DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Redes Neuronales Artificiales

Redes Neuronales
Primer Cuatrimestre de 2022

Integrante	LU	Correo electrónico
Lucas Iván Kruger	799/19	Lucaskruger10@gmail.com
Sebastián Cantini Budden	576/19	sebascantini@gmail.com
Juan Cruz Barcos	463/20	juancruzbarcos@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Introducción

En el presente informe vamos a estudiar la implementación de una red neuronal basada en el aprendizaje no supervisado tomando de base los modelos de oja y sanjer. El trabajo sera realizado en Python utilizando la librería NumPy, la cual está altamente optimizada para la multiplicación de matrices. Por lo tanto, nos habilita a entrenar la red más rápido. Esta implementación de bajo nivel de red para aprendizaje no supervisado está adaptada al pseudocódigo visto en clase, que nos permite entender mejor los mecanismos detrás de esta. Experimentaremos sobre un set de datos representado como Bag of Words, buscando una reducción de la dimensionalidad, y contrastaremos los resultados.

Los modelos mencionados anteriormente solo consisten de la capa de entrada y la de salida, por lo que trabajaremos con una capa de entrada de 851 neuronas (ya que es la cantidad de componentes con los que partimos) y una capa de salida con 9 neuronas.

2. Implementación

Para la implementación de nuestro algoritmo nos basamos en el siguiente algoritmo:

$$
\begin{aligned}
&\text{for } X \in dataset\text{:}\\
&\quad Y = X \bullet W\\
&\quad \text{for } j \in < 1 \dots M >\text{:}\\
&\qquad \text{for } i \in < 1 \dots N >\text{:}\\
&\qquad\quad \tilde{X}_i = 0\\
&\qquad\quad \text{for } k \in < 1 \dots Q >\text{:}\\
&\qquad\qquad \tilde{X}_i += Y_k \cdot W_{ik}\\
&\qquad\quad \triangle W_{ij} = \eta \cdot (X_i - \tilde{X}_i) \cdot Y_j\\
&W += \triangle W
\end{aligned}
$$

Figura 1: Pseudocódigo dado en clase

Para corregir las matrices de peso utilizamos los métodos de oja y sanjer. La diferencia entre el modelo de oja y el modelo de sanjer, es que el primero tiene a M como valor de Q. Es decir que en el ultimo ciclo iterara por cada uno de los componentes. Por otro lado, el modelo de sanjer tiene a j por como valor de Q, por ende la sumatoria arrancaría tomando en cuenta un único componente y progresaría hasta incluirlos todos. El impacto que esto tiene es que el modelo de sanjer encuentra los componentes principales en orden.

$$
\begin{aligned}
&\text{OjaGen:} \quad \triangle W_{ij} = \eta \cdot (X_i - \tilde{X}_i) \cdot Y_j \quad \tilde{X}_i = \textstyle\sum_k^M W_{ik} \cdot Y_k\\
&\text{Sanger:} \quad \triangle W_{ij} = \eta \cdot (X_i - \tilde{X}_i) \cdot Y_j \quad \tilde{X}_i = \textstyle\sum_k^j W_{ik} \cdot Y_k
\end{aligned}
$$

Figura 2: Ecuaciones del modelo OjaGen y Sanger

3. Experimentación

3.1. Learning Rate

El learning rate es un parámetro que impacta únicamente al aprendizaje. Determina la velocidad de convergencia del modelo. Es muy importante buscar un buscar un buen valor para este, ya que un learning rate muy alto puede hacer que nuestro modelo diverja mucho y salga de mínimos locales, y uno demasiado pequeño tomaría demasiadas iteraciones en converger o podría no salir de algunos mínimos locales. Probaremos diferentes learning rates para cada método y estudiaremos cómo varía el error respecto a las iteraciones con cada uno. Decidimos medir distintos learning rate desde 0.05 hasta 0.8 aumentando de a 0.05, esto nos permite tener un gran rango y encontrar el mejor learning rate para cada modelo. Consideramos que el sistema converge si luego de un numero de iteraciones la ortogonalidad es menor a 0.1, sin embargo, como un algunos casos el sistema puede oscilar, pusimos un máximo de 8000 iteraciones.

	Sanger		Oja	
lr	iters	orthogonality	iters	orthogonality
0.05	8000	3.0008434039444953	8000	3.6096062238593722
0.1	8000	2.5894664046776716	8000	0.3127470618703172
0.15	8000	2.899045393831873	5994	0.0999048409435386
0.2	8000	0.48490386456745094	4376	0.09984453426525584
0.25	8000	0.30652709456036287	3636	0.09990267356040475
0.3	8000	0.20534451635947748	3014	0.09980741044957345
0.35	8000	0.19052627310852033	2664	0.0998861806274936
0.4	8000	0.23695104829869307	2476	0.0998391520979227
0.45	8000	0.1767324545451615	2007	0.09981845395085528
0.5	8000	0.18246506056570994	1941	0.09980609977638341
0.55	8000	0.10053695409035501	1674	0.09962812137824817
0.6	8000	0.15449949884399977	1555	0.09988013193581491
0.65	8000	0.11956105378466067	1519	0.09961120808508818
0.7	8000	0.11752385366242046	1307	0.0997386284692024
0.75	8000	0.1401581382732140	1303	0.09978949586066144
0.8	8000	0.12918439869833043	1150	0.09995709277815455

Como podemos ver en estos resultados, el modelo de oja converge mucho mas rápido que Sanger, necesitando un learning rate de 0.15. Al buscar los componentes en orden, Sanger sacrifica tiempo de aprendizaje. Ambos modelos se benefician de learning rates altos, aunque hay que notar que luego de el learning rate de 0.55, en sanger, la ortogonalidad se vuelve inconsistente por lo que tomaremos este como el indicado para representar los resultados en gráficos tridimensionales. Oja es consistente en su convergencia por lo que tomaremos 0.8 como el learning rate óptimo para este modelo.

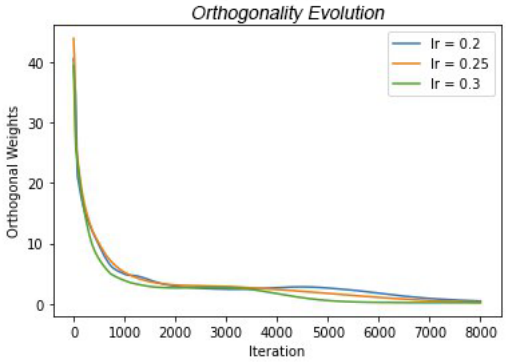

Figura 3: Los primeros tres componentes del output del modelo de sanger.

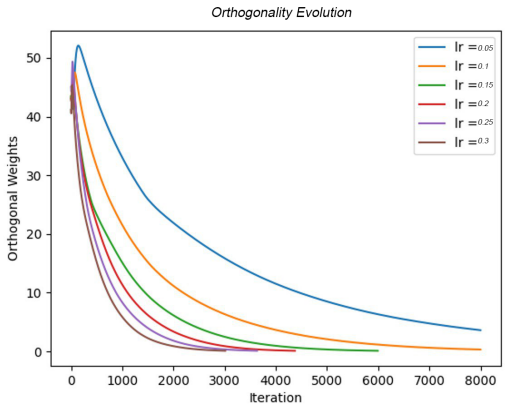

Figura 4: Los segundos tres componentes del output del modelo de sanger.

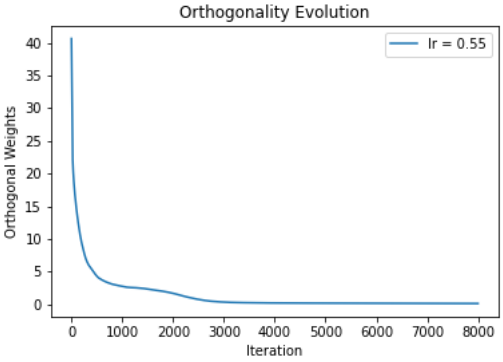

Figura 5: Los primeros tres componentes del output del modelo de sanger.

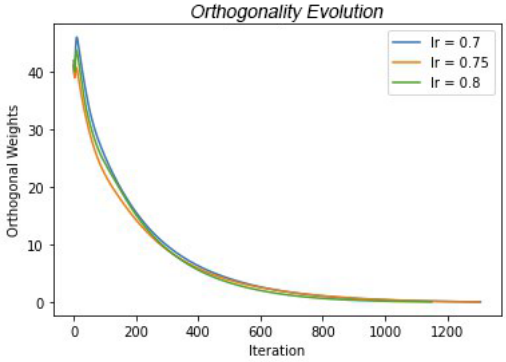

Figura 6: Los segundos tres componentes del output del modelo de sanger.

Como podemos ver en ambos metodos obtuvimos una clara convergencia a 0 en la ortogonalidad conseguida por el sistema. Llegando a valores como los vistos arrriba

3.2. Representación de respuestas en R^3

Para esta representación tomamos 3 gráficos. Como la respuesta de la red es el vector Y $\in R^9$, la figura 1 tendrá por ejes a Y_1, Y_2, Y_3, la figura 2 a Y_4, Y_5, Y_6, y la figura 3 a Y_7, Y_8 e Y_9.

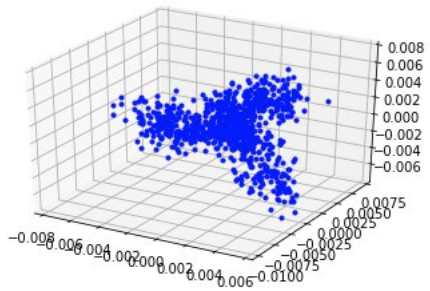

Figura 7: Los primeros tres componentes del output del modelo de sanger.

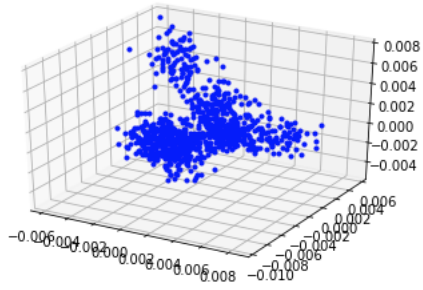

Figura 8: Los segundos tres componentes del output del modelo de sanger.

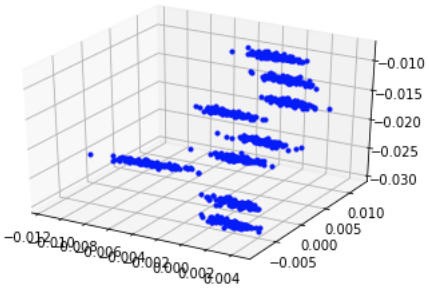

Figura 9: Los últimos tres componentes del output del modelo de sanger.

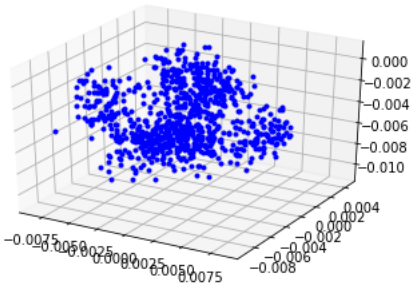

Figura 10: Los primeros tres componentes del output del modelo de oja.

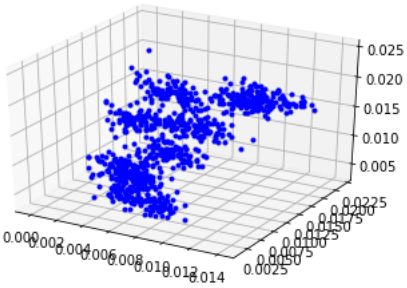

Figura 11: Los segundos tres componentes del output del modelo de oja.

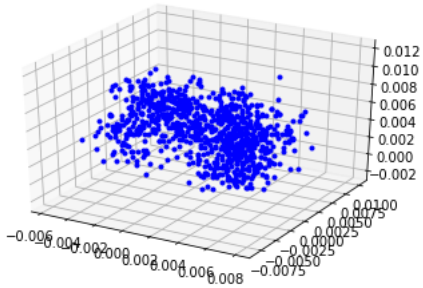

Figura 12: Los últimos tres componentes del output del modelo de oja.

Algo para notar es que las figuras que forman los resultados de sanger, son mas compactas y con una forma mas rígida mientras que los resultados de oja tienen un cuerpo mas amplio y mas abstractos. Esto se debe a que sanger obtiene los componentes de manera ordenada. Oja, por otro lado, no sabemos el orden de estos entonces no hay relación en la agrupación de estos.

Como podemos ver en los primeros componentes de sanger hay una clara diferenciación en los datos. A medida que nos acercamos a los últimos, estos van perdiendo esta diferenciación. En cambio en el método de oja vemos menos diferencia entre los datos. Esto puede deberse a que no tiene los componentes principales ordenados.

Esperábamos resultados mas apreciables en el método de oja, debido a que obtuvo mayor ortogonalidad. Pero nuestra hipótesis fue refutada, esto puede deberse a lo explicado antes del orden de los componentes.

4. Conclusión

Como pudimos observar los métodos de oja y sanger pueden ser utilizados para la reducción de la dimensionalidad. Estos algoritmos presentaron mejores resultados en el caso de sanger, pero a costo de muchas mas iteraciones. En ambos casos el learning rate necesario fue alto El haber trabajado directamente con el algoritmo nos permitió entender la arquitectura en profundidad.