guyvitelson_mmn11_ml_latest

March 19, 2025

1 11 - - 2025 - 203379706

##**If you run this within Google Collab, Dont Worry!** all the missing python files/directories/modules will be automatically feteched from my github repository

My GitHub Profile : https://github.com/v1t3ls0n

The Repository: https://github.com/v1t3ls0n/ml_intro_course_mmn11

Student ID: 203379706

1.1 Fetch Resources

1.1.1 External Code Imports (pip packages)

[1]:
```
import os
import shutil
import sys
import logging
import numpy as np # type: ignore
import matplotlib.pyplot as plt # type: ignore
import seaborn as sns # type: ignore
import time
import pandas as pd
```

1.1.2 Fetch Missing Files For Google Colab Env

[2]:
```
# %%capture run_output
# %matplotlib inline

if sys.platform != 'win32': # check if we are running on google collab
  repo_url = "https://github.com/v1t3ls0n/ml_intro_course_mmn11"
  repo_name = "ml_intro_course_mmn11"
  from tqdm.notebook import tqdm # type: ignore


  # Clone the repository if it doesn't exist
  if not os.path.exists(repo_name):
    os.system(f"git clone {repo_url}")
```

```
  # Construct the path to the repository directory
  repo_path = os.path.join(os.getcwd(), repo_name)

  # Add the repository directory to the Python path
  if repo_path not in sys.path:
    sys.path.insert(0, repo_path)

  # --- Extract 'core' and 'notebooks' directories ---
  def extract_directories(source_dir, destination_dir, dir_names):
      for dir_name in dir_names:
          source_path = os.path.join(source_dir, dir_name)
          destination_path = os.path.join(destination_dir, dir_name)
          if os.path.exists(source_path):
              shutil.copytree(source_path, destination_path, dirs_exist_ok=True)

  destination_path = "."
  # Extract the directories
  extract_directories(repo_path, destination_path, ["core"])
  project_root = os.path.abspath(os.path.join(os.getcwd(), '..'))
  sys.path.insert(0, project_root)
  if os.path.exists("ml_intro_course_mmn11"):
    shutil.rmtree("ml_intro_course_mmn11")
  if os.path.exists("sample_data"):
    shutil.rmtree("sample_data")
else:
  from tqdm import tqdm  # type: ignore
  current_dir = os.getcwd()  # Current working directory
  project_root = os.path.abspath(os.path.join(current_dir, '..'))  # Root␣
 ↪directory of the project
  sys.path.insert(0, project_root)
```

1.1.3 Internal Code Imports (original code)

[3]:
```
# ========== Internal Code Imports ==========

#Logger
from core.logger.config import logger

# Data Preprocessing
from core.data.mnist_loader import load_mnist
from core.data.data_preprocessing import preprocess_data

# Models
from core.models.perceptron.multi_class_perceptron import MultiClassPerceptron
from core.models.logistic_regression.softmax_lregression import␣
 ↪SoftmaxRegression
from core.models.linear_regression.linear_regression import  LinearRegression
```

```
# Performance & Plotting
from core.analysis.evaluation_functions import (
    evaluate_model,
    aggregate_iteration_losses,
    aggregate_iteration_losses_softmax
)

from core.analysis.plotting import (
    plot_confusion_matrix_annotated,
    plot_error_curves,
    plot_accuracy_vs_max_iter,
    plot_runtime_vs_max_iter,
    plot_performance_summary_extended,
    plot_train_curves_three_models,
    plot_metric_vs_learning_rate,
    plot_accuracy_vs_max_iter_4models,
    plot_runtime_vs_max_iter_4models,
    plot_accuracy_vs_runtime,
    plot_performance_summary_extended_by_runtime,
    plot_performance_summary_4models_by_runtime,
    plot_accuracy_vs_runtime_4models
)

logger = logging.getLogger("MyGlobalLogger") # configured in core/logger/config.
 ↪py
```

2 Overview

2.1 MNIST Digit Classification Report

2.1.1 Approach

Data Preprocessing The MNIST dataset was prepared by: - Splitting into training (60,000 samples) and test sets (10,000 samples). - Normalizing pixel values to the [0,1] range. - Flattening images into vectors (784 pixels plus 1 bias term). - Encoding labels into one-hot vectors.

Model Implementation

- **Multi-Class Perceptron:**
  - One-vs-all strategy implemented with standard Perceptron and Pocket Perceptron algorithms.
- **Softmax Regression:**
  - Implemented using cross-entropy loss and adaptive learning rates (AdaGrad).
  - Included early stopping based on loss improvement.
- **Linear Regression:**
  - Utilized mean squared error loss with gradient descent.
  - AdaGrad adaptive learning rate and early stopping were applied.

2.1.2 Results

- **Accuracy:**
  - Softmax Regression achieved the highest accuracy.
  - Multi-class Pocket Perceptron showed good performance, surpassing standard Perceptron.
  - Linear Regression exhibited relatively lower accuracy due to its limitations for classification tasks.

Confusion Matrices and Metrics

- Softmax Regression demonstrated the lowest misclassification rates across digits.
- Pocket Perceptron reduced errors compared to standard Perceptron, indicating improved robustness.
- Sensitivity and accuracy clearly highlighted Softmax Regression as superior for multi-class digit classification.

2.1.3 Discussion

- Softmax Regression proved best for digit classification, providing reliable probability estimations and stable convergence.
- Pocket Perceptron algorithm offered notable improvements over standard Perceptron, highlighting its utility in non-linearly separable scenarios.
- Linear Regression's limitations in classification tasks were evident, reaffirming theoretical expectations.

2.1.4 Conclusions

- Softmax Regression is the most suitable algorithm for multi-class digit recognition problems.
- Pocket Perceptron serves as an effective alternative, offering a balance between simplicity and performance.
- Linear Regression, while straightforward, is suboptimal for classification due to its inherent limitations.

3 Choose Run Parameters (Significant Effect On Model's Runtime!)

[4]:
```
######################################################################
# SEPARATE RUN PARAMETERS FOR PERCEPTRONS vs. REGRESSIONS
######################################################################

# Perceptrons (Clean & Pocket) iteration-based run
perceptron_max_iter_values = [50,100,2000,3000]  # for Clean PLA & Pocket PLA
# Logging the run parameters
logger.info(f"=== Perceptron Run Parameters ===")
logger.info(f"max_iter_values = {perceptron_max_iter_values}")
```

```
# Regression (Softmax & Linear) run parameters.
learning_rates = [0.1]  # for Softm vax & Linear Regression
iteration_counts = [50,100,2000,3000]
regression_run_configs = [
    {
        "label": f"LR={lr}/Iter={it}",
        "learning_rate": lr,
        "max_iter": it
    }
    for lr in learning_rates
    for it in iteration_counts
]

logger.info(f"=== Regression Run Parameters ===")
for cfg in regression_run_configs:
    logger.info(f"{cfg['label']} -> learning_rate={cfg['learning_rate']},␣
 ↪max_iter={cfg['max_iter']}")
```

```
INFO - === Perceptron Run Parameters ===
INFO - max_iter_values = [50, 100, 2000, 3000]
INFO - === Regression Run Parameters ===
INFO - LR=0.1/Iter=50 -> learning_rate=0.1, max_iter=50
INFO - LR=0.1/Iter=100 -> learning_rate=0.1, max_iter=100
INFO - LR=0.1/Iter=2000 -> learning_rate=0.1, max_iter=2000
INFO - LR=0.1/Iter=3000 -> learning_rate=0.1, max_iter=3000
INFO - max_iter_values = [50, 100, 2000, 3000]
INFO - === Regression Run Parameters ===
INFO - LR=0.1/Iter=50 -> learning_rate=0.1, max_iter=50
INFO - LR=0.1/Iter=100 -> learning_rate=0.1, max_iter=100
INFO - LR=0.1/Iter=2000 -> learning_rate=0.1, max_iter=2000
INFO - LR=0.1/Iter=3000 -> learning_rate=0.1, max_iter=3000
```

4 Load and Preprocess the MNIST Dataset

[5]:
```
'''
We'll load the MNIST dataset using our custom loader (`mnist_loader`) and then␣
 ↪apply preprocessing (`data_preprocessing`).
The preprocessing step normalizes each image to the range [0, 1] and adds a␣
 ↪bias term, resulting in input samples with 785 features.
This setup ensures that the training set contains 60,000 samples and the test␣
 ↪set 10,000 samples, preparing the data for the subsequent classification␣
 ↪tasks.
'''

# New section
# Load raw MNIST data (X: images, y: labels)
```

```
X_raw, y_raw = load_mnist()


logger.info("Raw MNIST data shapes: X_raw: %s, y_raw: %s", X_raw.shape, y_raw.
 ↪shape)

# Preprocess (normalize & add bias = True)
X = preprocess_data(X_raw, add_bias=True, normalize=True)
logger.info("Preprocessed shape: %s", X.shape)

# Split into train/test manually or with 60k/10k as the task suggests
X_train, y_train = X[:60000], y_raw[:60000]
X_test,  y_test  = X[60000:], y_raw[60000:]

logger.info("Train set: X_train: %s, y_train: %s", X_train.shape, y_train.shape)
logger.info("Test set: X_test: %s, y_test: %s", X_test.shape, y_test.shape)
```

```
INFO - Raw MNIST data shapes: X_raw: (70000, 784), y_raw: (70000,)
INFO - Preprocessed shape: (70000, 785)
INFO - Train set: X_train: (60000, 785), y_train: (60000,)
INFO - Test set: X_test: (10000, 785), y_test: (10000,)
```

5 Train

[ ]:
```
# ============================================================
# TRAINING CELL
# ============================================================

# 1) Dictionaries to store trained models
trained_models_clean   = {}
trained_models_pocket  = {}
trained_models_softmax = {}
trained_models_linear  = {}

# 2) Train Regression Models (Softmax & Linear)
logger.info("=== TRAINING REGRESSION MODELS (Softmax & Linear) ===")
for cfg in tqdm(regression_run_configs, desc="Train Regressions"):
    lr_val = cfg["learning_rate"]
    max_iter_val = cfg["max_iter"]
    label = cfg["label"]  # e.g. "LR=0.001/Iter=1000"

    # --- Softmax ---
    logger.info(f"--- Softmax {label} ---")
    s_model = SoftmaxRegression(
        num_classes=10,
        max_iter=max_iter_val,
        learning_rate=lr_val,
```

```
        adaptive_lr=True
    )
    s_model.fit(X_train, y_train)
    trained_models_softmax[(lr_val, max_iter_val)] = s_model

    # --- Linear ---
    logger.info(f"--- Linear Regression {label} ---")
    lin_model = LinearRegression(
        num_classes=10,
        max_iter=max_iter_val,
        learning_rate=lr_val,
        adaptive_lr=True,
        early_stopping=False
    )
    lin_model.fit(X_train, y_train)
    trained_models_linear[(lr_val, max_iter_val)] = lin_model

logger.info("Training complete for Softmax and Linear.")

# 3) Train Perceptron Models (Clean & Pocket)
logger.info("=== TRAINING PERCEPTRON MODELS (Clean & Pocket) ===")
for max_iter in tqdm(perceptron_max_iter_values, desc="Train Clean & Pocket"):
    logger.info(f"--- Clean PLA, max_iter={max_iter} ---")
    clean_perc = MultiClassPerceptron(num_classes=10, max_iter=max_iter,␣
 ↪use_pocket=False)
    clean_perc.fit(X_train, y_train)
    trained_models_clean[max_iter] = clean_perc

    logger.info(f"--- Pocket PLA, max_iter={max_iter} ---")
    pocket_perc = MultiClassPerceptron(num_classes=10, max_iter=max_iter,␣
 ↪use_pocket=True)
    pocket_perc.fit(X_train, y_train)
    trained_models_pocket[max_iter] = pocket_perc

logger.info("Training complete for Clean PLA and Pocket PLA.")
logger.info("=== ALL TRAINING COMPLETE ===")
```

```
INFO - === TRAINING REGRESSION MODELS (Softmax & Linear) ===
Train Regressions:   0%|          | 0/4 [00:00<?, ?it/s]INFO - --- Softmax
LR=0.1/Iter=50 ---
Train Regressions:   0%|          | 0/4 [00:00<?, ?it/s]INFO - --- Softmax
LR=0.1/Iter=50 ---
INFO - Iter 1/50, Loss: 2.3446, Avg Adaptive LR: 14.144338
INFO - Iter 11/50, Loss: 0.4314, Avg Adaptive LR: 3.019835
INFO - Iter 21/50, Loss: 0.3745, Avg Adaptive LR: 3.001877
INFO - Iter 31/50, Loss: 0.3500, Avg Adaptive LR: 2.999318
INFO - Iter 41/50, Loss: 0.3354, Avg Adaptive LR: 2.998457
```

```
INFO - SoftmaxRegression training completed in 2.01 seconds.
INFO - --- Linear Regression LR=0.1/Iter=50 ---
INFO - LinearRegressionClassifier training completed in 1.54 seconds.
Train Regressions:  25%|        | 1/4 [00:03<00:10,  3.55s/it]INFO - ---
Softmax LR=0.1/Iter=100 ---
INFO - Iter 1/100, Loss: 2.3862, Avg Adaptive LR: 14.077829
INFO - Iter 11/100, Loss: 0.4332, Avg Adaptive LR: 3.181821
INFO - Iter 21/100, Loss: 0.3769, Avg Adaptive LR: 3.174093
INFO - Iter 31/100, Loss: 0.3522, Avg Adaptive LR: 3.172495
INFO - Iter 41/100, Loss: 0.3366, Avg Adaptive LR: 3.171523
INFO - Iter 51/100, Loss: 0.3254, Avg Adaptive LR: 3.170821
INFO - Iter 61/100, Loss: 0.3168, Avg Adaptive LR: 3.170281
INFO - Iter 71/100, Loss: 0.3099, Avg Adaptive LR: 3.169847
INFO - Iter 81/100, Loss: 0.3042, Avg Adaptive LR: 3.169489
INFO - Iter 91/100, Loss: 0.2994, Avg Adaptive LR: 3.169185
INFO - SoftmaxRegression training completed in 3.90 seconds.
INFO - --- Linear Regression LR=0.1/Iter=100 ---
INFO - Iter 100/100, Loss: 0.5185, Gradient Norm: 13.5634, Avg Adaptive LR:
1.4043025216183762
INFO - LinearRegressionClassifier training completed in 3.07 seconds.
Train Regressions:  50%|        | 2/4 [00:10<00:11,  5.56s/it]INFO - ---
Softmax LR=0.1/Iter=2000 ---
INFO - Iter 1/2000, Loss: 2.3588, Avg Adaptive LR: 13.631132
INFO - Iter 11/2000, Loss: 0.4321, Avg Adaptive LR: 3.051432
INFO - Iter 21/2000, Loss: 0.3708, Avg Adaptive LR: 3.044953
INFO - Iter 31/2000, Loss: 0.3484, Avg Adaptive LR: 3.043681
INFO - Iter 41/2000, Loss: 0.3339, Avg Adaptive LR: 3.042849
INFO - Iter 51/2000, Loss: 0.3233, Avg Adaptive LR: 3.042241
INFO - Iter 61/2000, Loss: 0.3152, Avg Adaptive LR: 3.041768
INFO - Iter 71/2000, Loss: 0.3086, Avg Adaptive LR: 3.041387
INFO - Iter 81/2000, Loss: 0.3032, Avg Adaptive LR: 3.041070
INFO - Iter 91/2000, Loss: 0.2986, Avg Adaptive LR: 3.040803
INFO - Iter 101/2000, Loss: 0.2946, Avg Adaptive LR: 3.040572
INFO - Iter 111/2000, Loss: 0.2912, Avg Adaptive LR: 3.040371
INFO - Iter 121/2000, Loss: 0.2881, Avg Adaptive LR: 3.040193
INFO - Iter 131/2000, Loss: 0.2854, Avg Adaptive LR: 3.040034
INFO - Iter 141/2000, Loss: 0.2829, Avg Adaptive LR: 3.039891
INFO - Iter 151/2000, Loss: 0.2807, Avg Adaptive LR: 3.039762
INFO - Iter 161/2000, Loss: 0.2787, Avg Adaptive LR: 3.039644
INFO - Iter 171/2000, Loss: 0.2769, Avg Adaptive LR: 3.039535
INFO - Iter 181/2000, Loss: 0.2752, Avg Adaptive LR: 3.039435
INFO - Iter 191/2000, Loss: 0.2736, Avg Adaptive LR: 3.039343
INFO - Iter 201/2000, Loss: 0.2721, Avg Adaptive LR: 3.039257
INFO - Iter 211/2000, Loss: 0.2707, Avg Adaptive LR: 3.039176
INFO - Iter 221/2000, Loss: 0.2694, Avg Adaptive LR: 3.039101
INFO - Iter 231/2000, Loss: 0.2682, Avg Adaptive LR: 3.039030
INFO - Iter 241/2000, Loss: 0.2671, Avg Adaptive LR: 3.038963
INFO - Iter 251/2000, Loss: 0.2660, Avg Adaptive LR: 3.038900
```

```
INFO - Iter 261/2000, Loss: 0.2650, Avg Adaptive LR: 3.038841
INFO - Iter 271/2000, Loss: 0.2640, Avg Adaptive LR: 3.038784
INFO - Iter 281/2000, Loss: 0.2631, Avg Adaptive LR: 3.038730
INFO - Iter 291/2000, Loss: 0.2622, Avg Adaptive LR: 3.038678
INFO - Iter 301/2000, Loss: 0.2614, Avg Adaptive LR: 3.038629
INFO - Iter 311/2000, Loss: 0.2606, Avg Adaptive LR: 3.038582
INFO - Iter 321/2000, Loss: 0.2598, Avg Adaptive LR: 3.038538
INFO - Iter 331/2000, Loss: 0.2591, Avg Adaptive LR: 3.038494
INFO - Iter 341/2000, Loss: 0.2584, Avg Adaptive LR: 3.038453
INFO - Iter 351/2000, Loss: 0.2577, Avg Adaptive LR: 3.038413
INFO - Iter 361/2000, Loss: 0.2570, Avg Adaptive LR: 3.038375
INFO - Iter 371/2000, Loss: 0.2564, Avg Adaptive LR: 3.038338
INFO - Iter 381/2000, Loss: 0.2558, Avg Adaptive LR: 3.038302
INFO - Iter 391/2000, Loss: 0.2552, Avg Adaptive LR: 3.038268
INFO - Iter 401/2000, Loss: 0.2547, Avg Adaptive LR: 3.038235
INFO - Iter 411/2000, Loss: 0.2541, Avg Adaptive LR: 3.038202
INFO - Iter 421/2000, Loss: 0.2536, Avg Adaptive LR: 3.038171
INFO - Iter 431/2000, Loss: 0.2531, Avg Adaptive LR: 3.038141
INFO - Iter 441/2000, Loss: 0.2526, Avg Adaptive LR: 3.038112
INFO - Iter 451/2000, Loss: 0.2521, Avg Adaptive LR: 3.038083
INFO - Iter 461/2000, Loss: 0.2516, Avg Adaptive LR: 3.038056
INFO - Iter 471/2000, Loss: 0.2511, Avg Adaptive LR: 3.038029
INFO - Iter 481/2000, Loss: 0.2507, Avg Adaptive LR: 3.038003
INFO - Iter 491/2000, Loss: 0.2503, Avg Adaptive LR: 3.037977
INFO - Iter 501/2000, Loss: 0.2498, Avg Adaptive LR: 3.037952
INFO - Iter 511/2000, Loss: 0.2494, Avg Adaptive LR: 3.037928
INFO - Iter 521/2000, Loss: 0.2490, Avg Adaptive LR: 3.037905
INFO - Iter 531/2000, Loss: 0.2486, Avg Adaptive LR: 3.037882
INFO - Iter 541/2000, Loss: 0.2483, Avg Adaptive LR: 3.037860
INFO - Iter 551/2000, Loss: 0.2479, Avg Adaptive LR: 3.037838
INFO - Iter 561/2000, Loss: 0.2475, Avg Adaptive LR: 3.037816
INFO - Iter 571/2000, Loss: 0.2472, Avg Adaptive LR: 3.037795
INFO - Iter 581/2000, Loss: 0.2468, Avg Adaptive LR: 3.037775
INFO - Iter 591/2000, Loss: 0.2465, Avg Adaptive LR: 3.037755
INFO - Iter 601/2000, Loss: 0.2462, Avg Adaptive LR: 3.037736
INFO - Iter 611/2000, Loss: 0.2458, Avg Adaptive LR: 3.037716
INFO - Iter 621/2000, Loss: 0.2455, Avg Adaptive LR: 3.037698
INFO - Iter 631/2000, Loss: 0.2452, Avg Adaptive LR: 3.037679
INFO - Iter 641/2000, Loss: 0.2449, Avg Adaptive LR: 3.037662
INFO - Iter 651/2000, Loss: 0.2446, Avg Adaptive LR: 3.037644
INFO - Iter 661/2000, Loss: 0.2443, Avg Adaptive LR: 3.037627
INFO - Iter 671/2000, Loss: 0.2440, Avg Adaptive LR: 3.037610
INFO - Iter 681/2000, Loss: 0.2437, Avg Adaptive LR: 3.037593
INFO - Iter 691/2000, Loss: 0.2434, Avg Adaptive LR: 3.037577
INFO - Iter 701/2000, Loss: 0.2432, Avg Adaptive LR: 3.037561
INFO - Iter 711/2000, Loss: 0.2429, Avg Adaptive LR: 3.037545
INFO - Iter 721/2000, Loss: 0.2426, Avg Adaptive LR: 3.037530
INFO - Iter 731/2000, Loss: 0.2424, Avg Adaptive LR: 3.037515
```

```
INFO - Iter 741/2000, Loss: 0.2421, Avg Adaptive LR: 3.037500
INFO - Iter 751/2000, Loss: 0.2419, Avg Adaptive LR: 3.037485
INFO - Iter 761/2000, Loss: 0.2416, Avg Adaptive LR: 3.037471
INFO - Iter 771/2000, Loss: 0.2414, Avg Adaptive LR: 3.037456
INFO - Iter 781/2000, Loss: 0.2412, Avg Adaptive LR: 3.037443
INFO - Iter 791/2000, Loss: 0.2409, Avg Adaptive LR: 3.037429
INFO - Iter 801/2000, Loss: 0.2407, Avg Adaptive LR: 3.037415
INFO - Iter 811/2000, Loss: 0.2405, Avg Adaptive LR: 3.037402
INFO - Iter 821/2000, Loss: 0.2402, Avg Adaptive LR: 3.037389
INFO - Iter 831/2000, Loss: 0.2400, Avg Adaptive LR: 3.037376
INFO - Iter 841/2000, Loss: 0.2398, Avg Adaptive LR: 3.037364
INFO - Iter 851/2000, Loss: 0.2396, Avg Adaptive LR: 3.037351
INFO - Iter 861/2000, Loss: 0.2394, Avg Adaptive LR: 3.037339
INFO - Iter 871/2000, Loss: 0.2392, Avg Adaptive LR: 3.037327
INFO - Iter 881/2000, Loss: 0.2390, Avg Adaptive LR: 3.037315
INFO - Iter 891/2000, Loss: 0.2388, Avg Adaptive LR: 3.037303
INFO - Iter 901/2000, Loss: 0.2386, Avg Adaptive LR: 3.037291
INFO - Iter 911/2000, Loss: 0.2384, Avg Adaptive LR: 3.037280
INFO - Iter 921/2000, Loss: 0.2382, Avg Adaptive LR: 3.037269
INFO - Iter 931/2000, Loss: 0.2380, Avg Adaptive LR: 3.037258
INFO - Iter 941/2000, Loss: 0.2378, Avg Adaptive LR: 3.037247
INFO - Iter 951/2000, Loss: 0.2376, Avg Adaptive LR: 3.037236
INFO - Iter 961/2000, Loss: 0.2375, Avg Adaptive LR: 3.037225
INFO - Iter 971/2000, Loss: 0.2373, Avg Adaptive LR: 3.037215
INFO - Iter 981/2000, Loss: 0.2371, Avg Adaptive LR: 3.037204
INFO - Iter 991/2000, Loss: 0.2369, Avg Adaptive LR: 3.037194
INFO - Iter 1001/2000, Loss: 0.2368, Avg Adaptive LR: 3.037184
INFO - Iter 1011/2000, Loss: 0.2366, Avg Adaptive LR: 3.037174
INFO - Iter 1021/2000, Loss: 0.2364, Avg Adaptive LR: 3.037164
INFO - Iter 1031/2000, Loss: 0.2362, Avg Adaptive LR: 3.037154
INFO - Iter 1041/2000, Loss: 0.2361, Avg Adaptive LR: 3.037145
INFO - Iter 1051/2000, Loss: 0.2359, Avg Adaptive LR: 3.037135
INFO - Iter 1061/2000, Loss: 0.2358, Avg Adaptive LR: 3.037126
INFO - Iter 1071/2000, Loss: 0.2356, Avg Adaptive LR: 3.037116
INFO - Iter 1081/2000, Loss: 0.2354, Avg Adaptive LR: 3.037107
INFO - Iter 1091/2000, Loss: 0.2353, Avg Adaptive LR: 3.037098
INFO - Iter 1101/2000, Loss: 0.2351, Avg Adaptive LR: 3.037089
INFO - Iter 1111/2000, Loss: 0.2350, Avg Adaptive LR: 3.037080
INFO - Iter 1121/2000, Loss: 0.2348, Avg Adaptive LR: 3.037071
INFO - Iter 1131/2000, Loss: 0.2347, Avg Adaptive LR: 3.037063
INFO - Iter 1141/2000, Loss: 0.2345, Avg Adaptive LR: 3.037054
INFO - Iter 1151/2000, Loss: 0.2344, Avg Adaptive LR: 3.037046
INFO - Iter 1161/2000, Loss: 0.2343, Avg Adaptive LR: 3.037037
INFO - Iter 1171/2000, Loss: 0.2341, Avg Adaptive LR: 3.037029
INFO - Iter 1181/2000, Loss: 0.2340, Avg Adaptive LR: 3.037021
INFO - Iter 1191/2000, Loss: 0.2338, Avg Adaptive LR: 3.037013
INFO - Iter 1201/2000, Loss: 0.2337, Avg Adaptive LR: 3.037005
INFO - Iter 1211/2000, Loss: 0.2336, Avg Adaptive LR: 3.036997
```

```
INFO - Iter 1221/2000, Loss: 0.2334, Avg Adaptive LR: 3.036989
INFO - Iter 1231/2000, Loss: 0.2333, Avg Adaptive LR: 3.036981
INFO - Iter 1241/2000, Loss: 0.2332, Avg Adaptive LR: 3.036973
INFO - Iter 1251/2000, Loss: 0.2330, Avg Adaptive LR: 3.036966
INFO - Iter 1261/2000, Loss: 0.2329, Avg Adaptive LR: 3.036958
INFO - Iter 1271/2000, Loss: 0.2328, Avg Adaptive LR: 3.036951
INFO - Iter 1281/2000, Loss: 0.2327, Avg Adaptive LR: 3.036943
INFO - Iter 1291/2000, Loss: 0.2325, Avg Adaptive LR: 3.036936
INFO - Iter 1301/2000, Loss: 0.2324, Avg Adaptive LR: 3.036929
INFO - Iter 1311/2000, Loss: 0.2323, Avg Adaptive LR: 3.036922
INFO - Iter 1321/2000, Loss: 0.2322, Avg Adaptive LR: 3.036915
INFO - Iter 1331/2000, Loss: 0.2320, Avg Adaptive LR: 3.036908
INFO - Iter 1341/2000, Loss: 0.2319, Avg Adaptive LR: 3.036901
INFO - Iter 1351/2000, Loss: 0.2318, Avg Adaptive LR: 3.036894
INFO - Iter 1361/2000, Loss: 0.2317, Avg Adaptive LR: 3.036887
INFO - Iter 1371/2000, Loss: 0.2316, Avg Adaptive LR: 3.036880
INFO - Iter 1381/2000, Loss: 0.2315, Avg Adaptive LR: 3.036873
INFO - Iter 1391/2000, Loss: 0.2313, Avg Adaptive LR: 3.036867
INFO - Iter 1401/2000, Loss: 0.2312, Avg Adaptive LR: 3.036860
INFO - Iter 1411/2000, Loss: 0.2311, Avg Adaptive LR: 3.036854
INFO - Iter 1421/2000, Loss: 0.2310, Avg Adaptive LR: 3.036847
INFO - Iter 1431/2000, Loss: 0.2309, Avg Adaptive LR: 3.036841
INFO - Iter 1441/2000, Loss: 0.2308, Avg Adaptive LR: 3.036834
INFO - Iter 1451/2000, Loss: 0.2307, Avg Adaptive LR: 3.036828
INFO - Iter 1461/2000, Loss: 0.2306, Avg Adaptive LR: 3.036822
INFO - Iter 1471/2000, Loss: 0.2305, Avg Adaptive LR: 3.036816
INFO - Iter 1481/2000, Loss: 0.2304, Avg Adaptive LR: 3.036810
INFO - Iter 1491/2000, Loss: 0.2303, Avg Adaptive LR: 3.036804
INFO - Iter 1501/2000, Loss: 0.2302, Avg Adaptive LR: 3.036798
INFO - Iter 1511/2000, Loss: 0.2301, Avg Adaptive LR: 3.036792
INFO - Iter 1521/2000, Loss: 0.2300, Avg Adaptive LR: 3.036786
INFO - Iter 1531/2000, Loss: 0.2299, Avg Adaptive LR: 3.036780
INFO - Iter 1541/2000, Loss: 0.2298, Avg Adaptive LR: 3.036774
INFO - Iter 1551/2000, Loss: 0.2297, Avg Adaptive LR: 3.036768
INFO - Iter 1561/2000, Loss: 0.2296, Avg Adaptive LR: 3.036763
INFO - Iter 1571/2000, Loss: 0.2295, Avg Adaptive LR: 3.036757
INFO - Iter 1581/2000, Loss: 0.2294, Avg Adaptive LR: 3.036751
INFO - Iter 1591/2000, Loss: 0.2293, Avg Adaptive LR: 3.036746
INFO - Iter 1601/2000, Loss: 0.2292, Avg Adaptive LR: 3.036740
INFO - Iter 1611/2000, Loss: 0.2291, Avg Adaptive LR: 3.036735
INFO - Iter 1621/2000, Loss: 0.2290, Avg Adaptive LR: 3.036729
INFO - Iter 1631/2000, Loss: 0.2289, Avg Adaptive LR: 3.036724
INFO - Iter 1641/2000, Loss: 0.2288, Avg Adaptive LR: 3.036718
INFO - Iter 1651/2000, Loss: 0.2287, Avg Adaptive LR: 3.036713
INFO - Iter 1661/2000, Loss: 0.2286, Avg Adaptive LR: 3.036708
INFO - Iter 1671/2000, Loss: 0.2286, Avg Adaptive LR: 3.036703
INFO - Iter 1681/2000, Loss: 0.2285, Avg Adaptive LR: 3.036697
INFO - Iter 1691/2000, Loss: 0.2284, Avg Adaptive LR: 3.036692
```

```
INFO - Iter 1701/2000, Loss: 0.2283, Avg Adaptive LR: 3.036687
INFO - Iter 1711/2000, Loss: 0.2282, Avg Adaptive LR: 3.036682
INFO - Iter 1721/2000, Loss: 0.2281, Avg Adaptive LR: 3.036677
INFO - Iter 1731/2000, Loss: 0.2280, Avg Adaptive LR: 3.036672
INFO - Iter 1741/2000, Loss: 0.2279, Avg Adaptive LR: 3.036667
INFO - Iter 1751/2000, Loss: 0.2279, Avg Adaptive LR: 3.036662
INFO - Iter 1761/2000, Loss: 0.2278, Avg Adaptive LR: 3.036657
INFO - Iter 1771/2000, Loss: 0.2277, Avg Adaptive LR: 3.036652
INFO - Iter 1781/2000, Loss: 0.2276, Avg Adaptive LR: 3.036648
INFO - Iter 1791/2000, Loss: 0.2275, Avg Adaptive LR: 3.036643
INFO - Iter 1801/2000, Loss: 0.2275, Avg Adaptive LR: 3.036638
INFO - Iter 1811/2000, Loss: 0.2274, Avg Adaptive LR: 3.036633
INFO - Iter 1821/2000, Loss: 0.2273, Avg Adaptive LR: 3.036629
INFO - Iter 1831/2000, Loss: 0.2272, Avg Adaptive LR: 3.036624
INFO - Iter 1841/2000, Loss: 0.2271, Avg Adaptive LR: 3.036619
INFO - Iter 1851/2000, Loss: 0.2271, Avg Adaptive LR: 3.036615
INFO - Iter 1861/2000, Loss: 0.2270, Avg Adaptive LR: 3.036610
INFO - Iter 1871/2000, Loss: 0.2269, Avg Adaptive LR: 3.036606
INFO - Iter 1881/2000, Loss: 0.2268, Avg Adaptive LR: 3.036601
INFO - Iter 1891/2000, Loss: 0.2268, Avg Adaptive LR: 3.036597
INFO - Iter 1901/2000, Loss: 0.2267, Avg Adaptive LR: 3.036593
INFO - Iter 1911/2000, Loss: 0.2266, Avg Adaptive LR: 3.036588
INFO - Iter 1921/2000, Loss: 0.2265, Avg Adaptive LR: 3.036584
INFO - Iter 1931/2000, Loss: 0.2265, Avg Adaptive LR: 3.036579
INFO - Iter 1941/2000, Loss: 0.2264, Avg Adaptive LR: 3.036575
INFO - Iter 1951/2000, Loss: 0.2263, Avg Adaptive LR: 3.036571
INFO - Iter 1961/2000, Loss: 0.2262, Avg Adaptive LR: 3.036567
INFO - Iter 1971/2000, Loss: 0.2262, Avg Adaptive LR: 3.036562
INFO - Iter 1981/2000, Loss: 0.2261, Avg Adaptive LR: 3.036558
INFO - Iter 1991/2000, Loss: 0.2260, Avg Adaptive LR: 3.036554
INFO - SoftmaxRegression training completed in 78.14 seconds.
INFO - --- Linear Regression LR=0.1/Iter=2000 ---
INFO - Iter 100/2000, Loss: 1.0308, Gradient Norm: 19.5337, Avg Adaptive LR:
1.3966674945235422
INFO - Iter 200/2000, Loss: 0.5402, Gradient Norm: 13.8895, Avg Adaptive LR:
0.9907905664429116
INFO - Iter 300/2000, Loss: 0.3321, Gradient Norm: 10.6174, Avg Adaptive LR:
0.8101783950153758
INFO - Iter 400/2000, Loss: 0.2244, Gradient Norm: 8.4339, Avg Adaptive LR:
0.7023652851444648
INFO - Iter 500/2000, Loss: 0.1658, Gradient Norm: 6.9645, Avg Adaptive LR:
0.6287117510518254
INFO - Iter 600/2000, Loss: 0.1309, Gradient Norm: 5.9195, Avg Adaptive LR:
0.5742803246572616
INFO - Iter 700/2000, Loss: 0.1086, Gradient Norm: 5.1425, Avg Adaptive LR:
0.5319345164557255
INFO - Iter 800/2000, Loss: 0.0936, Gradient Norm: 4.5472, Avg Adaptive LR:
0.49777954817742803
```

```
INFO - Iter 900/2000, Loss: 0.0830, Gradient Norm: 4.0733, Avg Adaptive LR:
0.46946861229196296
INFO - Iter 1000/2000, Loss: 0.0751, Gradient Norm: 3.6802, Avg Adaptive LR:
0.44550575086018046
INFO - Iter 1100/2000, Loss: 0.0692, Gradient Norm: 3.3597, Avg Adaptive LR:
0.4248789312457259
INFO - Iter 1200/2000, Loss: 0.0648, Gradient Norm: 3.1025, Avg Adaptive LR:
0.40688084231996935
INFO - Iter 1300/2000, Loss: 0.0612, Gradient Norm: 2.8705, Avg Adaptive LR:
0.39099588485459913
INFO - Iter 1400/2000, Loss: 0.0584, Gradient Norm: 2.6814, Avg Adaptive LR:
0.37683939248516457
INFO - Iter 1500/2000, Loss: 0.0562, Gradient Norm: 2.5198, Avg Adaptive LR:
0.36411917380693276
INFO - Iter 1600/2000, Loss: 0.0543, Gradient Norm: 2.3692, Avg Adaptive LR:
0.35260812985773154
INFO - Iter 1700/2000, Loss: 0.0528, Gradient Norm: 2.2507, Avg Adaptive LR:
0.3421256799671795
INFO - Iter 1800/2000, Loss: 0.0514, Gradient Norm: 2.1293, Avg Adaptive LR:
0.3325270426964418
INFO - Iter 1900/2000, Loss: 0.0504, Gradient Norm: 2.0389, Avg Adaptive LR:
0.3236949113667735
INFO - Iter 2000/2000, Loss: 0.0493, Gradient Norm: 1.9403, Avg Adaptive LR:
0.3155319454048128
INFO - LinearRegressionClassifier training completed in 64.18 seconds.
Train Regressions:  75%|      | 3/4 [02:32<01:08, 68.01s/it]INFO - ---
Softmax LR=0.1/Iter=3000 ---
INFO - Iter 1/3000, Loss: 2.4383, Avg Adaptive LR: 14.561848
INFO - Iter 11/3000, Loss: 0.4727, Avg Adaptive LR: 3.375636
INFO - Iter 21/3000, Loss: 0.3708, Avg Adaptive LR: 3.352319
INFO - Iter 31/3000, Loss: 0.3474, Avg Adaptive LR: 3.350694
INFO - Iter 41/3000, Loss: 0.3325, Avg Adaptive LR: 3.349654
INFO - Iter 51/3000, Loss: 0.3217, Avg Adaptive LR: 3.348898
INFO - Iter 61/3000, Loss: 0.3133, Avg Adaptive LR: 3.348315
INFO - Iter 71/3000, Loss: 0.3067, Avg Adaptive LR: 3.347846
INFO - Iter 81/3000, Loss: 0.3012, Avg Adaptive LR: 3.347458
INFO - Iter 91/3000, Loss: 0.2965, Avg Adaptive LR: 3.347129
INFO - Iter 101/3000, Loss: 0.2925, Avg Adaptive LR: 3.346847
INFO - Iter 111/3000, Loss: 0.2890, Avg Adaptive LR: 3.346600
INFO - Iter 121/3000, Loss: 0.2859, Avg Adaptive LR: 3.346382
INFO - Iter 131/3000, Loss: 0.2832, Avg Adaptive LR: 3.346188
INFO - Iter 141/3000, Loss: 0.2807, Avg Adaptive LR: 3.346014
INFO - Iter 151/3000, Loss: 0.2785, Avg Adaptive LR: 3.345855
INFO - Iter 161/3000, Loss: 0.2765, Avg Adaptive LR: 3.345711
INFO - Iter 171/3000, Loss: 0.2746, Avg Adaptive LR: 3.345579
INFO - Iter 181/3000, Loss: 0.2729, Avg Adaptive LR: 3.345457
INFO - Iter 191/3000, Loss: 0.2713, Avg Adaptive LR: 3.345344
INFO - Iter 201/3000, Loss: 0.2698, Avg Adaptive LR: 3.345239
```

```
INFO - Iter 211/3000, Loss: 0.2684, Avg Adaptive LR: 3.345141
INFO - Iter 221/3000, Loss: 0.2671, Avg Adaptive LR: 3.345049
INFO - Iter 231/3000, Loss: 0.2659, Avg Adaptive LR: 3.344963
INFO - Iter 241/3000, Loss: 0.2648, Avg Adaptive LR: 3.344882
INFO - Iter 251/3000, Loss: 0.2637, Avg Adaptive LR: 3.344806
INFO - Iter 261/3000, Loss: 0.2627, Avg Adaptive LR: 3.344733
INFO - Iter 271/3000, Loss: 0.2617, Avg Adaptive LR: 3.344665
INFO - Iter 281/3000, Loss: 0.2608, Avg Adaptive LR: 3.344599
INFO - Iter 291/3000, Loss: 0.2599, Avg Adaptive LR: 3.344537
INFO - Iter 301/3000, Loss: 0.2591, Avg Adaptive LR: 3.344478
INFO - Iter 311/3000, Loss: 0.2583, Avg Adaptive LR: 3.344421
INFO - Iter 321/3000, Loss: 0.2575, Avg Adaptive LR: 3.344367
INFO - Iter 331/3000, Loss: 0.2568, Avg Adaptive LR: 3.344315
INFO - Iter 341/3000, Loss: 0.2561, Avg Adaptive LR: 3.344265
INFO - Iter 351/3000, Loss: 0.2554, Avg Adaptive LR: 3.344217
INFO - Iter 361/3000, Loss: 0.2547, Avg Adaptive LR: 3.344171
INFO - Iter 371/3000, Loss: 0.2541, Avg Adaptive LR: 3.344127
INFO - Iter 381/3000, Loss: 0.2535, Avg Adaptive LR: 3.344084
INFO - Iter 391/3000, Loss: 0.2529, Avg Adaptive LR: 3.344043
INFO - Iter 401/3000, Loss: 0.2524, Avg Adaptive LR: 3.344003
INFO - Iter 411/3000, Loss: 0.2518, Avg Adaptive LR: 3.343965
INFO - Iter 421/3000, Loss: 0.2513, Avg Adaptive LR: 3.343927
INFO - Iter 431/3000, Loss: 0.2508, Avg Adaptive LR: 3.343891
INFO - Iter 441/3000, Loss: 0.2503, Avg Adaptive LR: 3.343856
INFO - Iter 451/3000, Loss: 0.2498, Avg Adaptive LR: 3.343823
INFO - Iter 461/3000, Loss: 0.2494, Avg Adaptive LR: 3.343790
INFO - Iter 471/3000, Loss: 0.2489, Avg Adaptive LR: 3.343758
INFO - Iter 481/3000, Loss: 0.2485, Avg Adaptive LR: 3.343727
INFO - Iter 491/3000, Loss: 0.2481, Avg Adaptive LR: 3.343696
INFO - Iter 501/3000, Loss: 0.2477, Avg Adaptive LR: 3.343667
INFO - Iter 511/3000, Loss: 0.2472, Avg Adaptive LR: 3.343639
INFO - Iter 521/3000, Loss: 0.2469, Avg Adaptive LR: 3.343611
INFO - Iter 531/3000, Loss: 0.2465, Avg Adaptive LR: 3.343584
INFO - Iter 541/3000, Loss: 0.2461, Avg Adaptive LR: 3.343557
INFO - Iter 551/3000, Loss: 0.2457, Avg Adaptive LR: 3.343531
INFO - Iter 561/3000, Loss: 0.2454, Avg Adaptive LR: 3.343506
INFO - Iter 571/3000, Loss: 0.2450, Avg Adaptive LR: 3.343481
INFO - Iter 581/3000, Loss: 0.2447, Avg Adaptive LR: 3.343457
INFO - Iter 591/3000, Loss: 0.2444, Avg Adaptive LR: 3.343434
INFO - Iter 601/3000, Loss: 0.2440, Avg Adaptive LR: 3.343411
INFO - Iter 611/3000, Loss: 0.2437, Avg Adaptive LR: 3.343388
INFO - Iter 621/3000, Loss: 0.2434, Avg Adaptive LR: 3.343366
INFO - Iter 631/3000, Loss: 0.2431, Avg Adaptive LR: 3.343345
INFO - Iter 641/3000, Loss: 0.2428, Avg Adaptive LR: 3.343324
INFO - Iter 651/3000, Loss: 0.2425, Avg Adaptive LR: 3.343303
INFO - Iter 661/3000, Loss: 0.2422, Avg Adaptive LR: 3.343283
INFO - Iter 671/3000, Loss: 0.2420, Avg Adaptive LR: 3.343263
INFO - Iter 681/3000, Loss: 0.2417, Avg Adaptive LR: 3.343244
```

```
INFO - Iter 691/3000, Loss: 0.2414, Avg Adaptive LR: 3.343225
INFO - Iter 701/3000, Loss: 0.2412, Avg Adaptive LR: 3.343206
INFO - Iter 711/3000, Loss: 0.2409, Avg Adaptive LR: 3.343187
INFO - Iter 721/3000, Loss: 0.2407, Avg Adaptive LR: 3.343169
INFO - Iter 731/3000, Loss: 0.2404, Avg Adaptive LR: 3.343152
INFO - Iter 741/3000, Loss: 0.2402, Avg Adaptive LR: 3.343134
INFO - Iter 751/3000, Loss: 0.2399, Avg Adaptive LR: 3.343117
INFO - Iter 761/3000, Loss: 0.2397, Avg Adaptive LR: 3.343100
INFO - Iter 771/3000, Loss: 0.2394, Avg Adaptive LR: 3.343084
INFO - Iter 781/3000, Loss: 0.2392, Avg Adaptive LR: 3.343068
INFO - Iter 791/3000, Loss: 0.2390, Avg Adaptive LR: 3.343052
INFO - Iter 801/3000, Loss: 0.2388, Avg Adaptive LR: 3.343036
INFO - Iter 811/3000, Loss: 0.2386, Avg Adaptive LR: 3.343021
INFO - Iter 821/3000, Loss: 0.2383, Avg Adaptive LR: 3.343005
INFO - Iter 831/3000, Loss: 0.2381, Avg Adaptive LR: 3.342990
INFO - Iter 841/3000, Loss: 0.2379, Avg Adaptive LR: 3.342976
INFO - Iter 851/3000, Loss: 0.2377, Avg Adaptive LR: 3.342961
INFO - Iter 861/3000, Loss: 0.2375, Avg Adaptive LR: 3.342947
INFO - Iter 871/3000, Loss: 0.2373, Avg Adaptive LR: 3.342933
INFO - Iter 881/3000, Loss: 0.2371, Avg Adaptive LR: 3.342919
INFO - Iter 891/3000, Loss: 0.2369, Avg Adaptive LR: 3.342905
INFO - Iter 901/3000, Loss: 0.2367, Avg Adaptive LR: 3.342892
INFO - Iter 911/3000, Loss: 0.2366, Avg Adaptive LR: 3.342879
INFO - Iter 921/3000, Loss: 0.2364, Avg Adaptive LR: 3.342865
INFO - Iter 931/3000, Loss: 0.2362, Avg Adaptive LR: 3.342853
INFO - Iter 941/3000, Loss: 0.2360, Avg Adaptive LR: 3.342840
INFO - Iter 951/3000, Loss: 0.2358, Avg Adaptive LR: 3.342827
INFO - Iter 961/3000, Loss: 0.2357, Avg Adaptive LR: 3.342815
INFO - Iter 971/3000, Loss: 0.2355, Avg Adaptive LR: 3.342803
INFO - Iter 981/3000, Loss: 0.2353, Avg Adaptive LR: 3.342791
INFO - Iter 991/3000, Loss: 0.2352, Avg Adaptive LR: 3.342779
INFO - Iter 1001/3000, Loss: 0.2350, Avg Adaptive LR: 3.342767
INFO - Iter 1011/3000, Loss: 0.2348, Avg Adaptive LR: 3.342755
INFO - Iter 1021/3000, Loss: 0.2347, Avg Adaptive LR: 3.342744
INFO - Iter 1031/3000, Loss: 0.2345, Avg Adaptive LR: 3.342733
INFO - Iter 1041/3000, Loss: 0.2344, Avg Adaptive LR: 3.342722
INFO - Iter 1051/3000, Loss: 0.2342, Avg Adaptive LR: 3.342711
INFO - Iter 1061/3000, Loss: 0.2340, Avg Adaptive LR: 3.342700
INFO - Iter 1071/3000, Loss: 0.2339, Avg Adaptive LR: 3.342689
INFO - Iter 1081/3000, Loss: 0.2337, Avg Adaptive LR: 3.342678
INFO - Iter 1091/3000, Loss: 0.2336, Avg Adaptive LR: 3.342668
INFO - Iter 1101/3000, Loss: 0.2334, Avg Adaptive LR: 3.342657
INFO - Iter 1111/3000, Loss: 0.2333, Avg Adaptive LR: 3.342647
INFO - Iter 1121/3000, Loss: 0.2332, Avg Adaptive LR: 3.342637
INFO - Iter 1131/3000, Loss: 0.2330, Avg Adaptive LR: 3.342627
INFO - Iter 1141/3000, Loss: 0.2329, Avg Adaptive LR: 3.342617
INFO - Iter 1151/3000, Loss: 0.2327, Avg Adaptive LR: 3.342607
INFO - Iter 1161/3000, Loss: 0.2326, Avg Adaptive LR: 3.342598
```

```
INFO - Iter 1171/3000, Loss: 0.2325, Avg Adaptive LR: 3.342588
INFO - Iter 1181/3000, Loss: 0.2323, Avg Adaptive LR: 3.342578
INFO - Iter 1191/3000, Loss: 0.2322, Avg Adaptive LR: 3.342569
INFO - Iter 1201/3000, Loss: 0.2321, Avg Adaptive LR: 3.342560
INFO - Iter 1211/3000, Loss: 0.2319, Avg Adaptive LR: 3.342551
INFO - Iter 1221/3000, Loss: 0.2318, Avg Adaptive LR: 3.342542
INFO - Iter 1231/3000, Loss: 0.2317, Avg Adaptive LR: 3.342533
INFO - Iter 1241/3000, Loss: 0.2316, Avg Adaptive LR: 3.342524
INFO - Iter 1251/3000, Loss: 0.2314, Avg Adaptive LR: 3.342515
INFO - Iter 1261/3000, Loss: 0.2313, Avg Adaptive LR: 3.342506
INFO - Iter 1271/3000, Loss: 0.2312, Avg Adaptive LR: 3.342498
INFO - Iter 1281/3000, Loss: 0.2311, Avg Adaptive LR: 3.342489
INFO - Iter 1291/3000, Loss: 0.2310, Avg Adaptive LR: 3.342481
INFO - Iter 1301/3000, Loss: 0.2308, Avg Adaptive LR: 3.342472
INFO - Iter 1311/3000, Loss: 0.2307, Avg Adaptive LR: 3.342464
INFO - Iter 1321/3000, Loss: 0.2306, Avg Adaptive LR: 3.342456
INFO - Iter 1331/3000, Loss: 0.2305, Avg Adaptive LR: 3.342448
INFO - Iter 1341/3000, Loss: 0.2304, Avg Adaptive LR: 3.342440
INFO - Iter 1351/3000, Loss: 0.2303, Avg Adaptive LR: 3.342432
INFO - Iter 1361/3000, Loss: 0.2302, Avg Adaptive LR: 3.342424
INFO - Iter 1371/3000, Loss: 0.2301, Avg Adaptive LR: 3.342416
INFO - Iter 1381/3000, Loss: 0.2299, Avg Adaptive LR: 3.342408
INFO - Iter 1391/3000, Loss: 0.2298, Avg Adaptive LR: 3.342400
INFO - Iter 1401/3000, Loss: 0.2297, Avg Adaptive LR: 3.342393
INFO - Iter 1411/3000, Loss: 0.2296, Avg Adaptive LR: 3.342385
INFO - Iter 1421/3000, Loss: 0.2295, Avg Adaptive LR: 3.342378
INFO - Iter 1431/3000, Loss: 0.2294, Avg Adaptive LR: 3.342370
INFO - Iter 1441/3000, Loss: 0.2293, Avg Adaptive LR: 3.342363
INFO - Iter 1451/3000, Loss: 0.2292, Avg Adaptive LR: 3.342356
INFO - Iter 1461/3000, Loss: 0.2291, Avg Adaptive LR: 3.342349
INFO - Iter 1471/3000, Loss: 0.2290, Avg Adaptive LR: 3.342341
INFO - Iter 1481/3000, Loss: 0.2289, Avg Adaptive LR: 3.342334
INFO - Iter 1491/3000, Loss: 0.2288, Avg Adaptive LR: 3.342327
INFO - Iter 1501/3000, Loss: 0.2287, Avg Adaptive LR: 3.342320
INFO - Iter 1511/3000, Loss: 0.2286, Avg Adaptive LR: 3.342314
INFO - Iter 1521/3000, Loss: 0.2285, Avg Adaptive LR: 3.342307
INFO - Iter 1531/3000, Loss: 0.2284, Avg Adaptive LR: 3.342300
INFO - Iter 1541/3000, Loss: 0.2283, Avg Adaptive LR: 3.342293
INFO - Iter 1551/3000, Loss: 0.2282, Avg Adaptive LR: 3.342287
INFO - Iter 1561/3000, Loss: 0.2281, Avg Adaptive LR: 3.342280
INFO - Iter 1571/3000, Loss: 0.2280, Avg Adaptive LR: 3.342273
INFO - Iter 1581/3000, Loss: 0.2280, Avg Adaptive LR: 3.342267
INFO - Iter 1591/3000, Loss: 0.2279, Avg Adaptive LR: 3.342260
INFO - Iter 1601/3000, Loss: 0.2278, Avg Adaptive LR: 3.342254
INFO - Iter 1611/3000, Loss: 0.2277, Avg Adaptive LR: 3.342248
INFO - Iter 1621/3000, Loss: 0.2276, Avg Adaptive LR: 3.342241
INFO - Iter 1631/3000, Loss: 0.2275, Avg Adaptive LR: 3.342235
INFO - Iter 1641/3000, Loss: 0.2274, Avg Adaptive LR: 3.342229
```

```
INFO - Iter 1651/3000, Loss: 0.2273, Avg Adaptive LR: 3.342223
INFO - Iter 1661/3000, Loss: 0.2273, Avg Adaptive LR: 3.342217
INFO - Iter 1671/3000, Loss: 0.2272, Avg Adaptive LR: 3.342211
INFO - Iter 1681/3000, Loss: 0.2271, Avg Adaptive LR: 3.342205
INFO - Iter 1691/3000, Loss: 0.2270, Avg Adaptive LR: 3.342199
INFO - Iter 1701/3000, Loss: 0.2269, Avg Adaptive LR: 3.342193
INFO - Iter 1711/3000, Loss: 0.2268, Avg Adaptive LR: 3.342187
INFO - Iter 1721/3000, Loss: 0.2267, Avg Adaptive LR: 3.342181
INFO - Iter 1731/3000, Loss: 0.2267, Avg Adaptive LR: 3.342175
INFO - Iter 1741/3000, Loss: 0.2266, Avg Adaptive LR: 3.342170
INFO - Iter 1751/3000, Loss: 0.2265, Avg Adaptive LR: 3.342164
INFO - Iter 1761/3000, Loss: 0.2264, Avg Adaptive LR: 3.342158
INFO - Iter 1771/3000, Loss: 0.2263, Avg Adaptive LR: 3.342153
INFO - Iter 1781/3000, Loss: 0.2263, Avg Adaptive LR: 3.342147
INFO - Iter 1791/3000, Loss: 0.2262, Avg Adaptive LR: 3.342141
INFO - Iter 1801/3000, Loss: 0.2261, Avg Adaptive LR: 3.342136
INFO - Iter 1811/3000, Loss: 0.2260, Avg Adaptive LR: 3.342131
INFO - Iter 1821/3000, Loss: 0.2260, Avg Adaptive LR: 3.342125
INFO - Iter 1831/3000, Loss: 0.2259, Avg Adaptive LR: 3.342120
INFO - Iter 1841/3000, Loss: 0.2258, Avg Adaptive LR: 3.342114
INFO - Iter 1851/3000, Loss: 0.2257, Avg Adaptive LR: 3.342109
INFO - Iter 1861/3000, Loss: 0.2257, Avg Adaptive LR: 3.342104
INFO - Iter 1871/3000, Loss: 0.2256, Avg Adaptive LR: 3.342099
INFO - Iter 1881/3000, Loss: 0.2255, Avg Adaptive LR: 3.342093
INFO - Iter 1891/3000, Loss: 0.2254, Avg Adaptive LR: 3.342088
INFO - Iter 1901/3000, Loss: 0.2254, Avg Adaptive LR: 3.342083
INFO - Iter 1911/3000, Loss: 0.2253, Avg Adaptive LR: 3.342078
INFO - Iter 1921/3000, Loss: 0.2252, Avg Adaptive LR: 3.342073
INFO - Iter 1931/3000, Loss: 0.2252, Avg Adaptive LR: 3.342068
INFO - Iter 1941/3000, Loss: 0.2251, Avg Adaptive LR: 3.342063
INFO - Iter 1951/3000, Loss: 0.2250, Avg Adaptive LR: 3.342058
INFO - Iter 1961/3000, Loss: 0.2250, Avg Adaptive LR: 3.342053
INFO - Iter 1971/3000, Loss: 0.2249, Avg Adaptive LR: 3.342048
INFO - Iter 1981/3000, Loss: 0.2248, Avg Adaptive LR: 3.342043
INFO - Iter 1991/3000, Loss: 0.2247, Avg Adaptive LR: 3.342039
INFO - Iter 2001/3000, Loss: 0.2247, Avg Adaptive LR: 3.342034
INFO - Iter 2011/3000, Loss: 0.2246, Avg Adaptive LR: 3.342029
INFO - Iter 2021/3000, Loss: 0.2245, Avg Adaptive LR: 3.342024
INFO - Iter 2031/3000, Loss: 0.2245, Avg Adaptive LR: 3.342020
INFO - Iter 2041/3000, Loss: 0.2244, Avg Adaptive LR: 3.342015
INFO - Iter 2051/3000, Loss: 0.2244, Avg Adaptive LR: 3.342010
INFO - Iter 2061/3000, Loss: 0.2243, Avg Adaptive LR: 3.342006
INFO - Iter 2071/3000, Loss: 0.2242, Avg Adaptive LR: 3.342001
INFO - Iter 2081/3000, Loss: 0.2242, Avg Adaptive LR: 3.341997
INFO - Iter 2091/3000, Loss: 0.2241, Avg Adaptive LR: 3.341992
INFO - Iter 2101/3000, Loss: 0.2240, Avg Adaptive LR: 3.341988
INFO - Iter 2111/3000, Loss: 0.2240, Avg Adaptive LR: 3.341983
INFO - Iter 2121/3000, Loss: 0.2239, Avg Adaptive LR: 3.341979
```

```
INFO - Iter 2131/3000, Loss: 0.2238, Avg Adaptive LR: 3.341974
INFO - Iter 2141/3000, Loss: 0.2238, Avg Adaptive LR: 3.341970
INFO - Iter 2151/3000, Loss: 0.2237, Avg Adaptive LR: 3.341966
INFO - Iter 2161/3000, Loss: 0.2237, Avg Adaptive LR: 3.341961
INFO - Iter 2171/3000, Loss: 0.2236, Avg Adaptive LR: 3.341957
INFO - Iter 2181/3000, Loss: 0.2235, Avg Adaptive LR: 3.341953
INFO - Iter 2191/3000, Loss: 0.2235, Avg Adaptive LR: 3.341949
INFO - Iter 2201/3000, Loss: 0.2234, Avg Adaptive LR: 3.341944
INFO - Iter 2211/3000, Loss: 0.2234, Avg Adaptive LR: 3.341940
INFO - Iter 2221/3000, Loss: 0.2233, Avg Adaptive LR: 3.341936
INFO - Iter 2231/3000, Loss: 0.2232, Avg Adaptive LR: 3.341932
INFO - Iter 2241/3000, Loss: 0.2232, Avg Adaptive LR: 3.341928
INFO - Iter 2251/3000, Loss: 0.2231, Avg Adaptive LR: 3.341924
INFO - Iter 2261/3000, Loss: 0.2231, Avg Adaptive LR: 3.341920
INFO - Iter 2271/3000, Loss: 0.2230, Avg Adaptive LR: 3.341916
INFO - Iter 2281/3000, Loss: 0.2230, Avg Adaptive LR: 3.341912
INFO - Iter 2291/3000, Loss: 0.2229, Avg Adaptive LR: 3.341908
INFO - Iter 2301/3000, Loss: 0.2228, Avg Adaptive LR: 3.341904
INFO - Iter 2311/3000, Loss: 0.2228, Avg Adaptive LR: 3.341900
INFO - Iter 2321/3000, Loss: 0.2227, Avg Adaptive LR: 3.341896
INFO - Iter 2331/3000, Loss: 0.2227, Avg Adaptive LR: 3.341892
INFO - Iter 2341/3000, Loss: 0.2226, Avg Adaptive LR: 3.341888
INFO - Iter 2351/3000, Loss: 0.2226, Avg Adaptive LR: 3.341884
INFO - Iter 2361/3000, Loss: 0.2225, Avg Adaptive LR: 3.341880
INFO - Iter 2371/3000, Loss: 0.2225, Avg Adaptive LR: 3.341876
INFO - Iter 2381/3000, Loss: 0.2224, Avg Adaptive LR: 3.341873
INFO - Iter 2391/3000, Loss: 0.2224, Avg Adaptive LR: 3.341869
INFO - Iter 2401/3000, Loss: 0.2223, Avg Adaptive LR: 3.341865
INFO - Iter 2411/3000, Loss: 0.2223, Avg Adaptive LR: 3.341861
INFO - Iter 2421/3000, Loss: 0.2222, Avg Adaptive LR: 3.341858
INFO - Iter 2431/3000, Loss: 0.2222, Avg Adaptive LR: 3.341854
INFO - Iter 2441/3000, Loss: 0.2221, Avg Adaptive LR: 3.341850
INFO - Iter 2451/3000, Loss: 0.2220, Avg Adaptive LR: 3.341847
INFO - Iter 2461/3000, Loss: 0.2220, Avg Adaptive LR: 3.341843
INFO - Iter 2471/3000, Loss: 0.2219, Avg Adaptive LR: 3.341839
INFO - Iter 2481/3000, Loss: 0.2219, Avg Adaptive LR: 3.341836
INFO - Iter 2491/3000, Loss: 0.2218, Avg Adaptive LR: 3.341832
INFO - Iter 2501/3000, Loss: 0.2218, Avg Adaptive LR: 3.341829
INFO - Iter 2511/3000, Loss: 0.2217, Avg Adaptive LR: 3.341825
INFO - Iter 2521/3000, Loss: 0.2217, Avg Adaptive LR: 3.341822
INFO - Iter 2531/3000, Loss: 0.2216, Avg Adaptive LR: 3.341818
INFO - Iter 2541/3000, Loss: 0.2216, Avg Adaptive LR: 3.341815
INFO - Iter 2551/3000, Loss: 0.2215, Avg Adaptive LR: 3.341811
INFO - Iter 2561/3000, Loss: 0.2215, Avg Adaptive LR: 3.341808
INFO - Iter 2571/3000, Loss: 0.2215, Avg Adaptive LR: 3.341804
INFO - Iter 2581/3000, Loss: 0.2214, Avg Adaptive LR: 3.341801
INFO - Iter 2591/3000, Loss: 0.2214, Avg Adaptive LR: 3.341798
INFO - Iter 2601/3000, Loss: 0.2213, Avg Adaptive LR: 3.341794
```

```
INFO - Iter 2611/3000, Loss: 0.2213, Avg Adaptive LR: 3.341791
INFO - Iter 2621/3000, Loss: 0.2212, Avg Adaptive LR: 3.341788
INFO - Iter 2631/3000, Loss: 0.2212, Avg Adaptive LR: 3.341784
INFO - Iter 2641/3000, Loss: 0.2211, Avg Adaptive LR: 3.341781
INFO - Iter 2651/3000, Loss: 0.2211, Avg Adaptive LR: 3.341778
INFO - Iter 2661/3000, Loss: 0.2210, Avg Adaptive LR: 3.341774
INFO - Iter 2671/3000, Loss: 0.2210, Avg Adaptive LR: 3.341771
INFO - Iter 2681/3000, Loss: 0.2209, Avg Adaptive LR: 3.341768
INFO - Iter 2691/3000, Loss: 0.2209, Avg Adaptive LR: 3.341765
INFO - Iter 2701/3000, Loss: 0.2208, Avg Adaptive LR: 3.341761
INFO - Iter 2711/3000, Loss: 0.2208, Avg Adaptive LR: 3.341758
INFO - Iter 2721/3000, Loss: 0.2208, Avg Adaptive LR: 3.341755
INFO - Iter 2731/3000, Loss: 0.2207, Avg Adaptive LR: 3.341752
INFO - Iter 2741/3000, Loss: 0.2207, Avg Adaptive LR: 3.341749
INFO - Iter 2751/3000, Loss: 0.2206, Avg Adaptive LR: 3.341746
INFO - Iter 2761/3000, Loss: 0.2206, Avg Adaptive LR: 3.341743
INFO - Iter 2771/3000, Loss: 0.2205, Avg Adaptive LR: 3.341740
INFO - Iter 2781/3000, Loss: 0.2205, Avg Adaptive LR: 3.341736
INFO - Iter 2791/3000, Loss: 0.2205, Avg Adaptive LR: 3.341733
INFO - Iter 2801/3000, Loss: 0.2204, Avg Adaptive LR: 3.341730
INFO - Iter 2811/3000, Loss: 0.2204, Avg Adaptive LR: 3.341727
INFO - Iter 2821/3000, Loss: 0.2203, Avg Adaptive LR: 3.341724
INFO - Iter 2831/3000, Loss: 0.2203, Avg Adaptive LR: 3.341721
INFO - Iter 2841/3000, Loss: 0.2202, Avg Adaptive LR: 3.341718
INFO - Iter 2851/3000, Loss: 0.2202, Avg Adaptive LR: 3.341715
INFO - Iter 2861/3000, Loss: 0.2202, Avg Adaptive LR: 3.341712
INFO - Iter 2871/3000, Loss: 0.2201, Avg Adaptive LR: 3.341709
INFO - Iter 2881/3000, Loss: 0.2201, Avg Adaptive LR: 3.341706
INFO - Iter 2891/3000, Loss: 0.2200, Avg Adaptive LR: 3.341704
INFO - Iter 2901/3000, Loss: 0.2200, Avg Adaptive LR: 3.341701
INFO - Iter 2911/3000, Loss: 0.2200, Avg Adaptive LR: 3.341698
INFO - Iter 2921/3000, Loss: 0.2199, Avg Adaptive LR: 3.341695
INFO - Iter 2931/3000, Loss: 0.2199, Avg Adaptive LR: 3.341692
INFO - Iter 2941/3000, Loss: 0.2198, Avg Adaptive LR: 3.341689
INFO - Iter 2951/3000, Loss: 0.2198, Avg Adaptive LR: 3.341686
INFO - Iter 2961/3000, Loss: 0.2198, Avg Adaptive LR: 3.341684
INFO - Iter 2971/3000, Loss: 0.2197, Avg Adaptive LR: 3.341681
INFO - Iter 2981/3000, Loss: 0.2197, Avg Adaptive LR: 3.341678
INFO - Iter 2991/3000, Loss: 0.2196, Avg Adaptive LR: 3.341675
INFO - SoftmaxRegression training completed in 117.53 seconds.
INFO - --- Linear Regression LR=0.1/Iter=3000 ---
INFO - Iter 100/3000, Loss: 0.7045, Gradient Norm: 15.9743, Avg Adaptive LR:
1.3979463815447688
INFO - Iter 200/3000, Loss: 0.3775, Gradient Norm: 11.3865, Avg Adaptive LR:
0.9918208472885416
INFO - Iter 300/3000, Loss: 0.2592, Gradient Norm: 9.1784, Avg Adaptive LR:
0.8110873012350543
INFO - Iter 400/3000, Loss: 0.1986, Gradient Norm: 7.8071, Avg Adaptive LR:
```

```
0.703104381147389
INFO - Iter 500/3000, Loss: 0.1627, Gradient Norm: 6.8673, Avg Adaptive LR:
0.6292966829840383
INFO - Iter 600/3000, Loss: 0.1385, Gradient Norm: 6.1567, Avg Adaptive LR:
0.5747538485278758
INFO - Iter 700/3000, Loss: 0.1213, Gradient Norm: 5.5941, Avg Adaptive LR:
0.5323208681026976
INFO - Iter 800/3000, Loss: 0.1090, Gradient Norm: 5.1562, Avg Adaptive LR:
0.49810426582336054
INFO - Iter 900/3000, Loss: 0.0996, Gradient Norm: 4.7965, Avg Adaptive LR:
0.4697292357526956
INFO - Iter 1000/3000, Loss: 0.0924, Gradient Norm: 4.4997, Avg Adaptive LR:
0.44573554216118855
INFO - Iter 1100/3000, Loss: 0.0864, Gradient Norm: 4.2390, Avg Adaptive LR:
0.42507792087200574
INFO - Iter 1200/3000, Loss: 0.0821, Gradient Norm: 4.0369, Avg Adaptive LR:
0.4070402974506587
INFO - Iter 1300/3000, Loss: 0.0772, Gradient Norm: 3.8025, Avg Adaptive LR:
0.3911407209984137
INFO - Iter 1400/3000, Loss: 0.0734, Gradient Norm: 3.6049, Avg Adaptive LR:
0.3769569667387786
INFO - Iter 1500/3000, Loss: 0.0700, Gradient Norm: 3.4213, Avg Adaptive LR:
0.36421275399835285
INFO - Iter 1600/3000, Loss: 0.0673, Gradient Norm: 3.2648, Avg Adaptive LR:
0.3526889621450347
INFO - Iter 1700/3000, Loss: 0.0649, Gradient Norm: 3.1215, Avg Adaptive LR:
0.34218627650098443
INFO - Iter 1800/3000, Loss: 0.0628, Gradient Norm: 2.9903, Avg Adaptive LR:
0.3325771051971342
INFO - Iter 1900/3000, Loss: 0.0611, Gradient Norm: 2.8801, Avg Adaptive LR:
0.32373194838511077
INFO - Iter 2000/3000, Loss: 0.0596, Gradient Norm: 2.7823, Avg Adaptive LR:
0.31555907201227723
INFO - Iter 2100/3000, Loss: 0.0582, Gradient Norm: 2.6796, Avg Adaptive LR:
0.3079771645118265
INFO - Iter 2200/3000, Loss: 0.0572, Gradient Norm: 2.6108, Avg Adaptive LR:
0.3009182380646709
INFO - Iter 2300/3000, Loss: 0.0559, Gradient Norm: 2.5158, Avg Adaptive LR:
0.29432099415070795
INFO - Iter 2400/3000, Loss: 0.0551, Gradient Norm: 2.4538, Avg Adaptive LR:
0.288142205816417
INFO - Iter 2500/3000, Loss: 0.0541, Gradient Norm: 2.3810, Avg Adaptive LR:
0.2823368335314519
INFO - Iter 2600/3000, Loss: 0.0534, Gradient Norm: 2.3190, Avg Adaptive LR:
0.276870984682765
INFO - Iter 2700/3000, Loss: 0.0527, Gradient Norm: 2.2656, Avg Adaptive LR:
0.2717090318463083
INFO - Iter 2800/3000, Loss: 0.0521, Gradient Norm: 2.2103, Avg Adaptive LR:
```

```
0.26682415233196355
INFO - Iter 2900/3000, Loss: 0.0516, Gradient Norm: 2.1659, Avg Adaptive LR:
0.26219582309900336
INFO - Iter 3000/3000, Loss: 0.0510, Gradient Norm: 2.1225, Avg Adaptive LR:
0.25780108446018046
INFO - LinearRegressionClassifier training completed in 97.68 seconds.
Train Regressions: 100%|      | 4/4 [06:08<00:00, 92.02s/it]
INFO - Training complete for Softmax and Linear.
INFO - === TRAINING PERCEPTRON MODELS (Clean & Pocket) ===
Train Clean & Pocket:   0%|          | 0/4 [00:00<?, ?it/s]INFO - --- Clean PLA,
max_iter=50 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
INFO - --- Pocket PLA, max_iter=50 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
Train Clean & Pocket:  25%|        | 1/4 [00:47<02:22, 47.51s/it]INFO - ---
Clean PLA, max_iter=100 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
INFO - --- Pocket PLA, max_iter=100 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
```

```
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
Train Clean & Pocket:  50%|        | 2/4 [02:10<02:17, 68.52s/it]INFO - ---
Clean PLA, max_iter=2000 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
INFO - --- Pocket PLA, max_iter=2000 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
Train Clean & Pocket:  75%|      | 3/4 [24:05<10:37, 637.44s/it]INFO - ---
Clean PLA, max_iter=3000 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
INFO - --- Pocket PLA, max_iter=3000 ---
INFO - Training for digit 0…
INFO - Training for digit 1…
INFO - Training for digit 2…
INFO - Training for digit 3…
INFO - Training for digit 4…
```

```
INFO - Training for digit 5…
INFO - Training for digit 6…
INFO - Training for digit 7…
INFO - Training for digit 8…
INFO - Training for digit 9…
```

6 Evaluate

[ ]:
```
##################################################
# EVALUATION CELL (with pandas DataFrame)
##################################################


# 1) Evaluate Perceptrons: Clean & Pocket
accuracies_clean, accuracies_pocket = [], []
runtimes_clean,   runtimes_pocket   = [], []
sensitivities_clean, sensitivities_pocket = [], []
selectivities_clean, selectivities_pocket = [], []

conf_clean, conf_pocket = [], []
meta_clean, meta_pocket = [], []

for max_iter in tqdm(perceptron_max_iter_values, desc="Evaluate Clean &␣
 ↪Pocket"):
    # === Evaluate Clean PLA ===
    c_model = trained_models_clean[max_iter]
    cm_c, acc_c, s_c, sp_c, rt_c, ex_c = evaluate_model(
        c_model, X_test, y_test, classes=range(10), model_name="Clean PLA"
    )
    accuracies_clean.append(acc_c)
    runtimes_clean.append(rt_c)
    sensitivities_clean.append(np.mean(s_c))
    selectivities_clean.append(np.mean(sp_c))
    conf_clean.append(cm_c)

    cdict = {
        "max_iter": max_iter,
        "accuracy": acc_c,
        "runtime": rt_c,
        "avg_sensitivity": np.mean(s_c),
        "avg_selectivity": np.mean(sp_c),
        "method": "Clean PLA"
    }
    cdict.update(ex_c)
    meta_clean.append(cdict)

    # === Evaluate Pocket PLA ===
```

```
    p_model = trained_models_pocket[max_iter]
    cm_p, acc_p, s_p, sp_p, rt_p, ex_p = evaluate_model(
        p_model, X_test, y_test, classes=range(10), model_name="Pocket PLA"
    )
    accuracies_pocket.append(acc_p)
    runtimes_pocket.append(rt_p)
    sensitivities_pocket.append(np.mean(s_p))
    selectivities_pocket.append(np.mean(sp_p))
    conf_pocket.append(cm_p)

    pdict = {
        "max_iter": max_iter,
        "accuracy": acc_p,
        "runtime": rt_p,
        "avg_sensitivity": np.mean(s_p),
        "avg_selectivity": np.mean(sp_p),
        "method": "Pocket PLA"
    }
    pdict.update(ex_p)
    meta_pocket.append(pdict)

# Aggregated iteration-level training curves for Perceptrons
clean_train_curve = aggregate_iteration_losses(
    [trained_models_clean[m] for m in perceptron_max_iter_values]
)
pocket_train_curve = aggregate_iteration_losses(
    [trained_models_pocket[m] for m in perceptron_max_iter_values]
)

# 2) Evaluate Regression Models: Softmax & Linear
accuracies_softmax = []
runtimes_softmax   = []
sensitivities_soft = []
selectivities_soft = []
conf_soft          = []
meta_soft          = []

accuracies_linear = []
runtimes_linear   = []
sensitivities_lin = []
selectivities_lin = []
conf_linear       = []
meta_linear       = []

for cfg in tqdm(regression_run_configs, desc="Evaluate Regressions"):
    lr_val = cfg["learning_rate"]
    max_iter_val = cfg["max_iter"]
```

```
    label = cfg["label"]

    # === Evaluate Softmax ===
    s_model = trained_models_softmax[(lr_val, max_iter_val)]
    cm_s, a_s, se_s, sp_s, r_s, ex_s = evaluate_model(
        s_model, X_test, y_test, classes=range(10),
        model_name=f"Softmax ({label})"
    )
    accuracies_softmax.append(a_s)
    runtimes_softmax.append(r_s)
    sensitivities_soft.append(np.mean(se_s))
    selectivities_soft.append(np.mean(sp_s))
    conf_soft.append(cm_s)

    ms = {
        "label": label,
        "learning_rate": lr_val,
        "max_iter": max_iter_val,
        "accuracy": a_s,
        "runtime": r_s,
        "avg_sensitivity": np.mean(se_s),
        "avg_selectivity": np.mean(sp_s),
        "method": "Softmax"
    }
    ms.update(ex_s)
    meta_soft.append(ms)

    # === Evaluate Linear ===
    lin_model = trained_models_linear[(lr_val, max_iter_val)]
    cm_l, a_l, se_l, sp_l, r_l, ex_l = evaluate_model(
        lin_model, X_test, y_test, classes=range(10),
        model_name=f"Linear ({label})"
    )
    accuracies_linear.append(a_l)
    runtimes_linear.append(r_l)
    sensitivities_lin.append(np.mean(se_l))
    selectivities_lin.append(np.mean(sp_l))
    conf_linear.append(cm_l)

    ml = {
        "label": label,
        "learning_rate": lr_val,
        "max_iter": max_iter_val,
        "accuracy": a_l,
        "runtime": r_l,
        "avg_sensitivity": np.mean(se_l),
        "avg_selectivity": np.mean(sp_l),
```

```
        "method": "Linear Regression"
    }
    ml.update(ex_l)
    meta_linear.append(ml)


logger.info("Evaluation complete for Perceptrons & Regressions.")
```

7 Visualize (Generate Plots, Confusion Matricies, etc.)

[ ]:
```
#####################################################
# 1) CREATE A SINGLE PANDAS DATAFRAME FOR ALL RESULTS
#####################################################
all_rows = []

# (A) Clean PLA
for i, max_iter in tqdm(
    enumerate(perceptron_max_iter_values),
    desc="Collecting Clean PLA",
    total=len(perceptron_max_iter_values)
):
    all_rows.append({
        'model': 'Clean PLA',
        'max_iter': max_iter,
        'runtime': runtimes_clean[i],
        'accuracy': accuracies_clean[i],
        'sensitivity': sensitivities_clean[i],
        'selectivity': selectivities_clean[i]
    })

# (B) Pocket PLA
for i, max_iter in tqdm(
    enumerate(perceptron_max_iter_values),
    desc="Collecting Pocket PLA",
    total=len(perceptron_max_iter_values)
):
    all_rows.append({
        'model': 'Pocket PLA',
        'max_iter': max_iter,
        'runtime': runtimes_pocket[i],
        'accuracy': accuracies_pocket[i],
        'sensitivity': sensitivities_pocket[i],
        'selectivity': selectivities_pocket[i]
    })

# (C) Softmax
```

```
for i, row_meta in tqdm(
    enumerate(meta_soft),
    desc="Collecting Softmax",
    total=len(meta_soft)
):
    all_rows.append({
        'model': 'Softmax',
        'max_iter': row_meta['max_iter'],
        'runtime': runtimes_softmax[i],
        'accuracy': accuracies_softmax[i],
        'sensitivity': sensitivities_soft[i],
        'selectivity': selectivities_soft[i]
    })

# (D) Linear
for i, row_meta in tqdm(
    enumerate(meta_linear),
    desc="Collecting Linear",
    total=len(meta_linear)
):
    all_rows.append({
        'model': 'Linear',
        'max_iter': row_meta['max_iter'],
        'runtime': runtimes_linear[i],
        'accuracy': accuracies_linear[i],
        'sensitivity': sensitivities_lin[i],
        'selectivity': selectivities_lin[i]
    })

df_results = pd.DataFrame(all_rows)
logger.info("Combined Results DataFrame:\n%s", df_results)
display(df_results.head(20))

##############################################################################
# 2) CONFUSION MATRICES FOR ALL MODELS (GROUPED BY PLOT TYPE)
##############################################################################

logger.info("=== Plotting ALL Confusion Matrices ===")

# 2A) Perceptron: Clean
for idx, meta in tqdm(enumerate(meta_clean), total=len(meta_clean),␣
 ↪desc="Confusions: Clean PLA"):
    title = f"Clean PLA (max_iter={meta['max_iter']}, Acc={meta['accuracy']*100:
 ↪.2f}%)"
    plot_confusion_matrix_annotated(
        conf_clean[idx],
        classes=range(10),
```

```
        title=title,
        method=meta["method"],
        max_iter=meta["max_iter"]
    )

# 2B) Perceptron: Pocket
for idx, meta in tqdm(enumerate(meta_pocket), total=len(meta_pocket),␣
 ↪desc="Confusions: Pocket PLA"):
    title = f"Pocket PLA (max_iter={meta['max_iter']},␣
 ↪Acc={meta['accuracy']*100:.2f}%)"
    plot_confusion_matrix_annotated(
        conf_pocket[idx],
        classes=range(10),
        title=title,
        method=meta["method"],
        max_iter=meta["max_iter"]
    )

# 2C) Softmax
for idx, meta in tqdm(enumerate(meta_soft), total=len(meta_soft),␣
 ↪desc="Confusions: Softmax"):
    title = f"Softmax ({meta['label']}, Acc={meta['accuracy']*100:.2f}%)"
    plot_confusion_matrix_annotated(
        conf_soft[idx],
        classes=range(10),
        title=title,
        method=meta["method"],
        max_iter=meta["max_iter"]
    )

# 2D) Linear
for idx, meta in tqdm(enumerate(meta_linear), total=len(meta_linear),␣
 ↪desc="Confusions: Linear"):
    title = f"Linear ({meta['label']}, Acc={meta['accuracy']*100:.2f}%)"
    plot_confusion_matrix_annotated(
        conf_linear[idx],
        classes=range(10),
        title=title,
        method=meta["method"],
        max_iter=meta["max_iter"]
    )


################################################################################
# 3) ITERATION-LEVEL PLOTS (ALL MODELS)
################################################################################
```

```
logger.info("=== Iteration-Level Visualization (All Models) ===")

# 3A) Perceptron: Clean & Pocket
for max_iter, c_model in trained_models_clean.items():
    df_iter = c_model.get_iteration_df()
    if not df_iter.empty and "train_error" in df_iter.columns:
        title = f"Clean PLA max_iter={max_iter}: Train Error vs. Iteration"
        df_iter.plot(x="iteration", y="train_error", marker='o', figsize=(8,5),␣
 ↪title=title)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.show()

for max_iter, p_model in trained_models_pocket.items():
    df_iter = p_model.get_iteration_df()
    if not df_iter.empty and "train_error" in df_iter.columns:
        title = f"Pocket PLA max_iter={max_iter}: Train Error vs. Iteration"
        df_iter.plot(x="iteration", y="train_error", marker='o', figsize=(8,5),␣
 ↪title=title)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.show()

# 3B) Softmax
for (lr_val, max_iter_val), s_model in trained_models_softmax.items():
    df_iter = s_model.get_iteration_df()  # Must be implemented in your␣
 ↪SoftmaxRegression
    if not df_iter.empty:
        title = f"Softmax LR={lr_val}, max_iter={max_iter_val}: Train Loss vs.␣
 ↪Iteration"
        df_iter.plot(x="iteration", y="train_loss", marker='o', figsize=(8,5),␣
 ↪title=title)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.show()

        if "test_loss" in df_iter.columns:
            title = f"Softmax LR={lr_val}, max_iter={max_iter_val}: Train &␣
 ↪Test Loss"
            df_iter.plot(x="iteration", y=["train_loss","test_loss"],␣
 ↪marker='o', figsize=(8,5), title=title)
            plt.grid(True, linestyle='--', alpha=0.7)
            plt.show()

        if "avg_adaptive_lr" in df_iter.columns:
            title = f"Softmax LR={lr_val}, max_iter={max_iter_val}: Avg␣
 ↪Adaptive LR vs. Iteration"
            df_iter.plot(x="iteration", y="avg_adaptive_lr", marker='x',␣
 ↪figsize=(8,5), title=title)
```

```
            plt.grid(True, linestyle='--', alpha=0.7)
            plt.show()

# 3C) Linear
for (lr_val, max_iter_val), lin_model in trained_models_linear.items():
    df_iter = lin_model.get_iteration_df()  # Must be implemented in your␣
 ↪LinearRegression
    if not df_iter.empty:
        title = f"Linear LR={lr_val}, max_iter={max_iter_val}: Train Loss vs.␣
 ↪Iteration"
        df_iter.plot(x="iteration", y="train_loss", marker='o', figsize=(8,5),␣
 ↪title=title)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.show()

        if "test_loss" in df_iter.columns:
            title = f"Linear LR={lr_val}, max_iter={max_iter_val}: Train & Test␣
 ↪Loss"
            df_iter.plot(x="iteration", y=["train_loss","test_loss"],␣
 ↪marker='o', figsize=(8,5), title=title)
            plt.grid(True, linestyle='--', alpha=0.7)
            plt.show()

        if "avg_adaptive_lr" in df_iter.columns:
            title = f"Linear LR={lr_val}, max_iter={max_iter_val}: Avg Adaptive␣
 ↪LR vs. Iteration"
            df_iter.plot(x="iteration", y="avg_adaptive_lr", marker='x',␣
 ↪figsize=(8,5), title=title)
            plt.grid(True, linestyle='--', alpha=0.7)
            plt.show()


##############################################################################
# 4) PANDAS + SEABORN PLOTS
##############################################################################

logger.info("=== Pandas + Seaborn Plots ===")

# 4A) LINE PLOT: Accuracy vs. max_iter (Perceptrons Only)
df_perc = df_results[df_results['model'].isin(['Clean PLA','Pocket PLA'])].
 ↪copy()
df_perc.sort_values(['model','max_iter'], inplace=True)

plt.figure(figsize=(6,4))
sns.lineplot(
    data=df_perc,
```

```
    x='max_iter', y='accuracy',
    hue='model', marker='o'
)
plt.title("Perceptrons: Accuracy vs. max_iter (Pandas/Seaborn)")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

# 4B) BAR CHART: Average Accuracy by Model
df_mean = df_results.groupby('model', as_index=False)['accuracy'].mean()

plt.figure(figsize=(6,4))
sns.barplot(data=df_mean, x='model', y='accuracy')
plt.title("Average Accuracy by Model (Pandas/Seaborn)")
plt.ylim(0.7, 1.0)
plt.grid(True, axis='y', linestyle='--', alpha=0.7)
plt.show()

# 4C) SCATTER PLOT: Accuracy vs. Runtime, colored by model
plt.figure(figsize=(6,4))
sns.scatterplot(
    data=df_results,
    x='runtime', y='accuracy',
    hue='model', style='model',
    s=100
)
plt.title("Accuracy vs. Runtime (All Models) (Pandas/Seaborn)")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()


##############################################################################
# 5) CUSTOM SUMMARY PLOTS (AGGREGATED CURVES, ETC.)
##############################################################################

logger.info("=== Custom Summaries (Aggregated Curves, etc.) ===")

# 5A) Aggregated Perceptron Curves
plot_train_curves_three_models(
    clean_train_curve=clean_train_curve,
    pocket_train_curve=pocket_train_curve,
    softmax_train_curve=None,  # no Softmax aggregator
    title="Aggregated Perceptron Train Curves (Clean vs. Pocket)",
    max_iter=perceptron_max_iter_values[-1]
)

# 5B) Summaries for Perceptron
plot_accuracy_vs_max_iter(
```

```
    max_iter_values=perceptron_max_iter_values,
    accuracies_clean=accuracies_clean,
    accuracies_pocket=accuracies_pocket,
    accuracies_softmax=None
)

plot_runtime_vs_max_iter(
    max_iter_values=perceptron_max_iter_values,
    runtimes_clean=runtimes_clean,
    runtimes_pocket=runtimes_pocket,
    runtimes_softmax=None
)

plot_accuracy_vs_runtime(
    runtimes_clean=runtimes_clean,
    accuracies_clean=accuracies_clean,
    runtimes_pocket=runtimes_pocket,
    accuracies_pocket=accuracies_pocket,
    title="Perceptrons: Accuracy vs. Runtime"
)

plot_performance_summary_extended_by_runtime(
    runtimes_clean=runtimes_clean,
    accuracies_clean=accuracies_clean,
    sensitivities_clean=sensitivities_clean,
    selectivities_clean=selectivities_clean,
    runtimes_pocket=runtimes_pocket,
    accuracies_pocket=accuracies_pocket,
    sensitivities_pocket=sensitivities_pocket,
    selectivities_pocket=selectivities_pocket,
    title="Perceptrons: Performance vs. Runtime"
)

# 5C) Summaries for Softmax & Linear
plot_accuracy_vs_runtime(
    runtimes_clean=runtimes_softmax,
    accuracies_clean=accuracies_softmax,
    title="Softmax: Accuracy vs. Runtime"
)
plot_accuracy_vs_runtime(
    runtimes_clean=runtimes_linear,
    accuracies_clean=accuracies_linear,
    title="Linear: Accuracy vs. Runtime"
)
plot_accuracy_vs_runtime(
    runtimes_clean=runtimes_softmax,
    accuracies_clean=accuracies_softmax,
```

```
    runtimes_pocket=runtimes_linear,
    accuracies_pocket=accuracies_linear,
    title="Softmax vs. Linear: Accuracy vs. Runtime"
)
plot_performance_summary_extended_by_runtime(
    runtimes_clean=runtimes_softmax,
    accuracies_clean=accuracies_softmax,
    sensitivities_clean=sensitivities_soft,
    selectivities_clean=selectivities_soft,
    runtimes_pocket=runtimes_linear,
    accuracies_pocket=accuracies_linear,
    sensitivities_pocket=sensitivities_lin,
    selectivities_pocket=selectivities_lin,
    title="Softmax vs. Linear: TPR/TNR vs. Runtime"
)

# 5D) 4-Model Comparison
plot_performance_summary_4models_by_runtime(
    runtimes_clean, accuracies_clean, sensitivities_clean, selectivities_clean,
    runtimes_pocket, accuracies_pocket, sensitivities_pocket,␣
 ↪selectivities_pocket,
    runtimes_softmax, accuracies_softmax, sensitivities_soft,␣
 ↪selectivities_soft,
    runtimes_linear, accuracies_linear, sensitivities_lin, selectivities_lin,
    title="Performance vs. Runtime (4-Model Comparison)"
)

plot_accuracy_vs_runtime_4models(
    rt_clean=runtimes_clean,
    acc_clean=accuracies_clean,
    rt_pocket=runtimes_pocket,
    acc_pocket=accuracies_pocket,
    rt_softmax=runtimes_softmax,
    acc_softmax=accuracies_softmax,
    rt_linear=runtimes_linear,
    acc_linear=accuracies_linear,
    title="Accuracy vs. Runtime (4 Models)"
)

logger.info("=== All Visualizations Complete ===")
```