Assignment 4: Generalization Error, Distribution Shifts & Adversarial Attacks

1 Instructions

In previous assignments, you focused on the *correctness* of algorithms and the *geometry* of data. In this assignment, we turn to the central problem of Machine Learning: **Generalization**.

A model is only useful if it performs well on data it has never seen before. However, "unseen data" comes in many forms. It might look exactly like your training data, it might look slightly different (shifted), or it might be maliciously crafted to trick your model.

You will stress-test classifiers through three levels of difficulty:

1. **I.I.D. Generalization:** The "Happy Path" where the future looks like the past.
2. **Distribution Shift:** The "Real World" where the test data changes (rotation, noise, inversion).
3. **Adversarial Robustness:** The "Worst Case" where we actively try to break the model.

Note on Experiment 3:

While K-Nearest Neighbors (KNN) is excellent for understanding geometry, its decision boundary is often too "jagged" or discrete to easily fool with standard gradient-like attacks. Therefore, in Experiment 3, you will switch to a **Support Vector Machine (SVM)** with a Radial Basis Function (RBF) kernel. This model produces a smoother decision boundary, making it a better target for studying adversarial vulnerability.

1.1 Submission Requirements

You must submit two components to Gradescope:

1. **The Report:** A plaintext Markdown document containing the 5 sections described in Part 2.
   - **No Images:** Do not paste images into the report. Describe your findings with text and data tables.
   - **Character Limit:** 7,500 characters.
2. **Supplemental Material:** Your `.ipynb` notebook containing all code, plots, and raw results.

2 Part 1: Implementation (The Notebook)

You will author a Jupyter Notebook (`.ipynb`) to perform the following experiments.

Global Setup:

At the top of your notebook, ensure reproducibility:

- `np.random.seed(0)`
- Use `random_state=0` for all split functions.
- Load the **Digits Dataset** (`sklearn.datasets.load_digits`).

The "True" Held-Out Test Set:

Before running any experiments, split your data into two distinct sets:

1. **Development Set (80%):** This corresponds to the "Upper Level Training" set from the lecture slides. You will use this for training, validation, and hyperparameter tuning.
2. **Held-Out Test Set (20%):** You will **lock this away**. You may *only* use this set for the final evaluation of your retrained models.

2.1 Experiment 1: The "Happy Path" (I.I.D. Generalization)

2.1.1 Concept: Model Selection vs. Model Evaluation

In machine learning, we often have two distinct goals:

1. **Model Selection:** Finding the best hyperparameters (like k).
2. **Model Evaluation:** Estimating how well the final model will perform in the real world.

To do this correctly, we use the Development Set to find the best k. Once k is chosen, we **retrain** the model on *all* available development data to maximize performance. Only then do we evaluate on the Held-Out Test Set.

2.1.2 Task 1.1: Three Ways to Choose k for KNN

Using **only the Development Set**, compare three strategies to select the optimal k from `[1, 3, 5, 7, 9, 11, 15, 20]`.

1. **Method A: The Optimist (Train Only)**
   - Train on the entire Development Set.
   - Evaluate on the **same** Development Set.
   - Select the k with the highest accuracy.
2. **Method B: The Realist (Train/Validation Split)**
   - Split the Development Set again (e.g., 75% Train / 25% Validation).
   - Train on this sub-training set; evaluate on the validation set.
   - Select the k with the highest validation accuracy.
3. **Method C: The Scientist (Cross-Validation)**
   - Run 5-Fold Cross-Validation on the entire Development Set.
   - Select the k with the highest mean CV accuracy.

2.1.3 Task 1.2: Retraining & Final Evaluation

For each method (A, B, and C):

1. Take the optimal k you selected.
2. **Retrain** a new model using that k on the **full Development Set**.
3. **Evaluate** this final model on the **Held-Out Test Set**.

Notebook Output:

- **Plot:** A single plot showing the hyperparameter tuning curves (Accuracy vs k) for Method A, B, and C.
- **Values:** A printed table showing the selected k and the final **Held-Out Test Accuracy** for each method.

2.2 Experiment 2: The "Real World" (Distribution Shift)

2.2.1 Concept: Covariate Shift

In the real world, the I.I.D. assumption often breaks. The distribution of inputs (x) changes, even if the label (y) remains the same. This is called **Covariate Shift**. You will evaluate how robust your model is to three different types of shifts.

2.2.2 Task 2.1: The Robustness Suite

1. **Select Best Model:** Instantiate a KNN classifier using the **optimal** k found via Cross-Validation (Method C) in Experiment 1.
   - **Note:** You can use standard uniform weights for this experiment (default), or distance weights.
2. **Train:** Fit the model on the **full Development Set**.
3. **Test 1: Rotation Shift**
   - Evaluate on the **Held-Out Test Set**, applying rotation to the images.
   - **Angles:** $[0, 15, 30, 45, 60, 90, 180]$ degrees.
   - *Note:* Use `scipy.ndimage.rotate(..., reshape=False, mode='nearest')`.
4. **Test 2: Gaussian Noise Shift**
   - Evaluate on the **Held-Out Test Set**, adding random Gaussian noise.
   - **Standard Deviation (σ):** $[0, 1, 2, 4, 8, 16]$.
   - *Process:* $x_{new} = x + \text{np.random.normal}(0, \sigma)$.
   - *Constraint:* After adding noise, you must **clip** the values to the valid range $[0, 16]$.
5. **Test 3: Inversion Shift**
   - Evaluate on the **Held-Out Test Set**, inverting the colors.
   - *Process:* $x_{new} = 16 - x$.
   - Compare the accuracy on original vs. inverted images.

Notebook Output:

- **Plots:** Two plots:
  1. Rotation Angle vs. Test Accuracy.
  2. Noise Standard Deviation (σ) vs. Test Accuracy.
- **Values:** Print the exact accuracy values for **every** test case (all angles, all sigmas, and inversion). You will need these numbers to construct the full tables in your report.

2.3 Experiment 3: The "Worst Case" (Adversarial Robustness)

2.3.1 Concept: Smooth Decision Boundaries with Kernel SVM

KNN models can be difficult to attack using gradient-based or hill-climbing methods because their decision surfaces are "step functions." To properly study adversarial robustness, we need a model with a smoother decision boundary.

We will use a **Kernel Support Vector Machine (SVM)** with a Radial Basis Function (RBF) kernel.

- **What is it?** An SVM finds a boundary that maximizes the margin between classes. The "Kernel" trick allows it to draw curved, non-linear boundaries in high-dimensional space.
- **Optional Reading:** Scikit-Learn SVM Documentation or Wikipedia - Support Vector Machine.
  - *Note: Knowledge of the SVM math or methodology is not required for this assignment; we are treating it simply as a "smoother classifier" for our attack.*

2.3.2 Task 3.1: Tuning and Training the SVM

You must tune two hyperparameters for the SVM using `GridSearchCV`:

1. `C`: Controls regularization (trade-off between smooth boundary and classifying training points correctly).
2. `gamma`: Controls the kernel coefficient (how "far" the influence of a single training example reaches).

Steps:

1. **Define Parameter Grid:**
   - `C`: `[0.1, 1, 10, 100]`
   - `gamma`: `['scale', 0.001, 0.01, 0.1]`
   - `kernel`: `['rbf']`
2. **Initialize SVM:** Use `sklearn.svm.SVC` with `probability=True`.
   - *Critical:* `probability=True` is required so the model outputs confidence scores (`predict_proba`) needed for the attack.
3. **Grid Search:** Run 5-fold Cross-Validation on the **Development Set**.
4. **Refit & Test:** Take the best estimator (GridSearch does this automatically if `refit=True`), and evaluate its accuracy on the **Held-Out Test Set**.

Notebook Output:

- Print the best parameters found (`best_params_`).
- Print the final **Held-Out Test Accuracy** of the SVM.

2.3.3 Task 3.2: The Epsilon Attack (on SVM)

You will implement a "black box" attack against your **tuned SVM**. You will test this on **10 images from the Held-Out Test Set** that were originally classified **correctly** by the SVM.

The attack function will return three values:

1. **Adversarial Image (x_{adv}):** The final perturbed image.
2. **Success (Boolean):** `True` if the attack successfully flipped the label, `False` otherwise.

3. **Relative L2 Error:** The L2 norm of the difference divided by the L2 norm of the original image. This gives us a percentage-like measure of how much the image was changed.

$$\text{Error} = \frac{\|x_{adv} - x\|_2}{\|x\|_2}$$

Constraints:

- **Epsilon (ϵ):** ± 4 (pixel changes bounded relative to the original image).
- **Pixel range:** $[0, 16]$.

Algorithm:

```
Input: Original image $x$, True label $y$, Model $f$, Iterations $T = 1000$, Epsilon $\epsilon = 4$
Output: $x_{adv}$, Success (Bool), Rel L2 Error

1. $x_{adv} \leftarrow x$   (Initialize with original)
2. for $t = 1$ to $T$ do
3.     Pick random pixel index $i \in [0, 63]$
4.     Sample perturbation $\delta \sim \text{Uniform}(-2, 2)$
5.     $x_{temp} \leftarrow x_{adv}$
6.     $x_{temp}[i] \leftarrow x_{temp}[i] + \delta$
7.     Apply Constraints:
       $x_{temp} \leftarrow \text{clip}(x_{temp}, \text{min} = 0, \text{max} = 16)$   (Valid Range)
       $x_{temp} \leftarrow \text{clip}(x_{temp}, \text{min} = x - \epsilon, \text{max} = x + \epsilon)$   (Epsilon Box)
8.     Check Success:
       if $f.\text{predict}(x_{temp}) \neq y$ then return $(x_{temp}, \text{True}, \frac{\|x_{temp} - x\|_2}{\|x\|_2})$
9.     Hill Climbing (Soft Condition):
       $\text{prob}_{old} \leftarrow f.\text{predict\_proba}(x_{adv})[y]$
       $\text{prob}_{new} \leftarrow f.\text{predict\_proba}(x_{temp})[y]$
       if $\text{prob}_{new} < \text{prob}_{old}$ then $x_{adv} \leftarrow x_{temp}$
10. end for
11. return $(x_{adv}, \text{False}, \frac{\|x_{adv} - x\|_2}{\|x\|_2})$   (Attack Failed)
```

2.3.4 Task 3.3: Run the Attack

1. Select **10 images** from the **Held-Out Test Set** that your **SVM** correctly classified.
2. Run the attack on each.
3. Calculate the **Success Rate** (how many flipped?) and the **Average Relative L2 Error** for the successful attacks.

Notebook Output:

- **Visualization:** Show a figure with 3 columns for **one successful attack**:
    1. **Original Image** (Label: "X")
    2. **Adversarial Image** (Label: "X_{adv}")
    3. **Difference Map** ($x_{adv} - x$, scaled to be visible).
- **Values:** Print the Success Rate and the Average Relative L2 Error.

---

3 Part 2: Content Requirements (The Report)

Your report must be organized into **exactly five sections** with Markdown headers.

3.1 Section 1: Executive Summary & Key Insight

- **One Sentence Takeaway:** A single sentence summarizing an interesting or insightful result regarding the model's fragility.
- **Summary Paragraph:** Briefly describe the progression from I.I.D. success to O.O.D. and adversarial failure. Include one concrete metric (e.g., "Accuracy dropped from X% to Y%").

3.2 Section 2: I.I.D. Generalization (Model Selection)

- **Results Table:** A small markdown table showing the optimal k selected by Method A, B, and C, and the final **Held-Out Test Accuracy** for each.
- **Analysis:**
    - Why does Method A (Training Accuracy) select $k = 1$? Explain why this is "overfitting" in terms of the bias-variance tradeoff.
    - Which method provided the most reliable selection for the held-out test set?

3.3 Section 3: Distribution Shift

- **Results Tables:**
    - **Rotation:** A table showing accuracy for **all** tested angles $[0, 15, 30, 45, 60, 90, 180]$.
    - **Noise:** A table showing accuracy for **all** tested noise levels $[0, 1, 2, 4, 8, 16]$.
    - **Inversion:** The accuracy for inverted images.
- **Analysis:**
    - Compare the three shifts. Which one degraded performance the fastest?
    - Did the model perform better than random guessing on the Inverted images? Why or why not?

3.4 Section 4: Adversarial Robustness

- **Comparison (KNN vs SVM):** First, compare the standard Held-Out Test Accuracy of your best KNN model (from Exp 1/2) vs. your best SVM model (from Exp 3). Which model was more accurate on standard data?
- **Attack Results:** Report your SVM attack Success Rate (out of 10) and the **Average Relative L2 Error**.
- **Visual Analysis:** Look at your Difference Map.
  - Does the noise look like a structured digit (e.g., drawing a new line to make a 1 look like a 7)?
  - Or does it look like random "snow"?
  - Would a human be fooled by the adversarial image?

3.5 Section 5: Reflection

- **Measuring Performance:** Reflect on the difficulty of estimating "true" model performance.
  - When selecting models or deploying them in the real world, what factors should you consider beyond simple test accuracy?
  - When can you reasonably trust a performance estimate (like the one from Experiment 1), and when should you be cautious (like in Experiment 2)?
  - How does the possibility of adversarial attacks change your definition of a "good" model?

---

4 Grading Rubric

Each of the five sections will be weighted equally (**20% each**).

Criterion	Excellent (5)	Good (4)	Satisfactory (3)	Okay (2)	Poor (1)
Section 1: Executive Summary	Takeaway is insightful and captures the "fragility" theme well. Summary uses specific data to contrast the experiments.	Takeaway is clear. Summary covers the main arc of the assignment.	Takeaway is generic. Summary is present but vague.	Summary misses key elements of the analysis (e.g., ignores the attack).	Missing or unintelligible.

Criterion	Excellent (5)	Good (4)	Satisfactory (3)	Okay (2)	Poor (1)
Section 2: I.I.D. Generalization	Correctly identifies why $k = 1$ overfits (Method A). Correctly identifies Method C as robust. Data table includes final held-out accuracies.	Explanations are correct but slightly mechanical. Data table present.	Misses the connection between $k = 1$ and overfitting. Data table present.	Data table missing or values are incorrect.	Missing or fails to implement the methods.
Section 3: Distribution Shift	Analysis compares all three shifts. Tables include **all** angles and sigma values.	Good description of the curves. Comparison is reasonable. Tables complete.	Describes results but fails to compare the shifts meaningfully. Tables may be incomplete.	Data tables missing or results are clearly wrong.	Missing or fails to implement shifts.
Section 4: Adversarial Robustness	Compares KNN vs SVM accuracy. Insightful analysis of the difference map. Relative L2 Error and Success Rate are reported accurately.	Good description of visual results and SVM comparison. Metrics are present.	Metrics present. Visual analysis states the obvious without insight.	Attack metrics missing or implausible.	Missing or fails to implement attack.
Section 5: Reflection	Thoughtful reflection on the nuances of performance estimation and trust. Addresses real-world implications of distribution shift and adversarial risks.	Good reflection on metrics and trust.	Generic reflection (e.g., "Accuracy is important").	Minimal effort.	Missing.