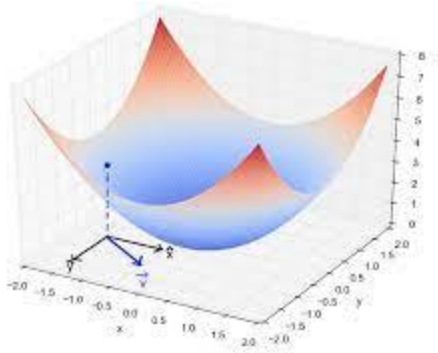
• Deep Learning

• Fundamentos de la IA

DuocUC®

ESCUELA DE
INFORMÁTICA Y
TELECOMUNICACIONES

Descenso del Gradiente



Próxima clase entregaré los detalles del encargo!

EVALUACIONES

Fechas	Ponderación	
Unidad 1 (prueba)	15-04-23	10%
Unidad 2 (Encargo + Presentación)	06-05-23	40%
Unidad 2 (Encargo + Presentación)	03-06-23	25%
Unidad 4 (Encargo + Presentación)	01-07-23	25%

70%

Examen transversal (Encargo + Presentación) **08-07-23**

30%

Calendario

MARZO

SM	LU	MA	MI	JU	VI	SA	DO
09			1	2	3	4	5
10	6	7	8	9	10	11	12
11	13	14	15	16	17	18	19
12	20	21	22	23	24	25	26
13	27	28	29	30	31		

ABRIL

SM	LU	MA	MI	JU	VI	SA	DO
13						1	2
14	3	4	5	6	7		9
15	10	11	12	13	14	15	16
16	17	18	19	20	21	22	23
17	24	25	26	27	28		30

MAYO

SM	LU	MA	MI	JU	VI	SA	DO
18	1	2	3	4	5	6	7
19	8	9	10	11	12	13	14
20	15	16	17	18	19	20	21
21	22	23	24	25	26	27	28
22	29	30	31				

JUNIO

SM	LU	MA	MI	JU	VI	SA	DO
22				1	2	3	4
23	5	6	7	8	9	10	11
24	12	13	14	15	16	17	18
25	19	20	21	22	23	24	25
26	26	27	28	29	30		

JULIO

SM	LU	MA	MI	JU	VI	SA	DO
26						1	2
27	3	4	5	6	7	8	9
28	10	11	12	13	14	15	16
29	17	18	19	20	21	22	23
30	24	25	26	27	28	29	30
31	31						

Encargo

- Se realiza en parejas
- 3 casos: A, B y C. El caso elegido se utilizará en las distintas evaluaciones del semestre.
- La evaluación 2 vale un 40% ¡ojito!
- La rúbrica estará disponible en AVA
- Para la evaluación 2, puede utilizar el mismo Jupyter-notebook para realizar la presentación. Si así lo decide, cuide el orden y claridad del mismo.
- El trabajo tiene que reflejar una propuesta original. Si utiliza el trabajo de otra persona o inteligencia artificial, deberá explicitarlo indicando claramente qué y su cita correspondiente. Caso contario será considerado plagio y se aplicaran las sanciones establecidas en el reglamento.

Caso A

Contexto del caso Forma A.

A continuación, se presenta el caso a ser utilizado en las tres evaluaciones sumativas y el examen transversal de la asignatura. Deberá ser capaz de implementar un proyecto de Deep Learning, entrenando un modelo ajustado a partir de los datos entregados para este caso. Desarrollará el proyecto utilizando las etapas de metodología CRISP.DM, para la toma de las mejores decisiones de la ejecución del proyecto.

Este caso propone la utilización de Deep Learning para la clasificación de imágenes de prendas de vestir.

Para contextualizar el caso, las empresas de moda han utilizado Deep Learning en su comercio electrónico para resolver muchos problemas, como el reconocimiento de ropa, la búsqueda de ropa y la recomendación. Un paso central para todas estas implementaciones es la clasificación de imágenes. Sin embargo, la clasificación de la ropa es una tarea desafiante ya que la ropa tiene muchas propiedades y la profundidad de la categorización de la ropa es muy complicada.

Para este caso utilizaremos un conjunto de datos denominado Fashion-MNIST, que consta de imágenes en escala de grises de 28 × 28 de 70 000 productos de moda de 10 categorías, con 7000 imágenes por categoría. El conjunto de entrenamiento tiene 60 000 imágenes y el conjunto de prueba tiene 10 000 imágenes. Fashion-MNIST está destinado a servir como un reemplazo directo del conjunto de datos MNIST original para comparar algoritmos de aprendizaje automático, ya que comparte el mismo tamaño de imagen, formato de datos y la estructura de las divisiones de entrenamiento y prueba. El conjunto de datos está disponible gratuitamente en https://github.com/zalandoresearch/fashion-mnist.

Caso B

Contexto del caso Forma B.

A continuación, se presenta el caso a ser utilizado en las tres evaluaciones sumativas y el examen transversal de la asignatura. Deberá ser capaz de implementar un proyecto de Deep Learning entrenando un modelo ajustado a partir de los datos entregados para este caso. Desarrollará el proyecto utilizando las etapas de metodología CRISP.DM, para la toma de las mejores decisiones de la ejecución del proyecto.

Para este caso un Banco requiere la digitalización de cheques utilizando técnicas de Deep Learning.

Para contextualizar el caso, el problema del reconocimiento de texto manuscrito en imágenes es muy importante en cuanto a aplicaciones en el área de visión por computadora. Un problema más acotado, pero no menos importante, es el de reconocimiento de secuencias de dígitos manuscritos de largo variable en imágenes. El presente trabajo consiste en investigar un nuevo enfoque para resolver el problema de reconocimiento de secuencias de dígitos manuscritos en imágenes. Se plantea dicho problema como uno de detección de objetos en que los dígitos corresponden a los objetos a detectar.

Para el entrenamiento se cuenta con un dataset llamado MNIST (el cual cuenta con 60.000 ejemplos para entrenar la red y otros 10.000 ejemplos para testear en modelo creado. Los dígitos se normalizaron en tamaño y se centraron en una imagen de tamaño fijo.

Descargar dataset a utilizar según el caso práctico entregado en el enunciado, o bien desde http://yann.lecun.com/exdb/mnist/ o https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

Caso C

Contexto del caso Forma C.

A continuación, se presenta el caso a ser utilizado en las tres evaluaciones sumativas y el examen transversal de la asignatura. Deberá ser capaz de implementar un proyecto de Deep Learning entrenando un modelo ajustado a partir de los datos entregados para este caso. Desarrollará el proyecto utilizando las etapas de metodología CRISP.DM, para la toma de las mejores decisiones de la ejecución del proyecto.

Para este caso el Gobierno de Chile requiere etiquetado de imágenes de distinto dominio para implementar inteligencia artificial en su página web.

Actualmente el mundo se encuentra en una era de un gran cúmulo de datos, donde muchos problemas de tomas de decisión son resueltos mejor por máquinas que por humanos, en términos de exactitud y escalabilidad. Muchos de estos datos están conformados por imágenes. En el mundo actual es necesario poder seleccionar estas imágenes y clasificarlas, detectar qué objetos la conforman, detectar estado de ánimo de una o varias personas e identificar rostros. Para realizar estas tareas se utilizan técnicas de Inteligencia Artificial. El Aprendizaje Automático es una de las ramas de la Inteligencia Artificial que tiene como objetivo crear técnicas que le permitan a las computadoras aprender, generalizando patrones a partir de conocimiento adquirido anteriormente. Dentro del Aprendizaje Automático se encuentra el Aprendizaje Activo, donde se mejora la precisión y el rendimiento de los modelos de predicción.

Para este se caso utilizaremos el conjunto de datos CIFAR-10.

CIFAR-10 consta de 60000 imágenes en color de 32x32 en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 imágenes de prueba.

El conjunto de datos se divide en cinco lotes de entrenamiento y un lote de prueba, cada uno con 10000 imágenes. El lote de prueba contiene exactamente 1000 imágenes seleccionadas al azar de cada clase. Los lotes de entrenamiento contienen las imágenes restantes en orden aleatorio, pero algunos lotes de entrenamiento pueden contener más imágenes de una clase que de otra. Entre ellos, los lotes de entrenamiento contienen exactamente 5000 imágenes de cada clase.

Descargar data set desde https://www.cs.toronto.edu/~kriz/cifar.html

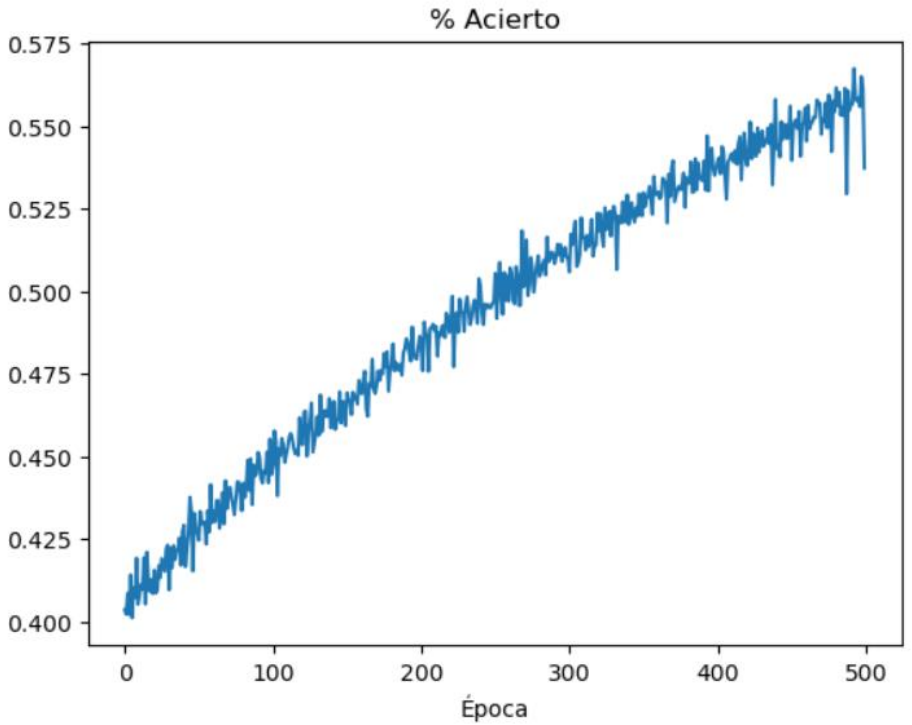
Quiz

Revisión Notebook

- ¿Qué efecto tiene no normalizar los datos?
- Cuál fue su propuesta de red neuronal?
  - ¿Qué precisión obtuvo con su propuesta?
  - ¿Qué hiper-parámetros modificó?
  - ¿Con cuáles hiper-parámetros observó una mejora?

Layer (type)	Output Shape	Param #
capa01 (Dense)	(None, 256)	786688
capa02 (Dense)	(None, 128)	32896
capa03 (Dense)	(None, 96)	12384
capa04 (Dense)	(None, 64)	6208
capa05 (Dense)	(None, 48)	3120
capa06 (Dense)	(None, 32)	1568
capa07 (Dense)	(None, 16)	528
capa_de_output (Dense)	(None, 10)	170

Total params: 843,562
Trainable params: 843,562
Non-trainable params: 0



Revisión Notebook

- Diferencia entre épocas e iteraciones

Época: Constituye una pasada de ida y vuelta (forward pass + backward pass) de todo el conjunto de datos
Iteración: Constituye una pasada de ida y vuelta (forward pass + backward pass) de un lote (batch) de datos

```
h = red.fit(X, Y,
        epochs=5,
        batch_size=1,
       )
```

```
Epoch 1/5
60000/60000 [==============================] - 41s 682us/step - loss: 0.1771 - accuracy: 0.9457
Epoch 2/5
60000/60000 [==============================] - 41s 686us/step - loss: 0.1128 - accuracy: 0.9663
Epoch 3/5
60000/60000 [==============================] - 41s 682us/step - loss: 0.0898 - accuracy: 0.9724
Epoch 4/5
60000/60000 [==============================] - 41s 677us/step - loss: 0.0778 - accuracy: 0.9761
Epoch 5/5
60000/60000 [==============================] - 41s 682us/step - loss: 0.0686 - accuracy: 0.9790
```

```
h = red.fit(X, Y,
        epochs=5,
        batch_size=64,
       )
```

```
Epoch 1/5
938/938 [==============================] - 1s 856us/step - loss: 0.3233 - accuracy: 0.9079
Epoch 2/5
938/938 [==============================] - 1s 845us/step - loss: 0.2880 - accuracy: 0.9175
Epoch 3/5
938/938 [==============================] - 1s 842us/step - loss: 0.2625 - accuracy: 0.9249
Epoch 4/5
938/938 [==============================] - 1s 842us/step - loss: 0.2423 - accuracy: 0.9306
Epoch 5/5
938/938 [==============================] - 1s 840us/step - loss: 0.2251 - accuracy: 0.9360
```

```
h = red.fit(X, Y,
        epochs=5,
        batch_size=2048,
       )
```

```
Epoch 1/5
30/30 [==============================] - 0s 3ms/step - loss: 0.4126 - accuracy: 0.8852
Epoch 2/5
30/30 [==============================] - 0s 3ms/step - loss: 0.4081 - accuracy: 0.8858
Epoch 3/5
30/30 [==============================] - 0s 3ms/step - loss: 0.4039 - accuracy: 0.8868
Epoch 4/5
30/30 [==============================] - 0s 3ms/step - loss: 0.3998 - accuracy: 0.8880
Epoch 5/5
30/30 [==============================] - 0s 3ms/step - loss: 0.3960 - accuracy: 0.8890
```

Carta

All Open Letters

Pause Giant AI Experiments: An Open Letter

We call on all AI labs to immediately pause for at least 6 months the training of AI systems more powerful than GPT-4.

Signatures
2446

Add your signature

AI systems with human-competitive intelligence can pose profound risks to society and humanity, as shown by extensive research[1] and acknowledged by top AI labs.[2] As stated in the widely-endorsed Asilomar AI Principles, *Advanced AI could represent a profound change in the history of life on Earth, and should be planned for and managed with commensurate care and resources*. Unfortunately, this level of planning and management is not happening, even though recent months have seen AI labs locked in an out-of-control race to develop and deploy ever more powerful digital minds that no one – not even their creators – can understand, predict, or reliably control.

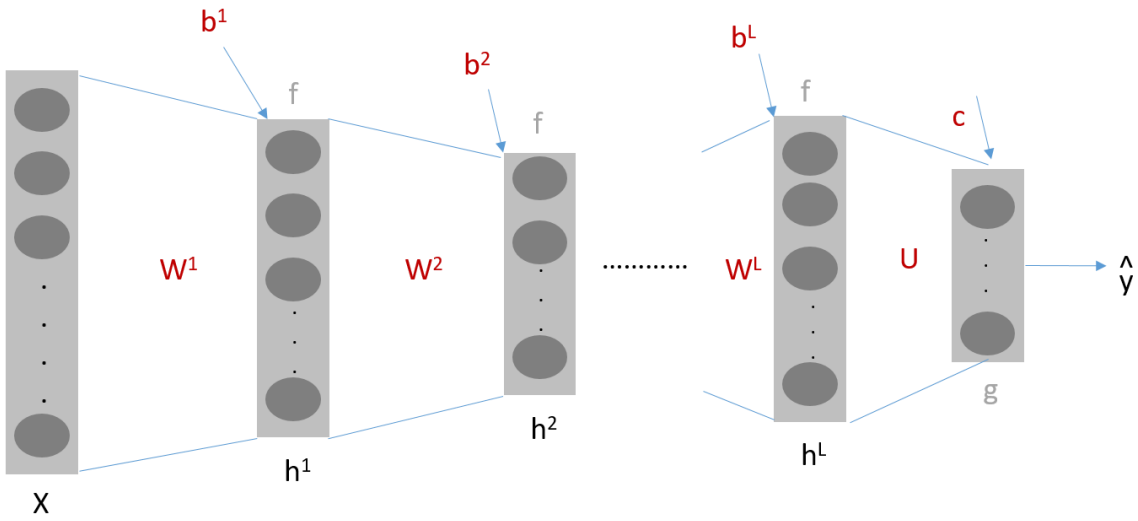
Contemporary AI systems are now becoming human-competitive at general tasks,[3] and we must ask ourselves: *Should* we let machines flood our information channels with propaganda and untruth? *Should* we automate away all the jobs, including the fulfilling ones? *Should* we develop nonhuman minds that might eventually outnumber, outsmart, obsolete and replace us? *Should* we risk loss of control of our civilization? Such decisions must not be delegated to unelected tech leaders. **Powerful AI systems should be developed only once we are confident that their effects will be positive and their risks will be manageable.** This confidence must be well justified and increase with the magnitude of a system's potential effects. OpenAI's recent statement regarding artificial general intelligence, states that *"At some point, it may be important to*

https://futureoflife.org/open-letter/pause-giant-ai-experiments/

Resumen Inicial

Dato $D = \{ (x_1,y_1), (x_2,y_2).....(x_i, y_i)...(x_n,y_n) \}$

Donde:
D: Es el conjunto de datos o ejemplos a utilizar para entrenar la red
x_i: Es un ejemplo o dato a ingresar a la red
y_i: Es el resultado que esperamos calcule la red



Función de Pérdida (Loss) -> [Losses (keras.io)]

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \text{error}(\hat{y}^{(i)} - y^{(i)})$$

Parámetros de Entrada

- La función de error depende de los valores de los parámetros (pesos y bias)
- El objetivo final es minimizar esta función (Una red aprende en la medida en que ajusta Θ)

→ OPTIMIZACIÓN!!!!!!

Función error y Descenso del Gradiente

Uno de los algoritmos de optimización más utilizados en Deep learning que permite a la red a minimizar la función de error es el descenso del gradiente.

En la práctica, no es tan así, en realidad se usan otros algoritmos de optimización basados en el descenso del gradiente, debido a que es un algoritmo demasiado lento o ineficiente, pero es importante de entender ya que sus variaciones, con pequeños cambios, tienen un impacto relevante en términos de eficiencia.

Minimizar la función de error = encontrar los valores de los parámetros que entreguen el menor valor para $\mathscr{L}$

Usualmente, minimizar una función se hace derivando e intentando igualar a 0

Sin embargo, esta función podría tener millones de parámetros, de hecho en la actualidad alcanzan a billones de estos

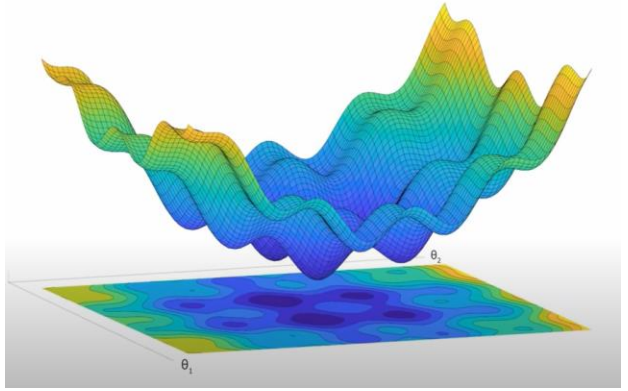
Lo que se hace es tratar de acercarse al menor error posible de la función $\mathscr{L}(\Theta)$

$$\mathscr{L} = \frac{1}{n} \sum_{i=1}^{n} \text{error}(\hat{y}^{(i)} - y^{(i)})$$

$$\theta = (W^{(1)} , b^{(1)}, W^{(2)} , b^{(2)}, ..., U, C)$$

DuocUC®

Función error y Descenso del Gradiente



El descenso del gradiente evalúa cómo cambia el error (mejora o empeora) cuando se mueven los parámetros.

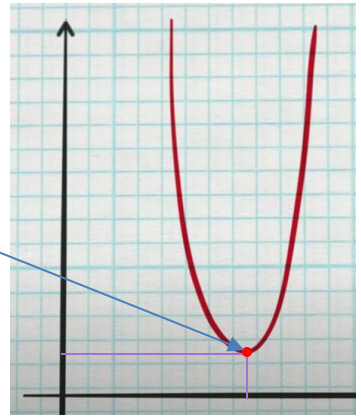
Variación de θ dependiendo de la variación de $\mathcal{L}$

Entonces, afinando definiciones, se puede especificar que entrenar una red es encontrar la combinación correcta de parámetros que minimizan la función de error, lo que se hace, en general, usando algún algoritmo de optimización basado fuertemente en el descenso del gradiente.

Optimizers (keras.io)

https://www.youtube.com/watch?v=A6FiCDoz8_4

Encontrar este valor



DuocUC®

Descenso del Gradiente

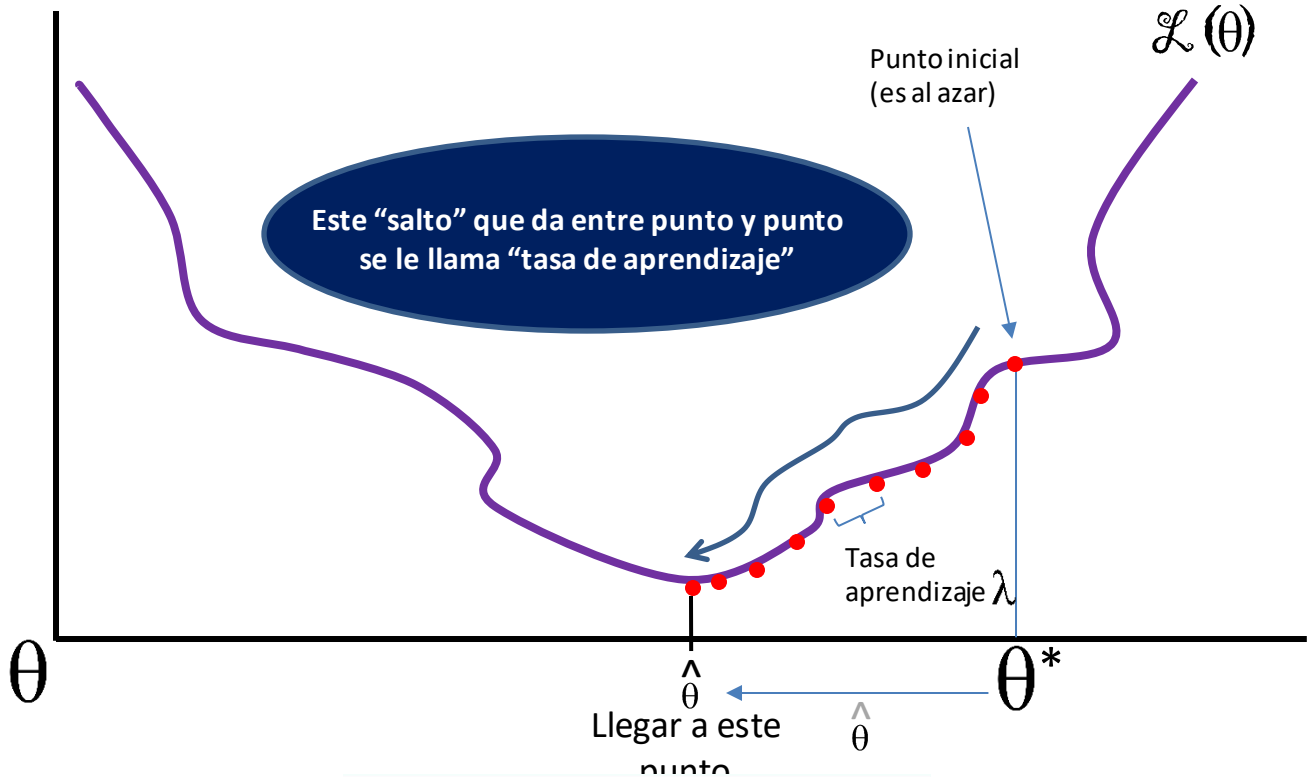
El algoritmo comienza en un punto al azar (el valor inicial de $W_{i,j}$) dentro de la función.
El objetivo es iterar, hasta llegar al punto tetha gorro, que es el punto que minimiza la función.
En el paso a paso, cada vez que ajusto o cambio el $W_{i,j}$, se calcula la derivada en este punto.
(Derivar significa encontrar la pendiente de la función en ese punto -> Caída a un valor -> en este caso Cero).

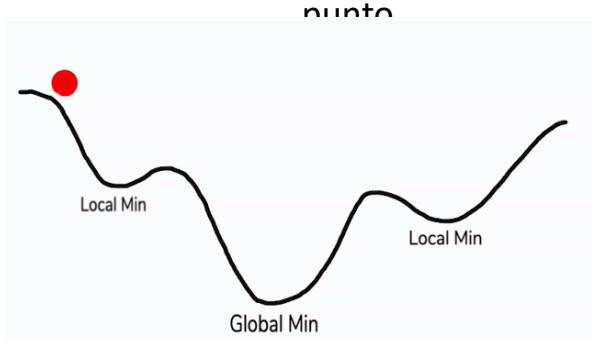
Al cambiar el valor del parámetro, desde ese nuevo punto se calcula la derivada a cero y luego se compara en qué sentido la función $\mathscr{L}$ crece y en cuál decrece.

La idea es mover la función en la dirección contraria a la que crece la pendiente (es decir en sentido decreciente)
Cada "Salto" o espacio que la función se mueve es denominada "tasa de aprendizaje", que indica en cuánto debo mover el parámetro.
Luego, se vuelve a cambiar el valor del parámetro y se itera tantas veces cómo se necesite hasta que el resultado sea igual a cero, o en la "vida real", hasta cuando alcance el criterio que se ha definido para detener el algoritmo.

El trabajo que realmente es complicado en el descenso del gradiente es calcular las derivadas, es lo realmente caro computacionalmente.
En una red con miles o millones de parámetros (pesos y bias), calcular el error "es muy caro" ya que tengo se debe pasar todos los ejemplos.

Un problema del descenso del gradiente es el "Mínimo Local", es decir, que el menor valor que el algoritmo encuentre en un punto sea o no el mínimo de la función (mínimo global). Esto en la práctica, no tiene solución. No hay forma de saber si el mínimo encontrado es un mínimo local o un mínimo global





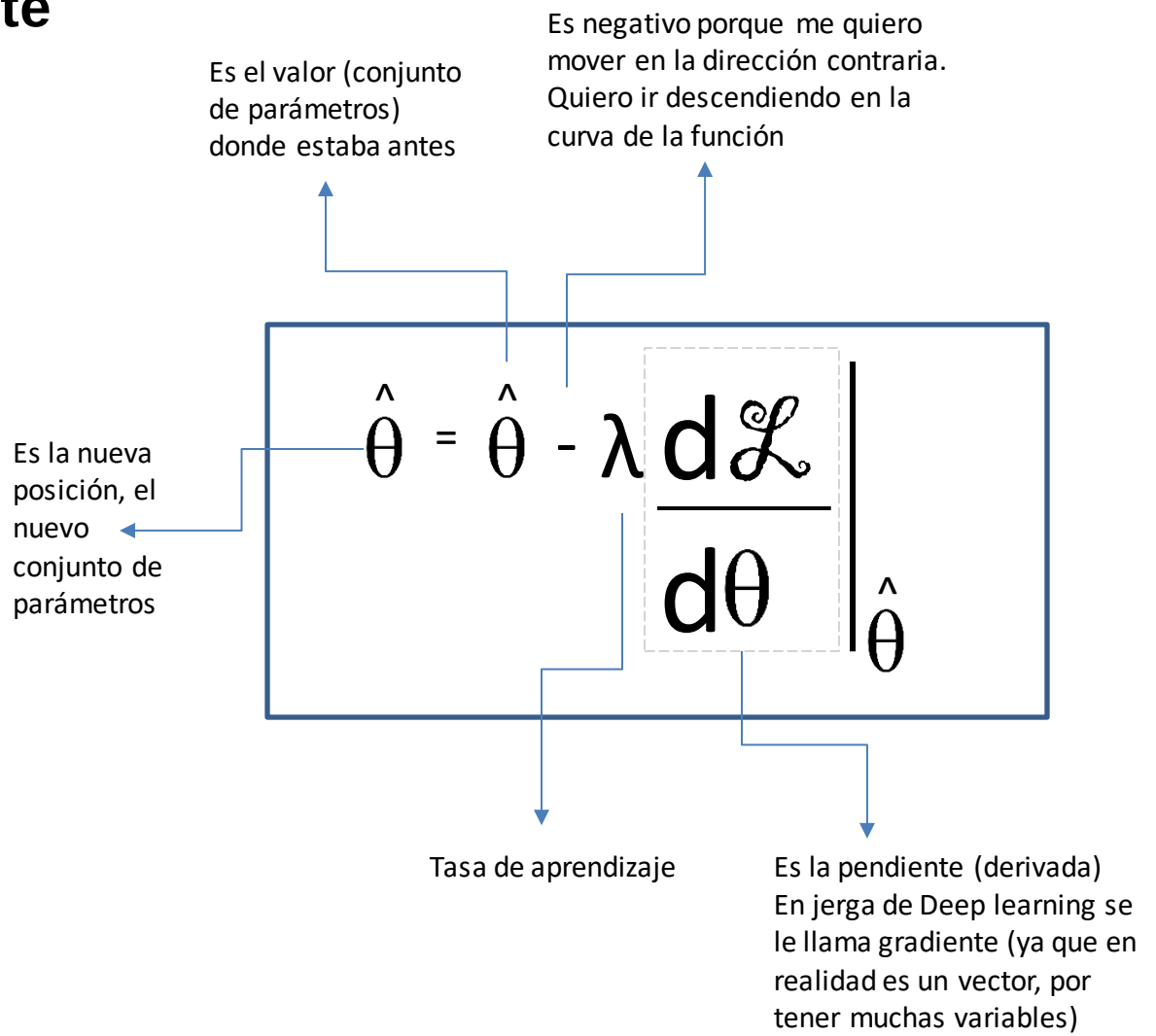
DuocUC®

Descenso del Gradiente

Una vez inicializados al azar el conjunto de parámetros $\hat{\theta}$, se itera hasta alcanzar algún criterio

La fórmula indica que dado un conjunto theta de parámetros, se Evalúa la pendiente de la función y se invierte el sentido (me muevo en la dirección contraria), dando saltos de largo lambda y el proceso termina al alcanzar el criterio definido (la función es minimizada)

En otras palabras, se actualiza el theta (conjunto de parámetros) con el theta de la iteración anterior y se calcula la derivada del error con respecto a theta, evaluada en theta gorro, saltando en la dirección contraria dando un salto definido por lambda



Descenso del Gradiente

Generalizando la fórmula del descenso del gradiente a la estructura general de la red:

Número de la capa

Derivada parcial, vale la pena recordar que la derivada parcial de una función que tiene muchas variables, es derivar solo una de ellas, dejando fijas el resto, como si fueran constantes

$$\hat{W}^i = \hat{W}^i - \lambda \frac{d\mathcal{L}}{dW^i}\Big|_{\hat{\theta}}$$

$$\hat{b}^i = \hat{b}^i - \lambda \frac{d\mathcal{L}}{db^i}\Big|_{\hat{\theta}}$$

Recordar que W y U son matrices, por lo tanto lo que pasa en la práctica es que se tienen que derivar cada uno de los valores de dicha matriz

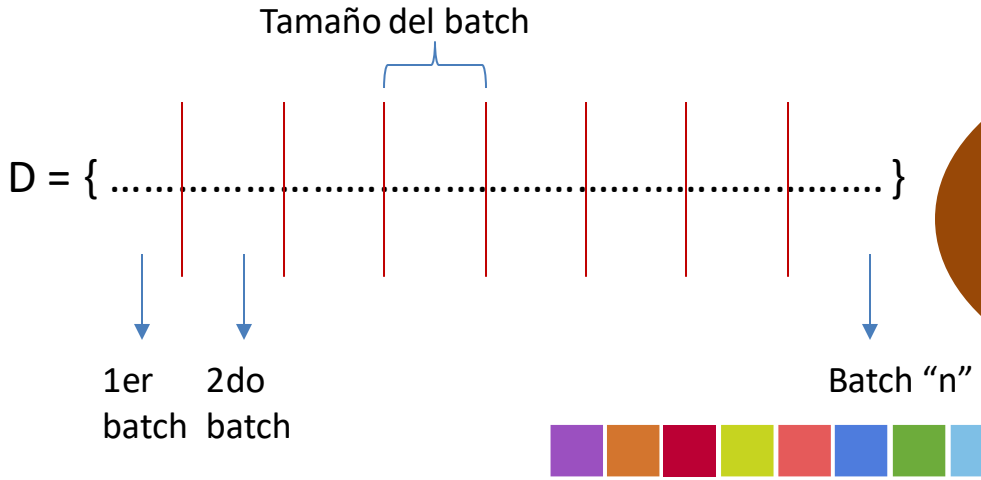
$$\hat{U} = \hat{U} - \lambda \frac{d\mathcal{L}}{dU}\Big|_{\hat{\theta}}$$

$$\hat{c} = \hat{c} - \lambda \frac{d\mathcal{L}}{dc}\Big|_{\hat{\theta}}$$

Descenso del Gradiente

En lugar de calcular el error promedio de todos los datos, lo que se hace es "estimar" el error. Para hacer esto, se toma un batch de los ejemplos y estos se usan para estimar el error (acá aparece el primer algoritmo de optimización que usa como base el descenso del gradiente, pero le introduce una innovación para hacerlo implementable, hacerlo más eficiente) este algoritmo se llama descenso estocástico del gradiente (SGD).
En otras palabras, no se espera a que pasen todos los datos para actualizar los parámetros, sino que se hace sobre una porción de ellos.
Es importantísimo que este batch sea al "azar"

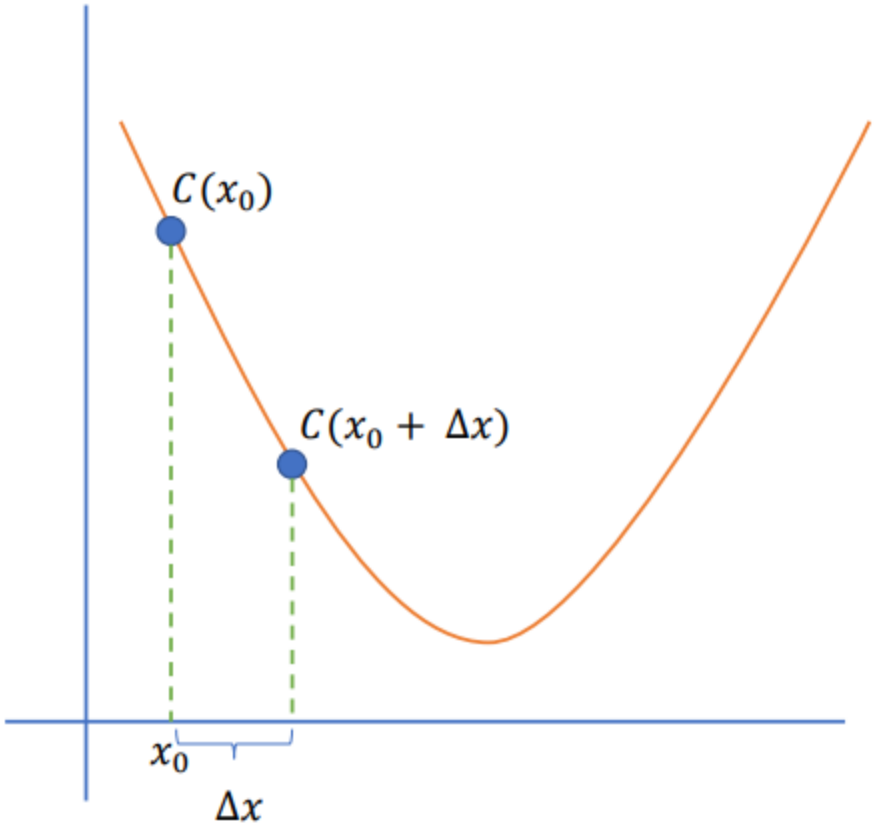
- En la práctica, se toman todos los ejemplos, se "**desordenan**", se ponen en una lista, se define el tamaño del batch… y dividir el set de datos en el tamaño de este batch
- Se toma el primer batch, se pasa por la red, se calcula el error y se ajutan los parámetros y luego…
- Se toma el segundo batch, se pasa paso por la red, se calcula el error y se ajustan los parámetros
- Y así sucesivamente hasta pasar todos los batches



Una vez que paso todos los batches por la red se completa una "época"

DuocUC®

Gradiente descendente



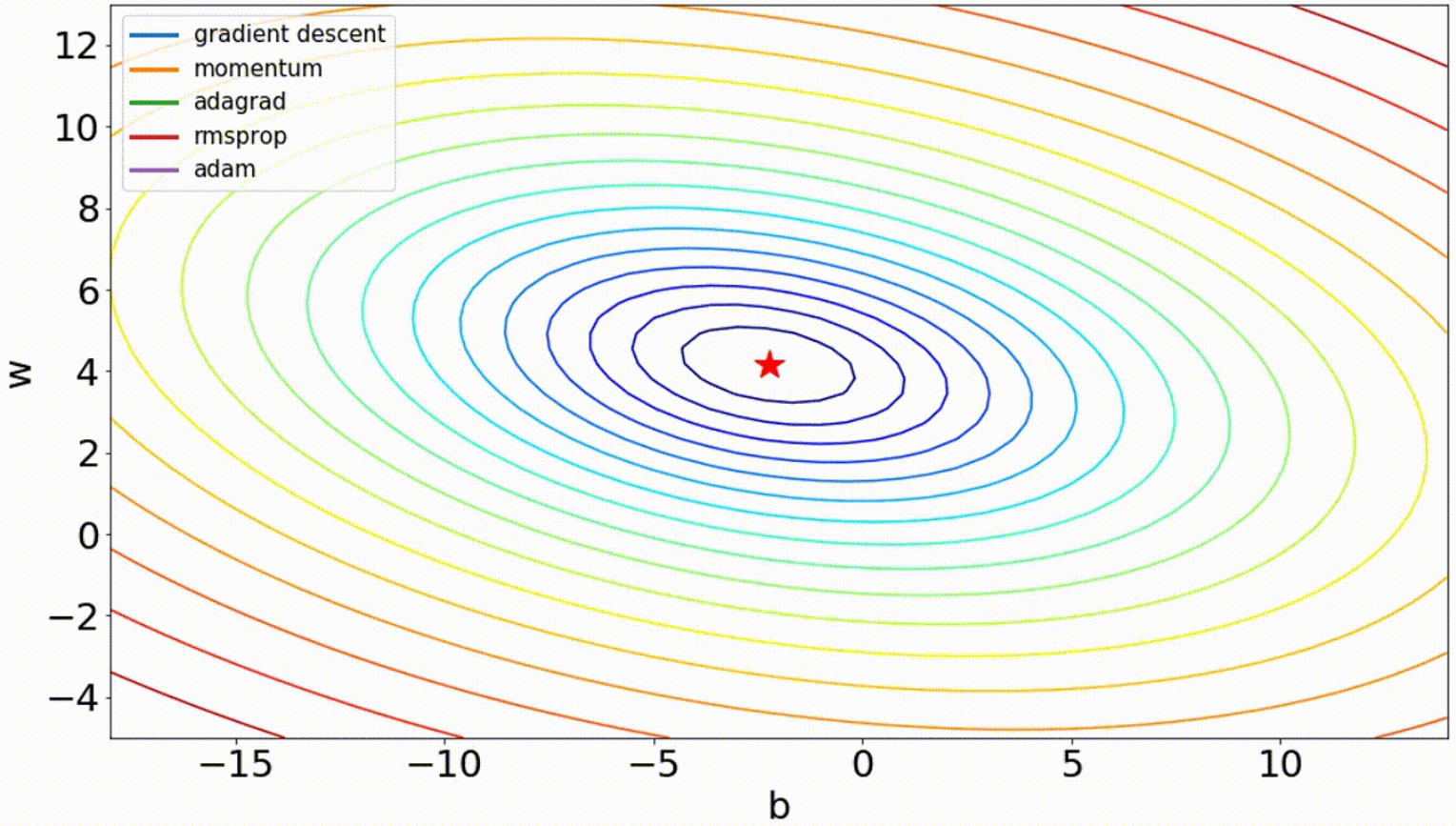
Derivada de C en el punto x_0

$$\frac{\partial C}{\partial x} = \frac{C(x_0 + \Delta x) - C(x_0)}{\Delta x} = \frac{\Delta C}{\Delta x}$$

$$\Delta C = \frac{\partial C}{\partial x} \Delta x$$

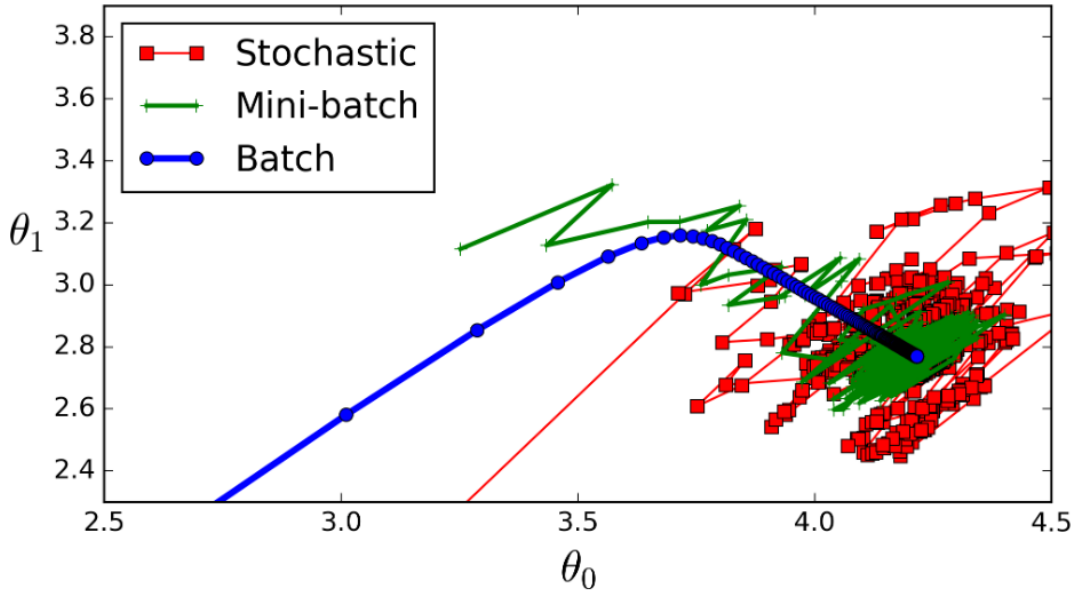
El cambio en la función depende de la derivada de la función y el cambio en los parámetros.

Descenso del Gradiente (Optimizadores)



Revisión Notebook

- **Ventajas** de usar un lote menor al número total de datos:
  - Menor uso de memoria
  - Entrenamiento más rápido
- **Desventajas**
  - La estimación del gradiente es menos precisa



Batch: utilizando todo el conjunto de datos (batch_size=60000).
Mini-batch: utilizando un conjunto más pequeño (batch_size <60000).
Stochastic: Utilizando batch_size = 1
https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

Consideraciones

- Si el error no baja, se debe ajustar levemente la tasa de aprendizaje
- Darle más “capacidad” a la red (aumentar capas o hacer las capas más grandes)
- Ya mencionamos SGD, pero también hay otros ejemplos de estos algoritmos: ADAM, Momentum, RmsProp entre otros Optimizers (keras.io)
- Esto algoritmos, buscan de manera dinámica la “buena” tasa de aprendizaje( ), un hiperparámetro clave en el entrenamiento de las redes neuronales profundas, como lo es también, el tamaño del paquete (batch) de los ejemplos