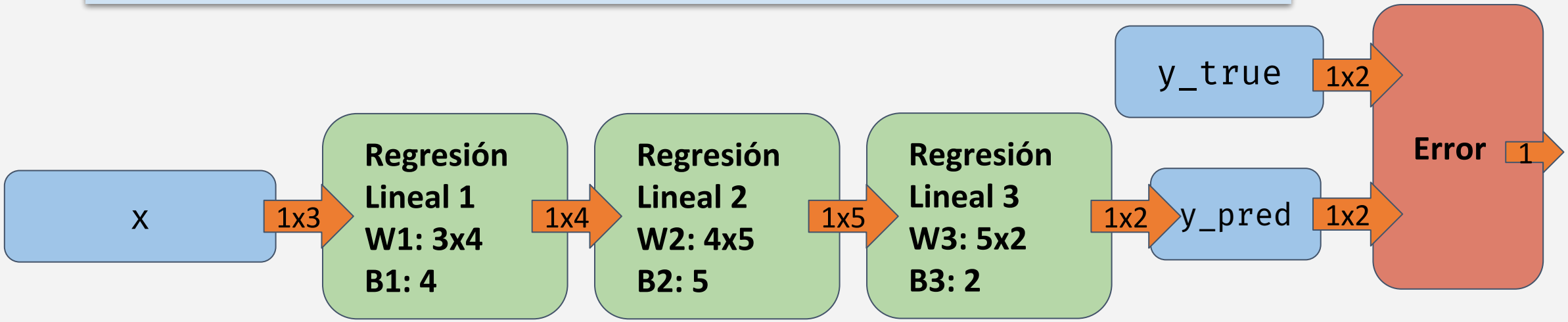
Aprendizaje Automático Profundo (Deep Learning)

Dr. Facundo Quiroga - Dr. Franco Ronchetti

Variantes de Descenso de Gradiente

¿Qué optimizamos?

- Al entrenar
  - **Bajamos** el **error**
  - **Cambiando** los **parámetros**
  - Matrices W1, W2, W3, B1, B2, B3
  - Cantidad de parámetros por matriz:
    - W1: 3*4 = 12, W2: 4x5 = 20, W3: 5x2 = 10
    - B1: 4, B2: 5, B3=2
      - 12+20+10+4+5+2 = 53 parámetros



Cuenta de parámetros

- # de parámetros en Keras

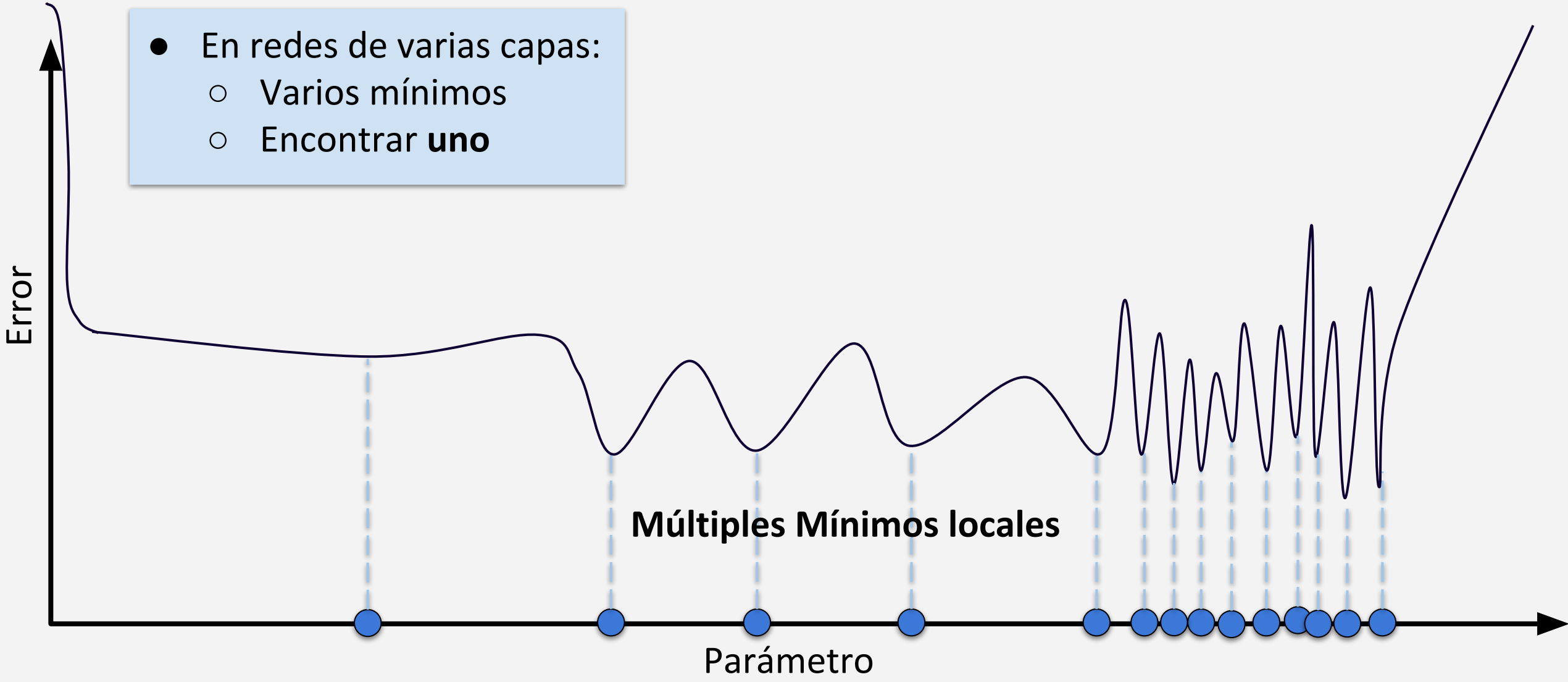
```
Layer (type)                 Output Shape              Param #
=================================================================
c1 (Conv2D)                  (None, 28, 28, 32)        320
mp1 (MaxPooling2D)           (None, 14, 14, 32)        0
c2 (Conv2D)                  (None, 7, 7, 64)          18496
mp2 (MaxPooling2D)           (None, 3, 3, 64)          0
flatten_7 (Flatten)          (None, 576)               0
fc1 (Dense)                  (None, 512)               295424
fc2 (Dense)                  (None, 10)                5130
=================================================================
Total params: 319,370
Trainable params: 319,370
Non-trainable params: 0
```

Caso convexo

- Óptimo global sólo con:
  - Regresión Lineal (1 capa)
  - Regresión Logística (1 capa)

Caso no convexo

- En redes de varias capas:
  - Varios mínimos
  - Encontrar **uno**



Descenso de gradiente - Versiones

- Descenso de gradiente tradicional

```
x,y = load_data()

α = 0.001 # learning rate
model = ….
epochs = 1000

for i in range(epochs):
  δw = …
  w = w - α * δw
```

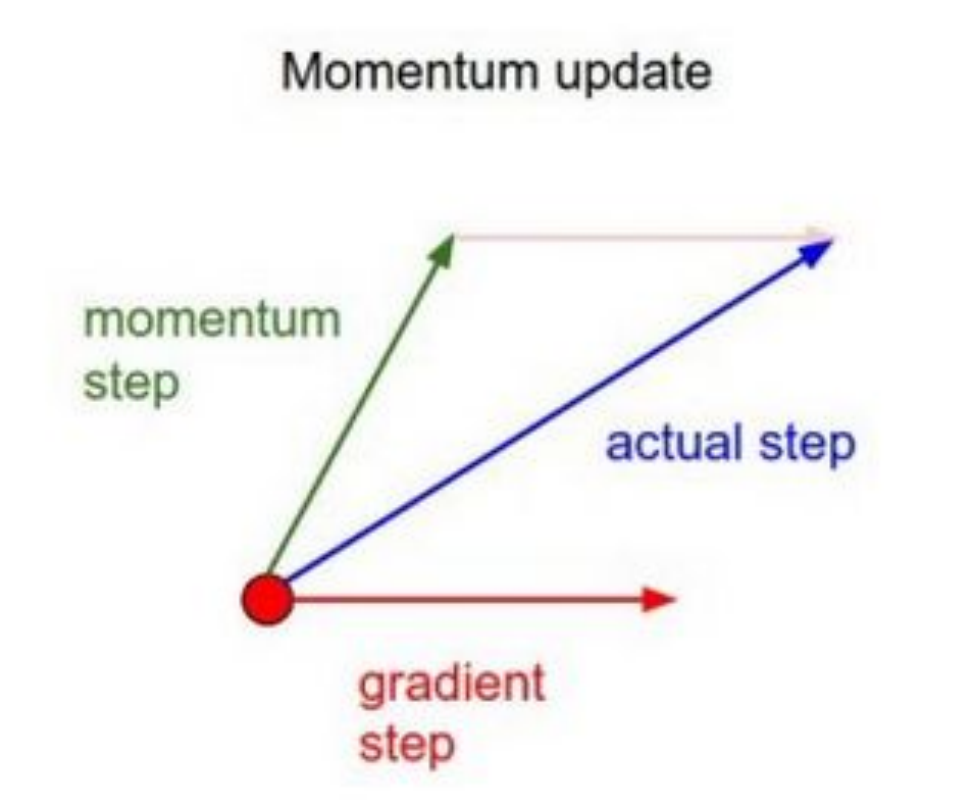
- Descenso de gradiente estocástico

```
x,y = load_data()
α = 0.001 # learning rate
model = ….
epochs = 1000

for i in range(epochs):
  for batch in batches:
    δw = …
    w = w - α * δw
```

Descenso de gradiente con Momentum

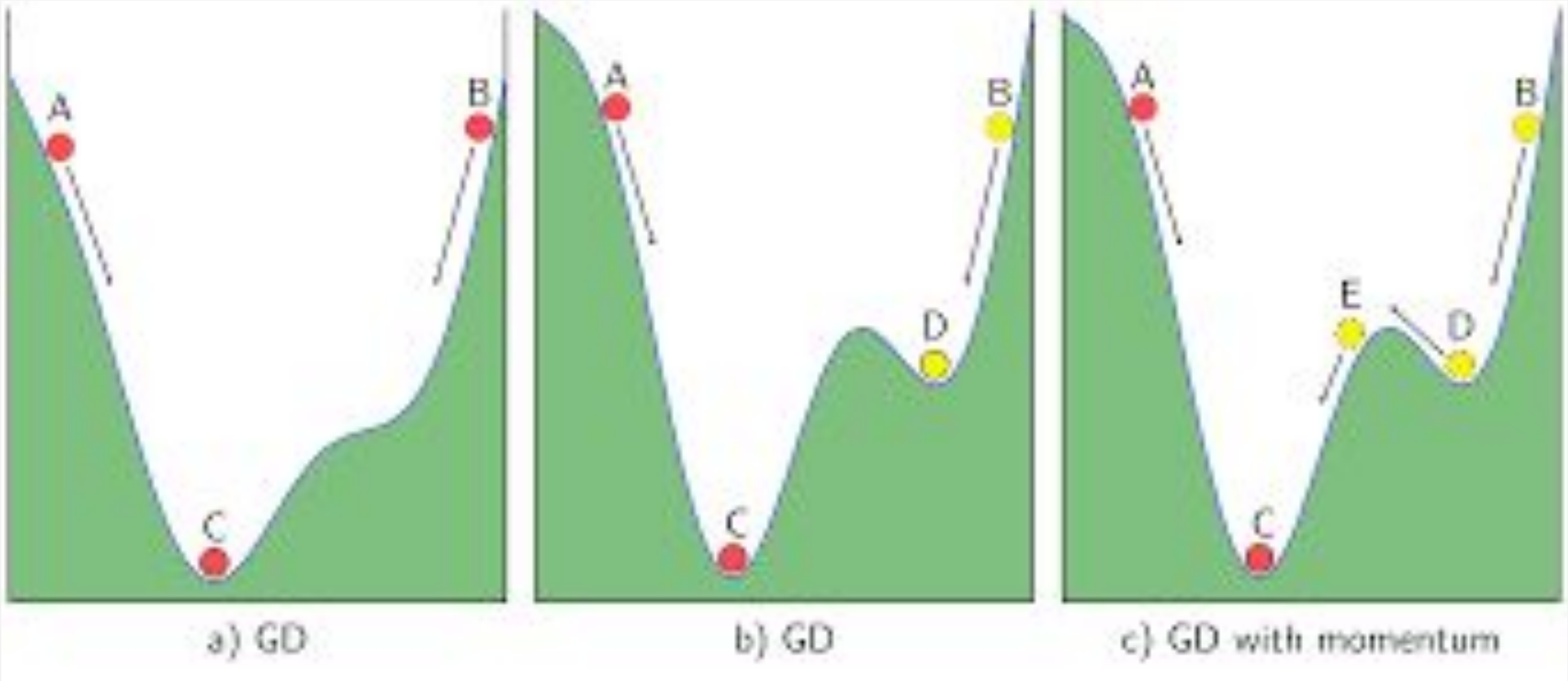
- Descenso de gradiente con velocidad o *momentum*
  - Matriz v de *momentum*
    - Dirección y velocidad promedio
  - Permite escapar mínimos locales “malos”




```
v=np.zeros(model.w.shape)
# mu: coef de fricción, entre 0 y 1
mu = 0.9
…
v = mu * v + alpha * dw
w = w - v
```

Descenso de gradiente con momentum

- Permite saltar mínimos locales *espurios* o “malos”



Optimizadores Avanzados: AdaGrad, RMSProp, Adam

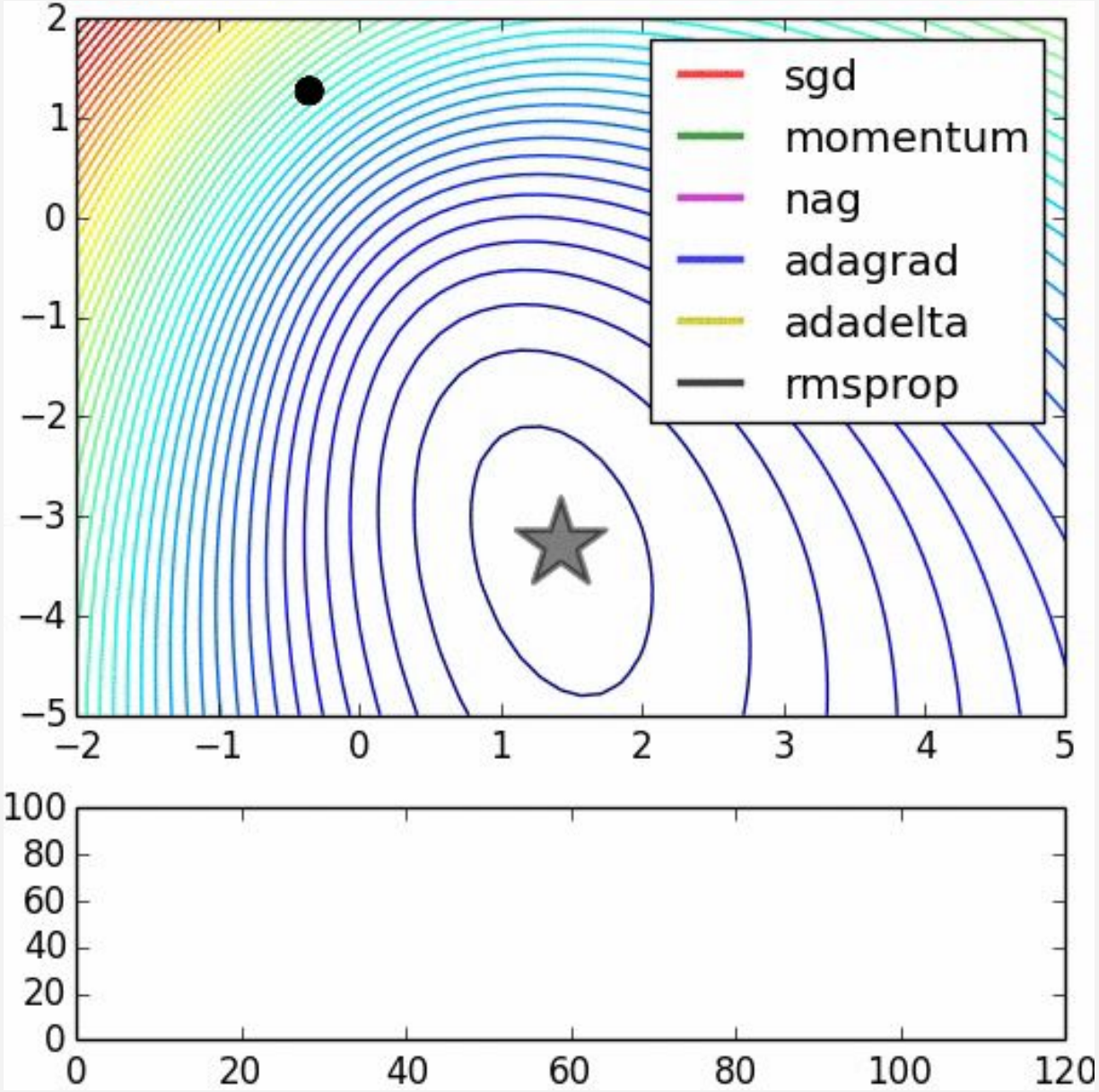
- AdaGrad
  - Normaliza la magnitud de los gradientes
    - Importa más la dirección
  - Mejora el comportamiento con gradientes muy bajos/altos

```
gradient_history=np.zeros_like(dw)
eps = 0.0001 # para evitar division por 0
...
gradient_history = gradient_history + dw**2
normalizing_factor = (np.sqrt(gradient_history)+eps)
w = w - (dw / normalizing_factor)
```

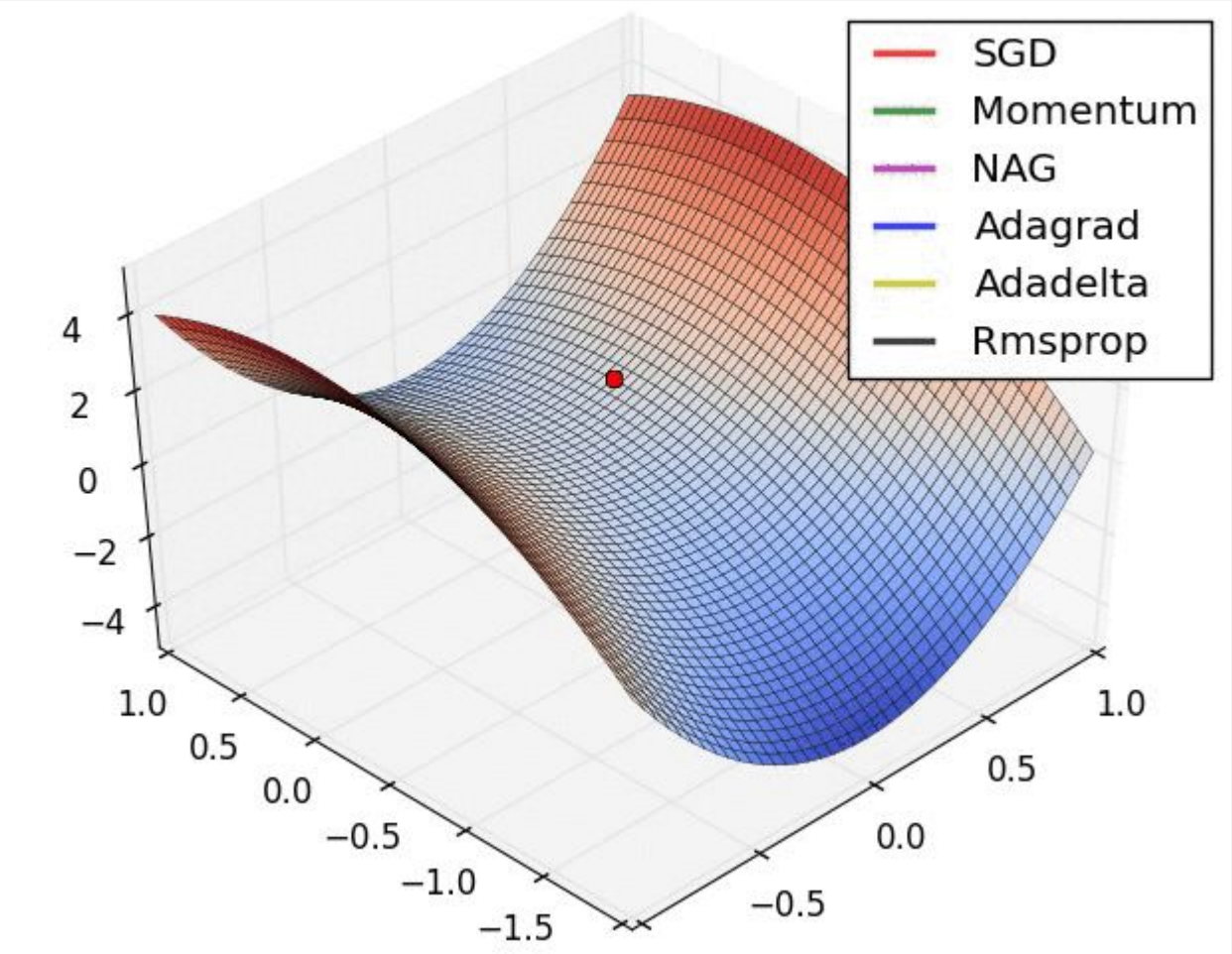
- ADAM
  - AdaGrad + Momentum
  - α no es tan importante
  - Algoritmo por defecto cuando no hay tiempo

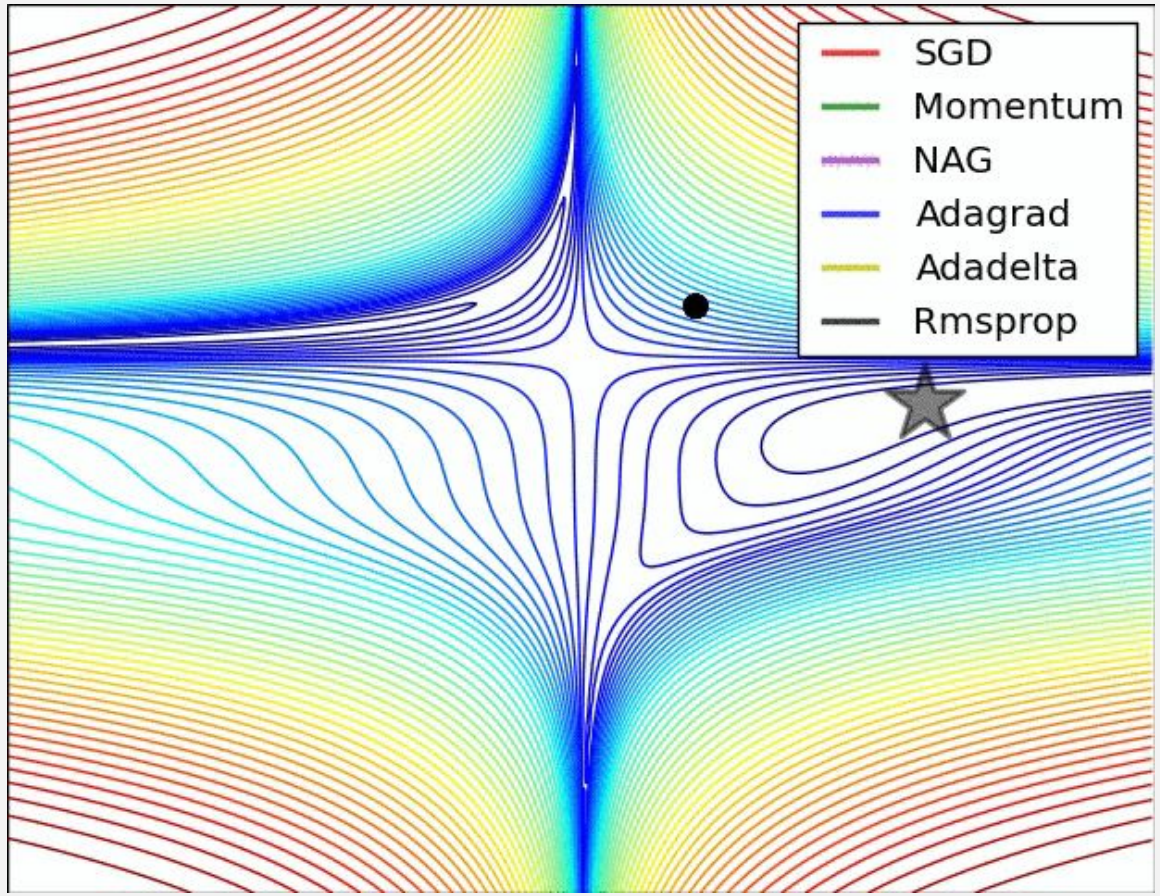
Optimizadores Avanzados

- Algoritmos más avanzados
  - Requieren menos pasos
  - Cada paso es más caro
    - más operaciones
    - más memoria
  - Adam es robusto y rápido
    - Utilizar para prototipar



Optimizadores Avanzados





Optimizadores Avanzados - En Keras

- Optimizadores modulares
  - Intercambiables
  - Desacoplados del cálculo de derivadas
  - Fácil implementar nuevos

```
opt = keras.optimizers.SGD(lr=0.01)
opt = keras.optimizers.SGD(lr=0.01, momentum=0.9)
opt = keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
opt = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9,
beta_2=0.999, amsgrad=False)
...
model.compile(...,optimizer=opt)
...
#alternativamente
model.compile(...,optimizer="sgd")
model.compile(...,optimizer="rmsprop")
model.compile(...,optimizer="adam")
```

Optimizadores Avanzados

- Descenso de gradiente tradicional
  - Necesita todos los ejemplos en memoria
- Descenso de gradiente estocástico (SGD)
  - Evita los problemas del tradicional
  - Sobreajusta menos
- Algoritmos de optimización avanzados
  - Intentan no depender de la tasa de aprendizaje
  - Más usados:
    - Momentum
    - Adagrad
    - Adam
  - Los modelos entrenados son ligeramente peores que con SGD