Tutorial for Task 2

Advanced Machine Learning, Fall 2020

Task 2: Overview

Disease Classification from Image Features:

- Pre-extracted features from image dataset
- Each image is given as a continuous feature vector of size 1000
- 4800 training samples, 4100 test samples
- 3 classes labeled as {0, 1, 2}
- Same class distribution for training and test data

Your Task: Multiclass classification

- Learn a mapping from the features to the set {0, 1, 2}

Multiclass Classification

We are going to see several ways of doing multiclass classification:

- One-vs-All Strategy
- One-vs-One Strategy
- Multinomial Logistic Regression

Multiclass Classification: One-vs-All

- For C classes, we train C different classifiers
- Classifier c is constructed using 1 as a label for class c and 0 for the remaining classes.
- After training, the classifier with highest value/confidence is chosen at test time
- Potential issue: We might have class imbalance depending on C.

Multiclass Classification: One-vs-All

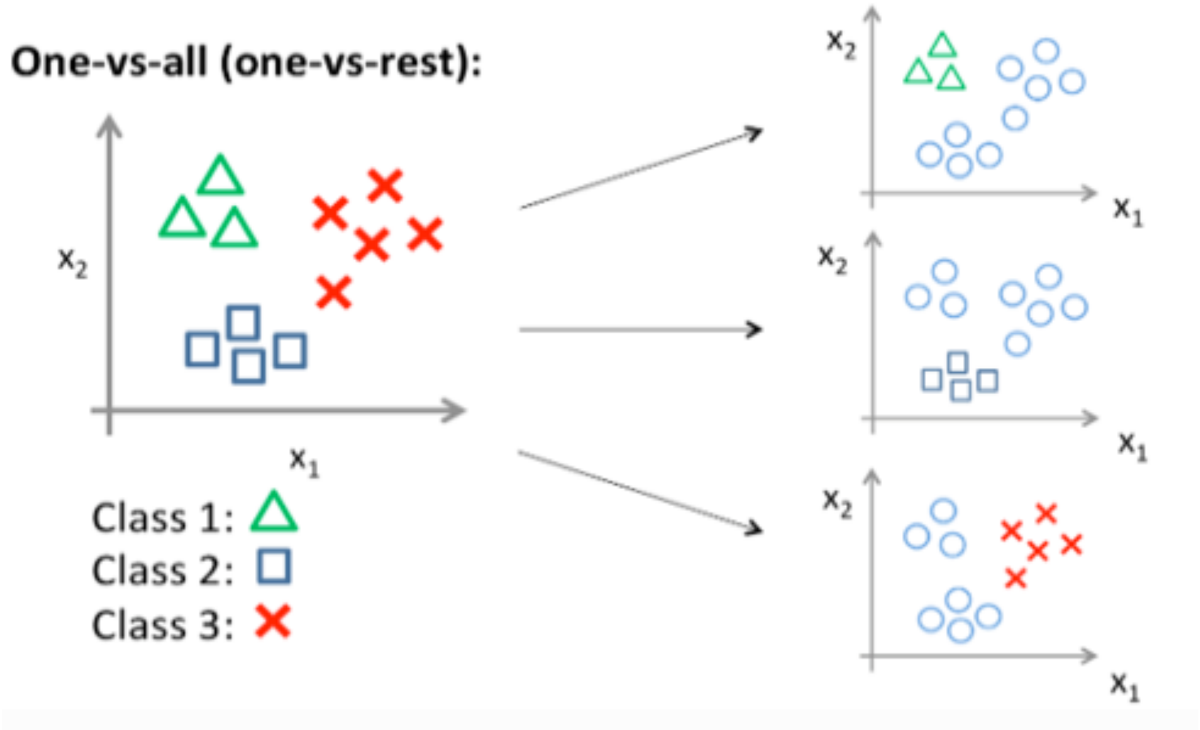

Photo credit: https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj4/html/jnanda3/index.html

Multiclass Classification: One-vs-One

- Create $\binom{C}{2}$ binary classifiers. Construct classifier c_{ij} using training data only from class i and j.
- For each new test point, evaluate all $\binom{C}{2}$ classifiers
- Usually, the outcome is decided using a voting scheme
- If you get equal votes for some classes, a tie-breaking rule is needed
- Each classifier uses much less data compared to One-vs-All
- Potential issue: We might need to train a lot of classifiers, depending on C.

Multiclass Classification: One-vs-One

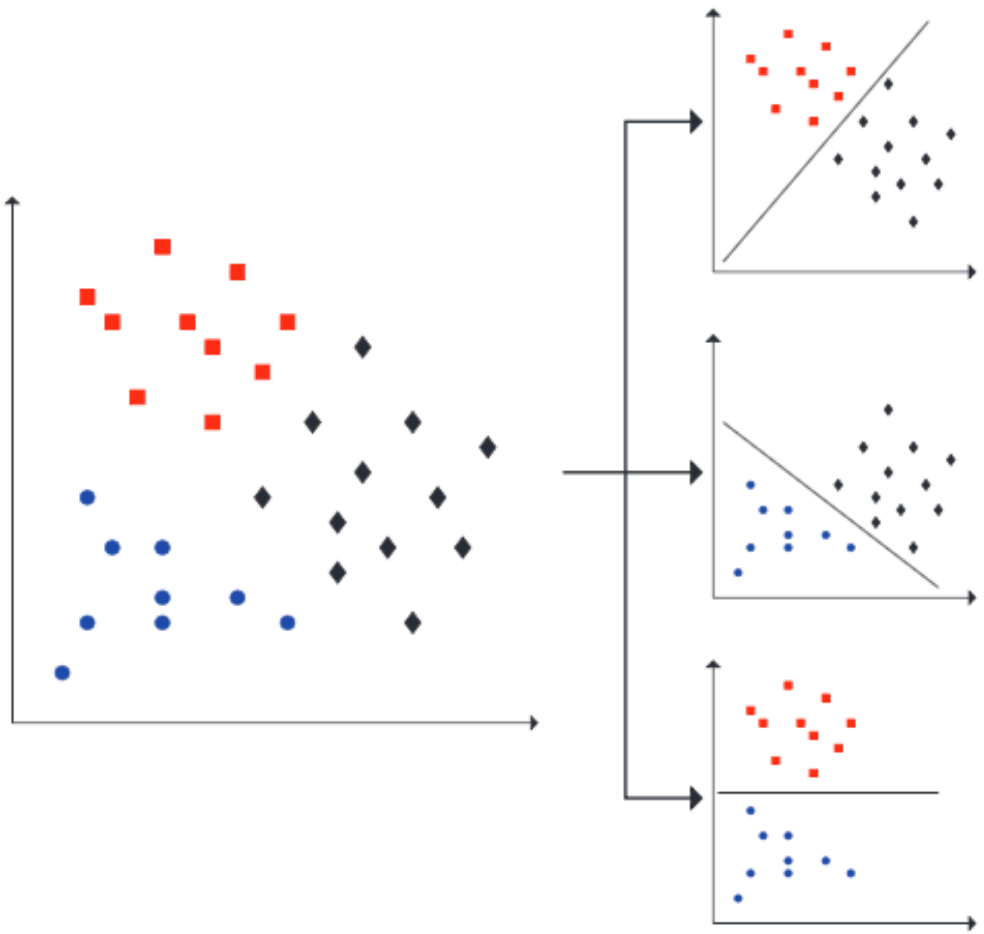

Fig. 1. An example of one-versus-one (OVO) decomposition of a three-class problem into three two-class problems.

Photo credit: Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data, Zhang et. al [2016]

Multinomial Logistic Regression

- Reminder: Binary classification loss

$$L(\theta) = -\sum_{i=1}^{n} y_i \log(p_\theta(x_i)) + (1 - y_i) \log(1 - p_\theta(x_i))$$

- Extend this to C classes

$$L(\theta) = -\sum_{i=1}^{n} \sum_{c=1}^{C} 1_{\{y_i = c\}} \log \left(\frac{\exp(\theta_c^T x_i)}{\sum_{k=1}^{C} \exp(\theta_k^T x_i)} \right)$$

Class Imbalance

- Class imbalance refers to the case when the classes are unequally distributed
- Some classes are under or over represented
- In the task 2, you have 4800 training samples
- Only 600 examples for class 0 and 2, 3600 examples for class 1
- Classic accuracy does not work well in this case
- Use Balanced Multiclass Accuracy!

Balanced Multiclass Accuracy

- Suppose you have the following confusion matrix
- 10 samples from class 1, 70 samples from class 2, 10 samples from class 3

$$M = \begin{bmatrix} 6 & 4 & 0 \\ 4 & 56 & 10 \\ 0 & 3 & 7 \end{bmatrix}$$

- To calculate Balanced Multiclass Accuracy (BMAC), we calculate the true positive rate of each class and take the mean

$$\text{BMAC} = \frac{1}{C} \sum_{c=1}^{C} \text{TPR}_c$$

$$\text{BMAC} = \frac{\frac{6}{10} + \frac{56}{70} + \frac{7}{10}}{3} = 0.7$$

How to deal with Class Imbalance?

Possible solutions to deal with class imbalance:

- Oversampling (Adding some instances from the underrepresented class)
- Undersampling (Remove some instances from the overrepresented class)
- Change your loss function using a cost sensitive classifier:

$$L(\theta) = \sum_{i=1}^{n} l_i(\theta) \quad \Rightarrow \quad L_w(\theta) = \sum_{i=1}^{n} w_i l_i(\theta)$$

Original cost function puts equal weight to every data point

You might want to put more emphasis on the underrepresented class

For example, in sklearn library, you have the parameter "class_weight"

Final Remarks

- You are allowed to use any library
- Pay attention to the class imbalance
- Overfitting to the public score might be dangerous

- Good luck and enjoy the project!