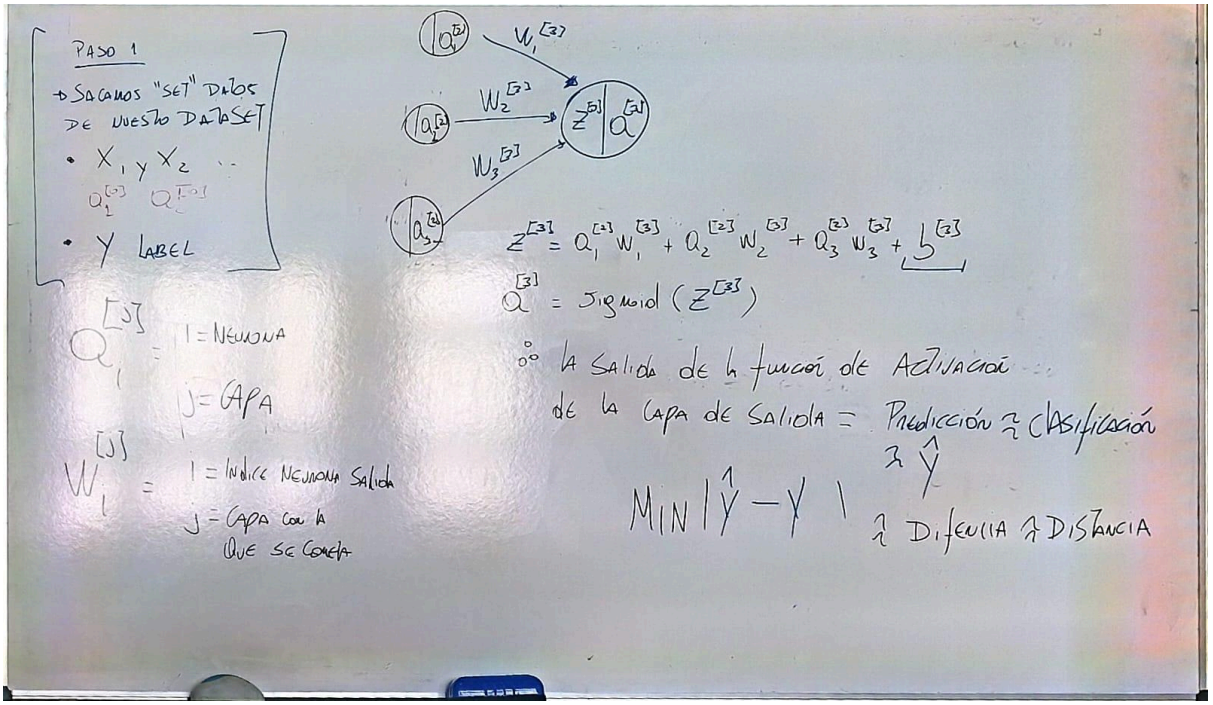
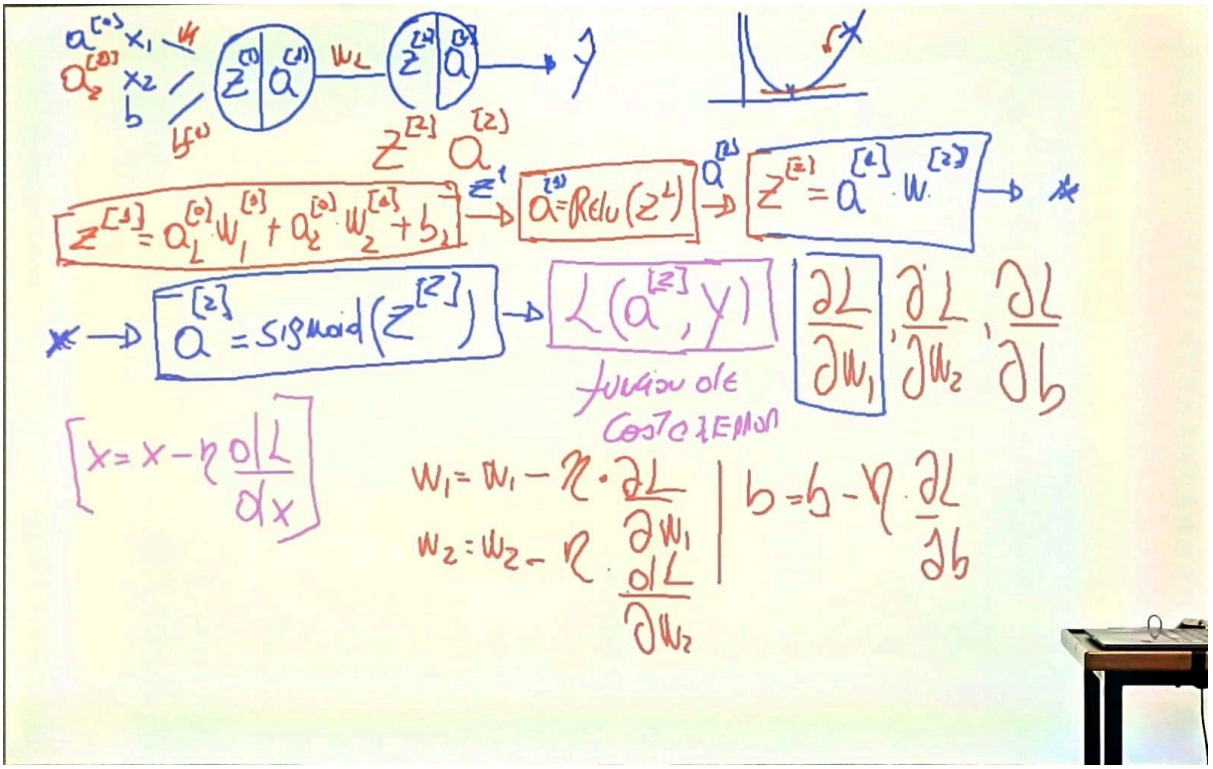
Resumen IA PP 2 ..... **2**
Suma ponderada y funcion de activacion 2
Optimización Primeros pesos Red Neuronal 3
¿Qué es Deep Learning? 3
Deep Learning y el Conjunto de Datos 4
En qué consiste el Dropout o Poda neuronal 4
Porque resurgen las RNA 5
Threshold Logic Unit TLU 5
Funciones de Activación 5
Componentes de una RNA Profunda 6
Regla de la cadena 6
Perceptron y MLP (Perceptrón Multicapa) 6
RELU 6
Ventajas 6
Problemas potenciales 6
Feedforward 6
Back Propagation 7
¿Cómo funciona? 7
Forward Propagation 8
CrossEntrpy 8
End to End Learning 8
Norma L2 9
¿Qué es? 9
Importancia 9
Como se aplica 9
Representation Learning 9
Modelo Secuencial 10
RNN Red Neuronal Recurrente 10
Long Short-Term Memory LSTM 10

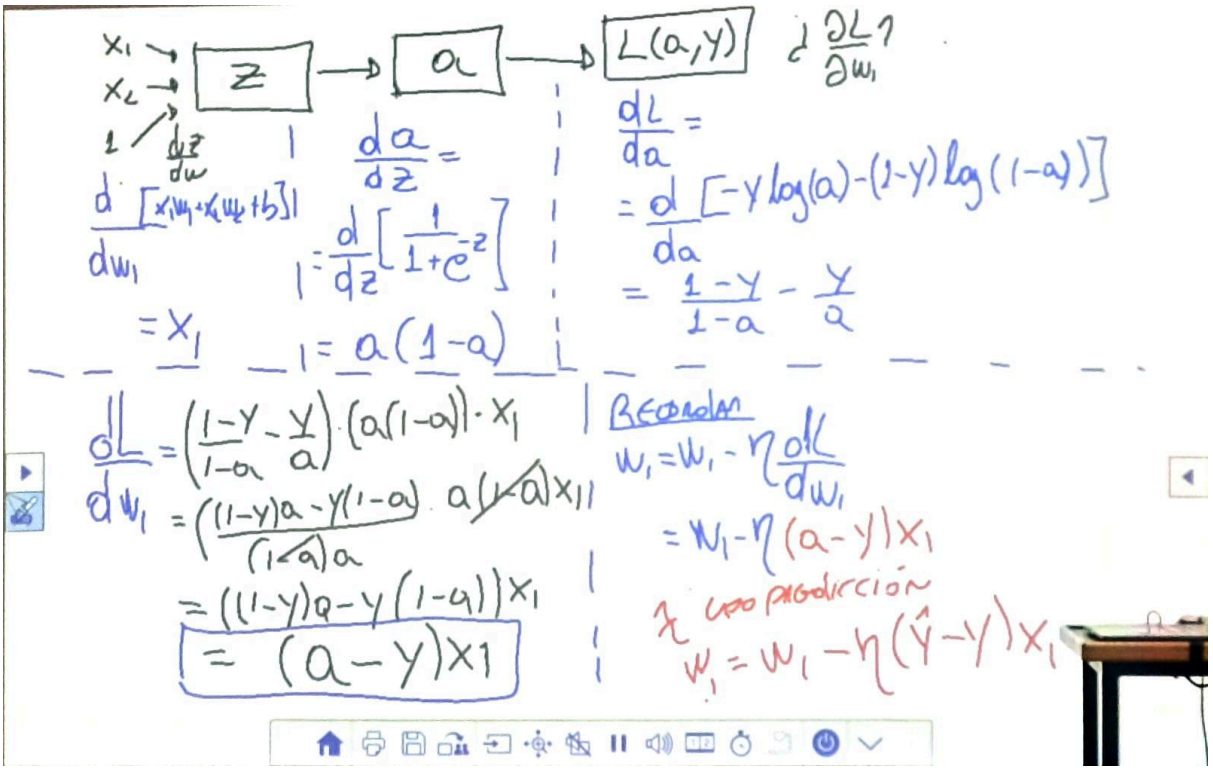
Resumen IA PP 2

Suma ponderada y funcion de activacion



Optimización Primeros pesos Red Neuronal





¿Qué es Deep Learning?

- Es un subdominio del Machine Learning que resuelve el problema de las técnicas de **Representation Learning** introduciendo

- Antes era de caja negra, pero dejo de serlo por la **Interpretabilidad y Explicabilidad**
- **N:** depende del numero de clases que vaya a predecir
- Necesito que a' = softwax -> [0,1] ~ Probabilidad
    - x % Clase 1
    - x % Clase 2
- En las capas de Deep Learning:
    - En la primera estan las entradas de caractersticas
    - Luego vienen las capas ocultas, donde las n cantidad de capas pueden tener distinta cantidad de neuronas
        - Ej: Capa 2 tiene 5 y capa 3 tiene 15

Deep Learning y el Conjunto de Datos

- Para que los algoritmos de DL funcionen adecuadamente se requiere un conjunto de datos grande. Los cuales deben tener:
    - **Representatividad**
        - Si el conjunto de datos no refleja adecuadamente las características de lo "real", los modelos de DL pueden aprender patrones incorrectos o sesgados.
    - **Generalización**
        - Los algoritmos de DL buscan aprender patrones que se apliquen a datos no vistos previamente. La selección de un mal conjunto de datos puede llevar a un overfitting
    - **Reducción de ruido**
        - Si el Dataset es muy grande, ayuda a mitigar este problema al proporcionar suficientes ejemplos para que el modelo aprenda los patrones subyacentes a pesar del ruido

En qué consiste el Dropout o Poda neuronal

- Es una técnica de DL para mejorar la generalización de los modelos de Redes neuronales
- Dependiendo del número del dropout, de activa un x número de neuronas, osea por ejemplo, de 10 neuronas, usando un dropout de 0.7, entonces solo se activan 3 neuronas.
- Las capas de dropout no tienen parámetros, ya que no son parámetros en si, sino que son operaciones.
- Dropout impacta en W , es que en la multiplicación de la suma ponderada, habrá un W*a*0
- **¿Qué es el Dropout?**
    - Técnica de regularización que se aplica durante el entrenamiento de una red neuronal.
    - Este desactiva aleatoriamente un porcentaje de las neuronas en cada capa oculta
    - Durante la inferencia (cuando se realizan predicciones) todas las neuronas están activas

- **Beneficios:**
    - Regularización: reduce la complejidad del modelo y evita el sobreajuste
    - Ensamble de modelos: Cada iteracion de entrenamiento crea un Submodelo, al combinarlos, se obtiene un modelo que generaliza mejor
- **Implementación**
    - from keras.layers import Dropout
    - model.add(Dense(64, activation='relu'))
    - model.add(Dropout(0.5)) # Desactiva el 50% de las neuronas

Porque resurgen las RNA

- Classification
    -
- Semantic Segmentation
    - Lo primero que hace es identificar el background
    - U.NET fue la primera red generativa sin necesidad de ser generativa
- Object Detection
- Inference Segmentation

Threshold Logic Unit TLU

- Forma básica de modelo de aprendizaje automático que consta de una sola entrada (y pesos correspondientes), y una función de activación.
- Cada entrada **X** está asociada a un peso **W**, en la cual la suma de las entradas ponderadas es X_i * W_i
- Compara la activación de entrada con un valor de umbral, si este es excedido, se genera una salida de **1,** sino, genera una salida de **0.**

Funciones de Activación

- **Sigmoidal:** toma un valor de entrada y lo define entre 0 u 1 $f(x) = \frac{1}{1 + e^{-x}}$
- **Unidad Lineal Rectificada RELU:** es una función de activación que se define como la parte positiva de su argumento.
    - Si entrada es positiva, la salida es igual a la entrada
    - Si es negativa o cero, la salida es cero
    - $f(x) = max(0, x)$
- **Tangente Hiperbólica TanH:** Toma un valor de entrada lo define entre -1 y 1, similar a la sigmoidal, pero más útil para situaciones en las que los valores negativos pueden ser útiles.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Componentes de una RNA Profunda

- Optimizer se refiere a mi función de Optimización
- Loss se refiere a mi función de error

Regla de la cadena

- Crucial para calcular gradientes durante backpropagation
- Para ajustar los pesos de los modelos, se necesita calcular derivadas parciales de la función de pérdida con respecto a los pesos por cada capa.
    - Esta regla permite descomponer estas derivadas parciales en términos de las derivadas de las funciones de activación

Perceptron y MLP (Perceptrón Multicapa)

- Las neuronas deben ser Fully Connected
- El proceso va hacia la derecha, a esto se le llama FeedForward Network `~ Secuencial
- **El producto del entrenamiento es un modelo**

RELU

- Si la entrada (x) es positiva, la función ReLU devuelve el mismo valor de entrada: (f(x) = x).
- Si la entrada (x) es negativa o cero, la función ReLU devuelve cero: (f(x) = 0).

Ventajas

- **Propagación de gradientes mejorada** Menos problemas de desvanecimiento de gradiente en comparación con funciones de activación s**igmoideas** que saturan en ambas direcciones
- **Cálculo eficiente:** Solo requiere comparaciones, sumas y multiplicaciones

Problemas potenciales

- **No Diferenciable en Cero:** aunque no lo es en 0, lo es en cualquier otro punto. Y el valor de la derivada en 0 puede definirse como **0** o **1**
- **No Entrada en Cero:** Las salidas RELU son siempre no negativas, lo que puede dificultar el aprendizaje durante la retropropagación.

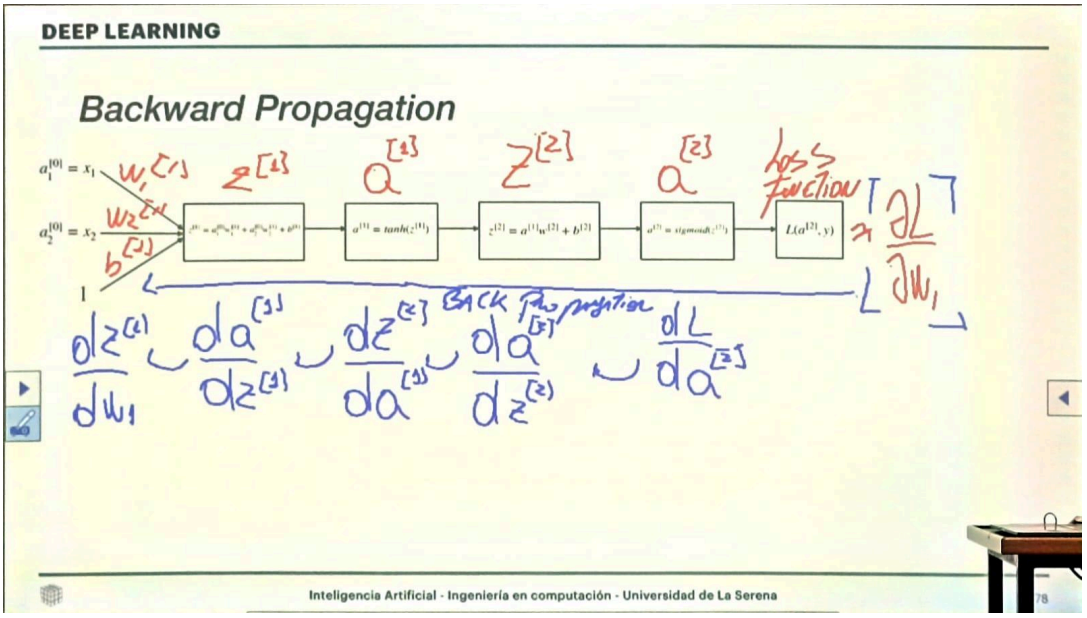
Feedforward

- A menudo denominadas Redes Multicapa.
- Se denominan Feedforward ya que toda la información fluye solo hacia adelante.
    - Los datos ingresan a los nodos de entrada.
    - Luego viajan a través de las capas ocultas
    - Eventualmente salen de los nodos de salida

- **Carece de enlaces que permitan que la información que sale del nodo de salida se envíe de vuelta a la red.**
- se utilizan para aproximar funciones y son fundamentales en aplicaciones como la detección de objetos en fotos (como se ve en Google Photos)

Back Propagation

- Proceso de propagar la pérdida total hacia atrás para cuanto contribuye cada neurona a la pérdida
- Ayuda a comprender el impacto de cada neurona en general, lo que nos permite ajustar los parámetros del modelo en consecuencia.
- Es importante porque nos permite atribuir el error a nodos (neuronas) en específico. Estos recibirán actualizaciones más significativas durante el ajuste de los pesos

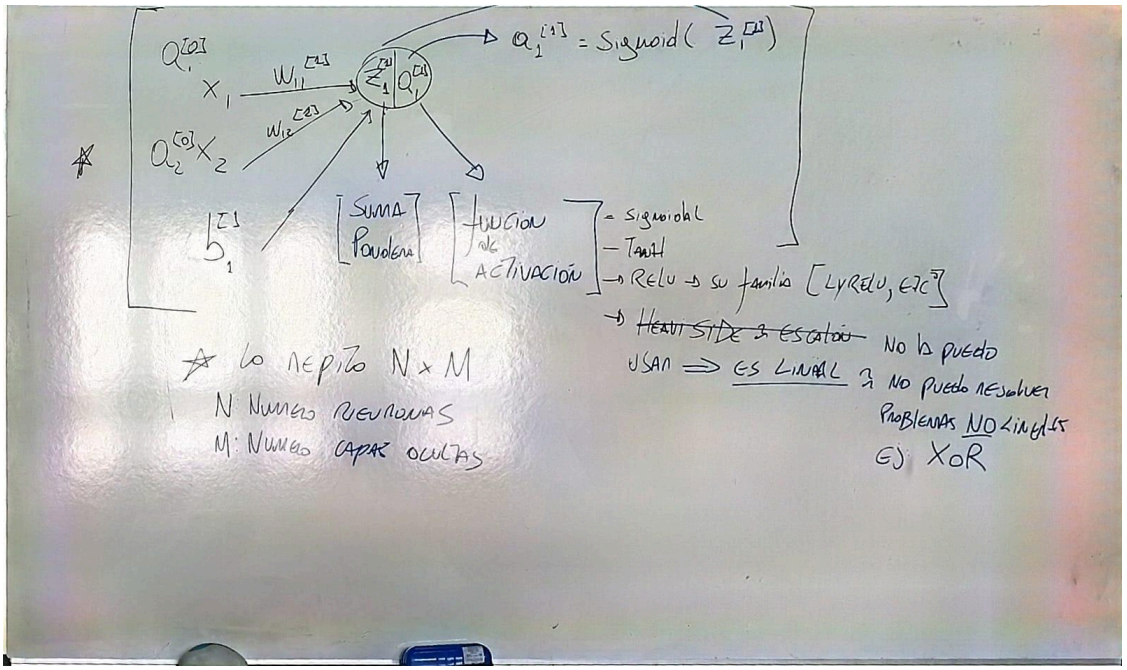


¿Cómo funciona?

1. **Paso hacia adelante (Forward Pass)**: Durante el paso hacia adelante, los datos de entrada fluyen a través de la red capa por capa, produciendo predicciones.
2. **Cálculo de la pérdida**: Comparamos la salida predicha con el objetivo real (salida esperada) utilizando una **función de pérdida** (como el error cuadrático medio o la entropía cruzada).
3. **Paso hacia atrás (Backward Pass o Backpropagation)**:
   - Calculamos el **gradiente** de la pérdida con respecto a cada peso y sesgo en la red.
   - El gradiente indica cuánto cambia la pérdida cuando ajustamos un peso o sesgo particular.
   - Utilizamos la **regla de la cadena** para calcular estos gradientes de manera eficiente.
4. **Actualización de pesos**: Finalmente, ajustamos los pesos y sesgos utilizando un algoritmo de optimización (como el descenso de gradiente) para minimizar la pérdida.

Forward Propagation

- Tiene que ver con el proceso de la preparación de una predicción para comparar contra la verdad.
    - Se compara, se optimiza y realiza Back Propagation
- Cuando ya esta entrenado el modelo, se llama clasificación o inferencia
- Consiste en el cálculo y la transmisión de los datos de entrada a través de las capas de la red para generar una salida
- Tenemos una red neuronal simple, luego de definirla, pasamos al **Forward Pass**, donde se calcula la salida de la red neuronal utilizando los valores actuales de los pesos, finalmente se ocupa una función de activación, ya sea sigmoidal, tanh, relu, etc.



CrossEntrpy

- Función de costo utilizada para medir un modelo de clasificación.
- La entropía cruzada evalúa cuánto se desvía la distribución (q) de la distribución (p).
- La fórmula para la entropía cruzada depende de si estamos tratando con distribuciones discretas o continuas, pero en ambos casos, su objetivo es medir la discrepancia entre las distribuciones.

End to End Learning

- Este es el concepto de Deep Learning, osea que yo le presento una entrada y tengo una salida.
- La unica manera en la que puedo influir en end to end learning es con los hiperparametros

Norma L2

- Son hiperparametros

¿Qué es?

- Es una forma de regularización que se utiliza para controlar la complejidad de los modelos y evitar el sobreajuste
- Se aplica agregando un término adicional a la función de pérdida durante el entrenamiento del modelo

Importancia

- **Reducción del sobreajuste:** Penaliza los valores grandes de los parámetros del modelo. Esto ayuda a evitar que los coeficientes se vuelvan demasiado grandes.
- **Estabilidad numérica:** Mejora la escalabilidad numérica durante el entrenamiento al evitar que los valores de los parámetros se disparen hacia infinito.

Como se aplica

- Se agrega como un término de penalización en la función de pérdida

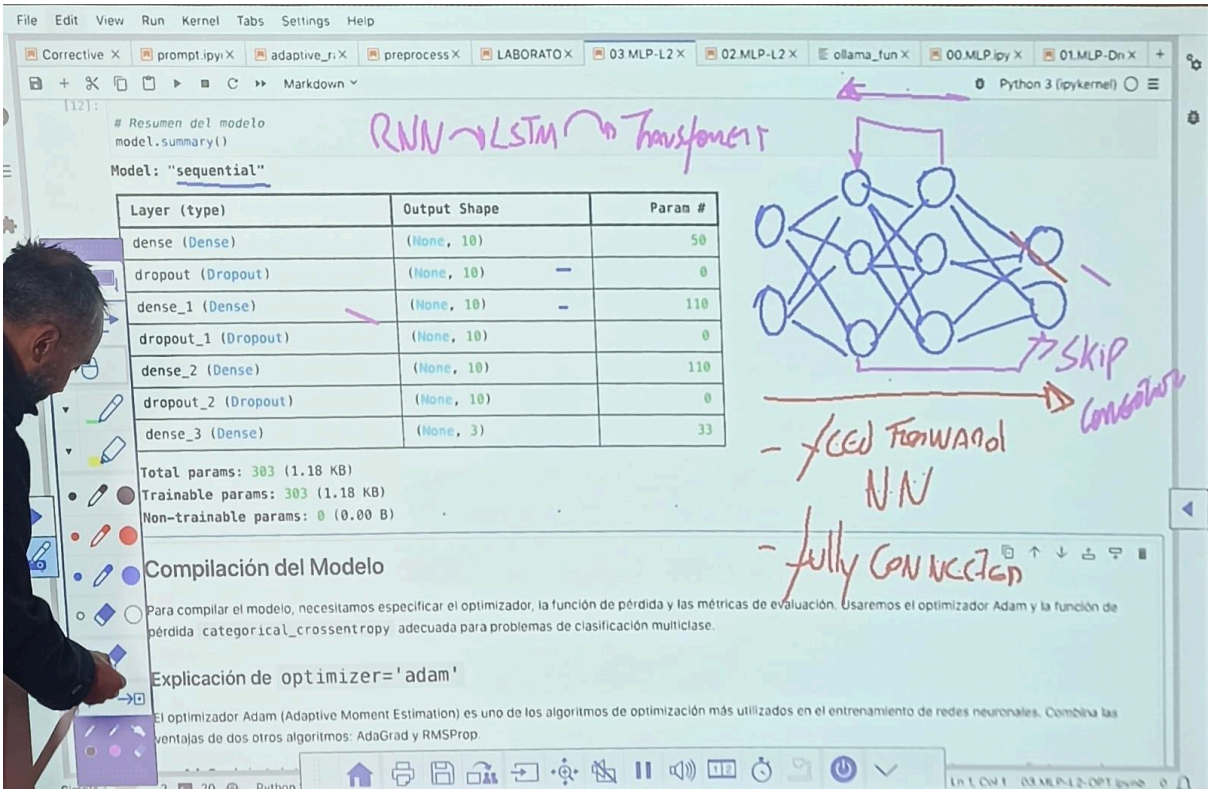
$$L(w) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda\|w\|_2^2$$

- **L(W):** Funcion de perdida
- **N:** numero de muestras
- **y_i:** etiqueta real
- **y_i sombrero:** predicción
- **W:** pesos del modelo
- **Lambda:** hiper parámetro de regularización

Representation Learning

- Solución a este problema es utilizar ML para descubrir la mejor forma de describir los datos

Modelo Secuencial



- Diseñadas específicamente para trabajar con datos secuenciales, como señales de audio.

RNN Red Neuronal Recurrente

- Base de muchos modelos secuenciales. Tienen conexiones recurrentes que les permiten mantener una especie de “memoria” a lo largo de las secuencias.
- Tienen problemas de gradiente

Long Short-Term Memory LSTM

- Son una mejora de las RNN. Resuelven problemas de gradiente y memoria a largo plazo