Practice

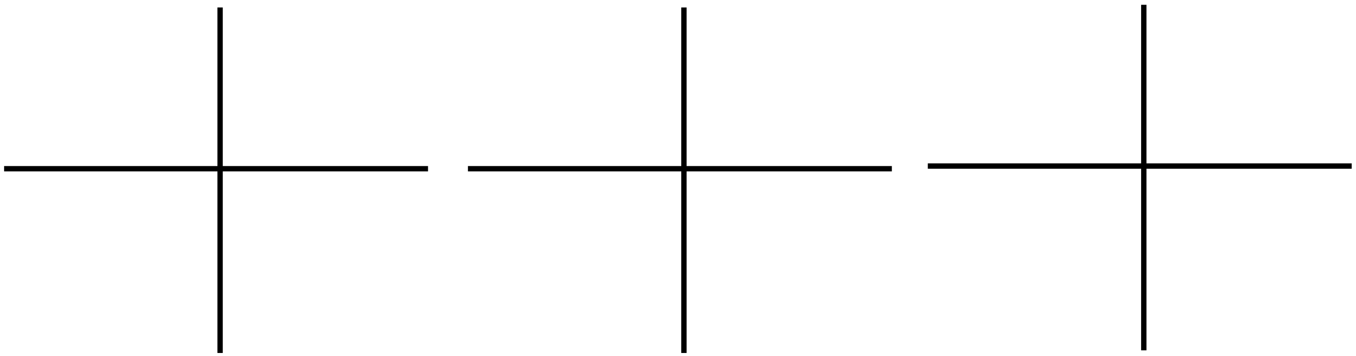
- Fill in the missing plots:

$$\Sigma = \mathbf{X}^T\mathbf{J}\mathbf{J}\mathbf{X} = \mathbf{Z}^T\mathbf{J}\mathbf{J}\mathbf{Z}$$

$$\mathbf{V}\mathbf{S}\mathbf{V}^T = \text{eig}(\Sigma) \qquad \mathbf{J} = I - \mathbf{1}\mathbf{1}^T/n$$

$$\mu_X = \mathbf{X}^T\mathbf{1}/n \qquad \mu_Z = \mathbf{Z}^T\mathbf{1}/n$$

$\mathbf{X}$ $\mathbf{Z}$ $\mu_X - \mu_Z$ $\mathbf{V}S^{-1/2}\mathbf{V}^T(\mu_X - \mu_Z)$



Matrix Completion

Machine Learning – CSE546

Kevin Jamieson
University of Washington

November 15, 2016

©Kevin Jamieson 2018

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{U} = [u_1, \ldots, u_r] \qquad \mathbf{V} = [v_1, \ldots, v_r]$$

$$\mathbf{A}^T\mathbf{A}v_i = \mathbf{S}_{i,i}^2 v_i$$

$$\mathbf{A}\mathbf{A}^T u_i = \mathbf{S}_{i,i}^2 u_i$$

$\mathbf{V}$ are the first r eigenvectors of $\mathbf{A}^T\mathbf{A}$ with eigenvalues $\text{diag}(\mathbf{S})$
$\mathbf{U}$ are the first r eigenvectors of $\mathbf{A}\mathbf{A}^T$ with eigenvalues $\text{diag}(\mathbf{S})$

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{U} = [u_1, \dots, u_r] \qquad \mathbf{V} = [v_1, \dots, v_r] \qquad \mathbf{S} = \text{diag}(s_1, \dots, s_r)$$

$$\mathbf{A} = \sum_{k=1}^{r} u_k v_k^T s_k \qquad\qquad s_1 \geq s_2 \geq \dots \geq s_r$$

Best rank-1 approximation $\sigma > 0$ and unit vectors $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ minimizes:
$\|\sigma x y^T - \mathbf{A}\|_F^2 =$

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{U} = [u_1, \dots, u_r] \qquad \mathbf{V} = [v_1, \dots, v_r] \qquad \mathbf{S} = \text{diag}(s_1, \dots, s_r)$$

$$\mathbf{A} = \sum_{k=1}^{r} u_k v_k^T s_k \qquad\qquad s_1 \geq s_2 \geq \cdots \geq s_r$$

Best rank-1 approximation $\sigma > 0$ and unit vectors $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ minimizes:

$$\begin{aligned}\|\sigma x y^T - \mathbf{A}\|_F^2 &= \sigma^2 + \text{Tr}(\mathbf{A}^T\mathbf{A}) - 2\sigma x^T \mathbf{A} y \\ &= \sigma^2 + \left(\sum_{k=1}^{r} s_k^2\right) - 2\sigma\left(\sum_{k=1}^{r} x^T u_k v_k^T y\, s_k\right)\end{aligned}$$

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{U} = [u_1, \dots, u_r] \qquad \mathbf{V} = [v_1, \dots, v_r] \qquad \mathbf{S} = \text{diag}(s_1, \dots, s_r)$$

$$\mathbf{A} = \sum_{k=1}^{r} u_k v_k^T s_k \qquad\qquad s_1 \geq s_2 \geq \dots \geq s_r$$

In general: $$\sum_{k=1}^{p} u_i v_i^T s_i = \arg \min_{\mathbf{Z}:\text{rank}(\mathbf{Z})=p} \|\mathbf{Z} - \mathbf{A}\|_F^2$$

Matrix completion

Given historical data on how users rated movies in past:

17,700 movies, 480,189 users, 99,072,112 ratings (Sparsity: 1.2%)

Predict how the same users will rate movies in the future (for $1 million prize)

Alice	1	?	?	4	?	
Bob	?	2	5	?	?	
Carol	?	?	4	5	?	
Dave	5	?	?	?	4	
⋮						...

Matrix completion

n movies, m users, $|S|$ ratings

$$\arg\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j,s) \in \mathcal{S}} ||(UV^T)_{i,j} - s_{i,j}||_2^2$$

How do we solve it? With full information?

Matrix completion

n movies, m users, |*S*| ratings

$$\arg\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j,s) \in \mathcal{S}} ||(UV^T)_{i,j} - s_{i,j}||_2^2$$

Matrix completion

n movies, m users, $|S|$ ratings

$$\arg\min_{U\in\mathbb{R}^{m\times d},V\in\mathbb{R}^{n\times d}} \sum_{(i,j,s)\in\mathcal{S}} ||(UV^T)_{i,j} - s_{i,j}||_2^2$$

Practical techniques to solve:

- Alternating minimization (Fix U, minimize V. Then fix V and minimize U)
- Stochastic gradient descent on U, V
- Nuclear norm regularization (convex)

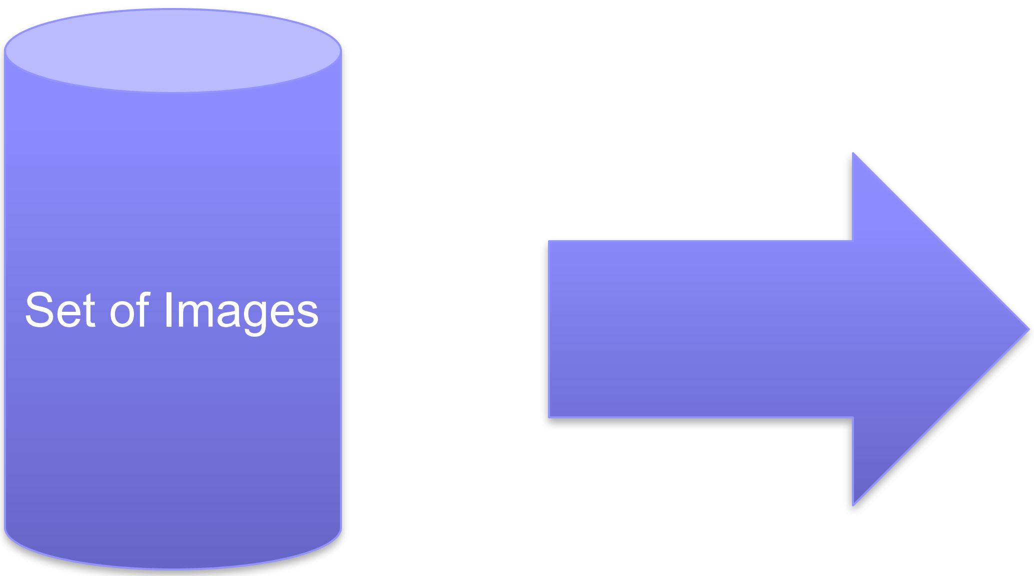
Clustering
K-means

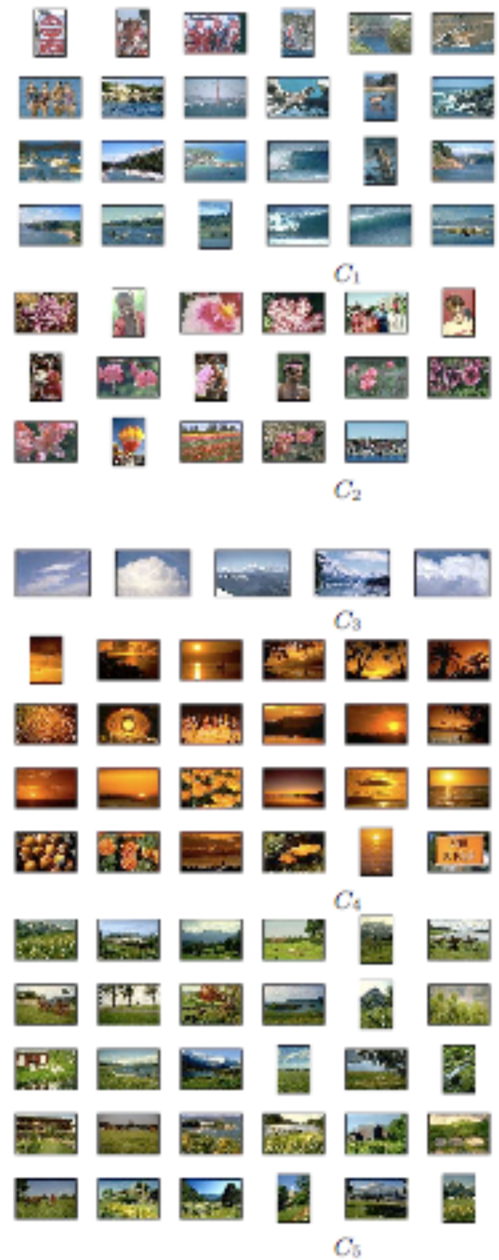
Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 15, 2016

©Kevin Jamieson 2018

Clustering images

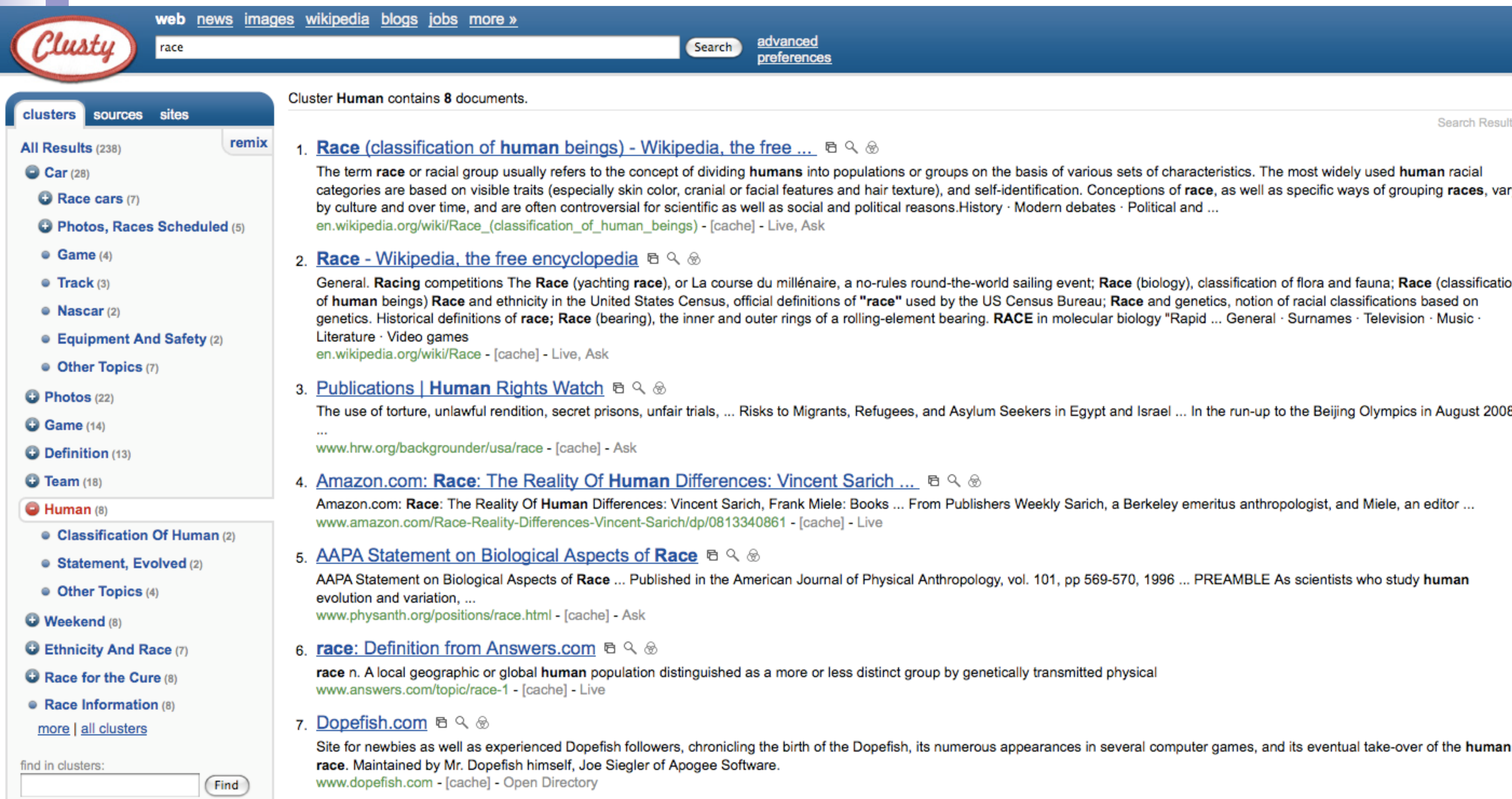




[Goldberger et al.]

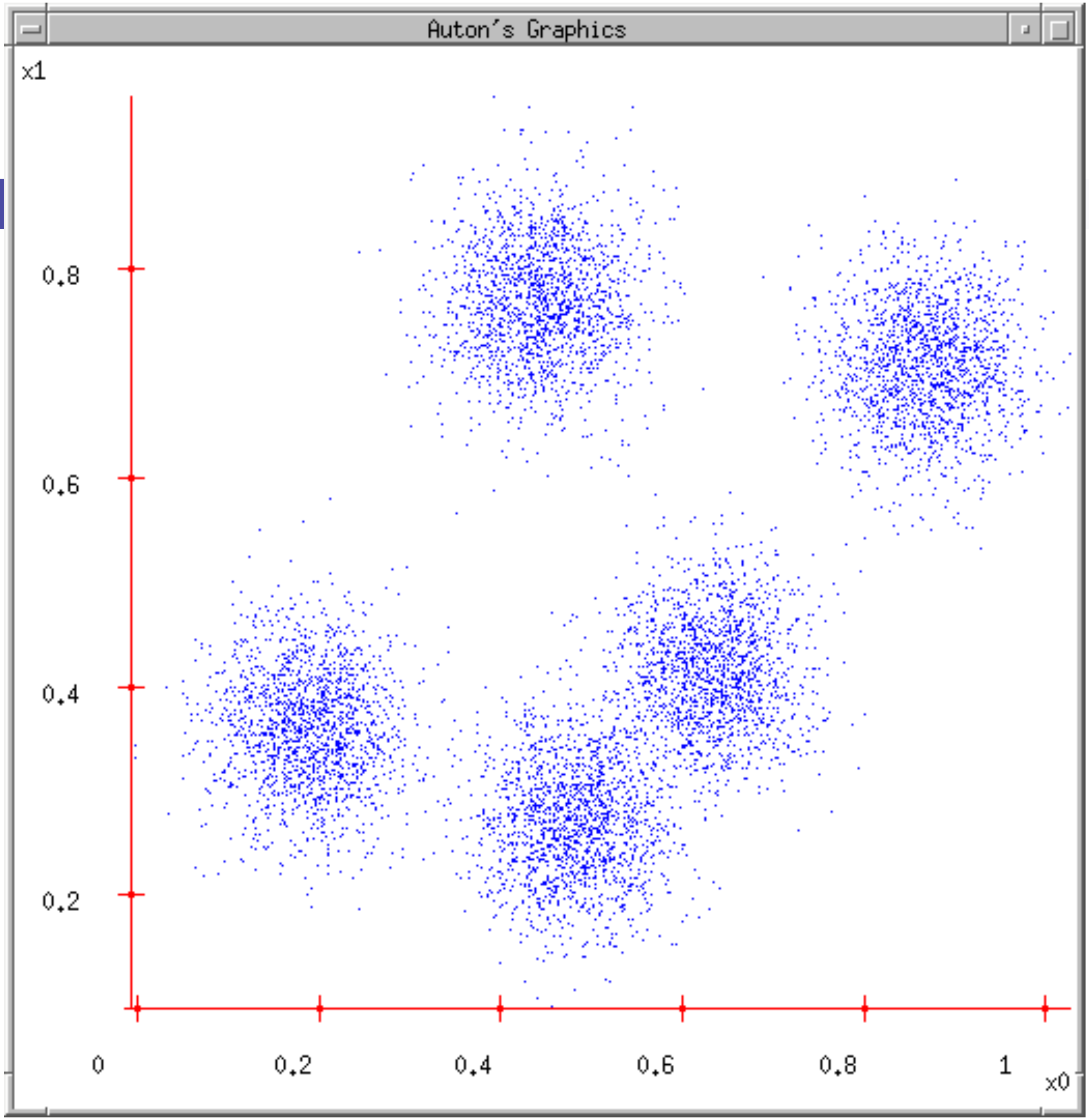
©Kevin Jamieson 2018

Clustering web search results



©Kevin Jamieson 2018

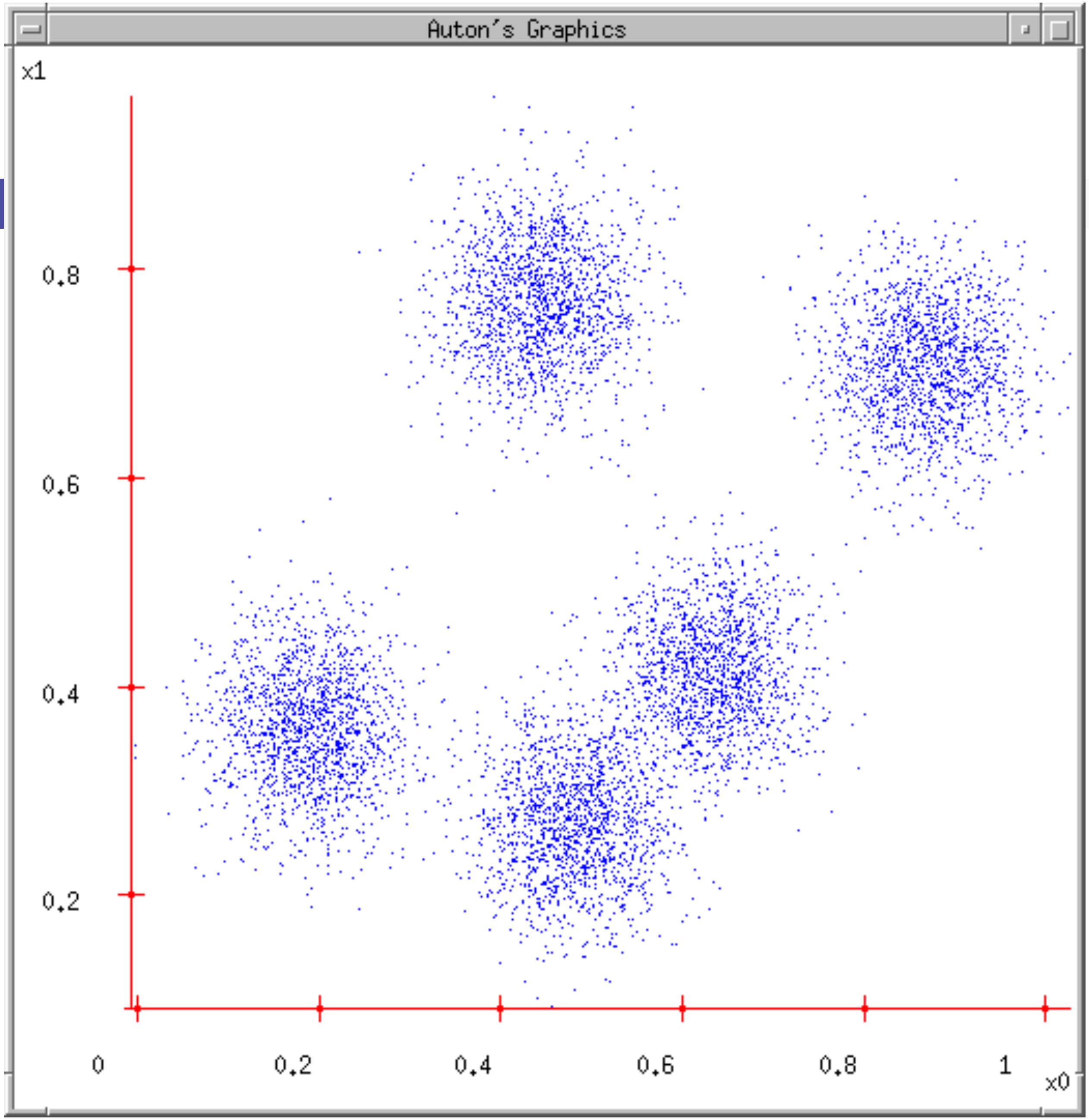
Some Data



©Kevin Jamieson 2018

K-means

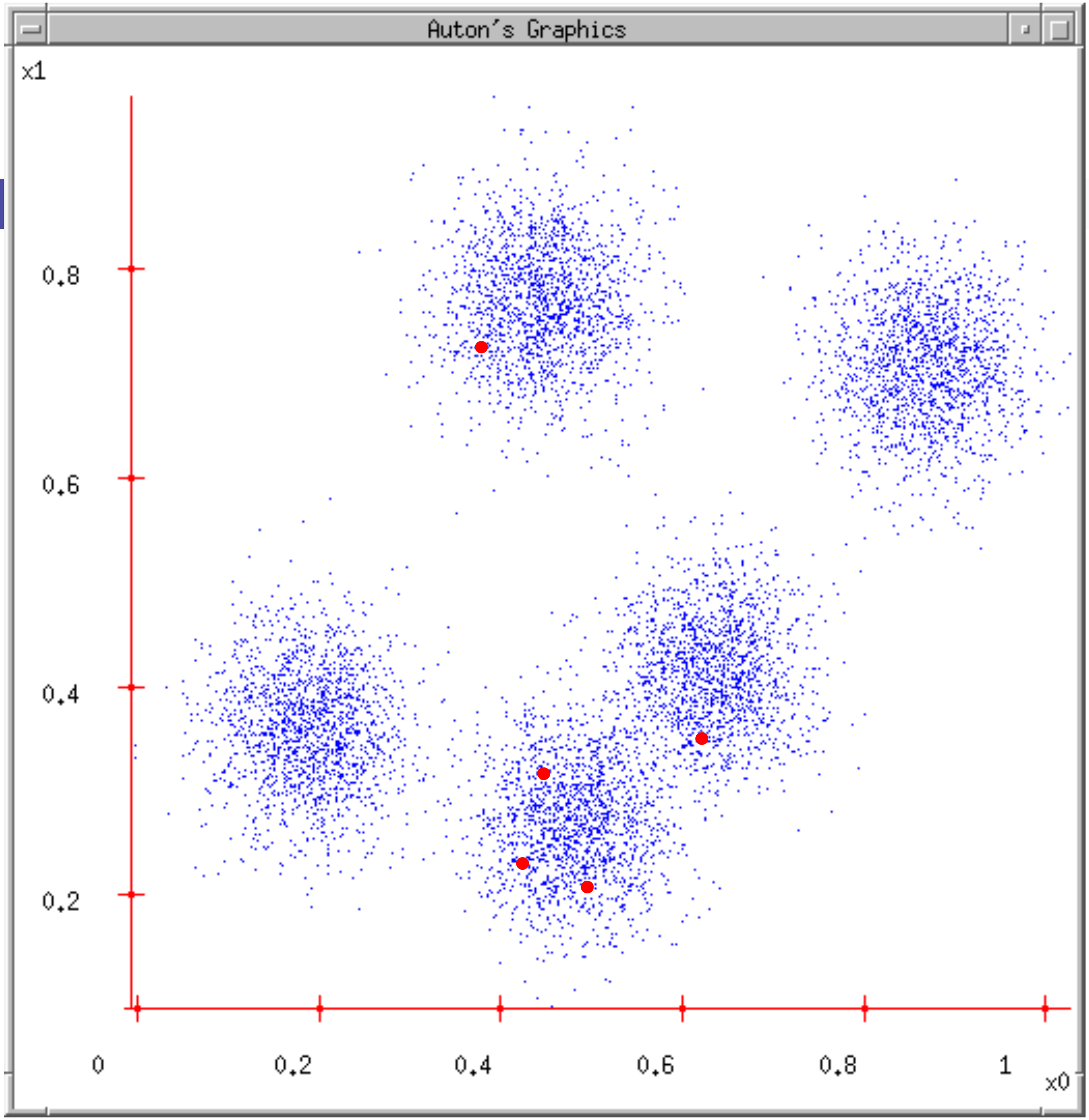
1. Ask user how many clusters they'd like. *(e.g. k=5)*



©Kevin Jamieson 2018

K-means

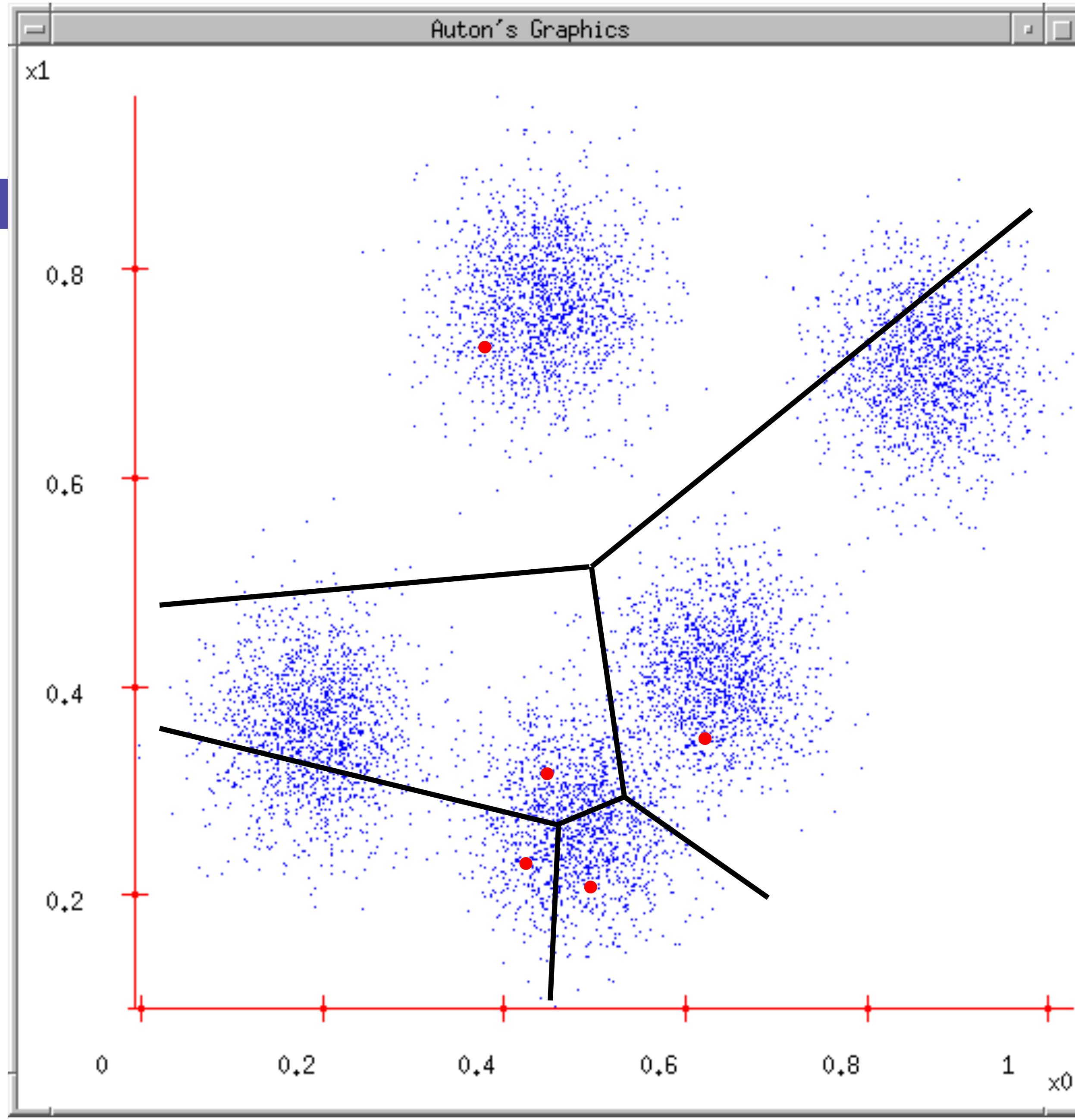
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



©Kevin Jamieson 2018

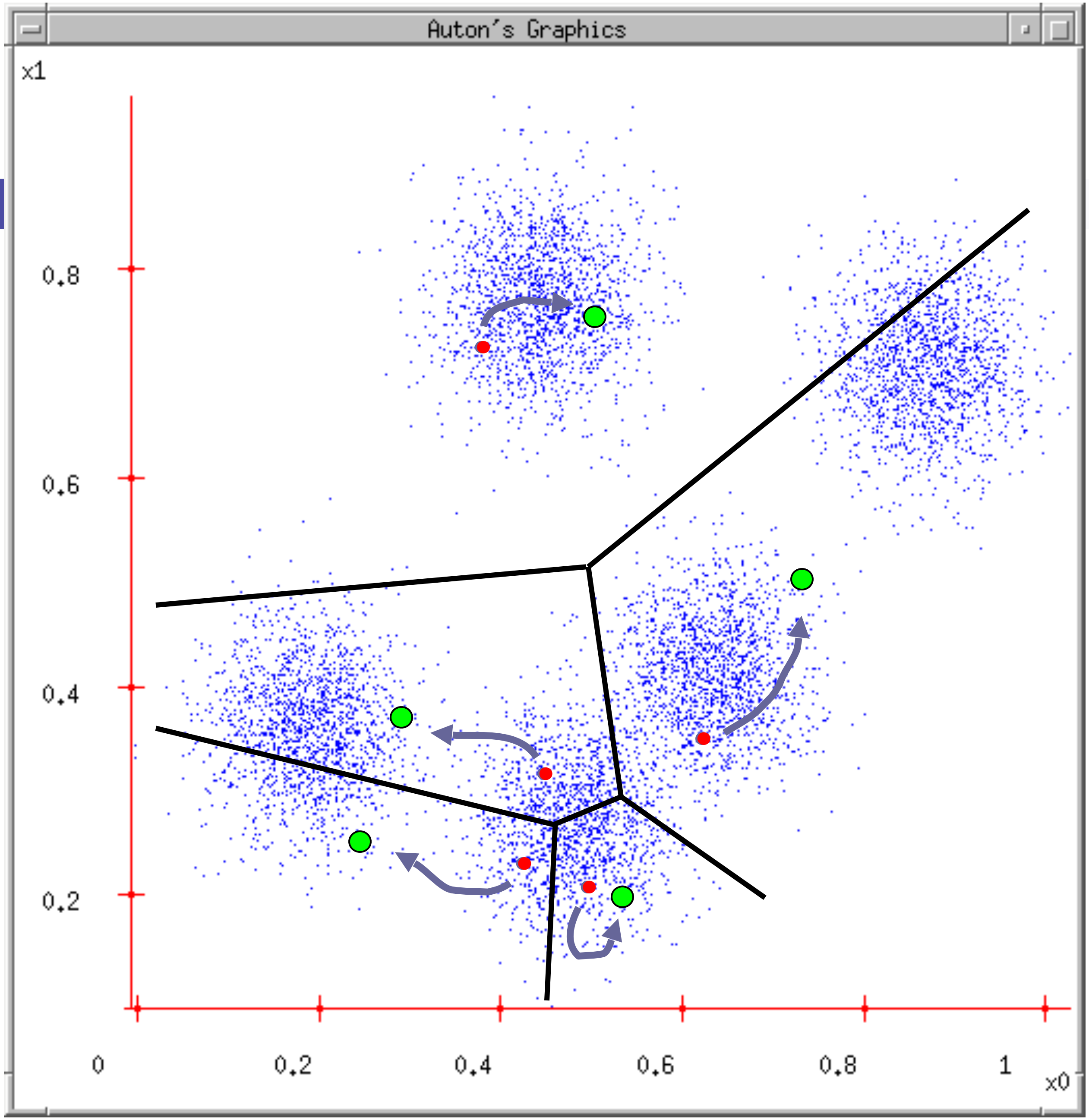
K-means

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

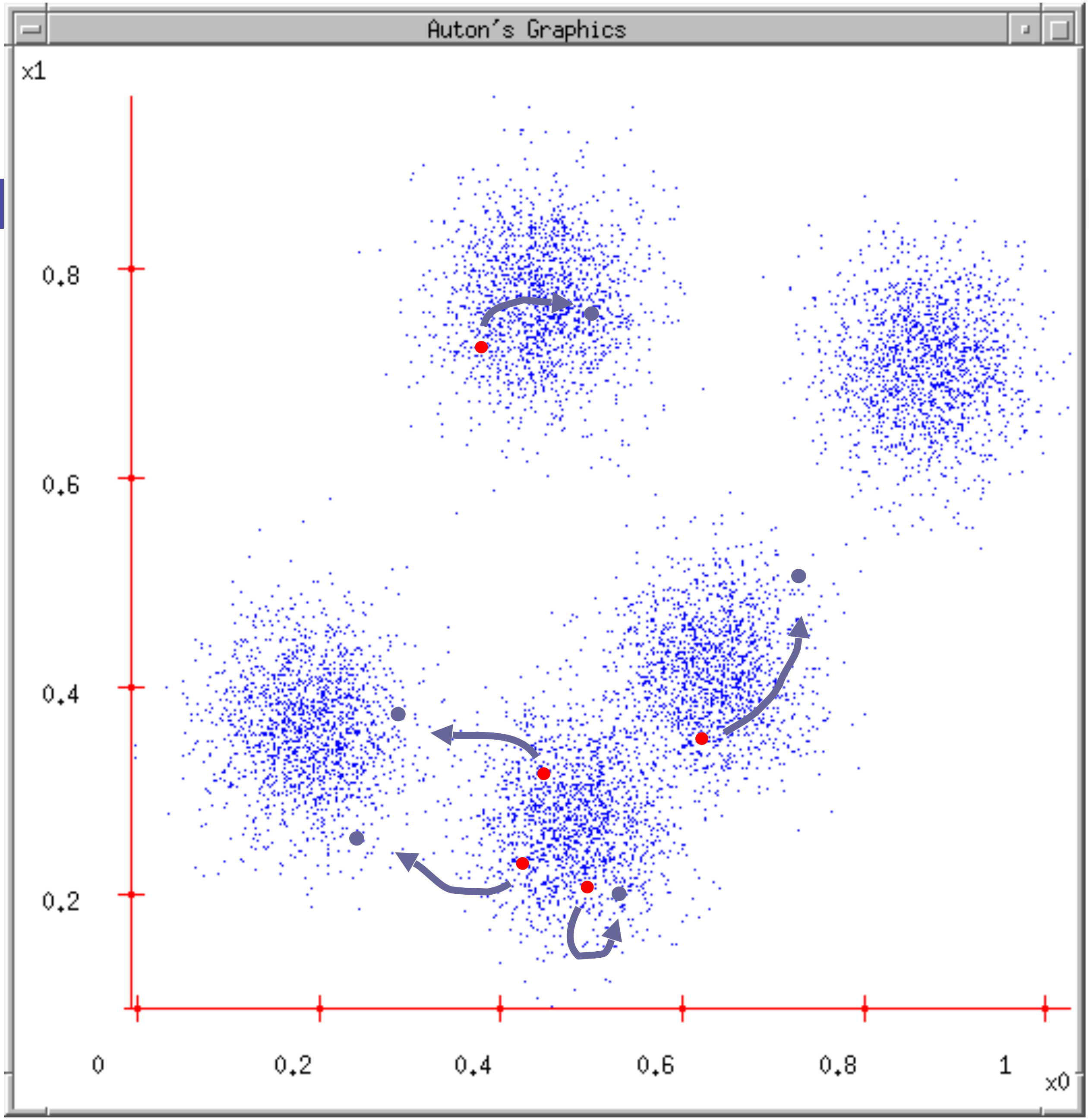
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



©Kevin Jamieson 2018

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

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$
- **Classify**: Assign each point $j \in \{1, \ldots N\}$ to nearest center:
  - $C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$
- **Recenter**: μ_i becomes centroid of its point:
  - $\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

©Kevin Jamieson 2018

Does K-means converge??? Part 1

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix μ, optimize C

©Kevin Jamieson 2018

Does K-means converge??? Part 2

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix C, optimize μ

©Kevin Jamieson 2018

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

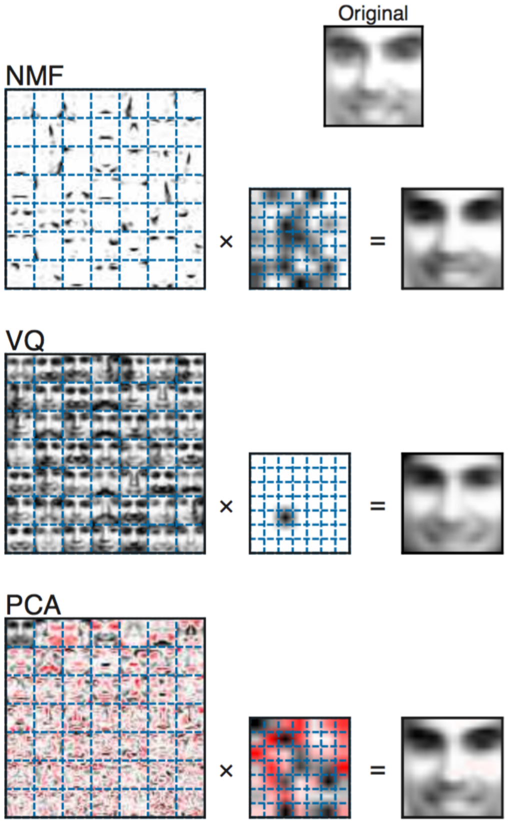
FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at 8 bits per pixel. The center image is the result of* 2×2 *block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at 8 bits per pixel. The center image is the result of* 2×2 *block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*



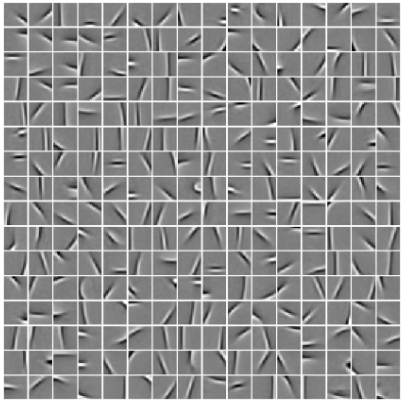
Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at 8 bits per pixel. The center image is the result of* 2×2 *block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Typical output of k-means on patches

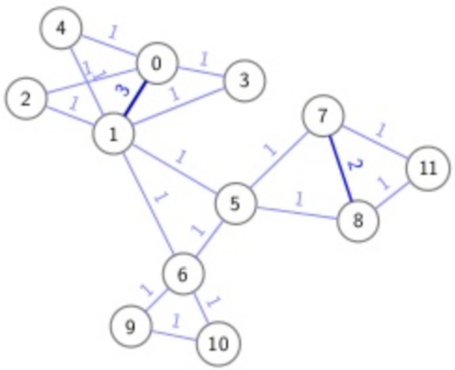


Similar reduced representation can be used as a feature vector

Coates, Ng, *Learning Feature Representations with K-means,* 2012

Spectral Clustering

Adjacency matrix: $\mathbf{W}$



$$\mathbf{W}_{i,j} = \text{weight of edge } (i,j)$$

$$\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{W}_{i,j} \qquad \mathbf{L} = \mathbf{D} - \mathbf{W}$$

Given feature vectors, could construct:

- k-nearest neighbor graph with weights in {0,1}
- weighted graph with arbitrary *similarities* $\mathbf{W}_{i,j} = e^{-\gamma||x_i - x_j||^2}$

Let $f \in \mathbb{R}^n$ be a function over the nodes

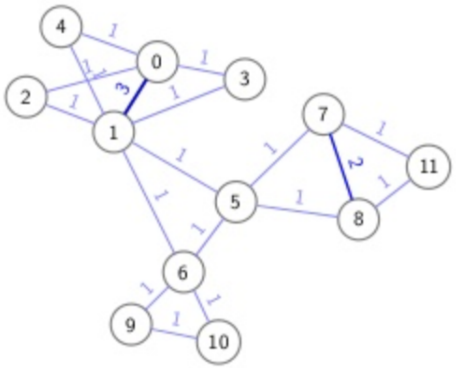
$$\begin{aligned} \mathbf{f}^T\mathbf{L}\mathbf{f} &= \sum_{i=1}^{N} g_i f_i^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} f_i f_{i'} w_{ii'} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} w_{ii'}(f_i - f_{i'})^2. \end{aligned}$$

Spectral Clustering

Adjacency matrix: $\mathbf{W}$

$$\mathbf{W}_{i,j} = \text{weight of edge } (i,j)$$

$$\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{W}_{i,j} \qquad \mathbf{L} = \mathbf{D} - \mathbf{W}$$



Given feature vectors, could construct:

- k-nearest neighbor graph with weights in {0,1}
- weighted graph with arbitrary *similarities* $\mathbf{W}_{i,j} = e^{-\gamma||x_i - x_j||^2}$

Let $f \in \mathbb{R}^n$ be a function over the nodes

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{i=1}^{N} g_i f_i^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} f_i f_{i'} w_{ii'}$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} w_{ii'}(f_i - f_{i'})^2.$$

Spectral Clustering

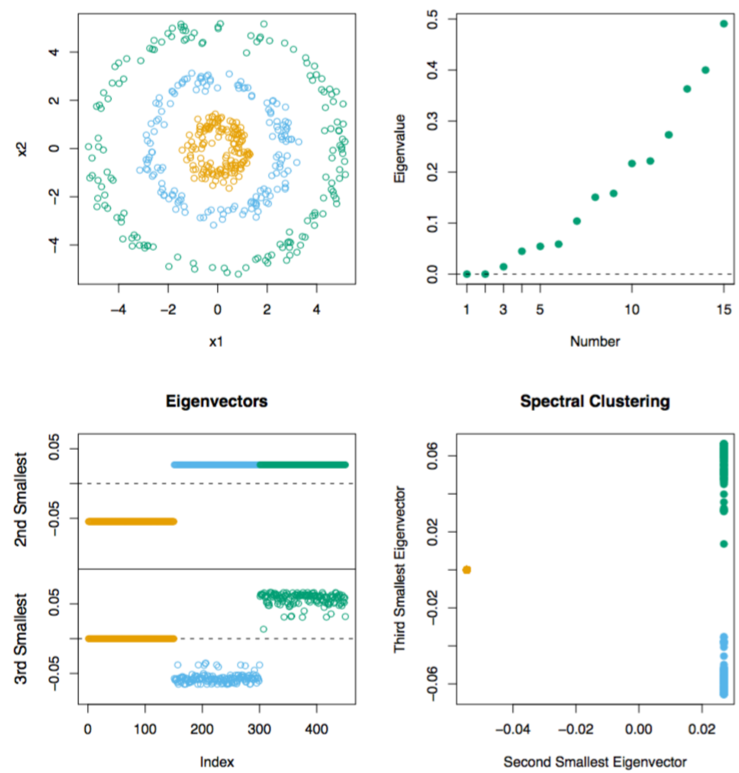
Adjacency matrix: $\mathbf{W}$

$$\mathbf{W}_{i,j} = \text{weight of edge } (i,j)$$

$$\mathbf{D}_{i,i} = \sum_{j=1}^{n} \mathbf{W}_{i,j} \qquad \mathbf{L} = \mathbf{D} - \mathbf{W}$$

Given feature vectors, could construct:

- (k=10)-nearest neighbor graph with weights in {0,1}



Mixtures of Gaussians

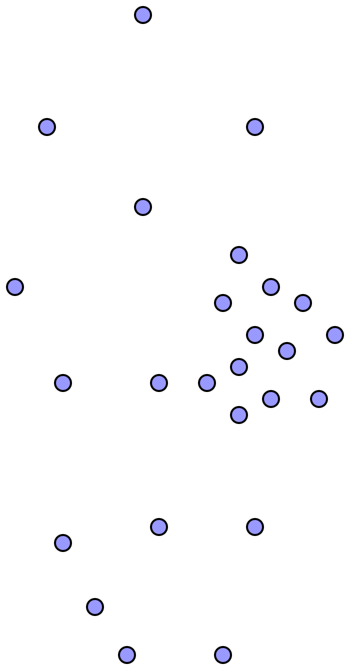
Machine Learning – CSE546

Kevin Jamieson
University of Washington

November 15, 2016

©Kevin Jamieson 2017

(One) bad case for k-means

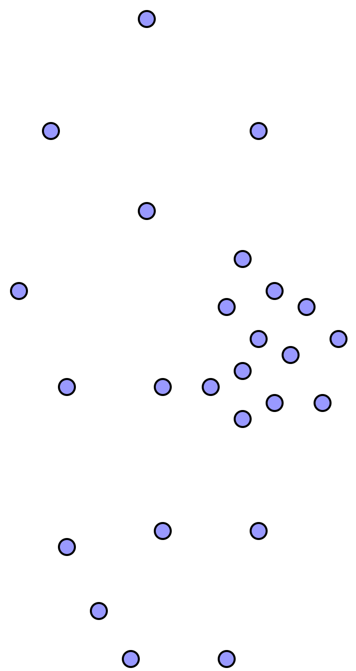


- Clusters may overlap
- Some clusters may be "wider" than others

©Kevin Jamieson 2017

(One) bad case for k-means

- Clusters may overlap
- Some clusters may be "wider" than others

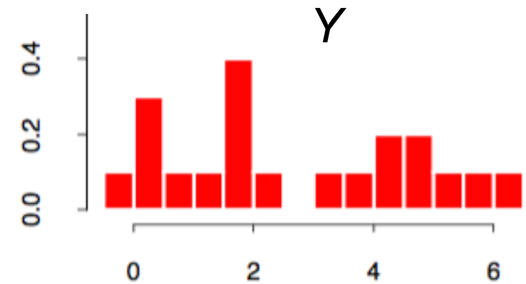


©Kevin Jamieson 2017

Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0,1\}$ with $\Pr(\Delta = 1) = \pi$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$\ell(\theta; \mathbf{Z}) = \sum_{i=1}^n \log[(1-\pi)\phi_{\theta_1}(y_i) + \pi\phi_{\theta_2}(y_i)]$$

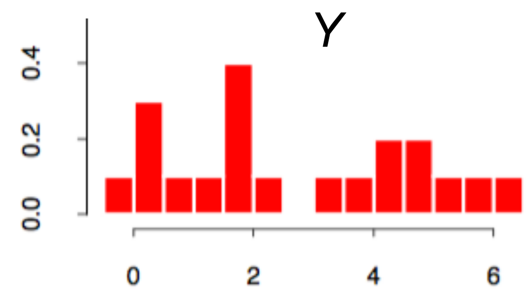
©Kevin Jamieson 2017

Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1 - \Delta) \cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0, 1\}$ with $\Pr(\Delta = 1) = \pi$

$$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

$\boldsymbol{\Delta} = \{\Delta_i\}_{i=1}^n$ is unobserved data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$\ell(\theta; y_i, \Delta_i = 0) =$$

$$\ell(\theta; y_i, \Delta_i = 1) =$$

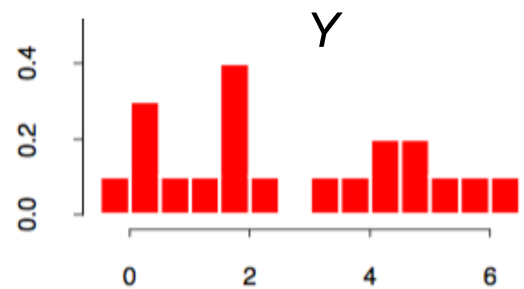
©Kevin Jamieson 2017

Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0,1\}$ with $\Pr(\Delta = 1) = \pi$

$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

$\mathbf{\Delta} = \{\Delta_i\}_{i=1}^n$ is unobserved data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$\ell(\theta; \mathbf{Z}, \mathbf{\Delta}) = \sum_{i=1}^n (1-\Delta_i)\log[(1-\pi)\phi_{\theta_1}(y_i)] + \Delta_i \log(\pi\phi_{\theta_2}(y_i)]$$

If we knew Δ, how would we choose θ?

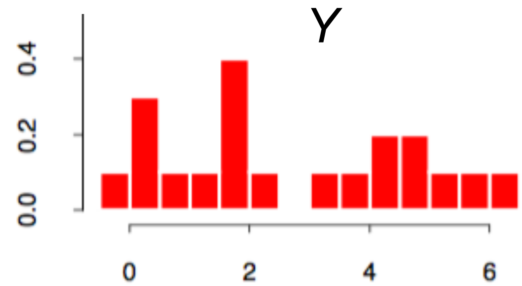
©Kevin Jamieson 2017

Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1-\Delta) \cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0, 1\}$ with $\Pr(\Delta = 1) = \pi$

$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

$\mathbf{\Delta} = \{\Delta_i\}_{i=1}^n$ is unobserved data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$
\ell(\theta; \mathbf{Z}, \mathbf{\Delta}) = \sum_{i=1}^n (1-\Delta_i) \log[(1-\pi)\phi_{\theta_1}(y_i)] + \Delta_i \log(\pi \phi_{\theta_2}(y_i)]
$$

If we knew θ, how would we choose $\mathbf{\Delta}$?

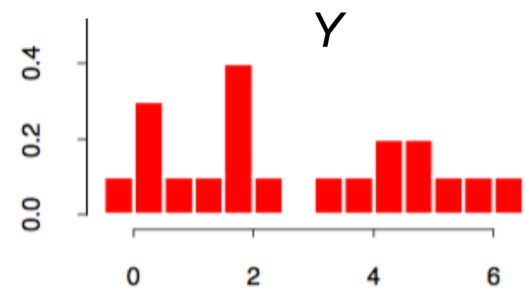
©Kevin Jamieson 2017

Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0,1\}$ with $\Pr(\Delta = 1) = \pi$

$$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

$\mathbf{\Delta} = \{\Delta_i\}_{i=1}^n$ is unobserved data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$\ell(\theta; \mathbf{Z}, \mathbf{\Delta}) = \sum_{i=1}^n (1-\Delta_i)\log[(1-\pi)\phi_{\theta_1}(y_i)] + \Delta_i \log(\pi \phi_{\theta_2}(y_i)]$$

$$\gamma_i(\theta) = \mathbb{E}[\Delta_i | \theta, \mathbf{Z}] =$$

©Kevin Jamieson 2017

Mixture models

Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \ldots, N. \tag{8.42}$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

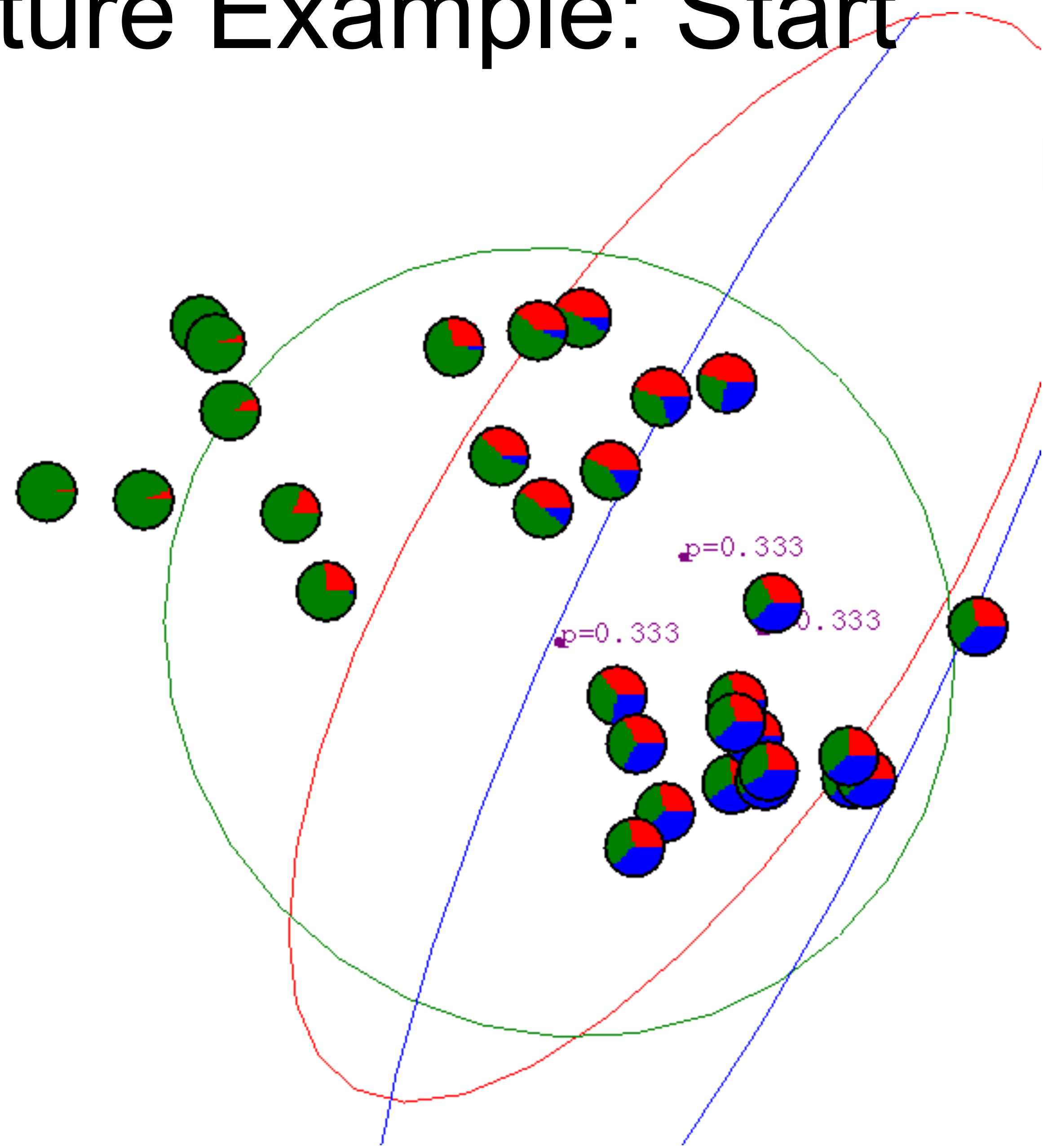
$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

   and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.

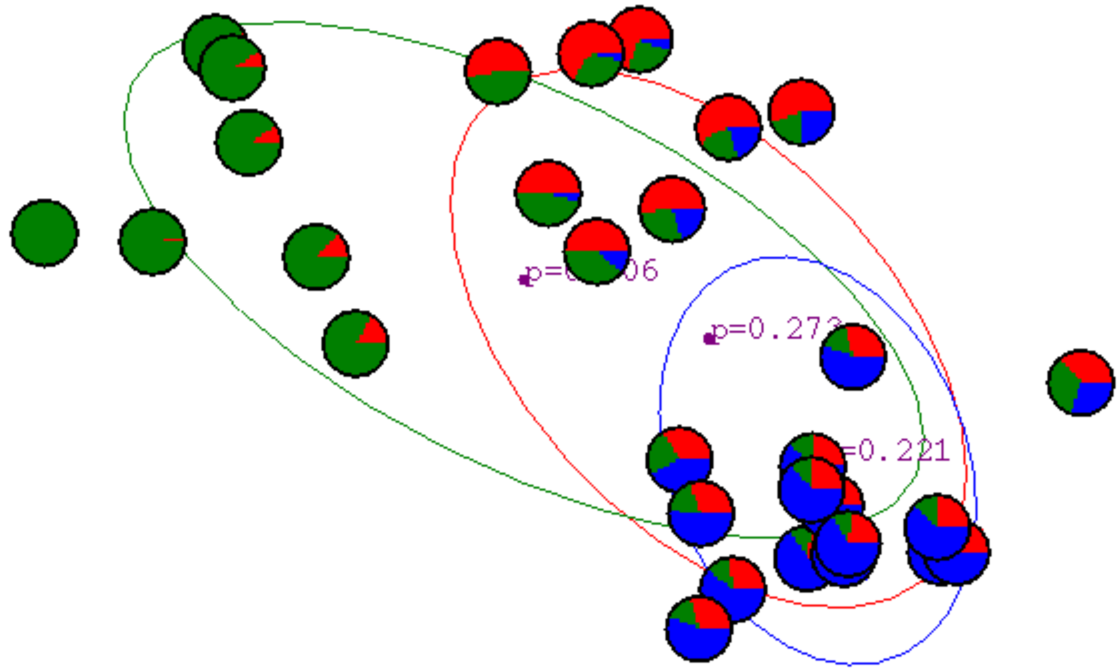
©Kevin Jamieson 2017

Gaussian Mixture Example: Start



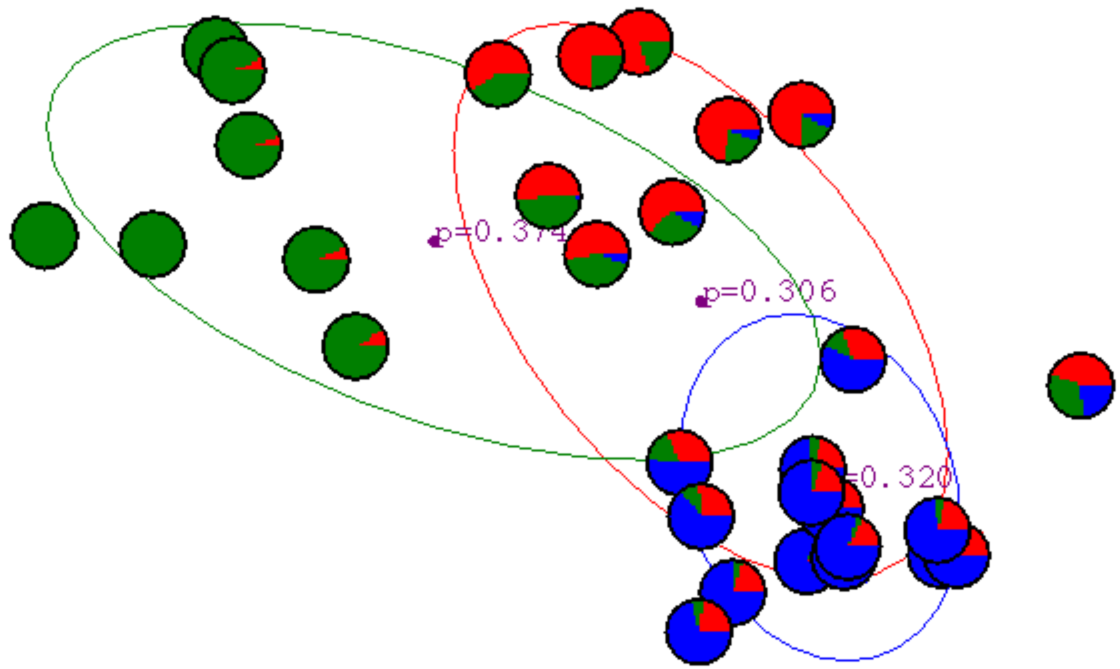
©Kevin Jamieson 2017

After first iteration



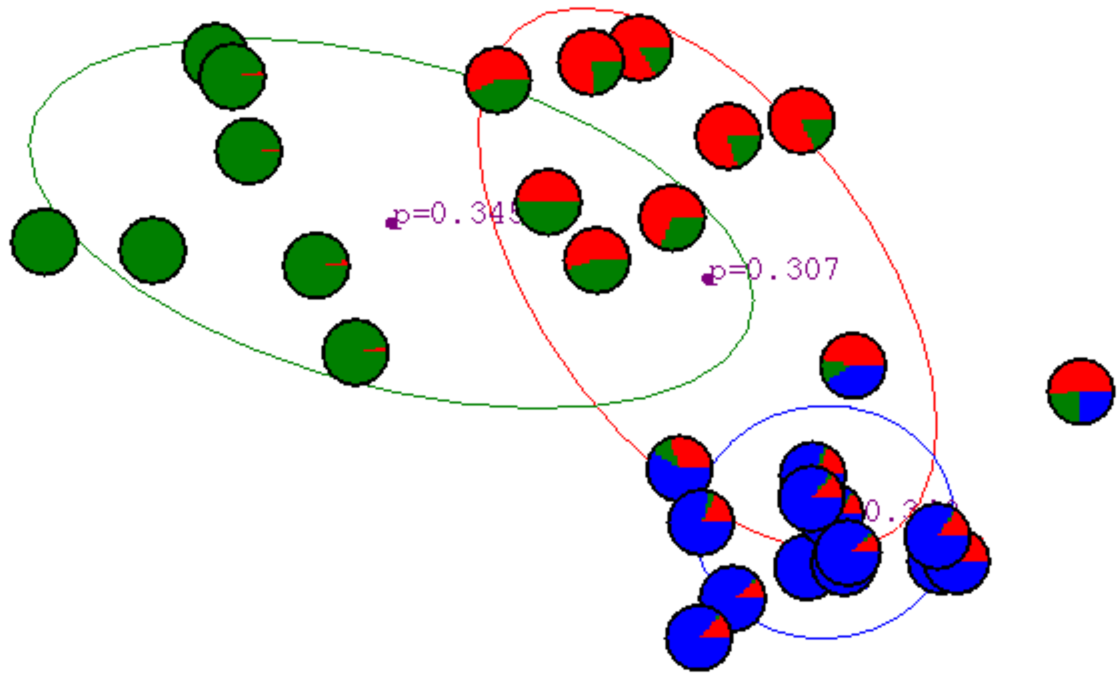
©Kevin Jamieson 2017

After 2nd iteration

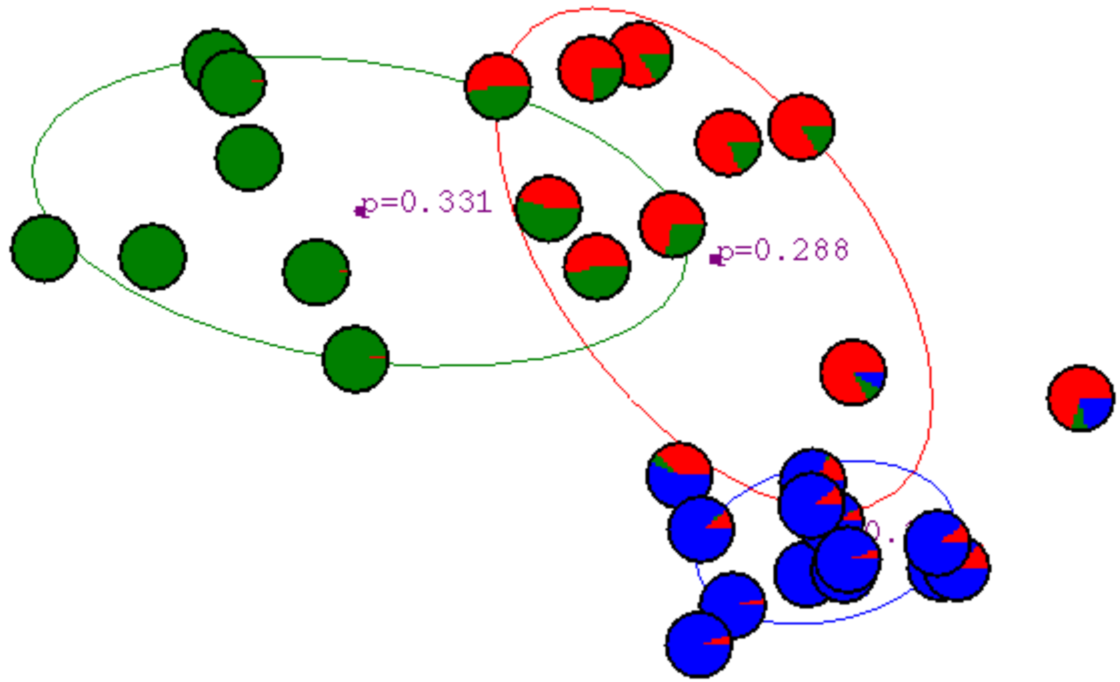


©Kevin Jamieson 2017

After 3rd iteration

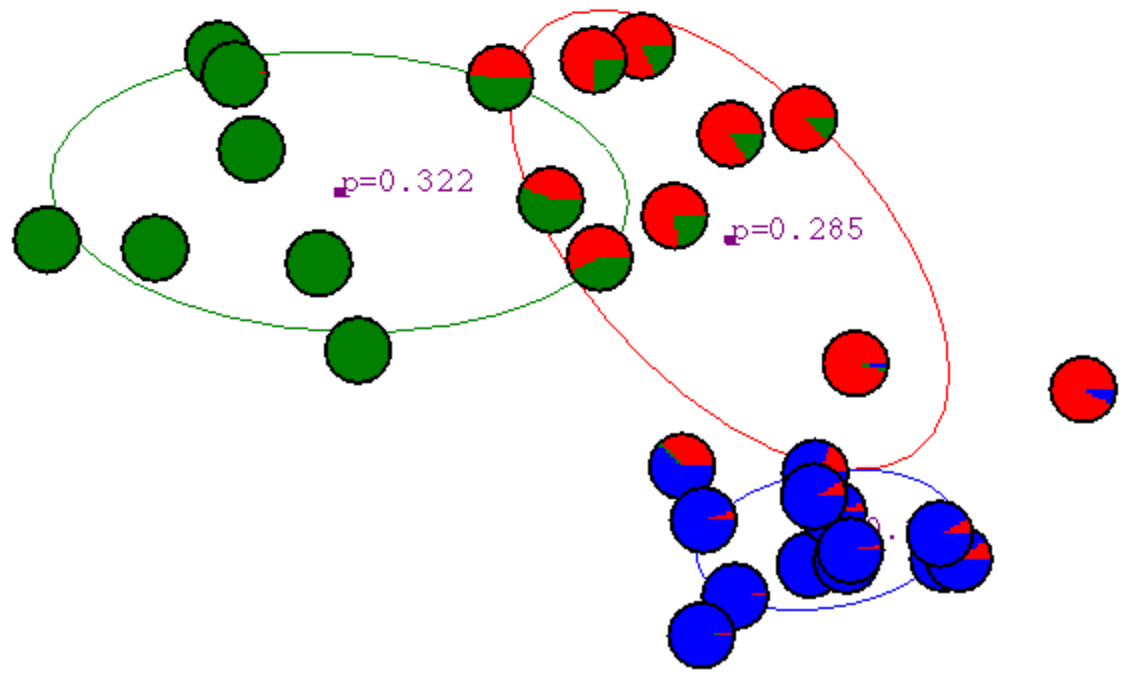


After 4th iteration



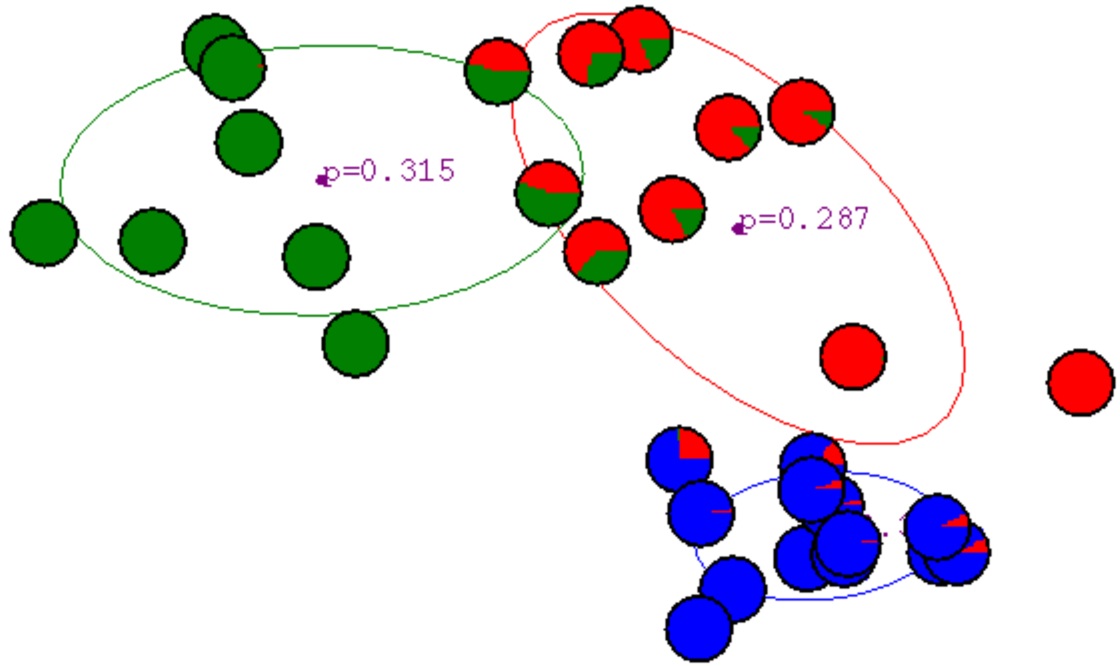
©Kevin Jamieson 2017

After 5th iteration



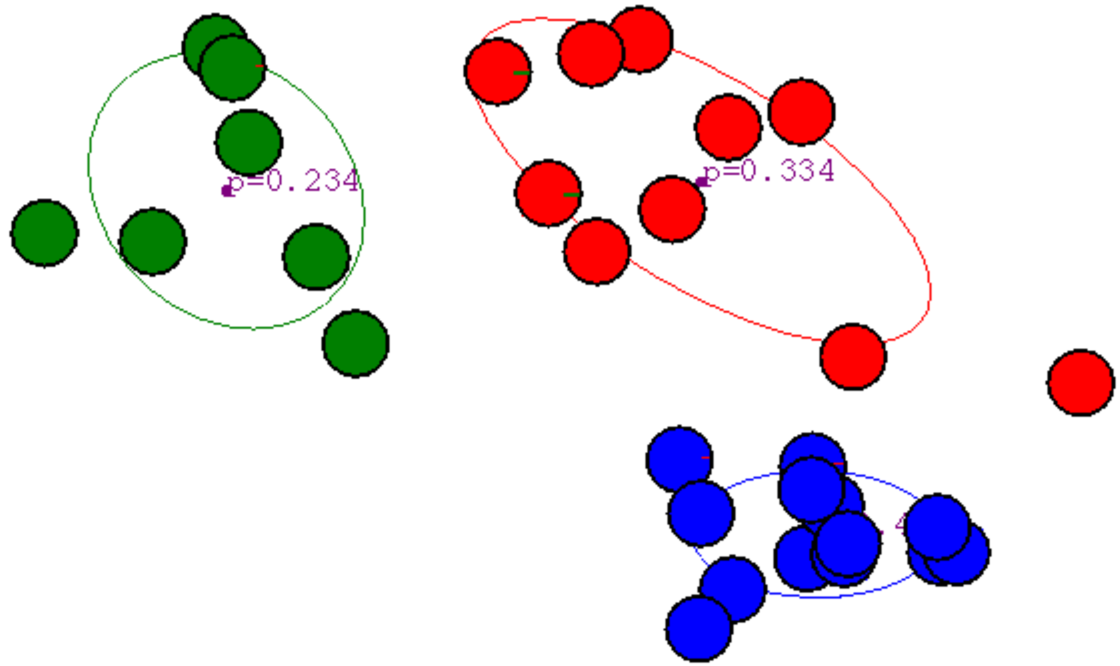
©Kevin Jamieson 2017

After 6th iteration



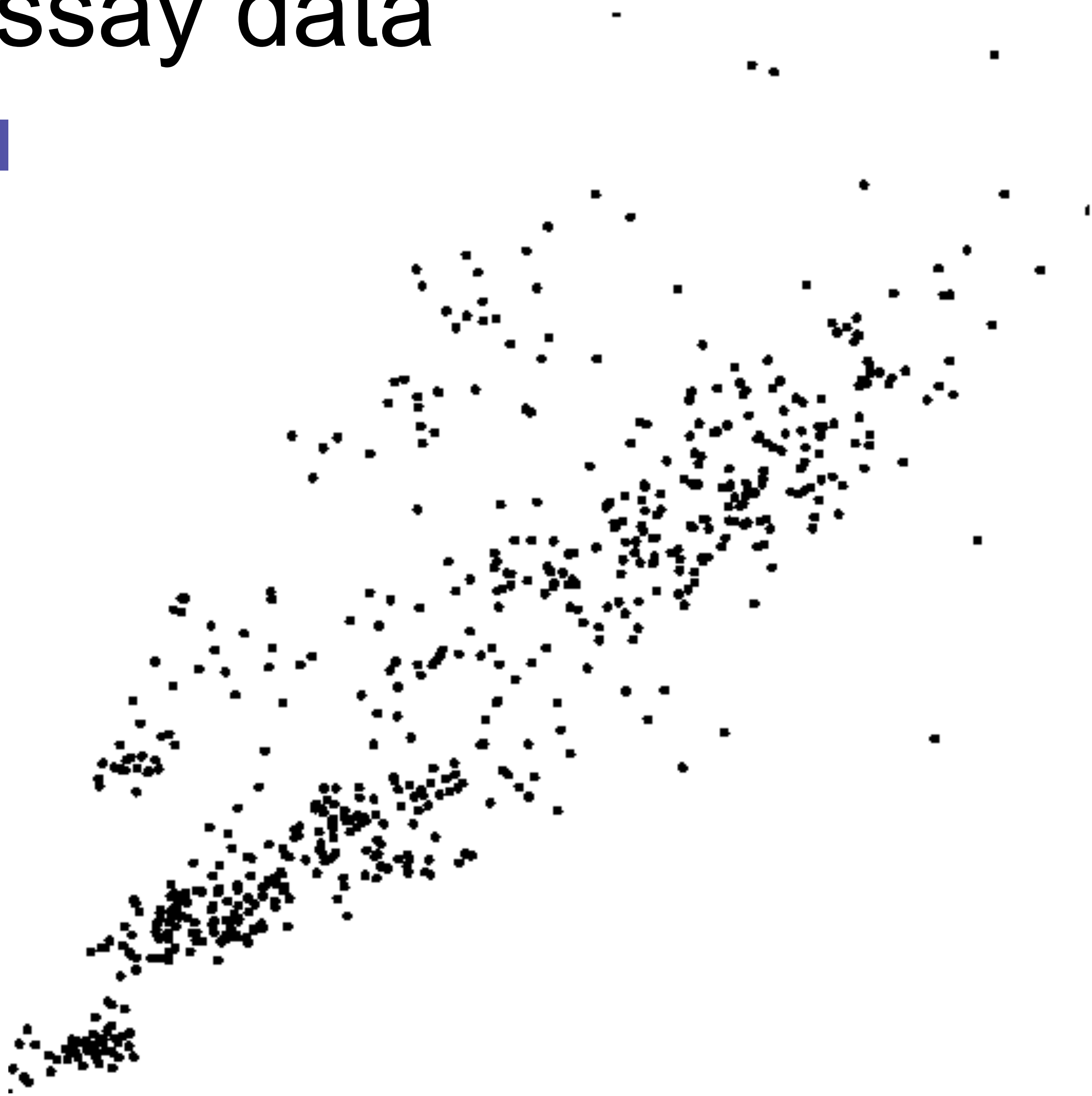
©Kevin Jamieson 2017

After 20th iteration



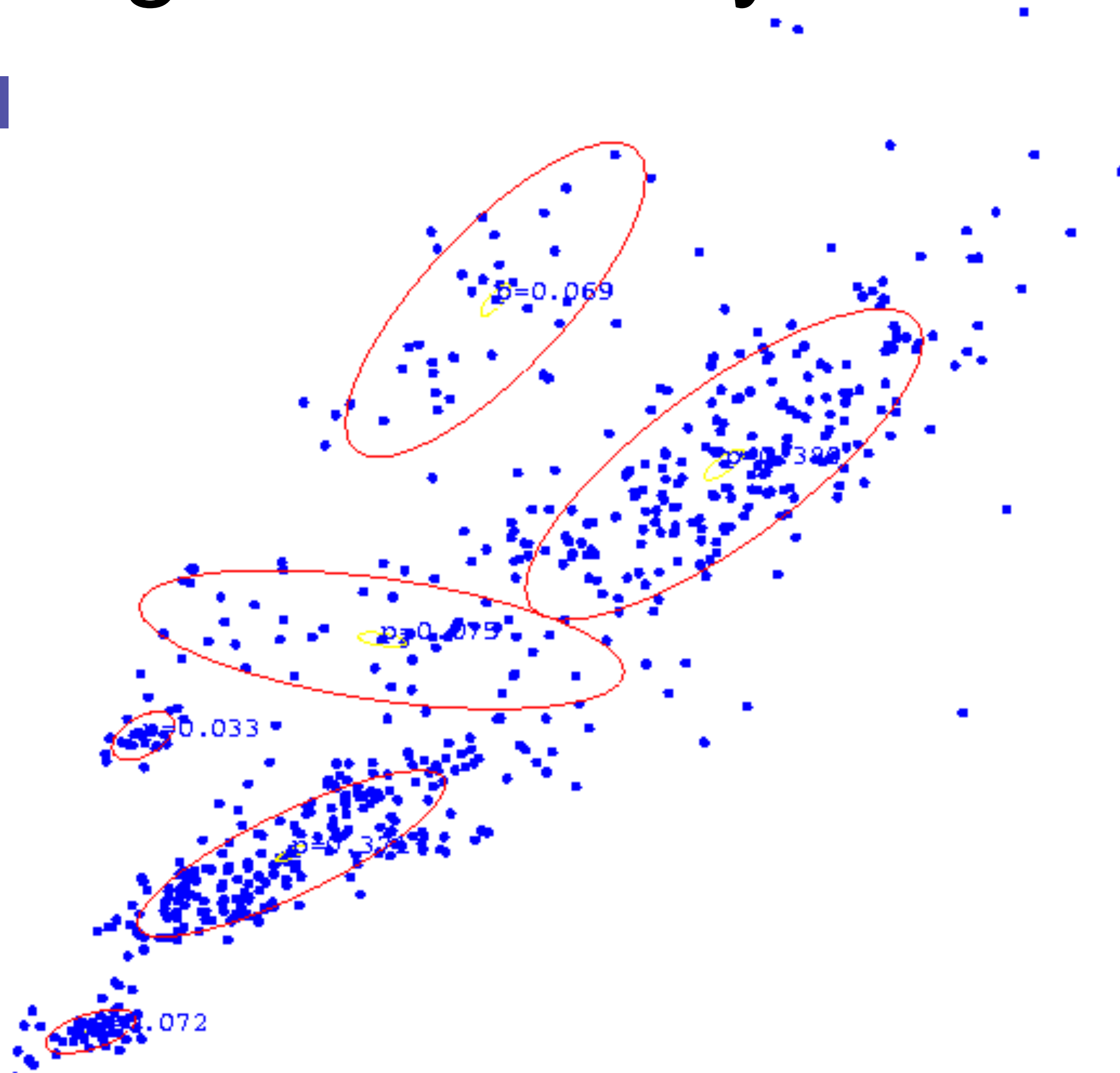
©Kevin Jamieson 2017

Some Bio Assay data



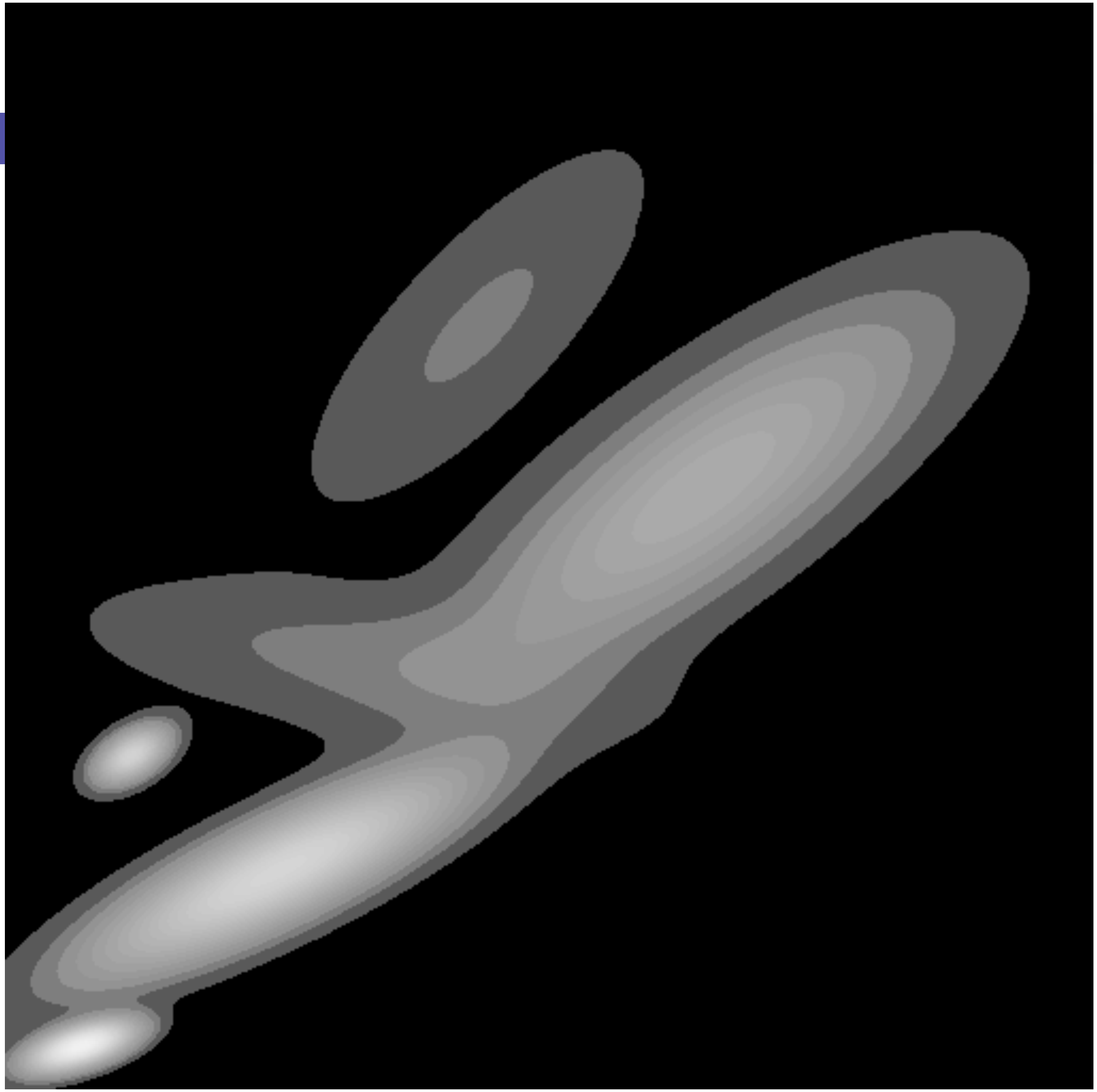
©Kevin Jamieson 2017

GMM clustering of the assay data



©Kevin Jamieson 2017

Resulting Density Estimator



©Kevin Jamieson 2017