COMP 551 - Applied Machine Learning Lecture 18 — Unsupervised learning

William L. Hamilton

(with slides and content from Joelle Pineau)

What is unsupervised learning?

- Given only input data: $D = <x_i>, i=1:n$, find some patterns or regularity in the data.

- Different classes of problems:
  1. Clustering
  2. Anomaly detection
  3. Dimensionality reduction

What is unsupervised learning?

- Given only input data: $D = <x_i>, i=1:n$, find some patterns or regularity in the data.

- Different classes of problems:
  1. Clustering
  2. Anomaly detection
  3. Dimensionality reduction

A simple clustering example

- A fruit merchant approaches you, with a set of apples to classify according to their variety.
  - Tells you there are five varieties of apples in the dataset.
  - Tells you the weight and diameter of each apple in the dataset.

- Can you label each apple with the correct variety?
  - What would you need to know / assume?

$$\text{Data} = \langle x_1, ? \rangle, \langle x_2, ? \rangle, \ldots, \langle x_n, ? \rangle$$

A simple clustering example

- You know there are 5 varieties.

- Assume each variety generates apples according to a (variety-specific) 2-dimensional Gaussian distribution.

A simple clustering example

- You know there are 5 varieties.
- Assume each variety generates apples according to a (variety-specific) 2-dimensional Gaussian distribution.
- If you know μ_i, σ_i^2 for each class, it's easy to classify the apples.
- If you know the class of each apple, it's easy to estimate μ_i, σ_i^2.

A simple clustering example

- You know there are 5 varieties.
- Assume each variety generates apples according to a (variety-specific) 2-dimensional Gaussian distribution.
- If you know μ_i, σ_i^2 for each class, it's easy to classify the apples.
- If you know the class of each apple, it's easy to estimate μ_i, σ_i^2.

But what if we know neither?

A simple algorithm: K-means clustering

- Objective: Cluster n instances into K distinct classes.
- Preliminaries:
  - Step 1: Pick the desired number of clusters, K.
  - Step 2: Assume a parametric distribution for each class (e.g., Normal).
  - Step 3: Randomly estimate the parameters of the K distributions.
- Iterate, until convergence:
  - Step 4: Assign instances to the most likely classes.
  - Step 5: Estimate the parametric distribution of each class based on the latest assignment.

K-means iterations: Simple K-means

- Simple K-means: just represent each cluster as a mean/centroid.
  - The likelihood that a point belongs to a cluster is simply proportional to the distance from the centroid.
  - Parametric cluster representation: $\{\mu_1, \ldots, \mu_k\}$

Feature vector for point $\mathbf{i}$.

Mean/centroid of cluster $\mathbf{k}$ at iteration $\mathbf{t}$.

- Assign instances to the most likely classes:

Cluster assignment of point $\mathbf{i}$ at iteration $\mathbf{t}$.

$$c_i^{(t)} = \arg\min_k \|\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)}\|$$

- Estimate the new parametric distribution of each class:

New mean/centroid for cluster $\mathbf{k}$.

Set of points currently assigned to cluster $\mathbf{k}$.

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{i \in \mathcal{C}_k^{(t)}} \mathbf{x}_i$$

K-means iterations: Gaussian K-means

- Gaussian K-means: represent each cluster as a normal distribution.
  - The likelihood that a point belongs to a cluster is defined by a normal distribution.
  - Parametric cluster representation: means $\{\mu_1, \ldots, \mu_K\}$ and co-variances $\{\Sigma_1, \ldots, \Sigma_K\}$

- Assign instances to the most likely classes:

Probability that point i was generated by a Gaussian with mean $\mu^{(t)}_k$ and co-variance $\Sigma_k^{(t)}$.

$$c_i = \arg\max_k P_{\mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}(\mathbf{x}_i)$$

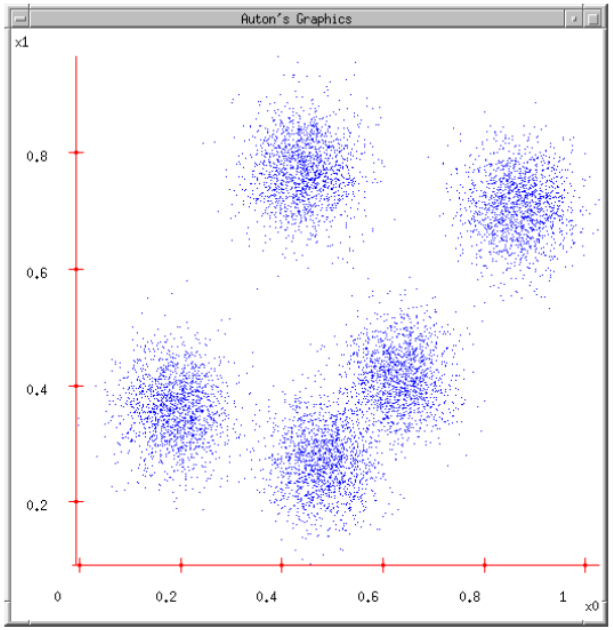
- Estimate the new parametric distribution of each class:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{i \in \mathcal{C}_k^{(t)}} \mathbf{x}_i \qquad \boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{|\mathcal{C}_k^{(t)}| - 1} \sum_{i \in \mathcal{C}_k^{(t)}} (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)})(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)})^\top$$

K-means algorithm

1. Ask user how many clusters.

Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm

1. Ask user how many clusters.

2. Randomly guess k centers:
   $\{ \mu_1, \ldots, \mu_k \}$

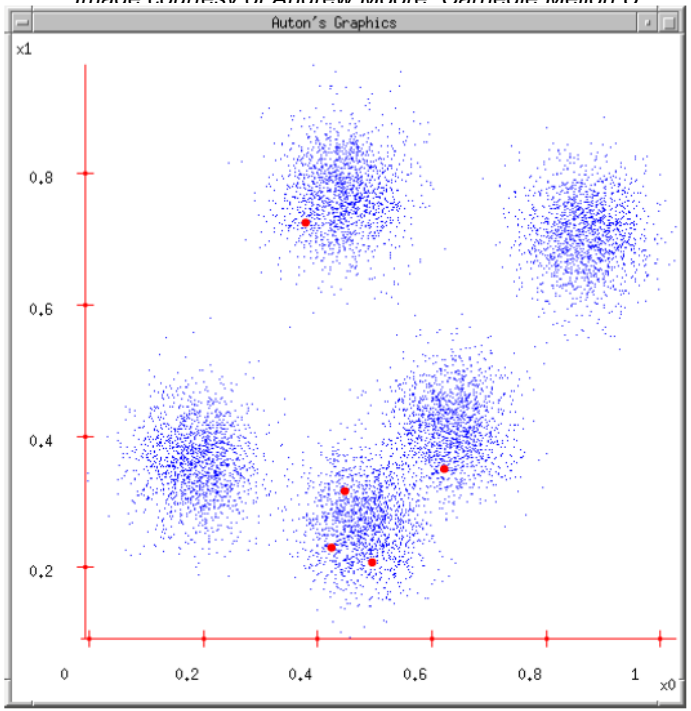
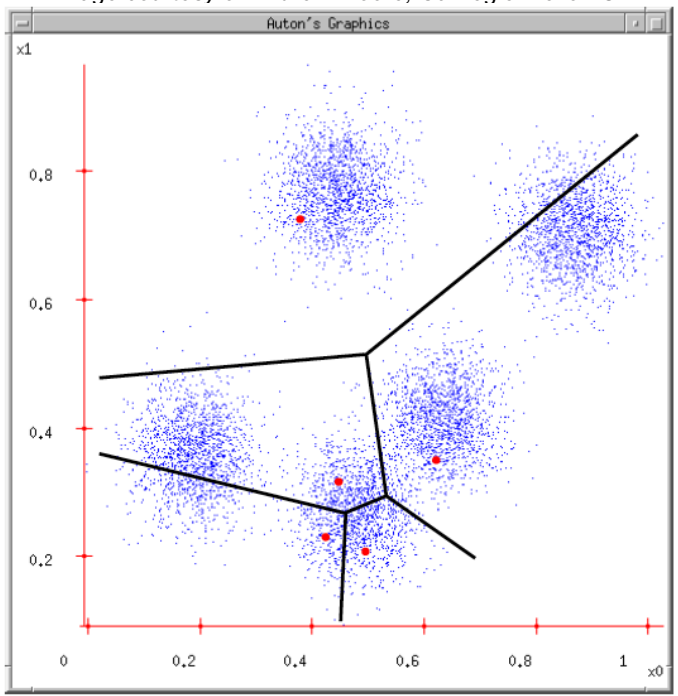


Image courtesy of Andrew Moore, Carnegie Mellon U.

K-means algorithm

1. Ask user how many clusters.
2. Randomly guess k centers:
   $\{ \mu_1, \ldots, \mu_k \}$
3. Assign each data point to the closest center.

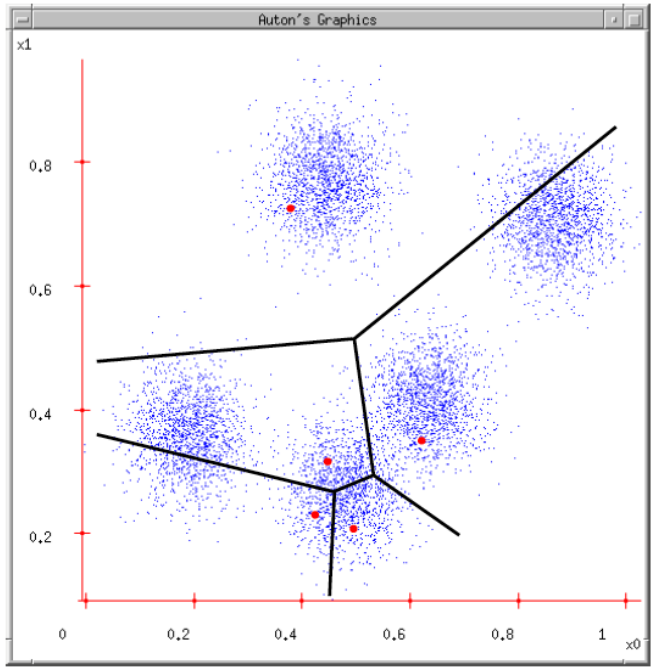
Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm

1. Ask user how many clusters.
2. Randomly guess k centers:
   $\{ \mu_1, \ldots, \mu_k \}$
3. Assign each data point to the closest center.
4. Each center finds the centroid of the points it owns… and jumps there.

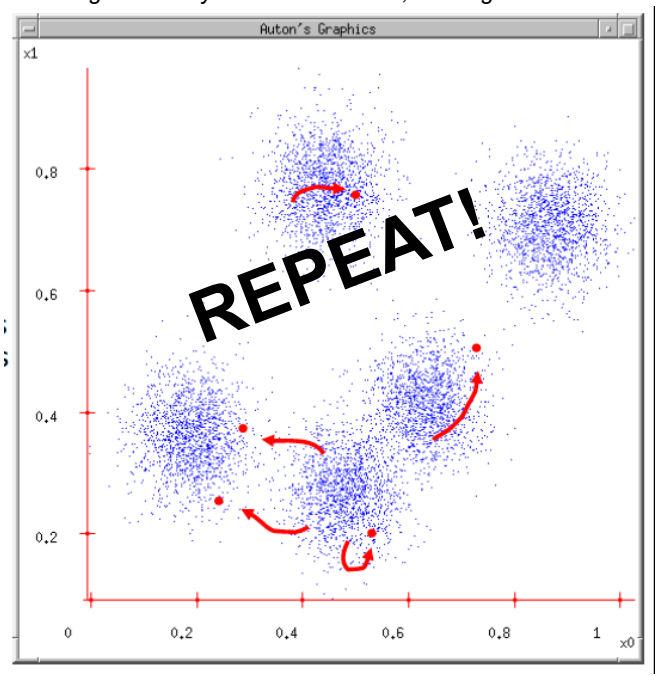
Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm

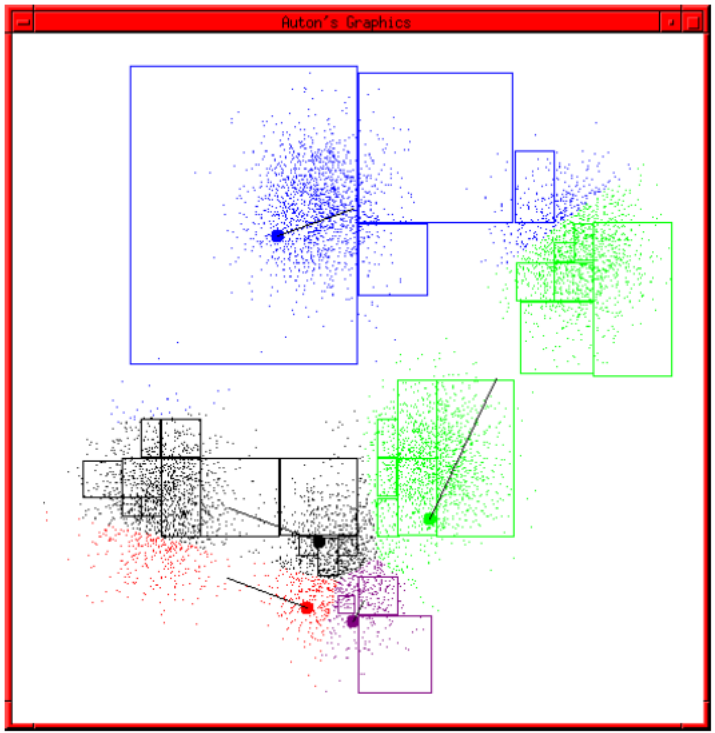
1. Ask user how many clusters.
2. Randomly guess k centers:
   $\{ \mu_1, \ldots, \mu_k \}$
3. Assign each data point to the closest center.
4. Each center finds the centroid of the points it owns… and jumps there.

Image courtesy of Andrew Moore, Carnegie Mellon U.



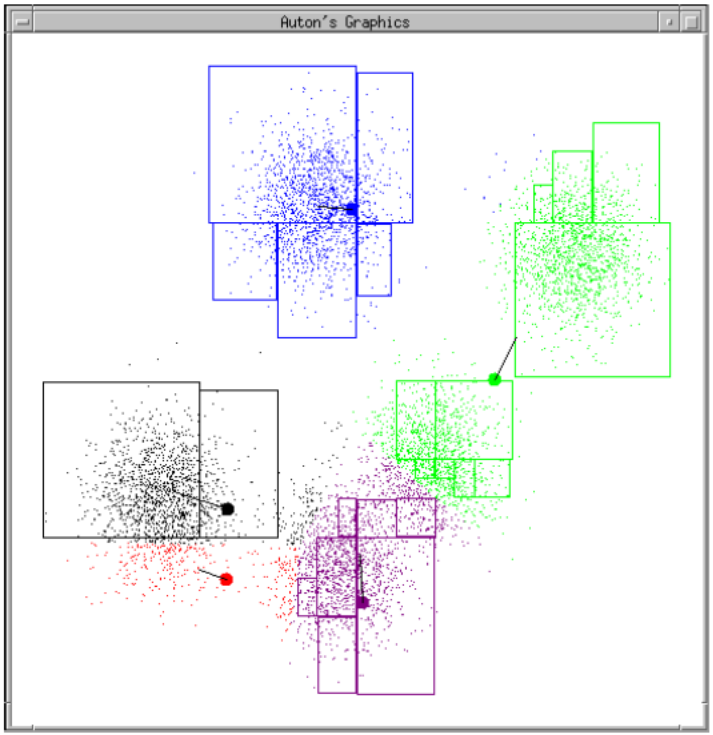
K-means algorithm starts

Image courtesy of Andrew Moore, Carnegie Mellon U.



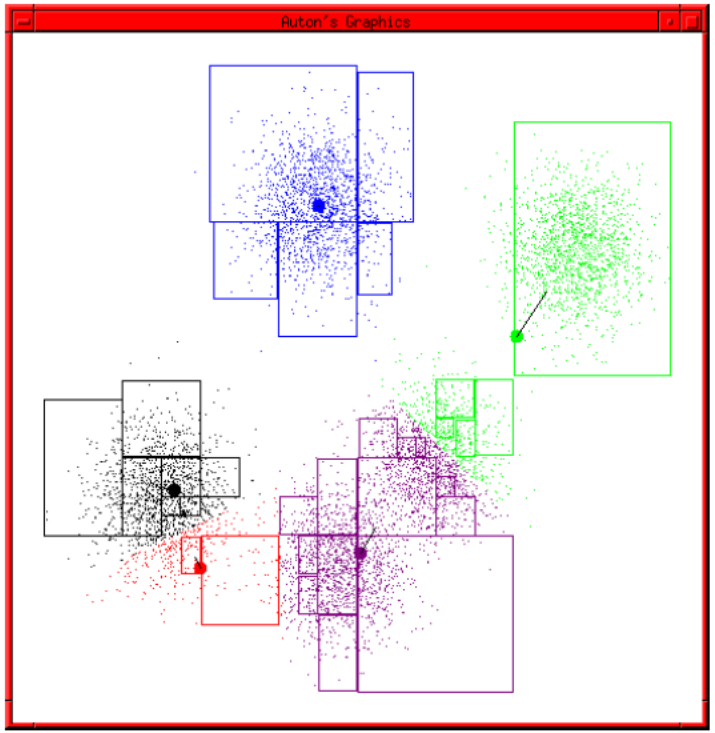
K-means algorithm continues (2)

Image courtesy of Andrew Moore, Carnegie Mellon U.



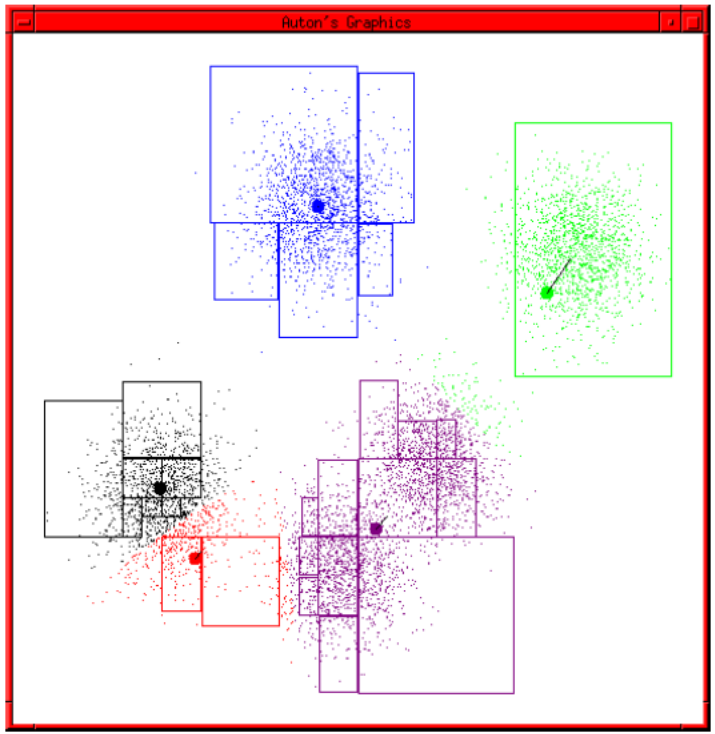
K-means algorithm continues (3)

Image courtesy of Andrew Moore, Carnegie Mellon U.



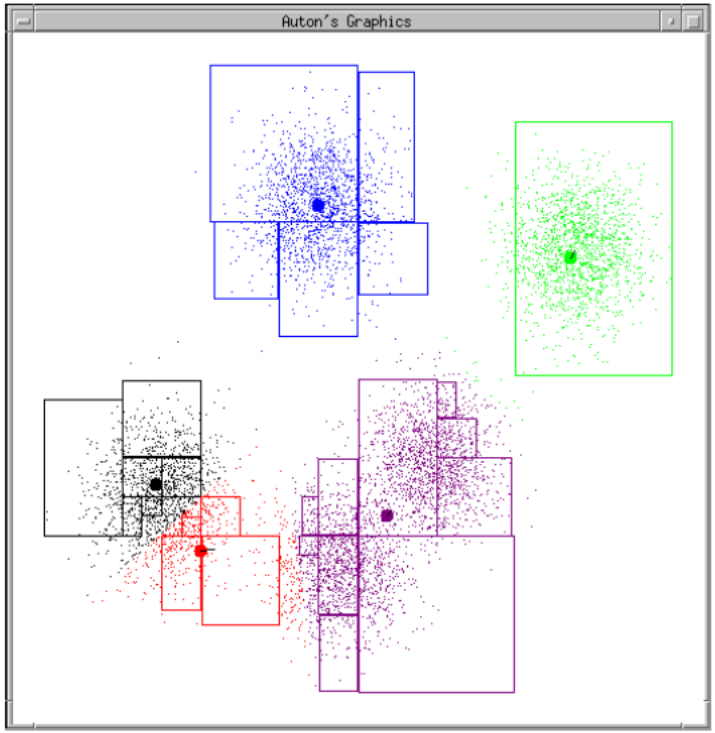
K-means algorithm continues (4)

Image courtesy of Andrew Moore, Carnegie Mellon U.



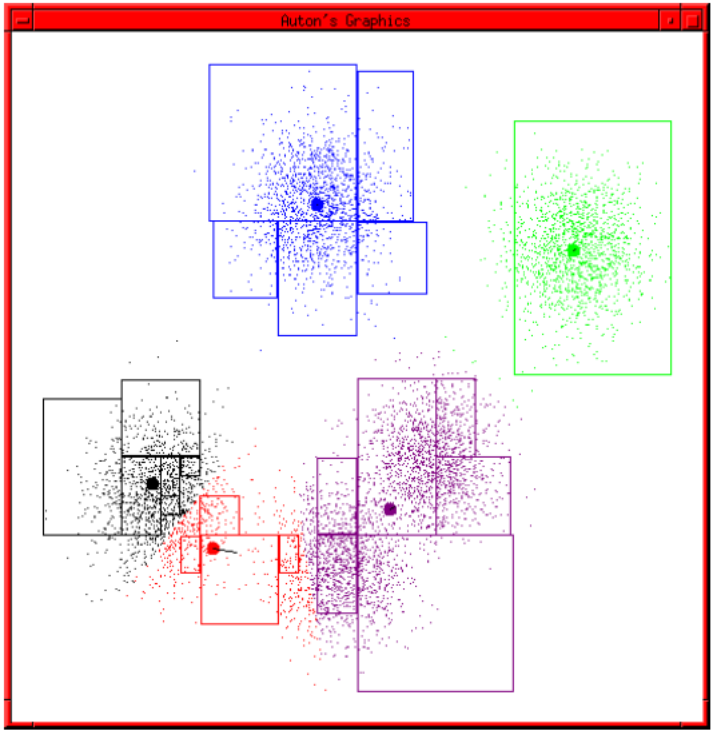
K-means algorithm continues (5)

Image courtesy of Andrew Moore, Carnegie Mellon U.



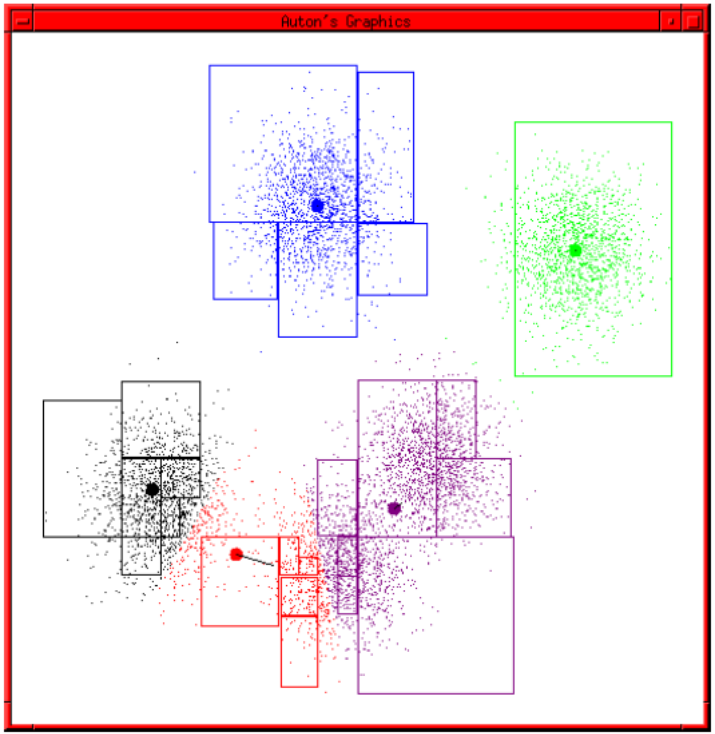
K-means algorithm continues (6)

Image courtesy of Andrew Moore, Carnegie Mellon U



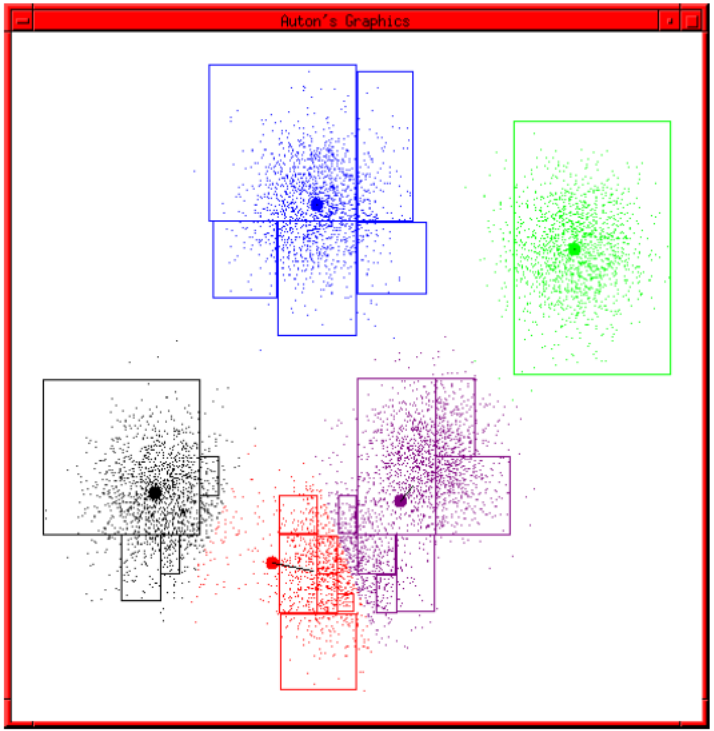
K-means algorithm continues (7)

Image courtesy of Andrew Moore, Carnegie Mellon U.



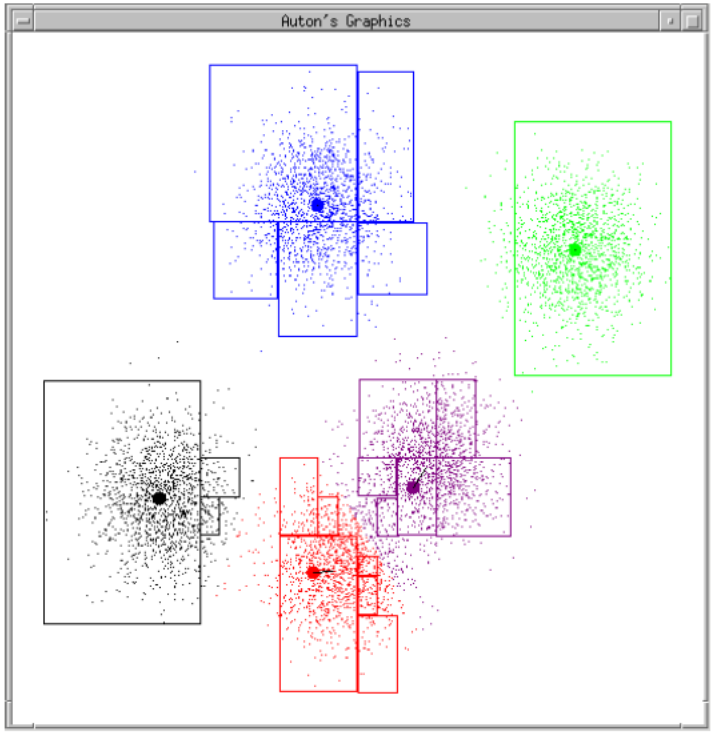
K-means algorithm continues (8)

Image courtesy of Andrew Moore, Carnegie Mellon U.



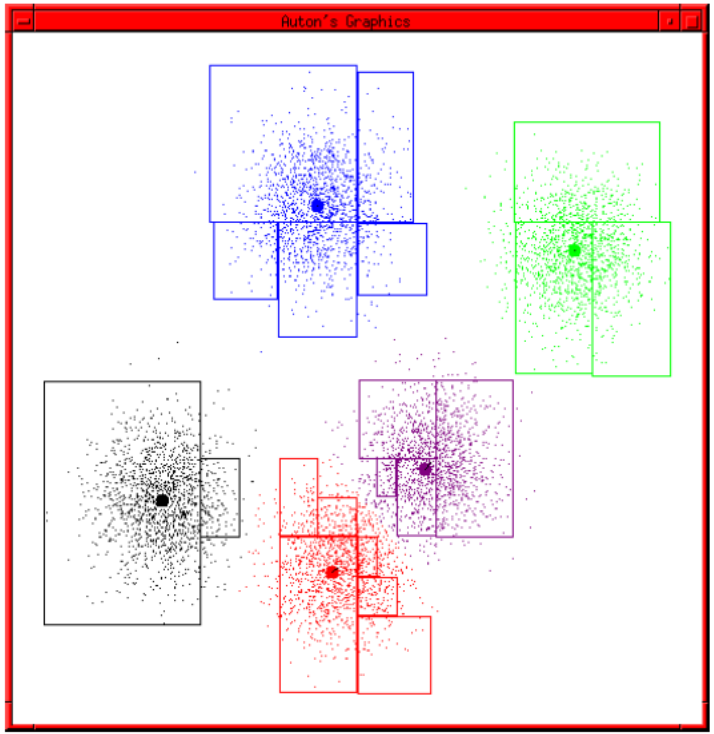
K-means algorithm continues (9)

Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm terminates

Image courtesy of Andrew Moore, Carnegie Mellon U.



Properties of K-means

- Optimality?
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place μ_1 on top of a randomly chosen datapoint.
    - Place μ_2 on top of datapoint that is furthest from μ_1.
    - Place μ_3 on top of datapoint that is furthest from both μ_1 and μ_2.

Properties of K-means

- Optimality?
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place μ_1 on top of a randomly chosen datapoint.
    - Place μ_2 on top of datapoint that is furthest from μ_1.
    - Place μ_3 on top of datapoint that is furthest from both μ_1 and μ_2.

- Complexity?

Properties of K-means

- Optimality?
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place μ_1 on top of a randomly chosen datapoint.
    - Place μ_2 on top of datapoint that is furthest from μ_1.
    - Place μ_3 on top of datapoint that is furthest from both μ_1 and μ_2.

- Complexity? $O(knm)$, where k=#centers, n=#datapoints, m=dimensionality of data

Properties of K-means

- Optimality?
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place μ_1 on top of a randomly chosen datapoint.
    - Place μ_2 on top of datapoint that is furthest from μ_1.
    - Place μ_3 on top of datapoint that is furthest from both μ_1 and μ_2.

- Complexity? $O(knm)$, where k=#centers, n=#datapoints, m=dimensionality of data
- K-means is an instance of a family of learning algorithms called "Expectation-Maximization" (aka EM).

A simple algorithm: K-means clustering

- Objective: Cluster n instances into K distinct classes.
- Preliminaries:
  - Step 1: Pick the desired number of clusters, K.
  - Step 2: Assume a parametric distribution for each class (e.g. Normal).
  - Step 3: Randomly estimate the parameters of the K distributions.
- Iterate, until convergence:
  - Step 4: Assign instances to the most likely classes based on the current parametric distributions.

    Maximization step
  - Step 5: Estimate the parametric distribution of each class based on the latest assignment.

    Expectation step

Expectation Maximization (more generally)

- Iterative method for learning the maximum likelihood estimate of a probabilistic model, when the model contains unobservable variables.

Expectation Maximization (more generally)

- Iterative method for learning the maximum likelihood estimate of a probabilistic model, when the model contains unobservable variables.

- Main idea:
  - If we had sufficient statistics for the data (e.g. counts of possible values), we could easily maximize the likelihood.
  - With missing data, we "fantasize" how the data should look based on the current parameter setting. I.e., compute expected sufficient statistics.
  - Then we maximize parameter setting, based on these statistics.

Expectation Maximization (more generally)

- Start with some initial parameter setting.

- Repeat (as long as desired):
  - Expectation (E) step: Complete the data by assigning "values" to the missing items.
  - Maximization (M) step: Compute the maximum likelihood parameter setting based on the completed data.

- Once the data is completed (E-step), computing the log-likelihood and new parameters (M-step) is easy! This is what we did for K-means.

Expectation Maximization: Properties

- Likelihood function is guaranteed to improve (or stay the same) with each iteration.
- Iterations can stop when no more improvements are achieved.
- Convergence to a local optimum of the likelihood function.
- Re-starts with different initial parameters are often necessary.
- Time complexity (per iteration) depends on model structure.

EM is very useful in practice!

Aside: Hierarchical clustering

- K-means generates a "flat" cluster structure. However, it is also possible to learn "hierarchical" clusters.

Aside: Hierarchical clustering

- K-means generates a "flat" cluster structure. However, it is also possible to learn "hierarchical" clusters.

- Two general approaches:
  - Top-down (agglomerative): Recursively merge a pair of clusters.
  - Bottom-up (divisive): Recursively split the existing clusters.

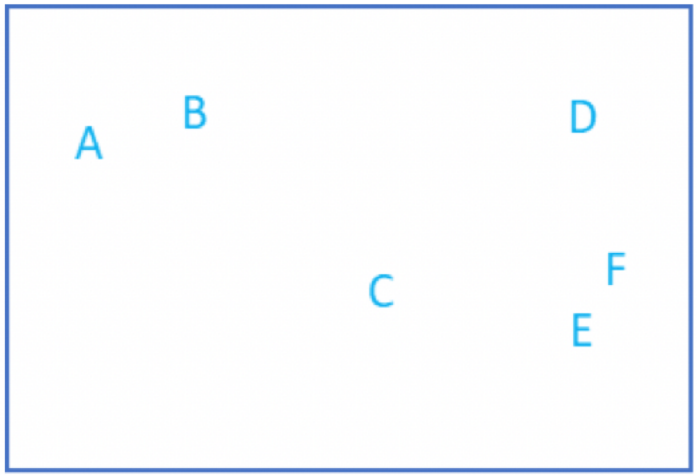
Aside: Hierarchical clustering

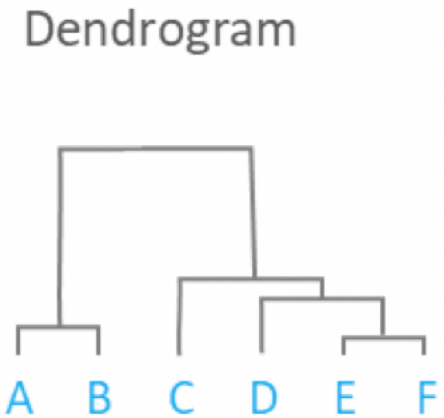
- K-means generates a "flat" cluster structure. However, it is also possible to learn "hierarchical" clusters.

- Two general approaches:
  - Top-down (agglomerative): Recursively merge a pair of clusters.
  - Bottom-up (divisive): Recursively split the existing clusters.

- Use dissimilarity measure to recursively split/merge pairs:
  - Measure pairwise distance between any points in the 2 clusters.
    - E.g. Euclidean distance, Manhattan distance.
  - Measure distance over entire clusters using linkage criterion.
    - E.g. Min/Max/Mean over pairs of points.

Aside: Hierarchical clustering





Anomaly detection

Anomaly detection

- K-means (and other discriminative approaches) tend to be ineffective when one class/cluster is much more rare than the other.

Anomaly detection

- K-means (and other discriminative approaches) tend to be ineffective when one class/cluster is much more rare than the other.

- A simple **generative** approach:
  - Fit a model, $p(x)$ using the input data.
  - Set a decision threshold ε and predict $Y = \{1 \text{ if } p(x) > \varepsilon, 0 \text{ otherwise}\}$.
  - Use a validation set to measure performance (can use cross-validation to set ε).

Anomaly detection vs Supervised learning

Anomaly detection

- Small number of positive examples (e.g. <10).
- Large number of negative examples (e.g. >100).

Supervised learning

- Similar number of positive and negative examples.

Anomaly detection vs Supervised learning

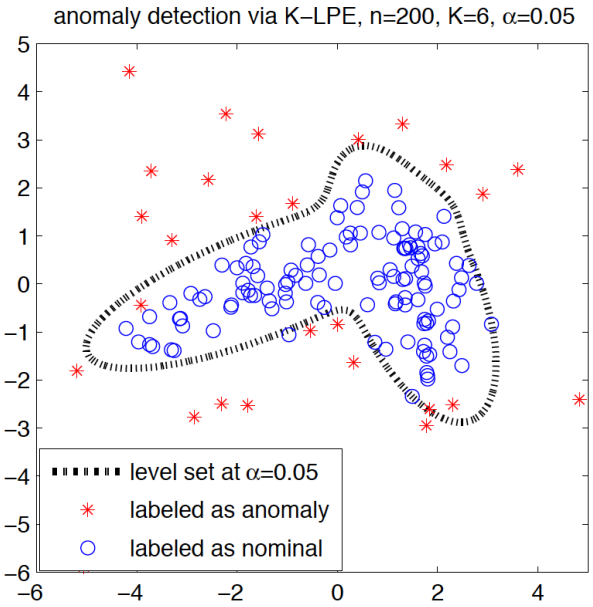
Anomaly detection

- Small number of positive examples (e.g. <10).
- Large number of negative examples (e.g. >100).
- Many different "types" of anomalies, so don't want to fit a model for the positive class.

Supervised learning

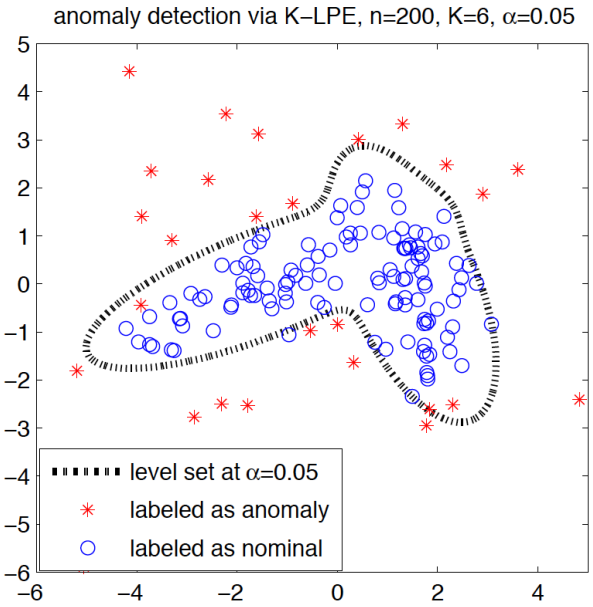
- Similar number of positive and negative examples.
- More homogeneity within classes, or enough data to sufficiently characterize each classes.

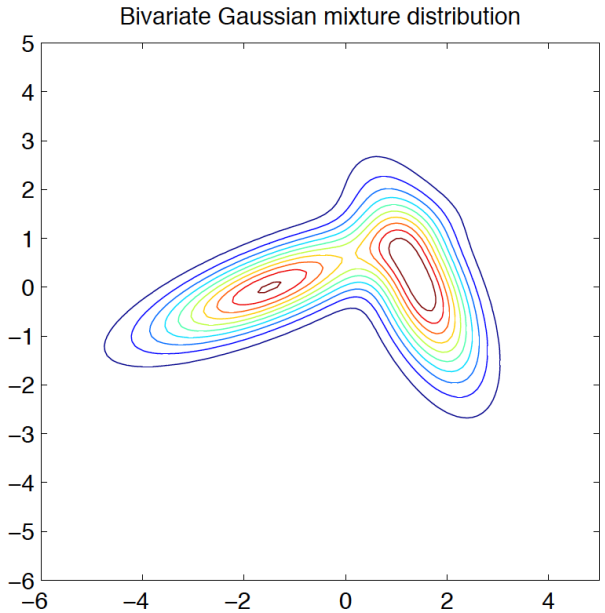
A simple example



From: *M. Zhao and V. Saligrama, "Anomaly Detection with Score functions based on Nearest Neighbor Graphs", Neural Information Processing Systems (NIPS) Conference, 2009*

A simple example





From: *M. Zhao and V. Saligrama, "Anomaly Detection with Score functions based on Nearest Neighbor Graphs", Neural Information Processing Systems (NIPS) Conference, 2009*

Gaussian Mixture Model

- Idea: Fit data with a combination of Gaussian distributions.

- What defines a set of Gaussians?

Gaussian Mixture Model

- **Idea:** Fit data with a combination of Gaussian distributions.

- Write $p(x)$ as a linear combination of Gaussians:

$$p(x) = \sum_{k=1:K} p(z_k)\, p(x \mid z_k)$$

where $p(z_k)$ is the probability of the k^{th} mixture component
and $p(x \mid z_k) = N(x \mid \mu_k, \sigma_k^2)$ is the prob. of x for the k^{th} mixture component.

- Estimate parameters $p(z_k)$, μ_k, σ_k^2, using Expectation-Maximization approach.

Gaussian Mixture Model

- **Idea:** Fit data with a combination of Gaussian distributions.

- Write $p(x)$ as a linear combination of Gaussians:

$$p(x) = \sum_{k=1:K} p(z_k)\, p(x \mid z_k)$$

where $p(z_k)$ is the probability of the k^{th} mixture component
and $p(x \mid z_k) = N(x \mid \mu_k, \sigma_k^2)$ is the prob. of x for the k^{th} mixture component.

- Estimate parameters $p(z_k)$, μ_k, σ_k^2, using Expectation-Maximization approach.

Generative modeling beyond anomalies

- Generative model $p(x)$ is not only useful for anomaly detection.

- Can be used to:
  - Provide priors in generative classification models.
  - Generate artificial samples.
  - Provide likelihood model for reinforcement learning agents.
  - ….

- Gaussian mixture models are very common/standard, but we will cover recent advancements in deep generative models after the midterm.

Practical issues

- In general, both clustering and anomaly detection are **very sensitive to the feature space!** Feature selection/construction is critical…

Practical issues

- In general, both clustering and anomaly detection are **very sensitive to the feature space!** Feature selection/construction is critical…

- But how do we do good feature selection in the unsupervised setting?
  - Can't use cross-validation…
  - Can't use mutual information or correlation with the target label….

Practical issues

- In general, both clustering and anomaly detection are **very sensitive to the feature space!** Feature selection/construction is critical…

- But how do we do good feature selection in the unsupervised setting?
  - Can't use cross-validation…
  - Can't use mutual information or correlation with the target label….

- Idea: Use unsupervised (deep) learning to infer high-quality low-dimensional representations (next lecture). **Dimensionality reduction.**