



Day 71 損失函數

損失函數的介紹與應用



陳宇春

出題教練

知識地圖 深度學習簡介

深度學習體驗 - 啟動函數與正規化

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

深度學習組成概念
Concept of DNN

感知器概念簡介



本日知識點目標

- 了解損失函數
- 針對不同的問題使用合適的損失函數

損失函數

- 機器學習中所有的算法都需要最大化或最小化一個函數，這個函數被稱為「目標函數」。其中，我們一般把最小化的一類函數，稱為「損失函數」。
它能根據預測結果，衡量出模型預測能力的好壞
- 損失函數大致可分為：分類問題的損失函數和回歸問題的損失函數
 - Numerical Issues

損失函數為什麼是最小化

- 期望：希望模型預測出來的東西可以跟實際的值一樣
- 預測出來的東西基本上跟實際值都會有落差
 - 在回歸問題稱為「殘差(residual)」
 - 在分類問題稱為「錯誤率(error rate)」
- 損失函數中的損失就是「實際值和預測值的落差」
- y 表示實際值， \hat{y} 表示預測值

$$\text{loss/residual} = y - \hat{y}$$

$$\text{error rate} = -\frac{\sum_{i=1}^n \text{sign}(y_i \neq \hat{y}_i)}{n}, \text{sign}(y_i \neq \hat{y}_i) = \begin{cases} 1, & y_i \neq \hat{y}_i \\ 0, & y_i = \hat{y}_i \end{cases}$$

損失函數的分類介紹 - mean_squared_error

- 均方誤差(mean_squared_error)：就是最小平方法(Least Square) 的目標函數 -- 預測值與實際值的差距之平均值。還有其他變形的函數, 如 mean_absolute_error、mean_absolute_percentage_error、mean_squared_logarithmic_error。

$$\sum (\hat{y} - y)^2 / N$$

- 使用時機：
 - n 個樣本的預測值 (y) 與 (y_{true}) 的差距
 - Numerical 相關
- Keras 上的調用方式：
 - from keras import losses
 - model.compile(loss= 'mean_squared_error', optimizer='sgd')
 - 其中，包含 y_{true} ， y_{pred} 的傳遞，函數是表達如下：
 - keras.losses.mean_squared_error(y_{true} , y_{pred})

損失函數的分類介紹 - Cross Entropy

- 當預測值與實際值愈相近，損失函數就愈小，反之差距很大，就會更影響損失函數的值
- 要用 Cross Entropy 取代 MSE，因為，在梯度下時，Cross Entropy 計算速度較快，
- 使用時機：
 - 整數目標：Sparse categorical_crossentropy
 - 分類目標：categorical_crossentropy
 - 二分類目標：binary_crossentropy。
- Keras 上的調用方式：
 - from keras import losses
 - model.compile(loss= 'categorical_crossentropy ', optimizer='sgd')
 - 其中, 包含y_true , y_pred的傳遞, 函數是表達如下：
 - keras.losses.categorical_crossentropy(y_true, y_pred)

損失函數的分類介紹: Hinge Error (hinge)

- 是一種單邊誤差，不考慮負值同樣也有多種變形，[squared_hinge](#)、[categorical_hinge](#)

$$\ell(y) = \max(0, 1 - t \cdot y)$$

- 使用時機：
 - 適用於『支援向量機』(SVM)的最大間隔分類法(maximum-margin classification)
- Keras 上的調用方式：
 - `from keras import losses`
 - `model.compile(loss= 'hinge', optimizer='sgd')`
 - 其中，包含 `y_true`，`y_pred` 的傳遞，函數是表達如下：
 - `keras.losses.hinge(y_true, y_pred)`

特別的案例：自定義損失函數

- 根據問題的實際情況，定制合理的損失函數
- 舉例：預測果汁日銷量問題，如果預測銷量大於實際銷量則會損失成本；如果預測銷量小於實際銷量則會損失利潤。
 - 考慮重點：製造一盒果汁的成本和銷售一盒果汁的利潤不是等價的
 - 需要使用符合該問題的自定義損失函數自定義損失函數為：

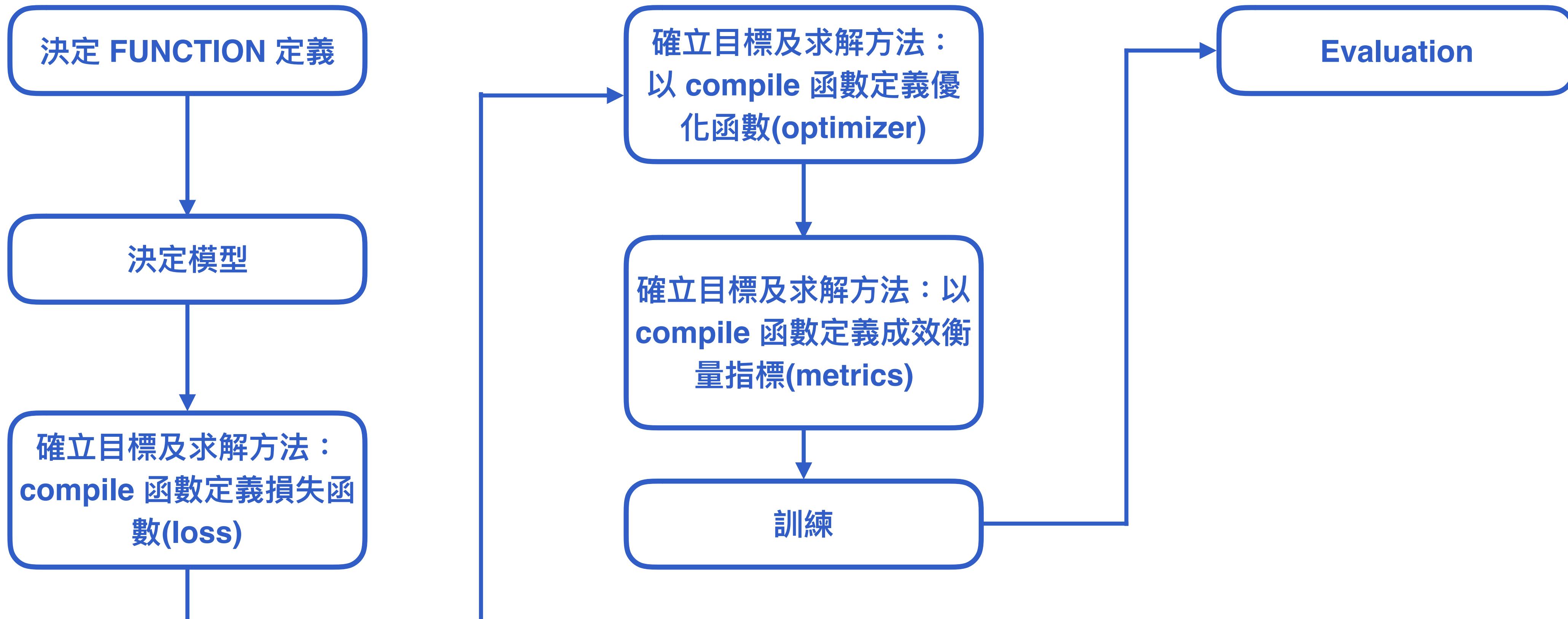
$$loss = \sum n f(y_{\sim}, y)$$

特別的案例: 自定義損失函數 (II)

- 接續上一頁
 - 損失函數表示若預測結果 y 小於標準答案 y_+ ，損失函數為利潤乘以預測結果 y 與標準答案之差
 - 若預測結果 y 大於標準答案 y_+ ，損失函數為成本乘以預測結果 y 與標準答案之差用
 - Tensorflow 函數表示為：

```
loss = tf.reduce_sum(tf.where(tf.greater(y, y_), COST*(y-y_), PROFIT*(y_-y)))
```

前述流程 / python程式 對照





前述流程 / python 程式 對照

載入之前訓練的模型

```
In [21]: try:  
        model.load_weights("SaveModel/cifarCnnModel.h5")  
        print("載入模型成功!繼續訓練模型")  
    except :  
        print("載入模型失敗!開始訓練一個新模型")
```

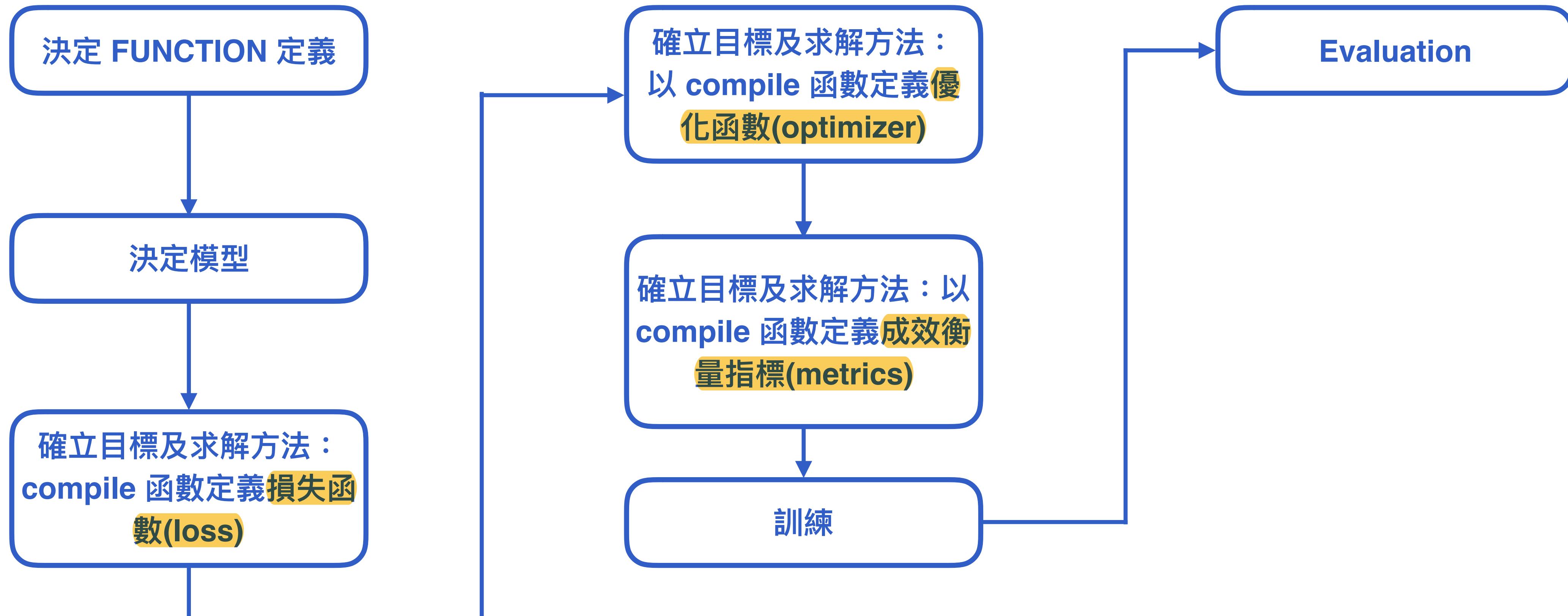
載入模型失敗！開始訓練一個新模型

訓練模型

```
In [42]: model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

...
作業解答：
請分別選用 "MSE", "binary _crossentropy"
查看Train/test accuracy and loss rate
...
```

複習：流程



複習：流程

- 損失函數中的損失就是「實際值和預測值的落差」，損失函數是最小化
- 損失函數大致可分為：分類問題的損失函數和回歸問題的損失函數



延伸 閱讀

- TensorFlow筆記-06-神經網絡優化- 損失函數，自定義損失函數，交叉熵（簡體）
- 使用損失函數（英文）

推薦延伸閱讀

自訂損失函數 1/2

- 對於預測優酪乳日銷量問題，如果預測銷量大於實際銷量則會損失成本；如果預測銷量小於實際銷量則會損失利潤。
- 在實際生活中，往往製造一盒優酪乳的成本和銷售一盒優酪乳的利潤不是等價的。因此，需要使用符合該問題的自訂損失函數
- 自訂損失函數為：
 - $\text{loss} = \sum n f(y_-, y)$
 - 其中，損失函數成分段函數：損失函數表示若預測結果 y 小於標準答案 y_- ，損失函數為利潤乘以預測結果 y 與標準答案之差若預測結果 y 大於標準答案 y_- ，
- 損失函數為成本乘以預測結果 y 與標準答案之差用Tensorflow函數表示為：
 - `loss = tf.reduce_sum(tf.where(tf.greater(y, y_), COST*(y-y_), PROFIT*(y_-y)))`
- 第1種情況：若優酪乳成本為1元，優酪乳銷售利潤為9元，則製造成本小於優酪乳利潤，因此希望預測結果 y 多一些
 - # 定義損失函數使得預測少了的損失大，於是模型應該偏向多的方向預測
 - `loss = tf.reduce_sum(tf.where(tf.greater(y, y_), COST*(y-y_), PROFIT*(y_-y)))`
 - `train_step = tf.train.GradientDescentOptimizer(0.001).minimize(loss)`
- 第2種情況：若優酪乳成本為9元，優酪乳銷售利潤為1元，則製造利潤小於優酪乳成本，因此希望預# 重新定義損失函數使得預測多了的損失大，於是模型應該偏向少的方向預測
 - `loss = tf.reduce_sum(tf.where(tf.greater(y, y_), COST*(y-y_), PROFIT*(y_-y)))`
 - `train_step = tf.train.GradientDescentOptimizer(0.001).minimize(loss)`

其他優化方法

```
# train_step = tf.train.GMomentumOptimizer(0.001, 0.9).minimize(loss)
# train_step = tf.train.AdamOptimizer(0.001).minimize(loss)
```



解題時間

It's Your Turn

請跳出PDF至官網Sample Code & 作業
開始解題

