# Homework #5 — MTH 602

**Assigned**: Monday, October 27, 2025
**Due**: Wednesday, Nov 5, 2025 (print off and submit your report in class if possible)

Hardcopy reports should be submitted to Scott by Wednesday, Nov 5, 2025 at 5pm. If you cannot make it to class, you must find some other way to submit the report. **You can work in groups of 1, 2, or 3.**

**Report**: Put all the figures and answers asked for in the questions into a single, well-organized PDF document (prepared using LaTeX, a Jupyter notebook, or something else). **Please submit your report by uploading it to your git project and printing it off to hand in.** Ensure your submission is *reproducible*: I should be able to clone your repository and generate your reported results by using available code. Paper and pencil questions can be included in the PDF report or turned in separately

## I. PAPER & PENCIL WORK (20 POINTS)

1. Book problem 5.13
2. Book problem 5.14

## II. USING GMM AS A GEOMETRIC AND PROBABILISTIC CLASSIFIER

### A. Recap of the previous assignment (context)

In the previous homework (HW4, problem 2, stage 2) you generated and analyzed a 2-component 2D Gaussian mixture model (GMM) and sampled from the fitted GMM. We also saw how sklearn was able to use the GMM model as a classifier by calling the function

```
predicted_labels = gmm.predict(X_test)
```

This assignment continues with this classification task. You will now use a fitted *two-component* GMM as a **classifier** in two ways:

- (A) to define a simple geometric decision boundary from the two means;
- (B) to define a probabilistic classifier using the GMM posterior.

Both approaches are discussed in the book as the two main ways of handling classification.

### B. Setup and notation

You will work in $\mathbb{R}^2$ with coordinates $\vec{x} = (x_1, x_2)$. Reuse your data generator and fitting code from the previous assignment (or use the snippets below). Unless otherwise specified, use a single fixed split with a fixed random seed for all methods discussed below. This will ensure reproducibility.

```
# Provided in HW4:
def sample_2d_gaussian(mu, Sigma, n, rng=None):
    if rng is None:
        rng = np.random.default_rng()
    return rng.multivariate_normal(mean=mu, cov=Sigma, size=n)
```

### C. Data for this assignment

Use a similar (**Note**: These values are different than homework 4) ground-truth mixture:

$$w = (0.2, 0.8), \quad \mu_1 = (0, 0),\ \Sigma_1 = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1.5 \end{bmatrix}; \qquad \mu_2 = (2, 2),\ \Sigma_2 = \begin{bmatrix} 1.2 & -0.5 \\ -0.5 & 0.8 \end{bmatrix}.$$

Draw $N = 3000$ labeled samples by first sampling the component index $y \in \{1, 2\}$ with probabilities $(0.2, 0.8)$ and then sampling $\vec{x}\,|\,y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$. Split once into train (80%) and test (20%) using a fixed random seed. See homework 4 and the GMM demo notebook for how to do this.

### D. Training the Gaussian mixture model (10 POINTS TOTAL)

1. Fit the GMM with $K$=2 components on the training points.

2. GMM components are unlabeled; align them to classes using only the training set data. Use the nearest-mean rule: with fitted means $\hat{\mu}^{(0)}, \hat{\mu}^{(1)}$ and known true means $\mu_1, \mu_2$, map the pair $(\hat{\mu}^{(0)}, \hat{\mu}^{(1)})$ to $(\mu_1, \mu_2)$ by the minimum total Euclidean distance. In code:

```
M  = g2.means_
d0 = np.linalg.norm(M[0]-mu1) + np.linalg.norm(M[1]-mu2)
d1 = np.linalg.norm(M[1]-mu1) + np.linalg.norm(M[0]-mu2)
if d0 <= d1:
    class_to_component = {1: 0, 2: 1}
    component_to_class = {0: 1, 1: 2}
else:
    class_to_component = {1: 1, 2: 0}
    component_to_class = {1: 1, 0: 2}
```

   These mappings will come in handy as you write code to solve this problem. In particular, we can now refer to $\hat{\mu}_1$ as the mean for the Gaussian describing class 1.

3. Verify the correctness of your GMM model and the class-to-component mapping. Provide strong evidence that everything is working. You should decide what evidence to provide, but it should convince any reasonable reader beyond any doubt that you have a working GMM model with the correct labels.

### E. Geometric classification (20 POINTS TOTAL)

Let $\hat{\mu}_1, \hat{\mu}_2 \in \mathbb{R}^2$ be the two fitted means after class label alignment (so $\hat{\mu}_1$ is the mean for the Gaussian describing class 1). Define

$$\hat{r} = \frac{\hat{\mu}_2 - \hat{\mu}_1}{\|\hat{\mu}_2 - \hat{\mu}_1\|_2}, \qquad m = \frac{\hat{\mu}_1 + \hat{\mu}_2}{2}.$$

Here, $\hat{r}$ is the unit direction vector pointing from $\hat{\mu}_1$ to $\hat{\mu}_2$ and $m$ is the midpoint between the means. Classify a point $x$ by which side of the perpendicular bisector it falls on:

$$\widehat{y}_{\text{geom}}(x) = \begin{cases} 1, & \langle x - m, \hat{r} \rangle < 0 \\ 2, & \langle x - m, \hat{r} \rangle \geq 0 \end{cases}.$$

The decision boundary you constructed here is called a perpendicular-bisector decision boundary.

**Tasks:**

1. Plot the geometric decision boundary together with the *training* points colored by their true class. This is a visual check that the decision boundary is reasonable.

2. Plot the geometric decision boundary together with the *testing* points colored by their true class. This is the result of using the classifier. This should also show misclassified data as being on the wrong side of the decision boundary.

3. Compute and report the confusion matrix for the testing data.

4. (**Optional**) Explore how different values of $(\hat{r}, m)$ impact the decision boundary. Plot a family of decision boundaries.

5. (**Optional**) Show that the perpendicular-bisector can be written in the book's linear-discriminant form $g(x) = w^\top x + w_0$ by $w = \hat{\mu}_2 - \hat{\mu}_1$ and $w_0 = -\frac{1}{2}(\hat{\mu}_2 + \hat{\mu}_1)^\top(\hat{\mu}_2 - \hat{\mu}_1)$.

### F. Probabilistic GMM classifier with thresholding & ROC (20 TOTAL POINTS)

Your GMM model can also be used as a probabilistic classifier. To see this, we first recall from the book (See Eq 3.115 and the first equality of 5.40) the following formula: For a $K$=2 component GMM with parameters $(\hat{\pi}_k, \hat{\mu}_k, \hat{\Sigma}_k)_{k=1}^2$, for any $x$,

$$P(k \mid x) \;=\; \frac{\hat{\pi}_k \mathcal{N}(x \mid \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{\ell=1}^2 \hat{\pi}_\ell \mathcal{N}(x \mid \hat{\mu}_\ell, \hat{\Sigma}_\ell)}\,.$$

$P(k \mid x)$ is the probability of the data $x$ belonging to class $k$. It can be computed directly from sklearn's GMM function:

```
comp_idx_for_class2 = class_to_comp[2]
proba_class2 = g2.predict_proba(X_test)[:, comp_idx_for_class2]
```

$P(2 \mid x)$ is the class-2 probability and, for a threshold $\tau \in [0, 1]$,

$$\widehat{y}_{\text{prob}}(x;\tau) = \begin{cases} 2, & P(2 \mid x) \geq \tau \\ 1, & P(2 \mid x) < \tau\,. \end{cases}$$

1. Compute the confusion matrix over the testing dataset using a reasonable threshold of $\tau = 0.5$.

2. Consider a range of thresholds $\tau$ from 0 to 1 and summarize the performance by plotting an ROC curve on the test dataset. Also report the AUC.

### G. Comparison: Probabilistic vs geometric classifier (10 TOTAL POINTS)

In a few sentences.... (i) compare the geometric and probabilistic classifiers in terms of performance, (ii) does this dataset constitute an easy or hard classification task, and (iii) is class-1 or class-2 more difficult to predict and intuitively why do you think so?

## III. PARTICLE PHYSICS: CLASSIFICATION OF A PARENT PARTICLE FROM TWO-BODY DECAY KINEMATICS (20 POINTS + BONUS)

**Grading.** Any reasonable attempt earns 20 points.

**High-level background information.** In high-energy particle physics, unstable "parent" particles are created in collisions (at places like Fermilab or CERN) and promptly decay into lighter "daughter" particles whose energies and momenta are measured by detectors. Because the parent often cannot be observed directly, we infer its identity from the daughters' kinematics using conservation of energy-momentum. In this challenge, each event contains two massless daughters (e.g., photons), and the unknown parent is one of two possible parent particles with different masses.

Its not expected that any student will have worked in particle physics or has detailed knowledge of this domain. Rather, you can approach this as a machine learning problem. You have raw data (energy and momentum of the

daughter particles) and you need to classify the parent particle. Students with a physics background may wish to construct "physics-informed" features such as invariant mass, transverse momentum, and opening angle, etc... and using these features as the raw data upon which the classifier is trained.

**Problem statement.** An unknown parent particle is either particle **A** or particle **B**. These particles have distinct masses.

Your dataset records instances of particle decay. Each particle-decay event has two massless daughters (e.g., photons). From the daughters' energy and momentum (for physics students, these are technically four-vectors), classify the parent particle as $A$ or $B$.

- `hw5_train.csv`: Each column is a recorded decay event. The first daughter particle has an energy, $E_1$, and momentum, $(p_{x1}, p_{y1}, p_{z1})$. The second daughter particle has an energy, $E_2$, and momentum, $(p_{x2}, p_{y2}, p_{z2})$. Therefore, each column's data is `E1,px1,py1,pz1,E2,px2,py2,pz2,label` (label $\in$ {A,B}).
- `hw5_test.csv`: same kinematics, no labels.
- The data is recorded in the "lab frame". In the rest frame of the parent particle, the decay is isotropic.

**Suggested line of attack.** As with all challenge problems, you should tackle this as you see fit and according to your skills. If you have more of an ML background, you might focus on building sophisticated classifiers. Presumably probabilistic ones that work in a non-linear feature space will be best. If you have a physics background, you might attempt to construct physics-inspired features such as the invariant mass, for example.

**Scoring for the bonus challenge.** For the test set, submit two numbers: your predicted counts of parent types A and B: `N_A_pred, N_B_pred`.

If you have a probabilistic classifer, you can either convert to a specific class using a threshold or compute so-called "soft" counts from the posteriors: $N\_B\_\text{pred} = \sum_{j=1}^{N_\text{test}} p(B \mid x_j)$ and $N\_A\_\text{pred} = N_\text{test} - N\_B\_\text{pred}$.

Submissions are scored by absolute count error

$$\text{Score} = \left|N\_A\_\text{pred} - N_A^\text{true}\right| + \left|N\_B\_\text{pred} - N_B^\text{true}\right|,$$

so lower is better. And of course $N\_A\_\text{pred} + N\_B\_\text{pred} = N_\text{test}$.