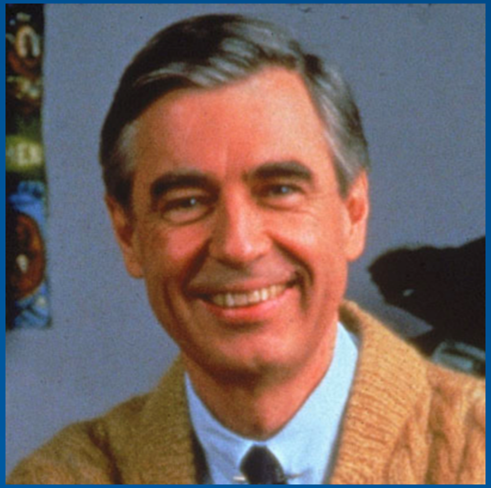
Week 2 : Clusters

RRRRRRRRRRRRRRRRRRRRRR! (I still like Pirates)

k Nearest Neighbors



Won’t you be my neighbor?

k Nearest Neighbor (kNN)

- Lazy algorithm
- Does no ‘learning’
- Still effective
- No loss metric
- Measure with validation or out of fold set
- Uses a distance function

kNN: Democracy in action

- Looks at a number of neighbors and takes a ‘vote’ for class
  - Regression just takes the known target value
- Winner is the one with most ‘votes’
  - Regression takes average of target value

Side note on computation

- If there are n samples there will be $(n^2-n)/2$ calculations
  - 10 points = 45 calculations
  - 100 points = 4950 calculations
  - Large n ~ $n^2/2$
- BUT remember: we do cross validation
  - 5 fold validation (each point used 4 times)
  - $2(n^2-n)$
    - 100 points: 19,600 calculations
- Not only must you compute, you must STORE the results
  - 32 bit float ~ 8.5 million points per gigabyte
  - And sort for each point! (n sorts)

It gets worse….

- No way to ‘transfer’ the model without the data
- No way to ‘save’ the model for re-use
- Regression takes the average of local values (no prediction)
- Boundary Conditions / Outliers cause extra problems

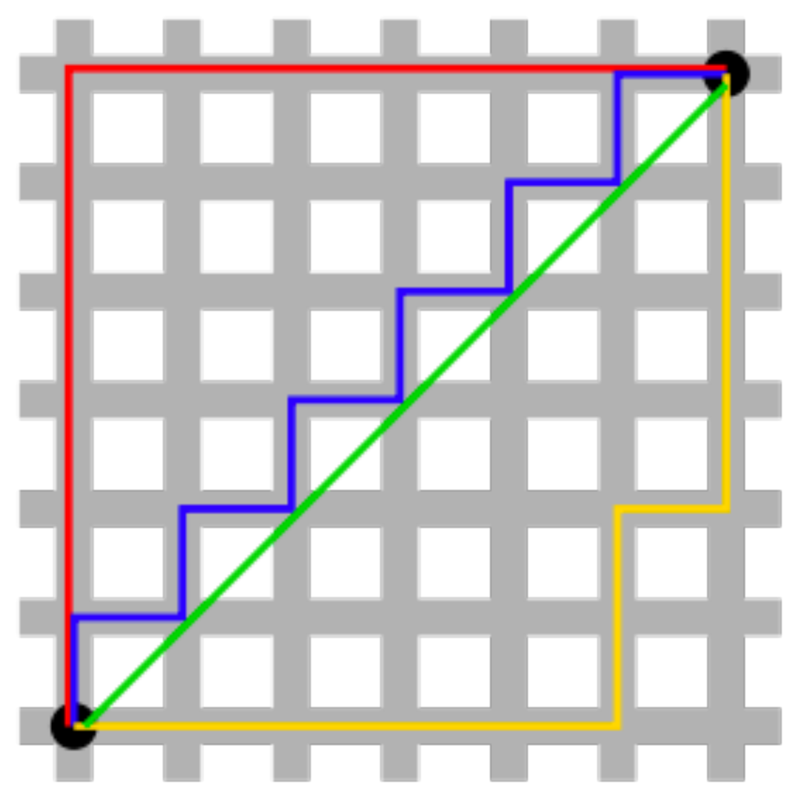
What do we mean by "nearest"?

- Euclidean distance (in N dimensions)
  - $\sqrt{\sum(p_i - q_i)^2}$
  - p_i is the current point and q_i is the point of comparison
- Manhattan distance (grid distance)
  - $\sum|p_i - q_i|$
- Minkowski distance (general case of the other two)
  - $(\sum|p_i - q_i|^m)^{\frac{1}{m}}$

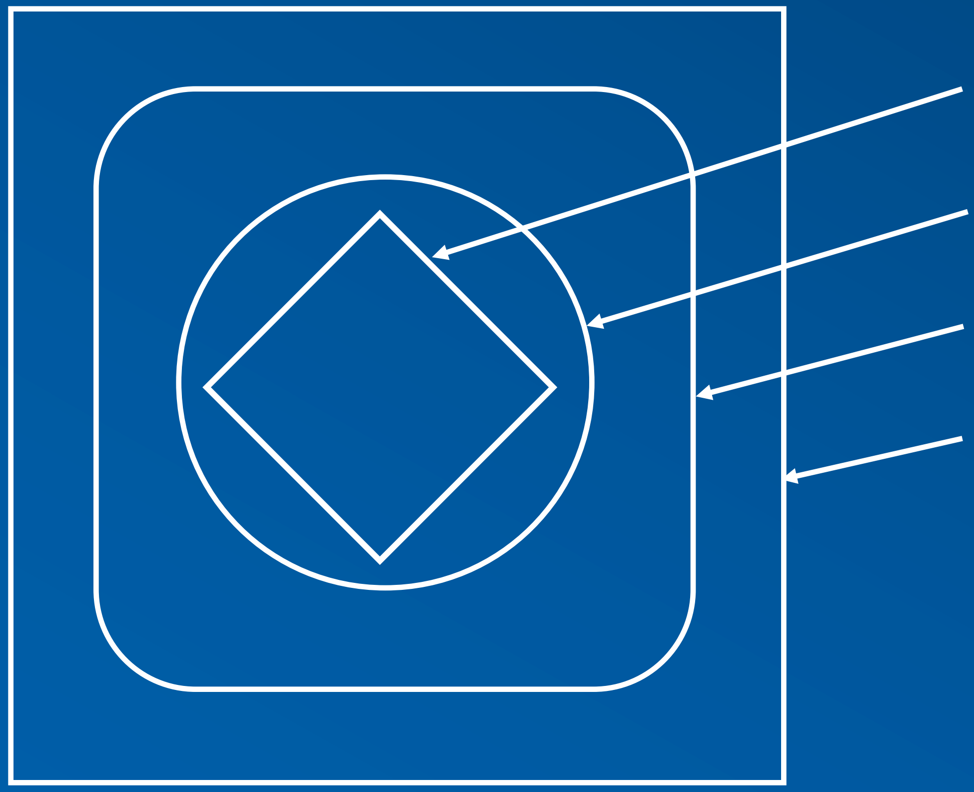
Distance functions visualized

Green = Euclidean

Other = Manhattan



‘Circles’ in higher m



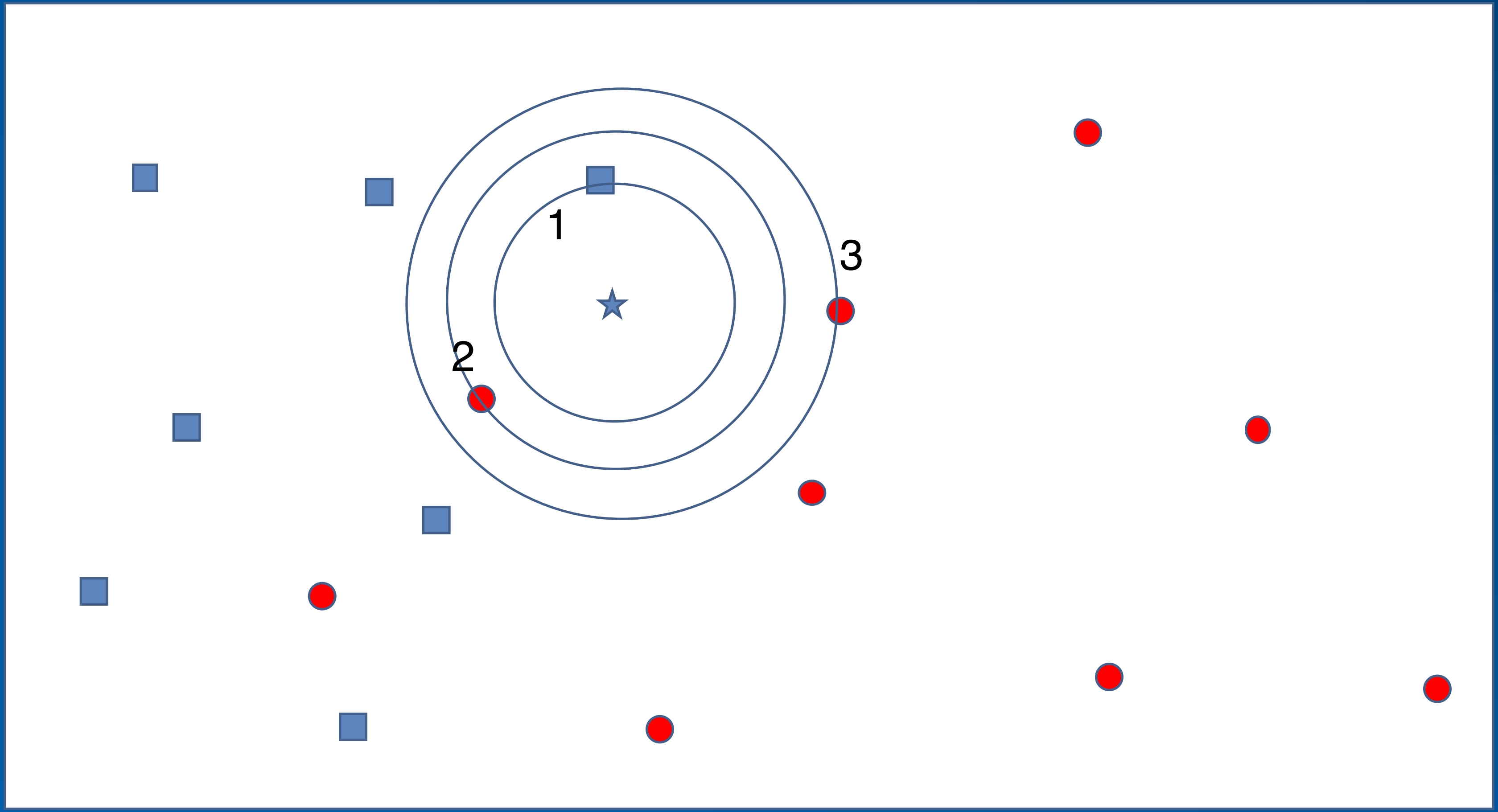
m = 1 – networks

m = 2 – normal distance

m = 4

m = ∞ – warehouse /storage

Visual example (Euclidean 2-D distance)



K = 1 (Blue)
K = 2 (Tie)*
K = 3 (Red)

- Even value k can lead to ties

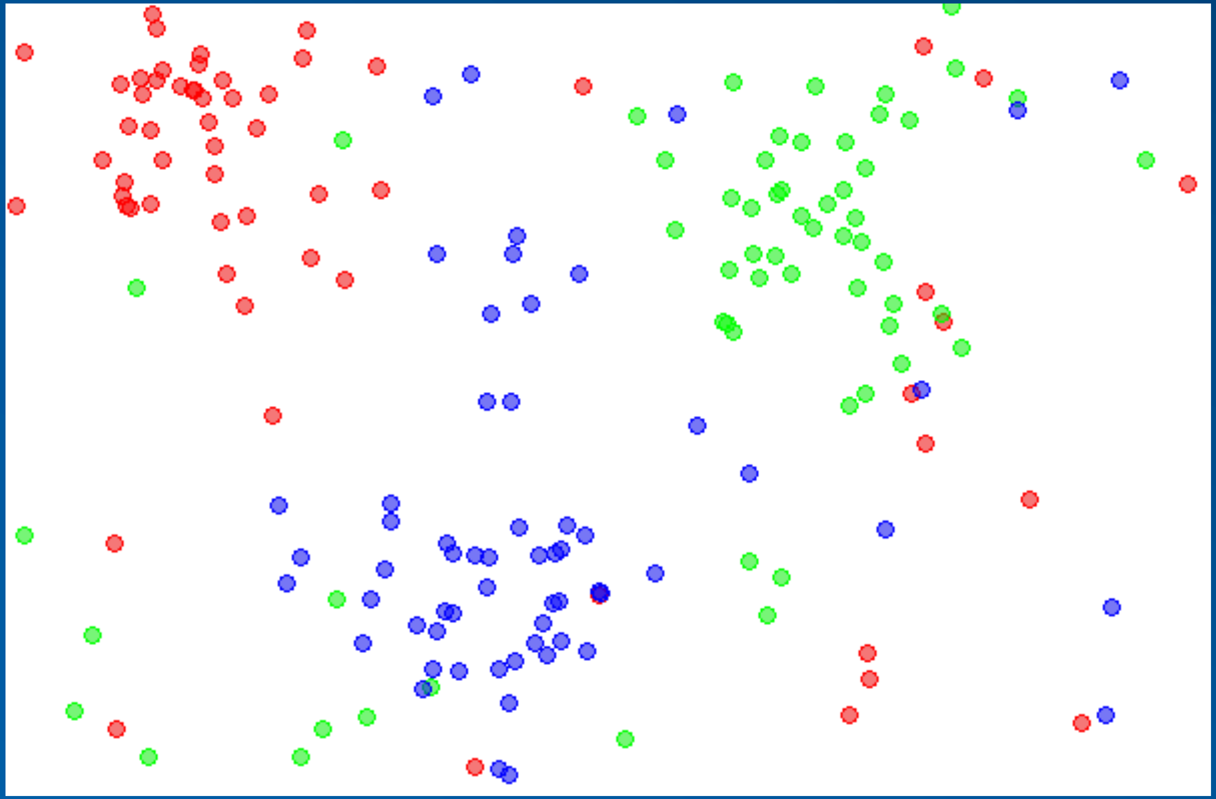
Choosing value of k

- $K = 1$ → severe overfitting, poor generalization
- $k \sim n$ = number of samples → severe underfitting, majority rules
- There is no generalized loss, so use accuracy or some other score metric

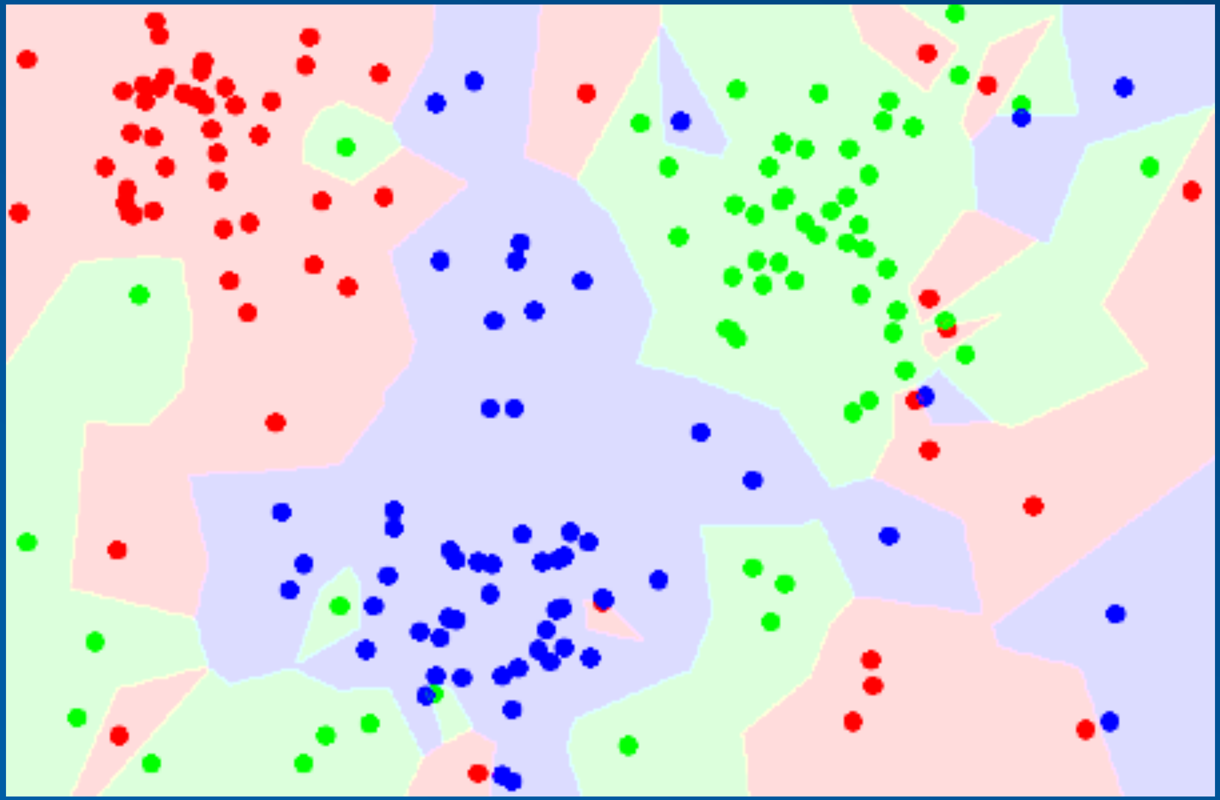
Weighted kNN

- Weight the ‘vote’ by the distance
- The farther away, the less the ‘vote’ counts
- Class/distance
  - Whatever the value of ‘class’, larger distance causes the class value to decrease

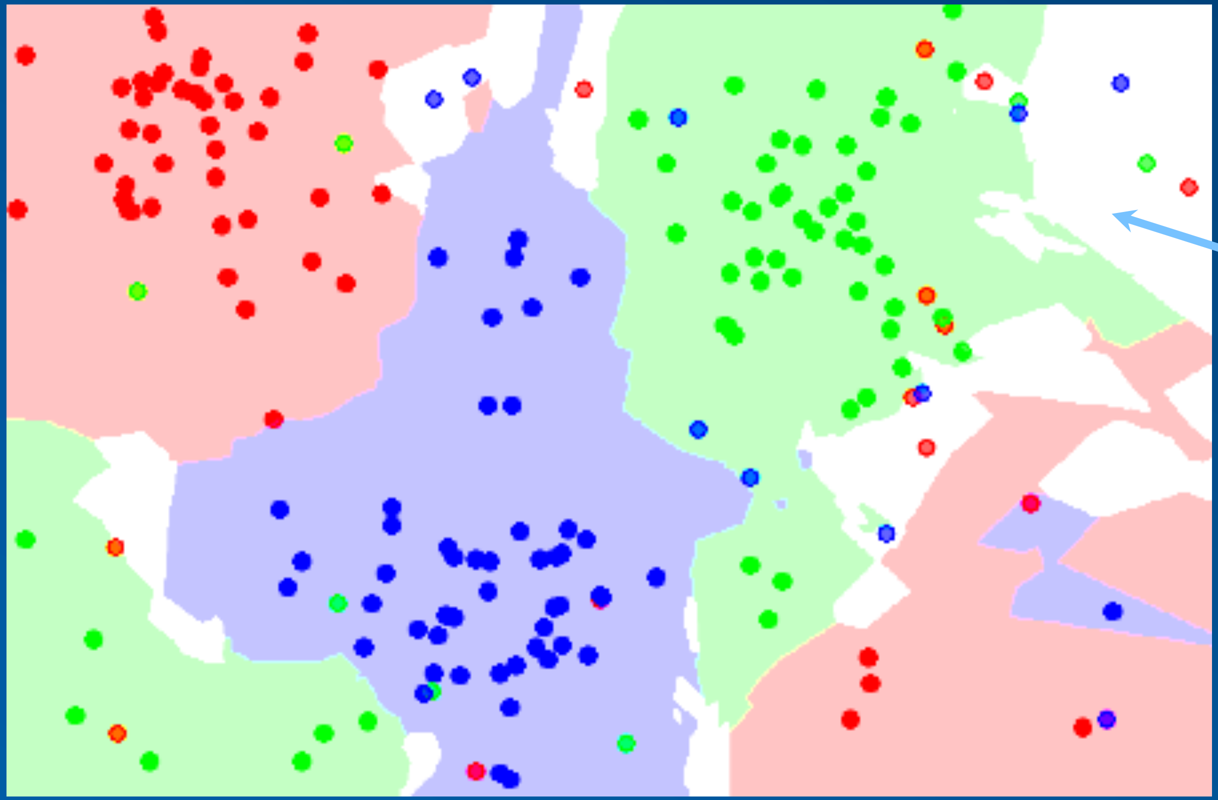
Data set



1 Nearest Neighbor



5 Nearest Neighbors



What is this
white space?

Choosing K: Cross Validation

- Don’t confuse k-folds with k-nearest neighbors! Different k!!
- For this lecture, use: f-folds
- To pick the ‘best’ k, divide your data into f folds (f > 3, usually 5+)
- Example: 5 Folds
  - Hold out Fold 1, build model on folds 2-5
  - Hold out Fold 2, build model on folds 1, 3-5
  - Hold out Fold 3, build model on folds 1-2, 4-5
  - Hold out Fold 4, build model on folds 1-3, 5
  - Hold out Fold 5, build model on folds 1-4
  - Average the accuracy of all 5 models

But we are going to use k-means!

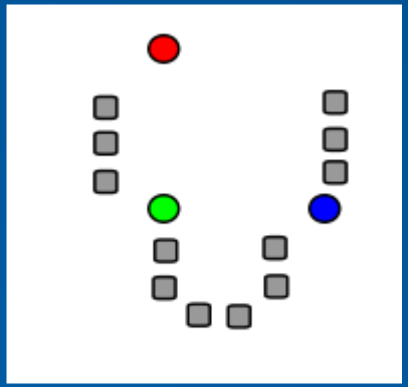
- Very similar to Nearest Neighbor, instead work on the average position
- Third k!! (no jokes here….)
    - K-nearest neighbor
    - K-fold cross validation
    - K-means

K means

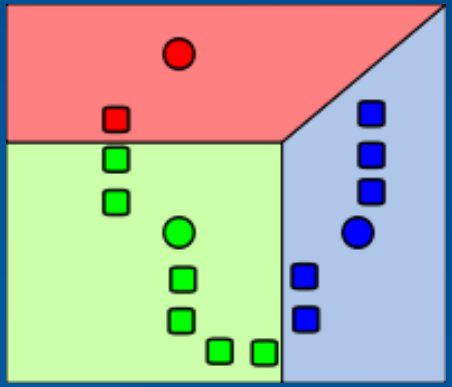
3-step process

1. Assign initial means/centers (various methods can be random)
2. Assign each point to the class of the nearest mean
3. Update the central point by taking the mean of each cluster generated from step 2
   - Repeat steps 2 & 3 until the mean stop moving (convergence)

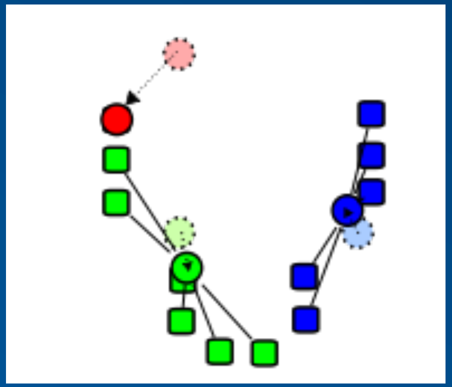
K means visual



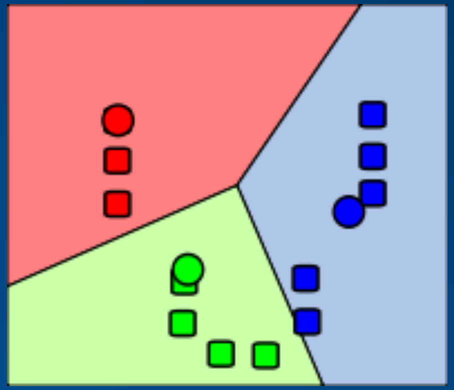
Step 1



Step 2



Step 3



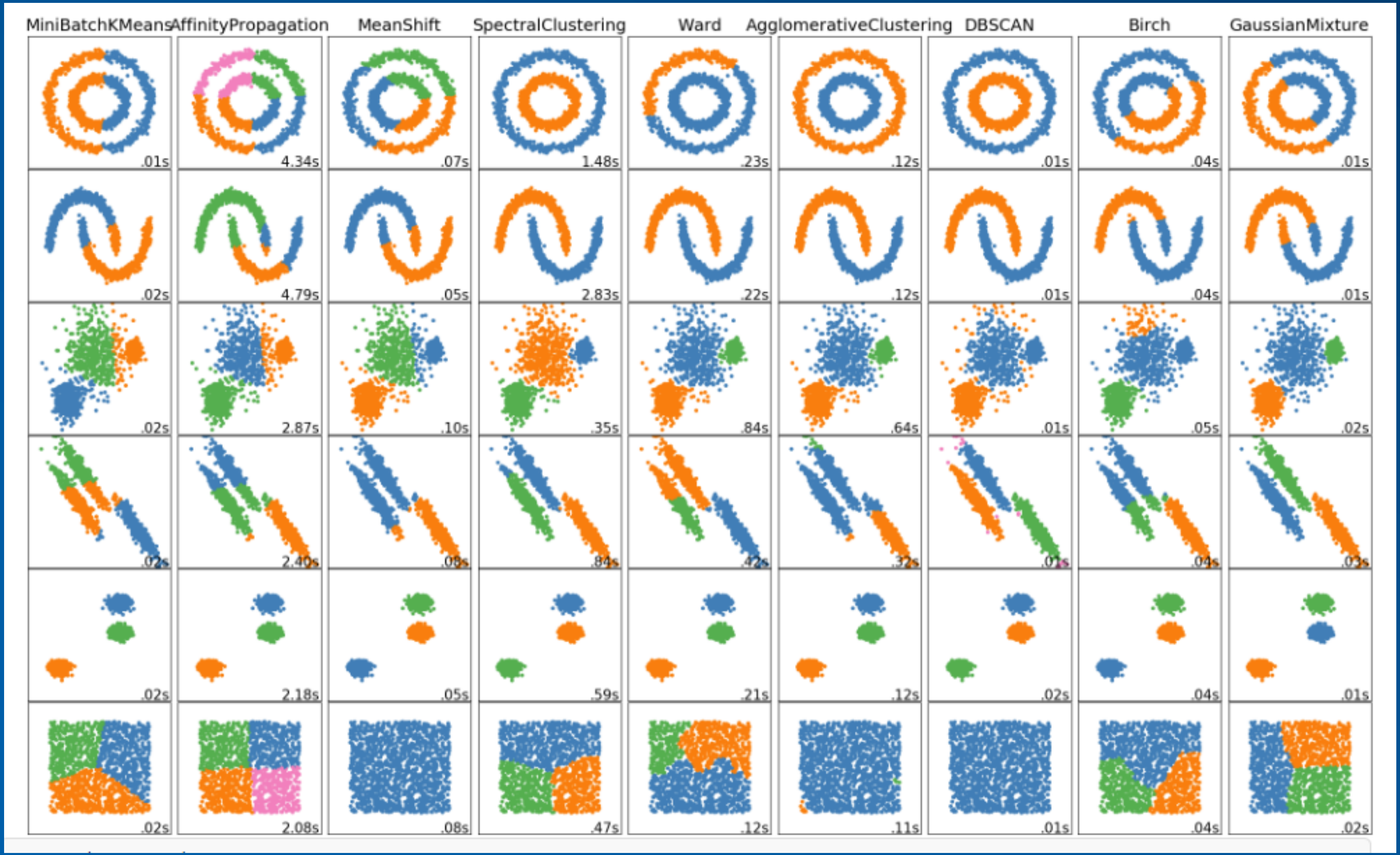
Step 2 (repeated)

- Center does not have to be a point
- Possible to get stuck in a 'loop'
- Stop when center moves less than a pre-defined distance
    - What does 'zero' mean

Same issues as Nearest Neighbors

- How many centers or means?
- Scaling is $O(n^2)$
    - OK, smarty, it is really $O(n * k * i * d)$
        - Number of points
        - Number of centers (k)
        - Number of features (d)
        - Number of iterations

Clustering difficulties



World Changers
Shaped Here

SMU®