# Redes Neuronales y Aprendizaje Profundo para Visión Artificial

**Dr.** Ariel H. Curiale

**Cuatrimestre:** 2do de 2019

---

## PRÁCTICA 0: INTRODUCCIÓN A PYTHON/NUMPY/SCIPY/MATPLOTLIB

---

1. Resuelva el siguiente sistema de ecuaciones de la forma más simple que pueda

$$\begin{aligned} x + z &= -2 \\ 2x - y + z &= 1 \\ -3x + 2y - 2z &= -1 \end{aligned}$$

2. Calcular el histograma, la media ($\mu$) y la desviación estándar ($\sigma$) de:
   `data= np.random.gamma(3,2,1000)`

   Verificar que sean las correctas.

3. Definir una función para encontrar las raíces de la ecuación de segundo grado

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

   El parámetro $a$ es obligatorio, $b$ y $c$ son opcionales con un valor por defecto de 0. La función debe devolver un array con las raíces.

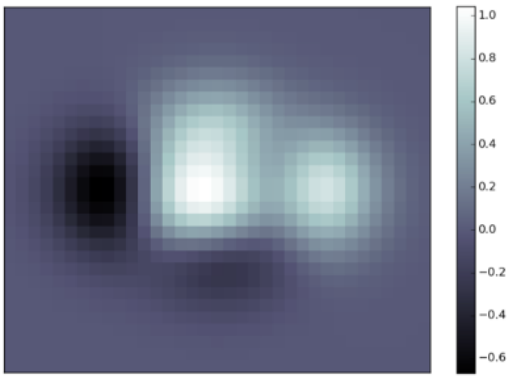
4. Crear una función que dados los coeficientes de un polinomio de segundo grado grafique la parábola ($x \in [-4.5, 4]$). Además, marcar las raíces en el gráfico, por ejemplo con puntos rojos y un texto indicando su valor con 3 decimales utilizando latex ($x_i = 1.22$).

5. Defina una clase `Lineal` que recibe parámetros a, b en su constructor, y un método que resuelve la ecuación $ax + b$. Definir el método para que pueda ser invocado como una función.

6. Herede de la clase `Lineal` una subclase `Exponencial` que resuelva la ecuación $ax^b$.

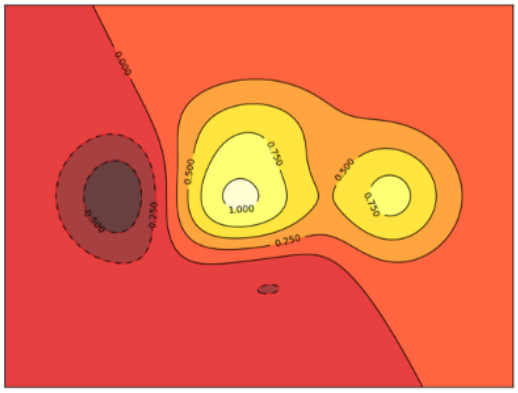
7. Crear un módulo `circunferencia.py` donde se define la constante `PI` y la función `area(r)` que calcula el área para una circunferencia de radio r. Importar el módulo con el alias `circle`. Verificar el valor de `PI` y calcular el área. Luego importar del modulo la constante `PI` y la función `area(r)` utilizando la sintaxis `from .... import ...` y verificar nuevamente la constante y calcular el área. ¿Son los mismos objeto?
   Ayuda: utilizar la función `is` que devuelve True si los objetos son el mismo (`Objeto1 is Objeto2`).

8. Crear un paquete llamado `geometria` el cual contenga al módulo `circunferencia` creado anteriormente y otro análogo que se llame `rectangulo`, también con la función `area` que permite calcular su área.
9. Generar la estructura de paquetes, módulos y archivos que necesite para que las siguientes instrucciones funcione correctamente:
   - `import p0_lib`
   - `from p0_lib import rectangulo`
   - `from p0_lib.circunferencia import PI, area`
   - `from p0_lib.elipse import area`
   - `from p0_lib.rectangulo import area as area_rect`
10. A partir de la definición de la función y el código de abajo intente obtener un gráfico como el siguiente teniendo en cuenta el colormap (bone), la interpolación de la imagen y el origen (pensar que función de numpy necesita para crear X e Y):



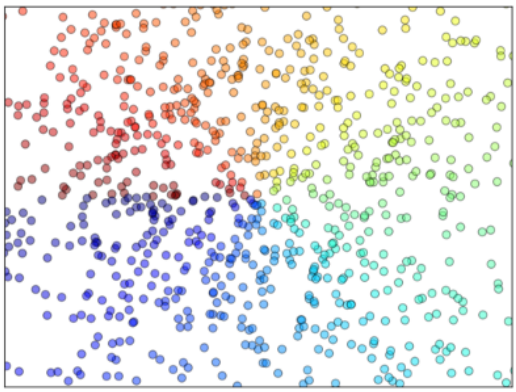

```
def f(x, y):
    return (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)

n = 10
x = np.linspace(-3, 3, 4 * n)
y = np.linspace(-3, 3, 3 * n)
X, Y = ........
plt.imshow(f(X, Y))
```

11. A partir de la definición de la función y el código anterior obtener un gráfico como el siguiente teniendo en cuenta el colormap (hot) y n=256:




Ayuda: usar los comandos `contourf, contour` y `clabel`.

12. A partir del código de abajo intente obtener un gráfico como el siguiente teniendo en cuenta el tamaño de los puntos, color y transparencia:



```
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)
```

¿Para qué sirve este gráfico? ¿Qué relaciona? Ayuda: Usar la función `scatter`.

13. Es posible simular el comportamiento de un cardumen con tres reglas que controlan la velocidad de cada pez dentro del cardumen:

    (a) Tender a acercarse al centro del cardumen $\delta v_i^1 = \frac{rc - r_i}{8}$ donde $rc$ es el centro del cardumen y $r_i$ es la posición del i-ésimo pez.

    (b) Evitar chocar con los vecinos más cercanos $\delta v_i^2 = \sum_j^{|r_j - r_i| < d}$ donde solo se tienen en cuenta los peses que están a una distancia $d$.

    (c) Tender a igular la velocidad media del cardumen $\delta_i^3 = \frac{vc - v_i}{8}$ donde $vc$ es la velocidad media del cardumen y $v_i$ es la velocidad del pez i-ésimo.

    El cambio de la velocidad del i-ésimo pez será la suma de los cambios de las tres reglas $\delta v_i = \delta v_i^1 + \delta v_i^2 + \delta v_i^3$. A su vez tenemos la restricción que la velocidad de los peces no puede superar una velocidad máxima ($\forall i, ||v_i||_2 < V$). Finalmente la evolución de la posición de cada pez se calcula como $r_i(t + \delta_t) = r_i(t) + v_i(t) \times \delta_t$. Armar una clase R2 para modelar el problema en 2D; una clase Pez que cuente con los atributos de posición y velocidad; y una

clase cardumen con los atributos privados maxVel, maxDist. La forma de utilizar la clase cardumen es:

```
Cardumen c
c.initialize(40, maxVel, maxDist)
for i in range(niter):
        c.doStep()
        c.print()
```

donde se inicializa de forma aleatoria la posición dentro del espacio 40x40 con una velocidad inicial random $< maxVel$, y `print` imprime la posicion y velocidad de cada pez.

14. El problema de cumpleaños está relacionado con cuál es la probabilidad de que en un grupo de personas haya al menos dos que cumplan años el mismo día. Se sabe que si en una misma sala si hay 23 personas reunidas, la probabilidad es del 50,7%. Para 57 o más personas la probabilidad es mayor del 99%. Si bien esto se puede deducir de manera analítica, es posible hacer una comprobación empírica. Para eso, considere 1000 grupos de un tamaño fijo de personas, tomando sus fechas de nacimiento al azar y analice en cuántos casos al menos dos personas cumplen el mismo día. Los tamaños de los grupos deben tomarse entre 10 y 60, aumentando de a 10. Hacer una tabla que indique la probabilidad en función del tamaño de grupo. Válgase de funciones para descomponer el problema en sub-problemas.

15. Se desea implementar un functor Noiser, cuando es activado con un double como argumento retorne un valor que resulte de sumar al argumento con una componente pseudoaleatoria. La componente pseudo-aleatoria estará limitada entre 2 umbrales minV y maxV que serán indicados en la construcción del objeto functor. Implementar la clase Noiser y verifique su funcionamiento utilizando la clase `np.vectorize` para aplicar el functor Noiser a cada elemento del array.