CS 273P: Machine Learning and Data Mining
**Homework 3**
Due date: **See Canvas**
Instructor: Xiaohui Xie

This homework is graded automatically on **Gradescope** using an autograder. All questions are unit-testable: you will implement specific functions that return specified outputs.

**Allowed libraries:** You may use `numpy` (required), and optionally `scipy` for numerical helpers. You may use `pandas` only for loading data. Do **not** use `scikit-learn` models for training; all learning algorithms in this homework must be implemented from scratch.

**Submission rules (READ CAREFULLY):**

1. Download the starter files from the GitHub link.
2. Complete your work by editing only the provided Python files: `problem1.py`, `problem2.py`, and `problem3.py`.
3. Create a plain text file called `statement.txt` containing your collaboration statement.
4. Submit to Gradescope by uploading a single **ZIP file** containing exactly:
   - `problem1.py`
   - `problem2.py`
   - `problem3.py`
   - `statement.txt`

   No folders/subdirectories; all files must be at the top level of the zip.
5. You may resubmit before the deadline; your **highest score** counts.

If you run into issues with the autograder or submission format, post on Ed Discussion so others can benefit from the discussion.

**Points:** This homework adds up to a total of **110 points**, as follows:

| | |
|---|---|
| Problem 1: Binary Logistic Regression (from scratch) | 40 points |
| Problem 2: Multiclass Softmax Regression (from scratch) | 40 points |
| Problem 3: Binary Hinge-loss SVM (from scratch) | 25 points |
| Statement of Collaboration | 5 points |

# Datasets (Shared)

We will use the Iris dataset (`data/iris.txt`). Let $X \in \mathbb{R}^{N\times 4}$ be the features and $y \in \{0, 1, 2\}^N$ be the class labels. Unless otherwise specified:

- Use **all 4 features**.
- Standardize features (zero mean, unit variance) using the data subset provided to the function.
- Add a bias term internally (do not modify $X$ on disk).

All functions must be deterministic given a seed.

# Problem 1: Binary Classification with Logistic Regression (40 points)

In this problem, you will implement binary logistic regression trained with gradient descent.

**Binary task.** Create a binary dataset from Iris using class pair $\{0, 1\}$ or $\{1, 2\}$. For `pair=(a,b)`, keep only samples with labels in $\{a, b\}$ and map the smaller label to 0 and the larger to 1.

All required functions for Problem 1 are in `problem1.py`.

## 1.1 Data preparation (8 points)

Implement:

1. `load_iris_binary(path, pair) -> (X, y)` (8 pts)
   Return standardized $X \in \mathbb{R}^{N\times 4}$ and binary $y \in \{0, 1\}^N$.

## 1.2 Core math (16 points)

Implement:

1. `sigmoid(z) -> ndarray` (3 pts)
   Stable sigmoid $\sigma(z) = 1/(1 + \exp(-z))$.
2. `logistic_loss(X, y, w, reg=0.0) -> float` (6 pts)
   Average negative log-likelihood with L2 regularization on weights excluding bias. Use $w \in \mathbb{R}^{d+1}$ with bias $w_0$.
3. `logistic_grad(X, y, w, reg=0.0) -> ndarray` (7 pts)
   Gradient of `logistic_loss` w.r.t. $w$.

## 1.3 Training + evaluation (16 points)

Implement:

1. `train_logreg(X, y, step_size=0.1, max_epochs=2000, tol=1e-6, batch_size=0, reg=0.0, seed=0) -> dict` (12 pts)
   Train from $w = 0$ using full-batch GD if `batch_size=0`, else mini-batch GD. Stop early when $|L_t - L_{t-1}| <$ `tol` (loss computed on full data). Return a dict containing at least: `{ "w": w, "loss_history": ..., "err_history": ..., "epochs": E }`.
2. `predict_proba(X, w) -> ndarray` (2 pts): return $p(y = 1 \mid x)$.
3. `predict(X, w, threshold=0.5) -> ndarray` (2 pts): return labels in {0,1}.

# Problem 2: Multiclass Classification with Softmax Regression (40 points)

In this problem, you will implement a multiclass linear classifier trained with softmax + cross entropy.

**Task.** Use the full Iris dataset with three classes $\{0, 1, 2\}$. Standardize features (zero mean, unit variance). Use all points for training.

All required functions for Problem 2 are in `problem2.py`.

## 2.1 Core math (22 points)

Implement:

1. `softmax(Z) -> ndarray` (6 pts)
   Stable softmax applied row-wise: input $Z \in \mathbb{R}^{N\times K}$, output probabilities $P \in [0, 1]^{N\times K}$.
2. `one_hot(y, K) -> ndarray` (4 pts)
   Convert labels $y \in \{0, \ldots, K - 1\}^N$ to $Y \in \{0, 1\}^{N\times K}$.

3. `softmax_loss(X, y, W, reg=0.0) -> float` (6 pts)
   Average cross-entropy loss with L2 regularization on weights excluding bias. Use $W \in \mathbb{R}^{(d+1)\times K}$ where the first row corresponds to bias weights.

4. `softmax_grad(X, y, W, reg=0.0) -> ndarray` (6 pts)
   Gradient of `softmax_loss` w.r.t. $W$.

### 2.2 Training + evaluation (18 points)

Implement:

1. `train_softmax(X, y, K, step_size=0.1, max_epochs=3000, tol=1e-6, batch_size=0, reg=0.0, seed=0) -> dict` (12 pts)
   Train from $W = 0$ using GD or mini-batch GD. Early stopping based on loss change. Return a dict containing at least: `{ "W": W, "loss_history": ..., "acc_history": ..., "epochs": E }`.

2. `predict_proba_softmax(X, W) -> ndarray` (3 pts)
   Return class probabilities $P \in [0, 1]^{N\times K}$.

3. `predict_softmax(X, W) -> ndarray` (3 pts)
   Return predicted labels in $\{0, \dots, K-1\}$ using argmax.

## Problem 3: Binary Classification with Hinge-loss SVM (25 points)

In this problem, you will implement a linear SVM-style classifier by minimizing hinge loss with L2 regularization (using gradient descent). We use the primal objective with slack via hinge loss.

**Binary task.** Use Iris class pair `pair=(0,1)`. Map labels to $y \in \{-1, +1\}$.
All required functions for Problem 3 are in `problem3.py`.

### 3.1 Core math (15 points)

Implement:

1. `hinge_loss(X, y, w, C=1.0) -> float` (7 pts)
   Objective:
   $$J(w) = \frac{1}{2}\|w_{1:}\|_2^2 + C \cdot \frac{1}{N}\sum_{j=1}^{N} \max(0, 1 - y^{(j)}(\, w_0 + x^{(j)} \cdot w_{1:}\,))$$
   where $y^{(j)} \in \{-1, +1\}$ and bias $w_0$ is not regularized.

2. `hinge_grad(X, y, w, C=1.0) -> ndarray` (8 pts)
   Subgradient of `hinge_loss` w.r.t. $w$ (choose any valid subgradient at the hinge point).

### 3.2 Training + prediction (10 points)

Implement:

1. `train_svm_hinge(X, y, C=1.0, step_size=0.1, max_epochs=5000, tol=1e-6, batch_size=0, seed=0) -> dict` (7 pts)
   Train from $w = 0$ using GD or mini-batch GD. Early stopping based on objective change. Return a dict containing at least: `{ "w": w, "loss_history": ..., "err_history": ..., "epochs": E }`.

2. `predict_svm(X, w) -> ndarray` (3 pts)
   Return labels in $\{-1, +1\}$ using $\mathrm{sign}(w_0 + Xw_{1:})$; break ties (0) as $+1$.

## Statement of Collaboration (5 points)

Add a plain text file `statement.txt` to your submission. List the names of collaborators and the nature of collaboration, or write "No collaboration." if you worked alone. Do not share code. Academic honesty policies apply.