Entrenamiento de Redes Neuronales - Parte 2

Gonzalo Uribarri • Gabriel B. Mindlin

Sistemas dinámicos e inteligencia artificial aplicados al modelado de datos

@gonzauri

Objetivos de la práctica de hoy

- Definir Batch y Epoch.
- Conocer algunos Optimizadores disponibles
- Implementar un primer modelo de Regresión en Keras.
- Observar la dificultad de predicción a largo plazo en un sistema caótico.

Batches y Epochs

Vimos en la teórica que la red se entrena calculando el **gradiente** de los pesos respecto a la función de costo mediante el proceso de **Backpropagation**.

Vimos en la teórica que la red se entrena calculando el **gradiente** de los pesos respecto a la función de costo mediante el proceso de **Backpropagation**.

Si mi set de entrenamiento está compuesto por **N** instancias.

¿Cómo calculo este gradiente?

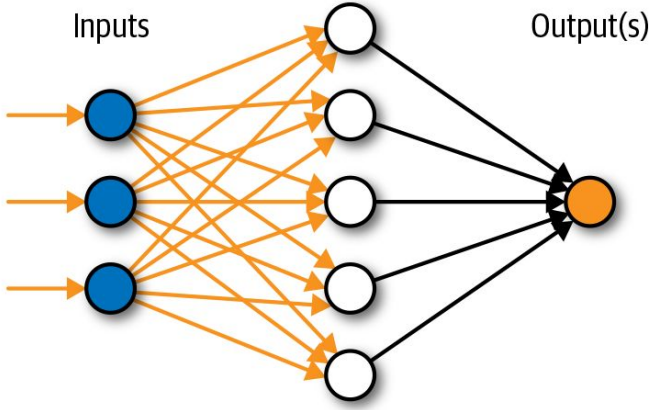
¿Instancia a instancia?

¿Promedio sobre todas?

Batches y Epochs

Batch

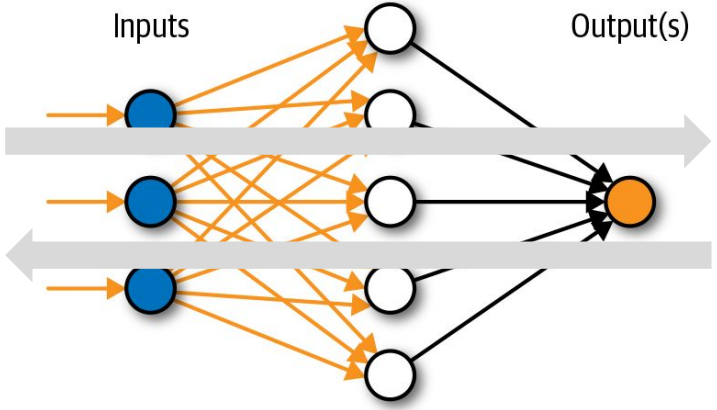
Conjunto de instancias.



Batches y Epochs

Batch

Conjunto de instancias.



Una **Iteración** (pasada): Computo el costo J, computo sus derivadas y actualizo los pesos de la red.

Batches y Epochs

Batch

Conjunto de instancias.

Batches y Epochs

Batch

Conjunto de instancias.

1 única instancia: **Stochastic**

m instancias (m<<N): **Mini-Batch**

N instancias (todo el training set): **Batch**

Batches y Epochs

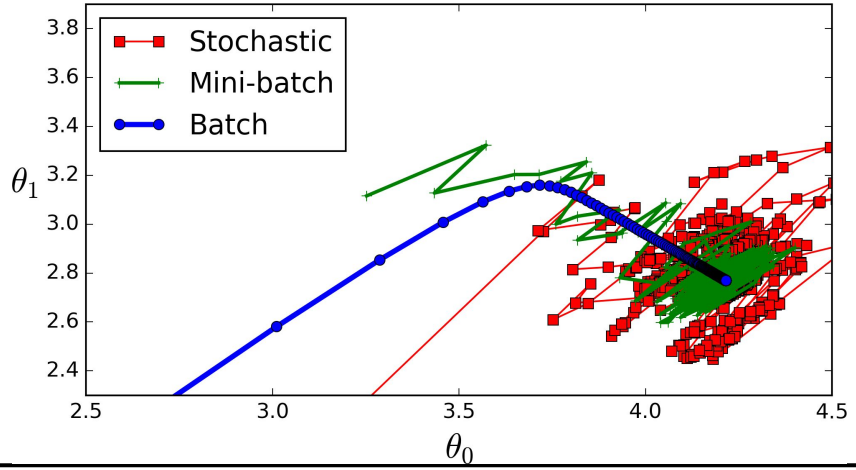
Batch

Conjunto de instancias.

1 única instancia: **Stochastic**

m instancias (m<<N): **Mini-Batch**

N instancias (todo el training set): **Batch**



Batches y Epochs

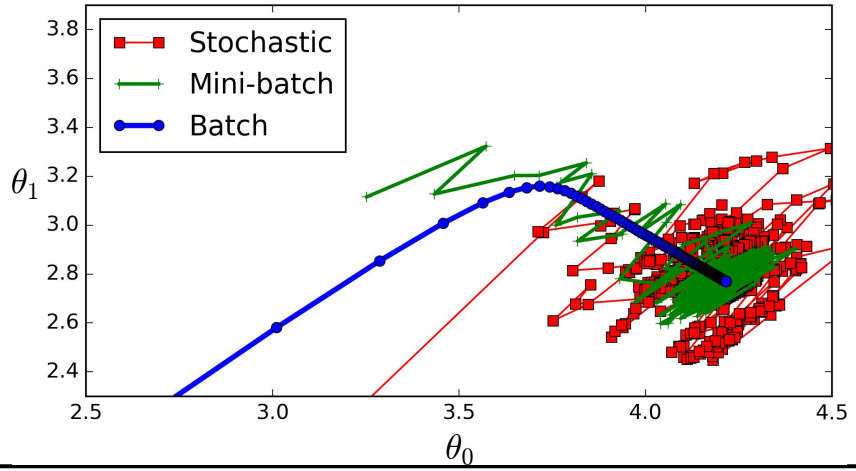
Batch

Conjunto de instancias.

1 única instancia: **Stochastic**

m instancias (m<<N): **Mini-Batch**

N instancias (todo el training set): **Batch**

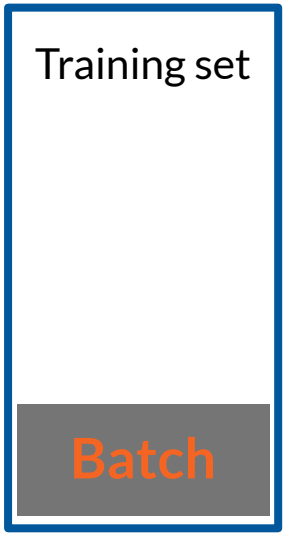


Batches y Epochs

Epochs: Es la cantidad de veces que pasamos **el training set completo** por la red.

Batches y Epochs

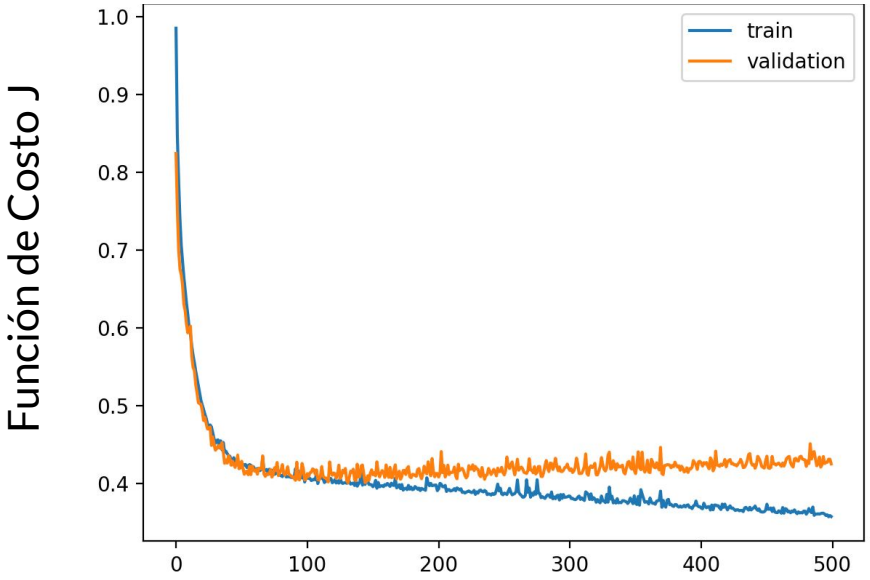
Epochs: Es la cantidad de veces que pasamos **el training set completo** por la red.



Noten que si el número de instancias **m** en el batch es mucho menor que el número de instancias **N** en todo el training set, vamos a necesitar varias **iteraciones** para completar un epoch.

Batches y Epochs

En general se precisan varios **Epochs** para entrenar la red.



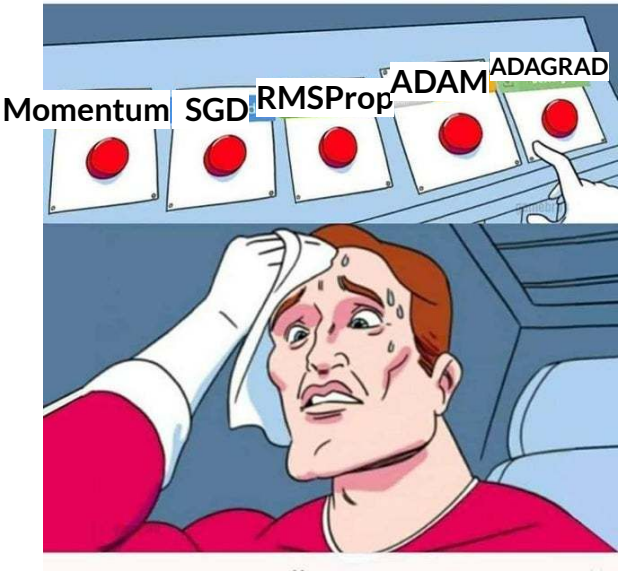
Optimizadores

Optimizadores

Existen **MUCHOS** métodos para actualizar los pesos, todos son algún tipo de variación del descenso por gradiente.

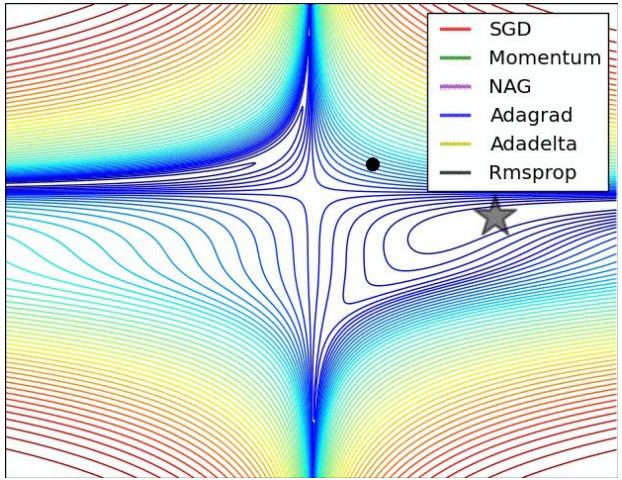
Optimizadores

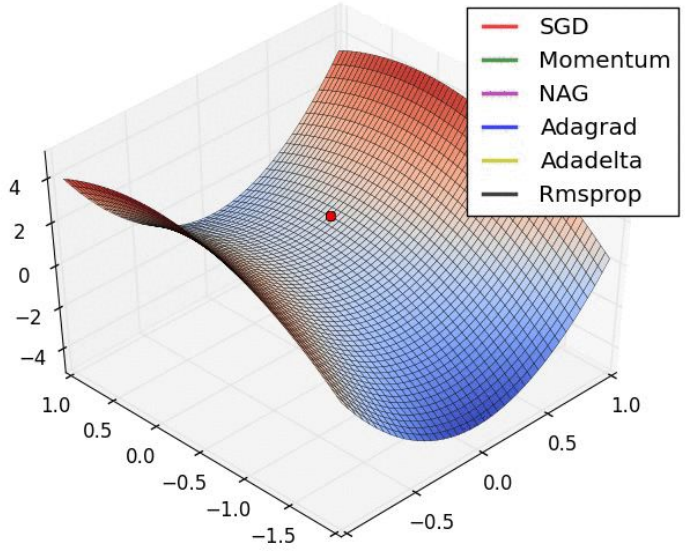
Existen **MUCHOS** métodos para actualizar los pesos, todos son algún tipo de variación del descenso por gradiente.



Optimizadores

Si les interesa, pueden leer mas al respecto en este artículo:
https://ruder.io/optimizing-gradient-descent/





Optimizadores

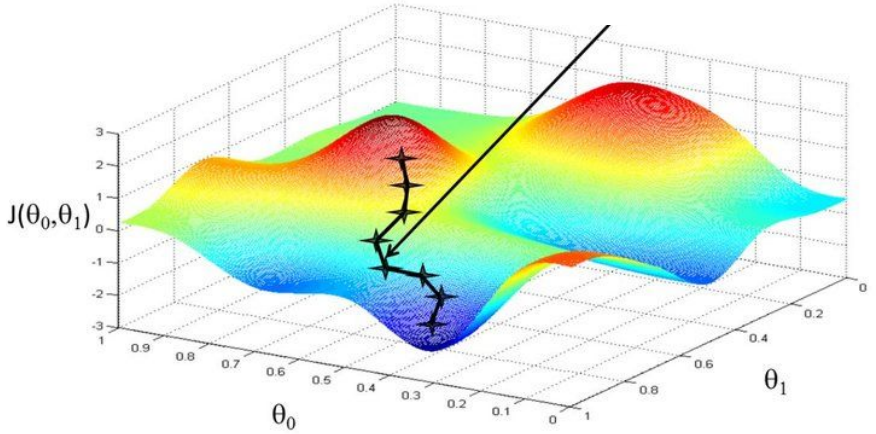
Vamos a comentar 3 métodos de optimización:

- Método 1: SGD
- Método 2: Momentum
- Método 3: ADAM

Optimizadores: SGD

A este proceso de actualizar los pesos a partir de calcular el gradiente en un batch (y no en todo el dataset) se lo llama **Stochastic Gradient Descent (SGD)**.

$$\theta = \theta - \alpha \nabla_\theta J(\theta)$$



Optimizadores: SGD

A este proceso de actualizar los pesos a partir de calcular el gradiente en un batch (y no en todo el dataset) se lo llama **Stochastic Gradient Descent (SGD)**.

$$\theta = \theta - \alpha \nabla_\theta J(\theta)$$

Keras

```
opti = tf.keras.optimizers.SGD( learning_rate=0.01)
model.compile(loss = 'mse', optimizer=opti)
```

Optimizadores: Momentum

La idea es incorporar **inercia** al término de actualización de los pesos, esto quiere decir que dependa del valor de actualización de la iteración anterior. Se busca acelerar el proceso de convergencia y ayudar a superar mínimos locales.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Optimizadores: Momentum

La idea es incorporar **inercia** al término de actualización de los pesos, esto quiere decir que dependa del valor de actualización de la iteración anterior. Se busca acelerar el proceso de convergencia y ayudar a superar mínimos locales.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Keras

```
opti = tf.keras.optimizers.SGD( learning_rate=0.01, momentum=0.9)
model.compile(loss = 'mse', optimizer=opti)
```

Optimizadores: Adam

Además de la **inercia,** el método ajusta el Learning Rate para cada parámetro teniendo en cuenta el cuadrado del gradiente correspondiente a ese parámetro.

Optimizadores: Adam

Además de la **inercia,** el método ajusta el Learning Rate para cada parámetro teniendo en cuenta el cuadrado del gradiente correspondiente a ese parámetro.

Inercia - $$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

2do momento - $$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Actualización - $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Optimizadores: Adam

Además de la **inercia,** el método ajusta el Learning Rate para cada parámetro teniendo en cuenta el cuadrado del gradiente correspondiente a ese parámetro.

Inercia - $$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

2do momento - $$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Actualización - $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Corregir inicio:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Optimizadores: Adam

Además de la **inercia,** el método ajusta el Learning Rate para cada parámetro teniendo en cuenta el cuadrado del gradiente correspondiente a ese parámetro.

Actualización - $$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

Keras

```
opti = tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999)
model.compile(loss = 'mse', optimizer=opti)
```

Optimizadores: Adam

Es hoy en día el optimizador más usado.

Adam: A method for stochastic optimization

DP Kingma, J Ba - arXiv preprint arXiv:1412.6980, 2014 - arxiv.org

We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters …

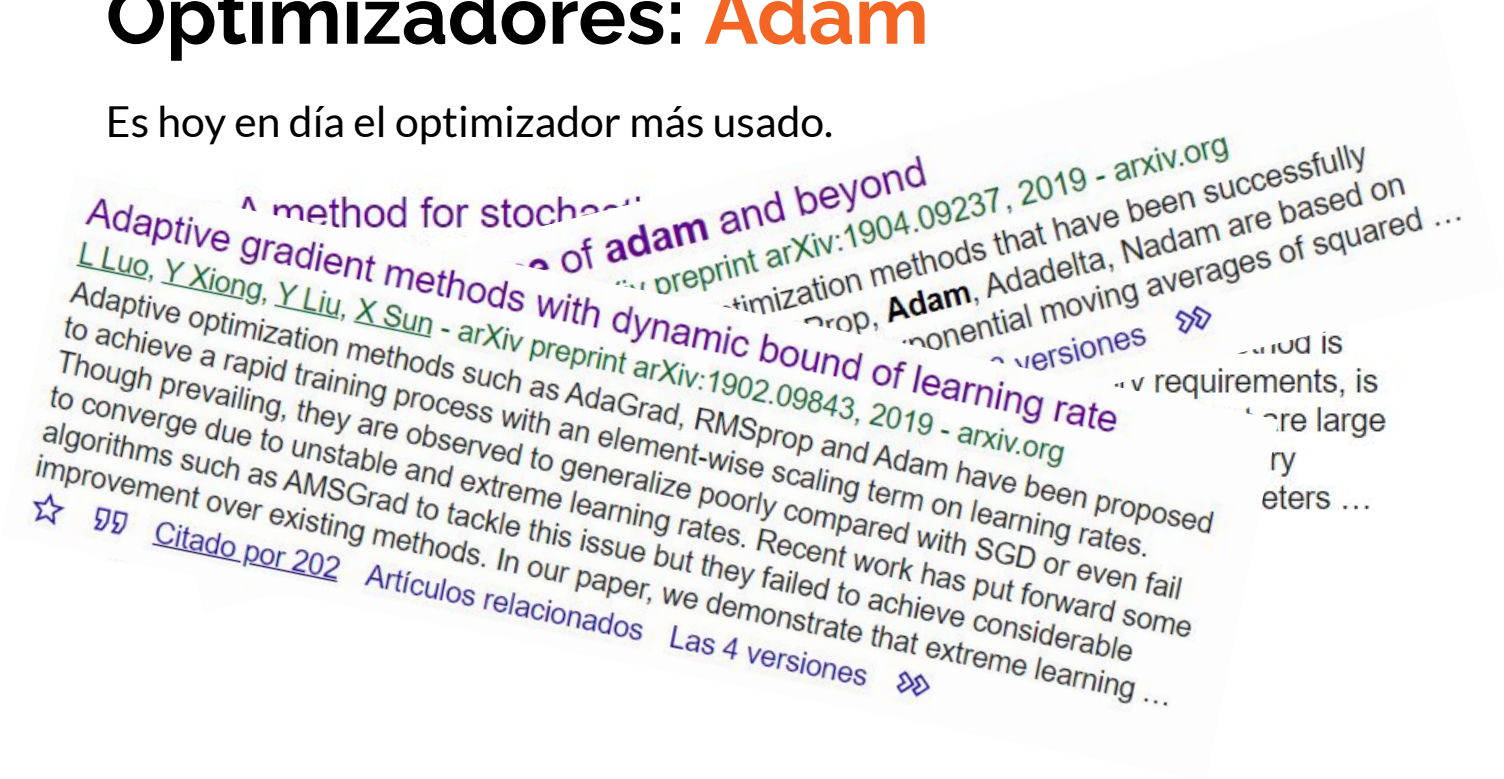
☆ 99 Citado por 58291 Artículos relacionados Las 18 versiones »

Optimizadores: Adam

Es hoy en día el optimizador más usado.

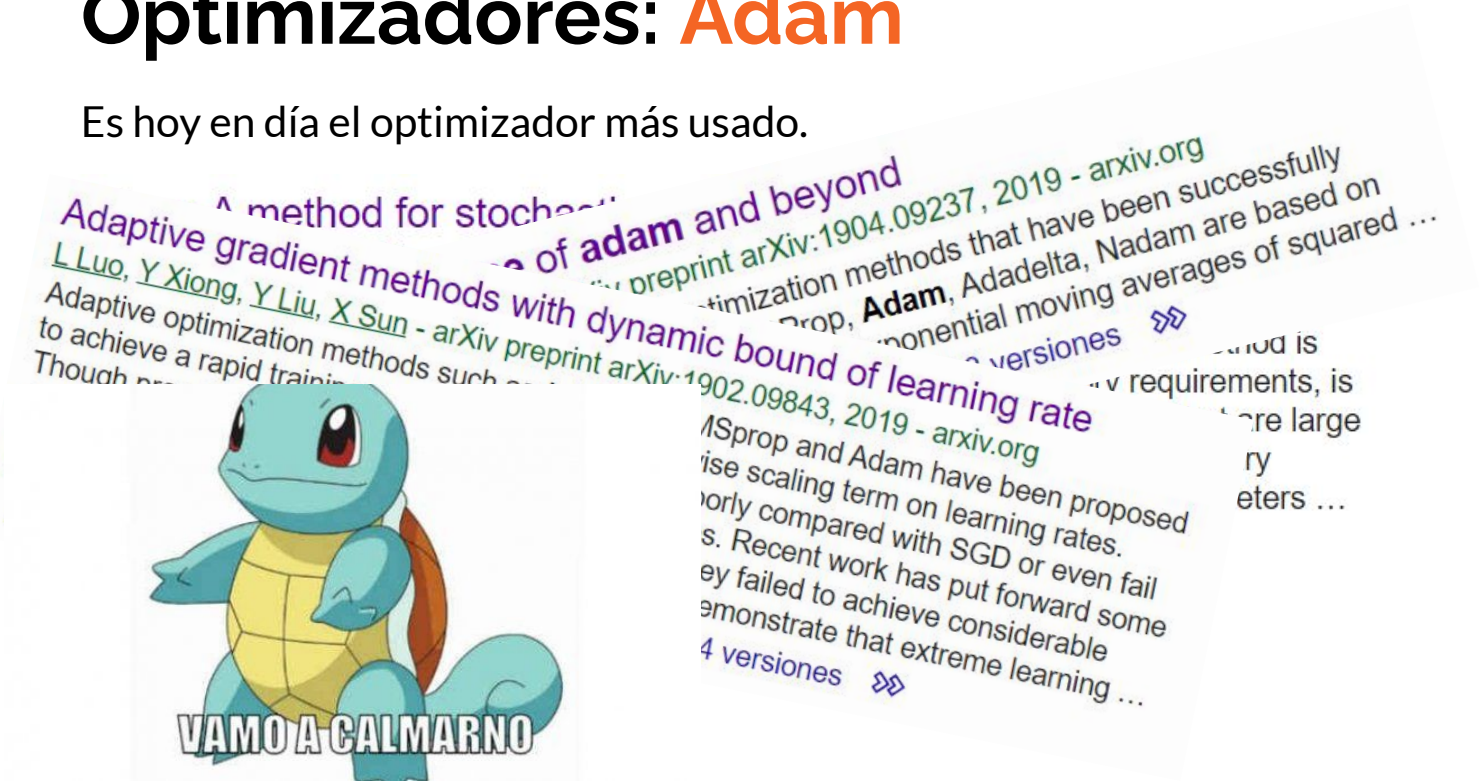
Optimizadores: Adam

Es hoy en día el optimizador más usado.



Optimizadores: Adam

Es hoy en día el optimizador más usado.



Trabajo en el Notebook

Link

https://drive.google.com/file/d/1NTeugGCdkJh0b_jD8Sn5MZo212fBItW/view?usp=sharing

Breakout Rooms

Trabajo en forma grupal, uno comparte pantalla. Pueden llamarme a la sala en cualquier momento.

Puesta en común y Conclusiones

(15 minutos) Comentarios sobre el trabajo realizado.