

# Redes Neuronales Básicas

## Parte 1 – Python y Numpy

**Oscar Fernando López Barrios - Carné 20679**

**Alejandro José Gómez Hernández - Carné 20347**

En clase vimos un modelo simple, utilizando solo Python y Numpy, para resolver Regresiones Lineales mediante redes neuronales. Utilizando el código desarrollado (o si lo desea uno propio), responda a las siguientes preguntas:

- 1) Cambie el número de observaciones a 100,000. Explique qué es lo que ocurre en términos de:
  1. El tiempo de ejecución para resolver el problema

**R//** En este caso el tiempo de ejecución aumenta de mayor manera debido a que al momento de realizar tantas observaciones más, los cálculos que se deben de realizar también aumentan. Por lo tanto, el tiempo que se ejecuta para resolver problemas es mayor.

2. El resultado final vrs lo encontrado en clase: es igual, o diferente. ¿Por qué?

**R//** En general el resultado final se diferencia al obtenido durante la clase debido a que se cuenta con una mayor cantidad de tiempo de ejecución, además que se puede notar que cuando se realiza esta nueva prueba con la cantidad definida la pérdida aumenta con respecto a la cantidad de las 1000 observaciones que se tenía antes en clase.

3. Las gráficas para representar los datos/resultados

**R//** Se puede observar a comparación de las gráficas anteriores que existen diferencias con respecto a como se mira la cantidad de datos que existen dispersos por toda la gráfica, en general se puede ver una cantidad de datos mayormente juntos y la gráfica se puede ver totalmente lineal.

- 2) Cambie el número de observaciones a 1,000,000. Explique qué es lo que ocurre en términos de:
  1. El tiempo de ejecución para resolver el problema

**R//** Así como pasó con el ajuste anterior el tiempo de ejecución fue mucho mayor al que brindaba la ejecución en la que se realizaron 1000 observaciones. En este caso realmente hay una gran diferencia debido a la gran cantidad de cálculos que se deben de realizar.

2. El resultado final vs lo encontrado en clase: es igual, o diferente. ¿Por qué?

**R//** En general el resultado es distinto a lo que se logró encontrar durante la clase, debido a que 100000 de iteraciones brindan resultados más precisos, esto debido a que la cantidad de iteraciones brinda resultados más acertados, principalmente porque en varios casos existe la forma de que a más cantidad de iteraciones se logra un resultado más preciso.

Además, se puede ver la diferencia que existe entre los pesos y los sesgos que se tienen:

1000 Observaciones

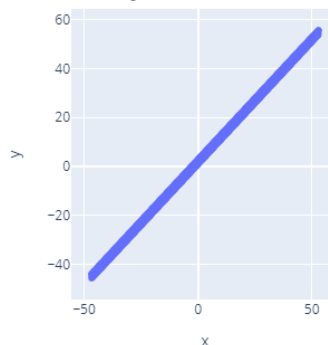
```
print(pesos, sesgos)
✓ 0.0s
[[ 2.01604999]
 [-3.00420609]] [3.19213183]
```

1000000 Observaciones

```
print(pesos, sesgos)
✓ 0.0s
[[ 2.00003351]
 [-2.99983041]] [3.18070216]
```

3. Las gráficas para representar los datos/resultados

**R//** En este caso la gráfica que brinda al momento de representar los resultados se puede ver de manera más clara la forma lineal de esta, además se puede ver como existe mayor densidad en los datos que contiene esta gráfica.



- 3) “Juegue” un poco con el valor de la tasa de aprendizaje, por ejemplo 0.0001, 0.001, 0.1, 1.  
Para cada uno de estos indique:

1. ¿Qué ocurre con el tiempo de ejecución?

**R//** Al momento de realizar estos cambios se puede visualizar por medio de la ejecución de tasas de aprendizaje más pequeñas el tiempo aumenta, esto debido a que al momento de tener una tasa de aprendizaje de menor tamaño las iteraciones que se deben de realizar para que esta logre el objetivo son mayores.

2. ¿Qué ocurre con la minimización de la pérdida?

**R//** La minimización de la pérdida realmente se ve afectada por los cambios en la tasa de aprendizaje, debido a que cuando se utilizan cantidades menores la tasa de pérdida es menor y cuando se utilizan tasa de aprendizaje mayor la tasa de pérdida es mayor.

3. ¿Qué ocurre con los pesos y los sesgos?

**R//** En el caso de los pesos y los sesgos se puede conocer que dependiendo de la cantidad de la tasa de aprendizaje que se tenga, puede ser mayor la divergencia entre los valores de estos. En el caso de una tasa de aprendizaje mayor puede llevar a valores fuera del rango que se espera.

```
print(pesos, sesgos)
✓ 0.0s
[[-2.43453136e+151]
 [ 3.48772067e+151]] [-4.38783404e+147]
```

4. ¿Qué ocurre con las iteraciones?

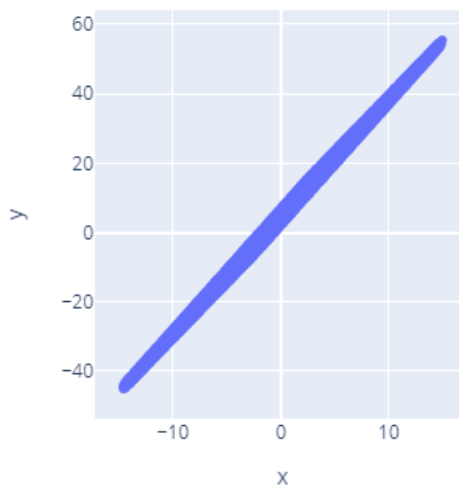
**R//** En el caso de las iteraciones, un número de tasa de aprendizaje menor hace que la cantidad de iteraciones deba de ser mayor para lograr un resultado óptimo con respecto a la forma en la que funciona el modelo que se está analizando.

5. ¿El problema queda resuelto o no?

**R//** La resolución del problema depende del número de la tasa de aprendizaje que se elija, esto debido a que dependiendo de la tasa de aprendizaje se puede obtener resultados, pero en algunos casos dependiendo de esta, los resultados son más precisos y mejores.

6. ¿Cuál es la apariencia de la última gráfica? ¿Se cumple con la condición de que sea de 45 grados?

**R//** Por medio de la ejecución se puede visualizar que la gráfica realmente si cumple con la condición de que sea de 45 grados.



- 4) Cambie la función de pérdida “L2-norm” a la misma pero sin dividir por 2. Explique lo que ocurre en términos de:

1. El tiempo que se tarda el algoritmo en terminar, comparado a lo que vimos en clase

**R//** El tiempo en el que se ejecuta realmente cambia a un tiempo un poco menor, debido a que existen menos cálculos que se deben de realizar la momento de lograr encontrar los valores de la pérdida.

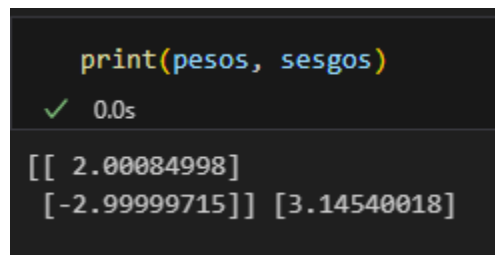
2. Si la pérdida se minimiza igual que lo que vimos en clase

**R//** En este caso sucede lo contrario, la pérdida que se tiene realmente aumenta y se puede notar por medio de los valores que brinda el modelo.

```
484.3362032229452
229.26143185032967
115.59404708820612
64.79474643500049
41.94879195233597
```

3. Si los pesos y sesgos son parecidos a los vistos en clase

**R//** En este caso los pesos son un poco distintos a los que se vieron en la clase, pero en este caso se puede visualizar mediante los resultados que los pesos y sesgos brindan mejores resultados a los que anteriormente se deberían de haber obtenido.



```
print(pesos, sesgos)
✓ 0.0s
[[ 2.00084998]
 [-2.99999715]] [3.14540018]
```

4. Si el problema se resuelve como ocurrió en clase

**R//** Realmente el problema se resuelve y brinda un resultado correcto, pero en este caso se puede observar como la cantidad de pérdida es mayor a la que se tenía al momento de realizar las pruebas durante la clase.

5. Si se obtiene un mejor resultado al hacer más iteraciones

**R//** Al momento de realizar un mayor número de iteraciones, estas logran que el modelo se ajuste de una manera mejor, pero esto no brinda la certeza de que el modelo esté hecho de manera correcta debido a que puede existir overfitting.

- 5) Cambie la función de pérdida de la “L2-norm” a la “L1-norm”. Explique lo que ocurre en términos de:

1. El tiempo que se tarda el algoritmo en terminar, comparado a lo que vimos en clase

**R//** En cuanto al tiempo que tarda en completarse un algoritmo, suele tardar más que los conceptos aprendidos en clase. La razón principal de esto es que la función de pérdida de la “L1-Norm” no es diferenciable en todos los puntos, lo que dificulta la aplicación de técnicas de optimización eficientes, como el descenso de gradiente. Como resultado, el algoritmo puede requerir más iteraciones para alcanzar la convergencia, lo que resulta en tiempos de ejecución más prolongados.

2. Si la pérdida se minimiza igual que lo que vimos en clase

**R//** La optimización de pérdidas con la función de L1-Norm es diferente de lo que se enseñó con la L2-Norm. La L1-Norm penaliza las discrepancias absolutas entre los valores previstos y los reales y, por lo tanto, puede proporcionar una solución más robusta para los valores atípicos. Sin embargo, la mencionada falta de diferenciabilidad puede dificultar la minimización de esta pérdida. Como resultado, la convergencia es más lenta y la pérdida final puede no ser tan pequeña como en el caso de L2-Norm.

3. Si los pesos y sesgos son parecidos a los vistos en clase

**R//** El cambio en la función de pérdida tendrá un impacto en los gradientes y las actualizaciones de los parámetros, lo que, a su vez, afectará los valores finales de los pesos y sesgos. En consecuencia, los pesos y sesgos obtenidos mediante el uso de la función de pérdida "L1-norm" serán distintos de aquellos obtenidos con la función "L2-norm" que se discutió en clase.

4. Si el problema se resuelve como ocurrió en clase

**R//** En relación a si el problema se resuelve de manera similar a lo visto en clase, la aproximación para encontrar una solución puede ser similar. Sin embargo, debido a las modificaciones en la función de pérdida y los cálculos asociados, la solución final puede diferir en términos de los valores específicos de los pesos y sesgos. Además, como se mencionó previamente, la convergencia puede ser más lenta y la pérdida final puede ser más alta en comparación con el uso de la función "L2-norm".

5. Si se obtiene un mejor resultado al hacer más iteraciones

**R//** En términos de obtener un mejor resultado a mayor número de iteraciones, podemos decir que no va a tener ningún impacto que realmente sea reflejado en los resultados con L1-Norm porque no es diferenciable en todos los puntos.

6. ¿Tendrá una de estas más limitaciones que la otra?

**R//** Sí, L1 tiene una ventaja sobre L2 porque maneja mejor los datos atípicos, pero tiene el problema de diferenciación mencionado previamente. Depende mucho del contexto y la data para saber con cual atacar.

- 6) Cree una función  $f(x_1, x_2) = 13 * x_1 + 7 * x_2 - 12$ .
1. ¿Funciona el algoritmo de la misma forma?

R// El algoritmo funciona de la misma forma. Lo único es que se pueden ajustar mejor los pesos/sesgos de la red neuronal y así minimizar la pérdida. Pero el algoritmo sigue funcionando igual.

## Parte 2 – Tensorflow2

En clase vimos un modelo simple de una red neuronal utilizando TensorFlow 2. Utilizando el código desarrollado (o si lo desea uno propio pero que funcione correctamente), responda a las siguientes preguntas:

1. Cambie el número de observaciones a 100,000. ¿Qué ocurre?

R// El algoritmo tarda más tiempo que antes.

2. “Juegue” un poco con la tasa de aprendizaje. Los valores como 0.0001, 0.001, 0.1, 1 son interesantes para observar ¿Qué diferencias se observan? ¿Se comporta bien el algoritmo?

R// Al modificar la tasa de aprendizaje, hay disparidades entre 0.0001, 0.001, 0.1 y 1. Una tasa de aprendizaje más reducida podría conducir a una convergencia más pausada, pero más precisa. Y una tasa de aprendizaje más elevada, podría propiciar una convergencia más veloz, pero fluctuante en resultados.

3. Cambie la función de pérdida. Una función alternativa es la “Huber Loss”.

La función de pérdida Huber es más adecuada que la L2.norm cuando tenemos valores atípicos, ya que es menos sensitiva a los mismos (en nuestro ejemplo no tenemos valores atípicos, pero seguramente se topará con ellos en el futuro). La L2-norm eleva todas las diferencias al **cuadrado**, por lo que los valores atípicos tienen mucha influencia sobre los resultados. La sintáxis correcta de la función de pérdida Huber es “huber\_loss”.

¿Cómo se comparan los resultados al cambiar la función de pérdida?

R// Los resultados son menos sensibles a data atípica. Llega a suavizar posibles cambios entre lo estimado/obtenido.

Referencia:

[https://www.tensorflow.org/versions/r1.15/api\\_docs/python/tf/keras](https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/keras)