深度學習教學小計畫

授課講師 陳少君

教材編寫 陳少君

緯育 TibaMe

即學・即戰・即就業

https://www.tibame.com/

學習目標：

- 了解教學方式與進度

Module 0.
教學目的方式與進度

目的方式與進度

深度學習養成班教學目的

為何而學

機器學習利用正確有效的資料訓練模型，並利用驗證資料及測試資料調整其參數，持續改進模型，使其分類、分群、迴歸等演算法能被有效應用

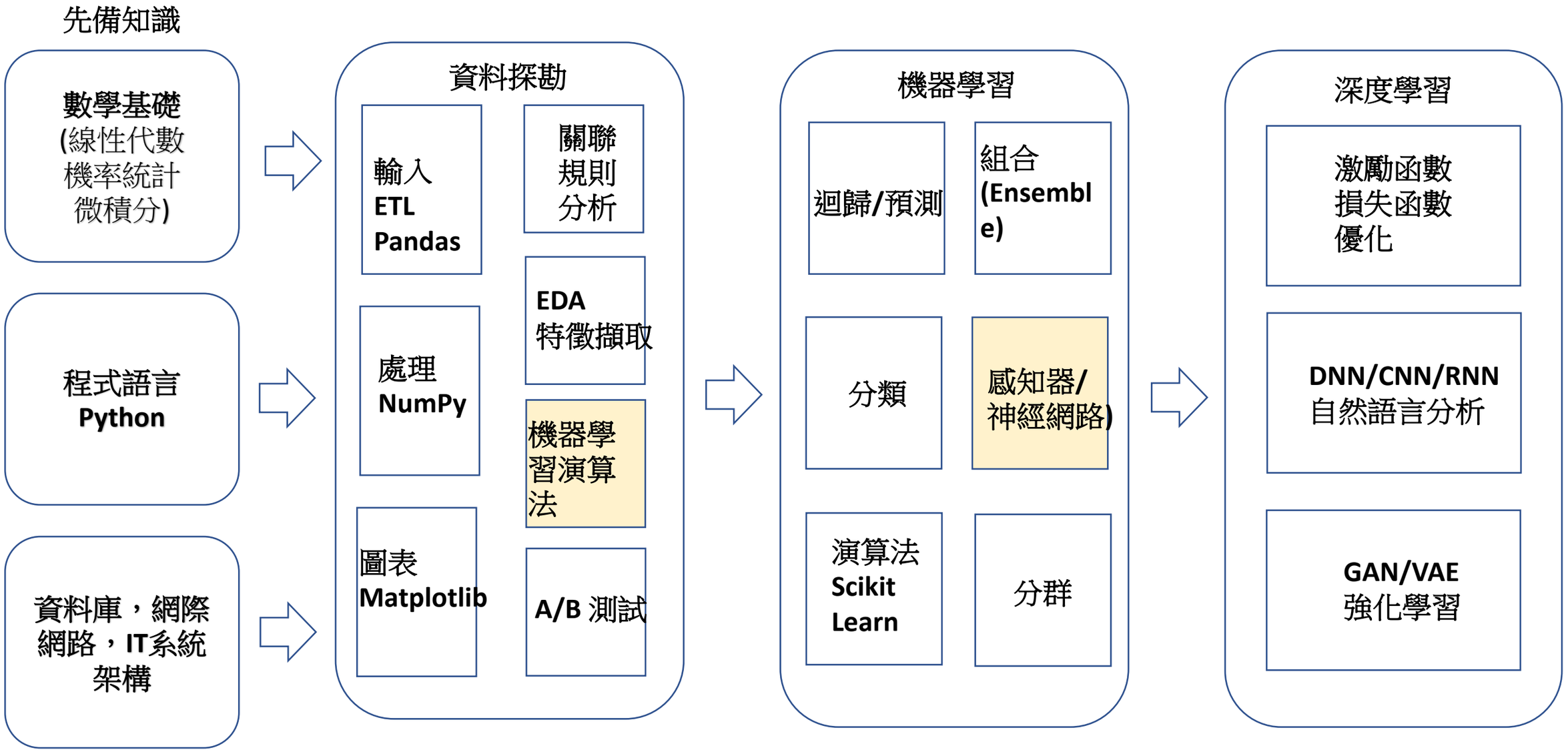
機器學習的一個分支：類神經網路，將成為未來深度學習的基礎。其他各演算法在各行各業也有其廣泛的應用領域。

承先啟後

先備知識：數學概念，Python/R 程式語言，資料探勘，機器學習

進階：深度學習之應用與演進

資料科學知識示意地圖



養成班教學進度

第一天(9月30日)：深度學習介紹
- 深度學習介紹
- Tensorflow 2.0介紹
- 損失函數
- 優化(1)

第二天(10月7日) ：神經網路，CNN
- 優化(2)
- Tensorboard
- CNN原理及應用

養成班教學進度

第三天(10月13日)：CNN應用，文字探勘

- CNN應用：物件偵測
- 文字探勘
- RNN介紹

第四天(10月21日) ：LSTM/GRU，降維，AutoEncoder，風格遷移

- LSTM/GRU
- 降維
- AutoEncoder
- 風格遷移

養成班教學進度

第五天(10月28日)：增強式學習，Keras，複習

- 增強式學習 (Reinforcement Learning)
- Keras
- 深度學習複習
- (Bonus: Transformer/BERT)

第一天

2 小時

深度學習概論
Tensorflow

2 小時

類神經網路
激活函數

2 小時

損失函數
優化演算法

預習：CNN
作業：作業一

從解聯立方程到機器(深度)學習

試解：

5x + 2y = 10

x + 4y = 11

機器學習不是在算參數權重(weight)嗎，把 x, y當成 w_1 w_2，把方程式左邊的係數(5,2) (1,4)定為輸入(x_1 x_2)，
右邊(10,11)為向量y，轉換成感知器的公式：

$5 * w_1 + 2 * w_2 = 10$

$1 * w_1 + 4 * w_2 = 11$

$$\begin{bmatrix}5 & 2\\ 1 & 4\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix}+\begin{bmatrix}-10\\ -11\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \qquad \begin{bmatrix}5 & 2\\ 1 & 4\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix}+\begin{bmatrix}0\\ 0\end{bmatrix}=\begin{bmatrix}10\\ 11\end{bmatrix}$$

如果我們故意不直接求解，而是用更多聯立方程式去逼近，比方$2*w_1 + 2*w_2 = 7$，我們是否也可解出 (w1,w2) ？
這樣作其實就是訓練感知器，如果在感知器後頭接上非線性激勵函數(sigmoid / tanh / relu)，
豈不形成類神經網路之一層？層層疊加就是深度學習網路之前饋(feed forward)部分
機器學習其實是個複雜的**函數映射**，線性代數適合**表述並計算出所需參數w**

看機器學習如何求解多元一次方程？ - 每日頭條 (kknews.cc)

Tensorflow vs PyTorch

兩大陣營各有千秋，簡單的說，Tensorflow 易懂好上手，Pytorch 適合做研究。Tensorflow 有 Tensorboard 的可視化功能，Pytorch 支持動態圖 (Dynamic Graph)。建議未來兩者都學。

以實例比較其不同：

PyTorch vs TensorFlow — spotting the difference | by Kirill Dubovikov | Towards Data Science

Pytorch 學習網站：

Learning PyTorch with Examples — PyTorch Tutorials 1.9.1+cu102 documentation

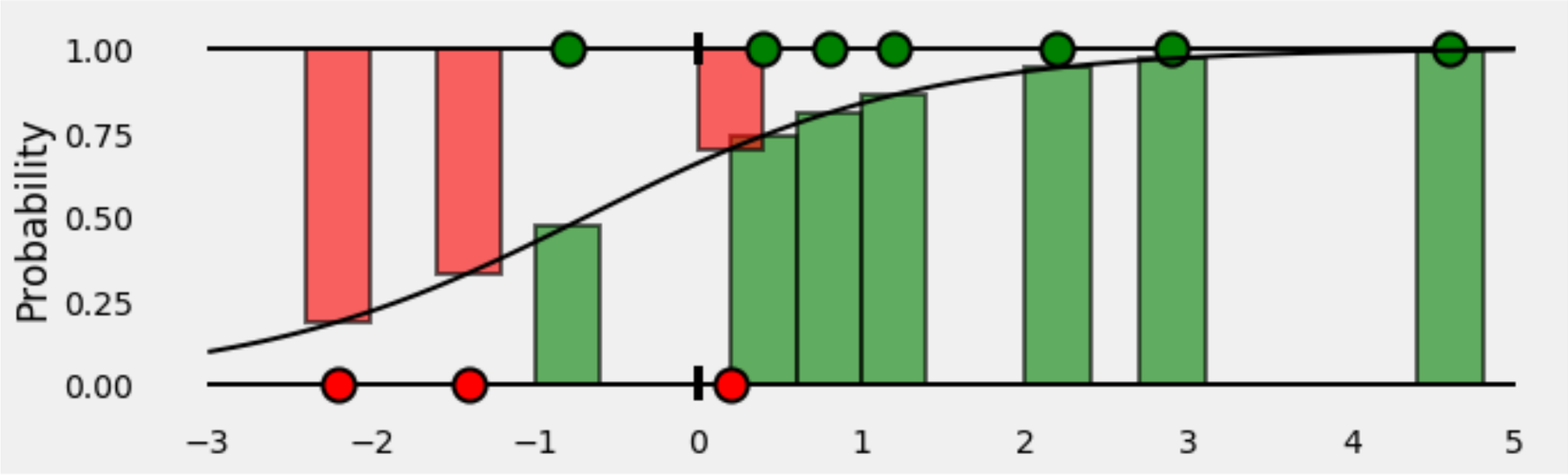
Tensorflow Important Classes

Class	範例/說明
Tensor	運算之主角
Operation	tf.add等運算元
Graph	Tensor的運算圖
Session	執行graph
Variable	像tensor之參數儲存處
Optimizer	以minimize method降低損失
Estimator	逐漸被淘汰之class較難開發卻適於軟體工程
Dataset	處理與載入資料
Iterator	將Dataset依序輸入模型
Saver	儲存參數供繼續訓練

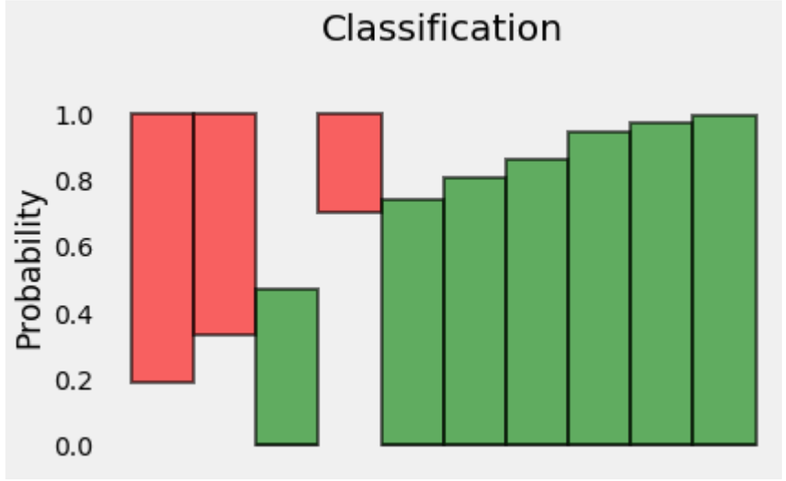
The 10 Most Important TensorFlow Classes - dummies

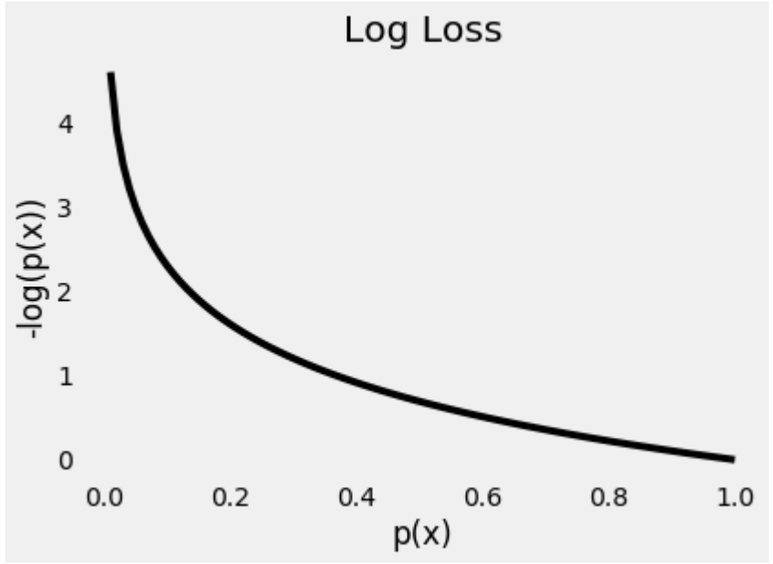
Log Loss (Cross Entropy)計算

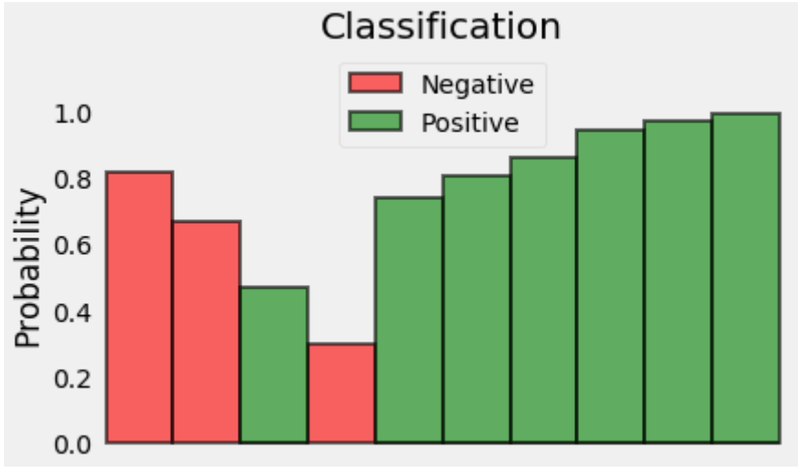
$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

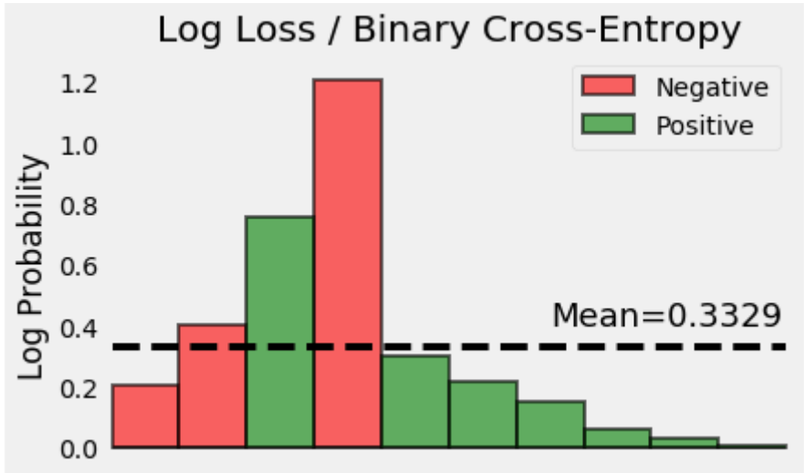


Log Loss (Cross Entropy) 計算









[Understanding binary cross-entropy / log loss: a visual explanation | by Daniel Godoy | Towards Data Science]

Loss Function的種類與選擇

大致分為

Probability Loss:

- Binary Cross Entropy：兩類分類
- Categorical Cross Entropy：多類分類，目標值(label)為獨熱(one hot)編碼
- Sparse Categorical Cross Entropy:(互斥)多類分類，目標值為整數編碼
- Poisson: 取張量元素平均值
- KL Divergence: 取機率對數加總之負數

Regression Loss

- MSE：Mean Squared Error
- MAE：Mean Absolute Error
- Cosine Similarity：向量之角度
- Huber：小的用平方大的用線性

Tensorflow Loss Functions | Loss Function in Tensorflow (analyticsvidhya.com)

稀疏(Sparse)類別不用獨熱(One Hot)

BinaryCrossentropy Loss Function ：

y_true = [0,1,0,0]

y_pred = [-18.6, 0.51, 2.94, -12.8] / /這是用 logit, API 可決定

SparseCategoricalCrossentropy Loss Function ：

y_true = [1,2]

y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]] //Probability

CategoricalCrossentropy Loss Function：

y_true = [[0,1,0], [0,0,1]]

y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]] //Probability

Logit 與 Probability

損失函數需要知道 y_pred 是否為 Logit，何謂 Logit？ (機率是 Probability(P))

Odds (賠率) = Probability / (1 –Probability)

Logit = Ln (Odds) = Ln (P/1- P)

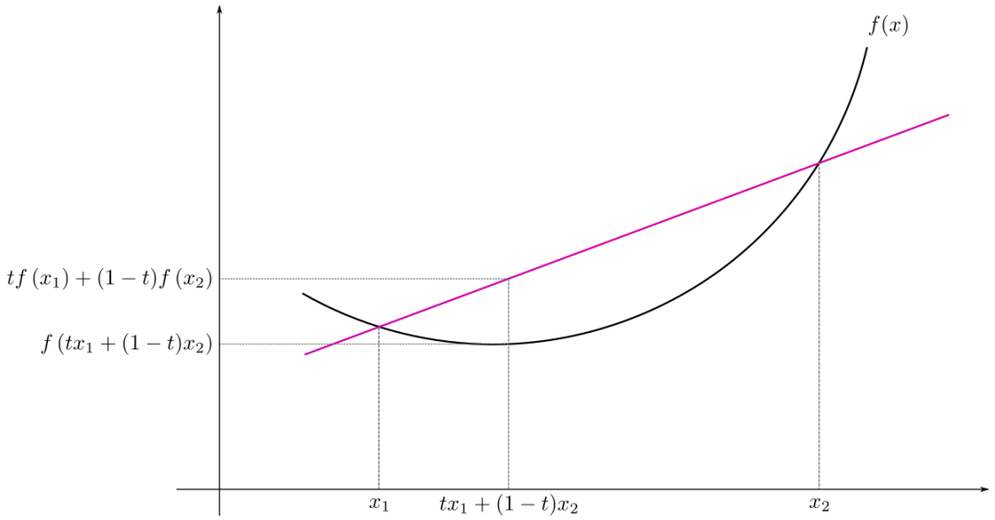
Ln 是底為歐拉數的自然對數

Logit 函數可將 P 轉成 Logit， Logit(0.5) = 0，所以如果 Logit 值 > 0，機率> 0.5
反函數就是 Sigmoid

LOGIT function calculator and graph (medcalc.org)

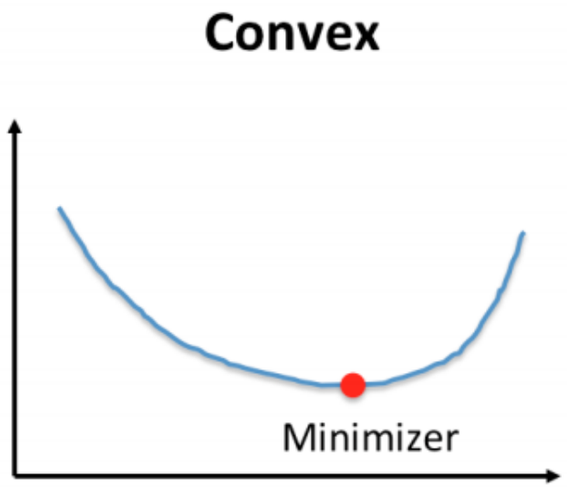
凸函數與優化

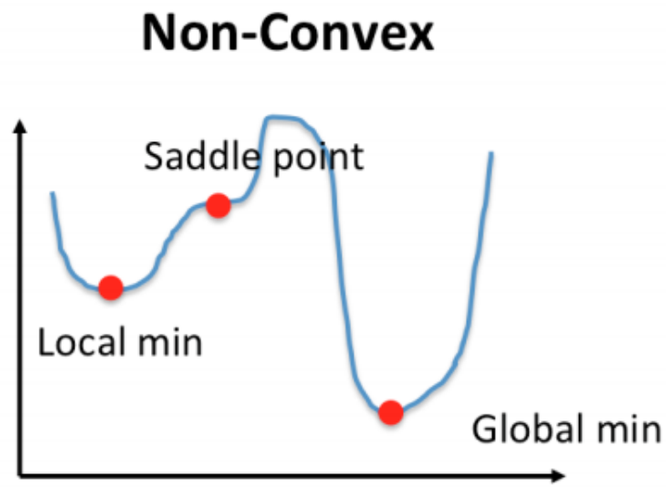
任何一條穿過凸函數曲線的直線，在相交兩點間，直線y之值永遠較凸函數f(x)之值大。



wikipedia.com

非凸函數之優化，可能造成本地優化(local optimum) 或鞍點(saddle point)





printerest.com

Adam 優化器

•**Adam** (Adaptive Moment Estimation) is a replacement optimization algorithm for stochastic gradient descent (**SGD**, fixed learning rate) for training deep learning models.

•Adam combines the best properties of the **AdaGrad** (每個參數都有 learning rate) and **RMSProp** (取每個參數average gradient 當個別 learning rate) algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

•Adam is relatively easy to configure where the default configuration parameters do well on most problems.

•在Tensorflow 的參數設置：learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.

Gentle Introduction to the Adam Optimization Algorithm for Deep Learning (machinelearningmastery.com)

第二天

2 小時

優化演算法(二)

2 小時

Tensorboard
CNN介紹

2 小時

CNN原理架構與應用

預習：Yolo, 文字探勘
作業：作業二

演算法中的正則化(Regularization)

目的：經由調整Loss Function (增加 L1, L2等)達到降低過適(Overfitting)。

L1: (LASSO) $||\mathbf{w}||_1 = |w_1| + |w_2| + ... + |w_N|$

L2: (RIDGE) $\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$

$$\hat{y} = w_1 x_1 + w_2 x_2 + ... + w_N x_N + b$$

Logistic Regression Loss Function: L(y_hat,y) = y log y_hat + (1 - y)log(1 - y_hat)

$$Loss = Error(y, \hat{y})$$

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

可參考：

[Regularization in Machine Learning - GeeksforGeeks]

第三天

2 小時

CNN物件偵測
(YOLO)

2 小時

期中複習
文字探勘

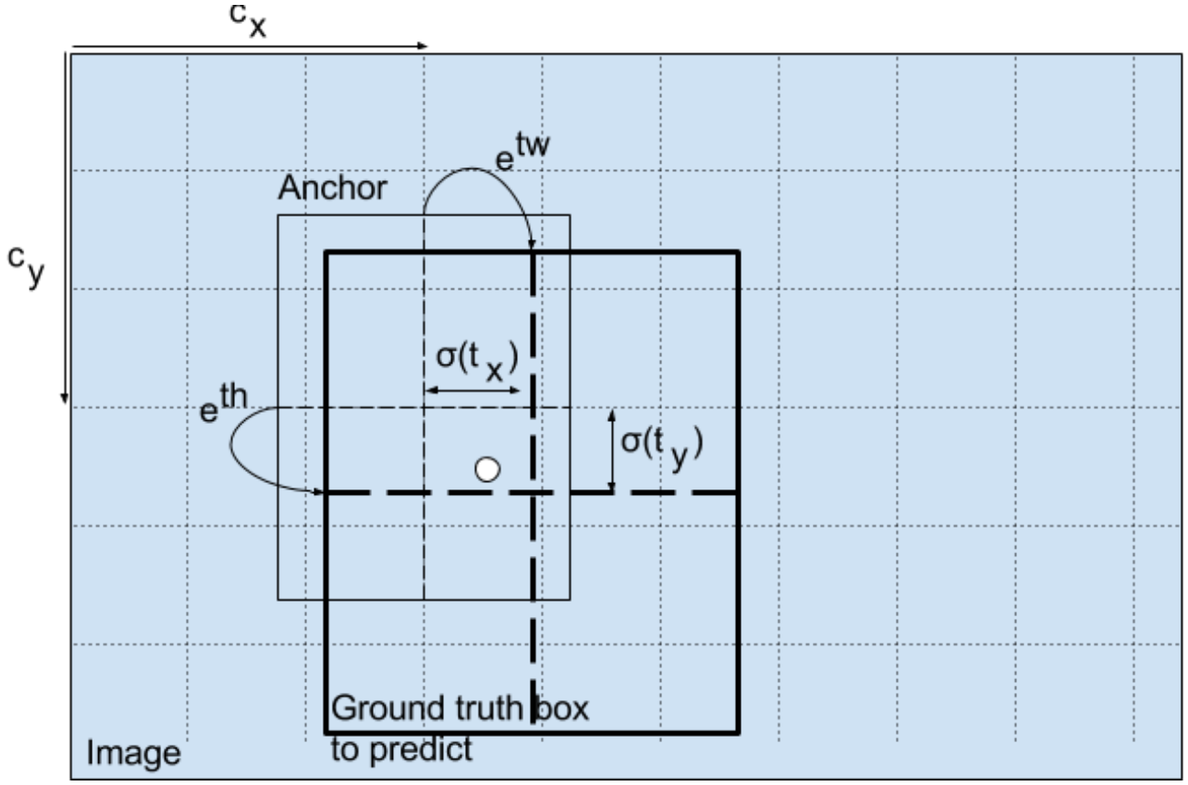
2 小時

文字探勘
CBOW
SKIPGRAM

預習：LSTM/GRU, 降維, AutoEncoder
作業：作業三

Why Anchor Box?

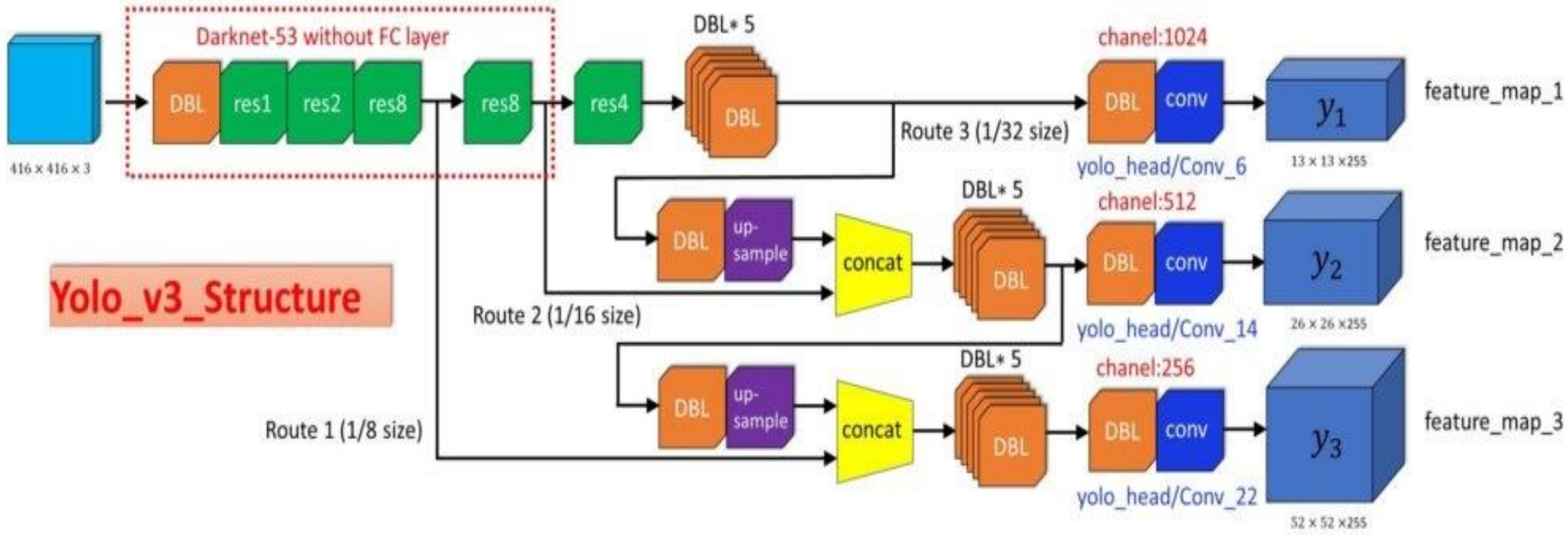
YOLO V3 的每個GRID CELL 都有 3x3個 以此CELL質心為中心的錨點框 (ANCHOR BOX)。利用他們預測此GRID CELL所負責的所有QUALIFIED BOUNDING BOXES。大中小加起來理論上有((13x13)+(26x26)+(52x52))x3 = 10647 個候選框

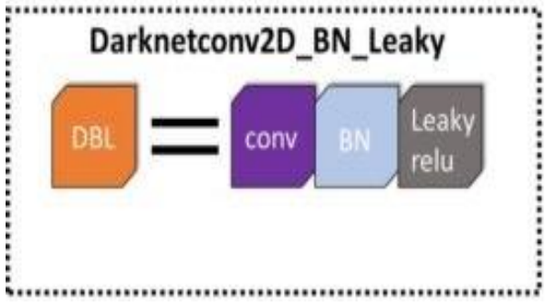


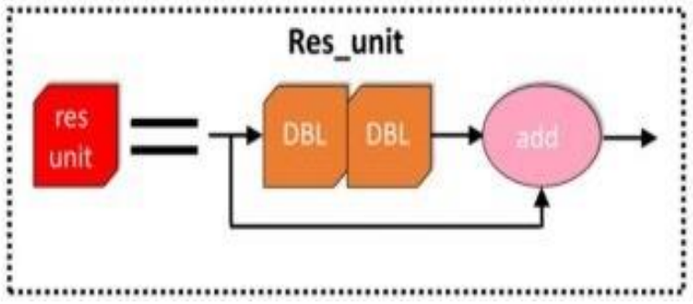
$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

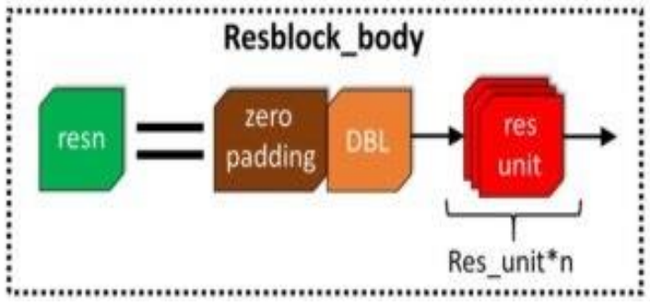
How to Implement a YOLO (v3) Object Detector from Scratch in PyTorch: Part 1 - KDnuggets

Yolo v3 Prediction









[#011 TF YOLO V3 Object Detection in TensorFlow 2.0 (datahacker.rs)]

NMS演算法

Input : $\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, N_t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq empty$ **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
      end
    end
  end
  return $\mathcal{D}, \mathcal{S}$
end

NMS

Gensim

Gensim = Generate Similar (產生類似的文件)

主要優點：
- 毋須將整個文件庫(Corpus)載入主記憶體中，可見進批次進行分析整理
- 可利用計算機硬體之多核結構加速運行
- 非監督式學習無須標籤
- 可與多種作業環境及平台整合

應用舉例：
- Word2Vec：文字轉向量技術
- TF-IDF (erm frequency–inverse document frequency)：文字轉向量技術
- 潛在語意分析(LSA : Latent Semantec Analysis)

學習：

Gensim - Introduction – Tutorialspoint

gensim: Core Concepts (radimrehurek.com)

第四天

2 小時

RNN原理
LSTM與GRU

2 小時

RNN進階應用
PCA 與 t-SNE

2 小時

AutoEncoder
圖像風格

預習：GAN，強化學習
作業：作業四

RNN Weights Saving/Sharing

T = 100 D = 10 M = 15 K = 1	輸入層 (Input Layer)	隱藏層 (Hidden Layer)	輸出層 (Output Layer)	總數
DNN	輸入攤平(flatten)	輸入(T*D) * 隱藏(M) * 步數(T)	步數(T) * 隱藏(M)	1.5M
DNN 權數公式	T * D = 1000	(T * D) * M * T = 1.5 Million	T * M * K = 1500	1.5M
RNN	Wxh = D * M = 150	Whh = M * M = 225	Who = M * K = 15	385

Spam Detection in RNN

DEMO9_SPAM_DETECTION.ipynb 說明

RNN 必須有 N x T x D
我們設定 T = data_train.shape[1]也就是整句話

我們希望Embedding的維數(Dimensionality) = 20，RNN 隱藏神經元 = 15
D = 20
M = 15
#先將所有字彙轉成20維度向量，然後再做LSTM/RNN訓練

```
i = Input(shape=(T,))
x = Embedding(V + 1, D)(i)
x = LSTM(M, return_sequences=True)(x)
x = GlobalMaxPooling1D()(x)
x = Dense(1, activation='sigmoid')(x)

model = Model(i, x)
```

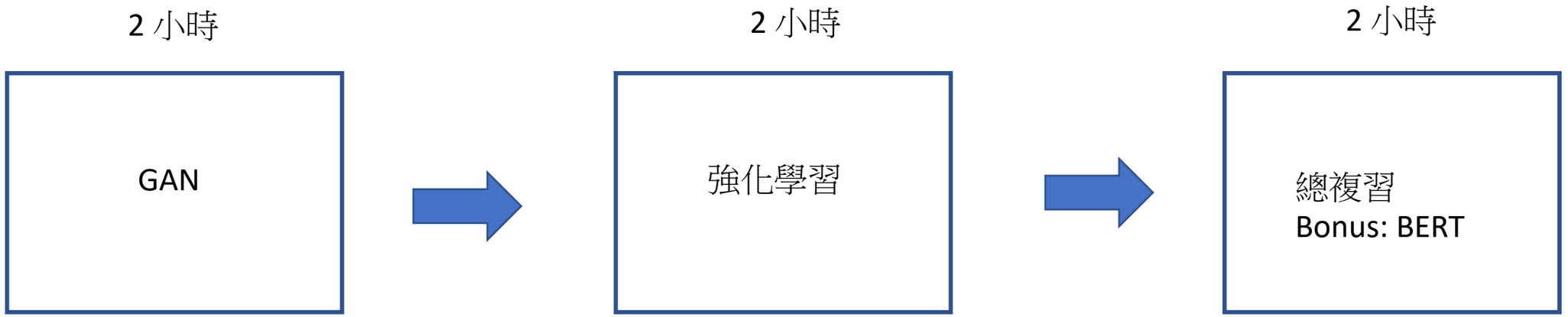
PCA 降維步驟

Steps Involved in the PCA

Step 1: Standardize the dataset.
Step 2: Calculate the covariance matrix for the features in the dataset.
Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix.
Step 4: Sort eigenvalues and their corresponding eigenvectors.
Step 5: Pick k eigenvalues and form a matrix of eigenvectors.
Step 6: Transform the original matrix.

Understanding Principle Component Analysis(PCA) step by step. | by The Nobles | Analytics Vidhya | Medium

第五天



預習：GAN，強化學習
作業：作業五

Cycle GAN Example

原理：將兩個互相映射的GAN，以強迫loss consistent (一致) 的方式造成來源和目的影像在轉換過程中擁有類似的風格或結構。

CycleGAN最大的特色是不需要成對的影像，就可以學習到風格是如何轉換的。如果一匹棕色的馬映射到一批斑馬，那麼一群棕色的馬就可以轉換成一群斑馬，不同情境下的馬，都會映射成類似情境的斑馬。維持轉換(映射)的一致性。

CycleGAN和Pix2Pix十分類似，不同點在於：

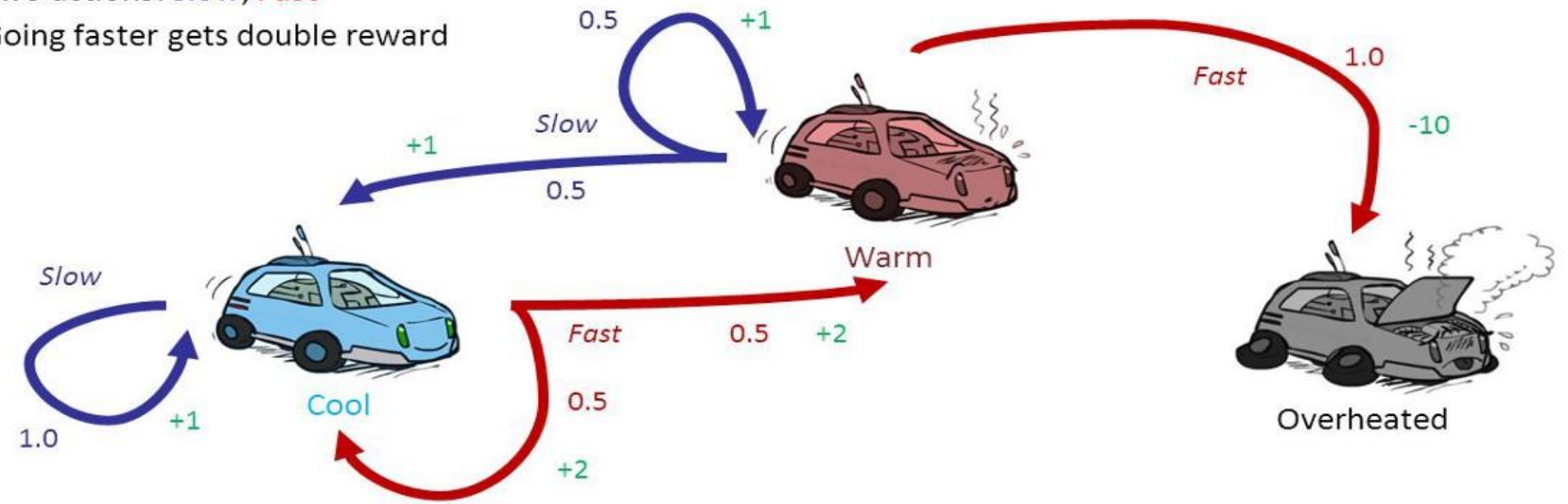
- CycleGAN 使用 Instance Normalization 而非 Batch Normalization
- CycleGAN 使用 ResNet Image Generator

CycleGAN | TensorFlow Core

馬可夫決策過程

Example: Racing

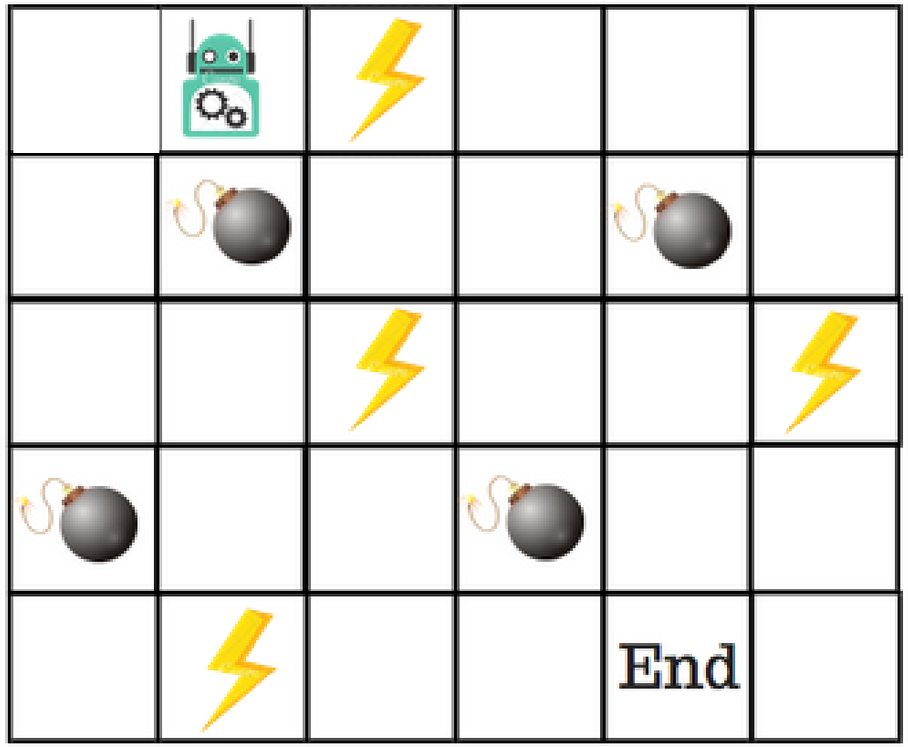
- A robot car wants to travel far, quickly
- Three states: Cool, Warm, Overheated
- Two actions: *Slow*, *Fast*
- Going faster gets double reward



medium.com

Markov 鍊模型：決策過程與過去無關，只和現在有關

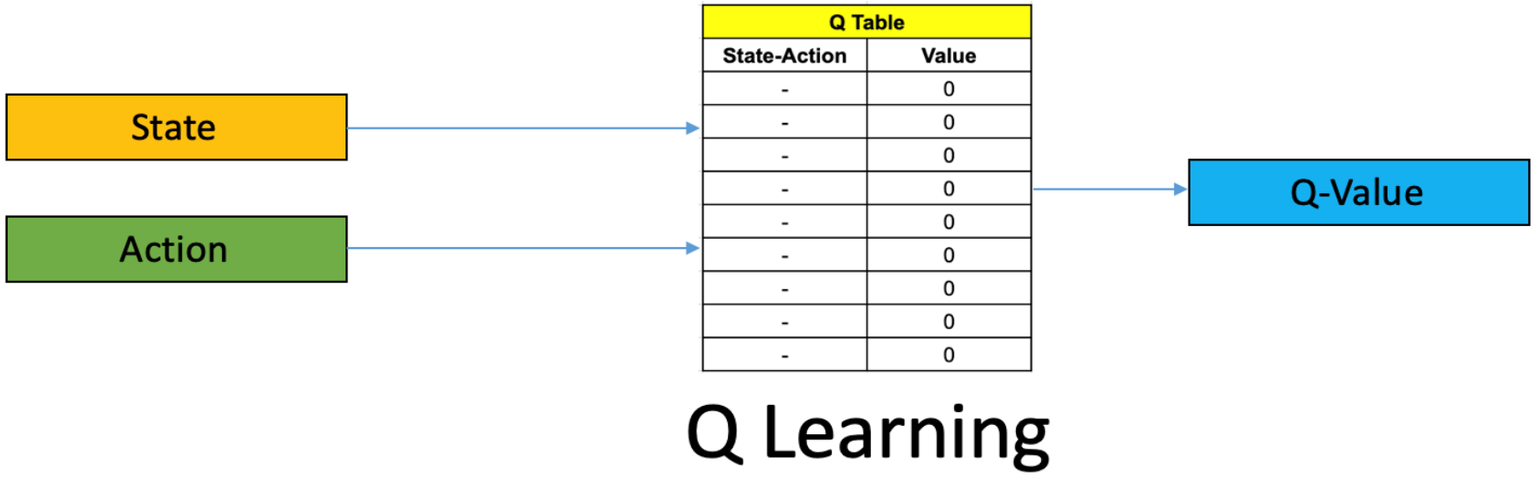
Q-Learning Table



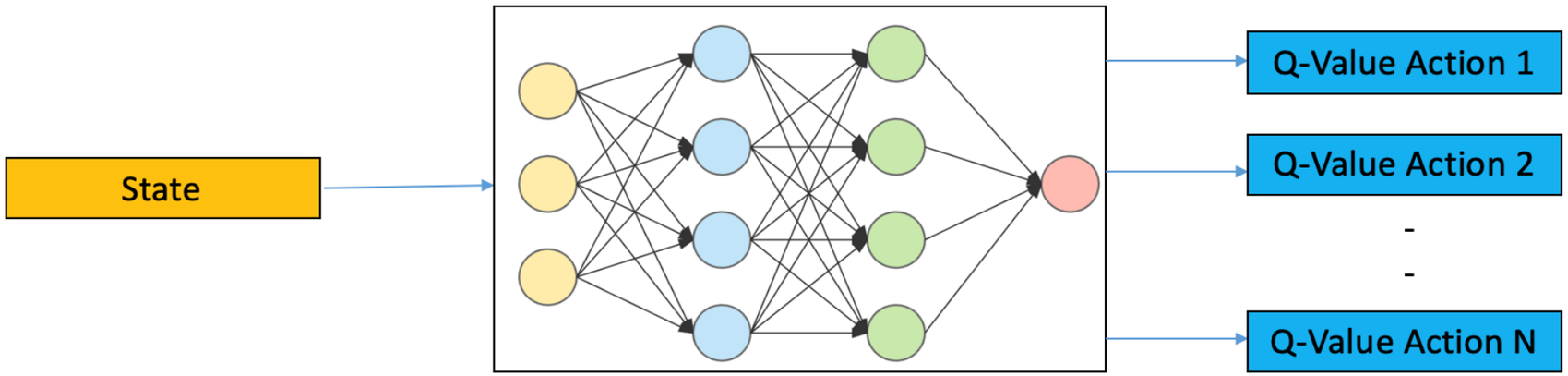
Actions :	↑	→	↓	←
Start	0	0	0	0
Nothing / Blank	0	0	0	0
Power	0	0	0	0
Mines	0	0	0	0
END	0	0	0	0

[An introduction to Q-Learning: reinforcement learning (freecodecamp.org)]

從 Q-table 到 DQN



Q Learning



Deep Q Learning

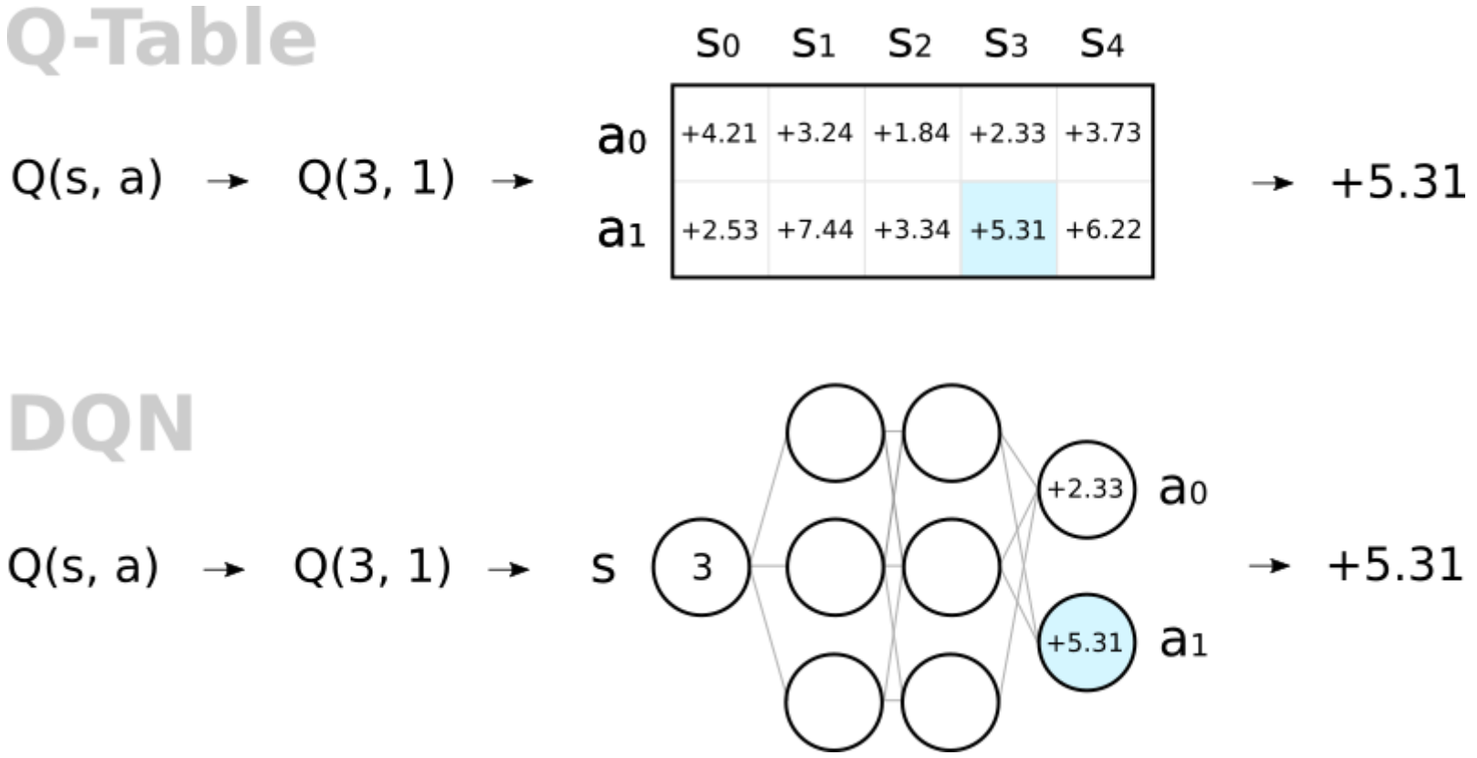
DQN其實就是用神經網路去模擬(逼近)Q-Table，DQN 直接產生各action再取最大值，而Q-Table 則找產生最大Q值的action。

一旦action被(policy)決定，Environment就會提供 reward 和下一個 state。Q-table 會根據新的state 去衡量所有可能的action，DQN則 輸入state 即可產生下一個action.

當State數變成無窮大時，Q-table將不敷使用，DQN經由訓練，其w與b的值將可取代被訓練後的Q(s,a)而達到同樣效果

Analyticsvidhya.com

從 Q-table 到 DQN：例子



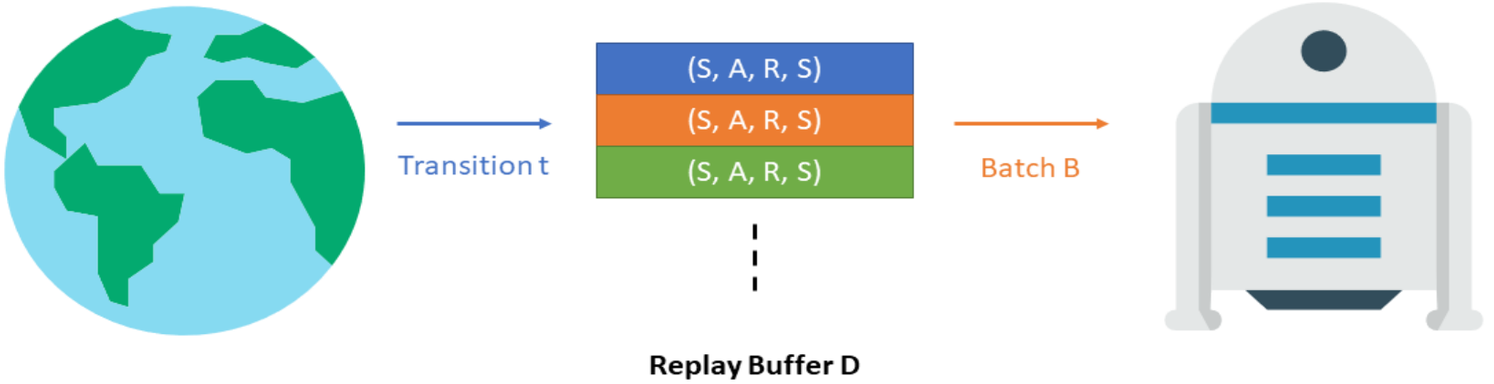
valohai.com

Experience Replay(經驗回放)

訓練時有效而多元，無須真實世界的模擬或執行，只雖機重覆回放(s,a,r,s’)值
收斂較快因而結果穩定

Experience Replay

- Save transitions $(S_t, A_t, R_{t+1}, S_{t+1})$ into buffer and sample batch B
- Use batch B to train the agent



endtoend.ai

學習目標：

- 了解Transformer
- 了解BERT

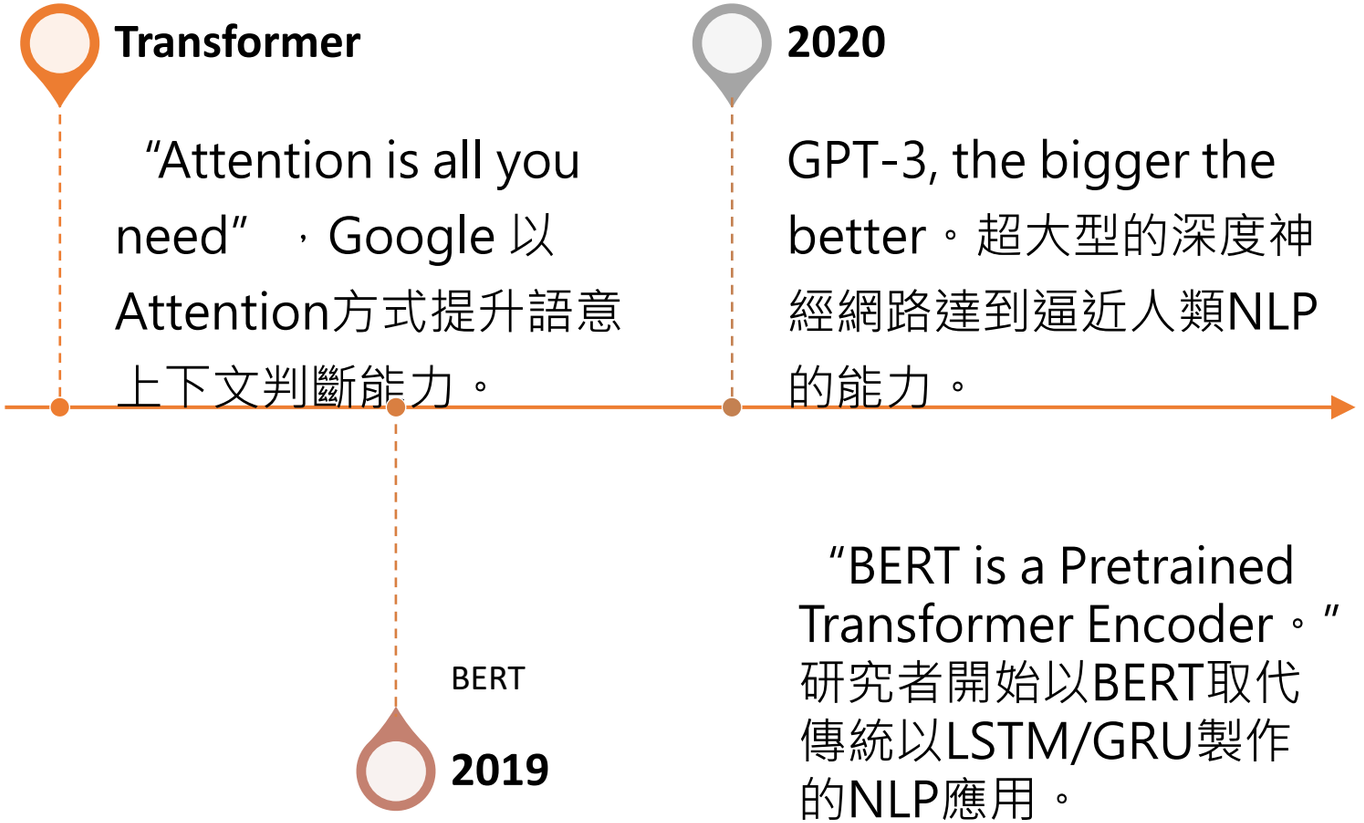
Bonus: BERT的原理與應用

BERT的原理與應用

內容大綱

- BERT的歷史淵源
- BERT的簡介
- BERT的應用：假新聞分類器
- BERT的其他應用

BERT的歷史淵源



BERT的簡介：BERT之前

- Elmo：雙向預訓LSTM based context-based Language Model
- ULM-FiT：在RNN/NLP 領域開始Transfer Learning 就像 CNN在 Computer Vision領域
- Transformer: 包括 Encoder和 Decoder，主要是建立自然語言翻譯的語言模型，他完全揚棄了 LSTM 模式而採用 Attention 模式。他的 Encoder-Deoder模式可用來做許多 downstream applications.
- OpenAI Transformer：利用 Transformer Decoder 做 Predicting Future Tokens。Encoder只留下self-attention layer. 應用包括 分類，蘊涵(entailment)，相似 (similarity)，選擇等。But it is forward direction only.

BERT的簡介： BERT的背景

- BERT (Bidirectional Encoder Representations from Transformers), 是一個 Pretrained Transformer 的 Encoder。Transformer 源自 Attention is all you need 的構想，揚棄傳統的RNN/LSTM/GRU的作法，不以順序(sequence)而以位置(position)達到平行訓練的目的，同時以一套Self-Attention的計算法更準地抓到了文句的上下文(Context)，進而提高幾乎所有NLP應用的準確度，包括分類與預測等等。

- 由於BERT系統龐大，即使是BASE model也有上億個神經元，非普通系統所能承受。我們一般都用Pretrained BERT Model再接上Fine Tuning Model來完成分類或其他應用，效果比起用RNN/LSTM好很多。

BERT的簡介： BERT的原理

- BERT的輸入有三個Embeddings，分別是 Token Embeddings (可能由Word2Vec, Glove等產生)，Position Embeddings紀錄順序位置，以及Segment Embeddings 紀錄相關或隨機字句。

- 在預訓時，BERT嘗試同時預測(15%)被掩蓋的字(Masked Language Model，MLM)以及下一句子預測 (Next Sentence Prediction)。

- BERT其實就是 預訓過的 Transformer Encoder，因為同時保有 字義，位置 和 段落 輸入的 embeddings，運用 Transformer Self-Attention 的能力，只要在後頭接上簡單的Linear Classifier or Regressor，就可完成許多NLP的應用。

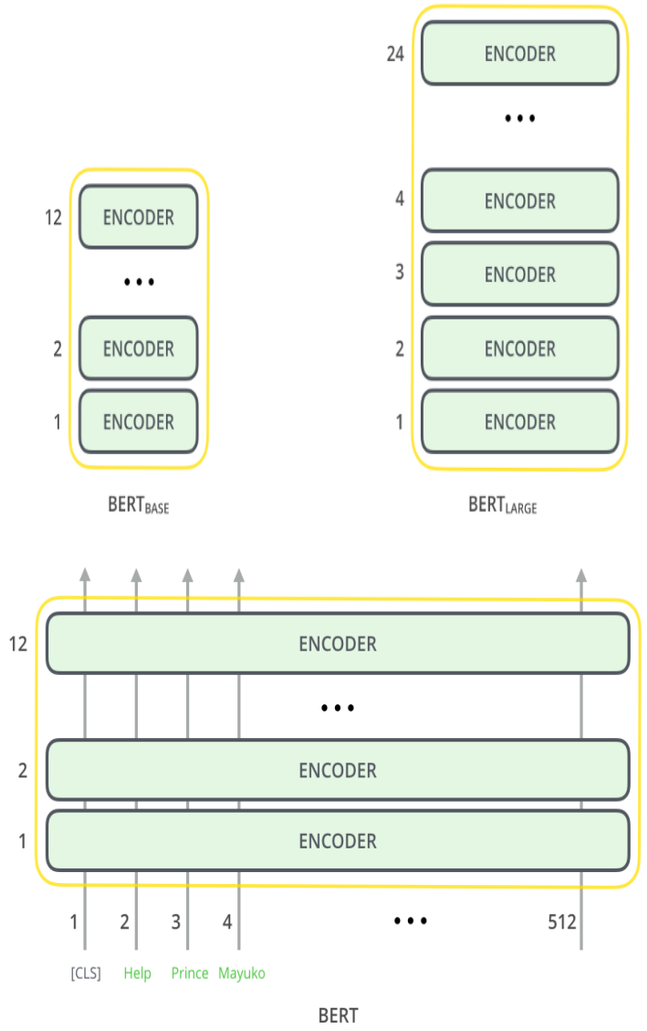
BERT的簡介：架構 – 輸入

常用的 BERT模型有兩種:

BERT BASE – 和OpenAI Transformer 大小差不多(12層)
BERT LARGE – 研發用非常大的模型(24層)

前面說過 BERT 基本上就是一個訓練過的 Transformer Encoder如右圖。BASE/LARGE 有 12/16 Attention Head， 784/1024 Hidden Unit。

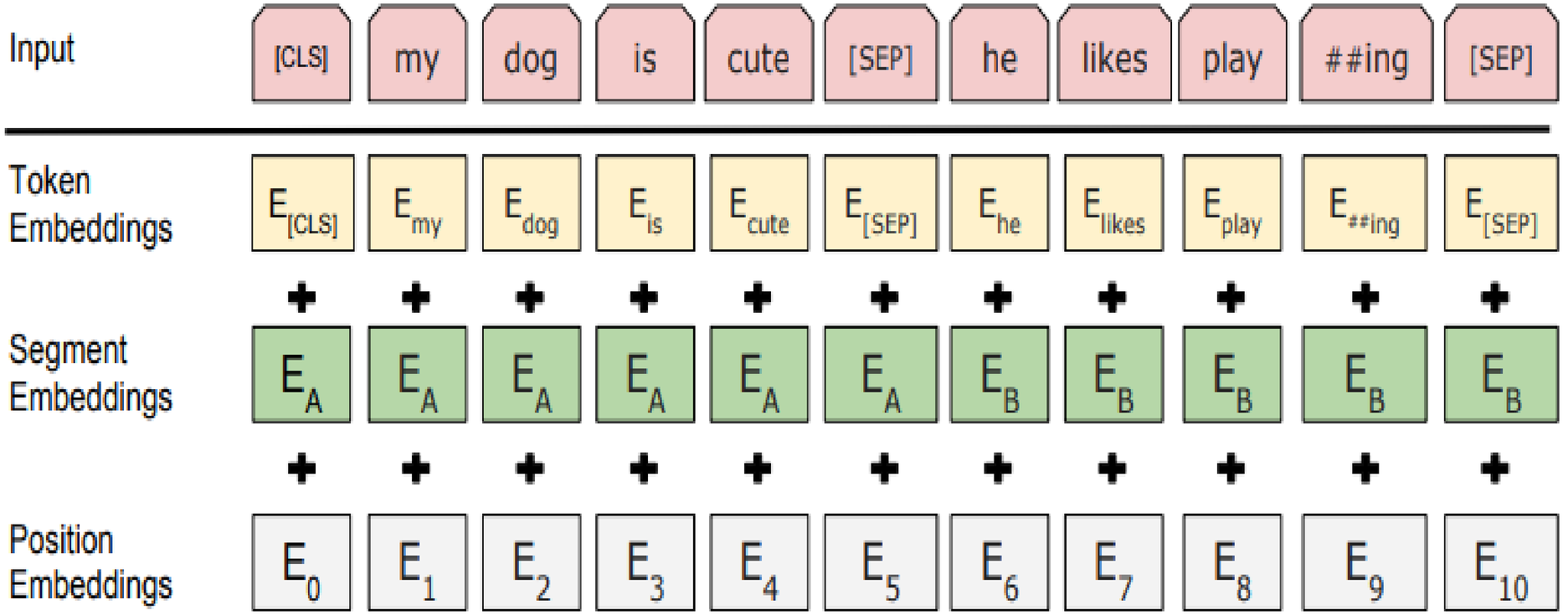
第一個input token 有一個特殊的 Classification Token [CLS]，Input token 不斷往上走，每一層都有self-attention層和Feed-forward層，然後在交給下一個Encoder.



BERT的簡介：輸入Embeddings

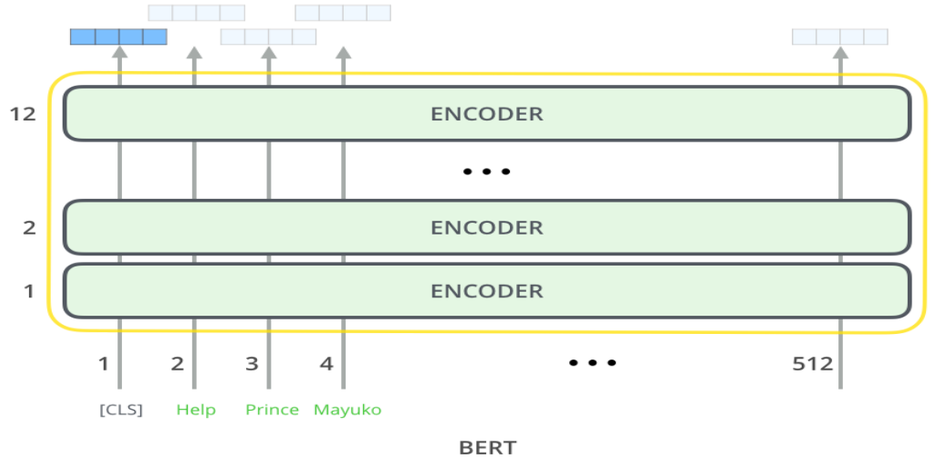
- BERT的輸入Embeddings其實是三個Embeddings的組合：
- Token Embedding: 就是你提供文章的句子加上[CLS][SEP][MASK]等特殊字符的向量，[CLS]代表句子的開端；[SEP]是相鄰(關)句子的間隔，像是句號；[MASK]是掩蓋掉的字符。
- Segment Embedding: 和[SEP]配合，表示第幾個句子。
- Position Embedding:輸入中的第幾個位置。

BERT的簡介：輸入Embeddings



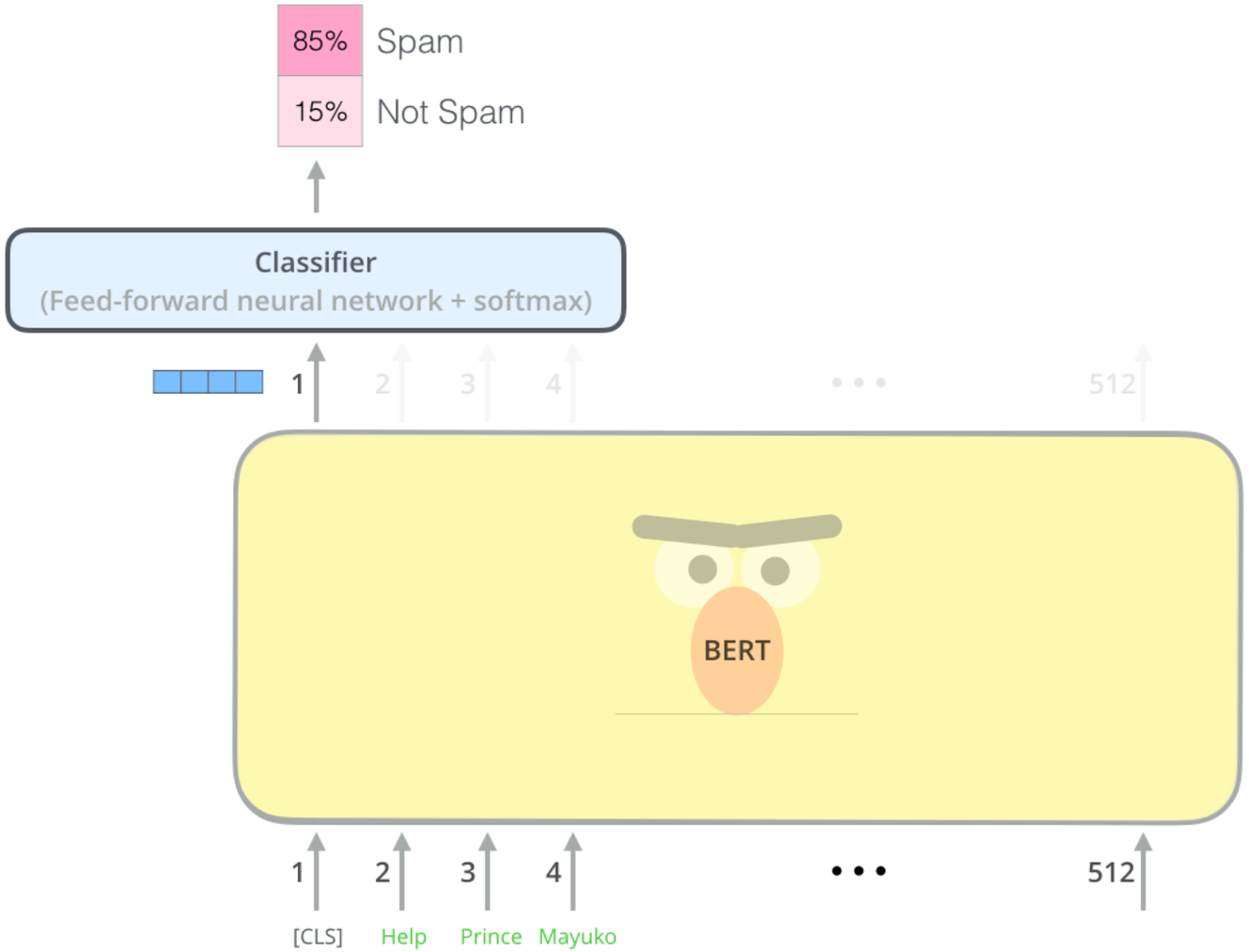
BERT的簡介：架構 – 輸出

輸出部分，512個輸出每個都有一個hidden_size=768(BERT Base)的向量，如果是做句子分類，其實只須注意第一個位置即可，也就是相對於[CLS]位置的那個。



這個向量可以輸入到我們設計的分類器. BERT作者僅用了單層的NN分類器即達到良好效果.

BERT的應用：假新聞分類器(圖示)



BERT的應用：假新聞分類器(簡述)

- 以新聞真偽這個應用的分類例題來說，我們主要在做dataset preprocessing以及 Classifier Model的 Fine Tune Training，中間主要的分類引擎是BERT BASE UNCASED。一旦將Label好的Real News 和 Fake News 的 Training Set, Test Set 和 Validation Set 分好，Preprocess就算完成。我們的PyTorch Fine Tune Python 程式主要就是將接口定義好，超參數設好，其實和其他的Classifier沒有甚麼不同。

- 例題：https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b
- **BERT fine-tune on fake news detection.ipynb**
- Use: PyTorch interface for BERT by Hugging Face

BERT的應用：假新聞分類器(前處理)

- 1. 下載Kaggle的Real/Fake News Dataset news.csv，放在Google Drive /transformers/data 裡：
  https ://www.kaggle.com/nopdev/real-and-fake-news-dataset

- 2. 在Colab 執行 **Preprocessing of Fake News Dataset.ipynb**

- 3. 要將Goggle Drive Mount 到 Colab的 Container，中間需要 Authorization Code

```
from google.colab import drive
drive.mount('/content/drive’  )
```

- 4. 按比例寫入 train.csv, test.csv 和 valid.csv

BERT的應用：假新聞分類器(載入資料)

- **tokenizer** = BertTokenizer.from_pretrained('bert-base-uncased’)
- text_field = Field(use_vocab=False, tokenize=**tokenizer**.encode, lower=False,
- include_lengths=False, batch_first=True,.....
- **train**, valid, test = TabularDataset.splits(path=source_folder, train='train.csv’ ……
- # Iterators
- train_iter = BucketIterator(**train**, batch_size=16, sort_key=lambda x: len(x.text),
- ……
- test_iter = ...
- valid_iter =……

.

BERT的應用：假新聞分類器(建模)

```
class BERT(nn.Module):

    def __init__(self):
        super(BERT, self).__init__()

        options_name = "bert-base-uncased"
        self.encoder = BertForSequenceClassification.from_pretrained(options_name)

    def forward(self, text, label):
        loss, text_fea = self.encoder(text, labels=label)[:2]

        return loss, text_fea
```

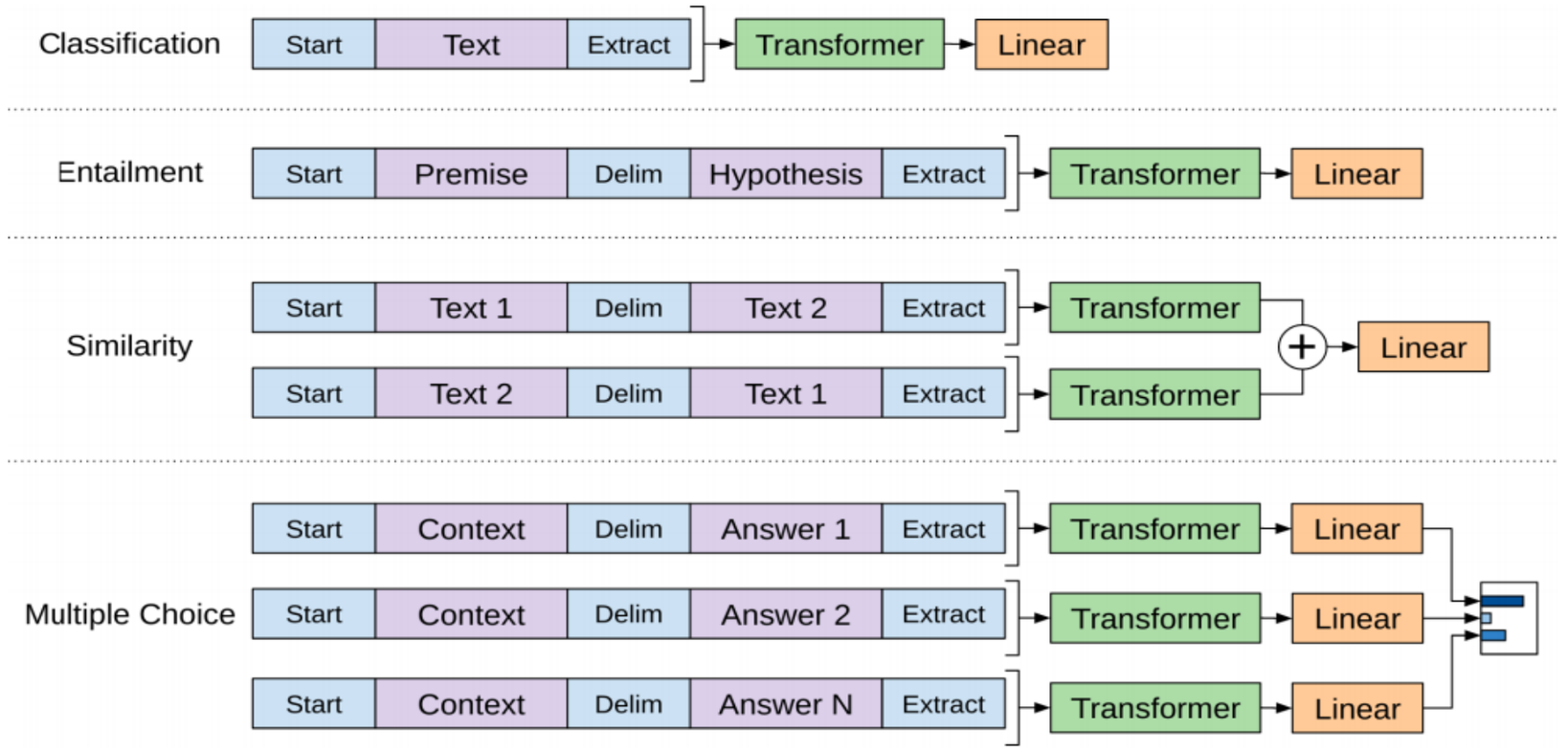
BERT的應用：假新聞分類器(訓練)

- model = BERT().to(device)
- optimizer = optim.Adam(model.parameters(), lr=2e-5)
- train(model=model, optimizer=optimizer)

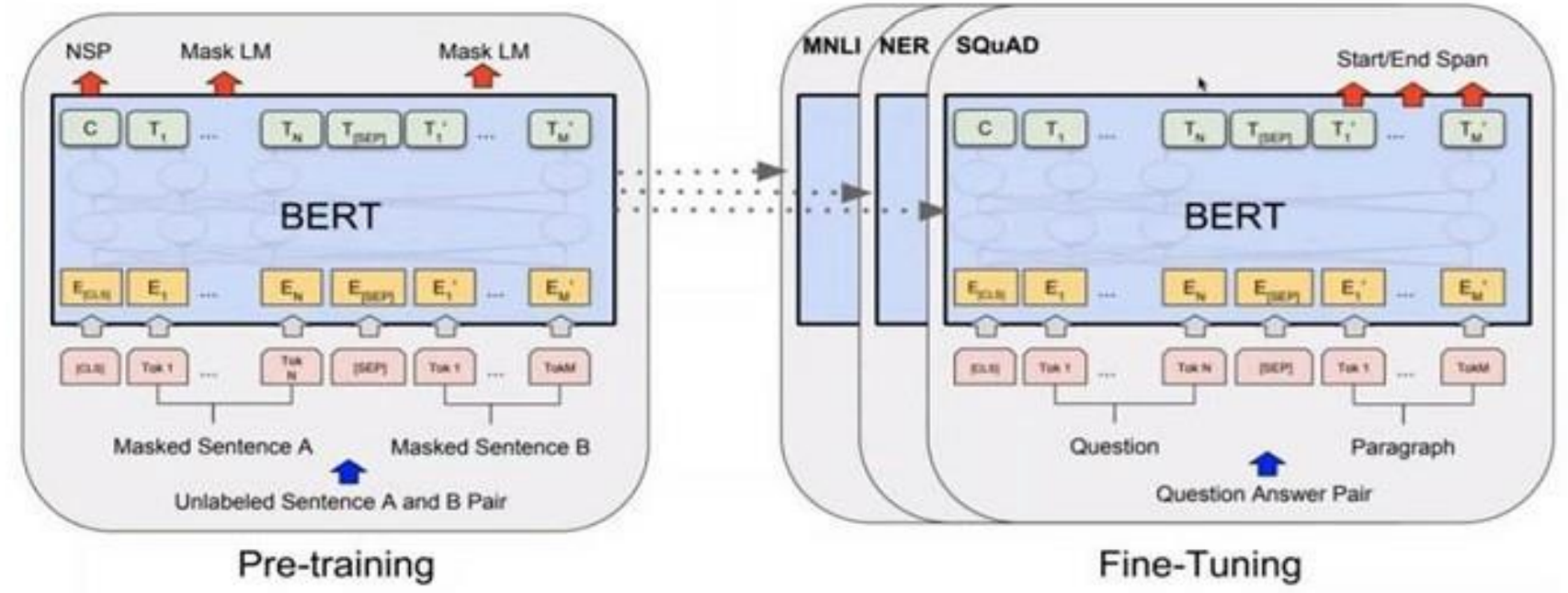
- # show loss, accuracy…
- # Evaluate
- 作業檔案: BERT fine-tune on fake news detection.ipynb

BERT的其他應用

BERT的作者號稱還可做以下這些NLP應用：分類，蘊涵，相似度比較，多選等，以下是簡單架構示意圖



BERT的的其他應用：模式



左邊是pre-trained: **NSP: Next Sentence Prediction, Mask LM: Masked Language Model**

右邊是fine-tuning應用：NER: Named Entity Recognition, SQuAD: 回答系統, MNLI：自然語言介面, 以及最常見的分類

BERT的其他應用：延伸練習

- 延伸學習任務一，用BERT MLM pretrained model 做**中間字的預測**，比方輸入
- text = "[CLS] Who was Jim Henson ? [SEP] Jim Henson was a puppeteer [SEP] "
- masked_index = 8
- tokenized_text[masked_index] = '[MASK]'
- 然後猜出是 Henson
- https://github.com/aniruddhachoudhury/BERT-Tutorials/tree/master/Blog%201
- 程式檔名：BERT_Tutorial_1.ipynb

- 延伸學習任務二，用 The Corpus of Linguistic Acceptability (CoLA) dataset **判斷文法是否正確**
- 程式檔名： BERT_Fine_Tuning_Sentence_Classification.ipynb

學習目標：

- 了解轉換學習
- 了解 TFLite

Bonus: Transfer Learning

轉換學習：TFLite 與 AIoT

Tensorflow Lite:

能夠讓行動(Mobile)和嵌入式(Embedded)應用使用機器學習(ML)的一組軟體工具，這些應用因而可在 iOS, Android, Raspberry Pi上執行。

但是如何將訓練好的模型輸出給行動應用開發者呢？

遷移學習：TFLITE與AIOT

高階步驟: (訓練端Training Side)
#1 正常訓練模型
從server/desktop訓練模型
把model 和 weights 遷移到行動裝置
一旦在行動裝置，model即可做
prediction/inference
#2 用Tensorflow Lite Converter 轉換模型
到.tflite 檔.
假設是 Keras API
.tflite 只是一個檔案模式, 就像 model.save() 產生 .h5 檔一樣
在 “import tensorflow as tf” 之後加幾行 code

遷移學習：TFLITE與AIOT

高階步驟: (行動開發者端)

#3 在行動開發者端, 使用 Tensorflow lite 庫.
支持 Java/C++ (Android) 和 Swift/Objective-C (iOS)
用 Tensorflow Lite Interpreter 載入 .tflite 檔並且用現有模型做預測。
數據需要和Tensorflow Lite interpreter相匹配.

遷移學習：TFLITE與AIOT

要改變的程式，訓練端

```
#convert the model to TFLite format

converter =
tf.lite.TFLiteConverter.from_keras_model(model)

tflite.model = converter.convert()

with open(  “converted_model.tffile”  , wb) as f:
      f.write(tflite_model)
```

遷移學習：TFLITE與AIOT

TFLite vs TF Serving

行動裝置也可用Serving直接對server做API calls.
TF Lite 讓你直接在行動/嵌入裝置使用模型，無需使用網路.
如果在地鐵或通訊不良處，依賴雲端不是很好的選擇。
但如果模型複雜需大量運算TF Lite 就不是很好的選擇.

遷移學習：TFLITE與AIOT

對無ML背景的行動裝置開發者:
預訓練好的模型包括: Object detection, pose estimation, smart reply.

https://www.tensorflow.org/lite/models

學習目標：

- 了解 推薦系統
- 實作推薦系統

Bonus: Recommender System

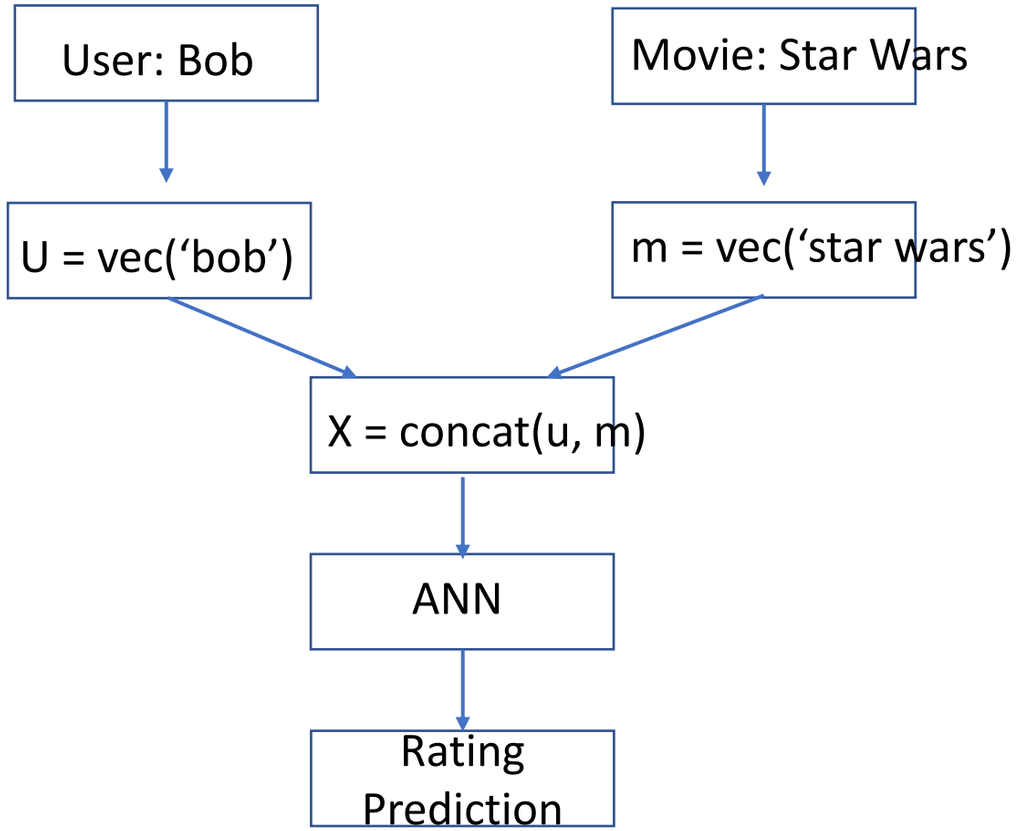
推薦系統的製作

推薦系統的製作

推薦系統三元素：(User, Item, Rating)通常以電影為例
Ratings 數據一定非完整
運用機器學習去趨近 f(user, item) → rating
如果我們已有預測函數 f(), 我們只需選擇user尚未看過的項目(比方說電影)從最高依降序推薦.
但 (user, item) 非數值而是範疇型(categorical), 然而NN需要做矩陣運算
在NLP中我們已得到靈感，用 embedding，將(user, item)轉成相關的向量

推薦系統的製作

推薦系統流程：



推薦系統的製作

Pseudocode：

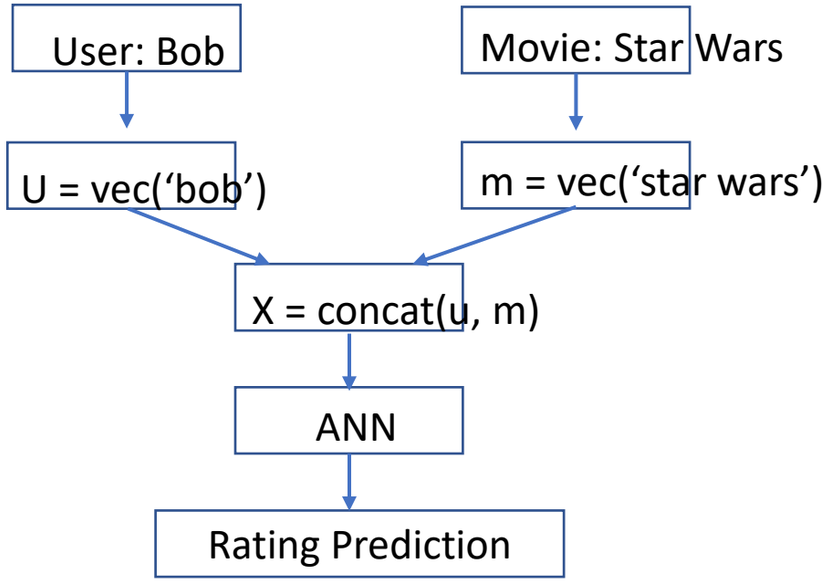
```
u = Input(shape((1,))
m = Input(shape((1,))
u_emb = Embedding(num_users, embedding_dim)(u)
m_emb = Embedding(num_movies, embedding_dim)(m)
#combine into a single feature factor
x=Concat(u_emb, m_emb)
#ANN
x=Dense(512, activation= ‘relu’ )(x)
X=Dense(1)(x)
```

推薦系統的製作

用Functional API, 而非 sequential API

Sequential API 無法合併兩個模型，
因為每一層只能連接到另一層.

Functional API 提供NN的彈性，
比方GAN的設計.



推薦系統的製作

Demo21_Recommender System.ipynb

Note: The ID 需要連續，因為它是對向量的索引.

需要先扁平化(flatten)再接續 (concatenate): Nx1xD → NxD

Homework: 在老師示範此檔案之後，嘗試尋求改進val_loss的任何方法。
Hint: 改變 hyper parameter.