



Day 87

初探深度學習使用 Keras

訓練神經網路的細節與技巧

使用 callbacks 函數做
reduce learning rate

游為翔

出題教練

知識地圖 深度學習訓練技巧

使用 callbacks 函數做 reduce learning rate

深度神經網路

Supervised Learning Deep Neural Network (DNN)

簡介 Introduction

套件介紹 Tools: Keras

組成概念 Concept

訓練技巧 Training Skill

應用案例 Application

卷積神經網路

Convolutional Neural Network (CNN)

簡介 introduction

套件練習 Practice with Keras

訓練技巧 Training Skill

電腦視覺 Computer Vision

深度學習訓練技巧

Training Skill of DNN

應注意的關鍵

防止過擬合 (Overfitting)

超參數 (Hyper-parameters)

學習率 (Learning Rate) 調整

相關訓練技巧

正規化
Regularization

批次標準化
Batch Normalization

回呼
Callback

隨機移除
Drop out

客製化損失函數
Customized Loss Function

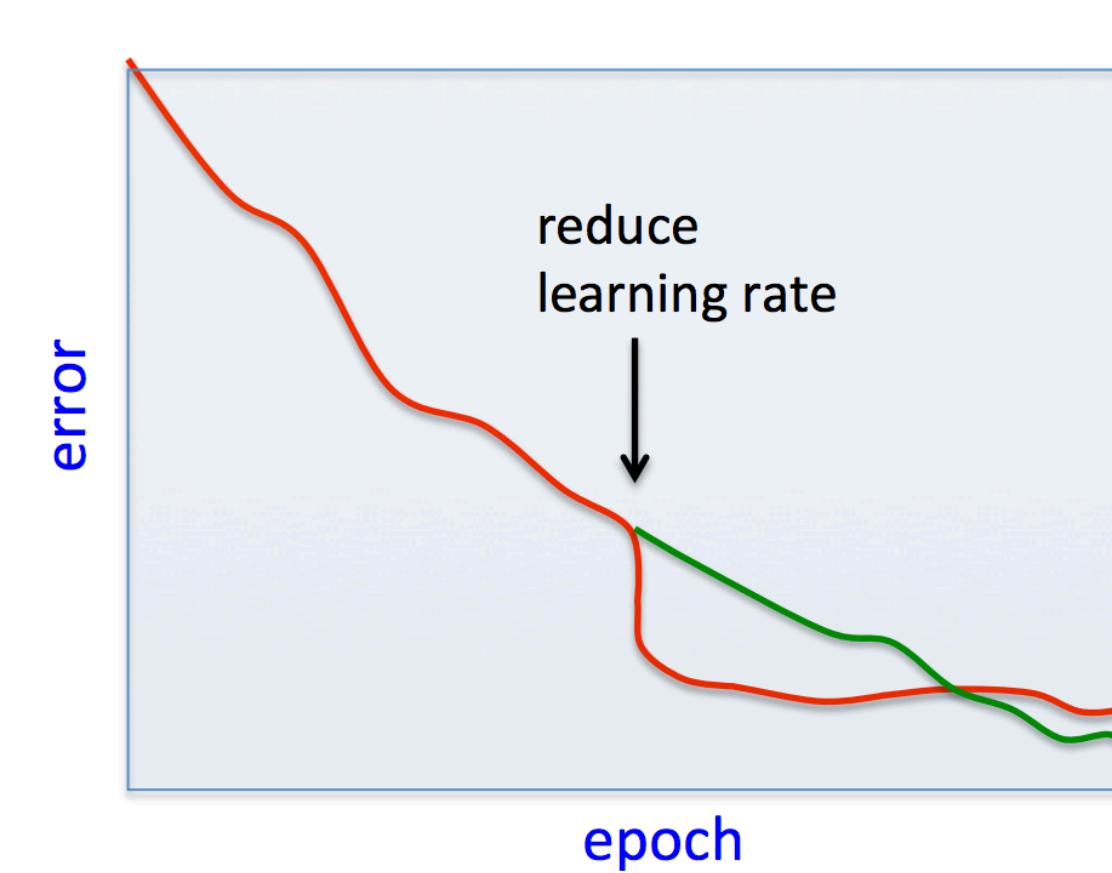
提前終止
Early Stopping

本日知識點目標

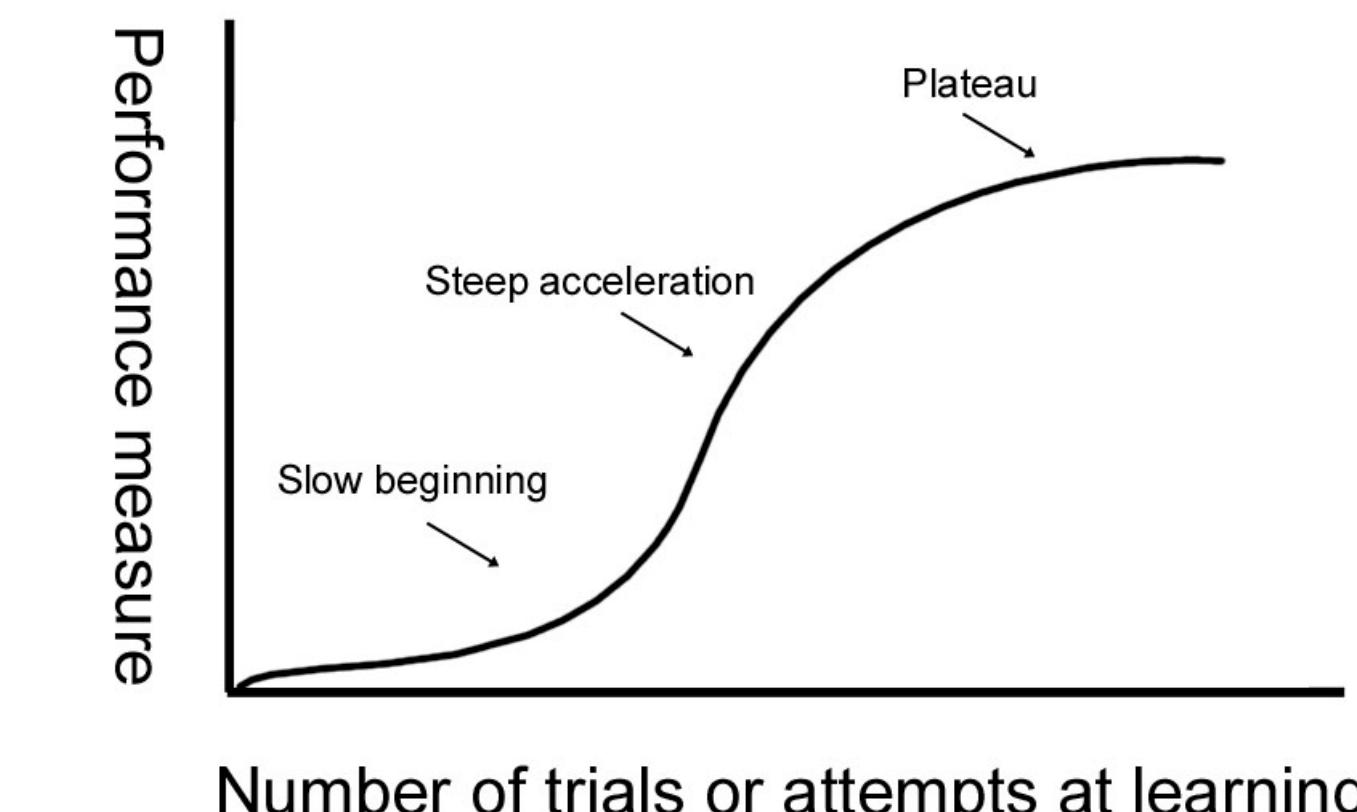
- 了解什麼是 Reduce Learning Rate
- 知道如何在 Keras 中，加入 ReduceLearningRate

Reduce Learning Rate

- Reduce Learning Rate: 隨訓練更新次數，將 Learning rate 逐步減小
 - 因為通常損失函數越接近谷底的位置，開口越小 – 需要較小的 Learning rate 才可以再次下降
- 可行的調降方式
 - 每更新 n 次後，將 Learning rate 做一次調降 – schedule decay
 - 當經過幾個 epoch 後，發現 performance 沒有進步 – Reduce on plateau



圖片來源：stats.stackexchange.com



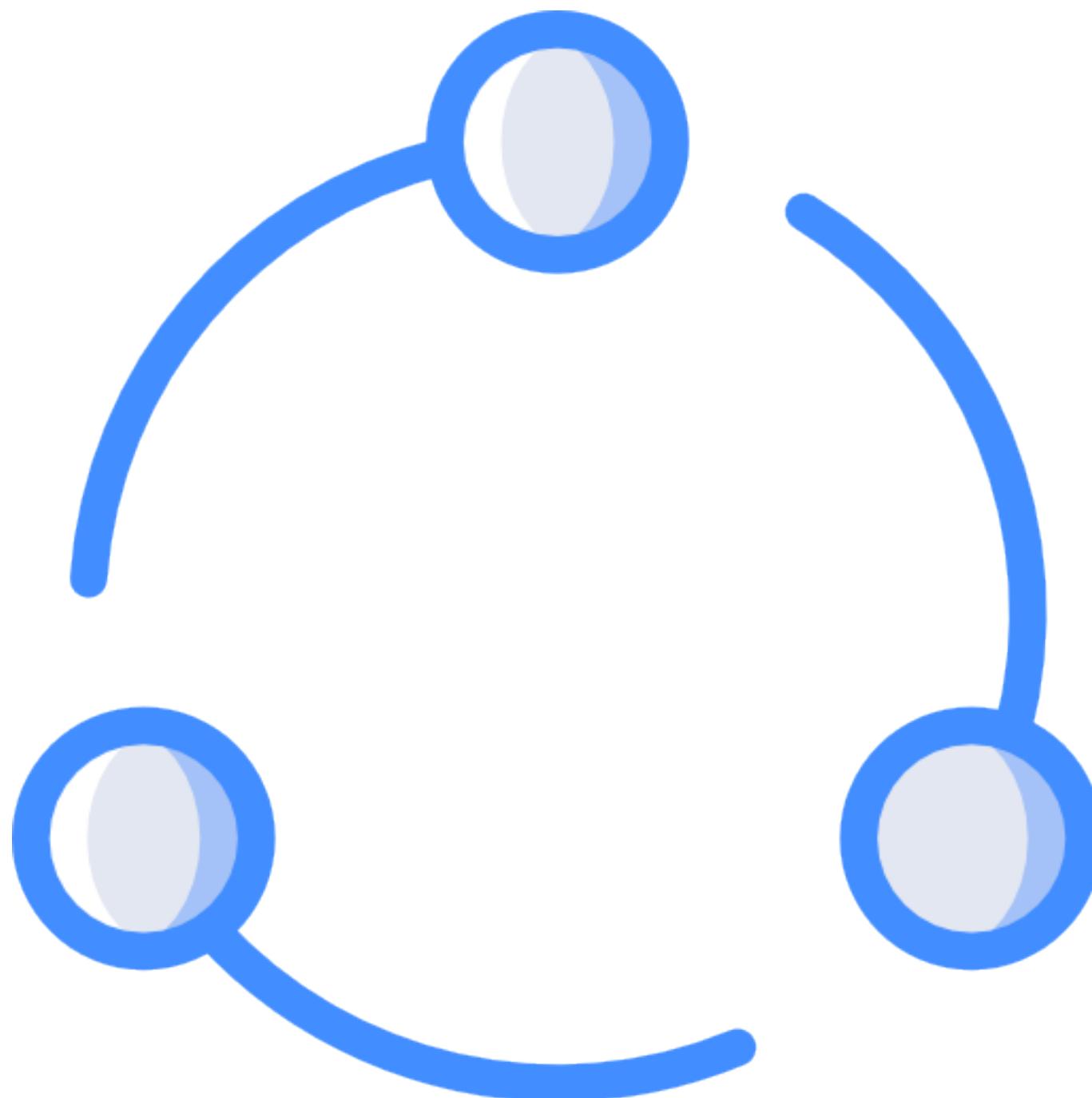
圖片來源：the-knowledgesmart-blog

Reduce Learning Rate on Plateau in Keras

```
from keras.callbacks import ReduceLROnPlateau
reduce_lr = ReduceLROnPlateau(factor=0.5,
                               min_lr=1e-12,
                               monitor='val_loss',
                               patience=5,
                               verbose=1)

model.fit(x_train, y_train,
          epochs=EPOCHS,
          batch_size=BATCH_SIZE,
          validation_data=(x_test, y_test),
          shuffle=True,
          callbacks=[reduce_lr]
         )
```

重要知識點複習：



- Reduce learning rate on plateau：模型沒辦法進步的可能
是因為學習率太大導致每次改變量太大而無法落入較低的
損失平面，透過適度的降低，就有機會得到更好的結果
- 因為我們可以透過這樣的監控機制，初始的 Learning
rate 可以調得比較高，讓訓練過程與 callback 來做適當
的 learning rate 調降。



延伸 閱讀

除了每日知識點的基礎之外，推薦的延伸閱讀能補足學員們對該知識點的了解程度，建議您解完每日題目後，若有
多餘時間，可再補充延伸閱讀文章內容。

推薦延伸閱讀

Github 原碼：LearningRateScheduler 與 ReduceLR

A. LearningRateScheduler

1. 在每個 epoch 開始前，得到目前 lr
2. 根據 schedule function 重新計算 lr，比如 epoch = n 時， new_lr = lr * 0.1
3. 將 optimizer 的 lr 設定為 new_lr
4. 根據 shhedule 函式，假設要自訂的話，它應該吃兩個參數：epoch & lr

B. ReduceLR

1. 在每個 epoch 結束時，得到目前監控目標的數值
2. 如果目標比目前儲存的還要差的話，wait+1；若否則 wait 設為 0，目前監控數值更新新的數值
3. 如果 wait >= patient，new_lr = lr * factor，將 optimizer 的 lr 設定為 new_lr，並且 wait 設回 0

- [參考連結 \(1\)](#)
- [參考連結 \(2\)](#)

```
lr = float(K.get_value(self.model.optimizer.lr)) A1
```

```
lr = self.schedule(epoch, lr) A2
```

```
K.set_value(self.model.optimizer.lr, lr) A3
```

```
logs['lr'] = K.get_value(self.model.optimizer.lr) B1
```

```
current = logs.get(self.monitor)
```

```
elif not self.in cooldown():
    self.wait += 1 B2
```

```
if self.wait >= self.patience:
```

```
old_lr = float(K.get_value(self.model.optimizer.lr))
```

```
if old_lr > self.min_lr:
```

```
new_lr = old_lr * self.factor
```

```
new_lr = max(new_lr, self.min_lr)
```

```
K.set_value(self.model.optimizer.lr, new_lr) B3
```



解題時間

It's Your Turn

請跳出PDF至官網Sample Code & 作業
開始解題

