TP 0

Tomás Hüttebräucker - Aprendizaje Profundo y Redes Neuronales Artificiales

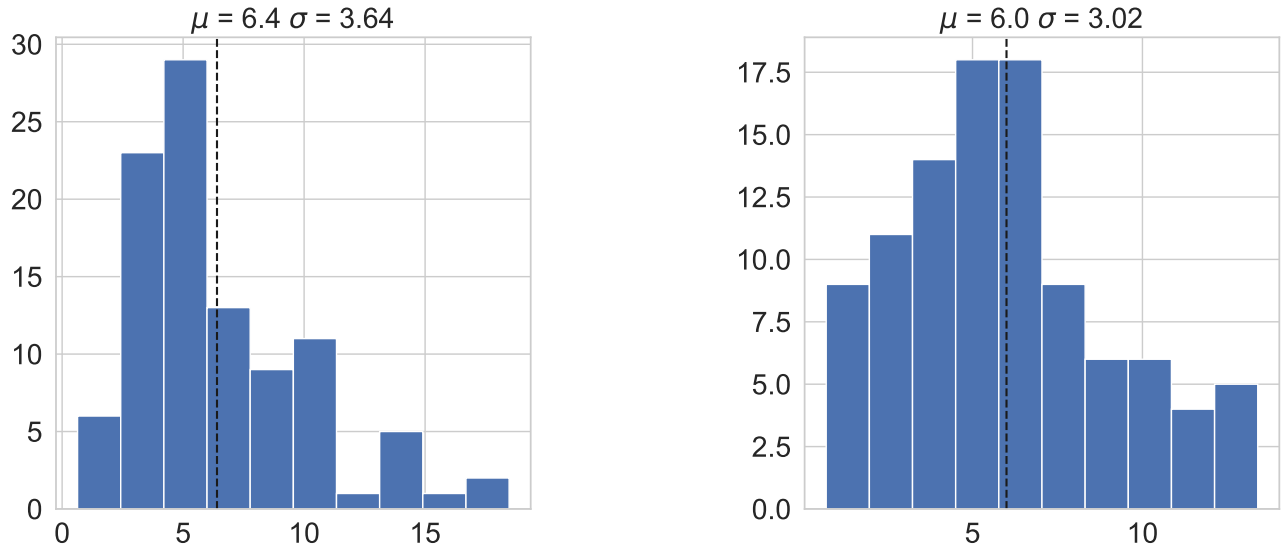
21 de agosto de 2020

Ejercicio 1

Utilizando la función `solve` del paquete `linalg` de la librería `numpy` se resuelve el problema y se obtiene como resultado x =1, y=-2 y z=-3

Ejercicio 2

La función `np.random.gamma(a,b,c)` genera c muestras de una distribución gamma con factor de forma a y factor de escala b. La media de una distribución gamma con factor de forma a y factor de escala b es ab y la varianza ab^2. Al crear muestras usando distribución `np.random.gamma(3,2,100)`, la media de las muestras debería aproximarse a ab =6 y la desviación estándar a $\sqrt{ab^2} = \sqrt{12} \approx 3{,}46$. Al tomar una cantidad finita de muestras la media y la desviación estándar de las mimas no necesariamente tomará esos valores, pero los aproxima.



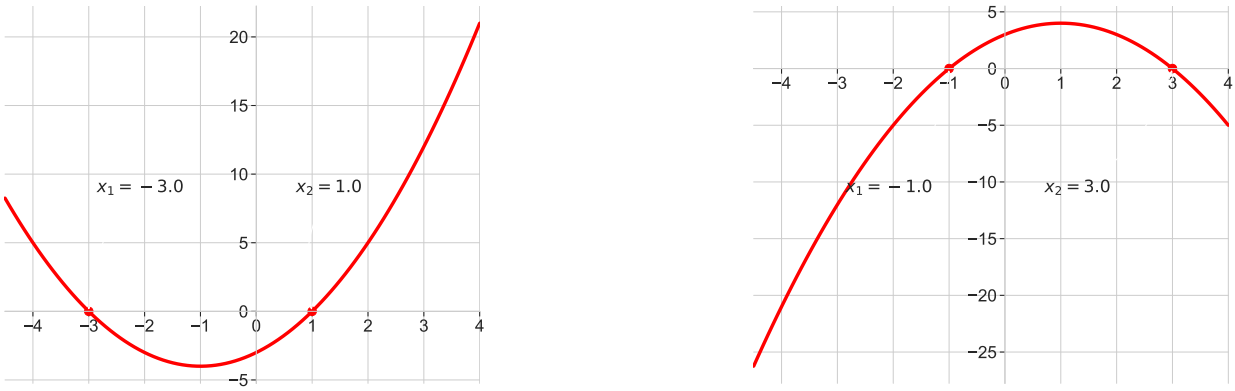
(a) (b)

Figura 1: Figuras del ejercicio 1

En la Figura 1 se muestran dos realizaciones diferentes de los números aleatorios, como se ve, tanto la media como la desviación estándar difieren significativamente, esto puede ser debido a que 100 muestras es una cantidad pequeña y no permite realizar una muy buena estadística.

Ejercicio 3-4

Usando la formula resolvente se programo una función que al pasarle como argumentos los coeficientes de un polinomio de grado 2, devuelva las raíces del mismo en forma de lista. Aparte se programo una función que al pasarle como argumento los coeficientes de un polinomio de grado 2 lo grafique y además, usando la función anterior, marque las raices del mismo en el gráfico. En la Figure 2 se muestra el gráfico realizado para dos polinomios diferentes.



(a) (b)

Figura 2: Figuras del ejercicio 4

Ejercicio 5-6

Se creó una clase `Lineal` con parametros `a` y `b` y con una funcion de llamada `call` que tome como argumento `x` y devuelva el valor $a \cdot b + b$. Luego se creó una clase `Exponencial` heredera de `Lineal` pero con una función `call` que devuelva el valor $a \cdot x^b$. En la Figura 3 se muestran gráficos de elementos de ambas clases para en el intervalo $[-5, 5)$

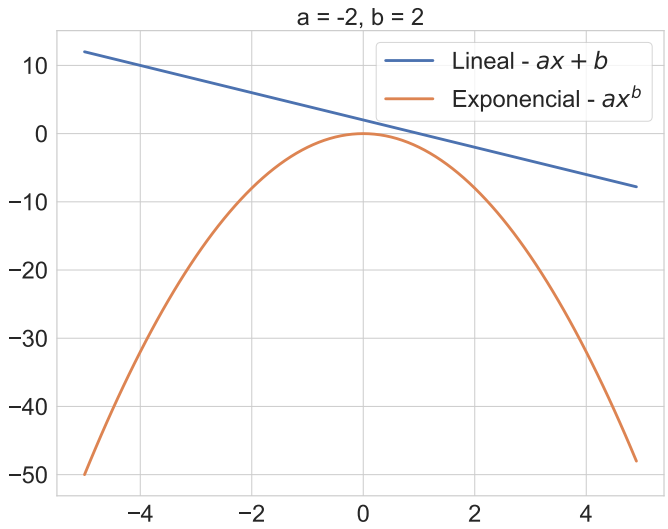


Figura 3: Graficos de los elementos de las clases `Exponencial` y `Lineal` con atributos `a`=-2 y `b`=2 al pasarles como argumento un `numpy array` de $[-5, 5)$.

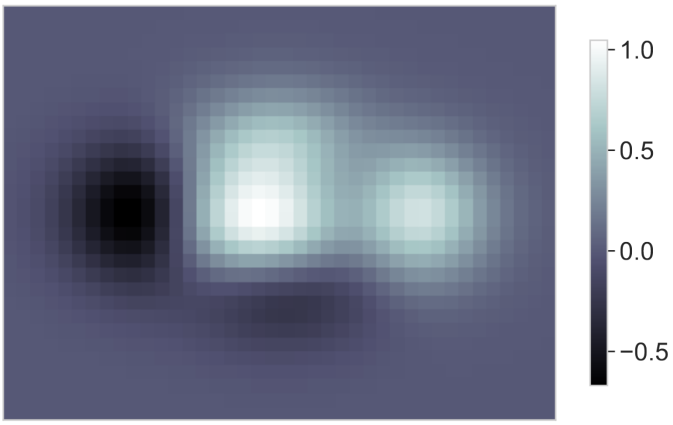
Ejercicios 7-8-9

Se verifico que importar la función `pi` era la misma sin importar de que forma se la importe, lo mismo paso con la función `area`. Se comprobó y se implementó correctamente la importación de paquetes previamente diseñados

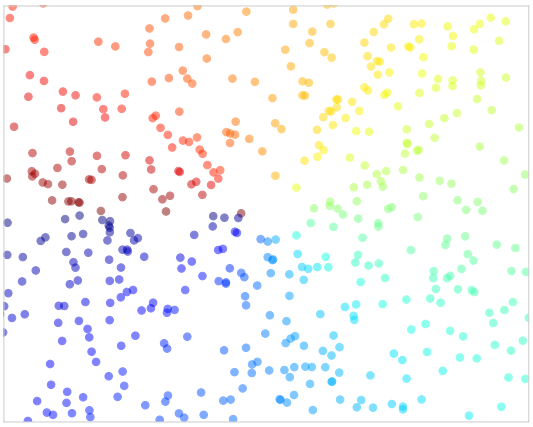
Ejercicio 10-11-12

Los gráficos mostrados en la Figura 4 quedaron bastante lindos y similares a los pedidos por el TP. Para el grafico de la Figura 4a se utilizó la función `plt.imshow` pasandole como argumento una función de dos arrays del mismo tamaño, para conseguirlos se utilizó la función `np.meshgrid` que permite crear n arrays de n dimensiones a partir de n arrays de una dimensión para su correcta utilización en funciones, se usó el parametro `origin='lower'` para invertir la imagen. Para crear la Figura 4c se crearon dos vectores aleatorios con distribución normal y se los grafico

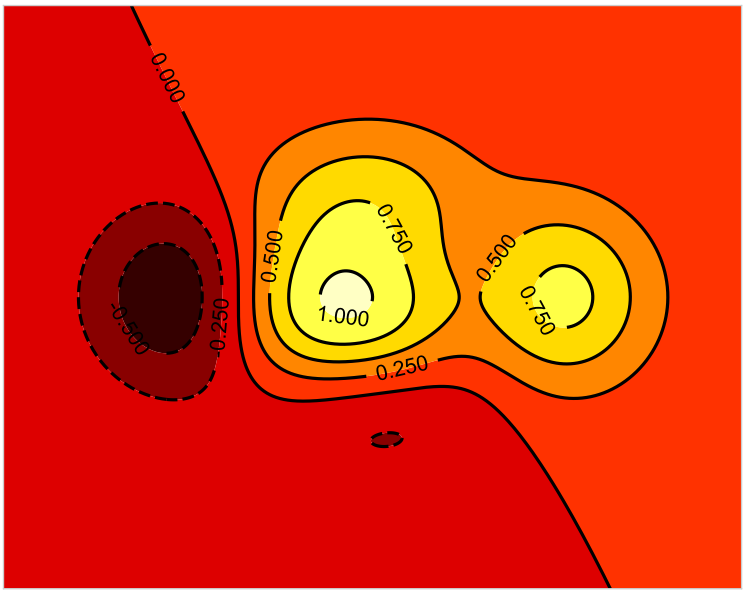
con `plt.scatter`, eligiendo `cmap=jet'` para la paleta de colores y los colores de acuerdo al angulo de los puntos. Para la Figura ?? se utilizaron las funciones `plt.contourf` y `plt.contourf`, eligiendo con `plt.clabel` manualmente las anotaciones.



(a) Gráfico obtenido al utilizar `plt.imshow` con la función f(x,y) definida en el ejercicio 10.



(b) `plt.scatter` con `cmap='jet'` de puntos aleatorios con distribución normal.

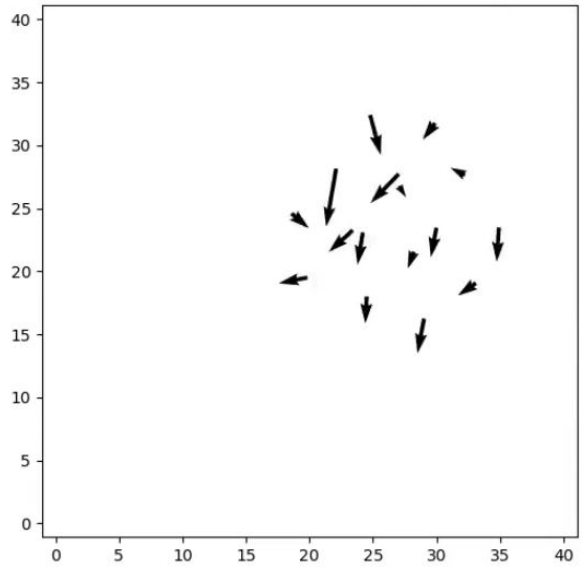


(c) `plt.contourf`+`plt.contour` con clabel elegidos manualmente (`manuat=True`).

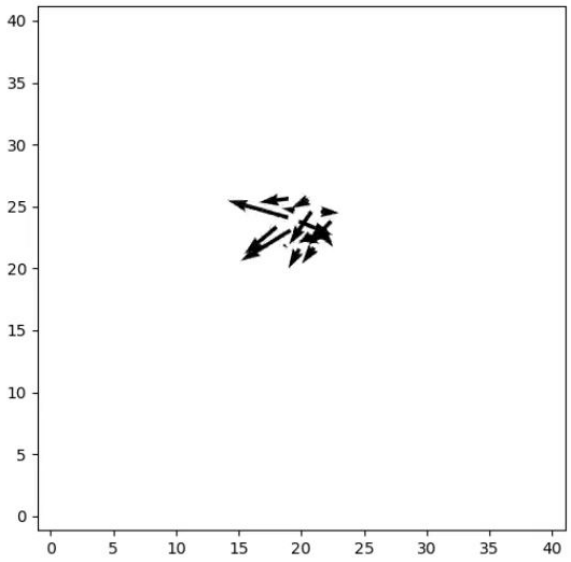
Figura 4: Figuras ejercicios 10-11-12

Ejercicio 13

Se implemento una clase `R2` con atributos `x` e `y` y diversas propiedades como suma, resta, división y multiplicación por un escalar, valor absoluto, etc. Luego se creo una clase `Pez` con atributos `pos` y `vel`, ambos de la clase `R2`. El `Pez`, además, puede moverse (modificar el atributo `pos`) y cambiar la velocidad (modificar el atributo `vel`), el cambio de velocidad del pez se ve afectado si este se encuentra cerca de una pared o si supera la velocidad máxima, definiendo la nueva velocidad de forma que se aleje de la pared y que no supere esta velocidad máxima definida respectivamente. La siguiente clase creada fue la clase `Cardumen`, que al iniciarse simplemente había que especificarle el tamaño del mismo, el atributo `size`. El cardumen puede inicializarse con una función `initialize` donde se definen los atributos `maxVel`, `maxDist`, `N` y se inicializa el atributo `peces`. El atributo `peces` es una lista de variables `Pez`, con sus respectivos `pos` y `vel` elegidos aleatoriamente de acuerdo a `N` y `maxVel`. El cardumen puede avanzar, con la funcion `doStep` que hace que todos los peces se muevan de acuerdo a diversos atributos del cardumen y los peces, como `centromasa` y `vmedia`. Se simuló el movimiento del cardumen variando los valores `maxDist` y se observo una clara influencia en el cardumen, estando el "tamaño"de este correlacionado con el atributo `maxDist`. Se incluye una carpeta con 2 videos de diferentes valores de `maxDist` donde se ve el efecto y se incluyen capturas de los videos en la Figure 5.



(a) `maxDist` = 5



(b) `maxDist` = 1

Figura 5: Figuras ejercicio 13 comparando las formas de los cardumenes con distintos atributos `maxDist`

Ejercicio 14

Se soluciono el problema usando el atributo `count` de las listas y unas secuencias `for` para revisarlas. En la tabla 1 se muestran las probabilidades obtenidas. Al ser muestras finitas, estos resultados cambian simulación a simulación.

Numero de personas	Probabilidad de mismo cumpleaños
10	11.3 %
20	41.7 %
30	69.4 %
40	91 %
50	96.9 %
60	99.2 %

Tabla 1: Probabilidad de que dos personas del mismo grupo compartan el día del cumpleaños en función del tamaño del grupo haciendo estadística sobre 1000 grupos.

Ejercicio 15

Se probó la función `Noiser f` y su versión vectorizada `f2` sobre una lista con elementos `float` y sobre la misma lista pero convertida a un `numpy array`. Al aplicar `f` a la lista hubo un error debido a que no se puede sumar un numero a una lista, al aplicar `f2` a la lista, en cambio, se consiguió el resultado esperado, sumándole a cada elemento de la lista un valor de ruido aleatorio. A aplicar `f` a un `numpy array`, el método funcionó, sin embargo no hizo lo esperado, ya que suma la misma componente de ruido a cada uno de los elementos de la lista, esto tiene sentido, ya que la suma entre un `numpy array` y un escalar esta permitida, pero definida como sumarle el mismo escalar a cada elemento de la lista. En cambio, al aplicar `f2` al mismo `numpy array`, vuelve a dar lo esperado, sumando un valor diferente a cada valor del array. Entonces concluimos que lo que hace la función al ser vectorizada es aplicarsele a cada elemento del argumento que se le pase por separado.