10-701: Introduction to Machine Learning Lecture 15: Clustering

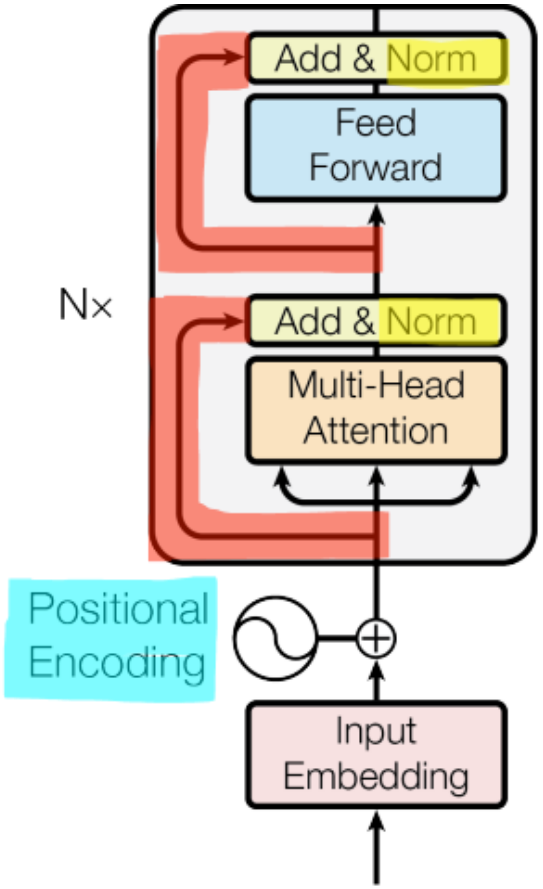
Pradeep Ravikumar & Geoff Gordon

Spring 2026

Front Matter

- Announcements
  - TA project proposals have been turned in; all teams should schedule a time to have a 15-30 minute meeting with their TA mentor before Friday March 20!
  - HW4 has been released and is due on Tuesday, March 24
- Recommended Readings
  - Daumé III, Chapter 15: Unsupervised Learning
  - Murphy, Section 11.1 – 11.4.2

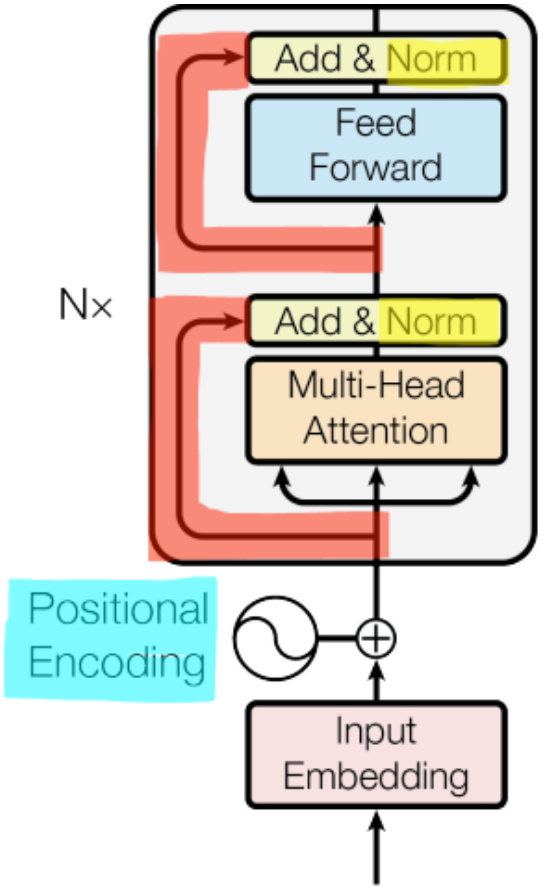
Recall: Transformers



- In addition to multi-head attention, transformer architectures use
  1. Positional encodings
  2. Layer normalization
  3. Residual connections
  4. A fully-connected feed-forward network

Source: https://arxiv.org/pdf/1706.03762.pdf

How on earth do we train these things?



- In addition to multi-head attention, transformer architectures use
  1. Positional encodings
  2. Layer normalization
  3. Residual connections
  4. A fully-connected feed-forward network

Source: https://arxiv.org/pdf/1706.03762.pdf

Learning Paradigms

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning
- Active learning
- Semi-supervised learning
- Online learning

Learning Paradigms

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$
  - **Clustering**
  - Dimensionality reduction
- Reinforcement learning
- Active learning
- Semi-supervised learning
- Online learning

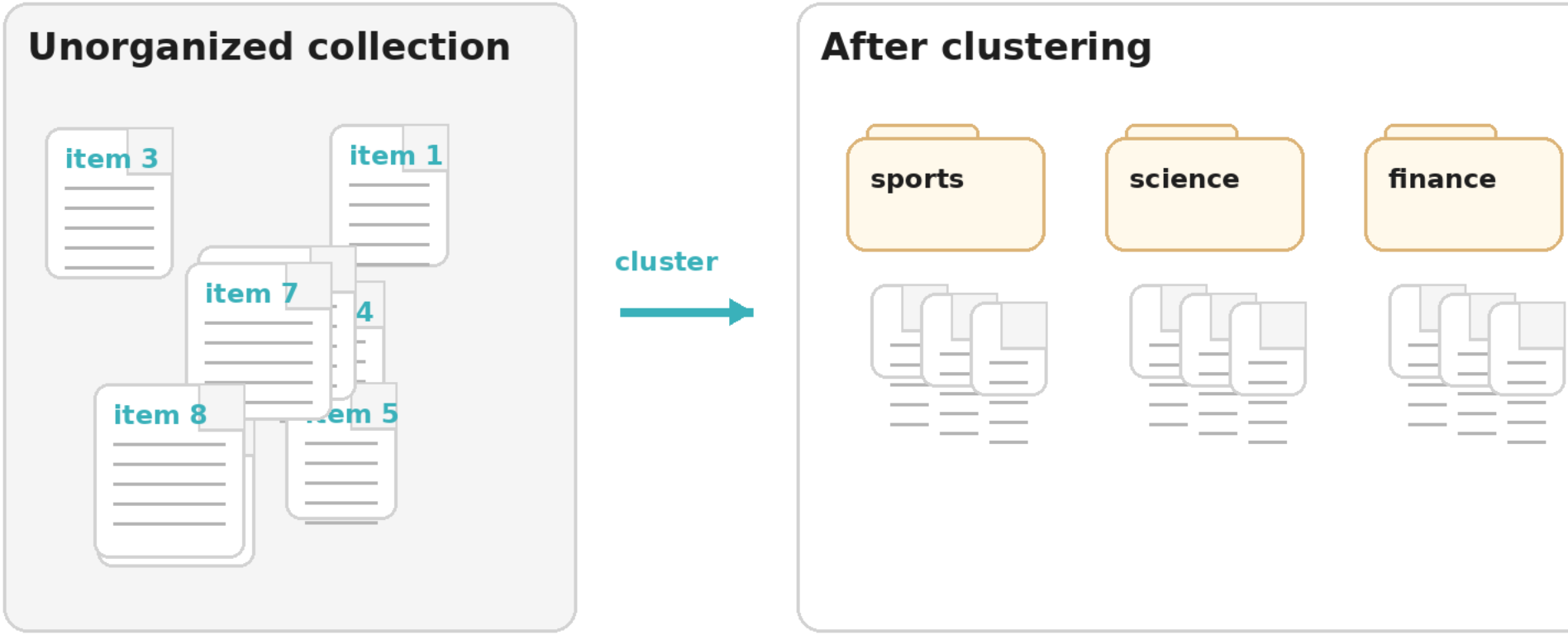
Clustering

- Goal: split an unlabeled data set into groups or clusters of “similar” data points
- Use cases:
  - Organizing data
  - Discovering patterns or structure
  - Preprocessing for downstream machine learning tasks

Use case: Organizing data

Clustering as an organizing tool

When you have lots of unlabeled items, clustering can group similar ones before you have a taxonomy.

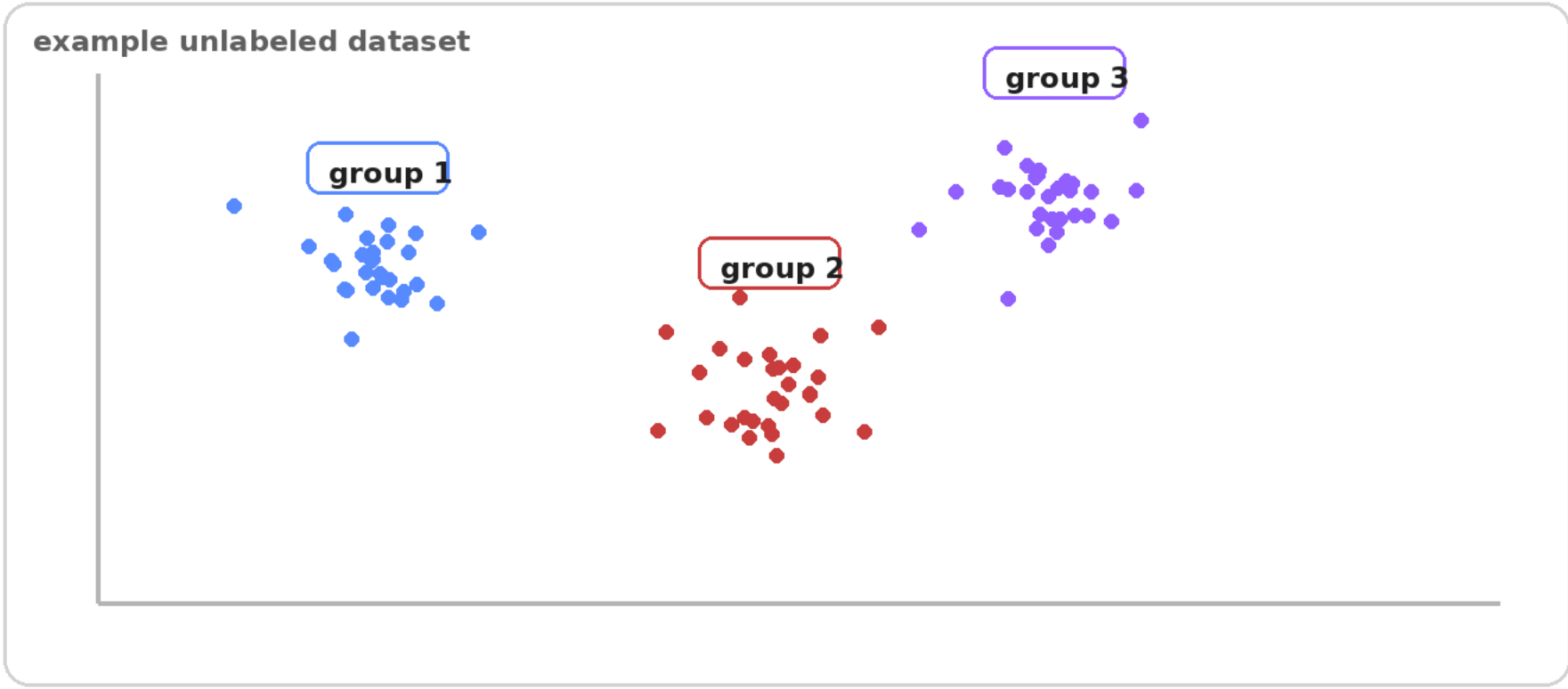


Examples: organizing news articles, support tickets, photos, product catalogs, or research papers.

Use case: Discovering patterns

Clustering can reveal hidden structure

Even without labels, clustered data can reveal subpopulations, recurring modes, or latent structure.

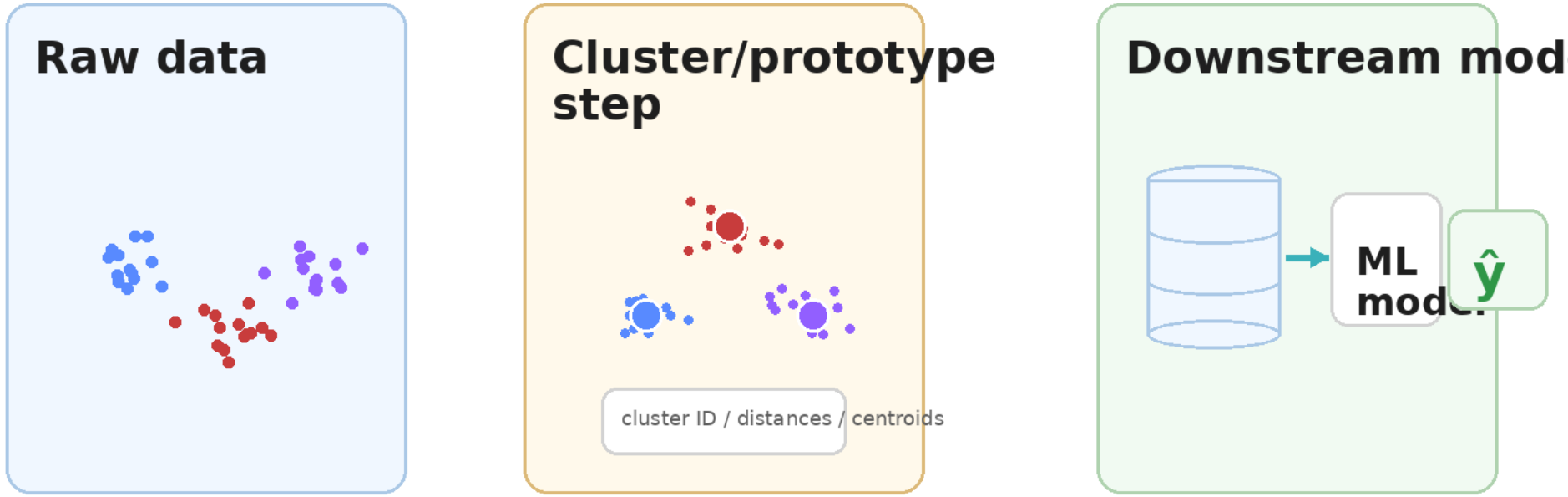


This is often the first clue that the data contains distinct regimes or populations worth modeling separately.

Use case: Preprocess for ML

Clustering as a preprocessing step

Clusters can compress, summarize, or featurize data before a downstream model sees it.



Examples: cluster first, then use prototypes or cluster-based features for compression, retrieval, or a later classifier.

Why care about clustering?

A common “first 48 hours” ML story

You inherit 100,000 unlabeled support tickets. No one has built a taxonomy yet. You still need to make sense of the data.

Day 1

Everything is a pile of text. No labels. No categories.

Day 2

Cluster the tickets. Recurring themes appear: billing, login, shipping, bugs, cancellations.

Day 3

Now humans can name clusters, prioritize them, and build a better downstream system.

Clustering matters because real data often arrives before labels do.

Recall: Similarity for kNN

- Classify a point as the label of the "most similar" training point
- **Idea: given real-valued features, we can use a distance metric to determine how similar two data points are**
- A common choice is Euclidean distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 = \sqrt{\sum_{d=1}^{D} (x_d - x'_d)^2}$$

- An alternative is the Manhattan distance:

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \sum_{d=1}^{D} |x_d - x'_d|$$

Partition-Based Clustering

- Given a desired number of clusters, K, return a partition of the data set into K groups or clusters, $\{C_1, \dots, C_K\}$, that optimize some objective function

1. What objective function should we optimize?

2. How can we perform optimization in this setting?

Option A

Option B

Which partition is best?

General Recipe for Machine Learning

- Define a model and model parameters
- Write down an objective function
- Optimize the objective w.r.t. the model parameters

Recipe for K-means

- Define a model and model parameters
  - Assume K clusters and use the Euclidean distance
  - Parameters: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and $z^{(1)}, \ldots, z^{(N)}$
- Write down an objective function

$$\sum_{i=1}^{N} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{z^{(i)}} \right\|_2$$

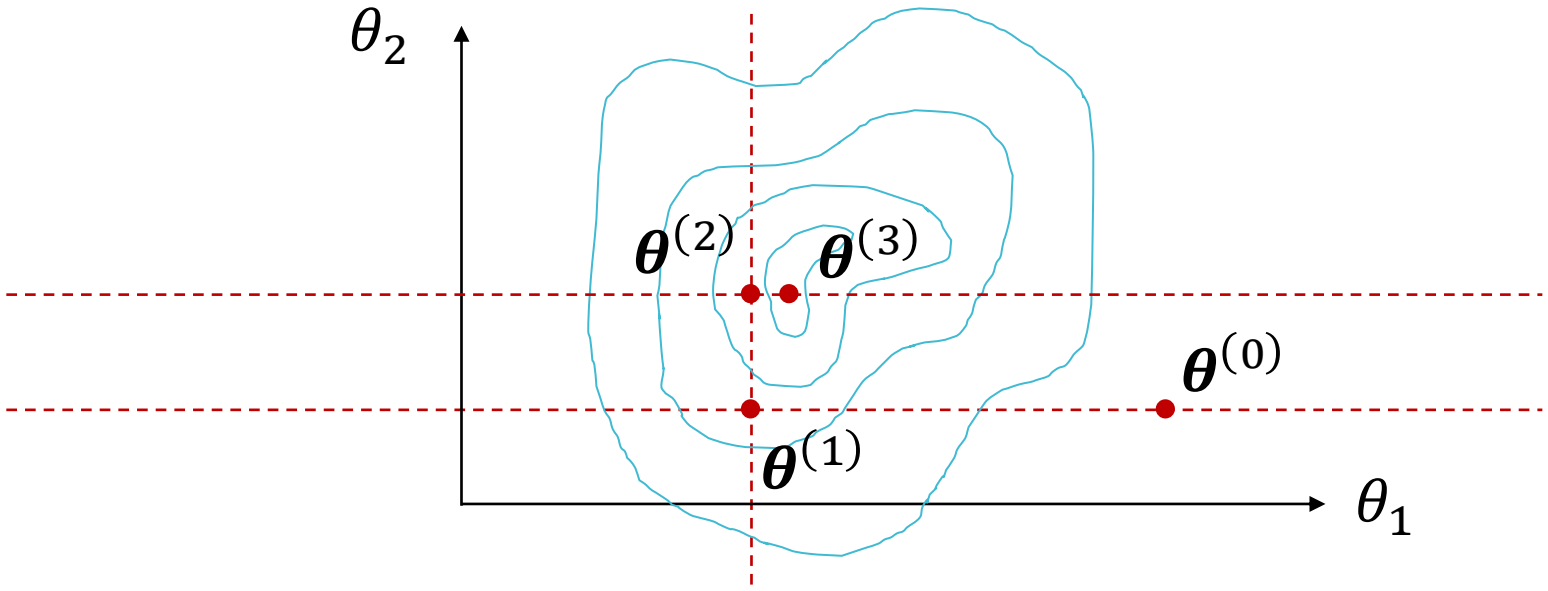
- Optimize the objective w.r.t. the model parameters
  - Use (block) coordinate descent

Coordinate Descent

- Goal: minimize some objective

$$\widehat{\boldsymbol{\theta}} = \text{argmin } J(\boldsymbol{\theta})$$

- Idea: iteratively pick one variable and minimize the objective w.r.t. just that variable, *keeping all others fixed*.



Block Coordinate Descent

- Goal: minimize some objective

$$\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}} = \operatorname{argmin} J(\boldsymbol{\alpha}, \boldsymbol{\beta})$$

- Idea: iteratively pick one *block* of variables ($\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$) and minimize the objective w.r.t. that block, keeping the other(s) fixed.
  - Ideally, blocks should be the largest possible set of variables that can be efficiently optimized simultaneously

Optimizing the K-means objective

$$\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K, z^{(1)}, \ldots, z^{(\mathrm{N})} = \operatorname{argmin} \sum_{i=1}^{N} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{z^{(i)}} \right\|_2$$

- If $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ are fixed

$$\hat{z}^{(i)} = \operatorname*{argmin}_{k \in \{1, \ldots, K\}} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k \right\|_2$$

- If $z^{(1)}, \ldots, z^{(N)}$ are fixed

$$\hat{\boldsymbol{\mu}}_k = \operatorname*{argmin}_{\boldsymbol{\mu}} \sum_{i : z^{(i)} = k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu} \right\|_2$$

$$= \frac{1}{N_k} \sum_{i : z^{(i)} = k} \boldsymbol{x}^{(i)}$$

K-means Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{N}, K$

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
2. While NOT CONVERGED
   a. Assign each data point to the cluster with the nearest cluster center:

$$z^{(i)} = \underset{k}{\operatorname{argmin}} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k\right\|_2$$

   b. Recompute the cluster centers:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i\,:z^{(i)}=k} \boldsymbol{x}^{(i)}$$

   where N_k is the number of data points in cluster k

- Output: cluster assignments $z^{(1)}, \ldots, z^{(N)}$

K-means: Example ($K = 3$)

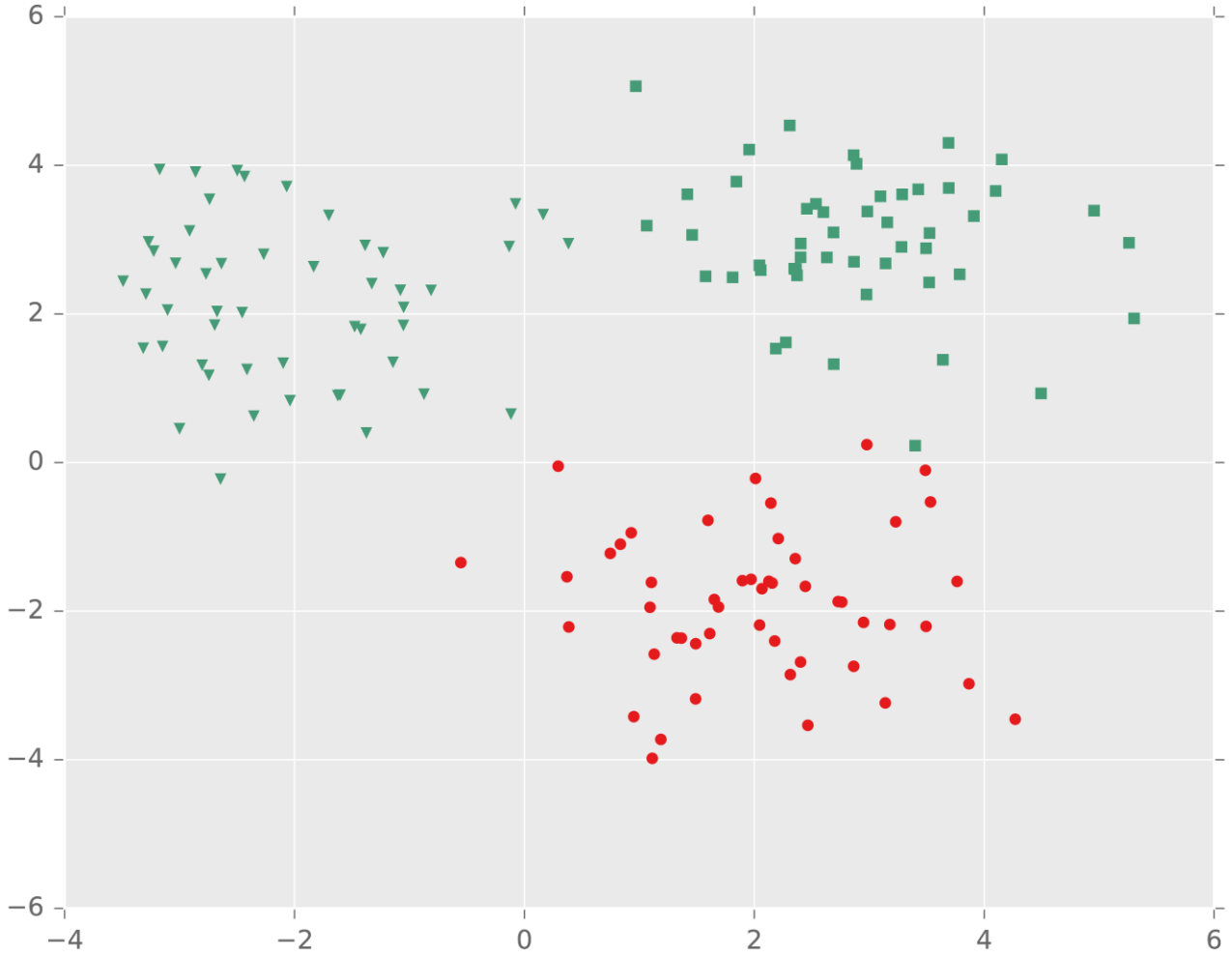


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

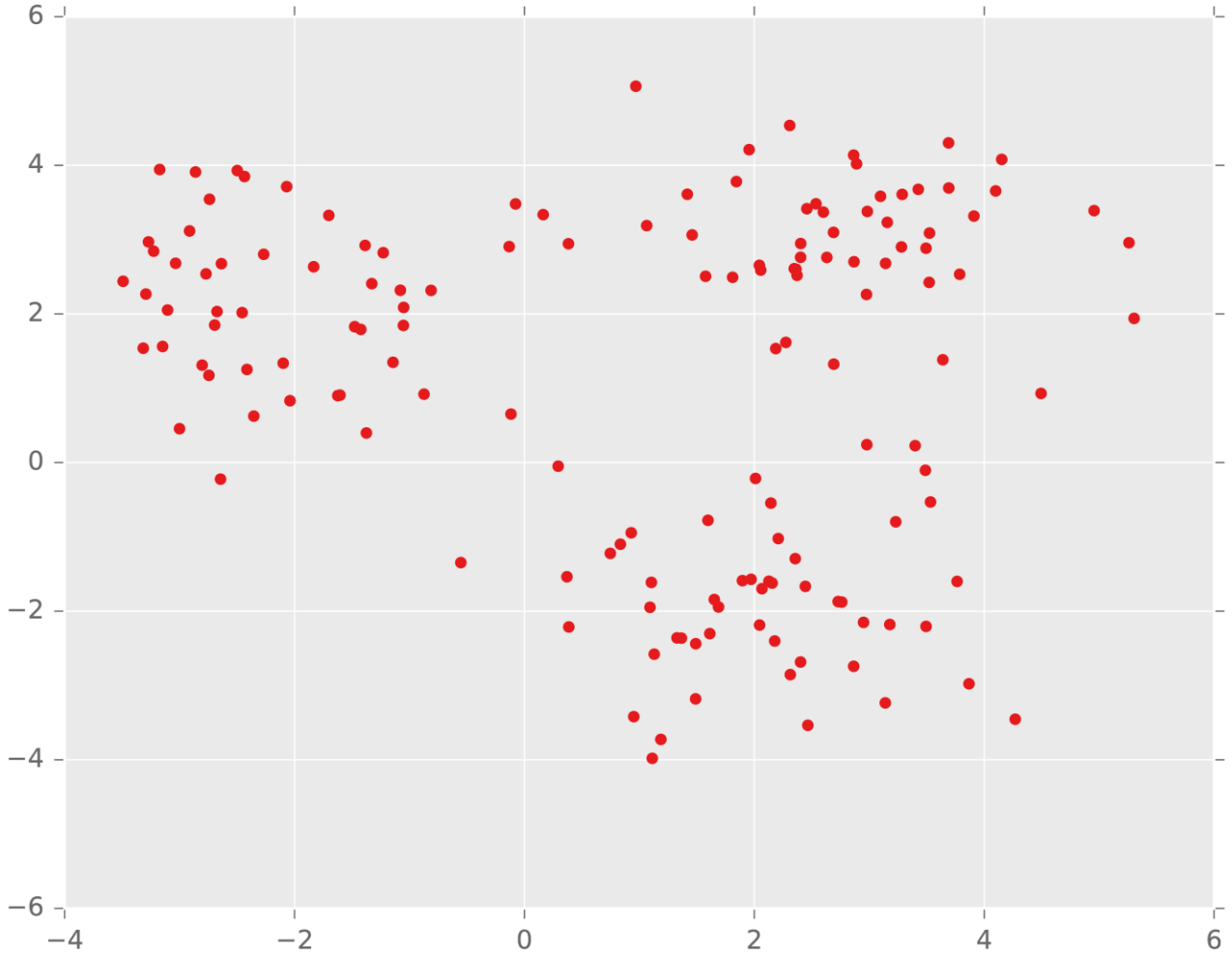


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

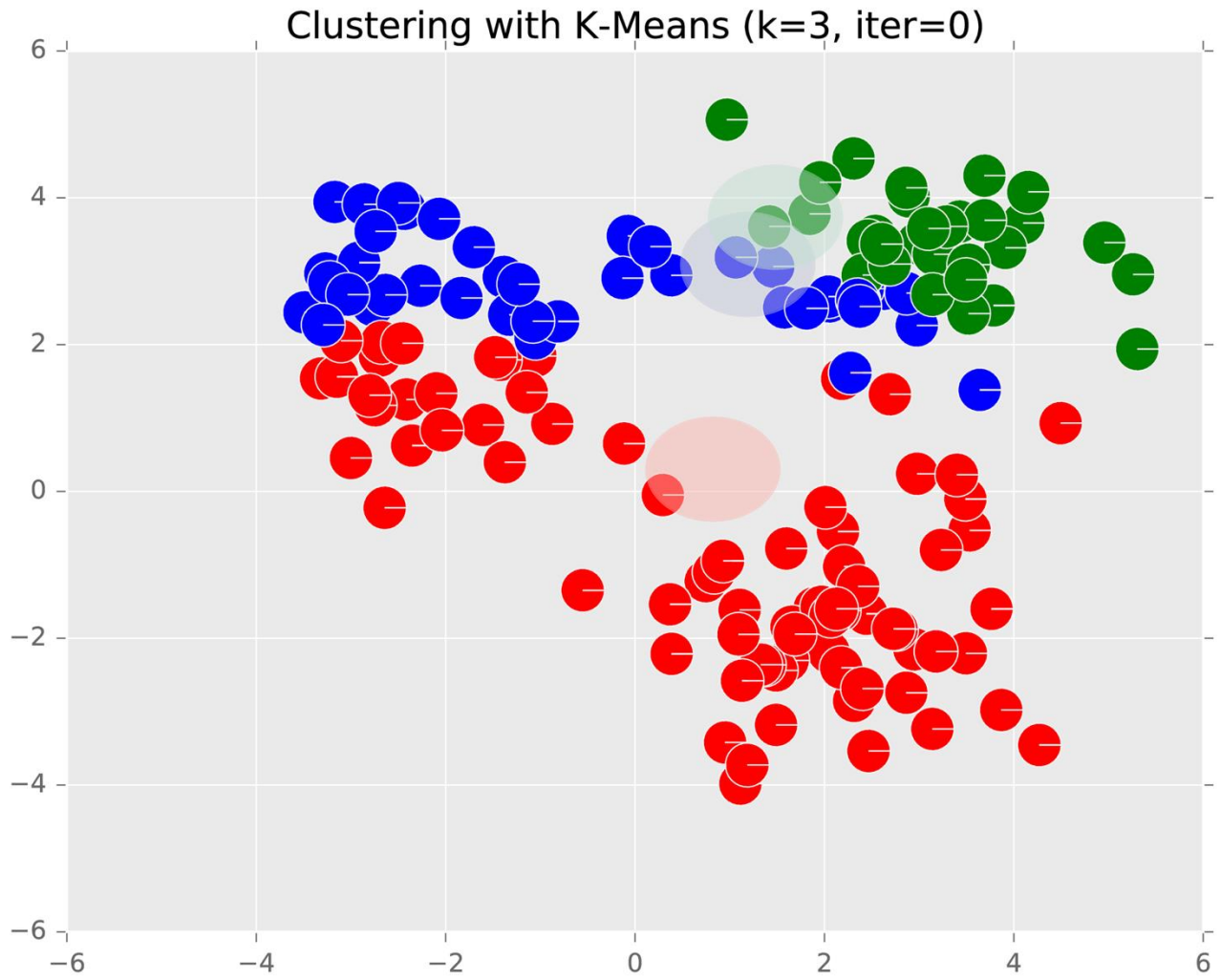


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

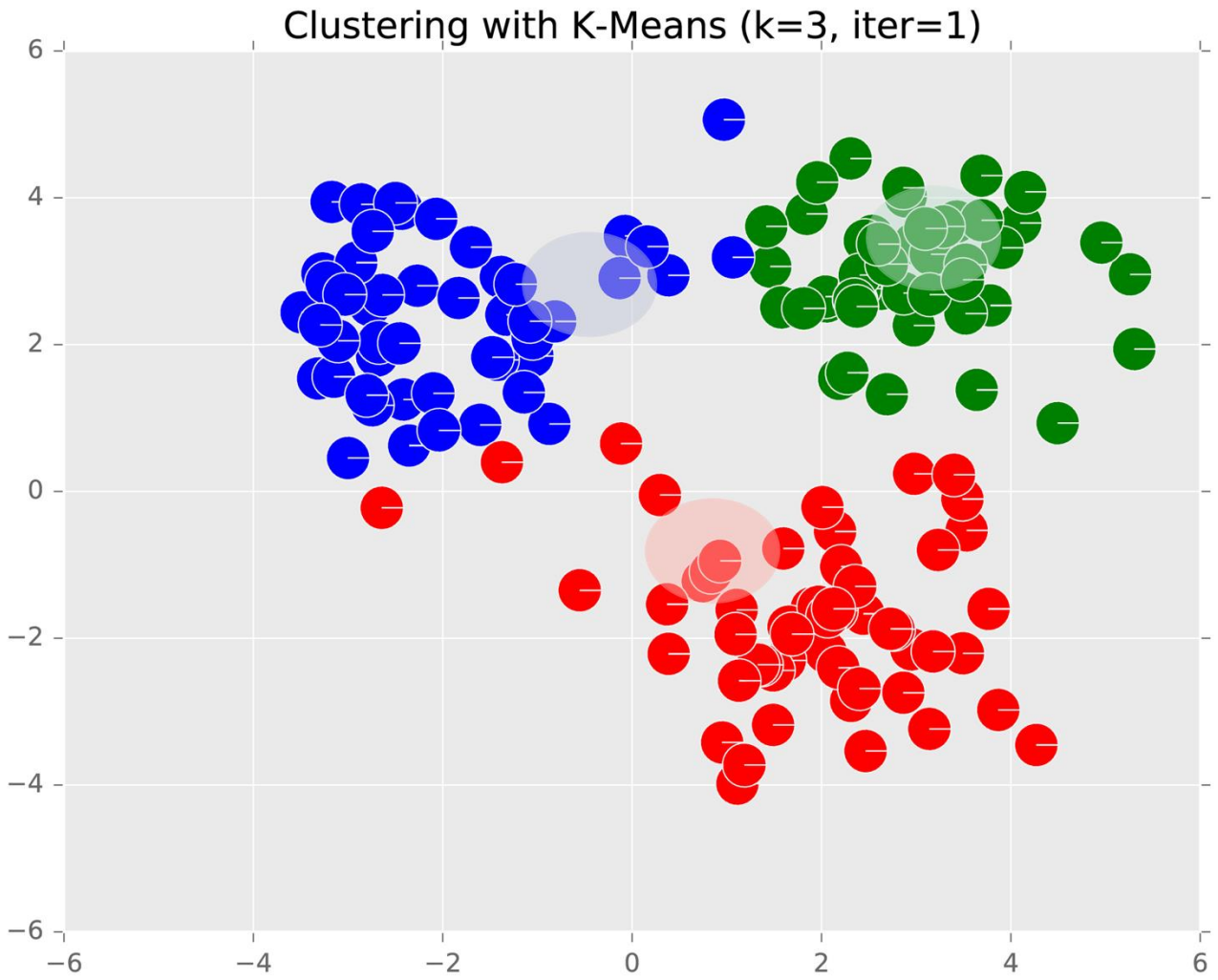


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

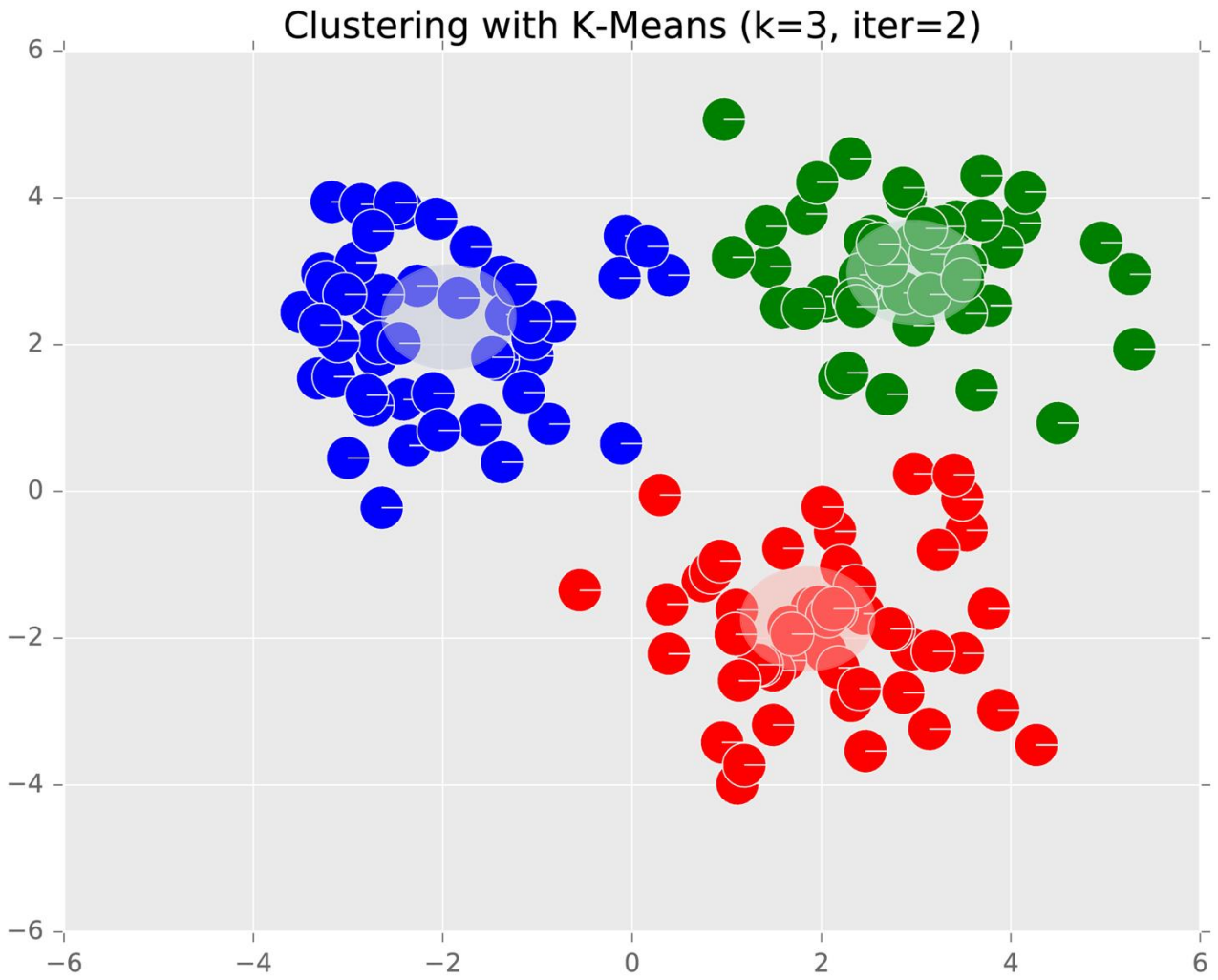


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

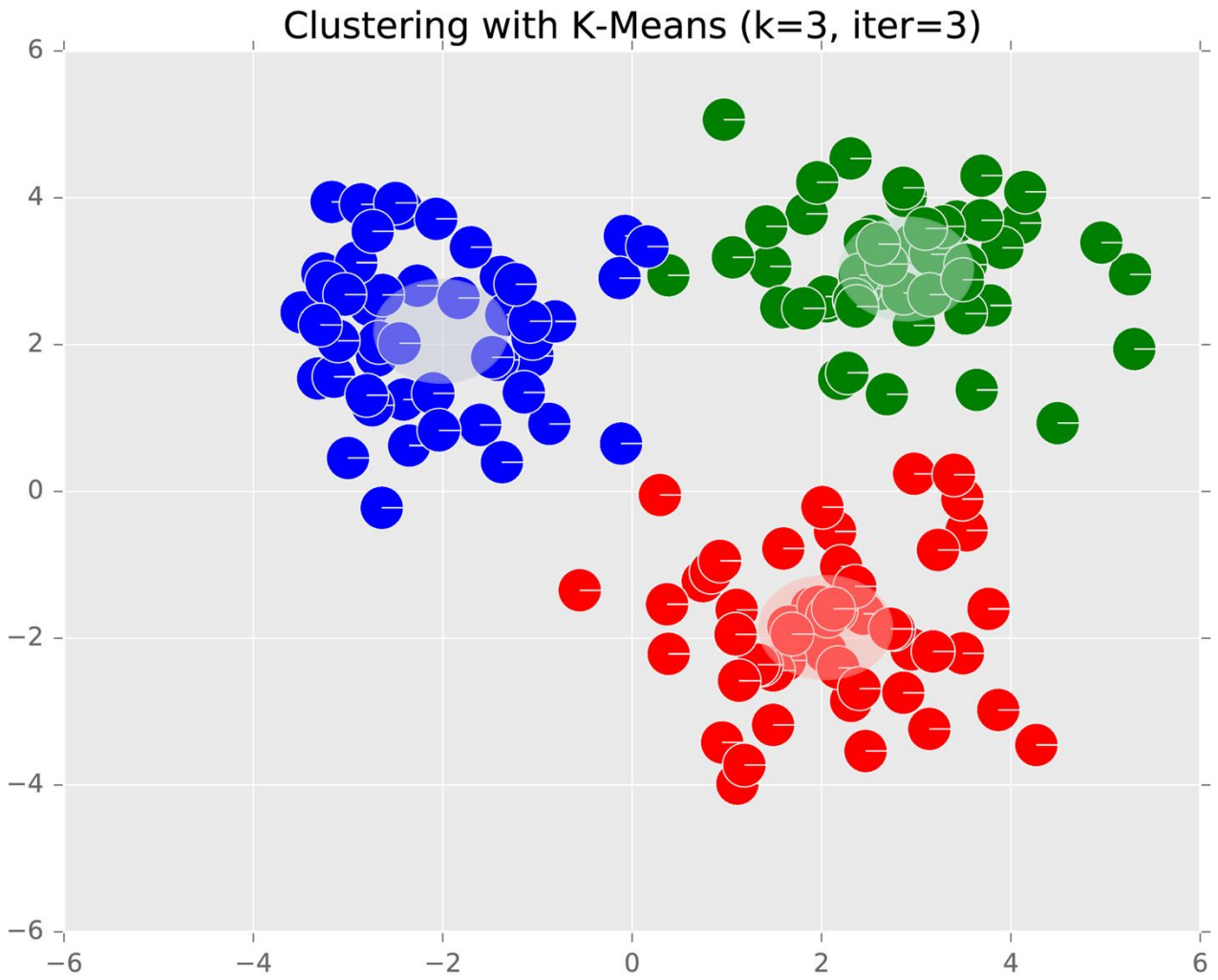


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

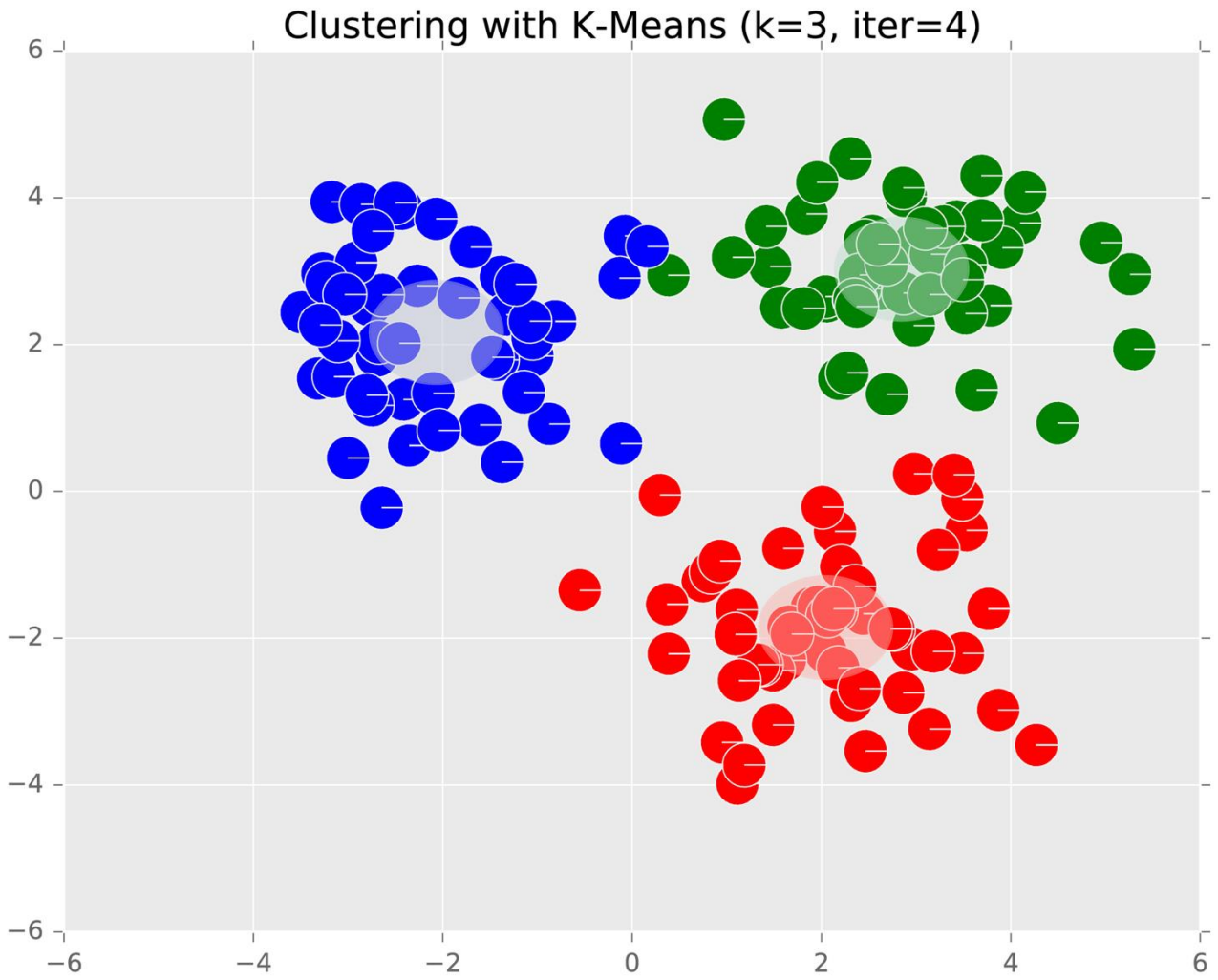


Figure courtesy of Matt Gormley

K-means: Example ($K = 3$)

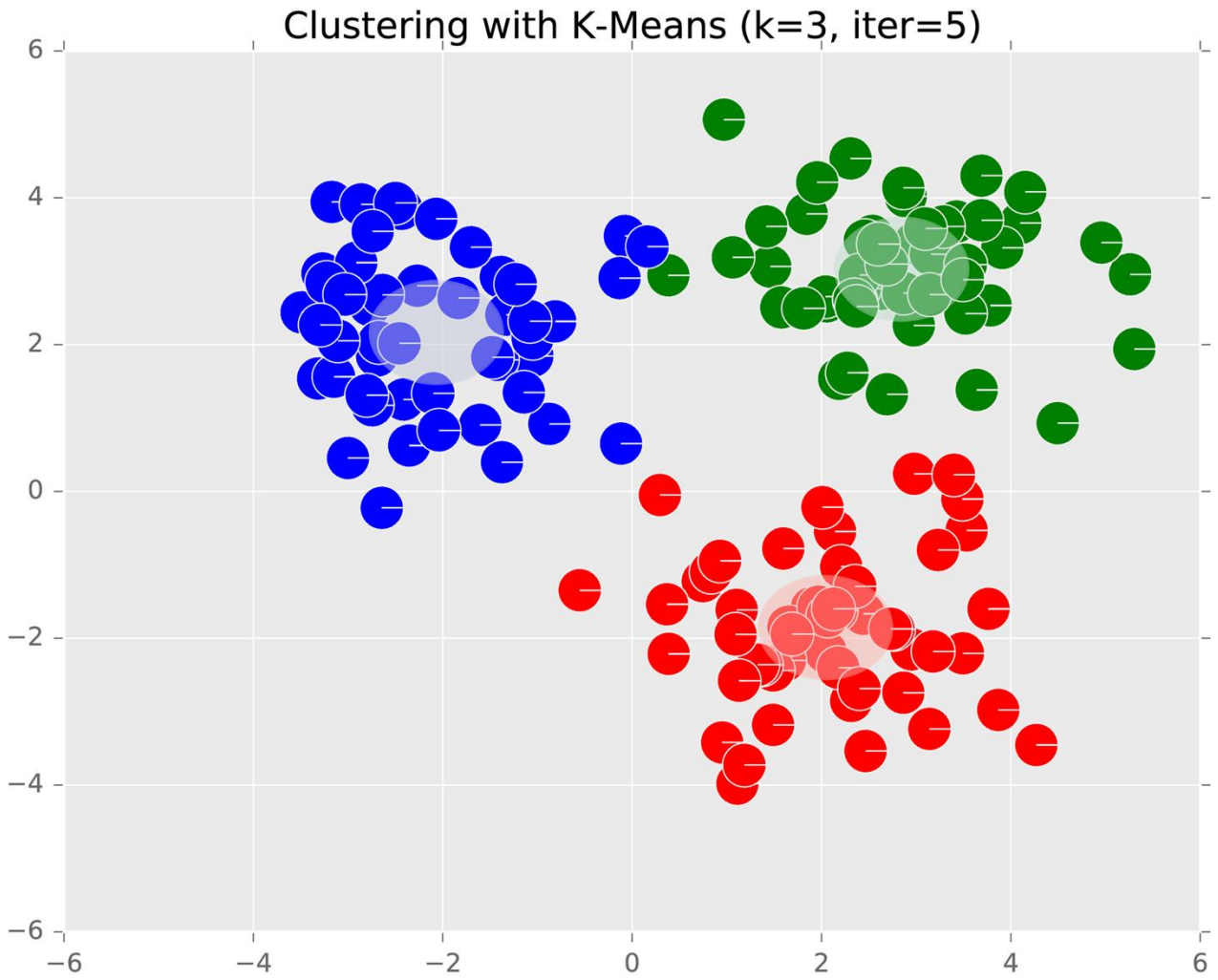


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

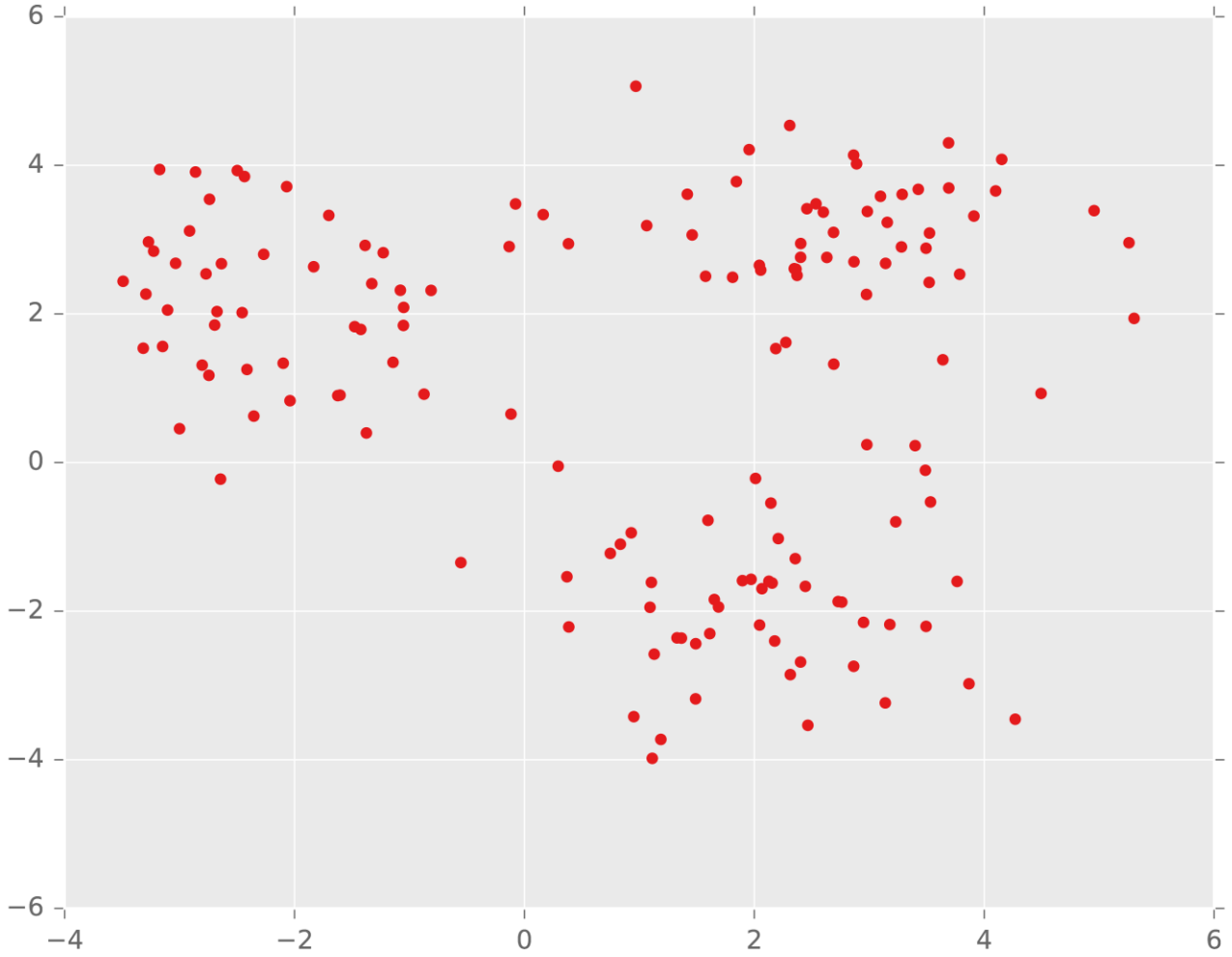


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

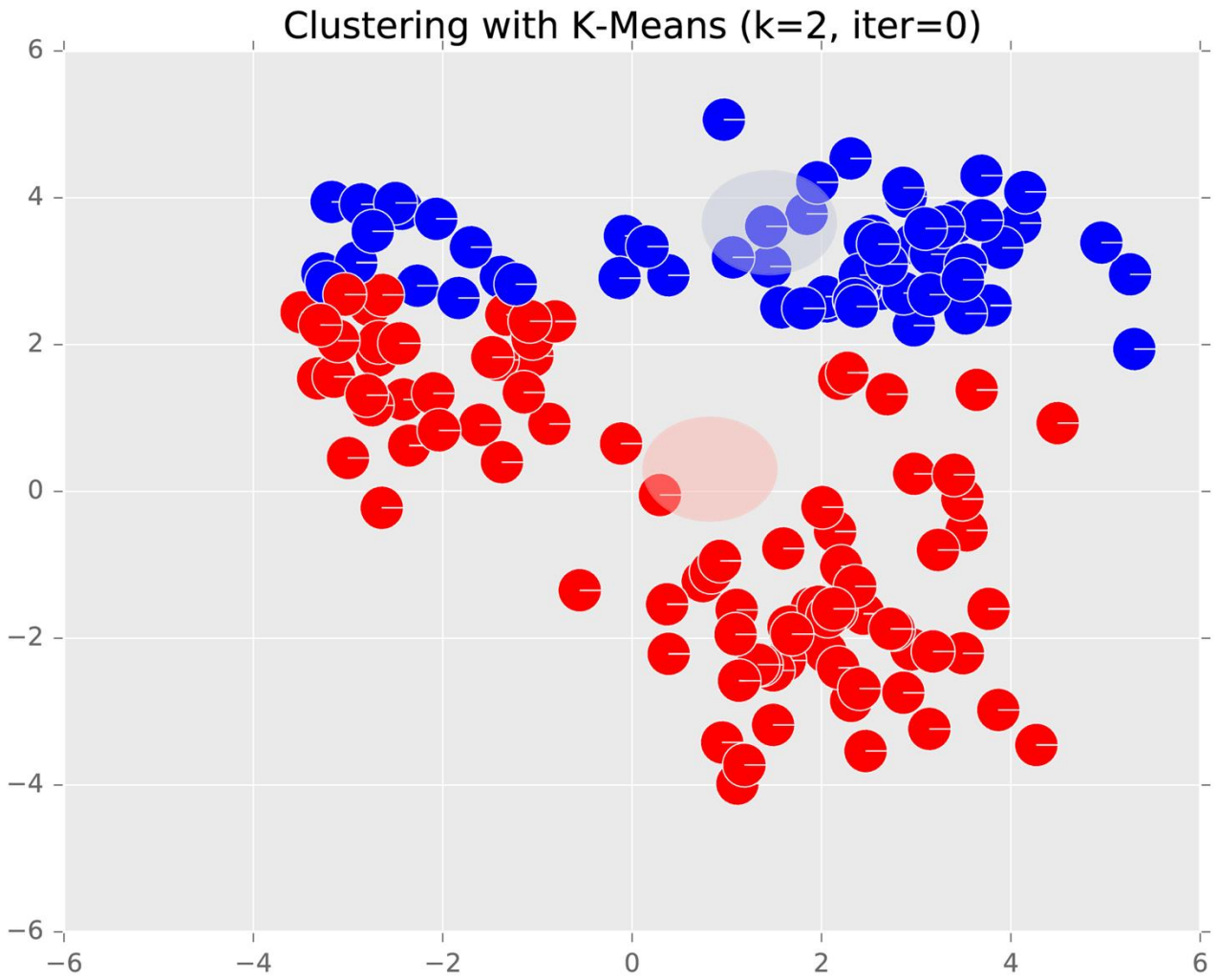


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

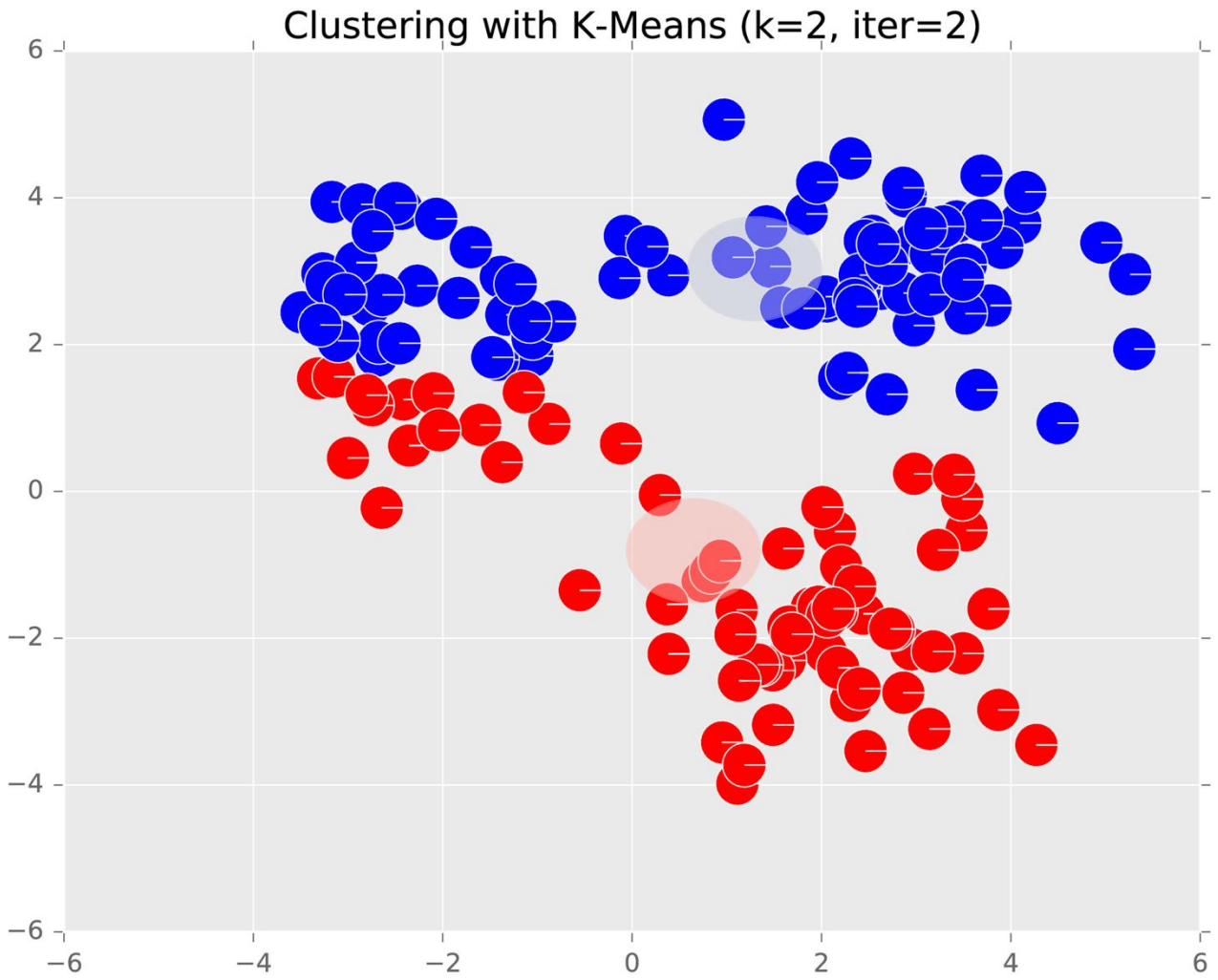


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

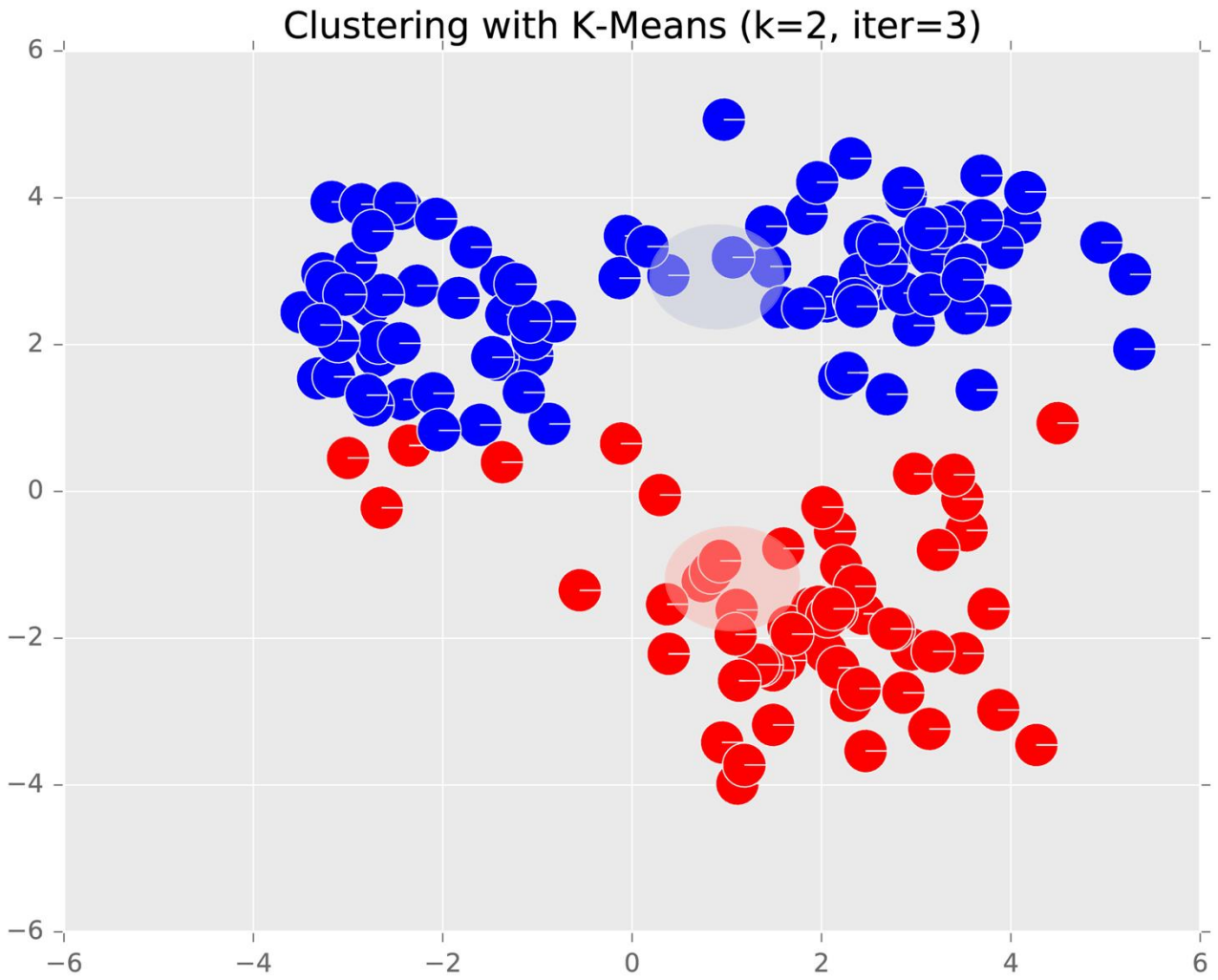


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

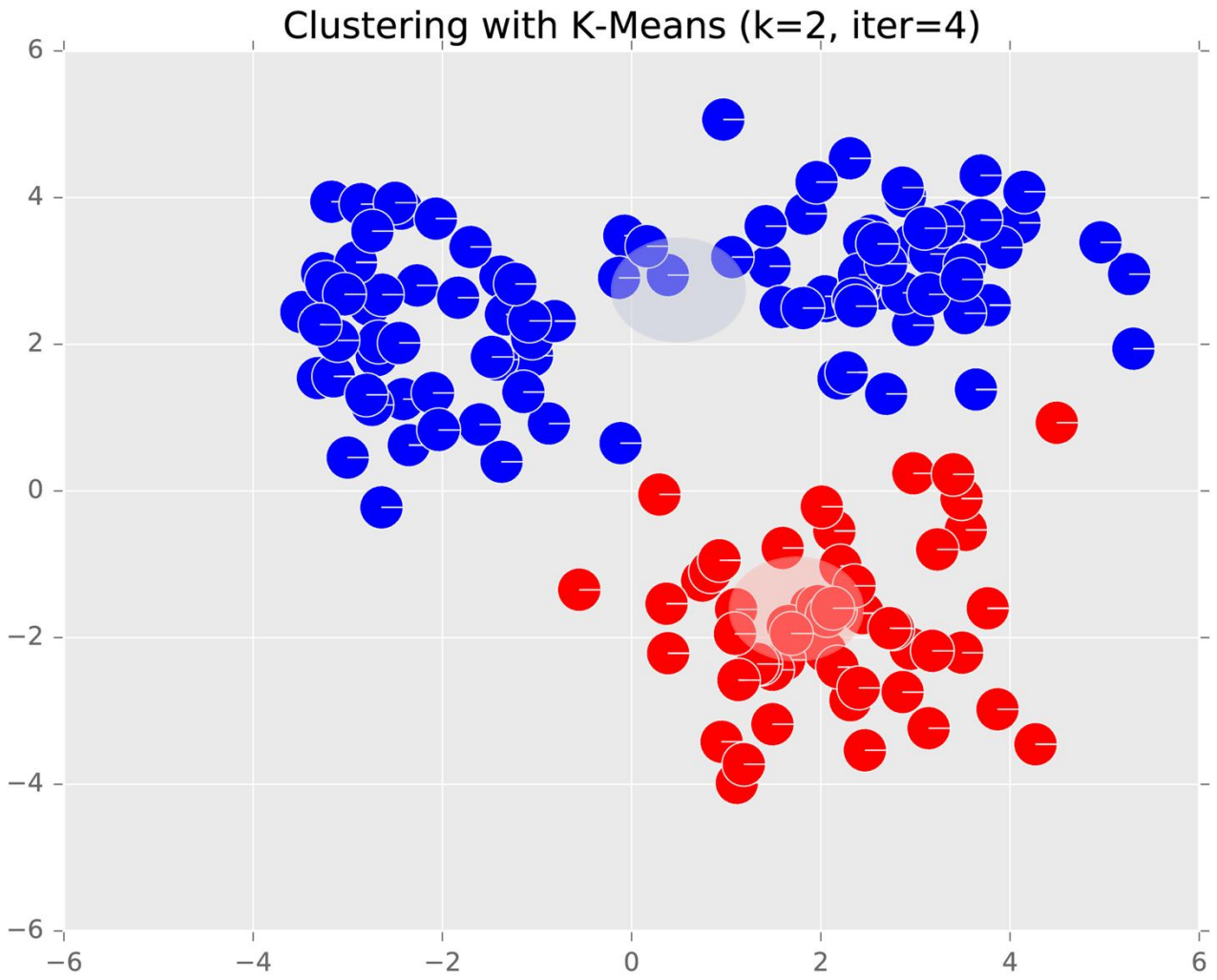


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

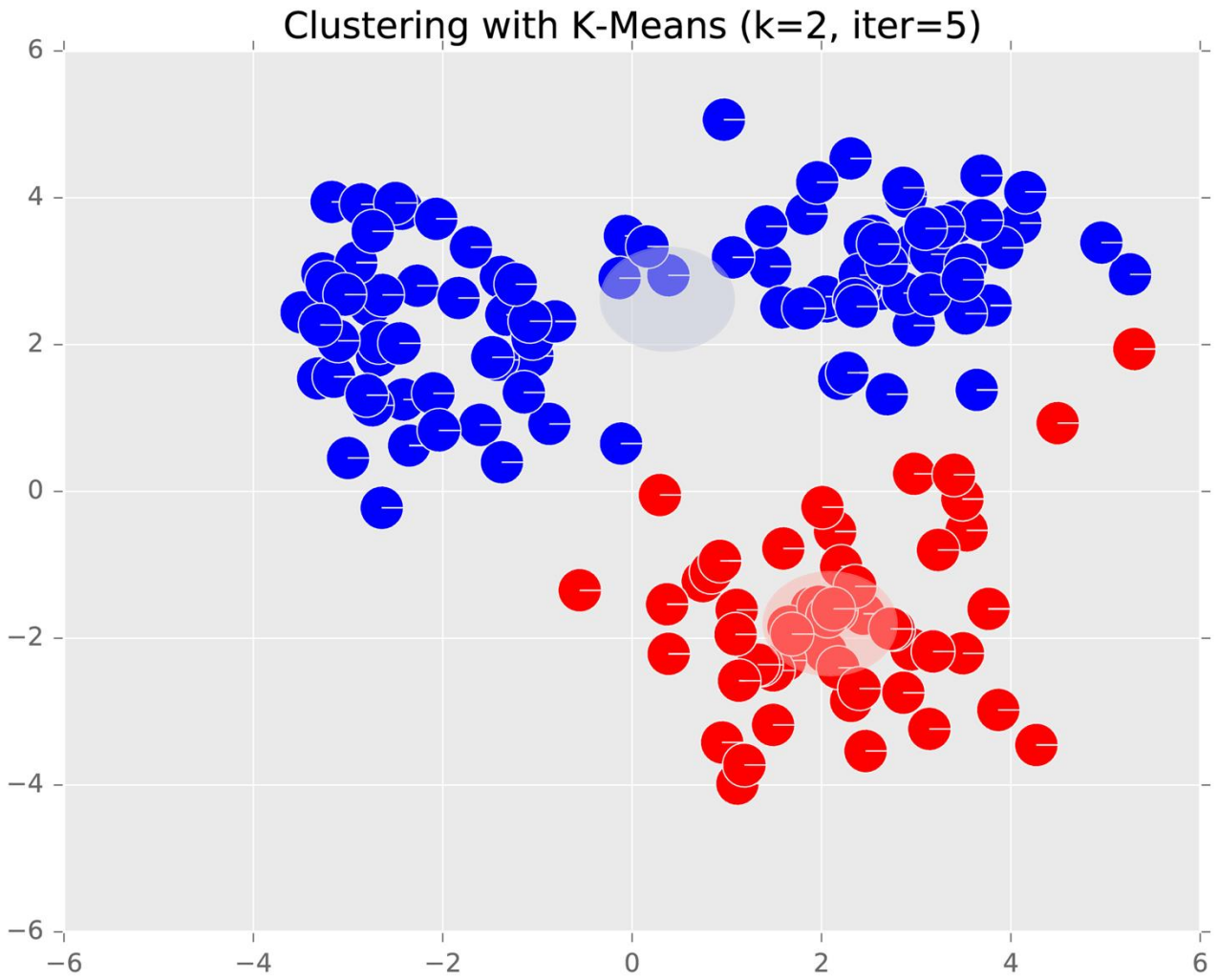


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

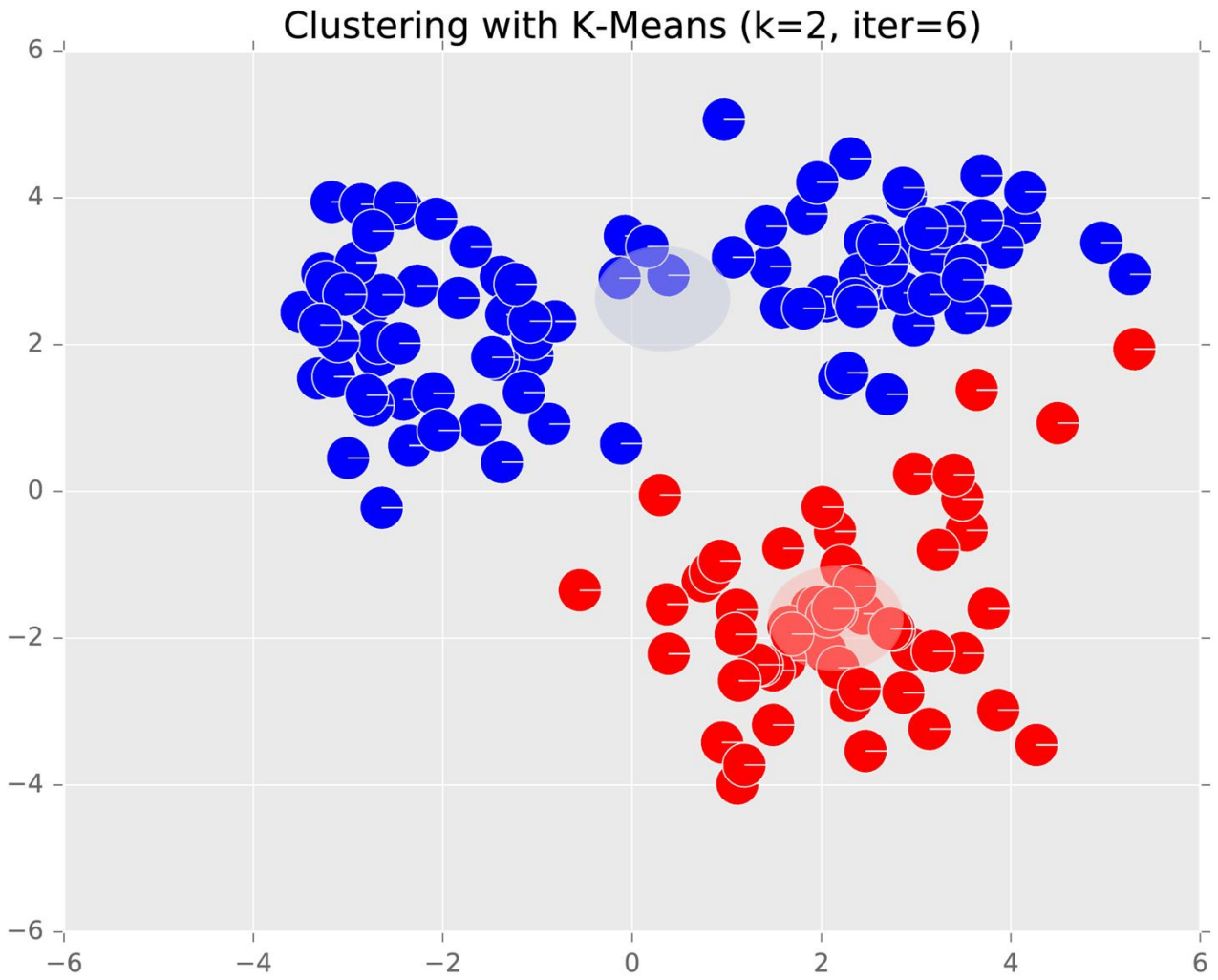


Figure courtesy of Matt Gormley

K-means: Example ($K = 2$)

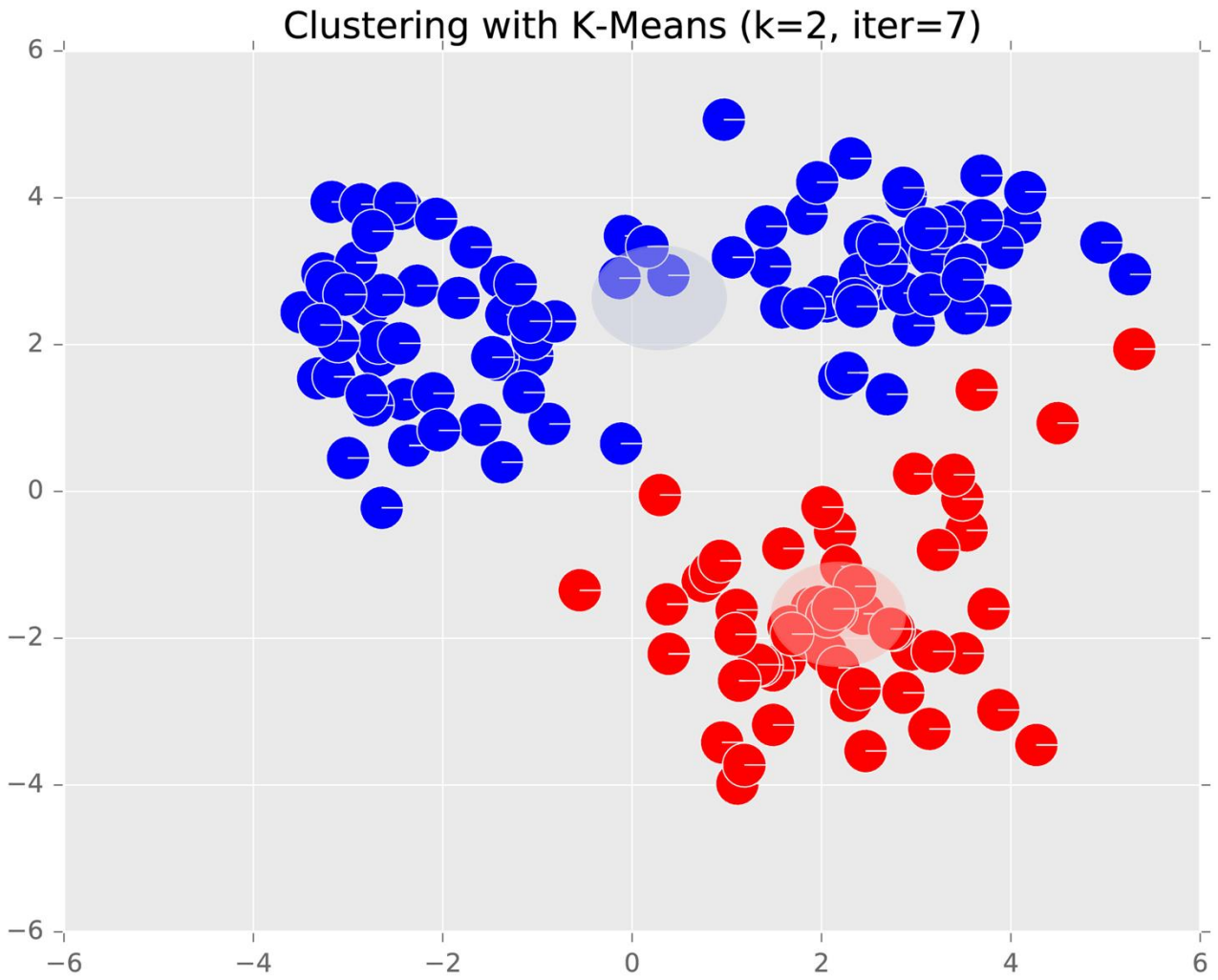
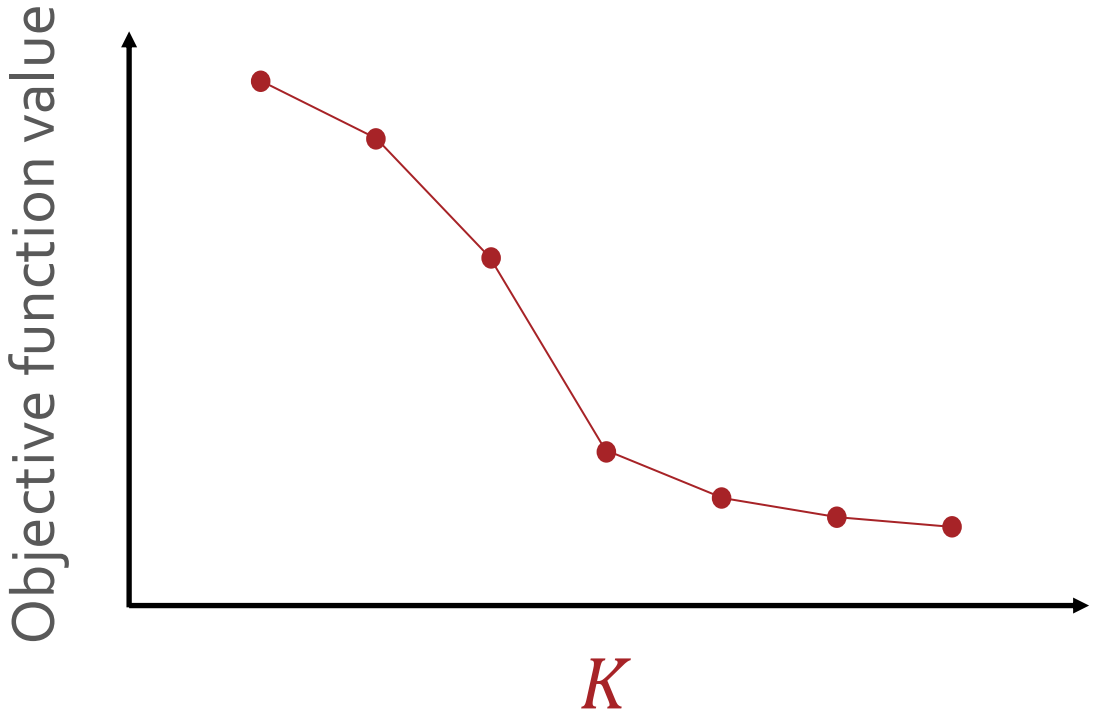


Figure courtesy of Matt Gormley

Setting K

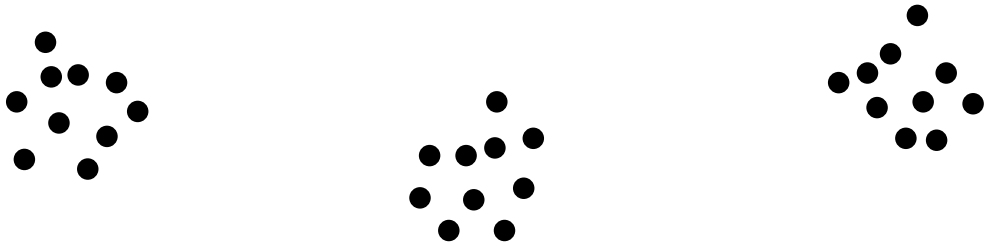
- Idea: choose the value of K that minimizes the objective function



- Better idea: look for the characteristic "elbow" or largest decrease when going from $K-1$ to K

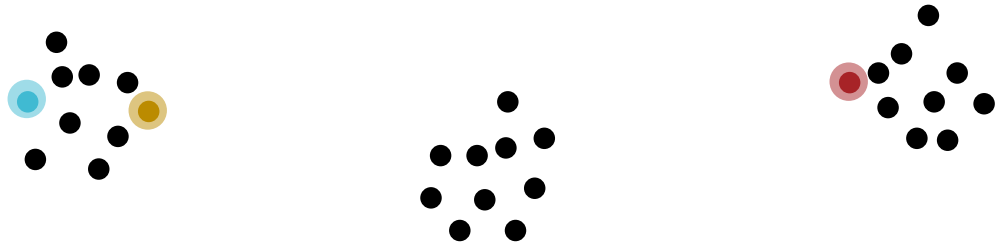
Initializing K-means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



Initializing K-means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)

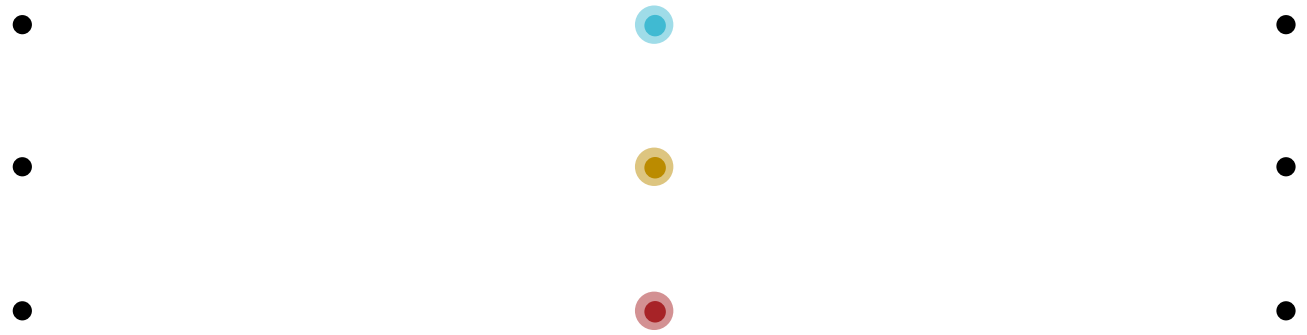


Initializing K-means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)

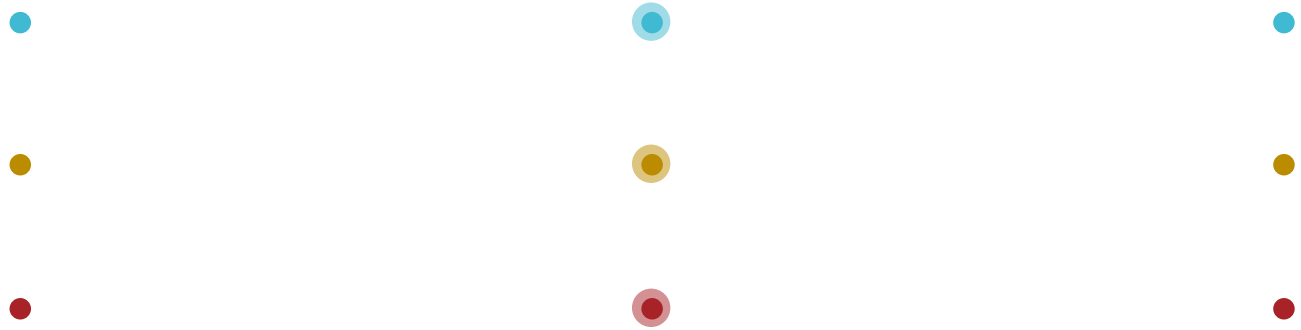
Initializing K-means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



Initializing K-means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



- Lloyd's method converges to a local minimum and that local minimum can be arbitrarily bad (relative to the optimal clusters)
- Intuition: want initial cluster centers to be far apart from one another

K-means++ (Arthur and Vassilvitskii, 2007)

1. Choose the first cluster center randomly from the data points.
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center.
3. Select the next cluster center proportional to $D(\boldsymbol{x})^2$.
4. Repeat 2 and 3 $K-1$ times.

- K-means++ achieves a $O(\log K)$ approximation to the optimal clustering in expectation
- Both Lloyd's method and K-means++ can benefit from multiple random restarts.

Myth Busters

MythBusters #1

“k-means always finds the best clustering.”

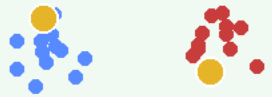
Myth: Lloyd’s method finds the global optimum of the k-means objective.

Reality: Lloyd’s method converges to a local minimum, and initialization can matter a lot.

Lecture takeaway: use smarter initialization (k-means++) and often multiple random restarts.

random init

k-means++ / restart



Short version: initialization changes the local minimum you end up in.

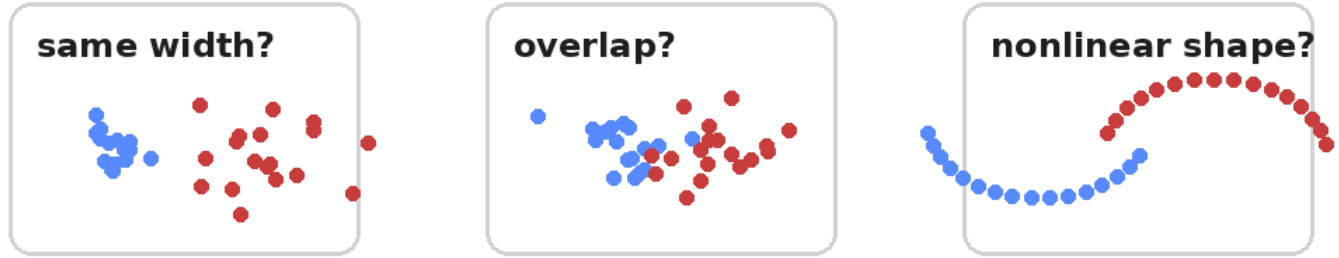
Myth Busters

MythBusters #2

“k-means can model any cluster shape.”

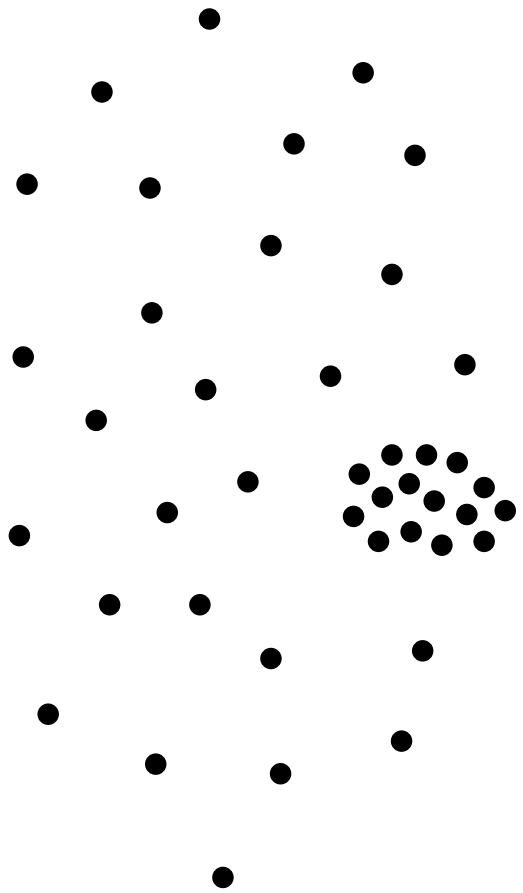
Myth: if you choose K well, k-means can fit essentially any clustering structure.

Reality: the lecture’s key limitations of k-means are: no overlap, roughly equal-width clusters, and linear separability.



This is why GMMs and other clustering models can be worth the extra complexity.

Shortcomings of K-means



- Clusters cannot overlap
- Clusters must all be of the same “width”
- Clusters must be linearly separable

Probabilistic or "Soft" Assignments

- Instead of $z^{(i)}$ being a deterministic scalar, let $\boldsymbol{z}^{(i)}$ be a 1-of-K vector indicating cluster membership
  - For example, $z^{(1)} = [0, 1, 0, \dots, 0]$ indicates that the first data point belongs to the second cluster
  - Let $\pi_k \coloneqq p\left(z_k^{(i)} = 1\right)$

Gaussian Mixture Models (GMMs)

Assume the following data-generating model for our dataset, $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$

1. Sample a cluster at random:
   $p\left(z_k^{(i)} = 1\right) = \pi_k$
2. Sample a data point from the chosen cluster:
   $p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1\right) \sim N(\mu_k, \Sigma_k)$

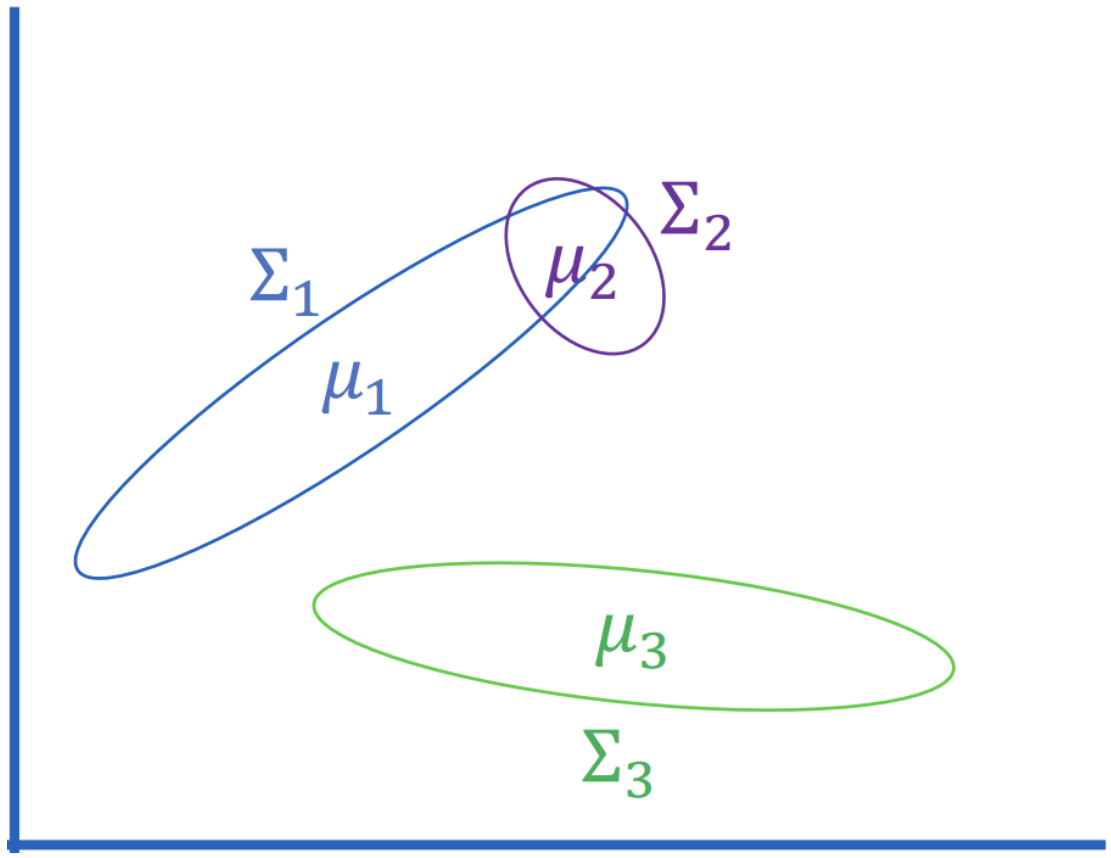
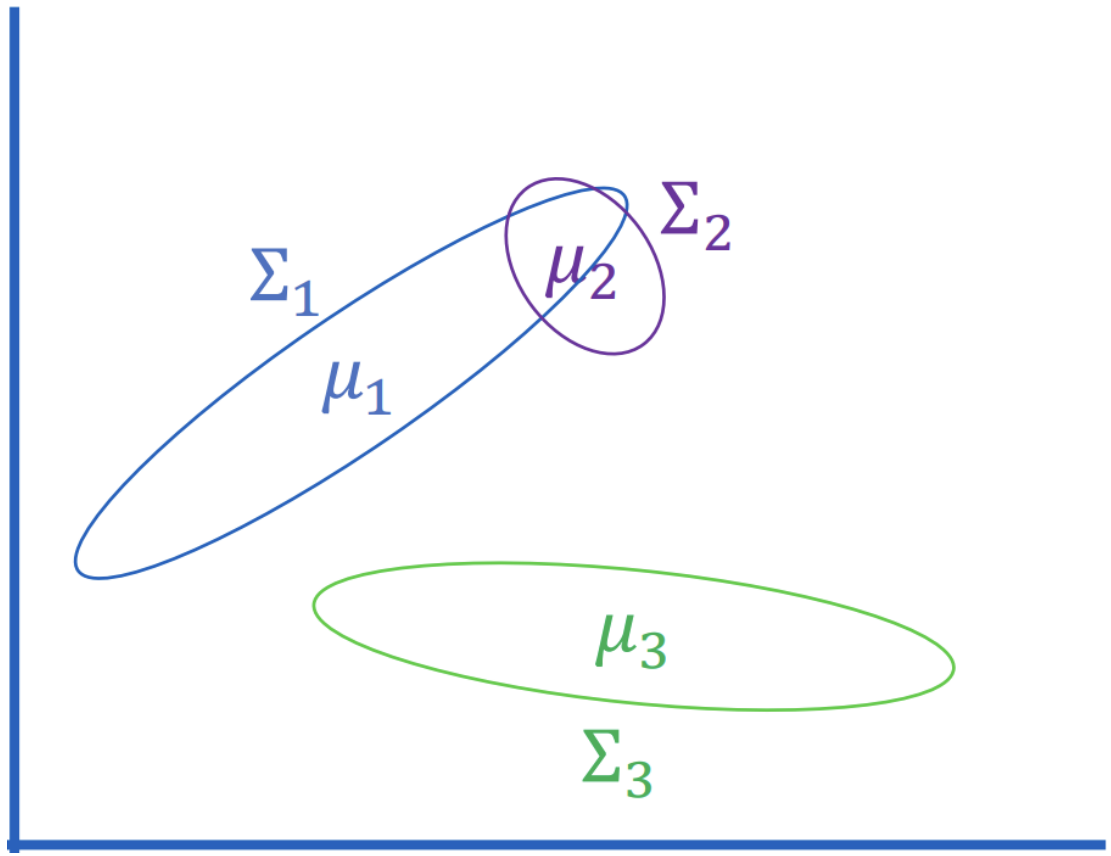


Figure courtesy of Pat Virtue

Gaussian Mixture Models (GMMs)

Assume the following data-generating model for our dataset, $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$

1. Sample a cluster at random:
   $p\left(z_k^{(i)} = 1\right) = \pi_k$
2. Sample a data point from the chosen cluster:
   $p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1\right) \sim N(\mu_k, \Sigma_k)$



Let $\theta = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K\}$

Figure courtesy of Pat Virtue

Maximizing the Likelihood?

- The log likelihood of $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\}_{i=1}^{N}$ is

$$\ell(\theta|\mathcal{D}) = \log \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}|\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i)}, \theta) + \log p(\boldsymbol{z}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log \prod_{k=1}^{K} p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right)^{z_k^{(i)}} + \log \prod_{k=1}^{K} p\left(z_k^{(i)} = 1\middle|\theta\right)^{z_k^{(i)}}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \log p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right) + \sum_{k=1}^{K} z_k^{(i)} \log p\left(z_k^{(i)} = 1\middle|\theta\right)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \left(\log N(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k) + \log \pi_k\right)$$

Maximizing the Complete Likelihood

- The log complete likelihood of $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\}_{i=1}^{N}$ is

$$\ell_c(\theta|\mathcal{D}_c) = \log \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\middle|\theta\right) = \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\middle|\theta\right)$$

$$= \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)}\middle|\boldsymbol{z}^{(i)}, \theta\right) + \log p\left(\boldsymbol{z}^{(i)}\middle|\theta\right)$$

$$= \sum_{i=1}^{N} \log \prod_{k=1}^{K} p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right)^{z_k^{(i)}} + \log \prod_{k=1}^{K} p\left(z_k^{(i)} = 1\middle|\theta\right)^{z_k^{(i)}}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \log p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right) + \sum_{k=1}^{K} z_k^{(i)} \log p\left(z_k^{(i)} = 1\middle|\theta\right)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \left(\log N\left(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k\right) + \log \pi_k\right)$$

Maximizing the Complete Likelihood is easy but requires $z^{(i)}$!

- The log complete likelihood of $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}\}_{i=1}^{N}$ is

$$\ell_c(\theta|\mathcal{D}_c) = \log \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}|\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i)}, \theta) + \log p(\boldsymbol{z}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log \prod_{k=1}^{K} p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right)^{z_k^{(i)}} + \log \prod_{k=1}^{K} p\left(z_k^{(i)} = 1\middle|\theta\right)^{z_k^{(i)}}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \log p\left(\boldsymbol{x}^{(i)}\middle|z_k^{(i)} = 1, \theta\right) + \sum_{k=1}^{K} z_k^{(i)} \log p\left(z_k^{(i)} = 1\middle|\theta\right)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \left(\log N\left(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k\right) + \log \pi_k\right)$$

- Parameters decoupled → set partial derivatives equal to 0

Maximizing the Marginal Likelihood

- The log *marginal* likelihood of $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ is

$$\ell(\theta|\mathcal{D}) = \log \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}|\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log \sum_{\boldsymbol{z}^{(i)}} p(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i)}, \theta) p(\boldsymbol{z}^{(i)}|\theta)$$

$$= \sum_{i=1}^{N} \log \sum_{\boldsymbol{z}^{(i)}} \prod_{k=1}^{K} \left(p\left(\boldsymbol{x}^{(i)}|z_k^{(i)} = 1, \theta\right) p\left(z_k^{(i)} = 1 \middle| \theta\right) \right)^{z_k^{(i)}}$$

$$= \sum_{i=1}^{N} \log \sum_{\boldsymbol{z}^{(i)}} \prod_{k=1}^{K} \left(N(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k) \pi_k \right)^{z_k^{(i)}}$$

- Parameters coupled and *constrained* → gradient ascent is possible but complicated and slow to converge

Recipe for GMMs

- Define a model and model parameters
  - Assume K Gaussian clusters
  - Parameters: $\theta = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K\}$
- Write down an objective function
  - Maximize the log marginal likelihood

$$\ell(\theta|\mathcal{D}) = \log \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}|\theta\right)$$

- Optimize the objective w.r.t. the model parameters
  - Expectation-maximization

Expectation-Maximization for GMMs: Intuition

- Insight: if we knew the cluster assignments, $\boldsymbol{z}^{(i)}$, we could maximize the log complete likelihood instead of the log marginal likelihood
- Idea: replace $\boldsymbol{z}^{(i)}$ in the log complete likelihood with our "best guess" for $\boldsymbol{z}^{(i)}$ given the parameters and the data
- Observation: changing the parameters changes our "best guess" and vice versa
- Approach: iterate between updating our "best guess" and updating the parameters

Expectation-Maximization for GMMs

- Iterative algorithm that alternates between two steps
  - Expectation or E-step: for fixed parameters θ, compute the *expected* assignment vectors conditioned on θ and the data set $\mathcal{D}$

$$E\left[z_k^{(i)} | \boldsymbol{x}^{(i)}, \theta\right] = p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \; \forall \, i \text{ and } k$$

  - Maximization or M-step: for fixed assignment vectors $\boldsymbol{z}^{(i)}$, set the parameters θ to *maximize* the complete log likelihood of the data set $\mathcal{D}$
- Under the hood: EM performs block-coordinate ascent on a lower bound of the log marginal likelihood

E-Step for GMMs

$$p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) = \frac{p\left(z_k^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}{p(\boldsymbol{x}^{(i)} | \theta)}$$

$$= \frac{p\left(z_k^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}{\sum_{j=1}^{K} p\left(z_j^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}$$

$$= \frac{\pi_k N\left(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j N\left(\boldsymbol{x}^{(i)}; \mu_j, \Sigma_j\right)} \quad \forall\, i \text{ and } k$$

M-Step for GMMs

Let $N_k = \sum_{i=1}^{N} p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right)$

$$\pi_k = \frac{N_k}{N}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N} p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \boldsymbol{x}^{(i)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N} p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \left(\boldsymbol{x}^{(i)} - \mu_k\right)\left(\boldsymbol{x}^{(i)} - \mu_k\right)^T$$

GMM Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{N}, K$

1. Initialize all parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K$
2. While NOT CONVERGED
   a. E-step: compute $p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \forall\ i \text{ and } k$
   b. M-step: update the parameters

- Output: parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K$ and assignments probabilities $p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \forall\ i \text{ and } k$

Initializing EM for GMMs

- Common heuristics for initialization
  - Cluster proportions typically initialized to be uniform
  - Cluster means
    - Randomly select data points to be cluster centers
    - Randomly sample locations in the range spanned by the data
  - Cluster covariances
    - Identity (or scaled identity) matrix
    - Random positive diagonal matrix
    - Randomly sample L, a lower triangular matrix with positive diagonal entries, and set to LL^{T}
    - Set to the empirical covariance of the data
  - Use multiple random restarts

Terminating EM for GMMs

- Common heuristics for termination
  - Stop if the log complete likelihood changes by less than some tolerance
  - Stop if the parameters and assignment probabilities change by less than some tolerance
  - Stop after a fixed number of iterations

GMMs: Example (Initial)

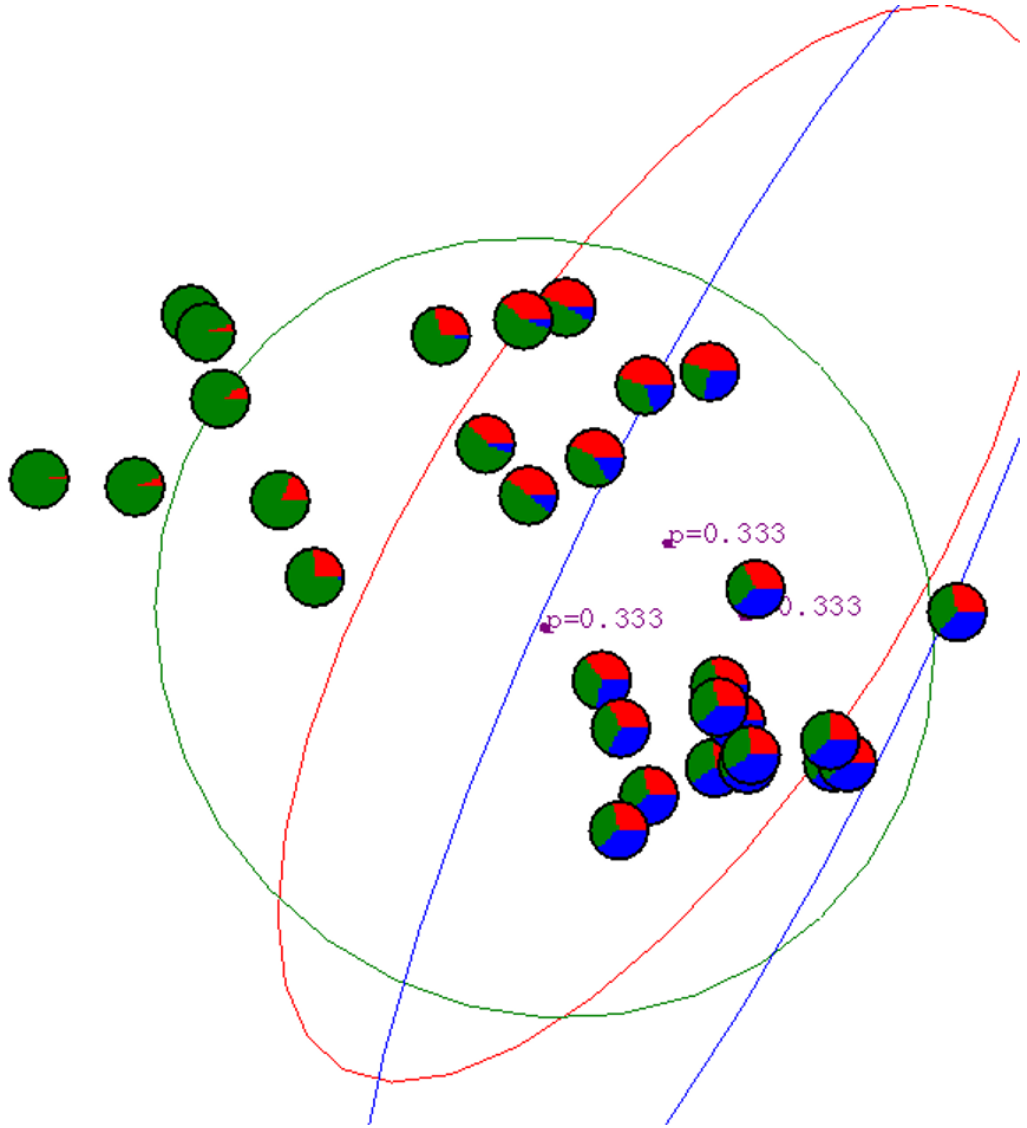


Figure courtesy of Pat Virtue

GMMs: Example (1 Iteration)

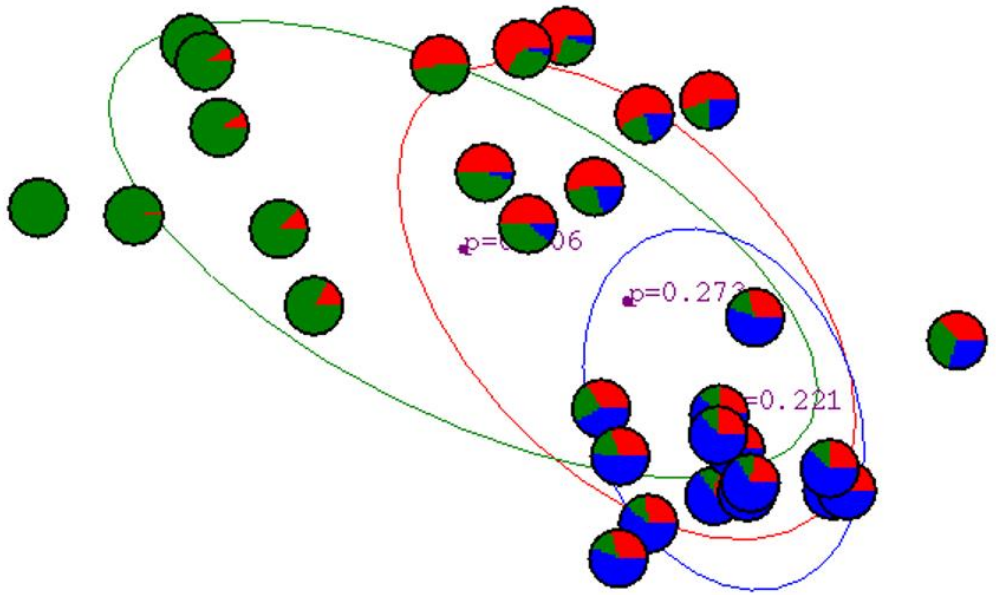


Figure courtesy of Pat Virtue

GMMs: Example (2 Iteration

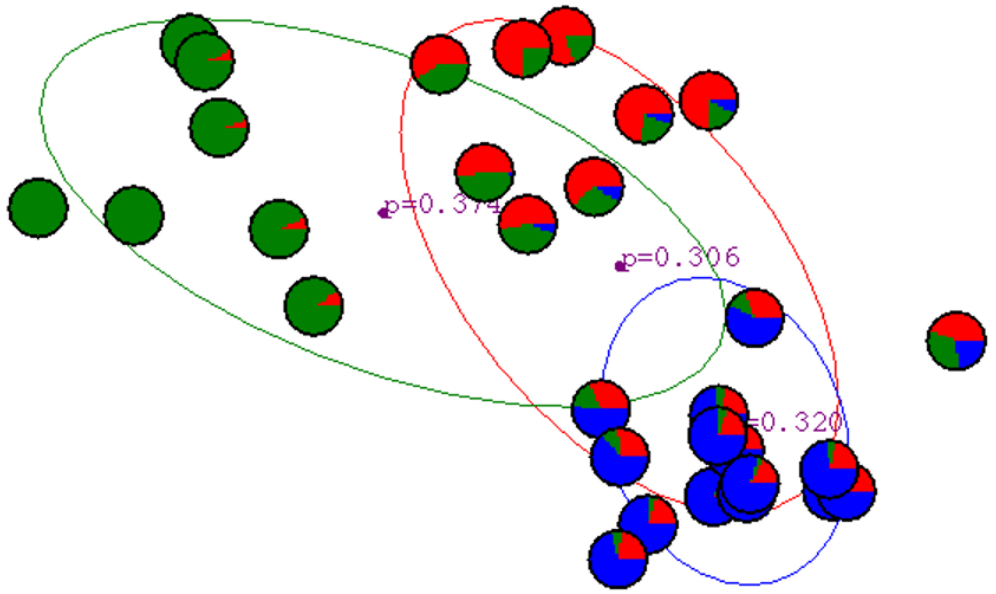


Figure courtesy of Pat Virtue

GMMs: Example (3 Iteration

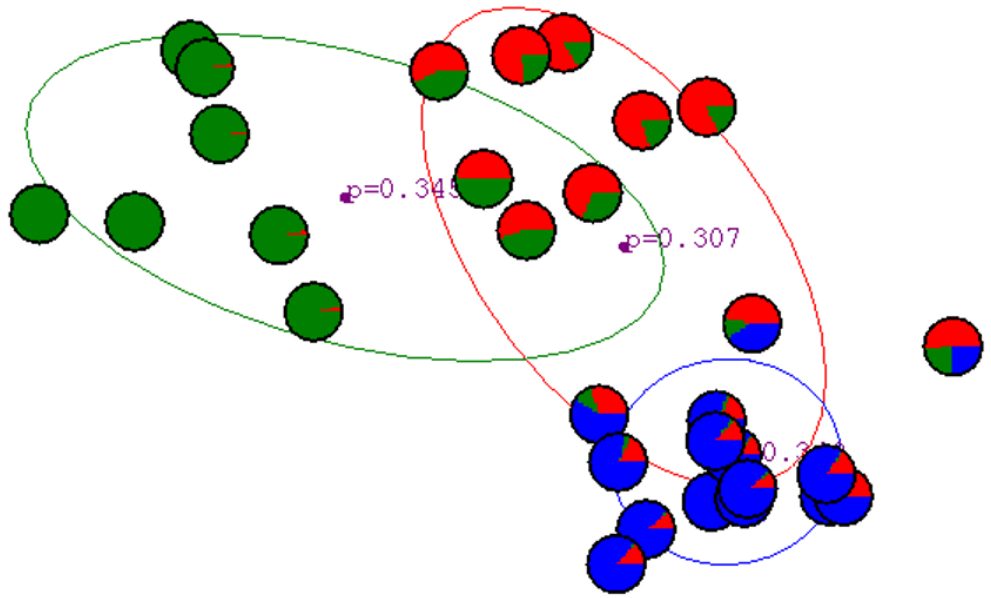


Figure courtesy of Pat Virtue

GMMs: Example (4 Iteration

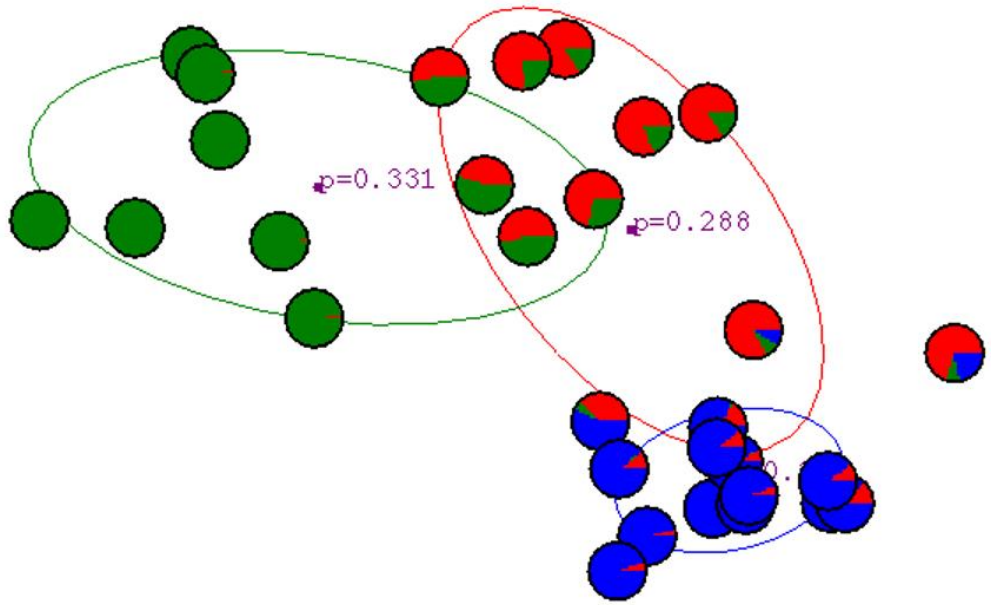


Figure courtesy of Pat Virtue

GMMs: Example (5 Iteration

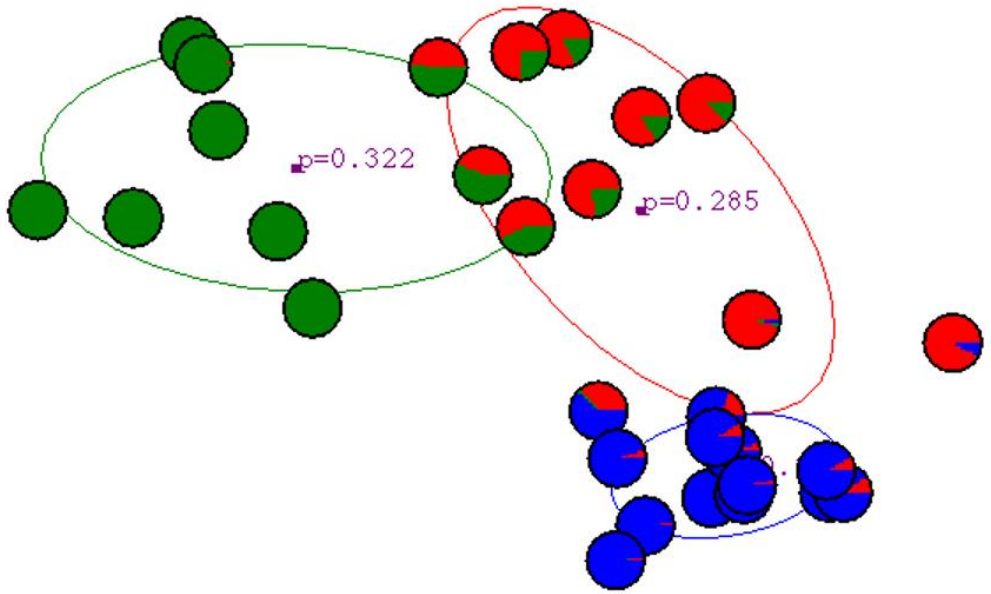


Figure courtesy of Pat Virtue

GMMs: Example (6 Iteration

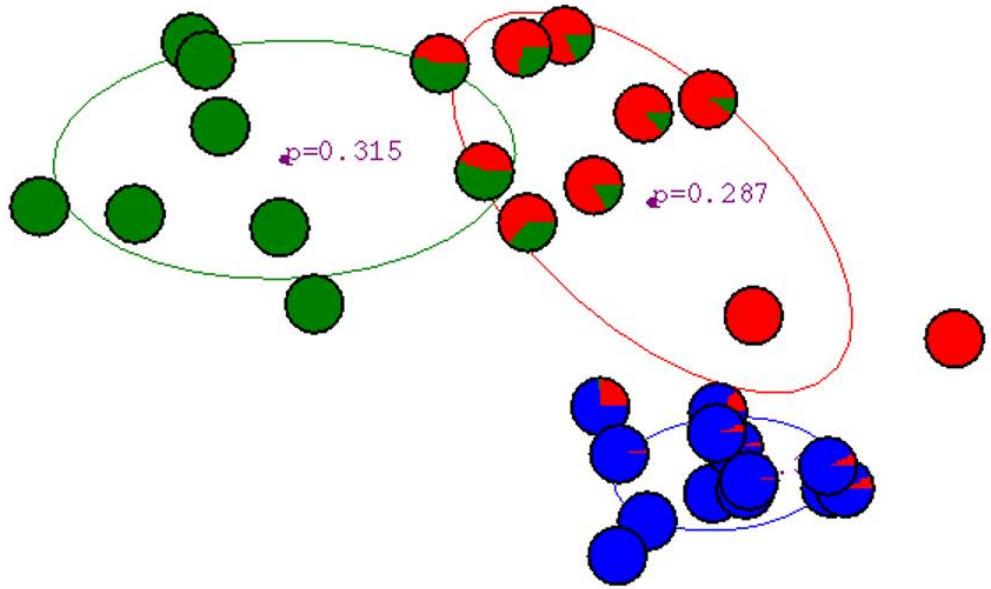


Figure courtesy of Pat Virtue

GMMs: Example (20 Iterations)

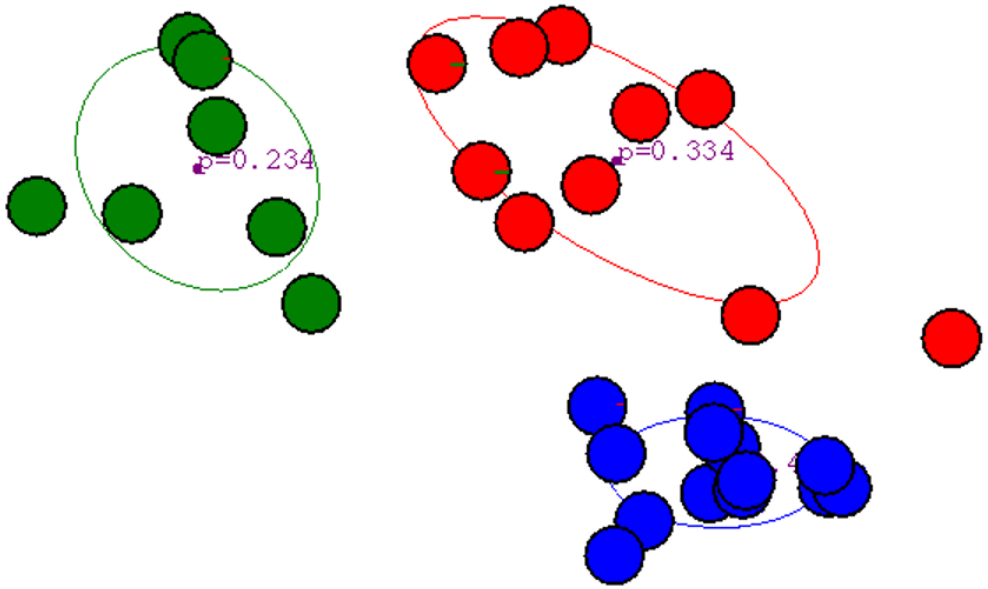


Figure courtesy of Pat Virtue

Quick Game

Two truths and a lie (clustering edition)

Vote for the lie: A, B, or C.

A. k-means uses block coordinate descent on a clustering objective.

B. GMMs use probabilistic assignments and are fit with EM.

C. k-means++ guarantees the global optimum of the k-means objective.

Quick Game

Two truths and a lie (clustering edition)

Vote for the lie: A, B, or C.

A. k-means uses block coordinate descent on a clustering objective.

B. GMMs use probabilistic assignments and are fit with EM.

C. k-means++ guarantees the global optimum of the k-means objective.

Reveal: C is the lie.

Key Takeaways

- Partition-based clustering
  - K-means (hard assignments)
    - Block-coordinate descent
    - Setting K
    - Initializing K means
  - Gaussian mixture models (probabilistic assignments)
    - Complete vs. marginal likelihood
    - Expectation-maximization for GMMs
    - Initializing EM for GMMs

Poll Link:

Final Poll

Final poll (pick one)

Which statement best captures why you might prefer a GMM over k-means on some datasets?

A. GMMs do not require choosing K.

B. GMMs guarantee the global optimum, while k-means does not.

C. GMMs allow soft assignments and cluster-specific covariances, so overlapping or unequal-width Gaussian clusters are possible.

D. GMMs are always faster to fit than k-means.

E. k-means and GMMs are equivalent except for the initialization step.