



Aprendizaje Automático Profundo (Deep Learning)

Dr. Facundo Quiroga - Dr. Franco Ronchetti



Descenso de Gradiente Estocástico

Motivación: Coste computacional de Descenso de Grad.

- Elementos
 - N ejemplos
 - P parámetros
 - I iteraciones
- En cada iteración
 - **Cálculo de $\delta E / \delta w_i$, $i=1, \dots, P$**
 - Asumo *mejor caso*
 - Lineal en N y P: **$O(NP)$**
 - **Actualización de w_i , $i=1, \dots, M$**
 - Lineal en la P: **$O(P)$**
- **Coste total**
 - $O(INP) + O(IP)$
 - **$O(INP)$**

Iteración 1

Horas	Nota
2	1
5	3.2
7	4.5
9	6
10	4
11	4.5
13.4	5.5
14	3
15	5



$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

Iteración 2

Horas	Nota
2	1
5	3.2
7	4.5
9	6
10	4
11	4.5
13.4	5.5
14	3
15	5



$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

Motivación: Coste computacional de Descenso de Grad.

- Elementos
 - N ejemplos
 - P parámetros
 - I iteraciones
- En cada iteración
 - **Cálculo de $\delta E / \delta w_i$** , $i=1, \dots, P$
 - Asumo *mejor caso*
 - Lineal en N y P: **$O(NP)$**
 - **Actualización de w_i** , $i=1, \dots, M$
 - Lineal en la P: **$O(P)$**
- **Coste total**
 - $O(INP) + O(IP)$
 - **$O(INP)$**

```
def descenso_gradiente(E,w,x,y)
    W = ...
    P = len(w)
    converge=False
    while not converge:
        for i in range(P):
             $\delta E \delta W[i]$  = derivada(E,i,x,y)
        for i in range(P):
            w[i] = w[i] -  $\delta E \delta W[i]$ 
        converge = ... (depende)
    return w
```

Entrenamiento por lotes

- Coste total **$O(INP)$**
 - Cálculo de derivadas **$O(NP)$**

- **Idea**

- Dividir datos en **lotes**
 - de **$B \ll N$** ejemplos
- Calcular derivadas con **lotes**

- Gradiente aproximado

- Ruidoso

- Pero funciona

- Cálculo de derivadas

- **$O(BP)$**

- Ejemplo con **$B=3$**

- Épocas = recorridos del conjunto de datos
- Iteraciones = act. de los **w_i**

Época 1

Horas	Nota
2	1
5	3.2
7	4.5
9	6
10	4
11	4.5
13.4	5.5
14	3
15	5

Iteración 1
(lote 1)

2	1
5	3.2
7	4.5

Iteración 2
(lote 2)

9	6
10	4
11	4.5

Iteración 3
(lote 3)

13.4	5.5
14	3
15	5

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

Época 2

Horas	Nota
2	1
5	3.2
7	4.5
9	6
10	4
11	4.5
13.4	5.5
14	3
15	5

Iteración 1
(lote 1)

2	1
5	3.2
7	4.5

Iteración 2
(lote 2)

9	6
10	4
11	4.5

Iteración 3
(lote 3)

13.4	5.5
14	3
15	5

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

$$\begin{aligned}w_1 &= w_1 - \delta E / \delta w_1 \\w_2 &= w_2 - \delta E / \delta w_2 \\&\dots \\w_p &= w_p - \delta E / \delta w_p\end{aligned}$$

Descenso de gradiente Clásico vs Por lotes

- Asumimos
 - model.w = matriz con todos los parámetros de la red
 - δw = matriz con la derivada de respecto a model.w
- Descenso de gradiente Clásico

```
x,y = load_data()

lr = 0.001 # learning rate
model = ...
epochs = 1000

for i in range(epochs):
     $\delta w = \text{model.derivatives}(x,y)$ 
     $\text{model.w} = \text{model.w} - \text{lr} * \delta w$ 
```

- Descenso de gradiente por lotes

```
x,y = load_data()

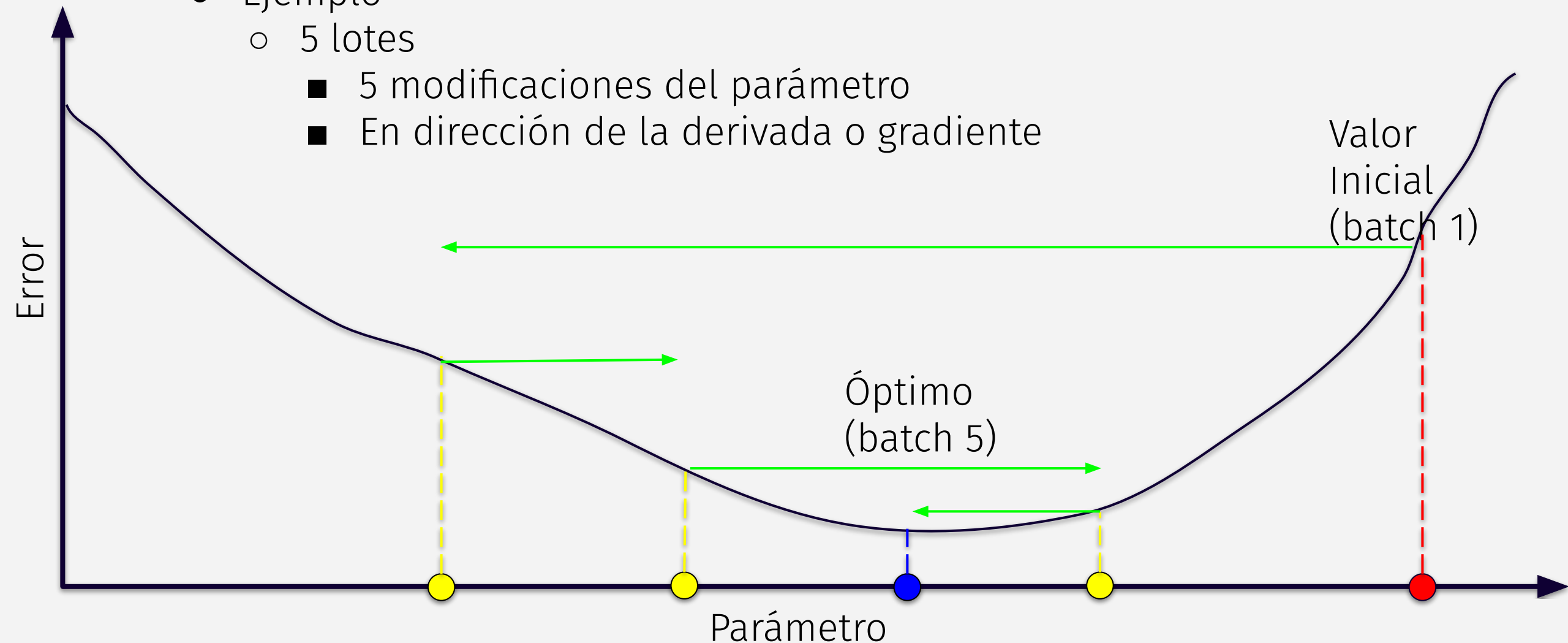
lr = 0.001 # learning rate
model = ...
epochs = 1000

n = x.shape[0]
batch_size = 32
batches = n // batch_size

for i in range(epochs):
    for batch in range(batches):
        batch_x, batch_y = get_batch(x,y,batch)
         $\delta w = \text{model.derivatives}(\text{batch\_x}, \text{batch\_y})$ 
         $\text{model.w} = \text{model.w} - \text{lr} * \delta w$ 
```

Descenso de gradiente por lotes

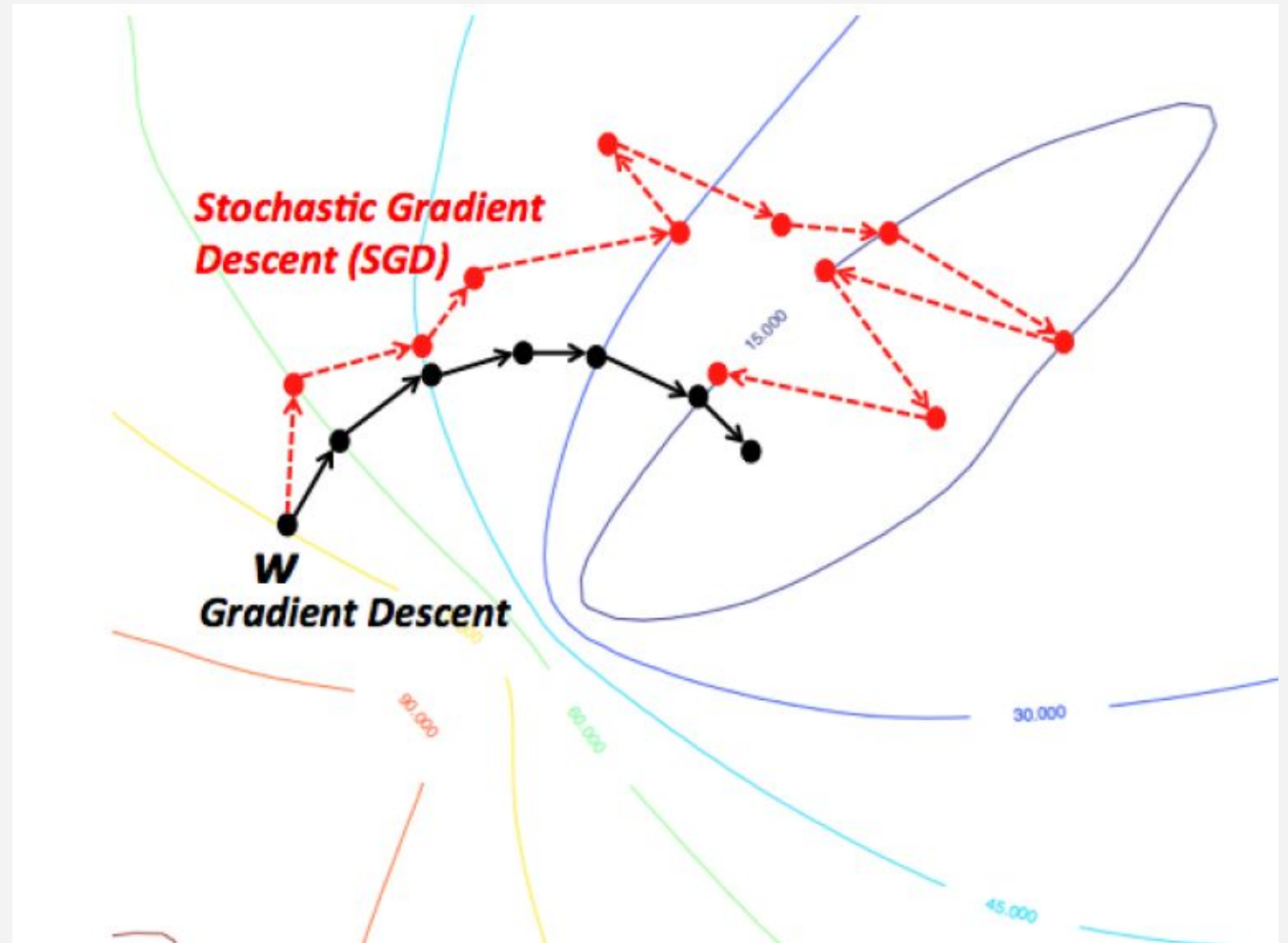
- Ejemplo
 - 5 lotes
 - 5 modificaciones del parámetro
 - En dirección de la derivada o gradiente



Coste computacional: Descenso de gradiente

estocástico

- Descenso clásico
 - Coste de iteración
 - **$O(NP)$**
- Descenso por lotes
 - Coste de iteración
 - **$O(BP)$**
 - **B** es un hiperparámetro
- Si **$B = N$**
 - Igual que clásico
- Si **$B = 1$**
 - Gradientes demasiado ruidoso
- Si $B =$ intermedio
 - Gradientes ruidosos
 - Pero buenos
 - Más pasos/iteraciones
 - Pero más rápidas

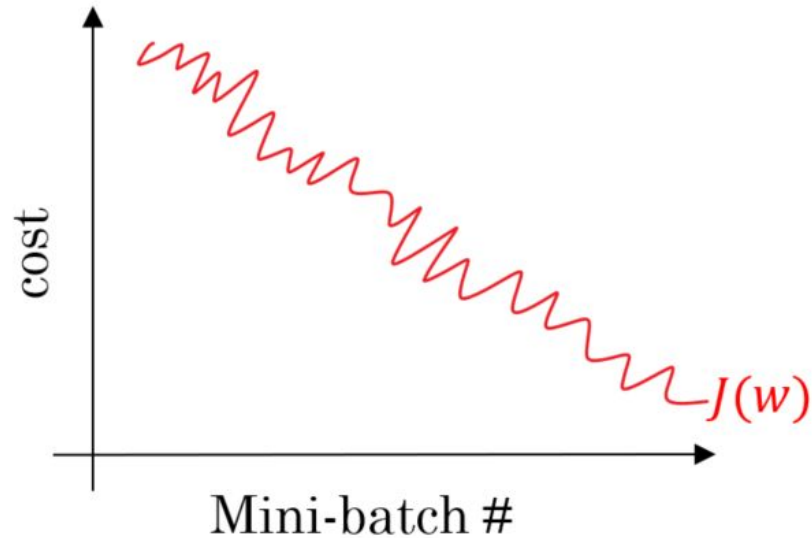


Por lotes = ruidoso = **estocástico**

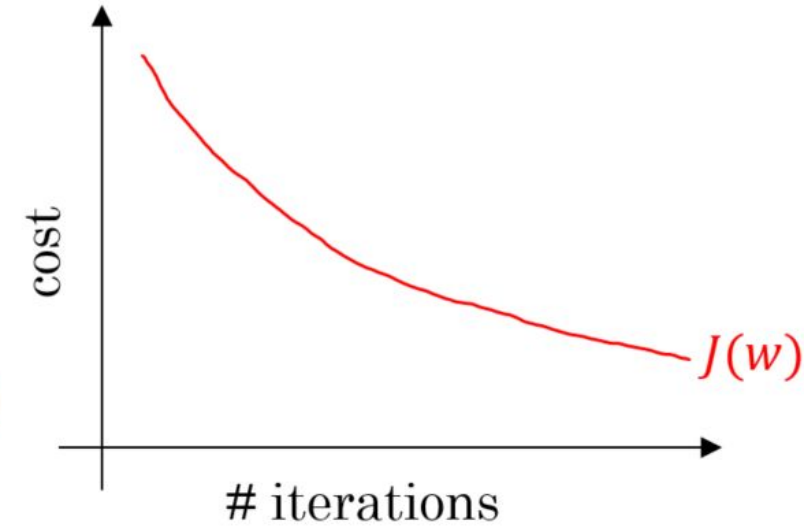
Descenso de Gradiente Estocástico - Efecto

- Error (cost)
 - Cambia de forma “ruidosa”
 - Algunas iteraciones aumentan el error
 - Pero decrece en promedio
- (minibatch = estocástico)
- (batch = clásico)

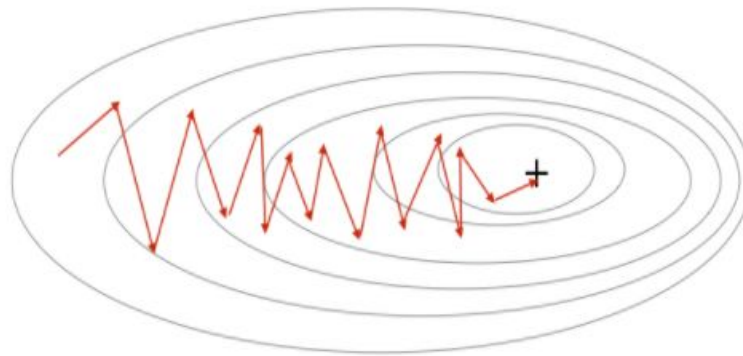
Mini-batch gradient descent



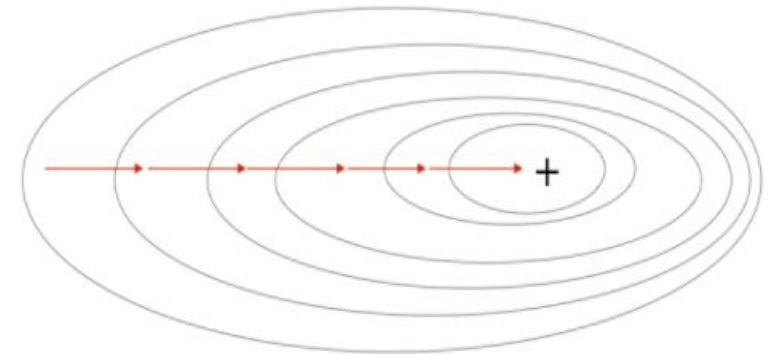
Batch gradient descent



Stochastic Gradient Descent

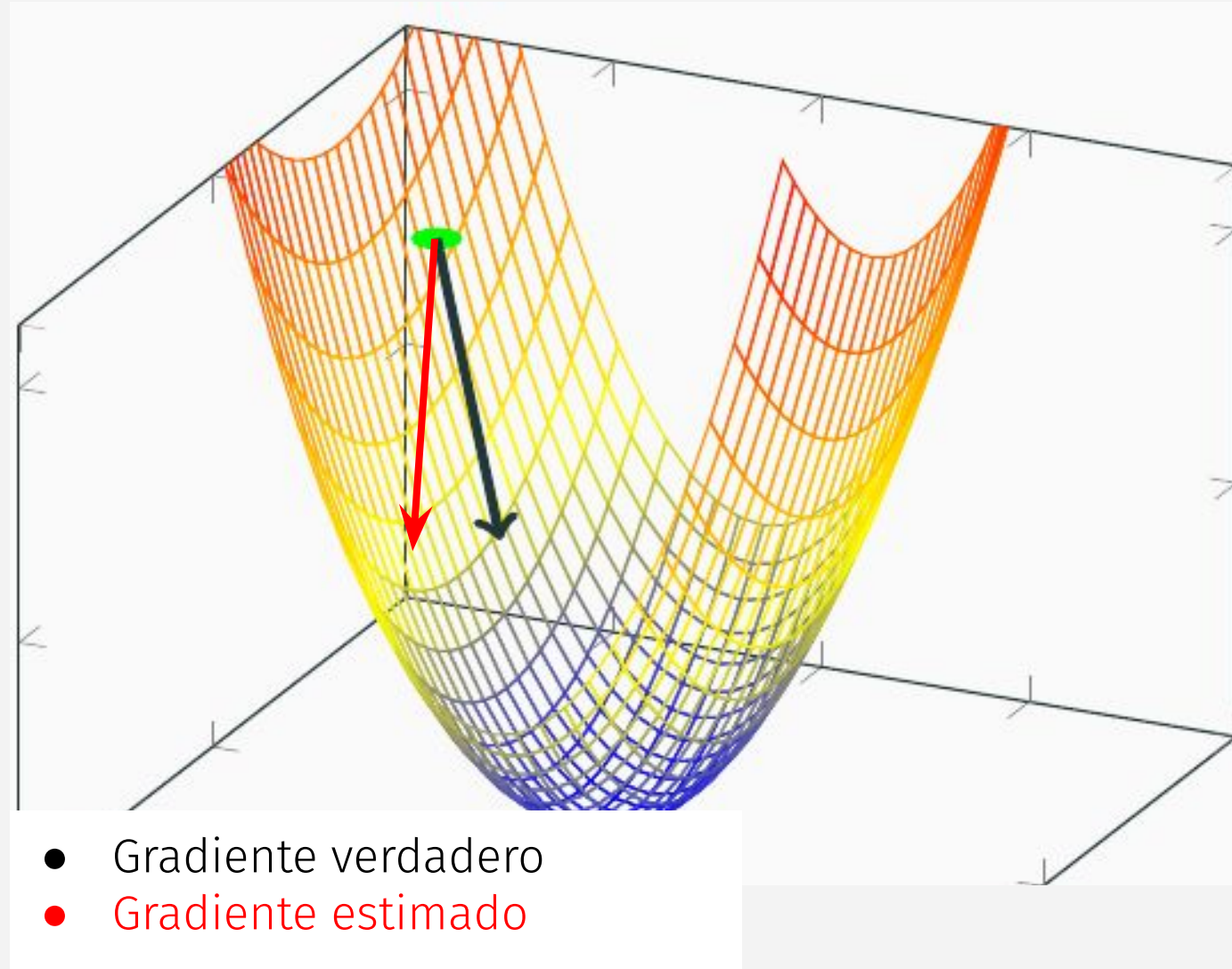


Gradient Descent



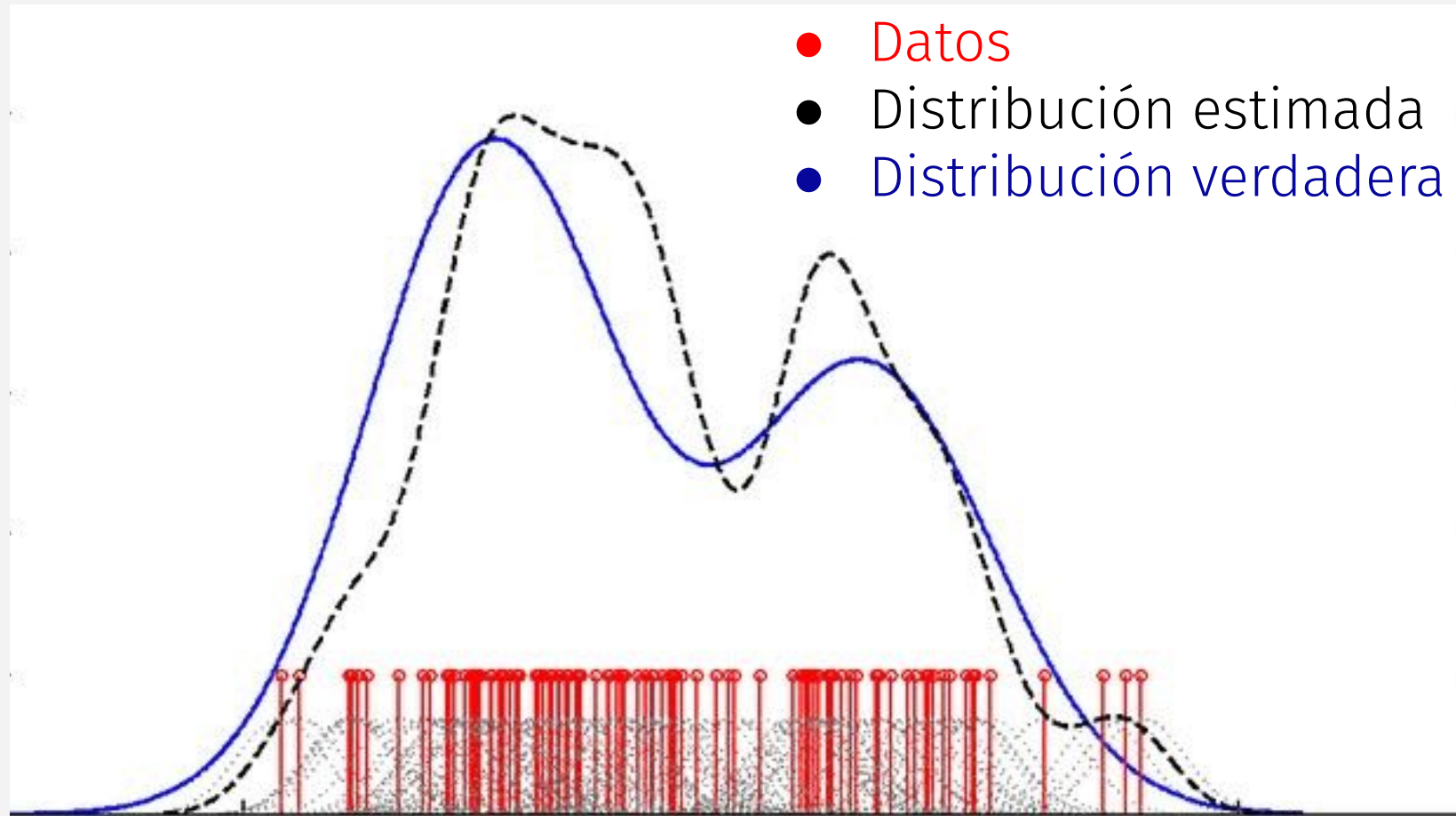
¿Por qué funciona?

- **Descenso de Gradiente Clásico**
 - Utiliza todas las **muestras**
 - Gradiente es exacto
 - Respecto de las **muestras**
 - Siguen siendo **muestras**
 - **No es exacto (es ruidoso)**
 - Respecto de la distribución
- **Descenso de Gradiente Estocástico**
 - Utiliza una **muestra** de la **muestra**
 - Gradiente **no** es **exacto**
 - **Significativo estadísticamente**
 - **Útil para entrenar**

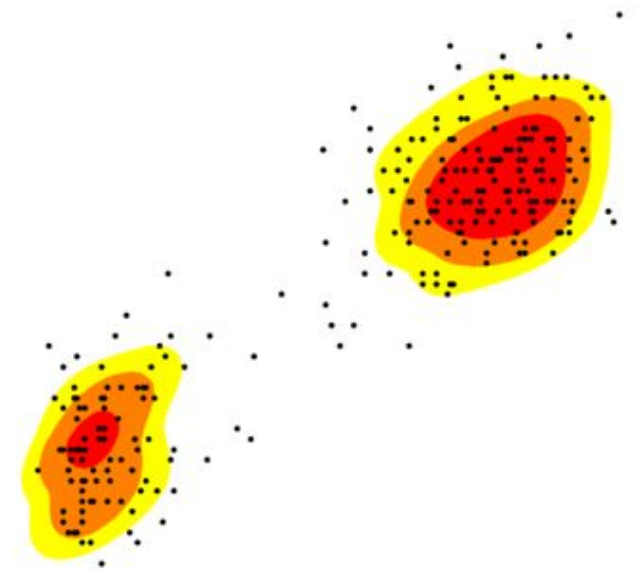


¿Por qué funciona?

- Distribuciones vs muestras
 - La **distribución** nunca se completamente
 - La estimamos a través de **muestras**

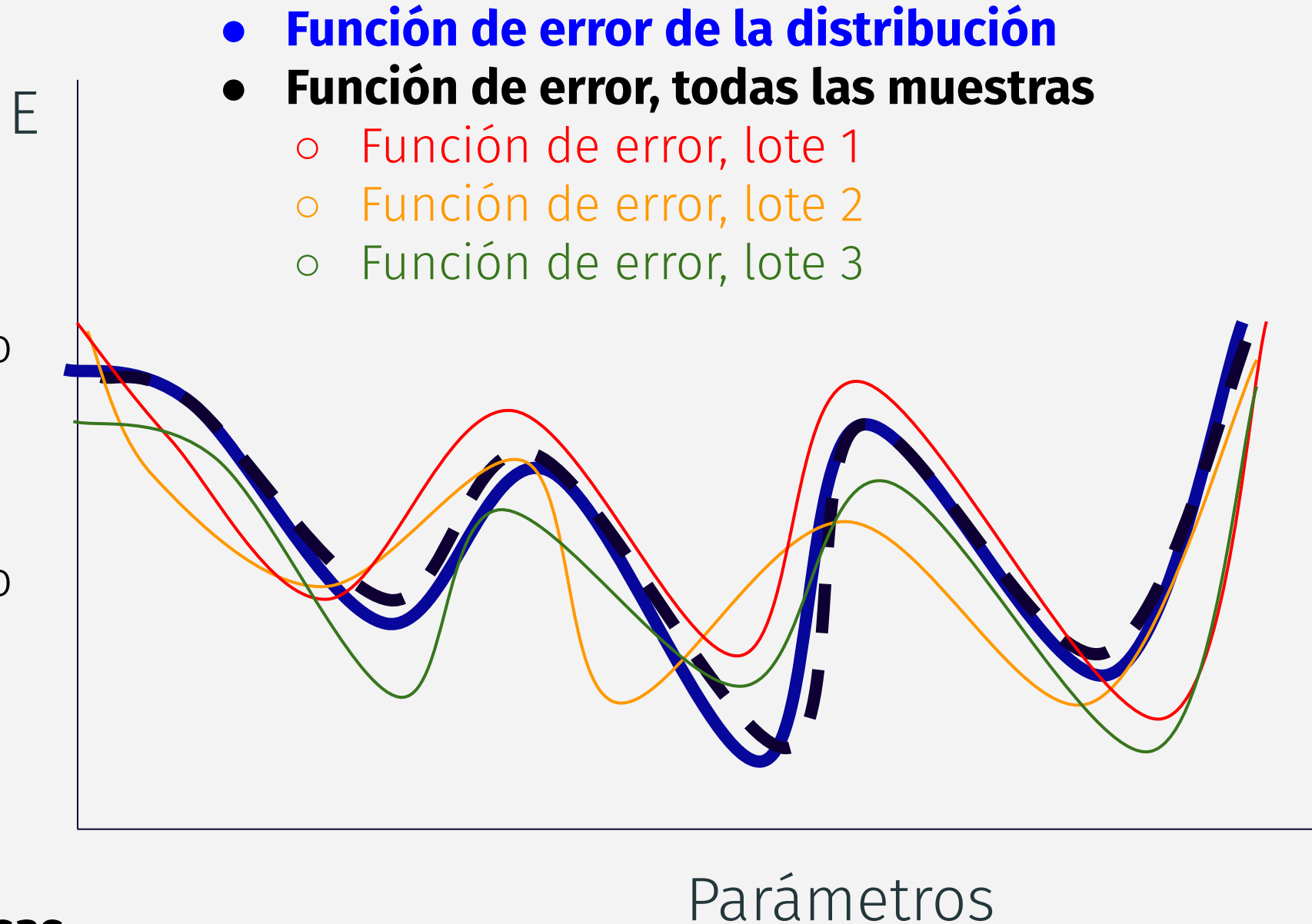


- Distribución estimada
- Datos



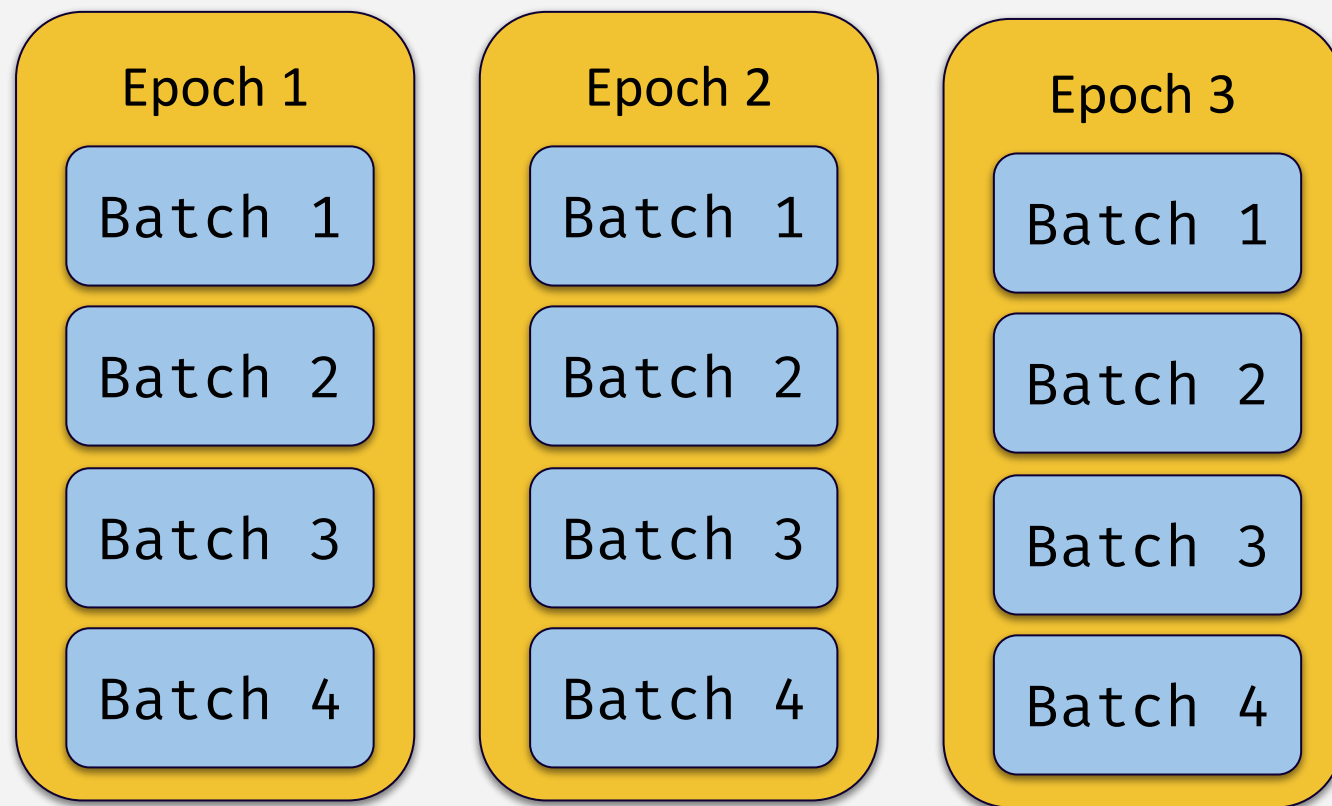
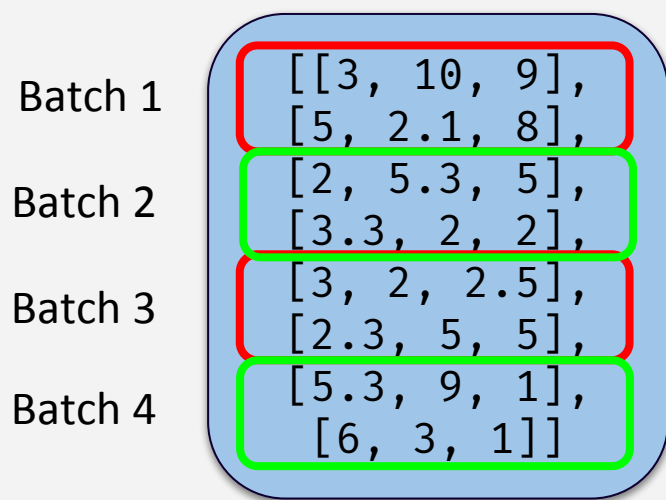
¿Por qué funciona?

- Función de error de la distribución
 - Imposible de conocer
- Función de error de la muestra completa
 - Utilizada por Descenso Clásico
- Función de error de cada lote
 - Utilizada por Descenso Estocástico
 - **Cada lote redefine la función de error**
 - Se **redefinen** las **derivadas**
 - **Equivalente: ruidosas**



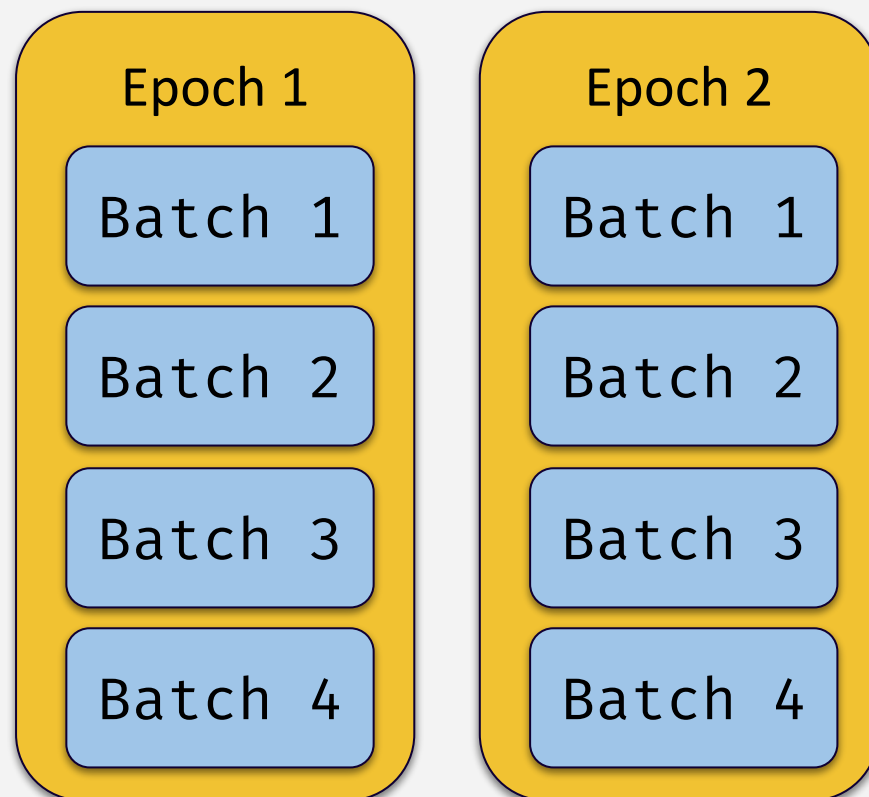
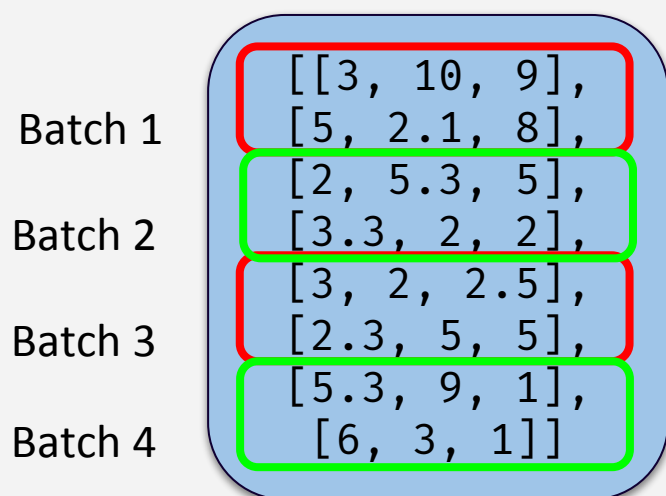
Épocas, tamaños de lote e iteraciones

- Épocas = epochs
 - Cantidad de iteraciones de entrenamiento
 - En relación al tamaño del dataset
- Ejemplo con **N=8, batch_size=2, y epochs = 3**
 - Iteraciones reales por epoch = 4
 - $N/\text{batch_size}$
 - Iteraciones reales totales = 12
 - $N/\text{batch_size} * \text{epochs}$



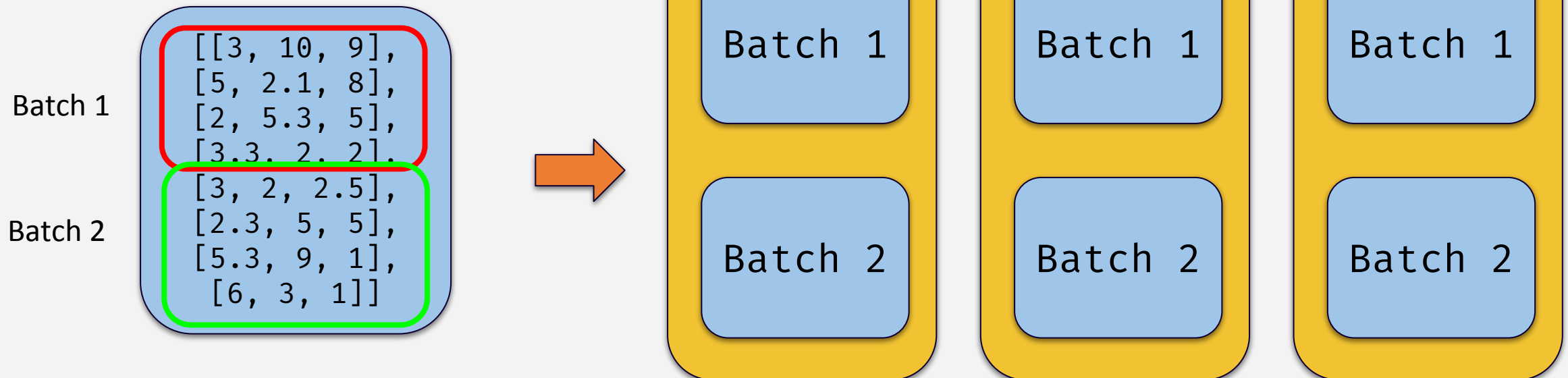
Épocas, tamaños de lote e iteraciones

- Épocas = epochs
 - Cantidad de iteraciones de entrenamiento
 - En relación al tamaño del dataset
- Ejemplo con **N=8, batch_size=2, y epochs = 2**
 - Iteraciones reales por epoch = 4
 - $N/\text{batch_size}$
 - Iteraciones reales totales = 8
 - $N/\text{batch_size} * \text{epochs}$



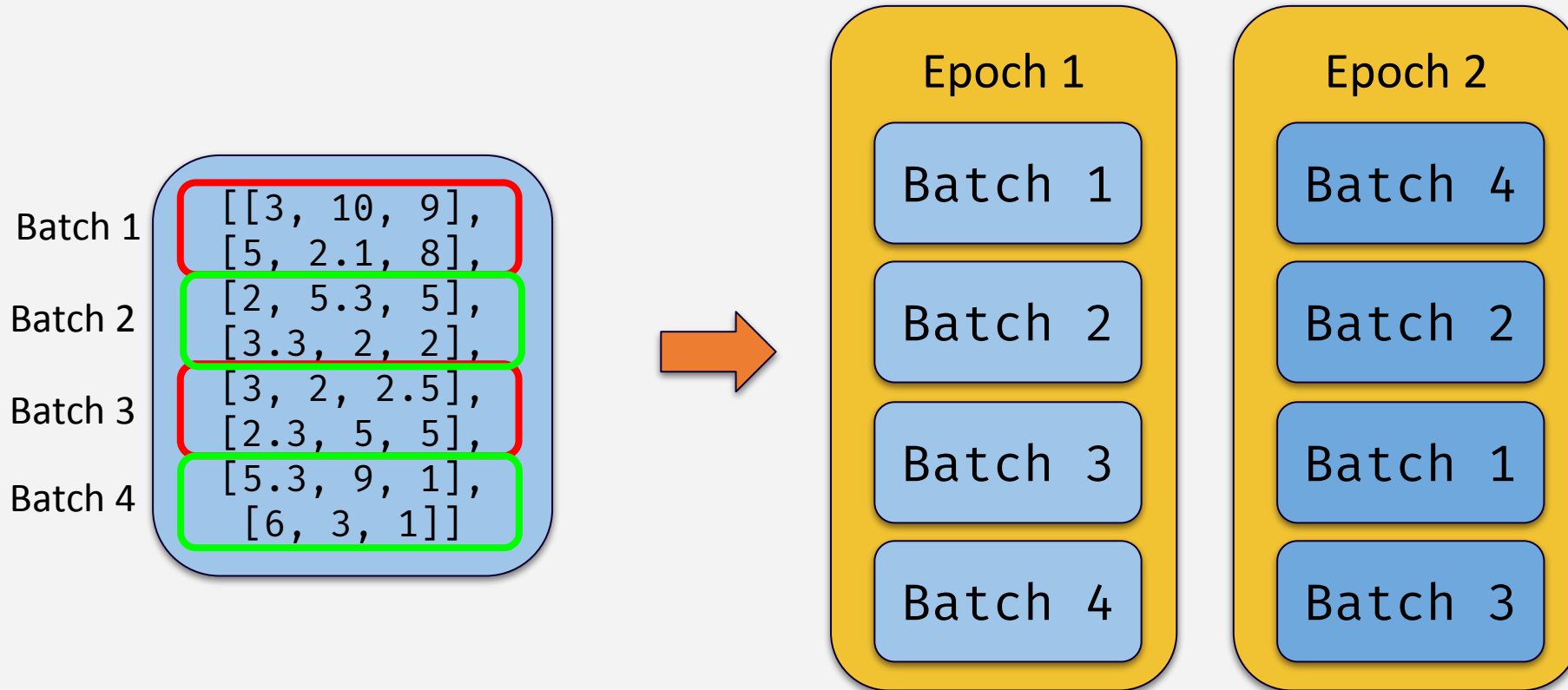
Épocas, tamaños de lote e iteraciones

- Épocas = epochs
 - Cantidad de iteraciones de entrenamiento
 - En relación al tamaño del dataset
- Ejemplo con **N=8, batch_size=4, y epochs = 3**
 - Iteraciones reales por epoch = 2
 - $N/\text{batch_size}$
 - Iteraciones reales totales = 6
 - $N/\text{batch_size} * \text{epochs}$



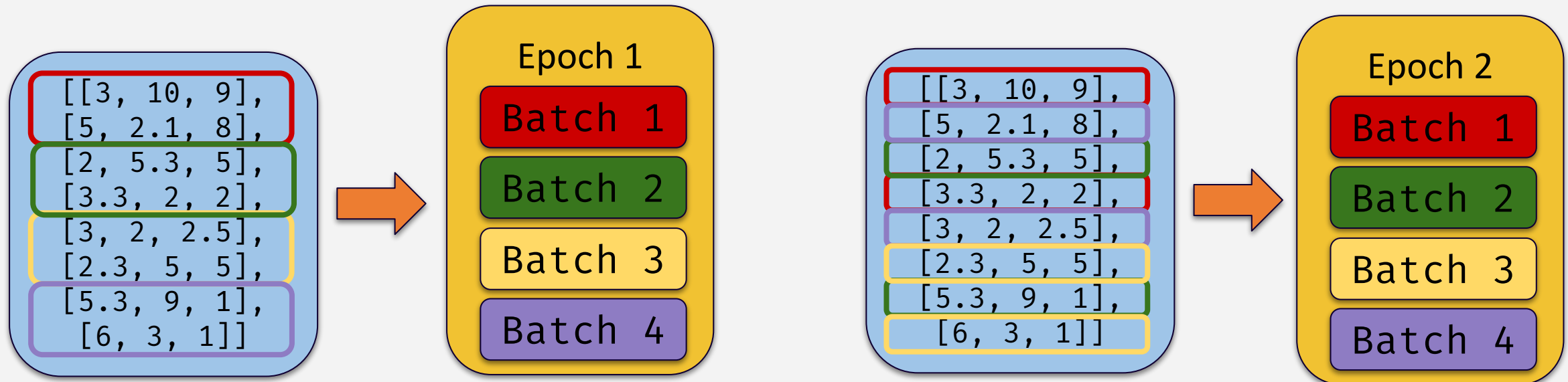
Desordenar lotes entre épocas

- Ayuda al entrenamiento
 - Genera ruido
 - Remueve efectos de orden



Desordenar ejemplos entre épocas

- Mismo objetivo que desordenar lotes
- Mismo efecto
 - Distinto orden de ejemplos
 - Distintos lotes
 - Aún más desorden



Paréntesis: nomenclatura

- Lote = Batch
 - Tamaño de lote = Batch Size = B
- Épocas = Epochs
 - Algunos las llaman **iteraciones** 🧐
- Descenso de Gradiente **Tradicional**
 - *También conocido como*
 - Descenso de Gradiente **Batch**
 - (Batch Gradient Descent)
 - Pero **NO** utiliza lotes 🧐
 - Proceso **Batch** vs **Online** (fábricas)

Paréntesis: nomenclatura

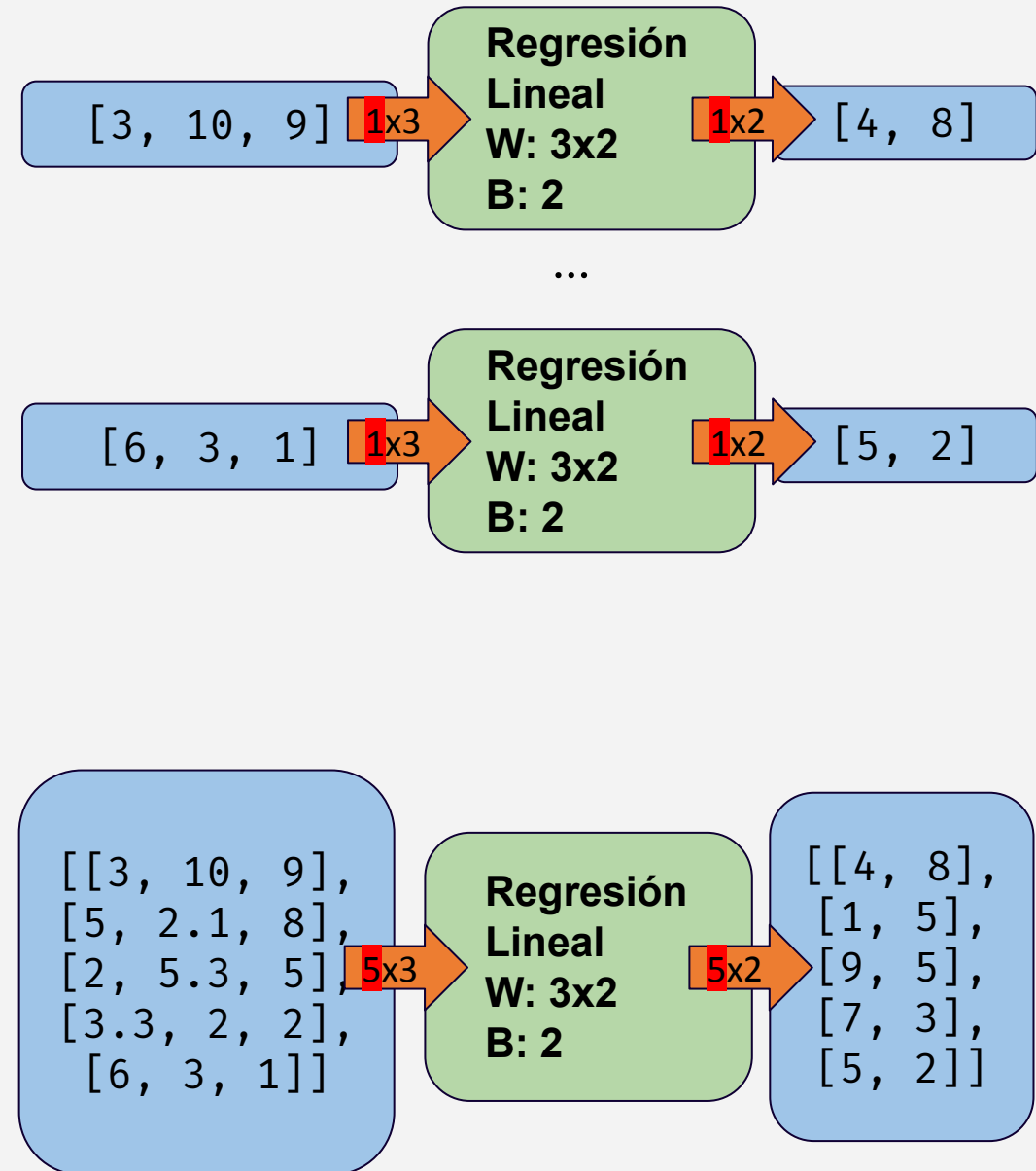
- Descenso de Gradiente **Estocástico**
 - Utiliza **lotes/batches**
 - *También conocido como*
 - Descenso de Gradiente **mini-batch** 🙋
- Descenso de Gradiente **Online**
 - Utiliza lotes de tamaño 1
 - $B = 1$
 - Simula reentrenar el modelo con nuevos ejemplos continuamente
 - *También conocido como*
 - Descenso de Gradiente **Estocástico** 🙋

Tamaño de lote

- **Predicción** por lotes
 - Más eficiente
 - Multiplicaciones de matrices
 - pre-alocación de memoria
 - Con GPUs:
 - Maximiza el uso de la memoria
 - Minimiza overhead copias GPU \leftrightarrow CPU
- Tamaño de lote
 - No se puede usar todo el dataset al mismo tiempo!
 - $N = O(\text{millones})$
 - No entra en memoria
- Generalmente **batch_size** = potencia de 2
 - 8, 16, 32, 64, 128, 256, 512
 - Más de 512 es redundante para entrenar

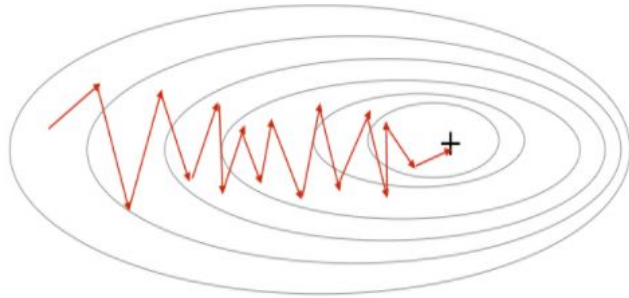
Predicción por lotes

- **Predicción de a un ejemplo**
 - Multiplicación de matrices muy pequeñas
 - Overhead de copias
 - Mala localidad de memoria
 - Con GPUs:
 - overhead copias GPU \leftrightarrow CPU
- **Predicción por lotes**
 - Resuelve estos problemas
 - Tamaño de lote
 - Tan grande como sea posible
 - Tamaño RAM

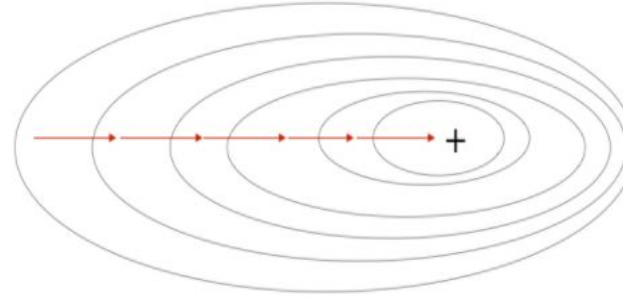


Resumen

Stochastic Gradient Descent



Gradient Descent



Descenso de gradiente tradicional	Descenso de gradiente Estocástico
Gradiente con todos los ejemplos	Gradiente con lotes de ejemplos
Gradientes más exactos	Gradientes ruidosos
No puede escapar de mínimos locales	Puede que escape de mínimos locales
Si f convexa, garantiza mínimo global	Mínimo global no garantizado
Lento, poco escalable	Rápido, escalable