Auburn University

Assignment 2

COMP 5630/ COMP 6630/ COMP 6630 - D01 (Fall 2025)

Machine Learning

Deadline: Oct 22, 2025, 11:59 PM CST

Overview

In this assignment, you will:

- Implement and evaluate binary logistic regression classifiers using the One-vs-Rest (OvR) approach.
- Train three models: Class 0 vs Rest, Class 1 vs Rest, and Class 2 vs Rest.
- Extract and compare feature importance from each model.
- Apply L1 regularization to observe its effect on model sparsity.
- Reflect on interpretability, performance, and selected features.

Part 1: Dataset and Preprocessing (10 points)

1. Load the Wine dataset from `sklearn.datasets.load_wine`.
2. Extract the feature matrix $\mathbf{X}$ and target vector $\mathbf{y}$.
3. Split the dataset into training and testing sets using `train_test_split` from `sklearn.model_selection`, with 80% for training and 20% for testing. Use `stratify=y` to maintain class proportions and `random_state=15` for reproducibility.
4. Standardize the features using `StandardScaler`.
5. For each class $c \in \{0, 1, 2\}$, create a binary classification problem:

$$y_c = \begin{cases} 1 & \text{if } y = c \\ 0 & \text{otherwise} \end{cases}$$

Deliverable:

- Code for loading and preprocessing.
- Shape of $\mathbf{X}$ and distribution of target classes.

Part 2: Logistic Regression with Gradient Descent (30 points)

Objective: Implement logistic regression from scratch using gradient descent. **You can not use Scikit-learn for this part.**

1. Implement the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

2. Implement the log-likelihood loss and its gradient:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

   where $\hat{y}^{(i)} = \sigma(\mathbf{w}^\top \mathbf{x}^{(i)})$.

3. Update weights using gradient ascent:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \nabla \mathcal{L}$$

4. Use a learning rate $\eta = 0.0001$ and run for 5000–10000 iterations.

5. Evaluate training loss and accuracy for each class-vs-rest model.

Deliverable:

- Code for gradient descent implementation.
- Training loss curves.
- Accuracy for each OvR classifier.

Part 3: Logistic Regression with Scikit-learn (No Regularization) (15 points)

Use `LogisticRegression(penalty='none')` to replicate your manual implementation.

For each class $c = 0, 1, 2$:

- Fit the model: $y_c = 1$ vs rest.
- Extract feature weights: `model.coef_`.
- Rank features by absolute value of weight.
- Report top 3 features with highest influence.

Deliverable:

- Table of weights and top 3 features for each class.
- Training accuracy.
- Confusion matrix on test set.

Part 4: Feature Importance with L1 Regularization in Scikit-learn (25 points)

Train logistic regression models with L1 regularization for each OvR classifier.

1. Use `LogisticRegression(penalty='l1', solver='liblinear', C=0.1)`.
2. Extract feature weights and identify which are zeroed out.
3. Compare the top 3 features with and without L1 regularization.
4. Evaluate model performance.

Deliverable: For each class c, provide a table like the following:

Feature	Weight (No Reg)	|Weight|	Weight (L1)	Zeroed?
flavanoids	-1.23	1.23	0.00	Yes
alcohol	0.89	0.89	0.75	No

Part 5: Evaluation and Comparison (10 points)

1. Compare training and test accuracy for all models:
   - No regularization.
   - L1 regularization.
2. How many features were eliminated by L1 for each class?
3. Did feature sparsity help or hurt performance?

Deliverable:

- Table of performance metrics.
- Number of features retained.
- Summary of top features across all classes.

Part 6: Analysis and Reflection (10 points)

Answer the following questions briefly:

1. Were the most important features consistent across all three class-vs-rest classifiers?
2. Did any features appear important for one class but not others?
3. How does L1 regularization affect interpretability?
4. If you had to select 5 total features for all classes, which would you pick and why?

Submission Checklist

- Code (in `.ipynb`).
- Feature weight tables for each class.
- Training and evaluation results.
- Plots (loss curves, if implemented).
- Written reflection.

Notes

1. If your code does not run on Colab, you will not get any credit for the code segment. We will only grade what is in your report.
   a. This includes any syntax errors due to indentation, unnamed/unknown libraries that were not listed in the README file, etc.
2. Please submit code only in Python and in the IPython notebook format. You can write your answers as part of the notebook if you do not want a separate report file, but it must be comprehensive.
   a. Any code not in Python will not be graded at all.
3. Please declare any use of GenAI usage. Your points will not be deducted for declaration. But if the TA or instructor discovers by themselves of GenAI usage and you don't declare, then your point will be deducted.