Schedule for the rest of the quarter

- 11/25 (Today): Clustering and latent variable models
- 11/27 (Thu): No class, happy Thanksgiving!
- 12/2 (Tue): Guest lecture by Leo on bandits [not on exam]
- 12/4 (Thu): Foundation models
- 12/8 (Mon): Final exam!

Clustering & Latent Variable Models

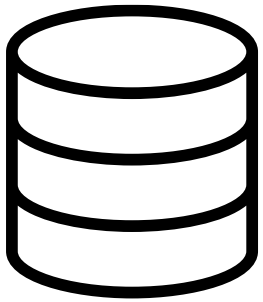
CSE 446/546

Sewoong Oh & Pang Wei Koh

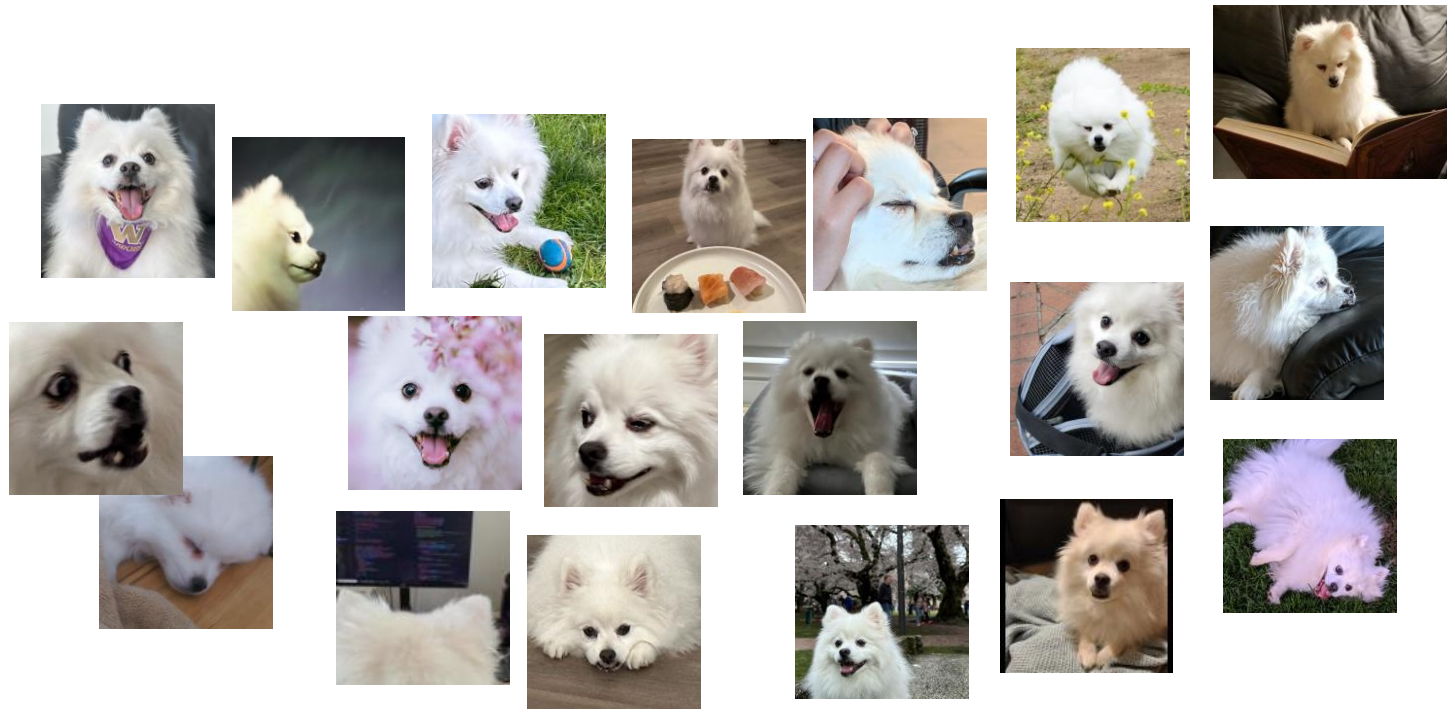
Clustering

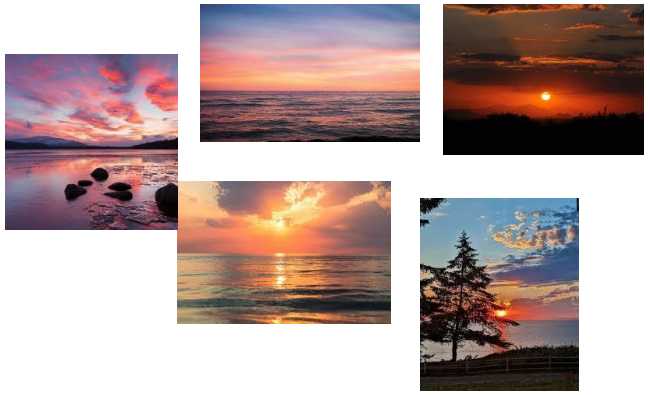
- Fundamental problem in unsupervised ML
- Goal: Group “similar” data points into clusters

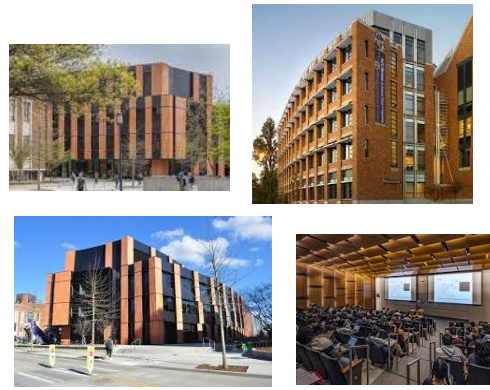
Clustering images



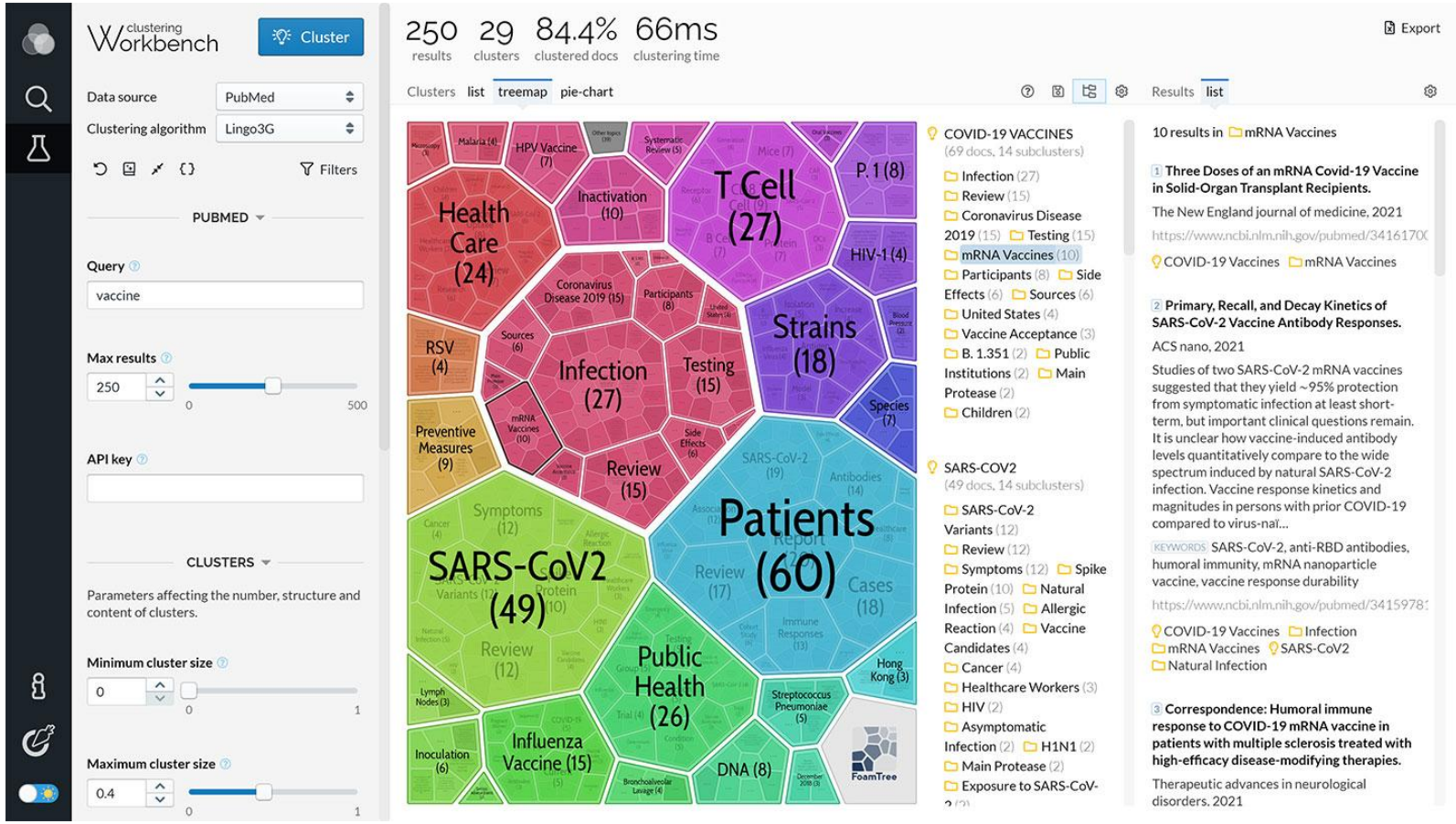
My photos





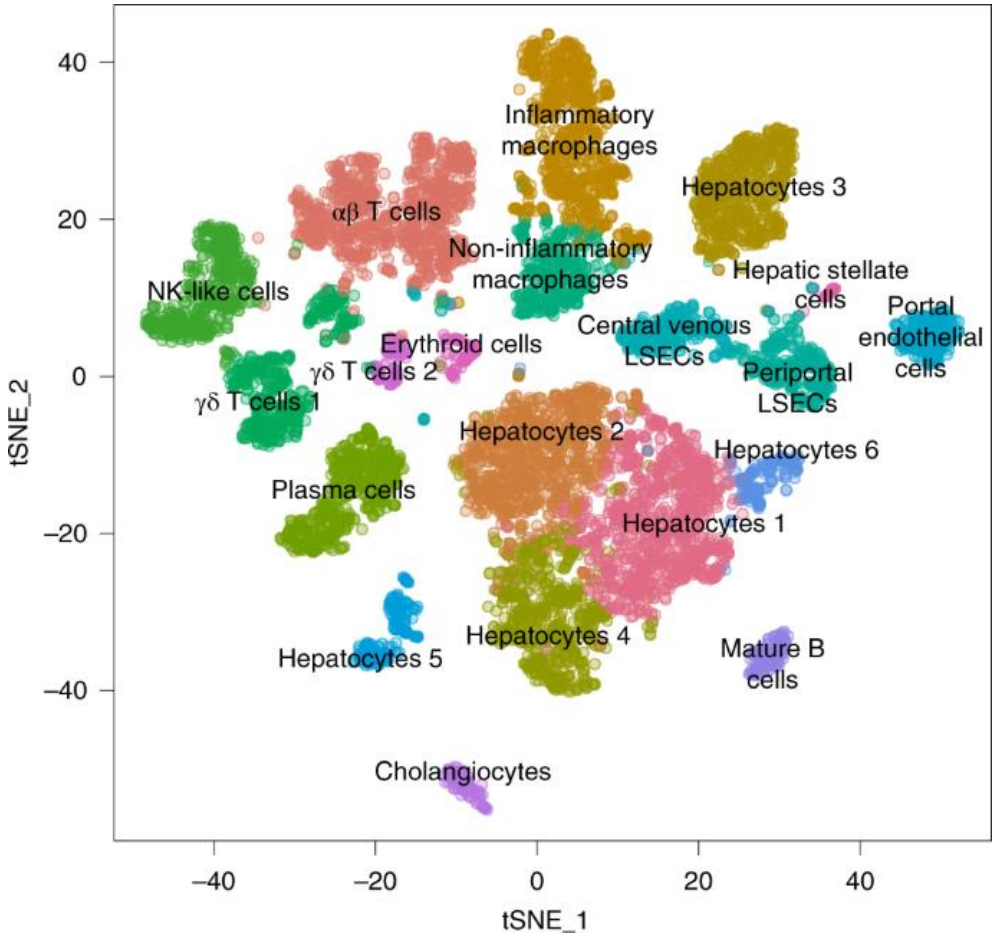


Clustering web search



https://carrotsearch.com/lingo3g/

Clustering cell types



https://www.nature.com/articles/s41596-021-00534-0

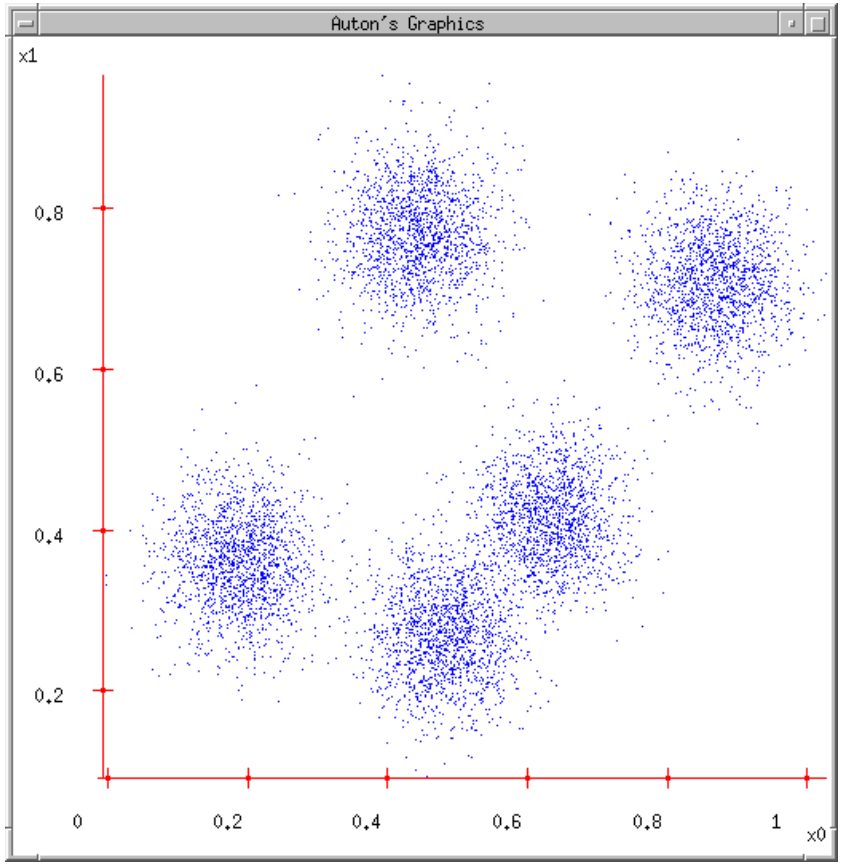
Clustering

- This lecture, we'll study two clustering methods:
  1. K-means
  2. Mixture of Gaussians

K-means

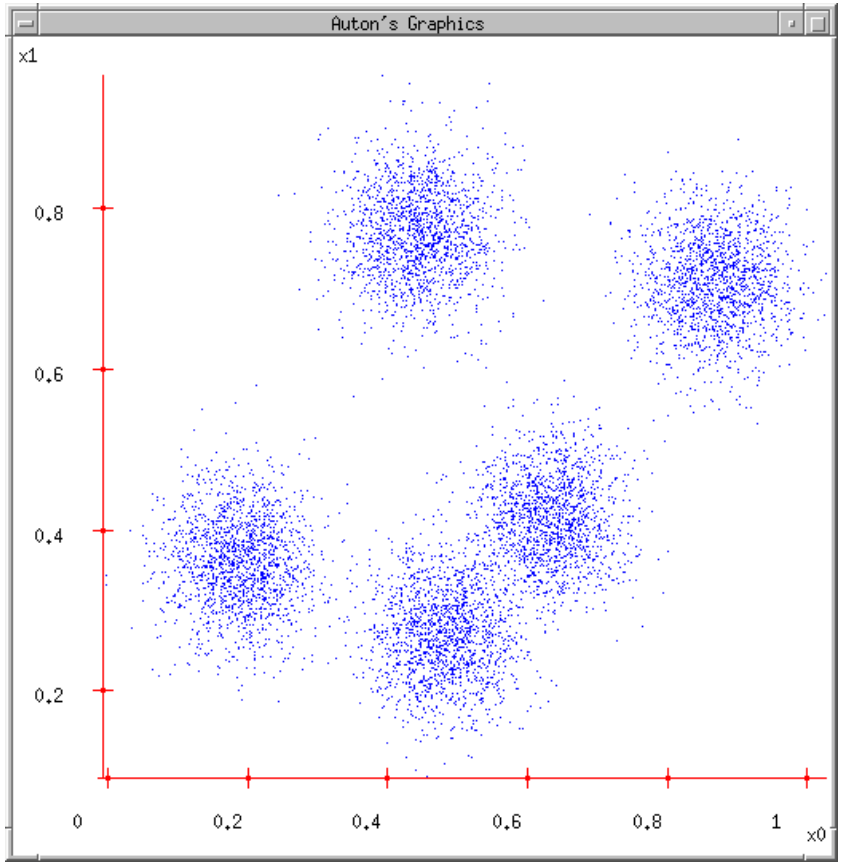
- Goal: Find cluster centers & cluster assignments that minimize the average squared distance between each point and its cluster center

$$\underset{\mu, C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||_2^2$$



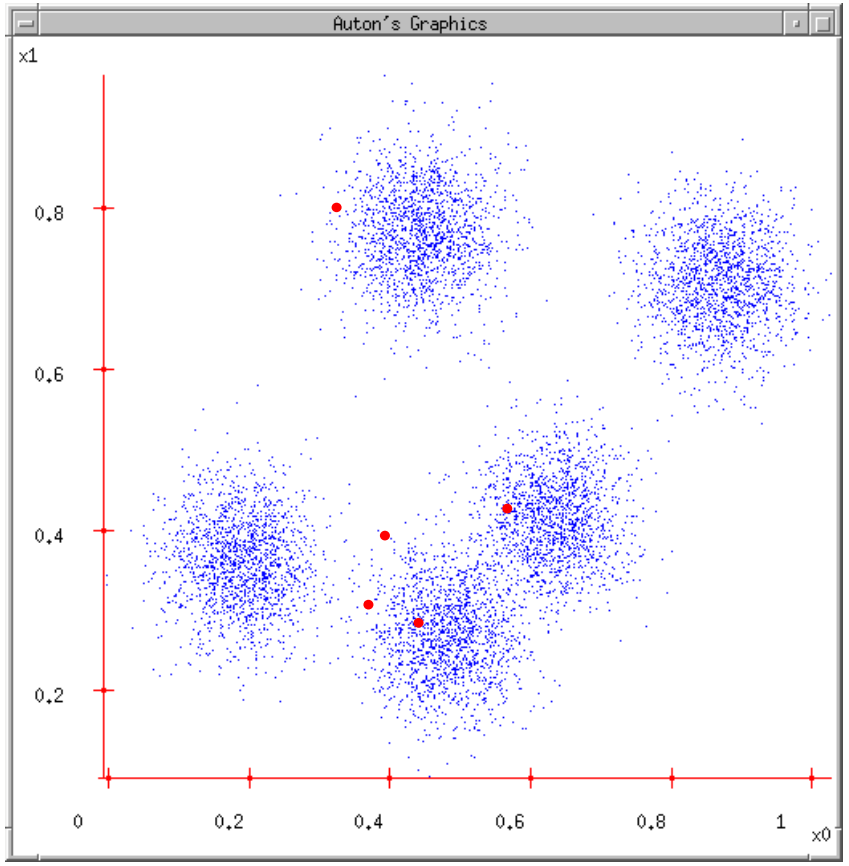
K-means

1. Ask user for # of clusters (e.g., k=5)



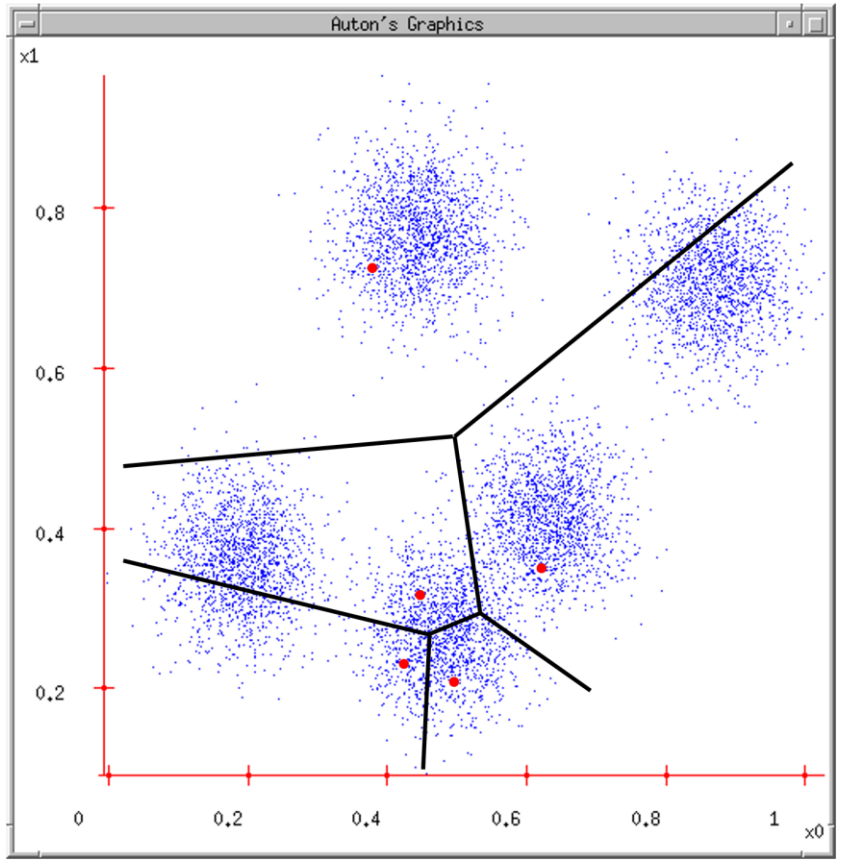
K-means

1. Ask user for # of clusters (e.g., k=5)
2. Randomly guess k cluster centers



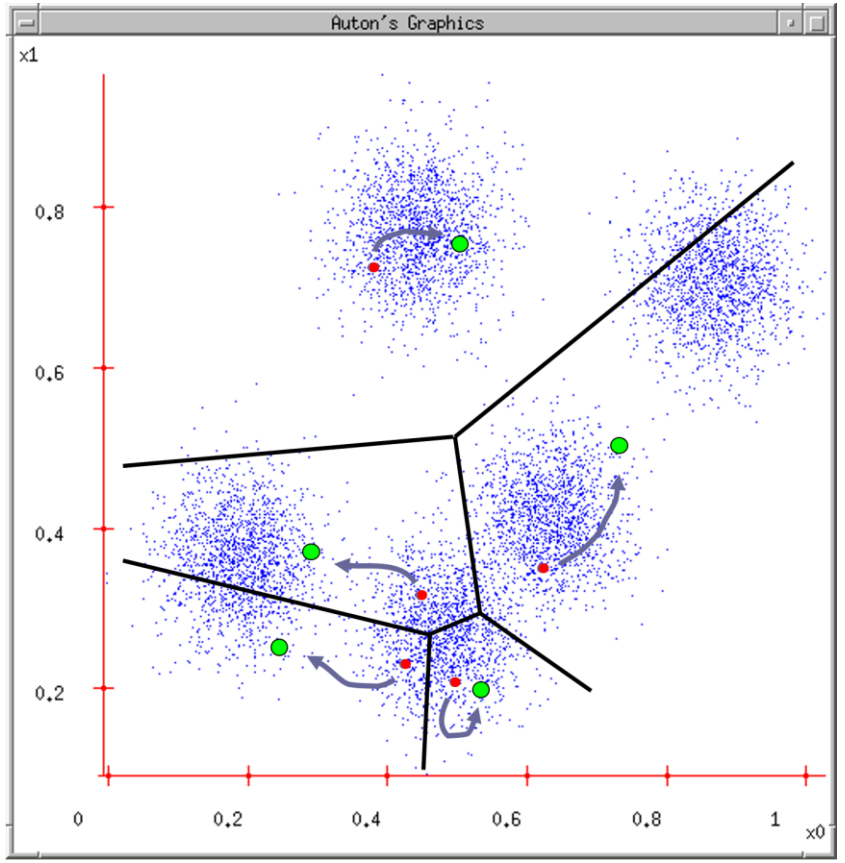
K-means

1. Ask user for # of clusters (e.g., k=5)
2. Randomly guess k cluster centers
3. Assign each data point to nearest cluster



K-means

1. Ask user for # of clusters (e.g., k=5)
2. Randomly guess k cluster centers
3. Assign each data point to nearest cluster
4. Each center moves to centroid of points it "owns"
5. Repeat until terminated!



K-means as optimization

- Randomly initialize cluster centroids
- Classify: Assign each point x_j to nearest center

- Recenter: Each center becomes centroid of its points

$$\underset{\mu, C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||_2^2$$

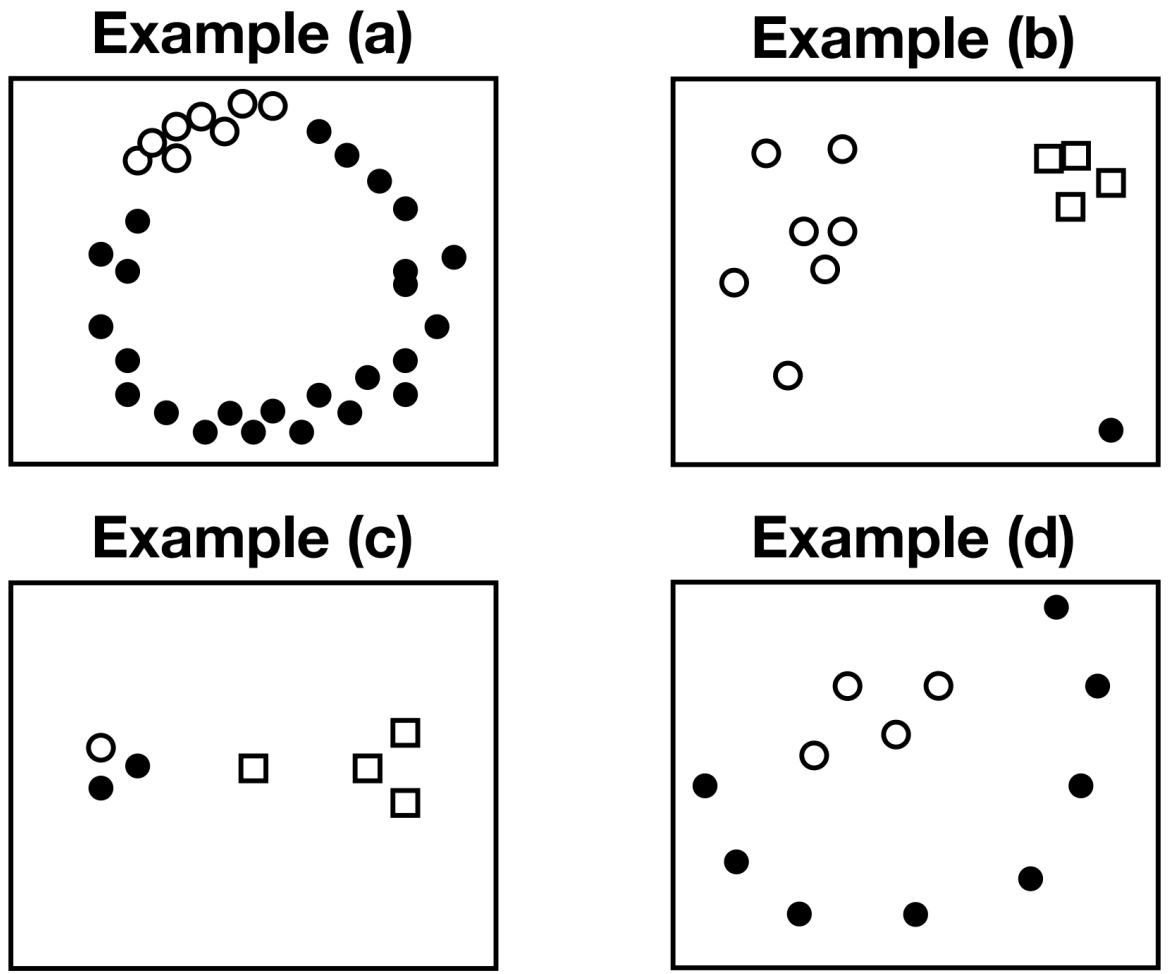
K-means is alternating minimization!

Does k-means converge?

$$\underset{\mu, C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||_2^2$$

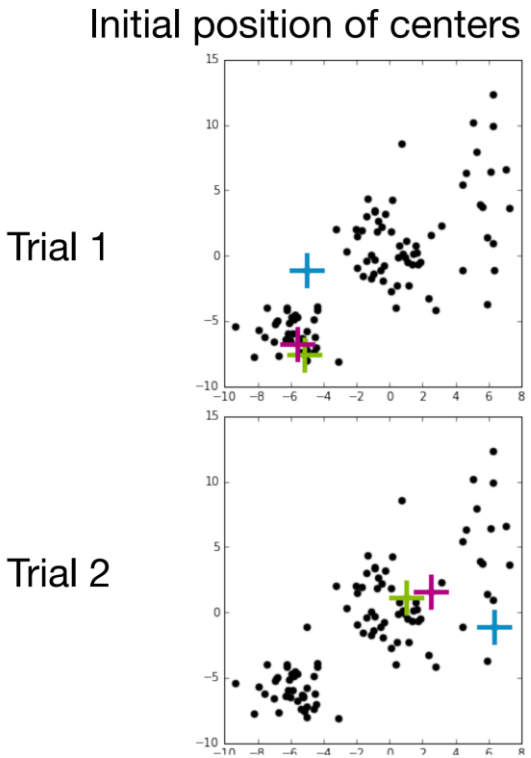
- After each iteration, objective always decreases or stays the same
- Does it terminate in finite time?
  - Finite set of values for cluster assignments
  - Objective function decreases or optimization terminates
  - Objective is lower bounded by 0
  - Therefore, converges in at most k^n steps

Has k-means converged?



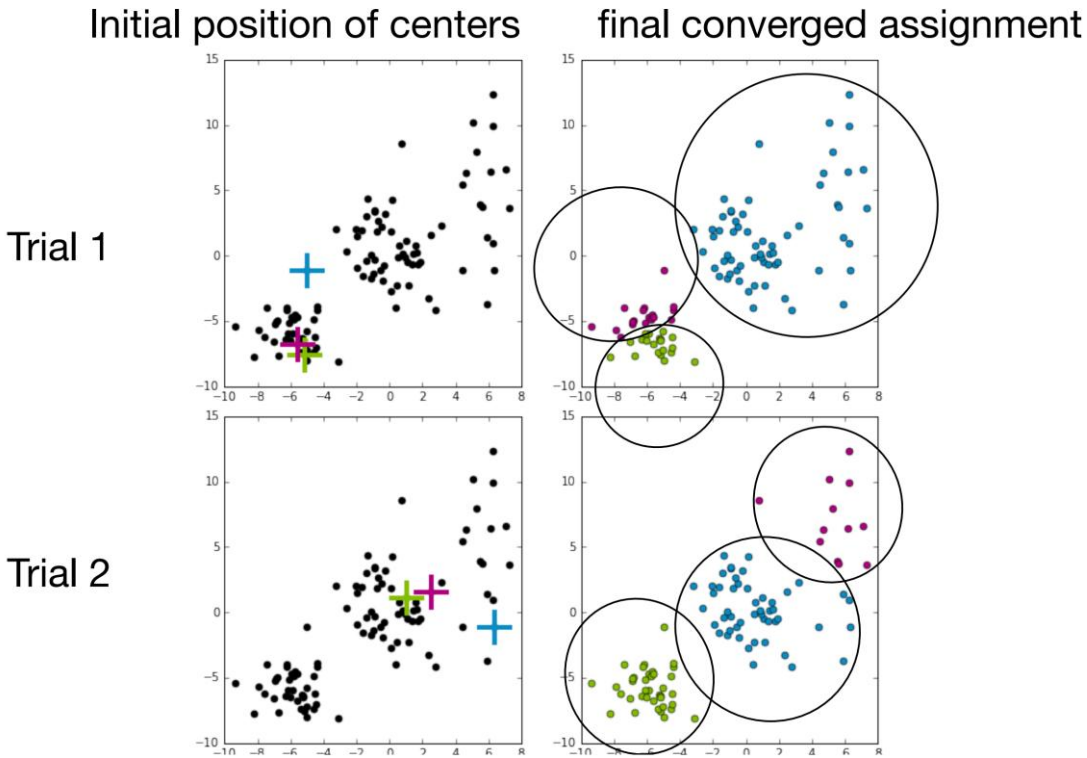
Downsides of k-means

- Requires number of clusters k to be specified by us
- Final solution depends on init



Downsides of k-means

- Requires number of clusters k to be specified by us
- Final solution depends on init



k-means++: A smarter initialization

1. Choose first cluster center μ_1 uniformly at random from data points
2. For $k = 2, \ldots, K$:
   - For each data point x_i, compute distance d_i to nearest cluster center
   - Choose new cluster from data points with probability of x_i chosen proportional to d_i^2

Apply standard k-means after this initialization

When does k-means fail?

Disparate cluster sizes

Differently shaped clusters

Gaussian mixture models (GMMs)

- Key idea: model each cluster as a Gaussian distribution with its own mean and covariance
- Define a distribution over data points that is a mixture of Gaussians

Gaussian mixture models (GMMs)

- Parameters:
  - Mixing weights π_j, for $j = 1, \ldots K$
  - Means μ_j, for $j = 1, \ldots K$
  - Covariances Σ_j, for $j = 1, \ldots K$
- Under the GMM, sample x_i is drawn as follows:
  - First sample a cluster $z_i \in \{1, \ldots, K\}$ w.p. π
  - Sample $x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$
- Given parameters of the GMM, how do we recover clusters?

How do we fit GMMs?

- The same way we always do: Maximum likelihood estimation!
  - Define the likelihood $\mathrm{P}_\theta(x)$
  - Find θ that maximizes the likelihood of observing the data $x_1, \ldots, x_n$
- For simplicity, assume $x \in \mathbb{R}$
- Likelihood:

Recap lecture 1: Fitting a single Gaussian

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial \mu} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \mu} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$\frac{\partial}{\partial \sigma} \log P(\mathcal{D} \mid \mu, \sigma) = \frac{\partial}{\partial \sigma} \left[-n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

Fitting multiple Gaussians

$$\log P(\mathcal{D}|\pi,\mu,\sigma) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \, P(x_i|\mu_k,\sigma_k)$$

- Can differentiate, but… no closed form solution (try this at home!)
- What if we knew which of the k Gaussians each x_i came from?

Expectation Maximization (EM) algorithm for GMMs

Repeat until convergence:

- E-step: For each x_i, guess which Gaussian it came from

- M-step: Fit Gaussian parameters accordingly

Why does EM work?

- How do we justify this procedure?
- Does it converge?
- What does it converge to?

- Key idea:
  - E-step forms a tight lower bound to $P(\mathcal{D}|\pi, \mu, \sigma)$
  - M-step optimizes this lower bound

Latent variable models

- Previously, we've studied models where all variables are observed

- Now, we have a latent (=unobserved) variable z

- EM is a general method for optimizing latent variable models

Detour: Jensen's inequality

*Named after Danish mathematician Johan Jensen, no relation to NVIDIA

The Evidence Lower Bound (ELBO)

EM is alternating maximization on the ELBO

$\log P(x;\theta) \geq \mathrm{ELBO}(\mathrm{x};\mathrm{Q},\theta)$

E-step: Optimize Q to create tight lower bound $\mathrm{ELBO}(\mathrm{x};\mathrm{Q},\theta_t)$

M-step: Optimize θ_{t+1} to maximize $\mathrm{ELBO}(\mathrm{x};\mathrm{Q},\theta_t)$

Convergence

- Likelihood never decreases
  - Lower bound is always tight
  - We always increase (or stay the same) on the lower bound
- Therefore, EM converges
- But it converges to a local maximum

- Do we need to use EM?

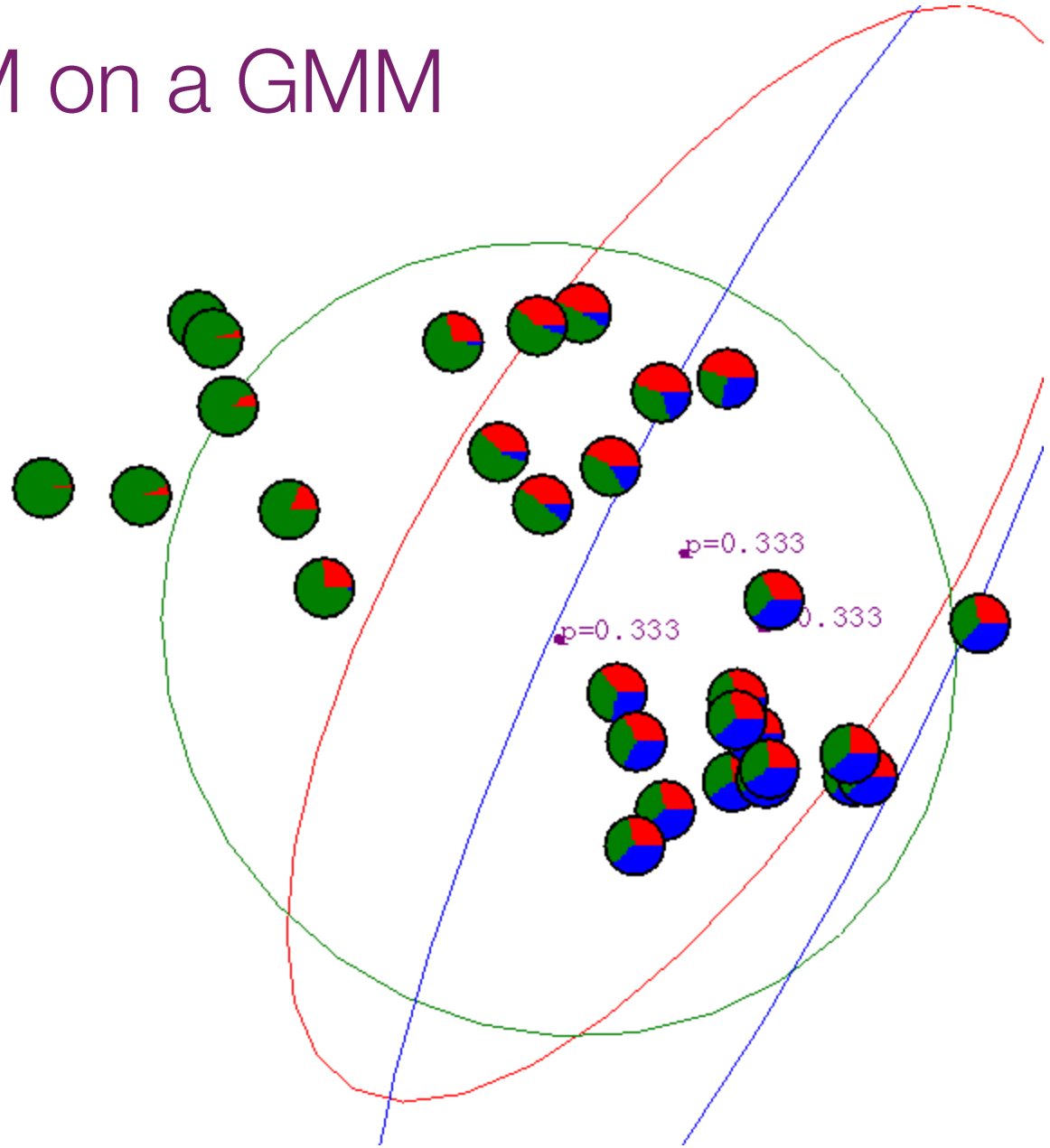
Multivariate Gaussians

Given $x \in \mathbb{R}^d$, $P(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$

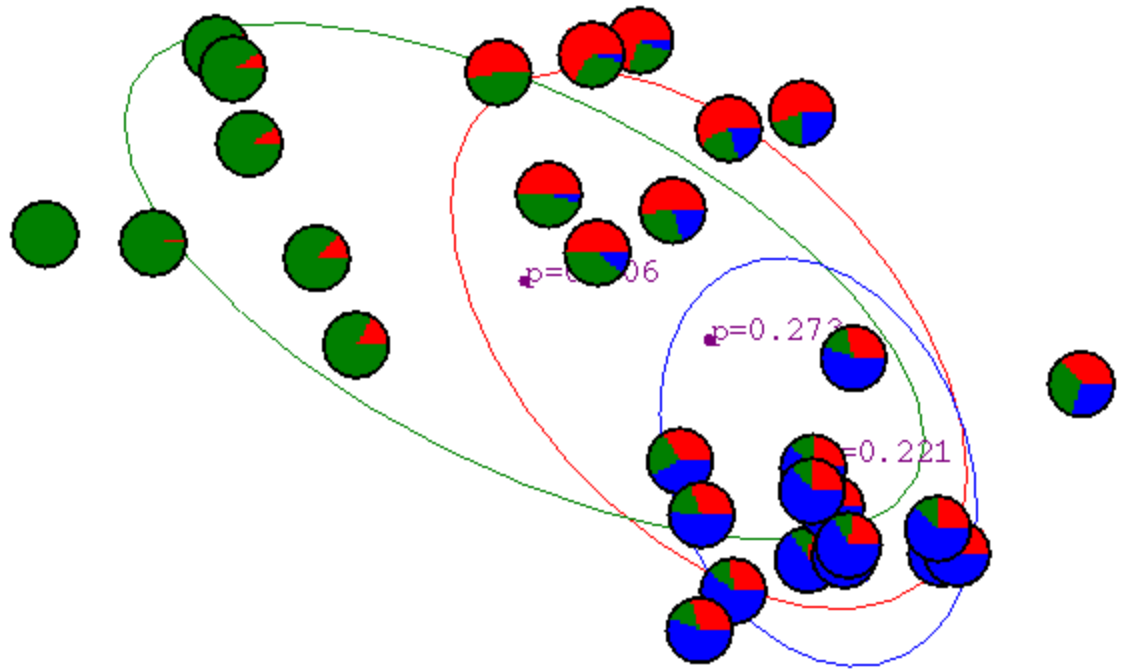
$$\hat{\mu}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\Sigma}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{\text{MLE}})(x_i - \hat{\mu}_{\text{MLE}})^\top$$

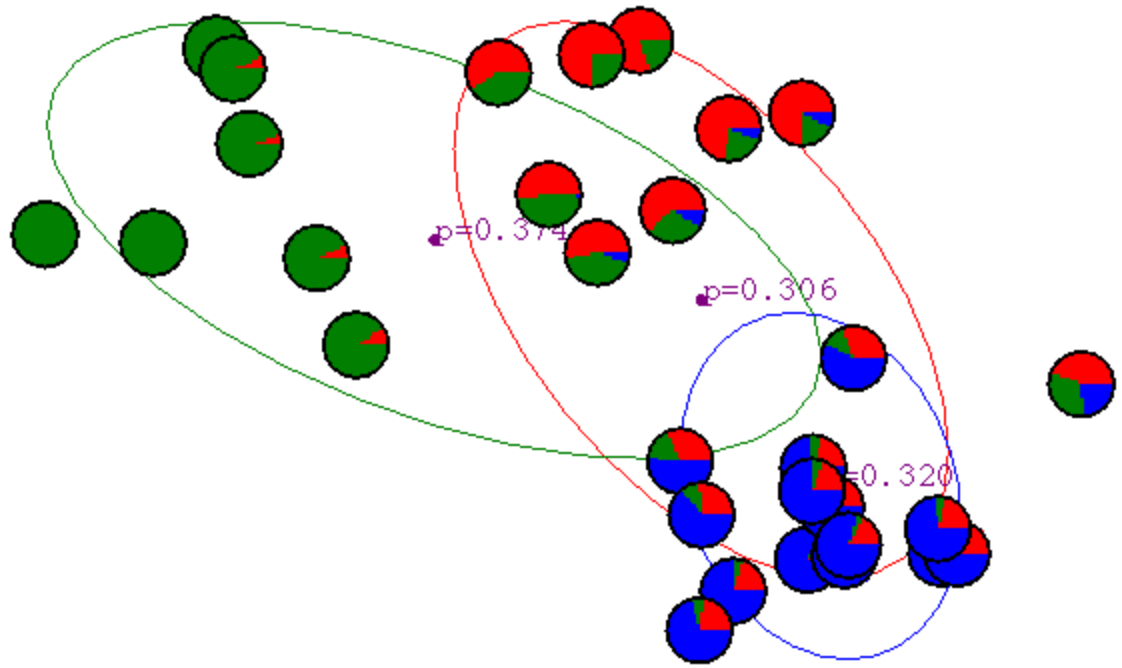
Example: Running EM on a GMM



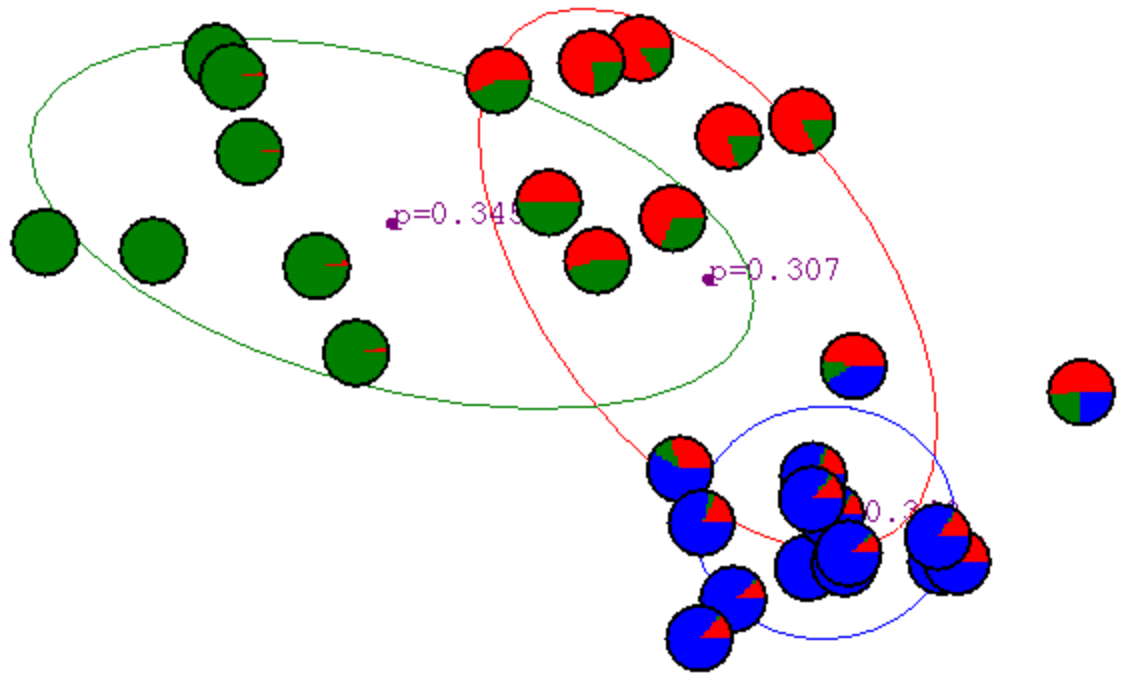
Example: Running EM on a GMM



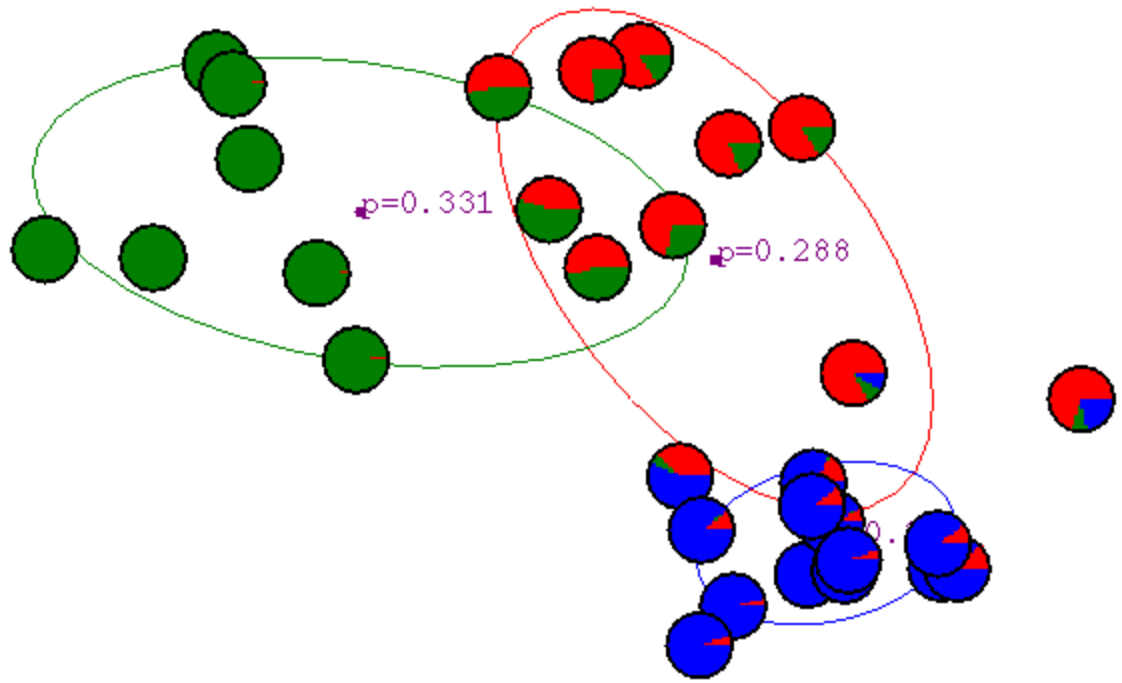
Example: Running EM on a GMM



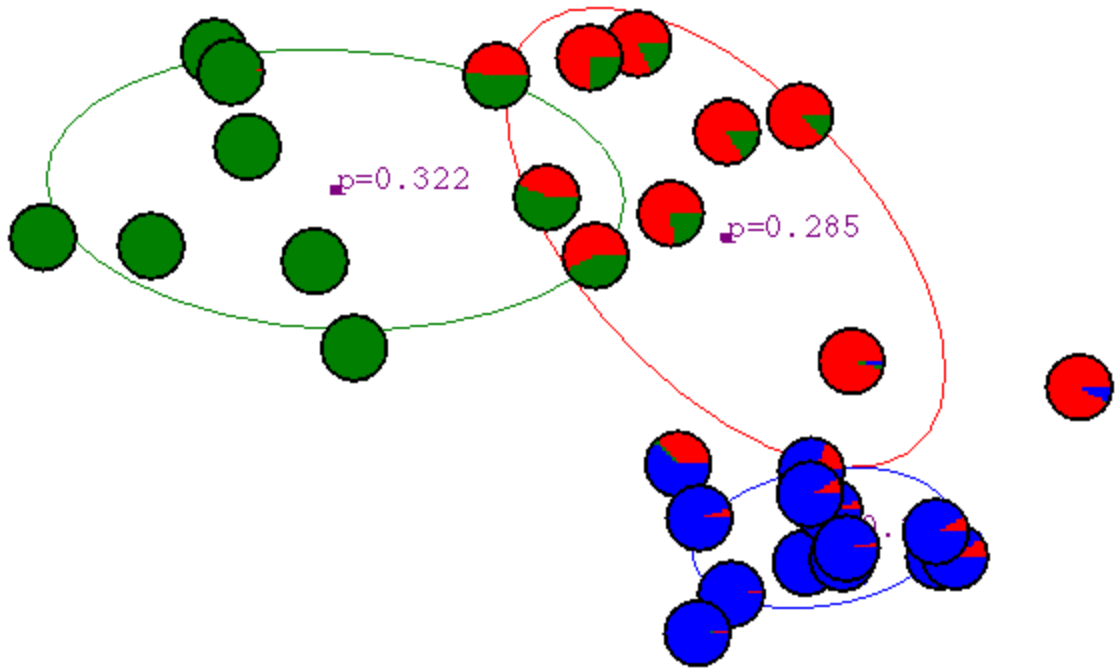
Example: Running EM on a GMM



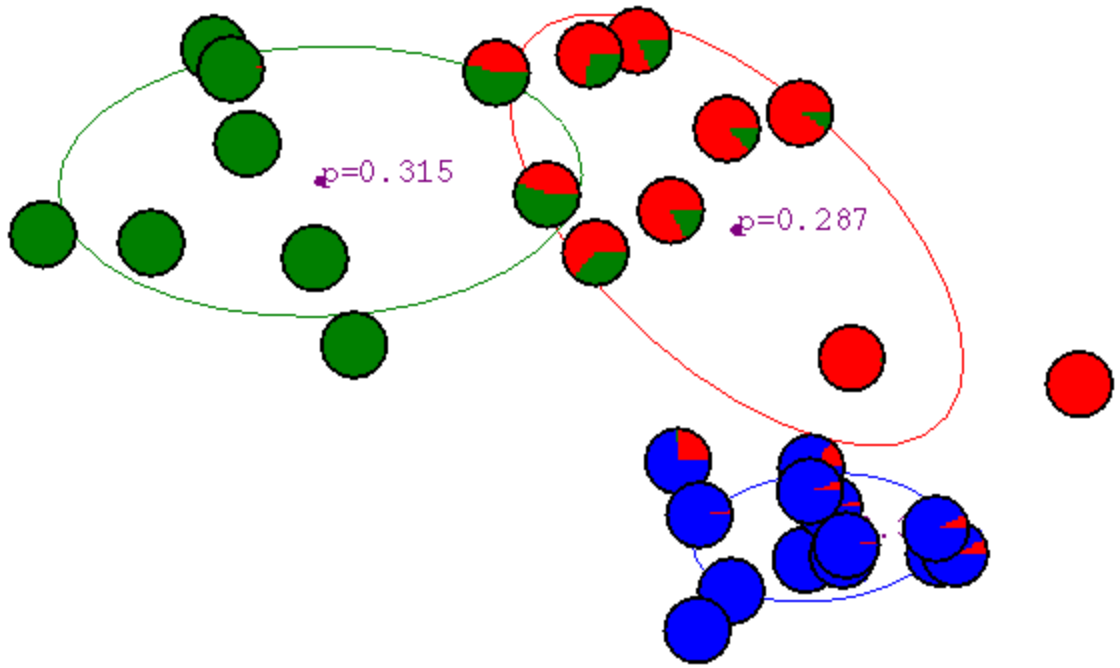
Example: Running EM on a GMM



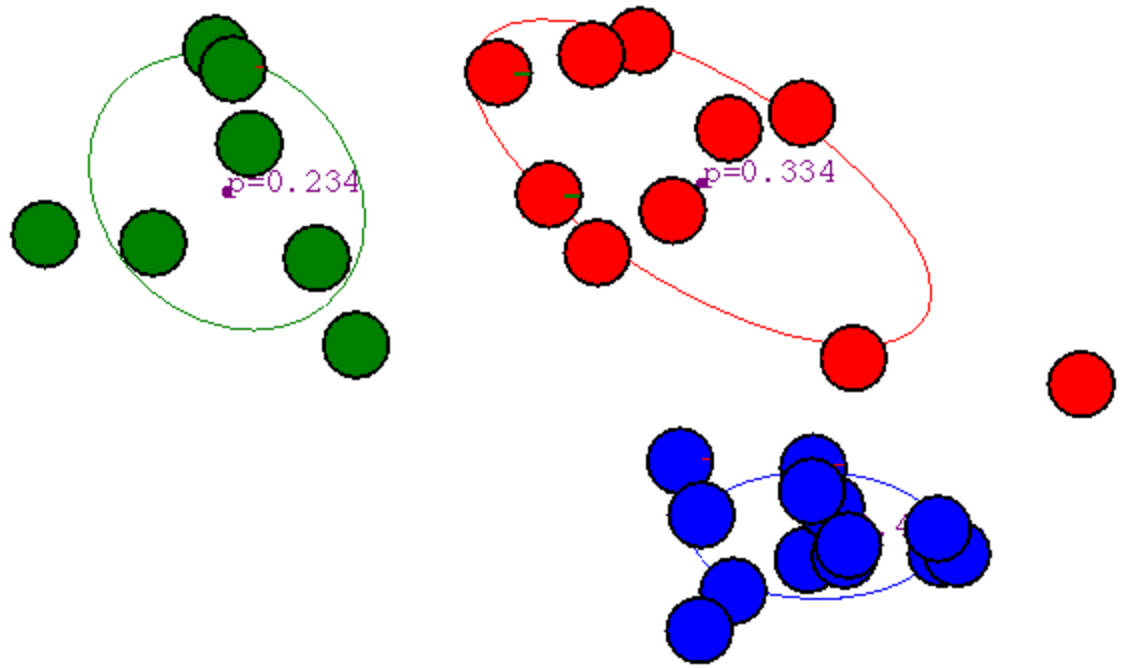
Example: Running EM on a GMM



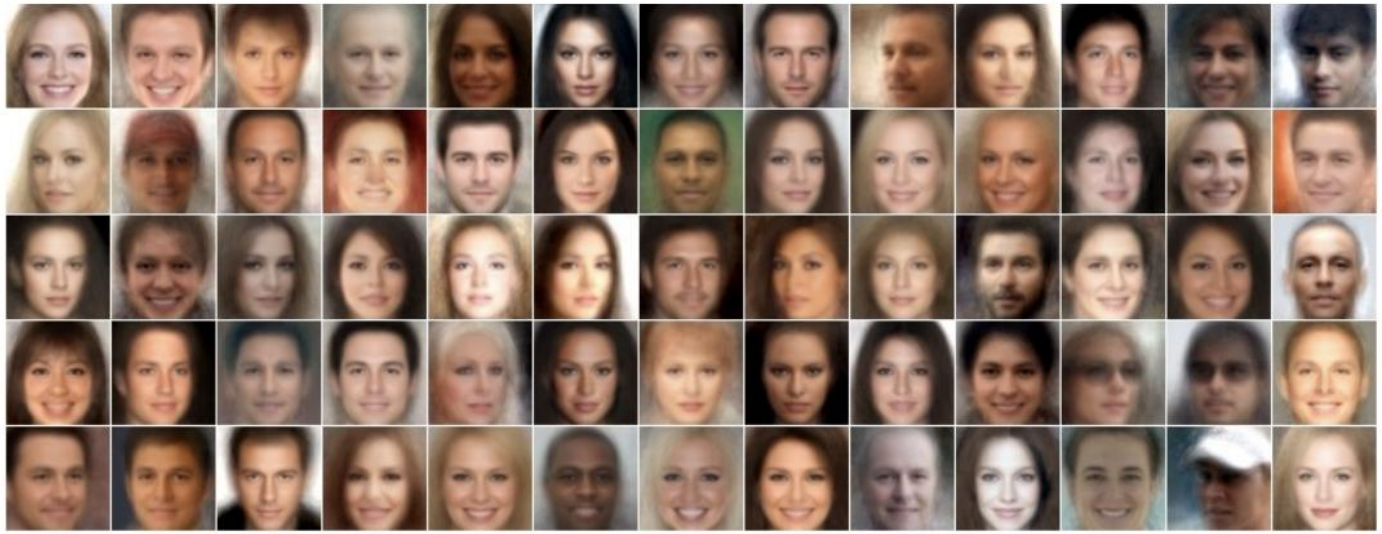
Example: Running EM on a GMM



Example: Running EM on a GMM



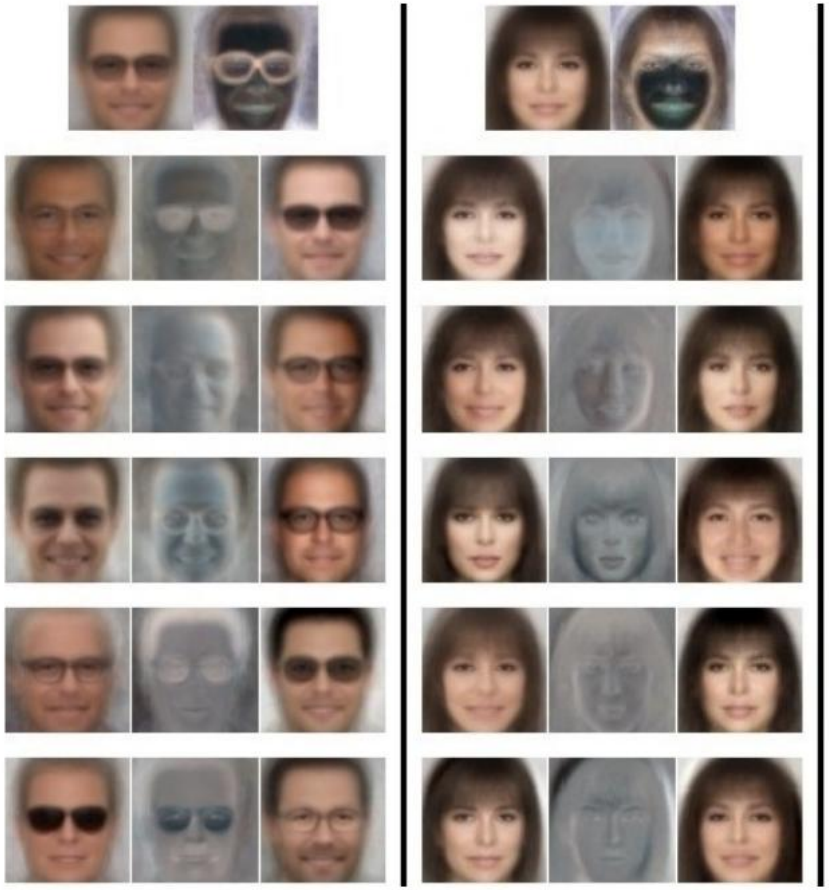
Example: GMMs on faces

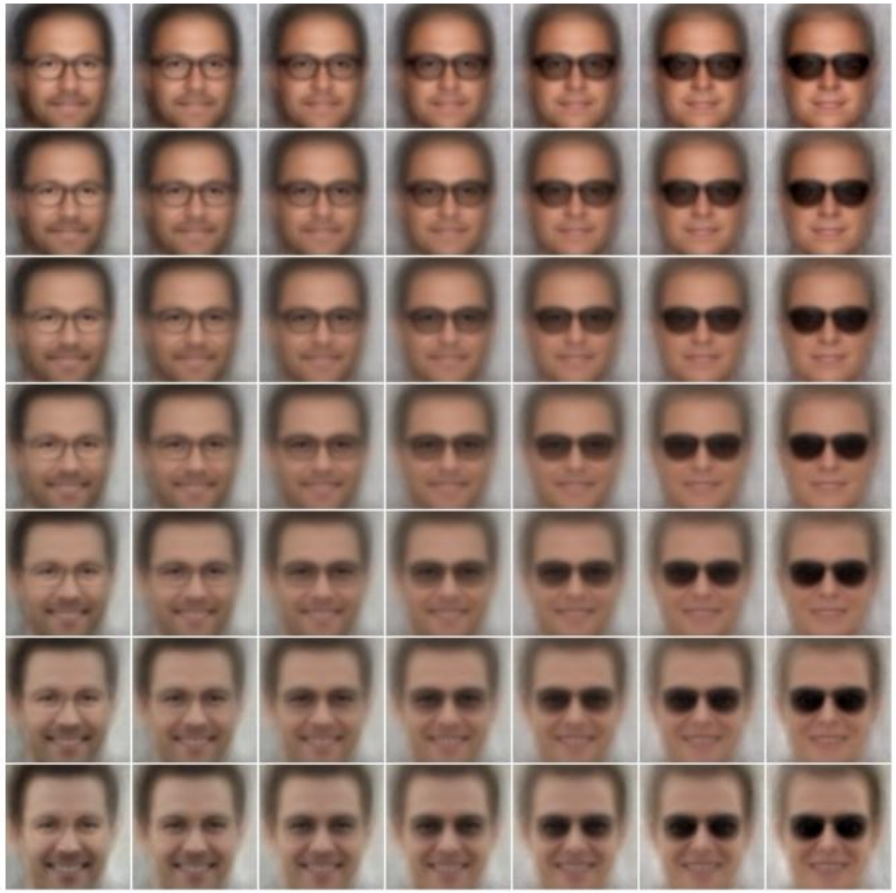


- Generated samples from GMM (K=1,000) trained on CelebA
- Images: 64 x 64 x 3
- Covariance: Restricted to rank-10 matrices

[Richardson & Weiss, On GANs and GMMs, 2018.]

Example: GMMs on faces





[Richardson & Weiss, On GANs and GMMs, 2018.]