Homework #2 — MTH 602

Assigned: Monday, September 22, 2025
Due: Wednesday, October 1, 2025 (print off and submit your report in class if possible)

Hardcopy reports should be submitted to Scott by Wednesday, October 1 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. FLOATING-POINT NUMBERS

1. (15 points) Consider an N-term Taylor series expansion of the exponential function

$$e^x \approx \sum_{n=0}^{N-1} \frac{x^n}{n!} = \hat{f}_N(x),$$

and the corresponding Python code:

```python
def taylor_exp(N, x):
    """
    Compute the N-term Taylor series expansion of e^x about 0, evaluated at x.

    Parameters
    ----------
    N : int
        Number of terms in the Taylor expansion.
    x : float
        Point at which to evaluate the expansion.

    Returns
    -------
    float
        Approximation of e^x using the first N terms of the series.
    """
    import math

    total = 0.0
    for n in range(N):
        total += (x**n) / math.factorial(n)
    return total
```

Now consider using this Python function to compute e^x at $x = -20$.

(a) (5 points) Plot the relative error

$$E_N = \frac{\left|\hat{f}_N(-20) - e^{-20}\right|}{|e^{-20}|}$$

as a function of N for $N = 1, 2, \ldots$. What is the smallest relative error you can achieve? Make the plot using a semilog-y scale.

(b) (5 points) You should find the relative error is much larger than the expected value of $\approx 10^{-16}$. Briefly explain the numerical reason for this behavior (hint: think about cancellation when x is negative and large in magnitude).

(c) (5 points) Without changing the function `taylor_exp(N,x)`, can you propose a way to accurately compute e^{-20}? If so, demonstrate the improvement by making another convergence plot as in part (a), and explain why your new method works.

(d) (**Optional**) In what sense are floating-point roundoff error and machine learning generalization error examples of "error accumulation," and how are they fundamentally different?

2. (15 points) Suppose you are going to compute a matrix–vector multiplication $\vec{y} = A\vec{x}$, where

$$A = \begin{bmatrix} 10^3 & 0 \\ 0 & 10^{-2} \end{bmatrix}.$$

Suppose further that $\vec{x}$ itself arises from previous numerical computations. In particular, each component of $\vec{x}$ comes from N floating-point operations (say, all multiplications), and the floating-point number system you are using has a machine epsilon ϵ_m.

(a) (5 points) Let $\delta\vec{x}$ be the error due to floating-point arithmetic. What is a reasonable estimate for the relative error

$$\frac{\|\delta\vec{x}\|_2}{\|\vec{x}\|_2}\,?$$

(b) (5 points) What is a reasonable estimate for the relative error in the computed $\vec{y}$ due to floating-point error? That is, estimate

$$\frac{\|\delta\vec{y}\|_2}{\|\vec{y}\|_2},$$

expressed in terms of N, ϵ_m, and the condition number of the matrix A.

(c) (5 points) Based on your estimate, what is the relative error in $\vec{y}$ when using single precision (e.g. on a GPU) if $N = 10^3$ floating-point operations are used?

3. (**Optional**) Deep neural networks involve thousands of sequential floating-point operations. How might the relative error analysis above change if $\vec{x}$ represents an activation vector in a deep network layer? Could single precision still be adequate?

II. PROBABILITY (CHAPTER 2)

1. (10 points) Verify that the uniform distribution

$$p(x) = \frac{1}{d-c}, \qquad x \in [c, d]$$

is correctly normalized. Find an expression for its expected value and variance.

2. (10 points) Show that expectation values are linear:

$$\mathbb{E}[\alpha f(x) + \beta g(x)] = \alpha\mathbb{E}[f(x)] + \beta\mathbb{E}[g(x)].$$

3. (**Optional**) A uniform prior distribution is often chosen in Bayesian inference when no prior knowledge is available, since it assigns equal belief to all values in an interval. If you were estimating a parameter θ with prior $p(\theta) = \text{Uniform}(c, d)$, how would the expected value and variance influence your posterior estimates after observing a very small amount of data?

4. (20 points) The book (https://www.bishopbook.com/) discusses the important concept of maximum likelihood estimation with an application to linear regression. Prove the fundamental results of this discussion by solving the following book problems:

   - Problem 2.16
   - Problem 2.18

III. IS THERE A SIGNAL IN MY TIME-SERIES DATA? APPLICATION OF PROBABILITY THEORY

Problem context: This matched filtering setup is known as Wiener filtering. The main idea is to define a detection statistic (called the signal-to-noise ratio) that can be used to classify the data as "signal" or "no signal". In a certain sense, it is the optimal linear classifier under the assumption of Gaussian noise with a known template. It is used throughout science, and forms the basis of gravitational-wave detection, for example. We will consider non-linear classifiers later in the course.

Consider a noisy time series dataset $\{t_i, y_i\}_{i=1}^N$ with

$$y_i = y(t_i) = s(t_i) + n(t_i) = s_i + n_i,$$

where the noise samples are i.i.d. Gaussian $n_i \sim \mathcal{N}(0,1)$. We observe on $t \in [0, 2\pi]$ using a uniform grid

$$t_i = \frac{2\pi(i-1)}{N-1}, \quad i = 1, \ldots, N, \qquad \Delta t \equiv \frac{2\pi}{N-1}.$$

As in the book's Section 2.3.2 (eq. 2.55), the noise probability density factorizes as

$$p(\mathbf{n} \mid 0, 1) = \prod_{i=1}^{N} \mathcal{N}(n_i \mid 0, 1).$$

The signal amplitude A could be large (e.g. $A = 10$), weak ($A = 3$), or even non-existent ($A = 0$). Here is Python code that you can use to generate mock datasets:

```
import numpy as np

def generate_data(A, 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, 2*np.pi, N)
    dt = (2*np.pi) / (N - 1)

    s_hat = np.sqrt(dt / np.pi) * np.sin(t)   # ||s_hat||_2 == 1 exactly on this grid
    s = A * s_hat
    n = rng.normal(loc=0.0, scale=1.0, size=N)
    y = s + n
    return t, y, s, n, s_hat, dt
```

We use the ordinary inner product between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^N$

$$\langle \mathbf{a}, \mathbf{b} \rangle \;=\; \sum_{i=1}^{N} a_i \, b_i,$$

and define a discrete (normalized) template $\hat{\mathbf{s}} \in \mathbb{R}^N$ by

$$\hat{s}_i \;=\; \sqrt{\frac{\Delta t}{\pi}} \; \sin(t_i) \; ,$$

for which $\|\hat{\mathbf{s}}\|_2^2 = \sum_{i=1}^{N} \hat{s}_i^2 = 1$ holds exactly on this grid, since

$$\sum_{i=1}^{N} \sin^2(t_i) \;=\; \frac{N-1}{2} \;=\; \frac{\pi}{\Delta t}.$$

We parameterize the signal amplitude by $A \geq 0$ via

$$\mathbf{s} \;=\; A \, \hat{\mathbf{s}}.$$

Define the detection statistic (the matched-filter signal-to-noise ratio) as

$$\rho(A) \;=\; \langle \mathbf{y}, \hat{\mathbf{s}} \rangle \;=\; \sum_{i=1}^{N} y_i \, \hat{s}_i \; .$$

Because $\mathbf{y}$ is a random variable (through $\mathbf{n}$), ρ is a random variable. It may also be helpful to view $\rho(A)$ as a function of the signal's amplitude A. We claim a signal is present if $\rho \geq \rho_0$, where ρ_0 is a chosen detection threshold.

- (10 points) Find the expectation value and variance of $\rho(A)$. Your formulas should be in terms of the signal amplitude A.
- (10 points) Set $N = 1000$. Plot the empirical distribution of $\rho(A)$ for $A \in \{0, 3, 10\}$. That is, for each A, generate many data realizations by calling `generate_data` and compute $\rho(A)$ for each one. Overlay each histogram with the corresponding theoretical Gaussian distribution using the mean and variance you derived above (label the curves clearly). Use these plots to confirm your formulas.
- (10 points) Now consider two possibilities: no signal ($A = 0$) or a signal ($A > 0$). From the previous question, observe that for large enough A, the $\rho(A = 0)$ distribution is well separated from the $\rho(A \neq 0)$ distribution. For example, if $A = 10$, setting $\rho_0 = 5$ should separate the no-signal and signal classes nearly perfectly. But for smaller values of A the distributions of $\rho(A = 0)$ and $\rho(A)$ overlap, leading to possible misclassification. There are four outcomes:

  1. True positive: You declare "signal present," and indeed a signal really was present in the data.
  2. True negative: You declare "no signal," and indeed there was no signal in the data (only noise).
  3. False positive: You declare "signal present," but in reality there was no signal (just noise mimicking a signal-like feature).
  4. False negative: You declare "no signal," but there actually was a real signal present.

  For the $A = 3$ case, empirically choose a value of ρ_0 that roughly balances the number of false positives and false negatives for 1000 mock datasets. Present your results by constructing a confusion matrix.

A. Class Competition (bonus +10 points)

A long data stream was recorded and stored in a file called `hw2.csv`. There are multiple signals in this data stream with amplitudes $A \in (0, 10)$. Your challenge is to provide an estimate for the number of signals. The group whose estimate is closest to the true value will earn 10 bonus points!