# 考題 for 機器學習專家

- Quiz time: 120 minutes
  - we suggested you spend 40 minutes in part 1 and 80 minutes in part 2

## Part 1: Maths and Common Senses (50%)

### Quiz 1-1: Back Propagation (20%)

- Given an neural network with one hidden layer, and the loss function is cross entropy loss
  - $a = W^{(1)}x + b^{(1)}$
  - $h = \sigma(a)$
  - $\theta = W^{(2)}h + b^{(2)}$
  - $\hat{y} = softmax(\theta)$
  - $L = \mathrm{CE}(y, \hat{y})$
  - $x \in \mathbb{R}^n$ 、$W^{(1)} \in \mathbb{R}^{d\times n}$ 、$b^{(1)} \in \mathbb{R}^d$ 、$h \in \mathbb{R}^d$ 、$W^{(2)} \in \mathbb{R}^{c\times d}$ 、$b^{(2)} \in \mathbb{R}^c$ 、$\theta \in \mathbb{R}^c$
- Calculate $\frac{\partial}{\partial \theta}L$ and $\frac{\partial}{\partial W^{(2)}}L$

**Note:** you can write answers on a paper and take a photo of the paper

### Quiz 1-2: Common Senses (30%)

**Note:** you can answer in Chinese or English.
**Note:** your answers can be as simplest as possible

- When will we use F1-score instead of precision(accuracy)?
- Why don't we use binary classification function as the activation function in neural networks?
  - Note: binary classification function $f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$
- What is the bias and variance of a machine learning algorithm?
- When training a single tree in random forest, we don't prune the tree, why?
- What is one-hot encoding?
- How to prevent overfitting in neural networks? (write down anything you know)

## Part 2: Python Skills (50%)

### Quiz 2-1: word counts (10%)

Download this English article from `https://goo.gl/BDB6bE` first.
We want to know the probabilities of each bigram and trigram.
Please write a function `ngram_probs` to calculate these counts.
**Note:** please convert text to lower cases before calculating.
**Note:** please keep punctuations in the text.
**Note:** in all quizs, you can use any Python packages on PyPI.

```python
def ngram_probs(filename='raw_sentences.txt'):
    return bigram_probs, trigram_probs
```

```python

cnt2, cnt3 = ngram_probs()
print(cnt2[('we', 'are')])
```

**Note:** an n-gram is a contiguous sequence of n items from a given sequence of text or speech. For example, `('we',)` is an unigram, `('we', 'are')` is a bigram, and `('we', 'are', 'here')` is a trigram.

## Quiz 2-2: next word probabilities (10%)

Write a function `prob3` to calculate the probabilities of next word of a bigram.
The probability of `'family'` after bigram `('we', 'are')` is the probability `('we', 'are', 'family')` divided by the probability `('we', 'are')`.

$$P(w_3|w_1, w_2) = \frac{P(w_1, w_2, w_3)}{P(w_1, w_2)}$$

$$P(\text{family}|\text{we are}) = \frac{P(\text{we are family})}{P(\text{we are})}$$

**Note:** In this quiz, you are requested to return **log probabilities.**

```python
def prob3(bigram, cnt2=cnt2, cnt3=cnt3):
    return prob

p = prob3(('we', 'are'))
print(p['family'])
```

## Quiz 2-3: predicting the next word (10%)

Write a function `predict_max` to complete the sentence by finding the max likelihood word.
For example, if the starting bigram is `('we', 'are')`, the max likelihood word is `'going'`.
Given the next word `'going'` , current bigram becomes `('are', 'going')`, and then we can predict the next word. The predicting process will end when the next word is `'.'` or length of the sentence is larger than 15.

```python
def predict_max(starting, cnt2=cnt2, cnt3=cnt3):
    return list_of_words

sent = predict_max(('we', 'are'))
assert sent[-1] == '.' or len(sent) <= 15
print(' '.join(sent))
```

## Quiz 2-4: beam search (20%)

Write a function `predict_beam` to complete the sentence by beam search https://en.wikipedia.org/wiki/Beam_search. “Complete” means find all sentences which end in “.” and have `sent_length` tokens (including “.”).

```python
def predict_beam(bigram, beam_size=4, sent_length=10, cnt2=cnt2, cnt3=cnt3):
    return list_of_sentence

for sent in predict_beam(('we', 'are')):
    assert sent[-1] == '.' or len(sent) < 10
    print(' '.join(sent))
```