Slides adapted from Prof. Carpuat

CMSC 422 Introduction to Machine Learning

Lecture 5 K-Means Clustering (Unsupervised Learning)

Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

Question

- When applying a learning algorithm, some things are properties of the problem you are trying to solve, and some things are up to you to choose as the ML programmer.
- Which of the following are properties of the problem?
  - The data generating distribution
  - The train/dev/test split
  - The learning model
  - The loss function

Recap

- Nearest Neighbors (NN) algorithms for classification
  - K-NN, Epsilon ball NN
  - Take a geometric view of learning
- Fundamental Machine Learning Concepts
  - Decision boundary
    - Visualizes predictions over entire feature space
    - Characterizes complexity of learned model
    - Indicates overfitting/underfitting

Exercise: When are DT vs kNN appropriate?

Properties of classification problem	Can Decision Trees handle them?	Can K-NN handle them?
Binary features		
Numeric features		
Categorical features		
Robust to noisy training examples		
Fast classification is crucial		
Many irrelevant features		
Relevant features have very different scale		

Exercise: When are DT vs kNN appropriate?

Properties of classification problem	Can Decision Trees handle them?	Can K-NN handle them?
Binary features	yes	yes
Numeric features	yes	yes
Categorical features	yes	yes
Robust to noisy training examples	no (for default algorithm)	yes (when k > 1)
Fast classification is crucial	yes	no
Many irrelevant features	yes	no
Relevant features have very different scale	yes	no

Today's Topics

- A new algorithm
  - K-Means Clustering
- Fundamental Machine Learning Concepts
  - Unsupervised vs. supervised learning
  - Decision boundary

Clustering

- Goal: automatically partition examples into groups of similar examples

- Why? It is useful for
  - Automatically organizing data
  - Understanding hidden structure in data
  - Preprocessing for further analysis

What can we cluster in practice?

- news articles or web pages by topic
- protein sequences by function, or genes according to expression profile
- users of social networks by interest
- customers according to purchase history
- …

Clustering

- Input
  - a set S of n points in feature space
  - a distance measure specifying distance $d(x_i, x_j)$ between pairs (x_i, x_j)
- Output
  - A partition $\{S_1, S_2, \ldots, S_k\}$ of S

Supervised Machine Learning as Function Approximation

Problem setting

- Set of possible instances X
- Unknown target function $f: X \rightarrow Y$
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \ldots (x^{(N)}, y^{(N)})\}$ of unknown target function f

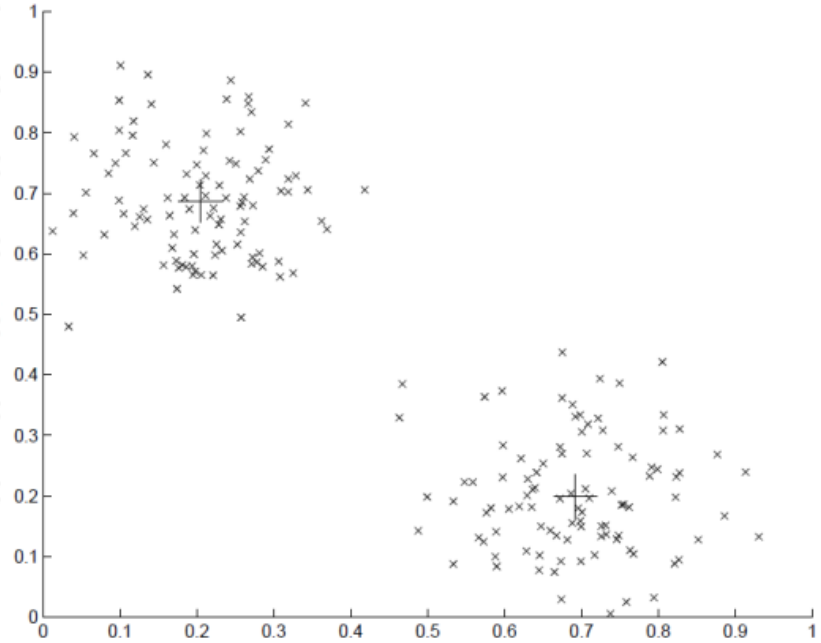
Output

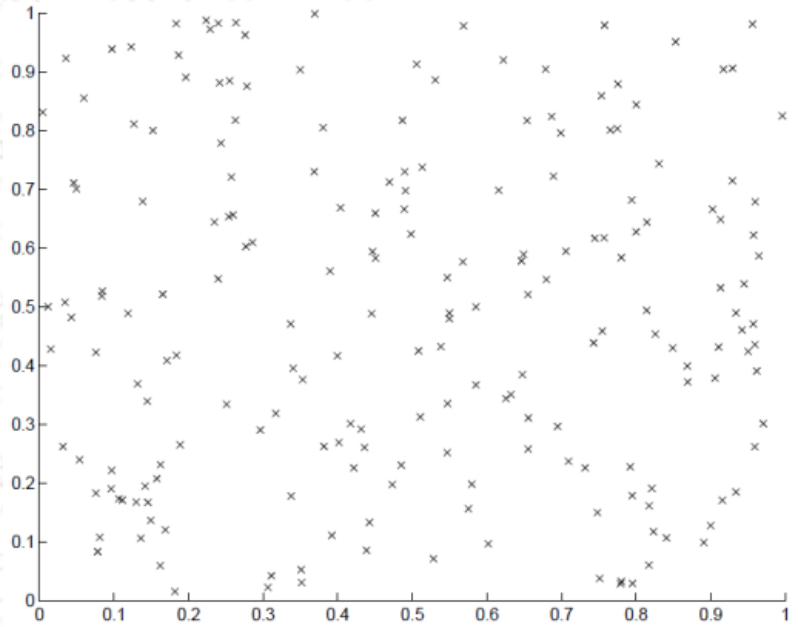
- Hypothesis $h \in H$ that best approximates target function f

Supervised vs. unsupervised learning

- Clustering is an example of unsupervised learning
- We are not given examples of classes y
- Instead we have to discover classes in data

2 datasets with very different underlying structure!





The K-Means Algorithm

Training Data

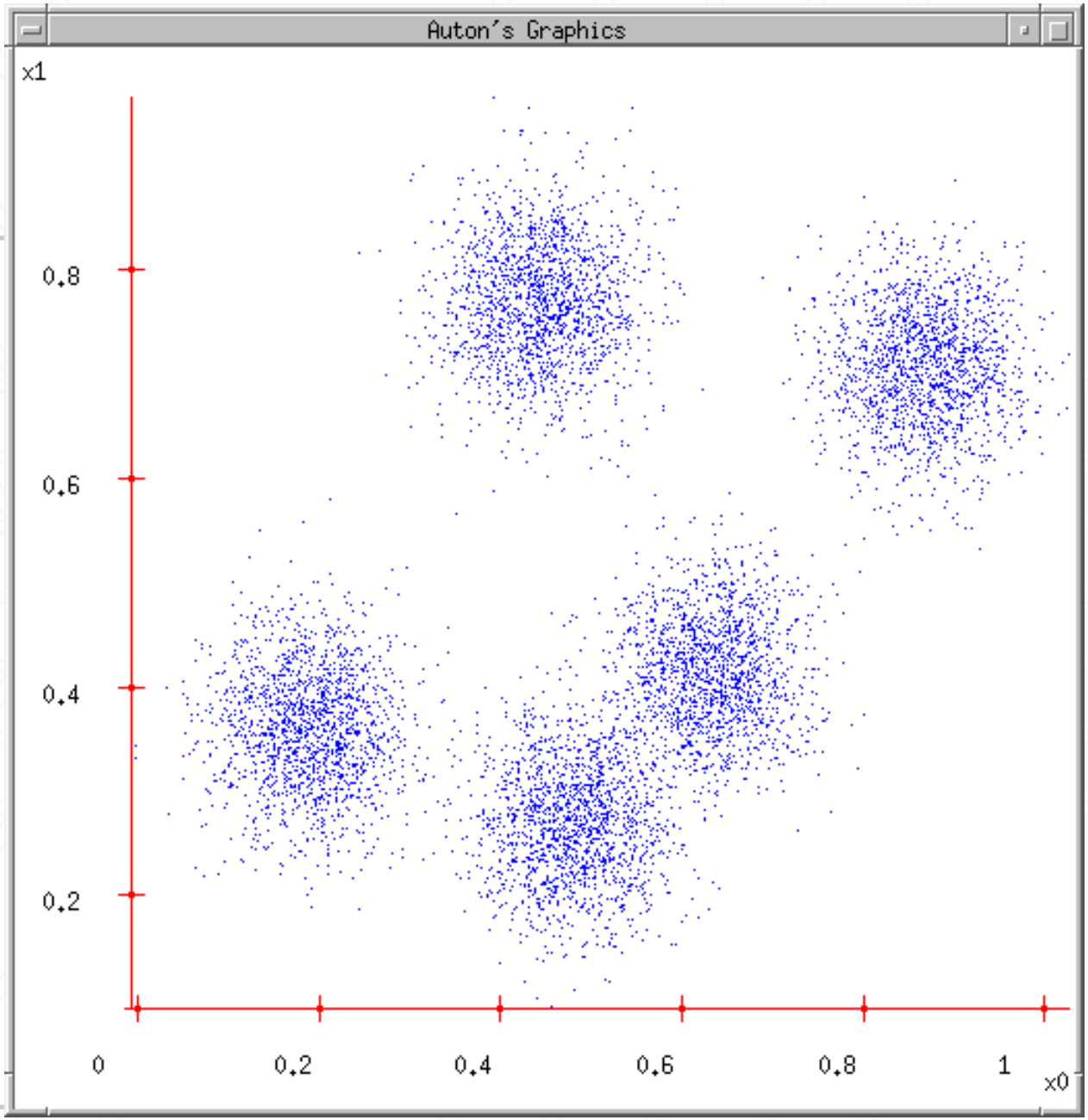
K: number of clusters to discover

Algorithm 4 K-MEANS($\mathbf{D}$, K)

1: **for** $k = 1$ **to** K **do**
2: $\boldsymbol{\mu}_k \leftarrow$ some random location // randomly initialize mean for kth cluster
3: **end for**
4: **repeat**
5: **for** $n = 1$ **to** N **do**
6: $z_n \leftarrow \text{argmin}_k \, ||\boldsymbol{\mu}_k - \boldsymbol{x}_n||$ // assign example n to closest center
7: **end for**
8: **for** $k = 1$ **to** K **do**
9: $\mathbf{X}_k \leftarrow \{ \boldsymbol{x}_n : z_n = k \}$ // points assigned to cluster k
10: $\boldsymbol{\mu}_k \leftarrow \text{MEAN}(\mathbf{X}_k)$ // re-estimate mean of cluster k
11: **end for**
12: **until** $\boldsymbol{\mu}$s stop changing
13: **return** z // return cluster assignments

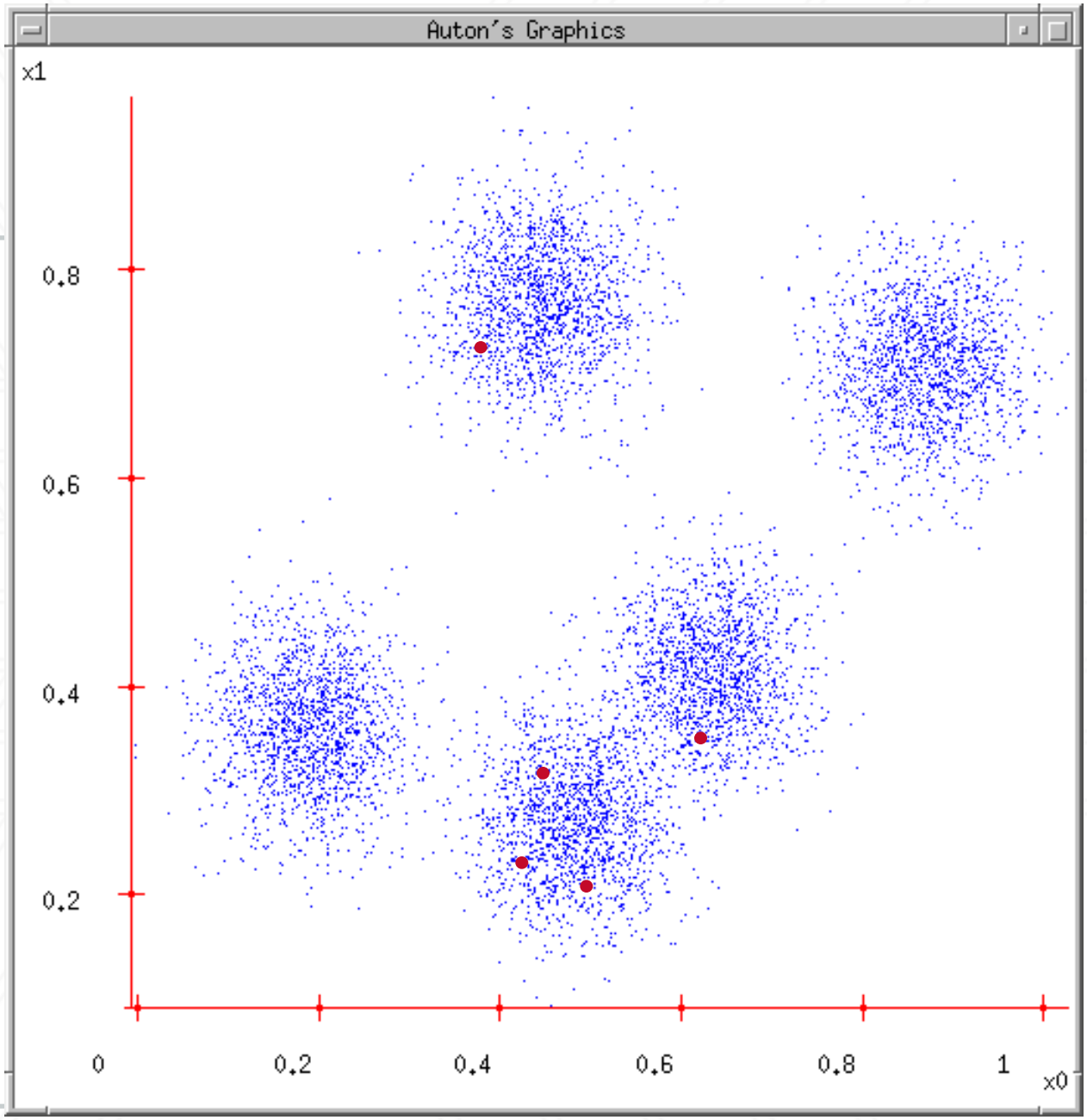
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*



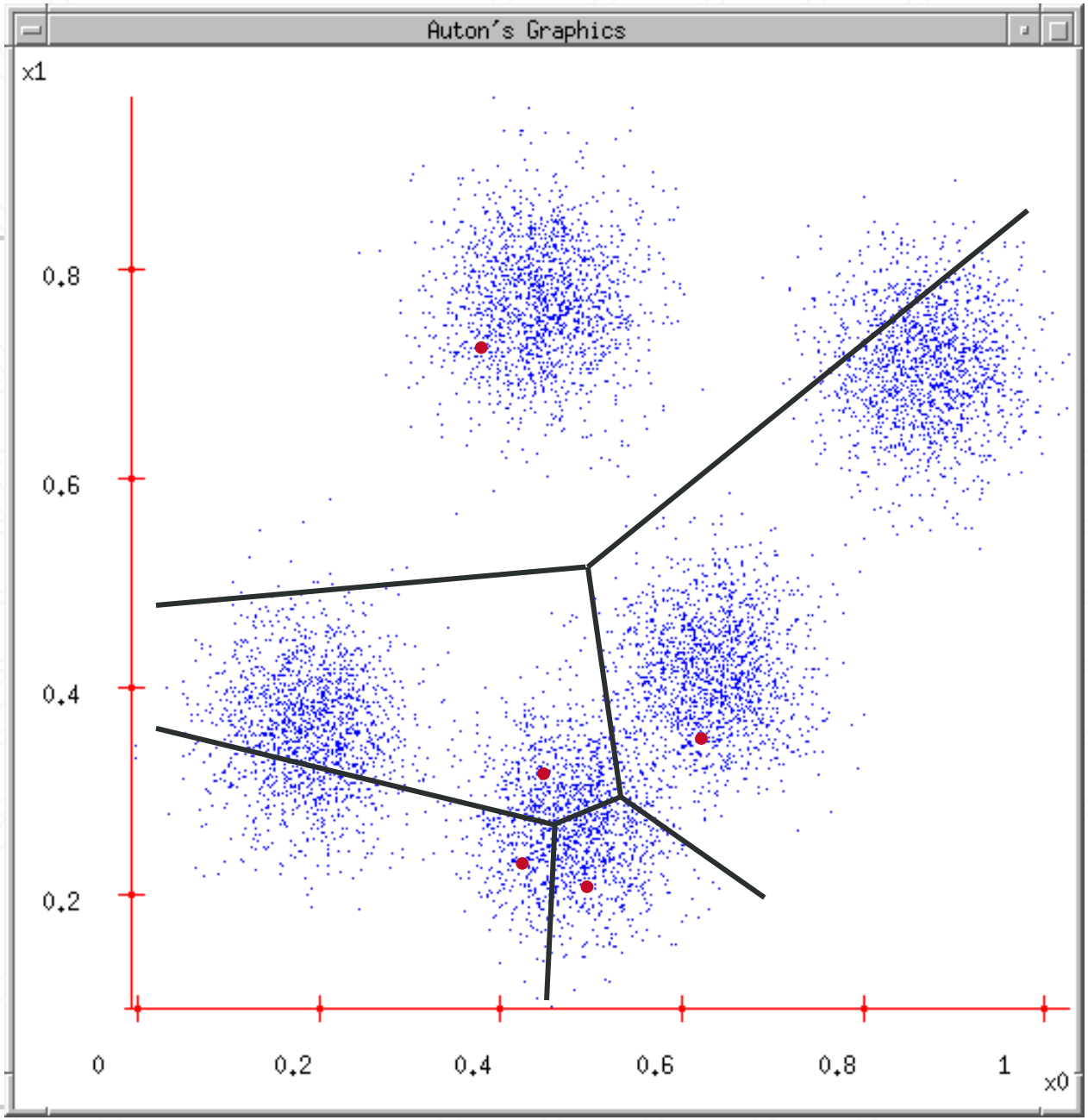
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



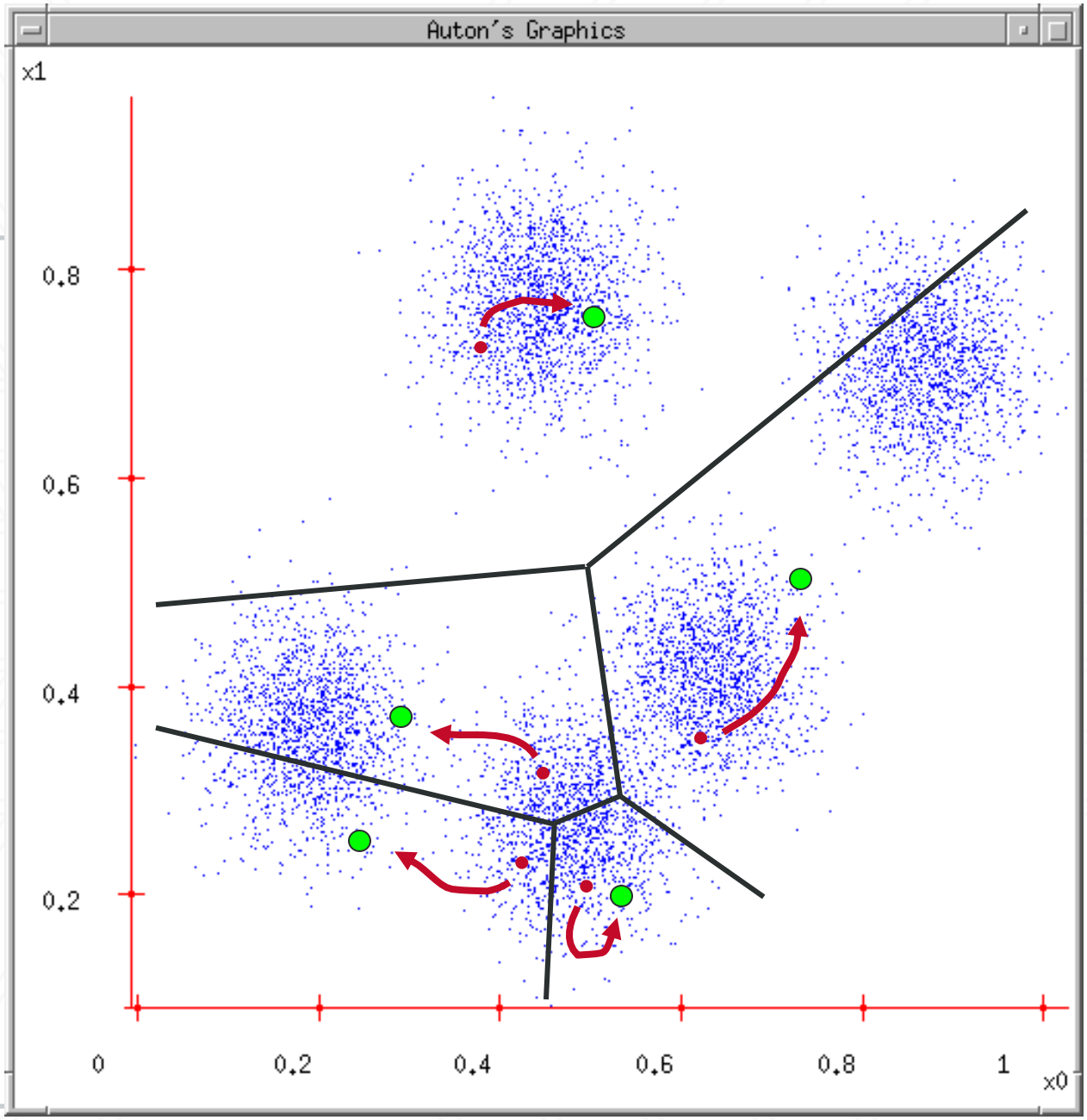
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



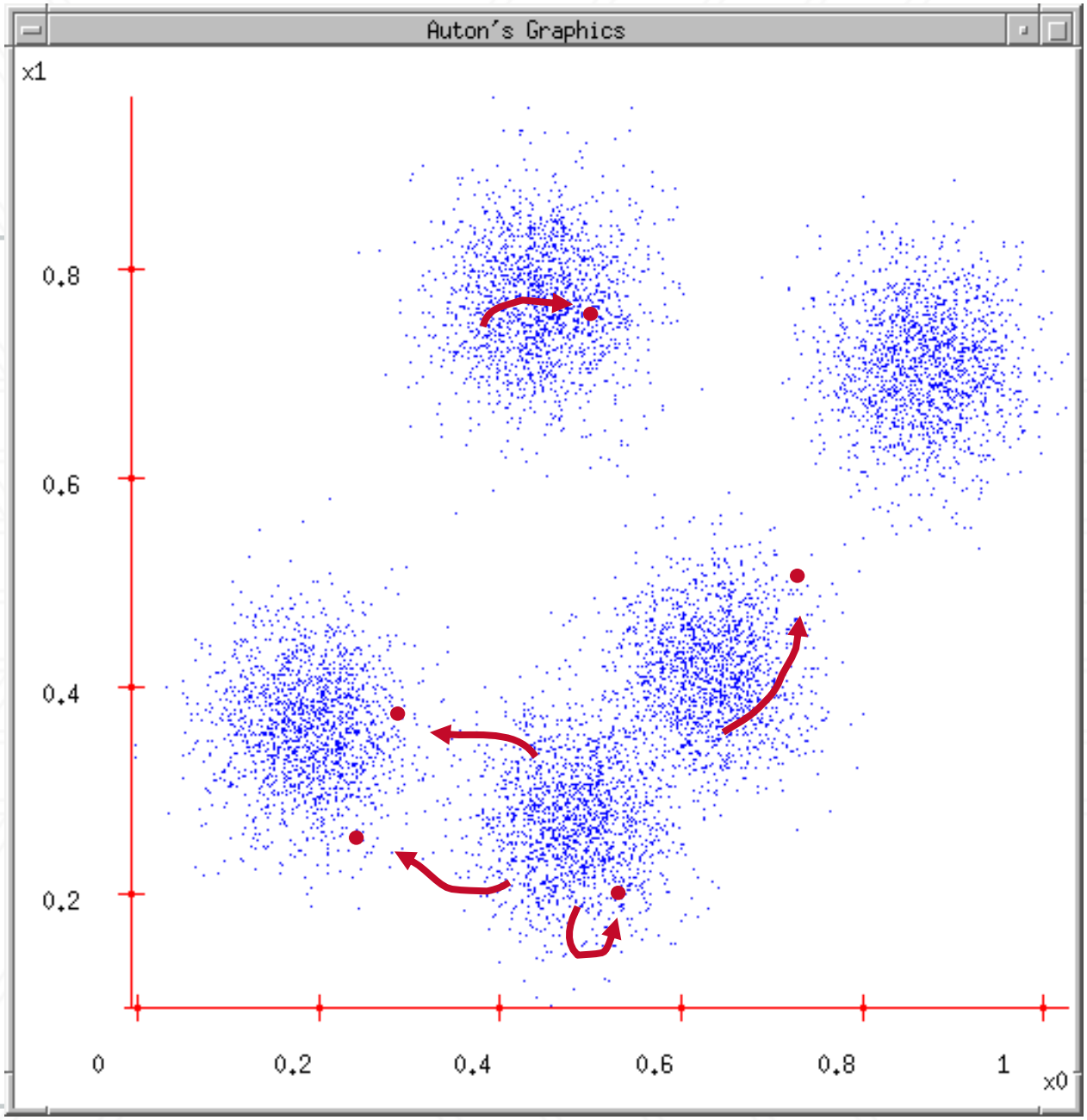
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



K-Means properties

- Time complexity: O(KNL) where
  - K is the number of clusters
  - N is number of examples
  - L is the number of iterations
- K is a hyperparameter
  - Needs to be set in advance (or learned on dev set)
- Different initializations yield different results!
  - Doesn't necessarily converge to best partition
- "Global" view of data: revisits all examples at every iteration

K-means Questions

Are we sure it will terminate?

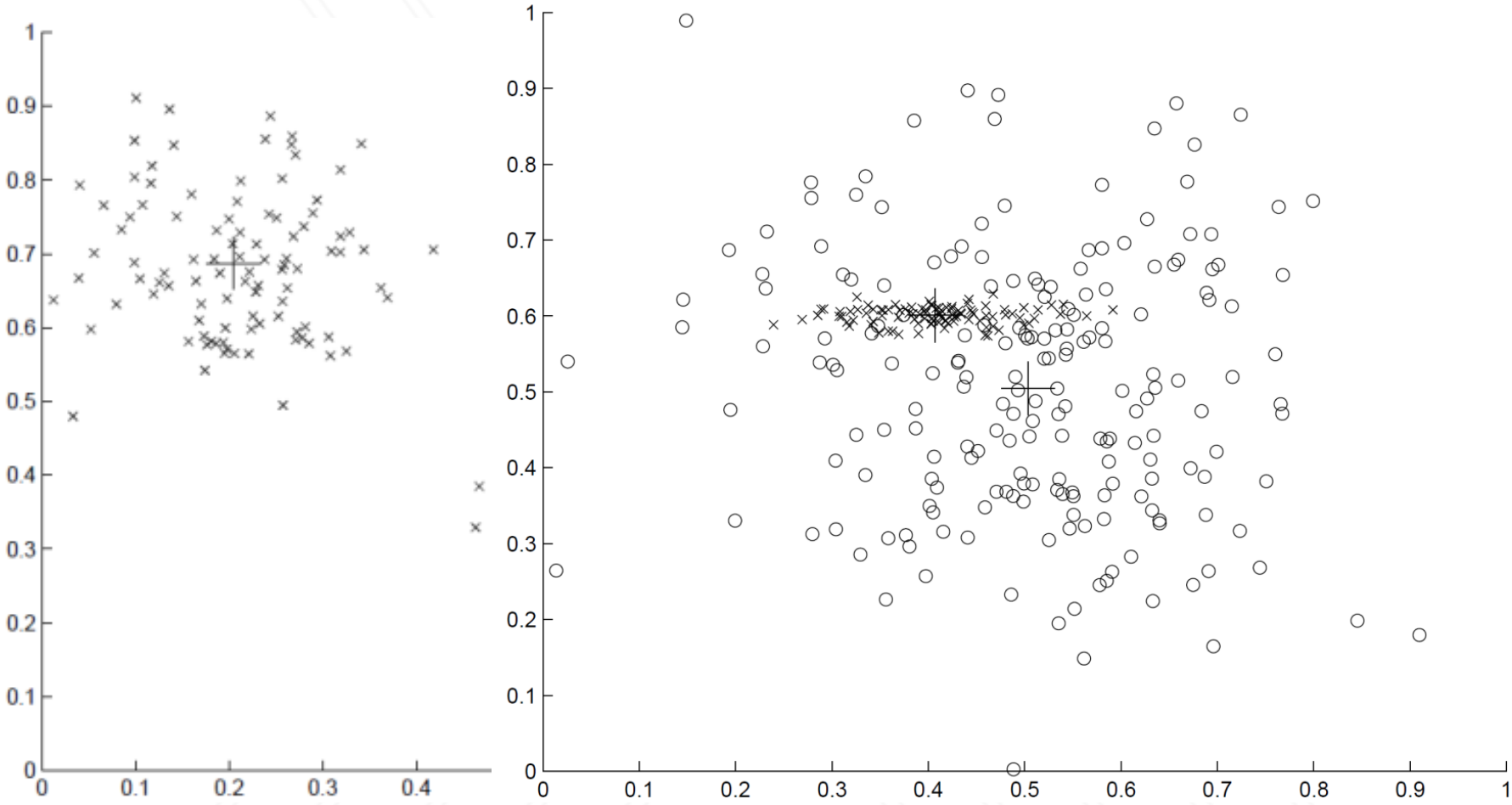
Are we sure it will find an optimal clustering?

How should we start it?

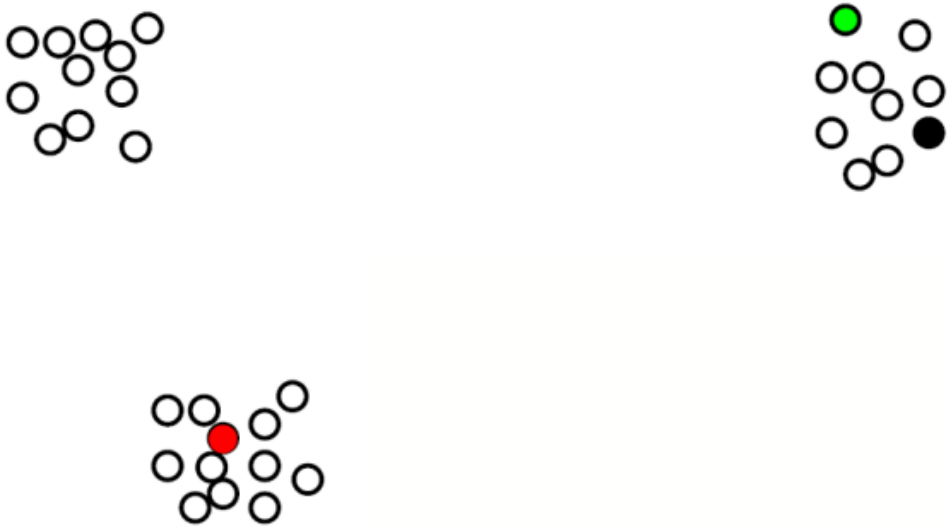
How could we automatically choose the number of centers?

….we'll deal with these questions over the next few slides

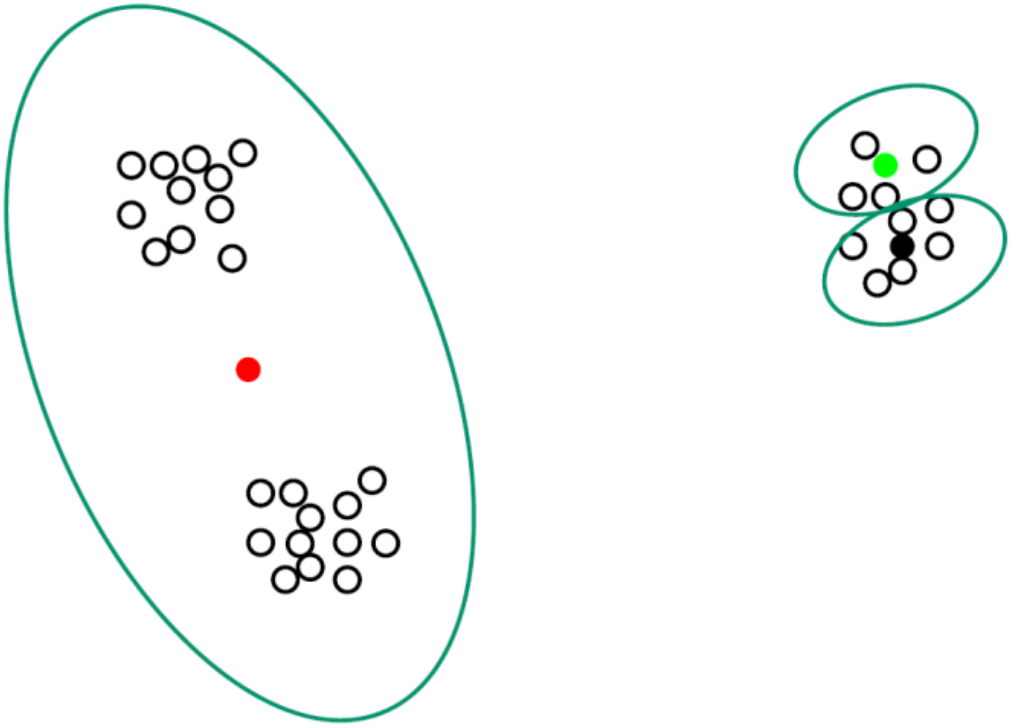
Can K-means always win?



Impact of initialization



Impact of initialization



Optimization View of K-means

Given a set of observations $(x_1, x_2, \ldots, x_n)$ where each observation is a d-dimensional real vector

k-means clustering aims to partition the n observations into $\mathrm{k}(\leq x)$ sets $S = \{S_1, S_2, \ldots, S_\mathrm{k}\}$ so as to minimize the within-cluster sum of squares.

Formally, the objective is to find:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg\min_{S} \sum_{i}^{k} |S_i| \operatorname{Var}(S_i)$$

where μ_i is the mean of points in S_i.

Trying to find good optima

Idea 1: Be careful about where you start

Idea 2: Do many runs of k-means, each from a different random start configuration

Many other ideas floating around.

Common uses of K-means

- Often used as an exploratory data analysis tool
- In one-dimension, a good way to quantize real-valued variables into k non-uniform buckets
- Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization)
- Also used for choosing color palettes on old fashioned graphical display devices!

Questions for you…

- Are there clusters that cannot be discovered using k-means?

- Do you know any other clustering algorithms?

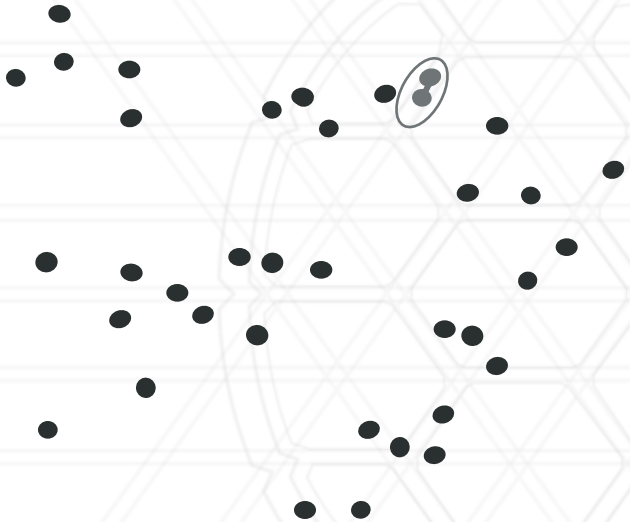
Single Linkage Hierarchical Clustering

1. Say "Every point is its own cluster"

Single Linkage Hierarchical Clustering

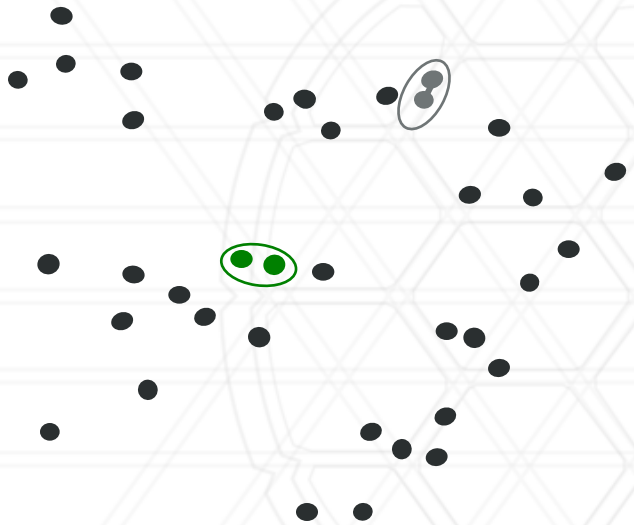
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters

Single Linkage Hierarchical Clustering



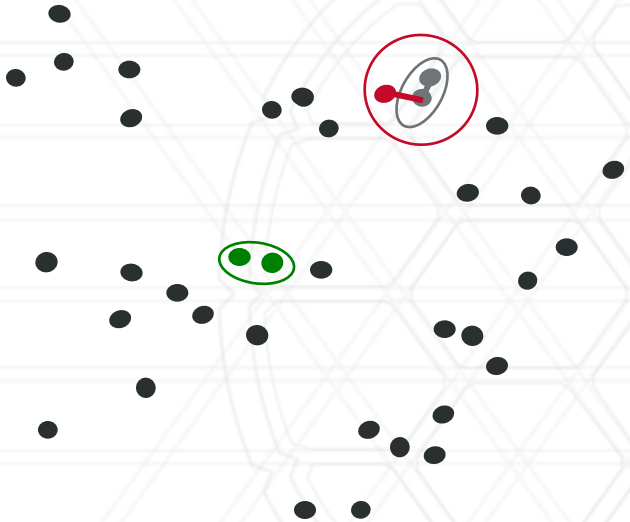
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster

Single Linkage Hierarchical Clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Single Linkage Hierarchical Clustering



1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Hierarchical Clustering

How do we define similarity between clusters?

- Minimum distance between points in clusters
- Maximum distance between points in clusters
- Average distance between points in clusters

1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat…until you've merged the whole dataset into one cluster

You're left with a nice dendrogram, or taxonomy, or hierarchy of datapoints (not shown here)

Aside: Curse of dimensionality

- Challenges of working with high dimensional spaces
  - Hard to visualize
  - Computational cost
  - Many of our intuitions about 2D or 3D spaces don't hold
    - High dimensional hyperspheres "look more like porcupines than balls"
    - Distances between two random points in high dimensions are approximately the same

  (CIML Section 3.5 + HW #3)

What you should know

- New Algorithms
  - K-NN classification
  - K-means clustering
- Fundamental ML concepts
  - How to draw decision boundaries
  - What decision boundaries tells us about the underlying classifiers
  - The difference between supervised and unsupervised learning

UNIVERSITY OF
MARYLAND

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu