

Assignment-2 - programming questions

1. Implement logistic regression trained by (A) full-batch gradient descent and (B) stochastic gradient descent (batch size = 1). Compare efficiency and behaviour.

Synthetic data: generate $N = 10,000$ samples in R^{10} from two Gaussians with overlap: class $+1 \sim \mathcal{N}(\mu, I)$, class $-1 \sim \mathcal{N}(-\mu, I)$ with $\mu = (0.5, \dots, 0.5)^\top$. Balance classes; use a fixed random seed.

```
import numpy as np; N,d=10000,10; mu=0.5*np.ones(d); np.random.seed(0)
Xp=np.random.randn(N//2,d)+mu; Xn=np.random.randn(N//2,d)-mu;
X=np.vstack([Xp,Xn]); y=np.hstack([np.ones(N//2),-np.ones(N//2)])
```

Split 80/20 train/test. Standardize features using train statistics. Labels $y \in \{-1, +1\}$. For $x \in R^d$, weight $w \in R^d$, bias $b \in R$, use the average logistic loss

$$\mathcal{L}(w, b; S) = \frac{1}{|S|} \sum_{(x,y) \in S} \log(1 + \exp(-y(w^\top x + b))).$$

Train both methods for 20 epochs with fixed learning rates $\eta_{\text{batch}} = 0.1$, $\eta_{\text{sgd}} = 0.05$. Shuffle data each epoch; for SGD, iterate samples in a random order without replacement.

Answer the following:

- (a) Report the number of gradient-evaluation operations (ops) per epoch for method A and method B. Also report the total ops over 20 epochs for each method.
 - (b) Report final train loss, test loss, and test accuracy for method A and method B.
 - (c) Provide a single plot: train loss versus ops, showing both methods on the same axes (ops on the x-axis).
2. Consider the function

$$R(\beta) = \sin(\beta) + \frac{\beta}{10},$$

and its regularized version

$$R_\lambda(\beta) = \sin(\beta) + \frac{\beta}{10} + \frac{\lambda}{2}\beta^2.$$

- (a) Plot $R(\beta)$ and $R_\lambda(\beta)$ for $\lambda = 0.1$ over $\beta \in [-6, 6]$ on the same graph. Identify approximate locations of minima.
 - (b) Derive $R'(\beta)$ and $R'_\lambda(\beta)$ analytically.
 - (c) Starting from $\beta_0 = 2.3$, perform 20 iterations of gradient descent with learning rate $\rho = 0.1$ for both $\lambda = 0$ and $\lambda = 0.1$. Record β_t at each step and plot the descent path on the curve.
 - (d) Repeat part (c) with a different initialization $\beta_0 = 1.4$, and overlay on the same plot(s).
3. Write a Python function from scratch (no libraries like `sklearn.tree`) that:
- Takes a dataset (as a list of dictionaries or pandas DataFrame),
 - Computes the best attribute to split on using Information Gain,
 - Returns the name of that attribute.

Example:

```
best_attr = find_best_split(df, target_col="PlayTennis")
print("Best attribute to split on:", best_attr)
```

Expected Output (for the PlayTennis dataset): Best attribute to split on: Outlook

Hint: Compute entropy and weighted average information gain manually.

4. (a) Load the Iris or Titanic dataset using `sklearn.datasets` or `pandas`.
 (b) Split into train/test sets and train a `DecisionTreeClassifier`.
 (c) Report:
 - Accuracy on the test set
 - Depth of the trained tree
 (d) Visualize the decision tree using `plot_tree()` or `graphviz`.
 (e) Explain in 3–4 lines how the top 3 features contribute to classification.
5. Implement Gaussian Discriminant Analysis (GDA) for a two-class classification problem using Python. You are given a dataset with features $X \in \mathbb{R}^{n \times d}$ and binary labels $y \in \{0, 1\}$.

Tasks:

(a) Estimate the GDA parameters:

$$\begin{aligned}\phi &= \frac{1}{n} \sum_{i=1}^n 1\{y^{(i)} = 1\}, \\ \mu_k &= \frac{\sum_{i=1}^n 1\{y^{(i)} = k\} x^{(i)}}{\sum_{i=1}^n 1\{y^{(i)} = k\}}, \\ \Sigma &= \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T\end{aligned}$$

(b) Use these parameters to compute the posterior probability:

$$P(y = 1 \mid x) = \frac{P(x \mid y = 1)\phi}{P(x \mid y = 1)\phi + P(x \mid y = 0)(1 - \phi)}$$

(c) Predict the class label as:

$$\hat{y} = \begin{cases} 1, & \text{if } P(y = 1 \mid x) > 0.5, \\ 0, & \text{otherwise.} \end{cases}$$

6. The Optimal Model [Programming Question]:

You have been given a dataset in `logistic_classification_data.csv`. This 2D data contains two classes (0 and 1) that are *not* linearly separable. Your task is to find the best-performing model.

Instructions:

- Load the data and split it into an 80% training set and a 20% testing set.
- Because the data is not linearly separable, you will need to use **polynomial features** (from `sklearn.preprocessing`) to create a more complex decision boundary.
- Build and evaluate six different models by combining `PolynomialFeatures` with `LogisticRegression`. You should test polynomial degrees $n = 1$ through $n = 6$.
Hint: Use `sklearn.pipeline.Pipeline` to chain your steps (e.g., `StandardScaler`, `PolynomialFeatures`, `LogisticRegression`).
- For each of the six models, calculate and record the **F1-Score** (a good metric for classification) on both the **training set** and the **testing set**.
- Report the most likely best polynomial degree (n) for this problem. Justify your choice by referencing your training and testing F1-Scores and explaining the tradeoff (i.e., why you didn't just choose the model with the highest training score).

7. Programming Task: Investigating Model Limitations

Objective: To programmatically investigate and visualize two major limitations of a standard `DecisionTreeRegressor`: its "step-function" nature and its inability to extrapolate.

(a) Load Data and Train Models:

- Load the **Boston Housing Dataset** using `sklearn.datasets`
- Use **only** the 'RM' column (average number of rooms) as X , and 'price' as y .
- Split the dataset into *Train* and *Test* data
- Train a '`LinearRegression`' model and '`DecisionTreeRegressor`' model on the training data. For clarity, set `max_depth = 3`.

(b) Analyze the Predictions:

- Create a test data `X_visualize` with 100 values, uniformly spaced from **3 to 12**. (Note: the range is wider than the training data's range).
- Get predictions for the **in-range 'RM' values**: [7.1], [7.2], and [7.3], and **extrapolated 'RM' values** [11.0], [11.5], and [12.0] from *both* the models.
- **Explanation:**
 - Why are the '`DecisionTreeRegressor`' model's predictions for the **in-range 'RM' values**: likely identical, while the '`LinearRegression`' model's predictions are clearly different and increasing?
 - Why are the '`DecisionTreeRegressor`' model's for **extrapolated 'RM' values**: all identical? What specific value does this prediction seem to be "stuck" at, and why?