

Apuntes del 3/04/2025

Steven Pérez Machado

Abstract—El presente documento son los apuntes realizados en la clase de jueves del 3/04/2025, donde se vieron varios temas, se comenzó con una sección de noticias, un repaso sobre los temas vistos en la clase del martes 1/04/2025 y luego como temas nuevos se estudiaron.

I. NOTICIAS

- Artículo que hace crítica al modelo de OpenAI Sora, que es el modelo que se encarga de realizar las imágenes y video, y lo que menciona el artículo es el modelo está plagado de Sexismo, Racismos y personas con alguna discapacidad, entonces ya que el modelo está hecho con datos si esos datos están sesgados, las respuestas de ese modelo igualmente serán sesgadas.

- Otra noticia que se menciona es lo de creación de imágenes de openAI, y como ha afectado a los creadores digitales y como estos artistas se quejan ya que el trabajo que a ellos les dedican tiempo la IA puede hacerla en segundos.

- Se menciona también un nuevo benchmark open ai (Papernbench), es como la capacidad de una IA replicar el paper de investigación de otra IA

II. REPASO CLASE DEL MARTES 1/04/2025(REDES NEURONALES)

A. Clasificación de MNIST

Se tienen formas de encontrar relaciones entre variables de un dataset y se tiene una forma de encontrar una relación lineal entre ellas, la cual es

$$X_1, X_2, \dots, X_n$$

$$W_1, W_2, \dots, W_n$$

Donde la relación entre ambas es de forma lineal, es decir X_1 con W_1 y X_2 con W_2 y así sucesivamente.

Con la idea anterior se lleva a otro problema para predecir valores continuos en la recta, pero se llega a un punto donde si se aplica una función no lineal se puede aplicar una no linealidad a la salida, entonces se tiene una relación lineal en el input a raíz de la función $wx + b$, pero en la salida se le puede aplicar una regresión logística que nos sirve para resolver problemas de clasificación binaria.

Lo que se busca es aplicarle a imágenes el mismo concepto de un clasificador, entonces se tiene un dataset ya hecho con imágenes que son originalmente de tamaño 128x128 pixeles y luego transformadas a 28x28(784 features) pixeles y están distribuidas en un sólo canal.

B. Regresión logística(clasificación binaria)

En la regresión logística tenemos un vector de entradas que representa cada uno de los features, y lo que se hace para convertir la imagen y que sirva de input para la regresión logística es que se le aplique un flatten, es decir aplastar la imagen en una tira de caracteres y esto se le pasa a la regresión logística para que uno de estos sea un feature.

Regresión Logística

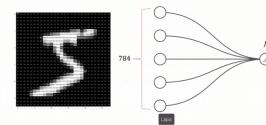


Fig. 1. Regresión logística

Va a tener un total de 784 features, donde cada pixel lo que me representa un sólo feature con un peso de X_1 asociado a este, ese procedimiento se hace hasta conseguir el $X=784$, el $W=784 + b$, donde b significa el bayaz.

C. Regresión logística multinomial(multiclas)

Ya sabemos hacer una clasificación binaria(0 y 1), pero queremos hacerlo para más datos, es decir que nos diga sólo si el evento ocurre o no ocurre, pero para todos los valores que se desean.

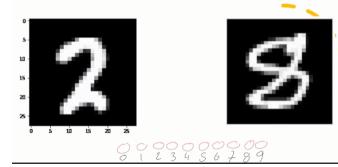


Fig. 2. Regresión logística multinomial

El ejemplo que se propuso en clase fue tratar de conseguir un One hot vector, donde se tenían 10 regresiones y cada una de estas se especializa en visualizar o definir si es un número o no, después se le pasaba la imagen como entrada a cada una de esas regresiones logísticas, esto lo que nos responde es cada uno de los veredictos de las regresiones.

Con el one hot vector puedo ver a cuál de todas las regresiones pertenece una codificación en particular.

D. Representación de 1 Feature(pixel)

Si la imagen es lo realmente pequeña para poder representarla con un pixel, lo que se hace es que esa salida se le pasa

o se le distribuye a todas las regresiones logísticas, para que determinen si son las clases que están determinadas a ver o no.

E. Representación de dos pixeles

En el mismo caso de que sean dos pixeles los que se tienen para representar la imagen, lo que se hace es que se distribuye entre todas las regresiones, y como conclusión tenemos que esto idea es expandible para todos los features que se tengan.

F. Conversión de vector a matriz

En lugar de cada regresión lineal de forma vectorial y hacer más sencillas las operaciones que se van a realizar, se recomienda hacer la conversión de cada uno de los vectores de pesos de cada regresión logística en una matriz.

G. Matriz de pesos $W()$

Cada uno de los pesos se convierte en una matriz y cada fila representa una neurona o regresión logística que se está observando y los features la cantidad de columnas que se tiene, es decir que j representa el tamaño de la siguiente capa y n representa el tamaño de los features.

H. ¿Cuál debería ser el tamaño de la matriz?



Fig. 3. Tamaño de la matriz

Es decir que por cada salida se tiene una nueva fila en la matriz y por cada regresión logística una columna nueva.

Por cada regresión logística se tiene un escalar, por lo que se tiene un vector de bayas, es decir lo que voy a tener es un vector de bayas.

I. Clasificación binaria(de nuevo)

Se aplica no linealidad a los datos de entrada, ahora se van a requerir los siguientes datos para poder resolver muchos más problemas:

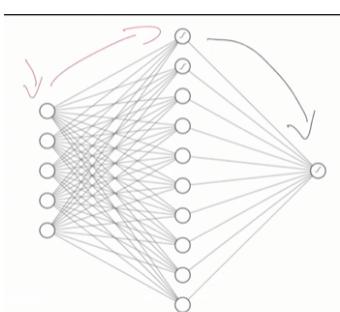


Fig. 4. Clasificación binaria

- Capa de entrada - Capa intermedia - Capa de salida

Pero la última capa es calculada basándose en la interacción de la capa anterior, y esta capa anterior ofrece no-linealidad

Lo que necesito para ir resolviendo los problemas es ir modificando cada uno de las capas de salida, ya sea aumentando o disminuyendo, o cambiando la función de activación.

J. Redes neuronales

Tenemos no linealidad que nos sirve para resolver problemas complejos.

Se tiene un nuevo hiper parámetro que es la cantidad de capas que necesito, aunque va mucho de la mano con la capacidad computacional con la que se cuenta.

Lo importante aquí es que se necesita que esas capas sean distintas, porque se va a propagar el error por cada una de las regresiones logísticas para poder optimizar los parámetros, si no son distintas no puede aplicar el descenso del gradiente.

Propagation: propaga el error o derivadas por cada una de las neuronas, se hace la sumatoria ponderada para saber cuanto afecta o contribuye a mi función de perdida

III. CLASE DEL 3/4/2025(REDES NEURONALES)

A. El perceptrón(Regresión logística):

Se descubrió en los años 60 frank roserblant propone una regresión logística que ayuda a resolver a varios problemas, y en lugar de utilizar otro loss function, esta utiliza Hinge Loss.

Minsky y otros señalan que tiene varios problemas, uno de ellos que los requerimientos computacional eran bastante altos para la época, y con una regresión logística no se puede resolver el XOR, por lo que para ellos que un algoritmo no solucione eso es inaceptable, por esto se detienen todas las investigaciones de la IA, y a este periodo se le conoció como "Invierno de la IA".

B. Predicción de compuertas lógicas

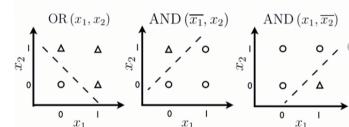


Fig. 5. Compuertas lógicas

En la imagen se muestran las compuertas lógicas OR y AND representadas en gráficos bidimensionales, donde:

- OR (X_1, X_2): Los triángulos representan la salida 1 y los círculos la salida 0. La compuerta OR da como resultado 1 si al menos una de las entradas es 1.
- AND (X_1^-, X_2): Muestra la compuerta AND con la primera entrada negada. El resultado es 1 (triángulo) solo cuando la primera entrada es 0 y la segunda es 1.
- AND (X_1, X_2^-): Representa la compuerta AND con la segunda entrada negada. El resultado es 1 (triángulo) solo cuando la primera entrada es 1 y la segunda es 0.

La línea punteada en cada gráfico representa el límite de decisión que separa las dos clases de salida (0 y 1). Esta visualización ayuda a entender cómo las compuertas lógicas clasifican sus entradas en el espacio bidimensional.

C. Problema del XOR

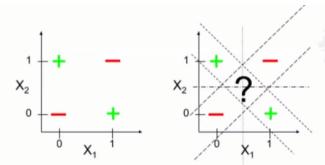


Fig. 6. Problema del XOR

Su principal problema es que no es linealmente separable, entonces el algoritmo del perceptrón ya no sabía qué tenía que hacer, es aquí donde aparecen las redes neuronales o perceptrón multicapa a resolver el problema, ya que estos si pueden resolver problemas multicapa.

Perceptrón o multicapa, puede resolver problema no lineales, lo que nos da la capacidad de ampliar los problemas que se pueden resolver con este método

D. Inspiración Biológica

Las redes neuronales tienen su inspiración biológica en las células neuronales, ya que estas están conectadas entre sí, estas neuronas tienen un núcleo que procesa la información y tiene dendritas que son las responsables de conectar con otras neuronas.

Si hacemos la comparación con las regresiones logísticas, estas dendritas hacen referencia a los inputs y el núcleo hace referencia a lo que tenemos en la función lineal, y luego se decide si se deja pasar esa información.

E. Funciones de activación

En la regresión logística lo llamamos función no lineal (sigmoid)

Depende de la señal activa o no la neurona, y deja pasar la información y la transforma o bloquea



Fig. 7. Funciones de activación

Función RELU: Lo que se tiene aquí es que cualquier valor que venga de la función $f(x)=wx+b$, si es menor a 0, entonces se va a encontrar en el eje negativo de x y su resultado va a ser cero también, si es el resultado de la función es mayor a cero, entonces su resultado va a ser el mismo resultado que haya salido.

De las función de RELU hay muchas variantes, por ejemplo en la **función Leaky RELU** lo que hace es que agrega una pendiente a la sección negativa para que no siempre sea cero, si no que le agrega una pendiente constante.

Función TANH esta función lo que hace es que está acotada entre valores de $]-1,1[$, tiene una forma similar a la sigmoid, pero su rango de salida es distinto.

Función Binary Step Function lo que hace es que si el valor es mayor a cero, entonces su resultado es 1, y si es menor a 0, entonces su resultado es cero.

Función lineal que es la que hemos venido trabajando, que básicamente es dejarla como está.

Función SELU y ELU son de la misma familia, son funciones muy costosas, pero su funcionamiento es muy óptimo.

Función Sigmoid que también es la que se ha venido trabajando por varias clases atrás.

Y por último se tiene la **función Parametric RELU** que a diferencia de la función de Leaky RELU es que el parámetro del alfa que está multiplicando al resultado de la función es aprendible, es decir que se convierte en un valor más de la red neuronal, entonces se ajusta dependiendo de las necesidades.

F. Perceptrón multicapa (MLP)

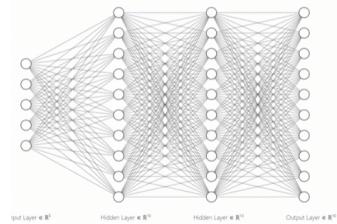


Fig. 8. Perceptrón

El Perceptrón Multicapa (MLP) es una evolución importante del perceptrón simple que permite resolver problemas no linealmente separables.

Características principales:

- A diferencia del perceptrón simple, el MLP está compuesto por múltiples capas de neuronas, lo que le permite resolver problemas más complejos.

Estructura:

- Está formado por tres tipos principales de capas: - Capa de entrada - Capas intermedias (también llamadas capas ocultas) - Capa de salida

Funcionamiento:

- La capa de salida se calcula basándose en las interacciones de las capas anteriores, donde cada capa intermedia proporciona no-linealidad al sistema.

Flexibilidad:

- La capacidad del sistema puede ajustarse modificando: - El número de capas - El tamaño de las capas - Las funciones de activación utilizadas

Un aspecto importante a considerar es que las capas deben ser distintas entre sí para poder aplicar el descenso del gradiente y optimizar los parámetros correctamente.

La propagación del error es un concepto fundamental en el MLP, donde se propagan las derivadas por cada una de las

neuronas y se realiza una sumatoria ponderada para determinar la contribución a la función de pérdida.

La imagen se interpreta de la siguiente manera, en el input Layer es la capa de entrada, donde cada X_i representa un vector, y esto dinámicamente no va a cambiar porque si cambian pierdo todos los pesos y eso no nos funciona. El Hidden Layer es la capa oculta, es decir son las capas que computan o realizan operaciones y cálculos, pero no son las finales. Por último está la capa de salida que me va a indicar a mí cuál es el tamaño de la salida que se está buscando, y va a cambiar dependiendo del problema que se esté trabajando.

¿Cómo se calcula la pasada?

Se tiene la siguiente fórmula $h(0) = \text{sigmoid}(XW^0 + b^0)$, donde $h(0)$ es la capa oculta y $\text{sigmoid}(XW^0 + b^0)$ son los features, primero debo hacer es computar o calcular la regresión lineal, luego se calcula la activación con ese resultado, y con eso se tiene ya calculado el primer Hidden Layer.

Luego para el siguiente Hidden Layer se realiza un proceso muy similar, se tiene $h(1) = \text{sigmoid}(h(0)W^1 + b^1)$, es decir se tiene que dentro de la función sigmoid voy a usar el resultado de $h(0)$, entonces se debe calcular una para que sea parámetro de la que sigue.

El proceso anterior se repite hasta llegar a $h(n) = \text{sigmoid}(h(n-1)W^n + b^n)$

G. Función de perdida

Cómo valoramos que nuestra red neuronal funciona bien o funciona mal, para esto siempre necesitamos encontrar una de perdida.

Para calcular esta función de costo debemos optimizar los parámetros de W y b de cada una de las neuronas, esto lo hacemos con la función de L , es de decir, aplicamos el descenso del gradiente, se calculan las derivadas parciales

$$\frac{\partial L}{\partial w_i^j}$$

$j = \text{capa}$

$i = \text{neurona}$

Fig. 9. función de costo

H. Maldición de dimensionalidad

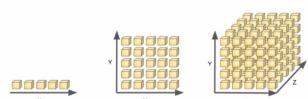


Fig. 10. maldición de la dimensionalidad

Conforme se va aumentando la dimensión se vuelve más difícil encontrar una solución. Aumenta la computabilidad, lo que hace que también se vuelva más complicado de encontrar patrones.

I. Comportamiento Jerárquico

Los humanos aprendemos cosas simples para despuésirlas transformando en algo más complejo. Las redes neuronales funcionan igual empiezan aprendiendo las primeras capas con conceptos simples para después tratar de completar la tarea para la cual ha sido especializada.