Multiclass Classification:
One-vs-all (OvR) & All-pairs (OvO) using Binary Logistic Regression

•••

Erica Brown, Mia Begic, Karolina Palac, Musa Tahir

BROWN
Data Science Institute

Mathematical Background & Numerical Techniques

Binary Logistic Regression Representation

Sigmoid Function:

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

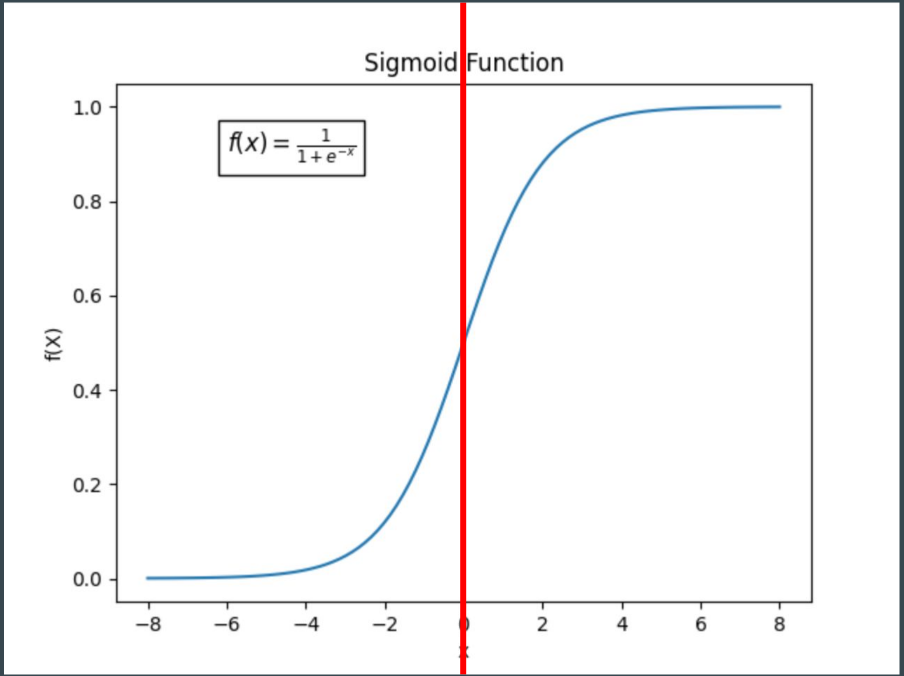
z = <w,x>

Input: $X = \mathbf{R}^d$

Output: [0, 1] → probabilities

$Y = [-1, 1]$

Decision Boundary: <w, x> = 0



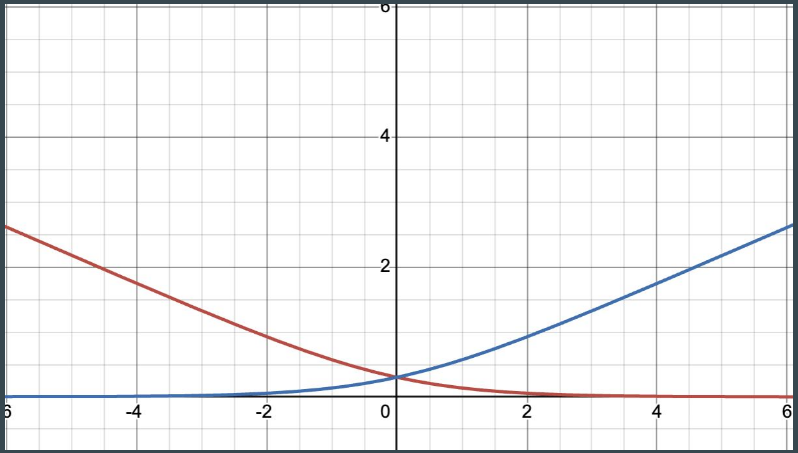
https://www.codecademy.com/resources/docs/ai/neural-networks/sigmoid-activation-function

Loss

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x}\rangle))$$

Log loss is convex!

Penalizes degree of incorrectness



y = 1
y = -1

Optimization

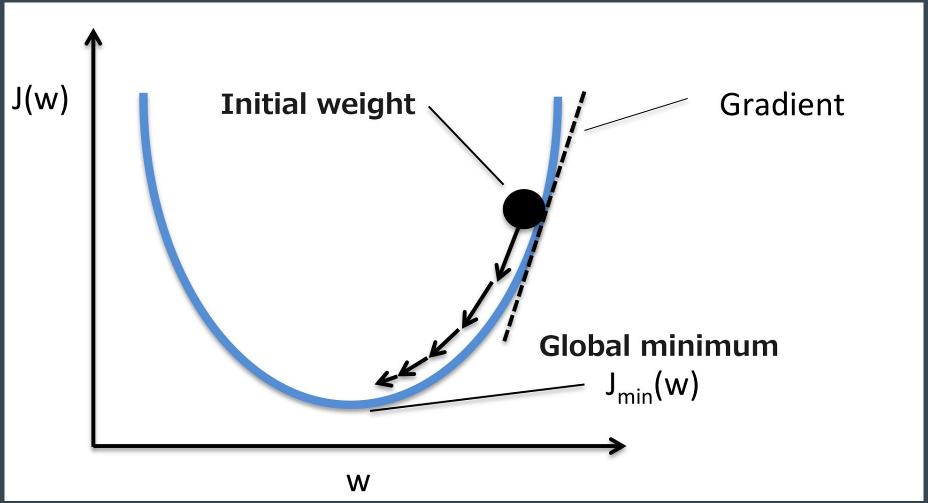
- **Empirical Risk Minimization** → minimize expected loss over ***all*** available data
- Log loss is convex → at most one global minima
- Use **Stochastic Gradient Descent** to update weights:

$$w_j = w_j - \alpha \frac{\partial L}{\partial w_j}$$

- where α is learning rate, which is multiplied by gradients of L(w) with respect to each weight
- $\alpha = 0.01$

Convergence Criteria

- **Max epoch limit = 1000**
- **Convergence Threshold = 1e-4**



https://ai.stackexchange.com/questions/16348/what-is-convergence-in-machine-learning

One-vs-all & All-Pairs Approaches

Binary Classification → Multiclass Classification

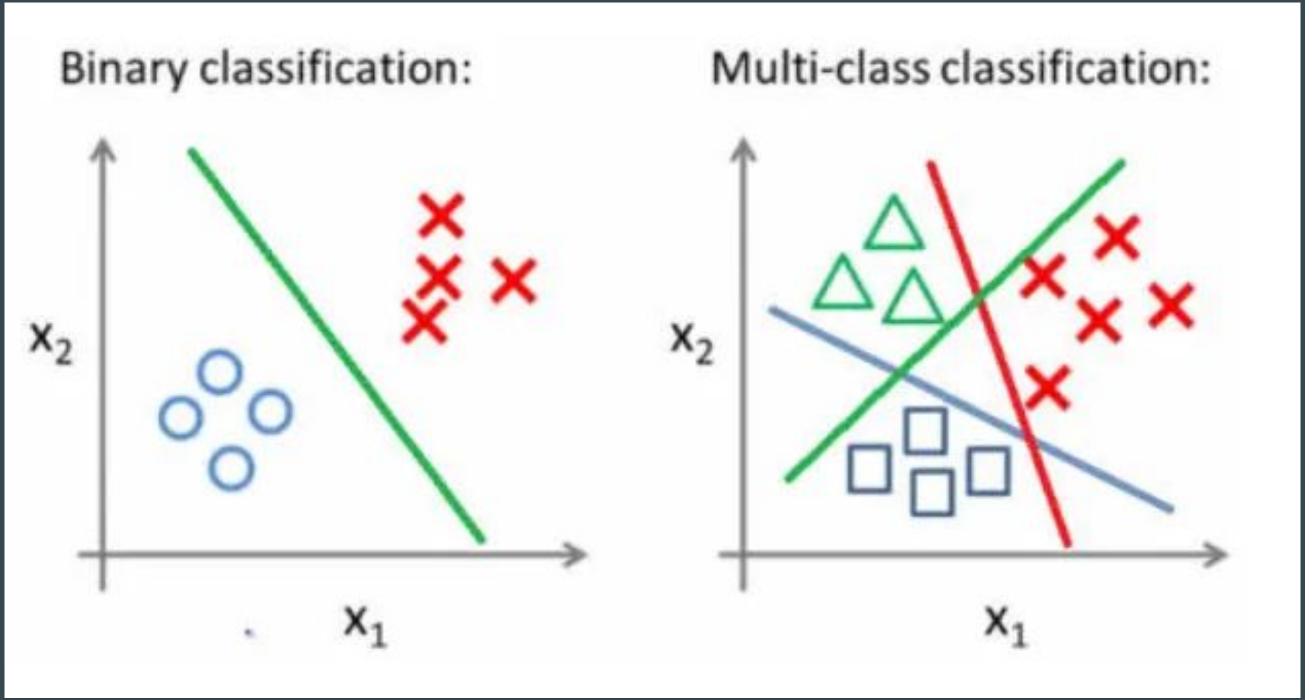


Image Source: Binary vs Multi-class classification. Retrieved from utkuufuk.com

One-vs-All Visualization

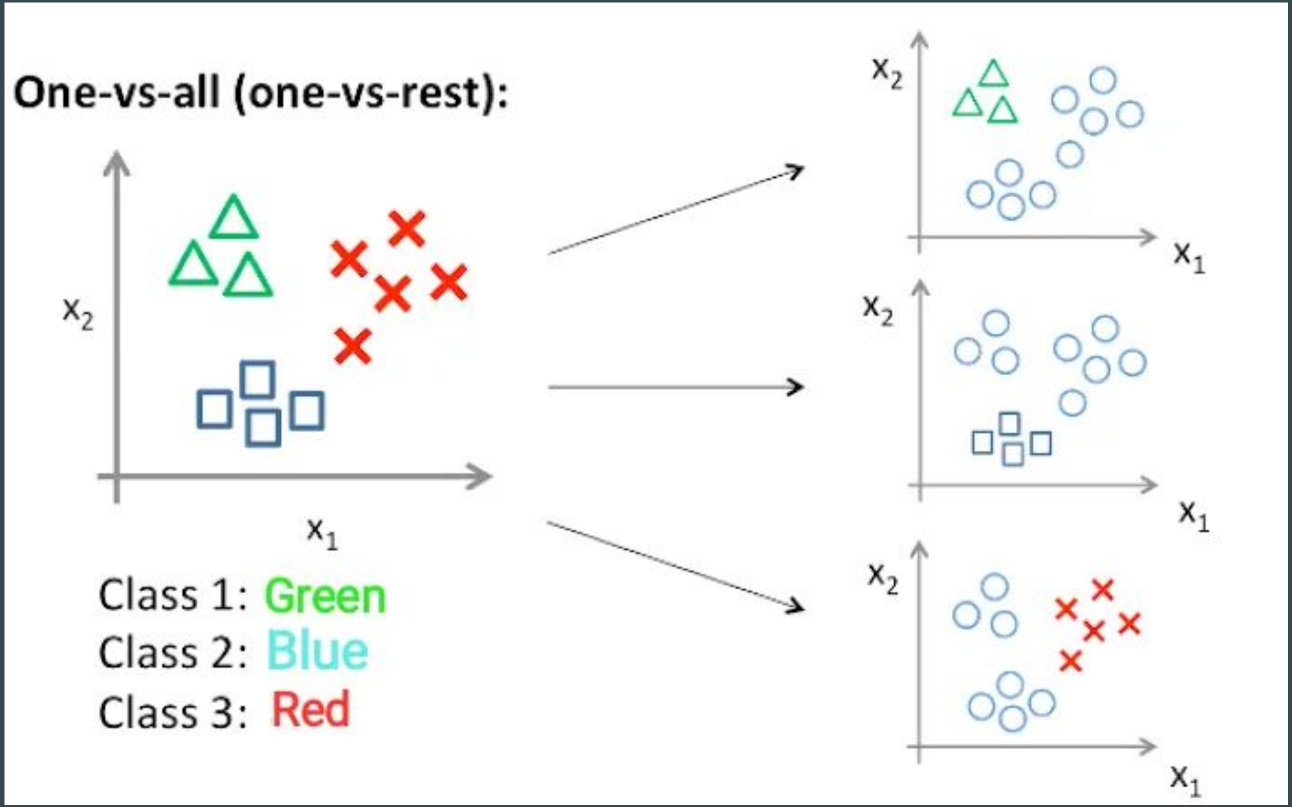


Image Source: One-vs-all classification. Retrieved from cc.gatech.edu

Musa: One-vs-All

Training data: $\mathbf{X}$ (features), $\mathbf{y}$ (labels) with k classes
Logistic regression model for binary classification

1. Initialize an empty list, *models*, to store each class's logistic regression model
2. For each class i in range 1 to k :
   a. Create a new binary label vector y_i where:
      - $y_i[j] = 1$ if $y[j] = i$ (current class)
      - $y_i[j] = 0$ otherwise (all other classes)

   b. Train a logistic regression model $model_i$ using $\mathbf{X}$ and y_i

   c. Store $model_i$ in the list *models*
3. For a given input $\mathbf{x}$:
   For each model i in range 1 to k :
      - Use each model in *models* to predict the probability that $\mathbf{x}$ belongs to each class
      - Store the probabilities in a list *probabilities*

   Select the class with the highest probability from *probabilities* as the predicted class
4. Return the predicted classes

All-Pairs Visualization

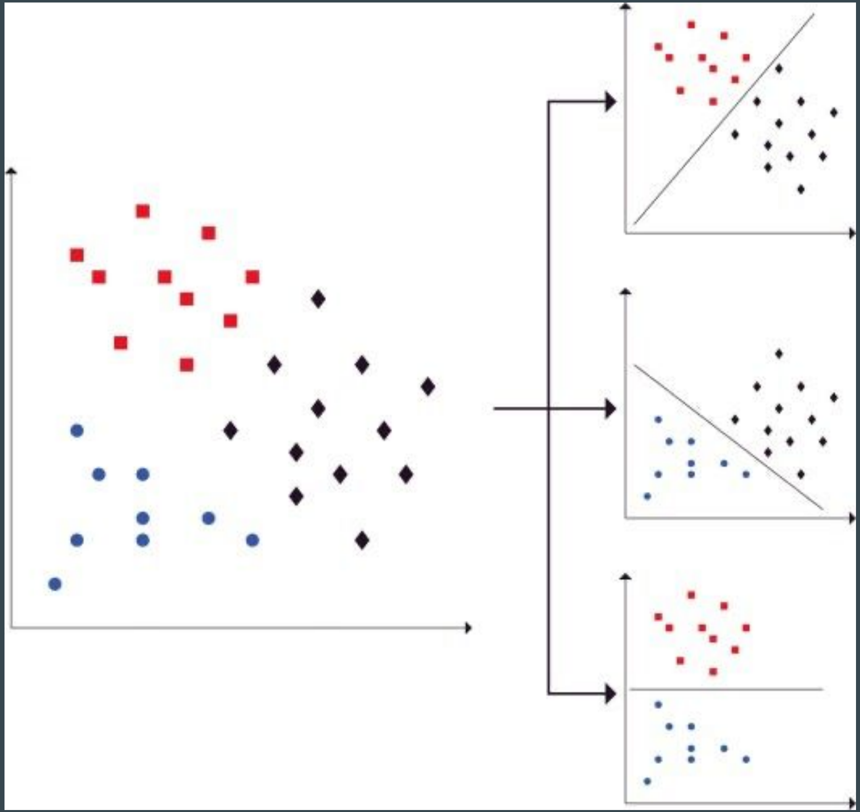


Image Source: All-Pairs classification. Retrieved from ScienceDirect.com

Erica: All-Pairs

Train method
Input training data **X** features and **Y** labels
Binary logistic regression model

- Validate input data
- Create all possible class pairs (class_i, class_j) where class_i < class_j
- For each pair of classes :
    - Create mask [Y==class_i | class_j]
    - Filter X and Y to get SX and SY
    - Convert SY to binary values [1,0]
    - Initialize binary classifier
    - Train with SX and SY
    - Store trained classifier

Predict method
Input training data **X** features and **Y** labels
Binary logistic regression model

- Validate input data
- Initialize vote array with zeros to store votes for each class for each sample
- For each pair of classes (class_i, class_j) and their classifiers:
    - Use classifier to predict binary labels [1,0]
    - If 1, add vote to class_i
    - If 0, add vote to class_j
- For each sample in X, assign class_label with highest vote count
- Return **predicted class label**

Previous Work

Dataset: Obesity Risk Prediction

Obesity Risk Prediction

Target variable: NObeyesdad (obesity level)

- 7 distinct obesity categories ranging from insufficient weight to severe obesity
  - 0: Insufficient Weights
  - 1: Normal Weight
  - 2: Overweight Level 1
  - 3: Overweight Level 2
  - 4: Obesity Type 1
  - 5: Obesity Type 2
  - 6: Obesity Type 2

Categorical and quantitative data

[Obesity Risk Prediction Cleaned | Kaggle]()

Previous Work: Scikit-learn & Kaggle Notebook

Mr. Amine

```
OvR_LR: Pipeline(steps=[('scaler', MinMaxScaler()),
                ('lr',
                 OneVsRestClassifier(estimator=LogisticRegression(max_iter=100000,
                                                                  penalty=None,
                                                                  solver='saga')))])
OvO_LR: Pipeline(steps=[('scaler', MinMaxScaler()),
                ('lr',
                 OneVsOneClassifier(estimator=LogisticRegression(max_iter=10000,
                                                                 penalty=None,
                                                                 solver='saga')))])
```

OvR_LR: 0.7508143567369544
OvO_LR: 0.9578617970056797

Mia: Preliminary Results

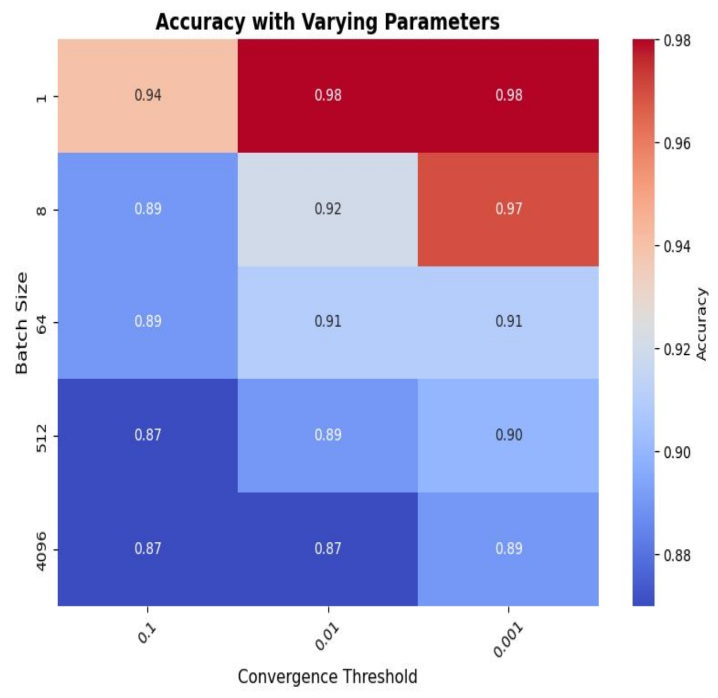
<u>One-vs-all</u>

Our

- training accuracy: 73.26%
- test accuracy: 73.21%

Kaggle Data Scientist:

- training accuracy: 73.5%
- test accuracy: 71.53%



<u>Heatmap</u>

- Peak at 98% for batch size 1 and convergence threshold 0.001 and 0.01

Mia: Preliminary Results

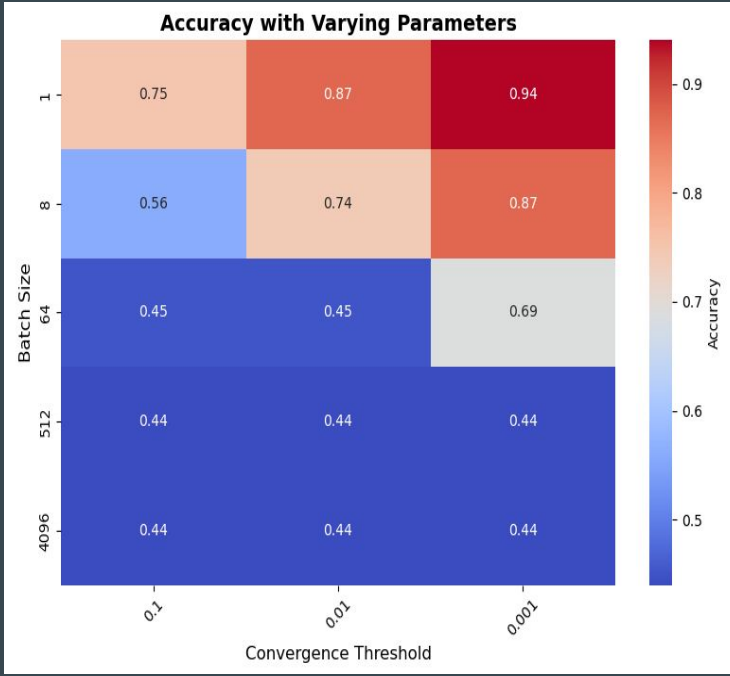
<u>All-pairs</u>

Our

- training accuracy: 96.4%
- test accuracy: 95.22%

Kaggle Data Scientist:

- training accuracy: 94.42%
- test accuracy: 94%



<u>Heatmap</u>

- Peak at 94% for batch size 1 and convergence threshold 0.001

Mia: Summary

Strengths of the Algorithms:

- **One-vs-all** simplifies implementation and interpretation, excelling with smaller datasets and fewer classes
- **All-pairs** offers a more granular view of inter-class relationships, particularly for datasets with complex boundaries

Challenged Encountered:

- Fine-tuning convergence

What Stood Out:

- Trade-off between simplicity and scalability

Why All-Pairs Excels

- More robust voting system
  - All-Pairs takes into account many more pairwise classifiers that combine to make arguably a more informed decision
  - One-vs-all is less robust, since the decision for each class depends on a single classifier's input, which may be biased due to noise or overlapping features
- Simpler Decision Boundaries
  - All-pairs trains models to distinguish between just two features at a time, which makes it easier to learn precise decision boundaries between two closely related classes (e.g. normal weight, overweight)
  - One-vs-all must learn more complicated decision boundaries that may not be feasible for data that isn't linearly separable (using binary logistic regression)

Convergence Criteria

- **Convergence Threshold = 1e-4**
- **Max epoch limit = 1000**

Avoid overfitting!

- **L2 Regularization**

$$R(\mathbf{w}) = \lambda\|\mathbf{w}\|_2^2 \qquad \|\mathbf{w}\|_2^2 = \sum_{i=1}^{d} w_i^2$$

λ → controls contribution

- **Implemented with SGD**

$$\frac{\partial L_S(h_{\mathbf{w}}) + R(h_{\mathbf{w}})}{\partial w_{st}}$$

$$\frac{\partial \lambda \sum_{i=1}^{d} w_i^2}{\partial w_j} = 2\lambda w_j$$

Factors Influencing Minor Differences

- Regularization techniques (e.g. L1, L2)
- Hyperparameter settings (e.g. convergence thresholds)
- Weight Initialization
- Data Preprocessing