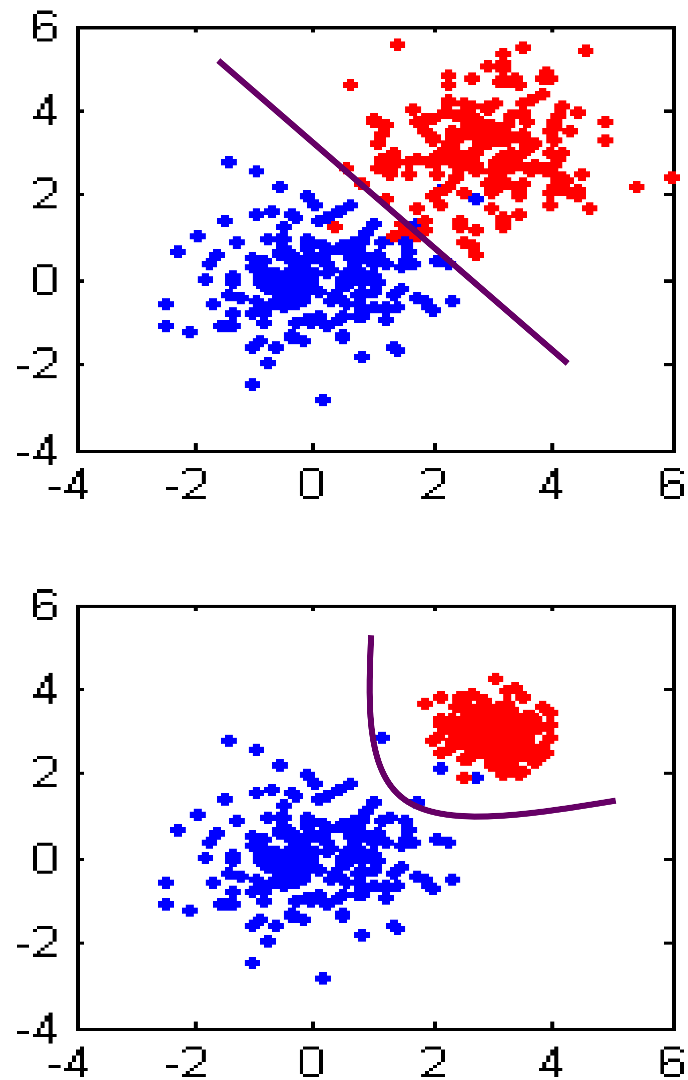
Machine Learning

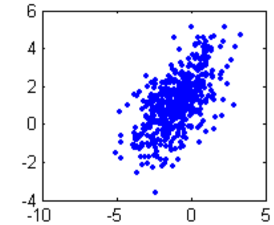
4771

Instructor: Itsik Pe'er

Reminder: Gaussians Classifiers

- Dependent features
- ML/Bayesian estimation of parameters
- Mahalanobis distance

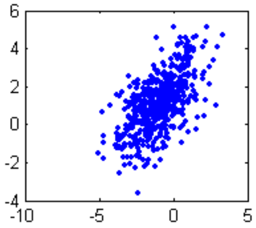




Gaussians: Key Points

- If $z \in \boldsymbol{R}^D$ multivariate normal $z \sim N(\mu, \Sigma)$ (also $z \sim MVN(\mu, \Sigma)$)

$$p(z|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right)$$



Gaussians: Key Points

- If $z \in \boldsymbol{R}^D$ multivariate normal $z \sim N(\mu, \Sigma)$ (also $z \sim MVN(\mu, \Sigma)$)

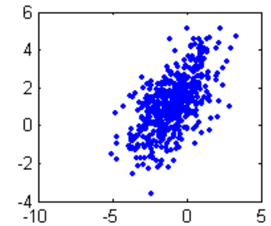
$$p(z|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T \Sigma^{-1} (z-\mu)\right)$$

- Affine transformation $w = Az + b \sim MVN(A\mu, A\Sigma A^T)$:

$$p_w(w) \propto p_z\left(A^{-1}(w-b)\right)$$

$$\propto \exp\left(-\frac{(A^{-1}(w-b)-\mu)^T \Sigma^{-1} (A^{-1}(w-b)-\mu)}{2}\right)$$

$$= \exp\left(-\frac{\left(w-(A\mu+b)\right)^T {A^{-1}}^T \Sigma^{-1} A^{-1} \left(w-(A\mu+b)\right)}{2}\right)$$



Concatenating Gaussians

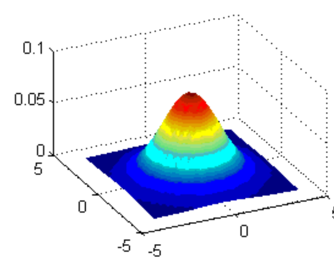
- Have input and output, each Gaussian:

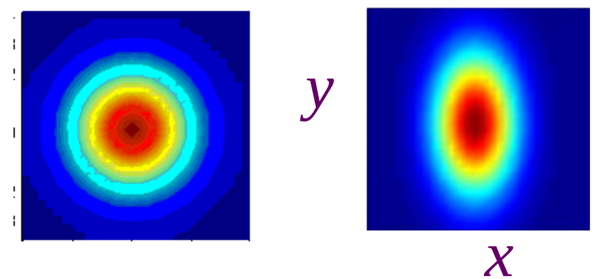
$$\{(x_1, y_1), \dots, (x_N, y_N)\} \;\; x \in \boldsymbol{R}^{D_x}, y \in \boldsymbol{R}^{D_y}, D = D_x + D_y$$

$$\text{concatenate } z_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

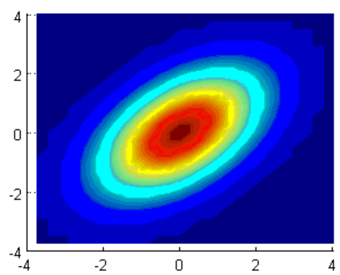
$$p(z|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T \Sigma^{-1} (z-\mu)\right)$$

Independent:

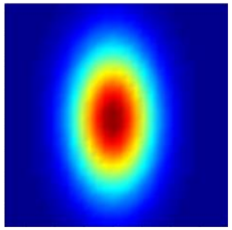




Dependent:

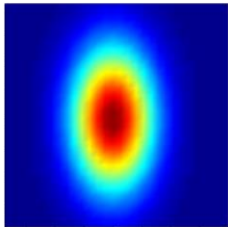


Marginals/Conditionals for Independent Gaussians



$$p(x,y) = \frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)^T \begin{bmatrix} \Sigma_{xx} & 0 \\ 0 & \Sigma_{yy} \end{bmatrix}^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}} =$$

Marginals/Conditionals for Independent Gaussians



$$p(x,y)=\frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)^T\begin{bmatrix}\Sigma_{xx} & 0\\0 & \Sigma_{yy}\end{bmatrix}^{-1}\left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}}=F_{xx}F_{yy}$$

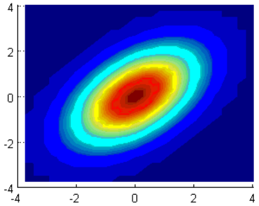
$$\begin{bmatrix}\Sigma_{xx} & 0\\0 & \Sigma_{yy}\end{bmatrix}^{-1}=\begin{bmatrix}\Sigma_{xx}^{-1} & 0\\0 & \Sigma_{yy}^{-1}\end{bmatrix}$$

$$F_{xx}=\frac{\exp\left(-\frac{(x-\mu_x)^T{\Sigma_{xx}}^{-1}(x-\mu_x)}{2}\right)}{(2\pi)^{\frac{D_x}{2}}\sqrt{|\Sigma_{xx}|}}$$

$$F_{yy}=\frac{\exp\left(-\frac{\left(y-\mu_y\right)^T{\Sigma_{yy}}^{-1}\left(y-\mu_y\right)}{2}\right)}{(2\pi)^{\frac{D_y}{2}}\sqrt{|\Sigma_{yy}|}}$$

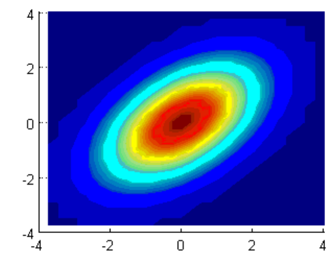
Marginals/Conditionals for Dependent Gaussians

$$p(x,y) = \frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)^T \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}}$$

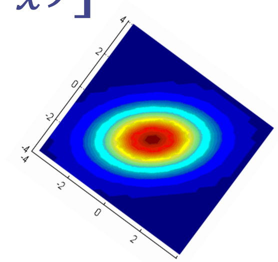


Dependent Gaussian Marginals/Conditionals

$$p(x,y) = \frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)^T\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\\Sigma_{yx} & \Sigma_{yy}\end{bmatrix}^{-1}\left(\begin{bmatrix}x\\y\end{bmatrix}-\begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}}$$



Use affine transformation $\begin{bmatrix}x\\y\end{bmatrix} \to \begin{bmatrix}x\\w = y - \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x)\end{bmatrix}$



$$cov(x,w) = \Sigma_{yx} - \Sigma_{yx}\Sigma_{xx}^{-1}\,\Sigma_{xx} = 0$$

for normal variables, uncorrelated=independent

$$\mu_w = \mu_y$$

Gaussian Marginals/Conditionals

- Conditional & marginal from joint: $p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x,y)}{\int_y p(x,y)}$
- Gaussian: $p(z|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right)$

$$p(x,y) = \frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)^T \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}}$$

$$p(x) = \frac{1}{(2\pi)^{\frac{D_x}{2}}\sqrt{|\Sigma_{xx}|}} \exp\left(-\frac{1}{2}(x-\mu_x)^T\Sigma_{xx}^{-1}(x-\mu_x)\right) = N(\mu_x, \Sigma_{xx})$$

Regression with Gaussians

- Conditional & marginal from joint: $p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x,y)}{\int_y p(x,y)}$
- Gaussian: $p(z|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right)$

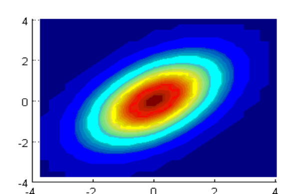
$$p(x,y) = \frac{\exp\left(-\frac{1}{2}\left(\begin{bmatrix}x\\y\end{bmatrix} - \begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)^T \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}^{-1} \left(\begin{bmatrix}x\\y\end{bmatrix} - \begin{bmatrix}\mu_x\\\mu_y\end{bmatrix}\right)\right)}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma|}}$$

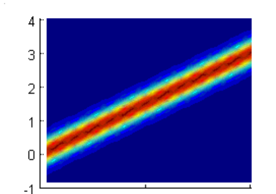
$$p(x) = \frac{1}{(2\pi)^{\frac{D_x}{2}}\sqrt{|\Sigma_{xx}|}} \exp\left(-\frac{1}{2}(x-\mu_x)^T\Sigma_{xx}^{-1}(x-\mu_x)\right) = N(\mu_x, \Sigma_{xx})$$

$$p(y|x) = N\left(\mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x), \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}\right)$$

- Here argmax is conditional expectation:

$$\hat{y} = \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x-\mu_x)$$

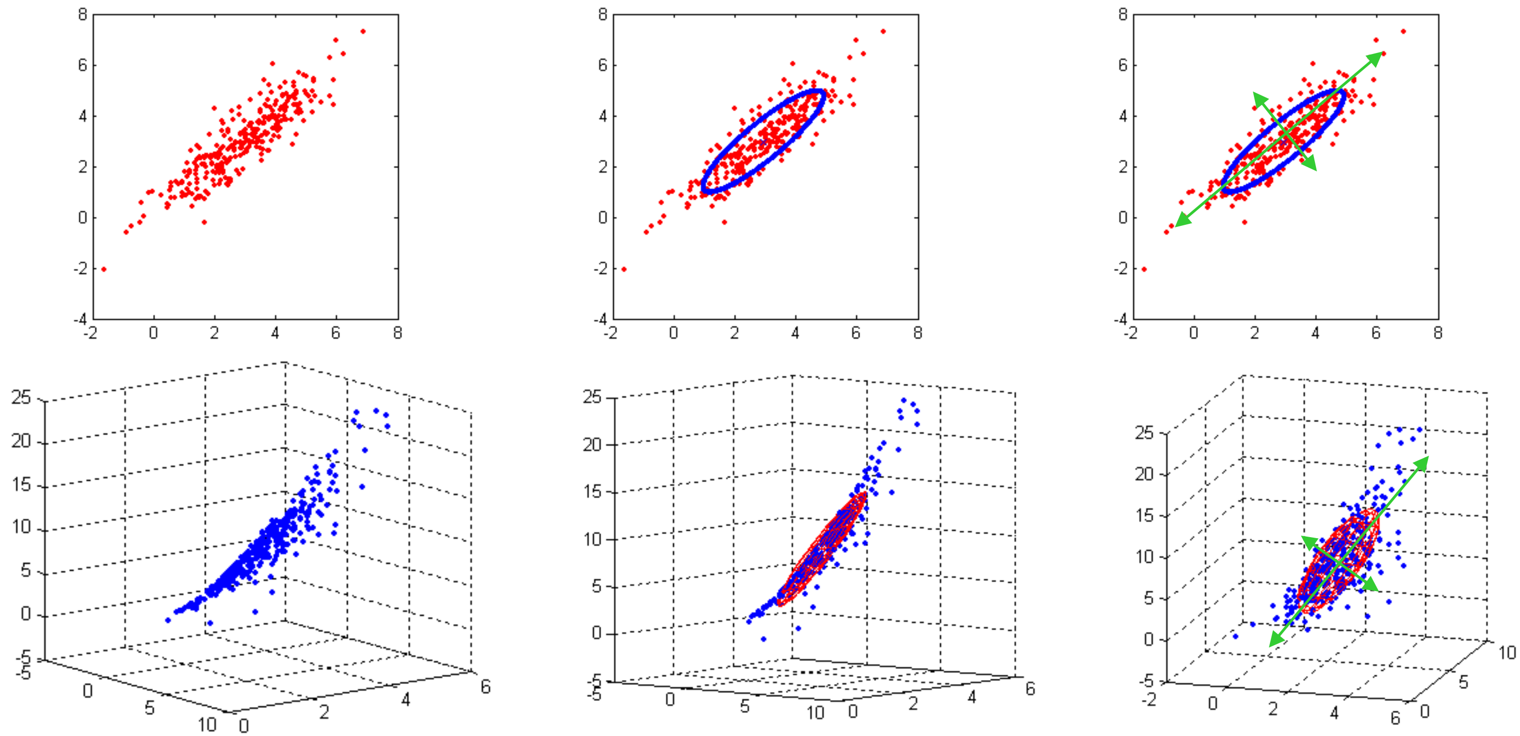




Principal Components Analysis

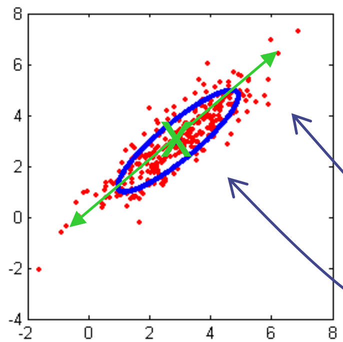
- Gaussians: for Classification, Regression… & Compression!
- Data can be constant in some directions, changes in others
- Use Gaussian to find directions of high/low variance
- Intuition: Regression = know x, measure noisy y

PCA= measure noisy x, y

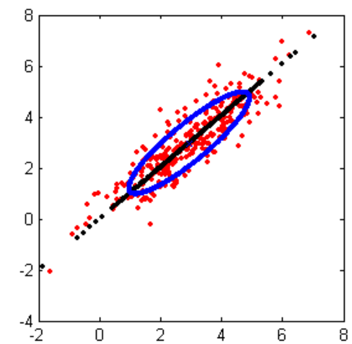


Principal Components Analysis

- Idea: instead of writing data in all its dimensions, only write it as mean + steps along one direction

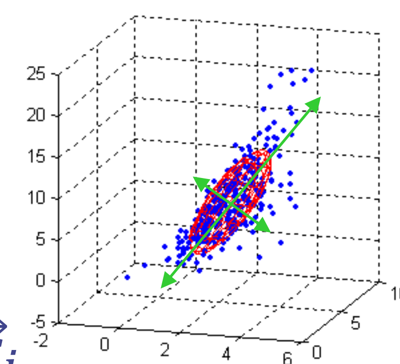


$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \approx \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} + c_i \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$



- More generally, keep a subset of dimensions

$$\vec{z}_i = \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$



- Compression method: instead of $\vec{z}_i$, only save $\vec{c}_i$
- Optimal directions: along eigenvectors of Σ
- Which directions to keep: highest eigenvalues (new variances)

Principal Components Analysis

- If we have eigenvectors, mean and coefficients:
  $\vec{z}_i = \vec{\mu} + \sum_{j=1}^{C} c_{ij}\vec{v}_j$
- Get eigenvectors $\Sigma = V\Lambda V^T$

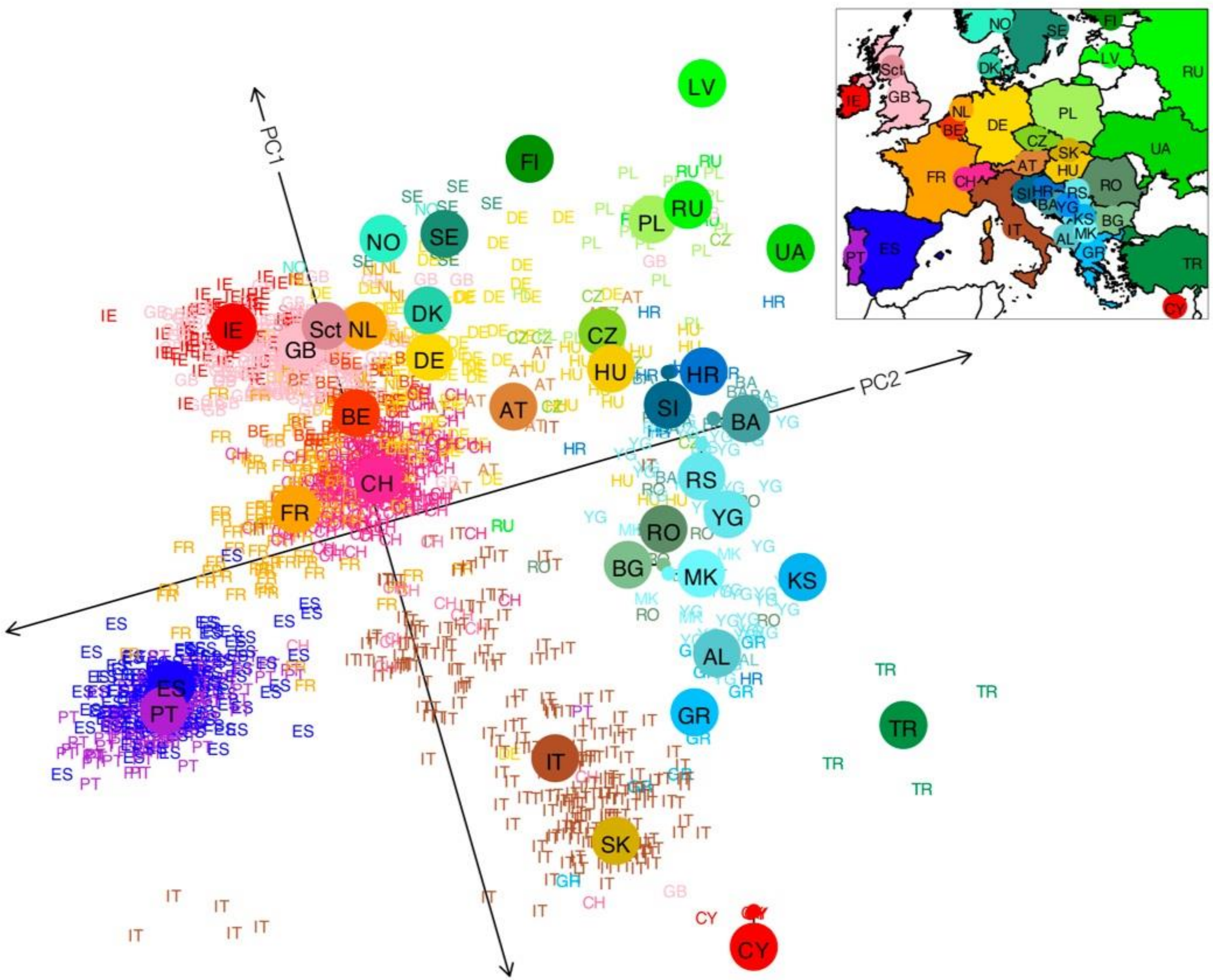
$$\begin{bmatrix} \Sigma(1,1) & \Sigma(1,2) & \Sigma(1,3) \\ \Sigma(2,1) & \Sigma(2,2) & \Sigma(2,3) \\ \Sigma(3,1) & \Sigma(3,2) & \Sigma(3,3) \end{bmatrix} = [[\vec{v}_1\,] \quad [\vec{v}_2\,] \quad [\vec{v}_3\,]] \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} [[\vec{v}_1\,] \quad [\vec{v}_2\,] \quad [\vec{v}_3\,]]^T$$

- Eigenvectors are orthonormal: $\vec{v}_i^T\vec{v}_j = \delta_{ij}$
- In coordinates of v, Gaussian is diagonal, cov = Λ
- All eigenvalues are non-negative $\lambda_i \geq 0$
- Higher eigenvalues are higher variance, use the top C ones

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \cdots$$

- To compute the coefficients: $c_{ij} = (\vec{z}_i - \vec{\mu})^T\vec{v}_j$

PCA in Genetics



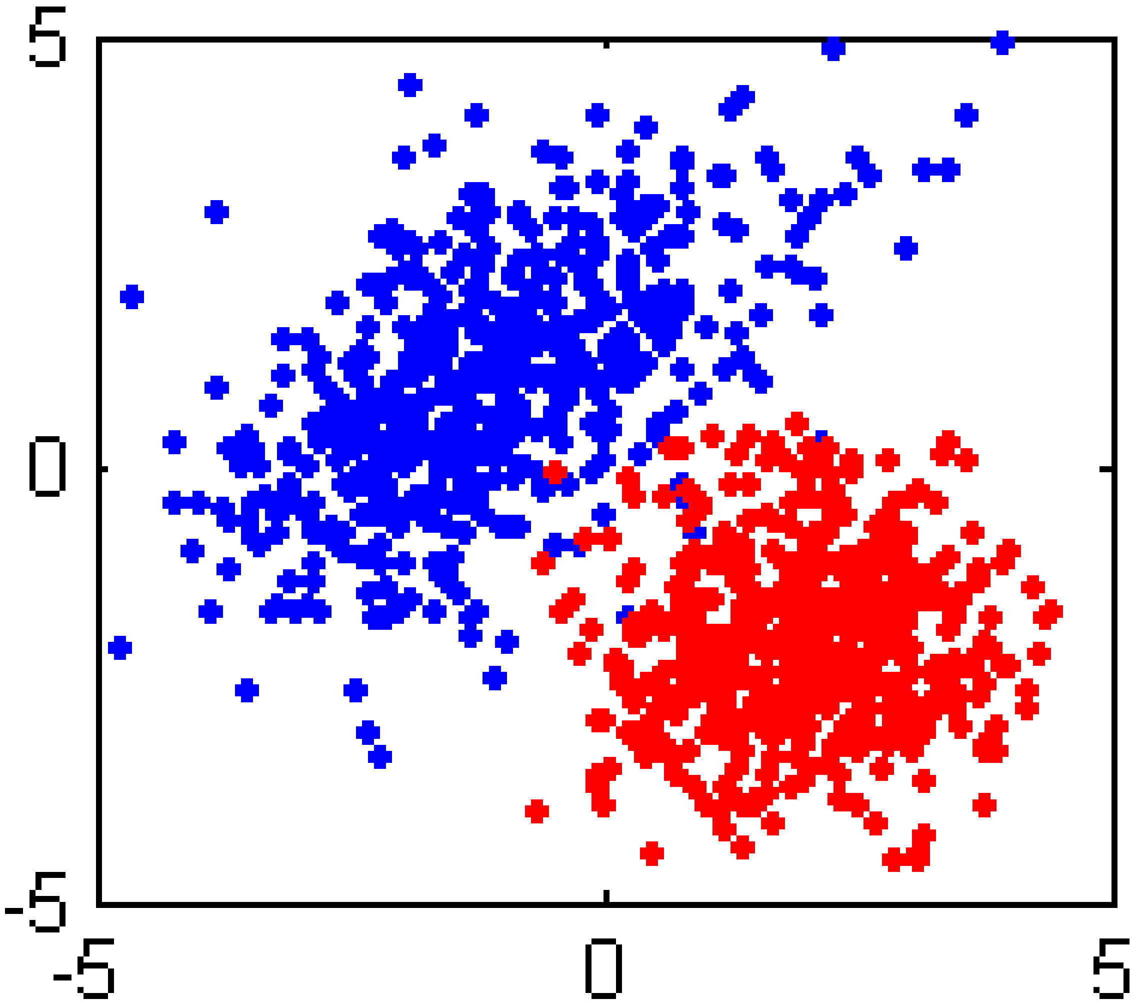
How Many Eigenvectors Needed?

- Not necessary to fully decompose
- Greedy algorithm:
  - Iteratively find largest λ & peel off vector

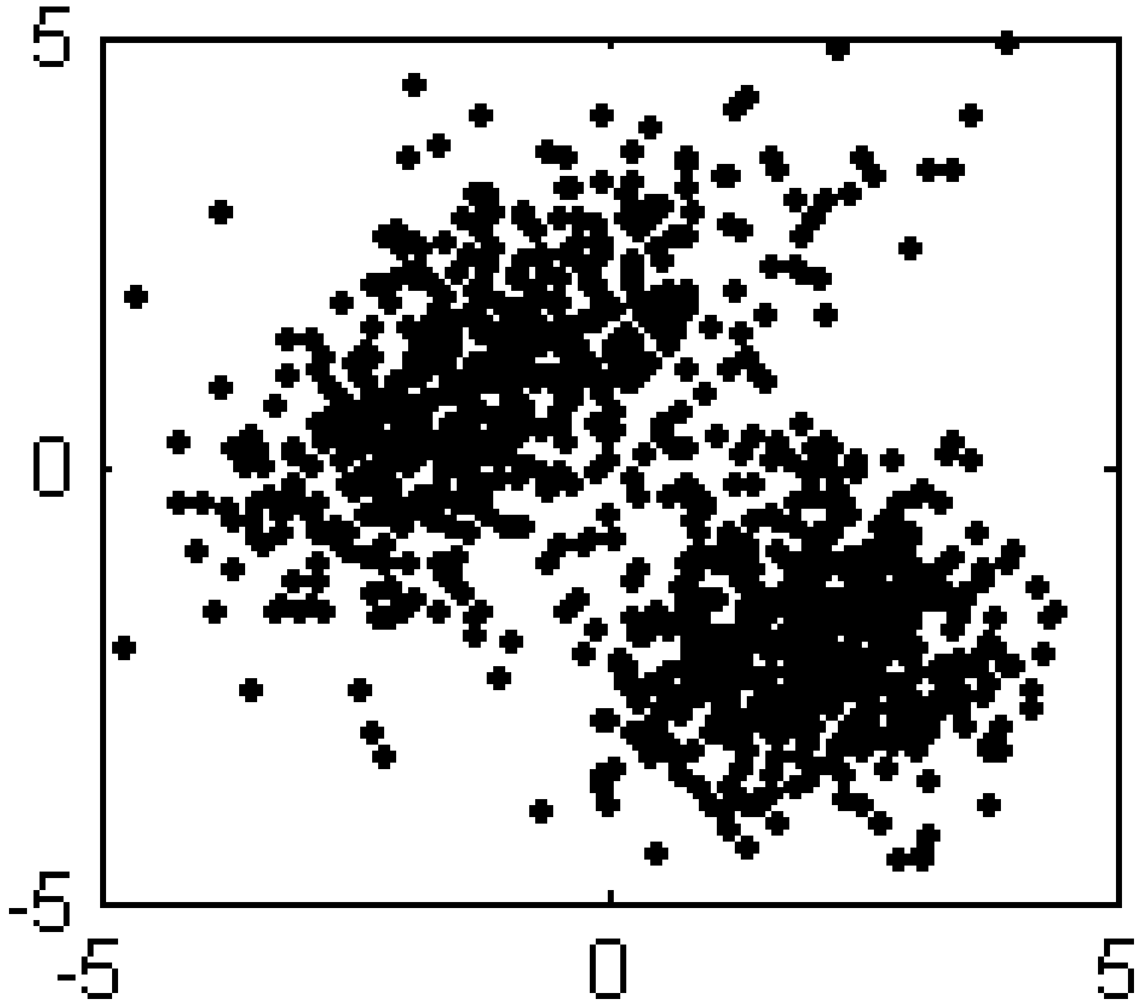
How Many Eigenvectors Needed?

- Not necessary to fully decompose
- Greedy algorithm:
  - Iteratively find largest λ & peel off vector
  - Repeatedly multiply a start vector & normalize
- Tracy-Widom statistics:
  - Null distribution of %variance λ_k explains

Two Gaussians



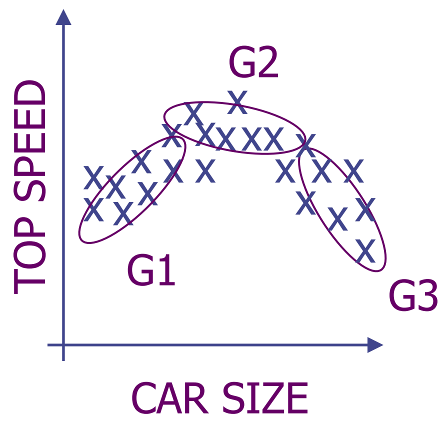
Two Unknown Gaussians



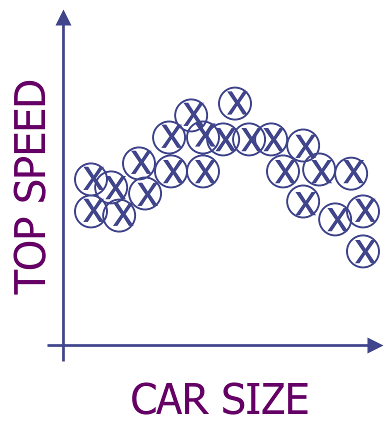
Mixtures for More Flexibility

- With mixtures (e.g. mixtures of Gaussians) we can handle more complicated (e.g. multi-bump, nonlinear) distributions.

  subpopulations: G1=compact car
  G2=mid-size car
  G3=cadillac

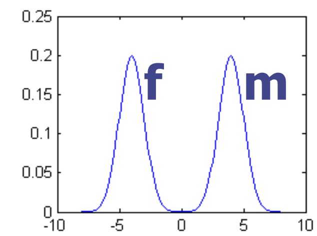


- In fact, if we have enough Gaussians (maybe infinite) we can approximate any distribution...

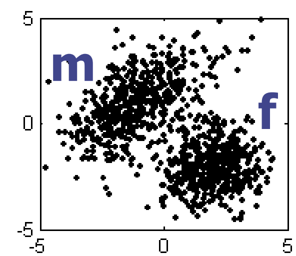


Mixtures as Hidden Variables

- Consider a dataset with K subpopulations but don't know which subpopulation each point belongs to



e.g. looking at height of adult people, we see K=2 subpopulations: males & females



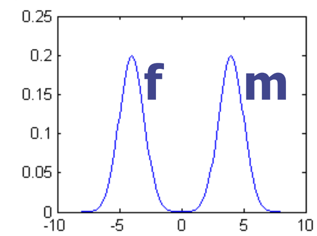
e.g. looking at pitch and height of people we see K=2 subpopulations: males & females

- Because of the 'hidden' variable (y can be 1 or 2), these distributions are not Gaussians but Mixture of Gaussians

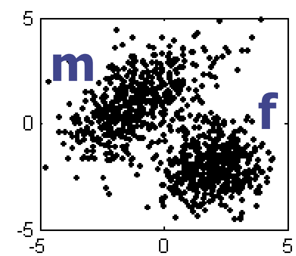
$p(\vec{x}) =$

Mixtures as Hidden Variables

- Consider a dataset with K subpopulations but don't know which subpopulation each point belongs to



e.g. looking at height of adult people, we see K=2 subpopulations: males & females



e.g. looking at pitch and height of people we see K=2 subpopulations: males & females

- Because of the 'hidden' variable (y can be 1 or 2), these distributions are not Gaussians but Mixture of Gaussians

$$p(\vec{x}) = \sum_y p(y)p(\vec{x}|y) = \sum_y \pi_y N\left(\vec{x}|\vec{\mu}_y, \Sigma_y\right)$$

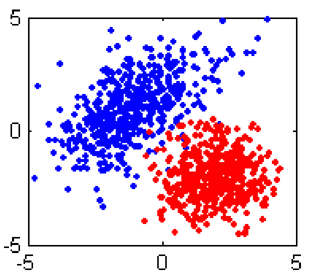
$$= \sum_y \pi_y \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma_y|}} \exp\left(-\frac{1}{2}\left(\vec{x}-\vec{\mu}_y\right)^T \Sigma_y^{-1}\left(\vec{x}-\vec{\mu}_y\right)\right)$$

Hidden / Unlabeled = Clustering

- Recall classification problem:

  maximize the log-likelihood of data given models:

  $l =$

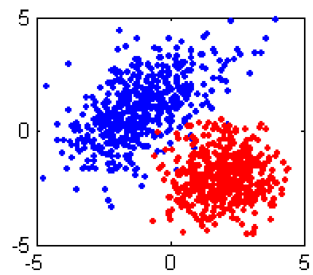


Hidden / Unlabeled = Clustering

- Recall classification problem:

  maximize the log-likelihood of data given models:

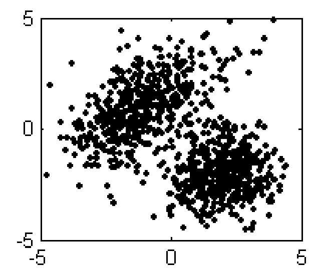
$$l = \sum_{n=1}^{N} \log p(\vec{x}_n, y_n | \pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \log \pi_{y_n} + \log N\left(\vec{x}_n \,\middle|\, \vec{\mu}_{y_n}, \Sigma_{y_n}\right)$$



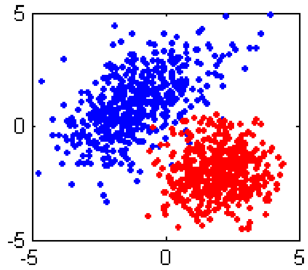
- If we don't know the class treat it as a hidden variable

  maximize the log-likelihood with unlabeled data:

$$l = \sum_{n=1}^{N} \log p(\vec{x}_n | \pi, \mu, \Sigma) =$$



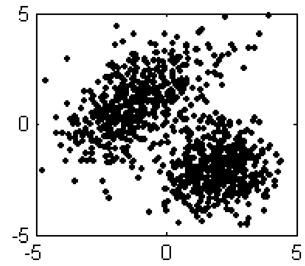
Hidden / Unlabeled = Clustering



- Recall classification problem:

  maximize the log-likelihood of data given models:

$$l = \sum_{n=1}^{N} \log p(\vec{x}_n, y_n | \pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \log \pi_{y_n} + \log N\left(\vec{x}_n \,|\vec{\mu}_{y_n}, \Sigma_{y_n}\right)$$



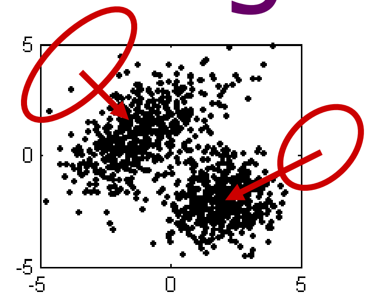
- If we don't know the class treat it as a hidden variable

  maximize the log-likelihood with unlabeled data:

$$l = \sum_{n=1}^{N} \log p(\vec{x}_n | \pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \sum_{y=1}^{K} p(\vec{x}_n, y \,| \pi, \mu, \Sigma)$$
$$= \sum_{n=1}^{N} \log(\pi_1 N(\vec{x}_n \,|\vec{\mu}_1, \Sigma_1) + \cdots + \pi_K N(\vec{x}_n \,|\vec{\mu}_K, \Sigma_K))$$

- Instead of classification, we now have a clustering problem

Hidden / Unlabeled = Clustering



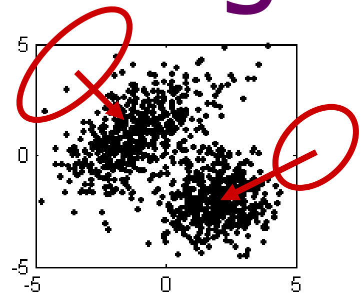
- Represent each hidden y integer (1 to K) with a hidden binary indicator vector z
  $\vec{z} \in \boldsymbol{B}^K$, $\sum_{i=1}^{K} \vec{z}(i) = 1$ or

$\vec{z} \in \{\vec{\delta}_1, \dots, \vec{\delta}_K\}$ where $\vec{\delta}_i(i) = 1, \vec{\delta}_i\,(j) = 0\ for\ i \neq j$

- Each likelihood requires summing over the possible z

$p(\vec{x}|\theta) =$

Hidden / Unlabeled = Clustering



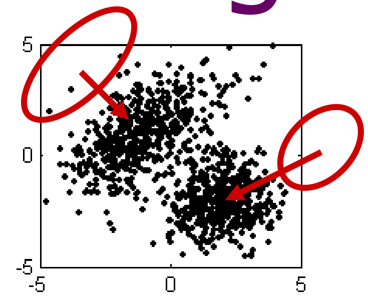
- Represent each hidden y integer (1 to K) with a hidden binary indicator vector z
  $\vec{z} \in \boldsymbol{B}^K$, $\sum_{i=1}^{K} \vec{z}(i) = 1$ or

$\vec{z} \in \{\vec{\delta}_1, \dots, \vec{\delta}_K\}$ where $\vec{\delta}_i(i) = 1, \vec{\delta}_i\,(j) = 0\ for\ i \neq j$

- Each likelihood requires summing over the possible z

$$p(\vec{x}|\theta) = \sum_z p(\vec{z}|\theta)p(\vec{x}|\vec{z},\theta) = \sum_{i=1}^{K} p\left(\vec{z} = \vec{\delta}_i\middle|\theta\right)p\left(\vec{x}\middle|\vec{z} = \vec{\delta}_i,\theta\right)$$

Hidden / Unlabeled = Clustering



- Represent each hidden y integer (1 to K) with a hidden binary indicator vector z

  $\vec{z} \in \boldsymbol{B}^K, \sum_{i=1}^{K} \vec{z}(i) = 1$ or

  $\vec{z} \in \{\vec{\delta}_1, \ldots, \vec{\delta}_K\}$ where $\vec{\delta}_i(i) = 1, \vec{\delta}_i(j) = 0 \; for \; i \neq j$

- Each likelihood requires summing over the possible z

$$p(\vec{x}|\theta) = \sum_z p(\vec{z}|\theta)p(\vec{x}|\vec{z},\theta) = \sum_{i=1}^{K} p\left(\vec{z} = \vec{\delta}_i \middle| \theta\right) p\left(\vec{x} \middle| \vec{z} = \vec{\delta}_i, \theta\right)$$

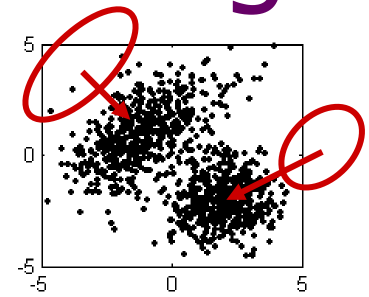
mixing proportions (prior) =$\boldsymbol{\pi} = p\left(\vec{z} = \vec{\delta}_i \middle| \pi\right)$

mixture components =$p\left(\vec{x} \middle| \vec{z} = \vec{\delta}_i, \theta\right)$

posteriors (responsibilities) =$\tau_{n,i} =$

log likelihood =

Hidden / Unlabeled = Clustering



- Represent each hidden y integer (1 to K) with a hidden binary indicator vector z
  $\vec{z} \in \boldsymbol{B}^K, \sum_{i=1}^{K} \vec{z}(i) = 1$ or

$\vec{z} \in \{\vec{\delta}_1, \dots, \vec{\delta}_K\}$ where $\vec{\delta}_i(i) = 1, \vec{\delta}_i(j) = 0 \; for \; i \neq j$

- Each likelihood requires summing over the possible z

$$p(\vec{x}|\theta) = \sum_z p(\vec{z}|\theta) p(\vec{x}|\vec{z},\theta) = \sum_{i=1}^{K} p\left(\vec{z} = \vec{\delta}_i \middle| \theta\right) p\left(\vec{x} \middle| \vec{z} = \vec{\delta}_i, \theta\right)$$

mixing proportions (prior) = $\boldsymbol{\pi} = p\left(\vec{z} = \vec{\delta}_i \middle| \pi\right)$

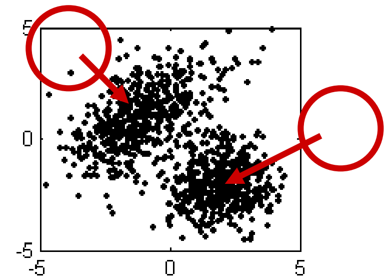
mixture components = $p\left(\vec{x} \middle| \vec{z} = \vec{\delta}_i, \theta\right)$

posteriors (responsibilities) = $\tau_{n,i} = p\left(\vec{z} = \vec{\delta}_i \middle| \vec{x}_n, \theta\right) = \frac{p\left(\vec{x}_n \middle| \vec{z} = \vec{\delta}_i, \theta\right) p\left(\vec{z} = \vec{\delta}_i \middle| \theta\right)}{p(\vec{x}_n|\theta)}$

log likelihood = $\sum_{n=1}^{N} \log p(\vec{x}_n|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \log \sum_{i=1}^{K} \pi_i p(\vec{x}_n|\vec{\mu}_i, \Sigma_i)$

- Can't easily take derivatives of log-likelihood and set to 0.
- Not nice, seems to need gradient ascent…
- Or, can we break up mixture into smaller Gaussian steps?

K-Means Clustering



- An old "heuristic" clustering algorithm
- Gobble up data with a divide & conquer scheme
- Assume each point x has an discrete multinomial vector z
- Chicken and Egg problem:

  If know classes, we can get model (max likelihood!)

  If know the model, we can predict the classes (classifier!)

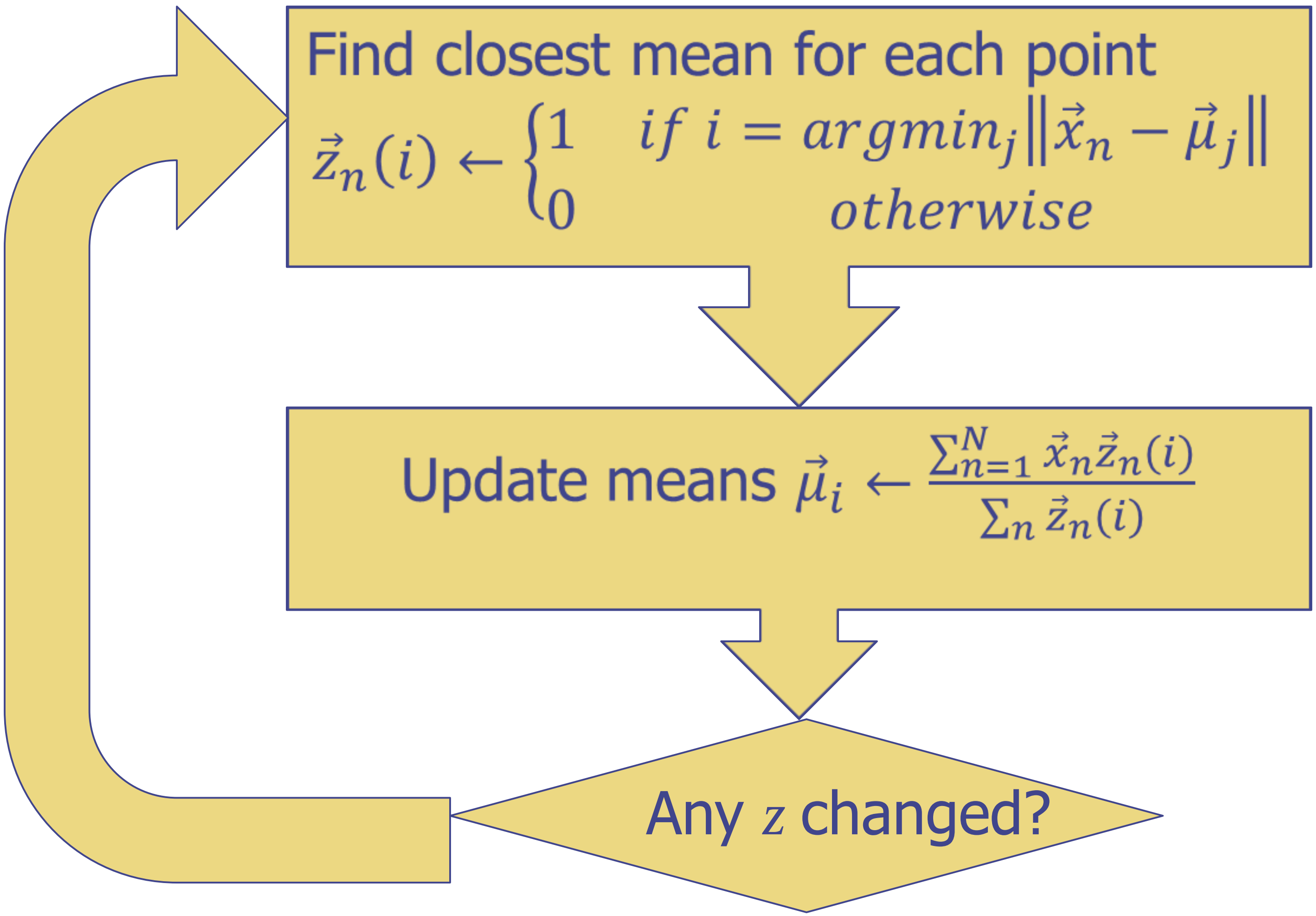
The K-Means Algorithm

Find closest mean for each point

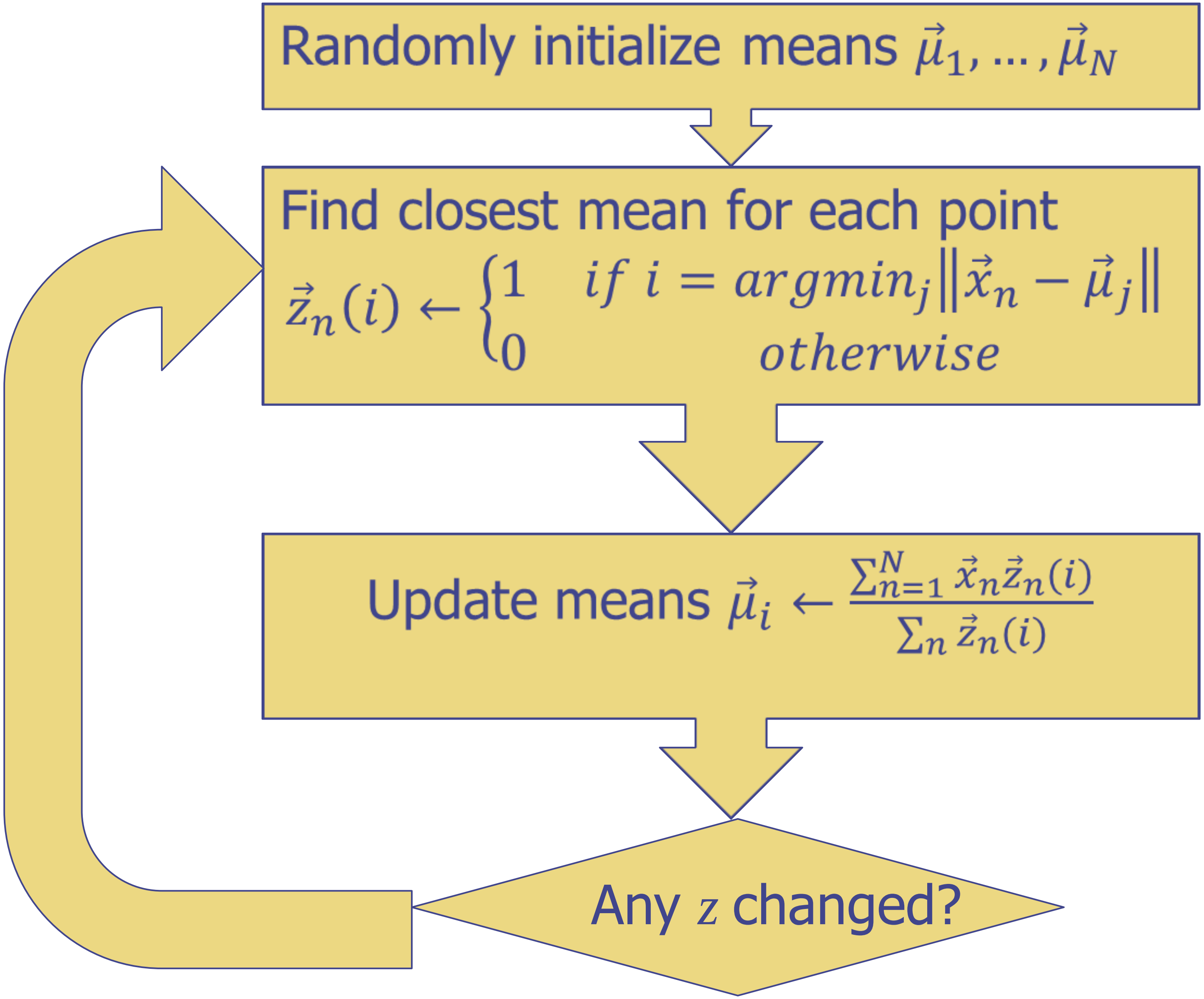
$$\vec{z}_n(i) \leftarrow \begin{cases} 1 & if\ i = argmin_j \left\|\vec{x}_n - \vec{\mu}_j\right\| \\ 0 & otherwise \end{cases}$$

Update means $\vec{\mu}_i \leftarrow \frac{\sum_{n=1}^{N} \vec{x}_n \vec{z}_n(i)}{\sum_n \vec{z}_n(i)}$

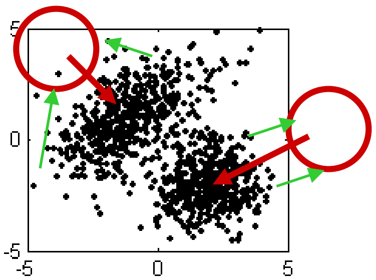
The K-Means Algorithm



The K-Means Algorithm



K-Means Clustering

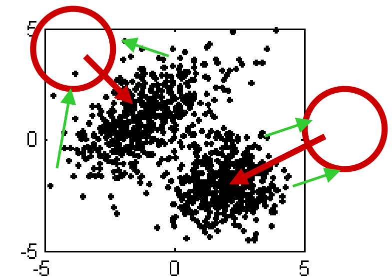


- Geometric, each point goes to closest Gaussian
- Recompute the means by their assigned points
- Minimizing $\min_{\mu} \min_{z} J(\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{z}_1, \ldots, \vec{z}_N)$ cost function:

$$J(\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{z}_1, \ldots, \vec{z}_N) = \sum_{n=1}^{N} \sum_{i=1}^{K} \vec{z}_n(i) \|\vec{x}_n - \vec{\mu}_i\|^2$$

$\vec{z}_n(i) = \begin{cases} 1 & if\ i = argmin_j \left\|\vec{x}_n - \vec{\mu}_j\right\| \\ 0 & otherwise \end{cases}$ and $\vec{\mu}_i = \frac{\sum_{n=1}^{N} \vec{x}_n \vec{z}_n(i)}{\sum_n \vec{z}_n(i)}$

K-Means Clustering

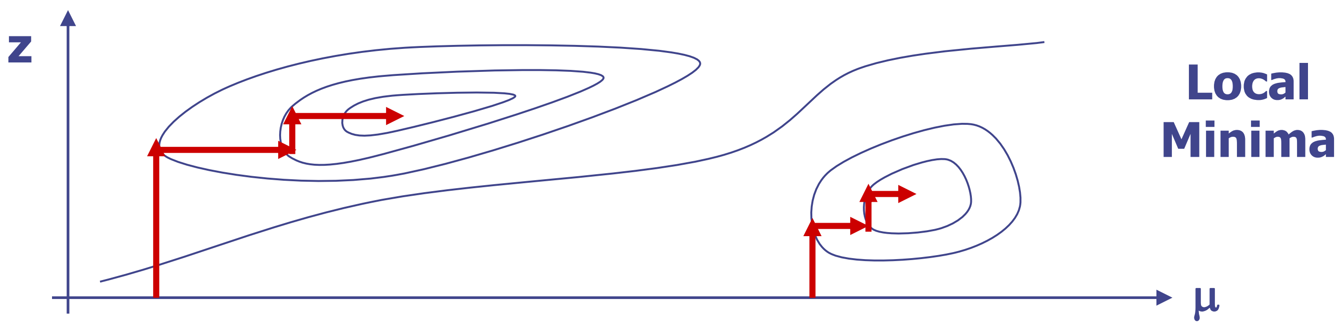


- Geometric, each point goes to closest Gaussian
- Recompute the means by their assigned points
- Minimizing $\min_{\mu} \min_{z} J(\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{z}_1, \ldots, \vec{z}_N)$ cost function:

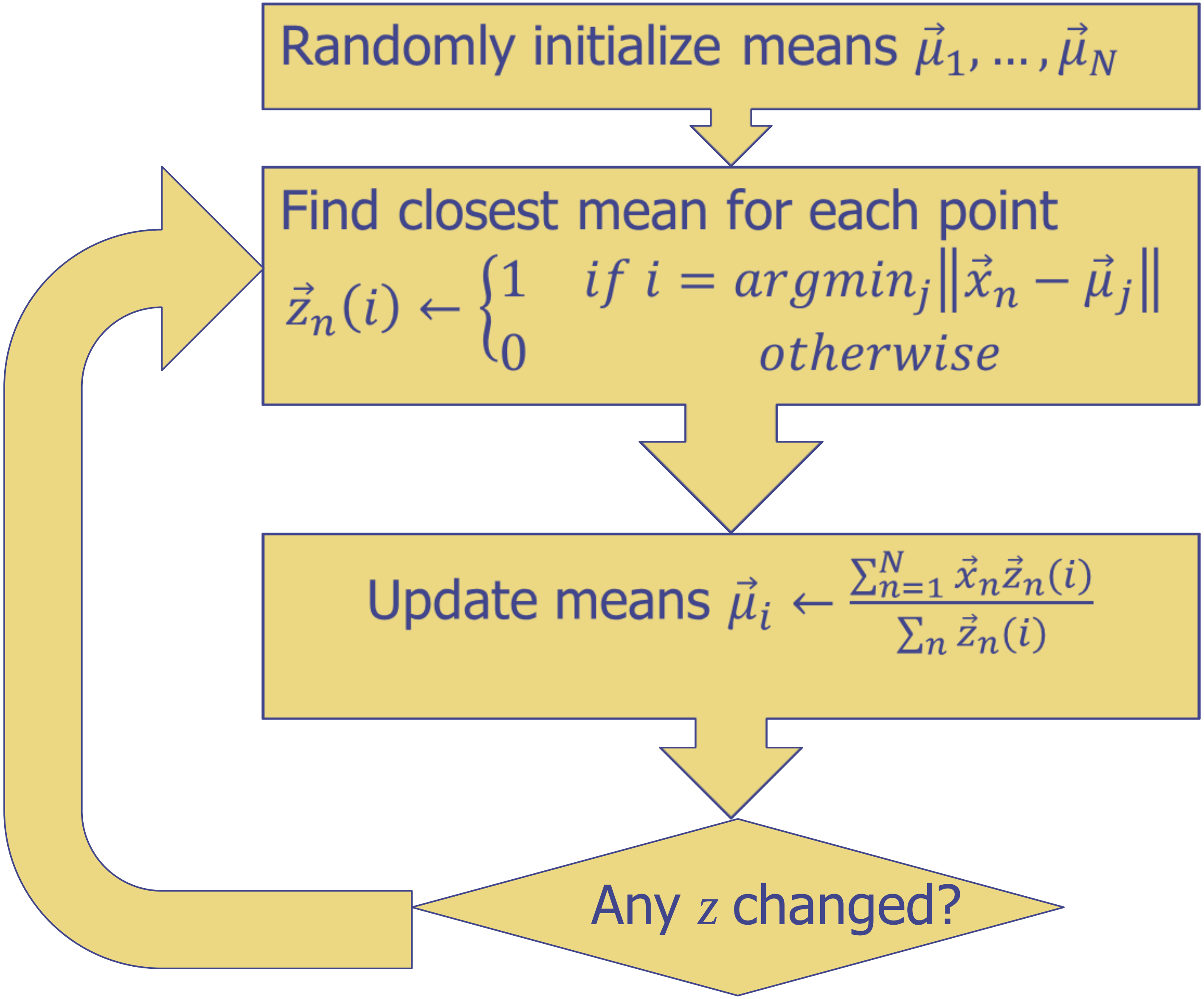
$$J(\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{z}_1, \ldots, \vec{z}_N) = \sum_{n=1}^{N} \sum_{i=1}^{K} \vec{z}_n(i) \|\vec{x}_n - \vec{\mu}_i\|^2$$

$$\vec{z}_n(i) = \begin{cases} 1 & if\ i = argmin_j \|\vec{x}_n - \vec{\mu}_j\| \\ 0 & otherwise \end{cases} \quad \text{and } \vec{\mu}_i = \frac{\sum_{n=1}^{N} \vec{x}_n \vec{z}_n(i)}{\sum_n \vec{z}_n(i)}$$

- Guaranteed to improve per iteration and converge
- Like Coordinate Descent (lock one var, maximize the other)
- A.k.a. Axis-Parallel Optimization or Alternating Minimization



The K-Means Algorithm



The Gaussian K-Means Algorithm

Randomly initialize all $\vec{\mu}_i$; set $\Sigma_i \leftarrow I$

Find ML classification for each point

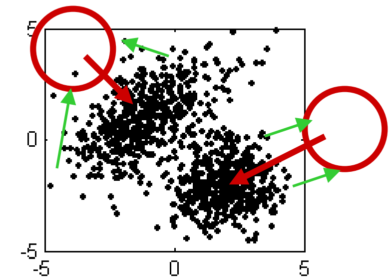
$$\vec{z}_n(i) \leftarrow \begin{cases} 1 & i = argmax_j N(\vec{x}_n|\vec{\mu}_j, \Sigma_j) \\ 0 & otherwise \end{cases}$$

Update $\vec{\mu}_i \leftarrow \sum_{n=1}^{N} \vec{x}_n \vec{z}_n(i) \,/\, \sum_n \vec{z}_n(i)$

$$\vec{\Sigma}_i \leftarrow \frac{\sum_{n=1}^{N} (\vec{x}_n - \vec{\mu}_i)^T (\vec{x}_n - \vec{\mu}_i) \vec{z}_n(i)}{\sum_n \vec{z}_n(i)}$$

Any z changed?

K-Means Clustering



- Geometric, each point goes to closest Gaussian
- Recompute the means by their assigned points
- Minimizing $\min_{\mu,\Sigma} \min_{z} J(\vec{\mu}_1, \dots, \vec{\mu}_K, \Sigma_1, \dots, \Sigma_K, \vec{z}_1, \dots, \vec{z}_N)$cost:

$$J(\vec{\mu}_1, \dots, \vec{\mu}_K, \vec{z}_1, \dots, \vec{z}_N) = \sum_{n=1}^{N} \sum_{i=1}^{K} \vec{z}_n(i)(\vec{x}_n - \vec{\mu}_i)^T \Sigma^{-1} (\vec{x}_n - \vec{\mu}_i)$$

$$\vec{z}_n(i) = \begin{cases} 1 & if\ i = argmin_j (\vec{x}_n - \vec{\mu}_i)^T \Sigma^{-1} (\vec{x}_n - \vec{\mu}_i) \\ 0 & otherwise \end{cases}$$

- Guaranteed to improve per iteration and converge
- Like Coordinate Descent (lock one var, maximize the other)
- A.k.a. Axis-Parallel Optimization or Alternating Minimization