Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CO-6612, Introducción a las redes neuronales
Trimestre Enero-Marzo 2023
Tarea 5

Eros Cedeño 16-10216

Aproximador Universal y RBF

Las redes neuronales de funciones de base radial se caracterizan por tener un aprendizaje o entrenamiento híbrido. La arquitectura de estas redes se caracteriza por la presencia de tres capas: una de entrada, una única capa oculta y una capa de salida y son muy utilizadas para problemas de aproximación. En este informe se explican las consideraciones y resultados obtenidos al crear un Aproximador Universal y una RBF capaz de interpolar adecuadamente los datos recibidos en los archivos Spectra100.csv para luego realizar la comparación con la curva real caracterizada en el archivo SpectraReal.csv.

En los siguientes incisos se solicita elaborar las implementaciones de diferentes redes neuronales. Estas implementaciones se realizaron en el lenguaje de programación Python 3. Los requerimientos e instrucciones de ejecución están debidamente documentados en el repositorio entregado junto a este informe. Igualmente el repositorio se encuentra sincronizado en el siguiente link:

https://github.com/Eycer-usb/redes-neuronales

Inciso 2 a)

Se pide realizar la implementación de un Aproximador Universal simple y luego utilizarlo para interpolar los puntos en el archivo Spectra. La clase se encuentra en RBF/UniAprox.py. Para la aproximación se utilizó la función Gaussiana dentro de las 100 neuronas correspondientes a los 100 datos recibidos y luego se obtuvieron los pesos sinápticos resolviendo el sistema de ecuaciones que se muestra a continuación

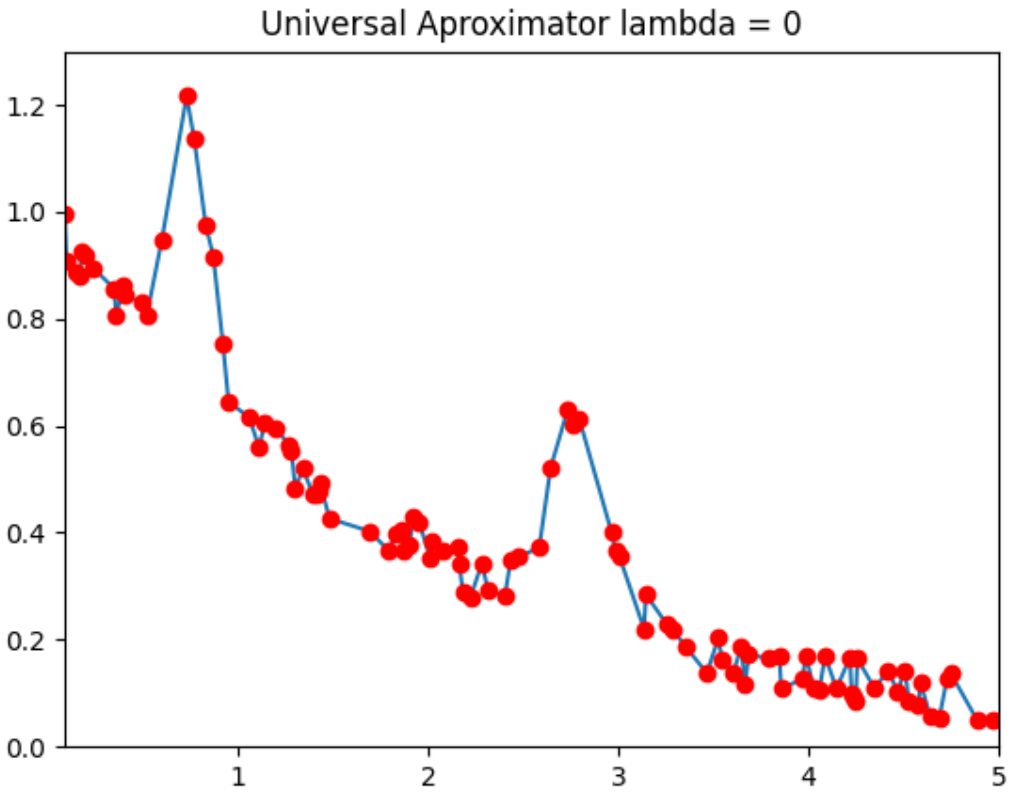
$$GW = d \Rightarrow W = G^{-1}d$$

En el programa cliente se instancia esta clase y se le pasa la primera columna de los datos de entrenamiento (correspondiente a las abscisas) que seran los estímulos y además se le envían las respuestas deseadas (las ordenadas) correspondiente a la segunda columna del csv.

Una vez realizado el entrenamiento de la red neuronal del aproximador universal se procede a evaluar su desempeño.

Evaluación empírica

Para la evaluación empírica se decidió graficar las respuestas obtenidas por la red a estímulos en el rango de 0 a con un intervalo de 0.5 entre cada punto y se grafica para comparar entre la curva obtenida y la curva deseada.



La curva obtenida muestra similitud con la gráfica real. Se observa que la curva pasa por todos los puntos de los datos de entrenamiento lo que nos sugiere un error cero en la interpolación sin embargo esta curva es altamente oscilatoria y con múltiples picos que alteran su suavidad. En general es común que en los datos hayan errores de medición, por lo que forzar la curva a pasar por los puntos de entrenamiento no es un criterio correcto ni

adecuado. Esto nos lleva al segundo inciso donde se introduce el factor de regularización para suavizar la curva.

Evaluación Analitica

Dado que se conoce la curva real del fenómeno estudiado se propone como métrica el error cuadrático medio entre la curva real y la curva estimada por cada punto. Con esta medida se podrá estimar la exactitud de la interpolación y comparar su desempeño en los incisos siguientes. En la imagen siguiente se muestra que el error promedio de la curva obtenida es de 0.0422

```
(env) [ec@Buzz RBF]$ python client.py
Programa Cliente Iniciado
Graficando Curva interpoladora y datos
0.042204760240124914
```

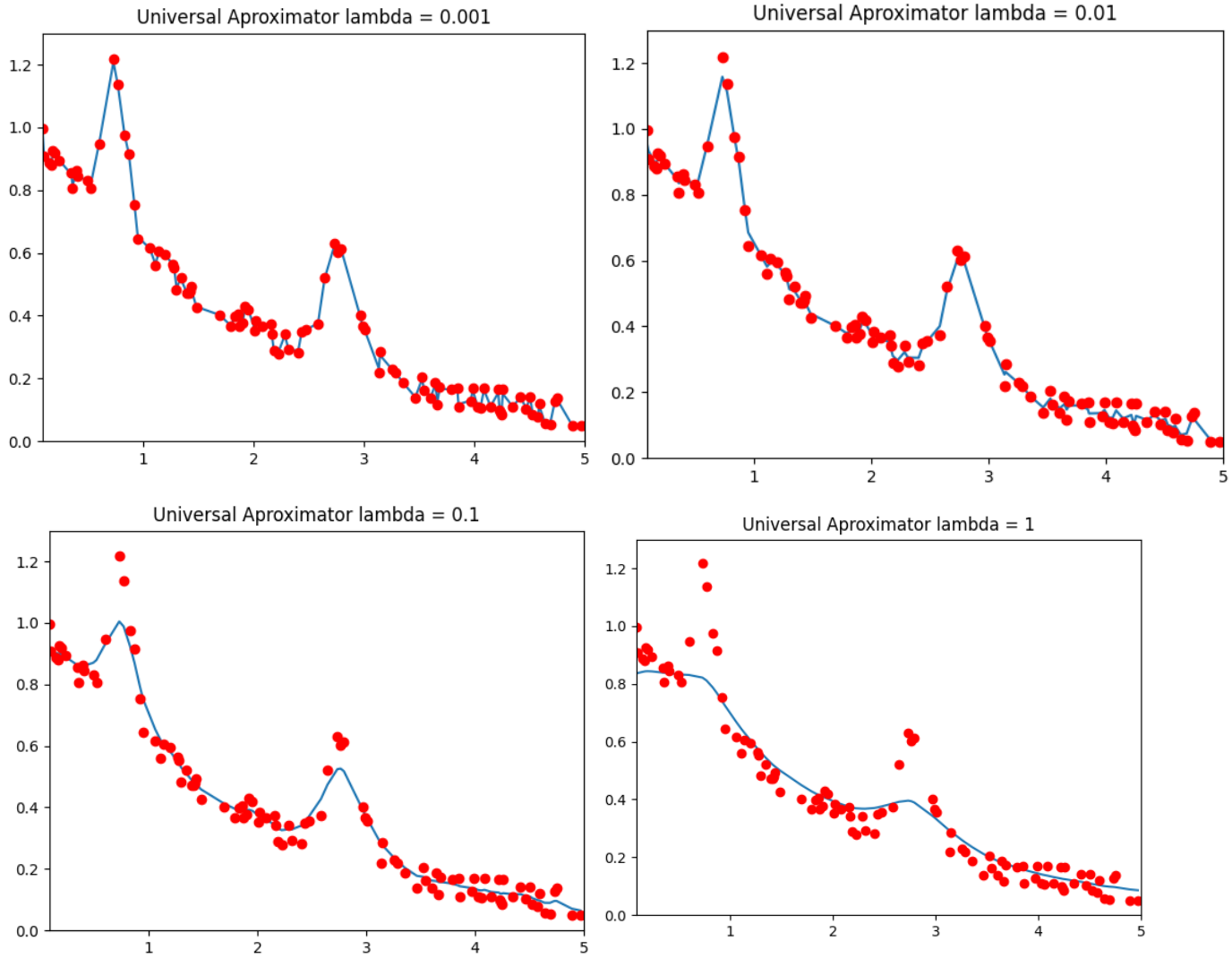
Inciso b)

Continuando con la implementación del inciso anterior se agrega como atributo de la clase el factor de regularización. Este factor de regularización (lamda λ) Es un parámetro de tolerancia en la distancia entre el dato de entrenamiento y el dato estimado por la red.

Para obtener los pesos basta con resolver el sistema de ecuaciones siguiente:

$$(G + I\lambda)W = d \Rightarrow W = (G + I\lambda)^{-1}d$$

Como el parámetro de regularización es arbitrario se realizó pruebas con distintos valores de λ que se muestran a continuación



En conclusión se escogió el factor $\lambda = 0.1$ Dado que la curva es adecuadamente suave, y aunque no alcanza el punto máximo de la curva real, este no se aleja demasiado de los valores reales. Para validarlo, se estudió el error cometido (métrica) respecto a la curva real y se obtiene un error suficientemente pequeño y el menor encontrado variando el lambda. Presenta un **error de 0.0349.**

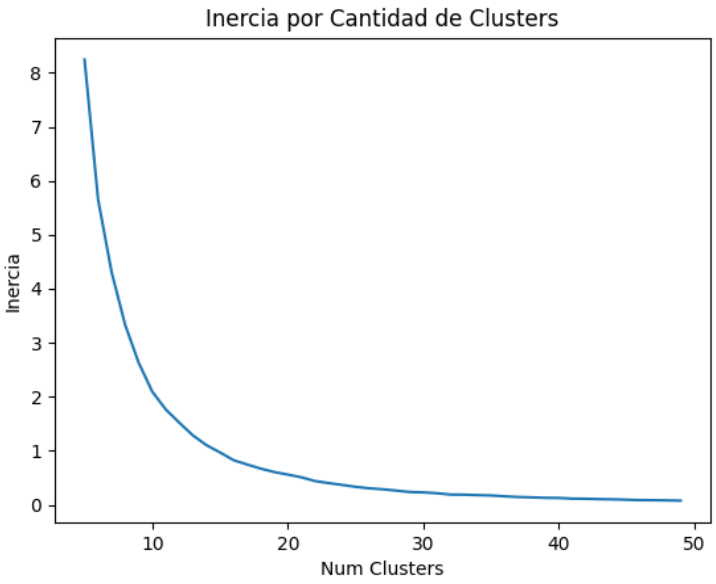
Inciso c)

Para este inciso se estudió el mismo problema, pero utilizando una RBF simple. El uso de la RBF presenta algunas ventajas como por ejemplo reducir el número de neuronas de la red. Sin embargo presenta un nuevo reto a considerar: Elegir la cantidad de neuronas adecuada para aproximar la función. Para resolver esto, se utilizó el *clustering* mediante el

algoritmo de las k-medias. Este algoritmo permite obtener k puntos tales que se minimiza la distancia media entre los datos y estos puntos también llamados centroides. Para determinar el número k de clusters se utilizó el método del Codo (Elbow Method) en el que se incrementa la cantidad de centroides y se estudia la inercia asociada a cada uno para luego seleccionar la menor cantidad de centros que minimice significativamente la inercia definida como se muestra a la derecha.

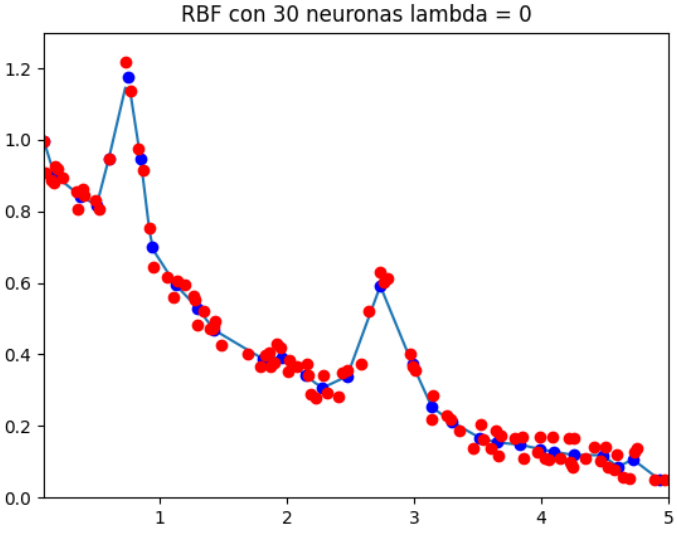
$$Inercia = \sum_{i=0}^{N} \|x_i - \mu\|^2$$

Para calcular los k centroides se utilizó la clase KMeans de la librería *scikit-learn*. Y se utilizó el algoritmo k-means++ que pone a disposición. Luego se graficaron las inercias obtenidas.



De la gráfica se observa que incrementar la cantidad de neuronas luego de 30 neuronas no aporta un cambio muy significativo. Por lo que según el método del codo este es un número óptimo de clusters.

Con la cantidad de clusters establecida en 30 se creó una RBF y a continuación se muestran los resultados obtenidos. Los puntos en Azul son los centroides obtenidos en el proceso de clustering.

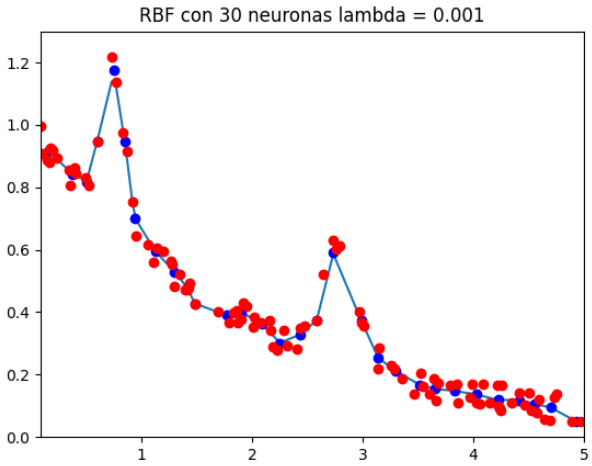


Como se observa en la imagen la gráfica obtenida no tiene demasiadas oscilaciones. Y al hacer el análisis métrico del error en comparación con el grafico real se obtiene un **error de 0.0384** unidades lo cual es un error bastante bajo

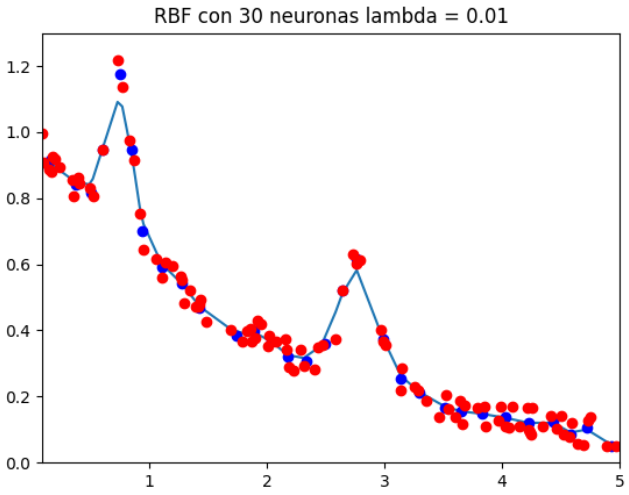
```
Graficando Curva interpoladora y datos
0.038465209612041265
```

Inciso d)

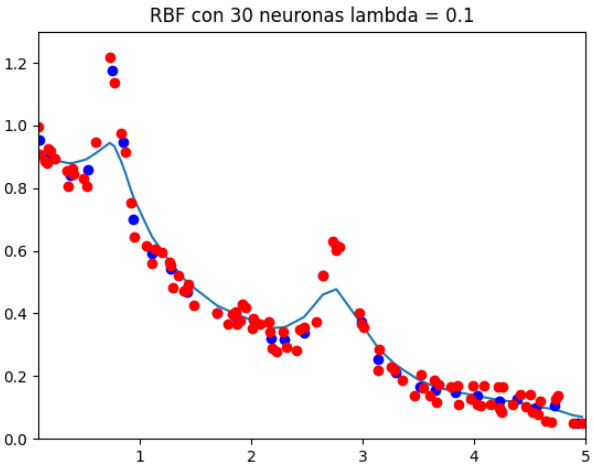
A la implementación del RBF anterior, se le agrega el factor de regularización. Conservando las 30 neuronas del inciso anterior se realizan pruebas sobre distintos valores de lambda que se muestran a continuación con su respectiva métrica de error.



Error 0.056



Error 0.063



Error 0.2101

Por lo tanto el valor del factor de regularización que mejor resultados dio fue el $\lambda = 0.001$ Dado que incrementar el factor de regularización implica que se está flexibilizando la curva y no incide adecuadamente sobre las medias obtenidas en el clustering.

Nota Adicional

Para finalizar el análisis y estudio de este tipo de dispositivos se implementa de nuevo el RBF pero esta vez el cálculo de los pesos asociados se realiza con la ayuda de un

dispositivo adaptativo lineal (Adaline) Los resultados obtenidos no son del todo satisfactorios sin embargo se describen a continuación por completitud.

La implementación con entrenamiento mediante un Adaline se encuentra en el archivo RBF/RBFAdaline.py. Al ejecutarse este instancia una clase y realiza la aproximación de los valores de los pesos considerando 50 neuronas, tasa de aprendizaje de 0.01, sesgo de 1 y máximo de epocas en 1000.