AI TEAM分享(一)
2018/04/24

大綱

- 業界AI發展分享 - 金融顛覆創新，德意志銀行AI之路(2018/04/14)
- 深度學習階段
- MOOC深度學習課程比較
- Python安裝及開發環境準備

業界AI發展分享

- 金融顛覆創新真的不容易，德意志銀行AI之路大公開

  - 德意志銀行創新技術產品部門管理總監Roberto Mancone

  - 德意志銀行目標是2020實現數位轉型

  - 想靠AI成為提升和強化金融專業的工具

  - 擁抱AI先從三大應用場景切入

  - 德意志銀行擁抱AI的四個教訓

深度學習階段

- 預備知識
  - **Programming skills : Python 3 (, 雲端服務)**
  - **Some math education in the past : algebra, geometry, calculus etc.**
- 建議學習階段

1. **Practical Deep Learning for Coders - Part 1 in fast.ai，共 7 個 lessons，約4~6週**
2. **Learn about calculus, linear algebra and matrix calculus**
3. **Deep Learning Specialization in Coursera，共5個courses，約 5 週**
4. **實際建置項目**

MOOC深度學習課程比較

	Fast.ai Practical Deep Learning for Coder Part1	Coursera Deep Learning Specialization	Udacity Deep Learning
講師	Jeremy Howard	Andrew Ng	Siraj Raval
課程概述	1. 學習如何在雲端部署工作站及安裝軟體，並使用深度學習快速建構解決方案。 2. 每周會挑選一個新的問題，然後教授你如何使用技術來提高模型的性能，例如使用預卷積特徵，偽標記和其他非常有用的技巧。 3. 第一部分结束將能够使用深度學習在工作領域構建實用的應用程式。	1. 詳細介绍了深度學習中許多概念所需的數學知識，這是了解深度學習的基礎所必需的，對通過程式的方式實踐數學知識非常有幫助。所有的公式都已經提供好了，所以就算没有太多的數學知識，也可以專注於實現他們。 2. 涵蓋了所有技術，像是正則化，dropout等來提升模型的性能，最好的地方就是使用Python和numpy實作所有技術· 3. 使用TensorFlow實作CNN，RNN等，並提供大量他的經驗·	1. 與其他兩門課程不同，此課程不會採取非常明確的自下而上或自上而下方式。 2. 課程分為五個部分，並附贈100美元的Amazon Credit。 3. 涵盖了很多深度學習技術，如CNN，RNN，GAN，自編碼器等。

MOOC深度學習課程比較

	Fast.ai Practical Deep Learning for Coder Part1	Coursera Deep Learning Specialization	Udacity Deep Learning
優勢	1. 建立世界級的圖像辨識以及NLP模型 2. 了解與使用主流框架學習深度學習，如Keras，TensorFlow，PyTorch 3. 不需對數學有深入的了解即可快速應用深度學習技巧 4. 龐大的社群與論壇	1. 上完課後基礎知識將非常強大 2. 了解與使用主流框架學習深度學習，如TensorFlow 3. 所有作業都是jupyter notebook形式並運行在Coursera伺服器上 4. 有證書	1. 在本课程的前幾個星期，學習如何使用python構建神經網絡，課程其餘部分將集中使用TensorFlow 2. 提交的所有項目都提供批改。 3. 龐大的社群與論壇
限制	1. 依賴框架，若要更深入調整模型需要提高數學知識 2. 無證書	1. 了解背後的數學遠裡可能非常具有挑戰性 2. 由於採用由下往上的方式，上到課程中段時候依然很難在你的領域中利用DL建構解決方案	1. 課程提供了一些技巧來告訴你如何最佳建置深度學習模型，但這並沒有作為課程的重點 2. 一些項目太過於簡單以至於很難運用在現實需求
費用	免費	Full Catalog Access $49/month Subscription courses $49	$599

PYTHON安裝及開發環境準備

- Python安裝，https://www.python.org/downloads/
  - Windows OS
    - 使用Anaconda
    - 不使用Anaconda —> 透過folder管理不同環境
  - Mac OS
    - 系統預設就有Python
    - 使用Anaconda

PYTHON安裝及開發環境準備

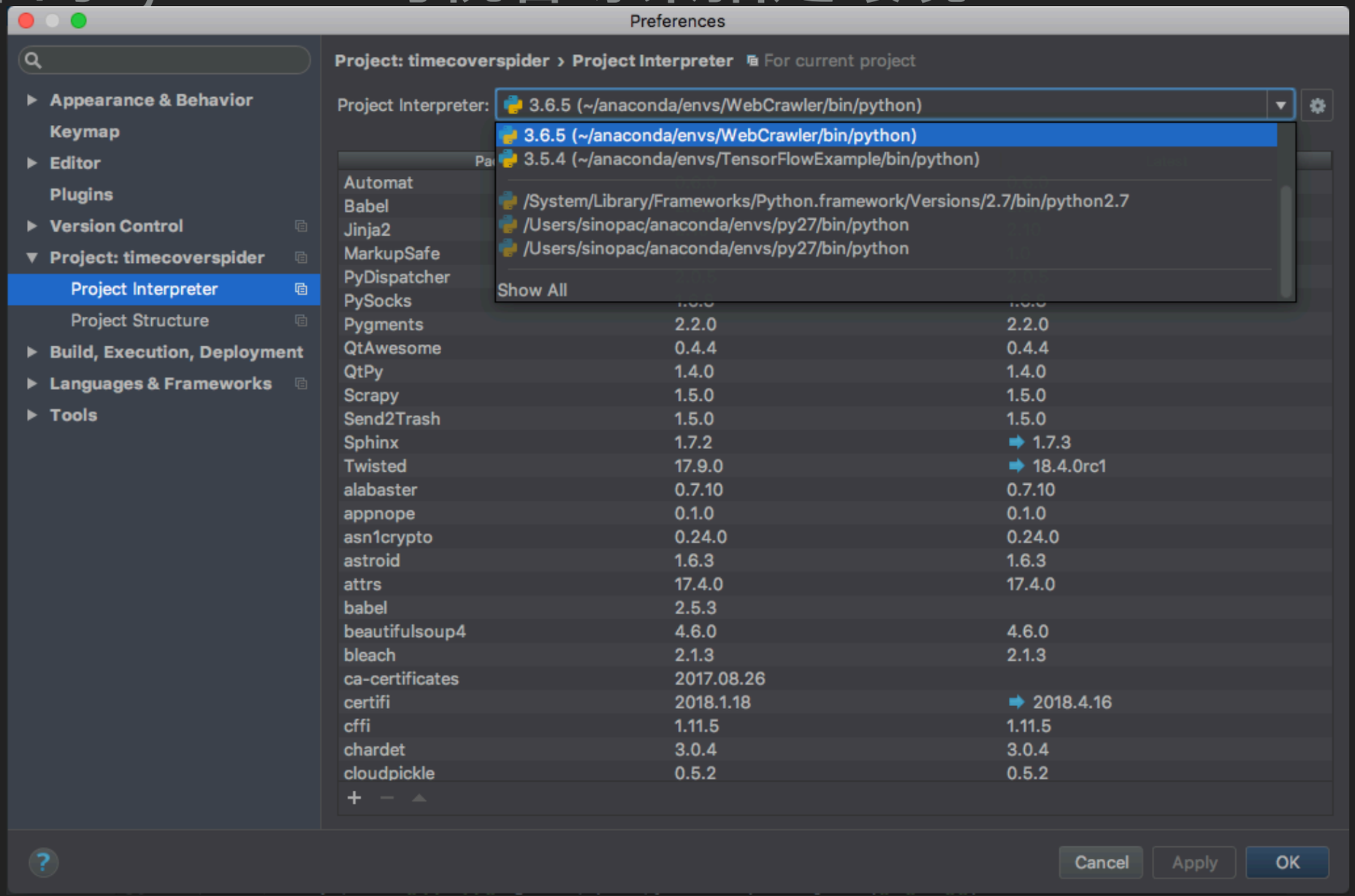
- 安裝TensorFlow & Keras
  - 新增Anaconda環境 conda create -n tensorflow pip python=3.5
  - 啟動環境 activate tensorflow
  - pip install --ignore-installed --upgrade tensorflow
  - pip install --ignore-installed --upgrade tensorflow-gpu ，使用GPU版本還需要安裝CUDA以及CuDNN https://www.tensorflow.org/install/install_windows#requirements_to_run_tensorflow_with_gpu_support

PYTHON安裝及開發環境準備

- 使用Jupyter Notebook (Jupyter Lab已發佈)
    - 本機
        1. 透過Anaconda安裝
    - 線上環境
        1. JupyterLab
        2. Google Colab

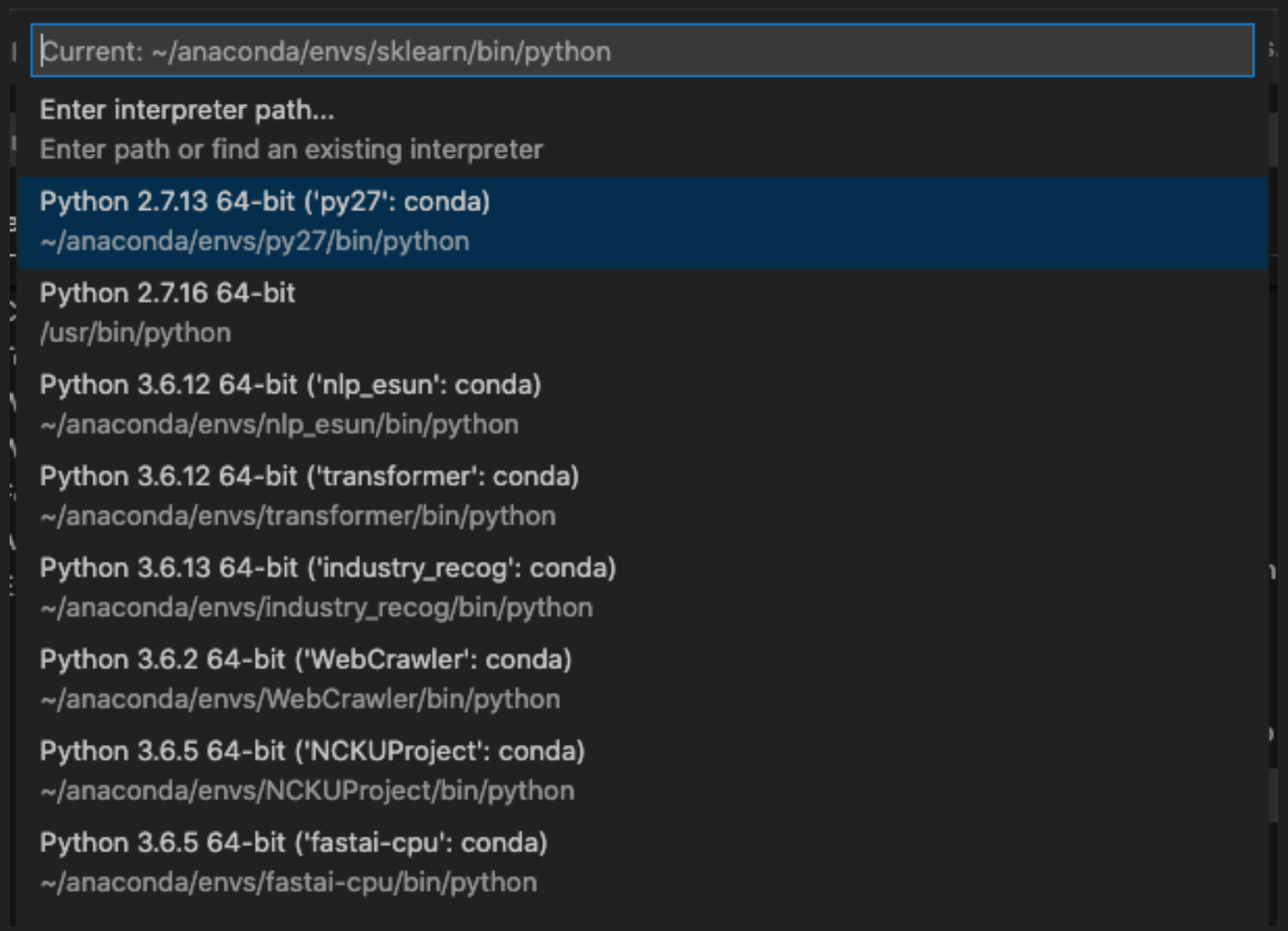
PYTHON安裝及開發環境準備

- 使用PyCharm
  - Preferences –> Project Interpreter，指定使用哪個版本的Python，可視各專案指定環境



PYTHON安裝及開發環境準備

- 使用VS Code
  - Preferences > Python: Select Interpreter 指定使用哪個環境



特徵選擇

- 尋找最好的特徵集合
  - 剔除不相關(irrelevant)或冗餘(redundant)
- 現實中的情況往往是特徵太多
- 好處
  - 降低過擬合
  - 更好的解釋性
  - 加快

特徵選擇 - 包裝法

- Full Search
  - 找尋所有可能的特徵組合
  - 假設有10個特徵，每個特徵都有"選"或"不選"，那就有1023種組合(扣掉全不選)
  - 運算複雜度 -> 2^N

特徵選擇 - 包裝法

- Greedy Forward Selection
  - 每次都只挑看起來最好的特徵
  - 假設有10個特徵，對每一個特徵都訓練一個模型並使用交叉驗證計算分數，最後挑出分數最高的，接著再從剩下 9 個中，對每一個特徵+已經挑出來的特徵，也都訓練一個模型，再挑出分數最高的，如此循環
  - 運算複雜度 -> N^2

特徵選擇 - 包裝法

- Greedy Forward Selection

輸入：所有特徵 feature
輸出：由高分到低分依序排列的特徵 result

初始化一個空集合 result
複製所有 feature 到 candidate
重複 N 次以下動作，N 為 feature 裡的特徵總數：
    初始化分數表 score
    對 candidate 裡的每一個特徵進行以下動作：
        依序從 candidate 裡選出一個特徵
        使用集合 result + 選出來的特徵訓練模型，得出交叉驗證的分數
        將得到的分數存在分數表 score
    將分數最高的特徵加入集合 result
    將分數最高的特徵從 candidate 移除

特徵選擇 - 包裝法

- Stepwise Forward Selection
  - 跟Greedy有點像
  - 假設有10個特徵，對每一個特徵都訓練一個模型並使用交叉驗證計算分數，排序後取分數最高的特徵直接選入集合
  - 接著用集合內的特徵和第二好的特徵訓練一個模型，如果分數增加量高於設定好的閾值，那就把這個特徵加入，如果沒有丟棄，如此循環
  - 運算複雜度 -> 2N

特徵選擇 - 包裝法

- Stepwise Forward Selection

輸入：所有特徵 feature、閾值
輸出：由高分到低分依序排列的特徵 result

初始化一個空集合 result
複製所有 feature 到 candidate
重複 N 次以下動作，N 為 feature 裡的特徵總數：
    依序從 candidate 選出一個特徵
    使用選出來的特徵訓練模型，得出交叉驗證的分數
    將得到的分數存在分數表 score
根據分數，由高分到低分，重新排序 candidate 裡的特徵
將分數最高的特徵加入集合 result
將分數最高的特徵從 candidate 移除
將最高分存在變數 best
重複 M 次以下動作，M 為 candidate 裡的特徵總數：
    依序從 candidate 選出一個特徵
    使用集合 result + 選出來的特徵訓練模型，得出交叉驗證的分數
    如果分數增加量大於預先設定的閾值，則執行以下動作：
        將特徵加入集合 result
        更新最高分 best

特徵選擇 - 包裝法

- Simplified Greedy Forward Selection
  - 根據經驗將特徵排序，或隨機打亂，將第一個加入集合
  - 使用集合內的特徵以及排序第二的特徵訓練一個模型並使用交叉驗證計算分數，如果分數增加量高於設定好的閾值，那就把這個特徵加入，如果沒有丟棄，如此循環
- 運算複雜度 -> N

特徵選擇 - 包裝法

- Simplified Greedy Forward Selection

輸入：所有特徵 feature、閾值
輸出：由高分到低分依序排列的特徵 result

初始化最高分變數 best
初始化一個空集合 result
複製所有 feature 到 candidate
將 candidate 裡的特徵隨機排序
重複 N 次以下動作，N 為 candidate 裡的特徵總數：
    依序從 candidate 選出一個特徵
    使用集合 result + 選出來的特徵訓練模型，得出交叉驗證的分數
    如果分數增加量大於預先設定的閾值，則執行以下動作：
        將特徵加入集合 result
        更新最高分變數 best

特徵選擇 - 嵌入法

- 過濾法與學習器沒有關係，特徵選擇只是用統計量做篩選
- 包裝法則固定了學習器，特徵選擇只是在特徵空間上進行搜索
- 嵌入法最大的突破在於，特徵選擇會在學習器的訓練過程中自動完成

  - Ridge Regression和LASSO，加入懲罰項

$$Ridge\ Regreesion:\ min_w(\mathbf{y}-\mathbf{X}w)^T(\mathbf{y}-\mathbf{X}w)+\alpha||w||_2^2$$

$$LASSO:\ min_w(\mathbf{y}-\mathbf{X}w)^T(\mathbf{y}-\mathbf{X}w)+\beta||w||_1$$