SIMC Sample problem: Who are colluding?

Preamble.

You have been tasked to help identify students who might have been colluding during tests. The suspicion is that a few groups of students use a secretive and successful mode of communication during tests to share their answers. If they did, then we expect their responses to multiple questions on the test to be similar.

Fortunately, you know machine learning! Let us decompose into several more manageable tasks.

Loading the data.

```
loaded = np.load('sample.npz')
for k in loaded.keys():
     print(f"key {k} has shape: {loaded[k].shape}")
    >>> key sample_small has shape: (50, 25)
    >>> key sample_larger has shape: (10000, 50)
```

Task 1: Plotting the design matrix.

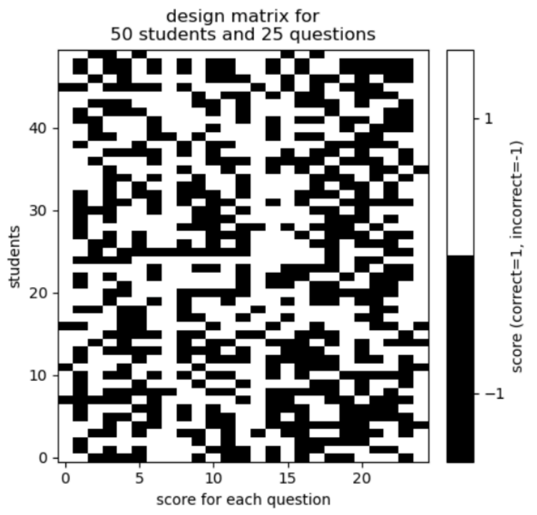

Figure 1: Small design matrix that compactly shows the test scores (right/wrong = $1/-1$) for 50 students on a test with 25 questions.

Getting to know the design matrix.

Linear algebra is essential to machine learning. Linear algebra does not merely condense large datasets into compact matrices. Linear algebra helps us (1) understand key mathematical and statistical properties of these matrices and (2) standardize numerical operations that can be optimized to execute in parallel on computing hardware (e.g., multi-core CPUs and GPUs).

We start by representing the students' scores on a test as a large **design matrix $\boldsymbol{X}$**. Each row of $\boldsymbol{X}$ represents a particular student's scores. Each of the 25 columns of a specific row indicates the student's scores for the 25 different questions on the test. So if the 10th student scores full credit for question 20, then (row=10, column=20) of this matrix shows the value 1 (conversely, value -1 if the student does not obtain full credit).

To help us abstract and generalize the problem (an important part of machine learning!), we introduce index notation, which is commonly used in linear algebra. Let's represent this design matrix as $\boldsymbol{X} \equiv X_{ij}$, where the first and second subscripts (i, j) denote the value at the i^{th} row and j^{th} column. For the example above, $X_{10,20} = 1$ if the 10th student scores full credit for question 20, otherwise $X_{10,20} = -1$.

For completeness, let there be M students (i.e., rows of $\boldsymbol{X}$ are $i = 1, \ldots, M$), N questions (i.e., columns of $\boldsymbol{X}$ are $j = 1, \ldots, N$).

Plot your design matrix `sample_small`, similar to the one shown in Figure 1. Remember: always label your axes!

Task 2: Write a simple routine to compute the following similarity matrix:

$$S_{ij} \equiv \boldsymbol{X}\boldsymbol{X}^T = \sum_{k=1}^{N} X_{ik} X_{jk} \,. \tag{1}$$

- You should present your results visually, similar to Figure 2.
- Reminder: while the indices of a matrix are mathematically denoted $j = 1, 2, \ldots, N$, the same index starts from zero instead when represented in `python 'X[0], X[1], ... X[N-1]'`.

Clustering with K-means.

To readily identify anomalies in the students' test-taking behaviors, we will use a prevalent (but elementary) unsupervised clustering algorithm: the **K-means** algorithm. As a clustering algorithm, it finds similar subgroups (i.e., clusters) within a larger group of items. Being an unsupervised algorithm, it can find clusters without needing you to **label** specific items (i.e., this student did not cheat, that one cheated, etc.).

In this context, the K-means algorithm tries to find K groups of students that are more similar within each group than between groups. Put differently, if we assign $K = 3$, then the algorithm tries to partition the students into three groups where the scores of in-group students are more mutually similar than the scores of out-group students. The reason the algorithm is termed K-**means** is because it first starts off by assuming $K = 3$ sets of random scores as the **mean score** within the three groups of students (i.e., $\boldsymbol{\mu}_K$). The algorithm then proceeds in two alternating steps:

- A) **Update membership**: Each student is tentatively assigned as a member to the K^{th} group whose corresponding mean $\boldsymbol{\mu}_K$ is closest to each student's score. Each student's group membership is often called the **labels**.
- B) **Update means**: based on the group assignment in step (A), the means of each of the K groups (i.e., $\boldsymbol{\mu}_K$) are updated.
- C) Go back to step (A) if the means $\boldsymbol{\mu}_K$ have changed appreciably; otherwise, the algorithm ends.

This simple set of alternating steps (**update membership** then **update means**, and so on) is surprisingly effective in partitioning students into distinct and meaningful groups. However,

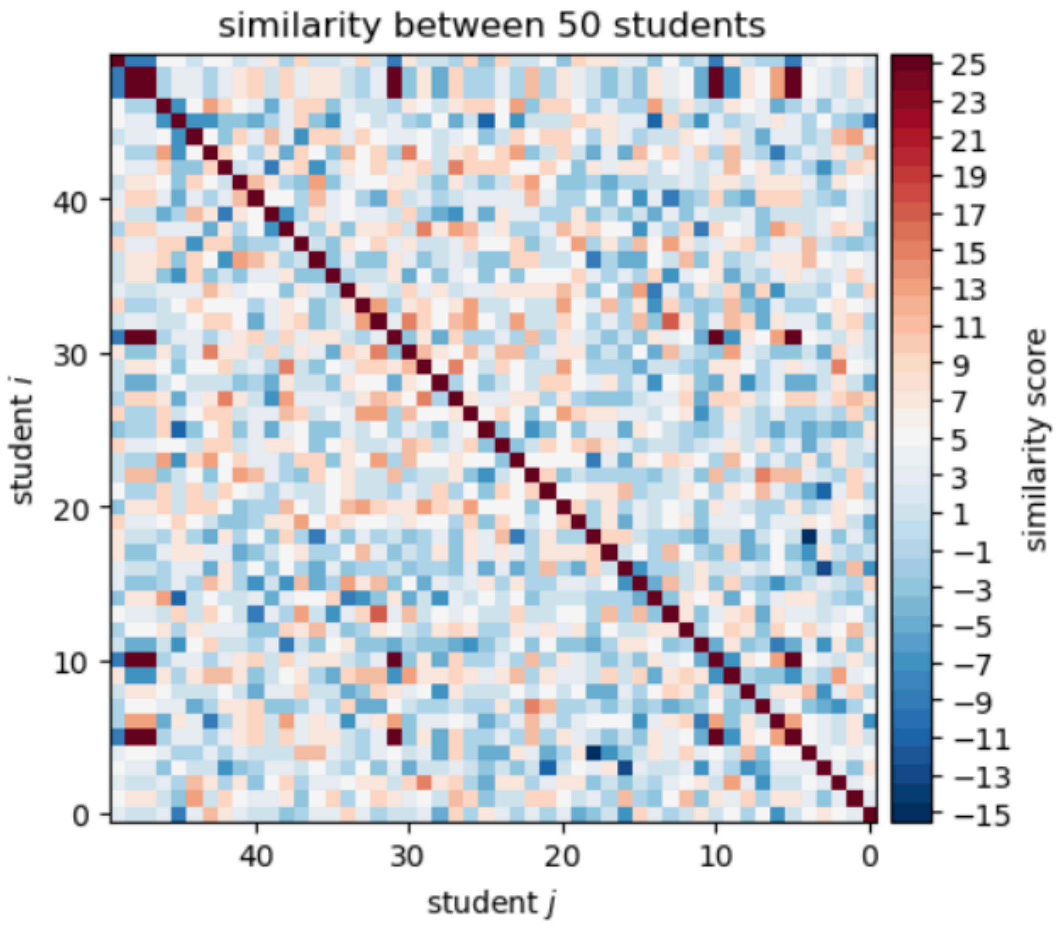

Figure 2: Similarity matrix for the small set of 50 students.

the K-means algorithm is very sensitive to how the initial K means are randomly picked. Hence, implementations typically try different initial sets of K means, and pick the 'best' grouping amongst these different initializations (in `sklearn`'s 'KMeans' implementation, the user sets the number of initializations with the `n_init` parameter).

Task 3: Creating clusters (subgroups) of students with similar answers.

You can find an implementation of this algorithm in `scikit-learn`.

Use the K-means algorithm to find groups of students with similar responses. However, as you might have realized, there is some luck and skill in picking the 'best' K.

- When K is too small, the scores amongst in-group students tend to be dissimilar.
- When K is too large, one or more groups might have very similar means. This indicates that you have over-fragmented an actual group of students.
- When K is **way too large** (e.g., $K \sim M$), each group only has one student, hence defeating the purpose of clustering.
- You should be able to group the students by their wrong/correct responses into a few **clusters**, similar to the plots below when varying $K = 3, 12, 40$. You might have to vary the K (i.e., `n_cluster` in `sklearn`).

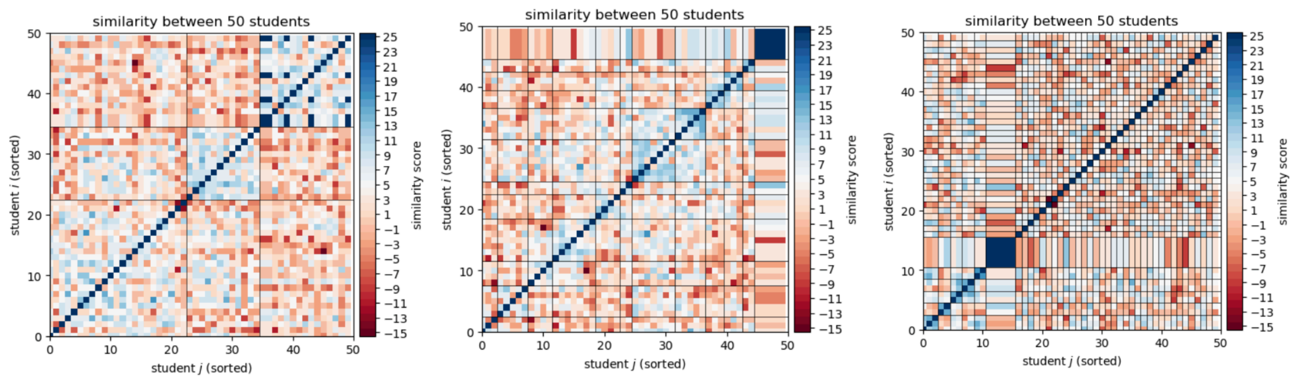

Figure 3: K-means cluster with too few (left), about right (middle), too many (right) K's.

Task 4: Give a plausibility argument (*without statistical rigor*), given your analyses up to now, about who might be colluding by copying each others' answers.

Just because two students have the same responses on a test does not mean that they have colluded. There's always a chance they will arrive at the same answers coincidentally. *Given only the data and analyses so far*, can you come up with a convincing argument about whether you can or cannot exclude the possibility that some of the students have colluded? *Hint: you might not actually get too far.*

Task 5: Assuming you have an infinite number of non-colluding students (impossible!), what is the average and standard deviation of obtaining a particular similarity score for N questions?

You might have realized that Task 4 is challenging without a sound statistical model for student collusion. After all, so far, you have only found students with similar scores, but not a causal proof of *how they came to have the same scores.*

One approach in statistics is instead to consider the seemingly contrary scenario, known as a *null hypothesis*, that none of the students colluded! Then, we determine in this null hypothesis the likelihood that two (or more students) would have scores that appear to be very similar (or identical) even though they did not collude. If two actual students' scores in an actual setting do appear this similar *despite a very low likelihood*, then it is likely they had, in fact, colluded.

Given that the average probability of getting the correct answer on any of the N questions in a test is p, where $0 < p < 1$, **derive a mathematical expression in terms of p for the expected average $\langle S \rangle$ and standard deviation σ_N in the score similarity S between two students who independently did the test.** The following might be definitions might be helpful:

$$\langle S \rangle = \sum_{s}^{\text{all cases}} \Pr(S = s)\, s \ , \tag{2}$$

s.t. $\Pr(S = s)$ is the probability for a particular score s,

$$\sigma^2 = \langle S^2 \rangle - \langle S \rangle^2 \ . \tag{3}$$

You can assume (somewhat incorrectly) that a student's response to one question is independent of the student's responses to other questions. This is equivalent to modeling a student's scores as a random sequence of -1s (with probability $1-p$) and +1s (with probability p).

Task 6: Use Monte Carlo to estimate the average and standard deviation of obtaining a particular similarity score for N questions.

A Monte Carlo simulation is a powerful computational tool for answering questions in uncertain scenarios. This method was nicknamed by physicists after the Monte Carlo Casino in Monaco. The idea behind this method is to use a computer to generate many random, probable scenarios rapidly and evaluate answers within these scenarios.

In this case, using simple random number generators in Python (e.g., `numpy.random.rand`), can you check your statistical expression in Task 5? Remember that your answer in Task 5 will depend on the number of questions N on a test, where the probability of getting a question correct is p. Here's one approach:

- Generate many students' random scores to N (say 100) questions. Here, you will should assume a particular p (say 0.5).
- Then compute the average $\langle S \rangle$ and standard deviation σ_N score similarities S between all pairs of such students. Remember, you will have to generate enough number of random student pairs to have stable averages and standard deviations.

Task 7: Given a particular p for N questions, mathematically express the probability distribution that two non-colluding students will have the same similarity score.

By assuming (somewhat crudely) that a student's response to one question is independent of the student's responses to other questions, argue that we can invoke the central limit theorem to derive an expression for this probability.

You will need to use your mathematical expressions for the average and standard deviation from Task 5 here.

Task 8: Can you identify the possibly colluding students amongst this group of 10,000 students who each took a 50-question exam?

Here are a few hints.

- The scores of these 10,000 students are found in `sample_large`.
- Using your results from tasks 1-3, find students who might be colluding.
- Then using your results from tasks 5-7, make a statistical case for whether the identified students have been colluding or not.
- Note that the difficulty of each question is not the same. Hence, to use your expression from Task 7 to establish the null hypothesis, you can estimate each question's difficulty p from all the students' average score.