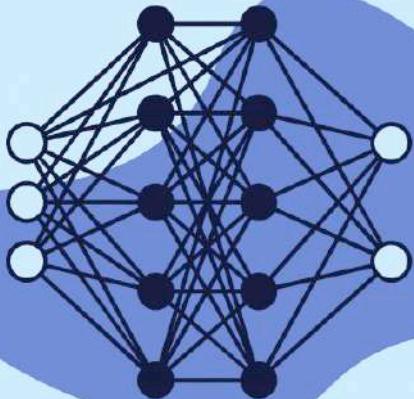


 BEST COURSES

Deep Learning

 class central



DEEP LEARNING

EL OBJETIVO PRINCIPAL ES DEFINIR CORRECTAMENTE EL PROBLEMA DE APRENDIZAJE A RESOLVER.

ELEMENTOS DEL PROBLEMA:

- CONJUNTO DE DATOS DISPONIBLES $D = \{(x_i, y_i)\}_{i=1}^N$
- FUNCIÓN DESCONOCIDA A APRENDER y
- ENTRENAMIENTO $f(x) \rightarrow y$
- FUNCIÓN DE APROXIMACIÓN GENERALIZADA $\hat{f}(x) \rightarrow f(x)$
 $\hat{f}(x)$ ES UNA FUNCIÓN QUE USA EL CONJUNTO DE DATOS D PARA APROXIMAR $f(x)$ Y GENERALIZAR EL ERROR AL USAR DATOS NUEVOS QUE NO ESTABAN EN EL ENTRENAMIENTO.

Modelos de Aprendizaje Básicos: INTERPOLACIÓN

USAR LOS DATOS QUE CONOCENOS PARA ENCONTRAR UNA FUNCIÓN QUE LOS DESCRIBA Y NOS DÉ UNA APROXIMACIÓN DE CÓMO SE PODRÍAN COMPORTAR DATOS NUEVOS.

EJEMPLO

	V_1	V_2	V_3	V_4	f_1	f_2	f_3	POSIBLES SALIDAS
x_1	0	0	0	0	1	1	1	
x_2	0	0	0	1	0	0	1	
x_3	0	0	1	0	0	0	1	
x_4	0	0	1	1	1	0	1	
⋮								
x_n	1	1	1	1	1	1	1	

2⁴ POSIBLES SALIDAS

4 VARIABLES CON 2 POSIBLES VALORES

EXISTEN 2⁴ FUNCIONES QUE AJUSTAN LOS DATOS

$$|D| = 2^4 \text{ CARNAVIDAD EXPONENCIAL}$$

DADO UN CONJUNTO DE ENTRENAMIENTO $D = \{(x_i, y_i)\}_{i=1}^N$, $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$
UNA FUNCIÓN $f: \mathbb{R}^m \rightarrow \mathbb{R}$ CON ERROR 0 SERÍA CUALQUIER FUNCIÓN
QUE PASE POR TODOS LOS PUNTOS DEL CONJUNTO D .

↳ PARA CADA ENTRADA x_i SE ASIGNA UN VALOR DE y_i .

CONVERGENCIA: SE ACERCA EXPONENCIALMENTE AL VALOR REAL DE LA FUNCIÓN.

DIVERGENCIA: SE ALEJA EXPONENCIALMENTE AL VALOR REAL DE LA FUNCIÓN.

GENERALIZACIÓN.

Ej. LA CAPACIDAD QUE TIENE UNA ARQUITECTURA DE ENCONTRAR UNA FUNCIÓN $\hat{f}(x)$ QUE REALMENTE SE APROXIME A LA FUNCIÓN REAL DEL FENÓMENO, USANDO SOLO UNA MUESTRA LIMITADA DE DATOS.

- LA FUNCIÓN REAL QUE GENERA LOS DATOS DEPENDE DE LA NATURALEZA DEL PROBLEMA Y PUEDE INCLUIR SESGOS O PROPIEDADES NO ÓPTIMAS QUE EL MODELO SUPERVISADO DEBE APRENDER A RECONOCER.

POR EJEMPLO: LOS CRITERIOS QUE TIENE UN BANQUERO PARA OTORGAR O NO UN CRÉDITO PUEDEN NO SER ÓPTIMOS PERO DEBEMOS APRENDERLOS PORQUE LOS DATOS LOS ESTÁN INCLUYENDO

ACCIONES QUE USAMOS PARA ALCANZAR GENERALIZACIÓN:

1. INCLUIR CONOCIMIENTO DEL PROBLEMA PARA CONSIDERAR PROPIEDADES DE UTILIDAD QUE NOS AYUDEN A REDUCIR EL ESPACIO DE BÚSQUEDA COMO:
 - INVARIANCIAS A ROTACIÓN O TRASLACIÓN
 - IDENTIFICACIÓN DE PATRONES
 - DERIVADAS PEQUEÑAS (QUE NO HAYA CAMBIOS BRUSCOS)

ESPACIO DE BÚSQUEDA

EXISTE UN CONJUNTO DE POSIBLES FUNCIONES QUE AJUSTAN LOS DATOS DE ENTRENAMIENTO, DENTRO DE ELLAS ESTÁ LA MEJOR FUNCIÓN $\hat{f}(x)$ QUE PODEMOS ENCONTRAR PERO CÓMO LA IDENTIFICAMOS. COMO EL CONJUNTO DE DATOS QUE CONOCEMOS ES FINITO NO TENEMOS SUFFICIENTE INFORMACIÓN PARA SABER QUÉ FUNCIÓN DE APROXIMACIÓN ES LA MEJOR.

COSAS QUE GARANTIZAN GENERALIZACIÓN:

- ASUMIR QUE TODOS LOS EJEMPLOS DENTRO Y FUERA DEL ENTRENAMIENTO, TIENEN LA MISMA DISTRIBUCIÓN.
- AJUSTAR LA COMPLEJIDAD DE LA FAMILIA DE FUNCIONES.
- Elección del espacio de hipótesis H a usar:
 - FUNCIONES LINEALES
 - ÁLGEBRAS DE DECISIÓN
 - ⋮
 - LAS NN PUEDEN APROXIMAR CUALQUIER FUNCIÓN CONTINUA.
- UTILIZAR EL CONOCIMIENTO DE LA FUNCIÓN OBJETIVO PARA ACOTARLA.
- USAR DATOS DE PRUEBA Y DE VALIDACIÓN PARA ENTRENAR Y USAR UN CONJUNTO TEST QUE NUNCA INFLUYA EN EL AJUSTE DEL MODELO.

AJUSTAR LOS HIPERPARÁMETROS DEL MODELO
SOLO SIRVE SI ESTÁN MINIMIZANDO EL ERRORE
DURANTE LA VALIDACIÓN Y NO SOLO EN
EL ENTRENAMIENTO.

GENERALIZACIÓN VS MEMORIZACIÓN

- Entre más grande es la familia de funciones que usamos para entrenar, más datos necesitamos para acotar la diferencia entre los errores fuera y dentro del entrenamiento.
- Lo mismo sucede para el conjunto de validación que usamos para elegir los hiperparámetros, tampoco puede ser tan pequeño.
- La distribución del conjunto de entrenamiento y la distribución de los datos reales son iguales.

MEMORIZAR:

ENCONTRAR UNA FUNCIÓN EN LA FAMILIA DE FUNCIONES QUE SE AJUSTA PERFECTAMENTE A LOS DATOS DE ENTRENAMIENTO SIN GENERALIZAR.

• CONDICIONES PARA GARANTIZAR GENERALIZACIÓN

→ APRENDIZAJE DE MÁQUINA ESTADÍSTICO (PROBABILIDAD AL RESCATE)

• TEORÍA DE MUESTREO

- TOMAR UNA FUNCIÓN LO SUFFICIENTEMENTE REPRESENTATIVA: "i.i.d."

- USAR LA CONCENTRACIÓN DE PROBABILIDAD ALREDEDOR DE LA MEDIA μ .

→ LOS DATOS PARA ENTRENAR Y PARA PRONOSTICAR DEBEN TENER LA MISMA DISTRIBUCIÓN DE PROBABILIDAD.

VARIANZA: DISPERSIÓN DE UN CONJUNTO DE DATOS, CUÁNTO SE ALEGAN LOS VALORES DE LA MEDIA.

$$\sum_{i=1}^n \frac{(x_i - \bar{x})^2}{n} \quad \bar{x} = \text{MEDIA}$$

MEDIA: VALOR TÍPICO DE UN CONJUNTO DE DATOS.

$$\sum_{i=1}^n \frac{x_i}{n} \quad (\text{VALOR ESPERADO})$$

VARIANZA: QUÉ TANTO SE ALEJAN LOS VALORES DE UNA VARIABLE ALÉATORIA DE SU MEDIA.

LEY DE LOS GRANDES NÚMEROS

$$\frac{p(1-p)}{n}$$

ENTRE MÁS GRANDE ES n , LA VARIANZA SE CONCENTRA MÁS EN SU MEDIA.

PROBABILIDAD DE QUE TODOS LOS VALORES SALGAN IGUALES



$$P(\text{AZUL}) = 0.8$$

1000

MUESTRA: $\bullet, \bullet, \bullet, \bullet, \dots, \bullet$

$x_1, x_2, x_3, \dots, x_{1000}$

$$p(x) = p(x_1 = \text{AZUL}) \cdot p(x_2 = \text{AZUL}) \cdots p(x_{1000} = \text{AZUL})$$
$$p(x) = 0.8^{1000}$$

QUEREMOS GARANTIZAR QUE EL ERROR QUE OBTENEMOS EN EL MUESTRÉO, NOS VA A DAR INFORMACIÓN DEL ERROR FUERA DEL MUESTRÉO (EN EL CONJUNTO REAL DE INTERÉS).

TAREA: VALIDAR EXPERIMENTALMENTE EL RESULTADO DE TEORÍA DE MUESTRÉO.

- EN MACHINE LEARNING INTENTAMOS APRENDER UNA DISTRIBUCIÓN DESCONOCIDA DE LOS DATOS QUE NOS DAN.

LA TEORÍA DE APRENDIZAJE DA GARANTÍAS DE QUE FUNCIONA PORQUE ASUMIMOS LA MISMA DISTRIBUCIÓN DE PROBABILIDAD DENTRO DEL CONJUNTO DE ENTRENAMIENTO Y FUERA DE ÉL.

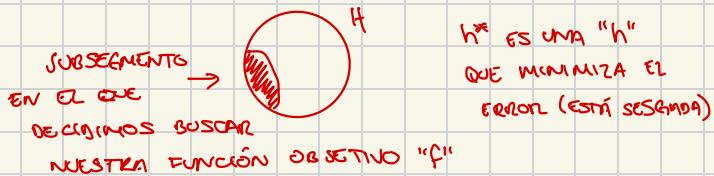
* ASUMIMOS QUE LO QUE ESTÁ FUERA DEL CONJUNTO DE ENTRENAMIENTO ES ARBITRARIO.

MACHINE LEARNING:

- FENÓMENO
- SERIE DE EXPERIMENTOS
- HIPÓTESIS DE SOLUCIÓN

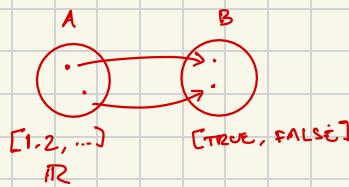
COSAS QUE ASUMIMOS:

- CONTINUIDAD
- SUS DERIVADAS NO SON TAN GRANDES
 - ASUMIMOS QUE UNA NUEVA "X" CERCA DE UNA "X" DE EJEMPLO, ESTARÁ TAMBÉN CERCA DE "y".
- INFORMACIÓN DEL FENÓMENO.
- "FISAMOS" MUCHAS PROPIEDADES DEL PROBLEMA (INVARIANTES)
- RESTRINGIMOS EL ESPACIO DE HIPÓTESIS.



PARA RESOLVER UN PROBLEMA CON MACHINE LEARNING, REQUERIMOS UN "BIAS INDUCTIVO".

MATEMÁTICAMENTE: UNA FUNCIÓN ES UN ELEMENTO "A" QUE MARCA A OTRO ELEMENTO "B"



ESTRICTAMENTE, MATEMÁTICAMENTE NO PODRIAMOS RESTRINGIR LA FUNCIÓN DESCONOCIDA PARA GARANTIZAR GENERALIZACIÓN.

Bias Inductivo.

UNA PRIMERA INTUICIÓN ES ELEGIR LA FUNCIÓN MÁS SIMPLE PERO...
SIMPLE EN TÉRMINOS DE QUÉ Y
POR QUÉ ESO FUNCIONA?

DEF.

→ DARLE INFORMACIÓN AL MODELO PARA TENER CRITERIOS
DE PREFERENCIA QUE AYUDEN A LA GENERALIZACIÓN PARA
REDUCIR EL ERROR FUERTE DE LOS DATOS DE ENTRENAMIENTO.

BIAS INDUCTIVO (SESGO):

- 1) SUPOSICIONES DEL FENÓMENO QUE HACEMOS PARA INTENTAR ACOTARLO.
- 2) QUEREMOS QUE LOS DATOS FUERA DEL CONJUNTO DE ENTRENAMIENTO SE PAREZCAN A LOS DATOS EN EL CONJUNTO QUE USAMOS PARA ENTRENAR
- 3) BUSCANDO DENTRO DEL ESPACIO DE POSIBLES FUNCIONES QUE APROXIMAN LOS DATOS "x" A LOS DATOS "y".
- 4) USAMOS LA INFORMACIÓN DEL ENTORNO DEL PROBLEMA PARA ACOTAR LAS FUNCIONES (EL ESPACIO DE FUNCIONES) DONDE BUSCAREMOS.



- CUÁLES SON LOS BIAS INDUCTIVOS QUE ASUMEN LOS TRANSFORMADORES Y POR QUÉ SON TAN BUENOS?

VARIABLES ALEATORIAS: Función DETERMINISTA (PARA CADA ENTRADA, LA FUNCIÓN PRODUCE EL MISMO RESULTADO) QUE PARA UN PUNTO DEL ESTADO MUESTRAL DE UN EXPERIMENTO ALEATORIO (P.E. LANZAR UN DADO) MAPEA (TRANSFORMA) A NÚMEROS REALES.

PROMEDIO PONDERADO DEL VALOR ESPERADO

$$\text{MEDIA: } \mathbb{E}[X] = \sum_x x P(x)$$

■ SI UN EXPERIMENTO ES COMPLETAMENTE AL AZAR,
LA PROBABILIDAD MÁS ALTA ES LA MEDIA.

$$[\text{COUTEO CON COEFICIENTE BINOMIAL}] \quad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

COMBINACIONES SIN
TENER CUENTA ORDEN.

Teoría de Aprendizaje Estadístico

Teoría de Muestras:

Por qué tomar una muestra i.i.d. de un conjunto de datos garantiza una buena aproximación al universo de datos.

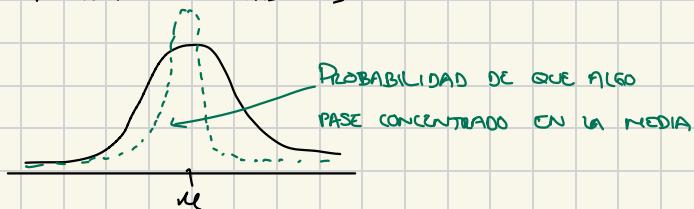
- Resultado: La probabilidad de la desviación de la función de aproximación a la función real cae exponencialmente conforme los datos de muestra aumentan.

$$P[\hat{p} - p | \epsilon] \leq 2e^{-\epsilon^2 N}$$

Desviación entre el valor de p en

el conjunto completo y \hat{p} en el
conjunto de muestra.

↳ Concentración de Probabilidad de una variable aleatoria

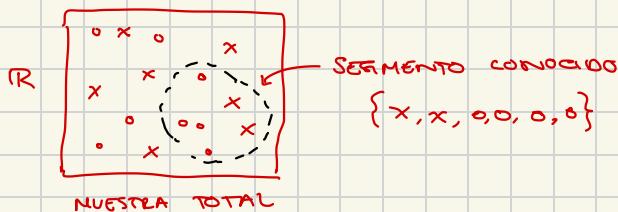


MACHINE LEARNING ESTADÍSTICO.

- DATASETS SUFFICIENTEMENTE GRANDES.
- COMPLICIDAD DE LA FUNCIÓN OBJETIVO.
- FLEXIBILIDAD DE LA FUNCIÓN OBJETIVO.

HEURÍSTICA: CONENZAR CON LOS MODELOS MÁS SENCILLOS.

MUESTREO:



CONDICIONES DE ALEATORIIDAD FUERTES:

MUESTREOS i → INDEPENDIENTE E

i → IDENTICAMENTE

d. → DISTRIBUIDO

↳ LAS MUESTRAS TIENEN LAS MISMAS
CARACTERÍSTICAS ESTADÍSTICAS (MEDIA, VARIANZA, ETC...)

- SELECCIÓN DE MUESTRAS ALEATORIAS.

INDEPENDENCIA:

- SI LOS DATOS TIENEN DEPENDENCIAS, LOS MODELOS PODRÍAN APRENDER RELACIONES ESPECÍFICAS QUE NO APLIQUEN A DATOS FUERA DEL ENTRENAMIENTO.

DISTRIBUCIÓN IDÉNTICA:

- ↳ ASEGURO QUE EL CONJUNTO DE ENTRENAMIENTO Y PRUEBA REFLEJAN EL MISMO FENÓMENO.
- ↳ GARANTIZAMOS QUE EL COMPORTAMIENTO APRENDIDO DURANTE EL ENTRENAMIENTO SERÁ APLICABLE A LOS DATOS NUEVOS.

PROBLEMA DE APRENDIZAJE ESTADÍSTICO

- APRENDER PATRONES A PARTIR DE DATOS Y HACER PREDICCIONES SOBRE NUEVOS DATOS.

→ SEA X UNA VARIABLE ALEATORIA QUE REPRESENTA UN VECTOR DE CARACTERÍSTICAS

→ SEA Y UNA VARIABLE ALEATORIA QUE REPRESENTA LA ETIQUETA A PREDICIR.

- SE ASUME QUE (X, Y) TIENEN UNA DISTRIBUCIÓN CONJUNTA DESCONOCIDA $P(X, Y)$

- EL OBJETIVO ES APRENDER UNA FUNCIÓN $f(X)$ TAL QUE, PARA UN CONJUNTO $X = x$, EL VALOR DE SALIDA ESTIMADO $\hat{y} = f(x) \approx y$ VERDADERA.

→ EL OBJETIVO ES ENCONTRAR UNA FUNCIÓN DE APROXIMACIÓN \hat{y} QUE MINIMICE EL ERROR DE PREDICCIÓN

$$E(\hat{y}) = \underset{(x,y) \sim P}{\mathbb{E}} [l(y, \hat{y}(x))]$$

$l(y, \hat{y}(x)) \rightarrow$ FUNCIÓN DE PÉRDIDA QUE CUANTIFICA LA DIFERENCIA ENTRE LA PREDICCIÓN \hat{y} Y EL VALOR REAL.

Sobre un conjunto de datos muestra:

$$E(\hat{y}) = \frac{1}{n} \sum_{i=1}^n l[(y_i, \hat{y}(x_i))]$$

→ SE DEFINE UN ESPACIO DE HIPÓTESIS H QUE REPRESENTA EL CONJUNTO DE POSIBLES FUNCIONES \hat{y} . SE ELIGE EN FUNCIÓN DE LA COMPLEJIDAD DSEADA Y LAS RESTRICCIONES DEL MODELO.

GARANTÍAS DE CONVERGENCIA.

TEOREMA DE CONVERGENCIA EN PROBABILIDAD.

- ESTABLECE QUE A MEDIDA QUE AUMENTAMOS EL TAMAÑO DE LA MUESTRA, EL MODELO ESTIMADOR CONVERGE AL VALOR VERDADERO DEL FENÓMENO (O VALOR ESPERADO) CON ALTA PROBABILIDAD.

↳ PROPIEDAD QUE PERMITE JUSTIFICAR EN ML QUE BAJO CERTAS CIRCUNSTANCIAS, LOS MODELOS PUEDEN GENERALIZAR BIEN A PARTIR DE LOS DATOS DE ENTRENAMIENTO.

DEF.

$$\lim_{n \rightarrow \infty} P(|X_n - c| \geq \epsilon) = 0$$

LA PROBABILIDAD DE QUE UNA VARIABLE ALEATORIA X_n SE DESVIÉ DE UNA CONSTANTE c POR MÁS DE UN VALOR ϵ , ES IGUAL A CERO A MEDIDA QUE EL TAMAÑO DE LA MUESTRA n CRECE.

INTERPRETACIÓN EN ML: UN ESTIMADOR SE APROXIMA AL VERDADERO COMPORTAMIENTO DE LOS DATOS, A MEDIDA QUE ESTOS AUMENTAN. LA FUNCIÓN DE PÉRDIDA PROMEDIO SOBRE EL CONJUNTO DE ENTRENAMIENTO, CONVERGE EN PROBABILIDAD A SU VALOR ESPERADO VERDADERO.

TEOREMA DE LÍMITE CENTRAL.

• BAJO CIERTAS CONDICIONES, LA SUMA O EL PROMEDIO DE UN GRAN NÚMERO DE VARIABLES ALEATORIAS i.i.d TIENDE A SEGUIR UNA DISTRIBUCIÓN NORMAL, INDEPENDIENTEMENTE DE LA DISTRIBUCIÓN ORIGINAL DE ESAS VARIABLES.

↳ PARA UN NÚMERO SUFFICIENTEMENTE GRANDE DE VARIABLES n , LA DISTRIBUCIÓN DE LA MEDIA MUESTRAL \bar{X}_n SE APROXIMA A UNA DISTRIBUCIÓN NORMAL ESTANDAR (DONDE LA PROBABILIDAD MÁS ALTA SE ENCUENTRA EN LA MEDIA μ).

Uso: OBTENER UNA DISTRIBUCIÓN NORMAL CUANDO EL NÚMERO DE VARIABLES ES GRANDE NOS PERMITE HACER INFERNCIAS PROBABILÍSTICAS SOBRE LA MEDIA DE UN CONJUNTO DE DATOS, INCLUSO SI LA DISTRIBUCIÓN ORIGINAL DE LOS DATOS NO ES NORMAL.

PERMITE APROXIMAR DISTRIBUCIONES COMPLEJAS A UNA DISTRIBUCIÓN NORMAL BAJO CIERTAS CONDICIONES

ML: DISTRIBUCIÓN DEL ERROR DE PREDICCIÓN. → LA SUMA O PROMEDIO DEL ERROR DE PREDICCIÓN EN MUCHOS EJEMPLOS INDEPENDIENTES TIENDE A SERVIR UNA DISTRIBUCIÓN NORMAL.

LEY DE LOS GRANDES NÚMEROS

- CONFORME AUMENTAMOS EL TAMAÑO DE LOS EXPERIMENTOS, LA VARIANZA DEL ESTIMADOR DE LA MEDIA SE COMIENZA A VOLVER CERO.

$$\text{Var}(\hat{\mu}) = \frac{\sigma^2}{n}$$

FUNCIÓN GENERADORA DE MOMENTOS = INFORMACIÓN DE UNA VARIABLE ALEATORIA.

LA MEDIA MUESTRAL $\hat{\mu}$ SE APROXIMA A LA VERDADERA MEDIA μ DE UNA VARIABLE ALEATORIA X_n CON ALTA PROBABILIDAD CUANDO LA MUESTRA AUMENTA.

DESIGUALDAD DE HOEFFDING:

$$\frac{1}{N} \sum_{i=1}^N l(h(x_i), y_i)$$

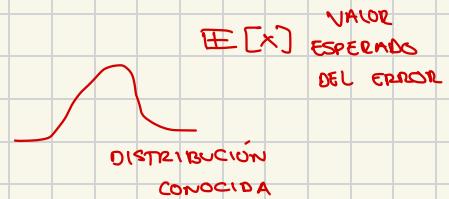
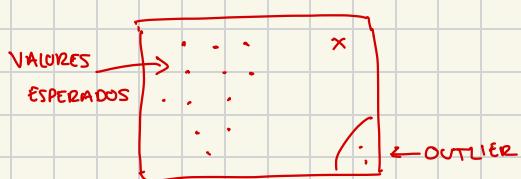
PROMEDIO

← FUNCIÓN DE PÉRDIDA

← ERROR EN LA HIPÓTESIS

DIFERENCIA ENTRE EL ERROR DENTRO DE LOS DATOS DE ENTRENAMIENTO Y LOS DATOS NO CONOCIDOS, ASUMIENDO QUE ESTOS POSEEN LA MISMA DISTRIBUCIÓN DE DATOS

EJEMPLOS: MISMO RANGO DE EDAD O DE ALTURA EN UNA POBLACIÓN.



DESANOS QUE EL MODELO SEA MUY BUENO (LE DÉ MÁS PESO O IMPORTANCIA) A LOS VALORES ESPERADOS AUNQUE SEA MUY MALO PARA VALORES DE OUTLIERS, PORQUE SON MENOS PROBABLES.

LA DESIGUALDAD DE Hoeffding ESTABLECE QUE LA DIFERENCIA ENTRE EL ERROR DE LA MUESTRA Y EL ERROR REAL PUEDE CONTROLARSE DE MANERA PROBABILÍSTICA.

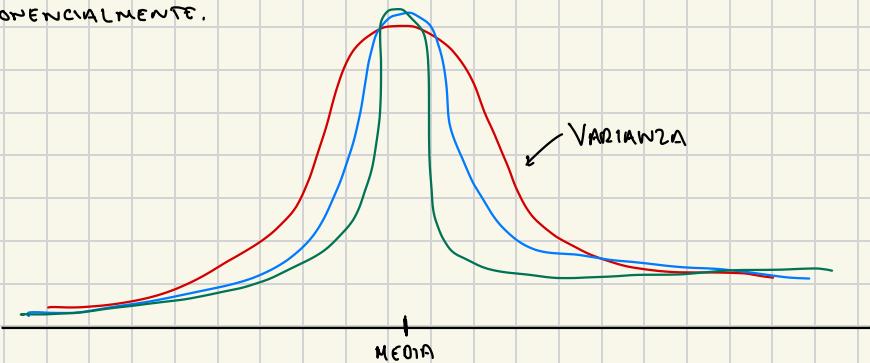
$$P[|\hat{\epsilon}_{\text{error,in}}^{(h)} - \hat{\epsilon}_{\text{error,out}}^{(h)}| > \epsilon] \leq 2e^{-2Ne^2}$$

↑
ESTIMADO EN EL ENTRENAMIENTO

↑
ERROR EN DATOS NO CONOCIDOS

↑
VALOR DE ERRORES ESPERADO

LA PROBABILIDAD DE QUE UN VALOR SE DESVIE DE SU MEDIA CAE EXPONENCIALMENTE.



FORMAS DE ELEGIR "n" VALORES DE 1000 MUESTRAS:

$$\binom{1000}{0} = 1$$

$$\binom{1000}{1000} = 1$$

$$\binom{1000}{1} = 1000$$

LA PROBABILIDAD DE QUE EL RESULTADO SE ALEJE DE LA MEDIA POR MÁS DE UN ϵ , CAE EXPONENCIALMENTE A MEDIDA QUE EL NÚMERO DE EXPERIMENTOS AUMENTA (n)

LA DESIGUALDAD DE Hoeffding nos dice para:

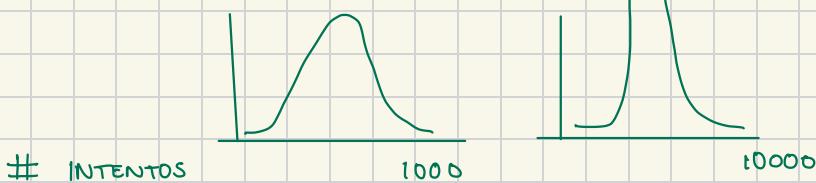
- ESTABLECER UN LÍMITE SUPERIOR DE GENERALIZACIÓN.
- COMPLEJIDAD NECESARIA DEL MODELO.
- CANTIDAD DE DATOS NECESARIOS.
- ESTABLECER UNIDADES PROBABILÍSTICAS EN LA DIFERENCIA DE LOS ERRORES IN/OUT.

CONCENTRACIÓN DE PROBABILIDAD:

- DESIGUALDAD DE Hoeffding.

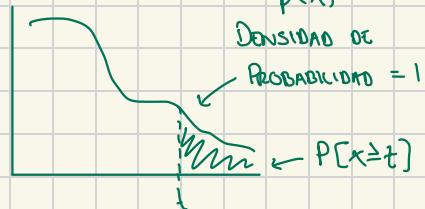
Conforme aumentamos el tamaño de N , la probabilidad comienza a hacerse determinista.

↳ Cuando echamos un dado muchas veces, el valor de la probabilidad se concentra en el valor de su media.



- DESIGUALDAD DE MARKOV PARA UNA VARIABLE ALEATORIA X .

$$P[X \geq t] \leq \frac{E[X]}{t}$$



- DESIGUALDAD DE CHEBYSHEV

$$P[|X - \mu| \geq t] \leq \frac{\text{Var}[X]}{t^2}$$



La probabilidad de caer lejos de la media se reduce cuadráticamente si conocemos la varianza de la variable aleatoria.

REINFORCEMENT LEARNING

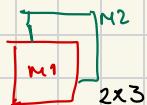
↳ LOS RESULTADOS NO SON I.I.D. PORQUE LOS DATOS TRAEN APRENDIZAJE DEL ESTADO Y DEL AGENTE EN EL TIEMPO.

LIBRERIAS PARA NL Y DL.

- 1) PyTorch
- 2) TensorFlow
- 3) JAX

PyTorch: ARREGLOS MULTIMENSIONALES

$$x \in \mathbb{R}^{2 \times 2 \times 3} \rightarrow 2 \text{ MATRICES DE } 2 \times 3$$



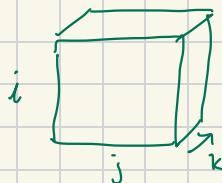
TENSOR $x \in \mathbb{R}^{2 \times 3}$:

$$[[1, 2, 3], [4, 5, 6]]$$

Dos vectores de
tamaño 3

ESTRUCTURA: $[[\cdot, \cdot, \cdot], [\cdot, \cdot, \cdot]]$

$$x = [:, :, :]$$



TENSOR 3D = CUBO DE DATOS
(IMÁGENES A COLOR)

• ES IMPORTANTE SIEMPRE QUE SEA POSIBLE, USAR LAS HERRAMIENTAS DE LAS LIBRERÍAS YA QUE CUENTAN CON CÓDIGO OPTIMIZADO.

→ UN TENSOR GENERALIZA EL CONCEPTO DE:

- ESCALARES
- VECTORES
- MATRICES

→ LOS TENSORES PUEDEN TENER MÁS DE 3 DIMENSIONES.

→ EN PYTORCH LOS TENSORES ALMACENAN EL MISMO TIPO DE DATOS.

LOS TENSORES SE PUEDEN COMPUTAR EN "CPU" O "GPU" (CON CUDA)

PROFILERS: MEDIDA DE TIEMPO DE EJECUCIÓN Y RENDIMIENTO.
% timeit func(values)

FUNCIONES PYTORCH:

- VIEW: LOS MISMOS DATOS EN UN ACOMODO DIFERENTE.

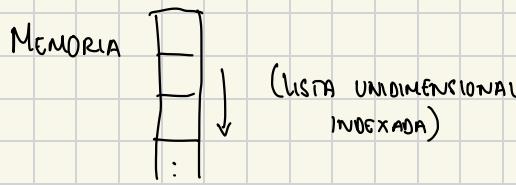
$$[4 \times 4] \rightarrow [2 \times 8]$$

(SIN NECESIDAD DE HACER COPIAS DE LOS DATOS)

- SHAPE: $x.shape = (2, 3, 4)$

- STRIDE: $x.stride = (12, 4, 1) \rightarrow$ ZANCADA

ACCESO A DATA EN MEMORIA:



- BROADCASTING: APLICAR OPERACIONES CON UN TENSOR A CADA UNO DE LOS ELEMENTOS DE OTRO TENSOR.
(OPERACIÓN VECTORIZADA)

REGLAS PARA HACER BROADCASTING:

- CADA TENSOR TIENE AL MENOS UNA DIMENSIÓN
- ITERAR SOBRE LAS DIMENSIONES DESDE EL FINAL
 - DIMENSIONES IGUALES
 - UNA DE LAS ES 1
 - UNA DE LAS NO EXISTE

Ejemplos:

TENSORES $a = [0, 1, 2]$

$$M = \begin{bmatrix} [1, 1, 1] \\ [1, 1, 1] \\ [1, 1, 1] \end{bmatrix}$$

DIMENSIÓN = [3]

DIMENSIONES = [3, 3]

BROADCASTING:

MATRIZ $\rightarrow [3, 3]$ } UNA DIMENSIÓN
VECTOR $\rightarrow \otimes [3]$ ES IGUAL Y

LA OTRA NO EXISTE.

RESULTADO: $\begin{bmatrix} [1, 2, 3] \\ [1, 2, 3] \\ [1, 2, 3] \end{bmatrix}$

TENSORES

$$a = [0, 1, 2]$$

$$M = \begin{bmatrix} [1, 1] \\ [1, 1] \\ [1, 1] \end{bmatrix}$$

DIMENSIÓN = [3]

DIMENSIONES = [3, 2]

Error:

$$M \rightarrow [3, 2] \quad \text{No se cumple}$$

$$a \rightarrow [3] \quad \text{Ninguna condición}$$

EJEMPLOS DE BROADCASTING:

$A + B \rightarrow A$ ES UNA MATRIZ CON DIMENSIONES
 $[4, 2, 1]$

4 MATRICES DE 2×1

$$\begin{bmatrix} [1] \\ [2] \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

B ES UN VECTOR CON DIMENSIÓN
 $[3]$ $B = [3, 2, 1]$

BROADCASTING

$C = A + B =$ SUMAR " B " A CADA ELEMENTO
DE LA MATRIZ " A "

$$[4, 2, 1] \\ [3]$$

C SE CONVIERTÉ EN UN TENSOR DE
4 MATRICES DE 2×3 .

$$C = \begin{bmatrix} 1+3, 1+2, 1+1 \\ 2+3, 2+2, 2+1 \end{bmatrix}, \begin{bmatrix} 3+3, 3+2, 3+1 \\ 4+3, 4+2, 4+1 \end{bmatrix}, \dots$$

$$C = \begin{bmatrix} 4, 3, 2 \\ 5, 4, 3 \end{bmatrix}, \begin{bmatrix} 6, 5, 4 \\ 7, 6, 5 \end{bmatrix}, \dots$$

SUMAR EL VECTOR B A CADA ELEMENTO
DEL TENSOR A

VC - Dimension

BALANCEAR

- MUCHAS MUESTRAS
- ESPACIO DE HIPÓTESIS REDUCIDO

1. DEFINIMOS UN ESPACIO DE HIPÓTESIS H
2. ENTRENAMOS
3. ELEGIMOS UNA h^*

SEGUO: ESTA HIPÓTESIS ES DEPENDIENTE DEL ENTRENAMIENTO Y DEL DATASET QUE USAMOS.

- NO ES UNA GARANTIA ELEGIR UNA h QUE MINIMICE EL ERROR EN EL ENTRENAMIENTO.

LA DIMENSIÓN VC = TAMAÑO MÁXIMO DE UN CONJUNTO DE DATOS PARA EL CUAL UN MODELO PUEDE ENCONTRAR UNA h QUE SE AJUSTE A ESOS DATOS. [CAPACIDAD DE UN MODELO PARA ENCONTRAR UNA FUNCIÓN DE APROXIMACIÓN]

Alta VC-Dimension:

- MÁS FLEXIBLE / MODELO MÁS COMPLEJO
- SE ADAPTA A UN CONJUNTO DE DATOS MÁS COMPLEJO.

- MÁS PROPENSO A SOBRE-ADJUSTE

Baja VC-Dimension:

- MENOS PROPENSO A SOBRE-ADJUSTE.
- DIFICULTADES PARA MODELAR DATOS MÁS COMPLEJOS.

LA VC-DIMENSION ES UNA MEDIDA TEÓRICA QUE AyUDA A COMPRENDER LA CAPACIDAD DE UN MODELO PARA GENERALIZAR UN CONJUNTO DE DATOS.

INTERPRETACIÓN.

CAPACIDAD DE UN MODELO PARA AJUSTARSE A DISTINTAS CONFIGURACIONES DE DATOS.

DEFINICIÓN.

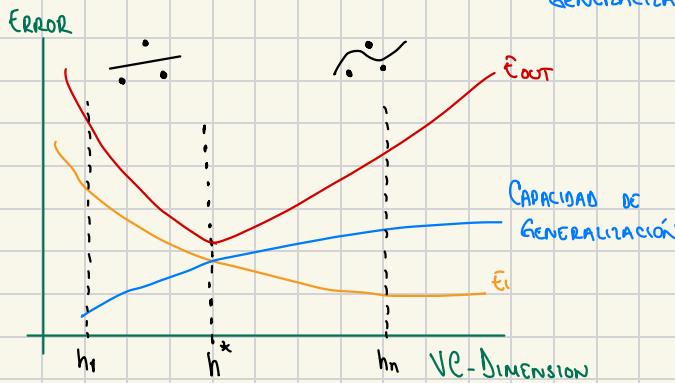
ES EL MÁXIMO NÚMERO DE PUNTOS QUE UN MODELO PUEDE SEPARAR DE TODAS LAS FORMAS POSIBLES, DE ACUERDO CON TODAS LAS COMBINACIONES POSIBLES DE ETIQUETADO.

- N PUNTOS O DATOS DE ENTRADA.
- ¿CUÁNTAS FORMAS DIFERENTES DE ETIQUETAR N TENEMOS?
↳ ¿EL MODELO PUEDE SEPARAR TODAS ESAS COMBINACIONES?

LA DIMENSIÓN VC CUANTIFICA LA VARIEDAD DE RELACIONES EN LOS DATOS QUE UN MODELO PUEDE CLASIFICAR.

ERROR DE GENERALIZACIÓN:

$$\text{Error en test} \leq \underbrace{\text{Error durante Entrenamiento}}_{\text{DATOS NECESARIOS PARA GENERALIZACIÓN}} + \underbrace{\text{Complejidad}/N}_{\text{CAPACIDAD DE GENERALIZACIÓN}}$$



VC DIMENSION EN NN.

- UNA RED CON MÁS NEURONAS Y CAPAS TENDRÁ UNA DIMENSIÓN VC MÁS ALTA → MAYOR CAPACIDAD PARA REPRESENTAR FUNCIONES COMPLEJAS.

↳ LAS NN SON MUY PROPENSAS A SOBRE-ADJUSTE POR LO QUE NECESITAN DE TÉCNICAS DE REGULARIZACIÓN, YA QUE SU ALTA COMPLEJIDAD AYUDA A QUE APRENDEAN DETALLES ESPECÍFICOS DE LOS DATOS DE ENTRENAMIENTO EN LUGAR DE GENERALIZAR.

- ESTO APARECE PRINCIPALMENTE EN CONJUNTOS DE DATOS PEQUEÑOS O RUIDOSOS.

- EN REDES CONVOLUCIONALES SE COMPARTEN PARÁMETROS (COMO LOS PESOS) POR LO QUE SU DIMENSIÓN VC SE CONSIDERA MÁS CONTROLADA.

- EN REDES RECURRENTES, LA DIMENSIÓN VC DEPENDE DE LAS UNIDADES RECURRENTES Y LA LONGITUD DE SECUENCIA QUE PROCESAN.

- TRANSFORMADORES: TIENEN UNA DIMENSIÓN VC ALTA QUE SE INCREMENTA CON LA PROFUNDIDAD Y EL NÚMERO DE CARREAS DE ATENCIÓN

DNNS
TRANSF



CNN
RNN



ENSAMBLE
ÁRBOLES

Propensos a sobreajuste
EN DATOS PEQUEÑOS O RUIDOSOS

Moderado debido
a restricciones
espaciales / temporales

Buenas generalizaciones
en datos tabulares.

→ CUAN MÁS ALTA ES LA DIMENSIÓN VC DE UNA FAMILIA DE MODELOS, MÁS IMPORTANTE ES APLICAR TÉCNICAS DE REGULARIZACIÓN PARA PREVENIR EL SOBREAJUSTE. LAS TÉCNICAS DE REGULARIZACIÓN CONTROLAN LA CAPACIDAD DE LOS MODELOS DE APRENDER DETALLES MUY ESPECÍFICOS DE LOS DATOS.

LA COMPLEJIDAD DE UN MODELO, AFECTA SU CAPACIDAD DE GENERALIZACIÓN.

LA VC-DIMENSION "k" SE REFIERE AL NÚMERO MÁXIMO DE PUNTOS QUE UN MODELO PUERDE ETIQUETAR CORRECTAMENTE SIN IMPORTAR CÓMO ESTÁN DISPUESTOS.

"k" ES UN BREAK POINT PARA UN CONJUNTO DE HIPÓTESIS H.

SE ASUME QUE EL MODELO YA NO PODRÍA APRENDER EL "k+1" DATOS.

PARA UN CONJUNTO DE DATOS BINARIO Y UN CONJUNTO DE DATOS n, EXISTE UNA FUNCIÓN DE CRECIMIENTO:

k-1

$$m_H(n) \leq \sum_{i=0}^{k-1} \binom{n}{i} = O(n^{k-1})$$

DICOTOMIAS POSIBLES

COMBINACIONES QUE GENERA

UNA FAMILIA DE FUNCIONES

PARA UN CONJUNTO DE DATOS.

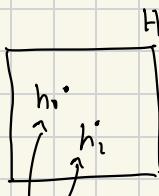
n > k

NÚMERO DE DICOTOMIAS GARANTIZADAS.

⊕

⊖

⊕



FUNCIONES HIPÓTESIS

a y b SON MIS CLASIFICACIONES

- TENGO 4 COMBINACIONES DE "a" Y "b" (DICOTOMIAS) QUE DIVIDEN LOS DATOS

$m_H(2) = 4$ PARA 2 PUNTOS x_1 Y x_2 .

- CADA FUNCIÓN $h \in H$ GENERA UNA DICOTOMIA EN PARTICULAR, UN PATRÓN DE SALIDA PARA LAS ENTRADAS x_i .

- ASIGNAMOS UN CONJUNTO X PARTICULAR QUE AYUDA A NUESTRAS HIPÓTESIS A ENCONTRAR LAS DICOTOMÍAS.

Las dicotomías son los posibles etiquetados que puede generalizar una familia de funciones. Nos interesa encontrar puntos que permitan generalizar la mayor cantidad de dicotomías para asegurar que durante el entrenamiento, nuestra familia de funciones pueda alcanzar el etiquetado presente en el conjunto de datos.

- La complejidad del aprendizaje está dada por la complejidad de los modelos y la cantidad de datos que tenemos disponibles.

La teoría de Aprendizaje Estadístico sugiere una relación entre la VC-Dimensión y el número de ejemplos necesarios para alcanzar generalización.

↳ Mayor complejidad → Mayor datos necesarios para Generalización

VC - Dimensión y Relación con la Navaja de Occam:

Principio de la Navaja de Occam: Al existir varias hipótesis que expliquen un fenómeno, debemos elegir la más simple.

- En Aprendizaje Estadístico se prefiere un modelo con menor complejidad para tratar de evitar sobreajustes a los datos de entrenamiento.
- El error de generalización se mantiene bajo, si la dimensión VC es baja en relación con el tamaño de la muestra.

FUNCTION DE CRECIMIENTO:

TEOREMA DE SAUER:

RELACION ENTRE EL NÚMERO DE BREAK POINTS
Y LA VC-DIMENSIÓN DE UN CONJUNTO DE
FUNCIONES HIPOTÉTICAS.

- **BREAK POINT:** Número "d" para el cual hay al menos una configuración de puntos que no puede ser correctamente etiquetada por el conjunto de hipótesis H.
- **VC-DIMENSIÓN:** Tamaño máximo de puntos que un modelo puede etiquetar.

LA VC-DIMENSIÓN DEL BREAK POINT "d" SATISFACE:

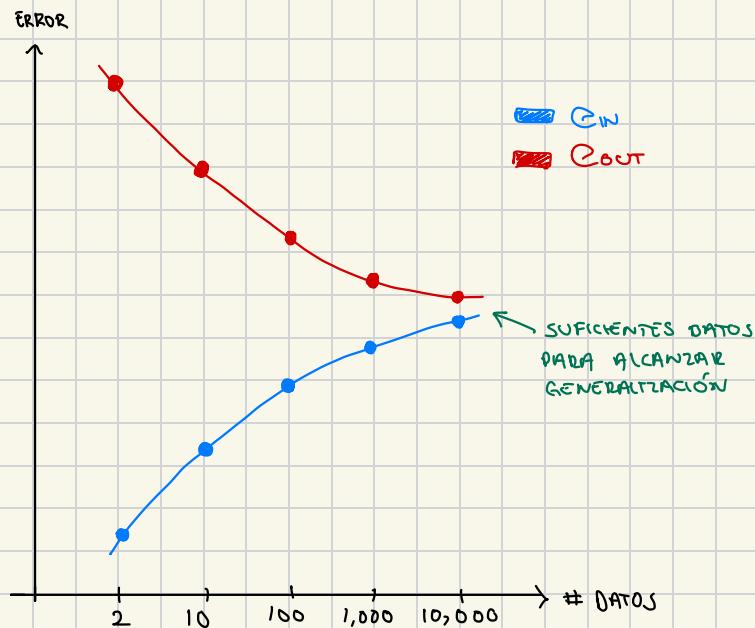
$$d_{vc} \leq d + 1$$

LÍMITE SUPERIOR DE LA VC-DIMENSIÓN
EN TÉRMINOS DEL BREAK POINT.

RESULTADO TEÓRICO:

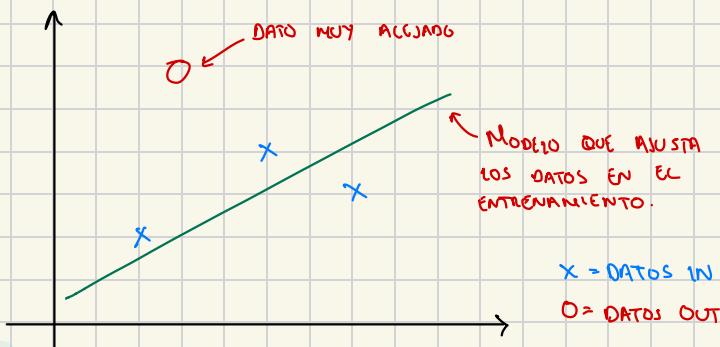
- COMPRENSIÓN DE LAS PROPIEDADES DEL CONJUNTO DE FUNCIONES HIPOTÉTICAS.
- DESCRIBE LA CAPACIDAD DE GENERALIZACIÓN DE UN MODELO.

FuncióN DE Crecimiento.



INTERPRETACIÓN:

- CUANDO TENEMOS POcos DATOS EL ERROR EN EL ENTRENAMIENTO ES PEQUEÑO PORQUE LAS FAMILIAS DE MODELOS QUE PUEDE AJUSTARLOS ES MUY GRANDE, INCLUSO UNA LÍNEA O UNA CONSTANTE PODEn APROXIMARSE A ELLos. SIN EMBArgo, LA PROBABILIDAD DE ENCONTRAR VALORES NO ESPERADOS, FUERA DE LOS DATOS DE ENTRENAMIENTO, ES ALTA.



FUNCIÓN DE CRECIMIENTO. $M_f(n)$

PARA UNA FAMILIA DE FUNCIONES \mathcal{F} Y UN CONJUNTO DE n PUNTOS, LA FUNCIÓN DE CRECIMIENTO $M_f(n)$ SE DEFINE COMO EL MÁXIMO NÚMERO DE PATRONES DE CLASIFICACIÓN EN DOS CLASES QUE PUEDEN SER SEPARADOS POR FUNCIONES EN \mathcal{F} .

$$M_f(n) = \max_{x_1, x_2, \dots, x_n} |\{(f(x_1), f(x_2), \dots, f(x_n)) : f \in \mathcal{F}\}|$$

CANTIDAD MÁXIMA DE MANERAS EN QUE LAS FUNCIONES DE \mathcal{F} PUEDEN CLASIFICAR ESOS PUNTOS.

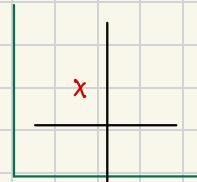
- CAPACIDAD DE UNA FAMILIA DE FUNCIONES PARA DIVIDIR EL ESPACIO DE DATOS DE DIFERENTES FORMAS A MEDIDA QUE AUMENTANOS LA CANTIDAD DE PUNTOS.

PARA UN CLASIFICADOR LINEAL

$$\begin{matrix} m=2 \\ =2^2 \end{matrix}$$



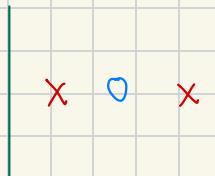
$$\begin{matrix} m=1 \\ =2^1 \end{matrix}$$



CUALQUIER LÍNEA PUEDE DIVIDIR LOS PUNTOS EN CUALQUIER CONFIGURACIÓN.

$$M_f(3) < 8$$

$$\begin{matrix} m=3 \\ =2^3 \end{matrix}$$



$$d_{VC} = 3$$

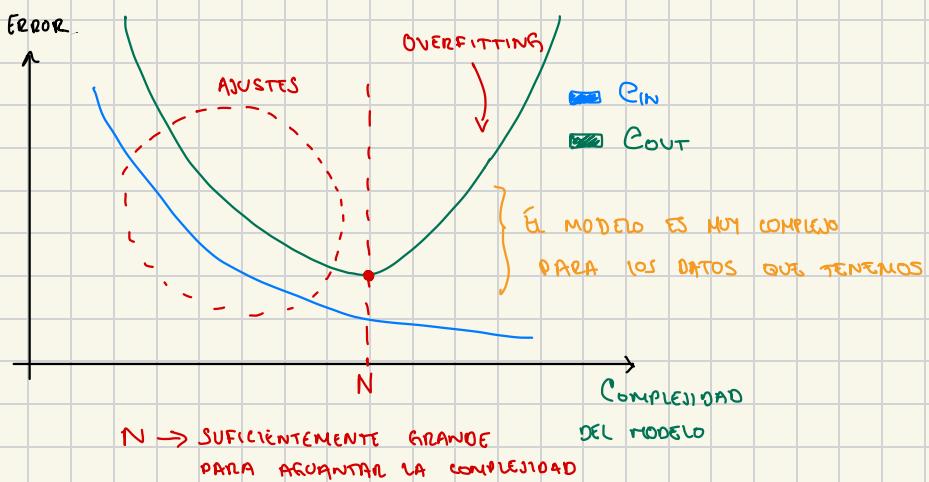
COTAS UNIFORMES:

COTAS DE GENERALIZACIÓN. \rightarrow EN DATOS NO CONOCIDOS.

$$P [\text{Error}_{in}(h_i) - \text{Error}_{out}(h_i) > \epsilon] \leq 2N^{VC}$$

\hookrightarrow COTA DE UNA UNIÓN DEL ERROR DE CADA HIPÓTESIS h DEL ESPACIO DE HIPÓTESIS H . LÍMITE DE PROBABILIDAD.

- ENTRE MÁS COMPLEJO ES EL ESPACIO DE HIPÓTESIS, MÁS DATOS SE NECESITAN PARA GARANTIZAR GENERALIZACIÓN.



VC-DIMENSION \rightarrow TAMAÑO MÁXIMO DE PUNTOS QUE PUEDEN SER CLASIFICADOS CORRECTAMENTE POR FUNCIONES QUE PUEDE APRENDER UN MODELO.

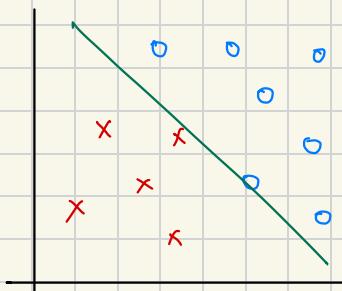
- CUANTO MENOR SEA LA VC-DIMENSION, MÁS ESTRECHAS SERÁN LAS COTAS DE GENERALIZACIÓN \rightarrow MAYOR CAPACIDAD DE GENERALIZACIÓN DEL MODELO.

\hookrightarrow

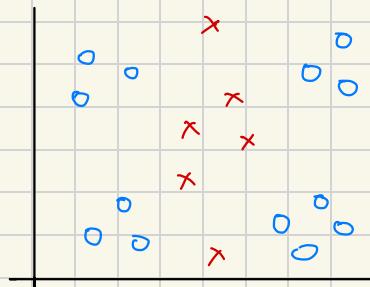
- ① INICIAR EL APRENDIZAJE CON MODELOS SENCILLOS.
- ② ASEGURARSE QUE ESOS MODELOS SON PUEDEN GENERAR LAS HIPÓTESIS QUE BUSCAMOS.

EJEMPLO:

- EL MODELO MÁS SENCILLO PARA CLASIFICACIÓN BINARIA PUEDE SER UN MODELO LINEAL:



SIN EMBARGO, SI LA DISPOSICIÓN DE LOS DATOS ES ASÍ:



UN MODELO LINEAL APLICADO DIRECTAMENTE SERÍA INCAPAZ DE APRENDER ESA INFORMACIÓN DE NADA REAL.

- POR OTRO LADO, ENTRE MÁS COMPLEJO SEA EL MODELO QUE USAMOS PARA APRENDER, MÁS DATOS VAMOS A REQUERIR PARA ENTRENAR.

RELACIÓN ENTRE VC-DIMENSIÓN Y $M_F(n)$

DIMENSIÓN VC = $d \rightarrow$ TAMAÑO MÁXIMO DE n PUNTOS QUE PUEDEN SER CLASIFICADOS DE TODAS LAS MANERAS POSIBLES.

$M_F(n) \rightarrow$ MUESTRA CÓMO CRECE LA CAPACIDAD DEL MODELO DESDE $n=1$ HASTA $n \leq d$ Y DESPUÉS DE $d+1$.

EJEMPLO: PARA UN CLASIFICADOR LINEAL EN 2D

$$d = 3$$

$$n = 1 \rightarrow M_F(1) = 2^1 = 2$$

$$n = 2 \rightarrow M_F(2) = 2^2 = 4$$

$$n = 3 \rightarrow M_F(3) = 2^3 = 8$$

$n = 4 \rightarrow M_F(4) = 6$ PORQUE HAY 2 CONFIGURACIONES QUE YA NO SE PUEDEN DIVIDIR CON UNA LÍNEA.

Por lo tanto, $M_F(n)$ es 2^n cuando $n=3$ (CRECIMIENTO EXPONENCIAL), pero crece $O(nd)$, es decir a lo más n^3 a partir de la dimensión VC que es $d=3$ (se reduce a un orden polinómico).

$$\cdot n \leq 3 \quad M_F(n) = 2^n$$

$$\cdot n < 3 \quad M_F(n) = n^3$$

Proceso de Entrenamiento usando batches (Potencias de 2)

Usamos tamaños de lote que son potencia de 2 como 32, 64, 128, etc.

- ↳ LAS GPUs ESTÁN OPTIMIZADAS PARA TRABAJAR CON TAMAÑOS DE DATOS QUE SON POTENCIAS DE DOS.
- ↳ MUCHAS OPERACIONES VECTORIZADAS SE REALIZAN DE FORMA PARALELA EN POTENCIAS DE DOS A NIVEL DE HARDWARE.

Operaciones CUDA

(Compute Unified Device Architecture)

- INSTRUCCIONES Y HERRAMIENTAS DESARROLLADAS PARA EJECUTAR TAREAS SOBRE GPUs.

- ↳ ES UNA ARQUITECTURA QUE EJECUTA MILES O MILLONES DE HILOS EN PARALELO.

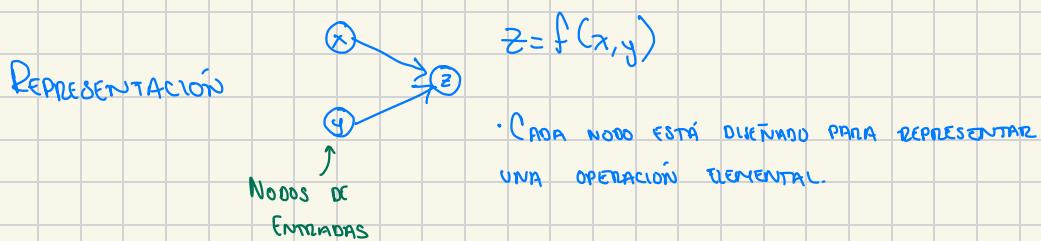
Hilos: Unidad individual de procesamiento.

GRAFOS COMPUTACIONALES.

GRAFOS DIRIGIDOS ACÍCLICOS (DAG) QUE SIRVEN COMO REPRESENTACIÓN ESTRUCTURADA DE VARIABLES INTERNAS Y PROCESOS QUE SE REALIZAN SOBRE LOS DATOS. ES UNA FORMA DE DESCRIBIR CÓMO LOS CÁLCULOS SE CONECTAN.

UNA VARIABLE PUEDE SER: Vector / Matriz / Escalar

- SE DESEA ENCONTRAR EL GRADIENTE CON RESPECTO A LOS PARÁMETROS O LAS ENTRADAS PARA ENCONTRAR LA MEJOR ACTIVACIÓN DE LA RED.
- PERMITE GENERAR ARQUITECTURAS CUSTOM.



DIFERENCIACIÓN AUTOMÁTICA.

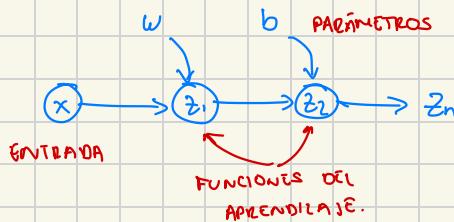
- REQUIERE QUE CADA OPERACIÓN DEL GRAFO SEA DIFERENCIABLE.
- SE CALCULAN GRADIENTES DE CADA NODO OBTENIENDO SUS DERIVADAS PARCIALES SOBRE CADA ENTRADA
- SE PROPAGA EL GRADIENTE HACIA ATRÁS.

OPERACIONES:

- FORWARD PROPAGATION: CALCULAR LAS FUNCIONES EN ORDEN TOPOLOGICO.
 - BACKWARD PROPAGATION: CALCULAR EL GRADIENTE δ^l EN ORDEN TOPOLOGICO INVERSO.
- El orden topológico del grafo lo podemos obtener con DFS.



→ CUALQUIER ARQUITECTURA QUE SE PUEDA REPRESENTAR COMO UN GRAFO COMPUTACIONAL DIFERENCIAL, SIRVE PARA CORRER UN ALGORITMO DE APRENDIZAJE.



Un nuevo nodo se obtiene mediante el cómputo de nodos predecesores.

Dentro de los frameworks donde se usan grafos computacionales, se debe especificar las derivadas parciales respecto a sus entradas, para poder usar el algoritmo de optimización.

Deep learning: Muchos fenómenos son de naturaleza jerárquica y los grafos computacionales tratan de abstractear esos procesos en ese orden.

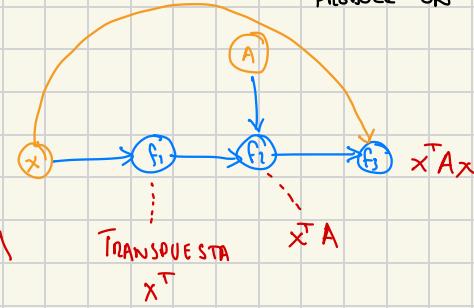
- El número de capas de un nodo aumenta su VC-Dimension (complejidad), pero buscamos que esa nueva complejidad vaya en la dirección correcta que ajusta el problema.

En un grafo computacional, la optimización se hace con los cálculos del gradiente. Los frameworks ya tienen operaciones con derivadas, cuando deseamos modificar una arquitectura, únicamente necesitamos incluir los cálculos de derivadas.

EJEMPLOS:

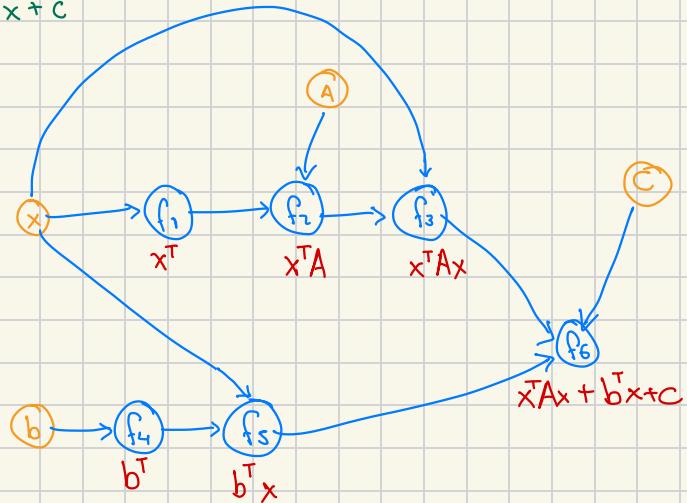
Expresión: $x^T A x$

$(1 \times n) (n \times n) (n \times 1) = 1 \times 1 \rightarrow$ OPERACIÓN QUE PRODUCE UN ESCALAR



Expresión: $x^T A x + b^T x + c$

GRAFO:



LA VC-DIMENSIÓN DE UN GRAFO COMPUTACIONAL ESTÁ ASOCIADA A SUS PARÁMETROS, ARQUITECTURA Y COMPLEJIDAD DEL MODELO.

$$x = [1, 2, 3]$$

$$A = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

$$x^T A = \begin{bmatrix} (1 \cdot 3) + (2 \cdot 4) + (5 \cdot 3) \\ (1 \cdot 6) + (2 \cdot 7) + (3 \cdot 8) \end{bmatrix} = \begin{bmatrix} 26 \\ 44 \end{bmatrix}$$

REDES NEURONALES

APROXIMADOR UNIVERSAL: CUALQUIER FUNCIÓN CONTINUA SE PUEDE REPRESENTAR CON UN MODELO DE NN DE UNA SOLA CAPA CON UN NÚMERO FINITO DE NEURONAS.

→ ESE NÚMERO FINITO DE NEURONAS PUEDE ESTAR EN EL ORDEN DE MILLONES DE NODOS.

↳ CON ESTO, EL MODELO SE VUELVE MUY COMPLICADO Y AL TENER DATOS FINITOS PARA ENTRENAR, ES ALTAMENTE PROBABLE CAER EN OVERFITTING.

PROOF:

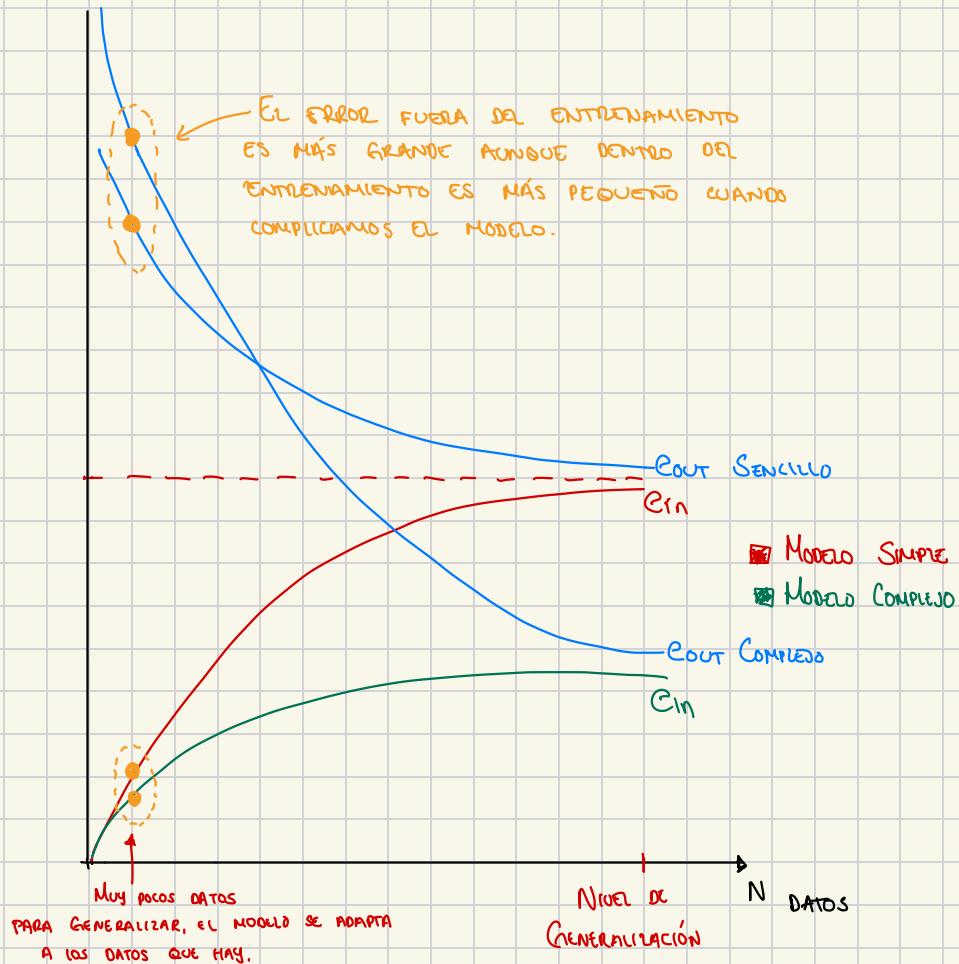
• TEOREMA DE APROXIMACIÓN UNIVERSAL

A MEDIDA QUE EL NÚMERO DE NEURONAS EN LA CAPA OCULTA TIENE A INFINITO, LA RED PUEDE REPRESENTAR CUALQUIER MAPEO CONTINUO.

Las neuronas son capaces de combinar funciones de actuación no lineales para formar expresiones complejas

DEFINIMOS UN ERROR CONTROLANDO $\epsilon > 0$ ARBITRARIAMENTE PEQUEÑO, A MEDIDA QUE LAS NEURONAS AUMENTAN, PODREMOS REDUCIR EL ERROR.

CURVAS DE APRENDIZAJE

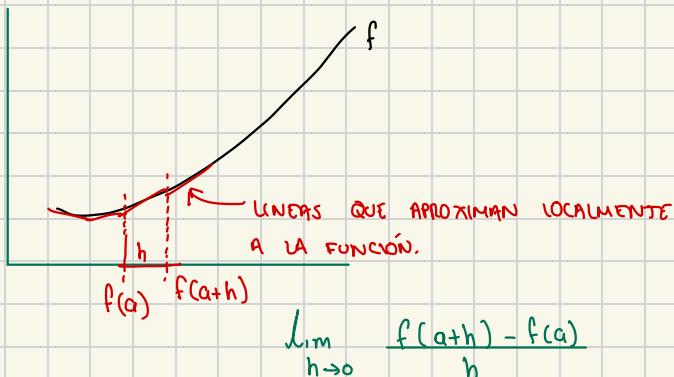


HIPERPARÁMETROS: Los definimos nosotros cuando proponemos una arquitectura.

PARÁMETROS: Son aprendidos por el modelo después de varias iteraciones.

Cálculo Diferencial Vectorial

DERIVADA - EVALUACIÓN DEL LÍMITE DEL CAMBIO DE LA FUNCIÓN CUANDO LOS VALORES SE VAN HACIENDO MÁS PEQUEÑOS (LA PENDIENTE).



DERIVADA PARCIAL

FIJAR TODAS LAS VARIABLES Y REVISAR CÓMO CAMBIA LA FUNCIÓN EN UNA COMPONENTE i^{th} EN PARTICULAR.

JACOBIANO : MATRIZ DE PRIMERAS DERIVADAS PARCIALES.

HESIANO : MATRIZ DE SEGUNDAS DERIVADAS PARCIALES.

REGLA DE LA CADENA PARA DERIVADAS.

UNA DERIVADA ES UNA APROXIMACIÓN LINEAL A UNA FUNCIÓN EN UN PUNTO LOCAL.

DEF.

$$\lim_{h \rightarrow 0} \frac{f(a+h) - f(a) - mh}{h} = 0$$

CAMBIO DE f CON
UN PASO DE TAMAÑO h

TRANSFORMACIÓN LINEAL

LA DERIVADA SE USA PORQUE ES LA MEJOR APROXIMACIÓN LINEAL AL CAMBIO DE LA FUNCIÓN EN UN PUNTO LOCAL.

DERIVADA DIRECCIONAL.

- CÓMO CAMBIA UNA FUNCIÓN EN LA DIRECCIÓN DE UN VECTOR \vec{v} CUANDO NOS MOVEMOS UN VALOR 'h'.

$$\lim_{h \rightarrow 0} \frac{f(a + h\vec{v}) - f(a)}{h} = [Df(a)]\vec{v}$$

REGLA DE LA CADENA.

- FÓRMULA PARA CALCULAR LA DERIVADA DE UNA COMPOSICIÓN DE FUNCIONES.

$$f(g(x)) \quad f: \mathbb{R} \rightarrow \mathbb{R}$$
$$g: \mathbb{R} \rightarrow \mathbb{R}$$

$$\partial f(g(x)) = f'(g(x)) \cdot g'(x) \leftarrow \begin{array}{l} \text{LA DERIVADA DE } g \\ \text{EVALUADA EN } x. \end{array}$$

\uparrow
LA DERIVADA DE f
EVALUADA EN $g(x)$

$$f = \sin u \quad g = 3x^2 + 2x$$

$$u = g = 3x^2 + 2x$$

$$f'(g(x)) \rightarrow \partial \sin(3x^2 + 2x) \quad g'(x) = 6x + 2$$

$$\partial \sin(u) = \cos u = \cos(3x^2 + 2x)$$

$$\partial f(g(x)) = \cos(3x^2 + 2x) \cdot 6x + 2$$

GRADIENTE DESCENDENTE. (Algoritmo de Optimización)

- Todos los modelos de Deep Learning comparten en alguna parte el cálculo de GRADIENTE DESCENDENTE ESTOCÁSTICO. [ALEATORIOIDAD / Incertidumbre]

GRADIENTE

(MÚLTIPLES ENTRADAS CON UNA ÚNICA SALIDA)

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

DERIVADA:

- TASA DE CAMBIO.
- QUÉ TAN RÁPIDO ESTÁ CAMBIANDO UNA FUNCIÓN Y EN QUÉ DIRECCIÓN.
- TRANSFORMACIÓN LINEAL QUE MEJOR APROXIMA A UNA FUNCIÓN.

DERIVADA PARCIAL:

- CÓMO CAMBIA LA FUNCIÓN CUANDO CAMBIO LA i -ESIMA VARIABLE Y FIJAMOS LAS DEMÁS.
[CAMBIAR UN PARÁMETRO Y FIJAR LOS DEMÁS]

MATRIZ JACOBIANO:

- MATRIZ QUE REPRESENTA LAS DERIVADAS PARCIALES DE UN CONJUNTO DE FUNCIONES VECTORIALES.

GRADIENTE DESCENDENTE:

- CÓMO CAMBIA LA FUNCIÓN LO MÁS RÁPIDO POSIBLE A SU MÍNIMO

$$\nabla f(a)^T h \rightarrow \text{QUÉ TANTO CAMBIA LA FUNCIÓN (SI CRECE O DECRECE)}$$

BUSCAMOS ENCONTRAR UNA h PARA QUE DECREZCA LA FUNCIÓN, PARA MOVERNOS EN DIRECCIÓN DEL MÁXIMO CAMBIO.

$$\nabla f(a)^T h = \|\nabla f(a)\| \|h\| \cos(\theta)$$

$\underbrace{\quad}_{\text{DIRECCIÓN DE LA FUNCIÓN}}$
BUSCAMOS
QUE ESTE INCREMENTO
SEA NEGATIVO,
 $\underbrace{\quad}_{\text{\|\|h\| NORMA O}} \quad \underbrace{\quad}_{\text{VALOR ABSOLUTO}}$

CUANDO $\theta = -\pi$ Y LA NORMA DE h $\|h\| = 1$

h DEBE ESTAR EN DIRECCIÓN OPUESTA AL GRADIENTE
PARA OBTENER:

$$- \|\nabla f(a)\|$$

$$h = \frac{-\nabla f(a)}{\|\nabla f(a)\|} \quad \leftarrow \text{RESUELVE EL PROBLEMA DE MINIMIZACIÓN DEL ERROR.}$$

CÁLCULO DE BACKPROPAGATION EN UN GRÁFO COMPUTACIONAL QUE REPRESENTA UNA RED NEURONAL MULTICAPA.

OBJETIVO: OPTIMIZAR LA FUNCIÓN DE ERROR $f_e(w_1, w_2, b)$ QUE DEPENDE DE LOS PARÁMETROS DEL MLP.

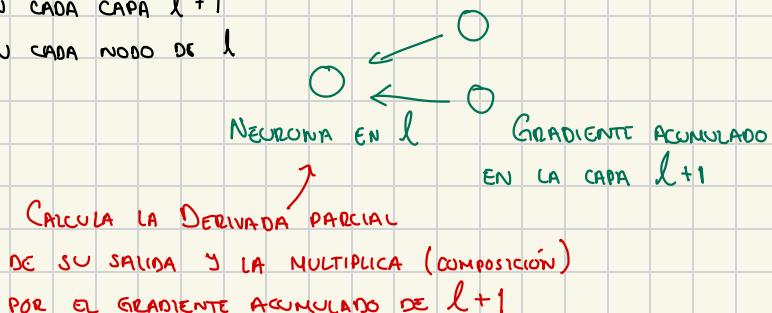
- EL ALGORITMO DE BACKPROPAGATION CALCULA DE MANERA EFICIENTE LOS GRADIENTES DE UNA FUNCIÓN DE COSTO CON RESPECTO A LOS PARÁMETROS DE UN MODELO.

↳ ORGANIZA EL CÁLCULO PARA EVITAR REDUNDANCIAS, REUTILIZANDO LOS RESULTADOS DE DERIVADAS INTERMEDIAS, APROVECHANDO LA ESTRUCTURA DE LA RED.

PARADIGMA DE PROGRAMACIÓN DINÁMICA:

EL ALGORITMO DE BACKPROPAGATION PUEDE VERSE COMO UN ALGORITMO DE PROGRAMACIÓN DINÁMICA, SUS SUBPROBLEMAS PUEDEN IDENTIFICARSE EN LOS CÁLCULOS DE LOS GRADIENTES PARCIALES

- CÁLCULOS EN CADA CAPA $l+1$
- CÁLCULOS EN CADA NODO DE l



SUBPROBLEMAS: CÁLCULOS POR CAPA EN ORDEN INVERSO.

¿CÓMO SE VAN RESOLVIENDO LOS SUBPROBLEMAS?

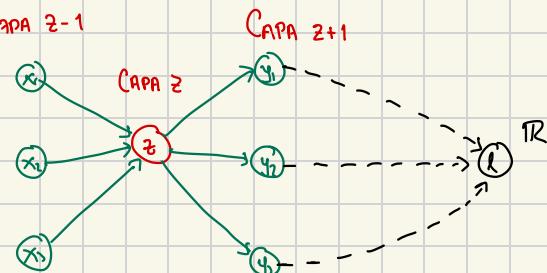
1. CADA NODO CONOCE SU VALOR DE ACTIVACIÓN Y EL DE LAS CAPAS ANTERIORES (FORWARD)

→ CÓMO CAMBIA CADA NODO RESPECTO A SUS ENTRADAS.

2. SE UTILIZA UNA REGLA DE LA CADENA PARA CALCULAR EL GRADIENTE APROVECHANDO LOS VALORES DE ACTIVACIÓN ALMACENADOS.

2.1 PARA CADA NODO EL GRADIENTE SE CALCULA EN FUNCIÓN DE LOS GRADIENTES DE LOS NODOS POSTERIORES.

¿CÓMO FLUYE EL GRADIENTE?



NOS INTERESA CALCULAR EL ERROR EN l CON RESPECTO A z

$$D_z l(\cdot) = \sum_{\text{TODOS LOS VECINOS DE } z} \sum [D_y l(\cdot)] [D_z y(\cdot)]$$

- z SUMA TODAS LAS CONTRIBUCIONES A l QUE TIENEN SUS VECINOS EN LA CAPA $z+1$

- HACE COMPOSICIÓN DE ESO CON CÓMO CAMBIA ÉL RESPECTO A SUS ENTRADAS.

- LE MANDA ESA INFORMACIÓN A SUS ENTRADAS EN $z-1$.

OPTIMIZACIÓN.

OPTIMIZACIÓN DE LA FUNCIÓN DE PÉRDIDA QUE DEPENDE DE LOS PARÁMETROS DEL MODELO.

MÍNIMO LOCAL: EL VALOR MÁS PEQUEÑO DENTRO DE UN INTERVALO DADO

MÍNIMO GLOBAL: EL VALOR MÁS PEQUEÑO DE LA FUNCIÓN EN TODO EL DOMINIO DE LA FUNCIÓN.

PROPIEDADES DE LOS MÍNIMOS:

CONDICIÓN NECESARIA - DERIVADA IGUAL A CERO.

VALORES DE HESSIANO - SEGUNDAS DERIVADAS PARCIALES.

GRADIENTE DESCENDENTE ESTOCÁSTICO.

- ESTIMADOR DEL GRADIENTE REAL OBTENIDO DE TOMAR SOLO UN BATCH DE MUESTRAS DE MANERA ALEATORIA DEL CONJUNTO DE ENTRENAMIENTO ORIGINAL.

El VALOR ESPERADO del GRADIENTE ESTOCÁSTICO VA EN LA MISMA DIRECCIÓN DEL GRADIENTE REAL.

EL TAMAÑO DEL BATCH ES RELEVANTE Y USUALMENTE VA EN MAGNITUDES DE 32, 64, 128 ... VS EVALUAR EL ERROR EN TODAS LAS MUESTRAS QUE PUEDEN SER MILES O MILLONES.

EPOCA: PASAR POR TODOS LOS DATOS DEL CONJUNTO DE ENTRENAMIENTO A TRAVÉS DE LAS PASADAS POR LOS BATCHES.

SOFTMAX.

FUNCIÓN DE ACTIVACIÓN QUE TRANSFORMA UN VECTOR DE VALORES REALES EN UNA DISTRIBUCIÓN DE PROBABILIDAD ENTRE 0 Y 1.

- PARTICULARMENTE ÚTIL EN CLASIFICACIÓN MULTICLASE.

$$\text{SOFT MAX}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

SUMA DE LAS
EXPONENCIALES DE TODAS
LAS ACTIVACIONES

REDES CONVOLUCIONALES CNN:

Operación de Convolución: Operación matemática que combina dos funciones $f(x)$ y $g(x)$ para crear una nueva $f(x) * g(x)$.

$$\text{Def. } f * g(x) = \int f(t) g(x-t) dt$$

↑
g se desplaza en el eje "x" sobre f.

f

1	2	1	2	1
0	1	0	1	2
1	2	1	1	0
0	1	1	2	1
2	2	1	0	0

*

g

1	0	1
0	1	0
1	0	1

=

5		
1	0	1

$$(f * g)(i, j) = \sum_{k=0}^{H_i} \sum_{l=0}^{W_j} g(k, l) f(i+k, j+l)$$

DIMENSIONES EN TENSORES:

$n \times n \times \text{DIMENSIONES} \times \text{TAMAÑO DEL BATCH}$

TAMAÑO DE LA MÁTRIZ NÚMERO DE MÁTRICES

PYTORCH CON-2D

INPUT (N, C_{in}, H, w)

CANALES

DIMENSIONES DEL KERNEL

CUANTOS FILTROS SE VAN A APLICAR A LA ENTRADA.

Operación de Convolución

$$X = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \quad \text{KERNEL} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

TAMAÑO DE LA SALIDA

$$\text{DIM}_X = 3 \times 4 \quad \text{DIM}_{\text{KERNEL}} = 2 \times 1$$

$$\text{DIM}_{\text{CONV}} = \left(\frac{\text{DIMENSIÓN ENTRADA} - \text{DIMENSIÓN KERNEL}}{\text{STRIDE}} \right) + 1$$

$$\text{FILAS} = (3 - 2) + 1 = 2$$

$$\text{COLUMNAS} = (4 - 1) + 1 = 4$$

$$\text{DIM}_{\text{CONV}} = 2 \times 4$$

$$\text{PRIMERA CELDA} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = (1)(1) + (2)(1) = 3$$

$$\text{CONN} = \begin{pmatrix} 3 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

INTUICIONES:

¿Por qué aprenden las CNN?

- IMPORTANCIA DE LA RELACIÓN ESPACIAL EN IMÁGENES.
 - Es útil cuando los datos tienen localidades espaciales significativas.
- EN COMPARACIÓN CON UN MLP, LAS REDES CONVOLUCIONALES HACEN UNA OPERACIÓN MÁS SENCILLA YA QUE EL KERNEL HACE QUE MUCHOS PESOS SE HAGAN CERO Y OTROS SE COMPARTAN.

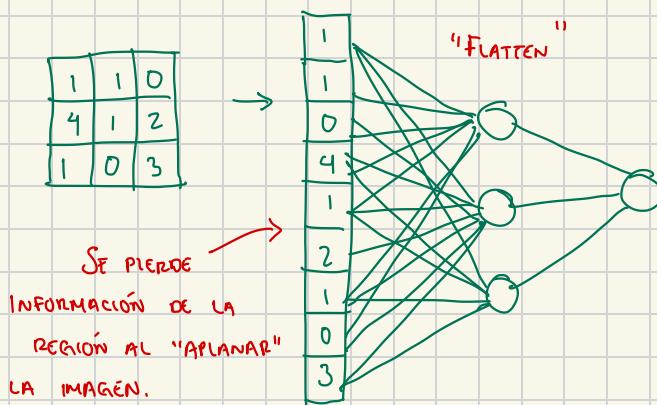
CUANDO INTAMOS QUE ALGUNOS PESOS SEAN IGUALES ESTAMOS INDUCIENDO UN PROCESO DE REGULARIZACIÓN.

- LOS FILTROS (KERNEL) NO SON FIJOS O PROPUESTOS, A PARTIR DE LOS DATOS SE APRENDEN CUÁLES SON LOS MEJORES VALORES PARA FILTRAR LA INFORMACIÓN.
- SE USAN VARIOS NIVELES DE CAPAS, DONDE CADA CAPA FUNCIONA COMO UN FILTRO ESPECÍFICO QUE SE FIJA EN PROPIEDADES MUY CONCRETAS DE LOS DATOS.
 - Logran aprender las características esenciales de los datos y "QUITAN" información poco relevante.
 - Extrae características que no son tan obvias incluso para expertos de los datos.

Aplicaciones:

- FILTRADO DE SEÑALES
- ELIMINACIÓN DE RUIDO
- DETECCIÓN DE OBJETOS
- ESTIMAR PROFUNDIDAD

LAS REDES CONVOLUCIONALES SE ORIGINARON PARA RESOLVER PROBLEMAS DE VISIÓN (IMÁGENES EN MATRICES)



PROPIEDADES DE LA OPERACIÓN DE CONVOLUCIÓN:

PARA ENCONTRAR EL GRADIENTE DESCENDENTE ; LA DERIVADA DE LA CONVOLUCIÓN ES LA CONVOLUCIÓN TRANSPUESTA.

→ EQUIVARIANZA A TRASLACIÓN

Si $f(x)$ ES EQUIVARIANTE A $T(x)$
→ $T(f(x)) = f(T(x))$

- LA OPERACIÓN DE CONVOLUCIÓN SE PUEDEN VER COMO LA MULTIPLICACIÓN DE LA SEÑAL POR UNA MATRIZ CIRCULANTE.

$$\begin{bmatrix} c_g \end{bmatrix} \begin{bmatrix} f \end{bmatrix} = \begin{bmatrix} h \end{bmatrix}$$

MATRIZ DE FILTROS g VECTOR DE UNA SEÑAL f . VECTOR DE CONVOLUCIÓN

→ TRASLACIÓN

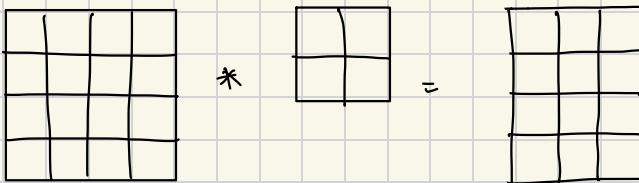
$$\begin{bmatrix} c_g \end{bmatrix} \begin{bmatrix} \downarrow \\ f \end{bmatrix} = \begin{bmatrix} \downarrow \\ h \end{bmatrix}$$

DESPLAZAR f
HACIA ABAJO

OPERACIÓN DE CONVOLUCIÓN EN MATRICES:

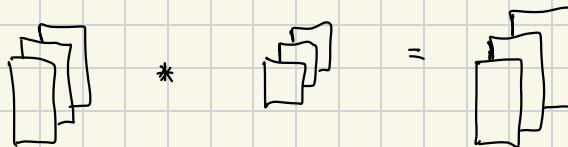
CONVOLUCIÓN 2D

MATRIZ DE ENTRADA + KERNEL = NUEVA MATRIZ CONVOLUCIONADA.



CONVOLUCIÓN 3D

CUBO DE ENTRADA + CUBO KERNEL = CUBO DE SALIDA



PRINCIPALES CAPAS:

1. CAPAS CONVOLUCIONALES: OPERACIÓN DE CONVOLUCIÓN CON UN KERNEL.

- SU OBJETIVO ES EXTRAER CARACTERÍSTICAS DE ALTO NIVEL.
- SE AGREGAN VARIAS CAPAS DE CONVOLUCIÓN PARA EXTRAER DIVERSOS NIVELES DE CARACTERÍSTICAS.

2. CAPAS DE AGRUPACIÓN:

- EXTRAER CARACTERÍSTICAS DOMINANTES INVARIANTES A ROTACIÓN Y POSICIÓN.
- DOS TIPOS: MAX / AVERAGE POOLING. (SUPRESIÓN DE RUIDO)

3. CAPAS FULLY CONNECTED.

- APLANAMIENTO Y CLASIFICACIÓN.
 - SOFTMAX: FUNCIÓN DE ASIGNACIÓN DE PROBABILIDADES.

PROPIEDAD DE EQUIVARIANZA A TRASLACIÓN.

- **VARIANZA:** LA SALIDA NO MANTIENE UNA RELACIÓN CONSISTENTE CUANDO SE APLICA UNA TRANSFORMACIÓN A LA ENTRADA COMO TRASLACIÓN O ROTACIÓN QUIL LOS MODELOS SON INCAPACES DE UTILIZAR PARA RECONOCER PATRONES.
- **INVARIANZA:** UN MODELO ES INVARIANTE SI LAS TRANSFORMACIONES DE LA ENTRADA NO MODIFICAN SUS SALIDAS. ES DECIR, EL MODELO ES CAPAZ DE IGNORAR LA TRANSFORMACIÓN.
- **EQUIVARIANZA:** LAS TRANSFORMACIONES A LA ENTRADA PRODUCEN SALIDAS EQUIVALENTES, MANTeniENDO UNA CORRESPONDENCIA PREDICIBLE.

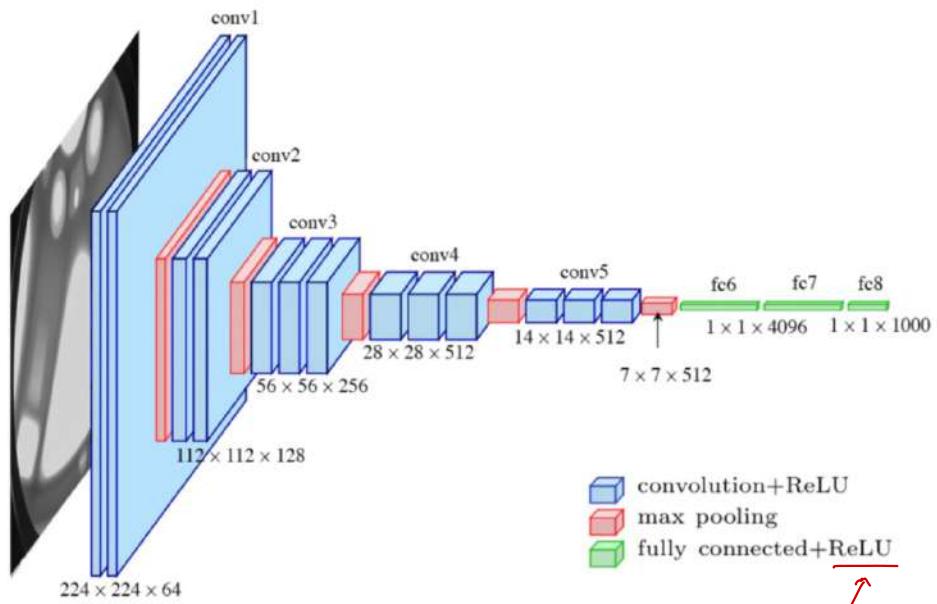
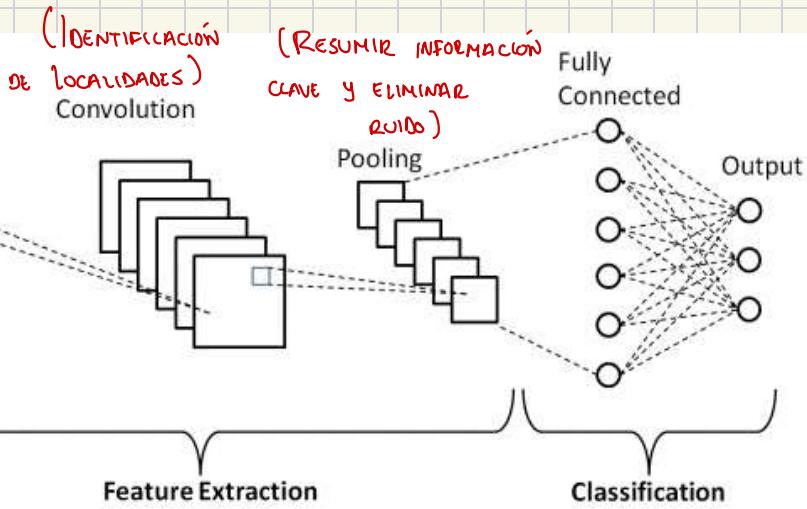
f ES EQUIVARIANTE A UNA TRANSFORMACIÓN T si:
 $f(T(x)) = T(f(x))$

EN CNNs TRASLADAR UN OBJETO EN UNA IMAGEN PRODUCIRÁ QUE LAS CARACTERÍSTICAS DETECTADAS POR LOS FILTROS SUFRIRÁN EL MISMO DESPLAZAMIENTO, EN LA MISMA DIRECCIÓN.

↳ ESTO SE LOGRA YA QUE EL MISMO FILTRO SE APLICA A DIFERENTES REGIONES, SIENDO CAPAZ DE DETECTAR LAS MISMAS CARACTERÍSTICAS SIN IMPORTAR SU POSICIÓN.

- LAS CNNs NO NECESITAN ENTRENAR CON TODAS LAS POSICIONES POSIBLES PARA RECONOCER OBJETOS.

NOTA: LA REDES CONVOLUCIONALES, POR EJEMPLO, SON INVARIANTES A TRASLACIÓN PERO NO A ROTACIÓN.



Parámetros compartidos

• EN LUGAR DE QUE CADA VALOR DE LA ENTRADA TENÍA UN PESO ASIGNADO COMO EN NNs CONVENCIONALES, UN MISMO FILTRO (KERNEL), QUE ES UNA MATRIZ DE PESOS, SE APLICA A UNA REGIÓN Y SE VA DESPLAZANDO POR TODA LA ENTRADA.

↳ ESTO REDUCE LA CANTIDAD DE PARÁMETROS Y BAJA LA COMPLEJIDAD DE LOS MODELOS.

CON FILTROS DE TAMAÑO 3×3 SOLO NECESITAMOS 9 PESOS, VERSUS 100 O MÁS SI TENEMOS UNA CONEXIÓN A UNA CAPA CON 100 NEURONAS.

Pooling:

SE PUEDE VER COMO UNA CONVOLUCIÓN NO PARAMÉTRICA
(NO SE ESTABLECEN PARÁMETROS, ES DECIR, SE APLICA
UN FILTRO PERO CON PROPIEDADES FIJAS, NO APRENDIDAS)

- COMPRESIÓN DE DATOS (RESUMIR INFORMACIÓN RESPETANDO LOCALIDADES)
- ELIMINACIÓN DE RUIDO
- AL REDUCIR EL TAMAÑO, OBLIGA A LA ARQUITECTURA A APRENDER CARACTERÍSTICAS RELEVANTES.
- AYUDA A ROBUSTECER LA CAPACIDAD DE INVARIANZA A PEQUEÑOS CAMBIOS EN LA ENTRADA.
- REDUCE LA COMPLEJIDAD COMPUTACIONAL SIN PERDER INFORMACIÓN CLAVE

CAPA FULL CONNECTED:

- RECIBE LAS CARACTERÍSTICAS DE LA ENTRADA YA SEGMENTADA Y ESTO PERMITE QUE SE HAGA APRENDIZAJE SOBRE UN ESPACIO DIFERENTE QUE YA FUE TRABAJADO POR LAS REDES CONVOLUCIONALES.

SE UTILIZAN PARA DETECTAR PATRONES LOCALES EN LOS DATOS Y ASIGNAR IMPORTANCIA A DIFERENTES ELEMENTOS.

- CAPTURAN DEPENDENCIAS ESPACIALES Y TEMPORALES.
- REDUCEN LAS IMÁGENES A UNA FORMA MÁS FÁCIL DE PROCESAR PERO SIN PERDER CARACTERÍSTICAS CRÍTICAS PARA LA PREDICCIÓN.
 - ↳ BUENAS PREDICCIONES Y ARQUITECTURAS ESCALABLES.
- A MAYOR NÚMERO DE CAPAS, MAYOR EXTRACCIÓN DE CARACTERÍSTICAS PERO MAYOR COSTO COMPUTACIONAL.

PADDING: INCREMENTAR LA MATRIZ ORIGINAL PARA, DESPUÉS DE APLICAR CONVOLUCIÓN CON EL KERNEL, OBTENER UNA MATRIZ CON LAS DIMENSIONES ORIGINALES DE LA ENTRADA.

STRIDE: TAMAÑO DEL PASO CON EL QUE EL KERNEL RECORRE LA MATRIZ DE ENTRADA.

GRADIENTE DESCENDENTE Y BACKPROPAGATION.

OBJETIVO: MEJORAR LA EXACTITUD DE LA PREDICCIÓN DE UNA NN.

FUNCIÓN DE COSTO: ERROR ENTRE EL VALOR REAL Y LA PREDICCIÓN.

BACKPROPAGATION: CALCULAR EL GRADIENTE A PARTIR DEL ERROR.

GRADIENTE DESCENDENTE: MINIMIZAR LA FUNCIÓN DE COSTO ACTUALIZANDO LOS PESOS.

- LA FORMA DE ACTUALIZAR LOS PESOS EN LA DIRECCIÓN CORRECTA ES A TRÁVES DEL CÁLCULO DE LAS DERIVADAS PARCIALES DE LA FUNCIÓN DE PÉRDIDA CON RESPECTO A LOS PESOS ASIGNADOS EN CADA NEURONA DE CADA CAPA.

DESVAJENCIAMIENTO DEL GRADIENTE

- GRADIENTES EXTREMAMENTE PEQUEÑOS PROVOCANDO QUE LAS ACTUALIZACIONES DE LOS PESOS SEAN INSIGNIFICANTES O NULAS, LO QUE DIFICULTA EL ENTRENAMIENTO EN ESAS CAPAS.
 - LAS PRIMERAS CAPAS APRENDEN MUY LENTO
 - EL MODELO TARDA MUCHO EN CONVERGIR

PARA VALORES ENTRE 0 Y 1, MULTIPLICAR VALORES POR NÚMEROS MENORES A 1 REDUCE LOS VALORES DEL GRADIENTE DE MANERA EXPONENCIAL.

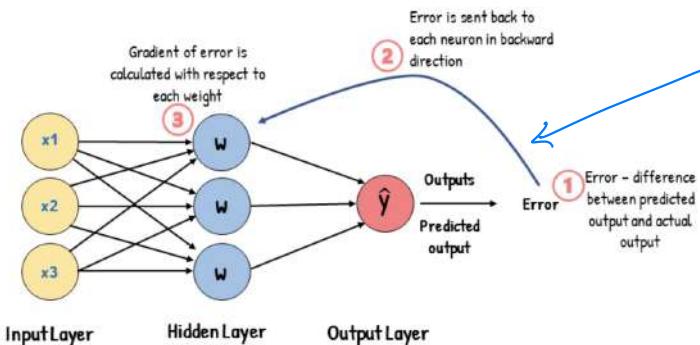
EXPLOSIÓN DEL GRADIENTE

- GRADIENTES QUE AUMENTAN EXPONENCIALMENTE HACIA LAS PRIMERAS CAPAS DE LA RED PROVOCANDO ACTUALIZACIONES ENORMES.
 - LOS PESOS SE DESESTABILIZAN PROVOCANDO UN MODELO INESTABLE.
 - LA FUNCIÓN DE PÉRDIDA OSCILA DE MANERA EXPONENCIAL O INCLUSO DIVERGE.
 - OCURRE CUANDO SE MULTIPLICA RAPIDAMENTE POR DERIVADAS MAYORES A 1.

CLIPPING: TÉCNICA QUE ESTABLECE UN UMbral A LOS GRADIENTES CON UN VALOR MÁXIMO POSIBLE.

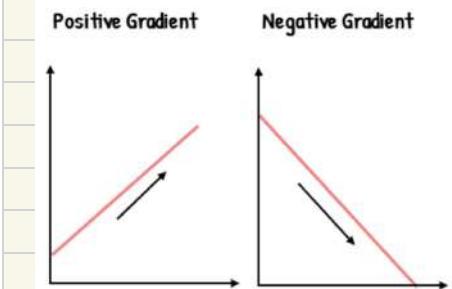
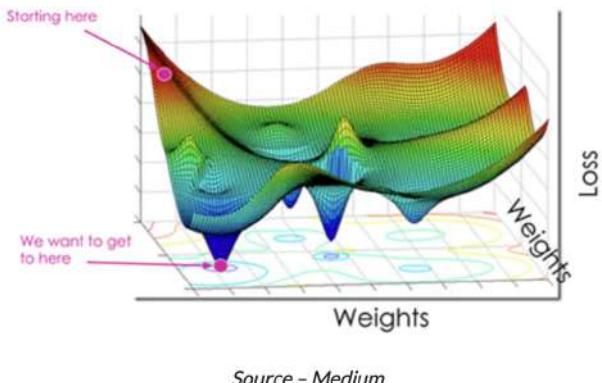
PROPAGACIÓN DEL ERROR ENTRE CAPAS:

Backpropagation



CÁLCULO DE
DERIVADAS
PARCIALES.

NAVEGACIÓN EN LA FUNCIÓN DE COSTO PARA ENCONTRAR EL ERROR MÍNIMO.



BUSCAMOS EL GRADIENTE
NEGATIVO PORQUE ESE IMPLICA
DECRECER EL ERROR.

OBTENCIÓN DE NUEVOS PESOS:

$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{dj}{dw}$$

TASA DE APRENDIZAJE

Gradient

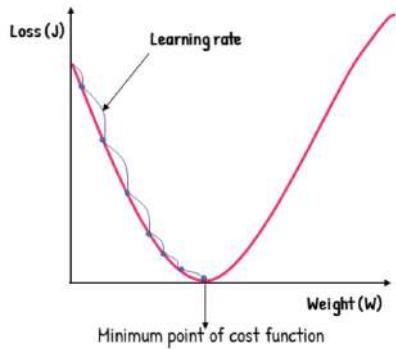
Learning Rate.

INDICA EL PASO Y LA VELOCIDAD CON LA QUE NOS MOVEMOS PARA ENCONTRAR EL MÍNIMO DE LA FUNCIÓN DE COSTO EN EL GRADIENTE DESCENDENTE.

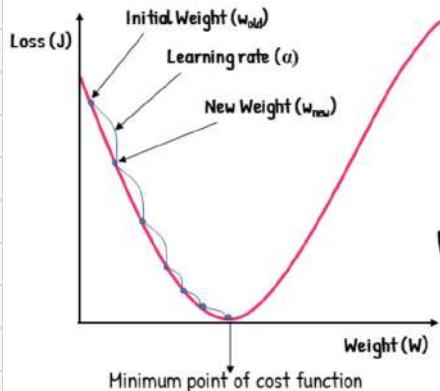
DEBEMOS BUSCAR UN BALANCE ENTRE LA OPTIMIZACIÓN DEL TIEMPO Y LA PRECISIÓN PORQUE UNA TASA DE APRENDIZAJE MUY RÁPIDA O CON UN PASO MUY GRANDE PUEDE ATORARNOS EN MÍNIMOS LOCALES. MIENTRAS QUE UNA TASA MUY PEQUEÑA IMPLICA MUCHAS ITERACIONES Y COSTO COMPUTACIONAL.

- CUANTO MAYOR SEA EL GRADIENTE (CAMBIO) DE LA FUNCIÓN DE COSTO, MÁS PRONUNCIADA SERÁ LA PENDIENTE Y MÁS RÁPIDO PODRÁ APRENDER UN MODELO
- SI EL GRADIENTE LLEGA A CERO O CONVERGE, EL MODELO PARA DE APRENDER.

Learning Rate



Gradient Descent



$$w_{\text{new}} = w_{\text{old}} - \alpha \frac{\delta J}{\delta w}$$

ALGORITMO DE APRENDIZAJE:

1. INICIALIZAR PESOS ALEATORIOS.
2. CALCULAMOS UN NUEVO PESO USANDO EL GRADIENTE Y LA TASA DE APRENDIZAJE.
 - 2.1. NAVEMOS POR LA FUNCIÓN DE COSTO VARIAS ITERACIONES.
3. OBTENEMOS EL MÍNIMO VALOR POSIBLE.
4. USAMOS LOS PESOS DEL MÍNIMO OBTENIDO PARA PROBAR EN VALIDACIÓN (O TESTING SI NO TENEMOS MUCHOS DATOS).

FuncióN DE ERROR:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N [f(x_i, \theta)]^2$$

GRADIENTE ESTOCÁSTICO (ALEATORIO)

$$\nabla_{\theta} L = \frac{1}{N} \sum_{i=1}^N \nabla [f(x_i, \theta)]^2$$

AJUSTE DE PARÁMETROS:

$$\theta^{\text{NEW}} = \theta^{\text{OLD}} + \eta \nabla_{\theta} L$$

VALOR ESPERADO (MEDIA PARA VARIABLES ALEATORIAS):

- GRADIENTE CON RESPECTO A LA DISTRIBUCIÓN DEL DATASET.

REGULARIZACIÓN:

ES UNA FORMA DE INOCIR BIAS AL MODELO PARA REDUCIR NUESTRO ESPACIO DE BÚSQUEDA Y DIRIGIR EL ERROR EN LA DIRECCIÓN CORRECTA DURANTE EL ENTRENAMIENTO.

REDUCEN EL ESPACIO DE BÚSQUEDA DE LAS FUNCIONES, ACOTANDO LAS Opciones DE LAS POSIBLES FUNCIONES DE APROXIMACIÓN QUE PUEDE ENCONTRAR EL MODELO.

EN MODELOS COMPLEJOS, LA REGULARIZACIÓN PERMITE REDUCIR LA VARIANZA DEL ERROR DE LAS FUNCIONES QUE AJUSTAN LA f^* VERDADERA. (UNA VARIANZA ALTA TIENDE A NO GENERALIZAR BIEN Y REQUERIR MUCHOS DATOS YA QUE SU VC-DIMENSION ES ALTA).

SON UTILIZADAS PARA EVITAR EL SOBREAJUSTE (MEMORIZAR DETALLES DE ENTRENAMIENTO, RUIDO O PATRONES IRRELACIONADOS). LA REGULARIZACIÓN IMPONE RESTRICCIONES QUE CONTROLAN LA COMPLEJIDAD DE LOS MODELOS.

EJEMPLOS:

- Weight Decay → Norma DE LOS PESOS DE LA RED NEURONAL.

- CNNs → POR LA ESTRUCTURA DE SU ARQUITECTURA:

LA COMPARTICIÓN DE PESOS IMPONE RESTRICCIONES AL REDUCIR EL NÚMERO DE PARÁMETROS SE CONTROLA LA COMPLEJIDAD.

AL CONECTAR EL FILTRO A REGIONES SE FUDIZA AL MODELO A CONCENTRARSE EN PATRONES LOCALES, EVITANDO DEPENDENCIAS DE POSICIÓN.

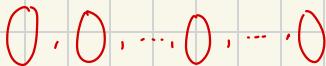
Skip Connections.

- EN DEEP LEARNING, CONFORME AUMENTAMOS EL NÚMERO DE CAPAS, LA COMPLEJIDAD DEBERÍA AUMENTAR Y POR LO TANTO, SE OBTIENE MAYOR EXPRESIVIDAD (APROXIMAR MÁS COSAS), SIN EMBARCO ESTO NO ES ASÍ SI NO SE INCLUYEN MÉTODOS DE REGULARIZACIÓN.

Modelo
CON 20 CAPAS



Modelo con
56 CAPAS



LA ARQUITECTURA CON 56
CAPAS, CONTIENE A LA DE 20
POR LO QUE DEBERÍA SER AL
MENOS IGUAL DE BUENA QUE
LA DE 56.

Skip Connection $f(x) + x$

ARGUMENTOS PARA DEMOSTRACIÓN:

- Copiamos la arquitectura y sus parámetros hasta la capa 20.
- A PARTIR DE LA CAPA 21, BUSCAMOS AJUSTAR LOS PESOS DE MANERA TAL QUE AJUSTEN LA FUNCIÓN IDENTIDAD (TODAS LAS CAPAS CON LOS MISMO PESOS).

↳ PROBLEMA: NO ENCONTREMOS ESOS PESOS CON GRADIENTE DESCENDENTE.

LAS SKIP CONNECTIONS SON CAPACES DE APROXIMAR LA FUNCIÓN IDENTIDAD (PASAR LA ENTRADA DIRECTAMENTE A LA SALIDA), LO QUE LE DA UNA CAPACIDAD A LA RED PARA DECIDIR SI UTILIZA LAS CAPAS INTERNALES O DECIDE IGNORARLAS.

Skip - Residual - Shortcut Connections.

- CONEXIÓN DIRECTA ENTRE CAPAS NO CONSECUTIVAS DE UNA RED.

VENTAJAS

- MITIGACIÓN DEL EFECTO DEL DESVANECLIMIENTO DEL GRADIENTE.
- FACILITA EL ENTRENAMIENTO.
- PRESERVA INFORMACIÓN ORIGINAL.
- PROPAGACIÓN DEL GRADIENTE MÁS EFECTIVA.

DESVANTAJAS

- COMPLEJIDAD ADICIONAL
- PODRÍA AGREGAR REDUNDANCIA INNECESARIA.

VANISHING ó DEGRADING GRADIENT.

- DIFICULTAD PARA RECIBIR GRADIENTES SIGNIFICATIVOS EN LAS PRIMERAS CAPAS DE LA RED. LOS GRADIENTES PUEDEN VOLVERSE EXTREMADAMENTE PEQUEÑOS O INCLUSO DESVANEZERSE.

LOS AJUSTES BASADOS EN GRADIENTES MUY DESQUEÑOS APENAS ACTUALIZAN SUS PESOS, DIFICULTANDO EL ENTRENAMIENTO DE LAS PRIMERAS CAPAS DE LA RED.

↳ LAS CONEXIONES RESIDUALES PERMITEN MANTENER UNA MAGNITUD MÁS GRANDE, ESTO ESTABILIZA EL ENTRENAMIENTO Y PERMITE UNA REDUCCIÓN MÁS RÁPIDA DEL ERROR.

CUANDO AGREGAMOS LAS CONEXIONES RESIDUALES, TENEMOS UN EFECTO QUE AFECTA LA MEDIA Y LA VARIANZA, AUMENTÁNDOLAS. PARA NEUTRALIZAR ESTE EFECTO, AGREGAMOS UNA CAPA DE NORMALIZACIÓN.

EXPRESSIVE POWER

POR QUÉ UNA NN CON SKIP CONNECTIONS TIENE EL MISMO PODER EXPRESIVO QUE LA MISMA RED SIN CONEXIONES RESIDUALES?

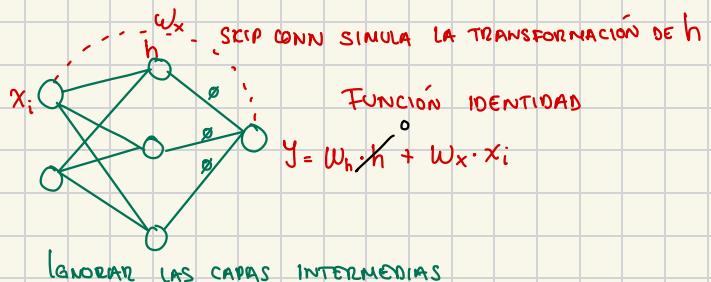
- LA RED CON SKIP CONNECTIONS PUEDE APRENDER LO MISMO QUE LA OTRA RED YA QUE LOS SALTOS NO REDUCEN EXPRESIVIDAD.

SALIDA DE LA CAPA CON SKIP CONN = $\underline{F(x) + x}$

SE COMBINA LA INFORMACIÓN PROCESADA CON LA ORIGINAL.

ARGUMENTOS:

- NECESITAMOS DEMOSTRAR QUE LA RED CON SKIP CONNECTIONS SE PUEDE COMPORTAR IGUAL QUE LA MISMA RED SIN ESAS CONEXIONES.
- SI LOS PESOS DE LAS NEURONAS INTERMEDIAS SE AJUSTAN A CERO, LA SALIDA DE LA RED SERÁ IGUAL A LA ENTRADA GRACIAS A LAS SKIP CONNECTIONS.



- LA FUNCIÓN IDENTIDAD QUE PUEDE REPRESENTAR UNA RED SIN SKIP CONNS CON UNA TRANSFORMACIÓN DE h , TAMBÍEN LA PUEDE REPRESENTAR LA RED CON SKIP CONNS AJUSTANDO w_x E IGNORANDO LAS CAPAS INTERMEDIAS, LO QUE LA NUELVE TAN SIMPLE COMO LA RED ORIGINAL.

- LAS SKIP CONNECTION NO REDUCEN LA CAPACIDAD DE UNA RED, SOLO EVITA LA DEGRADACIÓN DE INFORMACIÓN EN REDES PROFUNDAS.
- ADÉNÁS, PERMITEN COMBINAR APRENDIZAJES DE CARACTERÍSTICAS DE BAJO NIVEL CON OTRAS DE ALTO NIVEL (FORMAS COMPLEJAS), LO QUE AUMENTA SU PODER EXPRESIVO VS LA RED SIN SKIP CONN.

DROPOUT [REGULARIZACIÓN].

PREVENIR CO-ADAPTACIÓN Y OVERFITTING. (COMPENSA POCOS DATOS)

- AGREGAR ALÉATORIAMENTE ALGUNAS NEURONAS DE UNA CAPA OCULTA DURANTE ALGUNA ITERACIÓN.

↳ AL APAGAR ALGUNAS NEURONAS, ESTAS NO CONTRIBUYEN EN EL APRENDIZAJE, EVITANDO QUE CIERTA PARTE DE "RUIDO" SE APRENDA.

- PERMITE GENERALIZAR MEJOR YA QUE SE OBLIGA AL MODELO A APRENDER CARACTERÍSTICAS MÁS RÓBUSTAS

DROPOUT $y_i = \frac{x_i}{1-p}$ $p =$ PROBABILIDAD DE ELIMINACIÓN

ALÉATORIDAD EN EL PROCESO DE ENTRENAMIENTO.

SE EVITA QUE LA RED DEPENDA DEMASIADO DE CUALQUIER NEURONA EN PARTICULAR.

- SE PUEDE INTERPRETAR LA TÉCNICA DE DROPOUT COMO UN TIPO DE BAGGING QUE ENSAMBLA MODELOS MÁS PEQUEÑOS PERO QUE COMPARTEN SUS PARÁMETROS (ESTOS MODELOS NO SON INDEPENDIENTES)

- DIFERENTES COMBINACIONES DE NEURONAS ESTÁN ACTIVAS EN DIFERENTES ITERACIONES → LAS NEURONAS PRENDIDAS DEBEN APRENDER MEJOR PARA COMPENSAR LA LABOR DE LAS AUSENTES.

- SE PUEDE UTILIZAR LA MEDIA GEOMÉTRICA, EN LUGAR DE LA MEDIA ARITMÉTICA (PROMEDIO) PARA PREDICIR.

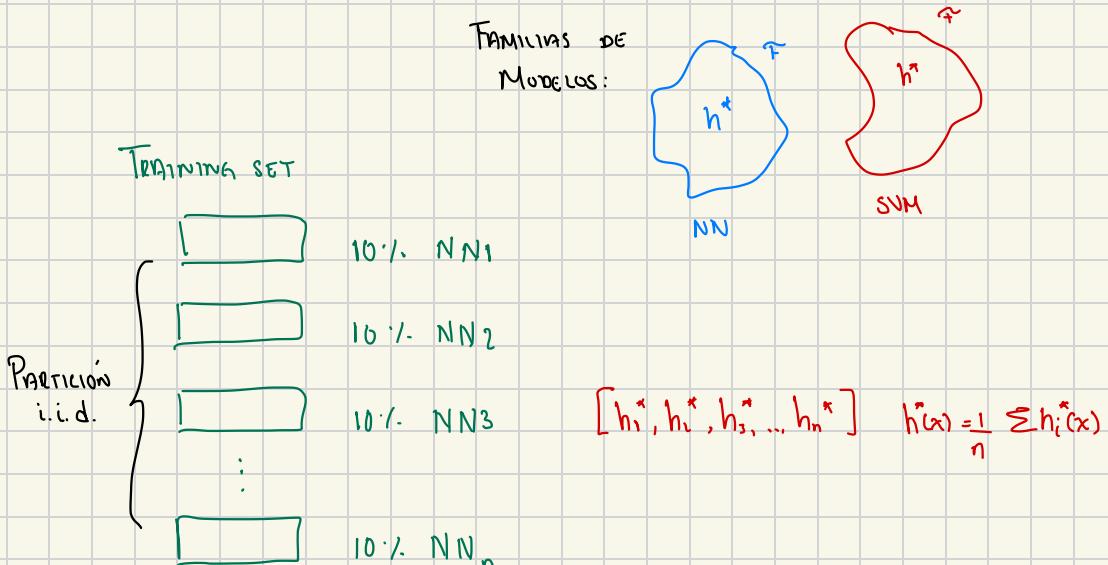
$$\frac{1}{M} \sum_{i=1}^M a_i \geq \left(\prod_{i=1}^M a_i \right)^{1/M}$$

LA MEDIA GEOMÉTRICA SE UTILIZA EN LUGAR DEL PROMEDIO CUANDO NOS INTERESA OBTENER UNA MEDIA QUE REFLEJE EL CRECIMIENTO O VARIACIÓN RELATIVA DE LOS DATOS, COMO TASAS DE CRECIMIENTO O RENDIMIENTO COMPLETO EN EL TIEMPO. LA MULTIPLICACIÓN PERMITE TENER EN CUENTA LA MAGNITUD DEL CRECIMIENTO, LA SUMA PUEDE VERSE AFECTADA POR VALORES ATÍPICOS.

DROPOUT ES UNA TÉCNICA DE REGULARIZACIÓN BARATA COMPUTACIONALMENTE QUE "SIMULA" BAGGING.

BAGGING.

MÉSMA DE MODELOS / MODELOS DE ENSEMBLE.



PREDICCIÓN FINAL:

CONFORME AUMENTA EL NÚMERO DE MODELOS, LA VARIANZA SE APROXIMA A CERO \rightarrow LAS VARIABLES ALATORIAS SE EMPLEAN A VOLVER DETERMINÍSTICAS.

1. Clasificación \rightarrow VOTACIÓN
2. Regresión \rightarrow PROMEDIO

TESTING:

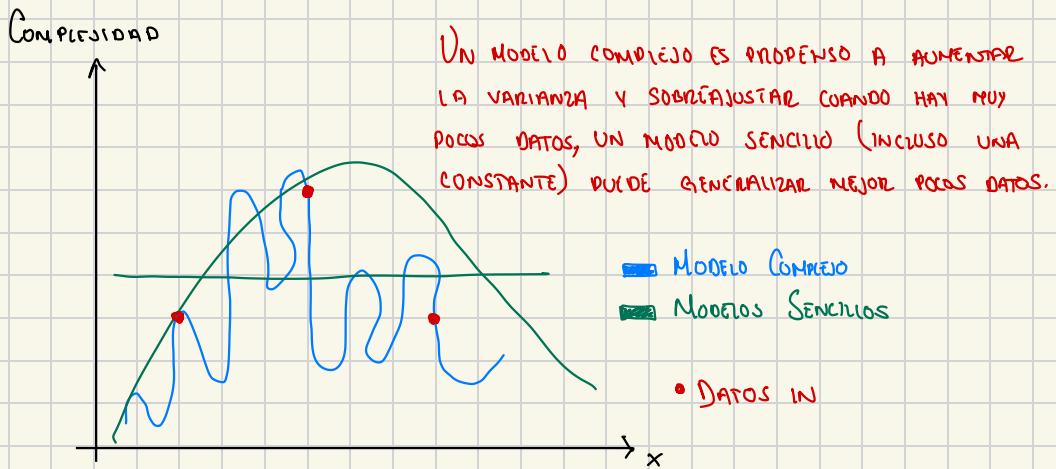
DURANTE EL TESTING, SE DESACTIVA DROPOUT Y SE AJUSTAN LOS VALORES DE ACTIVACIÓN PARA COMPENSAR LOS VALORES USADOS DURANTE EL ENTRENAMIENTO, ASSEGURANDO QUE LA SALIDA FINAL SEA EQUIVALENTE A LA MEDIA DE LAS SUBSEDES ENTRENADAS.

- ESCALADO DURANTE TESTING: SI SE USÓ UN DROPOUT DEL 50% (CADA NEURONA TENÍA UN 50% DE PROBABILIDAD DE ESTAR PRENDIDA EN EL ENTRENAMIENTO), ENTONCES EN EL MOMENTO DE LA EVALUACIÓN SE MULTIPLICAN TODAS LAS NEURONAS POR 0.5.

Warnings de Dropout.

1. DATASETS/ MODELOS PEQUEÑOS: Dropout podría disminuir la capacidad del modelo al perder información crucial.
2. LA TÉCNICA DE DROPOUT PUEDE PROVOCAR QUE LOS MODELOS CONVERGAN MÁS LENTO.

TRANE-OFF:



- LA TÉCNICA DE DROPOUT GENERA "SUBMODELOS" MÁS SENCILLOS Y PERMITE QUE AUNQUE TENGAMOS POCOS DATOS, LOS MODELOS NO "SE JUDGUEN LOCOS" SOBREALAJUSTANDO. DE ESTA FORMA, CADA INSTANCIA DE DROPOUT FUNCIONA MEJOR CON EL BATCH DE DATOS DISPONIBLE Y LUEGO SE SIMULA UN "ENSABLE" USANDO LA ARQUITECTURA COMPLETA

NOTAS DE DROPOUT.

- VUELVE AL MODELO ESTOCÁSTICO.
 - FOMENTA QUE EL MODELO NO SE ADAPTE A LAS FEATURES DISPONIBLES.
 - EVITA QUE HAYA CO-ADAPTACIÓN DEL MODELO A CIERTAS FEATURES.
 - DIFÍCILTA EL APRENDIZAJE DURANTE EL APRENDIZAJE PARA FORZAR UNA MEJOR GENERALIZACIÓN.
- Ayuda a regularizar modelos grandes (No es recomendado con modelos pequeños o poco datos → complejidad baja)

DROPOUT ES UNA TÉCNICA QUE HACE QUE UNA RED MUY GRANDE ACTÚE COMO UNA PEQUEÑA CON UN ENTRENAMIENTO ALEATORIO DE SUBSECCIONES DE LA RED Y, REDES PEQUEÑAS NO SE SOBREAJUSTAN.

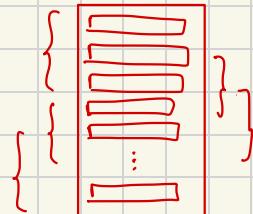
- Al apagar ciertas neuronas evitamos que la red aprenda el mismo ruido en cada entrenamiento.

BAGGING

f_1, \dots, f_n MODELOS DIFERENTES QUE TRATAN DE APROXIMAR los DATOS.



LOS MODELOS SE ENSAMBLAN $\hat{f} = \frac{1}{N} \sum_i f_i$



- ENTRE MÁS MODELOS INDEPENDIENTES USEMOS, MÁS PODEMOS REDUCIR LA VARIANZA EN EL ERROR DE PREDICCIÓN.

SE ENTRENAN CADA
MODELO CON UN
SUBCONJUNTO ALEATORIO
DEL DATASET COMPLETO.

BOOSTING

BATCH / LAYER NORMALIZATION.

NORMALIZACIÓN: AJUSTE DE LOS DATOS A UNA ESCALA COMPARABLE.

LOS ALGORITMOS DE APRENDIZAJE SUELEN SER SENSIBLES A LAS DIFERENCIAS DE MAGNITUDES DE LAS VARIABLES.

NORMALIZACIÓN DE LAS ENTRADAS:

PERMITE TENER CONTROL SOBRE LOS PESOS AL CONTROLAR LA ESCALA DE LOS DATOS Y MOVER LA DIRECCIÓN DE LA OPTIMIZACIÓN A LA DIRECCIÓN CORRECTA DURANTE EL ENTRENAMIENTO.

LOS ALGORITMOS DE APRENDIZAJE CONVERGEN MÁS RÁPIDO CUANDO LOS DATOS ESTÁN EN UNA ESCALA SIMILAR. DE LO CONTRARIO, PODRIAMOS OBTENER GRADIENTES GRANDES EN DIRECCIÓN A LAS VARIABLES CON MAGNITUDES MÁS GRANDES. CUANDO NORMALIZAMOS LE AYUDAMOS AL ALGORITMO DE APRENDIZAJE A OPTIMIZAR MEJOR LA APROXIMACIÓN CALCULADA YA QUE LA FUNCIÓN RECIBE DATOS QUE HOMOLOGAN LOS GRADIENTES PERMITIENDO QUE EL PROBLEMA DE OPTIMIZACIÓN TENGA INFORMACIÓN ESTANDARIZADA.

LAS VARIABLES DE MAYOR MAGNITUD PODRÍAN AFECTAR MÁS A LA FUNCIÓN DE OBJETIVO (DOMINANCIA). AL NORMALIZAR PROVOCAMOS UN PESO EQUITATIVO PARA TODAS LAS VARIABLES.

OVERFITTING: EL MODELO PUEDE AJUSTARSE DEMASIADO A LAS VARIACIONES EN LAS VARIABLES DE MAYOR ESCALA.

NORMALIZACIÓN:

- MEDIDA CERO
- DESVIACIÓN ESTÁNDAR DE 1.

PARA CADA VARIABLE x_i $z_i = \frac{x_i - \mu_i}{\sigma_i}$

μ_i = MEDIA DE LA VARIABLE ORIGINAL

σ_i = DESVIACIÓN ESTÁNDAR DE LA VARIABLE ORIGINAL.

BATCH Normalization:

PROPIUESTA PARA ABORDAR PROBLEMAS DE CONVERGENCIA Y ENTRENAMIENTO.

INTUICIÓN: SI NORMALIZAR LOS FEATURES O ATRIBUTOS DE LOS DIFERENTES DE ENTRADA AyUDA A QUE EL COMPORTAMIENTO DE LA PRIMERA CAPA FUNCIONE MEJOR, ENTONCES APLICAR NORMALIZACIÓN A LAS ACTIVACIONES DE LAS CAPAS INTERNAZ TAMBÍEN PODRÍA AyUDAR DURANTE EL ENTRENAMIENTO.

PROPIUESTA: SE APLICA A LA SALIDA DE UNA CAPA ANTES DE ENTRAR A LA SIGUIENTE → NORMALIZAR LAS ACTIVACIONES DE CADA NEURONA EN UNA CAPA CON LAS ESTADÍSTICAS DE UN BATCH.

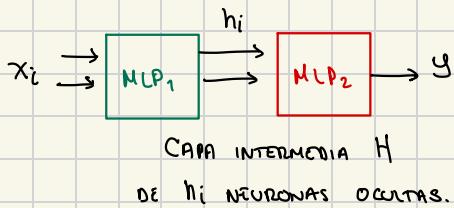
INTENCIÓN: REDUCIR EL FENÓMENO "INTERNAL COVARIATE SHIFT" QUE DESCRIBE CÓMO UNA CAPA PROFUNDA PUEDE SER AFECTADA POR CAMBIOS PEQUEÑOS DE LAS ACTIVACIONES DE CAPAS INICIALES EN LA RED.

INTERNAL Covariate Shift

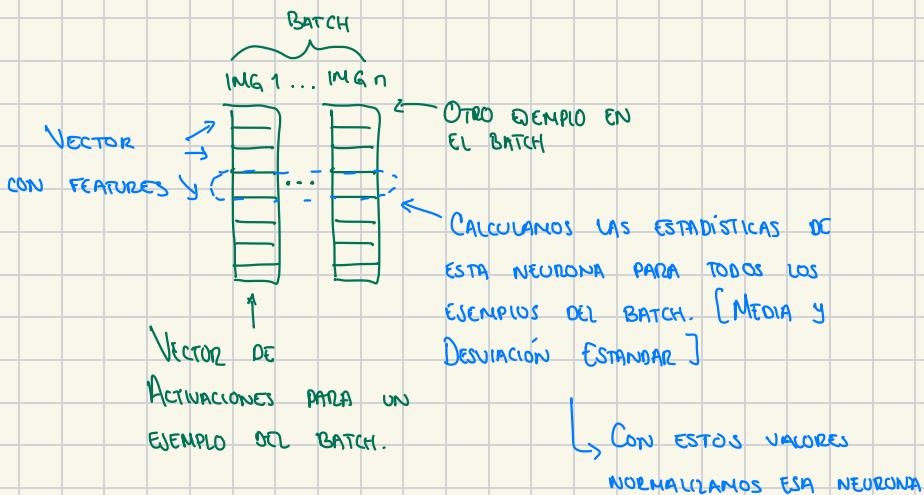
Conforme se hace el entrenamiento, la distribución de las capas intermedias sigue cambiando por efecto de la actualización de los pesos.

- PERMITE EL USO DE TASAS DE APRENDIZAJE MÁS ALTA, PREFERANDO LA CONVERGENCIA.
- ESTABILIZAN LAS DISTRIBUCIONES DE LAS ACTIVACIONES.

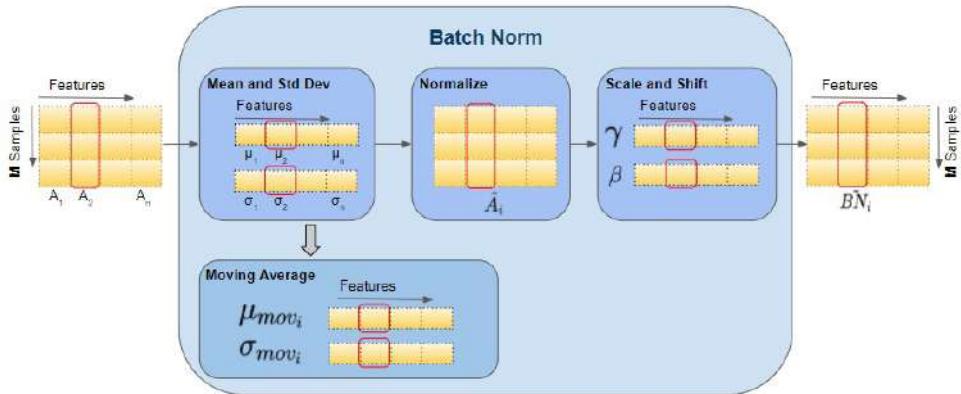
EN DEEP LEARNING, TENEMOS COMPOSICIONES DE MODELOS EN CAPAS. ENTonces, NO SOLO NORMALIZAMOS LA ENTRADA PRINCIPAL DE LOS DATOS SI NO QUE QUEREMOS INCLUIR BLOQUES DE NORMALIZACIÓN PARA NORMALIZAR LAS ENTRADAS INTERNAS DE CADA CAPA.



EFFECTO DE REGULARIZACIÓN: ACOTAMOS A LA FUNCIÓN PARA QUE ELUJA SOLO PARÁMETROS QUE SE ALINEEN A UNA ESCALA DADA (NORMALIZADA).



EFFECTO DE REGULARIZACIÓN: AL ENTRENAR CON MEDIA Y VARIANZA DIFERENTE EN CADA LOTE, LA RED EVITA SOBREAJUSTARSE A UN SOLO CONJUNTO DE VALORES.



1. OBTENEMOS LAS ESTADÍSTICAS DE $A_2 \rightarrow \mu_2$
 $\rightarrow \sigma_2$

2. NORMALIZAMOS CADA ELEMENTO DEL BATCH CON ESOS VALORES

→ Transformación a media 0 y desviación estándar 1.

3. Escalado y Desplazamiento (γ y β)

- AJUSTE DE VALORES NORMALIZADOS.
- PARÁMETROS APRENDIDOS.
- LOS VALORES NORMALIZADOS SE AJUSTAN PARA NO PERDER LA EXPRESIVIDAD DE LOS DATOS.

4. MOVING AVERAGE.

- CÁLCULO QUE UTILIZA LOS VALORES ESTADÍSTICOS ACTUALES Y PASADOS PONDERADOS PARA CADA FEATURE.
- SE UTILIZAN PARA APLICAR UNA NORMALIZACIÓN COHESIVA CON LA DISTRIBUCIÓN OBSERVADA DURANTE EL ENTRENAMIENTO, A LOS DATOS DE PRUEBA / VALIDACIÓN.

$H \rightarrow$ BATCH ORIGINAL PARA ENTRENAR (SUBCONJUNTO DEL DATASET)
 $H' \rightarrow$ BATCH NORMALIZADO

- EXISTEN 2 PARÁMETROS γ Y β QUE DURANTE EL ENTRENAMIENTO SON APRENDIDOS POR EL MODELO Y QUE NOS AYUDAN A CONTINUAR EL GRADO DE NORMALIZACIÓN MÁS APROPIADO PARA NUESTROS DATOS.

$$H^N = \gamma H' + \beta$$

\nearrow ↘

VARIANZA APRENENDIDA BIAS APRENDEDOR

- EN EL TESTING SE USAN LOS VALORES DE NORMALIZACIÓN APRENIDOS.

Si durante el entrenamiento γ alcanza un valor de 0 y β el valor M , obtendríamos una capa con todas las actuaciones normalizadas (usualmente aprenden valores intermedios).

Al normalizar se facilita el problema de optimización pero con γ y β buscando recuperar la expresividad del modelo aprendiendo momentos de la distribución que eran importantes.

- BATCH NORMALIZATION TAMBÍEN APORTA CONTROL SOBRE EL LEARNING RATE, AUNQUE ESTE DÉ PASOS GRANDES, AÚN ASÍ IGUAL APRENDER (MINIMIZAR EL ERROR) Y LOS CAMBIOS EN LA TASA DE APRENDIZAJE TAMBIÉN PUEDEN SER MÁS GRANDES (ACELERA LA VELOCIDAD DEL LEARNING RATE).

LOS VALORES ESTADÍSTICOS SE CALCULAN POR MINIBATCHES QUE SON LOS SEGMENTOS EN LOS QUE DIVIDIMOS EL CONJUNTO DE ENTRENAMIENTO DE FORMA ALEATORIA.

CONTROVERSIAS DE LA TÉCNICA:

- QUE LOS EJEMPLOS SEAN DISTINTOS EN CADA ÉPOCA Y QUE CON ELLOS SE CALCULE LA NORMALIZACIÓN, MEDE RUIDO.
- EL RUIDO DURANTE EL ENTRENAMIENTO PUEDE FUNCIONAR COMO UN FACTOR DE REGULARIZACIÓN.

↳ ESTO SUCDE PORQUE EL RUIDO IMPIDE QUE LAS NEURONAS MEMOTICEN INFORMACIÓN.

¿QUÉ VALORES DE α Y β USAMOS PARA HACER LAS PREDICCIONES? (PARA USAR EN TESTING?)

EN TESTING NO TENEMOS BATCHES, HACEMOS PREDICCIONES UNITARIAS.

EN LUGAR DE NORMALIZAR POR BATCH, APLICAMOS UNA TÉCNICA DE MOVING AVERAGE DE LAS MEDIAS ESTIMADAS.

$$\hat{\mu}_{\text{NEW}} = (1 - \alpha) \hat{\mu}_{\text{OLD}} + \alpha \hat{\mu}_t \quad \alpha \in (0, 1)$$

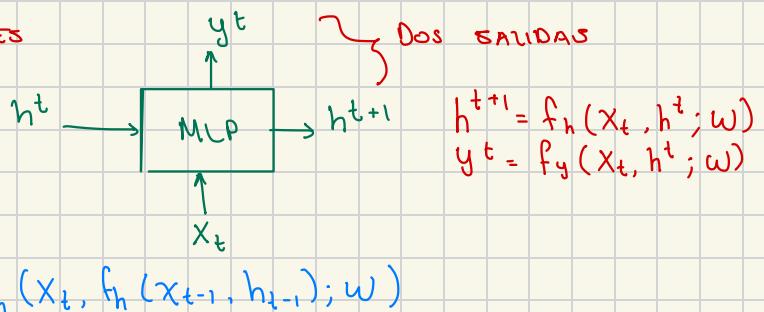
(PROMEDIO EXPONENCIAL PONDERADO)

NUEVO ARGUMENTO: Batch Normalization NO FUNCIONA POR REDUCIR EL FENÓMENO DE Cumulative Shift, SI NO PORQUE TIENE UN EFECTO DE SMOOTH EN EL ESPACIO DE PÉRDIDAS.

- BATCH NORMALIZATION NO SE APLICA A REDES RECURRENTES, EN SU LUGAR SE PUEDE APLICAR LAYER NORMALIZATION.

REDES RECURRENTES

RECURRENCIA:



LA ARQUITECTURA MLP COMPARTE LOS PARÁMETROS PARA TODOS LOS ELEMENTOS DE LA SECUENCIA EN EL TIEMPO.

- LA PRINCIPAL DIFICULTAD RADICA EN CÓMO PODEMOS IR ACTUALIZANDO EL MOVING AVERAGE A LO LARGO DE LA SECUENCIA.

↳ MANTIENE LAS MISMAS ESTADÍSTICAS DURANTE TODO EL TIEMPO.

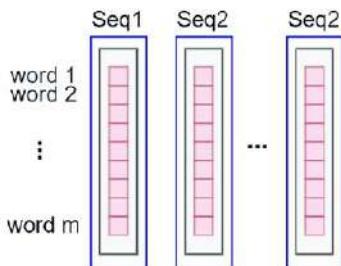
BATCH NORMALIZATION DEPENDE DEL TAMAÑO DEL LOTE, EN EL PROCESAMIENTO DE SECUENCIAS EL TAMAÑO DEL LOTE ES REDUCIDO.

Layer Normalization

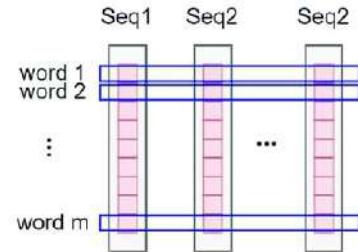
NORMALIZAR USANDO TODOS LOS VALORES QUE TONAN LAS NEURONAS DE TODA LA CAPA PARA UNA MUESTRA INDIVIDUAL.

OPERA A NIVEL DE TODAS LAS CARACTERÍSTICAS DENTRO DE CADA EJEMPLO EN Lugar DE HACER LOS CÁLCULOS PARA UNA SOLA CARACTERÍSTICA.

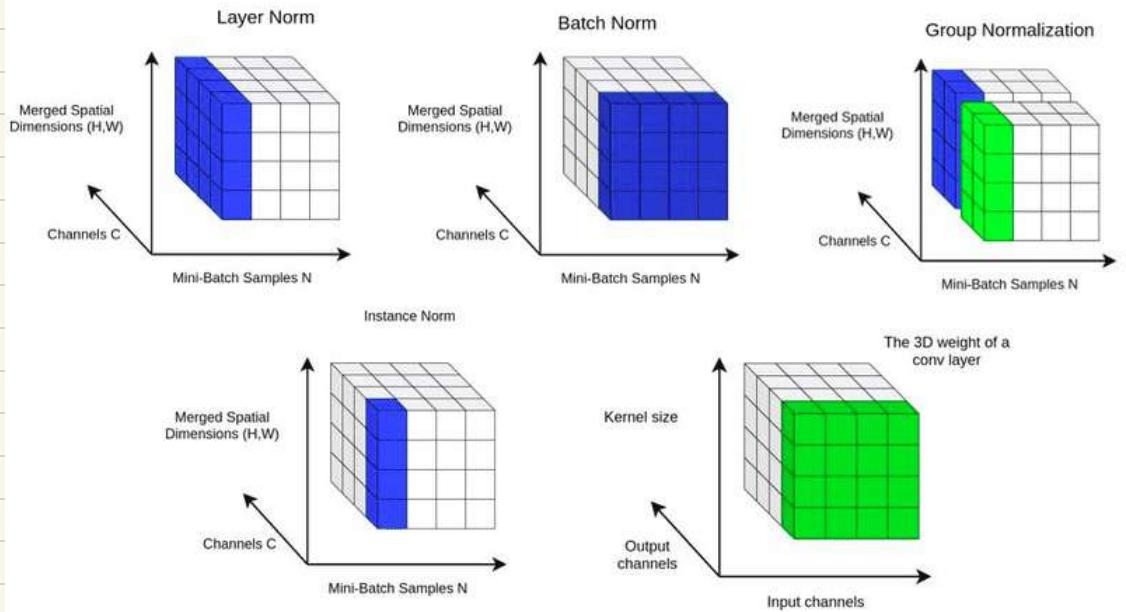
- FUNCIONA INCLUSO PARA LOTES DE TAMAÑO 1, POR LO QUE SE PUEDE APLICAR INCLUSO A REDES RECURRENTES O SECUENCIAS.



Layer Normalization (LN)



Batch Normalization (BN)



Weight Decay (Regularización L2)

- FUNCIONA PENALIZANDO LOS PESOS DURANTE LA OPTIMIZACIÓN, AÑADIENDO UN VALOR DE PENALIZACIÓN AL VALOR DE LA PÉRDIDA, BASADO EN LA MAGNITUD DE LOS PESOS.

$$L_2 = l_{\text{ORIGINAL}} + \frac{\lambda}{2} \sum_i w_i^2$$

$\lambda \rightarrow$ HIPERPARÁMETRO QUE CONTROLA LA INTENSIDAD DE LA PENALIZACIÓN.

↳ TRATAMOS DE MINIMIZAR LA MAGNITUD DE LOS PESOS PARA QUE TIENDAN A VALORES PEQUEÑOS.

- SE EVITA QUE EL MODELO DEPENDA EXCESIVAMENTE DE CUALQUIER CONEXIÓN O PESO EN PARTICULAR.

REGULARIZACIÓN L1

SIMILAR A L2 PERO APLICADA A LA NORMA DE LOS PESOS.

$$L_1 = l_{\text{ORIGINAL}} + \lambda \sum_i |w_i|$$

COMPLEJIDAD DE TÉCNICAS DE REGULARIZACIÓN:

- SKIP CONNECTIONS: SOLO SE AÑADE UNA OPERACIÓN DE SUMA DE VECTORES $O(n)$, $n = \text{NÚMERO DE NEURONAS CON CONEXIÓN RESIDUAL}$.
- BATCH NORMALIZATION: COMPLEJIDAD ASOCIADA AL CÁLCULO DE MEDIAS Y VARIANZAS DE CADA MINILOTE $O(N \cdot N)$ DONDE N ES EN NÚMERO DE NEURONAS Y M EL TAMAÑO DEL MINILOTE.
- LAYER NORMALIZATION: COMPLEJIDAD $O(d)$ PARA CADA CAPA DONDE d ES EL NÚMERO DE NEURONAS EN LA CAPA
- WEIGHT DECAY: $O(N)$ N ES EL NÚMERO DE PESOS DE LA RED.
- DROPOUT: AUMENTO DE LA COMPLEJIDAD DEBIDO A LAS MÁSCARAS DE Dropout $O(N)$

ATENCIÓN

MECANISMOS DE ATENCIÓN

- PROCESO SIMILAR AL QUE SIGUE UNA MEMORIA O UN DICCIONARIO.
A TRAVÉS DE LLAVES Y VALORES SE HACEN CONSULTAS.
- UTILIZANDO KEYS Y VALORES ESTAMOS ESQUEMATIZANDO DE UNA FORMA DONDE SEPARAMOS EL CONTENIDO (VALUES) DE LA FORMA EN LA QUE ACCEDEMOS AL CONTENIDO, DESACOPLANDO LA INFORMACIÓN PARA PODER CREAR LLAVES ROBUSTAS SIN DEPENDER DEL CONTENIDO (POR EJEMPLO, ROBUSTAS A RUIDO).
- EL MODELO DECIDE DINÁMICAMENTE QUÉ VALOR USAR BASADO EN UNA DISTANCIA DE SIMILITUD ENTRE QUÉ TANTO SE PARECE EL VECTOR DE QUERY AL CONJUNTO DE M KEYS. SI q ES MÁS SIMILAR A LA KEY i , ENTONCES SELECCIONAMOS EL VALOR v_i .

$$\text{ATTENTION } (q, (k_1, v_1), (k_2, v_2), \dots, (k_n, v_n))$$

\uparrow $\underbrace{\hspace{10em}}$
CONSULTA DICCIONARIO

$$= \sum_{i=1}^n \alpha_i (q, \{k_j\}) v_i = 1$$

LA SUMA PONDERADA
ES IGUAL A 1.

α_i ES UN PESO DE ATENCIÓN ENTRE 0 Y 1 → DISTRIBUCIÓN DE PROBABILIDAD.

SE UTILIZA SOFTMAX PARA DAR UN VALOR CERCANO A 1 A LA KEY MÁS SIMILAR Y UN VALOR CERCANO A 0 A TODAS LAS DEMÁS.

LA FUNCIÓN DE ATENCIÓN ES UNA FUNCIÓN DE SIMILITUD ENTRE EL QUERY Y LAS DIFERENTES KEYS.

FUNCIONES DE ATENCIÓN (SIMILITUD):

DADA UNA SECUENCIA DE ENTRADA, NOS INTERESA SER CAPACES DE PODER USAR EN CADA PASO LOS ELEMENTOS QUE SEAN RELEVANTES PARA EL CÁLCULO ACTUAL, SIN IMPORTAR EN QUÉ POSICIÓN O QUÉ TAN LEJOS SE ENCUENTREN.

EXISTE UN DICTIONARIO QUE CONTIENE TODA LA INFORMACIÓN POSIBLE DEL VOCABULARIO Y LA INTENCIÓN ES SER CAPACES DE ACCEDER A CUALQUIER ELEMENTO EN CUALQUIER MOMENTO.

OBJETIVO: EL OBJETIVO DEL MODELO ES APRENDER LA FUNCIÓN DE DISTANCIA ENTRE LOS "KEYS" Y LOS QUERIES USANDO SUS PARÁMETROS W .

→ LA ATENCIÓN DEPENDE DEL PROBLEMA, AUNQUE EL VOCABULARIO SEA EL MISMO.

LA FUNCIÓN DE SIMILITUD NOS DICE QUÉ TANTO SE PARECE EL QUERY A CADA ELEMENTO DEL DICTIONARIO. ESTA INFORMACIÓN ES LA BASE DE LA REPRESENTACIÓN CONTEXTUALIZADA QUE OBTENEMOS DE LA SECUENCIA.

LA FUNCIÓN CALCULA LA IMPORTANCIA RELATIVA ENTRE PARES DE TOKENS.

SCALED DOT-PRODUCT ATTENTION

$$\text{ATTENTION}(Q, K, V) = \text{SOFTMAX} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

FUNCIÓN PARAMETRIZADA

$$a(q, k) = \frac{(W_q q)^T (W_k k)}{\sqrt{d}}$$

PESOS PARA UNA TRANSFORMACIÓN LINEAL (LOS PASANOS A OTRO ESPACIO)

LA ATENCIÓN ES ADAPTATIVA EN EL SENTIDO DE QUE CADA MODELO APRENDE QUÉ TOKENS SON RELEVANTES EN FUNCIÓN DEL CONTEXTO Y EL OBJETIVO DE LA TAREA QUE ESTEMOS BUSCANDO RESOLVER (TRADUCCIÓN, CLASIFICACIÓN, ETC.)

- TOKENS CON SIGNIFICADOS SIMILARES ESTÁN CERCA EN EL ESPACIO VECTORIAL QUE LAS REPRESENTA.
- TOKENS CON RELACIONES OPUESTAS O DIFERENTES ESTÁN MÁS ALEJADOS.

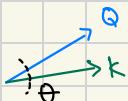
PRODUCTO PUNTO EQUIVALENTE PARA REPRESENTAR SIMILITUD.

- SIMILITUD CON DISTANCIA COSENO

- CON LAS MATRICES W, MOVEMOS LOS KEYS Y LOS VALUES A UN NUEVO ESPACIO DONDE VAMOS A MEDIR SIMILITUD.

- LA FUNCIÓN DE SIMILITUD BASADA EN COSENO TIENE LA VENTAJA DE QUE ES INVARIANTE A LA MAGNITUD DE LOS VECTORES.

→ El producto punto QK^T MIDE EL ÁNGULO ENTRE ELLOS
• Si el ángulo es PEQUEÑO, implica que SON SIMILARES o ESTÁN ALINEADOS, esto GENERA UN PRODUCTO PUNTO GRANDE e implica UNA ATENCIÓN ALTA.



¿CUÁNTO SE PARECE CADA ELEMENTO DE LA SECuencia A LOS DEMÁS?

- SIMILITUD ALTA = MISMO TOKEN \rightarrow SOFTMAX = 1
- SIMILITUD MODERADA = RELACIÓN, POR EJEMPLO RELACIÓN SEMÁNTICA ENTRE DOS PALABRAS "PERRO" y "GATO".
- EN EL CONTEXTO DE LENGUAJE NATURAL LO RELEVANTE ES QUÉ TAN ALINEADOS ESTÁN LOS VECTORES Y NO SU MAGNITUD (TAMAÑO), YA QUE LA DIRECCIÓN DEL VECTOR ES MÁS SIGNIFICATIVA PARA ENCONTRAR RELACIONES SEMÁNTICAS.

↳ LAS RELACIONES SEMÁNTICAS REPRESENTAN ASOCIACIONES DE SIGNIFICADO.

EJEM: "MAÑANA" y "DESPERTAR" PUEDEN ESTAR ASOCIADAS POR EL TIEMPO EN EL QUE OCURREN.

NORMALIZACIÓN DE LA FUNCIÓN DE ATENCIÓN

ESCALAMIENTO POR $\sqrt{d_k}$

- PERMITE EVITAR VALORES EXTREMAMENTE GRANDES O PEQUEÑOS, LO QUE PODRÍA TENER UN EFECTO DE Saturación EN LA FUNCIÓN DE SOFTMAX.

↳ Si el producto punto es muy grande, softmax tenderá a generar probabilidades extremas, generando un efecto de Pérdida de sensibilidad en las relaciones.

- GARANTIZA QUE LAS DISTRIBUCIONES DE ATENCIÓN SEAN MÁS SIGNIFICATIVAS.

Concepto de Query

EJEMPLO: SECUENCIA "EL PERRO PERSIGUE AL GATO"

Q ES UNA MATRIZ DE DIMENSIONES $n \times d_k \rightarrow Q \in \mathbb{R}^{n \times d_k}$

- Es una representación vectorial de las consultas de cada parte de la secuencia (TOKEN)

↳ n TOKENS (FILAS)

d_k TAMAÑO EN QUE SE CODIFICA CADA TOKEN (COLUMNAS)

LOS PESOS W_Q APRENDIDOS PARA Q SE REALIZAN CON EL OBJETIVO DE OBTENER PESOS QUE ENFATICEN CARACTERÍSTICAS ESPECÍFICAS DE CADA TOKEN SEGÚN LO QUE EL MODELO ESTÁ "PREGUNTANDO".

CONCEPTO DE Keys y Values

- TANTO LAS MATRICES K Y V , COMO LA MATRIZ DE QUERIES Q PASAN POR UN PROCESO DE PARAMETRIZACIÓN A TRAVÉS DE UNA TRANSFORMACIÓN LINEAL REALIZADA POR LAS MATRICES W_K , W_V Y W_Q QUE DESPUÉS DEL PROCESO DE ENTRENAMIENTO, TENDRÁN LOS PESOS OPTIMIZADOS PARA CADA TAREA EN PARTICULAR.

↳ LAS TRANSFORMACIONES SON INDEPENDIENTES Y APRENDIDAS.

EMBEDDING

DADO EL VOCABULARIO DEL PROBLEMA, GENERAMOS UNA CODIFICACIÓN PARA CADA ELEMENTO O SÍMBOLO DE LA SECUENCIA. EN UNA SECUENCIA, CADA UNA DE SUS PARTES SE CONOCE COMO "TOKEN".

- EL PROCESO DE EMBEDDING CONSISTE EN UNA TRANSFORMACIÓN DE LAS ENTRADAS A NÚMEROS.

MATRIZ DE ATENCIÓN

Es el resultado que guía en términos de probabilidades, cuánta atención le debe poner cada parte de la secuencia a las demás con base en la similitud adquirida.

- CAPTURA DEPENDENCIAS A CORTO Y LARGO PLAZO.

Modelos Transformers.

LA ARQUITECTURA PROUESTA EN EL ARTICULO "ATTENTION IS ALL YOU NEED"
SE BASA EN UN MECANISMO DE ATENCIÓN PARA PROCESAR DATOS SECUENCIALES
COMUESTO POR DOS PARTES PRINCIPALES:

- ENCODER → TRANSFORMA LA ENTRADA EN UNA REPRESENTACIÓN INTERNA.
- DECODER → GENERA LA SALIDA COMO UNA SECUENCIA TRADUCIDA.

CADA UNA ESTÁ COMUESTA POR STACKS QUE CONTIENEN DOS SUBCAPAS:

1. CAPA MultiHead ATTENTION
2. CAPA FEED FORWARD NN

COMPLEMENTOS (REGULARIZACIÓN)

- LAYER NORMALIZATION
- CONEXIÓN RESIDUAL

APLICACIONES

INICIALMENTE UTILIZADOS EN EL CONTEXTO DE LENGUAJE NATURAL.

PRECCIONES PARA EXPLICAR EL MODELO DE ATENCIÓN.

- ¿POR QUÉ LOS TRANSFORMADORES USAN MECANISMOS DE ATENCIÓN?

- PARA INCLUIR CONTEXTO

- EN RNN SE PERDÍAN RELACIONES CON UNA DISTANCIA.

↳ TENER ACCESO A LAS COSAS IMPORTANTES SIN IMPORTAR A QUÉ DISTANCIA ESTÉ.

ARQUITECTURA.

- ATENCIÓN.

KEY: VALUES → MEMORIA

QUERY → BÚSQUEDA

VALUES: CONTENIDO

KEY: FORMA DE ACCEDER AL CONTENIDO

LOS KEYS SE USAN PARA DESACOPLAR LA INFORMACIÓN DEL CONTENIDO.

FUNCIÓN DE SIMILITUD $\alpha(\cdot)$ ENTRE LA QUERY Y LAS KEYS
SE APRENDE POR EL MODELO DE ATENCIÓN PUES NO DEPENDEN
DEL PROBLEMA (SON FUNCIONES PARAMETRIZADAS)

1. Additive Attention

2. Scaled Dot Product Attention.

- distancia $\cos(\theta)$ ENTRE VECTORES.

TRANSFORMADORES.

Enfocarse en una parte del diccionario (una memoria) dado una consulta

Paralelización: A diferencia de las Redes Recurrentes que son secuenciales, los Transformadores pueden pasar todo el sistema en paralelo (al menos en el entrenamiento)

Softmax($QK'/\sqrt{d_k}$) la salida es una matriz con una distribución de probabilidad en cada renglón

Multihead

- Tener múltiples cabezas nos permite poner atención a diferentes cosas de acuerdo a los parámetros de cada una.
- Como en las CNN, que cada filtro aprende una característica, acá cada cabeza aprende una atención particular

MULTI HEAD ATTENTION

PROPIEDAD DE EQUIVARIANCIA EN UN MECANISMO DE ATENCIÓN.

- UNA PERMUTACIÓN DE LAS ENTRADAS, GENERA UNA PERMUTACIÓN EN LAS SALIDAS.

Posicional Encoding

- ASIGNAR UN VECTOR QUE CODIFIQUE LA POSICIÓN A LOS VECTORES DE ENTRADA PARA DARLE INFORMACIÓN SOBRE LA SECUENCIA.

Pares → Función Seno }
Impares → Función Coseno } PERMITEN RECUPERAR LAS POSICIONES ORIGINALES

Encoder - Decoder

Build A Large Language Model - Sebastian

ENTRENAMIENTO.

TEACHER FORCING

MECANISMOS DE ATENCIÓN.

SELF-ATTENTION: DAR DIFERENTES NIVELES DE ATENCIÓN A CADA PARTE DE LA ENTRADA.

CAPACIDAD DEL MODELO PARA CAPTURAR RELACIONES COMPLEJAS EN DATOS SECUENCIALES ASIGNANDO A LA RED PESOS DE ATENCIÓN DE MANERA ADAPTATIVA A DIFERENTES PARTES DE LA ENTRADA.

CADA PALABRA (ELEMENTO DE LA ENTRADA) TOMA 3 REPRESENTACIONES:

- CONSULTAS Q
 - LLAVE K
 - VALOR V
- } APRENDIDAS DURANTE EL ENTRENAMIENTO.

| INTERPRETACIÓN DE MATRICES:

Q = ELEMENTOS DE LA SECUENCIA QUE ESTAMOS COMPARANDO.

K = REPRESENTACIÓN DE LAS POSICIONES DE LAS PALABRAS EN LA SECUENCIA.

V = INFORMACIÓN REAL DE CADA ELEMENTO DE LA SECUENCIA QUE SE VA A PONDERAR.

CÁLCULO DE ATENCIÓN

$$\text{ATENCIÓN}_i = \text{SOFTMAX} \left(\frac{\mathbf{Q}_i \cdot \mathbf{K}_i^T}{\sqrt{d_k}} \right) \cdot \mathbf{V}_i$$

INDEPENDIENTES → PARALELIZACIÓN NORMALIZACIÓN
 DESVIACIÓN ESTÁNDAR.
 (d_k ES LA VARIANZA)

COMPARACIÓN DE LA CONSULTA DE UNA PALABRA CON LAS CLAVES DE TODAS LAS DEMÁS PALABRAS EN LA SECUENCIA.

d_k = DIMENSIÓN DE LAS CONSULTAS Y CLAVES.

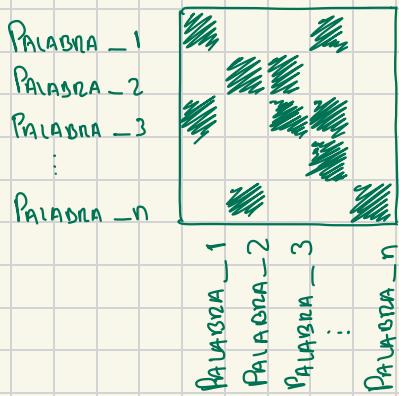
SECUENCIA DE SALIDA PONDERADA: PESOS QUE DETERMINAN LA CONTRIBUCIÓN DE CADA ELEMENTO DE LA SECUENCIA.

$$\text{SALIDA} = \text{ATENCIÓN} \cdot \mathbf{V}$$

MISMA SECUENCIA DE SALIDA PERO CON PESOS QUE DAN ÉNFASIS EN PARTES ESPECÍFICAS.

MATRIZ DE ATENCIÓN:

INDICA CUÁNTA ATENCIÓN SE DEBE PONER CADA ELEMENTO DE LA SECUENCIA A LOS OTROS.



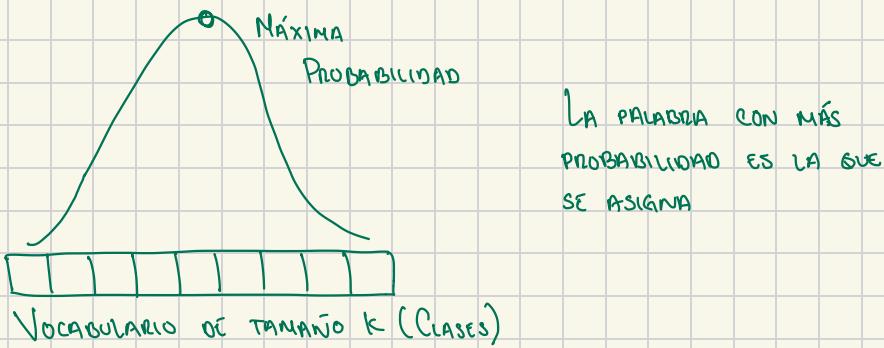
- AL PRINCIPIO, LOS VALORES DE LAS MATRICES Q , K , V PASAN POR UN PROCESO DE INICIALIZACIÓN A TRAVÉS DE LAS REPRESENTACIONES ENBEDIDAS.
- POSTERIORMENTE SE GENERAN LAS SALIDAS CON LOS PESOS ENCONTRADOS Y SE CALCULA UNA FUNCIÓN DE PÉRDIDA CON LAS SALIDAS Y LOS VALORES ESPERADOS.
- SE AJUSTAN LOS PARÁMETROS, INCLUYENDO LOS VALORES DE LAS MATRICES Q , K , V . (LA RED APRENDE A GENERAR REPRESENTACIONES MÁS ÚTILES) TANTO PARA EL ENCODER COMO EL DECODER.

DIMENSIÓN d_k = HIPERPARÁMETRO QUE SE USA PARA ASIGNAR EL TAMAÑO A LOS VECTORES DE EMBEDDING Y OTRAS REPRESENTACIONES EN EL MODELO.

$$K \approx 256 \text{ A } 1024$$

SOFTMAX: OBTENER UNA DISTRIBUCIÓN DE PROBABILIDAD COMO SALIDA.

FUNCIÓN QUE DEVUELVE UN CONJUNTO DE PROBABILIDADES SOBRE TODO EL UNIVERSO DE PALABRAS DISPONIBLES.



LA PALABRA CON MÁS PROBABILIDAD ES LA QUE SE ASIGNA

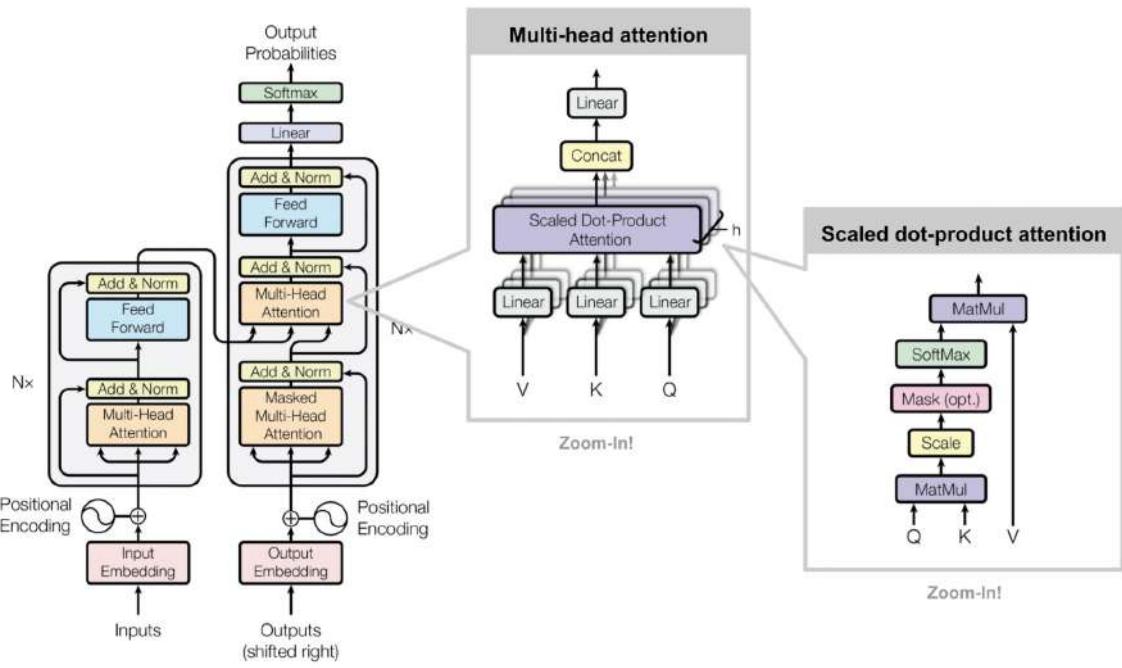
$k =$ TAMAÑO DEL VOCABULARIO, P.EJ. 10,000 PALABRAS.

- SE ASIGNA LA PALABRA
- SE MIDE EL ERROR CON LA PALABRA ESPERADA.
- SE PROPAGA GRADIENTE DESCENDENTE HACIA ATRÁS.

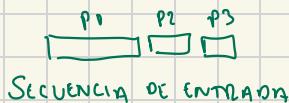
" d " = DIMENSIÓN DE LOS VECTORES EMBEBIDOS.

" s " = TAMAÑO DE LA SECUENCIA DE ENTRADA.

- LAS MATRICES Q, K, V CONTIENEN LOS VECTORES EMBEBIDOS DE LA SECUENCIA DE TAMAÑO " s ".



CADA CAPA TIENE UNA CONCAGENACIÓN
Y UNA NORMALIZACIÓN, ASÍ COMO
CONEXIONES RESIDUALES.



→ Encoder: p_1 APRENDE QUE TIENE ESTRECHA RELACIÓN CON p_3 (p_1 ESTÁ CONTEXTUALIZADA)



El decodificador USA LA SECUENCIA CONTEXTUALIZADA + EL INICIO + s_1 PARA PREDICIR s_2 (SEGUNDO ELEMENTO DE LA SALIDA).

EJEMPLO:

$p_1, p_2, p_3 \in$ PALABRAS EN ESPAÑOL

s_1, s_2, \in TRADUCCIÓN A INGLÉS

Model Architecture:

ESTRUCTURA: Encoder - Decoder.

Encoder: Procesa una secuencia de entrada con símbolos a una representación continua contextualizada de cada palabra en función del contexto global.

1. Capa Multiheads Attention:

- Cada cabeza de atención aprende diferentes representaciones ponderadas de la entrada como verbos, nouns, sujetos, etc.
- La entrada se proyecta en h entradas de atención.
- Las salidas de las cabezas se concatenan para obtener una salida final.
- Cada cabeza se especializa en capturar patrones específicos.

2. Capa Feed Forward:

- Transformaciones lineales y no lineales para aprender patrones más complejos en la secuencia.
→ Conexiones residuales y Normalización.
Presentes en cada capa.
- Ayudan a estabilizar y acelerar el entrenamiento.

Posicional Encoding.

Insertar información acerca de la posición en los embeddings en la secuencia. Esto se hace ya que self-attention es invariante a la permutación y no tiene información sobre el orden de los elementos.

Generalmente se usan funciones trigonométricas para crear patrones cílicos.

Flujo Ejemplo:



PALABRAS EN
LA SECUENCIA DE ENTRADA

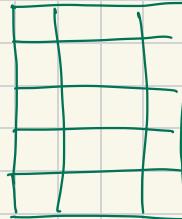


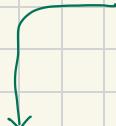
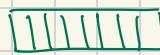
TABLA HASH CON
EL DICTIONARIO DE
PALABRAS DEL
DOMINIO DEL PROBLEMA.



VECTOR CON ENTRADAS
EMBEBIDAS



VECTOR CON INFORMACIÓN
DE POSICIÓN.



CAPAS DE
SKIP CONNECTION
+ NORMALIZACIÓN



MECANISMO DE ATENCIÓN
(MULTI-HEAD ATTENTION)
(Q, K, V)

SECUENCIA DE
SALIDA CONTEXTUALIZADA



SEMÁNTICA

...

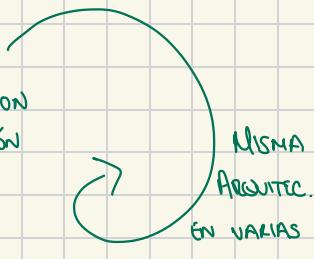
SINTAXIS



NN FEEDFORWARD
(PESOS COMPARTIDOS ENTRE TODOS
LOS ELEMENTOS DE LA SECUENCIA.)



Skip Connection
+ Normalización



MISMA
ARQUITEC.
EN VARIAS
CAPAS.

TRANSFORMACIÓN LINEAL
QUE PREPARA LOS VECTORES
PARA LA PREDICCIÓN

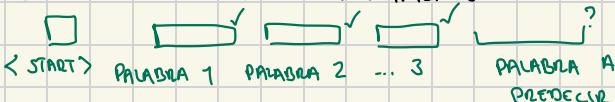
- MÚLTIPLES CAPAS QUE SE FIJAN EN DIFERENTES CARACTERÍSTICAS.

[MISMAS CAPAS QUE EL ENCODER]

DECODER: INICIA CON UN TOKEN DE INICIO DE SECUENCIA Y TOMA LA SALIDA CONTEXTUALIZADA DEL ENCODER Y SU SALIDA GENERADA HASTA ESE MOMENTO PARA PREDICIR LA SIGUIENTE PALABRA EN LA SECUENCIA.

- **PROPIEDAD AUTO-REGRESIVA:** GENERAR ELEMENTO POR ELEMENTO DE MANERA SECUENCIAL TIENENDO EN CUENTA EL CONTEXTO

SHIFT GENERADO PREVIAMENTE.



EL DECODER TIENE UNA CAPA ADICIONAL DE ATENCIÓN Y FEED FORWARD YA QUE PROCESA LA SALIDA DEL ENCODER Y LA SECUENCIA PARCIAL GENERADA HASTA EL MOMENTO.

1. **GENERACIÓN SECUENCIAL:** EL DECODER GENERA UN ELEMENTO BASÁNDOSE EN LA INFORMACIÓN DEL CONTEXTO QUE HA APRENDIDO.
2. **USO DE INFORMACIÓN GENERADA:** EL ELEMENTO GENERADO SE UTILIZA COMO ENTRADA PARA LA GENERACIÓN DEL SIGUIENTE ELEMENTO DE LA SECUENCIA.
3. **ITERACIÓN HACIA LA LONGITUD DESEADA:** EL PROCESO SE ITERA HASTA GENERAR UN ELEMENTO "TERMINADOR".

MÁSCARA DE ATENCIÓN: PERMITT ASEGURAR QUE LAS PREDICCIONES FUTURAS NO INFLUYAN EN LAS PREDICCIONES ACTUALES.

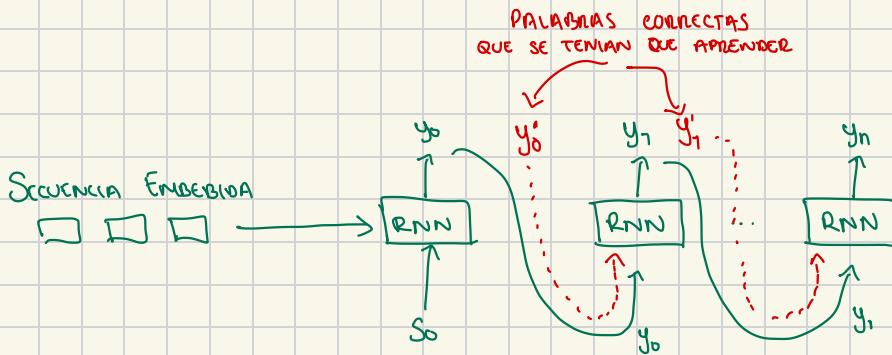
Salida: CAPA FULLY CONNECTED CON UNA FUNCIÓN SOFTMAX PARA OBTENER LA DISTRIBUCIÓN DE PROBABILIDADES SOBRE EL VOCABULARIO OBJETIVO.

ENMASCARAMIENTO:

PONE ATENCIÓN A SÍ MISMO Y A TODAS LAS PALABRAS
PREDECESORAS PARA GARANTIZAR QUE LA SECUENCIA
SE PRODUZCA CORRECTAMENTE

TEACHING FORCING.

TÉCNICA DE AYUDA QUE PROPORCIONA LAS PALABRAS ESPERADAS
DURANTE EL ENTRENAMIENTO PARA ACCELERAR EL APRENDIZAJE.



Predicción con TransFormers:

GREEDY: Se asigna la palabra más probable.

BEAM SEARCH (Generación por Haz):

TÉCNICA PARA ACCELERAR EL PROCESO DE GENERACIÓN A
TRAVÉS DE UN ÁRBOL DE PROBABILIDADES.

- APRENDIZAJE SEMI-SUPERVISADO SIN DECODER.

Función de Atención:

- Se ingresa un Query que se compara con pares Key: Value y regresa un valor ponderado.

a(Q, K:V)

↳ Dado un Query busca aprender la mejor "key" disponible a la que le debe poner atención.

Redes Neuronales Transformers:

Han funcionado muy bien con datos secuenciales como lenguaje natural y series de tiempo.

NLP (Lenguaje Natural):

Las secuencias en lenguaje natural dan contexto a las palabras y a la semántica o sintaxis.

~ 10 millones de parámetros en modelos como Chat GPT.

Ejemplos de Series de Tiempo:

- Tiempos
- Clima
- Bolsa de Valores

Redes Neuronales

↳ Redes Neuronales Convolucionales.

↳ Redes Neuronales Transformers.

COMPLEJIDAD RN TRANSFORMERS.

Espacial: Cuadrática con el tamaño de la secuencia.

NOTAS:

- LAS ARQUITECTURAS ENCODER - DECODER SURGIERON EN 2014.
- LOS MECANISMOS DE ATENCIÓN, SE USAN DESDE 2010 EN OTRAS ARQUITECTURAS COMO LSTM o RNN.

ENCODER: "Escucha" la entrada y la codifica en un conjunto de ideas clave.

DECODER: Toma las ideas clave y las decodifica usando el diccionario / vocabulario de salida. En lenguaje natural sería la traducción.

GRAPH NEURAL NETWORKS.

LIBRO: Graph Representation Learning - Hamilton.

- REDES COMPLEJAS

¿CÓMO LA TOPOLOGÍA IMPACTA EN EL FENÓMENO QUE REPRESENTA LA RED?

- LAS PROPIEDADES DE LOS GRAFOS SON SIMILARES
- Ayuda al aprendizaje al dar información sobre cómo los elementos del problema están conectados entre ellos.

↳ Los nodos en el grafo no son i.i.d.

El concepto de i.i.d. ↘
No implica que las muestras no tienen relación, se aplica al proceso de selección de las muestras que usamos para aprender y para garantizar generalización.

AGREGADOR UNIVERSAL

$$m_{N(u)} = \text{MLP}_\Theta \left(\sum_{v \in N(u)} \text{MLP}_\Phi(h_v) \right)$$

ESTADO DEL
NODO ↗

GENERA TODAS LAS POSIBLES FUNCIONES PERO PODRÍAMOS ESTAR PASÁNDONOS DE COMPLEJIDAD

• Neighborhood Attention

• CADA NODO APRENDE A QUÉ VECINO LE DEBE PONER ATENCIÓN CON BASE EN EL CONTENIDO DEL VECTOR DE CADA NODO (LOS EMISORIOS).

• UN TRANSFORMADOR ES UN GRAPH ATTENTION NETWORK CON LA PARTICULARIDAD DE QUE ESTÁ COMPLETAMENTE CONECTADO.

→ DESPUÉS DE UN K NÚMERO DE PASOS SUFFICIENTEMENTE GRANDE DE ACTUALIZACIONES, LOS EMISORIOS DEL GRAFO COMIENZA A PARECERSE MUCHO. (CONVERGEN A ESTACIONARIEDAD) → LOS NODOS EMPIEZAN A PERDER INFORMACIÓN SOBRE LA INFLUENCIA DE SUS VECINOS
↳ PARA TRATAR DE COMBATIR ESA CONVERGENCIA SE USAN SKIP CONNECTIONS.

- USAMOS LA TOPOLOGÍA DE CONEXIÓN ENTRE LOS ELEMENTOS COMO INFORMACIÓN PARA EL APRENDIZAJE.

→ SECUENCIA DE ACTUALIZACIONES.

$$f_{jk}(h_u^{(0)} \oplus h_u^{(1)} \oplus \dots \oplus h_u^{(k)}) \rightarrow \text{GRAFOS DE CONOCIMIENTO}$$

INVARIANZA

⇒ LO QUE IMPORTA ES CON QUIEN ESTÁS CONECTADO Y NO EN QUÉ ORDEN ESTÁN CONECTADOS.

Aprendizaje SEMI-SUPERVISADO: NO TENEMOS LA CLASE DE LOS FEATURES PERO SÍ TENEMOS INFORMACIÓN SOBRE COMO ESTA CONECTADO EL ELEMENTO CON EL RESTO DE LOS DATOS.

- SE USA LA TOPOLOGÍA DE CONEXIÓN PARA PREDICIR LA CLASE.

NEURAL NETWORKS AND DEEP LEARNING.

- BUSCAMOS DESCUBRIR CÓMO MODIFICAR LOS PESOS, GENERANDO QUE UN PEQUEÑO CAMBIO EN LOS PARÁMETROS PROVOCUE UN PEQUEÑO CAMBIO EN LA SALIDA.
- SIN EMBARGO, UN AJUSTE PARA UNA SALIDA PUEDE AFECTAR OTRA SALIDA, POR LO QUE SE BUSCA UTILIZAR UNA FUNCIÓN DE ACTIVACIÓN COMO LA SIGMOIDE, QUE SUANIZA LOS CAMBIOS

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

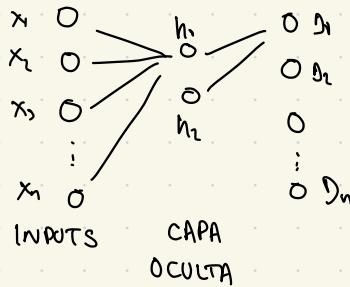
$$= \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

CUANDO z ES MUY NEGATIVO $\rightarrow \frac{1}{1+0} \approx 1$

CUANDO z ES MUY POSITIVO $\rightarrow \frac{1}{1+\infty} \approx 0$

GRAFOS COMPUTACIONALES.

MLP:



$$f(x) = \begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix}$$

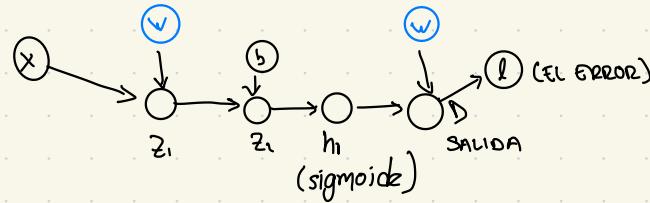
$$h_i = \text{sigm}(Wx + b_i)$$

$$D_1 = \text{sigm}(w_1 h_1 + w_2 h_2 + b_{D_1})$$

$$D_2 = \text{sigm}(w_1 h_1 + w_2 h_2 + b_{D_2})$$

* ENTRE MÁS NEURONAS TENGAN LA CAPA OCULTA, MAYOR ES LA UNIVERSALIDAD QUE PODÉMOS GARANTIZAR.

GRAFO COMPUTACIONAL DE MLP:



CÁLCULO DEL GRADIENTE:

$$\nabla l = \nabla_{z_i} l \cdot [D_{w_i}, z_i]$$

APLICANDO REGLA DE LA CADENA.

$l = \text{lost} \rightarrow \text{LA PÉRDIDA}$

$$z_i = w_i \cdot x$$

$$\lim_{\|H_i\| \rightarrow 0} \frac{(w_i + H_i)x - w_i x - [D_{w_i} z_i] H_i}{\|H_i\|} = 0$$

H_i ES EL CAMBIO O EL INCREMENTO DEL VECTOR DE PESOS w_i .

* EL GRADIENTE DEL ERROR "l" RESPECTO A "w" ES EL CAMBIO DE "l" CUANDO HAY UN PEQUEÑO CAMBIO EN "w" (EL VECTOR DE PESOS).

→ ES UNA TRANSFORMACIÓN LINEAL $T(A) = T(\alpha A + \beta B) = \alpha T(A) + \beta T(B)$

→ CÁLCULO RECURSIVO DEL GRADIENTE.

CALCULAMOS EL GRADIENTE PARA PROPAGAR "HACIA ATRÁS" LA DIRECCIÓN CORRECTA DONDE EL ERROR SE MINIMIZA.

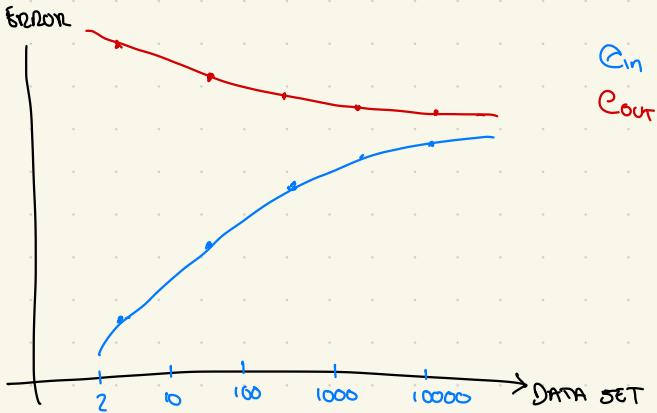
El cambio se representa como un producto exterior para obtener "una matriz de cambio" (el gradiente)

→ El gradiente es la información que le pasamos a "w" para que actualice sus pesos.

$$\rightarrow [\nabla_w l] = \nabla_x f \otimes x$$

↳ notación de producto exterior.

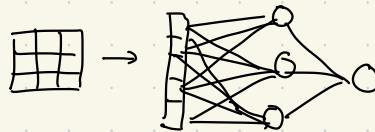
FUNCIÓN DE CRECIMIENTO.



REDES CONVOLUCIONALES.

SE ORIGINARON PARA RESOLVER PROBLEMAS DE VISION
(IMÁGENES EN MATRICES)

NN CONVENCIONAL



OPERACIÓN DE CONVOLUCIÓN:

MATRIZ DE
ENTRADA + KERNEL → NUEVA MATRIZ
convolucionada

(Convolución 2D)

(3D) CUBO DE
ENTRADA + CUBO
KERNEL → CUBO DE SALIDA

• LA DERIVADA DE LA CONVOLUCIÓN ES LA CONVOLUCIÓN TRANSPOSTA
(PARA EL GRADIENTE DESCENDENTE)

TEORÍA DE APRENDIZAJE:

¿POR QUÉ HACER LA OPERACIÓN DE CONVOLUCIÓN FUNCIONA PARA EL APRENDIZAJE?

↳ APLICAR UN KERNEL, REDUCE LAS CONEXIONES Y TOMA EL KERNEL COMO LOS PESOS DE LAS CONEXIONES QUE NO HACE ZERO.

(DEPENDENCIA SOBRE REGIONES LOCALES)

SE CONVIERTEN EN UN MODELO MENOS COMPLEJO.

↳ RELACIONES LOCALES (BIAS INDUCTIVO)

↳ VA A SER UNA MEJOR GENERALIZACIÓN SIEMPRE Y CUANDO EL KERNEL SE VAYA AJUSTANDO CON GRADIENTE DESCENDENTE AL QUE MINIMIZA EL ERROR.

- A DIFERENCIA DE LAS NN TOTALMENTE CONECTADAS, LAS CNN SON EQUI-VARIANTES A TRASLACIÓN.



- APLICANDO "STRIDE" LOGRAMOS DISMINUIR LA COMPLEJIDAD Y "HACER ZOOM" EN REGIONES MÁS GRANDES.

• LA CONVOLUCIÓN ES UNA OPERACIÓN PARTICULAR DE MATRICES QUE HACE MÁS SENCILLO EL CÁLCULO QUE NN TOTALMENTE CONECTADAS (ES COMO UNA SUBFAMILIA).

↳ LOS MODELOS COMPLEJOS SON CAPACES DE AJUSTAR EL RUMBO.

INVARIANZA

- LA OPERACIÓN DE CONVOLUCIÓN GENERA UNA "MATRIZ CIRCULANTE" QUE CONSISTE EN EL MISMO RENGLÓN PERO DESPLAZADO ALGUNAS POSICIONES.
- LA INVARIANZA CONSISTE EN QUE UNA OPERACIÓN $f(Tx) = f(x)$
- LA EQUIVARIANZA CONSISTE EN UNA TRANSPORTACIÓN $f(T(x)) = T(f(x))$
- LAS MATRICES CIRCULANTES TIENEN LA PROPIEDAD DE SER INVARIANTES A TRASLACIÓN $C Px = P C x$,
 - OPERACIÓN ↪ MATEZ CIRCULANTE
de Convolución
- CUANDO QUEREMOS HACER UN "BIAS INICIAL" PODEMOS RESTRINGIR EL ESPACIO DE FUNCIONES CON OPERACIONES COMO LA CONVOLUCIÓN.

TAREA:

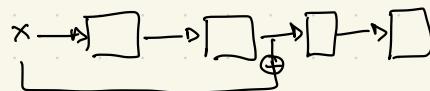
→ GRAFICAR MEDIA Y DESVIACIÓN ESTÁNDAR. → i.i.d.

→ IR SUSTITUYENDO EL NÚMERO DE DATOS 20, 100, 1000 ...

CONEXIONES RESIDUALES: (TÉCNICA PARA REGULARIZACIÓN)

- DEEP LEARNING USA MÚLTIPLES CAPAS CON LA INTENCIÓN DE QUE CADA CAPA ABSTRAIGA CARACTERÍSTICAS JERÁRQUICAS DEL PROBLEMA.
- SKIP CONNECTIONS:

EXISTE "UN SALTO" DÓNDE ENTRA COMO INPUT EL VECTOR ORIGINAL PERMITIENDO QUE LA TOPOLOGÍA "RECUERDE" LA INFORMACIÓN ORIGINAL. [Conexiones Residuales]



CONEXIÓN RESIDUAL

$F^R \rightarrow$ FUNCIONES RESIDUALES

↳ SON MÁS PODEROSAS PERO BAJO CERTAS CONDICIONES, SON LA MISMA FAMILIA DE FUNCIONES

$$F^R \subseteq F \quad y \quad F \subseteq F^R$$

- EL GRADIENTE DE UNA FAMILIA DE FUNCIONES RESIDUALES FLUYE DISTINTO, A PESAR DE QUE EL PODER COMPUTACIONAL ES EL MISMO.

DROPOUT: → TÉCNICA DURA REGULARIZACIÓN

- AL AZAR, PONEMOS EN "0" ALGUNAS NEURONAS DE LAS CAPAS OCULTAS, GENERANDO UNA FUNCIÓN ESTOCÁSTICA PARA LA MISMA ENTRADA Y LOS MISMOS PESOS.

$f(x; w) \rightarrow$ PROBLEMA DE MINIMIZACIÓN DEL ERROR.

$$\min_w \frac{1}{n} \sum_{i=1}^n l(f(x_i; w), y_i)$$

DEPENDE DE w

VS.

$f(x, d, w) \rightarrow$ PROBLEMA DE MINIMIZACIÓN DEL VALOR ESPERADO DEL ERROR

$$\min_w \mathbb{E}_d \left[\frac{1}{n} \sum_{i=1}^n l(f(x_i, d, w), y_i) \right]$$

↳ DIMENSIONES (VARIABLES ALEATORIAS)

- EL GRADIENTE DEL VALOR ESPERADO RESPECTO A LOS PESOS SE CALCULA USANDO UNA INSTANCIA DEL "DROPOUT" Y SE PROPAGA EL ERRORES HACIA ATRÁS SOBRE LAS NEURONAS QUE QUEDAN "PRENDIDAS".

INTUICIÓN

- ESTA TÉCNICA VUELVE MÁS COMPLEJA LA ARQUITECTURA PERO PERMITE PONER UN BIAS INDUCTIVO ANTE EL RUIDO YA QUE AL "APAGAR" ALGUNAS NEURONAS, DEJAMOS QUE LAS NEURONAS PRENDIDAS USEN INFORMACIÓN AISLADA.

PYTORCH → SE SUELE APLICAR DROPOUT DURANTE EL ENTRENAMIENTO Y SE APAGA EN EL TESTING, EVALUANDO CON UNA RED NEURONAL "NORMAL".

↳ EL ENTRENAMIENTO ES MÁS DIFÍCIL PERO QUITA EL RUIDO Y GENERALIZA EL ERROR EN EL TESTING.

CO-ADAPTACIÓN DE PARÁMETROS:

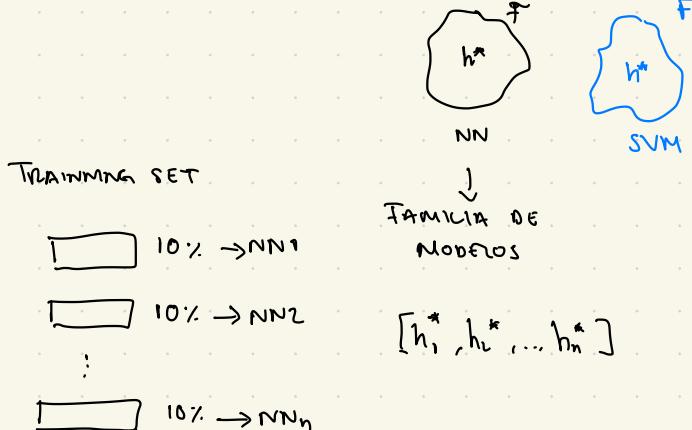
DE LAS CAPAS OCULTAS

DOS NEURONAS O MÁS COLABORAN JUNTAS PARA IDENTIFICAR CARACTERÍSTICAS DE UNA PARTE DEL PROBLEMA.

↳ CON DROPOUT, COMO SE APAGAN ALGUNAS NEURONAS, SE ROMPE ESA DEPENDENCIA.

DROPOUT AS BAGGING.

BAGGING → MEZCLA DE MODELOS. / MODELOS DE ENSAMBLE.



→ UN CONJUNTO DE ENTRENAMIENTO SE PARTEN (i.i.d)
Y CADA BATCH SE ENTRENÁ EN UN NUEVO
DE LA FAMILIA DE MODELOS

→ LA PREDICCIÓN FINAL SE HACE.

- 1) POR VOTACIÓN → CLASIFICACIÓN
- 2) POR PROMEDIO → REGRESIÓN

(LE PREGUNTAMOS A TODOS Y DEJAMOS LO QUE DIGA
LA MAYORÍA)

$$h^*(x) = \frac{1}{n} \sum h_i^*(x)$$

- EN BAGGING, CONFORME CRECE LA CANTIDAD DE MODELOS, LA VARIANZA SE APROXIMA A CERO, ESTO SIGNIFICA QUE LAS VARIABLES ALEATORIAS PIERDEN "ALEATORIEDAD" Y SE EMPIEZAN A VOLVER DETERMINISTAS.
- DROPOUT "SIMULA" SER UNA ARQUITECTURA DISTINTA CUANDO "APAGA" ALGUNAS NEURONAS.
- EN DROPOUT LOS PARÁMETROS NO SON INDEPENDIENTES, POR LO QUE EN ESENCIA, LOS MODELOS NO SON INDEPENDIENTES, COMO EN BAGGING.

TRADE-OFF

- ENTRE NUEVOS DATOS, UN MODELO MÁS COMPLEJO AUMENTA LA VARIANZA, UN MODELO MÁS SENCILLO (INCLUSO UNA CONSTANTE) PUEDE GENERALIZAR MEJOR EL APRENDIZAJE.
- DROPOUT ES UN REGULADOR QUE COMPENSA ESO YA QUE TENEMOS UN CONJUNTO FINITO DE DATOS PARA ENTRENAR Y SI ESTE ES MUY PEQUEÑO, AFECTA EL Sobre-ENTRENAMIENTO.

EN DROPOUT SE PREDICE CON LA MEDIA GEOMÉTRICA EN LUGAR DE LA MEDIA ARITMÉTICA.

BATCH NORMALIZATION.

Normalización \rightarrow MEDIA = 0

VARIANZA = IDENTIDAD = 1

$H \rightarrow$ CONJUNTO ORIGINAL (BATCH)

$H' \rightarrow$ BATCH NORMALIZADO

- EXISTEN 2 PARÁMETROS γ y β QUE APRENDEN CUÁL ES LA MEJOR MEDIA Y LA VARIANZA DEL DATASET

1) NORMALIZAR POR FEATURE

2) SE OBTIENE UN H'

3) SE APRENDE CON H' CUÁL ES LA MEJOR NORMALIZACIÓN.

$$H'' = \gamma H' + \beta$$

↑ ↘
VARIANZA BIAS APRENDIDO
APRENDIDA

- 4) DESPUÉS DEL ENTRENAMIENTO, USAN NUEVOS BATCHES EN EL TESTING Y SE USAN LOS VALORES PREVIAMENTE APRENDIDOS.

- Función de error

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N [f(x_n, \theta)]^2$$

↑
PARÁMETROS (PESOS)

- GRADIENTE

$$\nabla_{\theta} L = \frac{1}{N} \sum_{n=1}^N \nabla [f(x_n, \theta)]^2 \rightarrow \text{GRADIENTE ESTOCÁSTICO}$$

- PARÁMETROS:

$$\theta^{\text{new}} = \theta^{\text{old}} + \eta \nabla_{\theta} L$$

- VALEN ESPERANDO

↳ GRADIENTE CON RESPECTO A LA DISTRIBUCIÓN DEL DATASET

- BATCH \rightarrow SUBCONJUNTO i, j, k DEL DATASET ORIGINAL

↳ SE ENTRENNA UN SUBCONJUNTO DE ORDEN 2 COMO 16 o 32 INSTANCIAS Y SE APLICA GRADIENTE DESCENDENTE ESTOCÁSTICO A CADA UNO Y SE HACE ESTADÍSTICA CON ESO (SE SACAN PROMEDIOS).

MECANISMOS DE ATENCIÓN.

- Si ingresa un query que busca dentro de un objeto key:value y regresa el valor con el peso más grande (Función de atención)

→ KERNEL REGRESIÓN

↳ Aproximación de atención no-paramétrico

σ = parámetro estático que define la distancia de puntos que vamos a considerar.

- Los mecanismos de atención usan funciones de atención de tipo $a(q, k:v)$ con kernels que dan un query busca aprendiendo la mejor "key" disponible a la que le debe poner atención para regresar el "value" correcto.

→ Multi Head Attention:

- En lenguaje natural, se utilizan "cabezas" para "ponerle atención" a diferentes "keys" sobre un feature en particular como: verbos, sujetos, nouns, etc.

BIAS INDUCTIVOS (SUPOSICIONES)

- Queries y keys pasadas por filtros como convolución

Transformadores

- LOS MECANISMOS DE ATENCIÓN FUNCIONAN BIEN EN LENGUAJE NATURAL PORQUE TRABAJAN BIEN CON SECUENCIAS Y LAS SECUENCIAS EN LENGUAJE DAN CONTEXTO A LAS PALABRAS Y A LA SEMÁNTICA O SINTAXIS USADA.

≈ 10 MILLONES DE PARÁMETROS PARA MODELOS COMO CHAT GPT

→ EN LOS TRANSFORMADORES LA POSICIÓN DE LA SECUENCIA ES IMPORTANTE Y PARA ESO SE APLICA UN VARIENDEJAJE ADICIONAL PARA QUE SE APRENDA LA SECUENCIA ADECUADA DE LOS (V, K, Q) O MEDIANTE UNA TÉCNICA COMO POSITIONAL ENCODING QUE USA PROPIEDADES DE FRECUENCIAS CON DIMENSIONES TRIGONOMÉTRICAS.

→ LOS TRANSFORMADORES USAN MULTIMEDIA ATTENTION CONSIDERANDO EL ORDEN DE LA SECUENCIA (INFORMACIÓN POSICIONAL),

SECUENCIA DE SECUENCIA DE
ENTRADA SALIDA

PRIMERAS PALABRAS ... PALABRAS SIGUIENTES

— — — ... — — —

→ SON MUY BUENOS CON SERIES DE TIEMPO COMO TEMBLORES O BOLSA DE VALORES.

ENCODER:



ARQUITECTURA CON MultiHead ATTENTION (MECANISMO DE ATENCIÓN) (N, k, o)

RED NEURONAL
FEED FORWARD
CON PESOS COMPARTIDOS
ENTRE TODOS LOS
ELEMENTOS DE LA
SECUENCIA DE SALIDA.

SECUENCIA DE SALIDA
↓
SEMÁNTICA
↓
SINTAXIS

Skip connection +
NORMALIZACIÓN
(FACILITA EL ENTRENAMIENTO)
[ES UNA ARQUITECTURA MUY
GRANDE]

VECTORES DE SALIDA
PASADOS POR UNA TRANSFORMACIÓN
COMÚN, LISTOS PARA LA
PREDICCIÓN EN EL MISMO ESPACIO
EMBEDDED.

→ SKIP CONNECTION
Y NORMALIZACIÓN

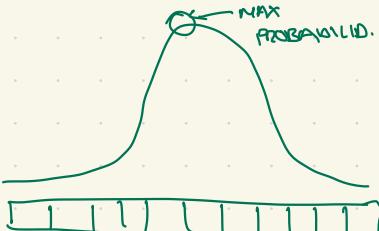
MISMA ARQUITECTURA
NUEVOS VECES
DONDE CADA CAPA
SE FIJA EN
CARACTERÍSTICAS
DIFERENTES

DECODER:

SECUENCIA DE SALIDA DEL TRANSFORMER (EMBEDDING)

ARQUITECTURA:

- MULTIHEAD ATTENTION {
- ENMASCARAMIENTO QUE PONE ATENCIÓN A TODAS LAS ENTRADAS PREDECESORAS.
 - **Modelo autorregresivo:** EN SERIES DE TIEMPO, LA PREDICCIÓN ACTUAL DEPENDE DE LA SECUENCIA QUE LLEVA
 - 
 - SALIDA CON UN SHIFT PARA QUE EL PRIMER ELEMENTO DE LA SECUENCIA SEA UN MARCADOR DE INICIO DE SECUENCIA.
 - LA SECUENCIA DE SALIDA TIENE QUE APRENDER A QUE ELEMENTOS DE LA SECUENCIA DE ENTRADA LE DEBE PONER ATENCIÓN.



d = DIMENSION DEL VOCABULARIO

P.EJ. 100,000 PALABRAS

[$K = \text{TAMÁO DEL VOCABULARIO O CLASES QUE SE PUEDEN APRENDER}$]

TRANSFORMACIÓN VIEJA → POR EJEMPLO

TRADUCCIÓN

→ CONVOLUCIÓN ES UN TIPO PARTICULAR DE RED NEURONAL

→ TRANSFORMADORES ES UN TIPO PARTICULAR DE CONVOLUCIÓN

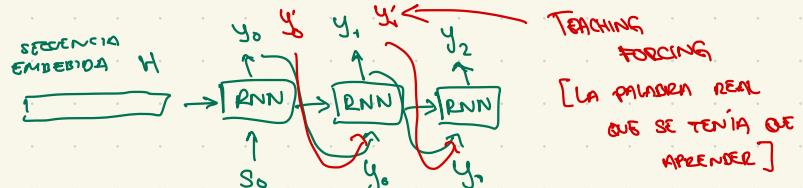
ATENCIÓN: DISTRIBUCIÓN DE PROBABILIDAD PARA DETERMINAR A QUE PALABRA TIENE QUE PONER ATENCIÓN EL QUERY DE INTERÉS ACTUAL.

→ LAS REDES RECURRENTES HACEN TANTO PARALELO PERO PRODUCEN UNA SECUENCIA SERIAL QUE NO SE PUEDE ENTRENAR EN PARALELO A DIFERENCIA DE LOS TRANSFORMADORES. CON UNA TÉCNICA LLAMADA "TEACHING FORCING" AyUDA AL ENTRENAMIENTO DÓNDE LAS PALABRAS NECESARIAS O CORRECTAS DE MANERA "FOTLANDIA"

- SE PUEDE APLICAR GRADIENTE DESCENDENTE PARA APRENDER SI LA ARQUITECTURA ES UN CIRCUITO COMPUTACIONAL CON TODOS SUS ELEMENTOS DIFERENCIABLES.

REPOSITORIO A REVISAR PARA TRANSFORMERS:

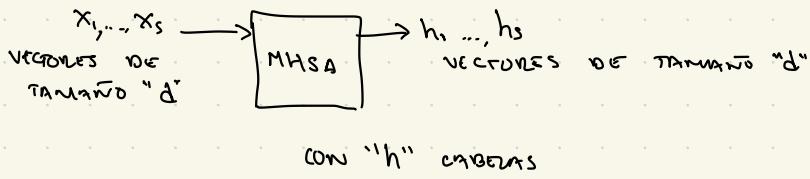
"THE ANNOTATED TRANSFORMERS"



PREDICCIÓN CON TRANSFORMERS:

- GREEDY \rightarrow LA "PALABRA" MÁS PROBABLE
- BEAM SEARCH \rightarrow UN ÁRBOL DE PROBABILIDADES.
LAS "k" PALABRAS MÁS PROBABLES
 $k|L|$ DONDE $|L|$ ES EL TAMAÑO
DEL VOCABULARIO
 \rightarrow GENERACIÓN HEURÍSTICA DE SECUENCIAS DE SALIDA.

Composición Condicional de los Transformers.



$$d \cdot h = D$$

① TRANSFORMACIÓN DE LAS "x"

$$\begin{bmatrix} & \end{bmatrix}_{q \times d} \begin{bmatrix} & \end{bmatrix}_{d \times s} = \begin{bmatrix} & \end{bmatrix}_{q \times s}$$
$$q = d/h$$

$$\text{MEMORIA} = q \times d + d \times s + q \times s$$

$$\hookrightarrow O\left(\frac{d^2}{n} + ds\right) \text{ POR CABEZA}$$

$$h\left(\frac{d^2}{n} + ds + \frac{ds}{n}\right)$$

$$\Theta(d^2 + hds)$$

② CALCULAR MECANISMO DE ATENCIÓN.

$$[V^{(i)}]^T [Q^{(i)}] = [A^{(i)}]$$

SxS

MECANISMO DE ATENCIÓN IGUAL
POR CABEZA

$O(h \cdot s^2)$

APLICAR SOFTMAX

③ CALCULAR LOS VALORES POMOENTADOS DE ATENCIÓN.

$$[A^{(i)}] [V^{(i)}] = [] \leftarrow \text{POR CABEZA}$$

S d/h → O(sd)

④ CONCATENAR CABEZAS Y HACER UNA TRANSFORMACIÓN LINEAL

$$\begin{matrix} [W] & [V] \\ d \times d & d \times s \end{matrix} = \begin{matrix} [] \\ d \times s \end{matrix} \rightarrow O(d^2 + ds + d \times s) \\ \approx O(d^2 + ds)$$

Complejidad espacial = $d^2 + hds + ds + hs^2 + sd + d^2 + ds$

$$O(d^2 + hds + hs^2)$$

→ Crecimiento con el tamaño de la secuencia

→ GENERALMENTE HAY UN MÁXIMO DE SECUENCIA
(UN CONTEXTO MÁXIMO)

→ VARIANTES ← (Sin decoder)

- APRENDIZAJE SEMI-SUPERVISEADO → "BERT" / "GPT"

↳ DATOS QUE NO ESTÁN ETIQUETADOS



DATOS "ENMASCARADOS" AL AZAR

PALABRAS DE LA SECUENCIA



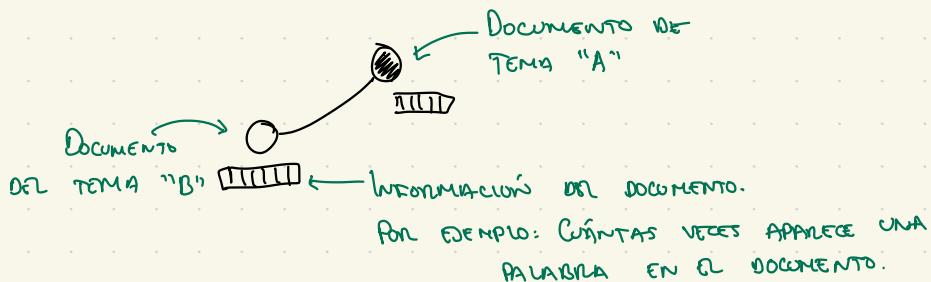
↳ MODIFICAMOS AL AZAR UNAS Y EL OBJETIVO ES RECUERDARLAS

"Fine - Tuning"

MIP
CLS → REPRESENTACIONES VECTORIALES DE SECUENCIAS EN TOKENS (NEUTRAS, SIN UN PROBLEMA DADO CON INFORMACIÓN MUY GENERALIZADAS DEL LENGUAJE)
BERT

GRAPH NEURAL NETWORKS

- INTERACCIONES ENTRE ELEMENTOS DENTRO DE SISTEMAS COMPLEJOS.
 - LO IMPORTANTE DE UN GRÁFO ES SU TOPOLOGÍA
 - LAS FUNCIONES EN UN GRÁFO DEBEN SER INVARIANTES A PERMUTACIÓN O ETIQUETADO DE NODOS / ARISTAS
- Ayuda a crear mejores modelos tener:
- + DATOS TABULARES
 - + INFORMACIÓN DE CÓMO INTERACTUAN ESOS DATOS (SU TOPOLOGÍA)



- LOS NODOS, AL ESTAR CONECTADOS, TIENEN PROPIEDADES QUE EVITAN QUE SEAN i.i.d (INDEPENDIENTES)

TIPOS DE INDEPENDENCIA:

- 1) ¿UN DATO DA INFORMACIÓN DE OTRO?
- 2) ¿USAMOS UN DATO PARA GENERAR EL SIGUIENTE?

Las propiedades de un dato nos dan información de otro

Como se agarran los dato

→ APRENDIZAJE SEMISUPERVISADO.

↳ NO TIENEN TODAS LAS ETIQUETAS PERO
NOS DABAN INFORMACIÓN DE QUIÉN ESTÁ CONECTADO
CON QUIÉN.

TIPOS DE USOS:

- 1) HACER ML CON LA ENTRADA EN FORMATO DE GRAFO
- 2) CONOCER LAS CONEXIONES DE LA ENTRADA PARA
HACER MEJORES MODELOS.

ALGORITMOS DE ML PARA GRAFOS:

- FUNCIONES PARA CREAR EMBEBIDOS DEL GRAFO DE
ENTRADA

↳ CAMBIA LA ENTRADA A UNA NUEVA
REPRESENTACIÓN "EMBEBIDA" DE CADA NODO

- SE LE APLICA APRENDIZAJE CON NN A ESTE
ENSEÑAZO DE CADA NODO Y SUS VECINOS.

PARÁMETROS DE UN GRAPH NEURAL NETWORK

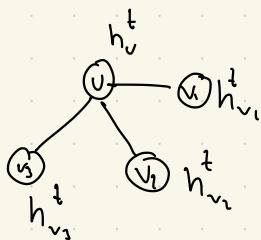
- RED NEURONAL
- FUNCIÓN PARA EMBEBER

• NORMALIZACIÓN

↳ PERMITE COMBINAR VALORES MUY GRANDES
CUANDO UN NODO TIENE MUCHOS VECINOS O
UN VECINO CON MUCHOS VECINOS Y DARLE
MÁS IMPORTANCIA A LOS VALORES QUE A
SUS PROPIEDADES ESTRUCTURALES.

APROXIMACIONES UNIVERSITARIAS EN GRAFOS NN.

(INVARIANTE A PERMUTACIÓN)



↓
NO DEPENDE DEL
ORDEN DE LOS
NODOS.

→ {ESPACIO DE FUNCIONES QUE GARANTIZAN
QUE LA FUNCIÓN DE AGREGACIÓN ES
INVARIANTE A PERMUTACIÓN}

$$\text{P. EJ. MLP } \left(\sum \text{MLP}(h_v) \right) \forall v \in N(u)$$

"AGREGADORES": FUNCIONES QUE NOS AYUDAN A ENTENDER
LA INFORMACIÓN DE LA ARQUITECTURA
DEL GRAFO.

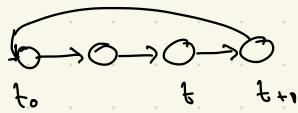
- Algunos Graph Neural Networks son TRANSFORMADORES CON MECANISMOS DE ATENCIÓN SOBRE LA ESTRUCTURA DEL GRAFO.

Graficación de Convolución.

Convolution:

Serial + Filters

HAY UN GRANFO NACIONAL AL ESTADO DE JA
SENÁZ



$$\begin{matrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_0 \end{bmatrix}$$

$$Q_F = \text{FILTRADO}$$

↑ SHIFT = y
SIGNAL x

$$y = Q_f x$$

MARIANZA 19

Traición

$$\rightarrow S Q_F x = Q_F S x$$

USAR CONVOLUCIÓN EN UNA SEÑAL MONTAÑA SOBRE UN GRAFO SEÑAL

$$\begin{bmatrix} & \end{bmatrix} \begin{bmatrix} & \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

MATEZIL DE
ADYACENCIA

SHIFT

- #### POTÈNCIAS DE UNA MÀNADA DE AVYACONCIL

- La convolución en Graph Neural Networks nos permite filtrar una señal para ver cómo cambia la señal considerando la influencia de sus vecinos cercanos.

- LA TRANSFORMACION DE FOURIER "DIAGONALIZA" A UNA CONVOLUCION.
- LA TRANSFORMACION DE FOURIER DE UNA CONVOLUCION = PRODUCTO DE LAS TRANSFORMADAS
- "DIAGONALIZAR" UNA TRANSFORMACION COMO CONVOLUCION, HACE MAS EFICIENTE SU CICLO.

→ FRECUENCIAS EN GRAFOS:

- 1) ALTA FRECUENCIA = LOS VALORES DE UN NODO Y SUS vecinos CAMBIAN MUCHO.
- 2) BAJA FRECUENCIA = TODOS LOS NODOS TIENEN EL MISMO VALOR (señal constante)

GRAPHICAL MODELS.

GRAFOS DIRIGIDOS O NO DIRIGIDOS QUE REPRESENTAN UNA DISTRIBUCION DE PROBABILIDAD.

ALGORITMOS (Sum-Product) (Version ARBOL)

- CON INFORMACION DINAMICA.

- ALGORITMOS DE PESO DE MENSAJES PARA ENCONTRAR LAS PROBABILIDADES MARGINALES DE UNA VARIABLE DE INTERES

