In question 2, you solve for $\alpha \in \mathbb{R}^n$: $\hat{f}(x) = \sum_{i=1}^n \alpha_i K(x, x_i)$ ← This is a function that can be evaluated on an arbitrarily fine grid

# Announcements

$\arg\min_a \mathbb{E}[(z-a)^2] = \mathbb{E}[z]$ $\qquad$ $\mathbb{E}[(z-\mathbb{E}[z])^2] \geq 0$

- Homework 3 due tonight! Milestones graded.
- HW 4 will be posted tonight. Start early.

$\mathbb{E}[Y|X=x] = \arg\min_f \mathbb{E}[(Y-f(X))^2]$

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \underset{d\times d}{\Sigma_{XX}} & \underset{d\times 1}{\Sigma_{XY}} \\ \Sigma_{YX} & \underset{1\times 1}{\Sigma_{YY}} \end{bmatrix} \right)$$

If $(X,Y)$ Gaussian it turns out that $\mathbb{E}[Y|X=x] = w^T x + b$ for $w \in \mathbb{R}^d$, $b \in \mathbb{R}^1$

Find $w, b$.

$$\mathbb{E}[Y|X=x] = \mu_Y + \Sigma_{YX}^T \Sigma_{XX}^{-1}(x - \mu_X)$$

$$\min_{w,b} \mathbb{E}[(Y - w^T X - b)^2] = \min_w \underbrace{\mathbb{E}[((Y-\mu_Y) - w^T(X-\mu_X))^2]}_{\ell(w)}$$

$$\nabla_w \ell(w) = \nabla_w \mathbb{E}[-2w^T(X-\mu_X)(Y-\mu_Y) + w^T(X-\mu_X)(X-\mu_X)w]$$

$$= \mathbb{E}[-2(X-\mu_X)(Y-\mu_Y) + 2(X-\mu_X)(X-\mu_X)w]$$

$$= -2\Sigma_{YX} + 2\Sigma_{XX} w = 0 \qquad w = \Sigma_{XX}^{-1}\Sigma_{YX}$$

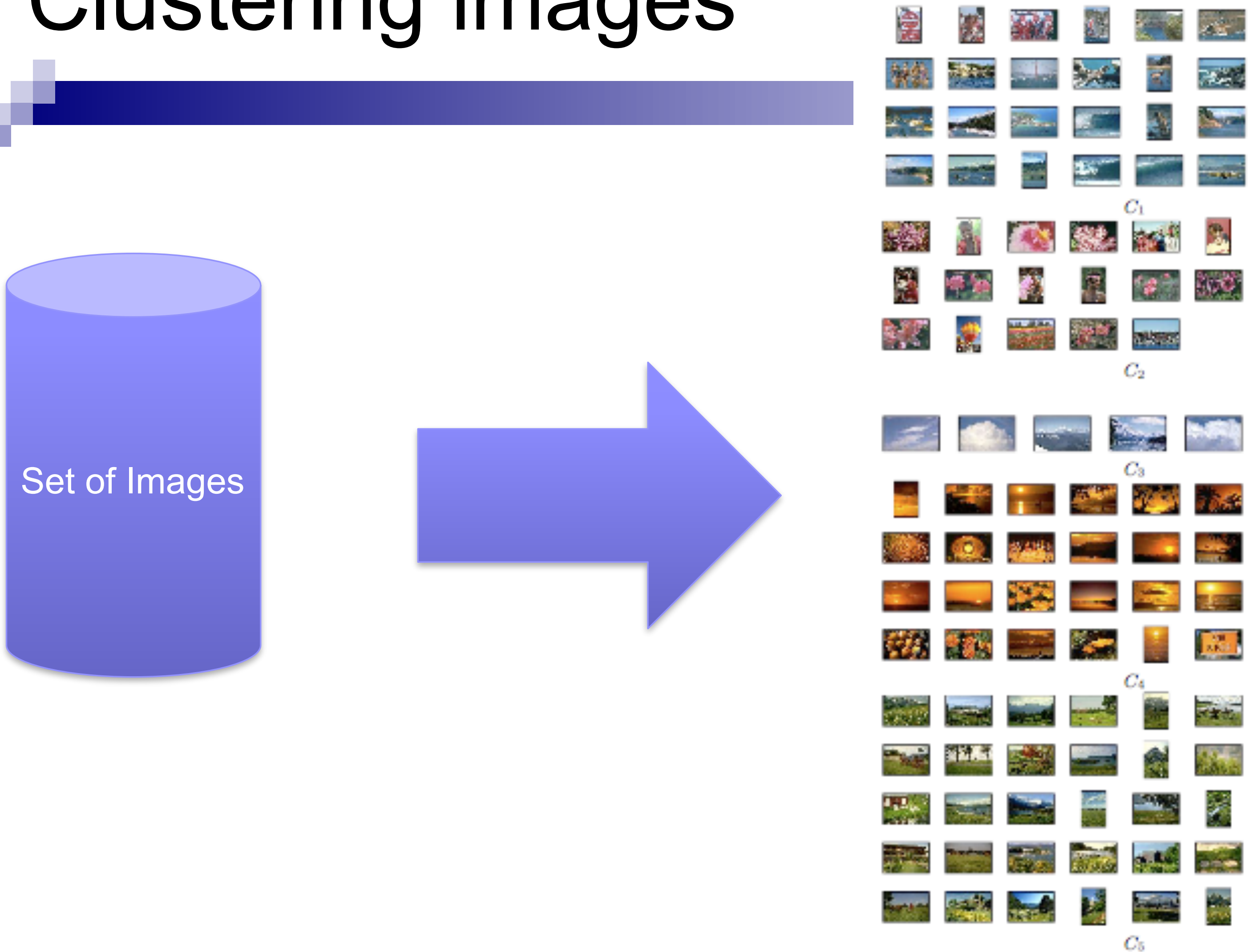
# Clustering

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 21, 2016


©Kevin Jamieson 2017

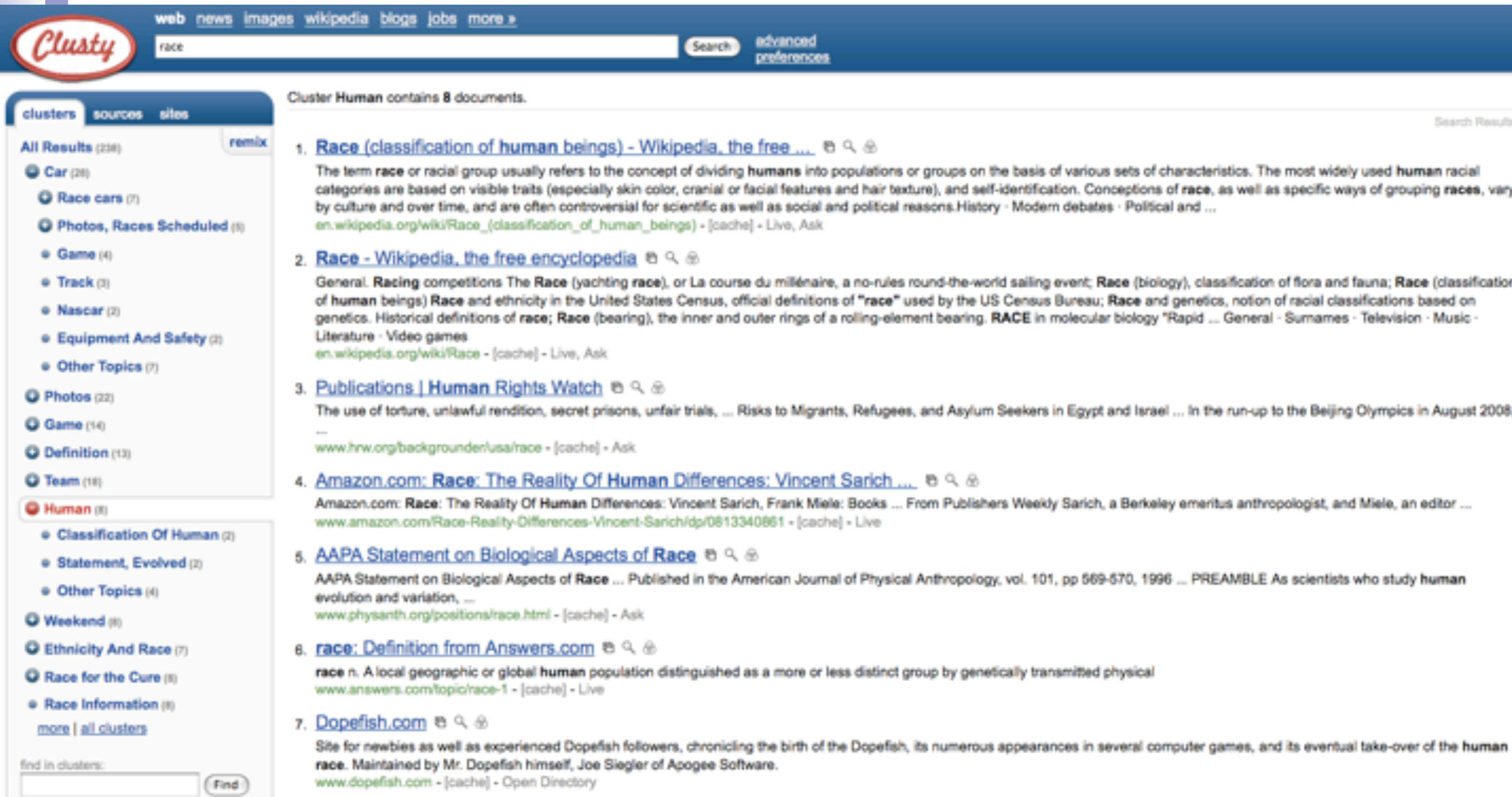
# Clustering images




[Goldberger et al.]


©Kevin Jamieson 2017

# Clustering web search results





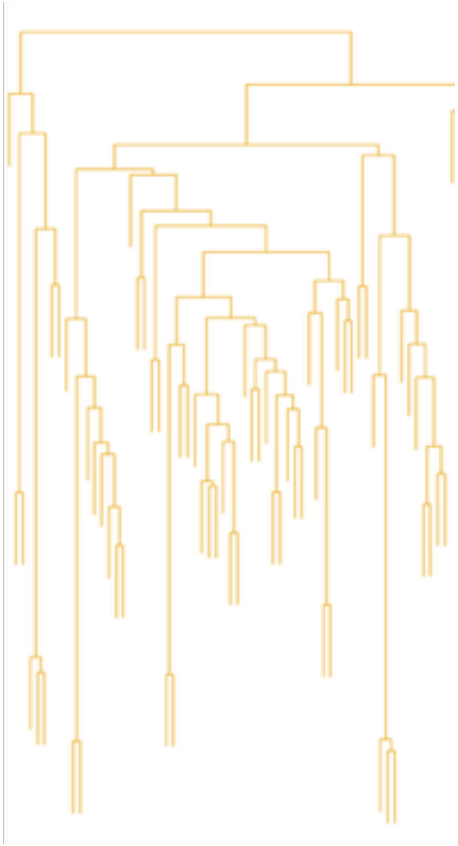

©Kevin Jamieson 2017

# Hierarchical Clustering

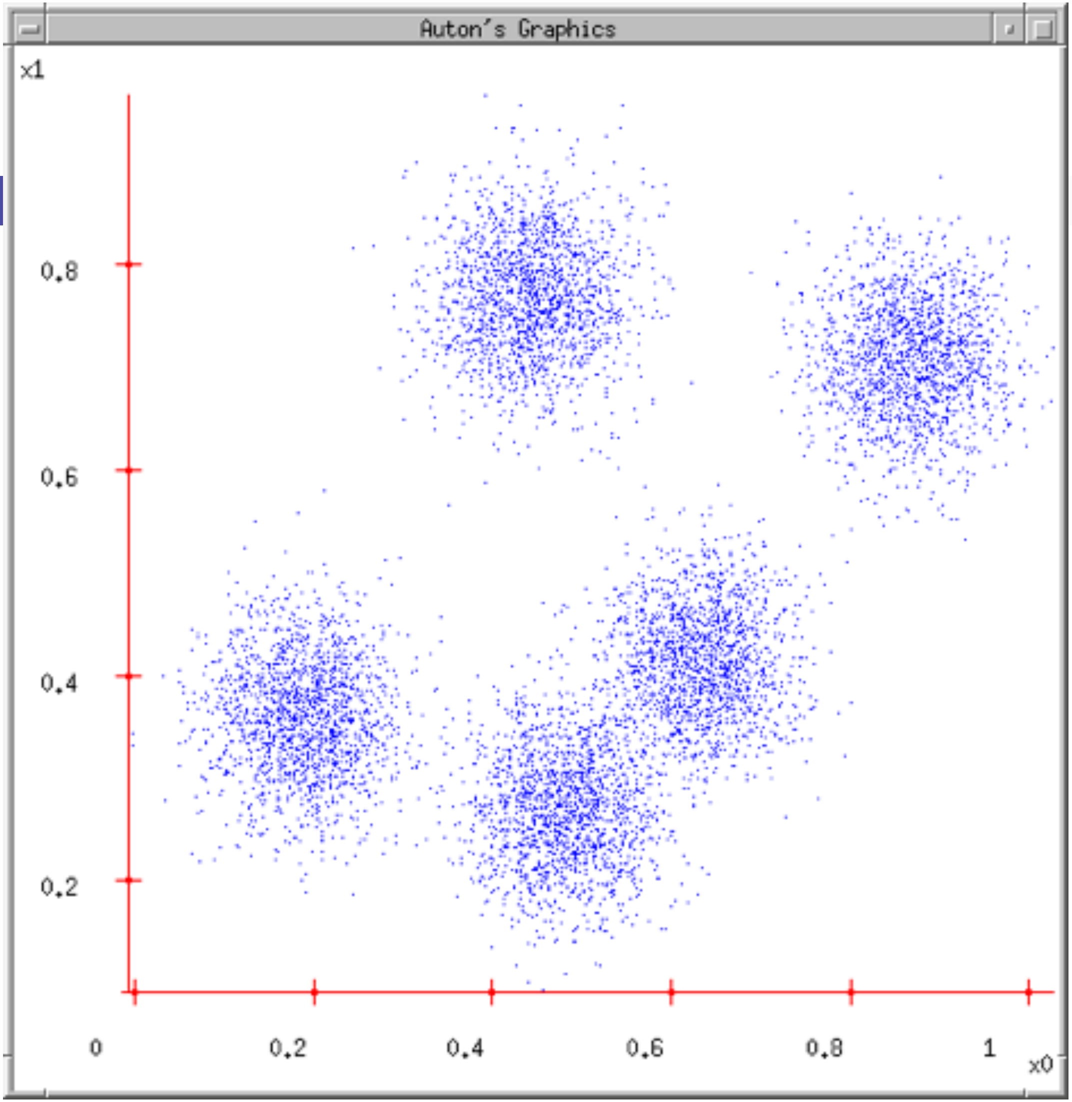
Pick one:
- Bottom up: start with every point as a cluster and merge
- Top down: start with a single cluster containing all points and split

Different rules for splitting/merging, no “right answer”

Gives apparently interpretable tree representation. However, warning: even random data with no structure will produce a tree that “appears” to be structured.



# Some Data

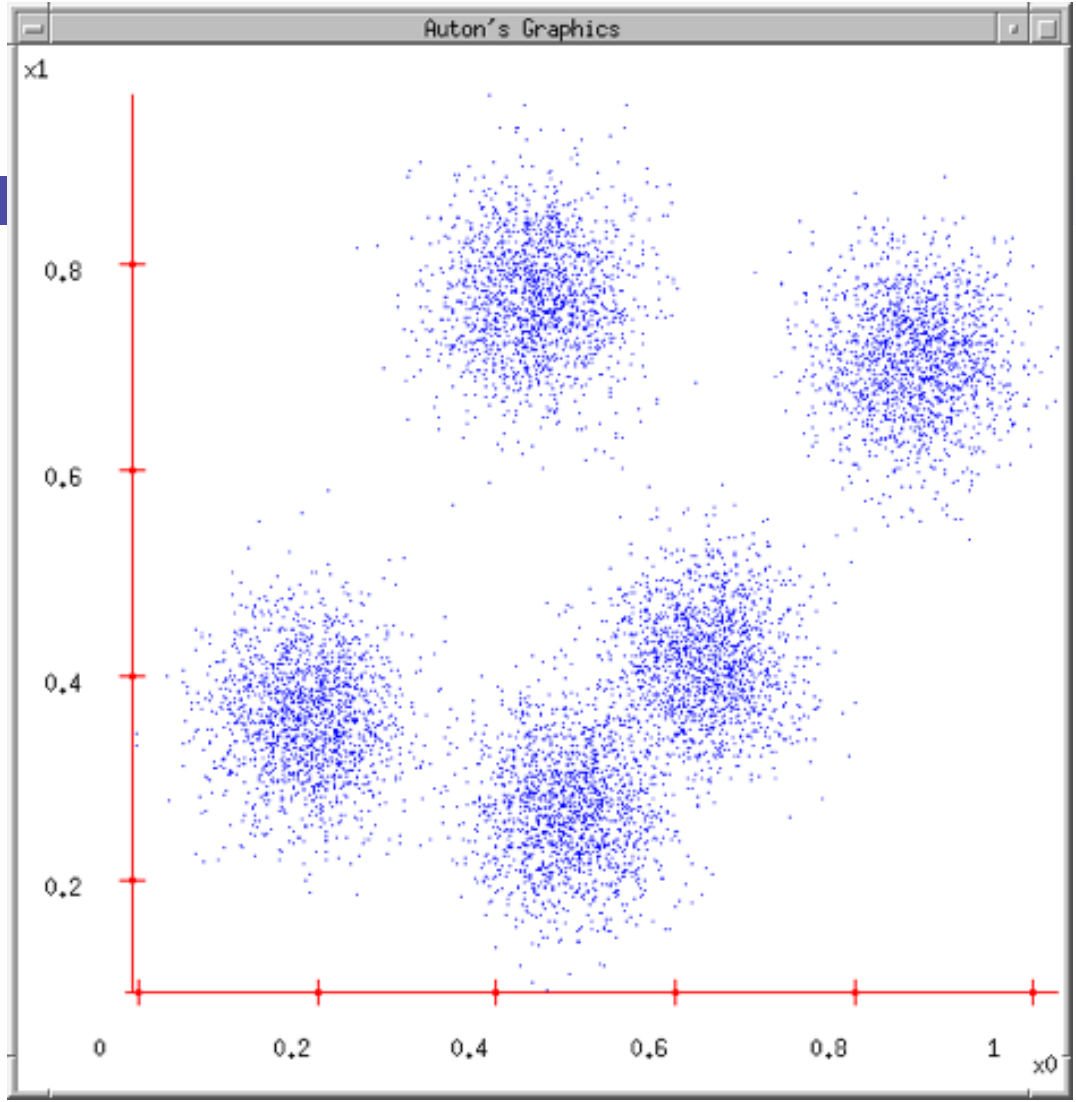




©Kevin Jamieson 2017

# K-means

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

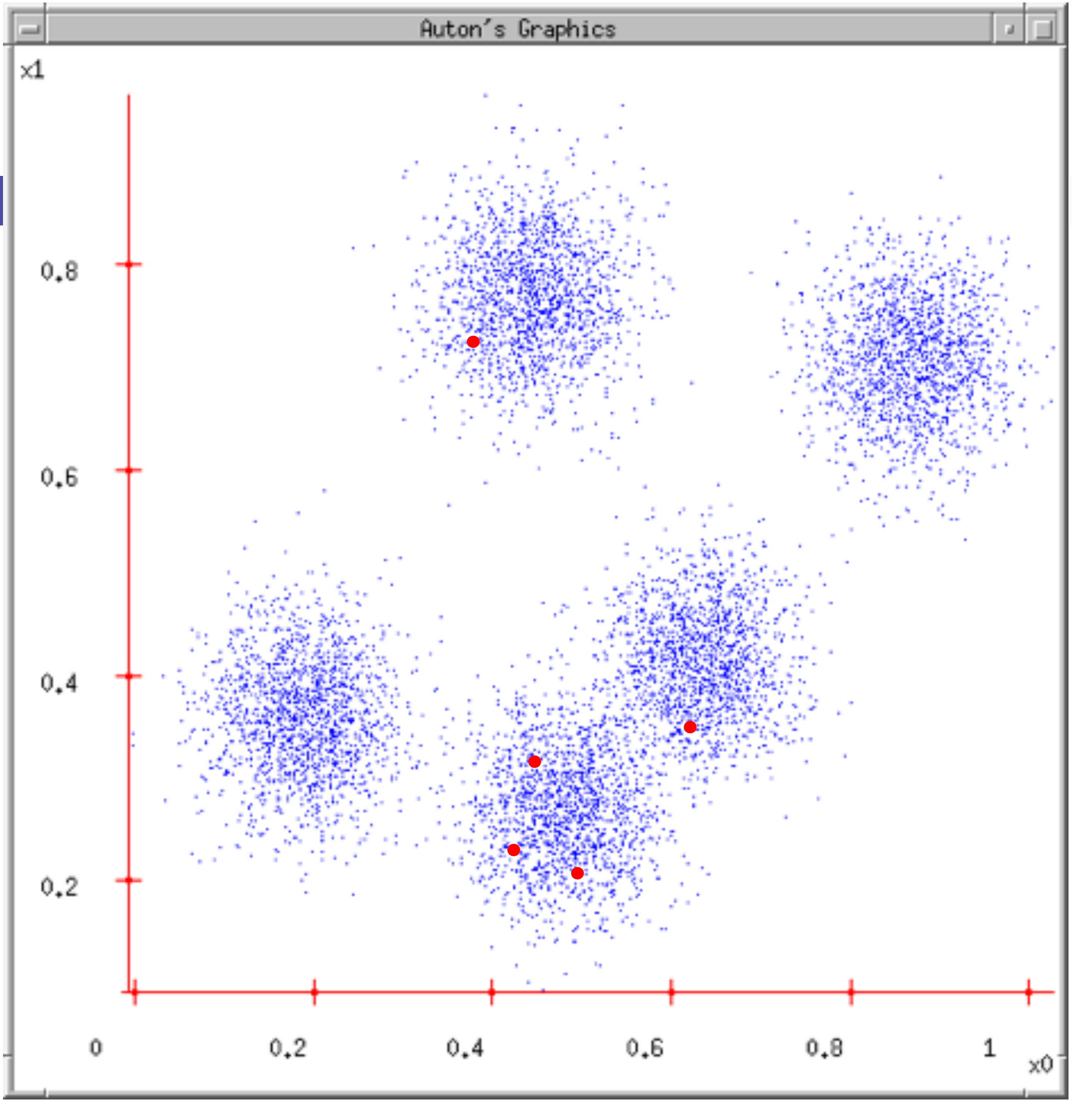





©Kevin Jamieson 2017

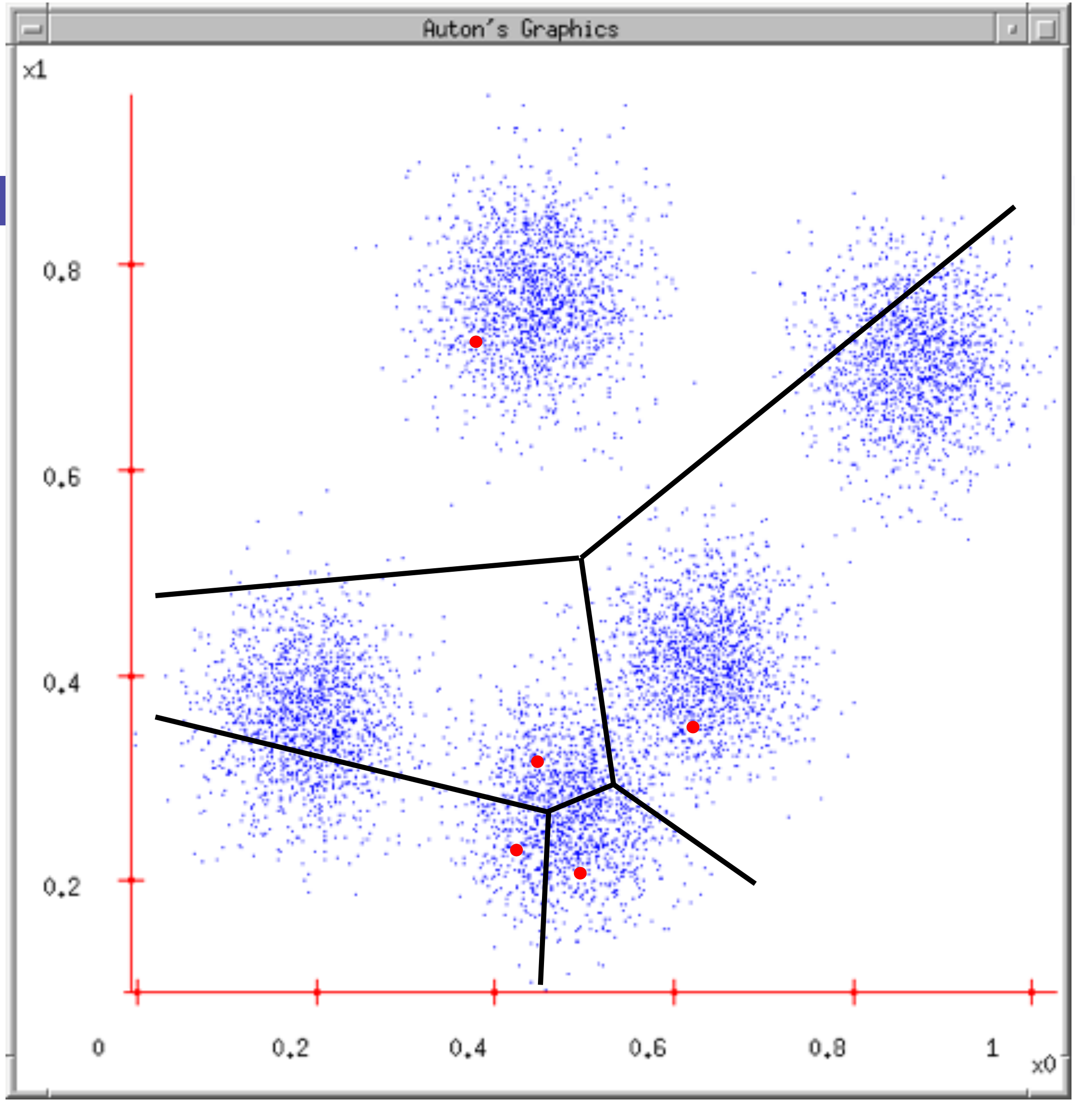
# K-means

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



# 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)

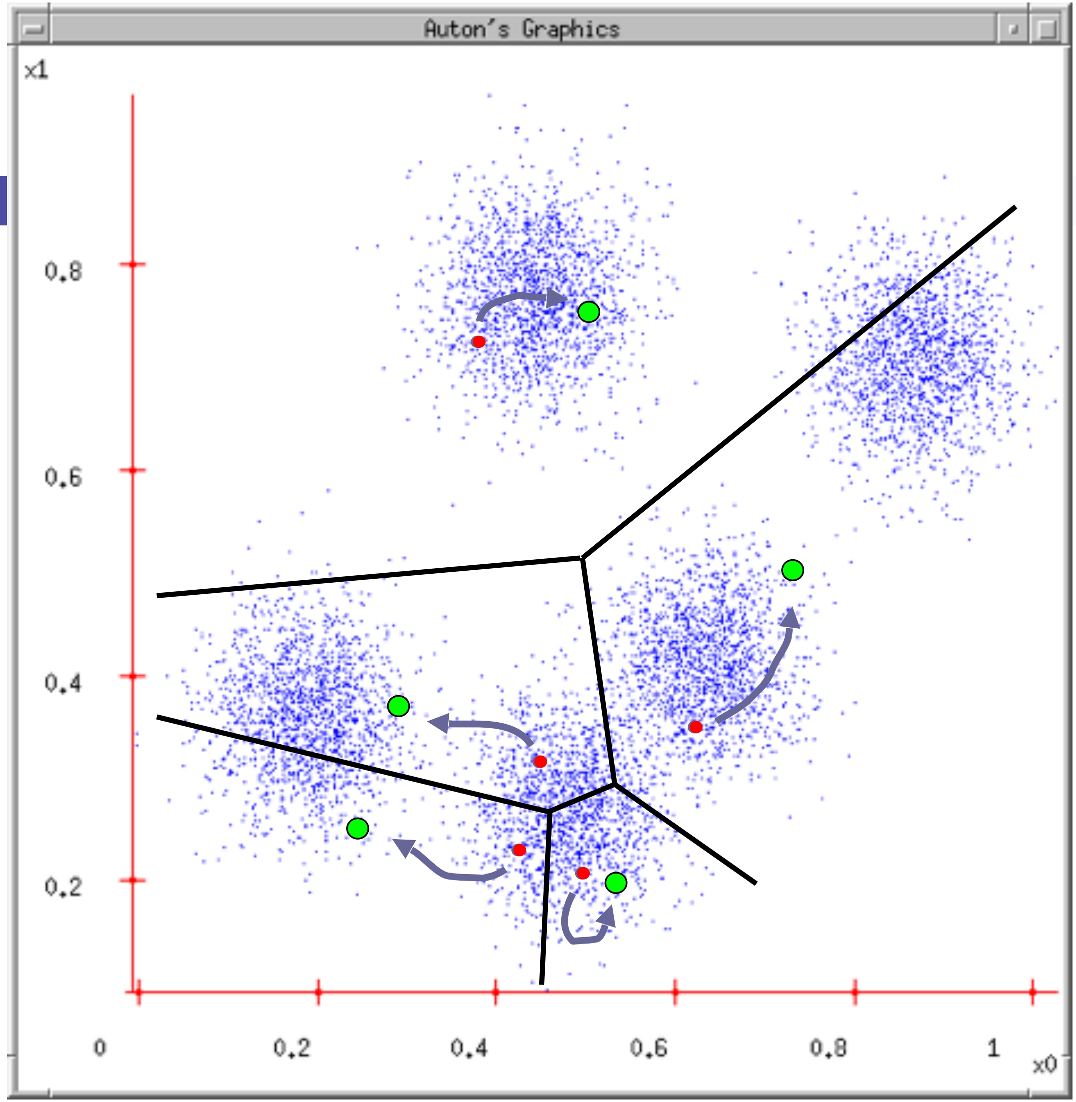




©Kevin Jamieson 2017

# 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

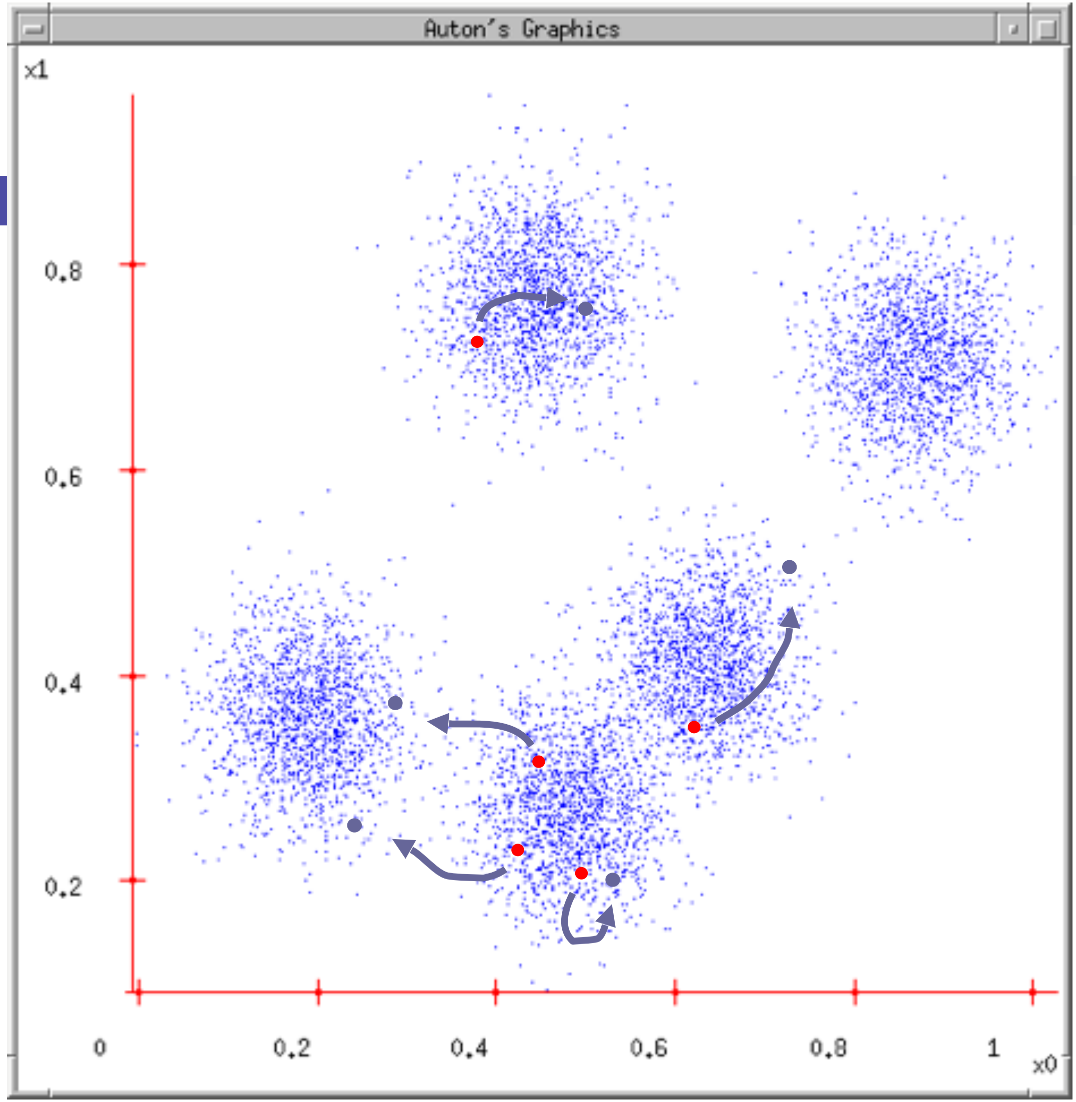





©Kevin Jamieson 2017

# 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!






©Kevin Jamieson 2017

# 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**: $\mu_i$ 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 2017

# What is K-means optimizing?

- Potential function F(μ,C) of centers μ and point allocations C:
  - $$F(\mu, C) = \sum_{j=1}^{N} ||\mu_{C(j)} - x_j||^2$$
- Optimal K-means:
  - $\min_{\mu} \min_{C} F(\mu,C)$


©Kevin Jamieson 2017

# 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 $\mu$, optimize C


©Kevin Jamieson 2017

# 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 2017

# 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.

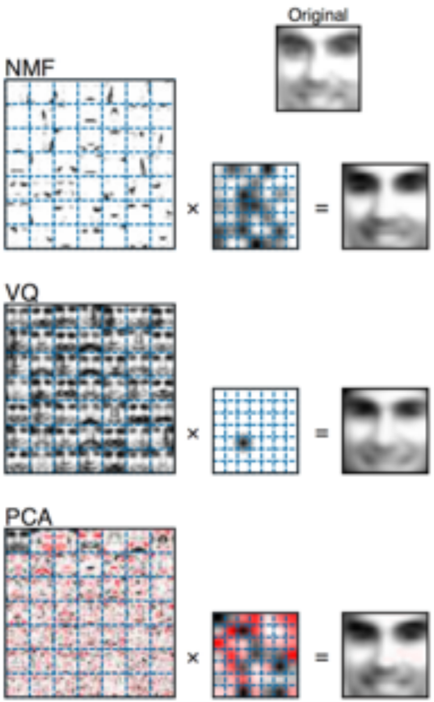
**FIGURE 14.9.** *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 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* $1024 \times 1024$ *grayscale image at 8 bits per pixel. The center image is the result of* $2 \times 2$ *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*



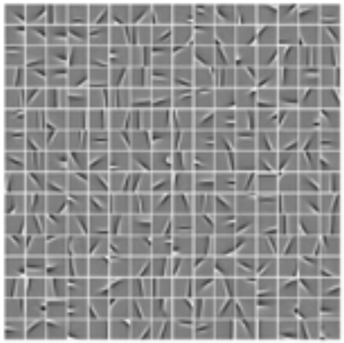
# Vector Quantization, Fisher Vectors

**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* $1024 \times 1024$ *grayscale image at 8 bits per pixel. The center image is the result of* $2 \times 2$ *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

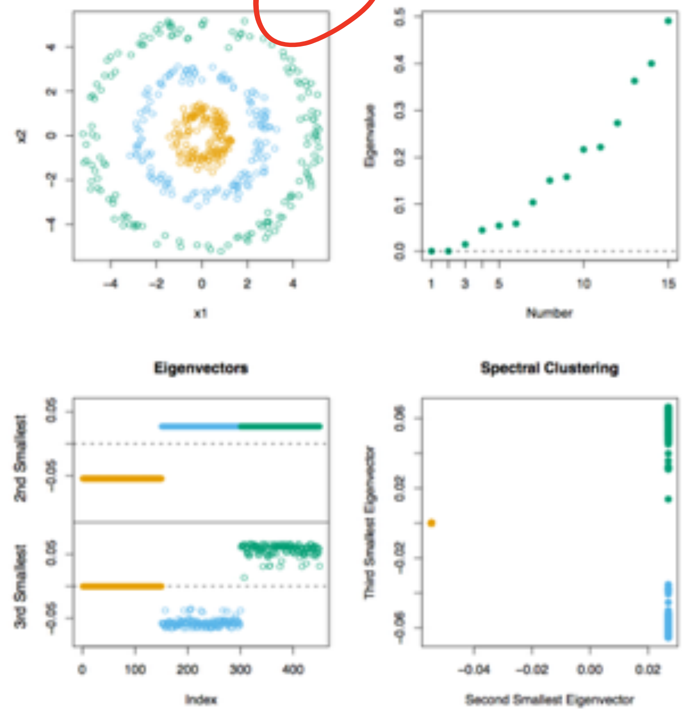
Adjacency matrix: $\mathbf{W}$

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

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

Given feature vectors, could construct:

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




Popular to use the Laplacian $\mathbf{L}$ or its normalized form $\widetilde{\mathbf{L}} = I - \mathbf{D}^{-1}\mathbf{W}$ as a regularizer for learning over graphs

# Mixtures of Gaussians

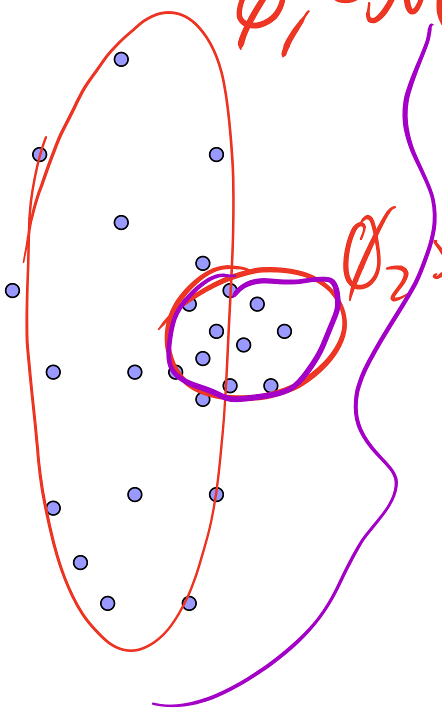
Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 21, 2016


©Kevin Jamieson 2017

# (One) bad case for k-means

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



$\phi_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

$\phi_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

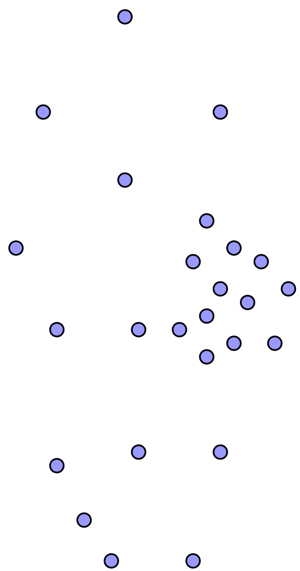
$= \pi_1 \phi_1 + \pi_2 \phi_2$



©Kevin Jamieson 2017

# (One) bad case for k-means

- Clusters may overlap
- Some clusters may be “wider” than others

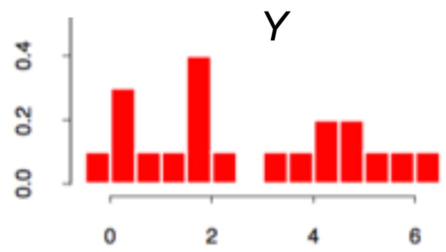



©Kevin Jamieson 2017

# Mixture models

$$
\begin{aligned}
Y_1 &\sim N(\mu_1, \sigma_1^2), \quad = \phi_{\theta_1} = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x-\mu_1)^2/2\sigma_1^2} \\
Y_2 &\sim N(\mu_2, \sigma_2^2), \\
Y &= (1-\Delta)\cdot Y_1 + \Delta \cdot Y_2,
\end{aligned}
$$

$\Delta \in \{0,1\}$ 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

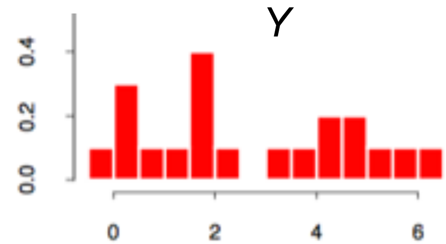
$\ell(\theta | x_i=1) = \theta$ $\quad \ell(\theta; x_i) = \theta^{x_i}(1-\theta)^{1-x_i}$

$\ell(\theta | x_i=0) = 1-\theta$

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

$\Delta \in \{0,1\}$ 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

$$\ell(\theta; y_i, \underline{\Delta_i = 0}) = \log\left(\phi_{\theta_1}(y_i)(1-\pi)\right)$$

$$\ell(\theta; y_i, \underline{\Delta_i = 1}) = \log\left(\phi_{\theta_2}(y_i)\pi\right)$$

$$\ell(\theta; y_i, \Delta_i) = (1-\Delta_i)\log((1-\pi)\phi_{\theta_1}(y_i)) + \Delta_i \log(\pi \phi_{\theta_2}(y_i))$$

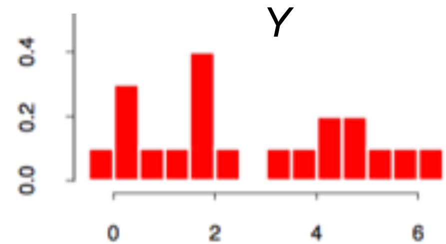

©Kevin Jamieson 2017

# 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

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

If we knew $\Delta$, how would we choose $\theta$?

$$\hat{\mu}_1 = \frac{1}{\sum_{i=1}^n (1-\Delta_i)} \sum_{i=1}^n (1-\Delta_i) y_i$$

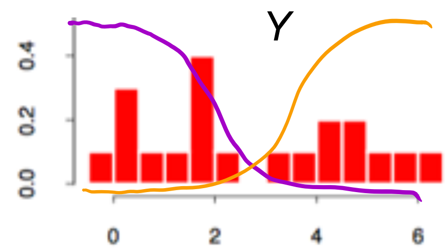

©Kevin Jamieson 2017

# 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

$\mathbf{\Delta} = \{\Delta_i\}_{i=1}^n$ 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 $\theta$, how would we choose $\mathbf{\Delta}$?

$$\mathbb{E}[\Delta_i \mid \theta, z] = \mathbb{P}(\Delta_i = 1 \mid \theta, z)$$

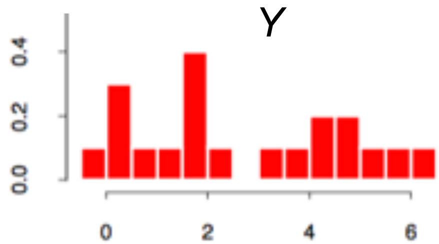
$$= \frac{\pi \phi_2(y_i)}{(1-\pi)\phi_1(y_1) + \pi \phi_2(y_i)}$$

# 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

$$E[\Delta_i \mid \theta, Z] = \hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \dots, N. \tag{8.42}$$

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

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

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

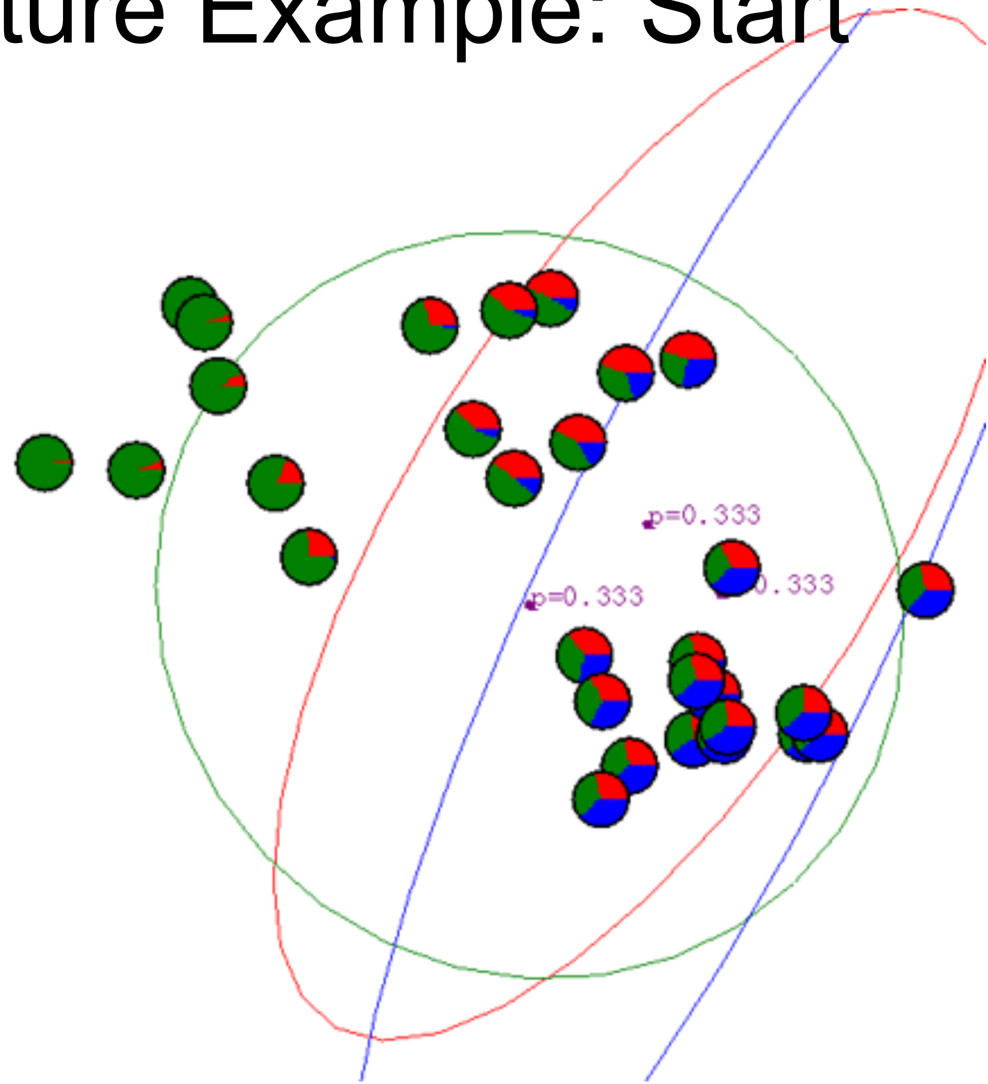
   and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.



©Kevin Jamieson 2017

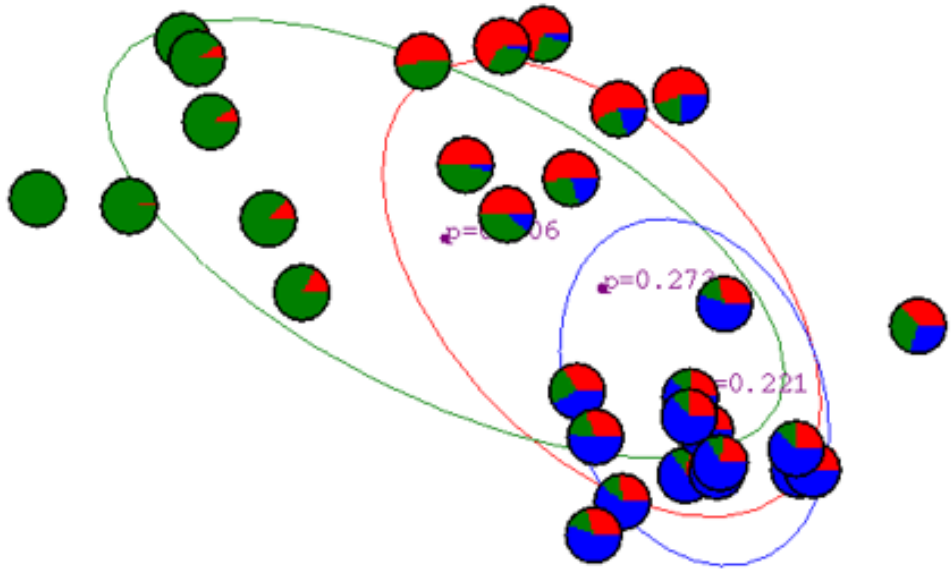
# Gaussian Mixture Example: Start





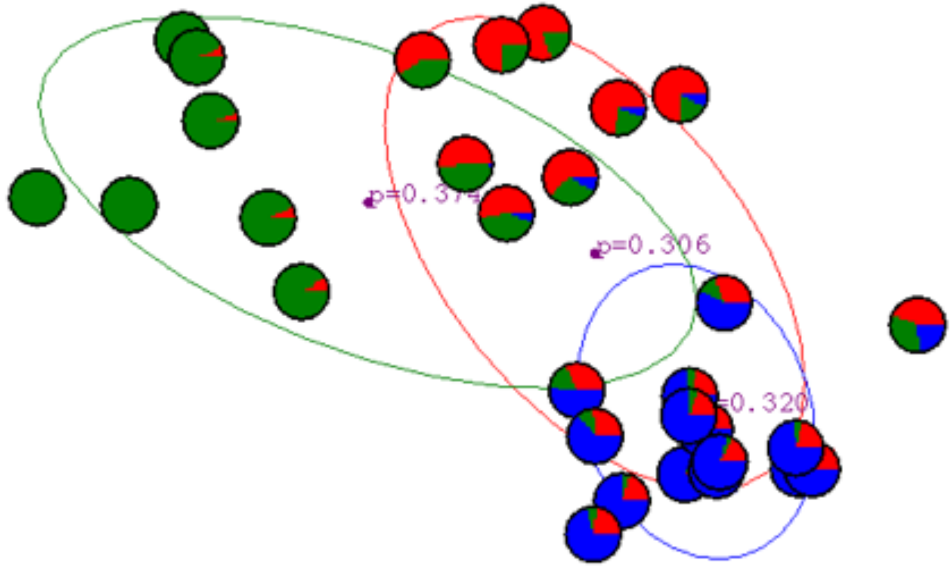

©Kevin Jamieson 2017

# After first iteration




©Kevin Jamieson 2017

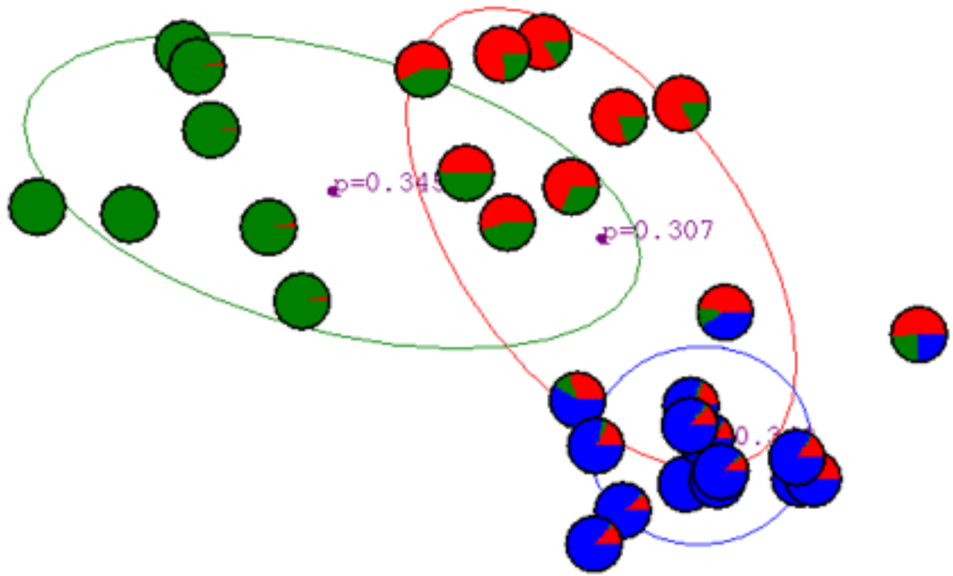
# After 2nd iteration





©Kevin Jamieson 2017

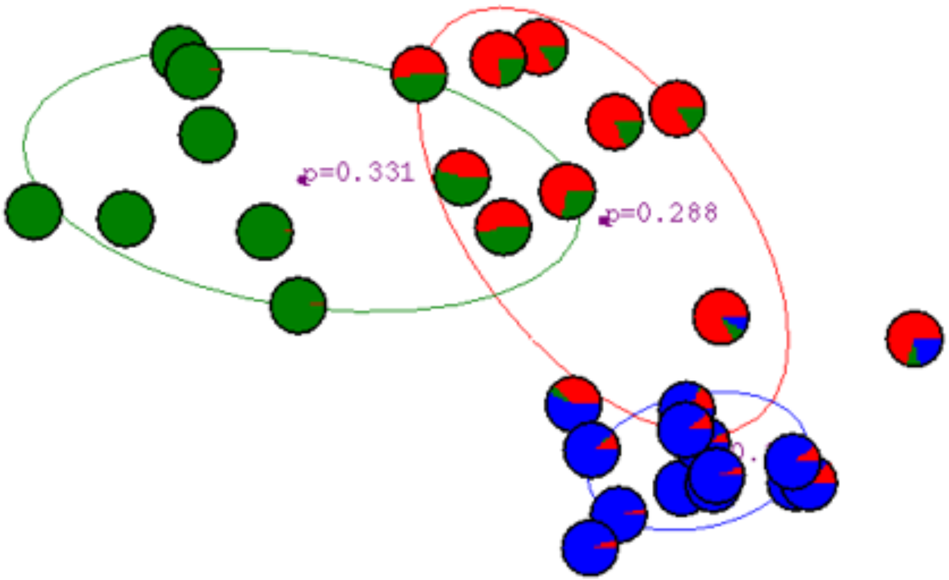
# After 3rd iteration






©Kevin Jamieson 2017

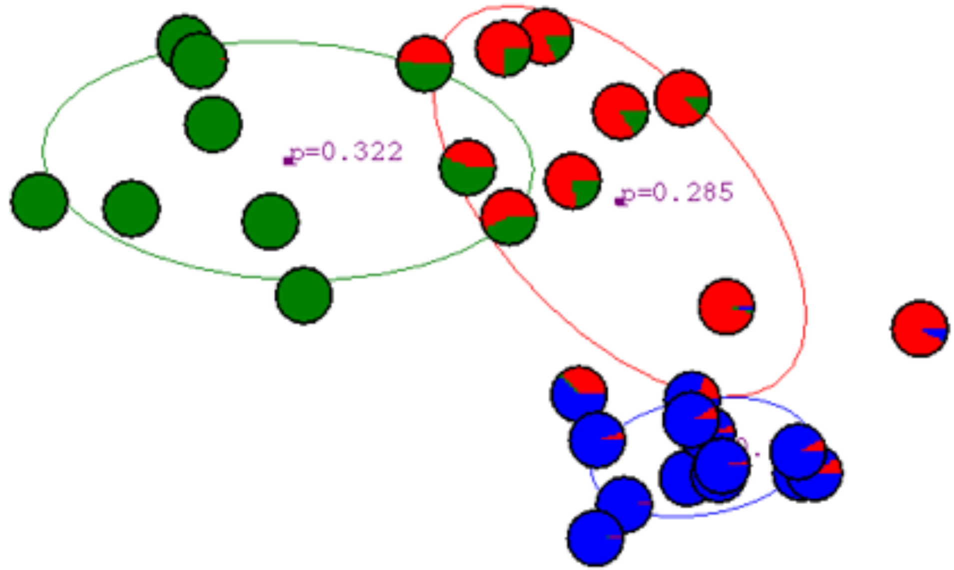
# After 4th iteration





©Kevin Jamieson 2017

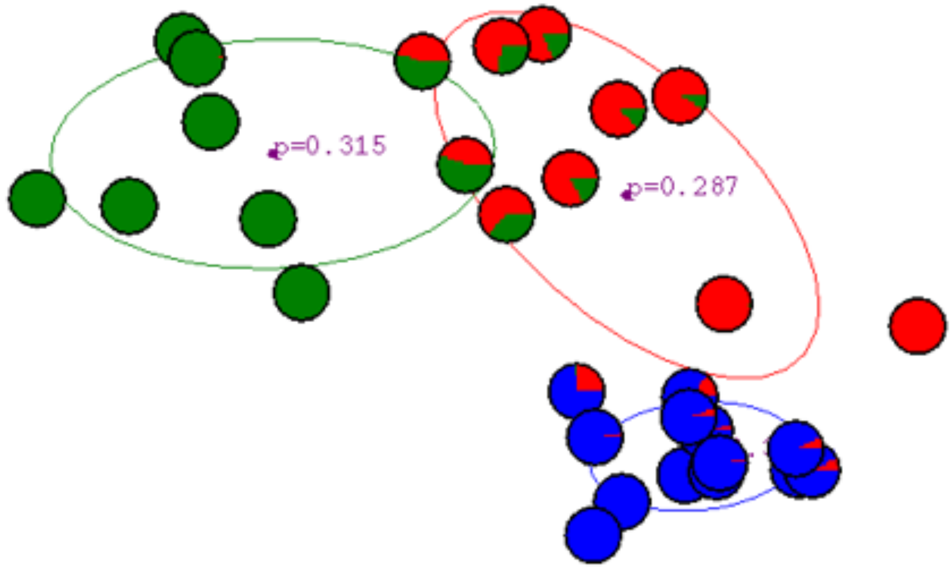
# After 5th iteration





©Kevin Jamieson 2017

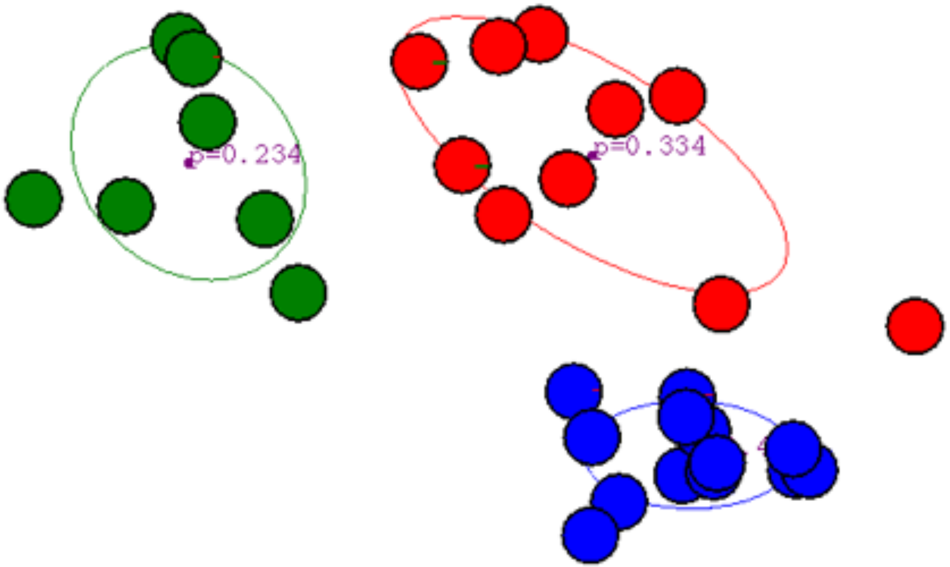
# After 6th iteration



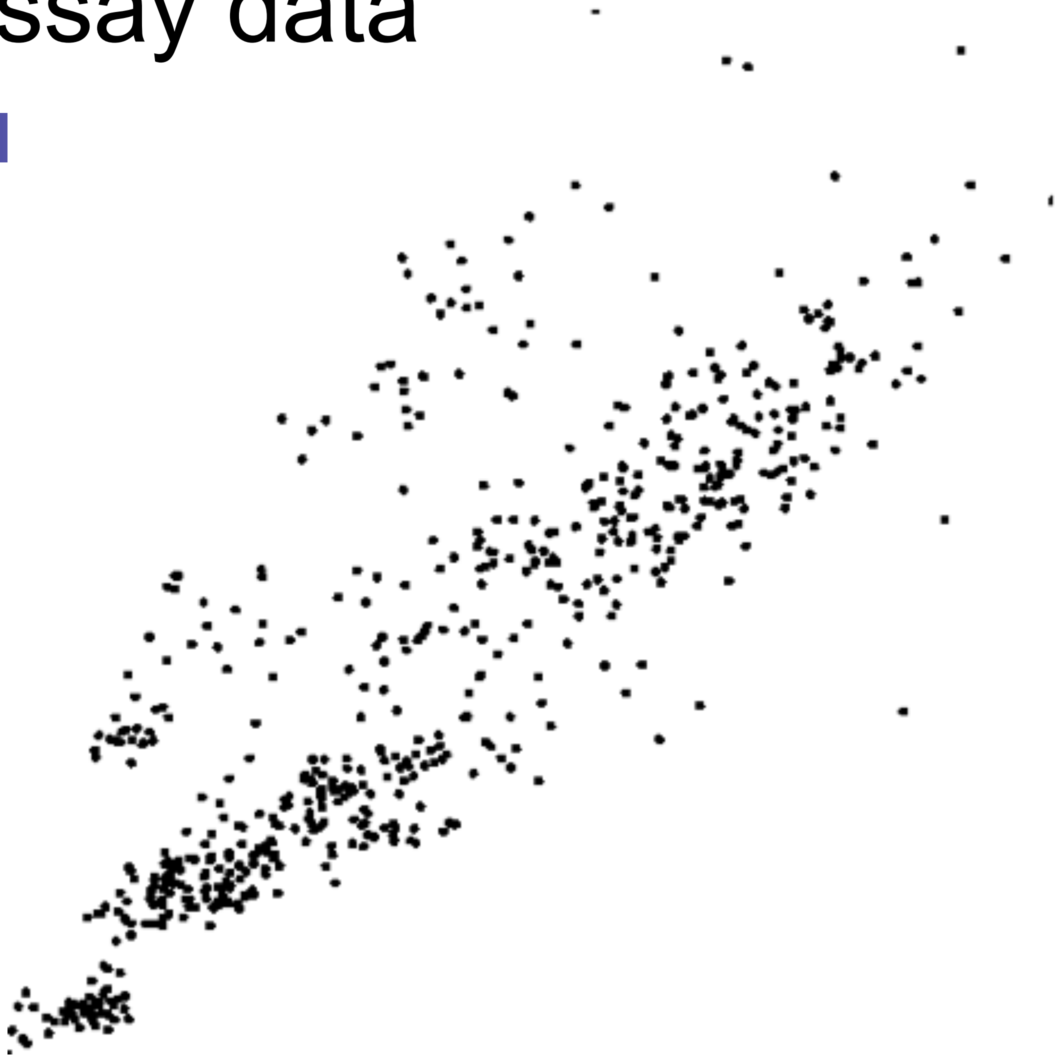



©Kevin Jamieson 2017

# After 20th iteration



# Some Bio Assay data



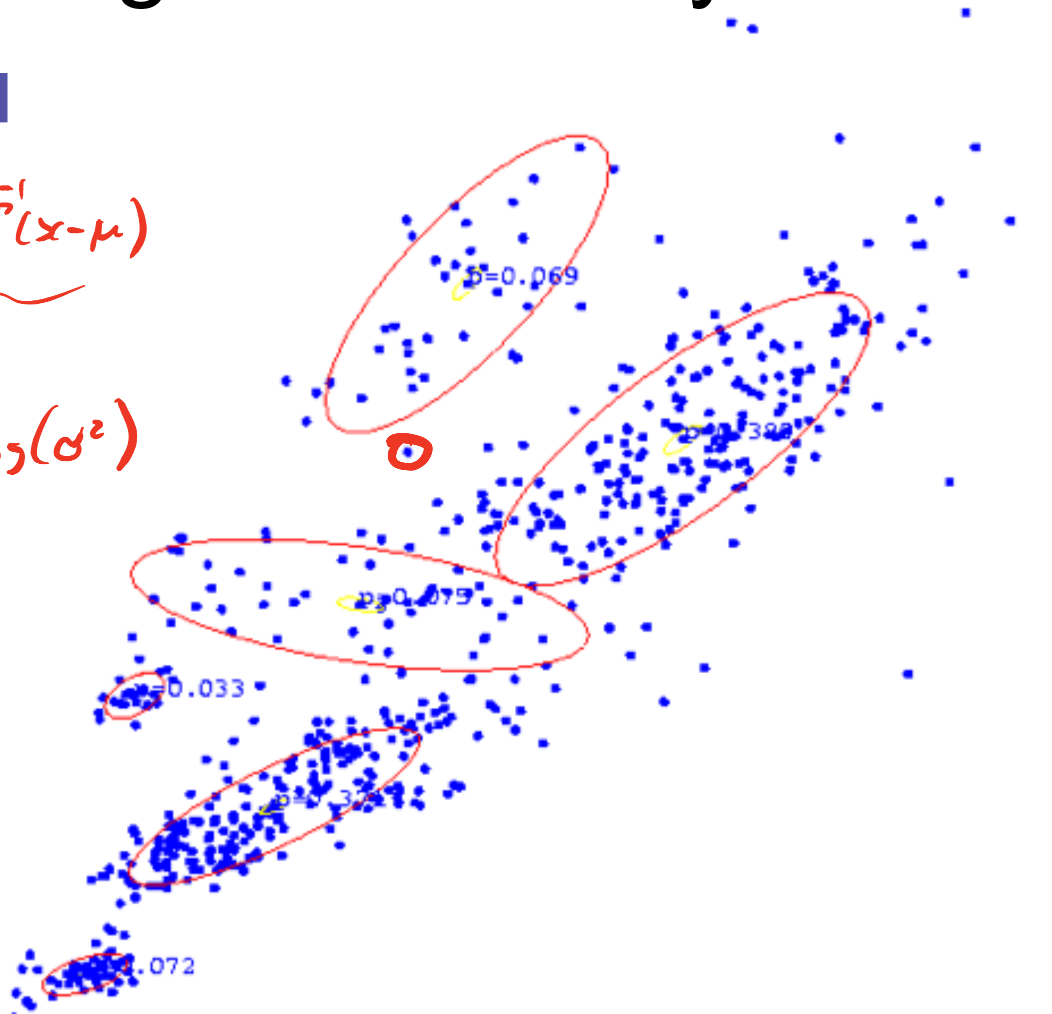


©Kevin Jamieson 2017

# GMM clustering of the assay data

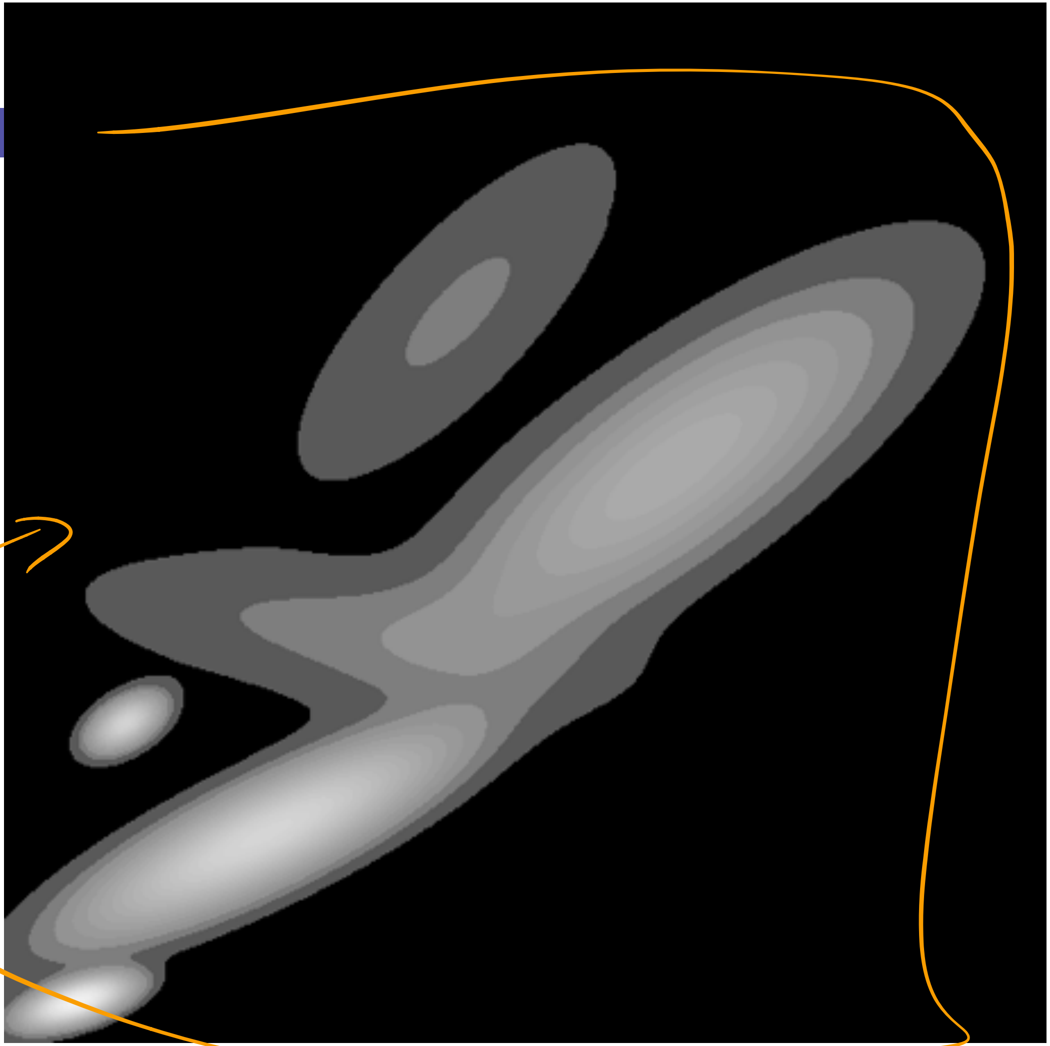
$$\underbrace{-\frac{1}{2}\log(2\pi|\Sigma|) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

$$\Sigma = \mathrm{diag}(\sigma^2)$$



# Resulting Density Estimator

$\sum_{i=1}^{k} \pi_i \phi_{\theta_i}(x)$



$x$

# Expectation Maximization Algorithm

The iterative gaussian mixture model (GMM) fitting algorithm is special case of EM:

**Algorithm 8.2** *The EM Algorithm.*

1. Start with initial guesses for the parameters $\hat{\theta}^{(0)}$.
2. *Expectation Step*: at the $j$th step, compute

   function of $\theta'$

   $$Q(\theta', \hat{\theta}^{(j)}) = \mathrm{E}(\ell_0(\theta'; \mathbf{T})|\mathbf{Z}, \hat{\theta}^{(j)}) \tag{8.43}$$

   as a function of the dummy argument $\theta'$.
3. *Maximization Step*: determine the new estimate $\hat{\theta}^{(j+1)}$ as the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over $\theta'$.
4. Iterate steps 2 and 3 until convergence.

$\mathbf{Z}$ is observed data

$\mathbf{\Delta}$ is unobserved data

$\mathbf{T} = (\mathbf{Z}, \mathbf{\Delta})$



©Kevin Jamieson 2017

# Missing data example

$x_i \sim \mathcal{N}(\mu, \Sigma)$ but suppose some entries of $x_i$ are *missing*

$x_i = \begin{bmatrix} y_i \\ \Delta_i \end{bmatrix}$

$\mathbf{Z}$ is observed data

$\boldsymbol{\Delta}$ is unobserved data

$\mathbf{T} = (\mathbf{Z}, \boldsymbol{\Delta})$

$$\ell(\theta|\mathbf{T}, \theta) = -\frac{1}{2}\log(2\pi|\Sigma|) + (x_i - \mu)^T\Sigma^{-1}(x - \mu)$$

E Step: $\mathbb{E}[\ell(\theta'; \mathbf{T})|\mathbf{Z}, \widehat{\theta}^{(j)}]$

Natural choice for $\widehat{\theta}^{(0)}$?


©Kevin Jamieson 2017

# Missing data example

$x_i \sim \mathcal{N}(\mu, \Sigma)$ but suppose some entries of $x_i$ are *missing*

$\mathbf{Z}$ is observed data

$\mathbf{\Delta}$ is unobserved data

$\mathbf{T} = (\mathbf{Z}, \mathbf{\Delta})$

$$\ell(\theta|\mathbf{T},\theta) = -\frac{1}{2}\log(2\pi|\Sigma|) + (x_i-\mu)^T\Sigma^{-1}(x-\mu)$$

E Step: $\mathbb{E}[\ell(\theta';\mathbf{T})|\mathbf{Z},\widehat{\theta}^{(j)}]$

Natural choice for $\widehat{\theta}^{(0)}$?

$$\mathbb{E}[Y|X=x] = \mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(x-\mu_X)$$

M Step: $\widehat{\theta}^{(j+1)} = \arg\max_{\theta'} \mathbb{E}[\ell(\theta';\mathbf{T})|\mathbf{Z},\widehat{\theta}^{(j)}]$


©Kevin Jamieson 2017

# Missing data example

$$x_i \sim \mathcal{N}(\mu, \Sigma)$$ but suppose some entries of $x_i$ are *missing*

**Z** is observed data

**Δ** is unobserved data

$$\mathbf{T} = (\mathbf{Z}, \boldsymbol{\Delta})$$

$$\ell(\theta|\mathbf{T}, \theta) = -\frac{1}{2}\log(2\pi|\Sigma|) + (x_i - \mu)^T \Sigma^{-1}(x - \mu)$$

E Step: $\mathbb{E}[\ell(\theta'; \mathbf{T})|\mathbf{Z}, \widehat{\theta}^{(j)}]$

Natural choice for $\widehat{\theta}^{(0)}$?

$$\mathbb{E}[Y|X = x] = \mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(x - \mu_X)$$

M Step: $\widehat{\theta}^{(j+1)} = \arg\max_{\theta'} \mathbb{E}[\ell(\theta'; \mathbf{T})|\mathbf{Z}, \widehat{\theta}^{(j)}]$

Connection to matrix factorization?



©Kevin Jamieson 2017

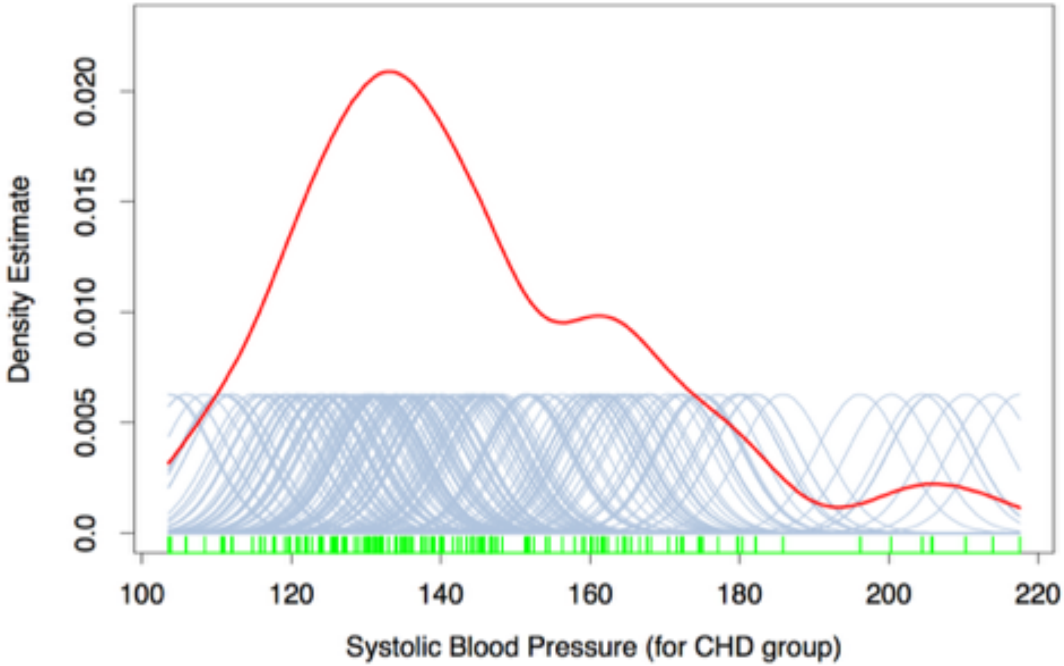
# Density Estimation

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 21, 2016


©Kevin Jamieson 2017

# Kernel Density Estimation



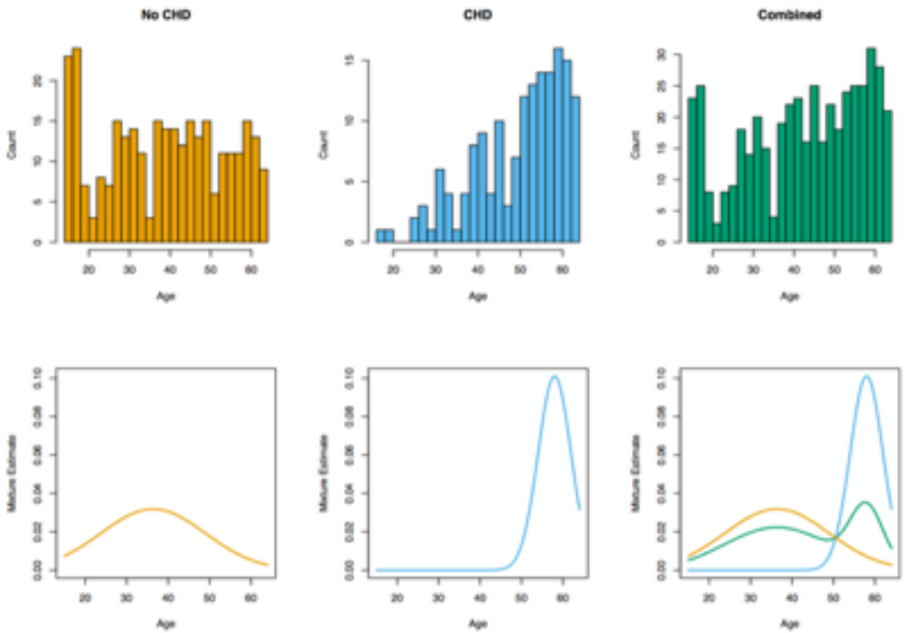

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$

A very “lazy” GMM



©Kevin Jamieson 2017

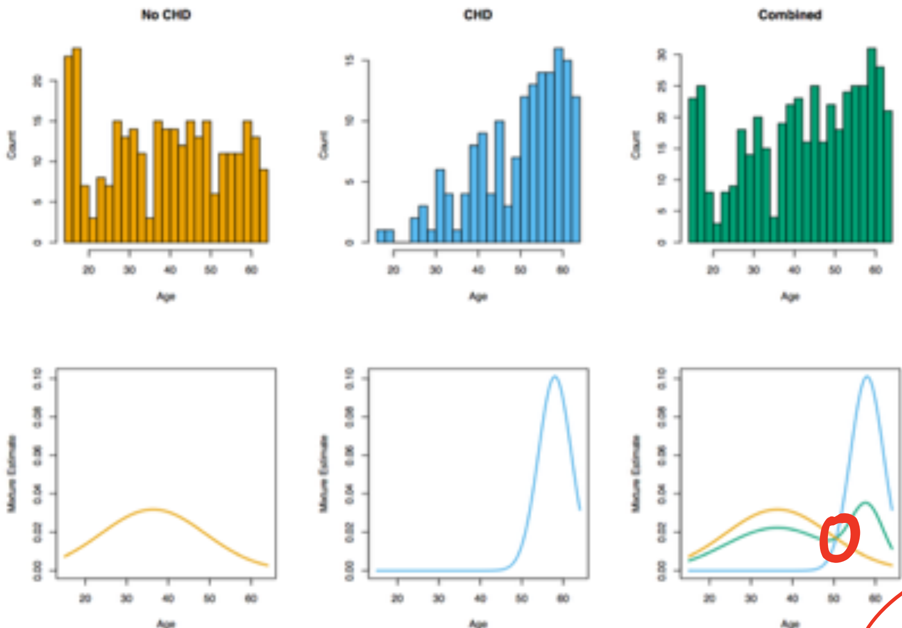
# Kernel Density Estimation




$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$


©Kevin Jamieson 2017

# Kernel Density Estimation




What is the Bayes optimal classification rule?

$$\hat{r}_{im} = \frac{\hat{\alpha}_m \phi(x_i; \hat{\mu}_m, \hat{\Sigma}_m)}{\sum_{k=1}^{M} \hat{\alpha}_k \phi(x_i; \hat{\mu}_k, \hat{\Sigma}_k)}$$

Predict $\arg\max_m \widehat{r}_{im}$

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$



©Kevin Jamieson 2017

# Generative vs Discriminative

$X \sim$ class 1 $\qquad Y \sim$ class 2

Generative rule: Fits densities to $X$ and $Y$ and then treats fitted densities as truth, and applies optimal classification

Discriminative: Makes no effort to fit densities, only fits function to decision boundary

$$\{z : P(X=z) = P(Y=z)\}$$



©Kevin Jamieson 2017