# CSCI-SHU 360: Machine Learning - Homework 1
# Due: Feb 11 23:59, 2026
# 100 Points

## Instructions

- **Collaboration policy:** Homework must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and you must write and use your own code in the programming parts of the assignment. It is acceptable for you to collaborate in figuring out answers and to help each other solve the problems, and you must list the names of students you discussed this with. We will assume that, as participants in an undergraduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online via **Gradescope**. You need to submit (1) a PDF that contains the solutions to all questions to the Gradescope **HW1 PaperWork** assignment, (2) `x.py` or `x.ipynb` files for the programming questions to the Gradescope **HW1 CodeFiles** assignment. We recommend that you type the solution (e.g., using LaTeX or Word), but we will accept scanned/pictured solutions as well (clarity matters).
- **Generative AI Policy:** You are free to use any generative AI, but you are required to document the usage: which AI do you use, and what's the query. You are responsible for checking the correctness.
- **Late Policy: No late submission is allowed.**

## 1 Simpson's Paradox [10 points]

Imagine you and a friend are playing the slot machine in a casino. Having played on two separate machines for a while, you decide to swap machines between yourselves, and measure for differences in "luck". The wins/losses for you and your friend for each machine are tabulated below.

| Machine 1 | Wins | Losses |
|---|---|---|
| You | 40 | 60 |
| Friend | 30 | 70 |

| Machine 2 | Wins | Losses |
|---|---|---|
| You | 210 | 830 |
| Friend | 14 | 70 |

Assuming that the outcomes of playing the slot machine are independent of their history and that of the other machine, answer the following questions.

1. *(3 points)* Estimate the winning probability of you and your friend for each of the machines. Compare your winning probability with your friend's on different machines. Who is more likely to win?
2. *(3 points)* Estimate the overall winning probability of you and your friend in the casino (assume that there are only two slot machines in the casino). Who is more likely to win?
3. *(4 points)* Compare your conclusions from (1) and (2). Can you explain them mathematically? I.e., write down the equations to compute the winning probabilities and compare the terms.

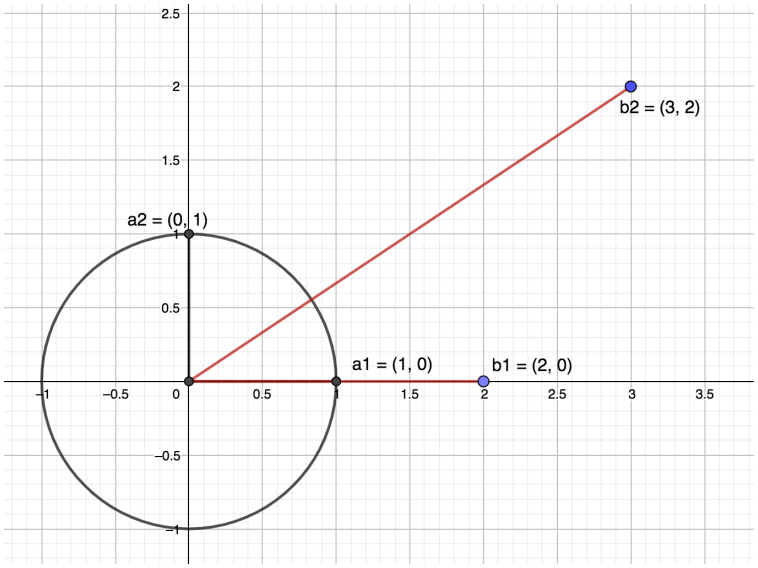


Figure 1: Problem 2

## 2 Matrix as Operations [25 points]

On the 2D plane, we have four vectors (all with the starting point on the origin) $a_1 = (1, 0)$, $a_2 = (0, 1)$, $b_1 = (2, 0)$ and $b_2 = (3, 2)$. See Fig. 1

1. *[5 points]* Write down one $2 \times 2$ matrix $W$ that transforms $a_1$ to $b_1$, and $a_2$ to $b_2$.
2. *[5 points]* Write down one rotation matrix $V$, which rotates clockwise by $\alpha$ degrees such that $tan(\alpha) = 2$. Then, write down one matrix $\Sigma$, which scales the x-axis by 4. Finally, write down one rotation matrix $U$, which rotates counter-clockwise by $\beta$ degrees, such that $tan(\beta) = 1/2$. Multiply three matrices together, namely $U\Sigma V$, what do you discover?
3. *[10 points]* Compute the eigenvalues and the corresponding eigenvectors of $W^T W$. Now consider the unit circle. Suppose every point on the unit circle gets transformed by $W$, what do you get after the transformation (consider the break-up transformations in the previous question)? What relationship do you find between the eigenvalues and the transformed shape?
4. *[5 points]* Compute the determinant of $W$. What's the area of the shape transformed from the unit circle? Can you think of some hypothesis about the relationship between the determinant and the area of a transformed shape? Based on such a hypothesis, can you use one sentence to explain that the determinant of a product of two matrices is equal to the product of the determinants of the two matrices or, in other words, $det(AB) = det(A)det(B)$?

## 3 Some Practices [10 points]

1. *[3 points]* For a discrete random variable $X$ of finite values, show that

$$(\mathbb{E}[X^3])^2 \leq \mathbb{E}[X^6]. \tag{1}$$

   You only need the materials taught in class.

2. *[3 points]* Prove the above in a different way. Again, you only need the materials taught in class.
3. *[4 points]* For two PSD matrices $A, B$ both of shape $n \times n$, show that $\lambda A + (1 - \lambda)B$ is also PSD for $0 \leq \lambda \leq 1$.

# 4 Programming Problem: Density Estimation of Multivariate Gaussian [25 points]

General instructions for programming problems (if you are not familiar with python): please install anaconda python (python version 3.x, and 3.8 or above is recommended) by following the instructions on https://www.anaconda.com/download/, and then install sklearn, numpy, matplotlib, seaborn, pandas and jupyter notebook in anaconda, for example, by running command "conda install seaborn". Note some of the above packages may have already been installed in anaconda, depending on which version of anaconda you just installed.

Please check the following tutorials if you are not familiar with numpy: http://cs231n.github.io/python-numpy-tutorial/ and https://docs.scipy.org/doc/numpy-1.15.0/user/quickstart.html

In this problem, you will estimate a multivariate Gaussian distribution from 2D points sampled a multivariate Gaussian distribution, and learn how to use python and matplotlib to generate simple plots in ipython notebook. If you are not familiar with jupyter notebook, please check a quick tutorial on https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html.

We provide an ipython notebook "2d_gaussian.ipynb" for you to complete. In your terminal, please go to the directory where this ipynb file is located, and run command "jupyter notebook". A local webpage will be automatically opened in your web browser. Click the above file to open the notebook.

You need to complete everything below each of the "TODO"s that you find (please search for every "TODO"). Once you have done that, please submit the completed ipynb file as part of your included .zip file.

In your write-up, you also need to include the plots and answers to the questions required in this session. Please include the plots and answers in the pdf file in your solution.

1. *[2 points]* Run the first cell in the ipython notebook to generate 1000 points $X$ (a $1000 \times 2$ matrix, each point has two coordinates) sampled from a multivariate Gaussian distribution. Estimate the mean and covariance of the sampled points and report them in your write-up.

2. *[3 points]* Plot the histogram for the x-coordinates of $X$ and y-coordinates of $X$ respectively. You can use the plt.hist() function from matplotlib.

3. *[5 points]* From the histogram, can you tell whether the x-coordinates of the 1000 points (the first column of $X$) follow some Gaussian distribution? If so, compute the mean and the variance. How about the y-coordinates? If so, compute the mean and the variance.

4. *[5 points]* Sample 1000 numbers from a 1D Gaussian distribution with the mean and variance of the x-coordinates you got from subproblem 3. Sample another 1000 numbers from a 1D Gaussian distribution with the mean and variance of the y-coordinates. You can use np.random.normal from numpy. Generate a new 2D scatter plot of 1000 points with the first 1000 numbers as x-coordinates and the second 1000 numbers as y-coordinates. Compared to the original 1000 points, what is the difference in their distributions? What causes the difference?

5. *[5 points]* Plot a line segment with $x = [-10, 10]$ and $y = 2x + 2$ on the 2D-Gaussian plot. The np.linspace() function may be helpful. Project $X$ onto line $y = 2x + 2$ and plot the projected points on the 2D space.

6. *[5 points]* Draw the histogram of the x-coordinates of the projected points. Are the x-coordinates of the projected points sampled from some Gaussian distribution? If so, compute the mean and the variance.

# 5 Programming Problem: Matrix/Tensor Transpose [20 points]

For computer vision (CV) tasks, the input data is typically a batch of images, which forms a tensor of shape $(B, C, H, W)$, where $B$ is the batch size, $C$ is the color channel (RGB), $H, W$ are the height and width of the image.

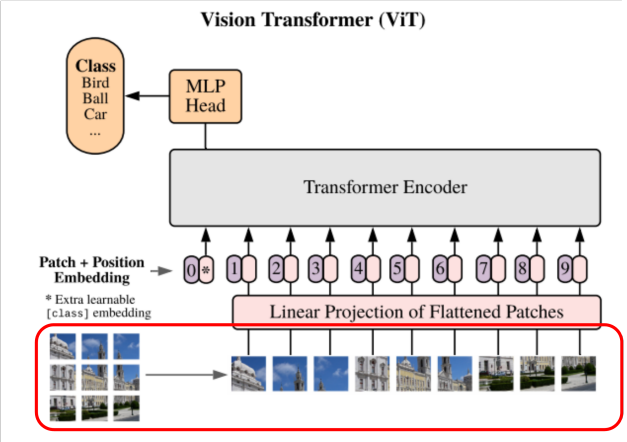


Figure 2: Patchify an Image: This figure shows the ViT model which we will cover much much later in class. For the first step, it requires cutting the input images into small patches (the red-circled part).

Modern CV models may require cutting the image into patches and treating each patch as a "visual token" for modeling. Intuitively, you may think of the modeling process as follows: each patch first processes its own information, and then shares the processed information with other patches. An illustration of such a "patchify" process can be found in Fig 2. In this problem, we will implement the "patchify" operation. *Please start from scratch with a python jupyter notebook, and all your results should be found in the output cells of the notebook.*

1. *[15 Points]* We start with numbers. Create an ordered list of numbers $(0, 1, 2, \ldots, N-1)$ and reshape it into a tensor of $B, C, H, W$ with $B = 2, C = 3, H = 4, W = 6, N = 144$. Write a python function that takes in such a tensor, and outputs the "patchified" results of shape $(B, M, C, H_p, W_p)$ where $M$ is the number of patches, and $H_p = 2, W_p = 3$ are the patch sizes. To receive the full credits, you need to use numpy.transpose or numpy.as_strided to solve this task.

2. *[5 Points]* Try your function on real images (any images you like) and show the resulting patches. In this case, you will need to first resize your image to $W = 128, H = 144$ and let the patch sizes be $W_p = 16$ and $H_p = 12$.

# 6 Programming Problem: K Means clustering [10 points]

For this question, use the data in clust_data.csv. We will attempt to cluster the data using k-means. But, what k should we use? *Please start from scratch with a python jupyter notebook, and all your results should be found in the output cells of the notebook.*

1. *[5 Points]* Apply k-means to this data 15 times, changing the number of centers from 1 to 15. Each time use nstart = 10 and store the within-cluster sum-of-squares/inertia value from the resulting object. The inertia measures how variable the observations are within a cluster. This value will be lower with more centers, no matter how many clusters there truly are naturally in the given data. Plot this value against the number of centers. Look for an "elbow", the number of centers where the improvement suddenly drops off. Based on this plot, how many clusters do you think should be used for this data?

2. *[2 Points]* Re-apply k-means for your chosen number of centers. How many observations are placed in each cluster? What is the value of inertia?

3. *[3 Points]* Visualize this data. Plot the data using the first two variables and color the points according to the k-means clustering. Based on this plot, do you think you made a good choice for the number of centers? Briefly explain your conclusion.