COMP 551 – Applied Machine Learning
Lecture 13: Unsupervised learning

Associate Instructor: Herke van Hoof

(*herke.vanhoof@mail.mcgill.ca*)

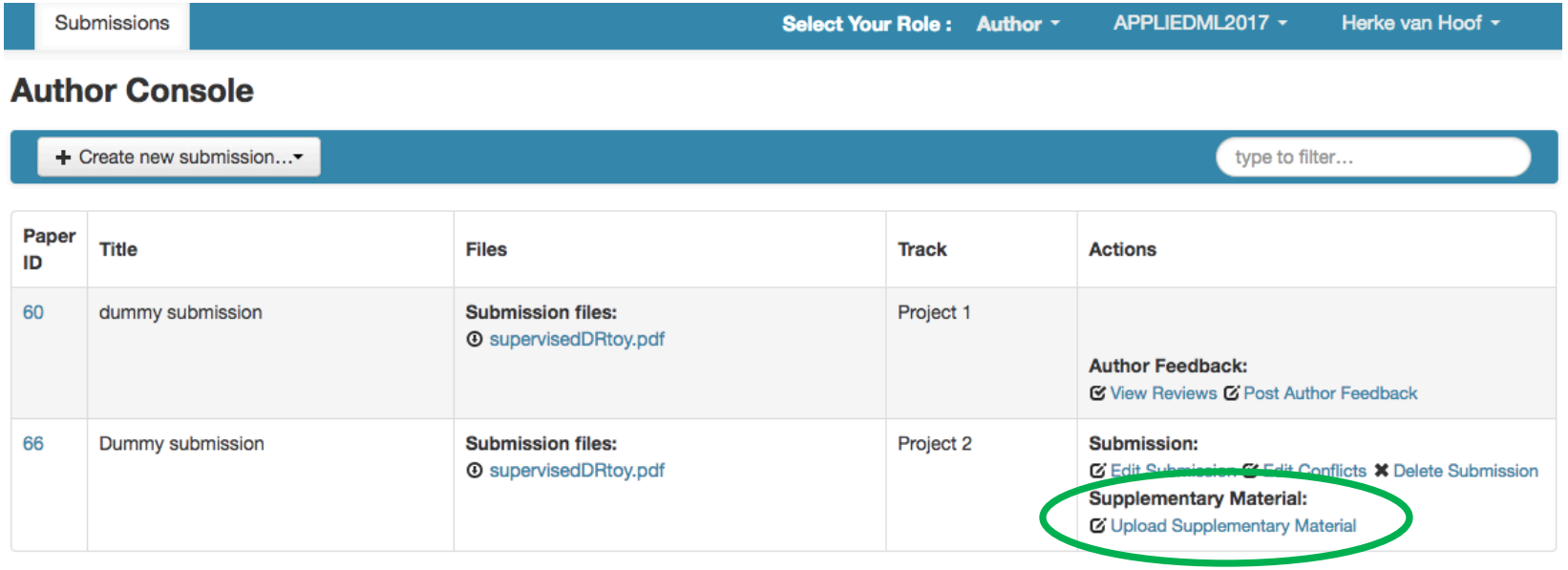
Slides mostly by: Joelle Pineau (*jpineau@cs.mcgill.ca*)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's quiz

- The hinge loss function is convex
  - True
- The number of Lagrange multipliers in the soft SVM problem is determined by the number of features in the input set.
  - False
- A quadratic programming problem can be solved in polynomial time.
  - True
- SVM with a Gaussian kernel requires specification of the variance of the kernel. This can be selected with cross-validation.
  - True
- One disadvantage of the "kernel trick" is that the memory requirement grows linearly with the numbers of features computed.
  - False

Uploading code in CMT



What is unsupervised learning?

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

- Typically use generative approaches: model the available data.

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

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 colour of each apple in the dataset.

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

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

A simple clustering example

- You know there are 5 varieties.

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

A simple clustering example

- You know there are 5 varieties.
- Assume each variety generates apples according to a (variety-specific) 2-D 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-D 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.

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.

A simple algorithm: K-means clustering

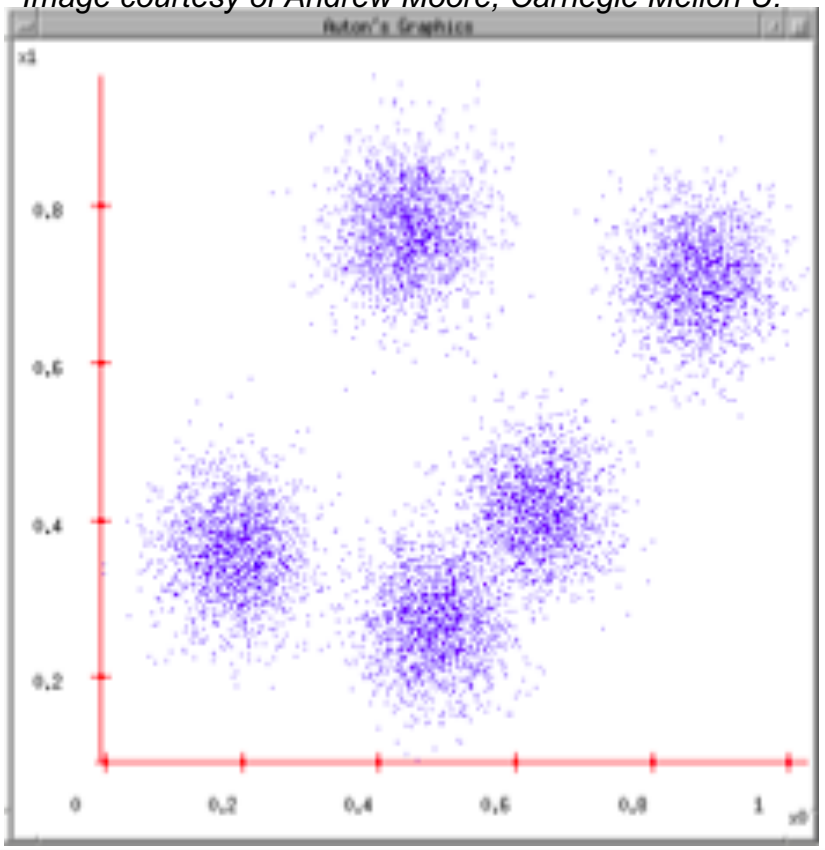
- Objective: Cluster *n* instances into *K* distinct classes.
- **Preliminaries**:
  - **Step 1**: Pick the desired number of clusters, *K*.
  - **Step 2**: Assume a parametric distribution for each class (e.g. Normal).
  - **Step 3**: Randomly estimate the parameters of the *K* distributions.
- **Iterate,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

K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.

Image courtesy of Andrew Moore, Carnegie Mellon U.



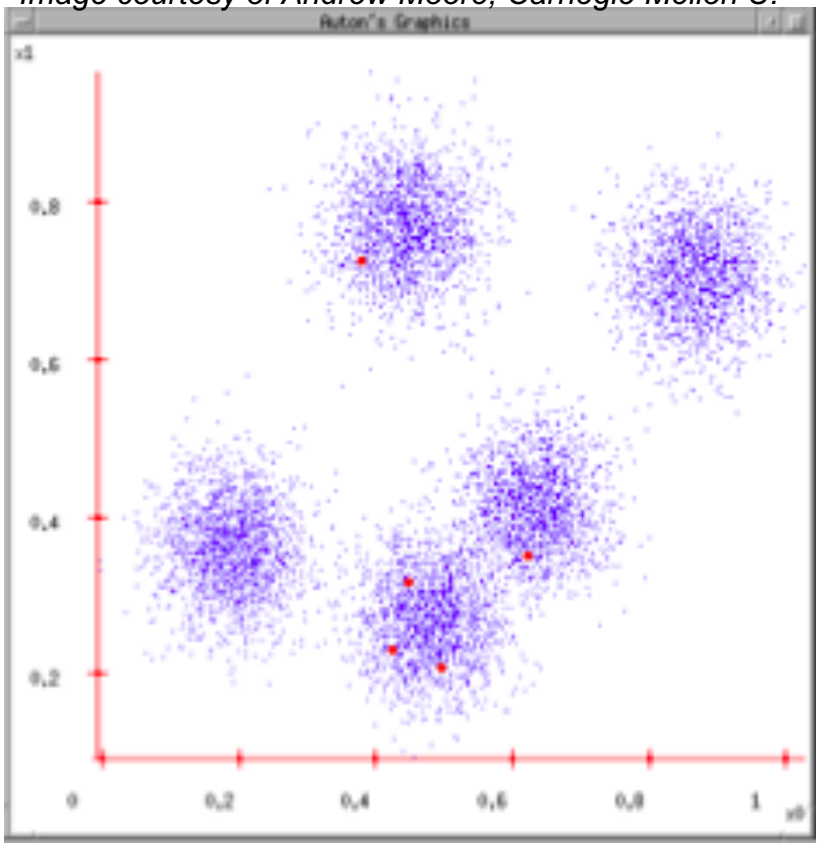
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:

   $\{\mu_1, \ldots, \mu_k\}$ (assume σ^2 is known).

Image courtesy of Andrew Moore, Carnegie Mellon U.



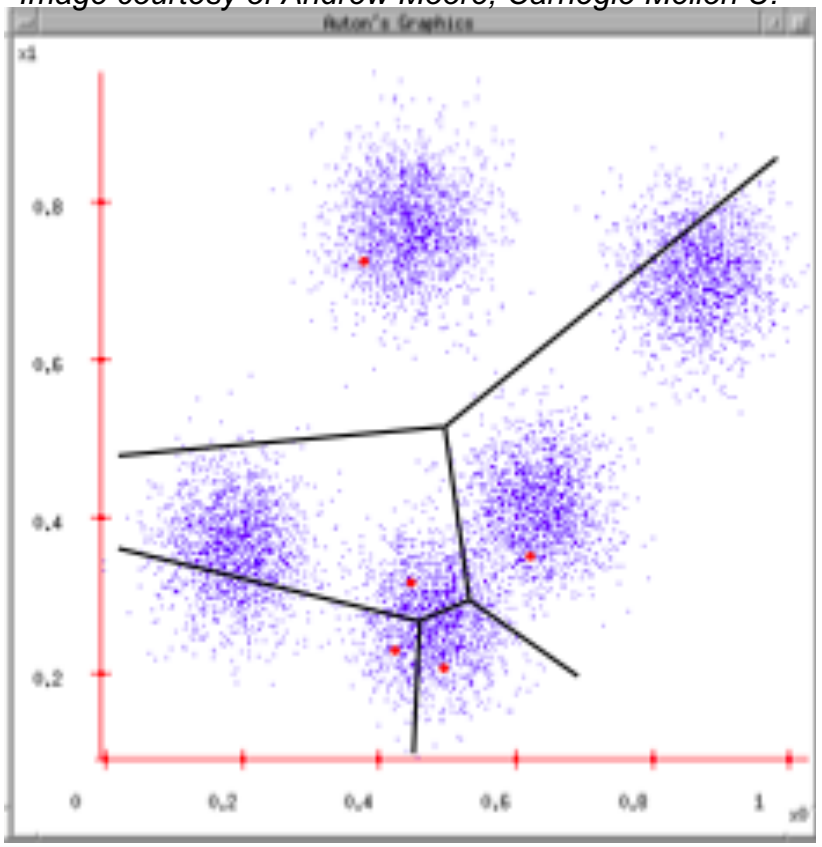
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:

   $\{\mu_1, \ldots, \mu_k\}$ (assume σ^2 is known).
3. Assign each data point to the center.

Image courtesy of Andrew Moore, Carnegie Mellon U.



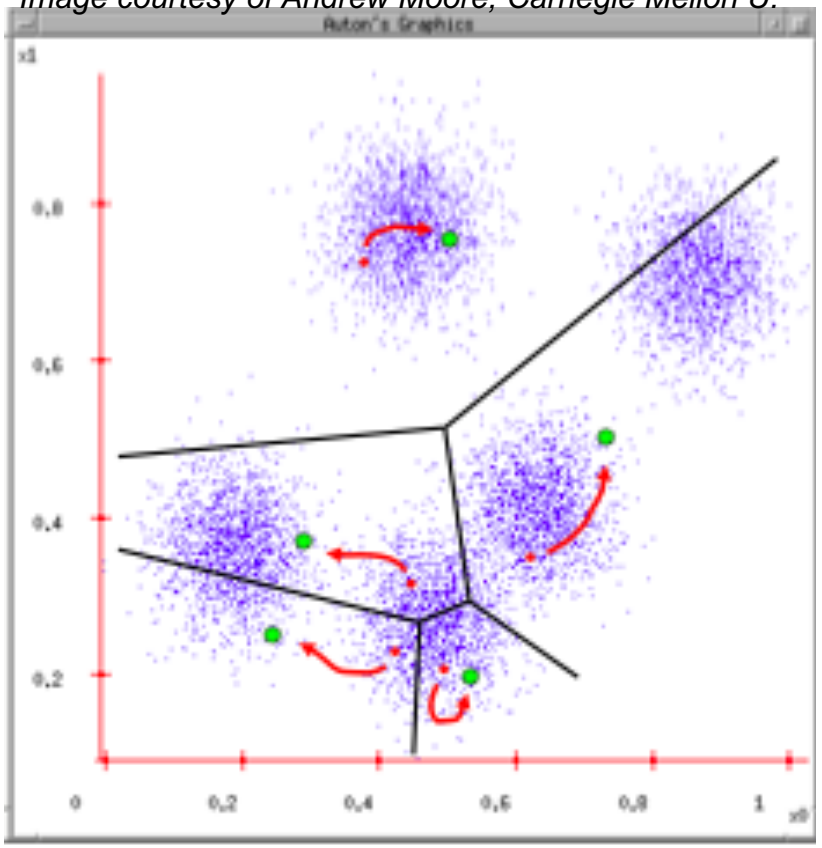
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:

   $\{\mu_1, \ldots, \mu_k\}$ (assume σ^2 is known).
3. Assign each data point to the center.
4. Each center finds the centroid of the points it owns.

Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm

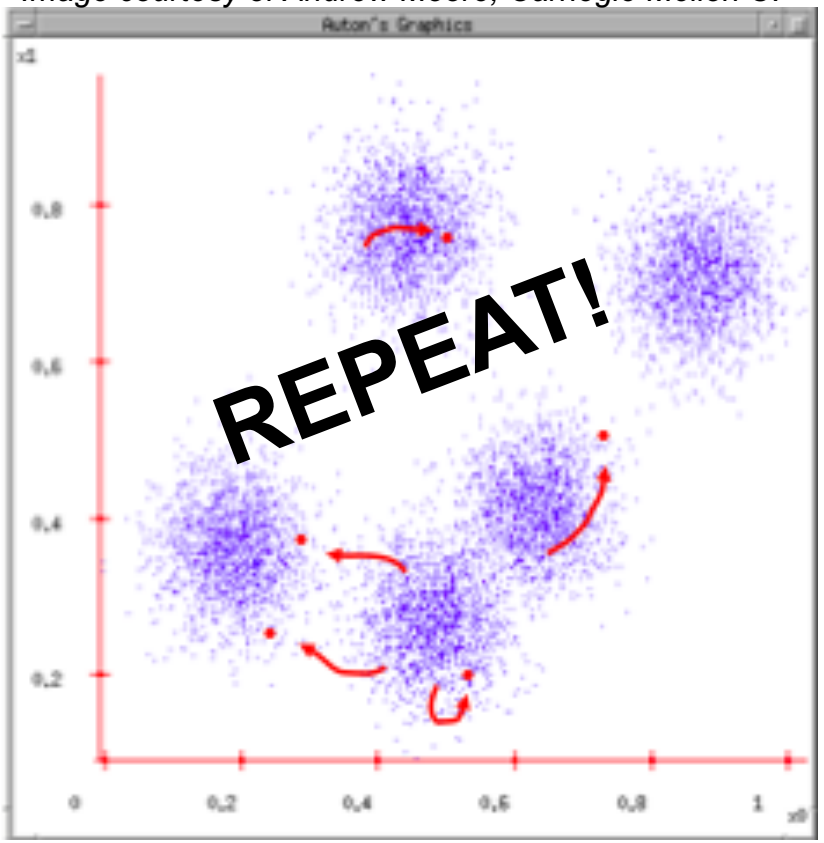
This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:

   $\{\mu_1, \ldots, \mu_k\}$ (assume σ^2 is known).
3. Assign each data point to the center.
4. Each center finds the centroid of the points it owns…

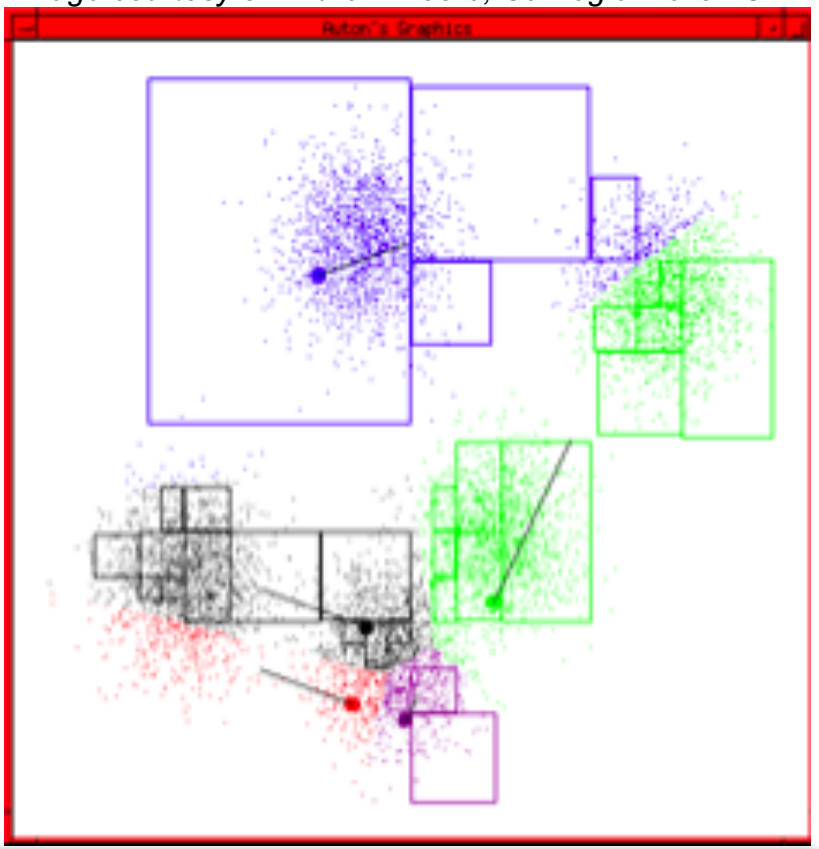
   and jumps there.

Image courtesy of Andrew Moore, Carnegie Mellon U.



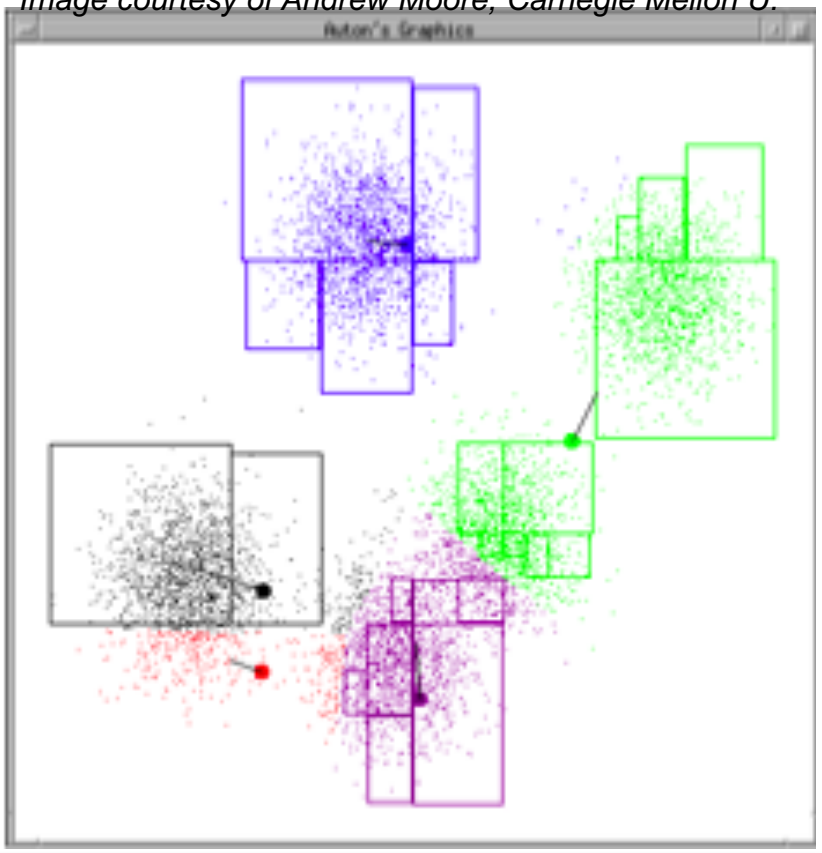
K-means algorithm starts

Image courtesy of Andrew Moore, Carnegie Mellon U.



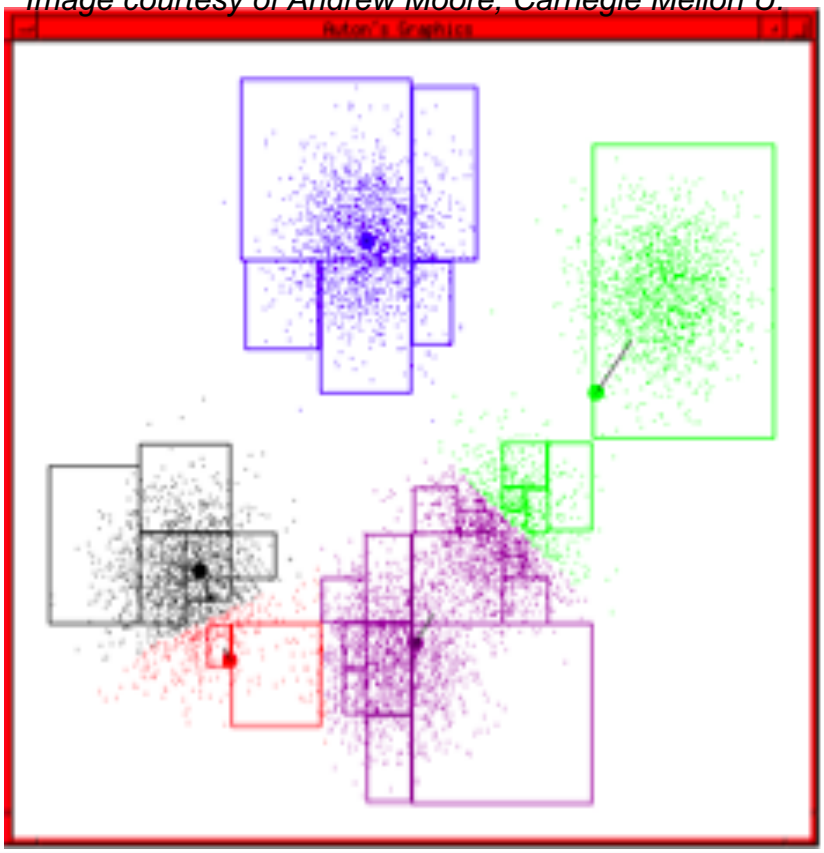
K-means algorithm continues (2)

Image courtesy of Andrew Moore, Carnegie Mellon U.



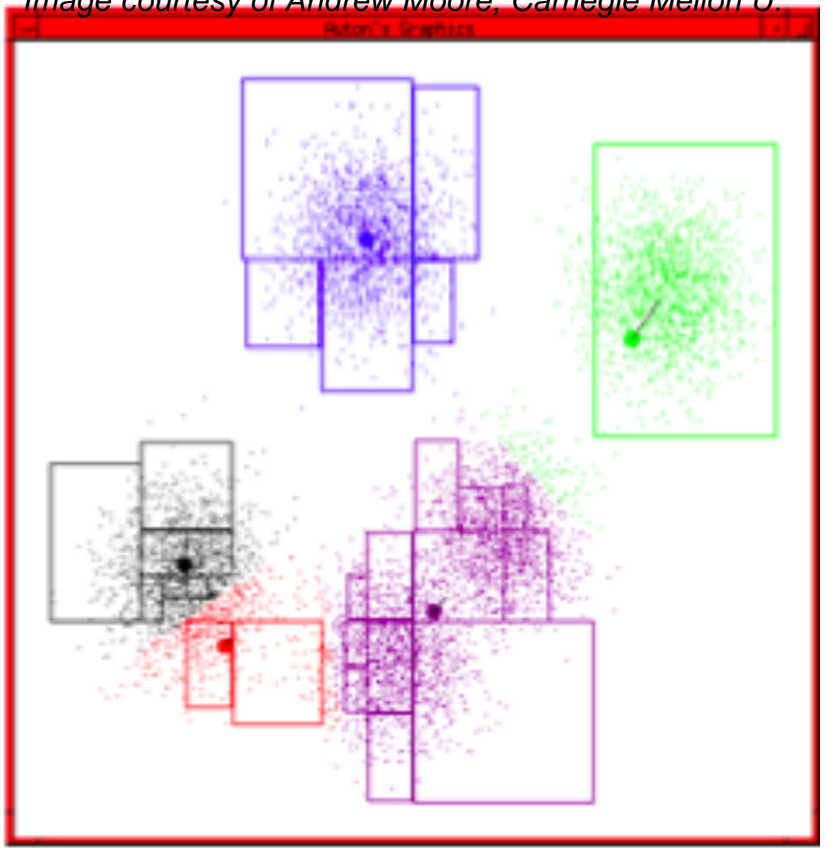
K-means algorithm continues (3)

Image courtesy of Andrew Moore, Carnegie Mellon U.



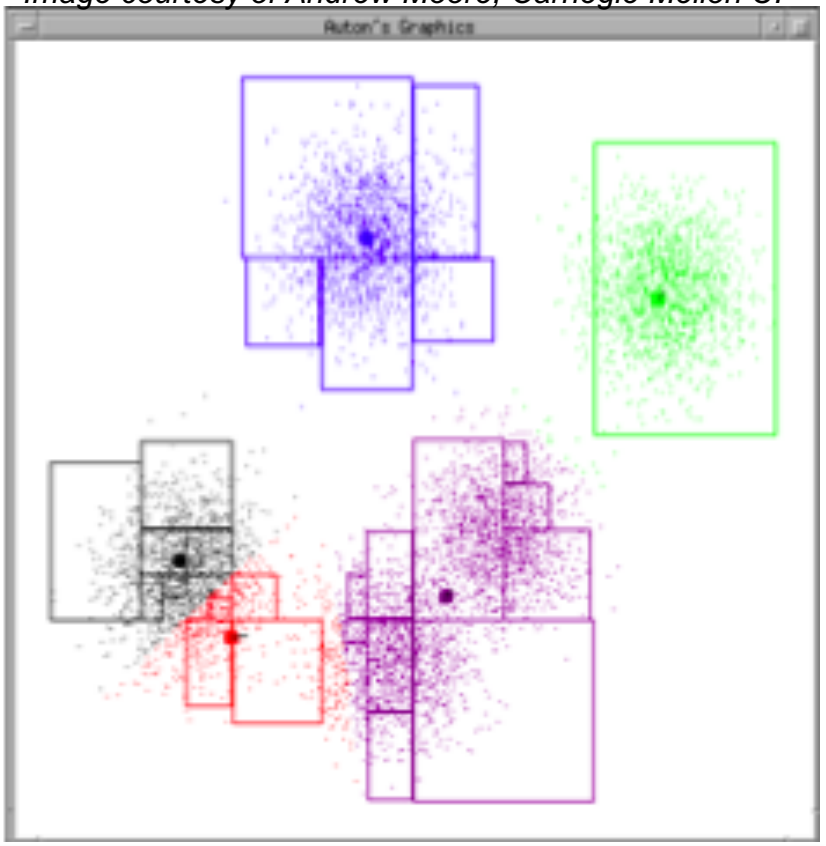
K-means algorithm continues (4)

Image courtesy of Andrew Moore, Carnegie Mellon U.



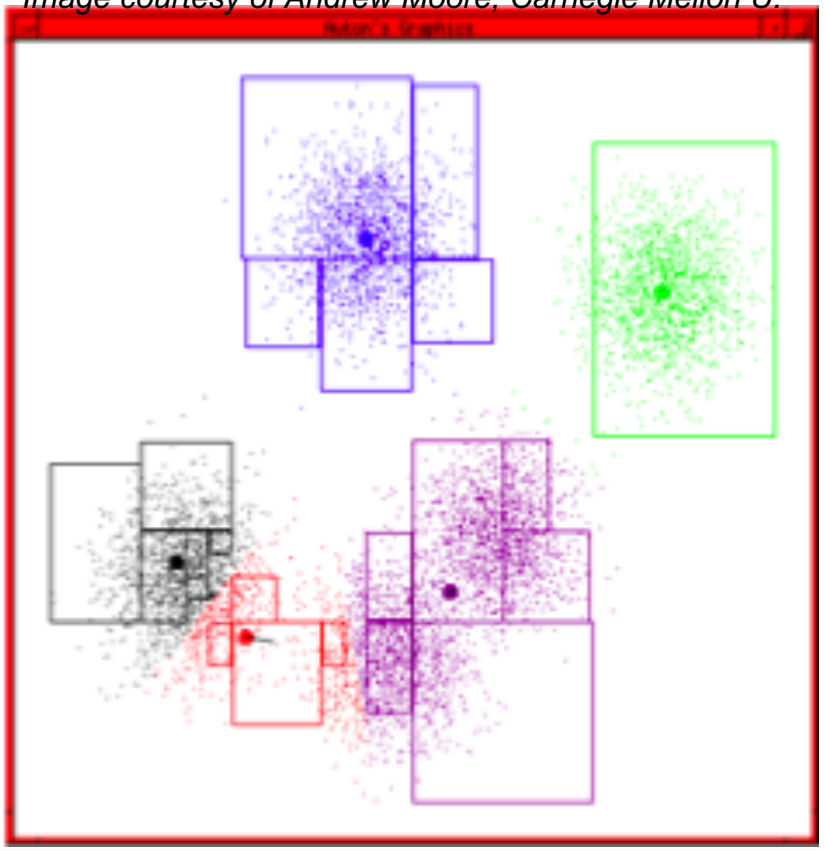
K-means algorithm continues (5)

Image courtesy of Andrew Moore, Carnegie Mellon U.



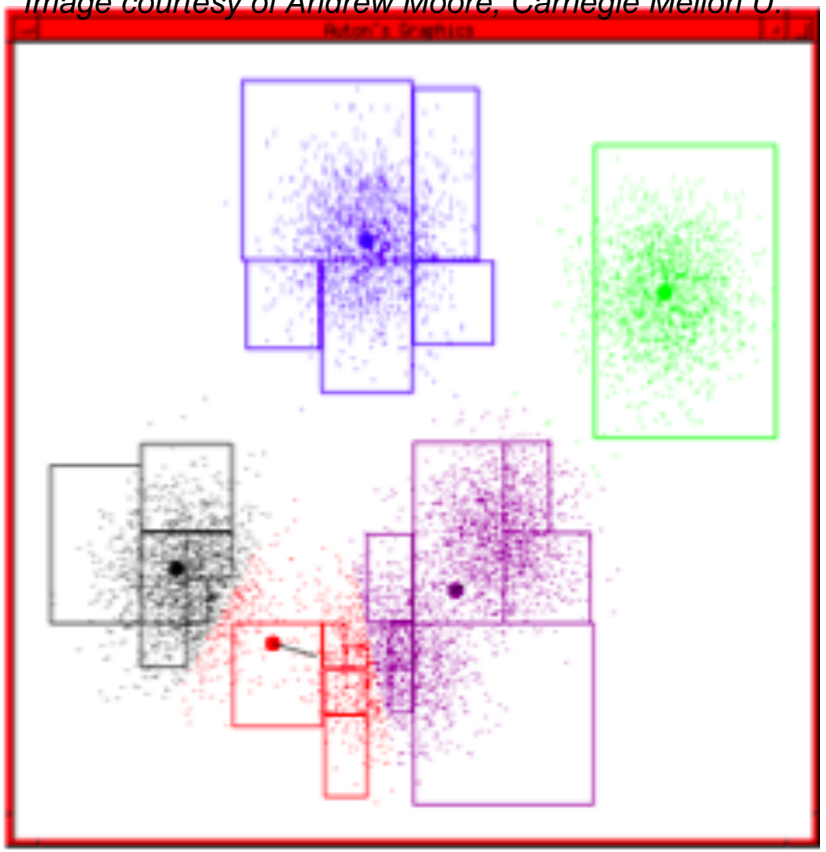
K-means algorithm continues (6)

Image courtesy of Andrew Moore, Carnegie Mellon U.



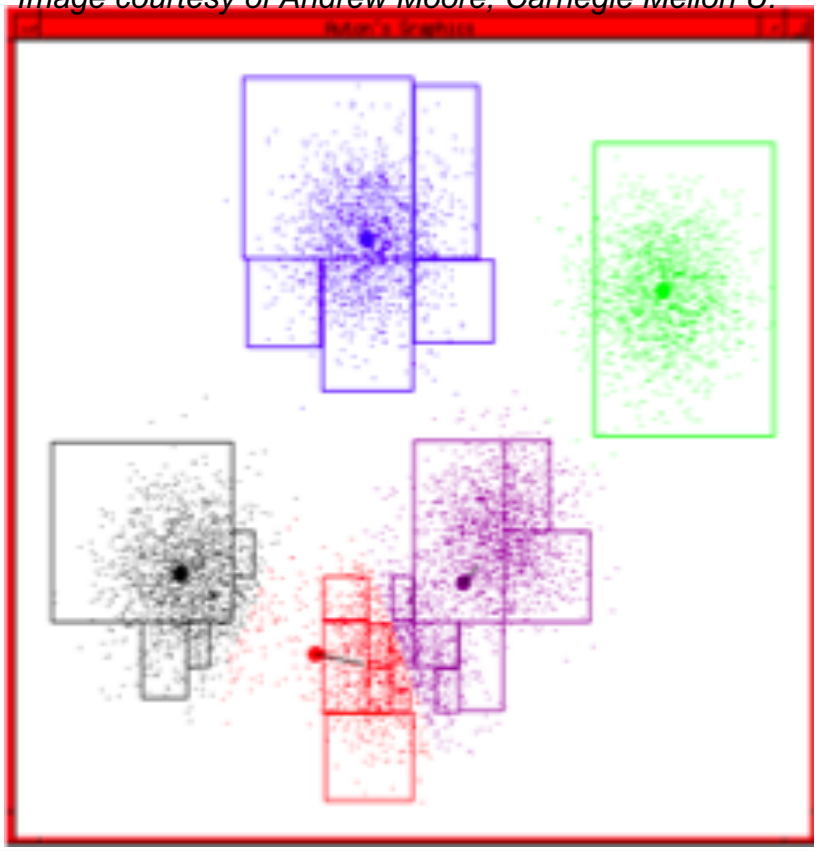
K-means algorithm continues (7)

Image courtesy of Andrew Moore, Carnegie Mellon U.



K-means algorithm continues (8)

Image courtesy of Andrew Moore, Carnegie Mellon U.

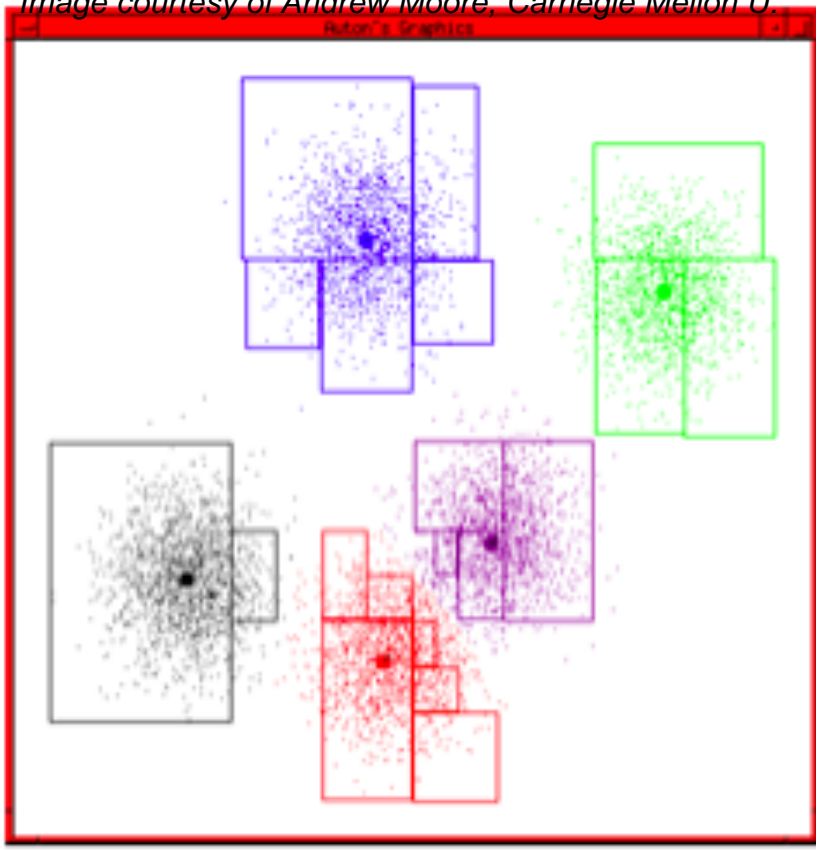


K-means algorithm continues (9)

Image courtesy of Andrew Moore, Carnegie Mellon U.

K-means algorithm terminates

Image courtesy of Andrew Moore, Carnegie Mellon U.



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

Properties of K-means

- **Optimality**?

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

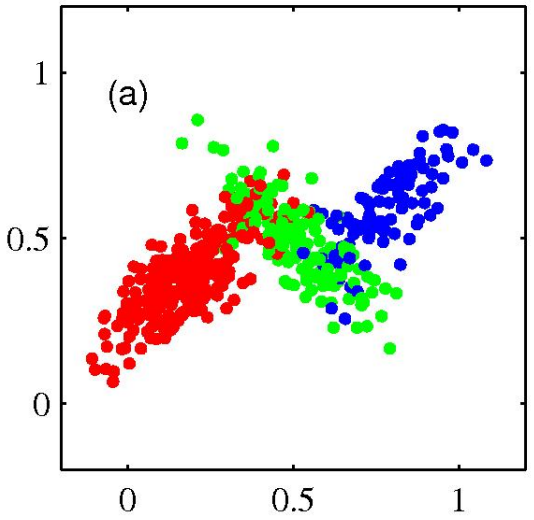
Properties of K-means

- **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: the good and the bad

- Good:
  - We realize that maximizing parameters is easy once we have assignments
- Bad:
  - What about points that are about equally far to two clusters?
  - We can only update the mean (not variance)
  - We have to assume equal variance between clusters

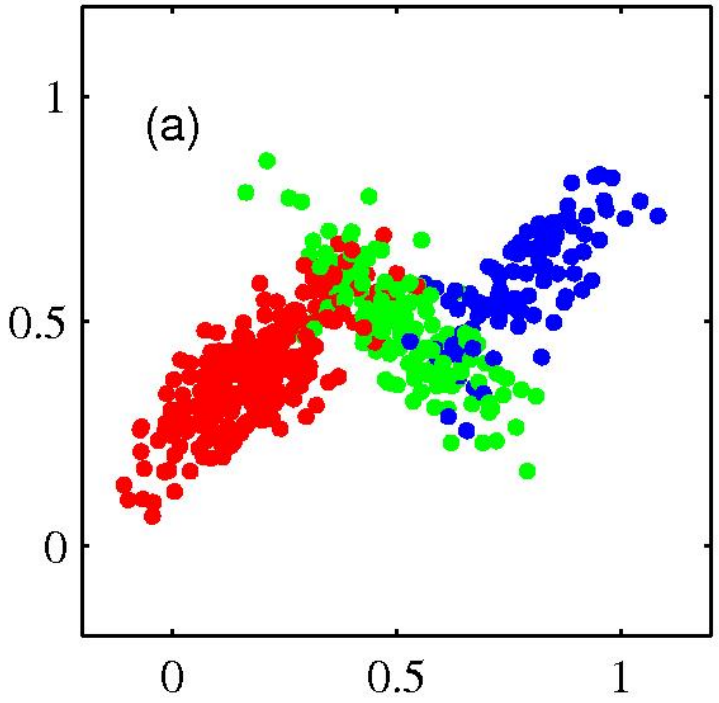


Copyright C.M. Bishop, PRML

Questions about K-means?

Beyond K-means

- How to fit data where variance is unknown or non-identical between clusters?



Copyright C.M. Bishop, PRML

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.

- Determining $p(z|x)$ is easy once we know parameters $p(z_k)$, μ_k, σ_k^2 (Bayes' rule)

Gaussian **Mixture** Model

- **Maximum likelihood** often gives a good parameter estimate

$$p(X|\theta) = \sum_{Z} p(X, Z|\boldsymbol{\theta})$$

- **Why is it hard here?**

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 knew all variables (e.g. cluster assignments), we could easily maximize the likelihood.
  - With unobserved variables, 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.

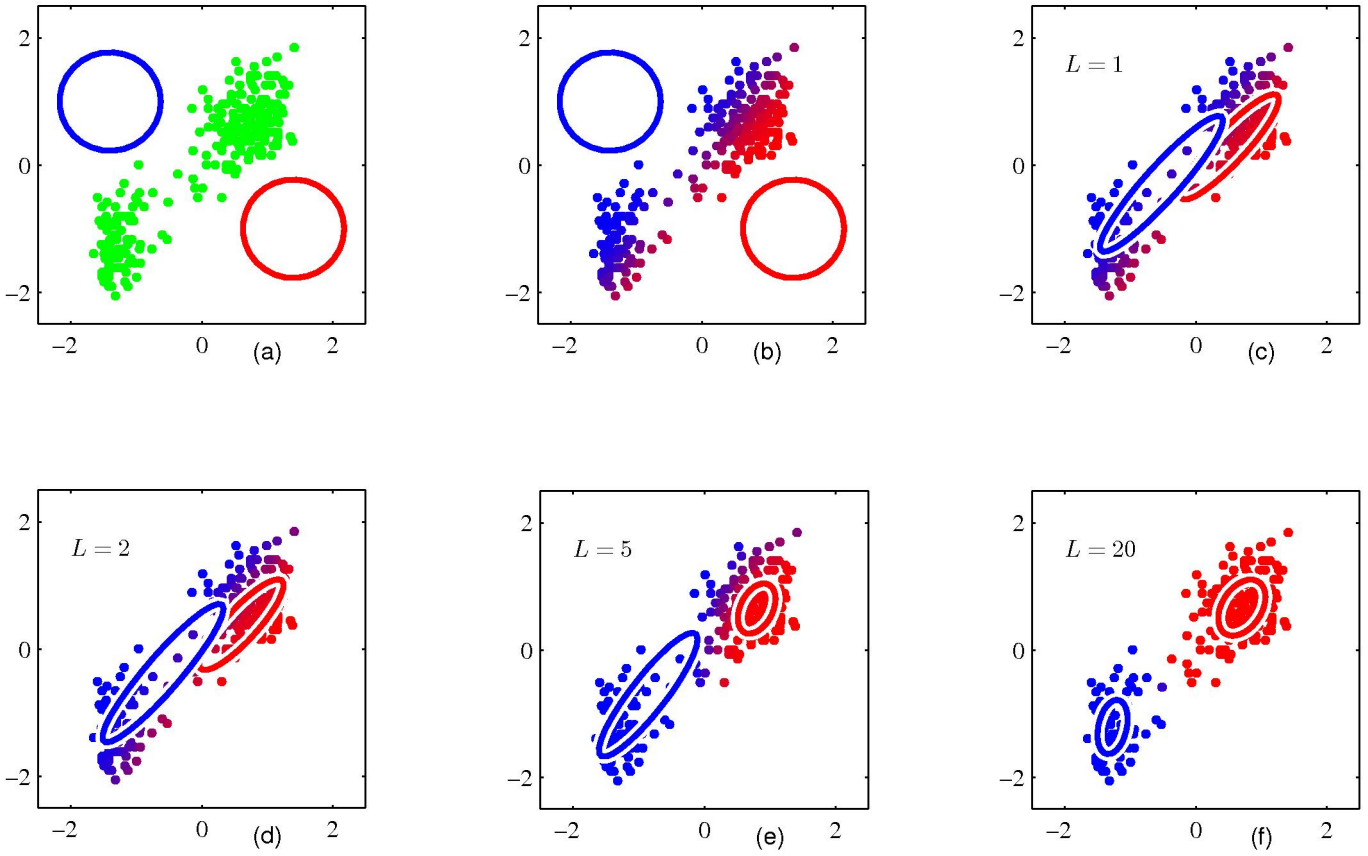
EM for clustering

- 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 initialize the parameters of the *K* distributions.
- **Iterate, until convergence**:
  - **Step 4**: Assign responsibility for instances to classes based on the current parametric distributions. **Soft assignment**

$$\gamma_i^j = p(z_i | x_j, \theta_{\text{old}})$$

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

EM for clustering



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.

$$Q(\theta, \theta_{\text{old}}) = \sum_{Z} p(Z|X, \theta_{\text{old}}) \log p(X, Z|\theta)$$

  - Maximization (M) step: Compute the maximum likelihood parameter setting based on the completed data.

$$\theta_{\text{new}} = \arg\max_{\theta} Q(\theta, \theta_{\text{old}})$$

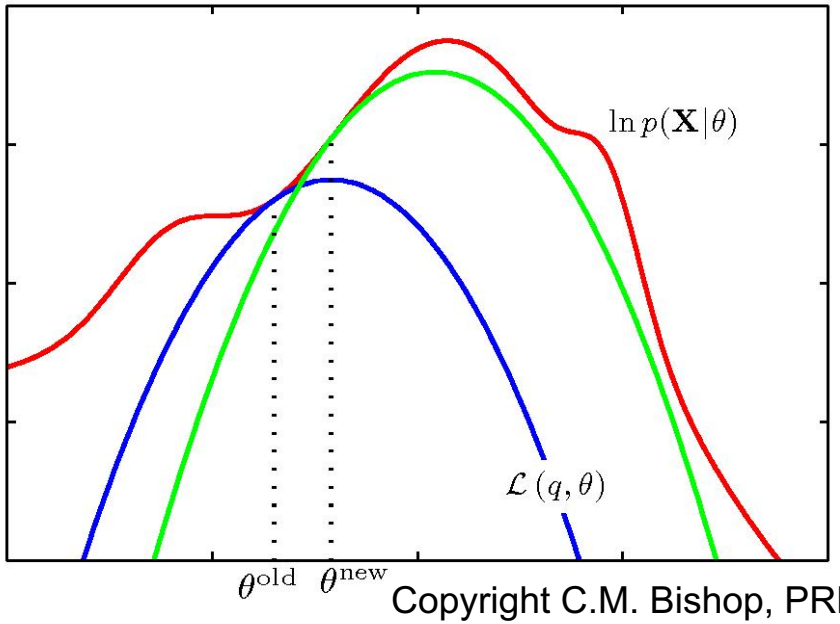
Once the data is completed (E-step), computing the log-likelihood and new parameters (M-step) is easy! This is what we did for K-means.

Why does it work?

- Instead of $p(X|\theta)$, we maximize

$$Q(\theta, \theta_{\text{old}}) = \sum_{Z} p(Z|X, \theta_{\text{old}}) \log p(X, Z|\theta)$$

- Original objective still improves if:
  - Maximum of Q is also the maximum of lower bound
  - Lower bound is exact at θ_{old}
  - (Not at local maximum)



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!

K-means or EM?

- K-means can be seen as a specific case of EM

  (where variance is fixed to a value that decreases to 0)
- K-means tends to converge faster
- EM can deal with unknown or non-identical variance
- K-means sometimes used to initialize EM

Anomaly detection

Anomaly detection

- Discriminative approaches tend to be ineffective when one class is much more rare than the other.

Anomaly detection

- Discriminative approaches tend to be ineffective when one class 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

http://opencourseonline.com/400/coursera-open-course-stanford-university-machine-learning-video-playlist-15-anomaly-detection

Anomaly detection vs Supervised learning

Anomaly detection

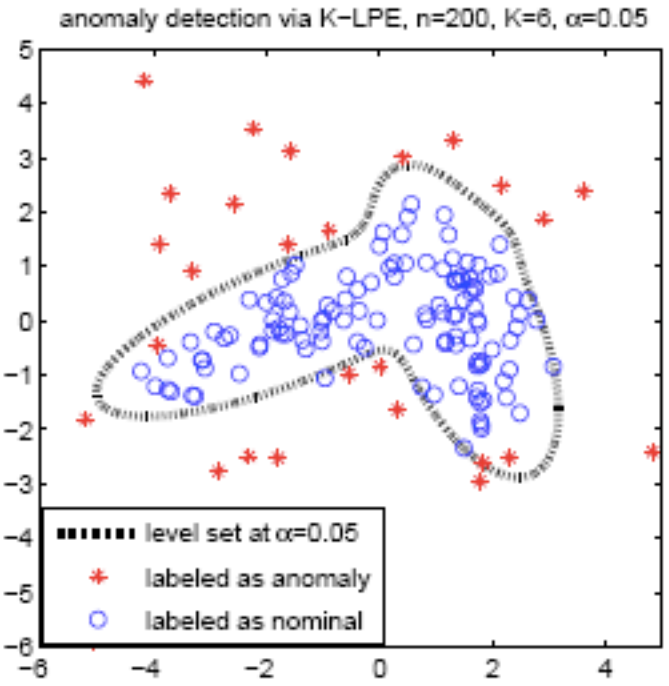
- Small number of positive examples (e.g. <10).
- Large number of negative examples (e.g. >100).
- Many different "types" of anomalies, so don't want to fit a model for the positive class.

Supervised learning

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

http://opencourseonline.com/400/coursera-open-course-stanford-university-machine-learning-video-playlist-15-anomaly-detection

A simple example

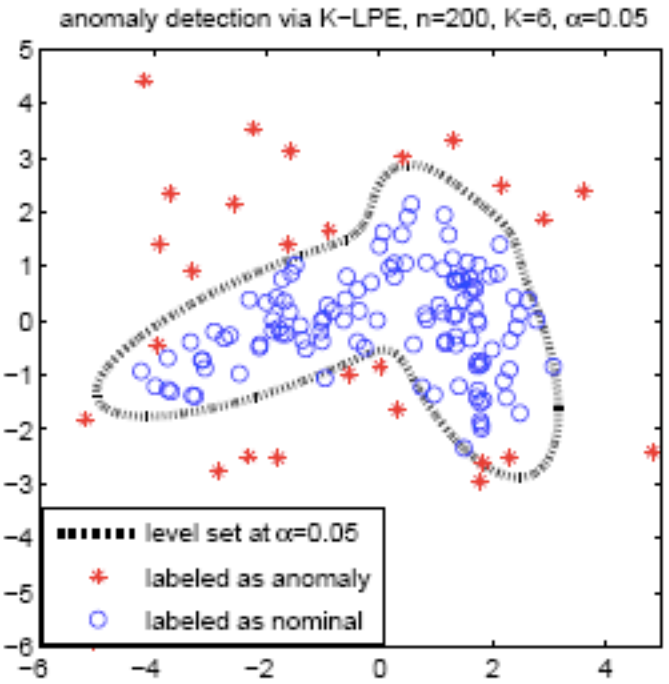


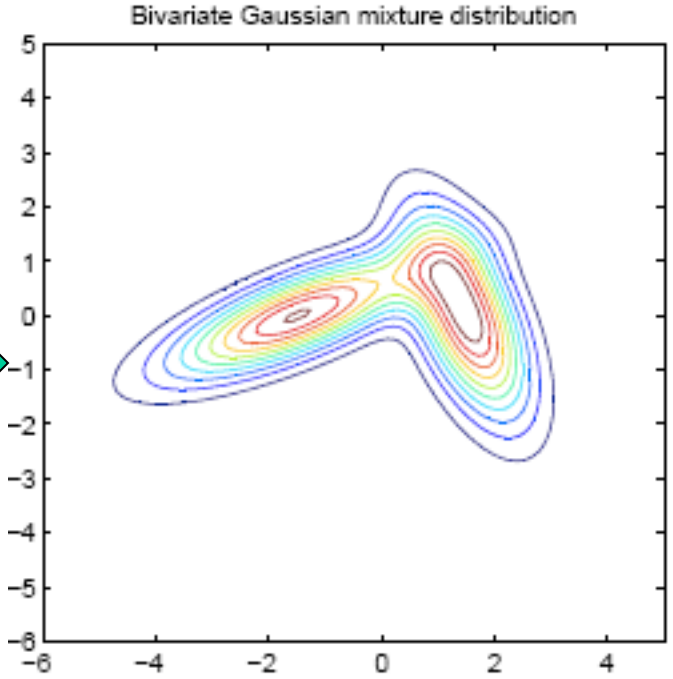
Does the distribution of nominal data look familiar?

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

Another GMM!



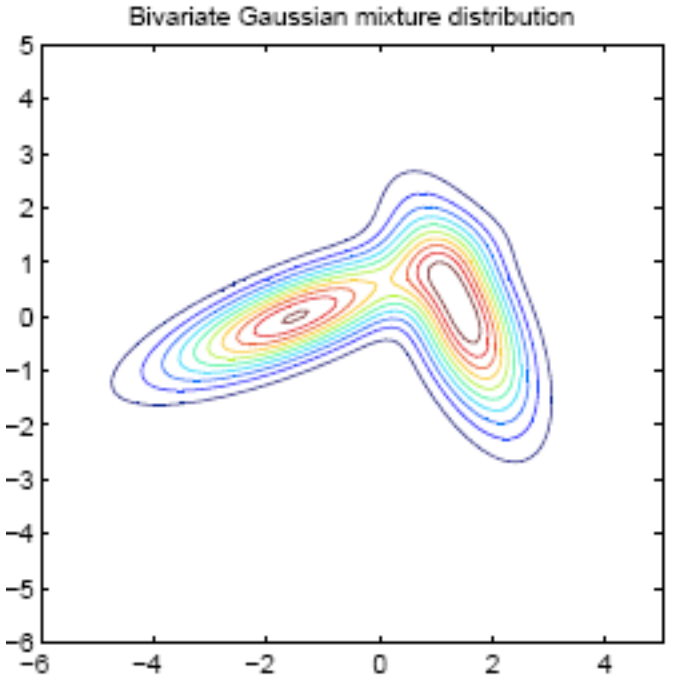


From: *M. Zhao and V. Saligrama, "Anomaly Detection with Score functions based on Nearest Neighbor Graphs", Neural Information Processing Systems (NIPS) Conference, 2009*

A simple example

- GMM can be fit again with EM
- Note that before we were mainly interested in the **cluster assignments** (which items go together)
- Here we are interested in the **final density**

Another GMM!



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

Anomaly detection

- Discriminative approaches tend to be ineffective when one class is much more rare than the other.
- A simple **generative** approach:
  - Fit a model, $p(x)$ using the input data **(using a GMM?)**.
  - Set a decision threshold ε and predict $Y=$ {*1 if* $p(x)>\varepsilon$*, 0 otherwise*}.
  - Use a validation set to measure performance (can use cross-validation to set ε).
- Note: GMM is used here to model the **nominal data only.** We **don't** attempt to model the anomalous data (see above)

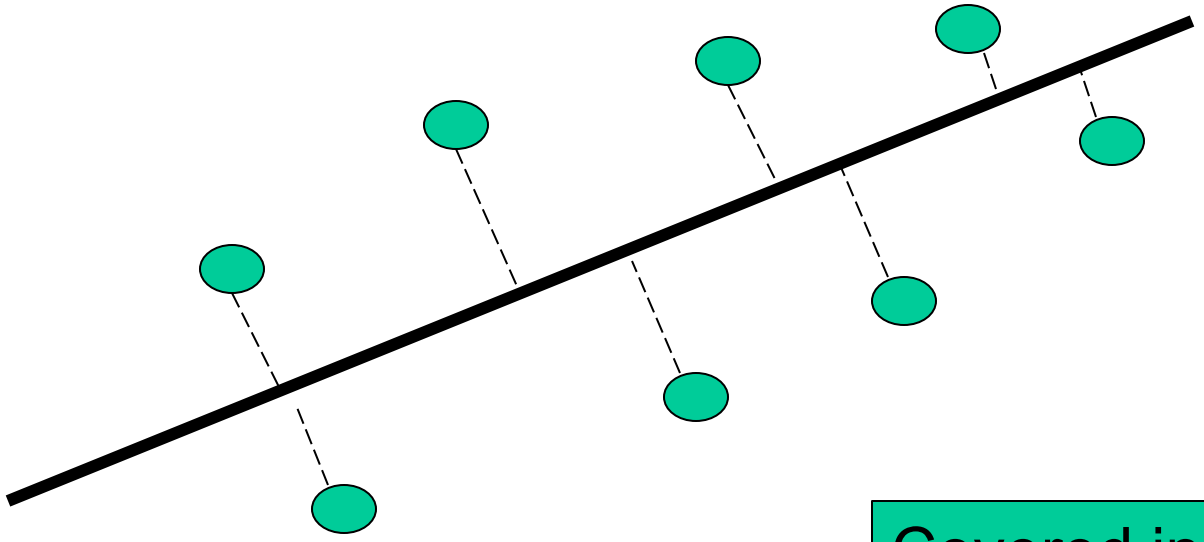
Practical issues

- Need $p(x)$ to be **low** for anomalous examples.
- Apply techniques for construction/selection of features to achieve this.
- Need a validation set to select features and learning parameters.

Dimensionality reduction

- Given points in an *m*-dimensional space (for large *m*), project to a low dimensional space while preserving trends in the data.

- Principal Components Analysis

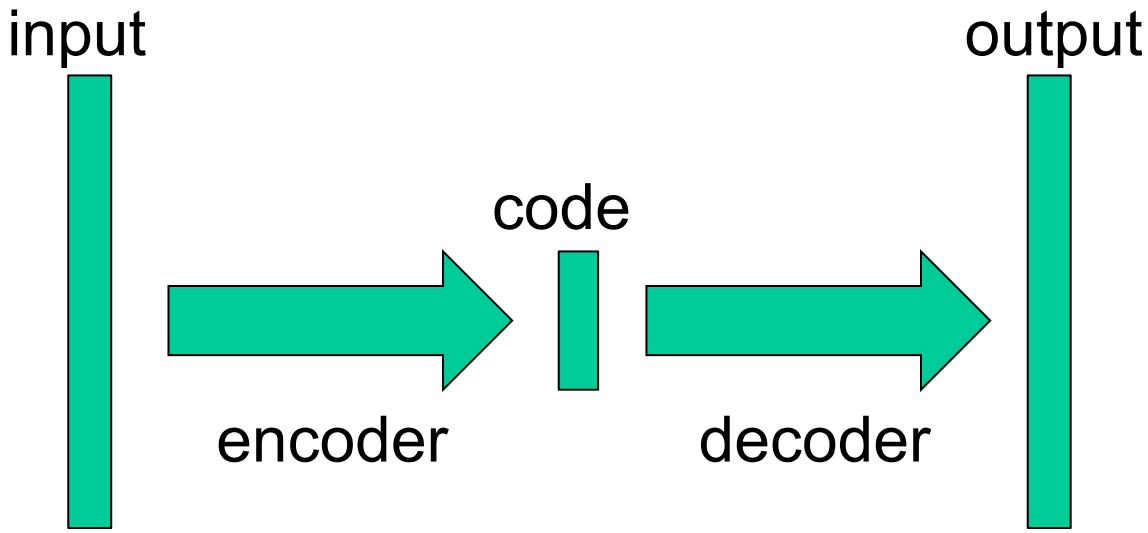


Covered in detail in Lecture 9!

Dimensionality reduction

- Learn neural networks to perform dimensionality reduction:

  auto-encoding



- Objective: recover input as output

Covered in detail in neural network lecture

Autoregressive models for time series

- The problem:
  - Given a time series: $X = \{x_1, x_2, \ldots, x_T\}$
  - Predict x_t from $x_{1:t-1}$.

Autoregressive models for time series

- The problem:
  - Given a time series: $X = \{x_1, x_2, \ldots, x_T\}$
  - Predict x_t from $x_{1:t-1}$.

- A simple autoregressive (AR) model:

  $X_t = w_0 + \varepsilon + \sum_{i=1:p} w_i x_{t-i} + \varepsilon_t$

  where w_i are the parameters and ε_t is white noise.

Autoregressive models for time series

- The problem:
  - Given a time series: $X = \{x_1, x_2, \ldots, x_T\}$
  - Predict x_t from $x_{1:t-1}$.

- A simple autoregressive (AR) model:

  $X_t = w_0 + \varepsilon + \sum_{i=1:p} w_i x_{t-i} + \varepsilon_t$

  where w_i are the parameters and ε_t is white noise.

- A more general model, autoregressive-moving average (ARMA):

  $X_t = w_0 + \varepsilon + \sum_{i=1:p} w_i x_{t-i} + \sum_{i=1:q} \theta_i \varepsilon_{t-i}$

  where w_i θ_i are the parameters, and $\varepsilon_t \sim N(0,\sigma^2)$ are assumed to be iid samples from a normal distribution.

What you should know

- The general form of the unsupervised learning problem
- Basic functioning and properties of useful algorithms:
  - K-means
  - Expectation-maximization
- A useful model
  - Gaussian mixture models
- Characteristics of common problems:
  - clustering, anomaly detection, dimensionality reduction, autoregression, autoencoding

Hierarchical clustering

- A hierarchy of clusters, where the cluster at each level are created by merging clusters from the next lower level.

Hierarchical clustering

- A hierarchy of clusters, where the cluster at each level are created by merging clusters from the next lower level.

- **Two general approaches:**
  - Recursively merge a pair of clusters.
  - Recursively split the existing clusters.

Hierarchical clustering

- A hierarchy of clusters, where the cluster at each level are created by merging clusters from the next lower level.

- **Two general approaches:**
  - Recursively merge a pair of clusters.
  - Recursively split the existing clusters.

- Use dissimilarity measure to select 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.

Hierarchical clustering of news articles

http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html