cs584 f25 – Assignment 2 (Generative Learning)

Due by: 10/12/2025

General Instructions

Objectives

By the end of this assignment, you will: (i) implement Gaussian discriminant analysis (1D, 2-class; nD, 2-class; nD, k-class); (ii) implement Naive Bayes with Bernoulli and Binomial features; (iii) evaluate performance using multiple metrics and cross-validation; (iv) explore both continuous and discrete feature datasets.

- Unless a question explicitly allows it, implement algorithms *from scratch* (NumPy/SciPy/Matplotlib allowed; `scikit-learn` only for dataset loading and sanity-check baselines).
- Use 10-fold cross-validation (CV) unless told otherwise. Fix a random seed and report it.
- Turn in a single Python notebook with answers, figures, and tables.
- Include a 1–2 paragraph reflection at the end: key findings, pitfalls, and lessons learned.

Submission Instructions

- Upload your submission to Canvas.
- Submit a fully executed Python notebook with no gaps in execution. To receive full credit, please ensure: The notebook includes all cell outputs and contains no error messages. Each problem is clearly labeled in Markdown or comments (e.g., "Problem 2"). Re-run your notebook before submitting to ensure that cells are numbered sequentially starting at `[1]`.

Experiment instructions

Datasets

Characteristics:

- You must use two or more external datasets: At least one with continuous features. At least one with discrete features.
- Suggested sources: UCI Machine Learning Repository. Examples: Continuous: Iris dataset. Discrete: Spambase dataset.

Notes:

- For discrete text-derived features, you may binarize word counts (non-zero frequency $\rightarrow$ 1, zero frequency $\rightarrow$ 0).
- You may convert continuous features to integers if appropriate. For example, approximate word counts from frequencies (multiply frequency by document length). You may use a random document length if unknown.

- For 2-class problems, you may select 2 classes from a dataset with more than 2.

Performance Metrics

- At minimum, report: Confusion Matrix, Precision, Recall, F1-score, Accuracy, (For some tasks) Precision–Recall curve and area under curve (AUC)
- For Cross-Validation (CV), report mean $\pm$ std across folds. When comparing models, include a table and a brief interpretation.

Programming Tasks

1. 1D 2-Class Gaussian Discriminant Analysis

1. Select a 2-class dataset with continuous 1D features.
2. Estimate model parameters and compute a discriminant function for each class distribution.
3. Classify examples and measure error. Report confusion matrix, precision, recall, F-measure, and accuracy.

2. nD 2-Class Gaussian Discriminant Analysis

1. Select a 2-class dataset with continuous nD features.
2. Estimate model parameters and compute a discriminant function for each class distribution.
3. Classify examples and measure error. Report confusion matrix, precision, recall, F-measure, and accuracy.
4. Plot the precision–recall curve and compute AUC.

3. nD k-Class Gaussian Discriminant Analysis

1. Select a k-class dataset with continuous nD features.
2. Estimate model parameters and compute a discriminant function for each class distribution.
3. Classify examples and measure error. Report confusion matrix, precision, recall, F-measure, and accuracy.

4. Naive Bayes with Bernoulli Features

1. Select a 2-class dataset with binary nD features (derived from text documents).
2. Estimate model parameters and compute a discriminant function under the Naive Bayes assumption.
3. Classify examples and measure error. Report confusion matrix, precision, recall, F-measure, and accuracy.

5. Naive Bayes with Binomial Features

1. Theory (written): Derive maximum likelihood parameter estimation for Naive Bayes with Binomial features: Write the log likelihood function. Compute its derivative. Equate derivative to zero and solve for parameters. There is no need to derive class priors.
2. Select a 2-class dataset with discrete nD features (derived from text documents).
3. Estimate model parameters and compute a discriminant function under the Naive Bayes assumption.
4. Classify examples and measure error. Report confusion matrix, precision, recall, F-measure, and accuracy.

6. Comparison with Baselines

To validate your implementations, compare results with `scikit-learn` baselines:

1. Gaussian Discriminant Analysis: Use `sklearn.discriminant_analysis.LinearDiscriminantAnalysis` (LDA). Report performance and note differences.
2. Naive Bayes: Use `sklearn.naive_bayes.GaussianNB` (for continuous), `BernoulliNB` (for binary), `MultinomialNB` (for counts). Compare results against your implementation.
3. Include a concise table for each problem showing: Your implementation's results, `sklearn` baseline results, a brief discussion of differences (precision, numerical stability, etc.)

7. Reflection

At the end of your notebook, include a 1–2 paragraph reflection: Key findings from implementing generative models. Pitfalls and challenges encountered. Lessons learned about generative learning methods.

Starter Code

Below is some basic starter code to help you load datasets, split into train/test, and prepare evaluation:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score,
accuracy_score

# Example: loading a dataset (replace with your dataset)
# Iris dataset (continuous)
from sklearn.datasets import load_iris
iris = load_iris()
X, y = iris.data, iris.target

# For a 2-class problem, select two classes
mask = (y == 0) | (y == 1)
X, y = X[mask], y[mask]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Example evaluation function
def evaluate(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='macro')
    rec = recall_score(y_true, y_pred, average='macro')
    f1 = f1_score(y_true, y_pred, average='macro')
    acc = accuracy_score(y_true, y_pred)
    return cm, prec, rec, f1, acc
```

```
# Placeholder for your implementation
# def gda_train(...):
# def gda_predict(...):
# def naive_bayes_train(...):
# def naive_bayes_predict(...):
```

Use this template to organize each problem section clearly. Expand with your own implementations for Gaussian discriminant analysis and Naive Bayes. Using this code is not mandatory.