DSC 10, Spring 2018
Lecture 25

Classification I

sites.google.com/eng.ucsd.edu/dsc-10-spring-2018

Credit: Anindita Adhikari and John DeNero

Announcements

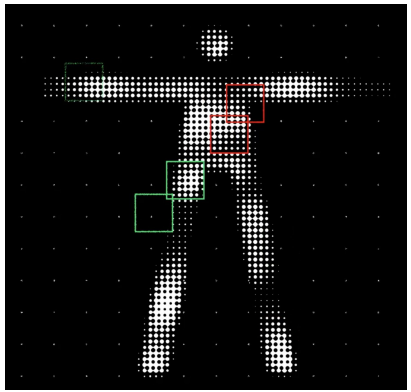
- HW 9 due Sunday - the final homework!
- Lab 10 due Wednesday
- Project 10 due Saturday of next weekend
  - Please re-click download link to get updated tests
- My office hours are tomorrow, 3-5pm

Regression

- Estimating/predicting a numerical response variable, y, based on other predictor variable(s), x
- Since **y** is numerical, makes sense to have predictions like “y will be between 13.8 and 15.1”

But what if y were categorical? How would you predict it?

Classification Examples



Classification

- Response variable is categorical; values are classes
- Binary response: Only two classes, 0 and 1

- Try to classify the response into one of the categories, based on:
  - Values of predictor variables, called attributes
  - Training set of data in which the classes of the individuals are known

Nearest Neighbor Classifier

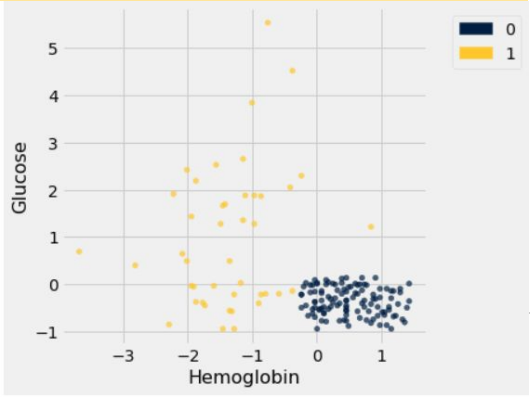
- New individual, unknown class
- Find individual in training set “closest” to this new individual
  - That’s the new individual’s “nearest neighbor”
- Assign the new individual the same class as the nearest neighbor

(Demo)

Discussion Question

What kind of test results will lead you to conclude that a patient does not have Chronic Kidney Disease (“0”)?

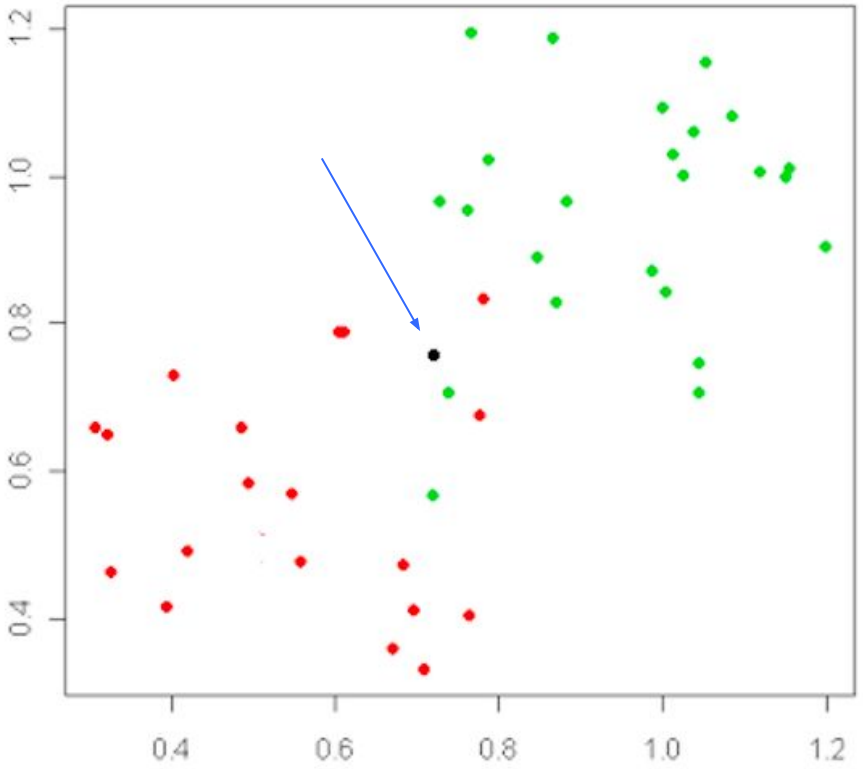
A: Hemoglobin below average Glucose below average
B: Hemoglobin above average Glucose below average
C: Hemoglobin below average Glucose above average
D: Hemoglobin above average Glucose above average
E: More than one possible answer



k-Nearest Neighbor Classifier

- New individual, unknown class
- Find the *k* closest individuals in the training set
  - They are the new individual's "*k* nearest neighbors"
- Assign the new individual the same class as the majority of the *k* nearest neighbors (*k* is usually taken to be an **odd** number)

Discussion Question



How would the unknown (black) point be classified by a

	1-nn classifier?	**3**-nn classifier?
A.	Green	Green
B.	Red	Red
C.	Green	Red
D.	Red	Green
E.	None of the above	

Implementing the Classifier

The Classifier (The Big Picture)

To classify a point:

- Find its *k* nearest neighbors
- Take a majority vote of the *k* nearest neighbors to see which of the two classes appears more often
- Assign the class that wins the majority vote

Rows of Tables (The Details)

Each row contains all the data for one individual

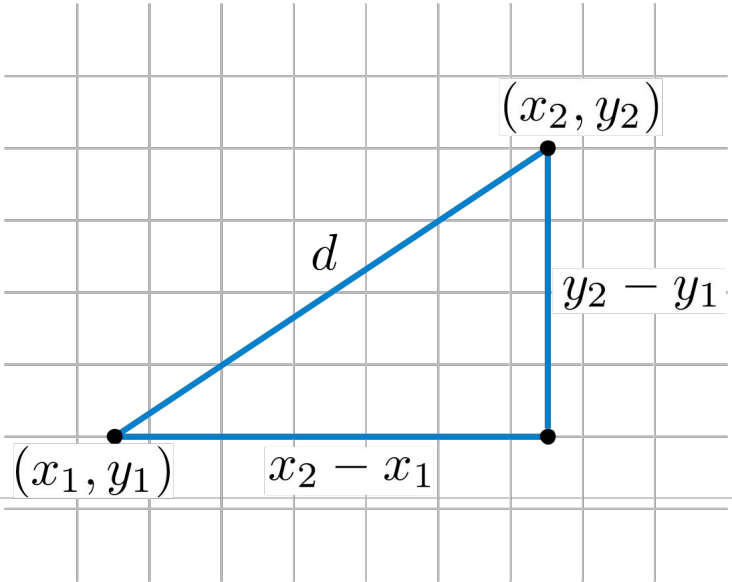
- `t.row(i)` evaluates to `i`th row of table `t`
- `t.row(i).item(j)` is the value of column `j` in row `i`
- If all values are numbers, then `np.array(t.row(i))` evaluates to an array of all the numbers in the row.

(Demo)

Distance Between Two Points

- Two attributes *x* and *y*:

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$



Distance Between Two Points

- Three attributes *x*, *y*, and *z*:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

- and so on ...

(Demo)

Finding the *k* Nearest Neighbors

To find the *k* nearest neighbors of a new point:

- Find the distance between the new point and each point in the training set
- Augment the training data table with a column containing all the distances
- Sort the augmented table in increasing order of the distances
- Take the top *k* rows of the sorted table

Taking a Majority Vote

To find the class to assign the new point:

- Find the majority of the top k rows' classes
- Assign this class to the new point

Discussion Question

```
def majority(topkclasses):
    ones = topkclasses.where('Class', are.equal_to(1)).num_rows
    zeros = topkclasses.where('Class', are.equal_to(0)).num_rows
    if ones > zeros:
        return 1
    else:
        return 0
```

How could you implement the majority function in one line of code?

A. return topkclasses.group('Class', max).sort('Class', descending=True).row(0).item(1)

B. return topkclasses.group('Class', max).sort('Class', descending=False).row(0).item(0)

C. return topkclasses.group('Class').sort('count', descending=True).row(0).item(0)

D. return topkclasses.group('Class').sort('count', descending=False).row(0).item(0)