Machine Learning

Multi-class classification

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Multi-class classification vs multi-label classification

- Multi-class classification
  - Each example is assigned to a single class only **from several categories**
  - Image: car, train, bicycle, etc
- Multi-label classification
  - Each example can be assigned to several categories
  - Movie Category: horror, romance, adventure and action
  - Car: Car, Honda, Honda-CRV, SUV car, 4-seats car, etc

Question!

- The customer would like a program that takes an image of a fruit. You would like to learn machine learning model(s) that classify to both the fruit type and its color. For example, red apple, yellow apple, pink panna, etc
  - Assume each fruit has a single label
- What are the different ways to model the problem? Pros/Cons?

Answer!

- One Multi-class classifier
  - Input is an image and its output is a single joint type (e.g. red apple, yellow banna, etc)
  - Cons: you will have many classes and need much more data to cover the space
- Two Multi-class classifiers
  - One classifier for the fruit type and another for the color
  - Cons: more processing
    - With deep network, we can do good network merging
- A Multi-label classifier (not the best to do - confusing)
  - Just let the model learns the 2 types together separately
  - Cons: need careful post-processing
    - E.g. select the highest color probability and the highest fruit probability

Multi-head Deep Network

Shared NN

Fruit NN

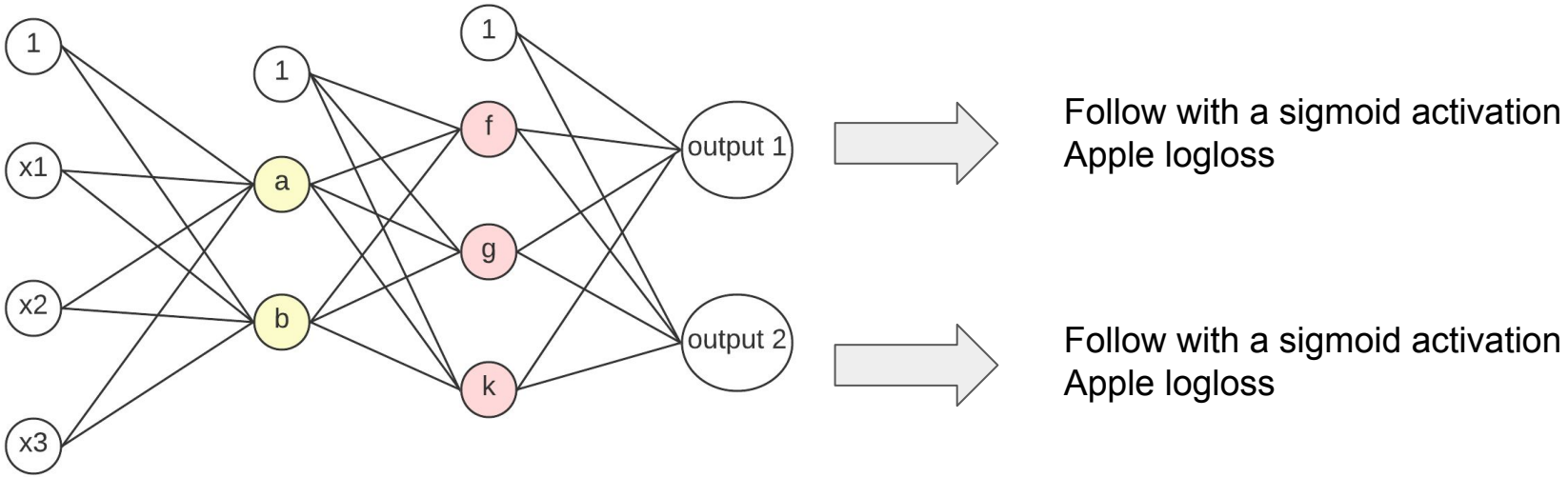
Apple: 0.8

Color NN

Red: 0.75

Extending NN to Multi Label classifier

- Extending NN to multilabel classifier is trivial due to independency
- Just learn jointly N output nodes
- For each node, apply a sigmoid loss



Extending a Binary Classifier to Multiclassifier

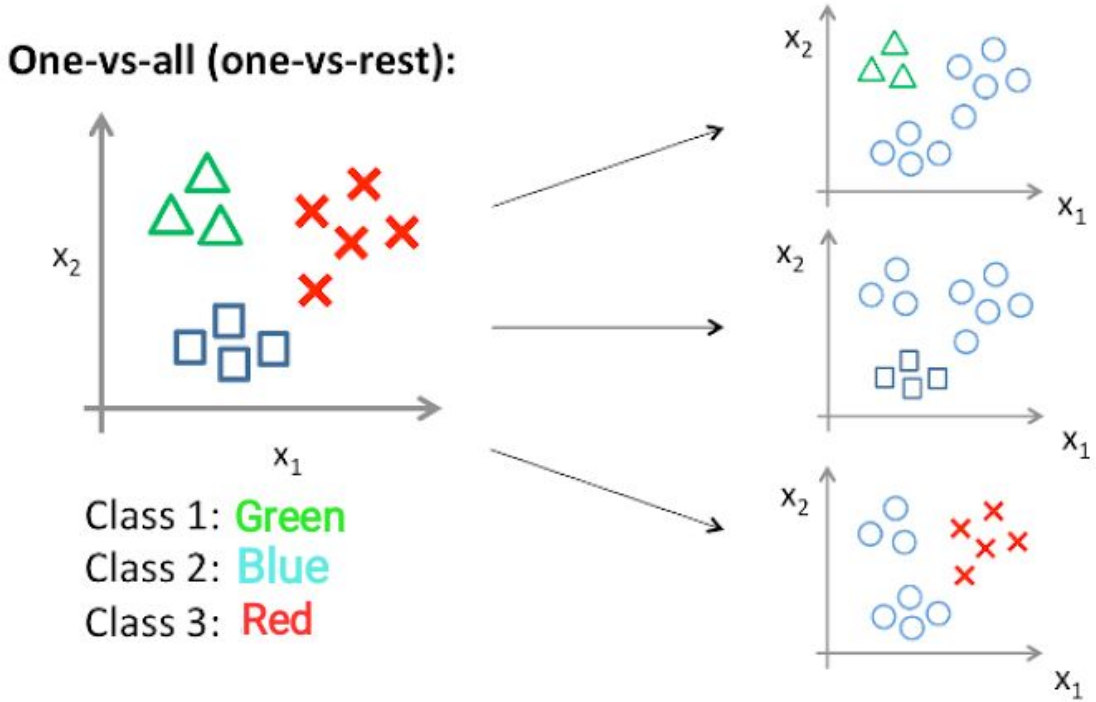
- Extending NN to multiclassifier needs some insights
- Assume we have an available code for (any) binary classification
- How can we extend it as black box to support multiclass classifier?
- For simplicity, Imagine we gave 3 classes A, B and C

One-vs-Rest (OvR)

- For n classes, you create n binary classifiers.
- Each classifier is trained to recognize a **single class** against **all** the other classes.
- For example, in a three-class problem with classes A, B, and C, you would create three classifiers:
  - A vs (B, C) that is: use **labels A as 1** and **labels B and C as 0**
  - B vs (A, C)
  - C vs (A, B)
- Choose the classifier with the highest probability/score

One-vs-Rest (OvR)

- *What are the **pros and cons** of this approach?!*



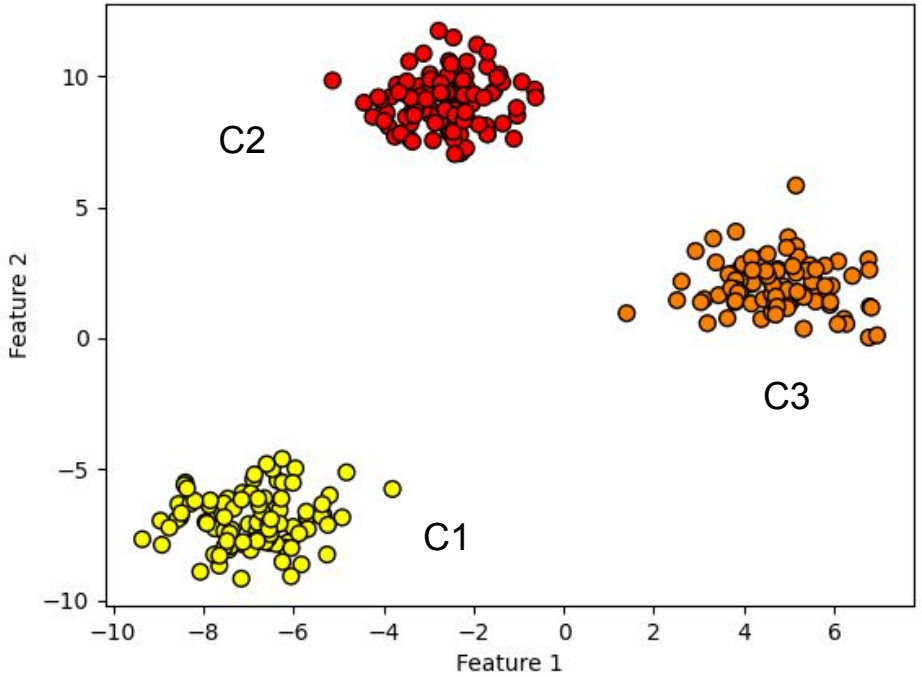
Src: Andrew NG Intro

One-vs-Rest (OvR)

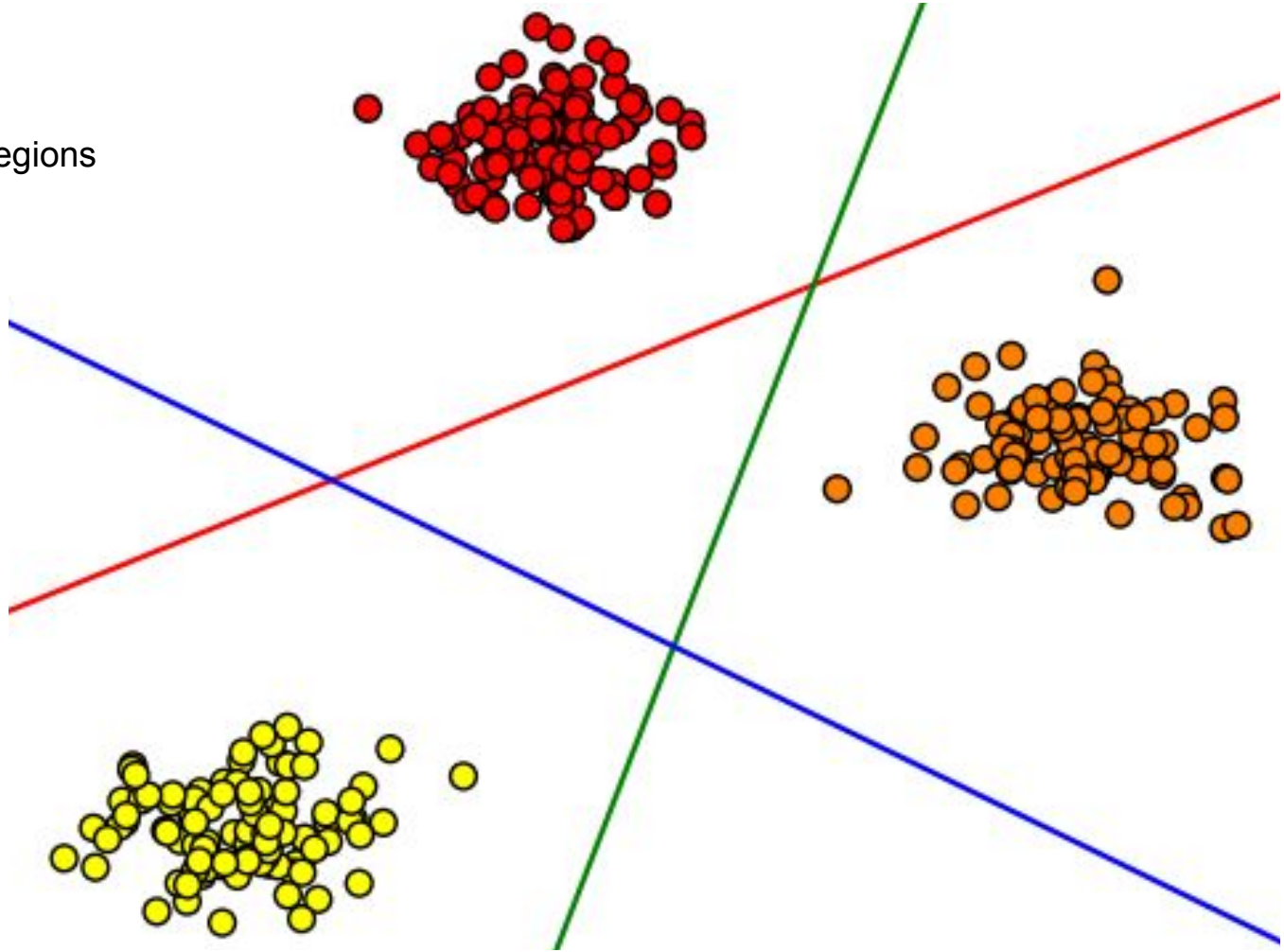
- Pros
  - Simple approach
  - Linearly scalable (N classifiers)
  - Flexible: use any binary classifier
- Cons
  - **Computational** Overhead: N classifiers not one
  - **Class Imbalance**: The rest classes will cause an imbalance!
  - **Calibration**: Getting well-calibrated probability estimates may be more challenging
  - **Inconsistency**: all N-1 classes predicts negative or many predicts positive
- What about the search space?
  - A binary classifier separates the 2 classes with a line
  - What happen with multiple lines?!

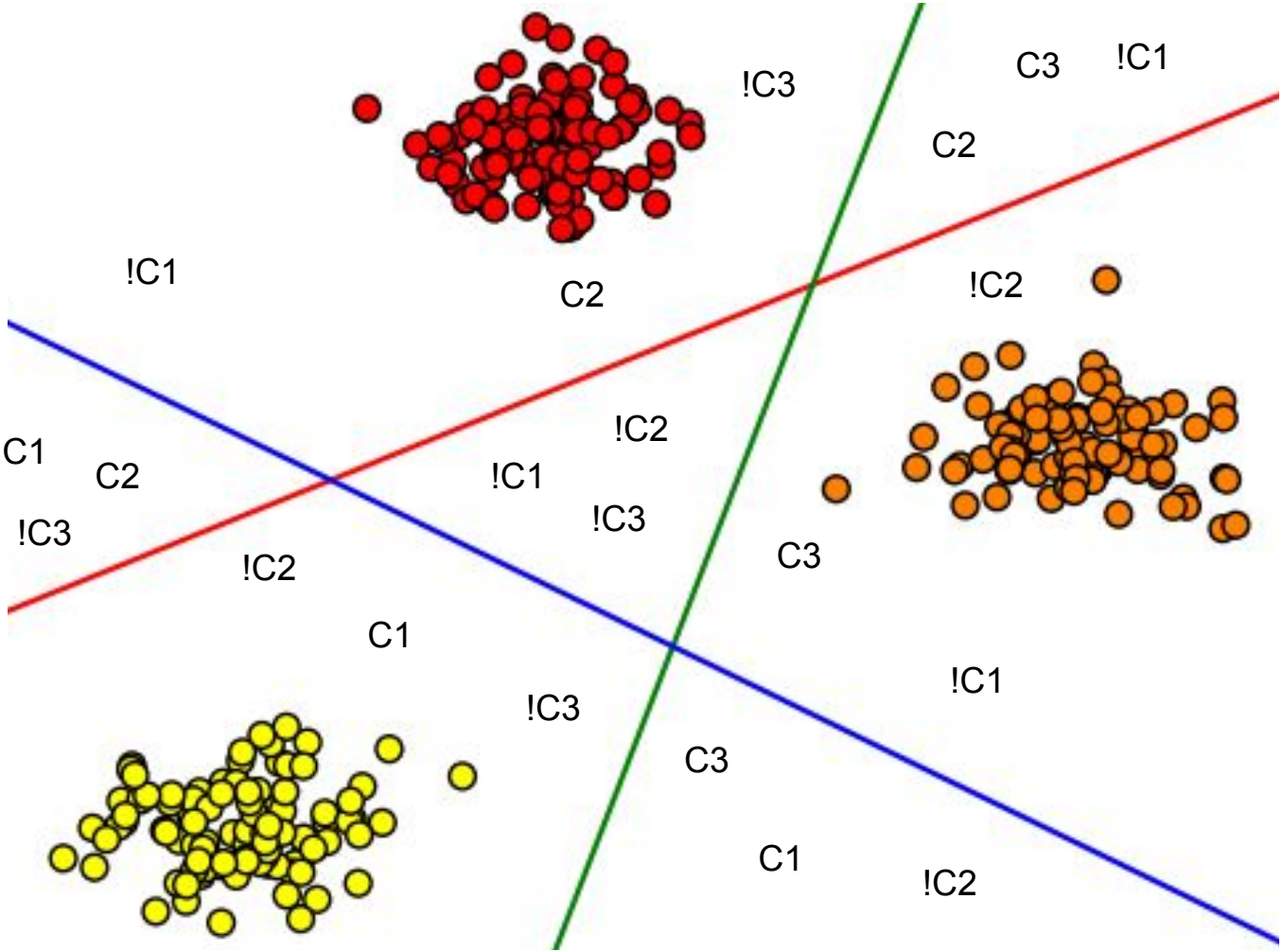
Search Space Investigations

- Assume we have data (x, y)
- They looks like these 3 clusters
- Draw 3 lines to separate them as one-vs-rest
- Identify 2 critical issues

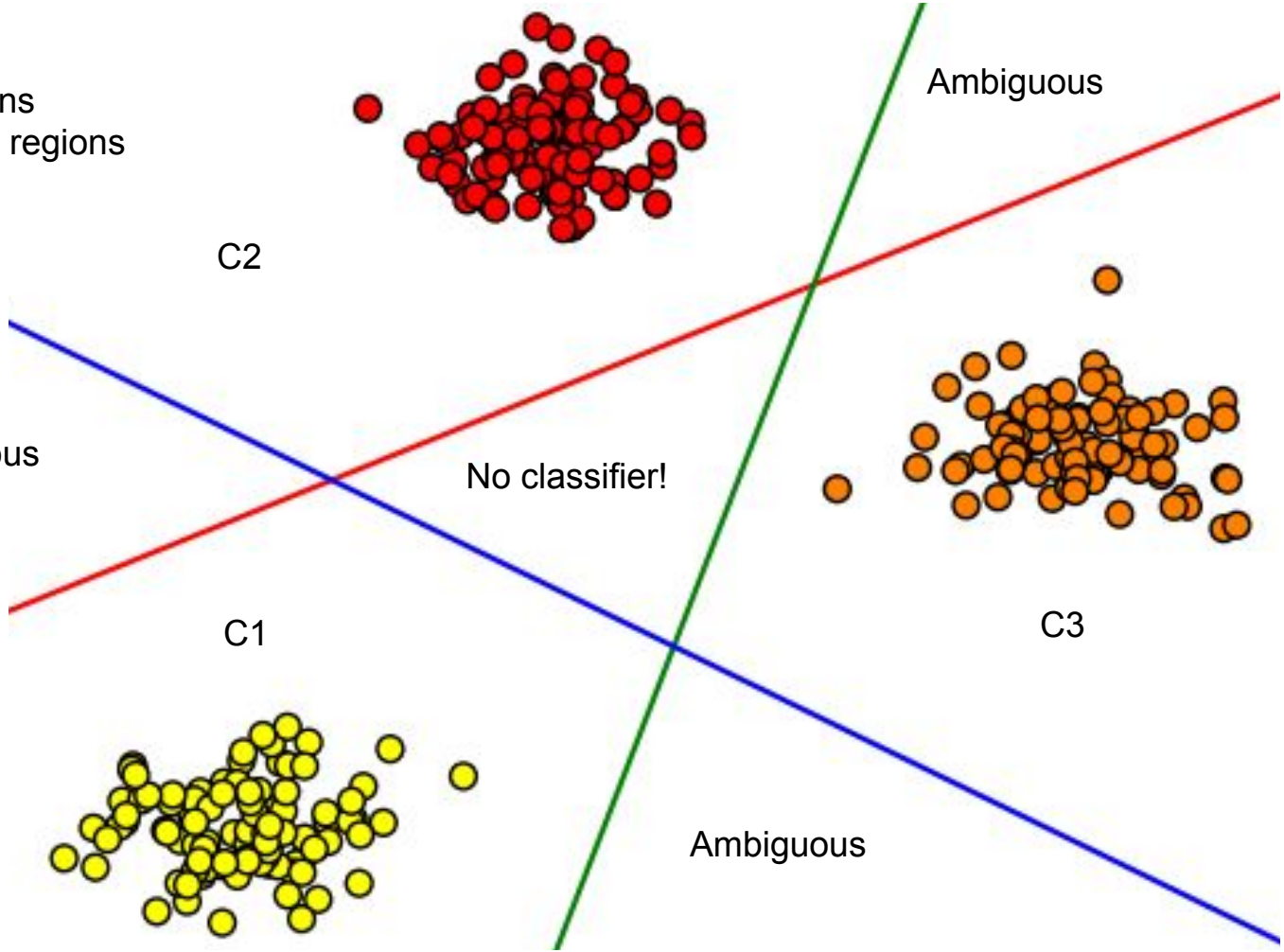


Analyze the 7 regions

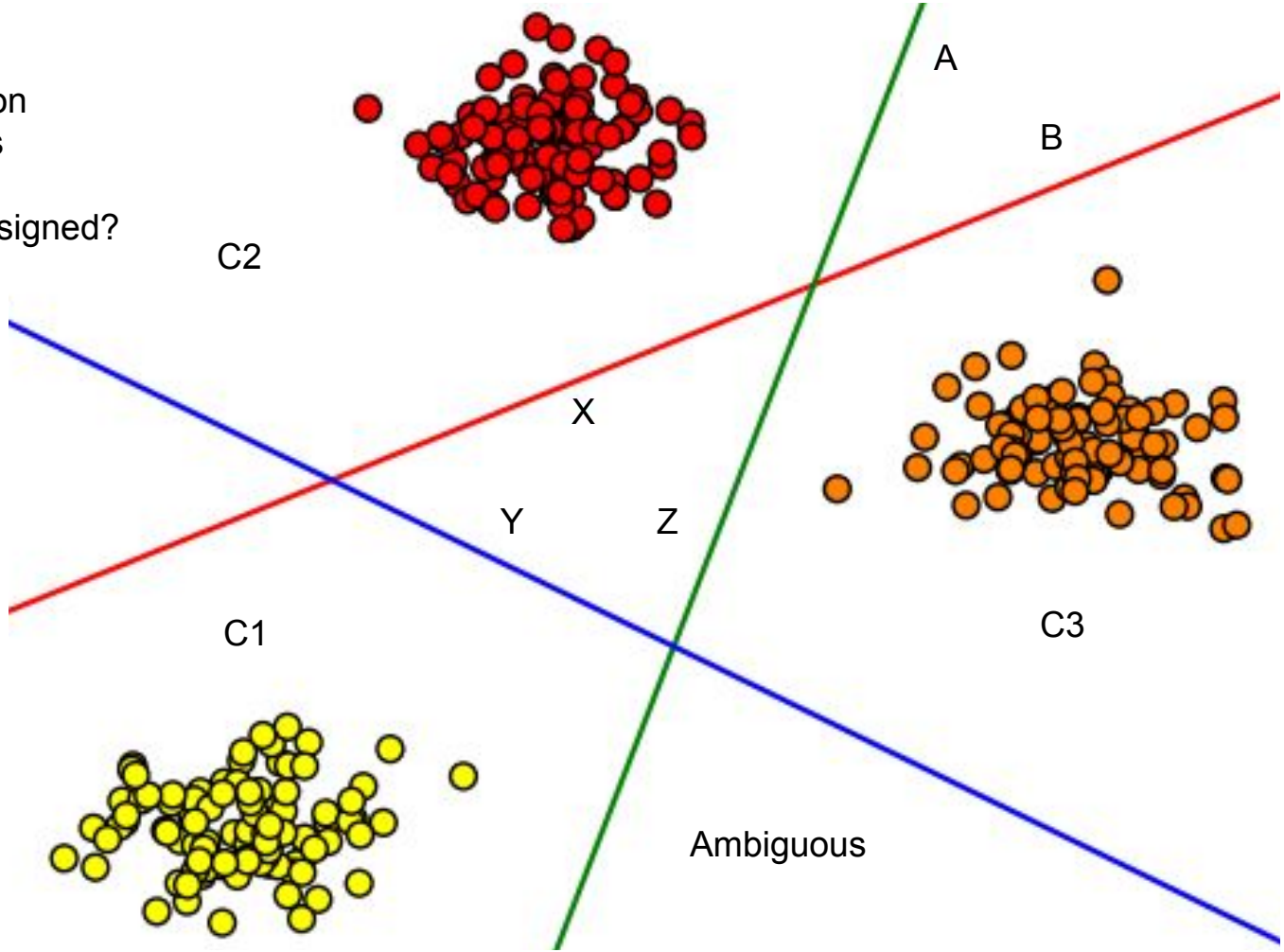


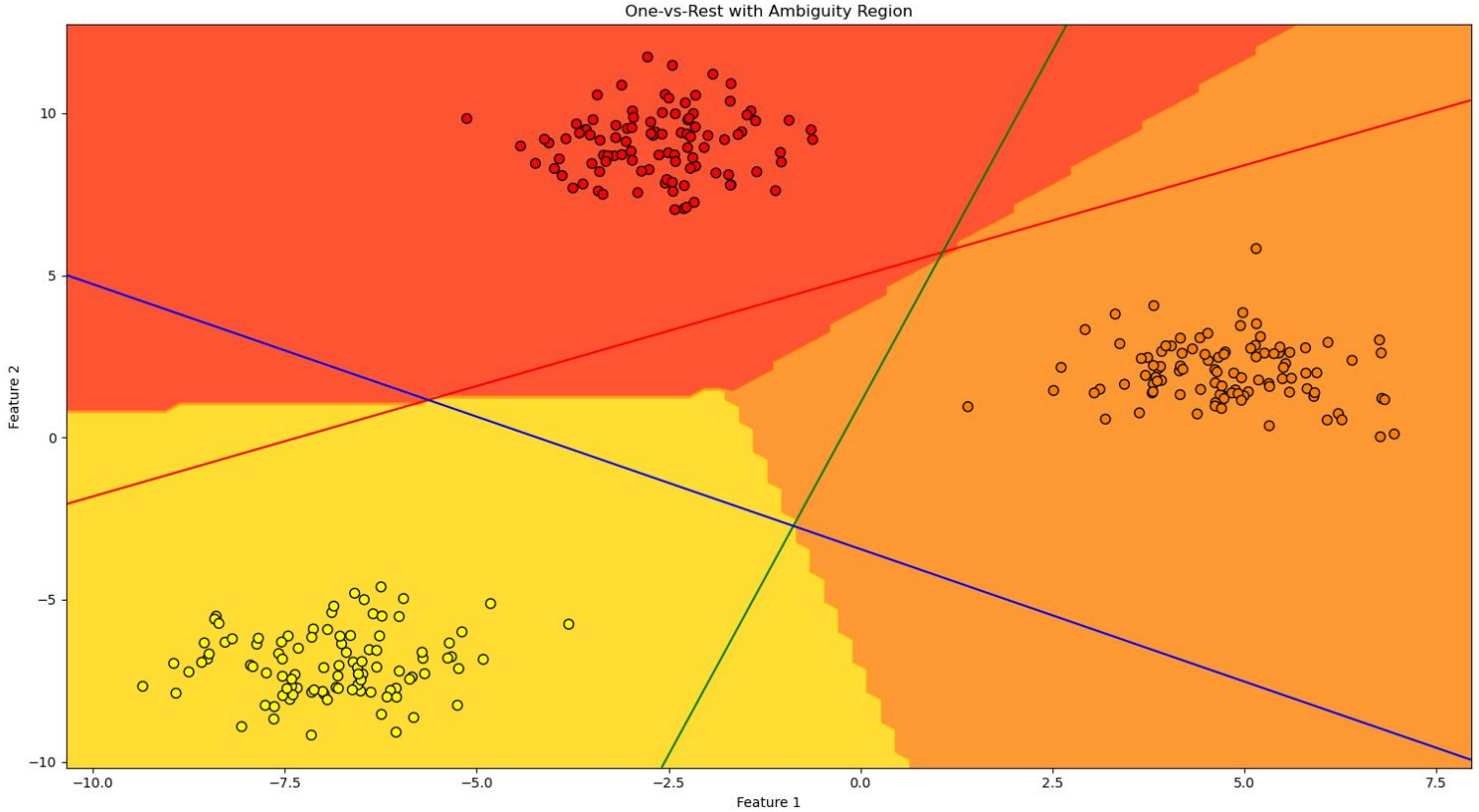

!C3
C3
!C1
C2
!C1
C2
!C2
C1
C2
!C2
!C1
!C3
!C3
C3
!C2
C1
!C1
!C3
C3
C1
!C2

- 3 Clear regions
- 3 Ambiguous regions
- 1 NA



- In your opinion
- Where points
  A, B, X, Y, Z
  should be assigned?





What if we assigned each point to the classifier with the **maximum** score/probability?

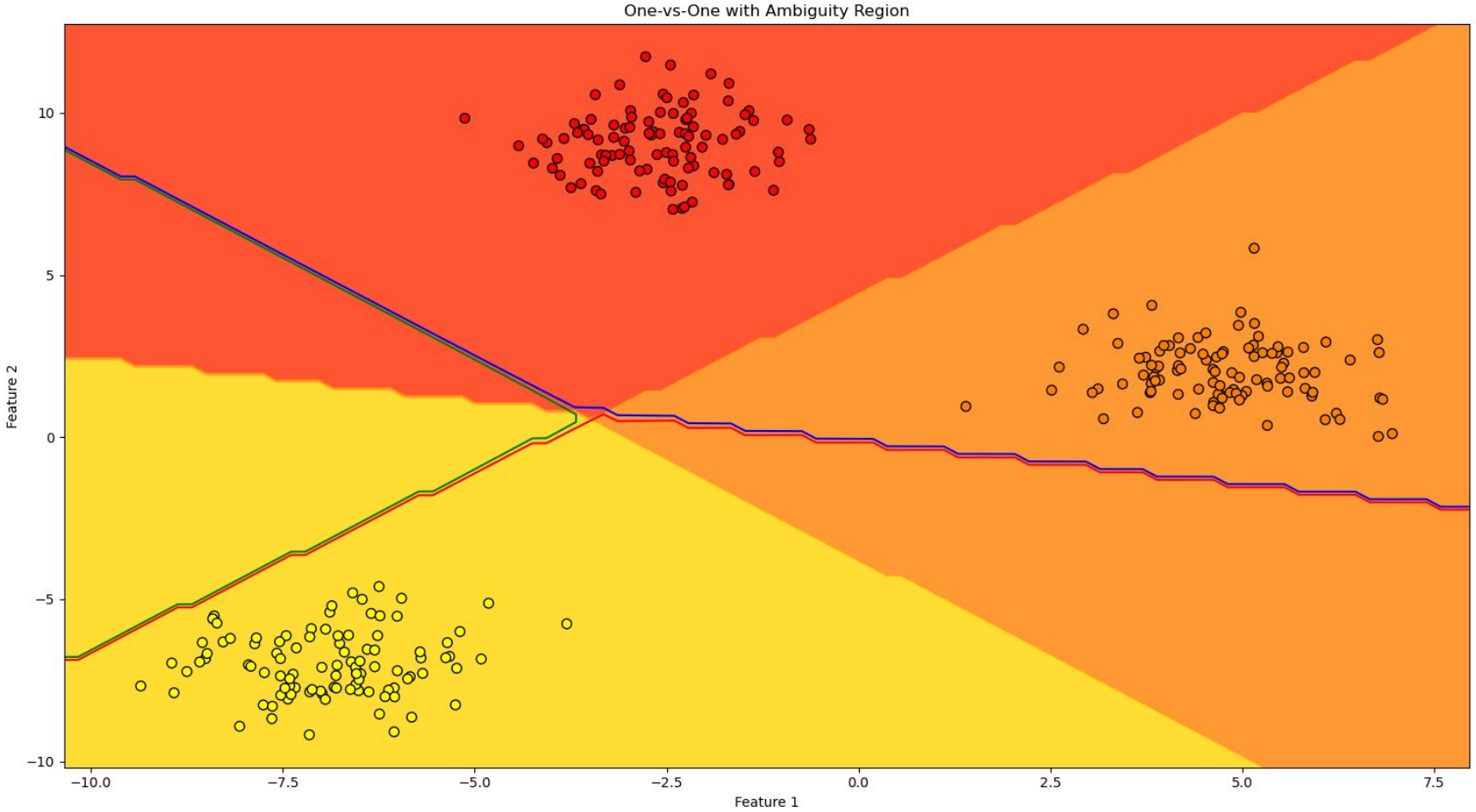
- The decision regions looks singly **connected and convex** (*See bishop 4.1.2*)
- Observe, ambiguous regions intuitively are split into halves. No area split into 3

One-vs-One (OvO)

- Create a classifier between every pair of the n classes [n * (n-1) / 2]
- For classes A, B, C, D
  - A vs B
  - A vs C
  - A vs D
  - B vs C
  - B vs D
  - C vs D
- **Voting**: use a voting system to make a prediction

One-vs-One (OvO)

- Pros
  - **Discriminative** Training: Binary classifier between 2 classes only
  - Per pair of classes, faster training
  - No imbalance in the training if the dataset is balanced
- Cons
  - n* (n-1) classifiers ⇒ a lot of models to manage, memory issues, scalability issues
  - Ambiguous areas (like OvR)
    - Feel free to draw. For each line mark Cx vs Cy
  - Possible ambiguity in Voting (more than a class with same vote)
  - Calibration


One-vs-One with Ambiguity Region
Feature 2
Feature 1
10
5
0
−5
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

What else?

- While there are some other ideas
- The most dominant approach is based on a **single model** where:
  - Learns **jointly** N linear functions (each is W^tX)
  - Assign the final class to the one with the **maximum** score (to solve ambiguity)
- How can we extend NN from a binary classifier to such Multi-Classifier?!
    - We can easily make N output nodes
    - But the key challenge is the loss function!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."