1 Part A: Calculus

1.1 Calculus-Q1

CQ1.1 Use the chain rule to calculate the gradient of

$$h(\boldsymbol{\sigma}) = \frac{1}{2}\|f(\boldsymbol{\sigma}) - y\|^3$$

where $\boldsymbol{\sigma} \in \mathbb{R}^m$ and f is some arbitrary function from $\mathbb{R}^m$ to $\mathbb{R}^n$.
CQ1.2 Compute the expression in the case where:

$$f(\boldsymbol{\sigma}) = \begin{bmatrix} \sigma_1 \cdot \sigma_2 \\ \sigma_1^2 + \sigma_2^2 \\ \sigma_1 \end{bmatrix}$$

$$\boldsymbol{\sigma} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$h(\sigma) = \frac{1}{2} \cdot \|r(\sigma)\|^3$

let:

$r(\sigma) = f(\sigma) - y, \quad g(u) = \frac{1}{2}\|u\|^3$

hence:

$h(\sigma) = g(r(\sigma))$

let us find gradiants:

$\|u\| = (u \cdot u^T)^{\frac{1}{2}}$

$g(u) = \frac{1}{2}\|u\|^3 = \frac{1}{2}(u \cdot u^T)^{\frac{3}{2}}$

$g'(u) = \frac{1}{2} \cdot \frac{3}{2} \cdot (u \cdot u^T)^{\frac{1}{2}} \cdot (u \cdot u^T)' = \frac{1}{2} \cdot \frac{3}{2} \cdot (u \cdot u^T)^{\frac{1}{2}} \cdot (u^2)'$

$= \frac{1}{\cancel{2}} \cdot \frac{3}{\cancel{2}} \cdot \|u\| \cdot \cancel{2}u = \frac{3}{2}\|u\| \cdot u =$

let $u = r(\sigma)$:

$g'(u) = \frac{3}{2}\|r(\sigma)\| r(\sigma)$

we evaluate $f'(\alpha)$ as the jacobian:

$$J_f(\sigma) = \begin{bmatrix} \frac{\partial f_1}{\partial \sigma_1} & \cdots & \frac{\partial f_1}{\partial \sigma_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial \sigma_1} & \cdots & \frac{\partial f_n}{\partial \sigma_m} \end{bmatrix} \in \mathbb{R}^{n \times m}.$$

$f'(\sigma) = \begin{pmatrix} 2 & 1 \\ 2 & 4 \\ 1 & 0 \end{pmatrix}$

we want to find gradiant $h(\sigma)$

evaluated at:

$$f(\boldsymbol{\sigma}) = \begin{bmatrix} \sigma_1 \cdot \sigma_2 \\ \sigma_1^2 + \sigma_2^2 \\ \sigma_1 \end{bmatrix}$$

$$\boldsymbol{\sigma} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

by the chainrule:

$$h'(\sigma) = g'(r(\sigma)) \cdot r'(\sigma)$$

$$\downarrow$$

$$h'(\sigma) = g'(r(\sigma)) \cdot f'(\sigma)$$

$$= \frac{3}{2} \|r(\sigma)\| \cdot r(\sigma) \cdot f'(\sigma)$$

$$= \frac{3}{2} \|r(\sigma)\| \cdot J_r(\sigma)^T \cdot r(\sigma)$$

let us evaluate:

$$\sigma = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad y = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$f(\sigma) = \begin{pmatrix} 2 \\ 5 \\ 1 \end{pmatrix} \rightarrow f'(\sigma) = \begin{pmatrix} 2 & 1 \\ 2 & 4 \\ 1 & 0 \end{pmatrix}$$

$$\downarrow$$

$$r(\sigma) = \begin{matrix} 1 \\ 4 \\ 0 \end{matrix}$$

$$h'(\sigma) = \frac{3}{2} \cdot \left\| \begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix} \right\| \cdot \begin{pmatrix} 2 & 2 & 1 \\ 1 & 4 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix}$$

$$\boxed{= \frac{3}{2} \cdot \sqrt{17} \cdot \begin{pmatrix} 10 \\ 17 \end{pmatrix}}$$

1.2 Calculus-Q2

The softmax function $S : \mathbb{R}^k \to [0,1]^k$, is defined as follows:

$$S(x)_j = \frac{e^{x_j}}{\sum_{l=1}^{k} e^{x_l}}$$

This function takes an input vector $x \in \mathbb{R}^d$ and outputs a probability vector (non-negative entries that sum up to 1), corresponding to the weight of original entries of x.
CQ2: Calculate the Jacobian of the softmax function S.

$$S'(x) = \begin{bmatrix} \frac{\partial S_1}{\partial x_1} & \cdots & \frac{\partial S_1}{\partial x_K} \\ & & \\ & & \frac{\partial S_K}{\partial x_K} \end{bmatrix}$$

let $Z = \sum_{i=1}^{K} e^{x_i}$.

$$\frac{\partial S_i}{\partial x_j} : \frac{d}{dx_j} \cdot \frac{e^{x_i}}{\sum x_j} \overset{*}{=} \frac{\frac{d}{dx_j}(e^{x_i}) \cdot Z - e^{x_i} \cdot \frac{d}{dx_j}(Z)}{Z^2}$$

$$\overset{**}{=} \frac{\delta_{ij} \cdot e^{x_i} \cdot Z - e^{x_i} e^{x_j}}{Z^2}$$

$$= \frac{e^{x_i}}{Z} \left(\delta_{ij} - \frac{e^{x_j}}{Z}\right)$$

$$* \ \frac{f(x)}{g(x)} = \frac{f'(x) g(x) - f(x) g'(x)}{g(x)^2}$$

$$\cdot \ \left(\sum e^{x_j}\right)' = \sum \left(e^{x_j}\right)' = \sum e^{x_j}$$

$$** \ \frac{d}{dx_j} e^{x_i} = \begin{cases} e^{x_i} & i=j \\ 0 & i \neq j \end{cases}$$

$$\delta_{ij} = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$$

$$\frac{d}{dx_j}(Z) = \frac{d}{dx_j} \sum e^{x_i} = \begin{cases} e^{x_j} & i=j \\ 0 & i \neq j \end{cases}$$

therefore

$$J_{ij} = \begin{cases} S(x)_i \cdot (1 - S(x))_j & i=j \\ -\frac{e^{x_i} \cdot e^{x_j}}{Z^2} = -S(x)_i S(x)_j & i \neq j \end{cases}$$

$$\boxed{J = \begin{bmatrix} S(x_1) \cdot (1 - S(x_1)) & -S(x_1) \cdot S(x_2) & \cdots & -S(x_1) S(x_K) \\ \vdots & S(x_2) \cdot (1 - S(x_2)) & & \\ & & \ddots & \\ -S(x_K) \cdot S(x_1) & & & S(x_K) \cdot (1 - S(x_K)) \end{bmatrix}}$$

part 2: scenarios:

SCENARIO 1: Two Prophets, One Game

```
Average test error:        0.3067
Approximation error:       0.2000
Estimation error:          0.0940
Best prophet chosen:       53/100 times
```

SCENARIO 2: Two Prophets, Ten Games

```
Average test error:        0.2213
Approximation error:       0.2000
Estimation error:          0.0240
Best prophet chosen:       88/100 times
```

SCENARIO 3: Many Prophets, Ten Games

```
Average test error:        0.0930
Approximation error:       0.0064
Estimation error:          0.0852
Best prophet chosen:       3/100 times
Within 1% of best:         8/100 times
```

analysis: if the error rates were uniformly distributed between [0, 0.5] instead of [0, 1], the **approximation error would stay largely the same**, since we have haven't changes the lower bound on True risk. meanwhile, the **estimation error would be substantially smaller**, since even when we select a model whose true risk is greater than that of the best available model, the true risk of the chosen model will be closer to 0.

SCENARIO 4: Many Prophets, Many Games

```
Average test error:        0.0064
Approximation error:       0.0064
Estimation error:          0.0008
Best prophet chosen:       50/100 times
Within 1% of best:         98/100 times
```

analysis: if we evaluate the genralisation gap of a model based on the train set, we expect it to be **greater** than if we had masured generalisation gap based on the model's perfomrnace on the test. that is becuase the train set is a relatively small subset compared to the poplation and the model selection was biased in favor of a model performing well on the train set. no such bias affects the performance on the test set, and hence the test set provides a better approxiamtion of the gnerealisation gap.

SCENARIO 5: School of Prophets

```
Grid search completed
  k values (prophets):     [2, 5, 10, 50]
  m values (train games):  [1, 10, 50, 1000]
  Number of trials:        100
```

Scenario 5: School of Prophets Results

Average Test Error

	m=1	m=10	m=50	m=1000
k=2	0.0987	0.0861	0.0738	0.0703
k=5	0.1006	0.0658	0.0409	0.0294
k=10	0.1019	0.0665	0.0239	0.0187
k=50	0.0960	0.0653	0.0178	0.0034

Approximation Error

	m=1	m=10	m=50	m=1000
k=2	0.0643	0.0730	0.0683	0.0707
k=5	0.0346	0.0340	0.0351	0.0298
k=10	0.0184	0.0176	0.0155	0.0182
k=50	0.0036	0.0036	0.0034	0.0030

Estimation Error

	m=1	m=10	m=50	m=1000
k=2	0.0341	0.0126	0.0048	0.0003
k=5	0.0667	0.0332	0.0053	0.0002
k=10	0.0836	0.0486	0.0083	0.0003
k=50	0.0925	0.0623	0.0143	0.0004

the grid search shows clear patterns: as we increase **k** (class size) we increase two facotrs:

1. the chance of the available prophets including a prothet with a lower true risk, thus **decreasing approxiamtion error**` indeed k is almost entirely responsible for the approxiamtion error.
2. increasing the chance of including a prophet that happened to perform well on the trian set, while having a greater true risk, thus **increasing estiamton error and test error**. meanwhile, increasing **m** (the training set size) makes it harder for a model to perform well on the train set despite having a higher true risk thus, it **decreases the estimation error and the test error**.

indeed, we see that when we increased the number of games for ERM from scenario 1 to 2, we lowered the estimation and test error as expected.

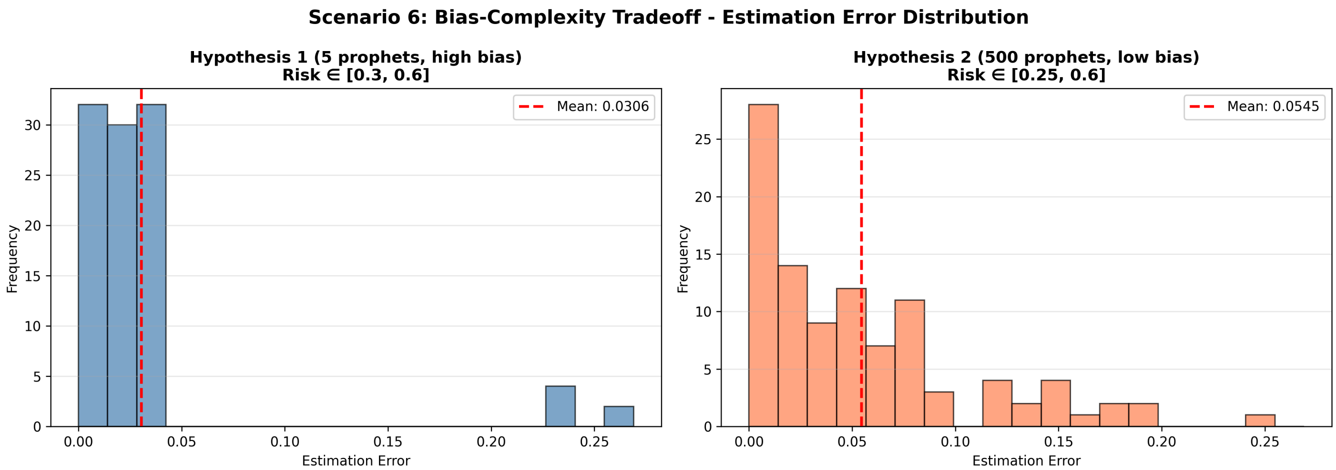
furthermore, scenarios 3 and 4 exactly show how the approxiamtion error is affected primarily by k, while incresing m led to lower test and estiamtion errors.

SCENARIO 6: Bias-Complexity Tradeoff

```
Hypothesis 1 (5 prophets, high bias):
  Average test error:      0.3570
```

```
  Approximation error:     0.3230
  Estimation error:        0.0306

Hypothesis 2 (500 prophets, low bias):
  Average test error:      0.3044
  Approximation error:     0.2500
  Estimation error:        0.0545
```



we see that while class 2 has models sampled from a 'better' range of true risk (lower bound 0.25 as opposed to 0.3), the estiamtion error is higher. this is as expected, since given a larger calss size there is a greater likelihood of choosing a sample with low error on the train set despite higher true risk. meanwhile, the fact that the models in calss 2 on average have lower true risk, leads to a lower test error.

part 3: pac learning analyis:

in this part we use the formula relating the number of samples to accuracy and cofidence: `m = 2 * np.log(2 * h / delta) / epsilon`, which can also be expressed as: `m = (2 / epsilon) * (np.log(2)+np.log(h)-np.log(delta))` we note that m is inversely proportional to epsilon and to the log of delta. it is proportional to the log of h.

QUESTION 1: Compute minimal number of samples

```
  Input Parameters:
    |H| (hypothesis class size) = 100
    epsilon (desired accuracy)  = 0.05
    delta (confidence level)    = 0.01


Output: m (rounded up) = 397
```

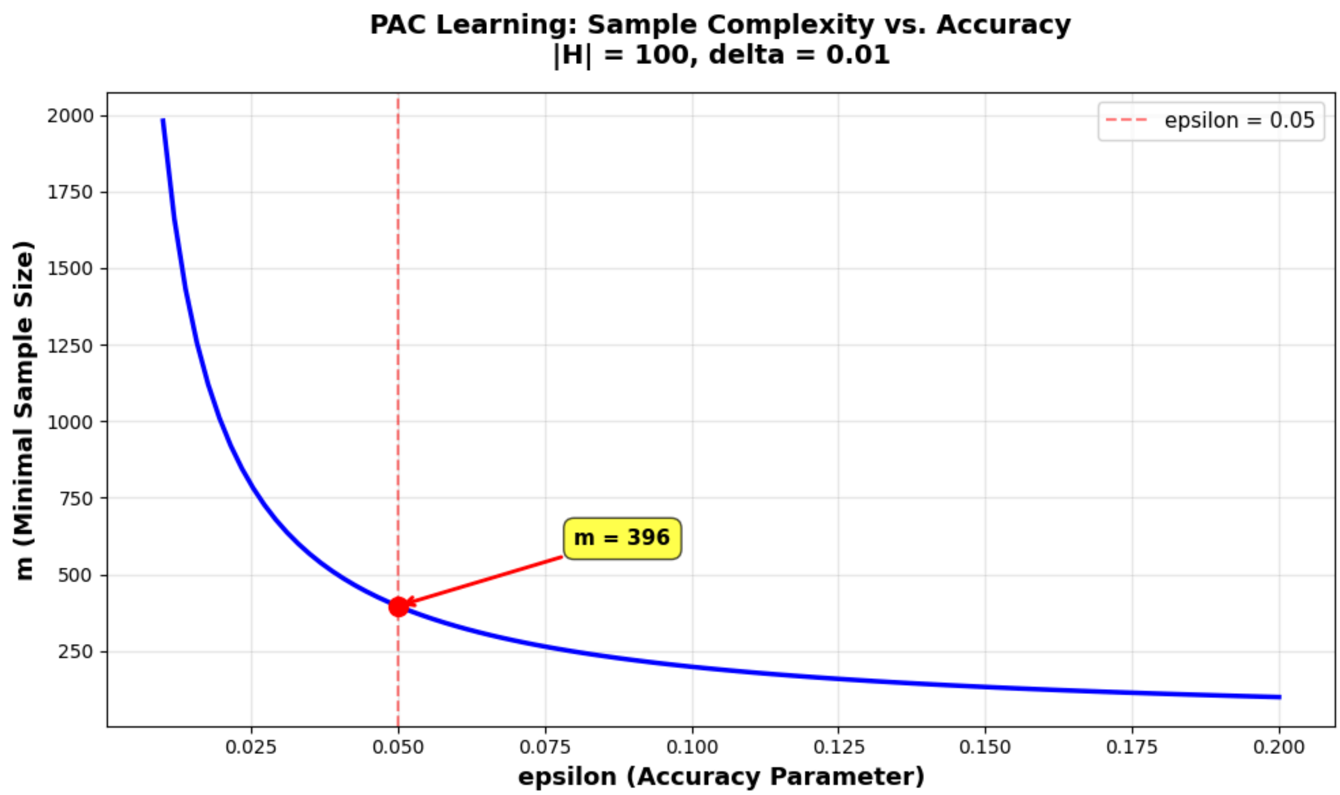
QUESTION 2: Analyze change when |H| is doubled

```
  Input Parameters:
    |H| (hypothesis class size) = 200 (doubled)


Output: m (rounded up) = 424
```

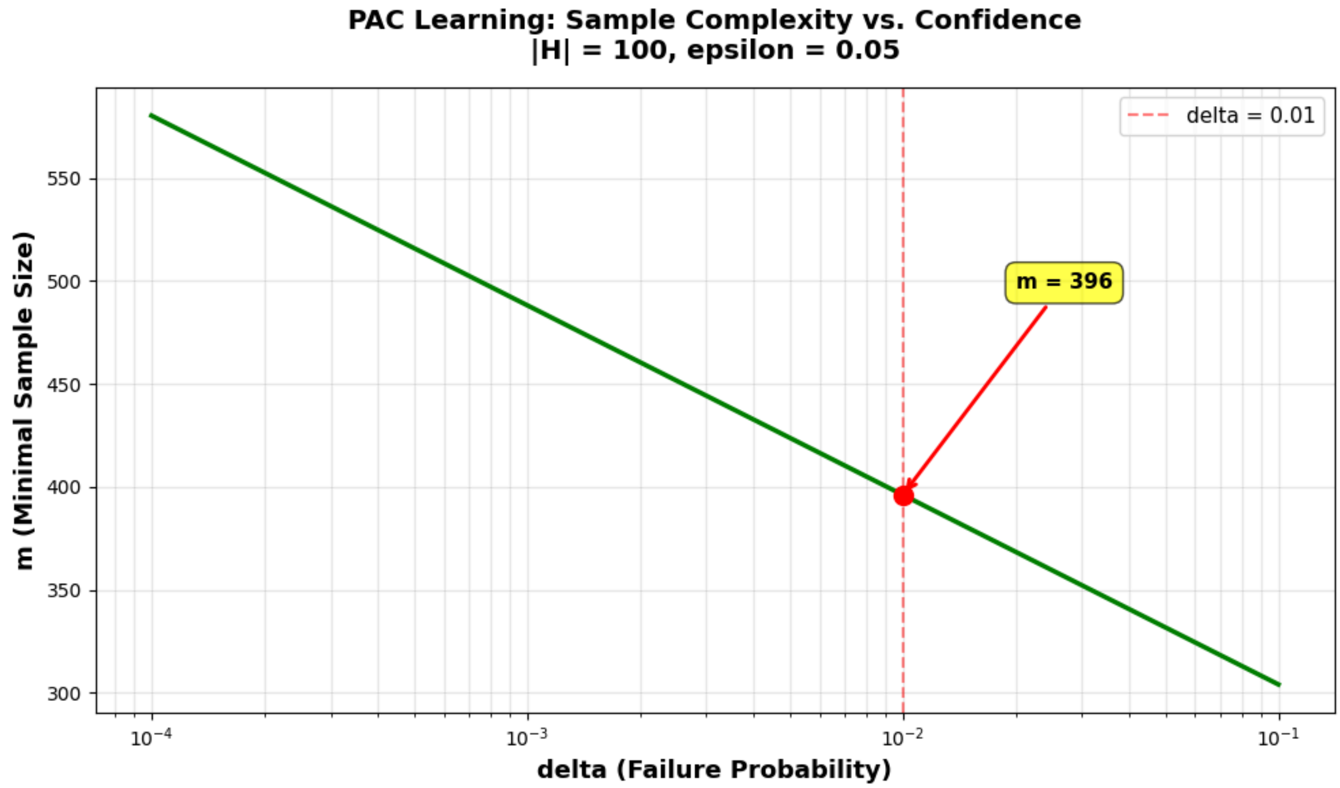
we see that if we double |H| we only need an additional 27 samples. if we refactor the equation we get `m = orignial_val + (2 / epsilon) * np.log(2)' which comes to 27

QUESTION 3: Plot m as a function of epsilon in [0.01, 0.2]



we clearly see m increase in inverse proportion to epsilon. hence, to halve the error, we need to double the number of samples.

QUESTION 4: Plot m as a function of delta in [10^-4, 0.1] (log scale)



we clearly see that curve follows an exact logarimic scale. hence, to increase the confidence 10 fold, we need only increase the numerator by a factor of log(10), making this scale well.