Homework #3, EECS 553, F26. Due **Wed. Sep. 30**, by 11:00PM

Problems

1. [6] **Bayes classifier and Bayes risk**

Consider a 3-class classification problem with (class conditional) **continuous uniform distributions**:

$$\pi_1 = 1/2, \quad \mathsf{p}(x \mid \mathsf{Y} = 1) = U[1,3]$$
$$\pi_2 = 3/8, \quad \mathsf{p}(x \mid \mathsf{Y} = 2) = U[2,5]$$
$$\pi_3 = 1/8, \quad \mathsf{p}(x \mid \mathsf{Y} = 3) = U[4,6].$$

(a) [3] Determine the **Bayes classifier** for this setting. Hint. It is a **piecewise function**.

(b) [3] Determine the **Bayes risk** (for the 0-1 loss). Hint. Your value should be less than 1/4.

2. [22] **Bayes classifier for Poisson data**

Consider a binary classification problem with $\mathcal{Y} = \{1, 2\}$ and $d = 1$. For class 1, the (conditional) distribution of X is $\mathsf{Poisson}\{\mu_1\}$ and for class 2, the (conditional) distribution of X is $\mathsf{Poisson}\{\mu_2\}$ where $0 < \mu_1 < \mu_2$. The prior class probabilities are π_1 and π_2, both of which are nonzero.

(a) [3] Determine the Bayes classifier (for the 0-1 loss) for this problem.

Hints. Taking the logarithm of $\pi_k g_k$ helps simplify. The solution has the form $f_*(x) = \begin{cases} 2, & x > x_* \\ 1, & \text{otherwise}, \end{cases}$ for some (possibly non-integer) value x_* that depends on the problem parameters.

(b) [3] Generalize to the $d > 1$ case where instead of the scalar X, you are given a vector $\mathbf{X}$ where each element of the vector is an independent realization drawn from $\mathsf{Poisson}\{\mu_k\}$.
Hint. The solution is fairly similar to the previous part, involving a single threshold.

(c) [10] Using your solution to (b), write a JULIA or Python function that performs Bayes classification given input data $\boldsymbol{x}$ and (known) model parameters $\mu_1, \mu_2, \pi_1, \pi_2$.
Your submitted file should be named `bayes_poisson.jl` or `bayes_poisson.py` and should contain the following function:

```
"""
    yh = bayes_poisson(x, μ1, μ2, π1, π2)

In:
- `x::Vector` vector of `d` elements drawn from Poisson(μk) distribution
- `μ1` positive mean for class 1
- `μ2` positive mean for class 2 with `μ2 > μ1`
- `π1` prior probability of class 1
- `π2` prior probability of class 2
Out:
- `yh` Bayes classification (1 or 2)
"""
function bayes_poisson(x::Vector, μ1::Real, μ2::Real, π1::Real, π2::Real)
```

Submit your solution to autograder.io

(d) [3] Write your own test cases to sanity check your code, *before* you submit to the autograder. At least one test case should return $\hat{\boldsymbol{y}} = 1$ and one should return $\hat{\boldsymbol{y}} = 2$. Submit a screen shot of your test code.

(e) [3] Find an expression for the risk of the Bayes classifier. Your expression should be computable to within machine precision (no infinite summations).

3. [24] **Naive Bayes classifier**

(a) [10] Write a JULIA or Python function that implements the Naive Bayes classifier. The function input is training data $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and labels $\boldsymbol{y} \in \mathbb{R}^n$ and a matrix of $m \geq 1$ test cases $\boldsymbol{X}_{\text{test}} \in \mathbb{R}^{d \times m}$.
Your submitted file should be named `naive2.jl` or `naive2.py` and should contain the following function:

```
"""
    ytest = naive2(Xtrain, ytrain, Xtest)

Naive Bayes binary classifier

In:
- `Xtrain` `d x n` matrix of training data;
   `n` sample vectors with `d` features
- `ytrain` length-`n` vector of training labels
   where `length(unique(ytrain)) == 2`
- `Xtest` `d x m` matrix of test data vectors

Out:
- `ytest` length-`m` vector of predicted labels
"""
function naive2(
    Xtrain::AbstractMatrix,
    ytrain::AbstractVector,
    Xtest::AbstractMatrix,
)
```

Submit your solution to autograder.io

(b) [3] Test your code with at least two cases: at least one case where you expect 0% errors, and one case where you expect $\approx 50\%$ errors. Submit a screen shot of your test code to gradescope.

(c) [3] Apply your Naive-Bayes classifier to real-world data: use 100 (each) MNIST images of digits 1 & 7 for training, and test with 500 (each) different images. First use PCA to do dimensionality reduction to $d = 2$.
One way to do this is to modify the code in the QDA demo. It is acceptable to use GenAI to convert that Julia code to Python if you want. It is *not* acceptable to use GenAI to help write the Naive Bayes version.
Submit a screenshot of your MNIST code.

(d) [8] Submit a plot of the training data and decision regions similar to the plot in the demo.
- (4) Superimpose the covariance estimate contours where $(1/2)(\boldsymbol{x} - \hat{\boldsymbol{\mu}})^\top \widehat{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{x} - \hat{\boldsymbol{\mu}}) = 1$.
- (2) Report the test error rate in the plot title.

4. [6] **Polynomial regression**

(a) [4] Consider fitting a polynomial $f(t) = \alpha_0 + \alpha_1 t + \cdots \alpha_m t^m$ to data pairs $(t_i, y_i),\ i = 1, \ldots, n$ where $y_i \approx f(t_i)$.
In practice, we often need regularization, but we do *not* want to regularize the constant term α_0 to avoid a systematic bias, and to make the solution equivariant to a vertical shift of the data.
Express concisely in matrix-vector notation the solution to the regularized least-squares problem

$$\arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{A}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \beta R(\boldsymbol{\alpha}), \quad R(\boldsymbol{\alpha}) \triangleq \sum_{j=1}^{m} \alpha_j^2,$$

where $\boldsymbol{A}$ has the usual definition for polynomial regression.
Hint. This is almost (but not quite) Tikhonov regularization. Rewrite R using a quadratic form possibly involving $\boldsymbol{I} - \boldsymbol{e}_1 \boldsymbol{e}_1^\top$, where $\boldsymbol{e}_1$ denotes the first unit vector in $\mathbb{R}^{m+1}$.

(b) [2] Is the minimizer unique? Discuss.

5. [5] **Weighted least squares regression**

Consider linear regression and let $c_1, \ldots, c_n > 0$ be known weights. Determine the solution of

$$\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{w}}, \hat{b}) = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R}} J(\boldsymbol{\theta}), \qquad J(\boldsymbol{\theta}) \triangleq \frac{1}{n} \sum_{i=1}^{n} c_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i\right)^2.$$

Express your solution in terms of the matrix $\boldsymbol{C} = \mathsf{Diag}\{c_1, \ldots, c_n\}$, an appropriate data matrix $\boldsymbol{X}$, and other notation as needed.

For grading consistency, please use an approach that estimates $\boldsymbol{w}$ and b together, as opposed to an approach that first eliminates b and then solves for $\boldsymbol{w}$.

Hint. As a sanity check, your solution should reduce to the ordinary least squares solution when $\boldsymbol{C}$ is the identity matrix.

6. [5] **Make your own problem**

Make a clearly stated problem that could be used on an EECS 553 exam based on the topics covered by the due date of this HW set, and provide your own solution to the problem.

Follow all the instructions from HW2 except create your problem in `hw3.tex` using the Overleaf link on EECS 553 Canvas home page.

Non-graded problems

7. [0] **Maximum likelihood covariance estimate**

Find the maximum-likelihood estimate for the covariance matrix $\boldsymbol{\Sigma}$ of a zero-mean multivariate Gaussian, given n IID samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ drawn from that distribution, assuming $n \geq d$.

Hint. Use the Matrix Cookbook for matrix derivatives.

8. [0] **Least-squares regression (under-determined cases)**

This problem considers a particular case of the linear least squares problem. Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and a vector $\boldsymbol{y} \in \mathbb{R}^n$, our goal is to find a vector $\boldsymbol{x} \in \mathbb{R}^n$ that minimizes $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. When $\boldsymbol{A}$ is an invertible square matrix, and $\boldsymbol{y}$ is in the range of $\boldsymbol{A}$, then $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$ is the unique solution. This problem considers the case where $\boldsymbol{A}$ is symmetric but *not* invertible. We will use the eigendecomposition of $\boldsymbol{A}$ to derive an analytical solution to linear least squares in this case.

Any diagonalizable matrix $\boldsymbol{A}$ has a decomposition of the form $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$, where $\boldsymbol{Q}, \boldsymbol{\Lambda} \in \mathbb{R}^{n \times n}$, $\boldsymbol{\Lambda}$ is diagonal, and $\boldsymbol{A}\boldsymbol{q}_i = \lambda_i \boldsymbol{q}_i$ for all $i = 1, \ldots, n$. Here, $\boldsymbol{q}_i$ is the ith column of $\boldsymbol{Q}$ and λ_i is the ith diagonal entry of $\boldsymbol{\Lambda}$. We call $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$ an *eigendecomposition* of $\boldsymbol{A}$. If $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a symmetric matrix (which guarantees diagonalizability), the *spectral theorem* guarantees the following:

1. $\boldsymbol{Q}$ has orthonormal columns[1] (also called a unitary matrix), *i.e.*, $\boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I} = \boldsymbol{Q}^\top\boldsymbol{Q}$.
2. $\lambda_i \in \mathbb{R}$ for all $i = 1, \ldots, n$

Consequently, we can write the eigendecomposition of a real-valued symmetric matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top.$$

Note: Since we have assumed $\boldsymbol{A}$ is non-invertible, there exists at least one $i \in \{1, \ldots, n\}$ for which $\lambda_i = 0$.

(a) Prove that $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ can be written $\|\boldsymbol{\Lambda}\widetilde{\boldsymbol{x}} - \widetilde{\boldsymbol{y}}\|_2^2$, where $\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}$ are some vectors in $\mathbb{R}^n$. Give the expressions for $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}}$.
Hint. If $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is a unitary matrix, it is a fact that $\|\boldsymbol{U}\boldsymbol{z}\|_2^2 = \|\boldsymbol{z}\|_2^2$ for all $\boldsymbol{z} \in \mathbb{R}^n$. If a norm has this property, it is called unitarily invariant.

(b) Using the above result, prove that $\boldsymbol{x}^* = \boldsymbol{Q}\boldsymbol{\Lambda}^\dagger\boldsymbol{Q}^\top\boldsymbol{y}$ is a minimizer of $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$, where we take $[\boldsymbol{\Lambda}^\dagger]_{i,j} = \begin{cases} \frac{1}{\Lambda_{i,j}}, & \Lambda_{i,j} \neq 0 \\ 0, & \text{otherwise.} \end{cases}$
Hint. Write out $\|\boldsymbol{\Lambda}\widetilde{\boldsymbol{x}} - \widetilde{\boldsymbol{y}}\|_2^2$ as a sum of scalars, and split the sum into terms where the eigenvalues are zero and non-zero.

(c) Prove a similar result for non-square $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ (i.e., not symmetric, not square) using the singular value decomposition. The proof follows a similar approach.

[1] The spectral theorem actually guarantees the eigenvectors are orthogonal. By convention, we take the eigenvectors to have unit norm. This implies we can write the eigenvalue decomposition as specified.