考題[第二版] for 機器學習專家

- 考試時間: 120分鐘
- 建議 part 1 使用40分鐘，part 2 使用80分鐘

Part 1: Maths and Common Senses (50%)

- 本大題可以在紙上作答，再拍照上傳
- 本大題可以用中文或英文作答

Quiz 1-1: Back Propagation (20%)

- 下面是單層 hidden layer 的類神經網路，loss function 使用 cross entropy
  - $L = \text{CE}(y, \hat{y})$
  - $\hat{y} = softmax(\theta)$
  - $\theta = W^{(2)}h + b^{(2)}$
  - $h = \sigma(a)$
  - $a = W^{(1)}x + b^{(1)}$
  - $x \in \mathbb{R}^n$、$W^{(1)} \in \mathbb{R}^{d \times n}$、$b^{(1)} \in \mathbb{R}^d$、$h \in \mathbb{R}^d$、$W^{(2)} \in \mathbb{R}^{c \times d}$、$b^{(2)} \in \mathbb{R}^c$、$\theta \in \mathbb{R}^c$
- 請計算 $\frac{\partial}{\partial \theta} CE(y, \hat{y})$ 及 $\frac{\partial}{\partial W^{(2)}} CE(y, \hat{y})$

Quiz 1-2: Common Senses (30%)

- 在評估模型表現時，為什麼要用 F1-score 而不是用 precision?
- 為什麼類神經網路不能用 binary classification function 作為 activation function?
  - Note: binary classification function $f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$
- 機器學習演算法的 bias 及 variance 是什麼意思?
- 訓練 random forest 時，為什麼 tree 不需要剪枝 (pruned)?
- 什麼是 one-hot encoding?
- 如何避免類神經網路 overfitting? 至少寫三個答案，但越多越好

Part 2: Python Skills (50%)

- 請先下載本大題使用的資料：https://goo.gl/BDB6bE
- 你可以使用任何PIP可安裝的package

Quiz 2-1: word co-occurrence matrix (20%)

- 有人說過，一個單字的意義，可以由上下文 (context) 來決定
- 我們定義「上下文」為單字的前二個字及後二個字
  - 例如 `I have a dog and a cat`，其中 `dog` 的上下文是 `have, a, and, a`
- co-occurrence 是記錄上下文關係的格式
  - 假設所有單字為 $w_1, w_2, ..., w_N$
  - co-occurrence matrix 是一個 $N \times N$ 的矩陣，其中 (i, j) 的位置代表 w_i, w_j 出現在上下文的次數

- 例如 `I have a dog and a cat`，會使 `(dog, have) += 1`、`(dog, a) += 1`、`(dog, and) += 1`、`(dog, a) += 1`

```
def q21_cooccur_matrix(filename='raw_sentences.txt', window=2):
    '''
    Arguments
        filename: the filename of an English article
        window: context window，定義「上下文」的範圍
    Returns
        vocab: 將單字對應到 id
        inv_vocab: 將 id 對應到單字
        cooccur_matrix: NxN np.ndarray，代表 co-occurrence matrix
    '''
    return vocab, inv_vocab, cooccur_matrix

vocab, inv_vocab, cooccur_matrix = q21_cooccur_matrix()
```

Quiz 2-2: word vectors (10%)

- 使用10維的PCA，將 `cooccur_matrix` 轉換成 shape `(N, 10)`

```
def q22_word_vectors(cooccur_matrix, dim=10):
    '''
    Arguments
        cooccur_matrix: co-occurrence matrix with shape (N, N)
        dim: PCA的維度，預設10維
    Returns
        word_vectors: word vector matrix with shape (N, 10)
    '''
    return word_vectors

word_vectors = q22_word_vectors(cooccur_matrix)
```

Quiz 2-3: word similarities (10%)

- 使用 cosine distance 找出某個單字最近的三個單字
- cosine distance 介於0到1之間，而 `1 - cosine_distance` 就代表相似度

```
def q23_similarity(word, wv=word_vector, vocab=vocab, inv_vocab=inv_voca
b):
    '''
    Arguments
```

```
        word: 要找的單字
        wv: word vector matrix with shape (N, 10)
        vocab: vocabulary
        inv_vocab: inverse vocabulary
    Returns
        ret: 長度為3的list，每個元素都是 tuple (word, similarity)
    '''
    return ret

for word, sim in q23_similarity(word):
    print(word, sim)
```

Quiz 2-4: Python data model (20%)

- 寫一個 `WordVector` class，將上面的過程包裝成物件，並方便查詢
  - 建立instance時，可以提供三項參數，`filename`、`window`、`dim`
- `wv = WordVector('raw_sentences.txt')` 產生word vector instance
- `wv['office']` 取得某個字的 vector，這裡的 vector 必須經過 normalize，意指長度為1
- `wv.most_similar('office')` 取得與某個字最接近的三個字，傳回格式為 tuple `(word, similarity)`

```
class WordVector(object):
    def __init__(self, filename, window=2, dim=10):
        pass

wv = WordVector('raw_sentences.txt')
print(wv['office'])
for word, sim in wv.most_similar('office'):
    print(word, sim)
```