KU x depa
MACHINE LEARNING
Training Course 2022

Logistic Regression

DR. SETHAVIDH GERTPHOL

KASETSART UNIVERSITY

Today's Outline

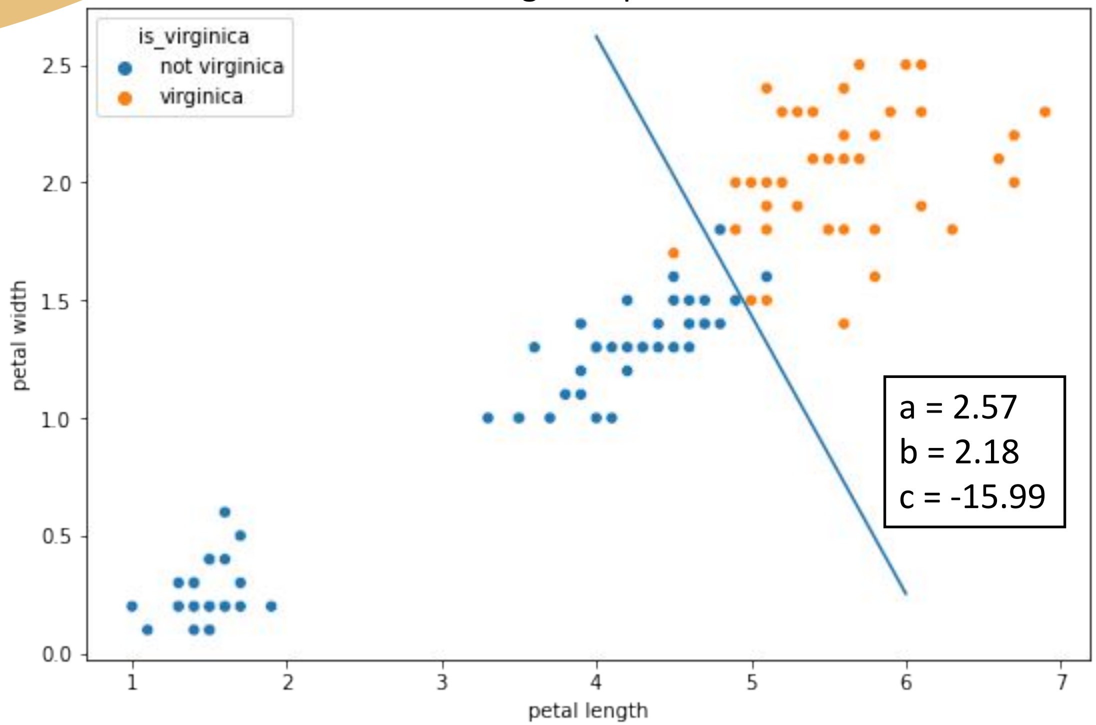
- Concept of Regression technique for classification
    - Decision boundary
    - Parameter randomization and optimization
    - Learning rate
    - Adjustment based on error
- Example using scikit-learn

Logistic Regression

- A technique for classification
- Create a linear decision boundary
  - Points on or above this boundary are predicted as one class
  - Points below this boundary predicted as the other class
- Use optimization technique to find best boundary

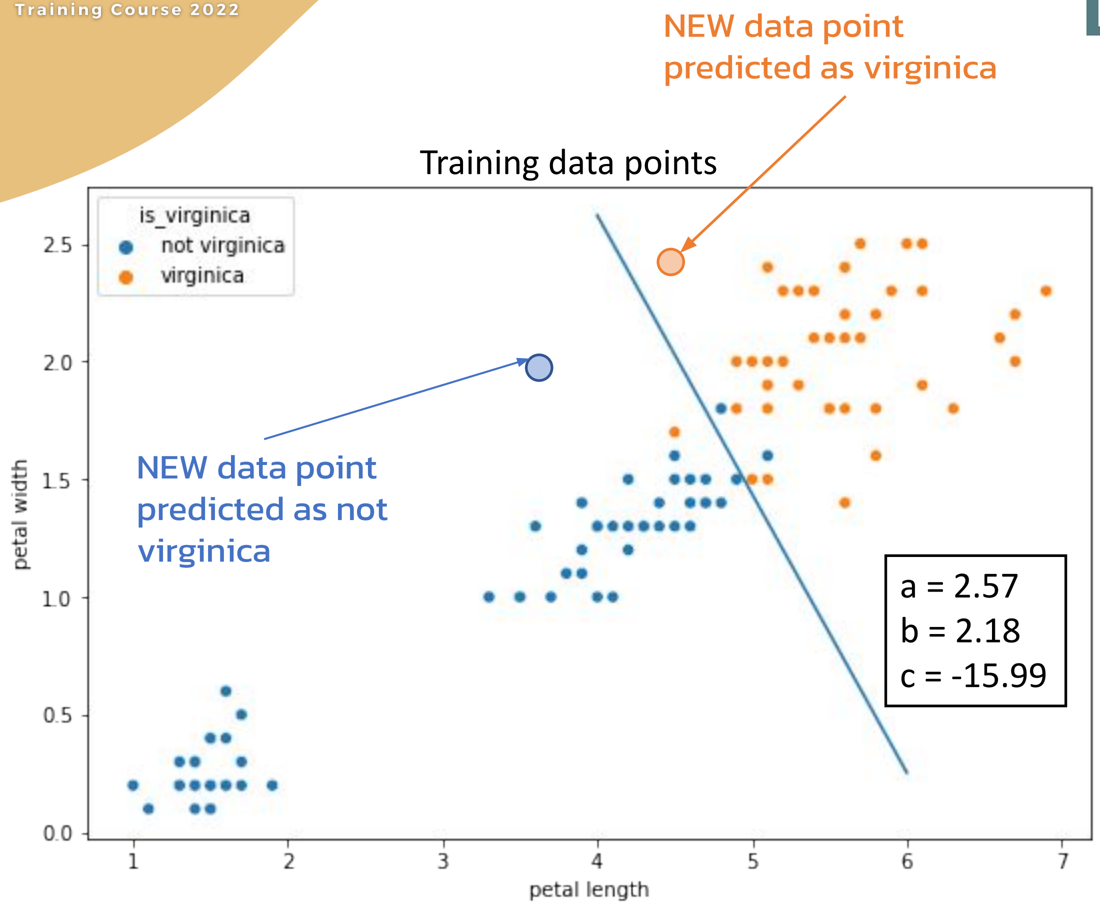
Line and weight vector

Training data points



- A line can be described using an equation y = ax+c where
  - x and y are two variables
  - c is an intercept of the line
  - a is the slope of the line
- Rearrange the equation to be a*x + b*y + c*1 = 0
  - a, b, c is called the weight (or coefficient) vector

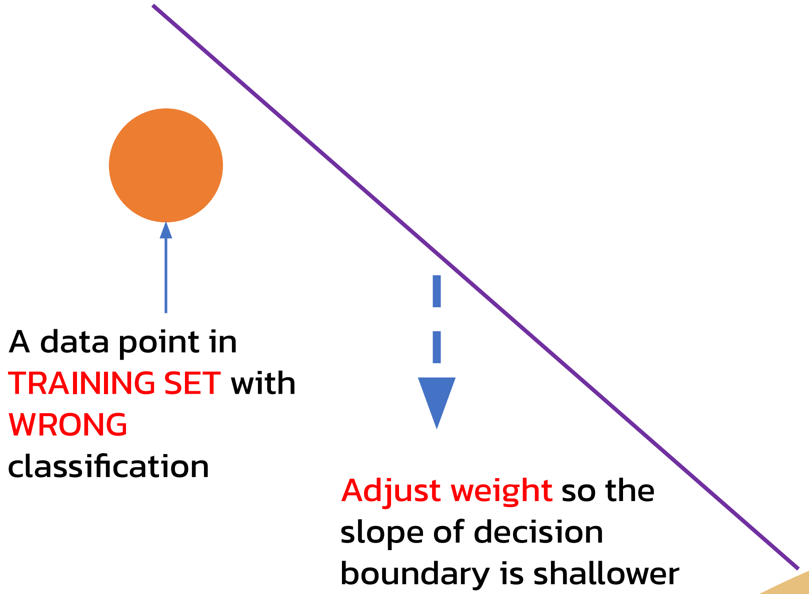
Decision boundary



- Suppose that we have a new data point x2, y2
  - If a*x2 + b*y2 + c*1 > 0, then the data point is above the line, and we can predict that it is of one class
  - If the data point is below the line, we can predict that it is of the other class
- So this line is a linear decision boundary

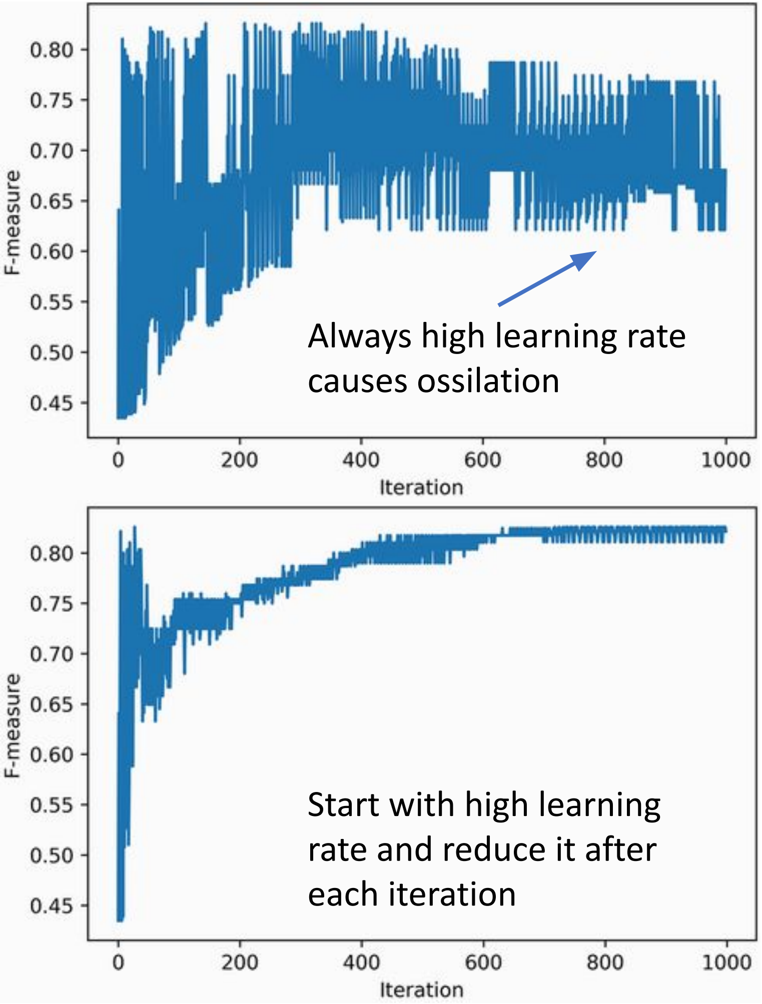
How to find a "good" weight vector?

1. Randomize initial values of weight factor
2. Loop a lot of times (ex. 1000 times) or until almost no change
    1) Calculate the accuracy (or other measure) of classification on the training set data points one-by-one
    2) Adjust the weight to improve the accuracy



How much weight to adjust?

- Learning rate: the rate of weight adjustment as a fraction of unit e.g., 0.1 or 0.01
- High rate: fast adjustment, but may overshoot
- Low rate: smaller adjustment, take longer time to reach good solution
- One technique: start with high rate then lower the rate after each loop



Model confidence

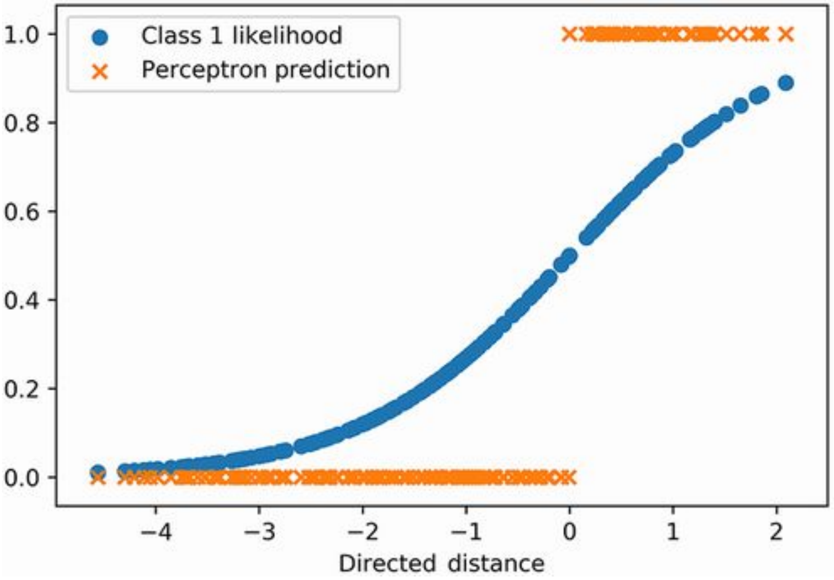
Class 1 likely

Class 1 maybe?

may not be class 1

- So far: simple perceptron (0–1 decision)
- Problem: no context about confidence of model prediction
  - Model should be less confidence if data point lies close to the decision boundary
- How can we calculate confidence?

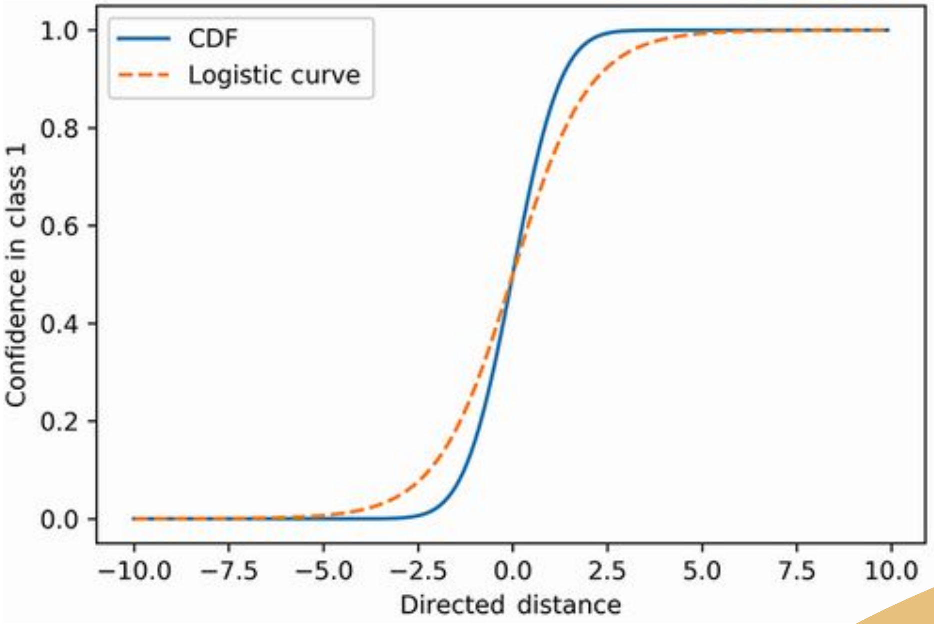
Probably not class 1



Logistic function

$$\text{confidence}(\text{distance}) = \frac{1}{1 + e^{-\text{distance}}}$$

- Characteristic of needed function
  - Zero distance from the decision line equation results in 50% confidence with no adjustment
  - Large distance from the decision line equation gives large, stable positive/negative confidence adjustment
  - Confidence adjusted linearly between the two extremes
- Possible 2 functions
  - CDF (cumulative density function)
  - Logistic function
  - distance is standardized



Using confidence to adjust weight

- Model confidence can be used to adjust weight vector
- If the model is confidence but wrong, we can strongly adjust the weight
- Weight adjustment is now a function of learning rate, iteration, confidence of prediction, and actual value
- More consistent model

?

- Predict class 1 with high confidence
- Data point is actually class 0
- large weight adjustment

?

- Predict class 1 with low confidence
- Data point is actually class 0
- small weight adjustment

Benefits and Limitations

- Works well where difference between classes follow mostly linear boundary
- Easy to use and give respectable performance, often used as baseline model
- Works with numeric data only
  - Must convert categorical values using one-hot encoding
- Don't forget to standardize or normalize numeric data

Sklearn

from sklearn.

Scikit-learn implementation

DEMO

Reference

- Leonard Apeltsin, "Data Science Bookcamp", Manning Publications, November 2021.