Introduction to Machine Learning

Xiaojin Zhu

jerryzhu@cs.wisc.edu

Read Chapter 1 of this book:

Xiaojin Zhu and Andrew B. Goldberg.

Introduction to Semi-Supervised Learning.

http://www.morganclaypool.com/doi/abs/10.2200/S00196ED1V01Y200906AIM006

Morgan & Claypool Publishers, 2009.

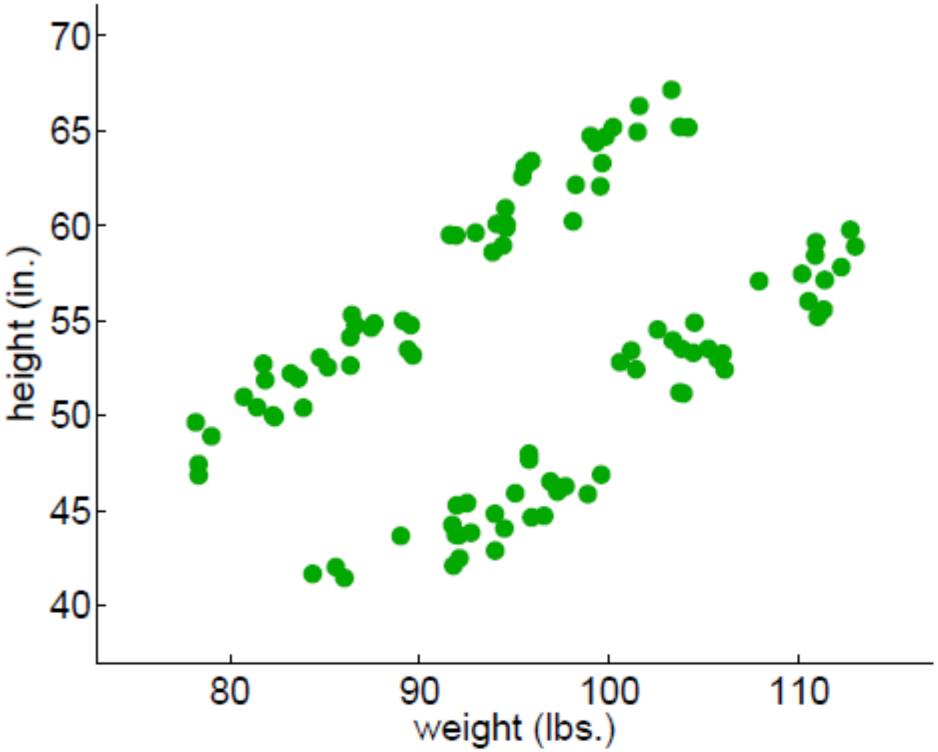
(download from UW computers)

Outline

- Representing “things”
  - Feature vector
  - Training sample
- Unsupervised learning
  - Clustering
- Supervised learning
  - Classification
  - Regression

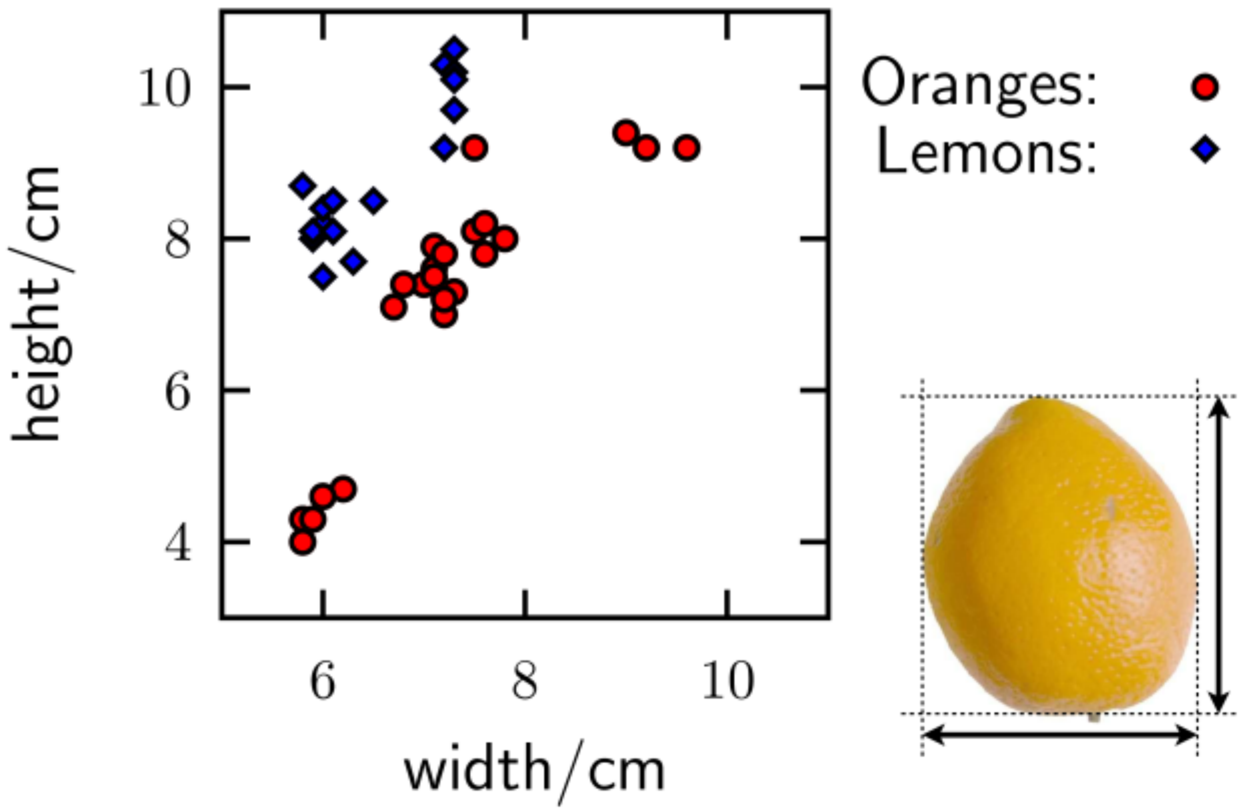
Little green men

- The weight and height of 100 little green men



- What can you learn from this data?

A less alien example



- From Iain Murray http://homepages.inf.ed.ac.uk/imurray2/

Representing "things" in machine learning

- An instance *x* represents a specific object ("thing")
- *x* often represented by a *D*-dimensional feature vector $x = (x_1, . . . , x_D) \in R^D$
- Each dimension is called a feature. Continuous or discrete.
- *x* is a dot in the D-dimensional feature space
- Abstraction of object. Ignores any other aspects (two men having the same weight, height will be identical)

Feature representation example

- Text document
  - Vocabulary of size D (~100,000): “aardvark … zulu”
- “bag of word”: counts of each vocabulary entry
  - To marry my true love → (3531:1 13788:1 19676:1)
  - I wish that I find my soulmate this year → (3819:1 13448:1 19450:1 20514:1)
- Often remove stopwords: the, of, at, in, …
- Special “out-of-vocabulary” (OOV) entry catches all unknown words

More feature representations

- Image
  - Color histogram
- Software
  - Execution profile: the number of times each line is executed
- Bank account
  - Credit rating, balance, #deposits in last day, week, month, year, #withdrawals …
- You and me
  - Medical test1, test2, test3, …

Training sample

- A *training sample is a collection of instances* $\mathbf{x}_1, \ldots, \mathbf{x}_n$, *which is the input to the learning process.*
- $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})$
- Assume these instances are sampled independently from an unknown (population) distribution $P(x)$
- We denote this by $\mathbf{x}_i \overset{\text{i.i.d.}}{\sim} P(x)$, where i.i.d. stands for independent and identically distributed.

Training sample

- A training sample is the “experience” given to a learning algorithm
- What the algorithm can learn from it varies
- We introduce two basic learning paradigms:
  - *unsupervised learning*
  - *supervised learning*

No teacher.

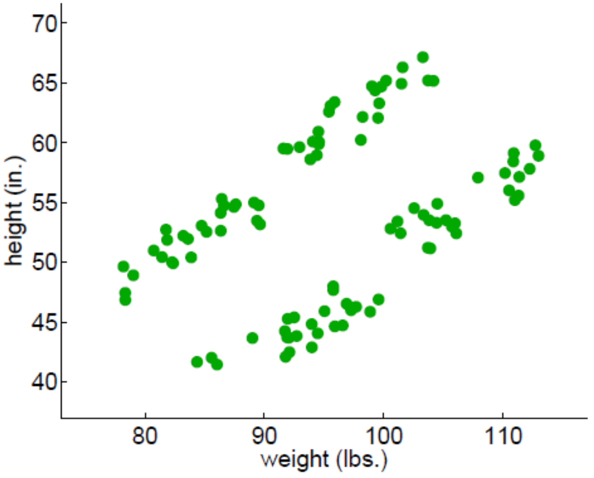
UNSUPERVISED LEARNING

Unsupervised learning

- Training sample $\mathbf{x}_1, \ldots, \boldsymbol{x}_n$, that's it
- No teacher providing supervision as to how individual instances should be handled
- Common tasks:
  - clustering, separate the *n* instances into groups
  - novelty detection, find instances that are very different from the rest
  - dimensionality reduction, represent each instance with a lower dimensional feature vector while maintaining key characteristics of the training samples

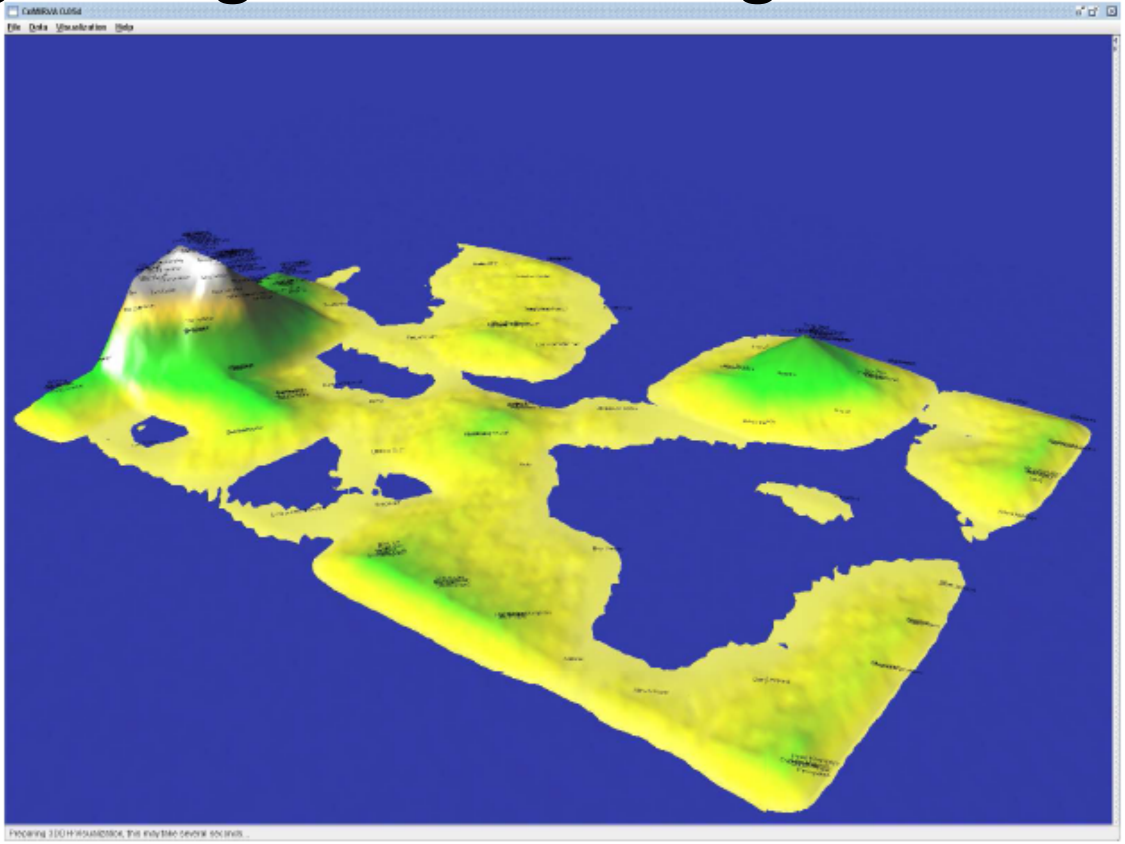
Clustering

- Group training sample into *k* clusters
- How many clusters do you see?
- Many clustering algorithms
  - HAC
  - k-means
  - …



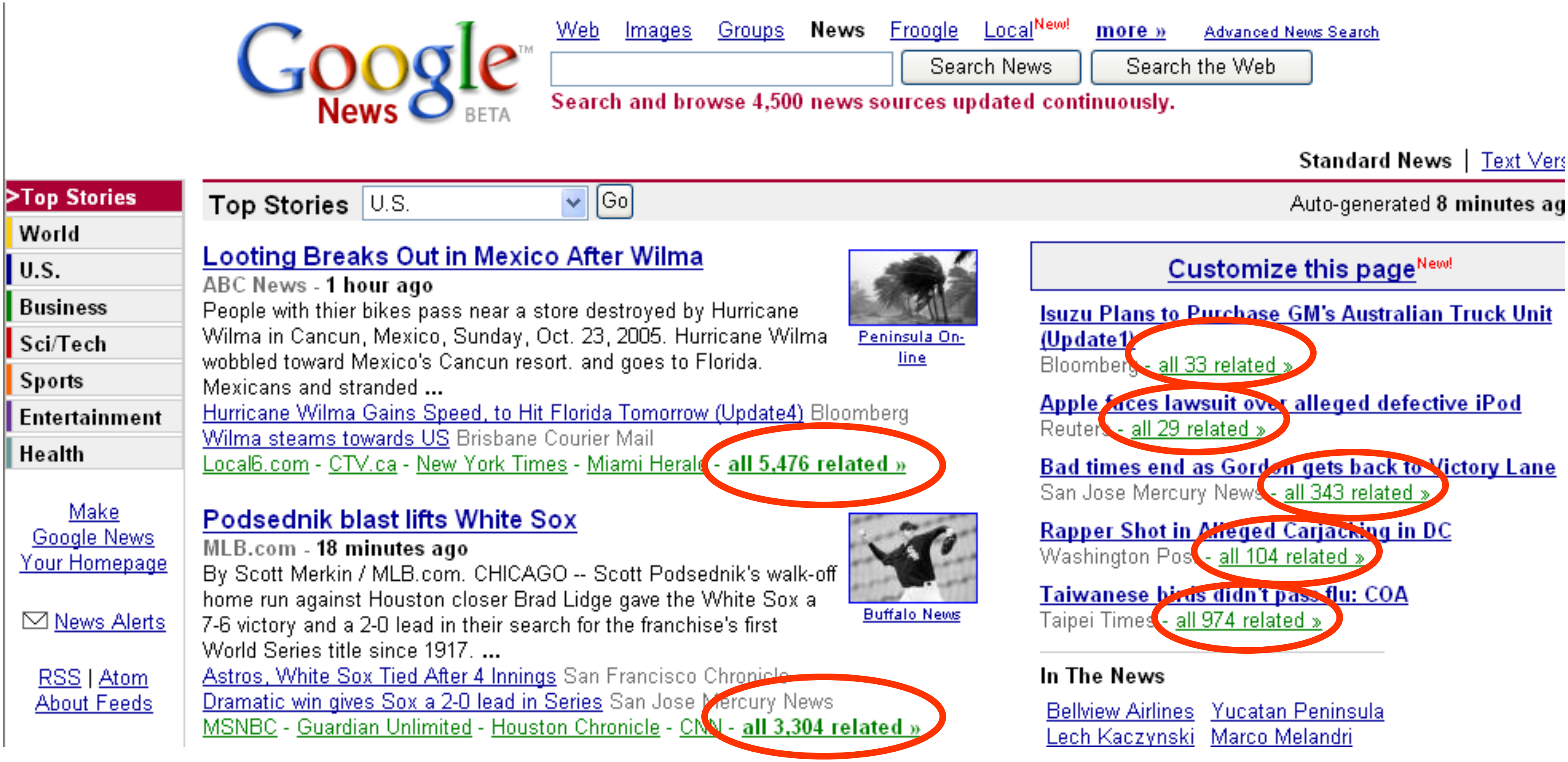
Example 1: music island

- Organizing and visualizing music collection



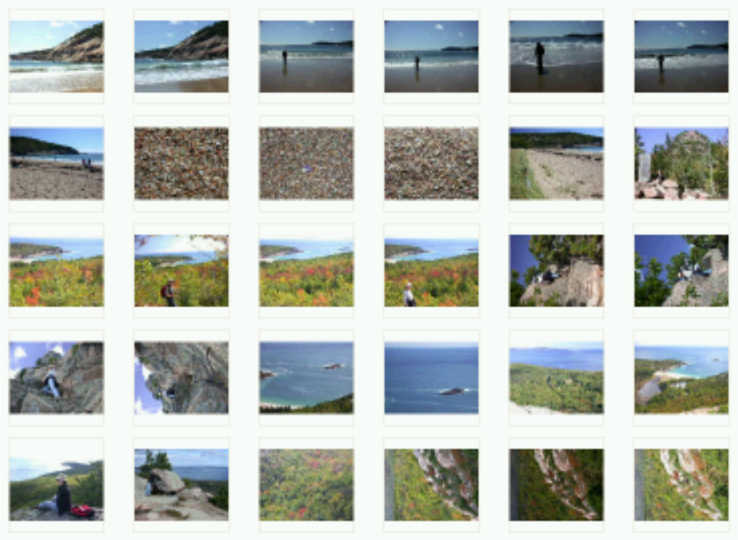
CoMIRVA http://www.cp.jku.at/comirva/

Example 2: Google News



Example 3: your digital photo collection

- You probably have >1000 digital photos, 'neatly' stored in various folders…
- After this class you'll be about to organize them better
  - Simplest idea: cluster them using image creation time (EXIF tag)
  - More complicated: extract image features



Two most frequently used methods

- Many clustering algorithms. We'll look at the two most frequently used ones:
  - Hierarchical clustering

    Where we build a binary tree over the dataset
  - K-means clustering

    Where we specify the desired number of clusters, and use an iterative algorithm to find them

Hierarchical clustering

- Very popular clustering algorithm
- Input:
  - A dataset $x_1, \ldots, x_n$, each point is a numerical feature vector
  - Does NOT need the number of clusters

Hierarchical Agglomerative Clustering

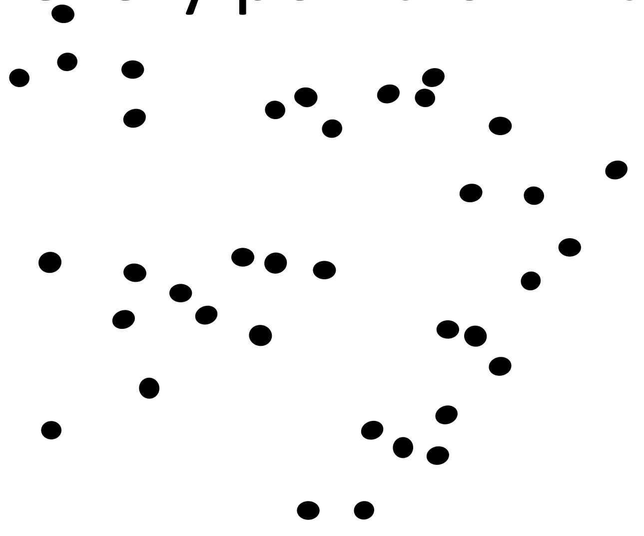
Input: *a training sample* $\{\mathbf{x}_i\}_{i=1}^n$; *a distance function* $d()$.
1. Initially, place each instance in its own cluster (called a singleton cluster).
2. ***while*** (*number of clusters* > 1) ***do***:
3. Find the closest cluster pair A, B, *i.e., they minimize* $d(A, B)$.
4. Merge A, B *to form a new cluster.*
Output: *a binary tree showing how clusters are gradually merged from singletons to a root cluster, which contains the whole training sample.*

- Euclidean (L2) distance

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{s=1}^{D} (x_{is} - x_{js})^2}.$$

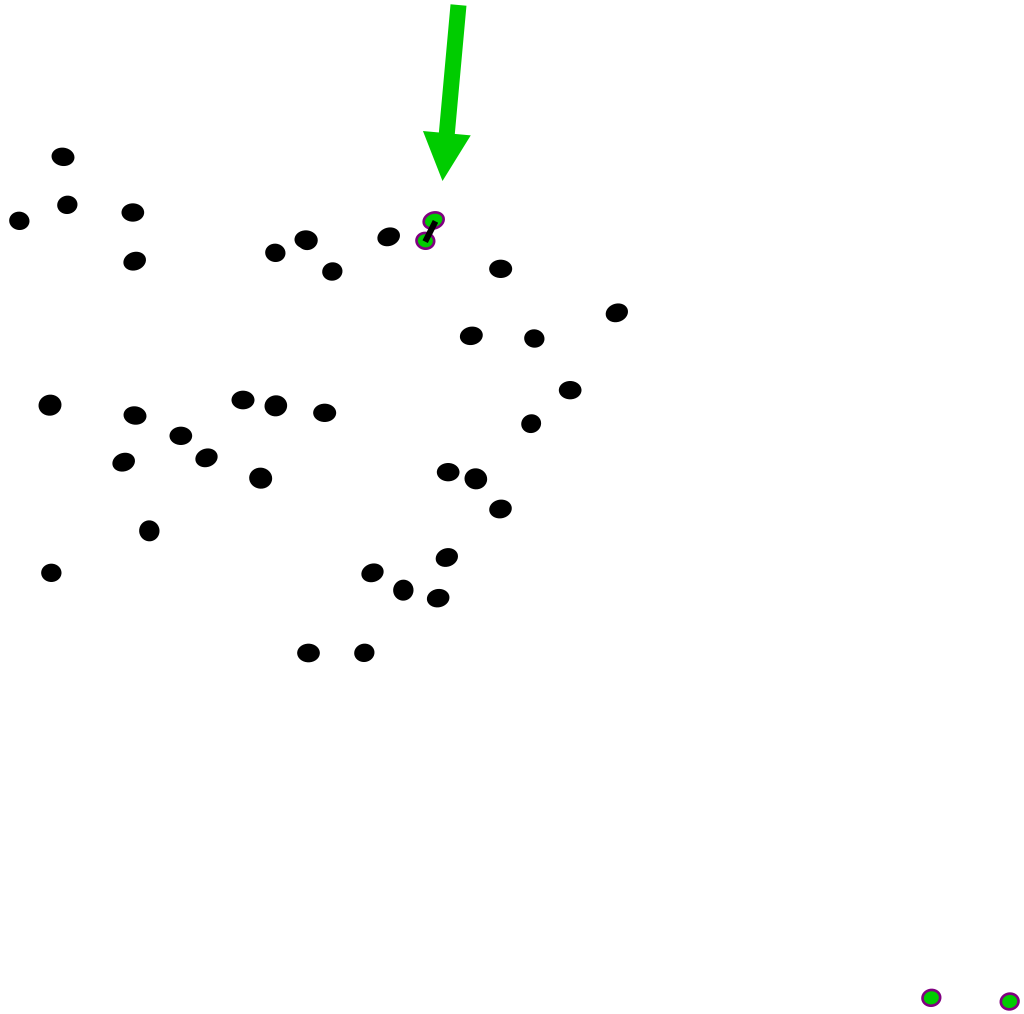
Hierarchical clustering

- Initially every point is in its own cluster



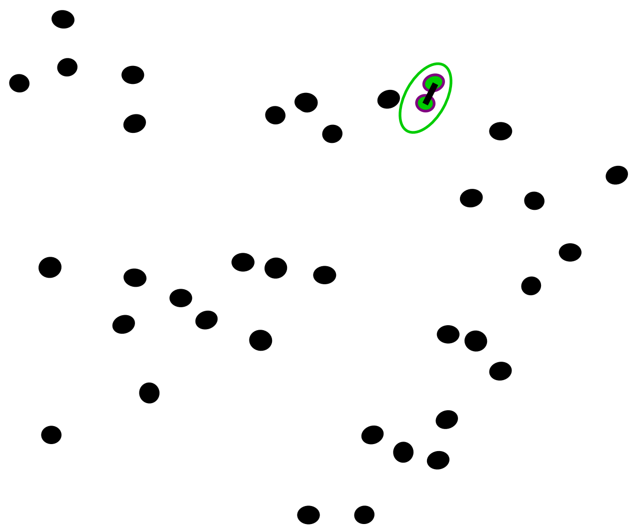
Hierarchical clustering

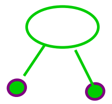
- Find the pair of clusters that are the closest



Hierarchical clustering

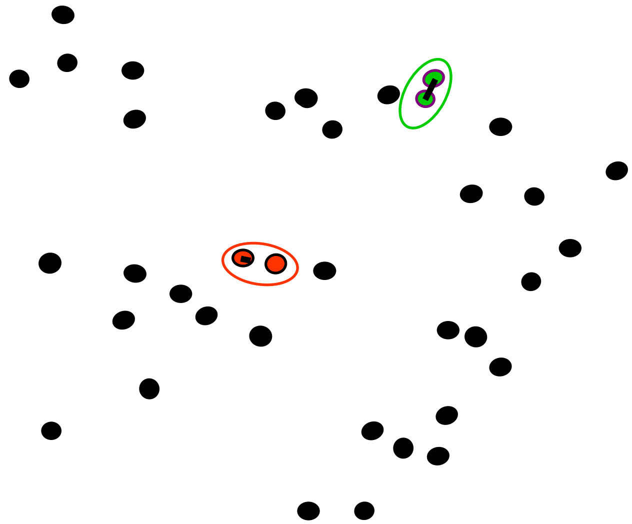
- Merge the two into a single cluster





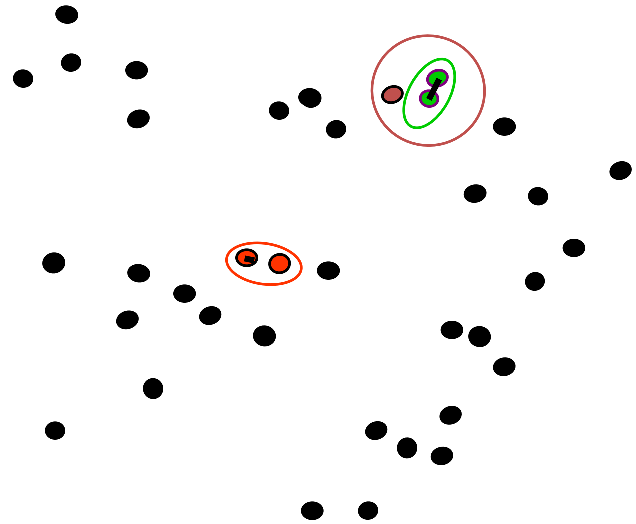
Hierarchical clustering

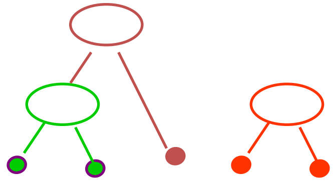
- Repeat…



Hierarchical clustering

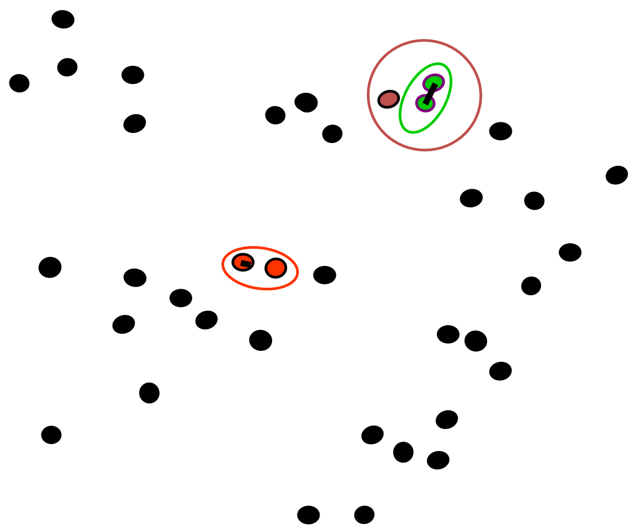
- Repeat…

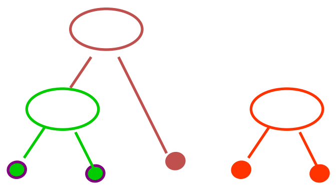




Hierarchical clustering

- Repeat…until the whole dataset is one giant cluster
- You get a binary tree (not shown here)





Hierarchical clustering

- How do you measure the closeness between two clusters?

Hierarchical clustering

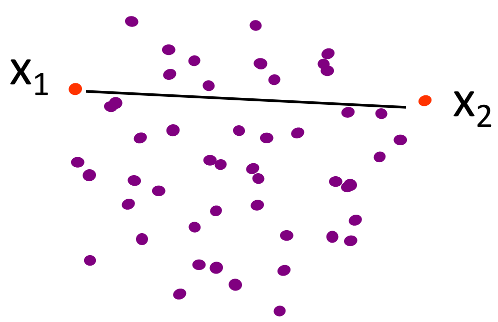
- How do you measure the closeness between two clusters? At least three ways:
  - Single-linkage: the shortest distance from any member of one cluster to any member of the other cluster. Formula?
  - Complete-linkage: the greatest distance from any member of one cluster to any member of the other cluster
  - Average-linkage: you guess it!

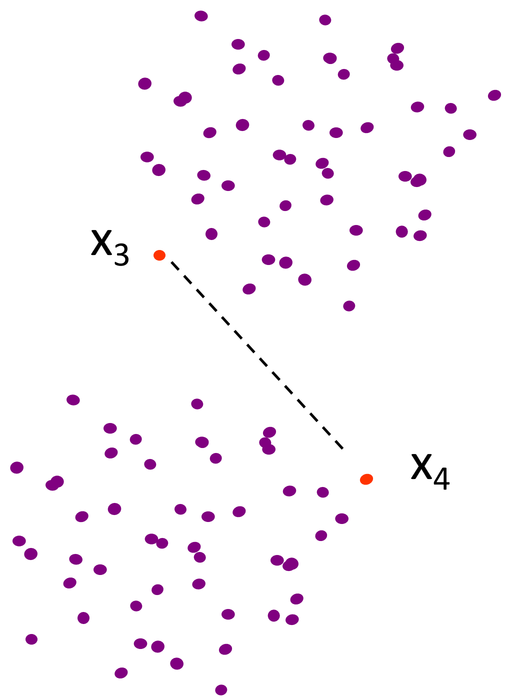
Hierarchical clustering

- The binary tree you get is often called a dendrogram, or taxonomy, or a hierarchy of data points
- The tree can be cut at various levels to produce different numbers of clusters: if you want k clusters, just cut the (k-1) longest links
- Sometimes the hierarchy itself is more interesting than the clusters
- However there is not much theoretical justification to it…

Advance topics

- **Constrained clustering**: What if an expert looks at the data, and tells you
  - "I think x1 and x2 **must** be in the same cluster" (must-links)
  - "I think x3 and x4 **cannot** be in the same cluster" (cannot-links)





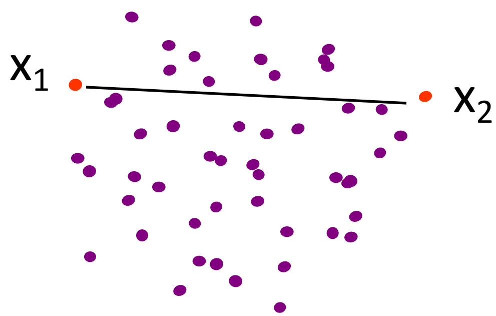
Advance topics

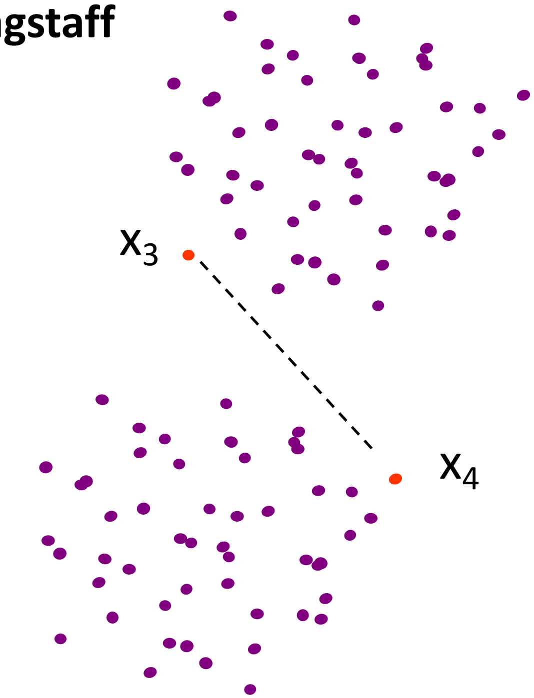
- This is clustering with supervised information (must-links and cannot-links). We can
  - Change the clustering algorithm to fit constraints
  - Or , learn a better distance measure
- See the book

Constrained Clustering: Advances in Algorithms, Theory, and Applications

Editors: Sugato Basu, Ian Davidson, and Kiri Wagstaff

http://www.wkiri.com/conscluster/



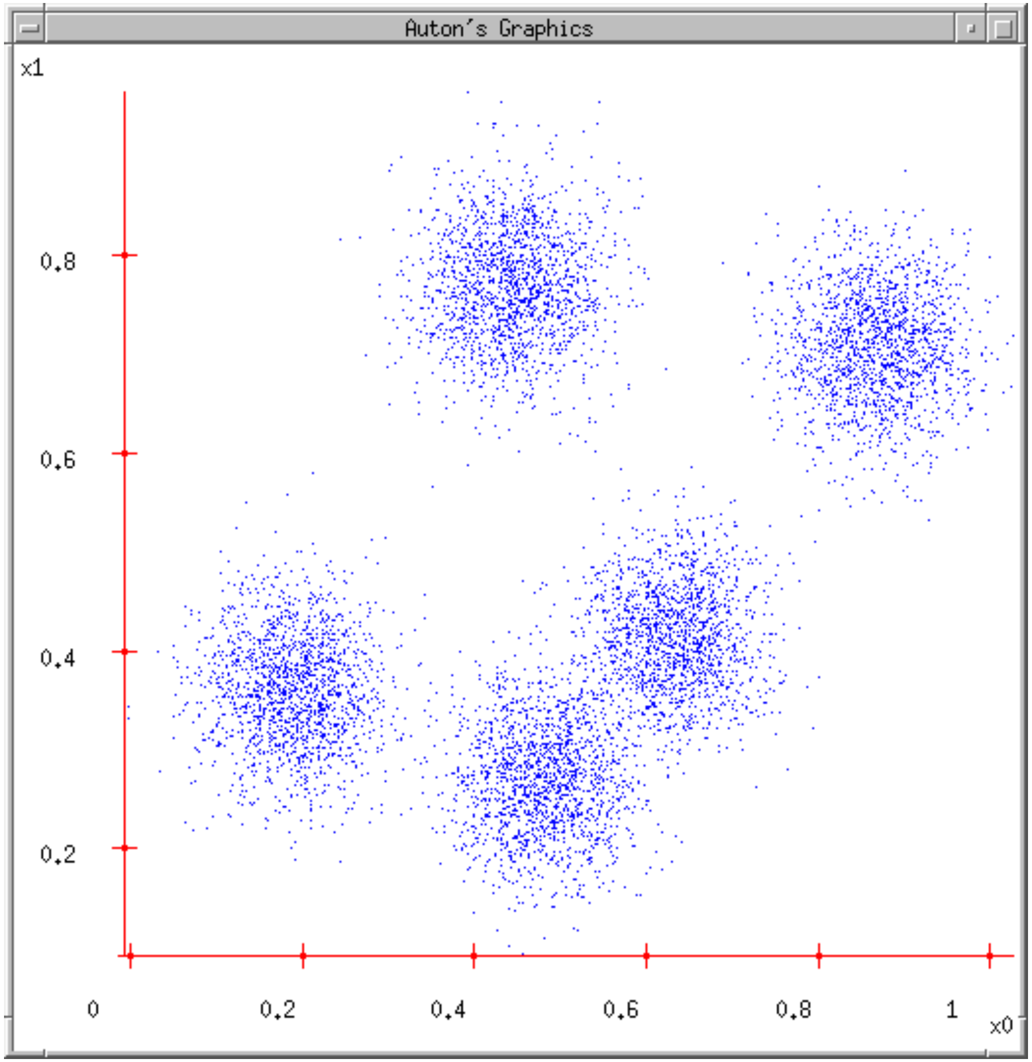


K-means clustering

- Very popular clustering method
- Don't confuse it with the k-NN classifier
- Input:
  - A dataset x_1, …, x_n, each point is a numerical feature vector
  - Assume the number of clusters, k, is given

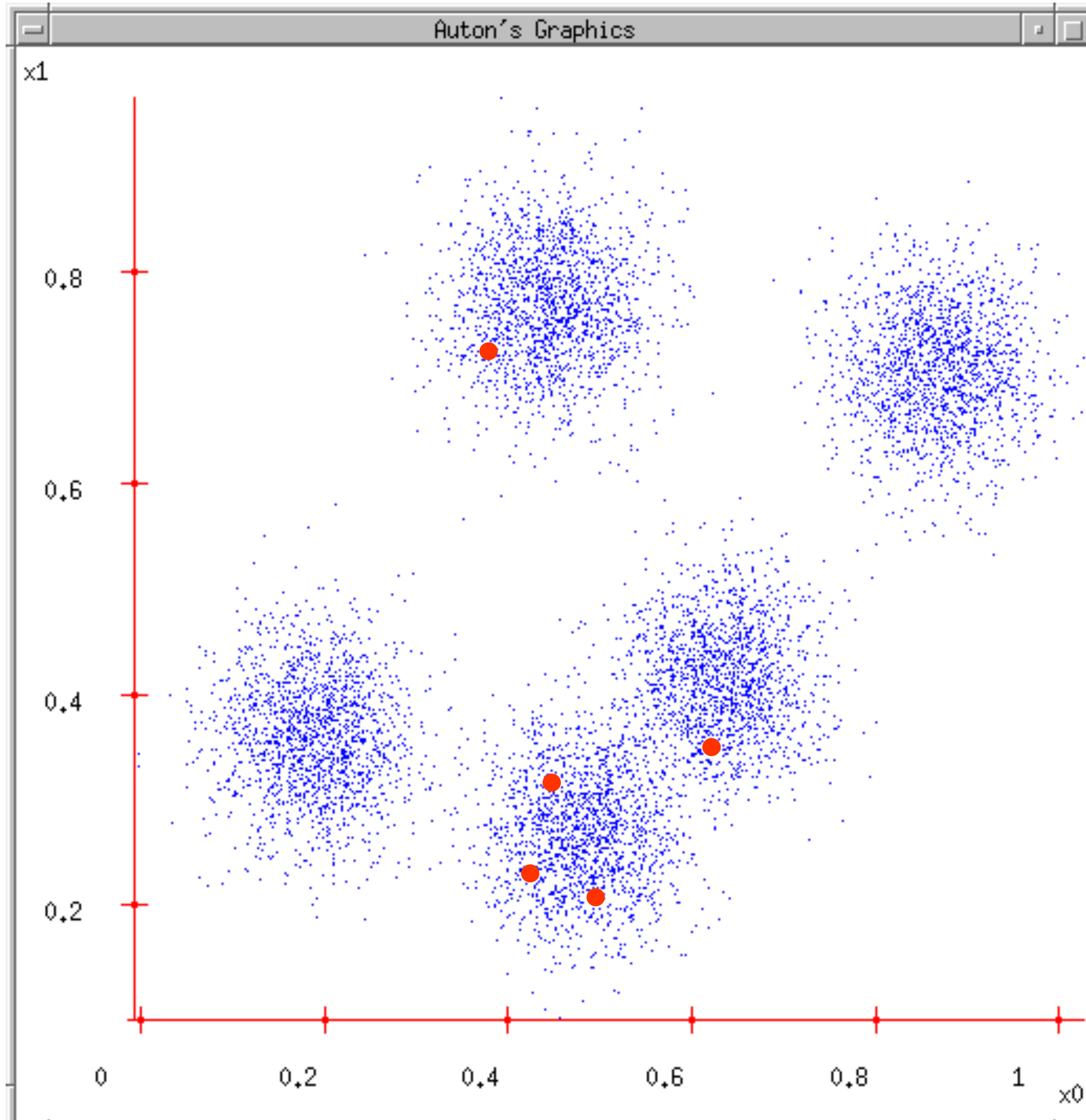
K-means clustering

- The dataset. Input k=5



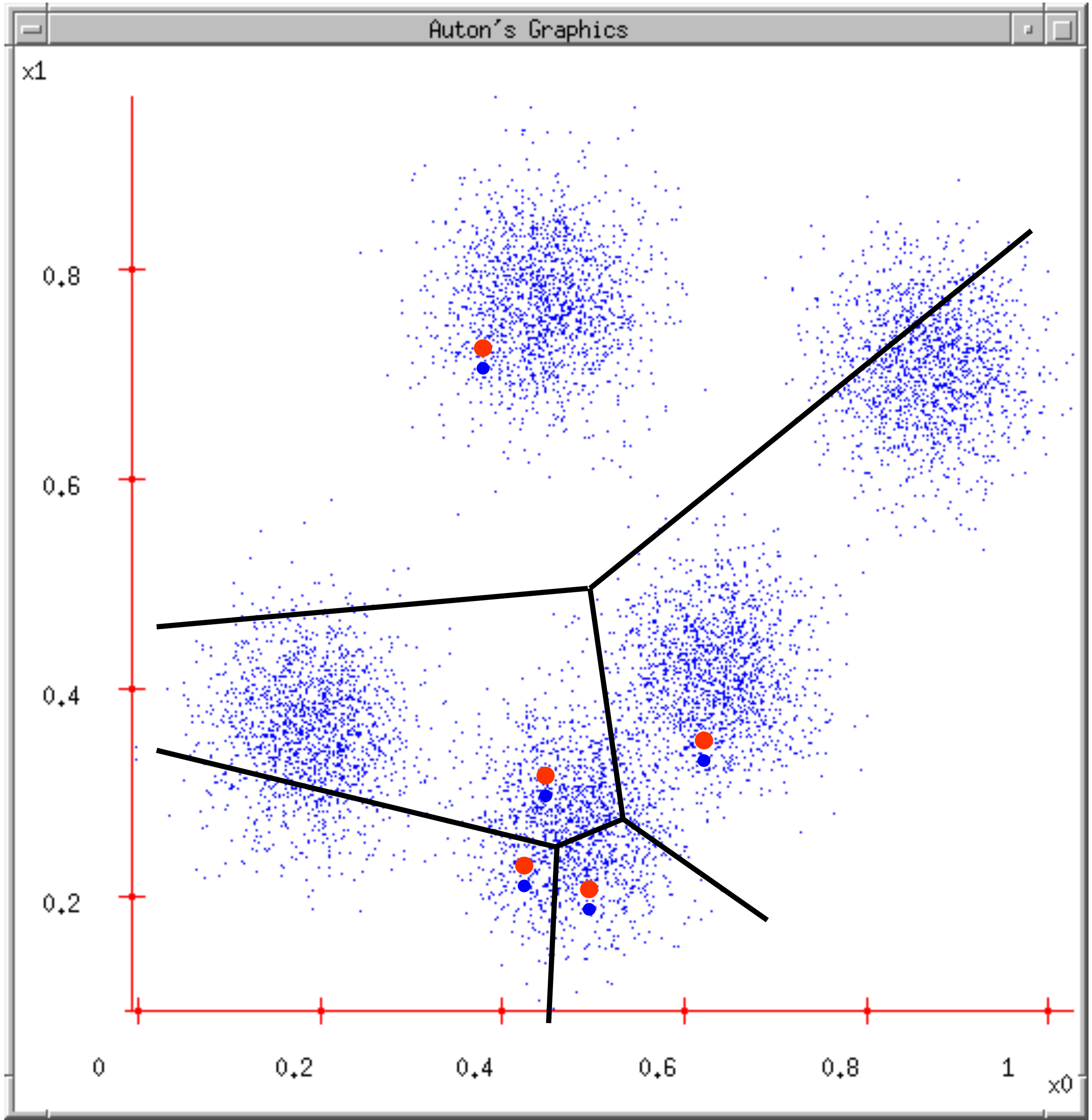
K-means clustering

- Randomly picking 5 positions as initial cluster centers (not necessarily a data point)



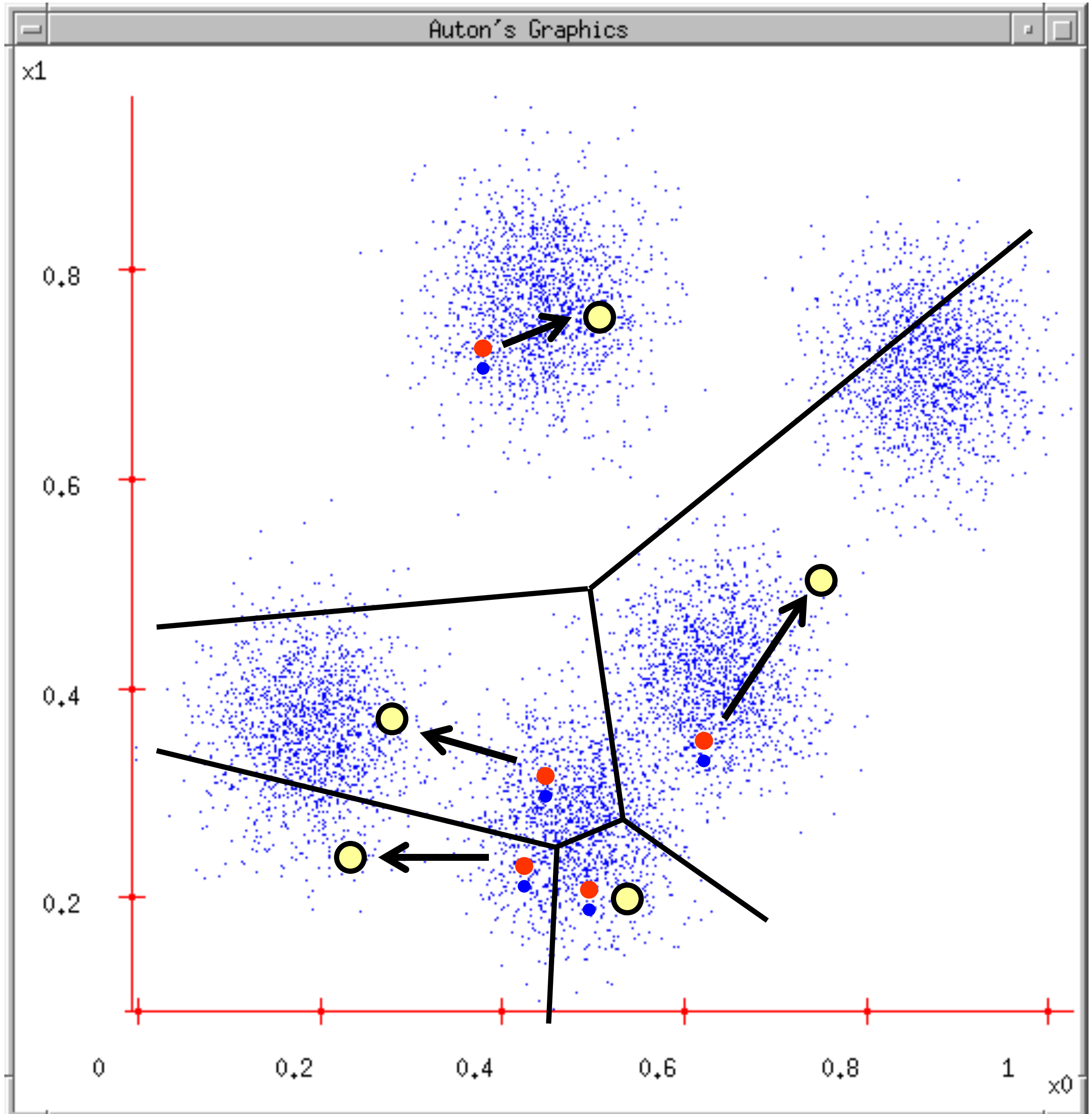
K-means clustering

- Each point finds which cluster center it is closest to (very much like 1NN). The point belongs to that cluster.



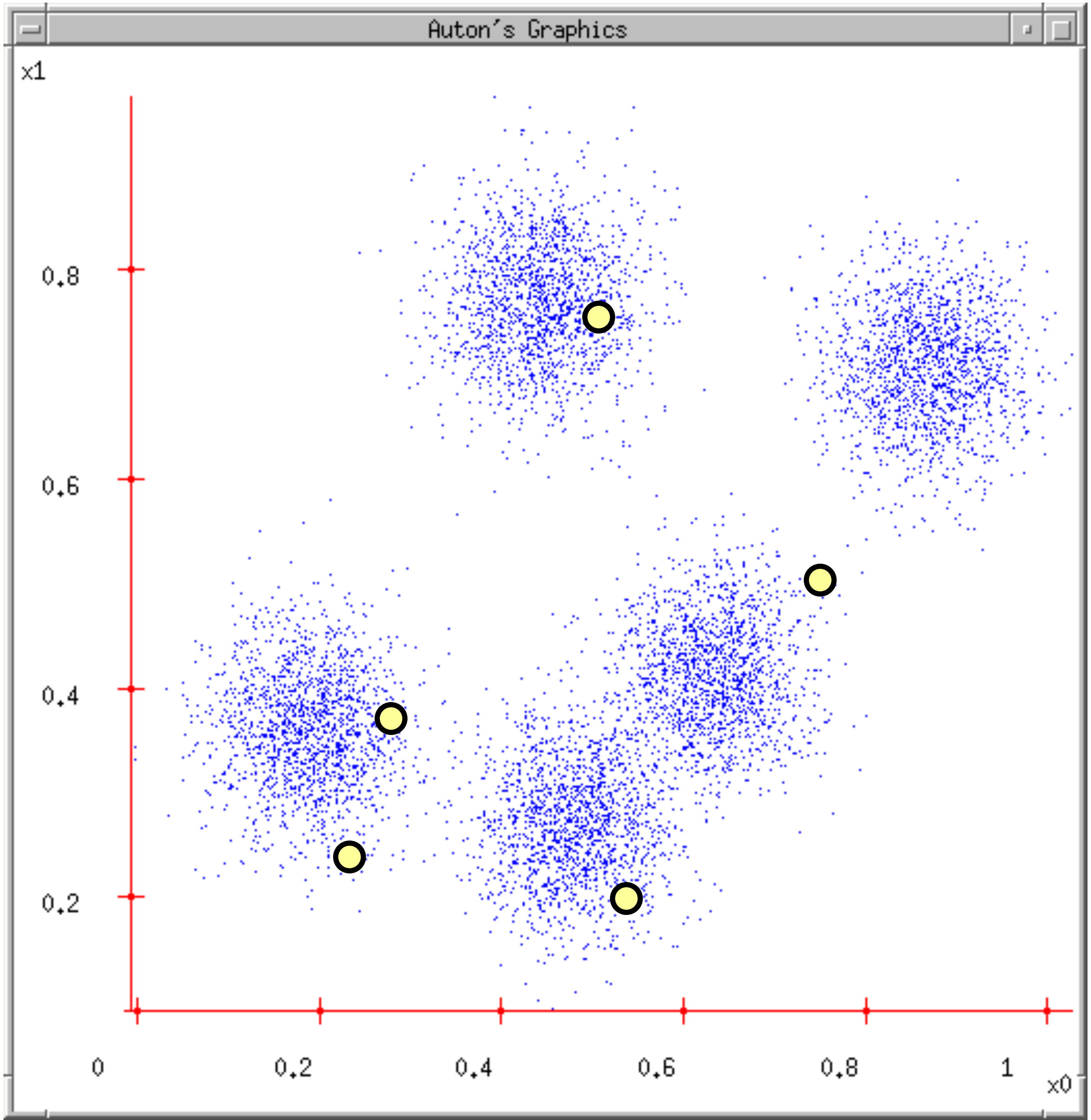
K-means clustering

- Each cluster computes its new centroid, based on which points belong to it

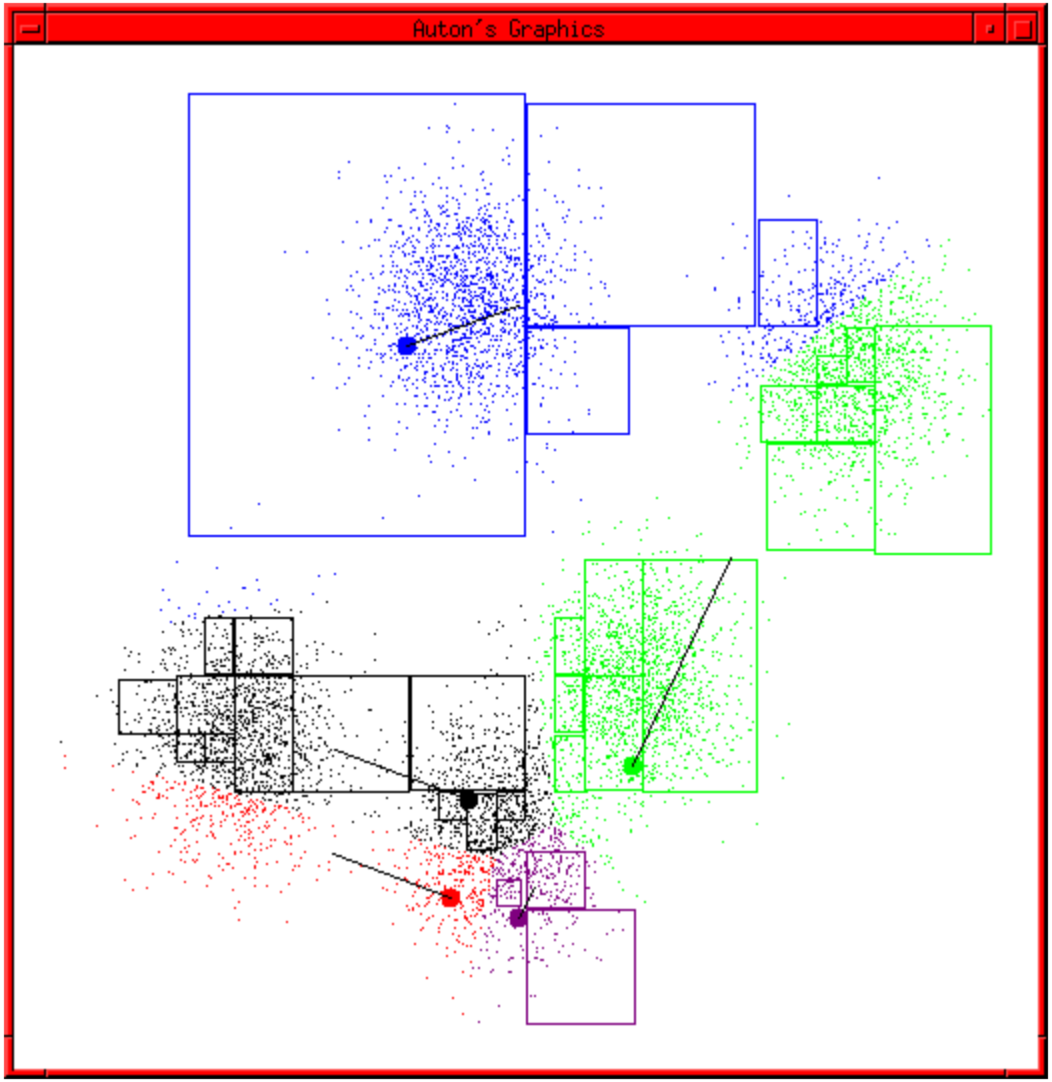


K-means clustering

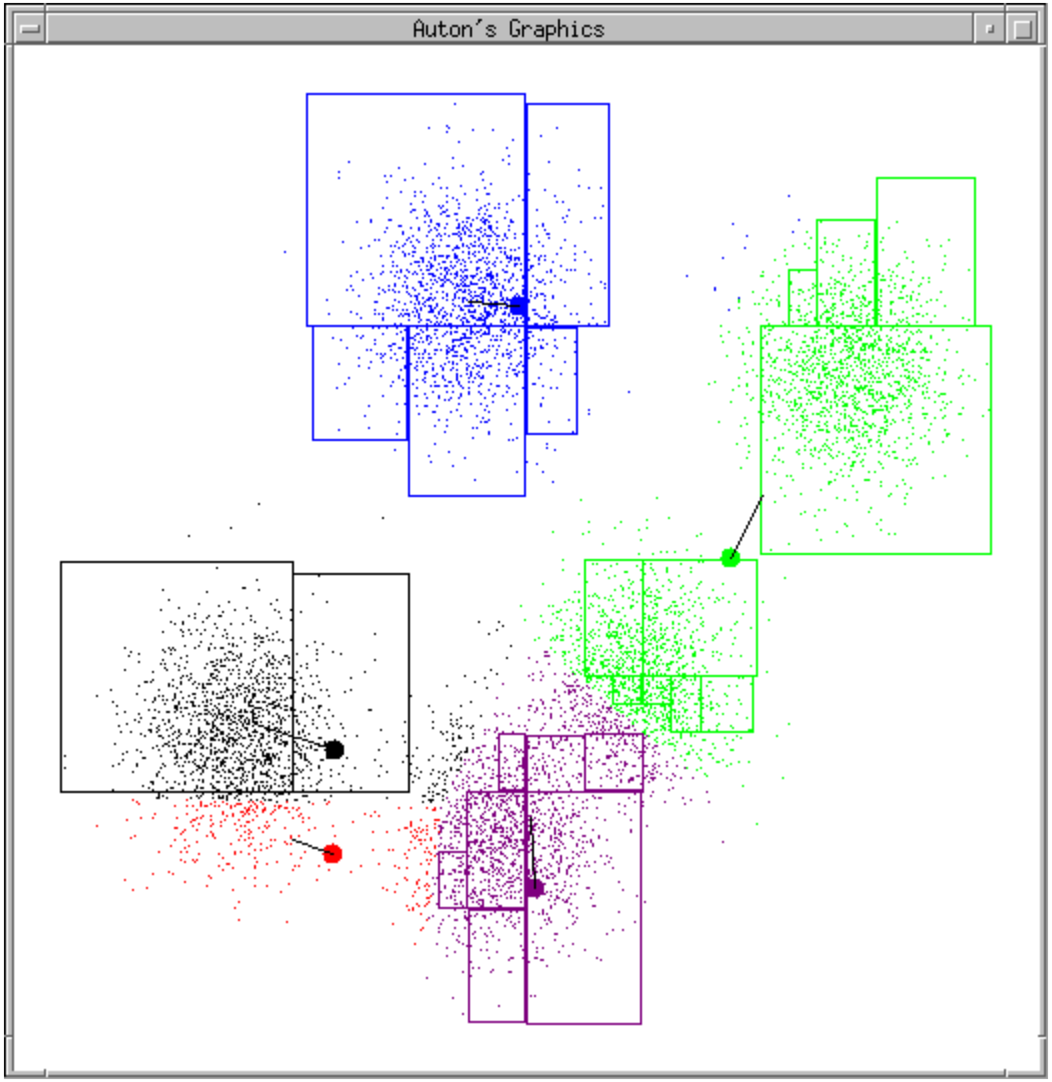
- Each cluster computes its new centroid, based on which points belong to it
- And repeat until convergence (cluster centers no longer move)…



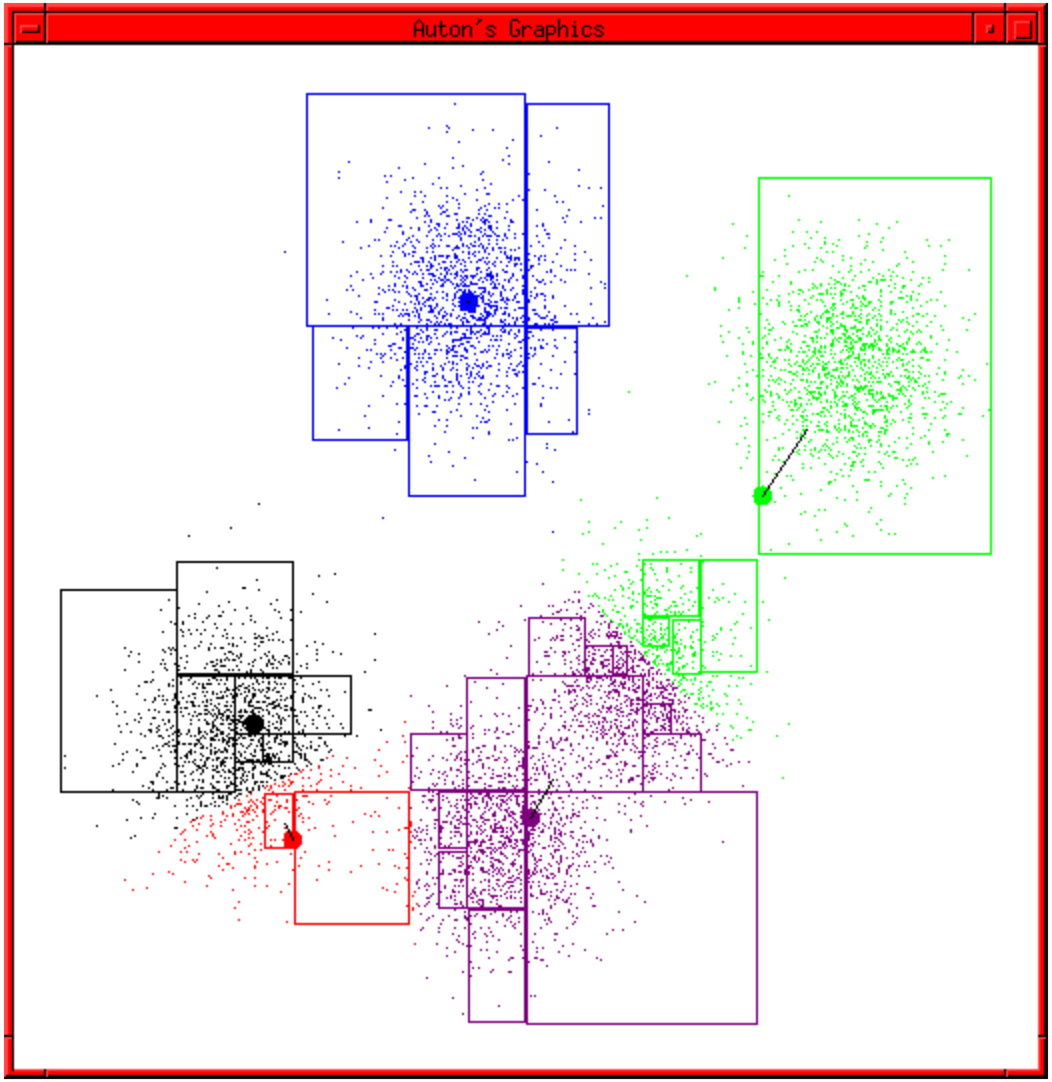
K-means: initial cluster centers



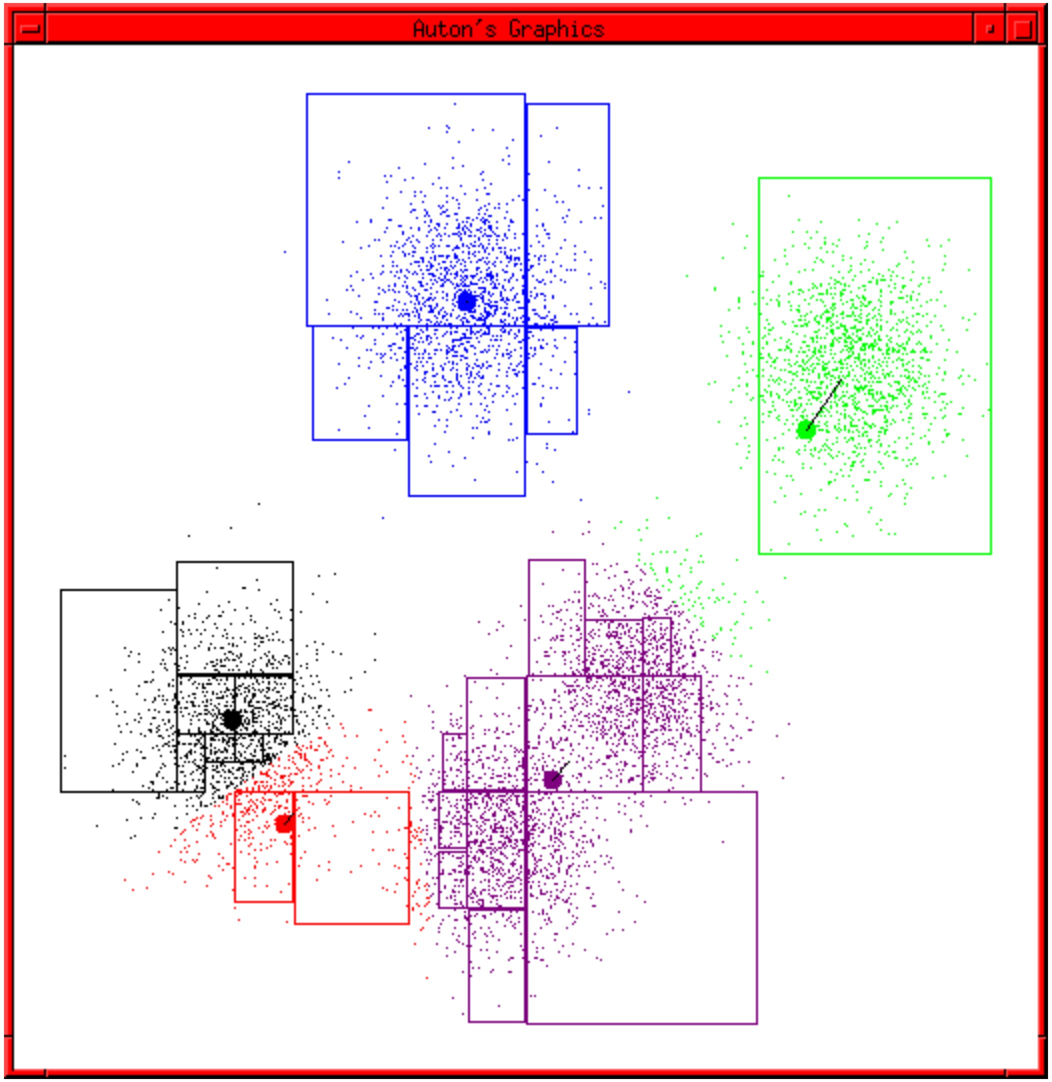
K-means in action



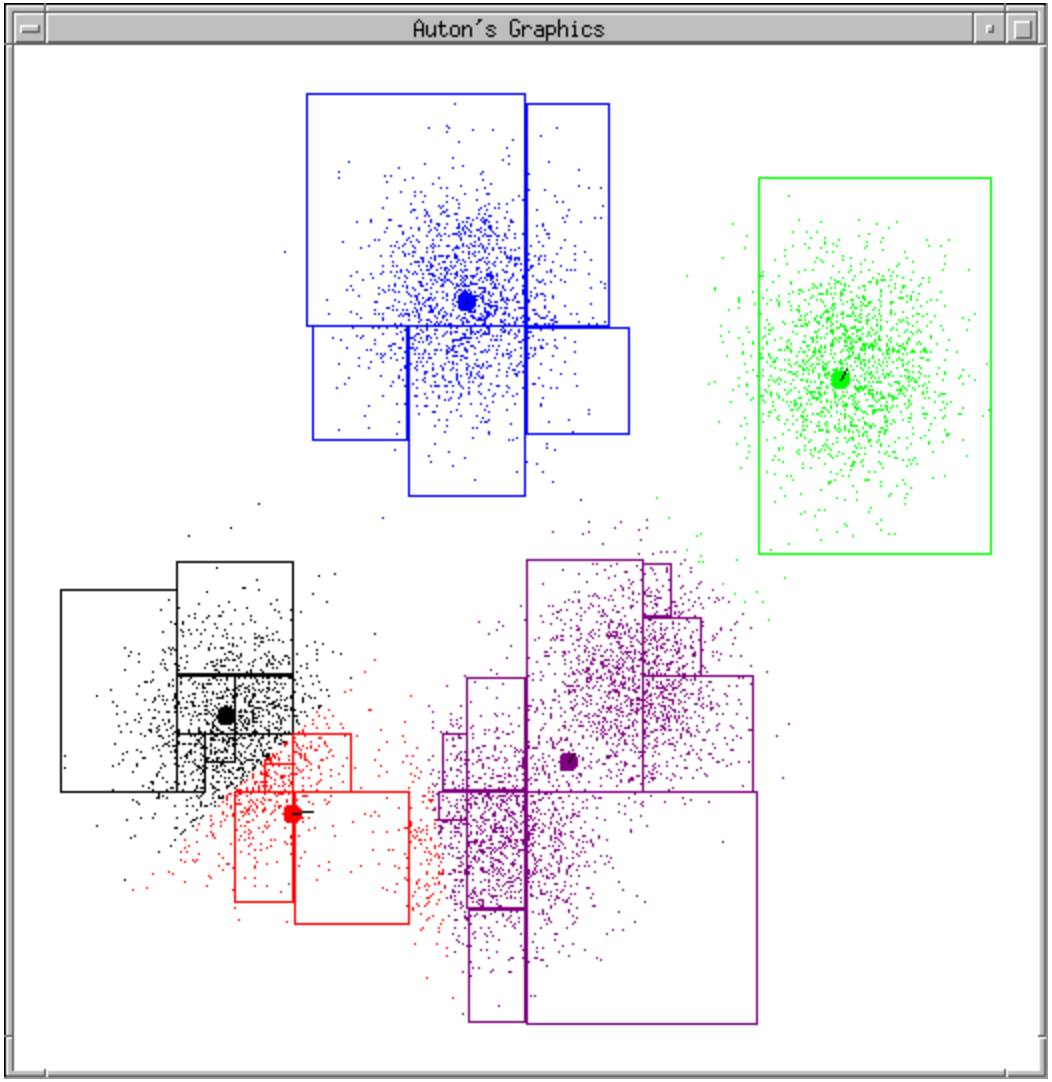
K-means in action



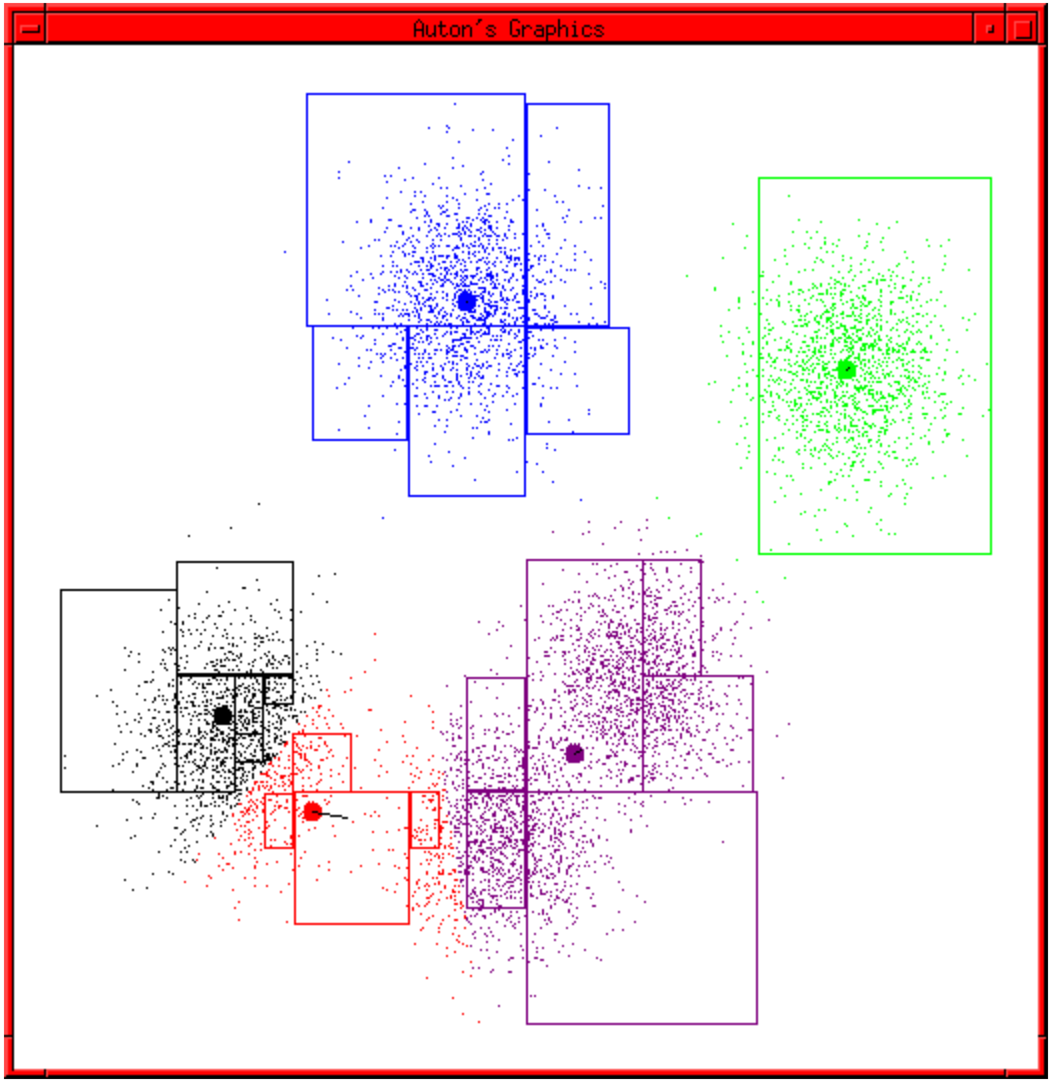
K-means in action



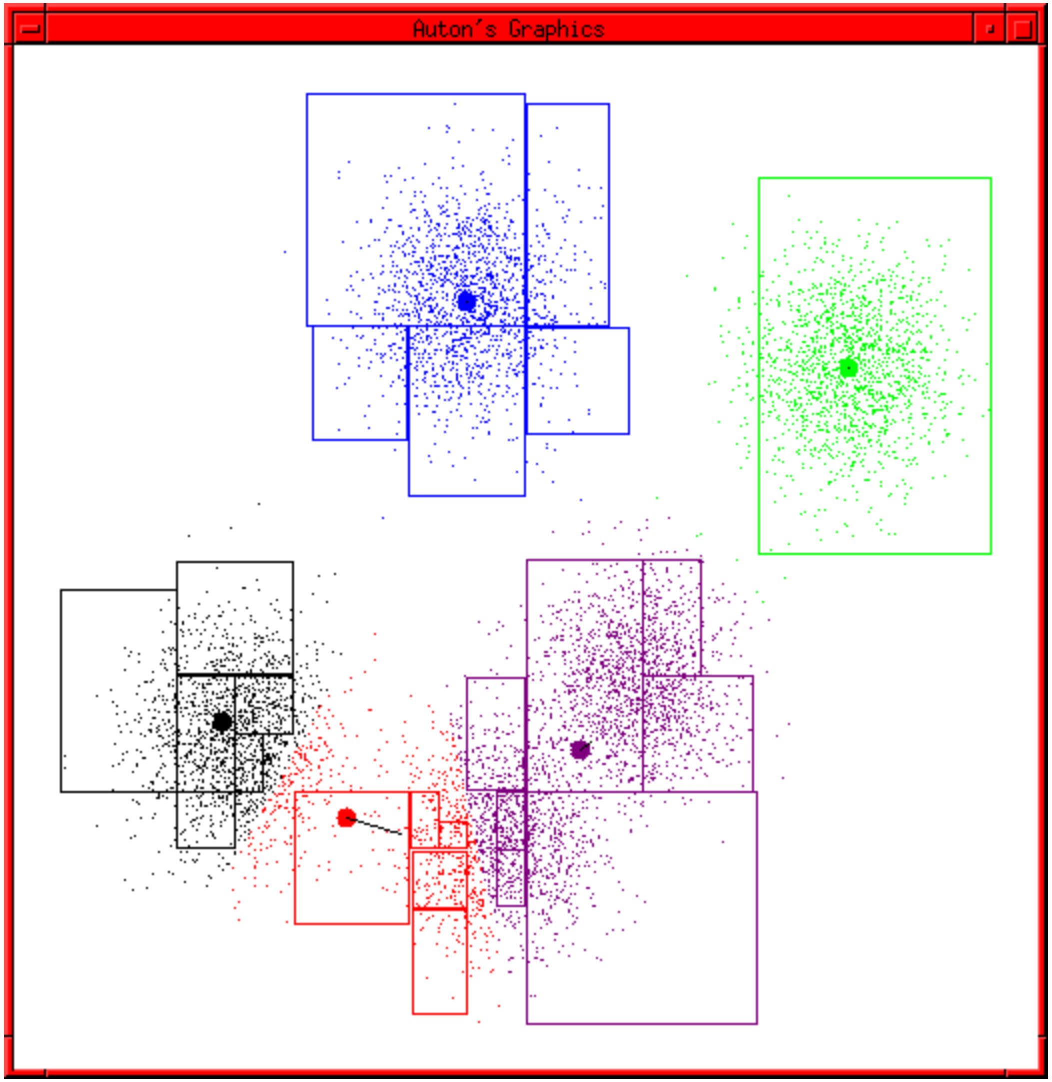
K-means in action



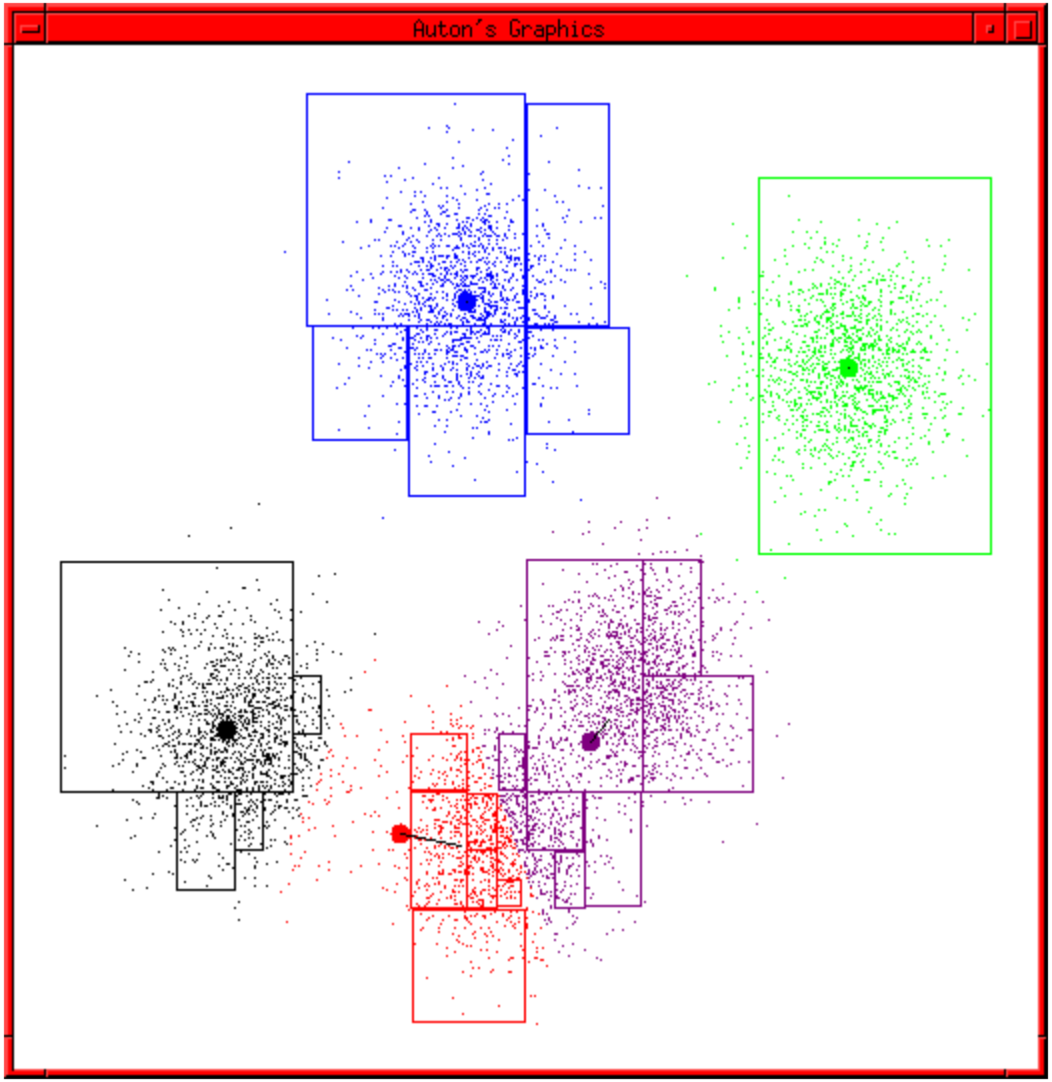
K-means in action



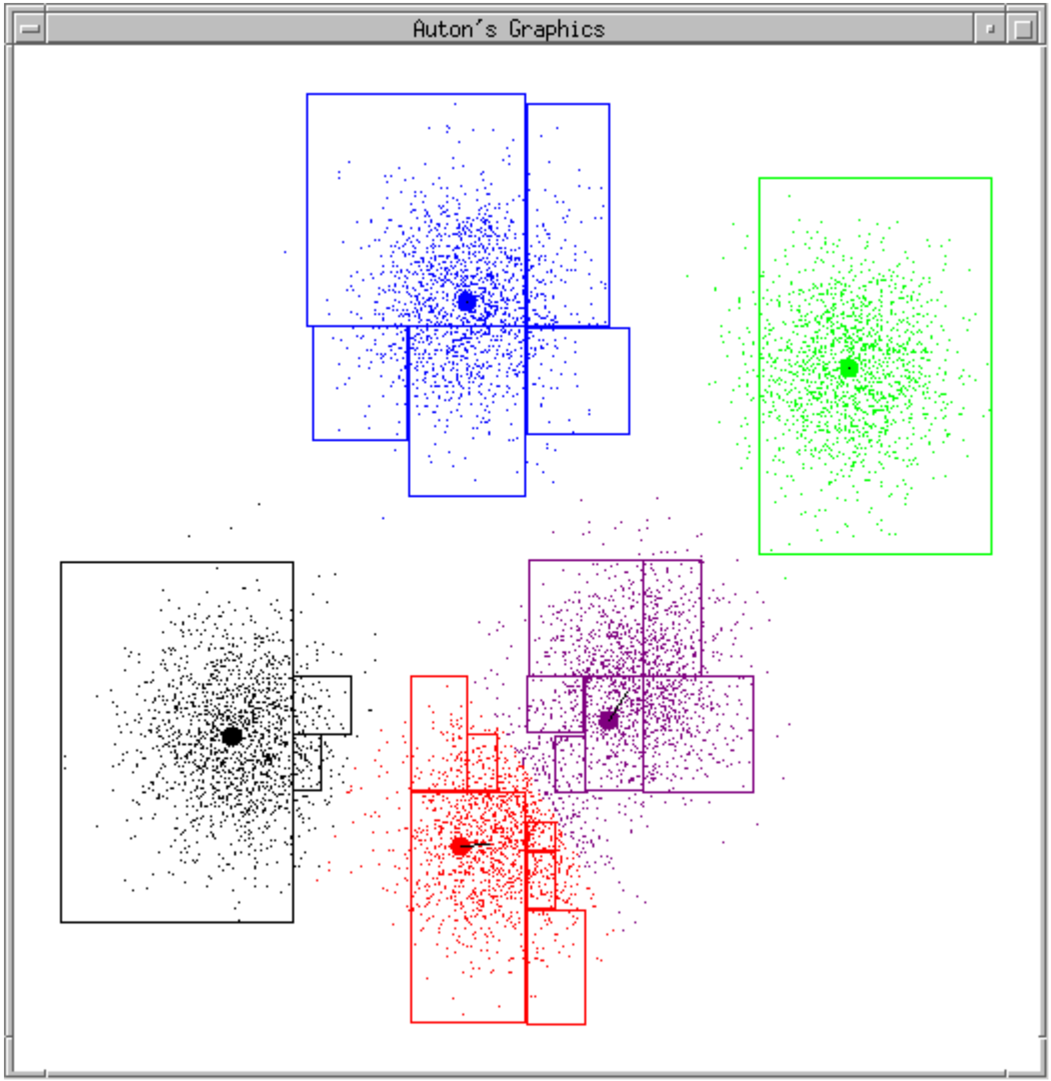
K-means in action



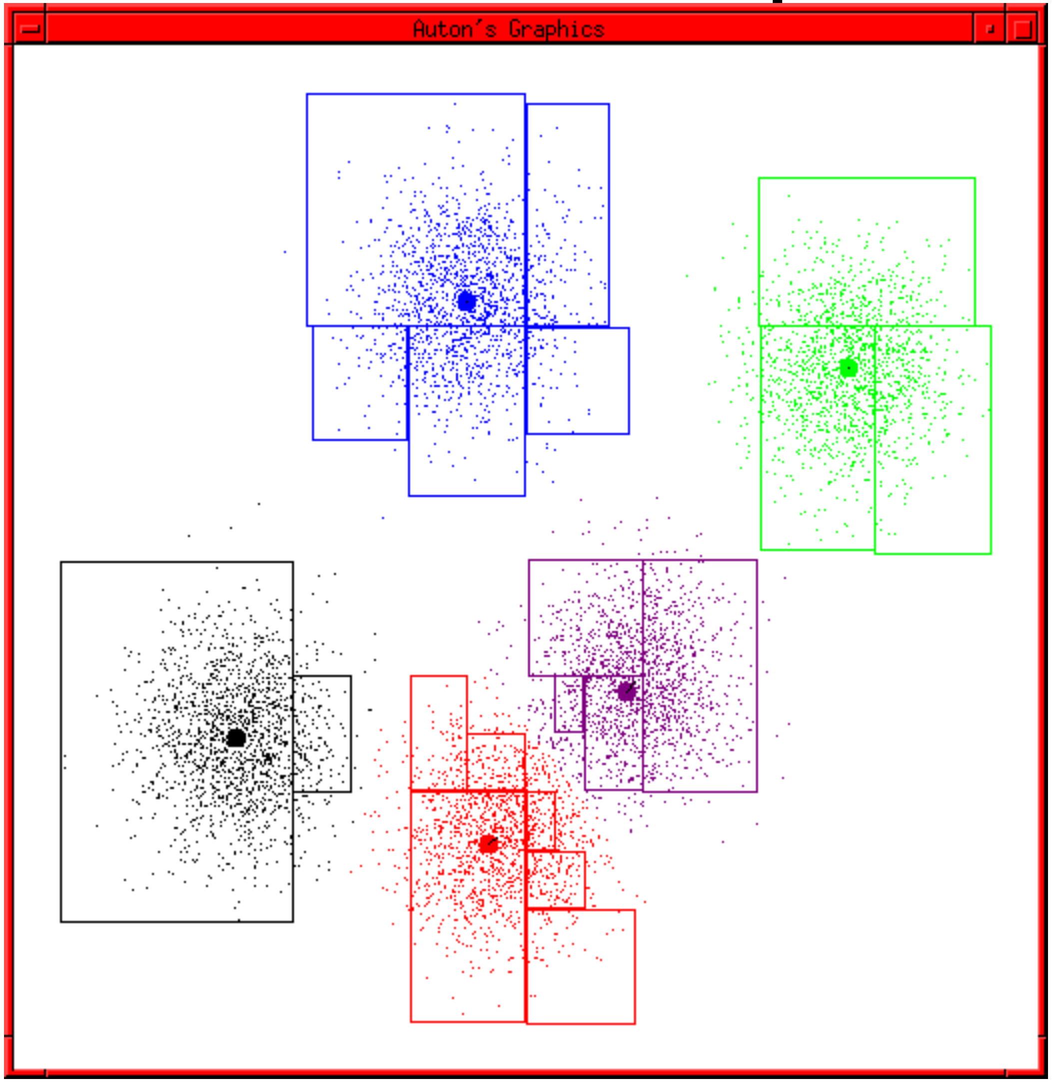
K-means in action



K-means in action



K-means stops



K-means algorithm

- Input: $x_1 \ldots x_n$, k
- **Step 1**: select k cluster centers $c_1 \ldots c_k$
- **Step 2**: for each point x, determine its cluster: find the closest center in Euclidean space
- **Step 3**: update all cluster centers as the centroids

$$c_i = \sum_{\{x \text{ in cluster } i\}} x \,/\, \text{SizeOf}(\text{cluster } i)$$

- Repeat step 2, 3 until cluster centers no longer change

Questions on k-means

- What is k-means trying to optimize?
- Will k-means stop (converge)?
- Will it find a global or local optimum?
- How to pick starting cluster centers?
- How many clusters should we use?

Distortion

- Suppose for a point x, you replace its coordinates by the cluster center $c_{(x)}$ it belongs to (lossy compression)
- How far are you off? Measure it with squared Euclidean distance: x(d) is the d-th feature dimension, y(x) is the cluster ID that x is in.

$$\Sigma_{d=1 \ldots D} [x(d) - c_{y(x)}(d)]^2$$

- This is the distortion of a single point x. For the whole dataset, the distortion is

$$\Sigma_x \Sigma_{d=1 \ldots D} [x(d) - c_{y(x)}(d)]^2$$

The minimization problem

$$\min \Sigma_x \Sigma_{d=1\ldots D} [x(d) - c_{y(x)}(d)]^2$$

$y(x_1)\ldots y(x_n)$

$c_1(1)\ldots c_1(D)$

…

$c_k(1)\ldots c_k(D)$

Step 1

- For fixed cluster centers, if all you can do is to assign x to some cluster, then assigning x to its closest cluster center y(x) minimizes distortion

$$\Sigma_{d=1 \ldots D} [x(d) - c_{y(x)}(d)]^2$$

- Why? Try any other cluster $z \neq y(x)$

$$\Sigma_{d=1 \ldots D} [x(d) - c_z(d)]^2$$

Step 2

- If the assignment of x to clusters are fixed, and all you can do is to change the location of cluster centers
- Then this is a continuous optimization problem!

$$\sum_x \sum_{d=1 \ldots D} [x(d) - c_{y(x)}(d)]^2$$

- Variables?

Step 2

- If the assignment of x to clusters are fixed, and all you can do is to change the location of cluster centers
- Then this is an optimization problem!
- Variables? $c_1(1), \ldots, c_1(D), \ldots, c_k(1), \ldots, c_k(D)$

$$\min \sum_x \sum_{d=1\ldots D} [x(d) - c_{y(x)}(d)]^2$$

$$= \min \sum_{z=1..k} \sum_{y(x)=z} \sum_{d=1\ldots D} [x(d) - c_z(d)]^2$$

- Unconstrained. What do we do?

Step 2

- If the assignment of x to clusters are fixed, and all you can do is to change the location of cluster centers
- Then this is an optimization problem!
- Variables? $c_1(1), \ldots, c_1(D), \ldots, c_k(1), \ldots, c_k(D)$

$$\min \sum_{x} \sum_{d=1\ldots D} [x(d) - c_{y(x)}(d)]^2$$

$$= \min \sum_{z=1..k} \sum_{y(x)=z} \sum_{d=1\ldots D} [x(d) - c_z(d)]^2$$

- Unconstrained.

$$\partial/\partial c_z(d) \sum_{z=1..k} \sum_{y(x)=z} \sum_{d=1\ldots D} [x(d) - c_z(d)]^2 = 0$$

Step 2

- The solution is

$$c_z(d) = \sum_{y(x)=z} x(d) \Big/ |n_z|$$

- The d-th dimension of cluster z is the average of the d-th dimension of points assigned to cluster z
- Or, update cluster z to be the centroid of its points. This is exact what we did in step 2.

Repeat (step1, step2)

- Both step1 and step2 minimizes the distortion
$$\Sigma_x \Sigma_{d=1\ldots D} [x(d) - c_{y(x)}(d)]^2$$
- Step1 changes x assignments y(x)
- Step2 changes c(d) the cluster centers
- However there is no guarantee the distortion is minimized over all… need to repeat
- This is hill climbing (coordinate descent)
- Will it stop?

Repeat (step1, step2)

- Both step1 and step2
- Step1 changes x assig
- Step2 changes c(d) th
- However there is no g repeat
- This is hill climbing (c
- Will it stop?

There are finite number of points

Finite ways of assigning points to clusters

In step1, an assignment that reduces distortion has to be a new assignment not used before

Step1 will terminate

So will step 2

So k-means terminates

What optimum does K-means find

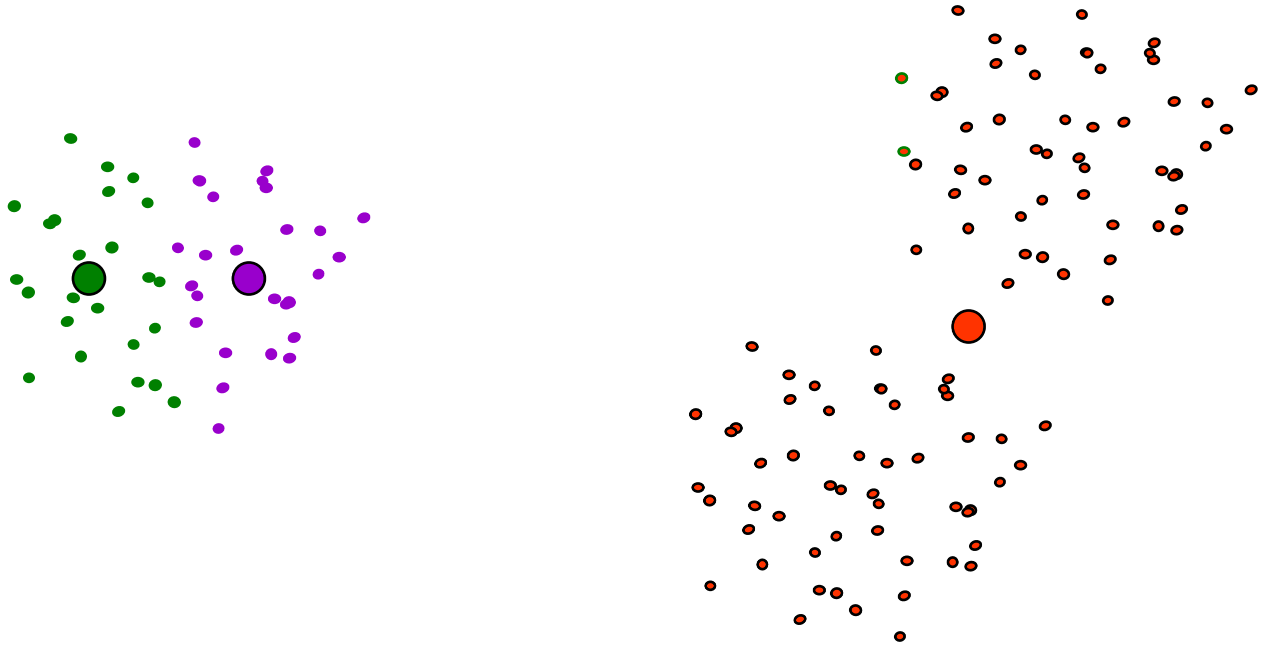
- Will k-means find the global minimum in distortion? Sadly no guarantee…
- Can you think of one example?

What optimum does K-means find

- Will k-means find the global minimum in distortion? Sadly no guarantee…
- Can you think of one example? (Hint: try k=3)

What optimum does K-means find

- Will k-means find the global minimum in distortion? Sadly no guarantee…
- Can you think of one example? (Hint: try k=3)



Picking starting cluster centers

- Which local optimum k-means goes to is determined solely by the starting cluster centers
  - Be careful how to pick the starting cluster centers. Many ideas. Here's one neat trick:
    1. Pick a random point x1 from dataset
    2. Find the point x2 farthest from x1 in the dataset
    3. Find x3 farthest from the closer of x1, x2
    4. … pick k points like this, use them as starting cluster centers for the k clusters
  - Run k-means multiple times with different starting cluster centers (hill climbing with random restarts)

Picking the number of clusters

- Difficult problem
- Domain knowledge?
- Otherwise, shall we find k which minimizes distortion?

Picking the number of clusters

- Difficult problem
- Domain knowledge?
- Otherwise, shall we find k which minimizes distortion? k = N, distortion = 0
- Need to regularize. A common approach is to minimize the Schwarz criterion

$$\text{distortion} + \lambda\,(\#\text{param})\,\log N$$

$$= \text{distortion} + \lambda\, D\, k\, \log N$$

#dimensions

#clusters

#points

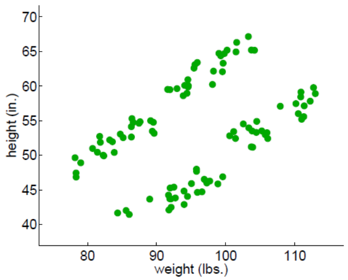
Beyond k-means

- In k-means, each point belongs to one cluster
- What if one point can belong to more than one cluster?
- What if the degree of belonging depends on the distance to the centers?

- This will lead to the famous EM algorithm, or expectation-maximization
- K-means is a discrete version of EM algorithm with Gaussian mixture models with infinitely small covariances… (not covered in this class)

Teacher shows labels

SUPERVISED LEARNING



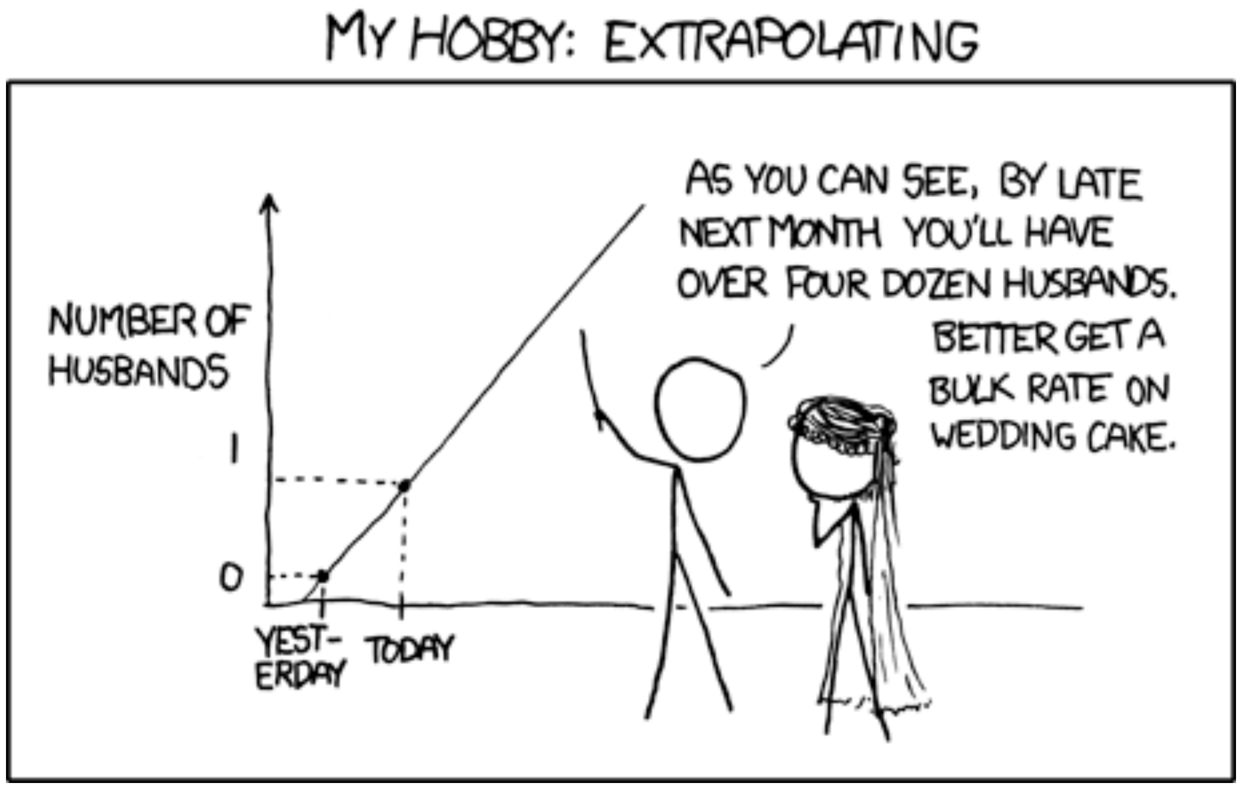
Label

- Little green men:
  - Predict gender (M,F) from weight, height?
  - Predict adult, juvenile from weight, height?
- A label *y* is the desired prediction on an instance ***x***
- Discrete label: classes
  - M,F; A,J: often encode as 0,1 or -1,1
  - Multiple classes: 1,2,3,…,C. No class order implied.
- Continuous label: e.g., blood pressure

Supervised learning

- A labeled training sample is a collection of instances $(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_n, y_n)$
- Assume $(\boldsymbol{x}_i, y_i) \overset{\text{i.i.d.}}{\sim} P(x,y)$. Again, $P(x,y)$ is unknown
- Supervised learning learns a function $f: X \rightarrow Y$ in some function family F, such that $f(\boldsymbol{x})$ predicts the true label y on future data $\boldsymbol{x}$, where $(\boldsymbol{x}, y) \overset{\text{i.i.d.}}{\sim} P(x,y)$
  - Classification: if y discrete
  - Regression: if y continuous

http://xkcd.com/605/



Evaluation

- Training set error
  - 0-1 loss for classification $\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) \neq y_i),$
  - squared loss for regression $\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2$
  - overfitting
- Test set error: use a separate test set
- True error of *f*: $\mathbb{E}_{(\mathbf{x},y)\sim P}\left[c(\mathbf{x}, y, f(\mathbf{x}))\right]$, where c() is an appropriate loss function
- Goal of supervised learning is to find

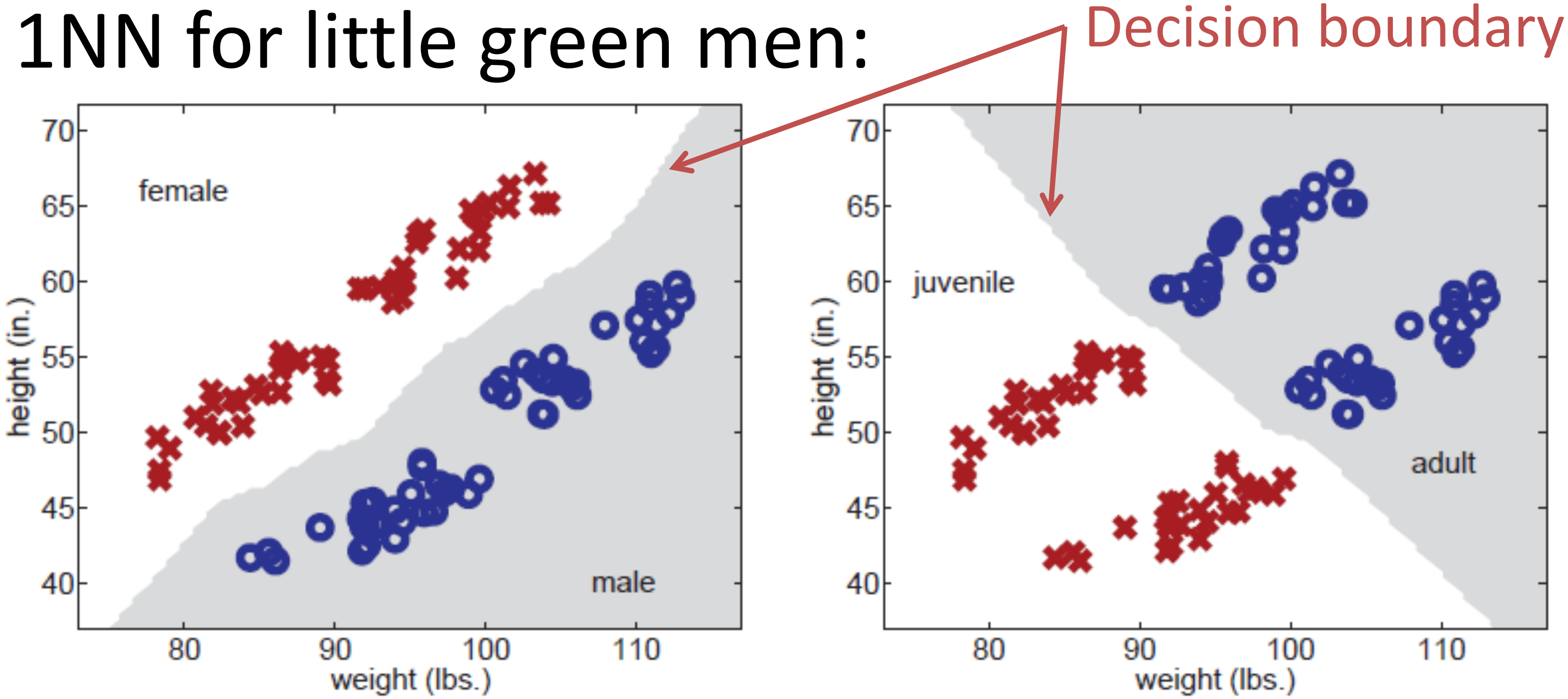
$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \mathbb{E}_{(\mathbf{x},y)\sim P}\left[c(\mathbf{x}, y, f(\mathbf{x}))\right]$$

k-nearest-neighbor (kNN)

Input: Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$; *distance function* $d()$;
number of neighbors k; *test instance* $\mathbf{x}^*$

1. *Find the* k *training instances* $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ *closest to* $\mathbf{x}^*$ *under distance* $d()$.
2. *Output* y^* *as the majority class of* $y_{i_1}, \ldots, y_{i_k}$. *Break ties randomly.*

- 1NN for little green men:



(a) classification by gender (b) classification by age

kNN

- Demo
- What if we want regression?
  – Instead of majority vote, take average of neighbors’ y
- How to pick *k*?
  – Split data into training and tuning sets
  – Classify tuning set with different *k*
  – Pick *k* that produces least tuning-set error

Summary

- Feature representation
- Unsupervised learning / Clustering
  - Hierarchical Agglomerative Clustering
  - K-means clustering
- Supervised learning / Classification
  - k-nearest-neighbor