THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Applied Machine Intelligence and Reinforcement Learning

Professor Hamza F. Alsarhan

SEAS 8505

Lecture 5

July 13, 2024

Welcome to SEAS Online at George Washington University

Class will begin shortly

Audio: To eliminate background noise, please be sure your audio is muted. To speak, please click the hand icon at the bottom of your screen (Raise Hand). When instructor calls on you, click microphone icon to unmute. When you've finished speaking, *be sure to mute yourself again*.

Chat: Please type your questions in Chat.

Recordings: As part of the educational support for students, we provide downloadable recordings of each class session to be used exclusively by registered students in that particular class for their own private use. Releasing these recordings is strictly prohibited.

Agenda

- Dimensionality Reduction
- Principal Component Analysis (PCA)
- Unsupervised Learning
- Homework Overview

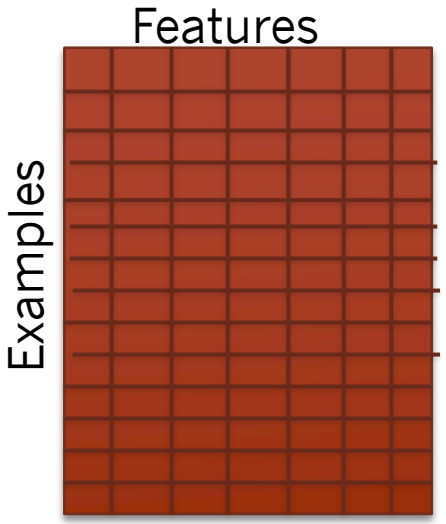
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Dimensionality Reduction

Professor Hamza F. Alsarhan
SEAS 8505

General Motivation for Dimensionality Reduction

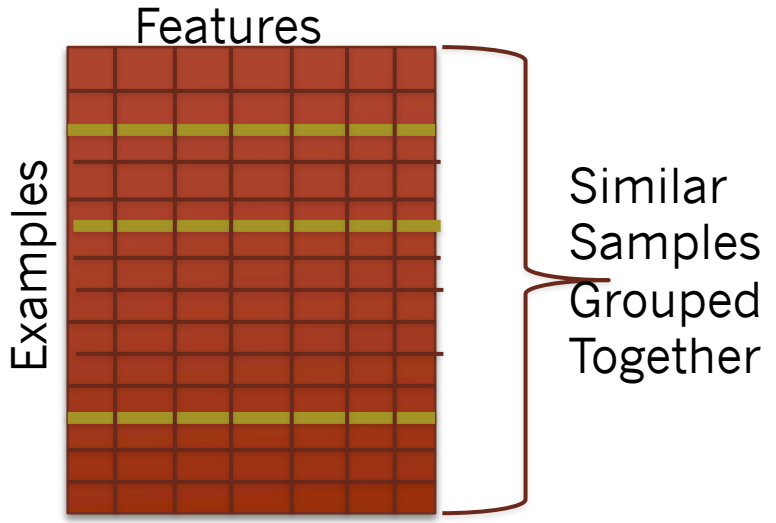
- Problem: Too much data!
- Solution: Reduce the amount of data
- Clustering: Reduce the number of examples
- Dimensionality Reduction: Reduce the number of features/attributes in the data



General Motivation for Dimensionality Reduction

- Problem: Too much data!
- Solution: Reduce the amount of data
- Clustering: Reduce the number of examples
- Dimensionality Reduction: Reduce the number of features/attributes in the data

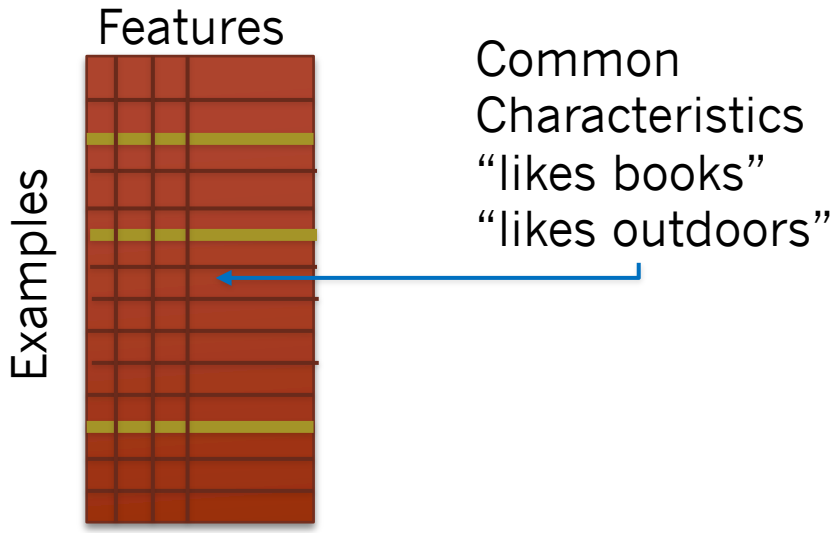
Market Segmentation –
What are some "natural" Groupings? →



General Motivation for Dimensionality Reduction

- Problem: Too much data!
- Solution: Reduce the amount of data
- Clustering: Reduce the number of examples
- Dimensionality Reduction: Reduce the number of features/attributes in the data

Reduce Number of Attributes. What info Is not needed? →



Dimensionality Reduction

- Given data points in d dimensions
- Convert them to data points in r < d dimensions
- With minimal loss of information
- Clustering:
  - Given a set of examples
  - Divide them into subsets of “similar” examples
  - So what does “similar” mean?
    - How do you measure similarity?
    - How do you know how well you did?

Why Reduce Dimensionality?

- Reduces time complexity: Less computation
- Reduces space complexity: Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

The Curse of Dimensionality

- Many Machine Learning problems involve thousands or even millions of features for each training instance
  - Makes training extremely slow
  - Makes it much harder to find a good solution
- This problem is often referred to as the curse of dimensionality
- It is often possible to reduce the number of features considerably and thereby turn an intractable problem into a tractable one

The Curse of Dimensionality

- Imagine instances described by 20 attributes but only 2 are relevant to target function

- Nearest Neighbor is easily misled with high dimensional x
- Easy problems in low dimension are hard in higher dimensions
- Lower dimension intuitions do not hold in higher dimensions

- **Examples:**
  - Normal Distribution
  - Uniform Distribution on a hypercube
  - Points on a hypergrid
  - Approximation of Sphere by Cube
  - Volume of hypersphere

In high dimensions KNN is led astray!

- **This is not a Phenomenon that is Unique to K-NN!**

Dealing with the Curse of Dimensionality

- Reduce dimensions

- Filter Approach
  - Pre-select features linearly
  - e.g. Information gain
  - Disadvantage: May get rid of an important feature
- Wrapper Approach
  - Forward selection
  - Backward elimination

Example – Think Back to MNIST

- The MNIST image pixels on the borders are almost always white
  - Removing these pixels from the training set would not result in a loss of much information (i.e. not needed for the classification task)

- In general, two neighboring pixels are often **highly correlated**
  - if you merge them into a single pixel (e.g., by taking the mean of the two pixel intensities) you will not **lose much information**

Compression

- Reducing dimensionality does cause some information loss (just like compressing an image to JPEG can degrade its quality), so even though it will speed up training, it may make your system perform slightly worse

- Introducing dimensionality reduction also makes your ML pipelines a bit more complex and thus harder to maintain
  - If training is already too slow, first try to train your system with the original data before considering using dimensionality reduction

Helps Visualization

- Dimensionality reduction is also extremely useful for data visualization

- Reducing the number of dimensions down to two (or three) makes it possible to plot a condensed view of a high-dimensional training set on a graph and often gain some important insights by visually detecting patterns, such as clusters

- Can also be used to communicate your conclusions to people who are not data scientists—in particular, decision makers who will use your results

Plenty of Space in High Dimensions

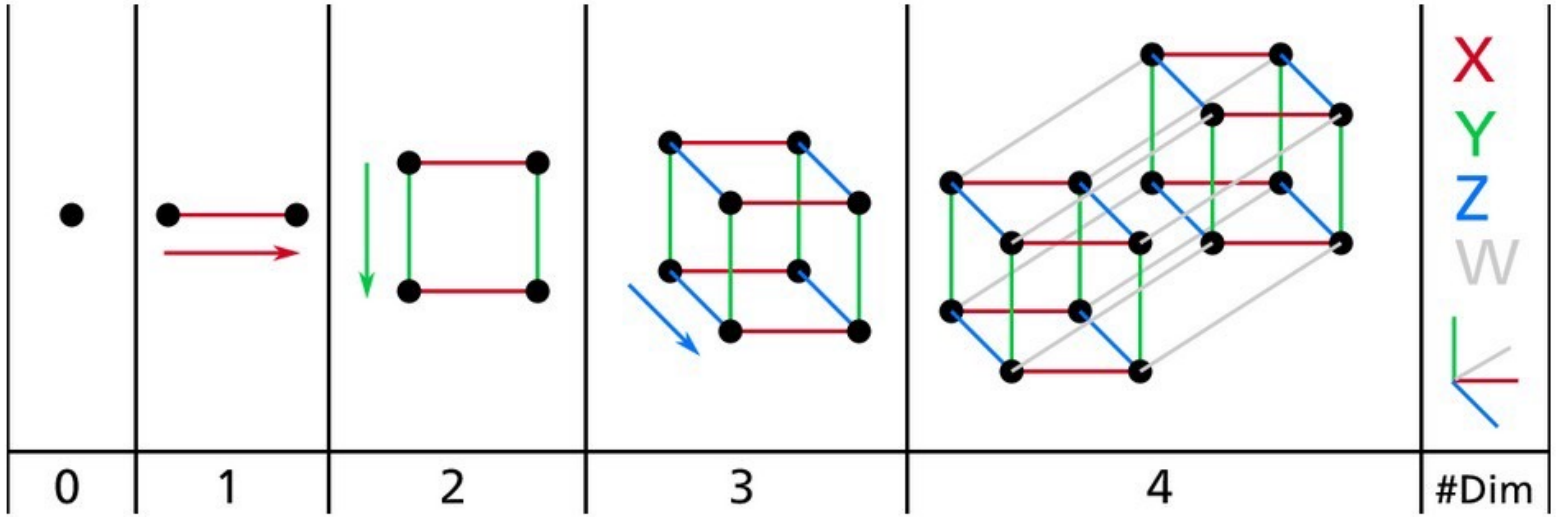


Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]

- High-dimensional datasets are at risk of being very sparse: most training instances are likely to be far away from each other
  - A new instance will likely be far away from any training instance, making predictions much less reliable than in lower dimensions
- The more dimensions the training set has, the greater the risk of overfitting it is

How to Deal with High-Dimensional Data

- You could increase the size of the training set to reach a sufficient density of training instances
  - Unfortunately, in practice, the number of training instances required to reach a given density grows exponentially with the number of dimensions.
  - Example: With just 100 features (significantly fewer than in the MNIST problem), you would need more training instances than atoms in the observable universe in order for training instances to be within 0.1 of each other on average, assuming they were spread out uniformly across all dimensions.
- Two main approaches to reduce dimensionality:
  - Projection
  - Manifold Learning

Projection

- In most real-world problems, training instances are not spread out uniformly across all dimensions.
- Many features are almost constant, while others are highly correlated
- As a result, all training instances lie within (or close to) a much lower-dimensional subspace of the high-dimensional space

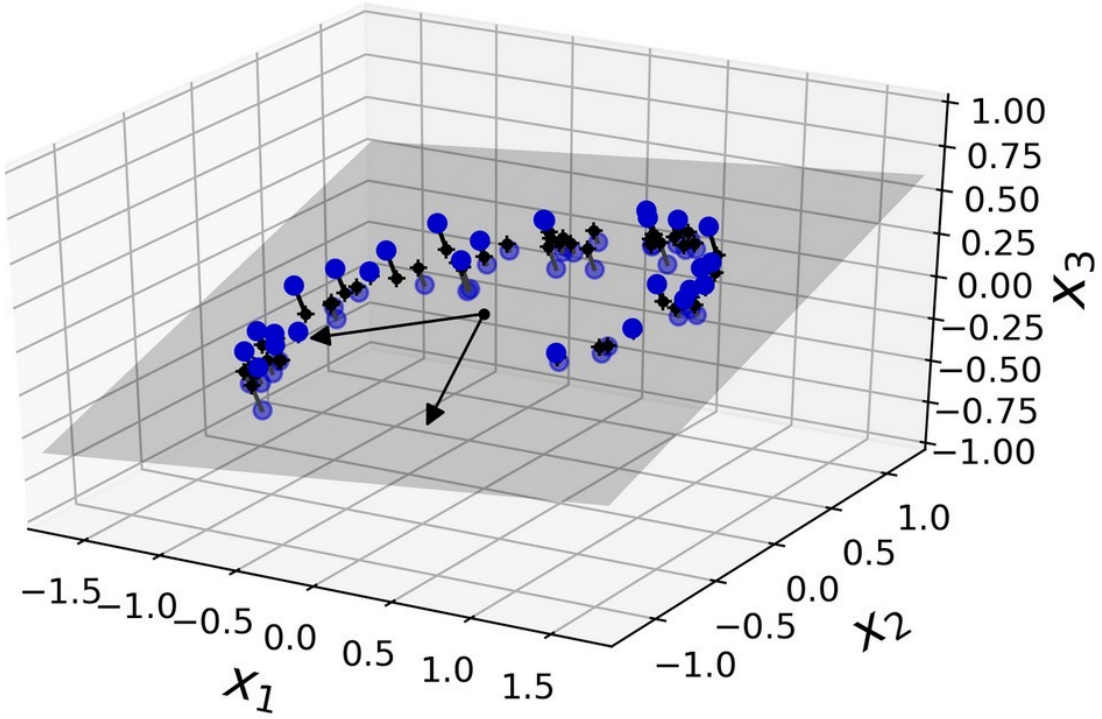


Figure 8-2. A 3D dataset lying close to a 2D subspace

Reducing the Dataset's Dimensionality (3D to 2D)

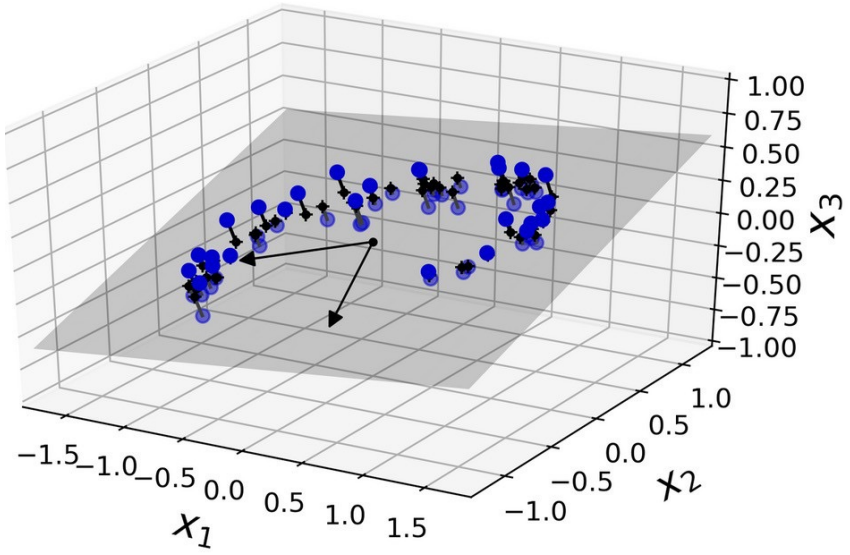


Figure 8-2. A 3D dataset lying close to a 2D subspace

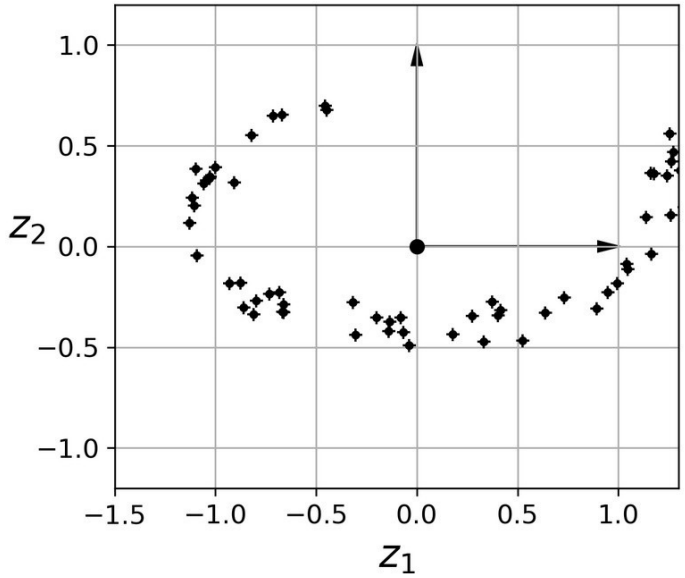


Figure 8-3. The new 2D dataset after projection

- In the left Figure, all training instances lie close to a plane
  - This is a lower-dimensional (2D) subspace of the high-dimensional (3D) space
- Project every training instance in Figure 8.2 perpendicularly onto that subspace to get the new 2D dataset shown in the Figure 8.3

2D Manifold

- Projection is not always the best approach to dimensionality reduction.
- In many cases the subspace may twist and turn, such as in the famous Swiss roll toy dataset.
- The Swiss roll is an example of a 2D manifold.
  - A 2D manifold is a 2D shape that can be bent and twisted in a higher-dimensional space.
  - This concept can be extended into higher dimensions.

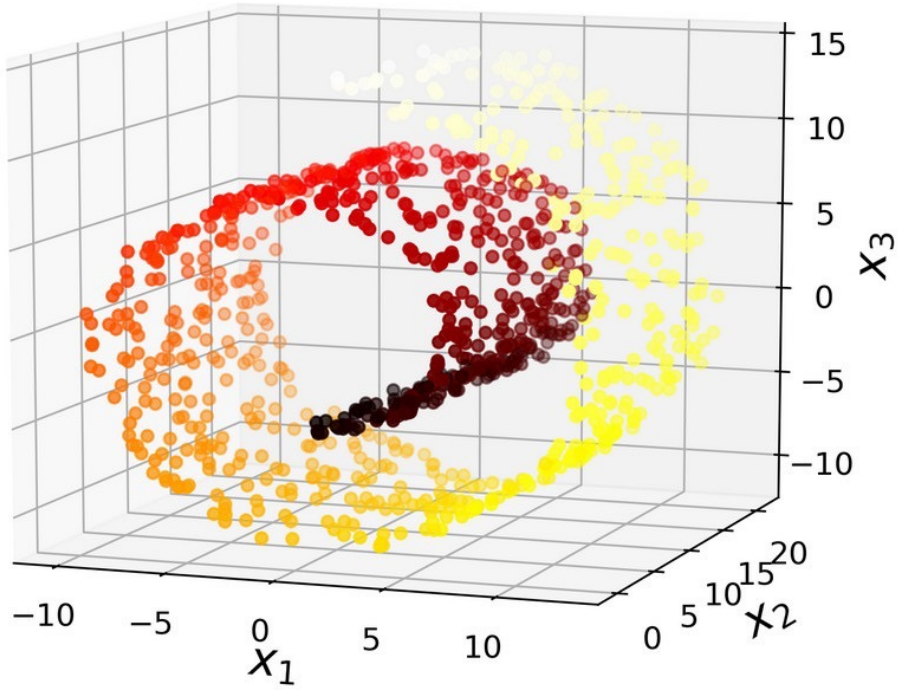


Figure 8-4. Swiss roll dataset

Unrolling a Manifold

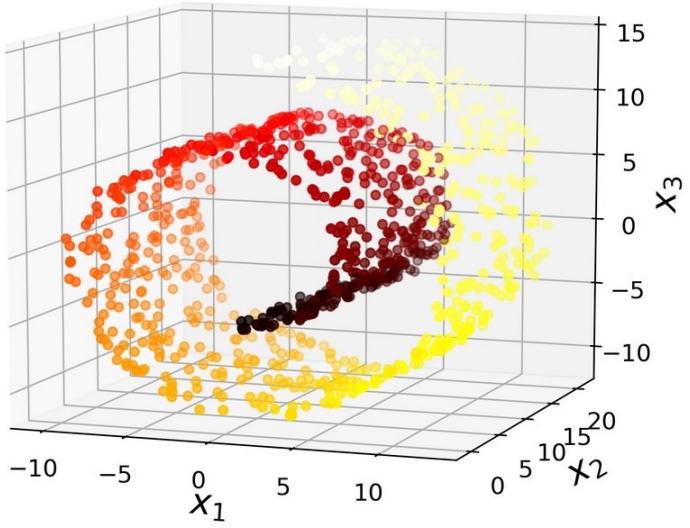


Figure 8-4. Swiss roll dataset

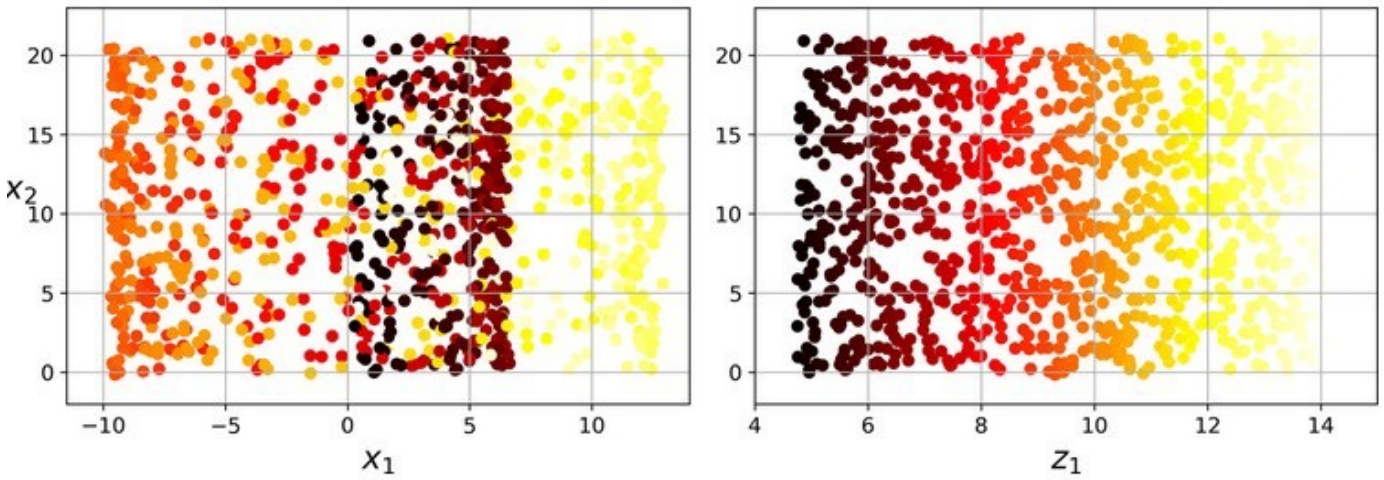


Figure 8-5. Squashing by projecting onto a plane (left) versus unrolling the Swiss roll (right)

- Simply projecting onto a plane (e.g., by dropping x3) would squash different layers of the Swiss roll together, as shown on the left side of Figure 8-5
- Ideally would like to unroll the Swiss roll to obtain the 2D dataset on the right side of Figure 8-5

Manifold Learning

- The manifold assumption, also called the manifold hypothesis, holds that most real- world high-dimensional datasets lie close to a much lower-dimensional manifold.
  - This assumption is very often empirically observed.
- Manifold learning involves dimensionality reduction algorithms which model the manifold on which the training instances lie
- The manifold assumption is often accompanied by another implicit assumption:
  - The task at hand (e.g., classification or regression) will be simpler if expressed in the lower-dimensional space of the manifold
  - This implicit assumption does not always hold

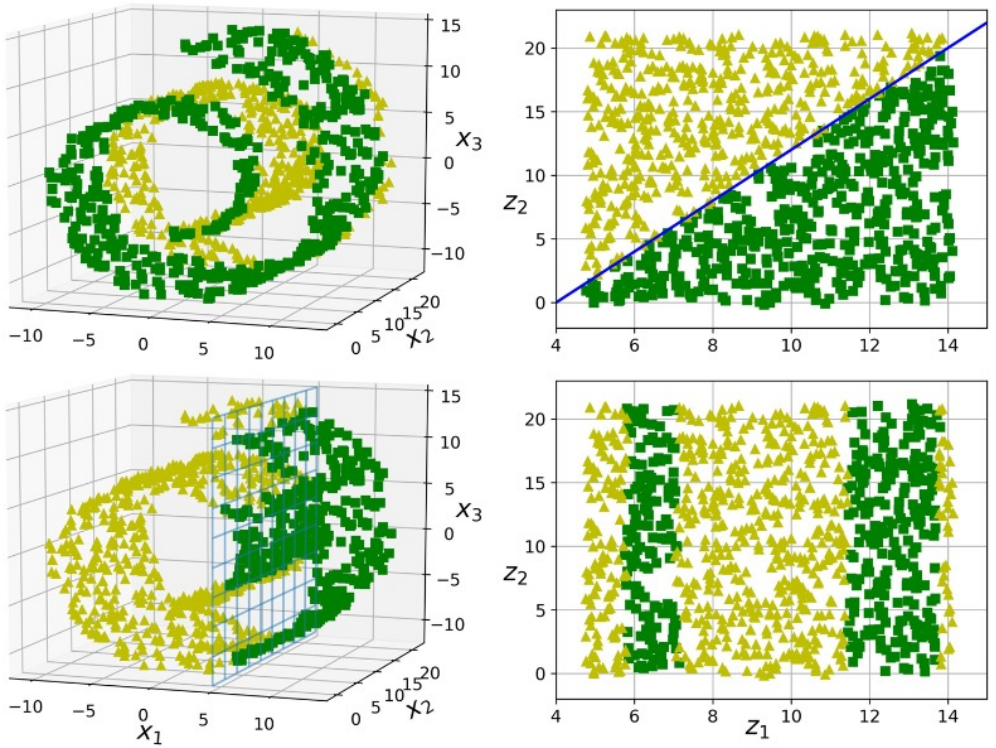


Figure 8-6. The decision boundary may not always be simpler with lower dimensions

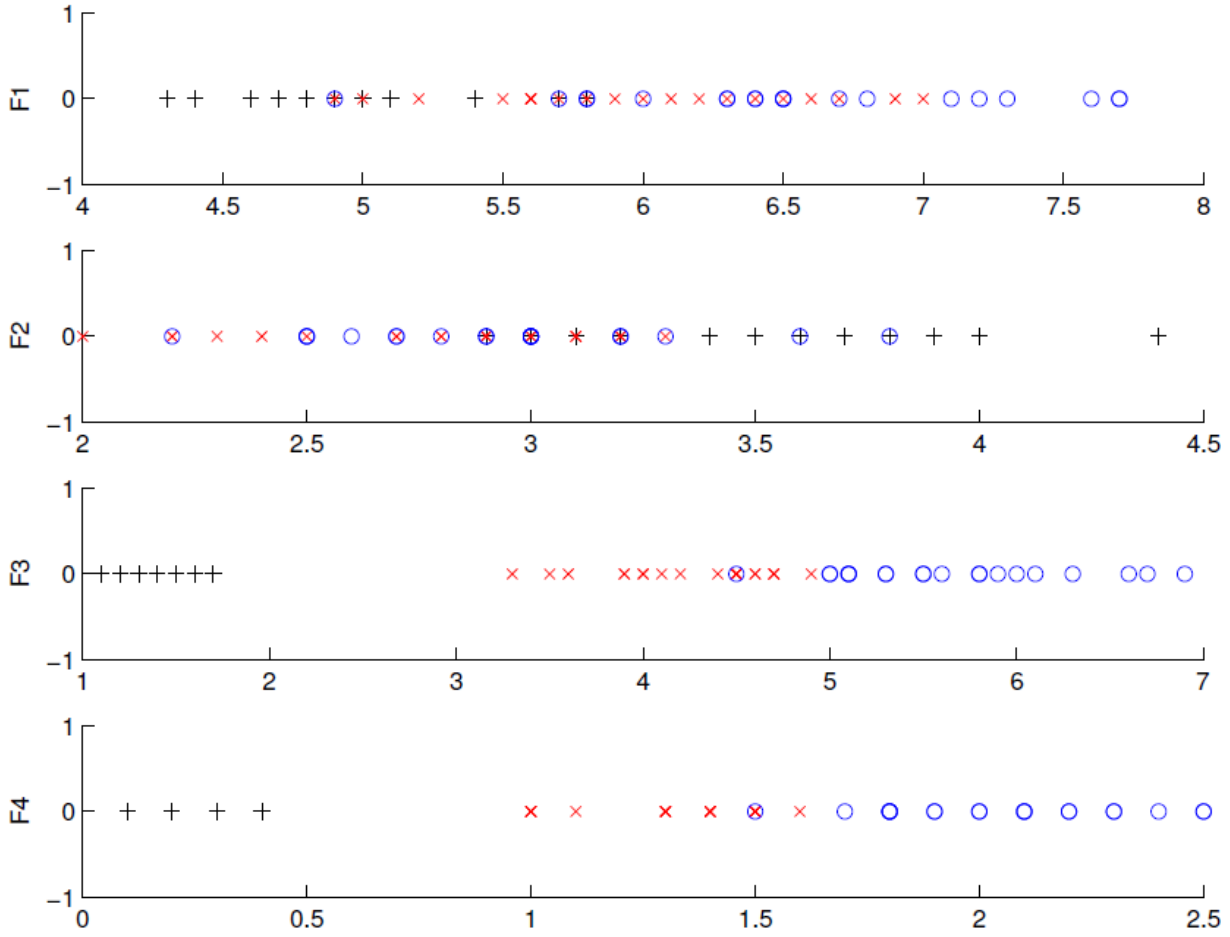
Relating Manifold Learning Back to MNIST

- Think about the MNIST dataset: all handwritten digit images have some similarities.
  - They are made of connected lines
  - The borders are white
  - More or less centered
- If you randomly generated images, only a ridiculously tiny fraction of them would look like handwritten digits.

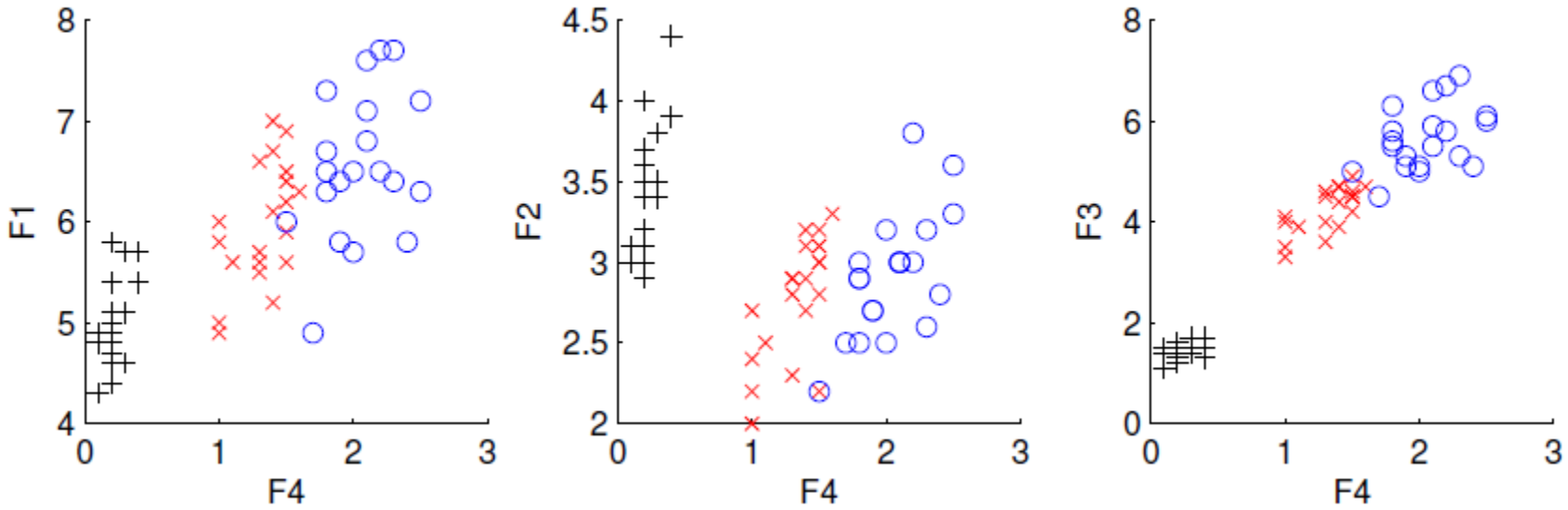
- The degrees of freedom available to you if you try to create a digit image are dramatically lower than the degrees of freedom you would have if you were allowed to generate any image you wanted.
  - These constraints tend to squeeze the dataset into a lower-dimensional manifold

Iris Data: Single Feature

- Plot of the training data for single features on Iris dataset; the three classes are shown with different symbols. It can be seen that F4 by itself allows quite good discrimination.



Iris Data: Add One More Feature to F4



- Plot of the training data with F4 as one feature, together with one of F1, F2, and F3. Using (F3, F4) leads to best separation.

Limitations of Feature Selection Techniques

- Good for small datasets in terms of computation and memory requirements
- Forward selection is faster than Backward Selection (why?)
- Does not work well with certain types of problems (e.g. image recognition)

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Principal Component Analysis (PCA)

Professor Hamza F. Alsarhan
SEAS 8505

Overview - Principal Component Analysis

- PCA is a method for dimensionality reduction
- Exploratory technique that can be used to reduce the dimensionality of the dataset to 2D or 3D
- Can be used to:
  - Reduce number of dimensions in data
  - Find patterns in high-dimensional data
  - Visualize data of high dimensionality
- Example applications:
  - Facial recognition
  - Image compression
  - Gene expression analysis

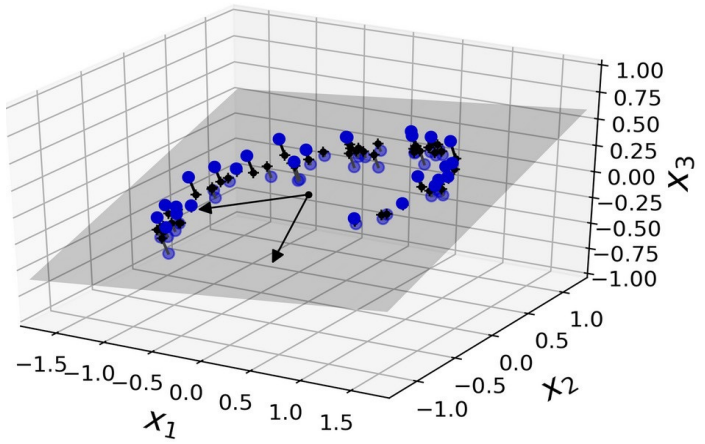
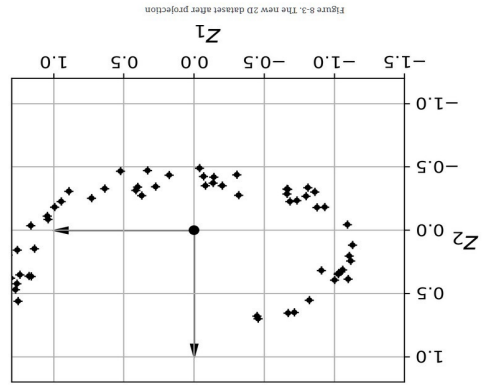


Figure 8-2. A 3D dataset lying close to a 2D subspace



Principal Components Analysis Overview

- Unsupervised method for identifying the important directions in a dataset
- We can then rotate the data into the (reduced) coordinate system that is given by those directions
- PCA is a method for dimensionality reduction
- **Algorithm:**
  - Find direction (axis) of greatest variance
  - Find direction of greatest variance that is perpendicular to previous direction and repeat
- Implementation: find eigenvectors of the covariance matrix of the data
  - Eigenvectors (sorted by eigenvalues) are the directions that has the highest variance

Principal Component Analysis

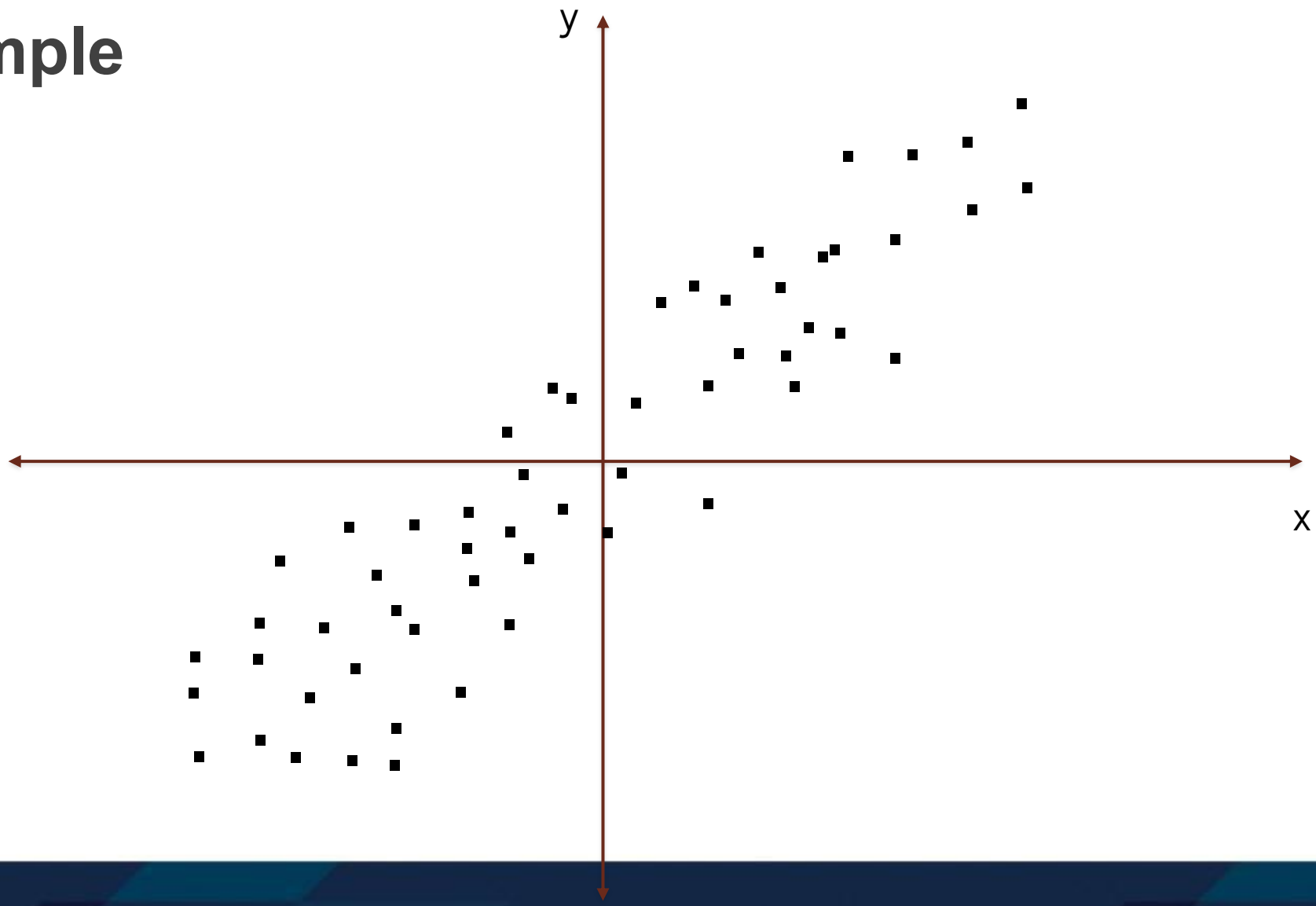
Goal: Find r-dim projection that best preserves the variance

1. Compute Mean of Vector μ and covariance matrix Σ of original points
2. Compute Eigenvectors and Eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

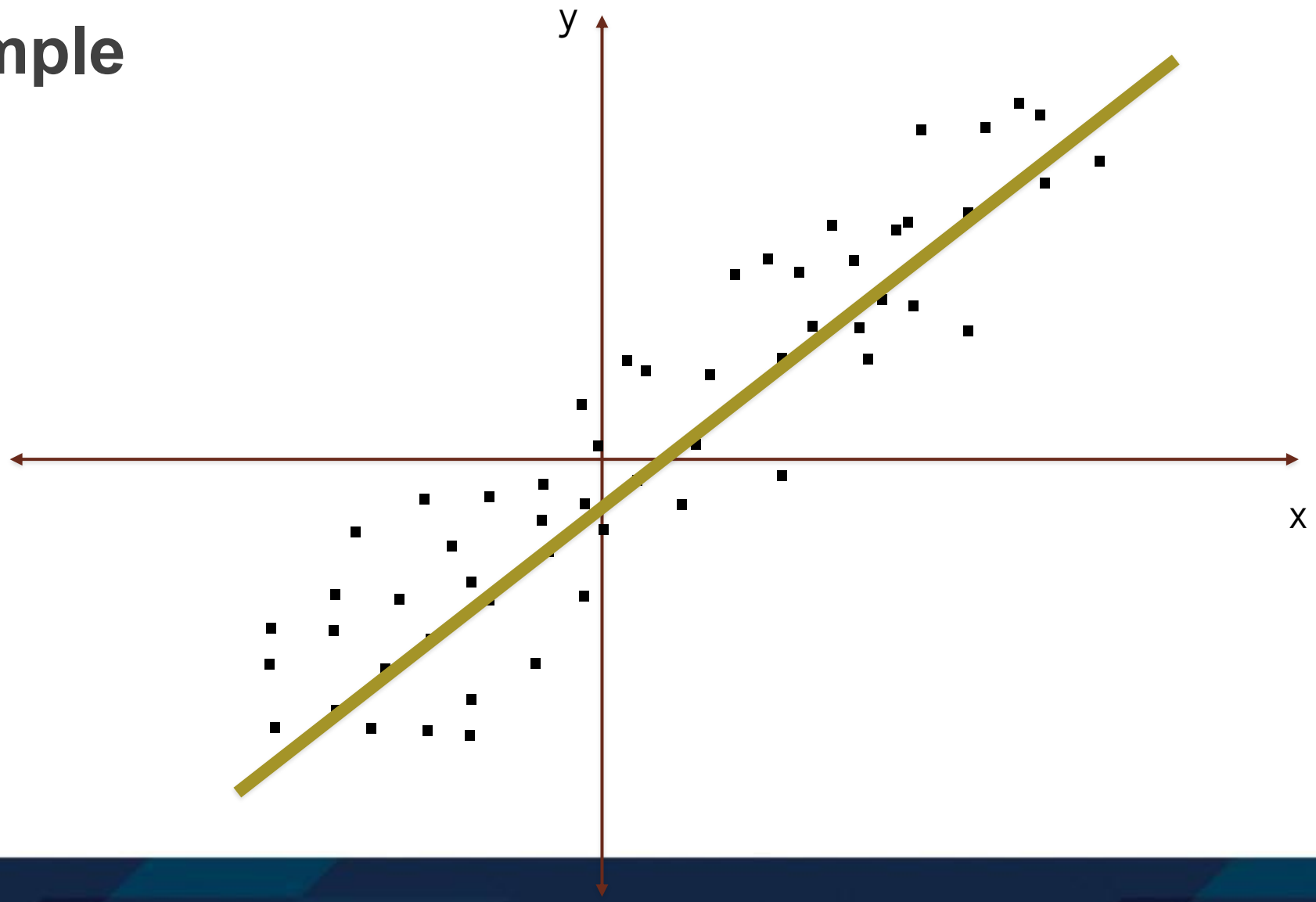
$$y = A(x - \mu)$$

Where y is the new point and x is the old one and the rows of A are the eigenvectors

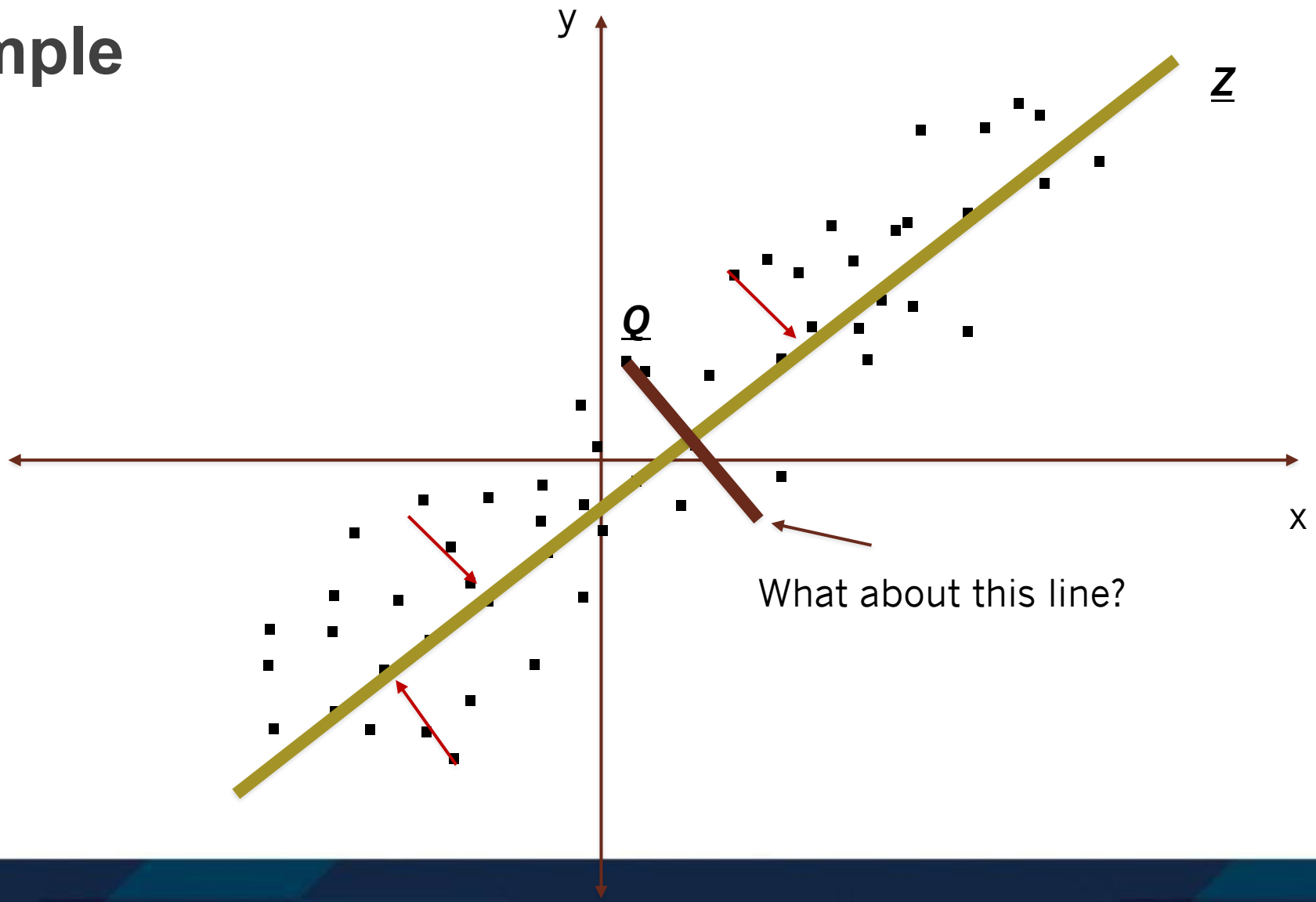
Example



Example



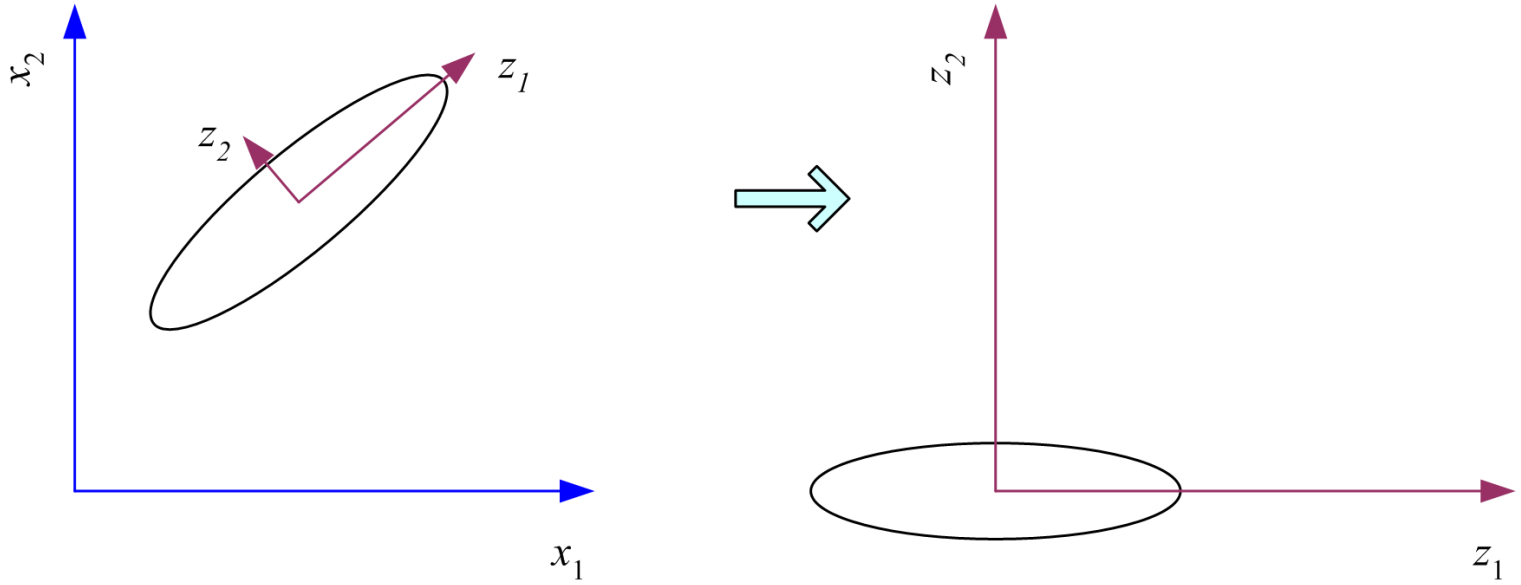
Example



What PCA Does

- $\mathbf{z} = \mathbf{W}^T(\mathbf{x} - \mathrm{m})$
- where the columns of **W** are the eigenvectors of ∑ (covariance matrix) and **m** is the sample mean
- Centers the data at the origin and rotates the axes

Principal component analysis centers the sample and then rotates the axes to line up with the directions of the highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one.



Preserving the Variance

- PCA identifies the axis that accounts for the largest amount of variance in the training set
- It also finds a second axis, orthogonal to the first one, that accounts for the largest amount of **remaining** variance
- If it were a higher-dimensional dataset, PCA would also find a third axis, orthogonal to both previous axes, and a fourth, a fifth, and so on—as many axes as the number of dimensions in the dataset.
  - The ith axis is called the ith principal component (PC) of the data

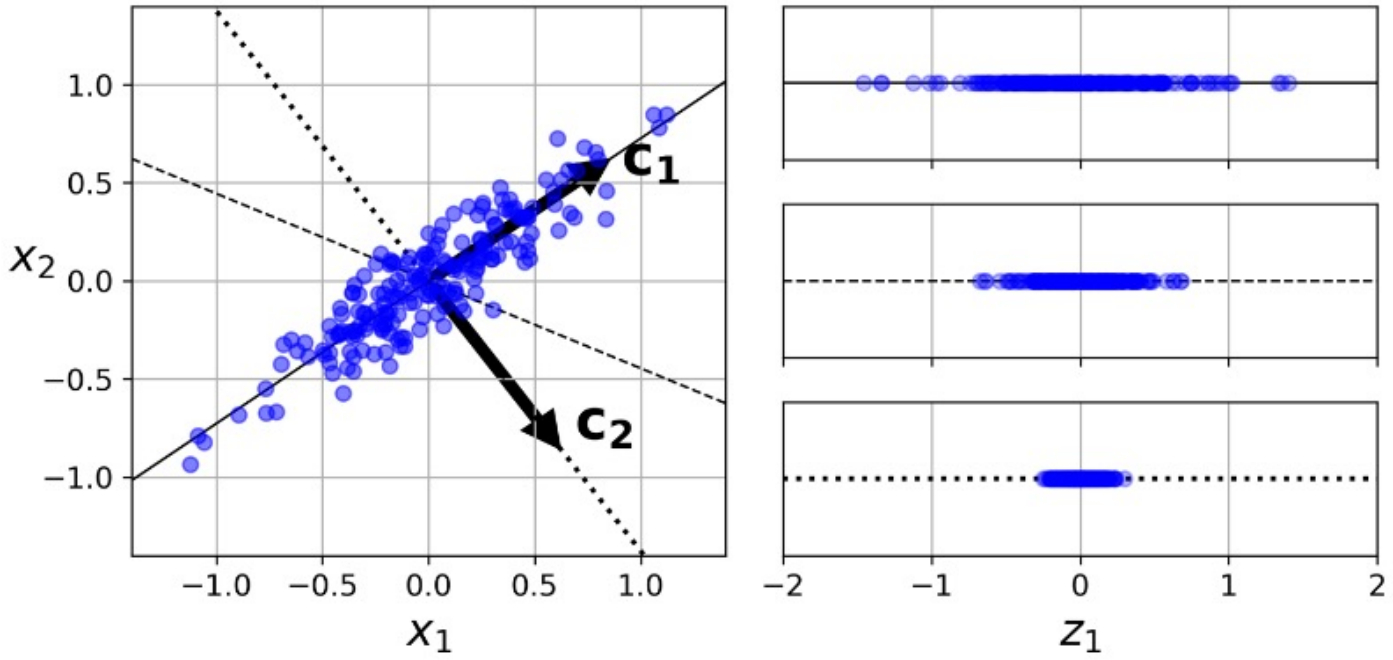
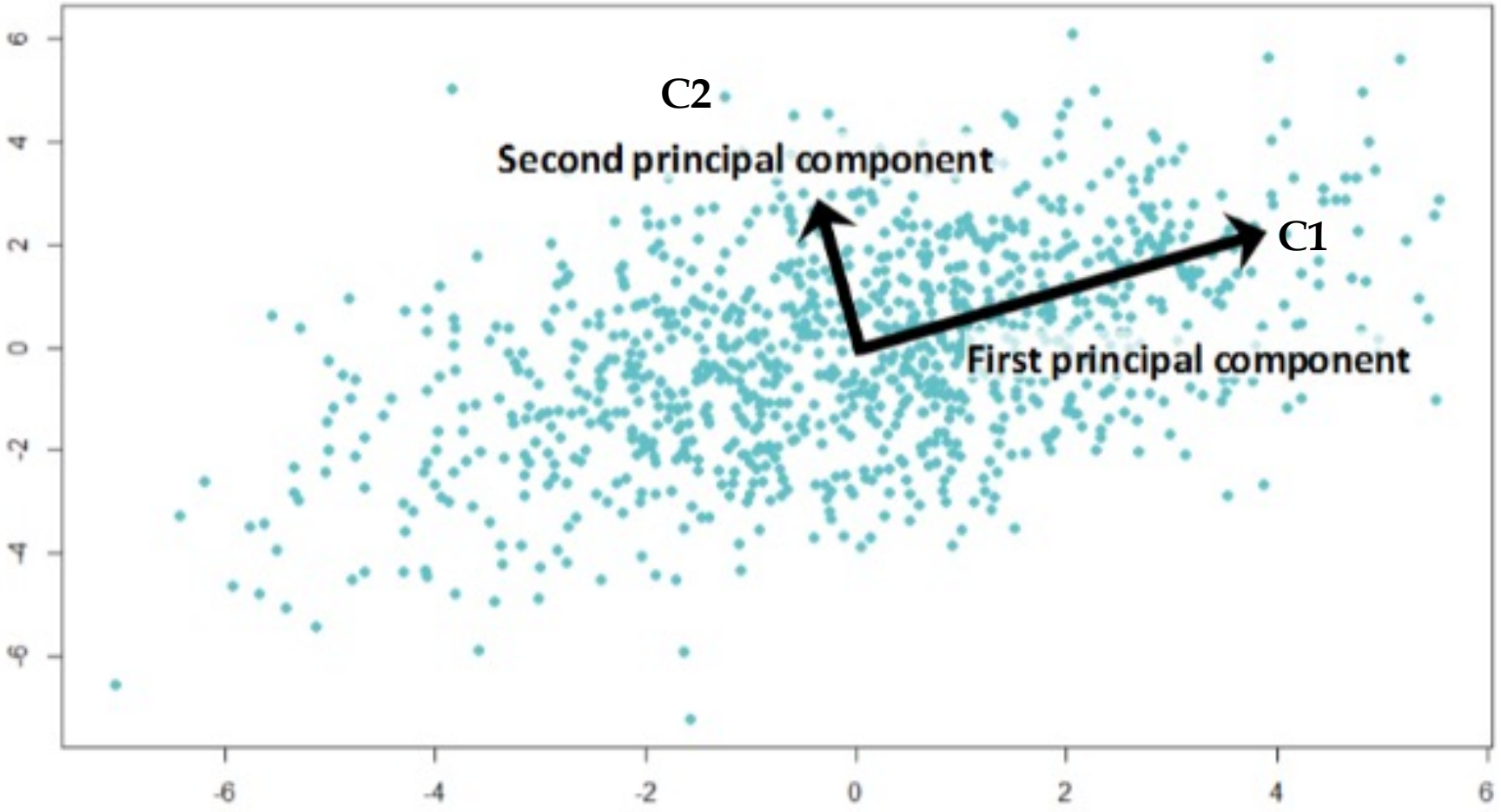


Figure 8-7. Selecting the subspace to project on

- **NOTE:** Singular Value Decomposition (SVD) can decompose the training set matrix X into the matrix multiplication of three matrices which contain the unit vectors that define all the principal components that we are looking for



1. **Principal Component (C1)**
   The eigenvalue with the largest absolute value will indicate that the data has the largest variance along its eigenvector, the direction along which there is greatest variation

2. **Principal Component (C2)**
   the direction with maximum variation left in data, orthogonal to C1.

In general, only few directions manage to capture most of the variability in the data.

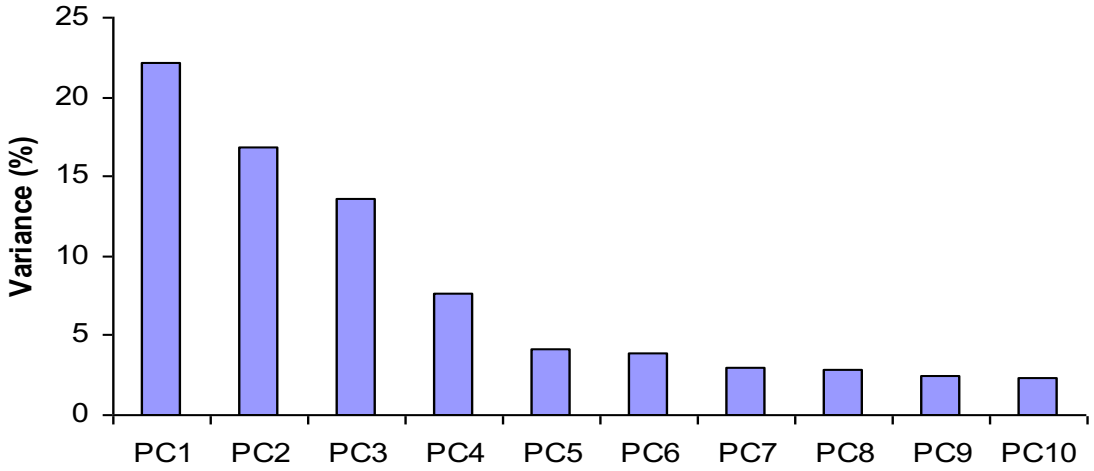
How to Choose k?

- Proportion of Variance (PoV) explained

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

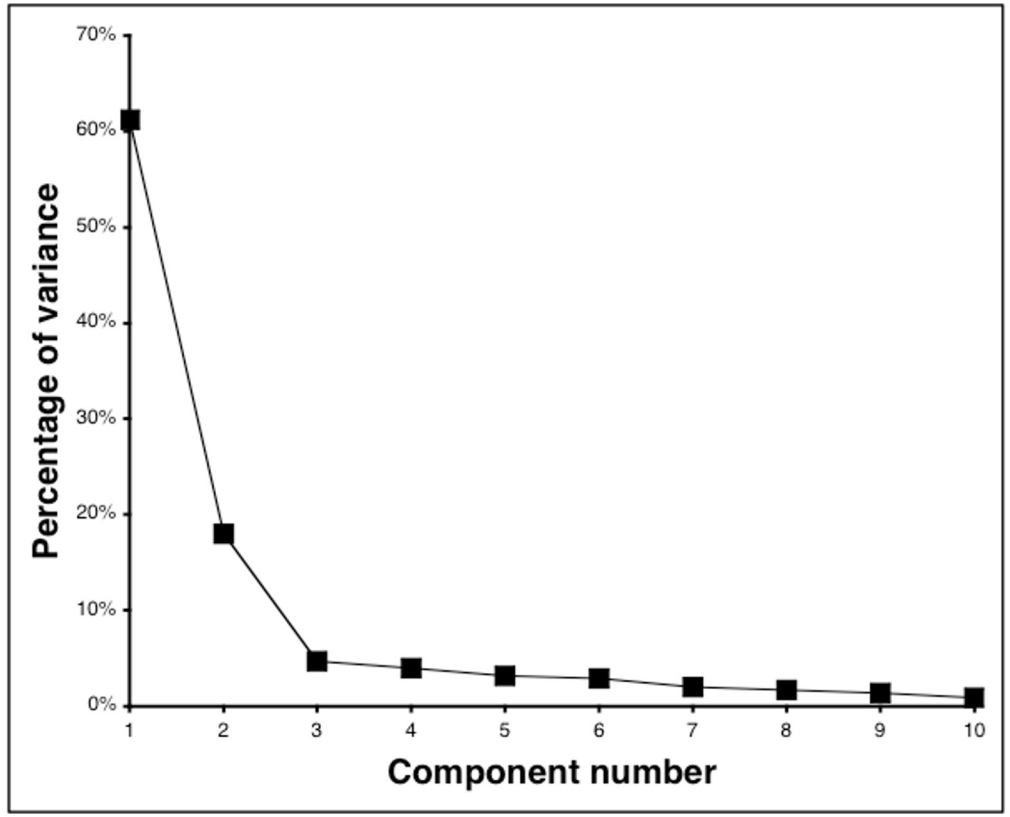
when λi are sorted in descending order

- Typically, stop at PoV>0.9
- Scree graph plots of PoV vs k, stop at "elbow"



Example: 10-Dimensional Data

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%



- Data is normally standardized or mean-centered for PCA

Projecting Down to d Dimensions

- Once you have identified all the principal components, you can reduce the dimensionality of the dataset down to d dimensions by projecting it onto the hyperplane defined by the first d principal components
  - This hyperplane ensures that the projection will preserve as much variance as possible
  - Choose the number of dimensions that add up to a sufficiently large portion of the variance (e.g., 95%)
- Can also plot the explained variance as a function of the number of dimensions
- There will usually be an elbow in the curve, where the explained variance stops growing fast

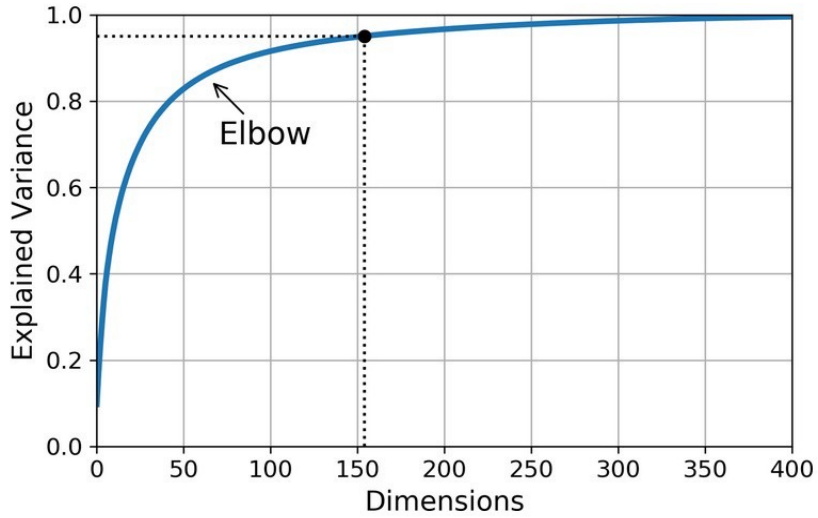


Figure 8-8. Explained variance as a function of the number of dimensions

PCA for Compression

- After dimensionality reduction, the training set takes up much less space.
- Applying PCA to the MNIST dataset while preserving 95% of its variance
  - Each instance will have just over 150 features, instead of the original 784 features.
  - Most of the variance is preserved and the dataset is now less than 20% of its original size!
  - This is a reasonable compression ratio, and this size reduction can speed up a classification algorithm tremendously

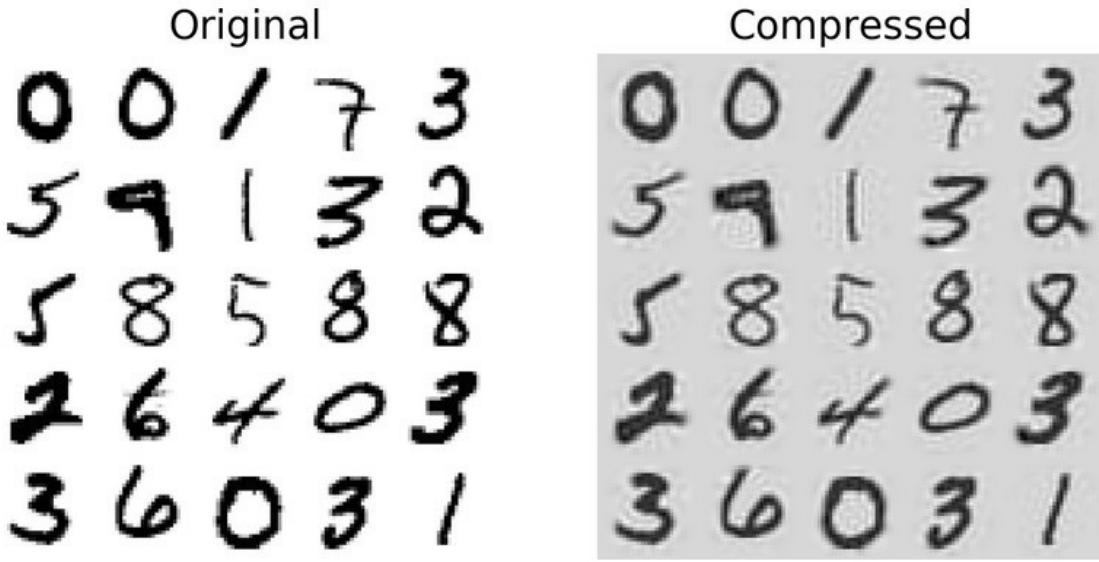


Figure 8-9. MNIST compression that preserves 95% of the variance

Kernel PCA

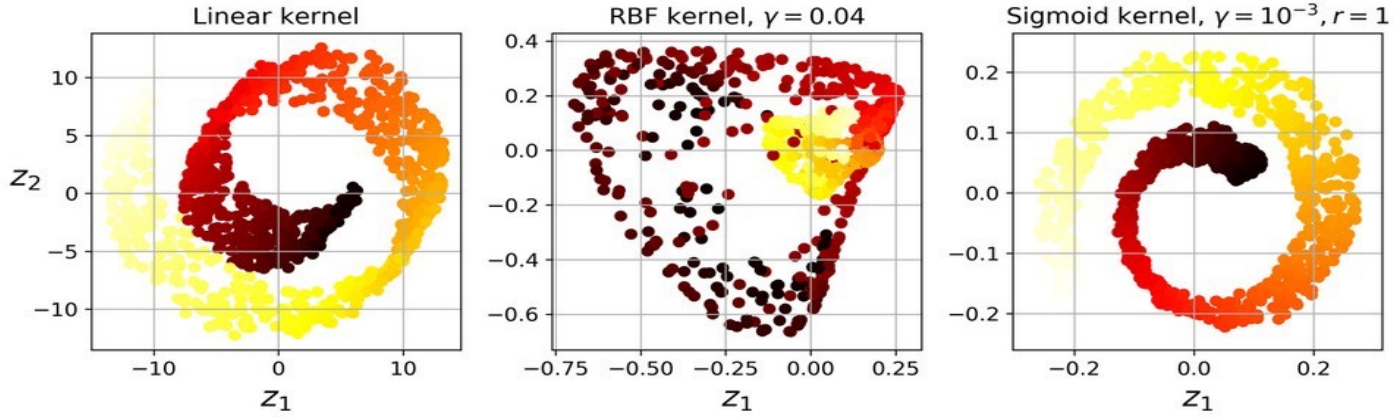


Figure 8-10. Swiss roll reduced to 2D using kPCA with various kernels

- The kernel trick is a mathematical technique that implicitly maps instances into a very high-dimensional space (called the feature space), enabling nonlinear classification and regression with Support Vector Machines (Chapter 5)
  - A linear decision boundary in the high-dimensional feature space corresponds to a complex nonlinear decision boundary in the original space
- Kernel PCA applies the same trick to PCA which makes it possible to perform complex nonlinear projections for dimensionality reduction
- It is often good at preserving clusters of instances after projection, or sometimes even unrolling datasets that lie close to a twisted manifold

Formal PCA & Observations

- Find a low-dimensional space such that when x is projected there, information loss is minimized.

- General observations about principal components
  - Summary variables
  - Linear combinations of the original variables
  - Uncorrelated with each other
  - Capture as much of the original variance as possible

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Unsupervised Learning

Professor Hamza F. Alsarhan
SEAS 8505

Why Unsupervised Learning

- Although most of the applications of Machine Learning today are based on supervised learning (and as a result, this is where most of the investments go to), the **vast majority of the available data is *unlabeled*** (i.e. we have the input features X, but we do not have the labels y)

- The computer scientist Yann LeCun famously said that “if intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake.”

Unsupervised Learning Tasks and Algorithms

- **Clustering**
  - The goal is to group similar instances together into *clusters*. Clustering is a great tool for data analysis, customer segmentation, recommender systems, search engines, image segmentation, semi-supervised learning, dimensionality reduction, and more.
- **Anomaly detection**
  - The objective is to learn what “normal” data looks like, and then use that to detect abnormal instances, such as defective items on a production line or a new trend in a time series.
- **Density estimation**
  - This is the task of estimating the *probability density function* (PDF) of the random process that generated the dataset. Density estimation is commonly used for anomaly detection: instances located in very low-density regions are likely to be anomalies. It is also useful for data analysis and visualization.

Clustering

- The task of identifying similar instances and assigning them to clusters, or groups of similar instances.
- Just like in classification, each instance gets assigned to a group. However, unlike classification, clustering is an unsupervised task.
- There is no universal definition of what a cluster is: it really depends on the context, and different algorithms will capture different kinds of clusters.
  - Some algorithms look for instances centered around a particular point, called a centroid.
  - Others look for continuous regions of densely packed instances: these clusters can take on any shape.
  - Some algorithms are hierarchical, looking for clusters of clusters.

Clustering is Used in a Wide Variety of Applications

- Customer segmentation
- Data analysis
- Dimensionality reduction
- Anomaly detection (also called outlier detection)
- Semi-supervised learning
- Search engines
- Image segmentation

Classification vs. Clustering

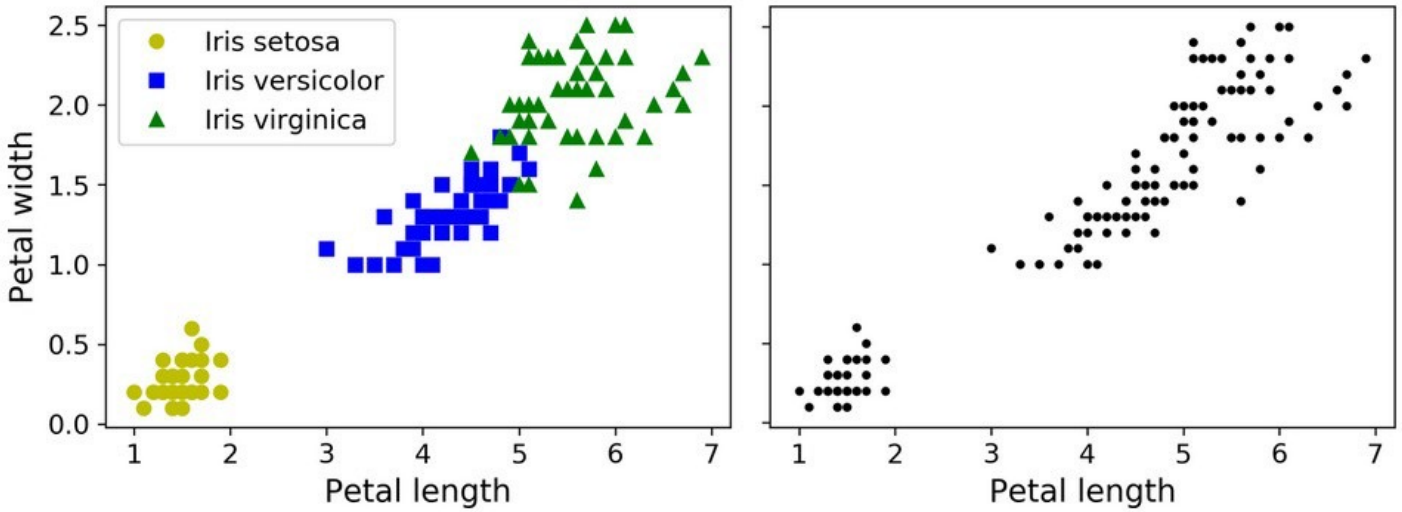


Figure 9-1. Classification (left) versus clustering (right)

- On the left is the iris dataset where each instance's species (i.e., its class) is represented with a different marker. It is a labeled dataset, for which classification algorithms such as Logistic Regression, SVMs, or Random Forest classifiers are well suited.
- On the right is the same dataset, but without the labels, so you cannot use a classification algorithm anymore.

Using Clustering for Image Segmentation

Figure 9-12. Image segmentation using K-Means with various numbers of color clusters

- Image segmentation is the task of partitioning an image into multiple segments.
- Semantic segmentation - all pixels that are part of the same object type get assigned to the same segment. (Focuses on class-level segmentation) – would identify all cars in an image as a class.
- Instance segmentation - all pixels that are part of the same individual object are assigned to the same segment. Would identify each car in an image separately as an object.
- Color segmentation - simply assign pixels to the same segment if they have a similar color.

Using Clustering for Preprocessing

- Clustering can be an efficient preprocessing step before a supervised learning algorithm.

- When using K-Means as a preprocessing step in a classification pipeline
  - Finding a good value for k is much simpler
  - There's no need to perform silhouette analysis or minimize the inertia
  - The best value of k is simply the one that results in the best classification performance during cross-validation

Thinking Back to MNIST

- Warning - For MNIST, since there are 10 different digits, it is tempting to set the number of clusters to 10. However, each digit can be written several different ways, so it is preferable to use a larger number of clusters, such as 50.

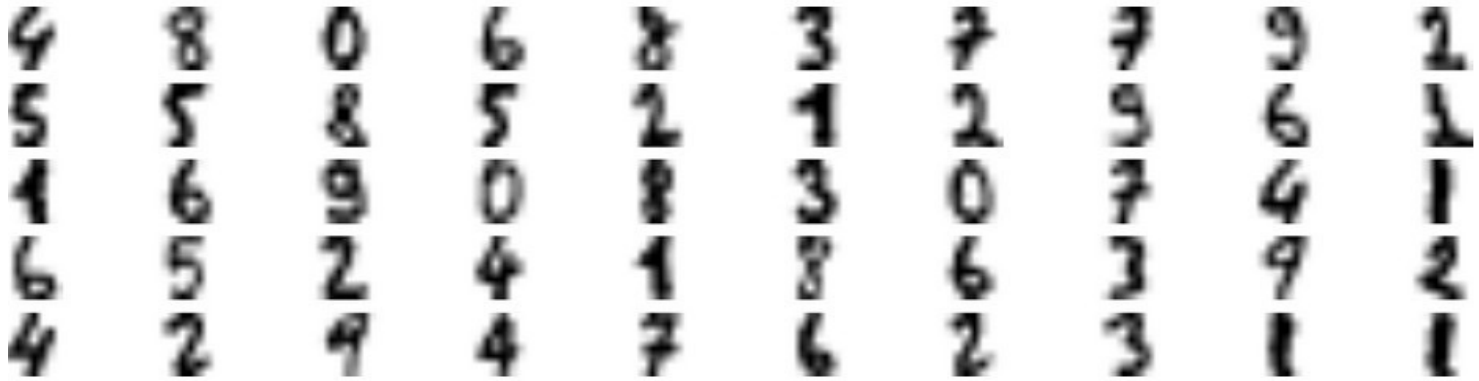


Figure 9-13. Fifty representative digit images (one per cluster)

Active Learning

- When a human expert interacts with the learning algorithm, providing labels for specific instances when the algorithm requests them.

- There are many different strategies for active learning, but one of the most common ones is called uncertainty sampling. Here is how it works:
  1. The model is trained on the labeled instances gathered so far, and this model is used to make predictions on all the unlabeled instances.
  2. The instances for which the model is most uncertain (i.e., when its estimated probability is lowest) are given to the expert to be labeled.
  3. You iterate this process until the performance improvement stops being worth the labeling effort.

Classes vs. Clusters

- Supervised: $X = \{\boldsymbol{x}^t, \boldsymbol{y}^t\}_t$
- Classes C_i $i=1,\ldots,K$

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x} \mid C_i) P(C_i)$$

where $p(\boldsymbol{x} \mid C_i) \sim N(\boldsymbol{\mu}_i, \sum_i)$

- $\Phi = \{P(C_i), \boldsymbol{\mu}_i, \sum_i\}_{i=1}^{K}$

- Unsupervised : $X = \{\boldsymbol{x}^t\}_t$
- Clusters G_i $i=1,\ldots,k$

$$p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid G_i) P(G_i)$$

- where $p(\boldsymbol{x} \mid G_i) \sim N(\boldsymbol{\mu}_i, \sum_i)$
- $\Phi = \{P(G_i), \boldsymbol{\mu}_i, \sum_i\}_{i=1}^{k}$

Labels $\boldsymbol{y}^t_i$?

Hierarchical Clustering

- Cluster based on similarities/distances
- Distance measure between instances $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$

  Minkowski (L_p) (Euclidean for $p = 2$)

$$d_m\left(\mathbf{x}^r, \mathbf{x}^s\right) = \left[\sum_{j=1}^{d}\left(x_j^r - x_j^s\right)^p\right]^{1/p}$$

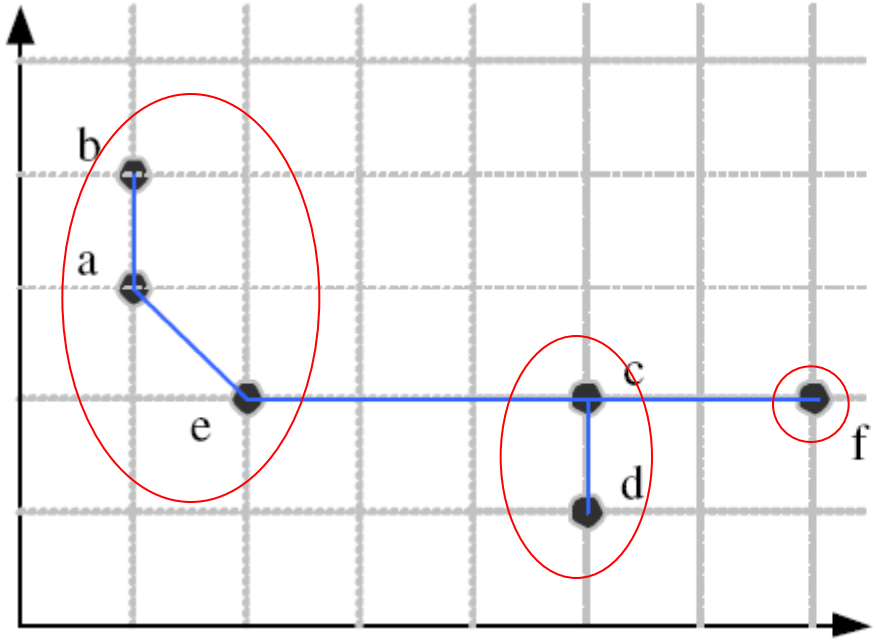
  City-block distance (easier to calculate):

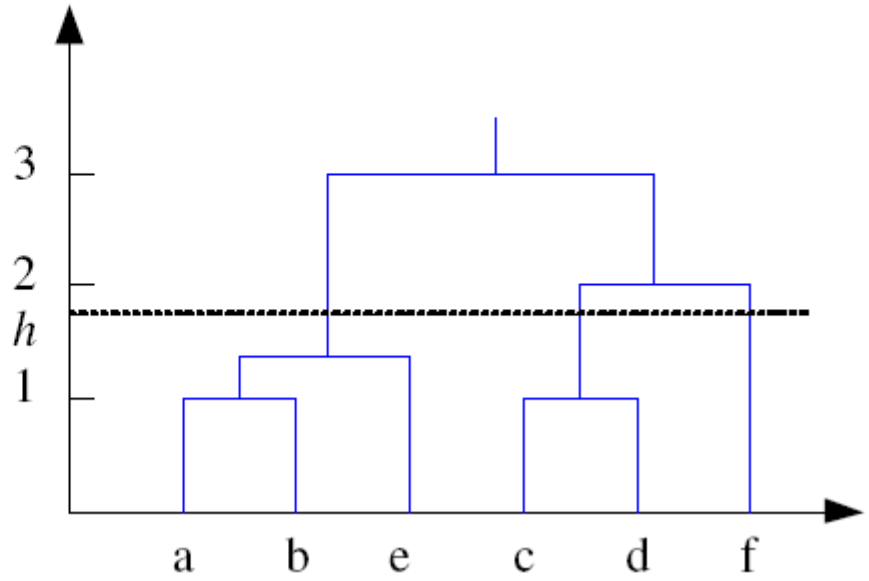
$$d_{cb}\left(\mathbf{x}^r, \mathbf{x}^s\right) = \sum_{j=1}^{d}\left|x_j^r - x_j^s\right|$$

Agglomerative Clustering

- Start with *N* groups each with one instance and merge two closest groups at each iteration
- Distance between two groups G_i and G_j:
  - Single-link: $$d(G_i, G_j) = \min_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$
  - Complete-link: $$d(G_i, G_j) = \max_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$
  - Average-link, centroid $$d(G_i, G_j) = \text{ave}_{\mathbf{x}^r \in G_i, \mathbf{x}^s \in G_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

Example: Single-Link Clustering



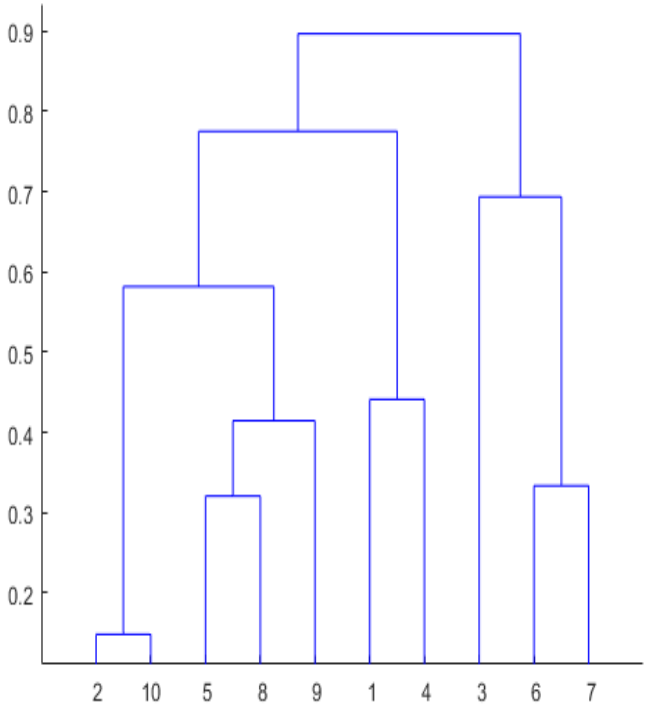


Dendrogram

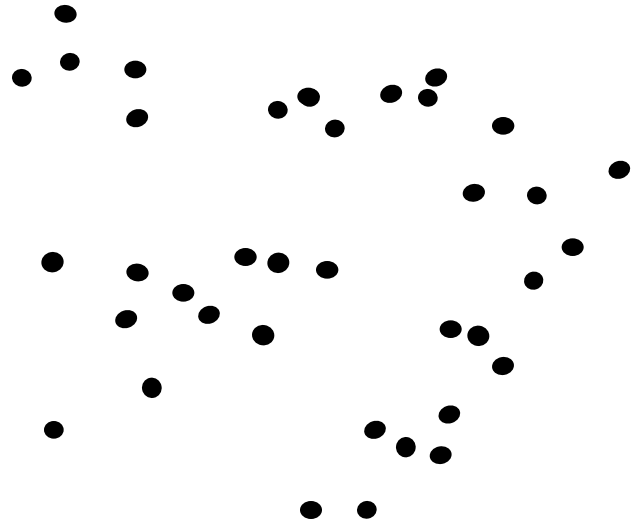
Based on Dendrogram

Hierarchical Clustering

- Agglomerative Clustering
  - Start with one cluster per example
  - Merge the two nearest clusters
  - (Criteria: min, max, average, mean distance)
  - Repeat until all are one cluster
  - Output is a Dendrogram
- Divisive Clustering
  - Start with everything in
  - one cluster
  - Split into two
  - (e.g. by min-cut)
  - etc.



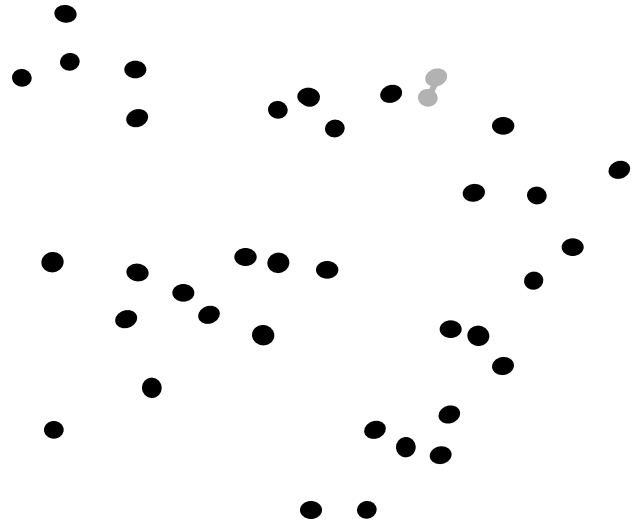
Single Linkage Hierarchical Clustering



1. Say “Every point is its own cluster”

Based on Moore

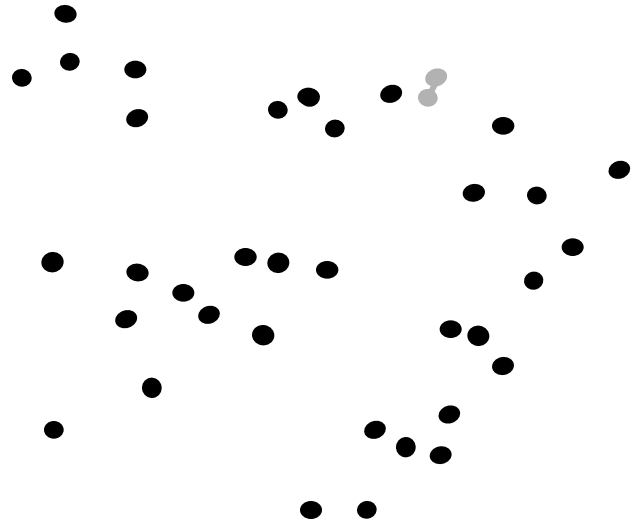
Single Linkage Hierarchical Clustering



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

Based on Moore

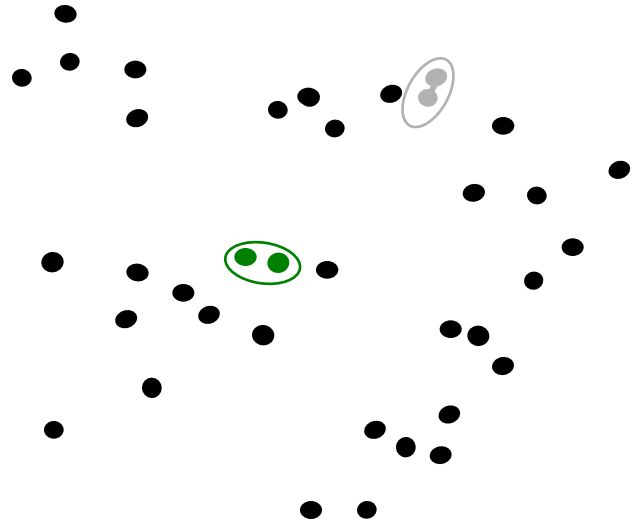
Single Linkage Hierarchical Clustering



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

Based on Moore

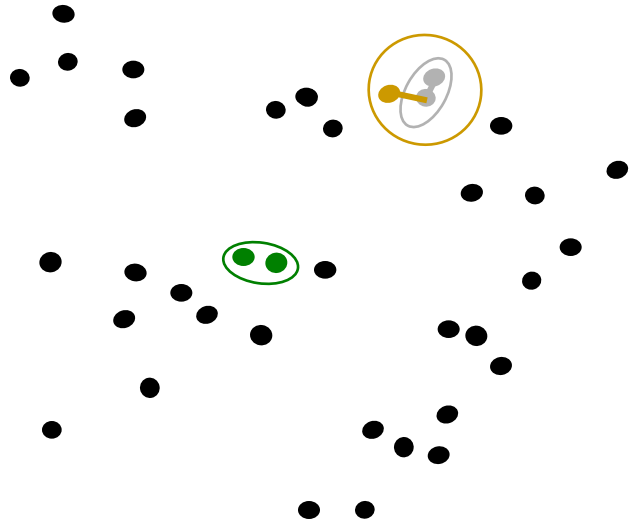
Single Linkage Hierarchical Clustering



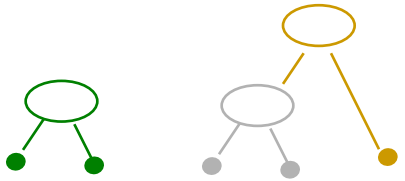
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat

Based on Moore

Single Linkage Hierarchical Clustering



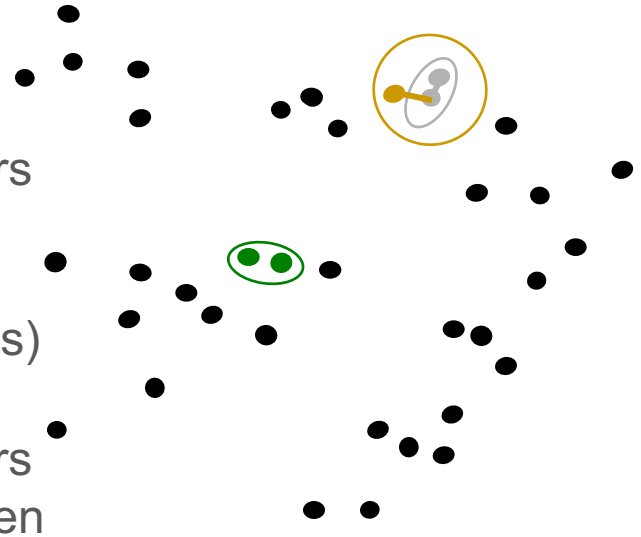
1. Say "Every point is its own cluster"
2. Find "most similar" pair of clusters
3. Merge it into a parent cluster
4. Repeat



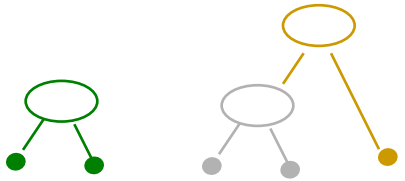
Based on Moore

Single Linkage Hierarchical Clustering

- How do we define similarity between clusters?
- Minimum distance between points in clusters (in which case we're simply doing Euclidian Minimum Spanning Trees)
- 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



Based on Moore

Single Linkage Comments

- It's nice that you get a hierarchy instead of an amorphous collection of groups
- If you want k groups, just cut the (k-1) longest links
- There's no real statistical or information-theoretic foundation to this.

Based on Moore

Choosing k

- Defined by the application, e.g., image quantization
- Plot data (after PCA) and check for clusters
- Incremental (leader-cluster) algorithm: Add one at a time until “elbow” (reconstruction error/log likelihood/intergroup distances)
- Manually check for meaning

K-Means Clustering

- Pick random examples as the initial means

- Repeat until convergence:
  - Assign each example to the nearest mean
  - New mean = The Average of examples assigned to it

K-Means

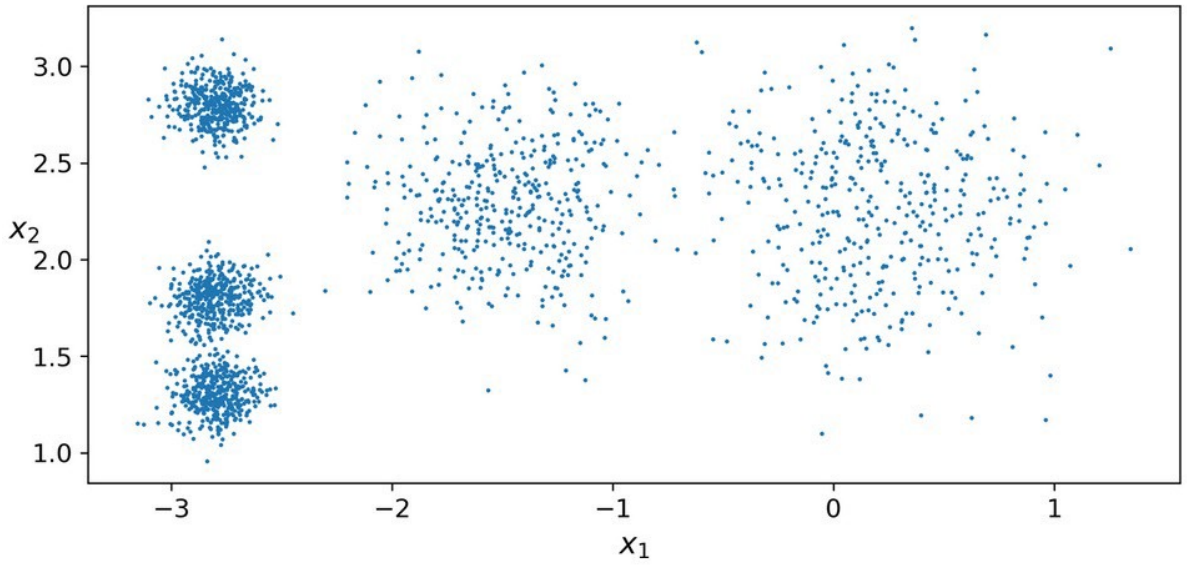


Figure 9-2. An unlabeled dataset composed of five blobs of instances

- You can see five blobs of instances in the figure above (unlabeled dataset).
- Training a K-Means clusterer on this dataset will try to find each blob's center and assign each instance to the closest blob.
- The K-Means algorithm is a simple algorithm capable of clustering this kind of dataset very quickly and efficiently, often in just a few iterations.

Training a K-Means Clusterer

1. Specify the number of clusters k that the algorithm must find.
2. Each instance is assigned to one of the clusters.
   - In the context of clustering, an instance's label is the index of the cluster that this instance gets assigned to by the algorithm: this is not to be confused with the class labels in classification (remember that clustering is an unsupervised learning task).
3. Assign new instances to the cluster whose centroid is closest.

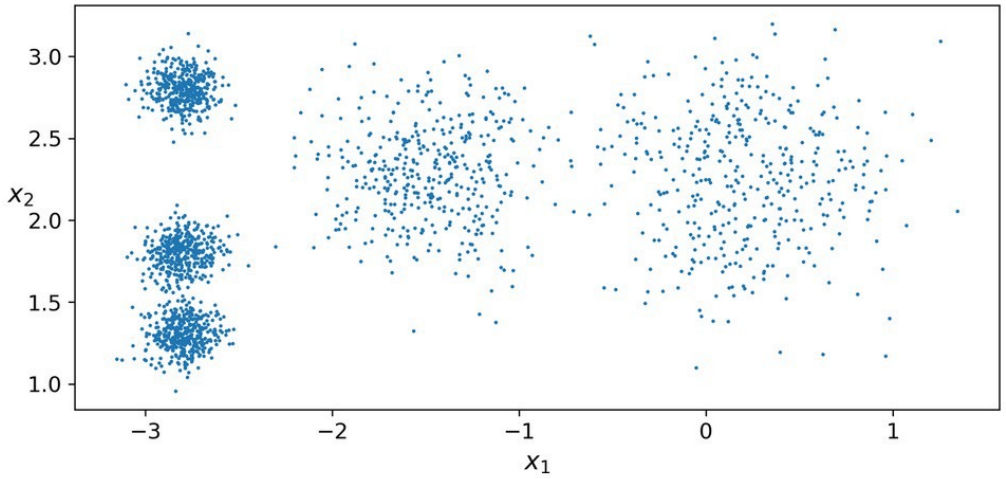


Figure 9-2. An unlabeled dataset composed of five blobs of instances

A Cluster's Decision Boundaries

Each centroid is represented with an X

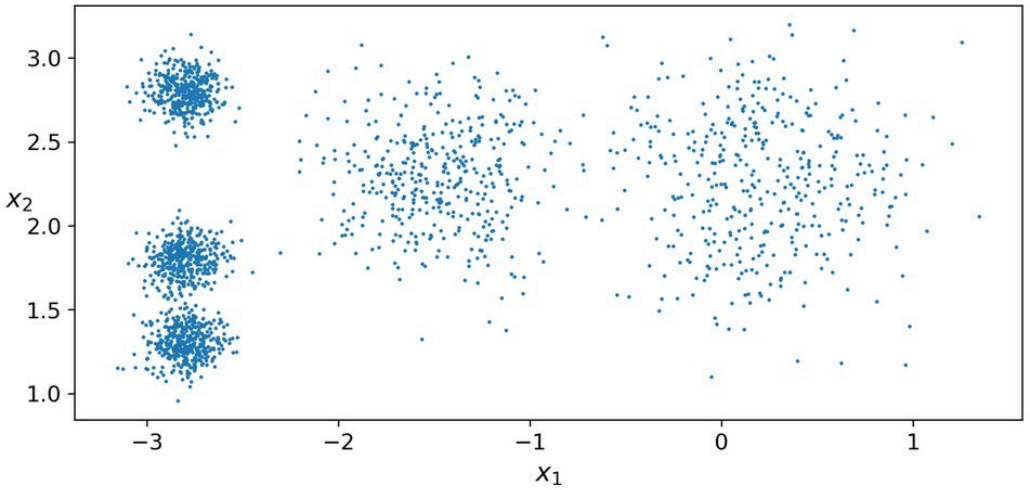


Figure 9-2. An unlabeled dataset composed of five blobs of instances

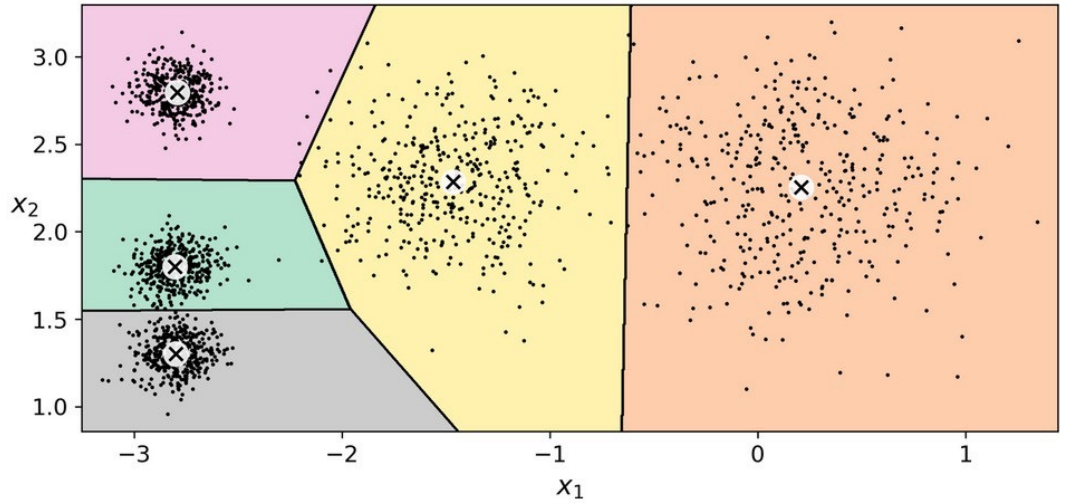


Figure 9-3. K-Means decision boundaries (Voronoi tessellation)

Characteristics of K-Means

- The K-Means algorithm does not behave very well when the blobs have very different diameters because all it cares about when assigning an instance to a cluster is the distance to the centroid.

- Instead of assigning each instance to a single cluster, which is called hard clustering, it can be useful to give each instance a score per cluster, which is called soft clustering. The score can be the distance between the instance and the centroid; conversely, it can be a similarity score (or affinity)
  - If you have a high-dimensional dataset and you transform it this way, you end up with a k-dimensional dataset: this transformation can be a very efficient nonlinear dimensionality reduction technique.

The K-Means Algorithm

- Start by placing the centroids randomly (e.g., by picking k instances at random and using their locations as centroids).
- Label the instances, update the centroids, label the instances, update the centroids, and so on until the centroids stop moving.
- The centroids are initialized randomly (top left), then the instances are labeled (top right), then the centroids are updated (center left), the instances are relabeled (center right), and so on.
- The algorithm is guaranteed to converge in a finite number of steps (usually quite small); it will not oscillate forever.

How To Do K-Means

1. The centroids are initialized randomly (top left)
2. Then the instances are labeled (top right)
3. Then the centroids are updated (center left)
4. The instances are relabeled (center right)
5. And on and on…

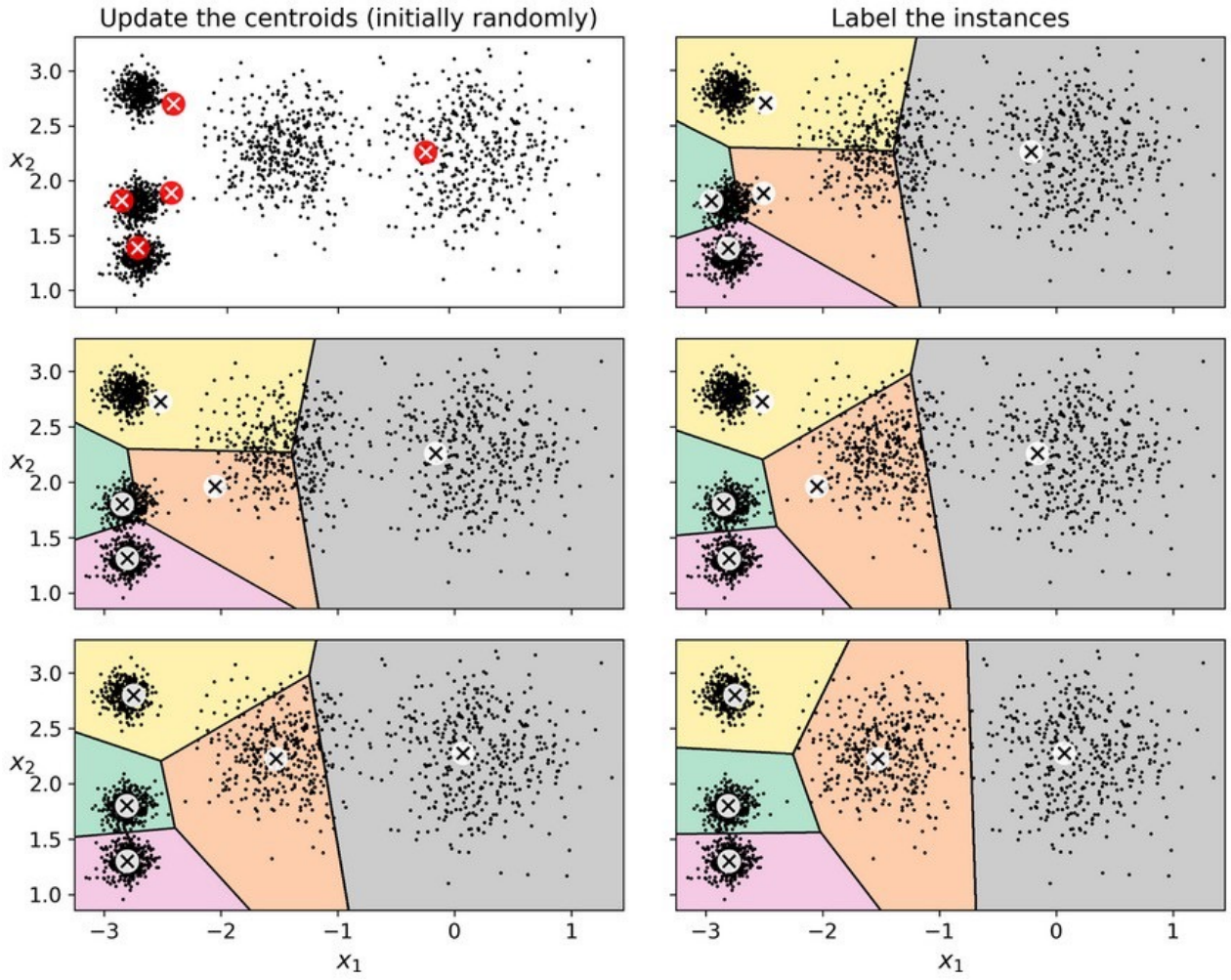


Figure 9-4. The K-Means algorithm

Local vs Global Optimum

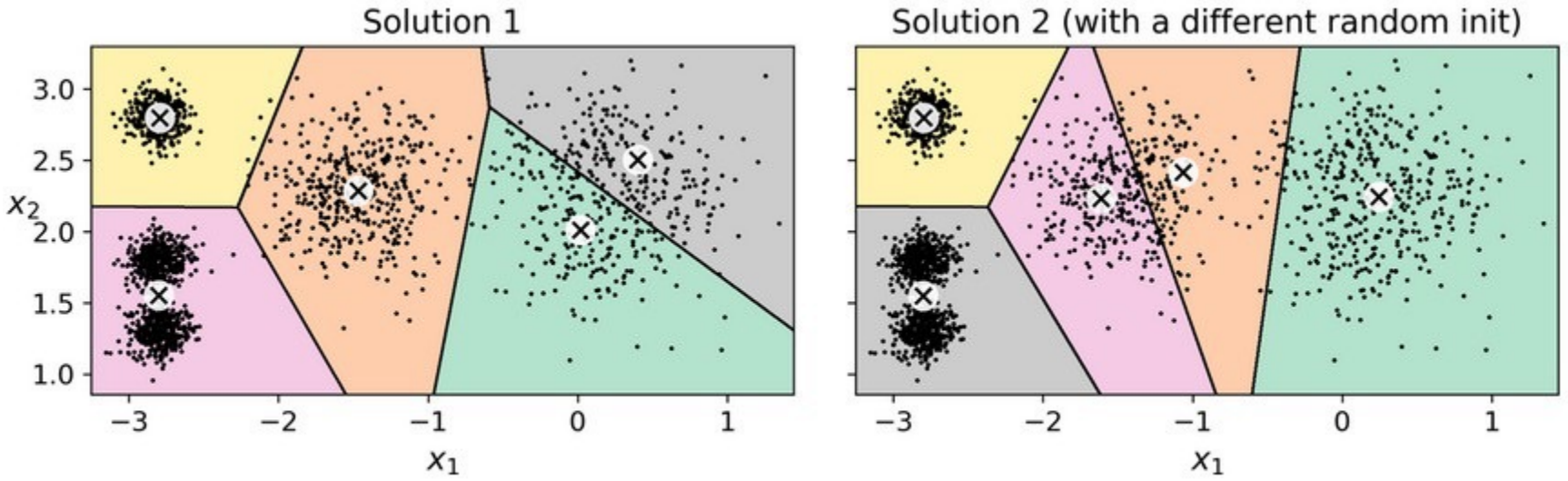
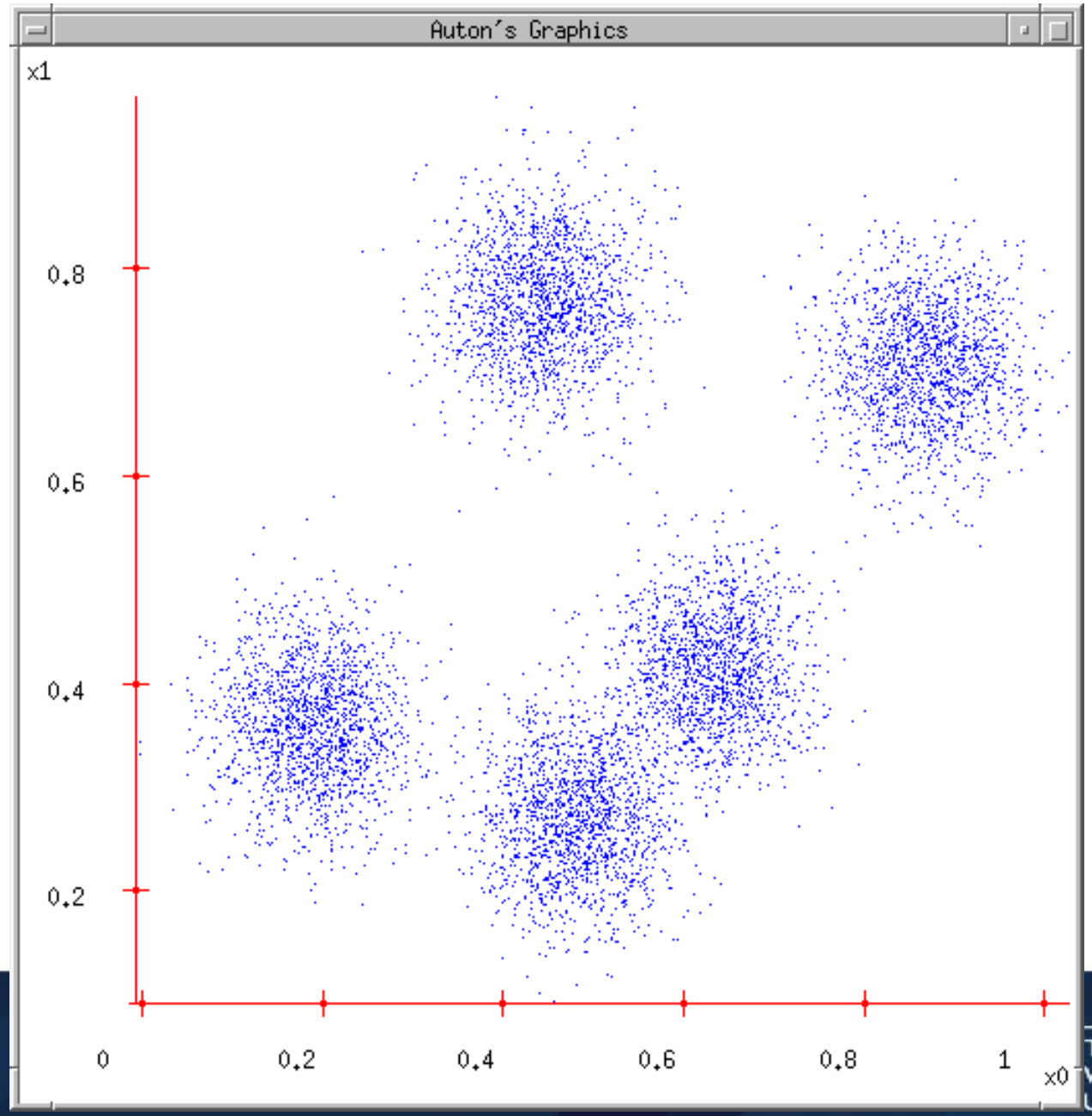


Figure 9-5. Suboptimal solutions due to unlucky centroid initializations

- Although the algorithm is guaranteed to converge, it may not converge to the right solution (i.e., it may converge to a local optimum)
- Whether it does or not depends on the centroid initialization

Some Data

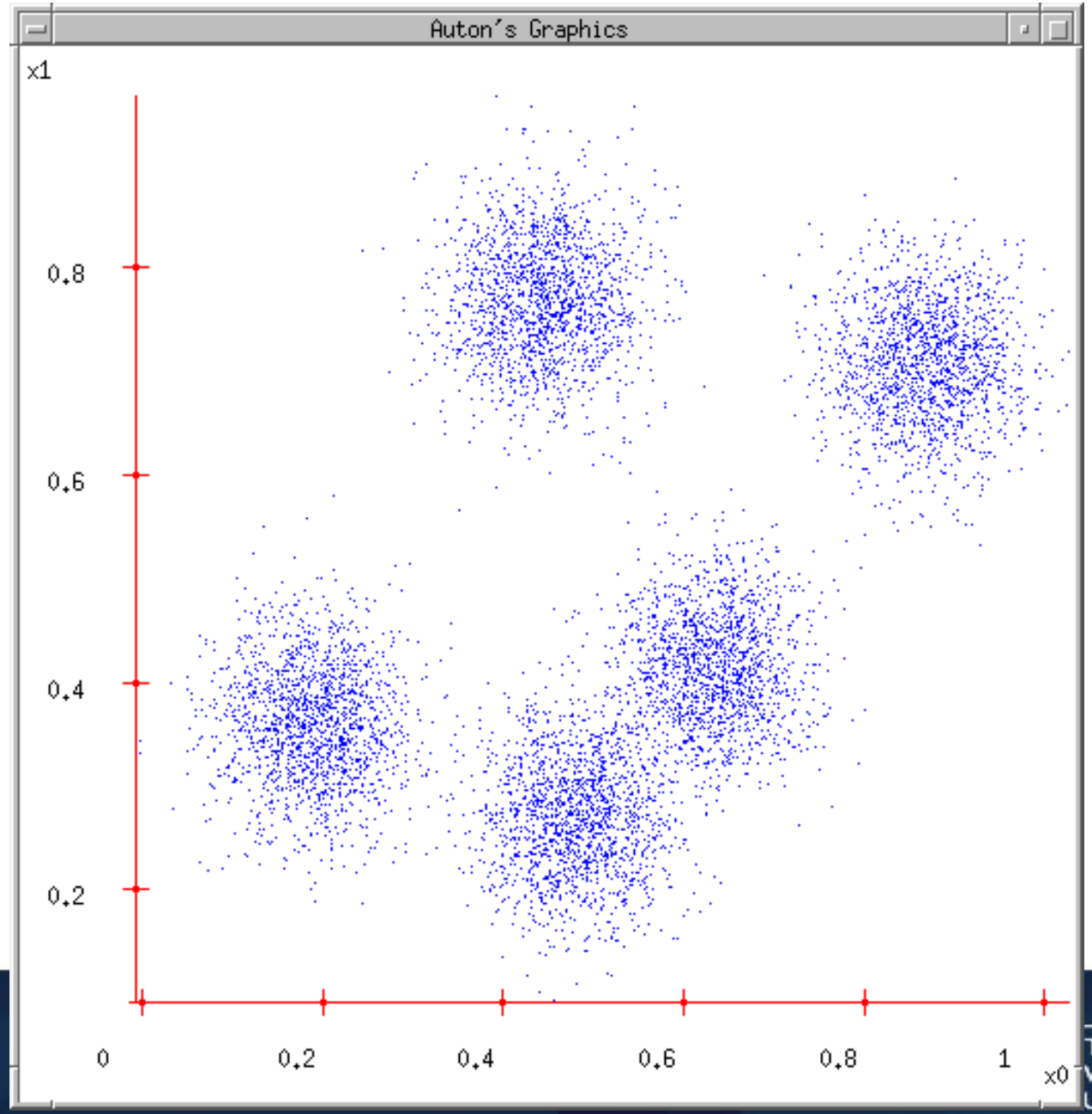
- This could easily be modeled by a Gaussian Mixture (with 5 components)
- But let's look at an satisfying, friendly and infinitely popular alternative…



Based on Moore

K-Means

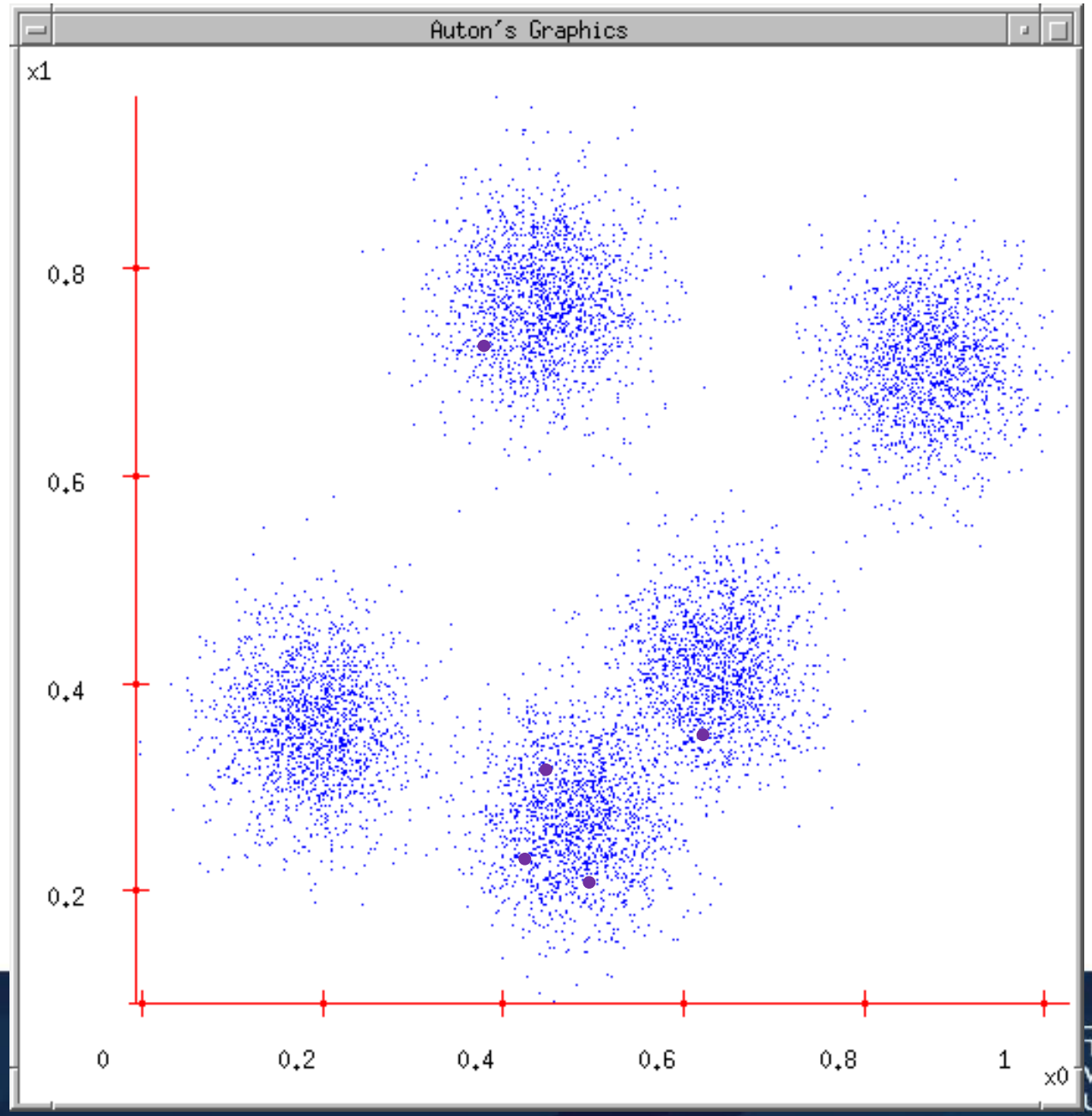
1. Ask user how many clusters they'd like. (e.g. k=5)



Based on Moore

K-Means

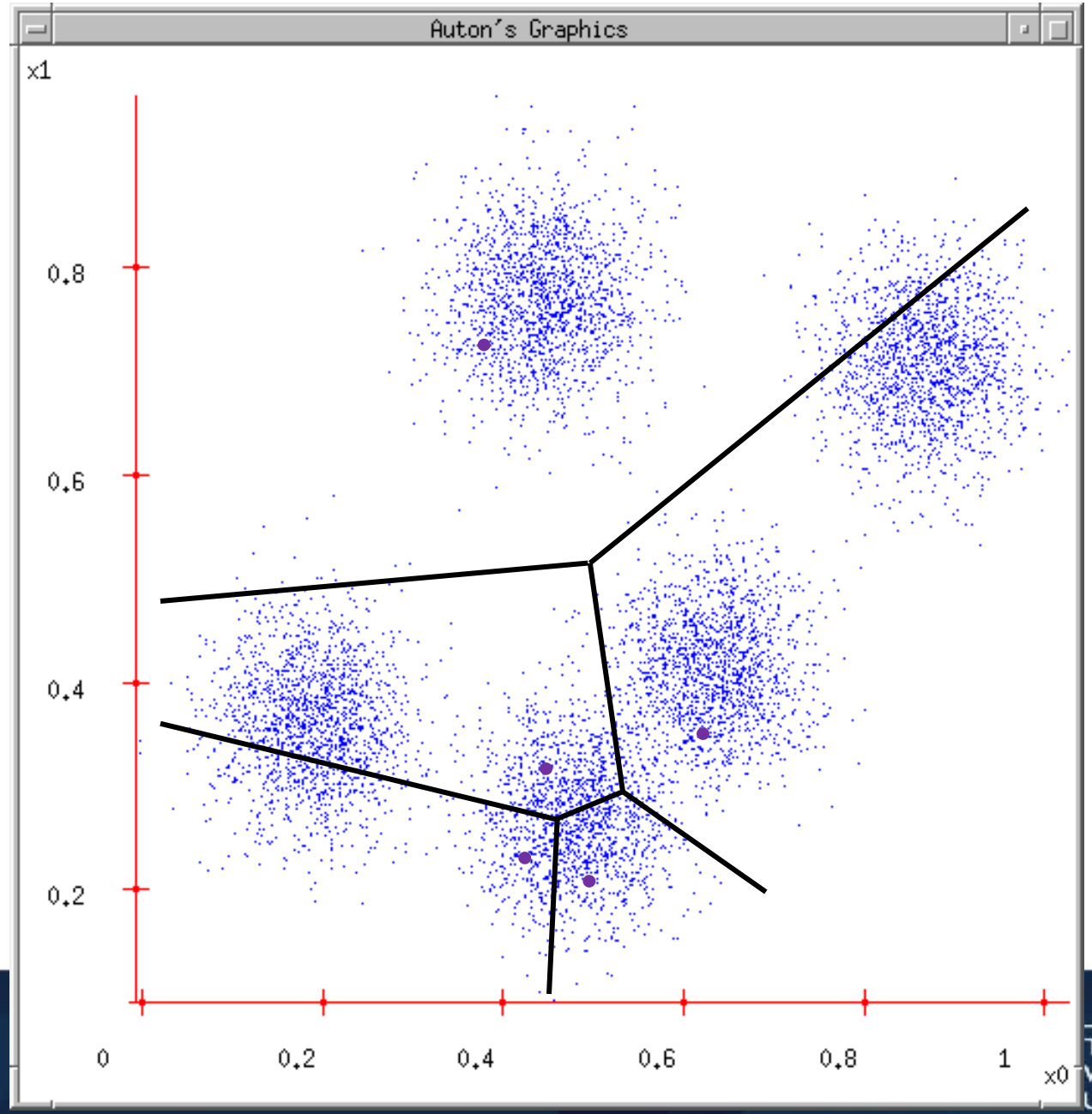
1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations



Based on Moore

K-Means

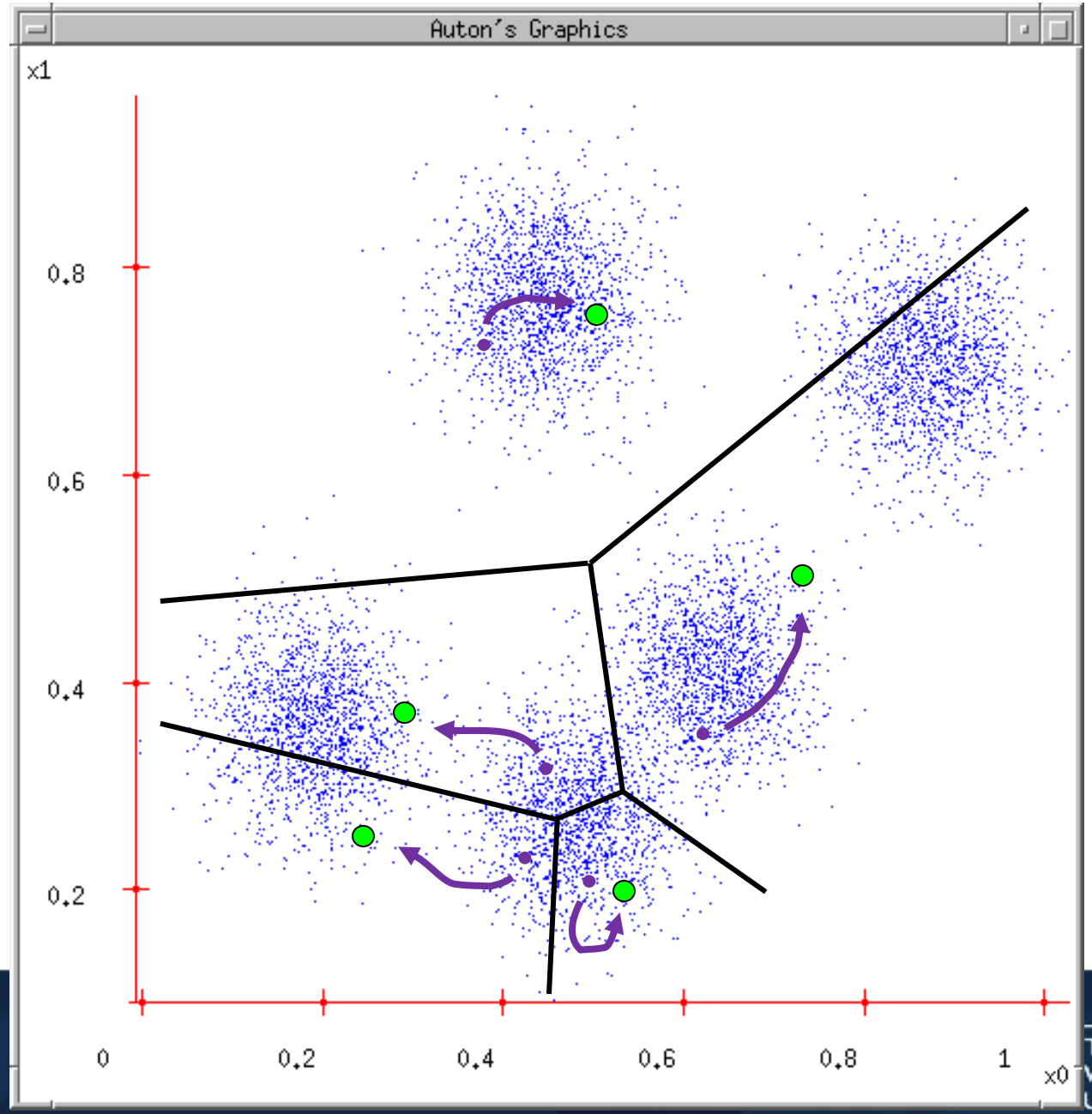
1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Based on Moore

K-Means

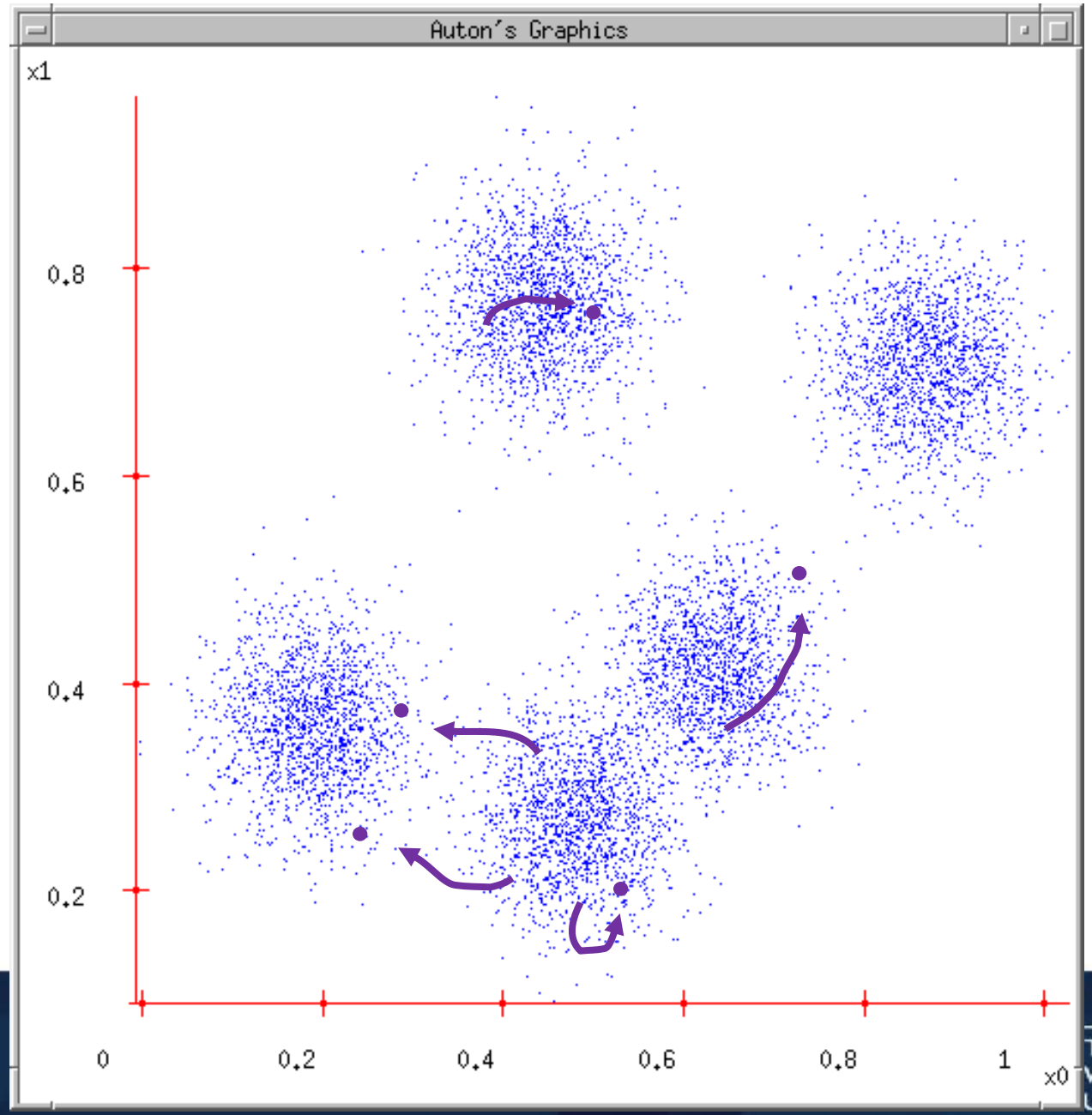
1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Based on Moore

K-Means

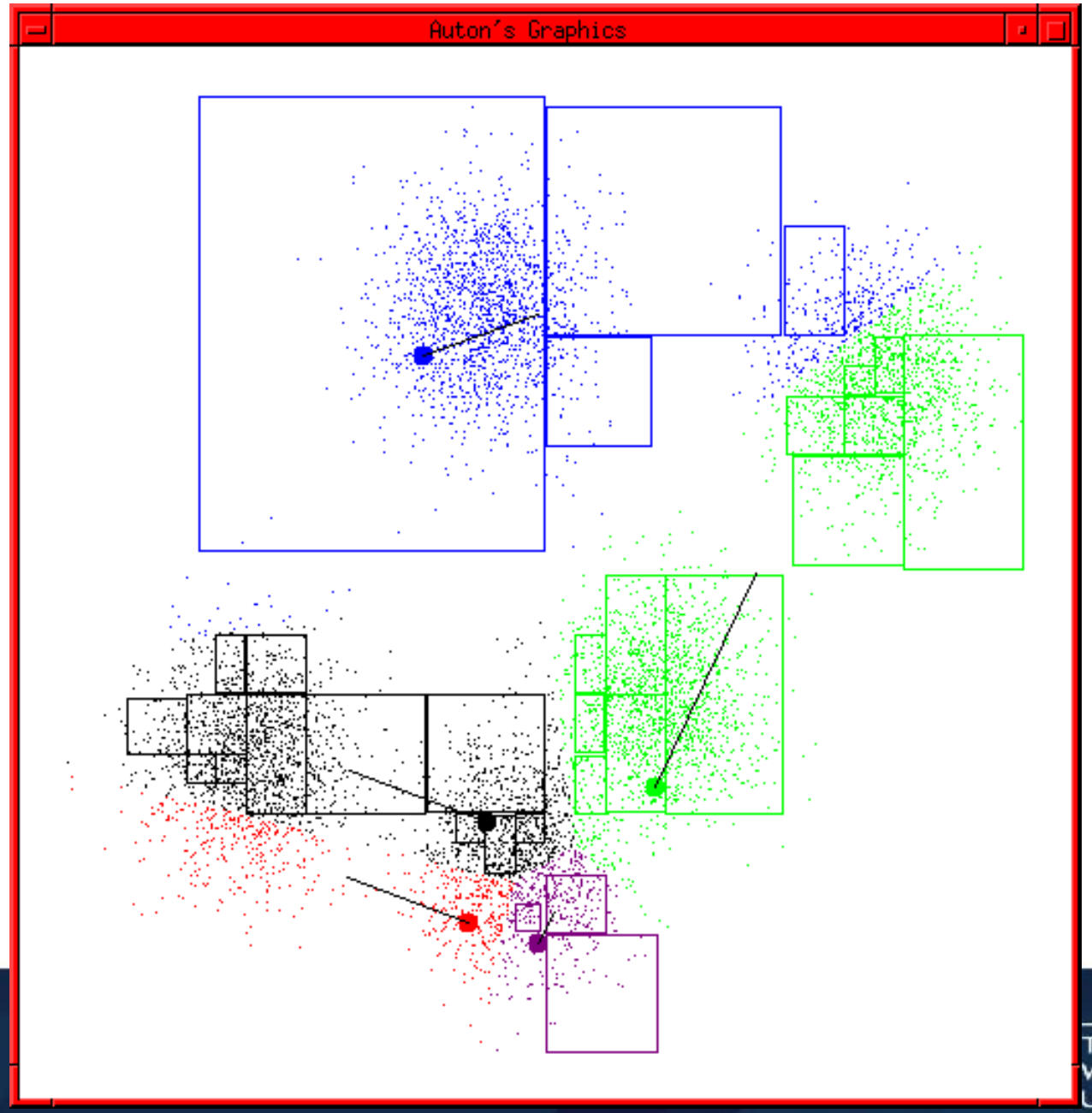
1. Ask user how many clusters they'd like. (e.g. k=5)
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Based on Moore

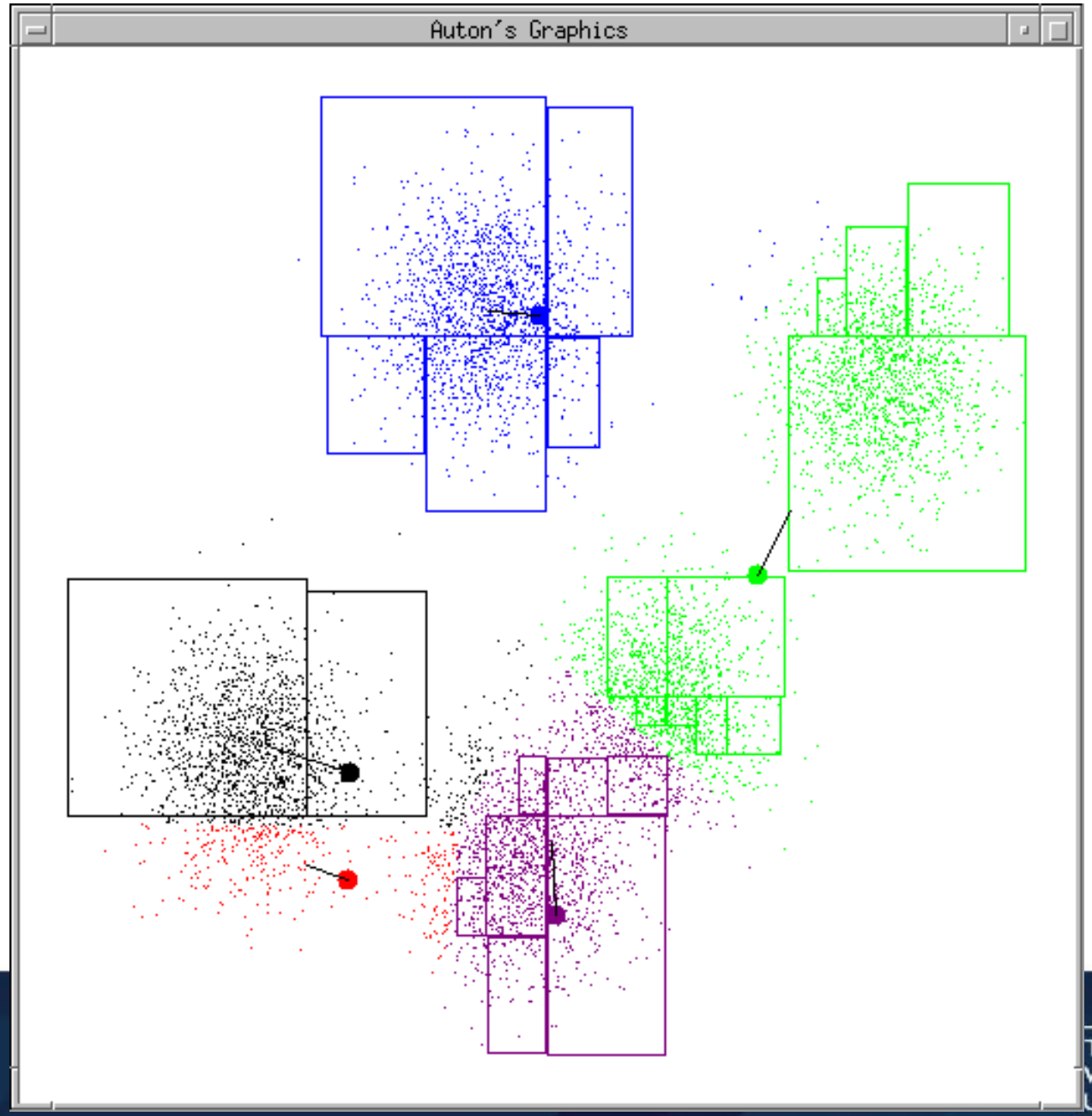
K-Means Starts

- Example generated by Dan Pelleg's super-duper fast K-means system:
  - Dan Pelleg and Andrew Moore. Accelerating Exact k-means Algorithms with Geometric Reasoning. Proc. Conference on Knowledge Discovery in Databases 1999, (KDD99) (available on www.autonlab.org/pap.html)



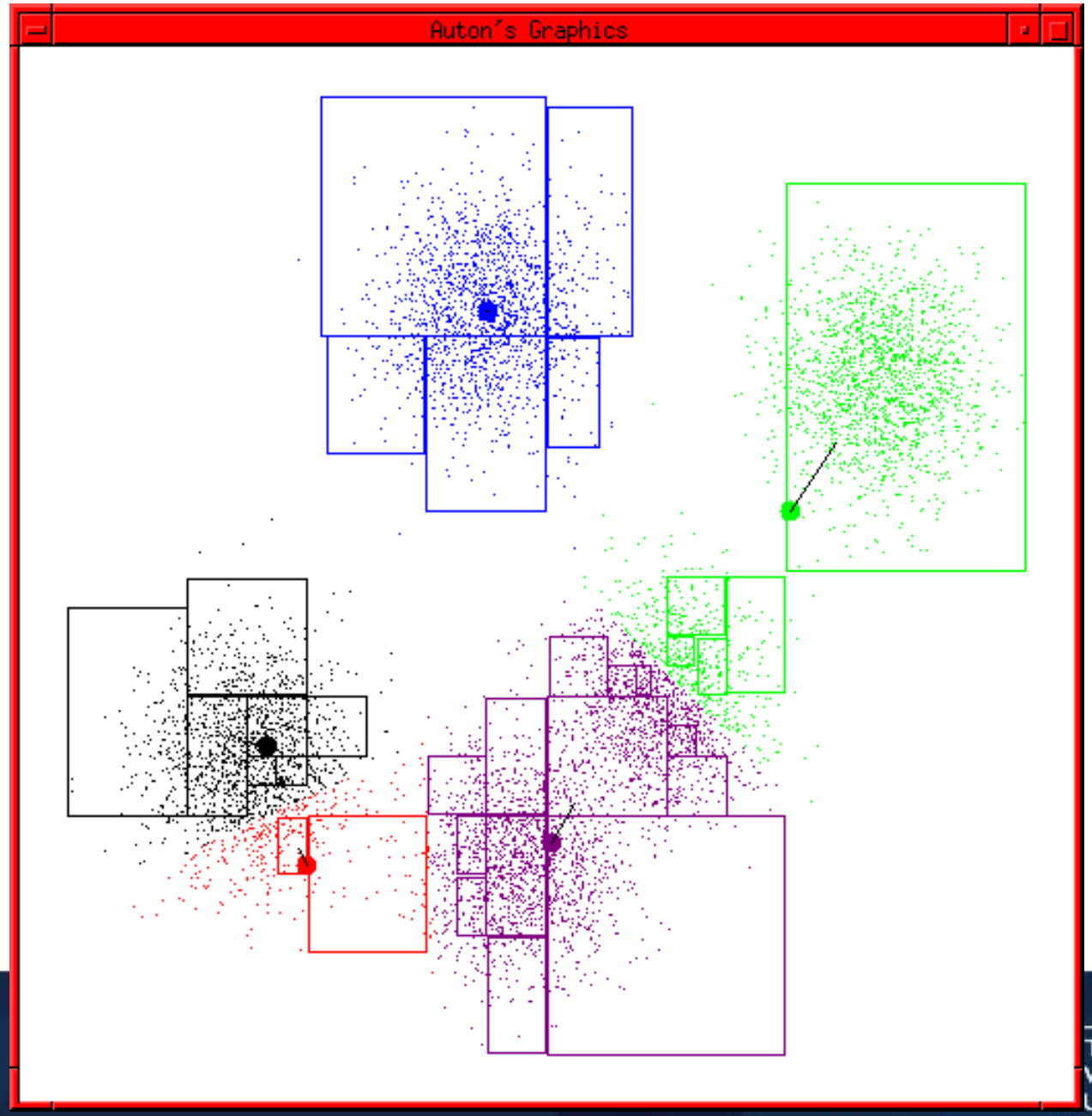
Based on Moore

K-Means Continues



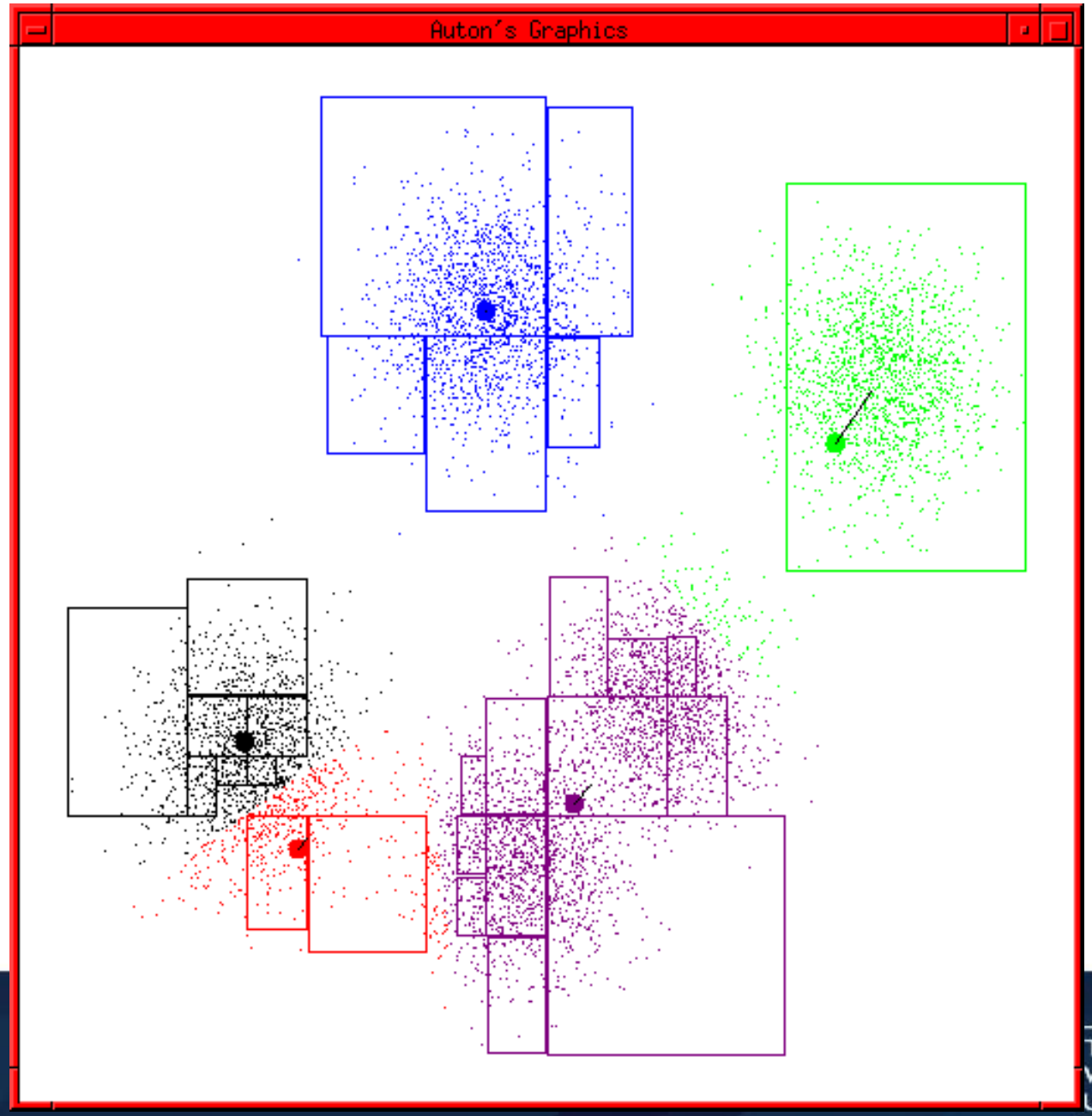
Based on Moore

K-Means Continues



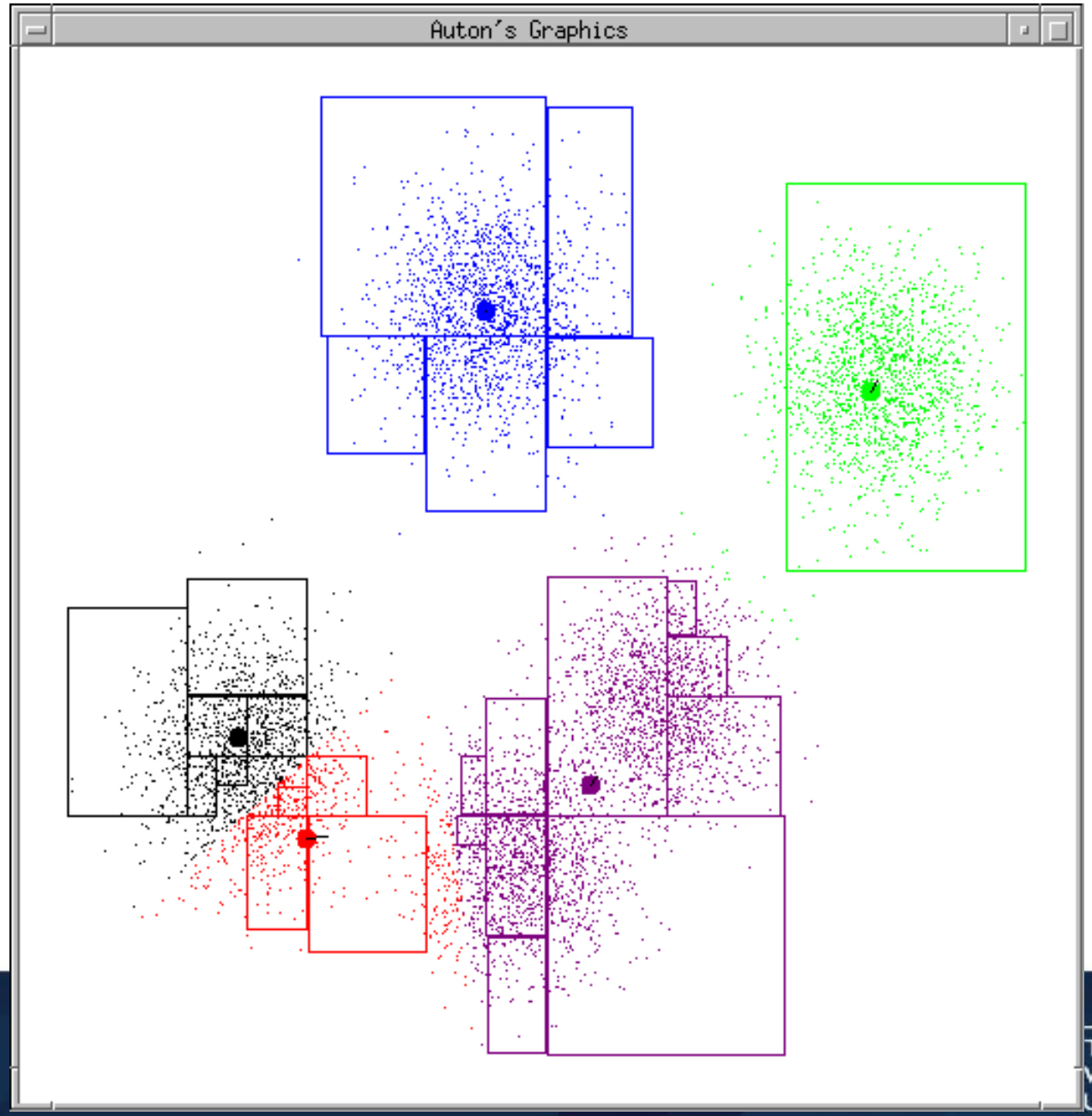
Based on Moore

K-Means Continues



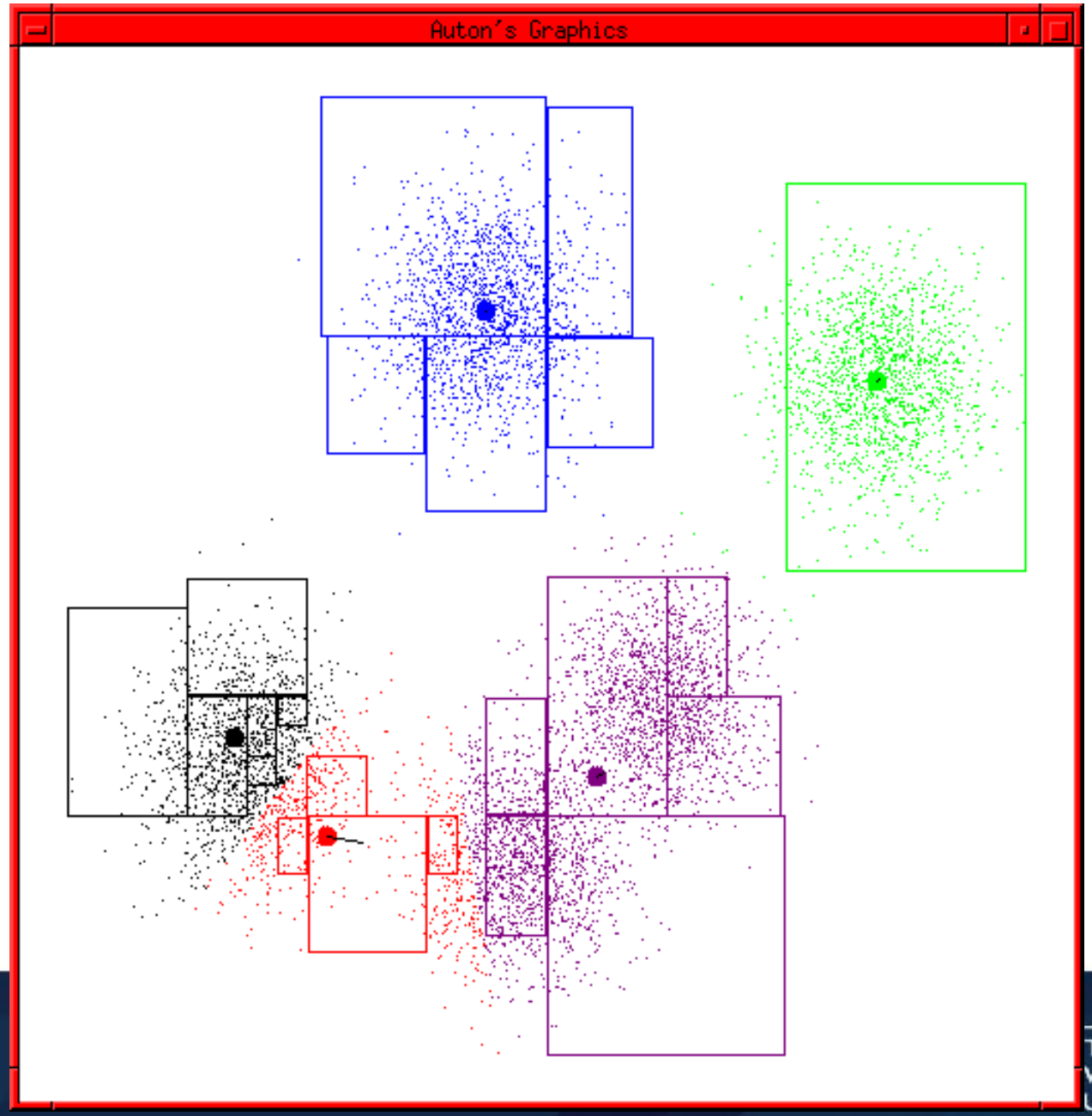
Based on Moore

K-Means Continues



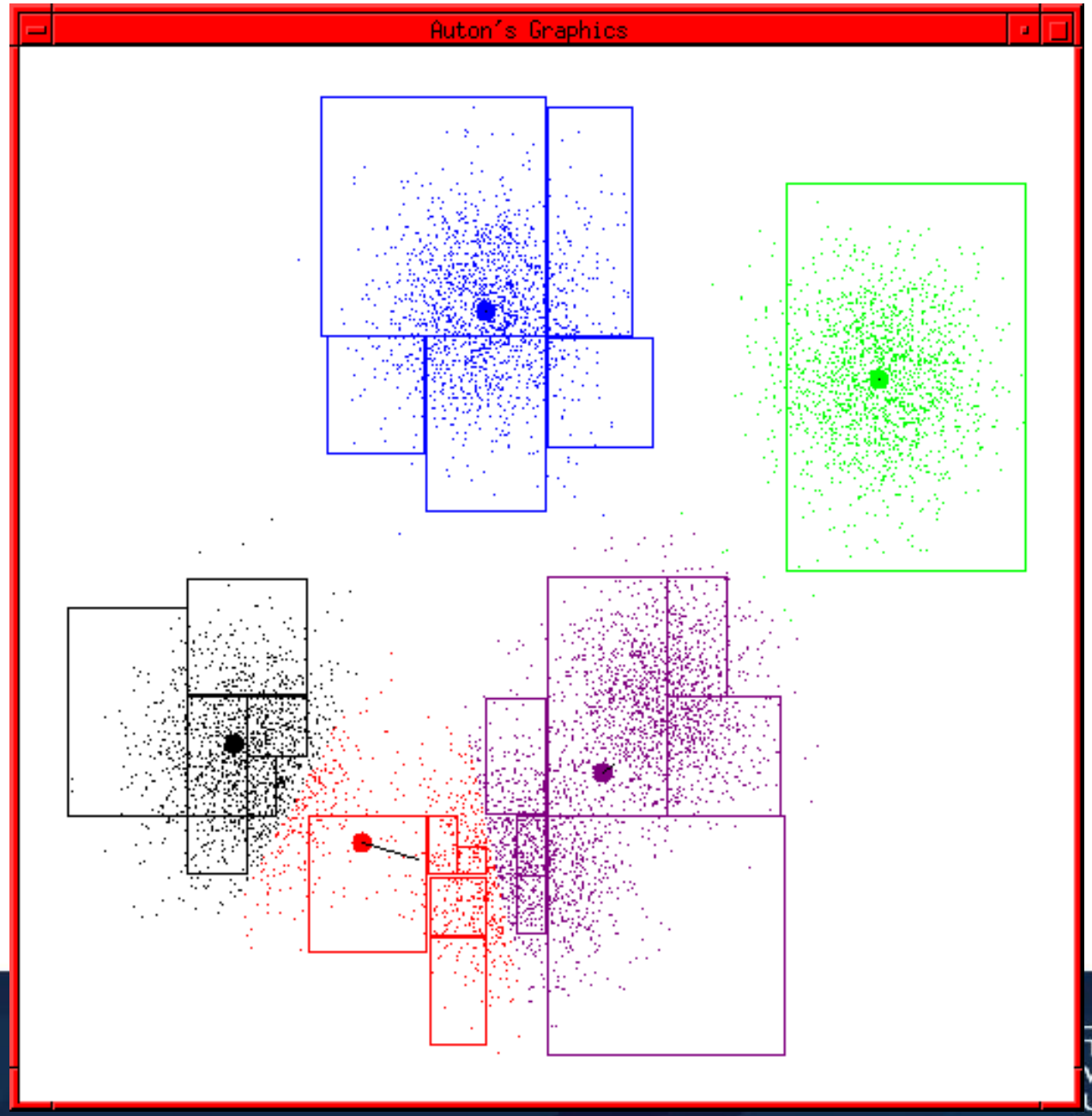
Based on Moore

K-Means Continues



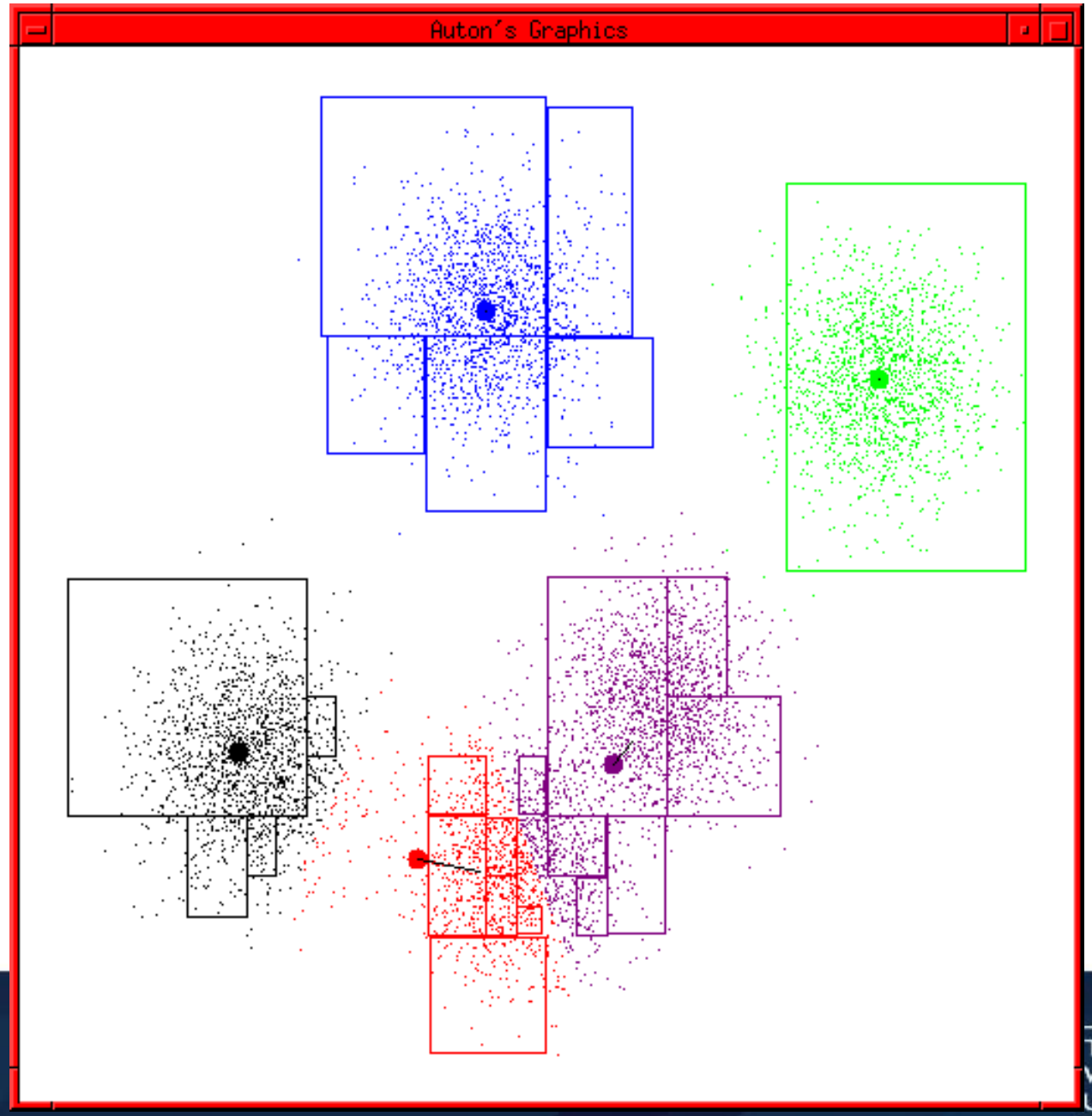
Based on Moore

K-Means Continues



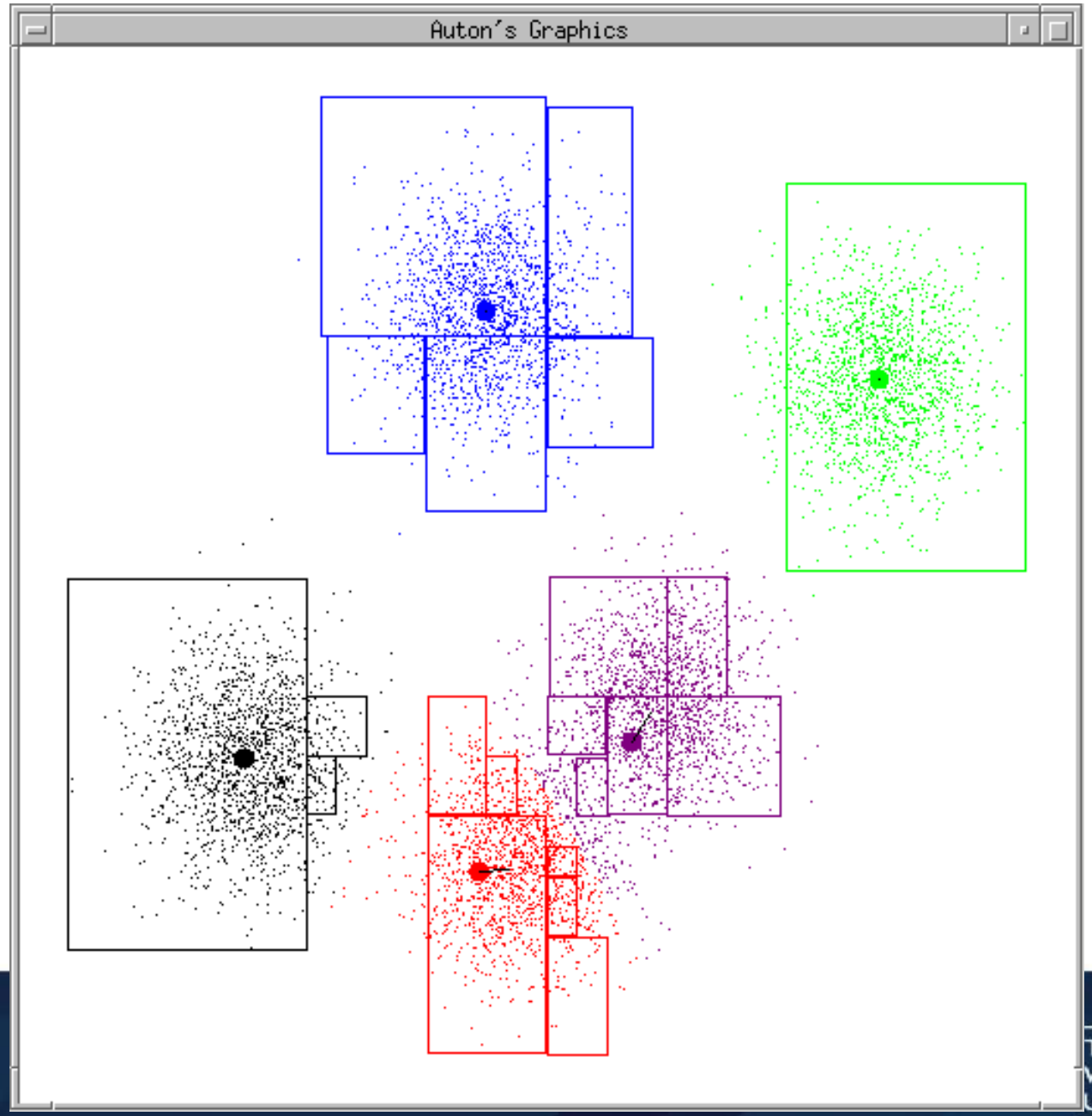
Based on Moore

K-Means Continues



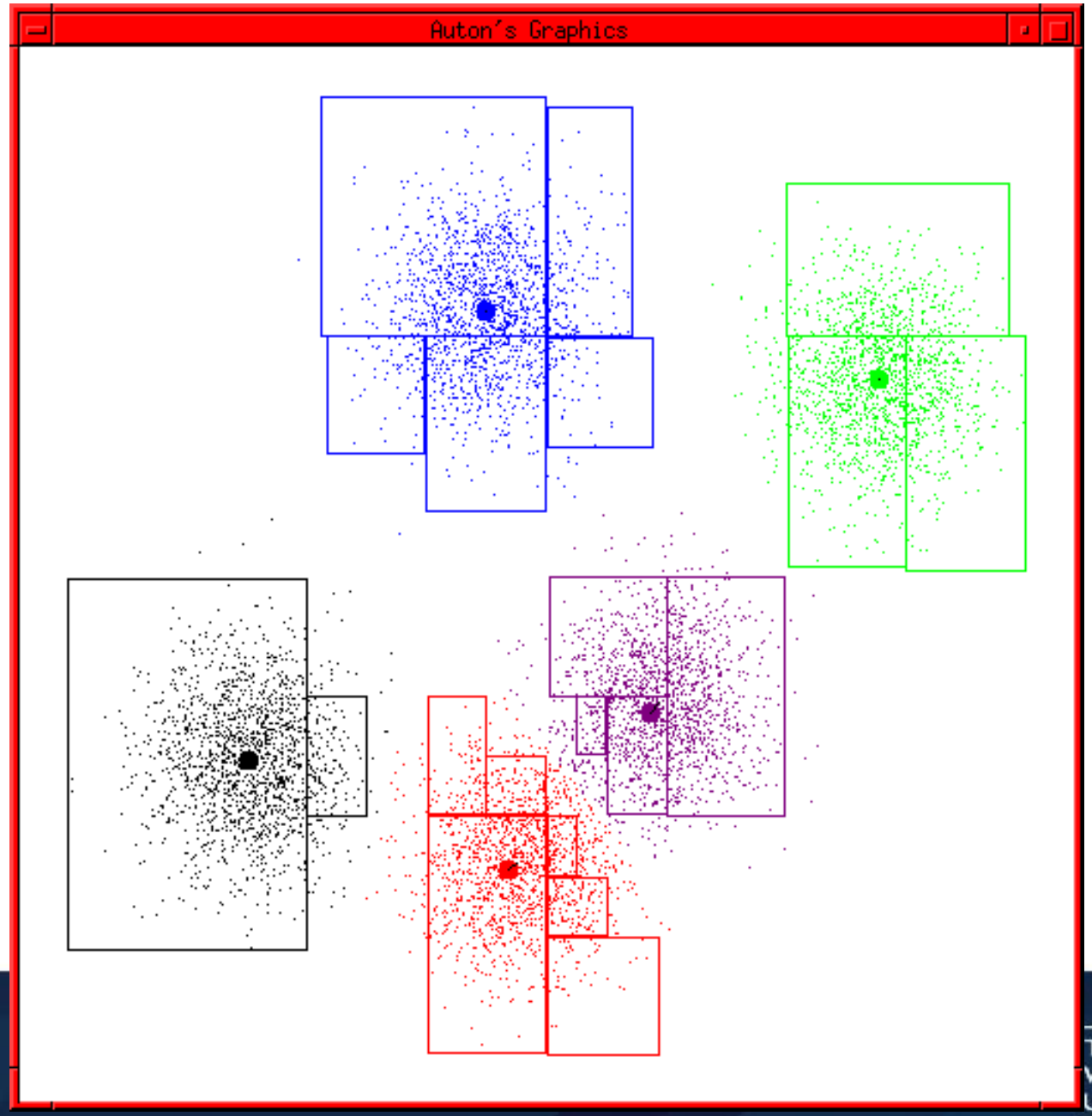
Based on Moore

K-Means Continues



Based on Moore

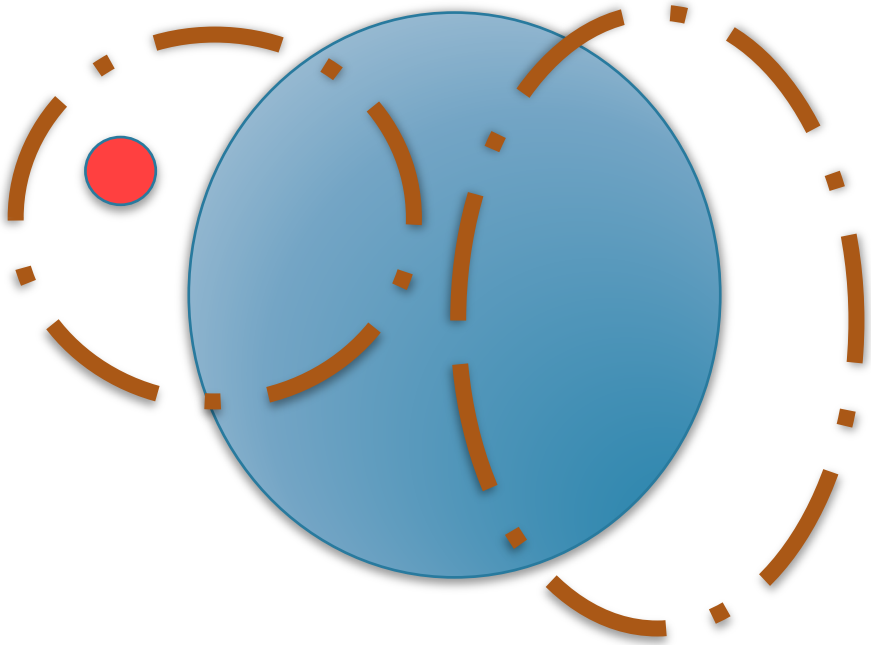
K-Means Terminates

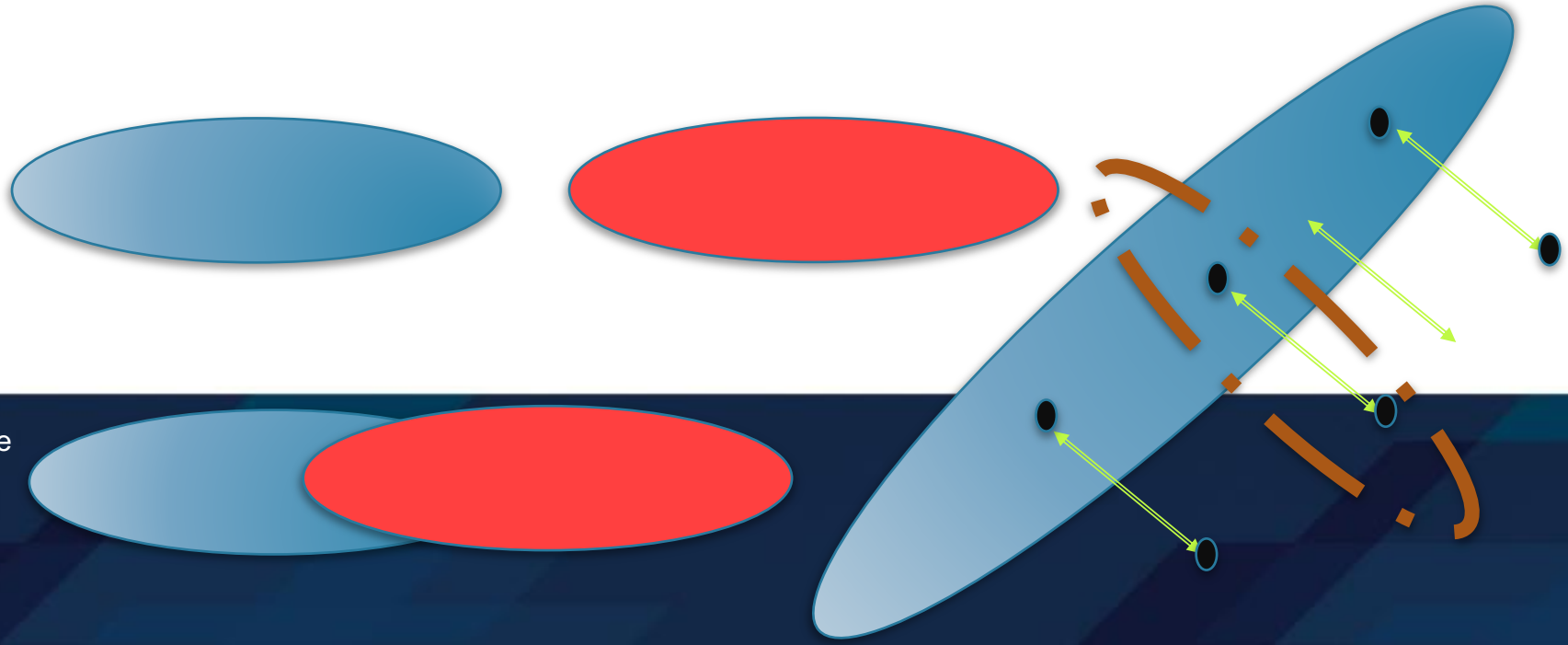


Based on Moore

K-Means Works If

- Clusters are spherical
- Clusters are well separated
- Clusters are of similar volumes
- Clusters have similar numbers of points





Based on Moore

K-Means Questions

- What is it trying to optimize?
- 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

Based on Moore

Will We Find the Optimal Configuration?

- Not necessarily.
- Can you invent a configuration that has converged, but does not have the minimum distortion?

Based on Moore

Will We Find the Optimal Configuration?

- Not necessarily.
- Can you invent a configuration that has converged, but does not have the minimum distortion? (Hint: try a k=3 configuration here…)

Based on Moore

Will We Find the Optimal Configuration?

- Not necessarily.
- Can you invent a configuration that has converged, but does not have the minimum distortion?

Based on Moore

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.

Based on Moore

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.

Neat trick:
Place first center on top of randomly chosen datapoint.
Place second center on datapoint that's as far away as possible from first center
:
Place j'th center on datapoint that's as far away as possible from the closest of Centers 1 through j-1
:

Based on Moore

Centroid Initialization Methods

- If you don't happen to know approximately where the centroids should be:
- Run the algorithm multiple times with different random initializations and keep the best solution
  - How exactly does it know which solution is the best?
  - It uses a performance metric! That metric is called the model's inertia, which is the mean squared distance between each instance and its closest centroid.
- An important improvement to the K-Means algorithm, K-Means++, was proposed in a 2006 paper by David Arthur and Sergei Vassilvitskii.
  - Introduced a smarter initialization step that tends to select centroids that are distant from one another, and this improvement makes the K-Means algorithm much less likely to converge to a suboptimal solution.

Choosing the Number of Centers – A Difficult Problem

$$BIC = \log(m)p - 2\log\left(\widehat{L}\right)$$

$$AIC = 2p - 2\log\left(\widehat{L}\right)$$

In these equations:

- m is the number of instances, as always.
- p is the number of parameters learned by the model.
- $\widehat{L}$ is the maximized value of the *likelihood function* of the model.

- Both the BIC and the AIC penalize models that have more parameters to learn (e.g., more clusters) and reward models that fit the data well.
- They often end up selecting the same model. When they differ, the model selected by the BIC tends to be simpler (fewer parameters) than the one selected by the AIC, but tends to not fit the data quite as well (this is especially true for larger datasets).

Mini-Batch K-Means

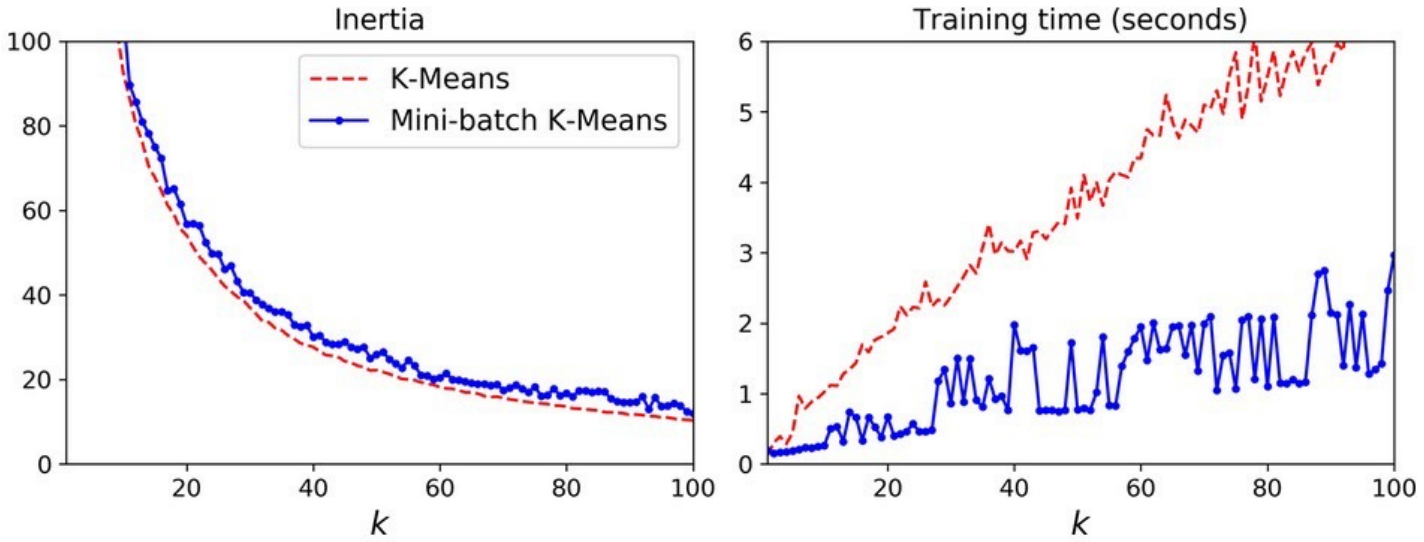


Figure 9-6. Mini-batch K-Means has a higher inertia than K-Means (left) but it is much faster (right), especially as k increases

- Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroids just slightly at each iteration.
  - This speeds up the algorithm typically by a factor of three or four and makes it possible to cluster huge datasets that do not fit in memory.
- Although the Mini-batch K-Means algorithm is much faster than the regular K-Means algorithm, its inertia is generally slightly worse, especially as the number of clusters increases.

Note – Computational Complexity

- K-Means is generally one of the fastest clustering algorithms.

- The computational complexity of the algorithm is generally linear with regard to the number of instances m, the number of clusters k, and the number of dimensions n.

Finding the Optimal Number of Clusters

- You might be thinking that we could just pick the model with the lowest inertia, right?
- Unfortunately, it is not that simple.
- The inertia for k=3 is 653.2, which is much higher than for k=5 (which was 211.6). But with k=8, the inertia is just 119.1.
- The inertia is not a good performance metric when trying to choose k because it keeps getting lower as we increase k.
- Indeed, the more clusters there are, the closer each instance will be to its closest centroid, and therefore the lower the inertia will be.

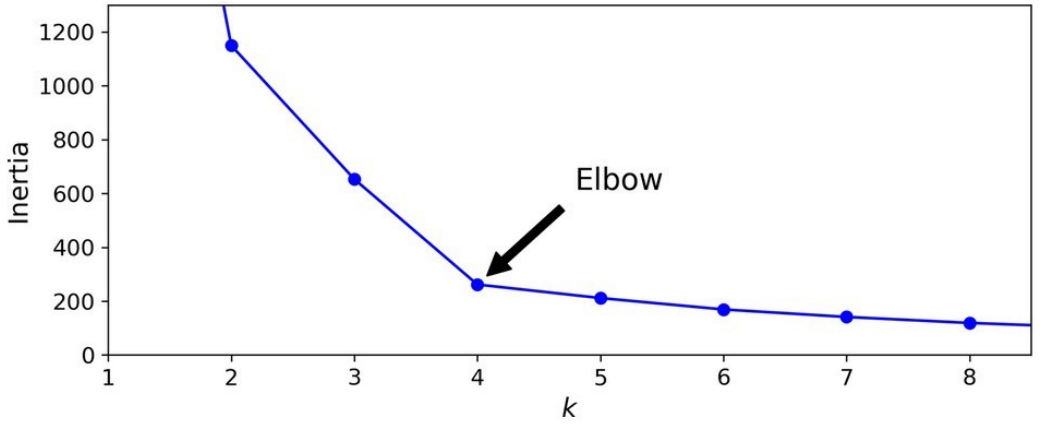


Figure 9-8. When plotting the inertia as a function of the number of clusters *k*, the curve often contains an inflexion point called the "elbow"

Finding the Optimal Number of Clusters

- A more precise approach (but also more computationally expensive) is to use the **silhouette score**, which is the mean silhouette coefficient over all the instances.
- An instance's silhouette coefficient is equal to (b – a) / max(a, b), where a is the mean distance to the other instances in the same cluster (i.e., the mean intra-cluster distance) and b is the mean nearest-cluster distance (i.e., the mean distance to the instances of the next closest cluster).
- The silhouette coefficient can vary between –1 and +1.

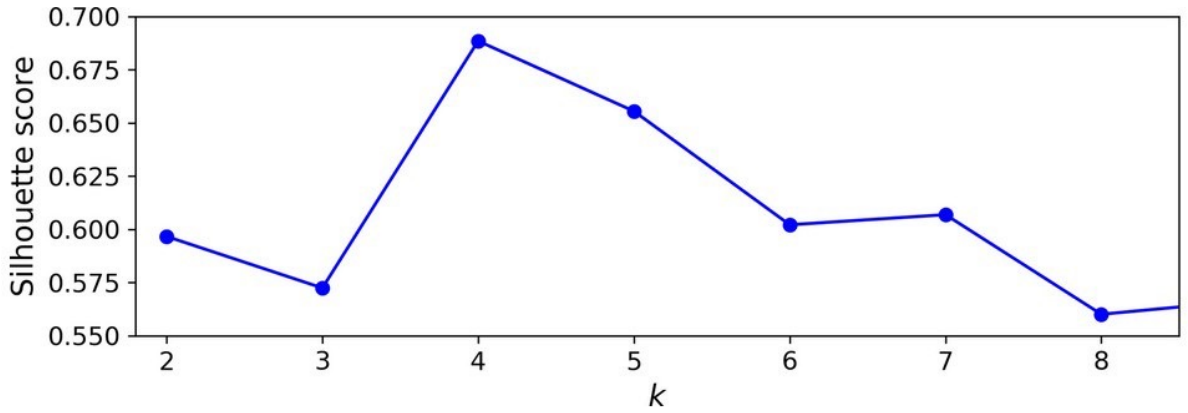


Figure 9-9. Selecting the number of clusters *k* using the silhouette score

- A coefficient close to +1 means that the instance is well inside its own cluster and far from other clusters, while a coefficient close to 0 means that it is close to a cluster boundary, and finally a coefficient close to –1 means that the instance may have been assigned to the wrong cluster.

Limits of K-Means

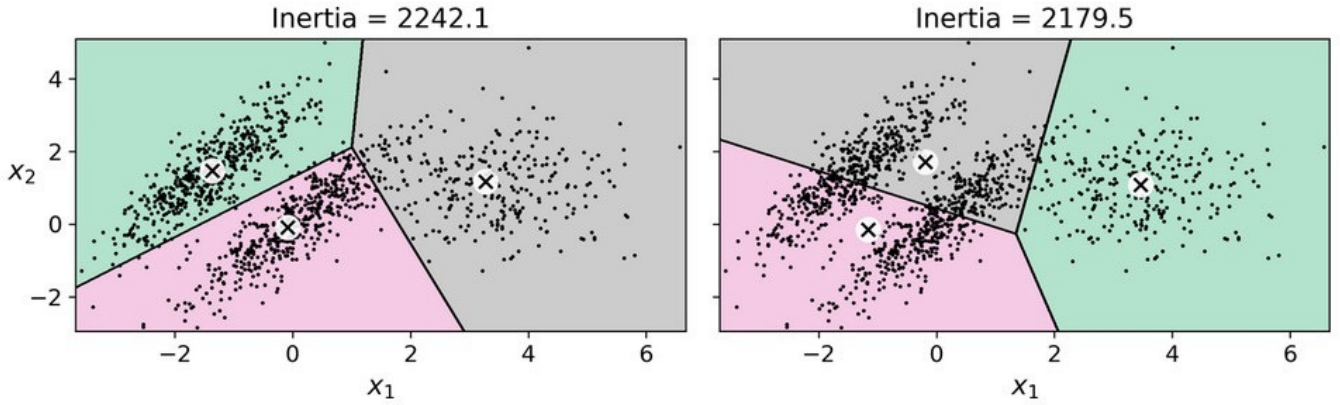


Figure 9-11. K-Means fails to cluster these ellipsoidal blobs properly

- It is necessary to run the algorithm several times to avoid suboptimal solutions.
- You need to specify the number of clusters, which can be quite a hassle.
- K-Means does not behave very well when the clusters have varying sizes, different densities, or nonspherical shapes.
- As you can see, neither of these solutions is any good. The solution on the left is better, but it still chops off 25% of the middle cluster and assigns it to the cluster on the right. The solution on the right is just terrible, even though its inertia is lower.
- Depending on the data, different clustering algorithms may perform better.

Tip – Input Feature Scaling

- It is important to scale the input features before you run K-Means, or the clusters may be very stretched and K-Means will perform poorly.

- Scaling the features does not guarantee that all the clusters will be nice and spherical, but it generally improves things.

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!

Based on Moore

Expectation Maximization (EM)

- What if you can't label your data but you need labels?
- You can treat the labels as "hidden variables"
- You can use Expectation Maximization
- Example: Jane's machine learning instructor Professor Xavier gives out an exam that consists of 50 true/false questions. Jane's class of 100 students takes the exam and Carlos goes to grade their solutions. If Professor Xavier made an answer key, this would be easy: he would just count the fraction of correctly answered questions each student got, and that would be their score. But, Professor Xavier was really busy and didn't have time to make an answer key. Can he still grade the exam?
- There are two insights that suggest that he might be able to do this:
- Let's say we know that Jane is an awesome student, and gets 100% on every exam. In that case, Professor Xavier can simply use Jane's answers as the answer key.
- What else can we assume? On average students are better than random guessing, we can hope that the majority answer for each question is likely to be correct.
- NOTE: In reality no, grading should not be done the way I am about to illustrate!

Chicken & Egg Problem

- If you knew the student's scores, you could estimate an answer key.
- If you had an answer key, you could compute student scores.
- A common approach to such problems is to iterate.
- Take a guess at the first, compute the second, recompute the first, and so on.

The Expectation Maximization (EM) Algorithm

Initialize parameters ignoring missing information

Repeat until convergence:

E step: Compute expected values of unobserved variables, assuming current parameter values

M step: Compute new parameter values to maximize probability of data (observed & estimated)

(Also: Initialize expected values ignoring missing info)

Connections

- K-Means is a special case of E-M
- For Gaussian mixtures, the hidden variables drive observation belongs to which component.
- If we knew these hidden variables we could use them as class labels in a supervised setting, we would know which parameters to adjust to fit that data point.
- With the E-M algorithm, in the E-step we estimate these labels given our current knowledge of components, and in the M-step we update our component knowledge given the labels estimated in the E-step.
- Note that these two steps are the same as the two steps of k-means; calculation of b^t_i (E-step) and re-estimation of m_i (M-step).

What You Should Know

- All the details of K-means
- The theory behind K-means as an optimization algorithm
- How K-means can get stuck
- The outline of Hierarchical clustering
- Be able to contrast between which problems would be relatively well/poorly suited to K-means vs Expectation Maximization vs Hierarchical clustering

The K-Means Algorithm

- Dimensionality Reduction
  - Reduce the number of features/attributes in the data
  - Dimensionality reduction methods find correlations between variables for grouping
  - PCA – exploratory technique

- Unsupervised Learning
  - Find the regularities in the input
  - Clustering methods find similarities between instances for grouping

- Clustering
  - K-Means
  - Mixture Models
  - Expectation Maximization (E-M)
  - Hierarchical/Agglomerative

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Homework Overview

Professor Hamza F. Alsarhan
SEAS 8505

This Week

- No HW or Discussion this week
- The Midterm Exam is released this evening at 8 pm ET and is available for you to start before 5 pm ET on Monday
- Be sure to start your exam early

- Once you have completed the midterm, read Chapters 8 and 9 from the textbook

Next Steps

- Come to office hours with any questions you may have.
- See you next class!

Thank you!