Homework #4 — MTH 602

Assigned: Monday, October 13, 2025
Due: Monday, October 27, 2025 (print off and submit your report in class if possible)

Hardcopy reports should be submitted to Scott by Wednesday, October 27 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 (10 POINTS)

1. (i) Show that a positive definite matrix of size $D \times D$ has $D(D+1)/2$ independent parameters. (ii) How many independent parameters are in a D-dimensional Gaussian? (Hint: if $D = 1$ there are 2 parameters.) (iii) How many independent parameters are in a Gaussian mixture model with K components if each component is a D-dimensional Gaussian?

II. CHAPTER 3: MULTIVARIATE GAUSSIAN MIXTURE MODELING (35 POINTS)

Problem context: Probability mixture modeling is used throughout science and engineering. For example, see a multi-component Gaussian mixture model for pulsar classification (https://arxiv.org/pdf/1205.6221, Fig. 2). In this assignment, the you will proceed from a 1-component to a 2-component mixture model, using diagnostics to check model correctness. In class the basics of 2D Gaussians and mixture models were outlined. To complete this assignment it may be helpful to consult the scikit-learn documentation on fitting GMMs from samples (https://scikit-learn.org/stable/modules/mixture.html).

The exercises use 2D data to explore Gaussian Mixture Models (GMMs). Hints given throughout this assignment will assume Python with **scikit-learn** and the following modules:

```
from sklearn.mixture import GaussianMixture
from scipy.special import logsumexp
import numpy as np
```

Use the following function to generate ground-truth samples in Stages 1 and 2.

```
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)
```

A. Stage 1 — One Gaussian: MLE vs. GMM with 1 component

Let the ground-truth Gaussian be

$$\mu = (2, -1), \quad \Sigma = \begin{bmatrix} 2 & 0.8 \\ 0.8 & 1 \end{bmatrix},$$

and write $\vec{x} = (x_1, x_2)$. Draw $N = 1000$ samples $\vec{x} \sim \mathcal{N}(\mu, \Sigma)$ using `sample_2d_gaussian`, then:

1. **Visualization:** Make a scatter plot of the samples. (Optional: overlay the true 1σ ellipse of Σ.)
2. **Analytic maximum-likelihood solution:** From Bishop, the MLEs are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})(x_i - \hat{\mu})^\top.$$

   Compute these, report their values, and compare to the ground truth.
3. **GMM with one component:** Fit a one-component GMM:

```
g1 = GaussianMixture(n_components=1, covariance_type='full',
                     n_init=10, random_state=0).fit(X)
```

   Extract μ_{gmm} and Σ_{gmm}. Then compare to the maximum-likelihood solution using

$$\|\hat{\mu} - \mu_{\text{gmm}}\|_2 \text{ (vector } L^2 \text{ norm)}, \qquad \|\hat{\Sigma} - \Sigma_{\text{gmm}}\|_2 \text{ (matrix 2-norm)}.$$

   Agreement should be very close.
4. **Sampling from the fitted model:** Use `g1.sample(M)` to draw new points; overlay them on the original scatter plot as color-coded points so as to distinguish them from the original 1000 samples.

STAGE 2 — TWO-COMPONENT MIXTURE IN 2D: FIT, COMPARE, SAMPLE, CLASSIFY

Let the ground-truth mixture be

$$w = (0.2, 0.8), \quad \mu_1 = (0,0),\ \Sigma_1 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1.5 \end{bmatrix}; \qquad \mu_2 = (4,3),\ \Sigma_2 = \begin{bmatrix} 1.2 & -0.5 \\ -0.5 & 0.8 \end{bmatrix},$$

with $\vec{x} = (x_1, x_2)$. Motivated by pulsar classification (Fig. 2 of the linked paper above), assume pulsars of "type 1" are drawn from $\mathcal{N}(\vec{x}\,|\,\mu_1, \Sigma_1)$ and "type 2" from $\mathcal{N}(\vec{x}\,|\,\mu_2, \Sigma_2)$. To make the connection to pulars more explicit, you can view $x_1 = P$ (period) and $x_2 = \dot{P}$ (spin-down). The observed pulsar population is thus

$$P(\vec{x}) = 0.2\,\mathcal{N}(\vec{x}\,|\,\mu_1, \Sigma_1) + 0.8\,\mathcal{N}(\vec{x}\,|\,\mu_2, \Sigma_2),$$

so "type 2" occurs 80% of the time.

Draw $N = 1000$ samples by first sampling the component index (probabilities 0.2 and 0.8), then sampling from the corresponding Gaussian via the provided function.

1. **Visualization**: Scatter plot the data. For plotting, color each point by its *true* component label (e.g., red for type 1, black for type 2).
2. **Modeling**: Fit GMMs with $K = 1$ and $K = 2$. Also report diagnostics to decide which model is better (e.g., log-likelihood, Bayesian information criterion, or qualitative sampling comparisons). For example you could do:

```
logL1, bic1 = g1.score(X)*len(X), g1.bic(X)
logL2, bic2 = g2.score(X)*len(X), g2.bic(X)
```

   Which model is preferred?
3. **Comparisons**: For $K = 2$, report learned weights $\hat{w}_k$, means $\hat{\mu}_k$, and covariances $\hat{\Sigma}_k$:

```
means2, covs2, weights2 = g2.means_, g2.covariances_, g2.weights_
```

   Compare these to the true values.

4. **Sampling from the fitted model**: Use `g2.sample(M)` to sample M times from the GMM. And overlay these new samples on the true data samples. Comment on coverage of both modes.

5. **Classification**: Now construct your own ground truth data by gathering X_{data} by sampling from $P(\vec{x})$. Make sure to store the pulsar type labels. Now, test the GMM model by predicting the pulsar classes:

```
predicted_labels = g2.predict(X_data)
```

Plot points colored by predicted labels as a visual check. Compute the classifier's accuracy against the true label and report the confusion matrix.

III. CHAPTER 4: POLYNOMIAL REGRESSION WITH REGULARIZATION (35 POINTS)

Context. In Homework 1 you fit noisy data $\{(t_i, y_i)\}_{i=1}^{N}$ with polynomial least squares, examined conditioning, and studied train/test error vs. degree M. In this assignment you will revisit the same task with *ridge* (ℓ_2) regularization and perform model selection. This corresponds to "regularized least squares" in Section 4.1.6 of the book.

Your testing and training dataset will be generated using a slightly modified version of homework 2's dataset:

```
def generate_data(scale_noise, N, seed=None):
    """ Generate a noisy time series dataset {t_i, y_i}_{i=1}^N.

    Parameters
    ----------
    A : float
        Amplitude of the signal (A >= 0)
    N : int
        Number of data points. Time points uniformly sampled from [0, 2pi]
    seed : int or None, optional
        Random seed for reproducibility """

    rng = np.random.default_rng(seed)
    t = np.linspace(0.0, 1.0, N)
    dt = (2*np.pi) / (N - 1)

    A=np.sqrt(np.pi/dt)
    s_hat = np.sqrt(dt / np.pi) * np.sin(2*np.pi*t)   # ||s_hat||_2 == 1 exactly on this grid
    s = A * s_hat
    n = rng.normal(loc=0.0, scale=scale_noise, size=N)
    y = s + n
    return t, y, s, n, s_hat, dt

# generate testing and training data
x_data,y_data,s,n,s_hat, dt = generate_data(.1,100)

t_train = x_data[::10]
y_train = y_data[::10]
s_train = s[::10]
plt.plot(t_train,y_train,'bo')
plt.plot(t_train,s_train,'r--')
plt.plot(x_data,s,'k--')

# generate testing data
# TODO for YOU: Write code to remove the training data below
# t_test and y_test will have 90 data points
t_test =
y_test =
```

Let $A \in \mathbb{R}^{N\times(M+1)}$ be your polynomial design (Vandermonde) matrix from HW 1 with columns $[1, t, t^2, \dots, t^M]$. Denote $\omega \in \mathbb{R}^{M+1}$ the coefficients. Consider the ridge optimization problem

$$\min_{\omega\in\mathbb{R}^{M+1}} \; \|A\omega - y\|_2^2 \; + \; \lambda \, \|\Gamma\omega\|_2^2, \qquad \lambda \geq 0.$$

If $\Gamma = \mathrm{diag}(1, 1, \dots, 1)$ is the identity, this matches Eq. (4.25) from the Bishop book. Often it is desirable to leave the constant term unpenalized, in which case $\Gamma = \mathrm{diag}(0, 1, 1, \dots, 1)$. In either case, the corresponding normal equations are

$$\hat{\omega}_\lambda \; = \; (A^\top A + \lambda \, \Gamma^\top \Gamma)^{-1} A^\top y,$$

which can be compared to Eq. (4.27) from the Bishop book.

In what follows, use $\Gamma = \mathrm{diag}(0, 1, 1, \dots, 1)$.

1. Ridge implementation, diagnostics, and interpretation

1. **Implementation & comparison.** Consider all polynomial orders $M \in \{1, 2, \dots, 9\}$ and regularization factors $\lambda \in \{0, 10^{-6}, 10^{-4}, 10^{-2}, 10^0, 10^2\}$. There are $9 \times 6 = 54$ (M, λ) combinations. For each:
   - Use the training and testing data as shown in the code block above. You should have 10 samples in your training set and 90 samples in your testing set. Use the same testing/training data for *all* (M, λ) so results are comparable.
   - Compute $\hat{\omega}_\lambda$ by solving the linear system
   - Report train and test MSE, $\kappa(A^\top A)$, and $\kappa\big(A^\top A + \lambda \, \Gamma^\top \Gamma\big)$ (2-norm condition numbers). Briefly discuss how ridge regularization affects conditioning.
   - For each fixed M, plot test MSE vs. λ (use a log λ axis).
   - Provide evidence (table or plot) showing how the condition number varies with λ.
2. **Equivalence check with `scikit-learn`.** Fit your data with `Ridge(alpha=`λ`, fit_intercept=False)`, `solver='auto'` on the same A matrix (intercept already included in A). Verify that the coefficients match your closed form solution (now penalize all coefficients with $\Gamma = \mathrm{diag}(1, 1, \dots, 1)$) for the same (M, λ) and unpenalized intercept.
3. **Bias–variance view (qualitative).** Using your results, concisely explain how increasing λ increases bias and reduces variance, and why the optimal $\lambda^\star$ differs across degrees M.

2. Model selection over polynomial degree and regularization

1. **Grid search over (M, λ).** Identify the value $(\hat{M}, \hat{\lambda})$ that minimizes test MSE. Report the test MSE at $(\hat{M}, \hat{\lambda})$.
2. **Final reflections** Consider three different fits: (i) the best one $(\hat{M}, \hat{\lambda})$, (ii) a poorly conditioned case $(M, \lambda) = (7, 0)$, (iii) an overly penalized case $(M, \lambda) = (7, 10^0)$.
   - Plot all three fitted curves on the same figure along with the *true (noise-free) signal.*
   - Print the coefficient vectors for all three fits.
   - Comment on how the regularization parameter λ affects the coefficients (magnitude and pattern) and how this manifests in the plotted fits. You may wish to revisit the discussion in Chapter 1 and Chapter 4's discussion on bias–variance.

IV. GMM & REGRESSION FOR MARINE CRAFT DYNAMICS (20 POINTS + BONUS)

Grading. Any reasonable attempt earns 20 points. The best *forecasting* model earns +10 bonus points, and the best *GMM noise model* earns +10 bonus points).

Context. You observe a single time series $y(t)$, vertical acceleration, from a marine planing craft operating in (approximately) regular waves. Beyond forecasting $y(t)$, we care about what the *noise tells us about the waves.* In particular, impulsive, non-Gaussian disturbances (e.g., slamming) carry information about sea state and operating regime. By explicitly modeling the noise as a *Gaussian mixture* (GMM), you will both (i) improve robustness of your signal estimate and (ii) provide a scientifically meaningful description of the disturbance environment. The *values* of the GMM parameters are of intrinsic interest: they summarize wave-related variability in the measurement channel.

Assume

$$y(t_i) \;=\; s(t_i;\theta) \;+\; \varepsilon_i, \qquad i = 1,\ldots,N,$$

where

- $y(t_i)$ is the measured signal at time t_i,
- $s(t;\theta)$ is a *smooth/structured* underlying signal you will model via regression in time,
- ε_i are i.i.d. disturbances with $\mathbb{E}[\varepsilon_i] = 0$ that can be reasonably modeled by a 1D Gaussian mixture (GMM) whose parameters you need to find.

The dataset `hw4_train.csv`= $\{(t_i, y_i)\}_{i=1}^{N}$ contains the necessary data.

Your goals are twofold: (A) build a good predictor $\hat{y}(t)$, and (B) learn a plausible GMM model for the noise that *explains the residual structure* and thus is informative about the environmental waves. Any reasonable approach is permitted.

Practical tip (normalization/standardization): Consider normalizing your time before building polynomial features: map t linearly to $[-1, 1]$ using the *training* sets $t_{\min}, t_{\max}$ and use the normalized test times. Build the Vandermonde on this normalized t. This greatly improves conditioning of $A^\top A$ and makes λ choices more stable. The data y can remain in its original form; but if you do standardize y, remember to unscale predictions before computing computing the challenge submission data below.

What to submit:

1. In your solution document, present your work approaching this problem. Describe what you believe to be the best predictor model $\hat{y}(t)$ and GMM noise model you've built. Include any tests you've done to provide convincing evidence that these models are trustworthy.

2. A brief (200–300 words) discussion: what were basis/features you've used for the time regression, any regularization was used (how and why), how you estimated/selected the GMM, and how you tuned hyperparameters (e.g., basis choice, regularization, etc...).

A. Challenge submission I: Using the predictive model

We will rank submissions by how well they reproduce the *Vibration Dose Value* (VDV) of the test acceleration signal. VDV emphasizes impulsive forces on the boat and is defined as

$$\text{VDV} \;=\; \left(\int_{t_0}^{t_1} |a(t)|^4 \, dt\right)^{1/4}.$$

Smaller values of VDV means the boat's ride is smoother.

Given the dataset's timestamps $\{t_j\}_{j=1}^{N}$ and your predicted accelerations $\hat{a}_j = \hat{y}(t_j)$, compute the discrete VDV using some numerical integration algorithm. A simple example for a uniform with step Δt is $\text{VDV}_{\text{pred}} = \left(\Delta t \sum_{j=1}^{N} |\hat{a}_j|^4\right)^{1/4}$

I will compute the ground-truth value VDV_{true} from the hidden test accelerations $a_j = y(t_j)$. Your submitted value will be compared to the ground-truth value.

B. Challenge submission II: Using the GMM model

Beyond forecasting, we are interested in what the residual noise says about the sea state as quantified through the Gaussian mixture model you've already built.

Define the *background sea state* as the Gaussian component with the **smallest variance**. Let its standard deviation be

$$\sigma_{\text{bg,pred}} = \min_k \sigma_k \quad \text{(over your GMM components).}$$

Your value of $\sigma_{\text{bg,pred}}$ will be compared to the ground-truth value $\sigma_{\text{bg,true}}$.