 10),
    'fecha_inicial': dt.datetime(2016, 1, 1),
    'parametro': 'wind_speed',
    'unidades': 'm/s'
}

pprint(datos)
```

1.Librería: Collections

namedtuple

Lo anterior es simple y rápido pero usando una namedtuple dispongo de algo parecido con algunas cosas extra:

```
from collections import namedtuple

Datos = namedtuple('Datos', 'valores frecuencia fecha_inicial parametro unidades')

datos = Datos(np.random.randn(100),
              dt.timedelta(minutes = 10),
              dt.datetime(2016, 1, 1),
              'wind_speed',
              'm/s')

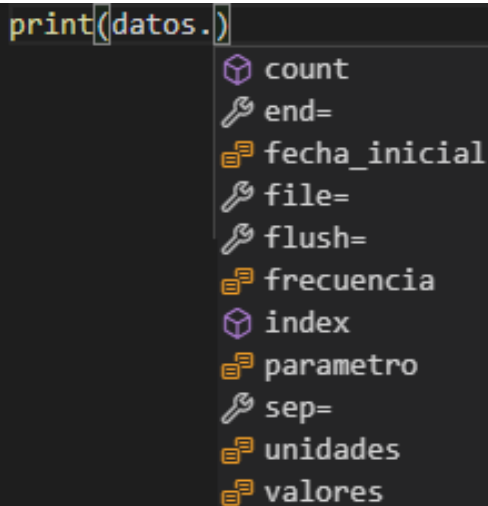
print(datos)
```

1.Librería: Collections

namedtuple

Ahora puedo acceder a los 'campos' o claves del diccionario usando **dot notation**

```
print(datos.)
```



The screenshot shows a code editor with a dark background. The first line of code is `print(datos.)`. Below the code, a list of attributes is displayed, each preceded by a small icon: a purple cube for `count` and `index`, a key icon for `end=`, `file=`, and `flush=`, and a yellow notepad icon for `fecha_inicial`, `frecuencia`, `parametro`, `unidades`, and `valores`, and `sep=`.

- count
- end=
- fecha_inicial
- file=
- flush=
- frecuencia
- index
- parametro
- sep=
- unidades
- valores

1.Librería: Collections

namedtuple

También Puedo extender desde Datos

```
class DatosExtendidos(Datos):  
    def media(self):  
        "Calcula la media de los valores."  
        return self.valores.mean()  
  
datos_ext = DatosExtendidos(**datos._asdict())  
#** Desempaqueta un diccionario como parámetros de una función  
  
print(datos_ext.media())
```

1.Librería: Collections

deque

Es una secuencia mutable (parecido a una lista), pero con una serie de ventajas. Es una cola/lista cuyo principio y fin es 'indistinguible', es *thread-safe* y está diseñada para poder insertar y eliminar de forma rápida en ambos extremos de la cola

1.Librería: Collections

deque

```
#deque
from collections import deque

dq = deque(range(10), maxlen = 10)
lst = list(range(10))
print(dq)
print(lst)

# los tres últimos elementos los anexa nuevamente al principio de la secuencia.
dq.rotate(3)
print(dq)
lst = lst[-3:] + lst[:-3] #Podría hacer lo mismo con listas y slicing
print(lst)
```


1.Librería: Collections

deque

Se pueden agregar elementos por derecha o izquierda.

De a uno o varios

```
dq.append(100)
print(dq)
dq.appendleft(10000)
print(dq)

dq.extend(range(10))
print(dq)
dq.extendleft([10, 100])
print(dq)
```

1.Librería: NumPy y matplotlib

NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices

Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, **pylab**, diseñada para recordar a la de MATLAB.

1.Librería: NumPy y matplotlib

Figure 1

```
#numPy
```

```
import numpy
```

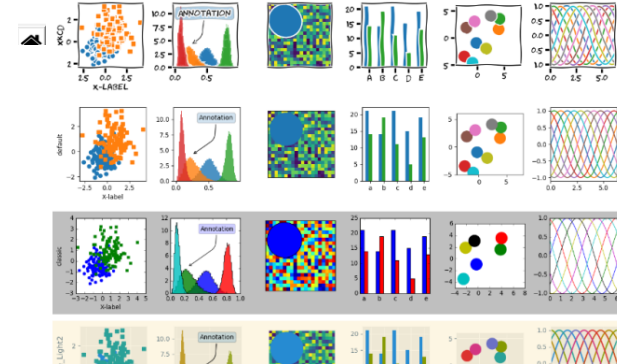
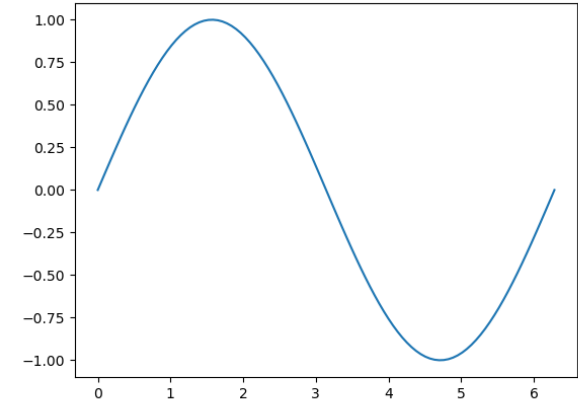
```
from matplotlib import pyplot
```

```
x = numpy.linspace(0, 2 * numpy.pi, 100)
```

```
y = numpy.sin(x)
```

```
pyplot.plot(x, y)
```

```
pyplot.show()
```



Matplotlib permite generar diversos gráficos2D y 3D

1.Librería: SQLAlchemy & pyodbc

Genero la conexión a la base de datos

```
from sqlalchemy.engine import create_engine
from sqlalchemy.sql import select
from sqlalchemy import Table, Column, Integer, String, MetaData

import urllib
params = urllib.parse.quote_plus("DRIVER={SQL Server Native Client 11.0};"
                                "SERVER=.\\SQLEXPRESS;"
                                "DATABASE=Security;"
                                "UID=test;"
                                "PWD=qwerty")

engine = create_engine("mssql+pyodbc:///odbc_connect={}".format(params))

conn = engine.connect()
```

1.Librería: SQLAlchemy & pyodbc

Genero la entidad tipo tabla

```
metadata = MetaData()
Usuario = Table('Usuario', metadata,
                Column('idUserio', String, primary_key=True),
                Column('Nombre', String))
```

1.Librería: SQLAlchemy & pyodbc

Ejecuto las acciones sobre la DB

```
s = select([Usuario])  
  
result = conn.execute(s)  
  
for row in result:  
    print(row)
```

1.Librería: Requests

Requests permite enviar solicitudes HTTP con extrema facilidad. Trabaja básicamente con json.

```
#Requests

import requests
r = requests.get('https://api.github.com/repos/psf/requests')
print(r.json()["description"])

A simple, yet elegant HTTP library.
```

1.Librería: Requests - Funcionalidades

- URLs y Dominios internacionales
- Keep-Alive* y Agrupamiento de conexiones (*Connection Pooling*)
- Sesiones con Cookies persistentes
- Verificación SSL al estilo navegador
- Autenticación Básica y Digest
- Elegantes Cookies en pares Llave/Valor
- Descompresión automática
- Cuerpos de respuestas Unicode
- Subida de archivos Multiparte
- Tiempos de espera de conexión
- Soporte para *.netrc*
- Seguridad para programación en hilos (*Thread-safety*)

1.Librería: Pillow



Python Imaging Library es una biblioteca gratuita para el lenguaje de programación Python que agrega soporte para abrir, manipular y guardar muchos formatos de archivo de imagen diferentes.

1.Librería: Pillow

```
#PIL

from PIL import Image
from PIL import ImageFont
from PIL import ImageDraw

img = Image.new('RGB', (800, 400), "white")
draw = ImageDraw.Draw(img)
font = ImageFont.truetype("LiberationSerif-Regular.ttf", 36)
draw.text((20, 20), "UAI-CURSO DE PYTHON", (0, 2, 27), font=font)
img.save('image.png')
```

Agenda

1. Descripción y ejemplos de algunas de las librerías más utilizadas: Pygame, RE, Collections, NumPy, SQLAlchemy, Request, Pillow, etc.
2. **Introducción al machine learning.**
 - A. Introducción al Deep learning.
 - i. Algoritmos de redes neuronales.
 - ii. Práctica de generación de red neuronal sobre placa microbit y/o simulación.
 - B. Framework Anaconda y miniconda

Concepto IA



La **inteligencia artificial (IA)** es la inteligencia exhibida por máquinas

En 1956, John McCarthy acuñó la expresión «inteligencia artificial», y la definió como «la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes»

Según Takeyas (2007) la IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: El razonamiento y la conducta.

Concepto IA



Sistemas que piensan como humanos

REDES NEURONALES

Sistemas que actúan como humanos

ROBÓTICA

Sistemas que piensan racionalmente

SISTEMAS EXPERTOS

Sistemas que actúan racionalmente (idealmente)

AGENTES INTELIGENTES

Concepto IA



Sentir

La visión por ordenador y el procesamiento de audio, por ejemplo, pueden percibir lo que sucede alrededor mediante la adquisición y el procesamiento de imágenes, sonidos y voz. El uso del reconocimiento facial en puestos de control de pasaportes es un ejemplo práctico de cómo la IA puede aumentar la productividad.



Comprender

El procesamiento de lenguajes naturales y los motores de inferencia permiten que los sistemas de IA puedan analizar y comprender la información recibida. Esta tecnología se utiliza para crear herramientas que traducen los resultados obtenidos por motores de búsqueda.



Actuar

Un sistema de IA puede emplear tecnologías como los sistemas expertos y los motores de inferencia para tomar decisiones o llevar a cabo acciones en el mundo físico. Buen ejemplo de ello son los pilotos automáticos y los sistemas de asistencia a la frenada en vehículos.

Nuevo factor de producción



Nuevo factor de producción

“Los avances en IA nos llevan a replantearnos relaciones económicas básicas y la forma en que se genera valor.”

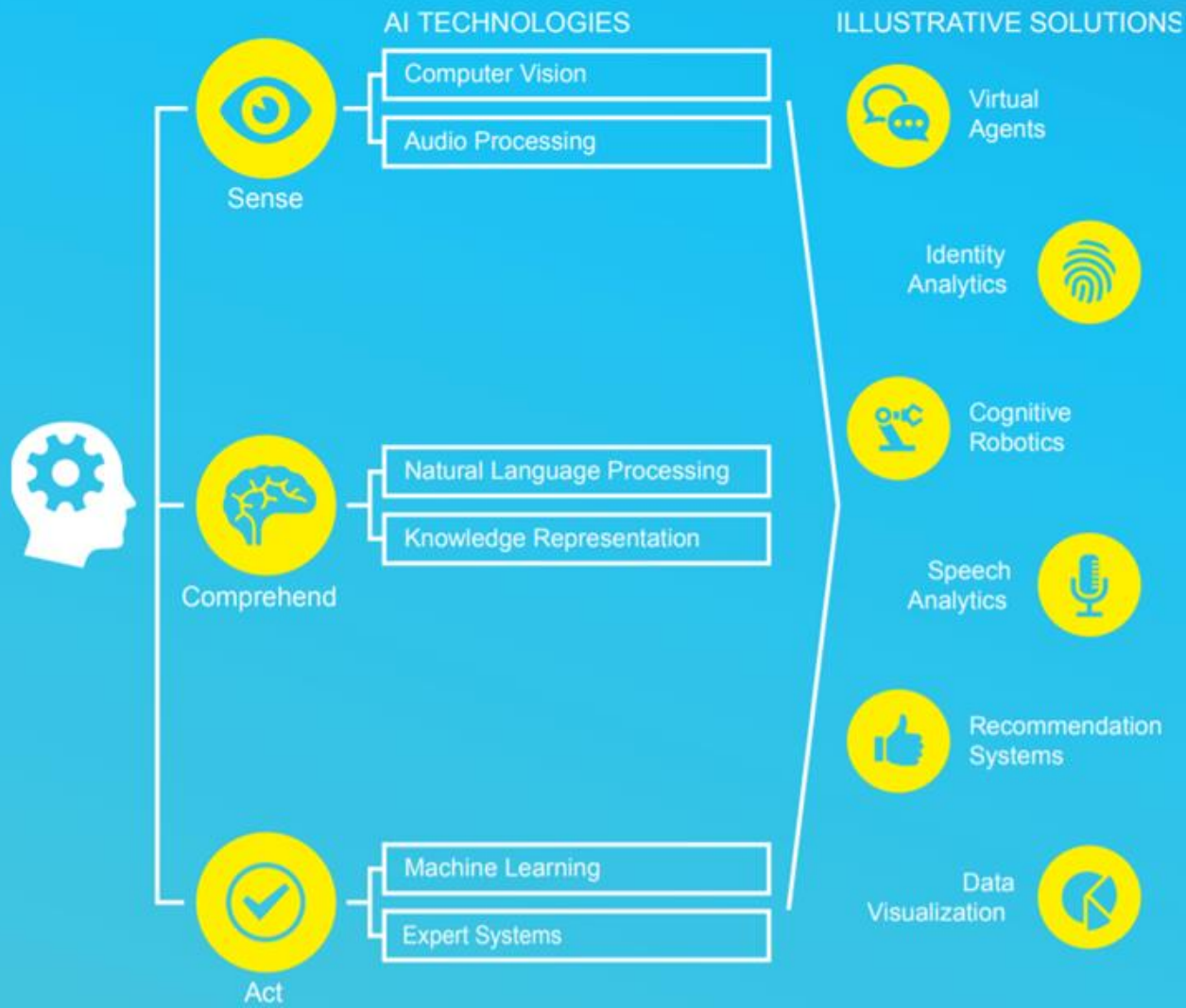
DAVID LEHRER, CEO DE CONATIX

Nuevo factor de producción

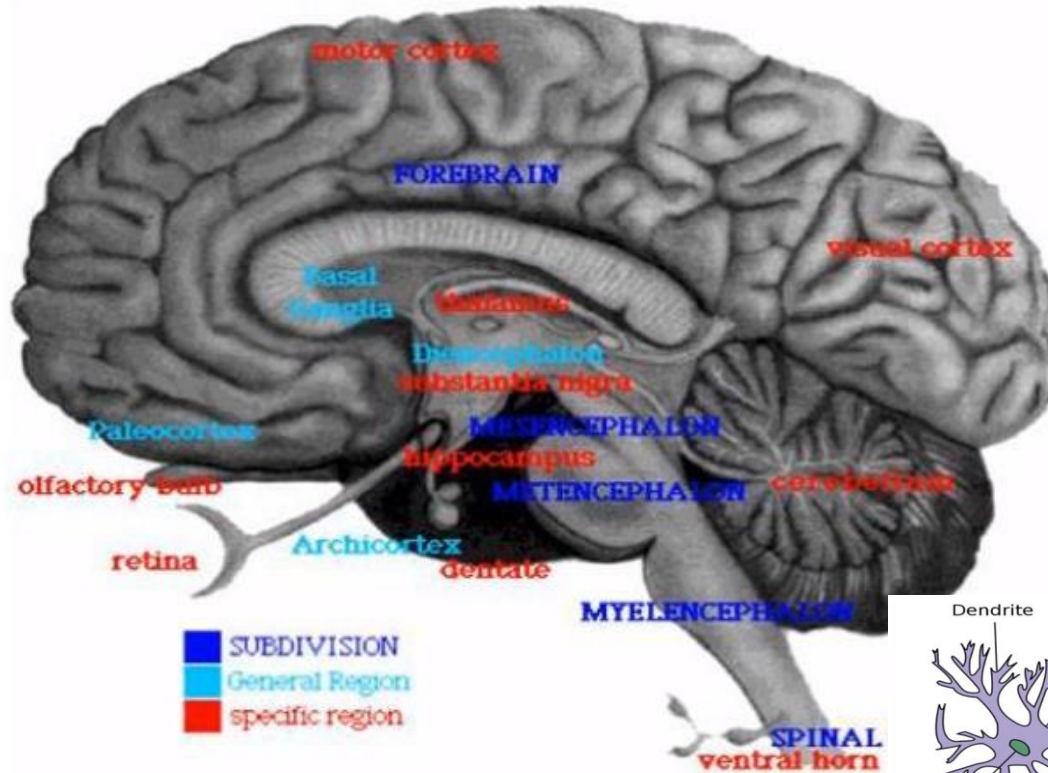
“En el futuro, la IA llegará a todos los ámbitos de la economía.”

NILS J. NILSSON, CATEDRÁTICO DE INGENIERÍA
INFORMÁTICA EN LA UNIVERSIDAD DE STANFORD

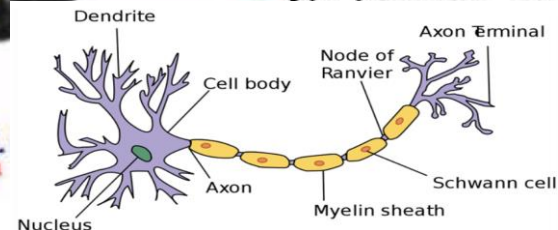
IA APPLICATIONS



Cerebro Humano



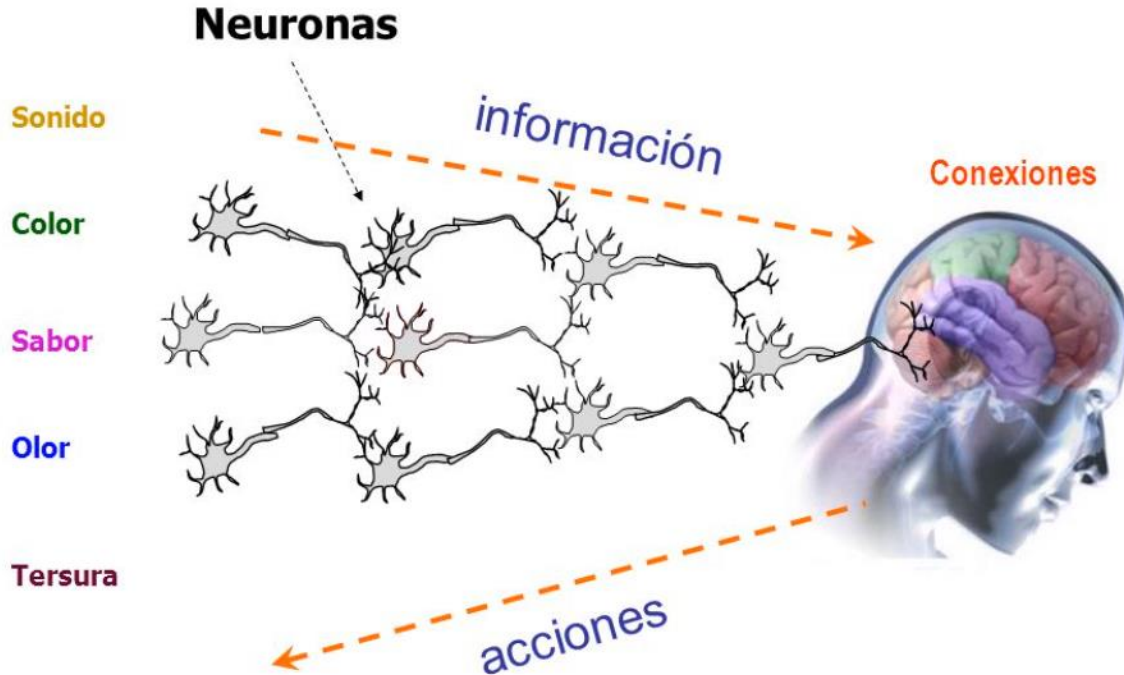
- 10^{11} Neuronas (procesadores)
- Poder desconocido
- 1000 – 10000 conexiones por neurona
- Capacidad basada en las conexiones.
- Cada neurona es muy compleja.
- Almacenamiento redundante.



stable y

EL CEREBRO

¿Cómo funciona ?



Sensores humanos

Sonido → 10^2 Hz, 10^4 Hz

Color → 10^{15} Hz

Sabor → ¿es una onda?

Olor → ¿es una reacción química?

Tersura → ¿es una onda?

EL CEREBRO

Características del Sistema Nervioso

- adaptabilidad
- aprendizaje continuo
- distribución del procesamiento y del “almacenamiento”
- alta redundancia
- plasticidad (creación/modificación de sinapsis).
- tolerante a fallas
- 10 a 100 billones de neuronas, cada una conectado a otras 10.000 neuronas
- Los humanos pierden aprox.. 1000 neuronas por día.

EL CEREBRO

Aprendizaje

Patrones de actividad.

- Ésta forma de aprendizaje es la que explica la memoria de corto plazo.

Cambios físicos y químicos en las neuronas.

- En este aprendizaje se modifican las conexiones entre unidades haciendo que grupos de neuronas se vuelvan más o menos eficientes para excitar o inhibir a otras neuronas.
- Explica la memoria de largo plazo.

Creación de nuevas conexiones.

- En este aprendizaje se crean nuevas conexiones.

EL CEREBRO

Aprendizaje

- El aprendizaje es un proceso por el cual el individuo cambia de **actitud**.
- Es un proceso unido a la experiencia.
- El proceso fundamental del aprendizaje es la **imitación**, es decir la repetición de un proceso observado.
- *El aprendizaje se define técnicamente como un cambio relativamente estable en la conducta del sujeto como resultado de la experiencia, producidos a través del establecimiento de asociaciones entre estímulos y respuestas mediante la práctica en un nivel elemental*

EL CEREBRO



Actuar como humano: La prueba de Turing

- Propuesto por Alan Turing (1950).
- Turing definió la conducta inteligente como “la capacidad de lograr eficiencia humana en todas las actividades cognoscitivas, suficiente para engañar a un evaluador”.

EL CEREBRO

La prueba de Turing

- Para que una computadora pase la prueba de Turing, debe por lo menos:
 - Procesar lenguaje natural
 - Representar el conocimiento
 - Razonar automáticamente
 - Auto aprender
- Para la prueba total de Turing, la computadora debe tener
 - Vista
 - Robótica

EL CEREBRO

- ¿Cómo se puede inhabilitar la prueba de turing?
- ¿Qué preguntas se puede hacer a una persona de tal forma que se pueda diferenciar de un computador?
- ¿Qué preguntas se puede hacer a un computador de tal forma que se pueda diferenciar de una persona?



EL CEREBRO

Racionalidad

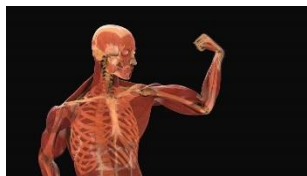
- La racionalidad se entiende referida a la acción y designa aquel tipo de acción configurada de forma que permita obtener ciertos objetivos predefinidos.
- La racionalidad consiste en la inteligente persecución de fines mediante acciones adecuadas.
- Un agente racional es aquel que saca el máximo provecho (objetivo, meta) de una dada situación (restricciones).
- Actuar racionalmente significa elegir una alternativa con la jerarquía más alta dentro de un grupo de alternativas factibles.



EL CEREBRO

Actuar racionalmente: El agente racional

- Actuar racionalmente es hacerlo de tal manera que se logran los objetivos deseados.
- Un agente es algo que percibe y actúa. De acuerdo a esto, la IA es el estudio y construcción de agentes racionales.
- Actuar racionalmente no implica sólo actuar por inferencias lógicas (ejemplo, reflejos).



EL CEREBRO

¿Inteligencia Artificial?

- ¿Puede pensar realmente un sistema inteligente artificial dentro de una computadora?
- ¿Tiene realmente libre albedrío?
- ¿Puede llegar a reemplazar al ser humano?
- ¿Hasta que punto puede un sistema ser inteligente?

