DATA2060:
Multiclass Classification on the Iris Dataset

Group Name: Overfitting Avengers
Group Members: Hyuk Ahn, Chuiyang Kong, Jincheng Yang
Date: 12/10/2024

Content

1. Introduction
2. Math of Logistic Regression
3. Numerical Techniques (One-Vs-All and All-Pairs)
4. Results (Comparison with previous works)
5. Challenges and Summary

Introduction of Multiclass classification

Aims: to classify an input that can belong to one of the multiple classes (≥ 2)

Our Algorithms:

- The Binary Classifier: Binary Logistic Regression
- One-vs-All
- All-Pairs

Representations:

- Samples' feature values $x \in \mathbb{R}^d$ and labels $y \in \{0, 1, \ldots, k\}$
- Iris Data Set

Representation of Iris Dataset

150 samples of flowers from the Iris genus

4-feature $x \in \mathbb{R}^4$:

Sepal length (in cm)

Sepal Width (in cm)

Petal Length (in cm)

Petal Width (in cm)

3 Classes indicating species:

0: Iris-setosa

1: Iris-versicolor

2: Iris-virginica

Math of Binary Logistic Regression

- Representations: x ∈ $\mathbb{R}^d$ and labels y ∈ {0, 1} (bias added to the last column of x)
- Labels are predicted based on the affine function and sigmoid function

  - Affine function:

$$y = \langle \mathbf{w}, \mathbf{x} \rangle \quad (1)$$

  - Sigmoid function:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \quad (2)$$

  - Hypothesis function:

$$h_w(x) = \frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{x} \rangle}} \quad (3)$$

  - Predicted Label:

$$y = \begin{cases} 0 & \text{if } h_{\mathbf{w}}(\mathbf{x}) \leq 0.5, \\ 1 & \text{if } h_{\mathbf{w}}(\mathbf{x}) > 0.5. \end{cases} \quad (4)$$

Math of Binary Logistic Regression

- Binary Log Loss:

$$L(h) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h(x_i) + (1 - y_i)\log(1 - h(x_i))\right] \quad (5)$$

- Weight updating:

$$w_j = w_j - \alpha \cdot \frac{\partial L}{\partial w_j} \quad (6)$$

- The gradient of the Binary Log Loss with respect to weights:

$$\frac{\partial L}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left[(h(x_i) - y_i) \cdot x_{ij}\right] \quad (7)$$

Optimizer for Logistic Regression: Pseudo-code

```
Given inputs:
    Training samples S, step size α, batch size b < |S|

Initialize: w = 0

for t = 1, 2, ..., T:
    Randomly shuffle S
    For i = 0 to |S|/b - 1:
        S' = Extracted current batch using i
        w = w − α · ∇L_S'(h_w) + regularization
Return w
```

Numerical Techniques

One-Vs-All and All-Pairs

- Reduces multi-class classification into multiple binary classification problems

- Different binary classifiers are trained for either:
  - Each class (One-Vs-All)
  - Each pair of classes (All-Pairs)

- Final predictions are made considering the outputs of all binary classifiers

Numerical Techniques: One-Vs-All

Basics of One-Vs-All

Role of each binary classifier:

Predicting whether an input belongs to its class or not

Training data for each binary classifier:

Transform dataset into binary-labeled dataset

Original Dataset

X	Y
15	0
23	1
35	2
45	3

Dataset for binary classifier for class 2

X	Y
15	-1
23	-1
35	1
45	-1

Table 1: Illustration of One-Vs-All dataset transformation

Pseudo-code of One-Vs-All

Input:

- X: Training data (number of examples, number of features)
- Y: Training labels ($y_i \in \{0, 1, ..., K\}$, where K == number of classes)
- K: number of classes

Output:

- Binary classifiers h_k for each class $k \in \{0, 2, ..., K\}$

<u>OneVsAll Training Algorithm(X, Y, k):</u>

```
Initialize list Classifiers to hold k number of binary classifiers

For each class k in {0, 1, ..., K}:
    Create new Y_k such that Y_k[i] = 1 if Y[i] == k else -1
    Initialize new binary classifier h_k
    Train the binary classifier h_k with inputs X and labels Y_k
    Append h_k to Classifiers

Return the list of trained classifiers
```

<u>OneVsAll Predict(x):</u>

```
Compute the probability of x for each class using each class' classifier

Assign x to the class with the highest probability

Return predicted class
```

Numerical Techniques: All-Pairs

Basics of All pairs

Role of each binary classifier:

Predicting whether an input belongs to class A or class B

Training data for each binary classifier:

1.Filter data based on class pairs

2.Transform dataset into binary-labeled dataset

X	Y
10	0
20	1
30	2

X	Y
10	0
20	1

All pairs train

- Iterate each pair of classes (A, B)
- For each pair, filter the data and assign binary labels

X	Y
10	0
20	1
30	2

X	Y
10	0
30	2

X	Y
10	0
30	1

- Train the binary classifier using logistic regression
- Stored the trained model for later prediction

All pairs predict

- Iterate through the binary classifiers trained earlier
- For each binary classifier, make prediction and accumulate votes & confidence score
- For each example, the class with the highest vote is selected as the final prediction
- Resolve tie using confidence score from classifiers

All Pairs pseudo code

Given inputs:

training set $S = (x_1, y_1), \ldots, (x_m, y_m)$

class set C = (0, 1, ..., k), where k == number of classes

binary classifier - logistic regression L

foreach $i, j \in C$ such that $i < j$

Initialize empty dataset $S_{i,j}$

for $t = 1, \ldots, m$

If $y_t = i$, then add $(x_t, 1)$ to $S_{i,j}$

If $y_t = j$, then add $(x_t, -1)$ to $S_{i,j}$

Train $h_{i,j} = L(S_{i,j})$

Outputs:

$h(x) \in argmax_{i \in Y} \; (\Sigma_{j \in Y} \; \text{sign}(j - i) h_{i,j}(x))$

Results
(comparison with previous works)

Scikit-Learn: SGDClassifier

<u>Important hyperparameters to consider</u>

loss = ‘log_loss’

fit_intercept = True

max_iter = [desired train epoch]

learning_rate = ‘constant’

eta0 = [learning rate]

<u>LogisticRegression vs. SGDClassifier</u>

Optimization through SGD

Scikit-Learn: OneVsRestClassifier and OneVsOneClassifier

The only hyperparameter to consider:

estimator = SGDClassifier(...)

Creates multiple binary classifiers based on the number of classes determined by the given Y

Previous work reproduced (One-Vs-All)

Comparison to
Scikit-learn's One-Vs-Rest
classifier with SGDClassifier

```
np.random.seed(0)
lr = 0.01
train_epochs = 1000
batch_size = 1
```

Figure 1: Hyperparameters for One-Vs-All and One-vs-Rest

```
Predictions of our One-vs-All model: [0 1 1 1 2 2 0 2 0 2 0 0 2 2 0 2 0 2 1 2 1 0 0 2 0]
Predictions of sklearn OneVsRest Classifier: [0 1 1 1 2 2 0 2 0 2 0 0 2 2 0 2 0 2 1 2 1 0 0 2 0]
num_samples of training dataset 125
num_samples of testing dataset 25
Differences 0
Accuracy of our One-vs-All model: 1.0
Accuracy of sklearn OneVsRest Classifier: 1.0
```

Figure 2: Results for One-Vs-All and One-vs-Rest on Iris Dataset

Previous work reproduced (All-Pairs)

Comparison to
Scikit-learn's One-Vs-One
classifier with SGDClassifier

```
np.random.seed(0)
lr = 0.01
train_epochs = 1000
batch_size = 1
```

Figure 3: Hyperparameters for One-Vs-One and All-Pairs

```
Predictions of our All-Pairs Model: [0 1 1 1 2 2 0 2 0 2 0 0 2 2 0 2 0 2 1 2 1 0 0 2 0]
Predictions of sklearn OneVsOne Classifier: [0 1 1 1 2 2 0 2 0 2 0 0 2 2 0 2 0 2 1 2 1 0 0 2 0]
num_samples of training dataset 125
num_samples of testing dataset 25
Differences 0
Accuracy of our All-Pairs model: 1.0
Accuracy of sklearn OneVsOne Classifier: 1.0
```

Figure 4: Results for All-Pairs and One-Vs-One on Iris Dataset

Challenges and Summary

Not Exact Match

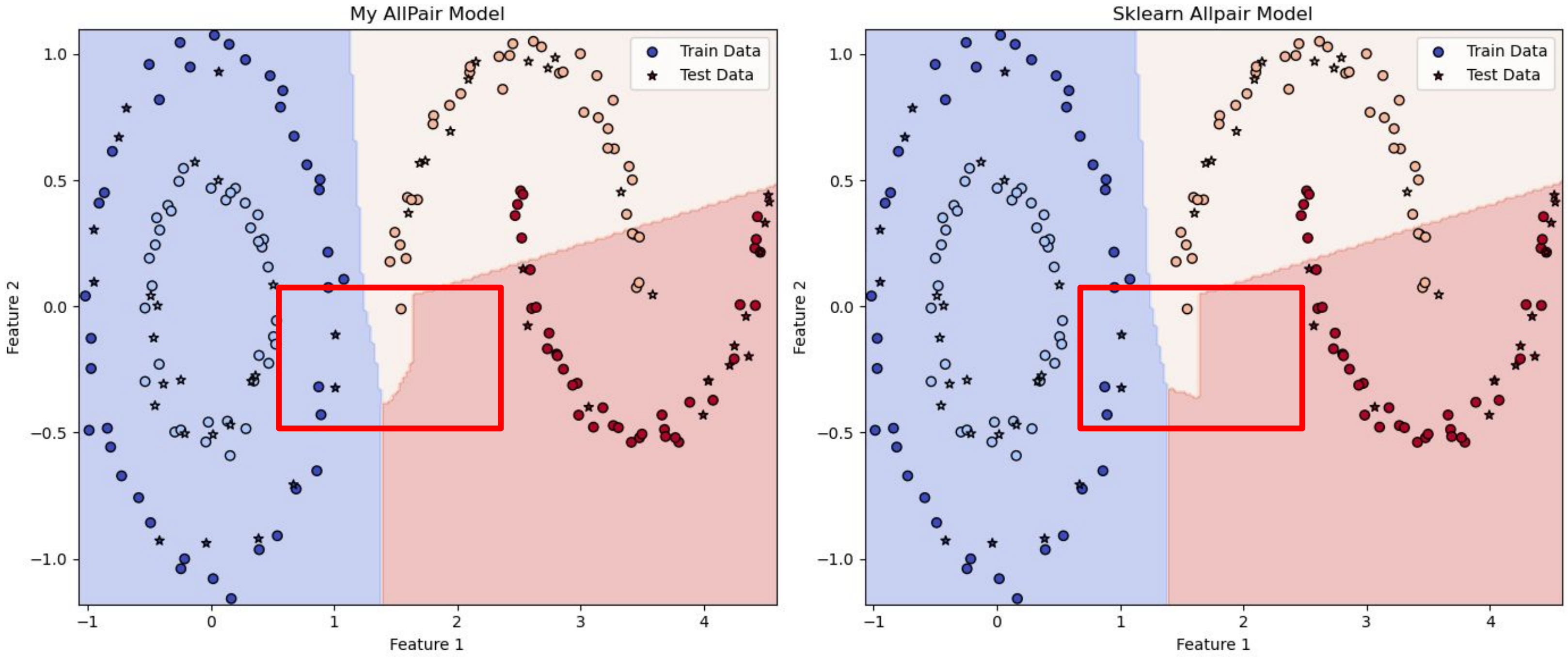

Figure 5: Unit tests visualizing decision boundaries for AllPair on 4-class non-linearly separated dataset

Shuffling Problem → Different Weights

```
 ===  0
[-0.3036488  -0.10085157]
[-0.30287925 -0.10010839]
1.6796336912342988e-05
-0.008073173358932628
 ===  1
[-0.24344459 -0.25772009]
[-0.24805068 -0.25899929]
0.5306252943864805
0.5349923408698549
 ===  2
[0.28011449 0.06059034]
[0.27916507 0.06756442]
-1.219446451611035
-1.2143026065971645
My Predictions: [0 1 2 2 0 1 1 2 2 2]
sklearn Predictions: [0 1 2 2 0 1 1 2 2 2]
num_samples 10
Differences 0
```

Classifier for class 0
Our weights
sklearn's weights
Classifier for class 1
Our bias
sklearn's bias

Turn
shuffle off

```
 ===  0
[-0.31197662 -0.10912839]
[-0.31197662 -0.10912839]
-0.026783970677386276
-0.02678397067738624
 ===  1
[-0.25499991 -0.25585737]
[-0.25499991 -0.25585737]
0.533835614573224
0.533835614573224
 ===  2
[0.29462046 0.07319256]
[0.29462046 0.07319256]
-1.1883256927760357
-1.1883256927760357
My Predictions: [0 1 2 2 0 1 1 2 2 2]
sklearn Predictions: [0 1 2 2 0 1 1 2 2 2]
num_samples 10
Differences 0
```

Figure 6: Unit tests outputting weights of different binary classifiers for OneVsAll on 10-Point dataset

What We Tried

- Use random state imported from sklearn instead of numpy
- Use shuffle function from sklearn
- Look through the source codes:
    - scikit-learn/sklearn/linear_model/**_stochastic_gradient.py**
    - scikit-learn/sklearn/linear_model/**_sgd_fast.pyx.tp**

```
for epoch in range(max_iter):
    sumloss = 0
    if verbose > 0:
        with gil:
            print("-- Epoch %d" % (epoch + 1))
    if shuffle:
        dataset.shuffle(seed)
    for i in range(n_samples):
        dataset.next(&x_data_ptr, &x_ind_ptr, &xnnz,
                     &y, &sample_weight)
```

Figure 7: Shuffling part in Scikit-learn

Summary

- We use OneVsAll and AllPair algorithms with Binary Logistic Regression to achieve multiclass classification.
- We compare our results with previous work on Iris dataset
- Shuffling issues result in some predictions that differ from Scikit-learn implementations.

Github repo: https://github.com/hyukahn16/data2060-final-project

References

[1] Fisher, R.A. (1936) 'Iris.' Available at: https://archive.ics.uci.edu/dataset/53/iris (Accessed November 2024).

[2] Scikit-learn. (2024) *1.12 Multiclass and multioutput algorithms* [Online]. Available at: https://scikit-learn.org/1.5/modules/multiclass.html#multiclass-classification (Accessed November 2024).

[3] Scikit-learn. (2024) *sklearn.metrics.log_loss* [Online]. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html (Accessed November 2024).

[4] Scikit-learn. (2024) *sklearn.multiclass.OneVsRestClassifier* [Online]. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.multiclass.OneVsRestClassifier.html (Accessed November 2024).

[5] Scikit-learn. (2024) *sklearn.multiclass.OneVsOneClassifier* [Online]. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.multiclass.OneVsOneClassifier.html (Accessed November 2024).

[6] Scikit-learn. (2024) *sklearn.linear_model.SGDClassifier* [Online]. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.SGDClassifier.html (Accessed November 2024).

[7] Shalev-Shwartz, S. and Ben-David, S. (2014) *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press.

Contributions

One-vs-All & Relevant Weight Unit Test

Hyuk Ahn

All-Pairs & Relevant Weight Unit Test

Jincheng Yang

Binary Logistic Regression & Rest Unit Tests

Chuiyang Kong

Reproduction on Iris Dataset

Chuiyang Kong

Slides Preparation

Hyuk Ahn, Chuiyang Kong, Jincheng Yang

Markdown Documentation

Hyuk Ahn, Chuiyang Kong, Jincheng Yang