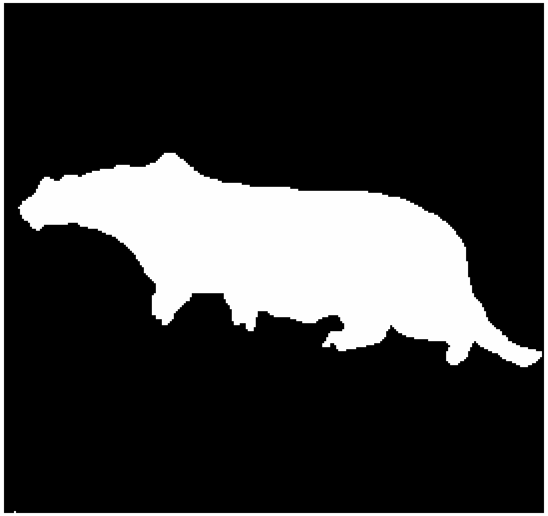
Mixture density estimation

Nuno Vasconcelos

ECE Department, UCSD

Recall

- last class, we will have "Cheetah Day"
- what:
  - 4 teams, average of 6 people
  - each team will write a report on the 4 cheetah problems
  - each team will give a presentation on one of the problems
- I am waiting to hear on the teams

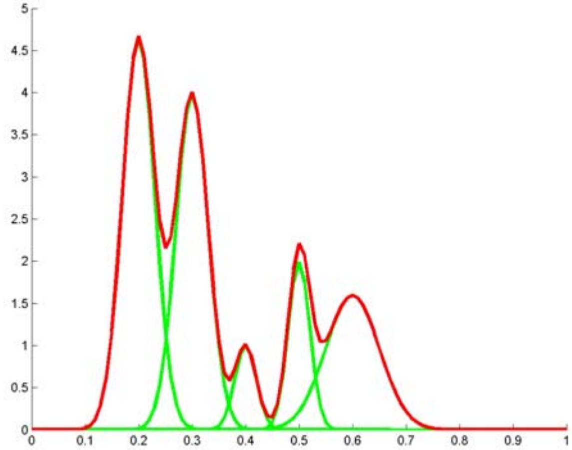


Plan for today

- last time we started talking about mixture models
- we introduced the basics of EM
- today to motivate EM:
  - "classification-maximization"
  - which is a general case of "K-means"
- we will then
  - introduce EM
  - solve EM for the case of learning Gaussian mixtures
- next class:
  - proof that EM maximizes likelihood of incomplete data

Mixture density estimate

- we have seen that EM is a framework for ML estimation with missing data
- canonical example:
  - want to classify vehicles into commercial/private
  - X: vehicle weight
  - multimodal density because there is a hidden variable Z (type of car)



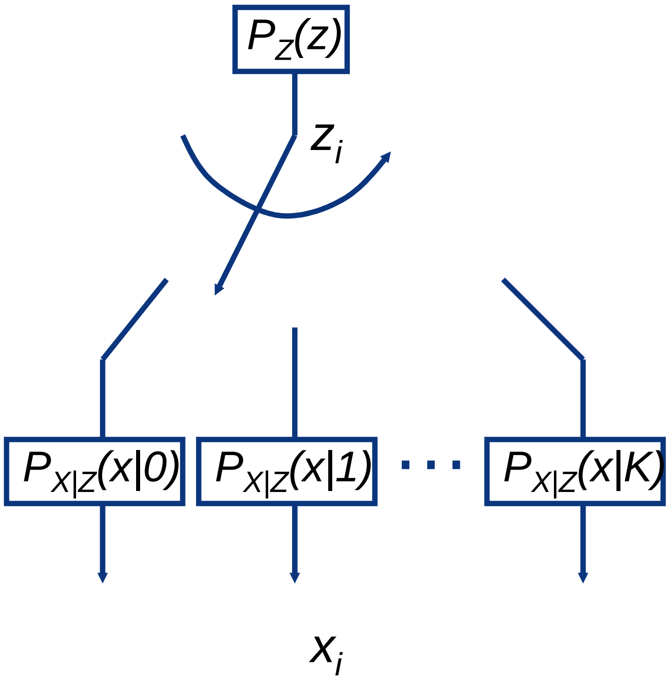
z in {compact, sedan, station wagon, pick up, van}

  - for a given car type the weight is approximately Gaussian (or has some other parametric form)
  - the density is a "mixture of Gaussians"

mixture model

- two types of random variables
  - Z – hidden state variable
  - X – observed variable
- observations sampled with a two-step procedure
  - a state (class) is sampled from the distribution of the hidden variable

$$P_Z(z) \rightarrow z_i$$



  - an observation is drawn from the class conditional density for the selected state

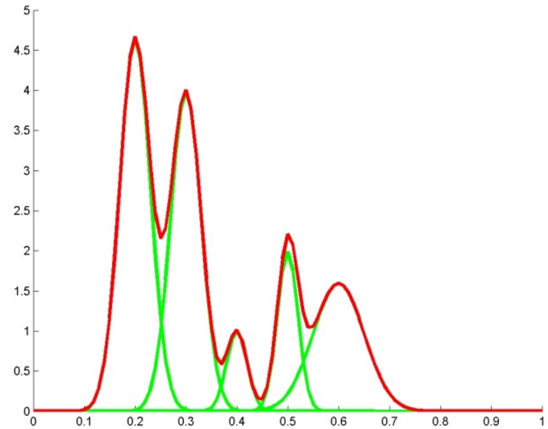
$$P_{X|Z}(x|z_i) \rightarrow x_i$$

mixture model

- the sample consists of pairs (x_i, z_i)

  $$D = \{(x_1, z_1), \ldots, (x_n, z_n)\}$$

  but we never get to see the z_i



- e.g. bridge example:
  - sensor only registers weight
  - the car class was certainly there, but it is lost by the sensor
  - for this reason Z is called hidden
- the pdf of the observed data is

$$P_{\mathbf{X}}(\mathbf{x}) = \sum_{c=1}^{C} P_{\mathbf{X}|Z}(\mathbf{x}|c) P_Z(c)$$

$$= \sum_{c=1}^{C} P_{\mathbf{X}|Z}(\mathbf{x}|c) \pi_c$$

\# of mixture components

component "weight"

c^{th} "mixture component"

The basics of EM

- as usual, we start from an iid sample $D = \{x_1,\ldots,x_N\}$
- goal is to find parameters Ψ^* that maximize likelihood with respect to D

$$
\begin{aligned}
\Psi^* &= \arg\max_{\Psi} P_{\mathbf{X}}(\mathcal{D};\Psi) \\
&= \arg\max_{\Psi} \int P_{\mathbf{X}|Z}(\mathcal{D}|z;\Psi)P_Z(z;\Psi)dz
\end{aligned}
$$

- the set

$$D_c = \{(x_1,z_1), \ldots, (x_N,z_N)\}$$

is called the complete data

- the set

$$D = \{x_1, \ldots, x_N\}$$

is called the incomplete data

Complete vs incomplete data

- in general, the problem would be trivial if we had access to the complete data
- we have illustrated this with the specific example of
  - Gaussian mixture of C components
  - parameters $\Psi = \{(\pi_1, \mu_1, \Sigma_1), \ldots, (\pi_C, \mu_C, \Sigma_C)\}$
- and shown that,
  - given the complete data D_c, we only need to split the training set according to the labels z_i

    $D^1 = \{x_i | z_i = 1\}$, $D^2 = \{x_i | z_i = 2\}$, … , $D^C = \{x_i | z_i = C\}$
  - and solve, for each c,

$$(\pi_c^\star, \mu_c^\star, \Sigma_c^\star) = \arg\max_{\pi,\mu,\Sigma} \mathcal{G}(\mathcal{D}^c, \mu, \Sigma)\pi$$

Learning with complete data

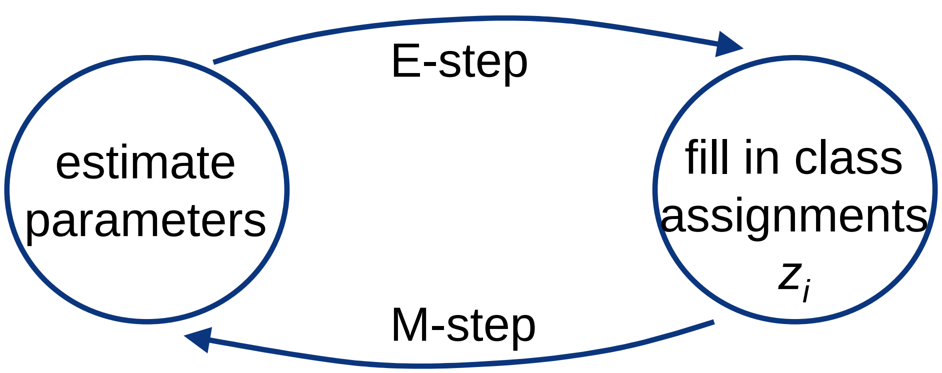
- the solution is

$$
\begin{aligned}
\pi_c^\star &= \frac{|\{\mathbf{x}_i \in \mathcal{D}^c\}|}{N} \\
\mu_c^\star &= \frac{1}{|\{\mathbf{x}_i \in \mathcal{D}^c\}|} \sum_{i|\mathbf{x}_i \in \mathcal{D}^c} \mathbf{x}_i \\
\Sigma_c^\star &= \frac{1}{|\{\mathbf{x}_i \in \mathcal{D}^c\}|} \sum_{i|\mathbf{x}_i \in \mathcal{D}^c} (\mathbf{x}_i - \mu_c^\star)(\mathbf{x}_i - \mu_c^\star)^T
\end{aligned}
$$

- hence, all the hard work seems to be in figuring out what the z_i are
- the EM algorithm does this iteratively

Learning with incomplete data (EM)

- the basic idea is quite simple
  1. start with an initial parameter estimate $\Psi^{(0)}$
  2. **E-step:** given current parameters $\Psi^{(i)}$ and observations in D, "guess" what the values of the z_i are
  3. **M-step:** with the new z_i, we have a complete data problem, solve this problem for the parameters, i.e. compute $\Psi^{(i+1)}$
  4. go to 2.
- this can be summarized as



Classification-maximization

- the question is how do we get the z_i in the E-step?
- we will look at this soon, when we derive EM
- for now let's start with a simpler algorithm, that I would call "Classification-Maximization"
- the idea is the following
  - after the M-step we have an estimate of all the parameters, i.e. an estimate for the densities that compose the mixture
  - we want to find the class-assignments z_i (recall that $z_i=k$ if x_i is a sample from the k^{th} component)
  - but this is a classification problem, and we know how to solve those: just use the BDR
- the steps are as follows

Classification-maximization

- C-step:
  - given estimates $\Psi^{(i)} = \{\Psi^{(i)}_1, \ldots, \Psi^{(i)}_C\}$
  - determine z_i by the BDR

$$z_l = \arg\max_c P_{\mathbf{X}|Z}\left(\mathbf{x}_l | c; \Psi_c^{(i)}\right) \pi_c^{(i)}, l \in \{1, \ldots, n\}$$

  - split the training set according to the labels z_i

$$D^1 = \{x_i | z_i = 1\}, \quad D^2 = \{x_i | z_i = 2\}, \quad \ldots \quad, D^C = \{x_i | z_i = C\}$$

- M-step:
  - as before, determine the parameters of each class independently

$$\Psi_c^{(i+1)} = \arg\max_{\Psi,\pi} P_{\mathbf{X}|Z}(\mathcal{D}^c | c, \Psi)\pi$$

For Gaussian mixtures

- C-step:
  - $$z_l = \arg\max_c \left\{ -\frac{1}{2}\left(\mathbf{x}_l - \mu_c^{(i)}\right)^T \left(\Sigma_c^{(i)}\right)^{-1} \left(\mathbf{x}_l - \mu_c^{(i)}\right) - \frac{1}{2} log \left|\Sigma_c^{(i)}\right| + \log \pi_c^{(i)} \right\}, l \in \{1, \ldots, n\}$$
  - split the training set according to the labels z_i

    $D^1 = \{x_i|z_i=1\}$, $D^2 = \{x_i|z_i=2\}$, … , $D^C = \{x_i|z_i=C\}$

- M-step:
  - $$\pi_c^{(i+1)} = \frac{|\{\mathbf{x}_i \in \mathcal{D}^c\}|}{n} \qquad \mu_c^{(i+1)} = \frac{1}{|\{\mathbf{x}_i \in \mathcal{D}^c\}|} \sum_{i|\mathbf{x}_i \in \mathcal{D}^c} \mathbf{x}_i$$

    $$\Sigma_c^{(i+1)} = \frac{1}{|\{\mathbf{x}_i \in \mathcal{D}^c\}|} \sum_{i|\mathbf{x}_i \in \mathcal{D}^c} \left(\mathbf{x}_i - \mu_c^{(i+1)}\right)\left(\mathbf{x}_i - \mu_c^{(i+1)}\right)^T$$

K-means

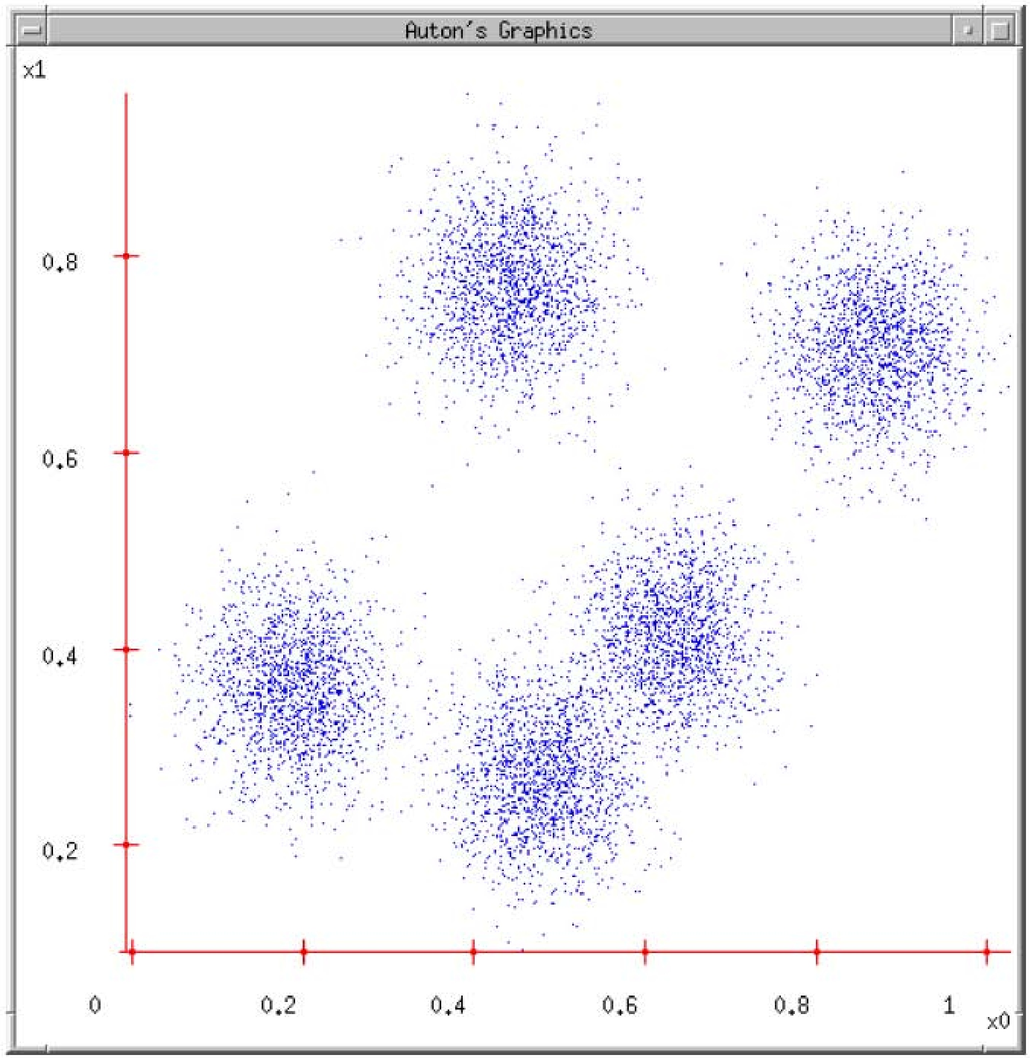
- when covariances are identity and priors uniform
- C-step:
  - $z_l = \arg\min_c ||\mathbf{x}_l - \mu_c^{(i)}||^2, \qquad l \in \{1, \ldots, n\}$
  - split the training set according to the labels z_i

    $D^1 = \{x_i|z_i=1\}$, $D^2 = \{x_i|z_i=2\}$, … , $D^C = \{x_i|z_i=C\}$
- M-step:
  - $\mu_c^{(i+1)} = \frac{1}{|\{\mathbf{x}_i \in \mathcal{D}^c\}|} \sum_{i|\mathbf{x}_i \in \mathcal{D}^c} \mathbf{x}_i$
- this is the K-means algorithm, aka generalized Loyd algorithm, aka LBG algorithm in the vector quantization literature:
  - “assign points to the closest mean; recompute the means”

K-means (thanks to Andrew Moore, CMU)

K-means

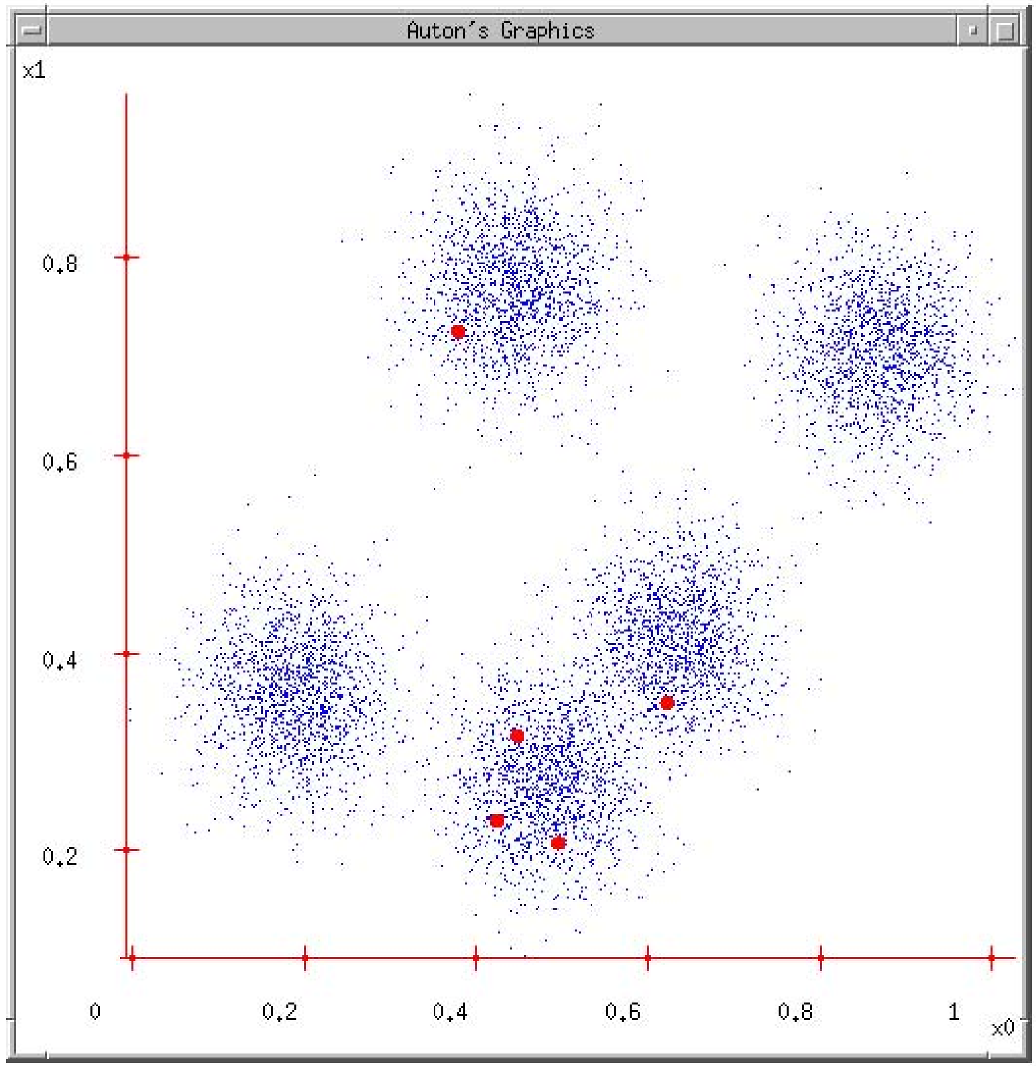
1. Ask user how many clusters they'd like. *(e.g. k=5)*



K-means (thanks to Andrew Moore, CMU)

K-means

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

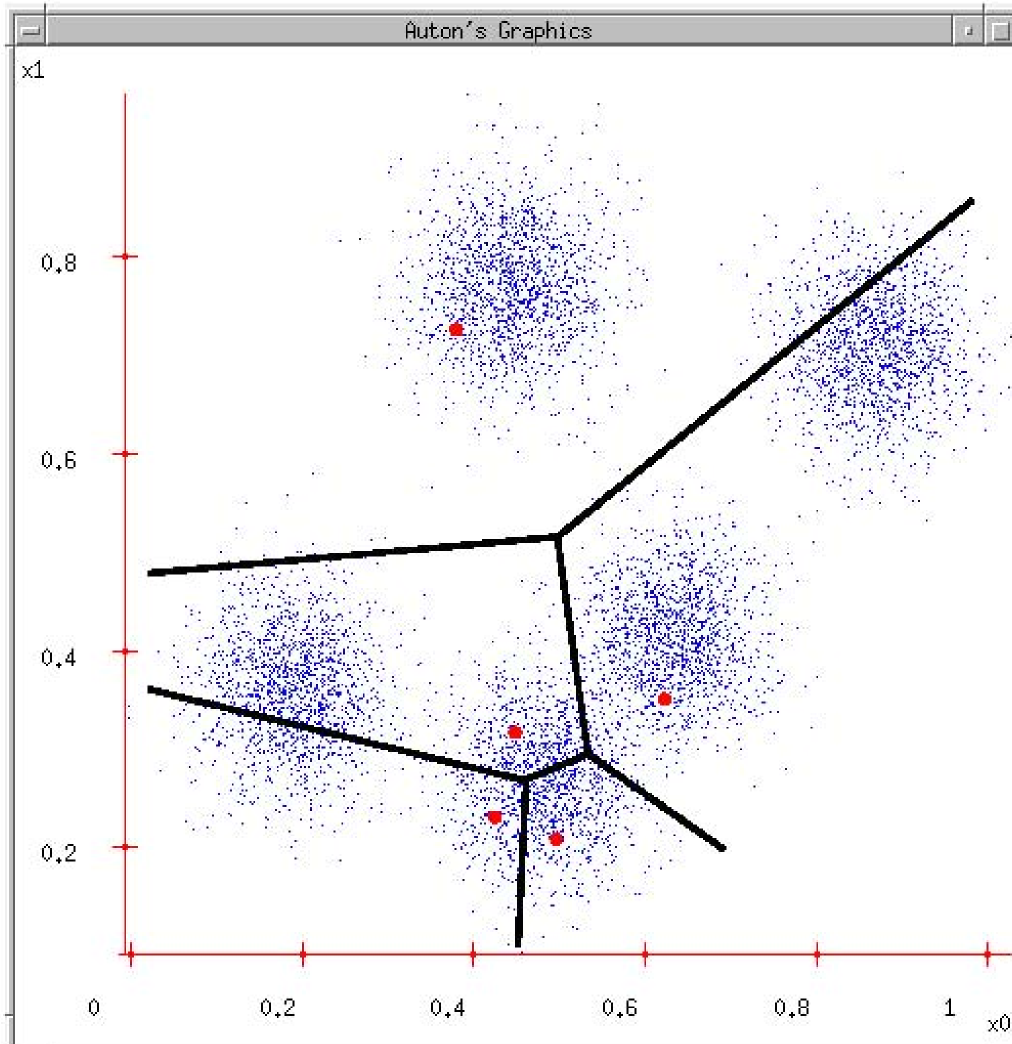


K-means and Hierarchical Clustering: Slide 7

K-means (thanks to Andrew Moore, CMU)

K-means

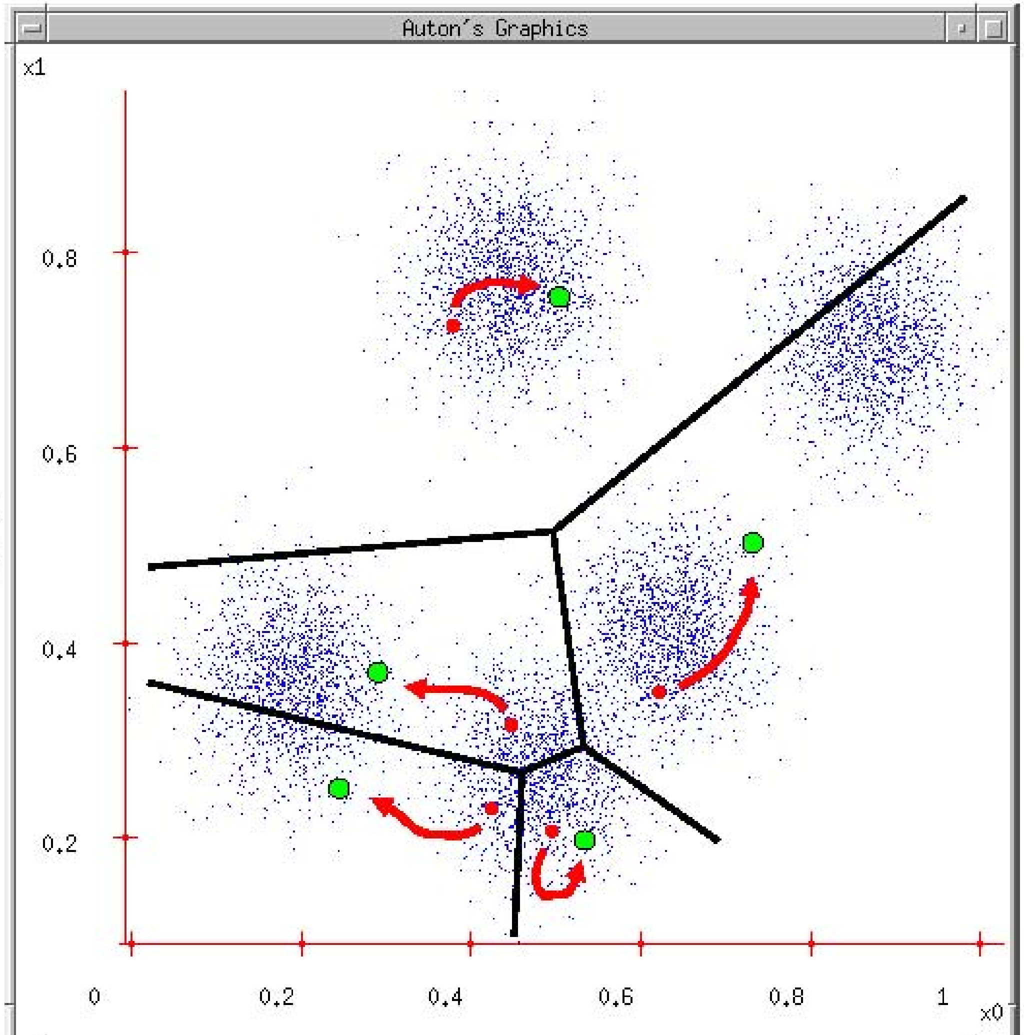
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



K-means (thanks to Andrew Moore, CMU)

K-means

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

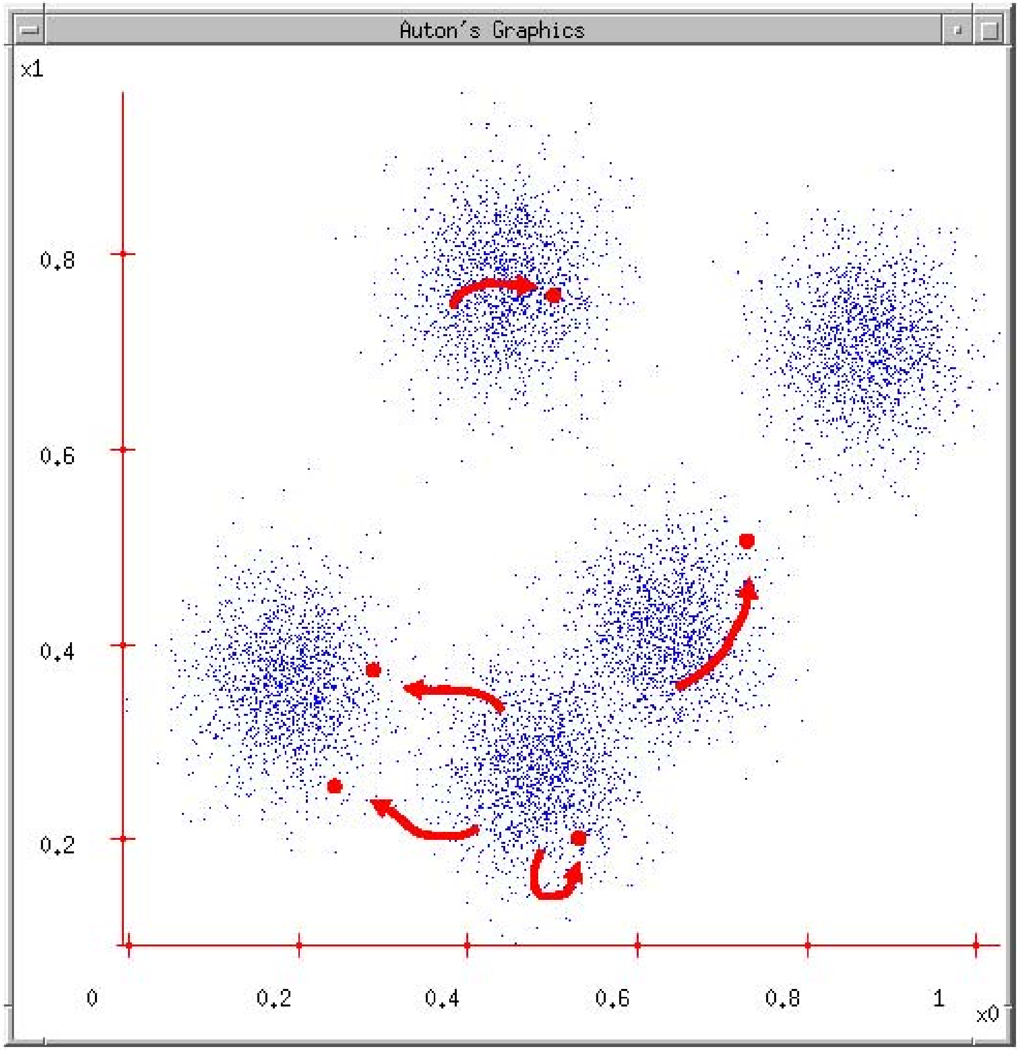


K-means and Hierarchical Clustering: Slide 9

K-means (thanks to Andrew Moore, CMU)

K-means

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



K means

- why do we care?
  - it is optimal if you want to minimize the expected value of the squared error
  - it is still the best way to initialize EM
- problems:
  - how many clusters?
    - various methods available, Bayesian information criterion, Akaike information criterion, minimum description length
    - guessing can work pretty well
  - local minimum only
  - how do I initialize?
    - random can be pretty bad
    - mean splitting can be significantly better

mean splitting

- for $K = 1$ we just need the mean of all points (μ^1)
- to initialize means for $K = 2$ perturb the mean randomly
  - $\mu_1^2 = \mu^1$
  - $\mu_2^2 = (1+\varepsilon)\ \mu^1 \qquad \varepsilon << 1$
- then run K means with $K = 2$
- initial means for $K = 4$
  - $\mu_1^4 = \mu_1^2$
  - $\mu_2^4 = (1+\varepsilon)\ \mu_1^2$
  - $\mu_3^4 = \mu_2^2$
  - $\mu_4^4 = (1+\varepsilon)\ \mu_2^2$
- then run K means with $K = 4$
- etc ….

Empty clusters

- can be a source of headaches
- at the end of each iteration of K means
  - check the number of elements in each cluster
  - if too low, throw the cluster away
  - reinitialize the mean with a perturbed version of that of the most populated cluster

- OK, this is k-means. What about EM?
- "filing in" the z_i with the BDR seems intuitive, but
  - Q_1: what about problems that are not about classification?
  - the missing data does not need to be class labels, it could be a continuous random variable
  - Q_2: how do I know that this converges to anything interesting?

Two open questions

- Questions
  - Q_1: what about problems that are not about classification?
  - Q_2: how do I know that this converges to anything interesting?
- we will look at Q_2 in the next class
- Q_1: EM suggests
  - do the most intuitive operation that is ALWAYS possible
  - don't worry about the z_i directly
  - "estimate the likelihood of the complete data by its expected value given the observed data" (E-step)
  - "then maximize this expected value" (M-step)
  - this leads to the so-called Q-function

The Q function

- is defined as

$$Q(\Psi;\Psi^{(n)}) = E_{Z|\mathbf{X};\Psi^{(n)}}\left[\log P_{\mathbf{X},Z}(\mathcal{D},\{z_1,\ldots,z_N\};\Psi)|\mathcal{D}\right]$$

- and is a bit tricky:
  - it is the expected value of likelihood with respect to complete data (joint X and Z)
  - given that we observed incomplete data (X=$\mathcal{D}$)
  - note that the likelihood is a function of Ψ (the parameters that we want to determine)
  - but to compute the expected value we need to use the parameter values from the previous iteration (because we need a distribution for Z|X)
- the EM algorithm is, therefore, as follows

Expectation-maximization

- E-step:
  - given estimates $\Psi^{(n)} = \{\Psi^{(n)}{}_1, \ldots, \Psi^{(n)}{}_C\}$
  - compute expected log-likelihood of complete data

$$Q(\Psi;\Psi^{(n)}) = E_{Z|\mathbf{X};\Psi^{(n)}}\left[\log P_{\mathbf{X},Z}(\mathcal{D},\{z_1,\ldots,z_N\};\Psi)|\mathcal{D}\right]$$

- M-step:
  - find parameter set that maximizes this expected log-likelihood

$$\Psi^{(n+1)} = \arg\max_{\Psi} Q(\Psi;\Psi^{(n)})$$

- let's make this more concrete by looking at the mixture case

Expectation-maximization

- to derive an EM algorithm you need to do the following

1. write down the likelihood of the COMPLETE data
2. E-step: write down the Q function, i.e. its expectation given the observed data
3. M-step: solve the maximization, deriving a closed-form solution if there is one

Any Questions?