CSCE 633: Machine Learning

Lecture 23: Unsupervised Learning: Principal Component Analysis (PCA) and K-Means

Texas A&M University

10-16-19

Goals of this lecture

- PCA
- Clustering, K-Means

Dimensionality Reduction

Broad question

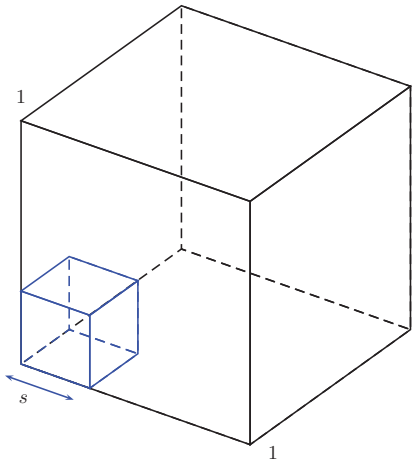
- How can we detect low dimensional structure in high dimensional data?

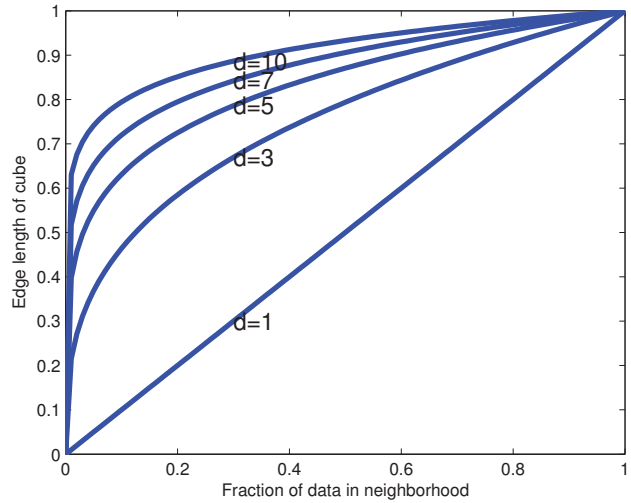
Motivations

- Exploratory data analysis & visualization: you can plot data now
- Compact representation: small memory/computational footprint, lossy data compression
- Robust statistical modeling: curse of dimensionality

What is curse of dimensionality

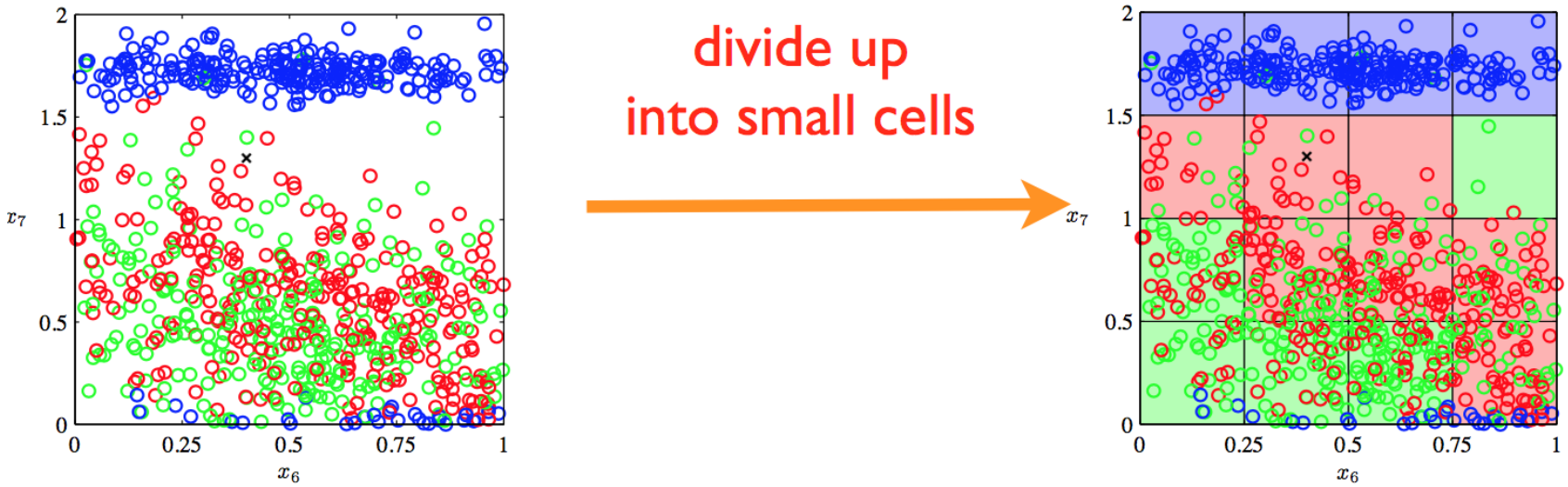
- In a high-dimensional space:
  - all intuition fails in higher dimensions
  - harder to generalize
  - harder to systematically search
  - harder to accurately approximate a target function
- On the positive side
  - "blessing of non-uniformity": examples aren't uniformly spread





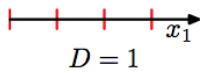
What is curse of dimensionality

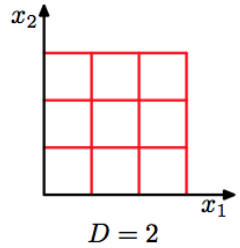
Example: a simple classification scheme (related to decision tree)

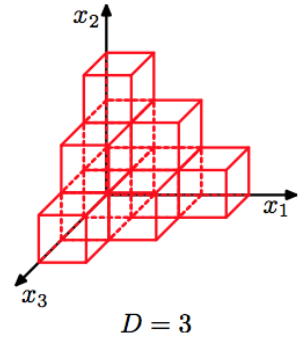


What is curse of dimensionality

Number of cells grows exponentially as dimensionality increases







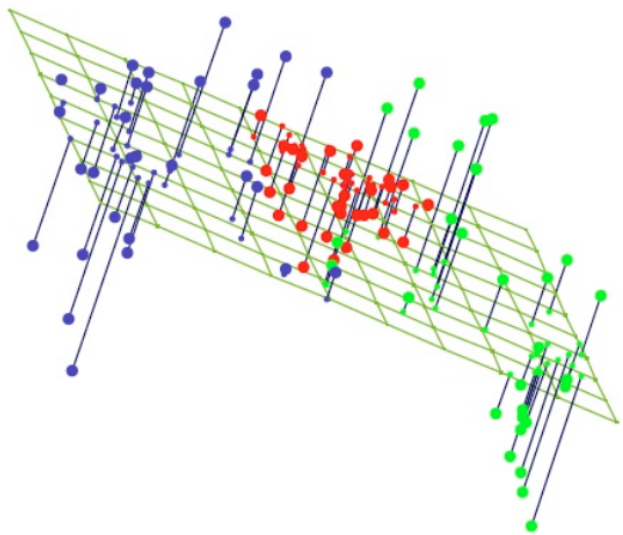
# of cells r^D

r: number of divisions in each dimension

- Large number of cells, even if D is moderately large
- So to cover the whole space reasonably well, you need exponentially number of training data points

Linear dimensionality reduction

- $\mathbf{x} \in \Re^D \rightarrow \mathbf{y} \in \Re^M$, $D \gg M$
- linear transformation of original space: $\mathbf{y} = \mathbf{U}^T\mathbf{x}$



An overview of dimensionality reduction

- Methods we have talked about so far use all predictors $X_1, \cdots, X_p$
- What if we transform them to a $Z_1, \cdots, Z_m$ that represent $M < p$ linear combinations of our original p predictors?
- That is:

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

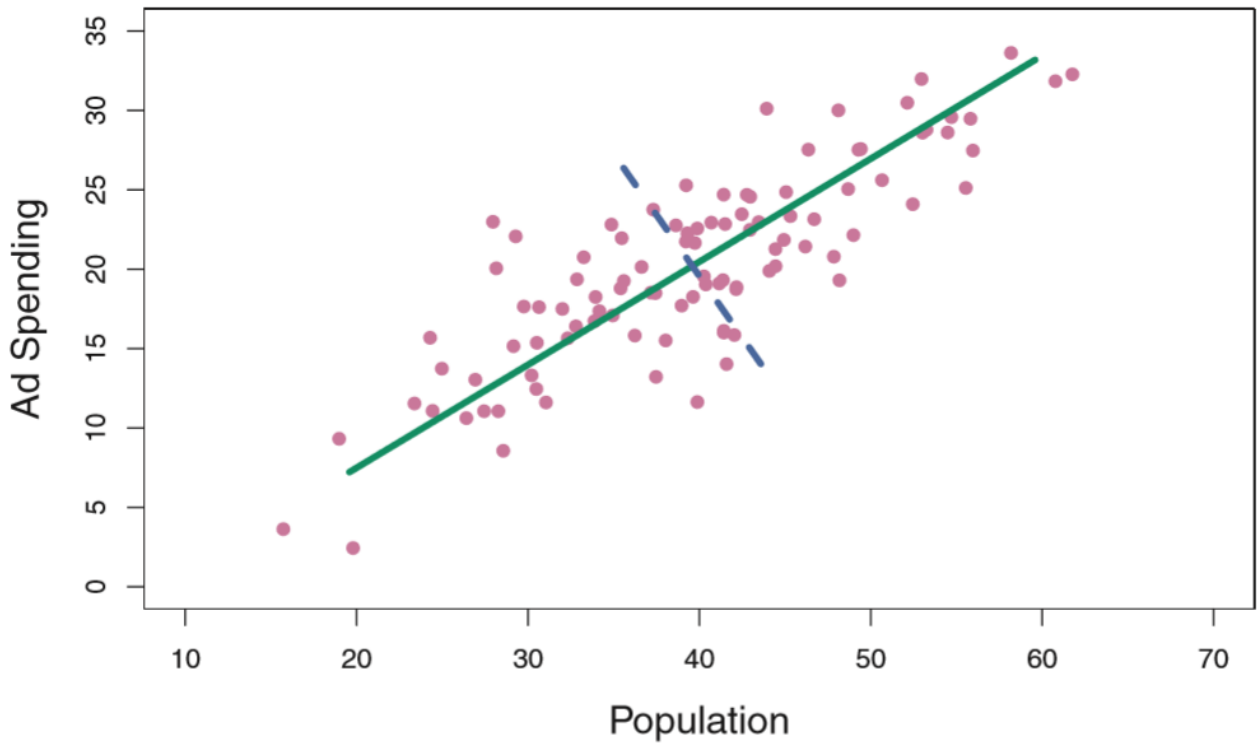
For constants $\phi_{1m}, \cdots, \phi_{pm}$ for $m = 1, \cdots, M$

We can use this in regression to fit:

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m Z_{im} + \epsilon_i$$

This results in constants $\beta_j = \sum_{m=1}^{M} \theta_m \phi_{jm}$
So dimension reduction gives same linear regression but constraints β

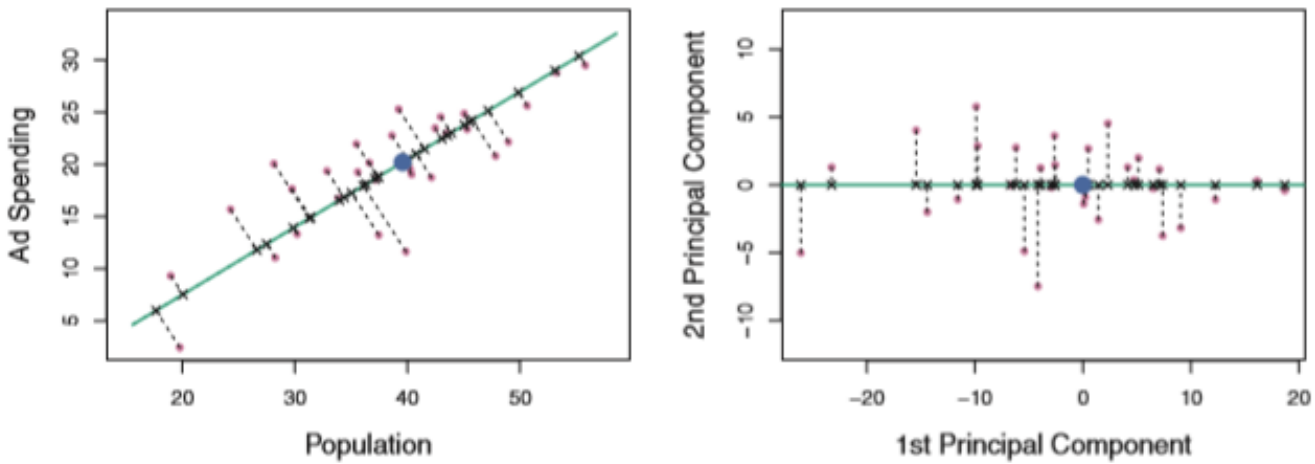
PCA



PCA: Population and Ads

- $Z_1 = 0.839 \times (pop - p\bar{o}p) + 0.544 \times (ad - \bar{ad})$
- $\phi_{11} = 0.839$, $\phi_{21} = 0.544$ are the principal component loadings
- What we want to do is maximize
  $Var(\phi_{11} \times (pop - p\bar{o}p) + \phi_{21}(ad - \bar{ad})$
- So each element i has
  $z_{i1} = 0.839 \times (pop_i - p\bar{o}p) + 0.544 \times (ad_i - \bar{ad})$, where the $z_{11}, \cdots, z_{n1}$ are the principal component scores

PCA Visuals



PCA: Challenges in Unsupervised Learning

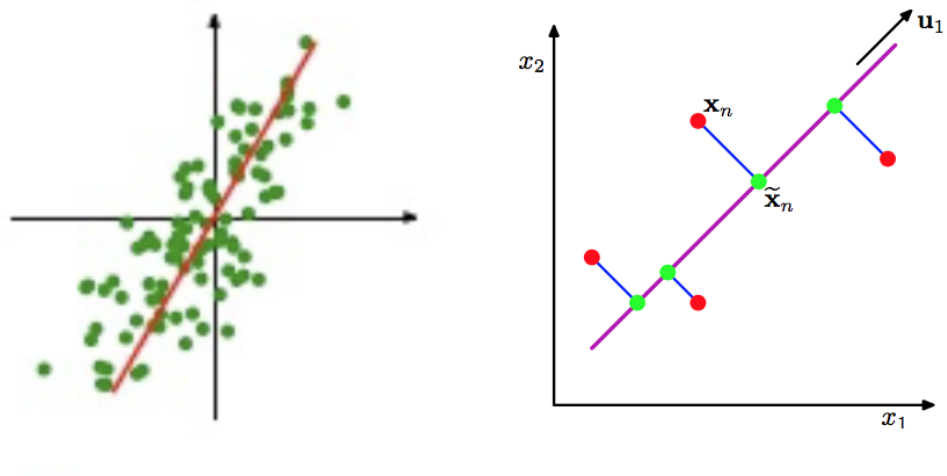
- Learning patterns in data without supervised labels
- More Subjective
- No simple goal (prediction of response)
- Exploratory data analysis means no way to check if your answer is correct
- One can't do a bunch of $2D$ scatter plots - $\binom{p}{2}$ plots

Principal Component Analysis (PCA): Representation

- Input: Data $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$, $\mathbf{x_n} \in \Re^D$,centered inputs
- Output: Projected data $\{\mathbf{y_1}, \ldots, \mathbf{y_N}\}$, $\mathbf{y_n} \in \Re^M$, $D \gg M$
- Projection into subspace: $\mathbf{U} \in \Re^{D \times M}$

$$\mathbf{y_n} = \mathbf{U}^T \mathbf{x_n} \ , \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}$$

- Evaluation metric: many possible metrics yielding the same solution
  - Derivation 1: Maximize captured variance
  - Derivation 2: Minimize projection error



Matrix Diagonalization

- Converting a square matrix into a special type of matrix, i.e. diagonal, which shares the same fundamental properties of the underlying matrix
- A square matrix $\mathbf{A} \in \Re^{D \times D}$ can be decomposed into

$$\mathbf{A} = \mathbf{P} \boldsymbol{\Lambda} \mathbf{P}^{-1}$$

  - $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_D)$: diagonal constructed from eigenvalues of $\mathbf{A}$
  - $\mathbf{P} = \begin{bmatrix} | & & | \\ \mathbf{e_1} & \ldots & \mathbf{e_D} \\ | & & | \end{bmatrix} \in \Re^{M \times M}$: matrix decomposed from the eigenvectors of $\mathbf{A}$

Principal Component Analysis (PCA): Optimization

- Compute covariance matrix

$$\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X} = \frac{1}{N}\sum_n \mathbf{x_n}\mathbf{x_n}^T \;, \quad \mathbf{X} = \begin{bmatrix} - \tilde{\mathbf{x}}_1^T - \\ \vdots \\ - \tilde{\mathbf{x}}_N^T - \end{bmatrix}$$

$$\mathbf{U} \in \Re^{D \times M}$$

- Diagonalize S, i.e. compute eigenvalues and eigenvectors

$$\mathbf{S} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{-1} \;, \quad \mathbf{P} = \begin{bmatrix} | & & | \\ \mathbf{u_1} & \dots & \mathbf{u_D} \\ | & & | \end{bmatrix} \in \Re^{D \times D}$$

- Use the eigenvectors corresponding to the M largest eigenvalues

$$\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u_1} & \dots & \mathbf{u_M} \\ | & & | \end{bmatrix} \in \Re^{D \times M}$$

Principal Component Analysis (PCA): Optimization

- Compute covariance matrix

$$\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X}\,, \quad \mathbf{X} = \begin{bmatrix} - \tilde{\mathbf{x}}_1^T - \\ \vdots \\ - \tilde{\mathbf{x}}_N^T - \end{bmatrix}$$

$$\mathbf{U} \in \Re^{D \times M}$$

- Diagonalize S, i.e. compute eigenvalues and eigenvectors

$$\mathbf{S} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{-1}\,, \quad \mathbf{P} = \begin{bmatrix} | & & | \\ \mathbf{u_1} & \dots & \mathbf{u_D} \\ | & & | \end{bmatrix} \in \Re^{D \times D}$$

- Use the eigenvectors corresponding to the M largest eigenvalues

$$\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u_1} & \dots & \mathbf{u_M} \\ | & & | \end{bmatrix} \in \Re^{D \times M}$$

Principal Component Analysis (PCA): Algorithm

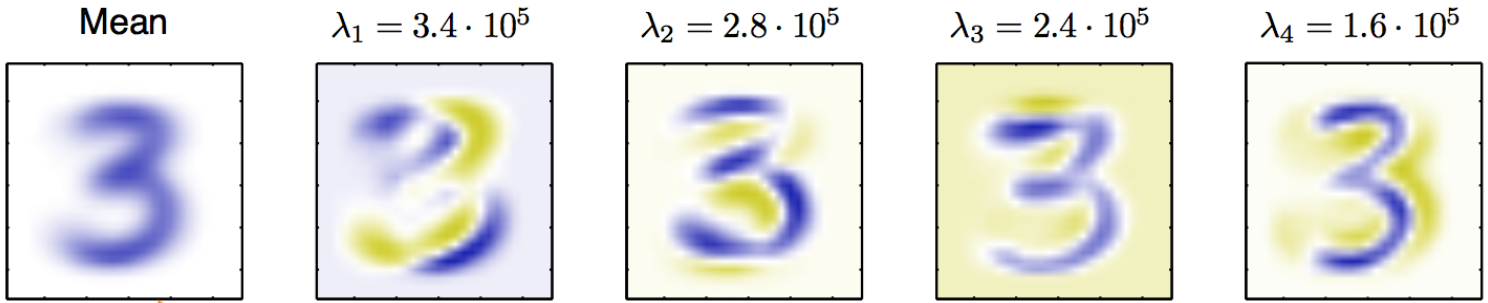
- Step 0: Mean normalize input features
- Step 1: Compute covariance matrix $\mathbf{S} = \frac{1}{N}\mathbf{X}^T\mathbf{X} = \frac{1}{N}\sum_n \mathbf{x_n}\mathbf{x_n}^T$
- Step 2: Diagonalize $\mathbf{S}$ and find eigenvector matrix $\mathbf{P}$
- Step 3: Take the first $M \ll D$ eigenvectors or *principal components* (corresponding to the M largest eigenvalues) and form reduced matrix $\mathbf{U}$
- Step 3: Project data into reduced space: $\mathbf{z_n} = \mathbf{U}^T\mathbf{x_n}$

Principal Component Analysis (PCA)

Original Images

Eigenvectors

they look like blurred original images



Used to centralize inputs

Principal Component Analysis (PCA)

Applications of PCA

Preprocessing

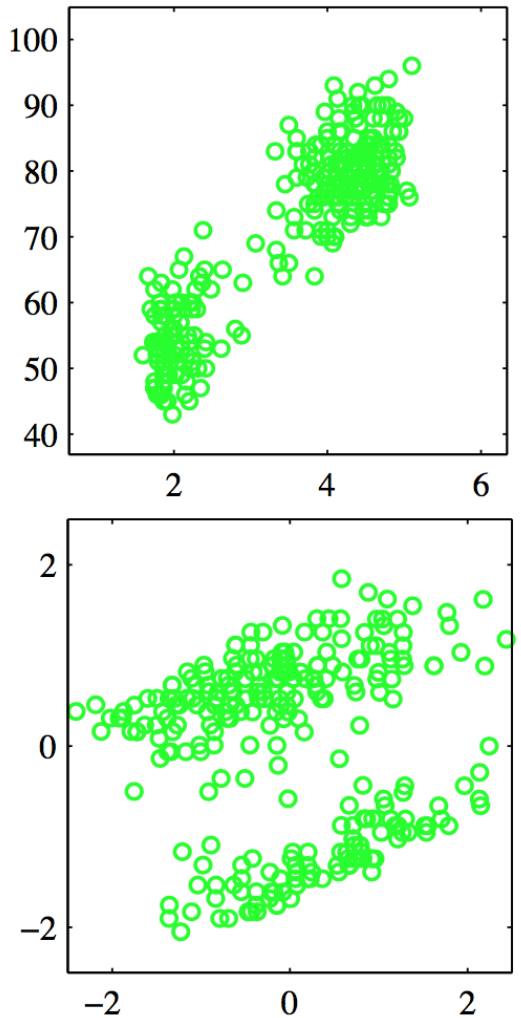
Diagonalize data

$$\boldsymbol{y}_i = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

Normalize data (whitening)

$$\boldsymbol{y}_i = \boldsymbol{\lambda}^{-1/2} \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}_i$$

Benefits:
1) depress noisy features
2) couple with other models



Principal Component Analysis (PCA)

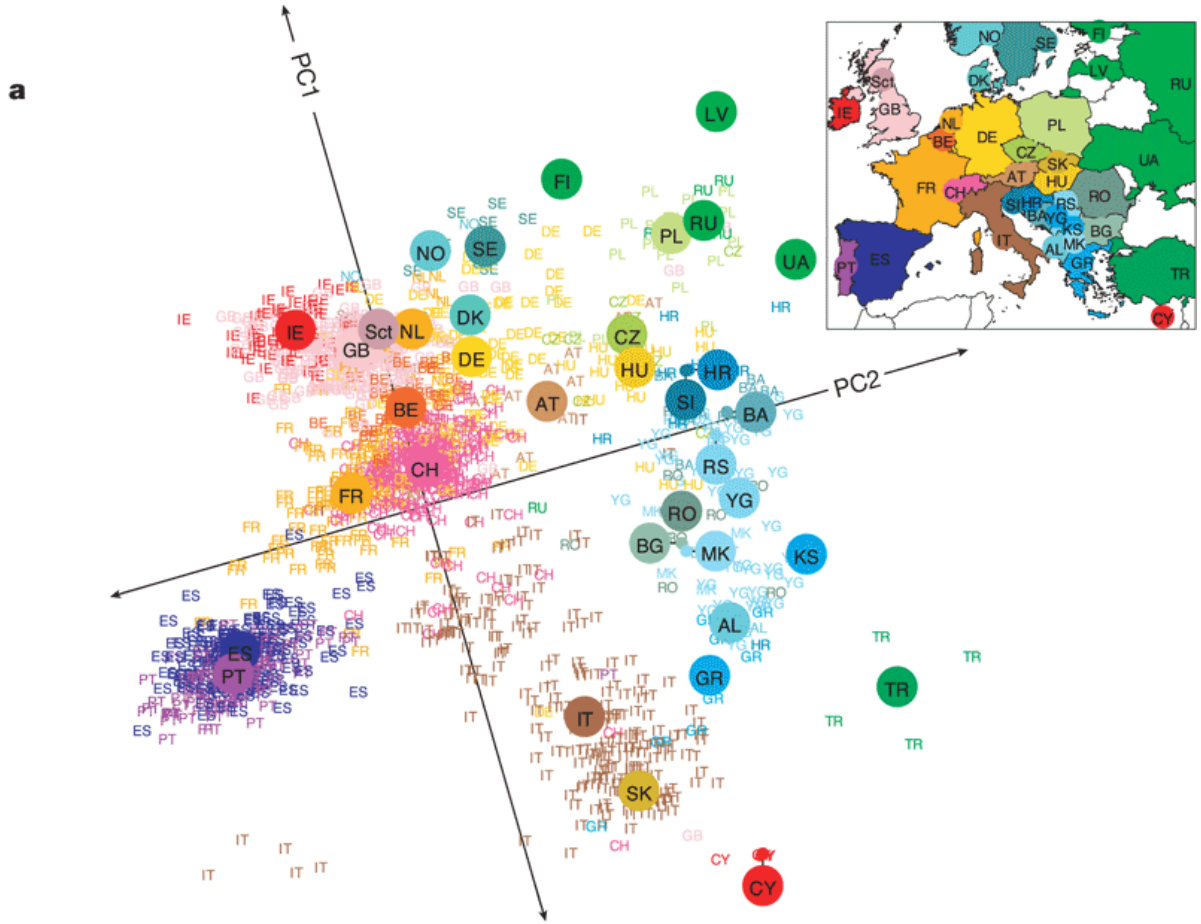
How to determined number of principal components M?

Plot eigenspectrum

$$\frac{\sum_{d=1}^{M} \lambda_d}{\sum_{d=1}^{D} \lambda_d} \geq \text{threshold, where common choices are 95\%, 99\%}$$

Clustering

Finding patterns/structure/sub-populations in data



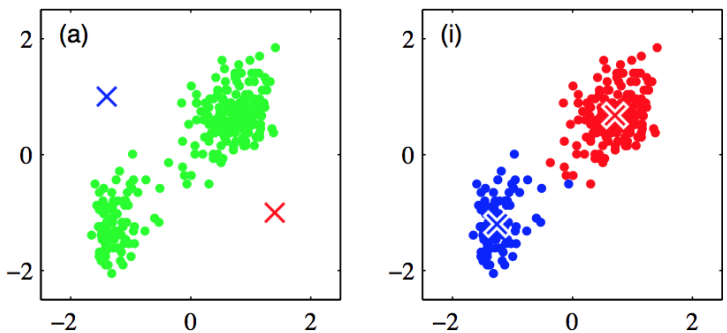
Clustering

- find patterns/structure/sub-populations in data ("knowledge discovery")
- training data does not include desired outputs
- less well-defined problem with no obvious error metrics
- topic modeling, market segmentation, clustering of hand-written digits, news clustering (e.g. Google news)

K-means Clustering

Representation Input: Data $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$

Output: Clusters $\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_K}$
Decision: Cluster membership, the cluster id assigned to sample $\mathbf{x_n}$, i.e. $A(\mathbf{x_n}) \in \{1, \ldots, K\}$
Evaluation metric: Distortion measure
$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x_n} - \boldsymbol{\mu_k}\|_2^2$, where $r_{nk} = 1$ if $A(\mathbf{x_n}) = k$, 0 otherwise
Intuition: Data points assigned to cluster k should be close to centroid $\boldsymbol{\mu_k}$



K-means Clustering

Evaluation Metric:

$$\min_{r_{nk}} J = \min_{r_{nk}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x_n} - \boldsymbol{\mu_k}\|_2^2$$

Optimization:

- Step 0: Initialize $\boldsymbol{\mu_k}$ to some values
- Step 1: Assume the current value of $\boldsymbol{\mu_k}$ fixed, minimize J over r_{nk}, which leads to the following cluster assignment rule
  $$r_{nk} = \left\{ \begin{array}{ll} 1, & \text{if } k = arg\min_j \|\mathbf{x_n} - \boldsymbol{\mu_j}\|_2^2 \\ 0, & \text{otherwise} \end{array} \right\}$$
- Step 2: Assume the current value of r_{nk} fixed, minimize J over $\boldsymbol{\mu_k}$, which leads to the following rule to update the prototypes of the clusters $\boldsymbol{\mu_k} = \frac{\sum_n r_{nk}\mathbf{x_n}}{\sum_n r_{nk}}$
- Step 3: Determine whether to stop or return to Step 1

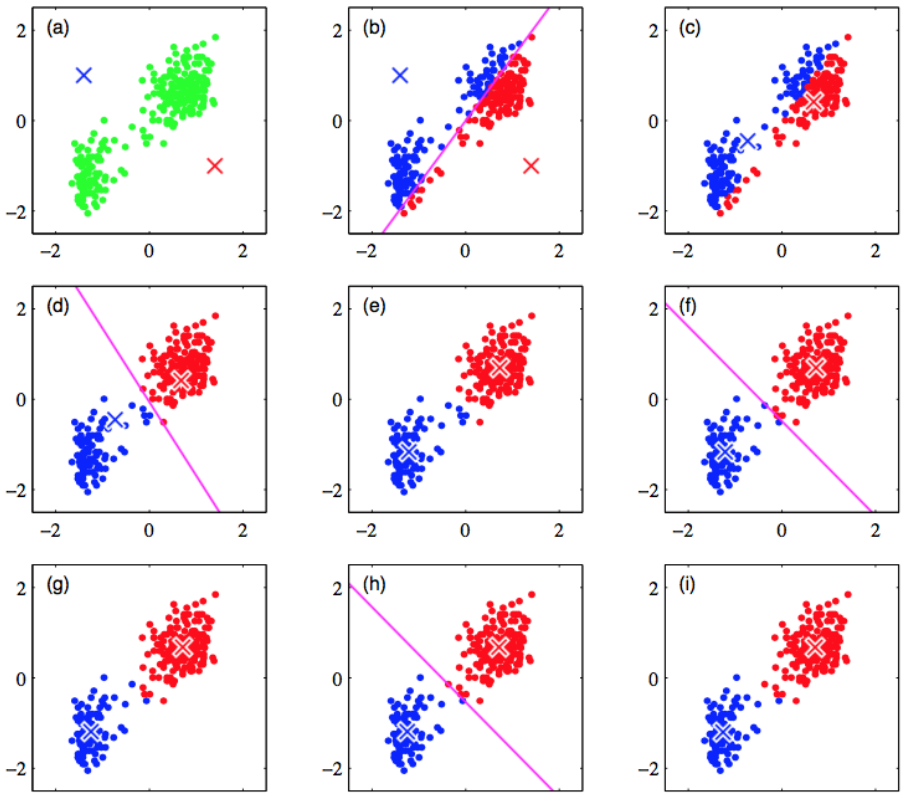
K-means Clustering

Remarks

- The centroid $\boldsymbol{\mu_k}$ is the means of data points assigned to the cluster k, hence the name K-means clustering.
- The procedure terminates after a finite number of steps, as the procedure reduces J in both Step 1 and Step 2
- There is no guarantee the procedure terminates at the global optimum of J. In most cases, the algorithm stops at a local optimum, which depends on the initial values in Step 0 $\rightarrow$ random restarts to improve chances of getting closer to global optima

K-means Clustering

Example



K-means Clustering

Application: vector quantization

- We can replace our data points with the centroids $\boldsymbol{\mu_k}$ from the clusters they are assigned to $\rightarrow$ vector quantization
- We have compressed the data points into
  - a codebook of all the centroids $\{\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_K}\}$
  - a list of indices to the codebook for the data points (created based on r_{nk})
- This compression is obviously lossy as certain information will be lost if we use a very small K

K-means Clustering

Question: vector quantization with K-means

Assume that the images bellow are created by vectoring the original image with K-means using different values of K. What is the correct combination?

Original Image

A) $K = 25$ $K = 10$ $K = 3$

B) $K = 3$ $K = 10$ $K = 25$

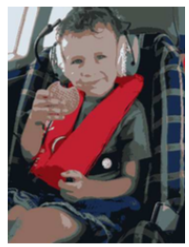
K-means Clustering

Question: vector quantization with K-means

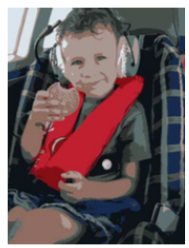
Assume that the images bellow are created by vectoring the original image with K-means using different values of K. What is the correct combination?

Original Image

A) $K = 25$ $K = 10$ $K = 3$



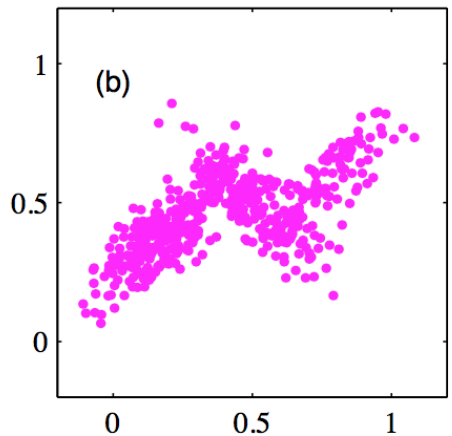
B) $K = 3$ $K = 10$ $K = 25$



Correct answer is A

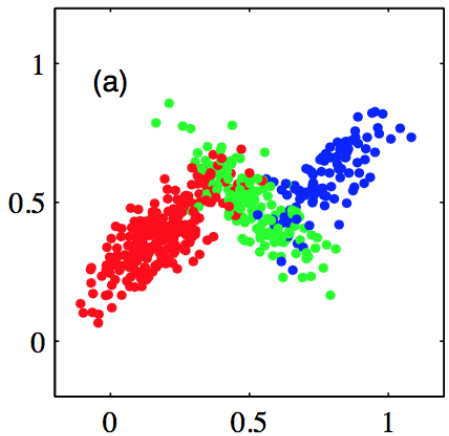
Probabilistic interpretation of clustering

- We want to find $p(\mathbf{x})$ that best describes our data
- The data points seem to form 3 clusters
- However, we cannot model $p(\mathbf{x})$ with simple and known distributions, e.g. one Gaussian



Probabilistic interpretation of clustering

- Instead, we will model each region with a Gaussian distribution $\rightarrow$ Gaussian mixture models (GMMs)
- Question 1: How do we know which (color) region a data point comes from?
- Question 2: What are the parameters of Gaussian distributions in each region?
- We will answer both in an unsupervised way from data $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$



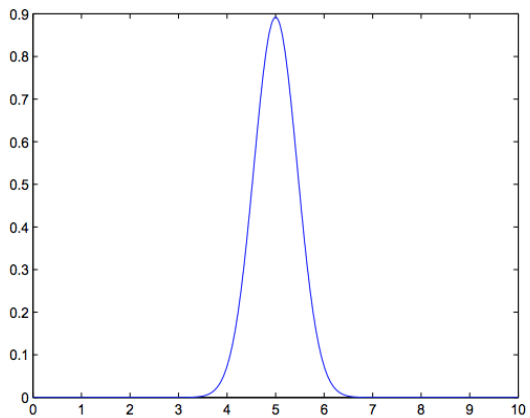
Multivariate Gaussian distribution

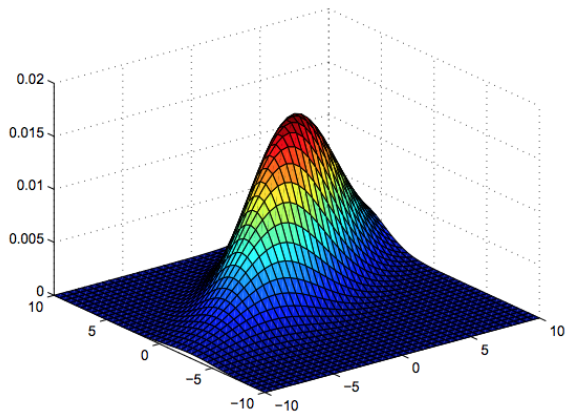
Univariate Gaussian distribution

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

Multivariate Gaussian distribution

$$p(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$





Multivariate Gaussian distribution

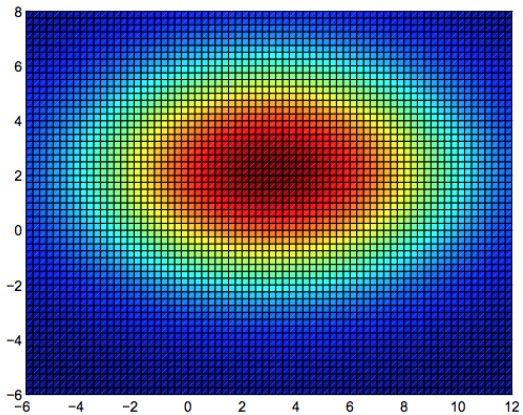
Covariance matrix

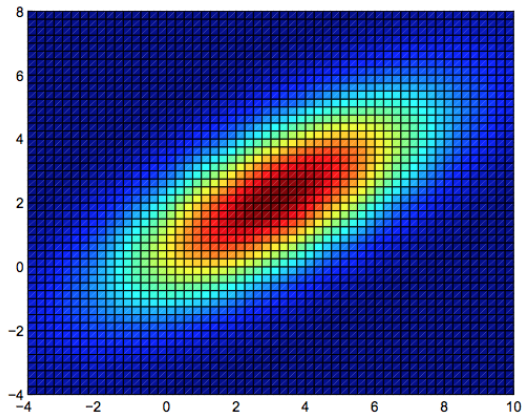
- Covariance between two random variables X and Y
  $Cov(X, Y) = \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y))) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)$
- The covariance matrix provides a way to summarize the covariances of all pairs of variables $(\boldsymbol{\Sigma})_{ij} = Cov(X_i, X_j)$
- $\boldsymbol{\Sigma}$ is always positive definite

Multivariate Gaussian distribution

Isocontours

- For a function $f : \mathbb{R}^2 \to \mathbb{R}$ an isocontour is a set of the form $\{\mathbf{x} \in \mathbb{R}^2 : f(x) = c\}$

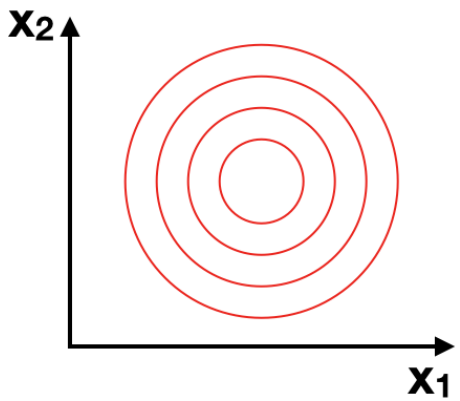


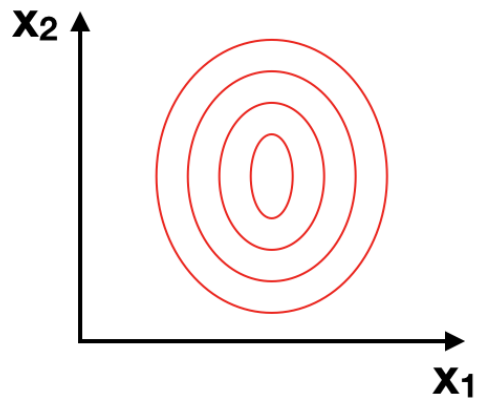


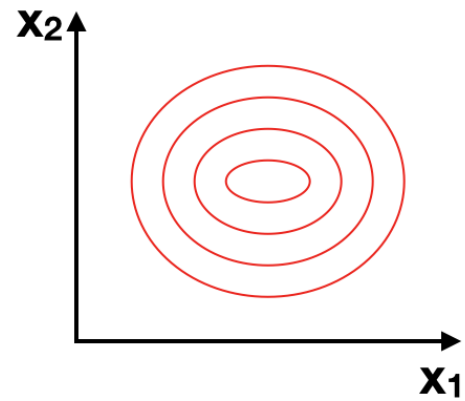
Multivariate Gaussian distribution

The diagonal covariance case

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$



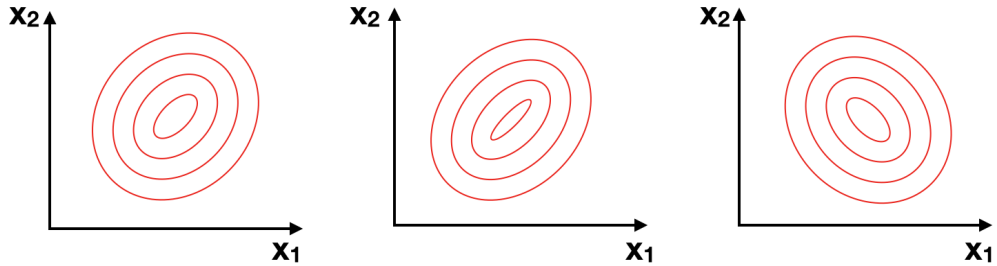




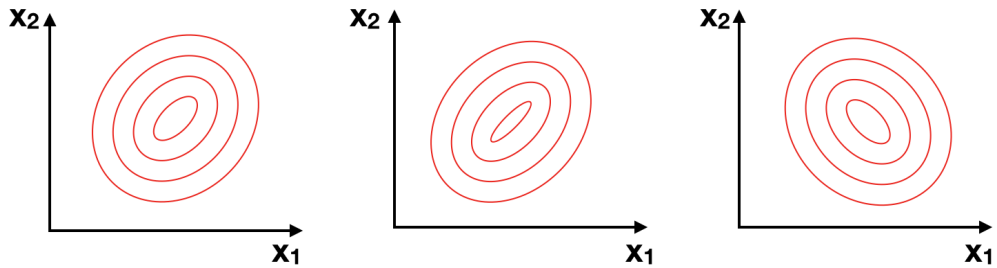
$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Multivariate Gaussian distribution

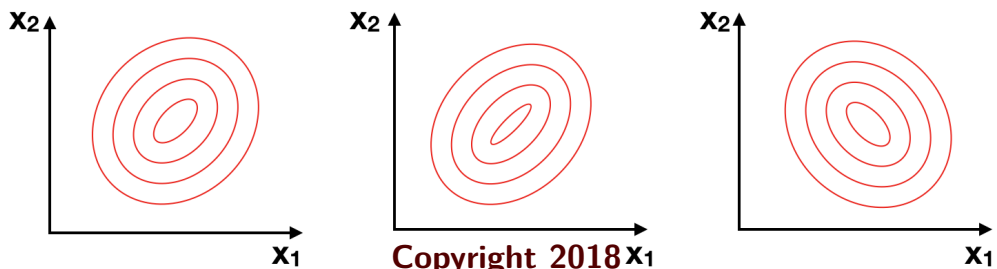
Question: Which is correct in this non-diagonal covariance case?



A) $\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$



B) $\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$

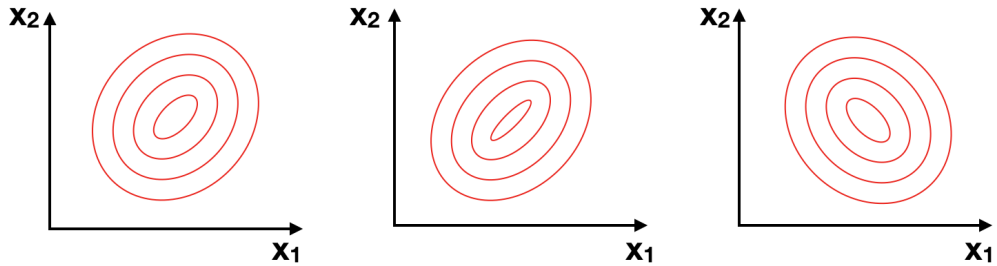


Multivariate Gaussian distribution

Question: Which is correct in this non-diagonal covariance case?

Correct answer is C

By increasing the off-diagonal elements from 0.5 to 0.8, the distribution is more and more thinly peaked along the line where x_1 is equal to x_2



c) $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$ $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\mathbf{x}$

$$p(\mathbf{x}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$$

- K: number of Gaussians
- $\boldsymbol{\mu_k}$,$\boldsymbol{\Sigma_k}$: mean & covariance of k^{th} component
- ω_k: component weights, how much component k contributes to the final distribution

$$\omega_k > 0\ ,\quad \forall k \quad \text{and} \quad \sum_{k=1}^{K} \omega_k = 1$$

ω_k can be represented by the prior distribution: $\omega_k = p(z = k)$, which decides which mixture to use

GMM as the marginal distribution of a joint distribution

- Consider the following joint distribution $p(\mathbf{x}, z) = p(z)p(\mathbf{x}|z)$
- z is a discrete random variable taking values between 1 and K, "selects" a Gaussian component
- We denote prior $\omega_k = p(z = k)$
- Assume Gaussian conditional distributions $p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$
- Then the marginal distribution of $\mathbf{x}$ is

$$p(\mathbf{x}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})$$

(which is the Gaussian mixture model)

GMM as the marginal distribution of a joint distribution

- The joint distribution between $\mathbf{x}$ and z (representing color) are

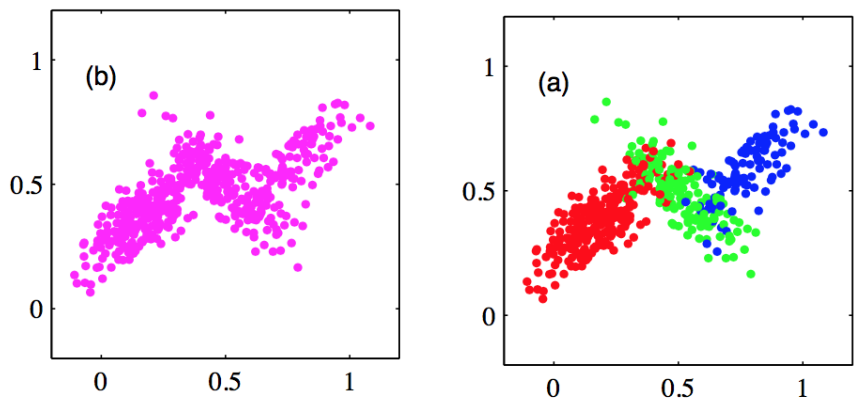
$$p(\mathbf{x}|z = red) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})$$

$$p(\mathbf{x}|z = blue) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_2}, \boldsymbol{\Sigma_2})$$

$$p(\mathbf{x}|z = green) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu_3}, \boldsymbol{\Sigma_3})$$

- The marginal distribution is thus

$$\begin{aligned} p(\mathbf{x}) =& p(red)\mathcal{N}(\mathbf{x}; \boldsymbol{\mu_1}, \boldsymbol{\Sigma_1}) + p(blue)\mathcal{N}(\mathbf{x}; \boldsymbol{\mu_2}, \boldsymbol{\Sigma_2}) \\ &+ p(green)\mathcal{N}(\mathbf{x}; \boldsymbol{\mu_3}, \boldsymbol{\Sigma_3}) \end{aligned}$$



Parameter estimation for GMMs: the easy case with complete data

We know the component in which each sample belongs to

- Data $\mathcal{D} = \{(\mathbf{x_1}, z_1), \ldots, (\mathbf{x_N}, z_N)\}$
- We want to find $\boldsymbol{\theta} = \{\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}, \omega_k\}$
- Maximum log-likelihood: $\boldsymbol{\theta}^* = arg\max \log(p(\mathcal{D})|\boldsymbol{\theta})$

Solution

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}} \qquad \boldsymbol{\mu_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \mathbf{x_n}$$

$$\boldsymbol{\Sigma_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x_n} - \boldsymbol{\mu_k})(\mathbf{x_n} - \boldsymbol{\mu_k})^T$$

where $\gamma_{nk} = 1$ if $z_n = k$

Parameter estimation for GMMs: the easy case with complete data

Understanding the intuition

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}} \qquad \boldsymbol{\mu_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \mathbf{x_n}$$

$$\boldsymbol{\Sigma_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x_n} - \boldsymbol{\mu_k})(\mathbf{x_n} - \boldsymbol{\mu_k})^T$$

- For ω_k: count the number of data points whose z_n is k and divide by the total number of data points
- For μ_k: get all the data points whose z_n is k, compute their mean
- For Σ_k: get all the data points whose z_n is k, compute their covariance

Parameter estimation for GMMs: incomplete data

Trick: estimation with soft γ_{nk}

$$\gamma_{nk} = p(z_n = k|\mathbf{x_n})$$

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}$$

$$\boldsymbol{\mu_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \mathbf{x_n}$$

$$\boldsymbol{\Sigma_k} = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\mathbf{x_n} \boldsymbol{\mu_k})(\mathbf{x_n} \boldsymbol{\mu_k})^T$$

Every data point $\mathbf{x_n}$ is assigned to a component fractionally according to $p(z_n = k|\mathbf{x_n})$, also called responsibility

Parameter estimation for GMMs: incomplete data

Trick: estimation with soft γ_{nk}

Since we do not know θ to begin with, we cannot compute the soft γ_{nk}

But we can invoke an iterative procedure and alternate between estimating γ_{nk} and using the estimated γ_{nk} to compute $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_k}$

- Step 0: guess $\boldsymbol{\theta}$ with initial values
- Step 1: compute γ_{nk} using current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using computed γ_{nk}
- Step 3: go back to Step 1

Questions: i) is this procedure correct, for example, optimizing a sensible criteria? ii) practically, will this procedure ever stop instead of iterating forever? The answer lies in the Expectation Maximization (EM) algorithm, a powerful procedure for model estimation with unknown data

Takeaways and Next Time

- PCA
- K-Means
- Next Time: More Unsupervised Learning