FIT2086 Lecture 10
Introduction to Unsupervised Learning

Daniel F. Schmidt

Faculty of Information Technology, Monash University

October 6, 2019

Outline

Revision from last week (1)

- Machine learning methods
- Cross validation for model selection
  - Withhold data to estimate prediction error
  - K-fold CV divides data up into K equal sized groups
  - Train on $K-1$ folds, predict on the remianing fold
- Decision Trees
  - Split the data up by asking questions of the predictors
  - Number of leaves determines complexity of tree
  - Easy to interpret, flexible
- Methods for learning trees
  - Greedy growing of trees – find best split at each step
  - Backwards pruning of large tree
  - Use CV to select number of leaves in the tree

Revision from last week (2)

- Trees have low bias, high variance
- One solution: random forests
  - Grow many trees with guided random search
  - Aggregate predictions from the trees
  - Stable, low variance, but loses interpretability
- k nearest neighbours (kNN) methods
  - Assume individuals similar in predictors are similar in targets
  - Find k "most similar" individuals in data to new individual
  - Use their targets to predict target for new individual
- Use CV to select k, other tuning parameters

Outline

Unsupervised learning (1)

- We have n items, each with q associated attributes, formed into a matrix

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} = \begin{pmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,q} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,1} & y_{n,2} & \cdots & y_{n,q} \end{pmatrix}$$

- Each $\mathbf{y}_i$ is a "data-point" in q-dimensional space
- Unlike supervised learning, we do not nominate any one of these as a "target"
- Instead we want to discover structure in the data

Unsupervised learning (2)

- What is unsupervised learning used for?
- Classifying or categorising objects (taxonomy)
  - For example, species of animals
- Filling in missing entries in the data matrix
  - Matrix completion problem
  - Recommender systems
  - Imputation (estimating missing data in predictor matrix before supervised learning)
- Image processing
  - Noise removal
  - Compression
  - Image analysis and recognition

Unsupervised learning vs supervised learning

- Supervised learning: target Y and explanatory variables $X_1, \ldots, X_p$
  - We then try and find the conditional distribution

$$p(Y \mid X_1, \ldots, X_p)$$

    using a specific form of model (linear regression, tree, etc.)
  - Model describes relationship between Y and $X_1, \ldots, X_p$

- Unsupervised learning: only have explanatory variables $X_1, \ldots, X_q$
  - We try and discover the joint distribution

$$p(X_1, \ldots, X_q)$$

    using a specific form of model
  - The details of the model reveal internal structure of data

Clustering

- Assumptions
  - Population consists of K sub-populations $(K > 1)$
  - We are given observations from the pooled population only
    - No sub-population information is available
- Aim
  - Discover the number of sub-populations K
  - Estimate models for each of the sub-populations
- Sometimes called intrinsic classification
  $\Rightarrow$ Class labels are learned from the data

K-means Clustering (1)

- Perhaps most commonly used clustering technique
- Models data as having K "centroids" defined by mean vectors

$$\mathbf{M} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_K \end{pmatrix} = \begin{pmatrix} \mu_{1,1} & \cdots & \mu_{1,q} \\ \vdots & \ddots & \vdots \\ \mu_{K,1} & \cdots & \mu_{K,q} \end{pmatrix}$$

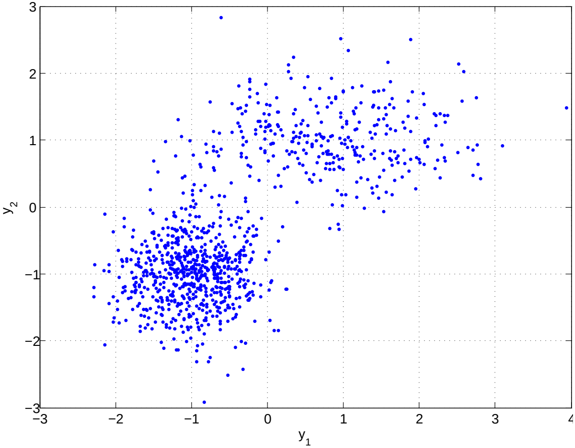
- Assigns items to class with most similar mean vector
- Similarity between item i and centroid k is

$$d_k(i) = \left(\sum_{j=1}^{q} (y_{i,j} - \mu_{k,j})^2 \right)^{\frac{1}{2}}$$

$\Rightarrow$ Euclidean distance between the vectors.

K-means Clustering (2)

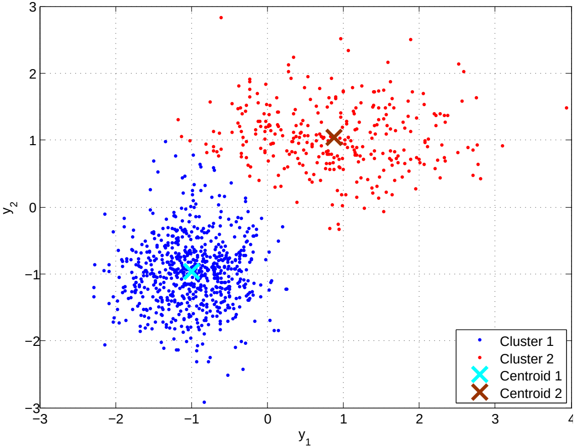
- Artificial data example



- Chosen so that the “clusters” are obvious for demonstration purposes

K-means Clustering (3)

- K-means clustering with $K = 2$



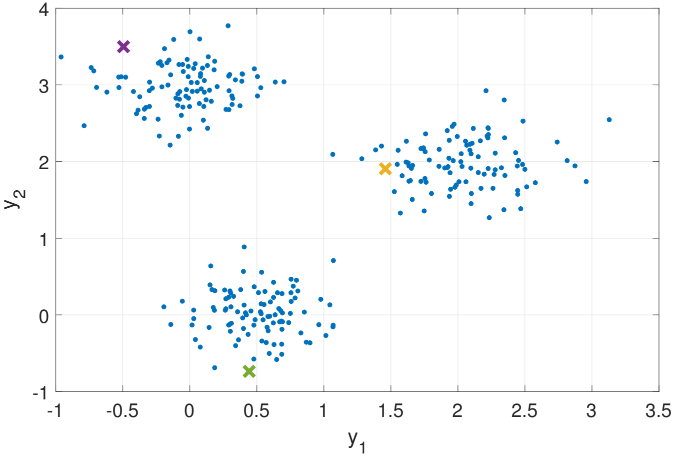
- Centroids chosen to minimise the within-cluster sum-of-squares

K-means Algorithm (1)

- The k-means algorithm is very simple:
  1. Initialise $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ randomly
  2. Loop until convergence
     1. Compute distances $d_k(i)$ from each data point $\mathbf{y}_i$ to each centroid $\boldsymbol{\mu}_k$
     2. Assign datapoints to cluster with closest centroid
     3. Re-estimate each $\boldsymbol{\mu}_k$ using the datapoints assigned to cluster k
- Converges quickly to a stable solution
  $\Rightarrow$ might not be the global-minima
- Sensitive to starting points

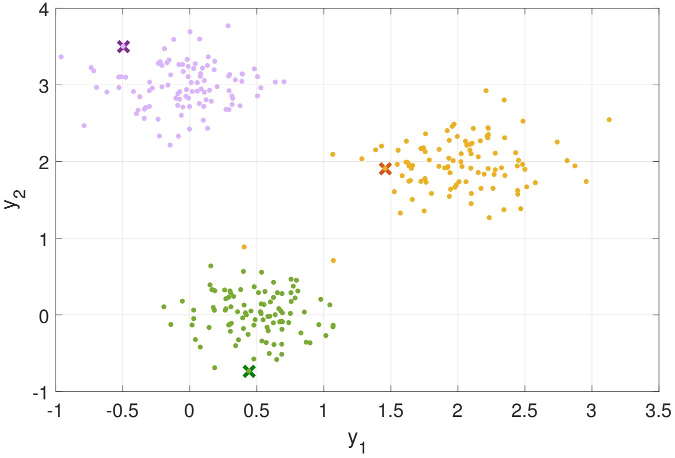
K-means Algorithm (2)

- Example: $K = 3$, initial starting points for centroids $\boldsymbol{\mu}_k$



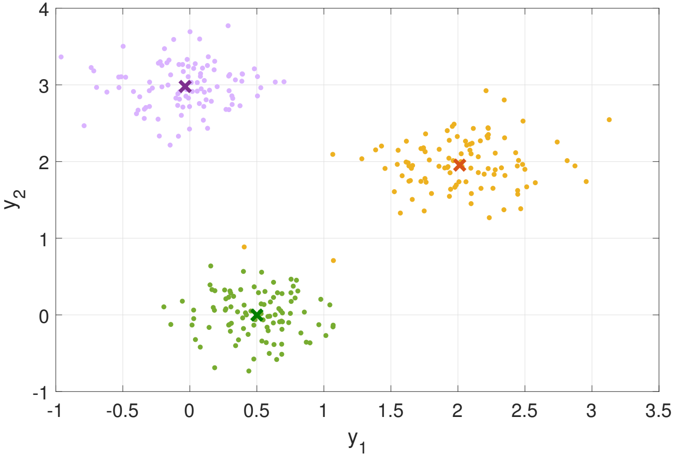
K-means Algorithm (3)

- Example: assigning points to clusters with closest centroid



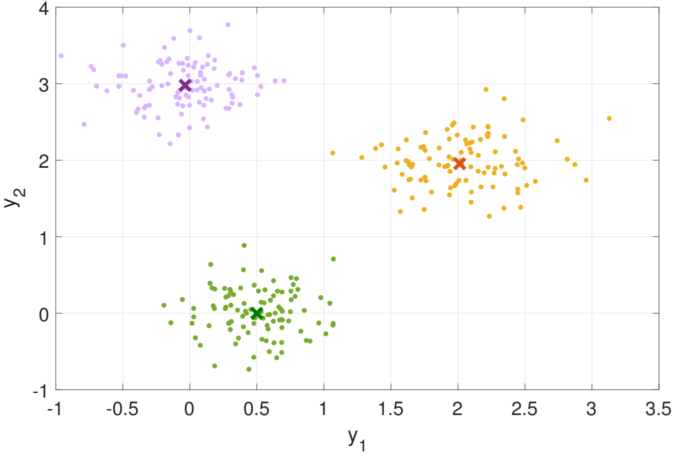
K-means Algorithm (4)

- Example: re-estimating centroids from data in the clusters



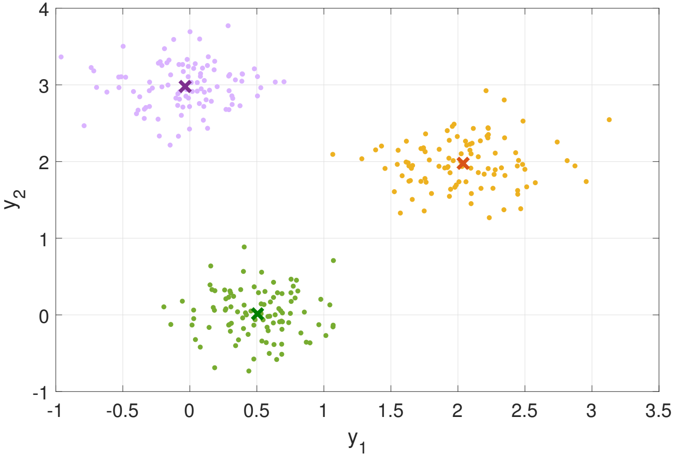
K-means Algorithm (5)

- Example: assigning points to clusters with closest centroid



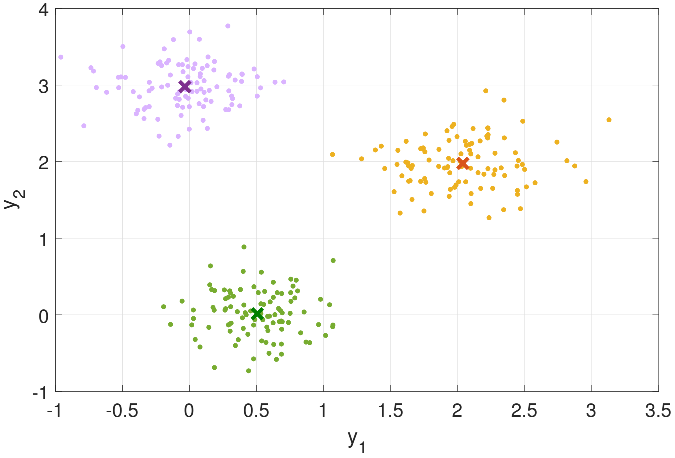
K-means Algorithm (6)

- Example: re-estimating centroids from data in the clusters



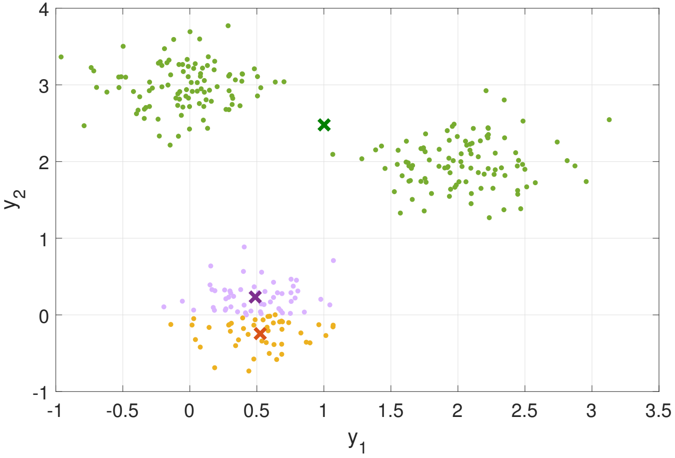
K-means Algorithm (7)

- Example: after 3 iterations, centroids are stable



K-means Algorithm (9)

- The k-means algorithm is sensitive to starting points



K-means Algorithm (10)

- k-means tries to optimise the function

$$D(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K) = \sum_{i=1}^{n} \min_k \{d_k(i)\}$$

$\Longrightarrow$ Tries to minimise distance of each point to nearest centroid

- Bad seeding leads to local minima
- k-means++ algorithm improves convergence dramatically
  - Randomly choose centers to be far apart from each other

Further Clustering

- Alternative similarity measures
  - Weighted Euclidean distance
  - "Cityblock" distance
  - Hamming distance (for pure binary data)
  - and many more ...
- Some potential issues
  - "Hard" classification of items to clusters
  - Difficult to handle mixed attributes (continuous, discrete)
  - No explicit statistical interpretation
  - How to choose K using just the data?
- Mixture modelling a flexible alternative

Mixture Modelling (1)

- Models data as a mixture of probability distributions

$$p(y_{i,j}) = \sum_{k=1}^{K} \alpha_k p(y_{i,j} \mid \boldsymbol{\theta}_{k,j})$$

where

  - K is the number of classes
  - $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ are the mixing (population) weights
  - $\boldsymbol{\theta}_{k,j}$ are the parameters of the distributions

- Has an explicit probabilistic form
  $\Longrightarrow$ allows for statistical interpretion

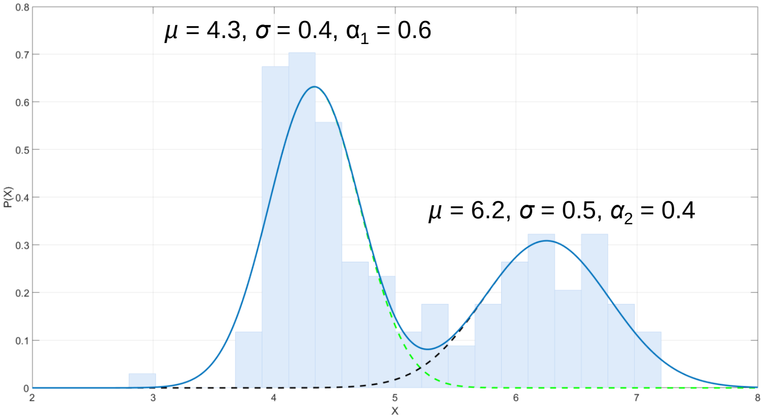
Mixture Modelling (2)

- How is this related to clustering?
- Each class is a cluster
  - Class-specific probability distributions over each attribute
    - e.g., normal, inverse Gaussian, Poisson, etc.
  - Mixing weight is prevalance of items in the class
    - Fraction of our population in that particular subpopulation

- The resulting mixture model has
  - K different classes (subpopulations)
  - q different models for each class, one for each attribute
    - $\boldsymbol{\theta}_{k,j}$ are parameters of model for attribute j in class k
  - $K \times q$ total probability models

Mixture Modelling (3)

- Example: two normal distributions



Mixture Modelling (4)

- Measure of similarity of item to class

$$p_k(\mathbf{y}_i) = \prod_{j=1}^{q} p(y_{i,j} \mid \boldsymbol{\theta}_{k,j})$$

$\implies$ probability of item's attributes under class distributions

- For Gaussian models, this is equivalent to Euclidean distance
- For non-Gaussian models (Bernoulli, Poisson, etc.) it is often a generalisation of the Euclidean distance
  - Related to something called Kullback–Leibler divergence

Mixture Modelling (5)

- Membership of items to classes is soft

$$r_{i,k} = \frac{\alpha_k \, p_k(\mathbf{y}_i)}{\sum_{l=1}^{K} \alpha_l \, p_l(\mathbf{y}_i)}$$

- Application of Bayes' theorem
- Posterior probability of belonging to class k
  - α_k is *a priori* probability item belongs to class k
  - $p_k(\mathbf{y}_i)$ is probability of data item $\mathbf{y}_i$ under class k

$\Longrightarrow$ Assign to class with highest posterior probability

- Total number of samples in a class is then

$$n_k = \sum_{i=1}^{n} r_{i,k}$$

Multivariate Normal Distribution (1)

- So far we have considered seperate univariate distributions for each attribute
- However, it would be useful to model attributes as related
- Multivariate normal distributions are important in statistics
  - Generalize normal distributions to more than one dimension
  - Allow for correlation between random variables
- Are important in mixture model
- They model relationships between multiple random variables
  - The attributes of an individual are likely related
  - For example, height and weight will show correlation

Multivariate Normal Distribution (2)

- If $\boldsymbol{Y} = (Y_1, \ldots, Y_q)$ are RVs with pdf

$$\left(\frac{1}{2\pi}\right)^{\frac{q}{2}} \sqrt{|\boldsymbol{\Sigma}^{-1}|} \exp\left(-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu})\right)$$

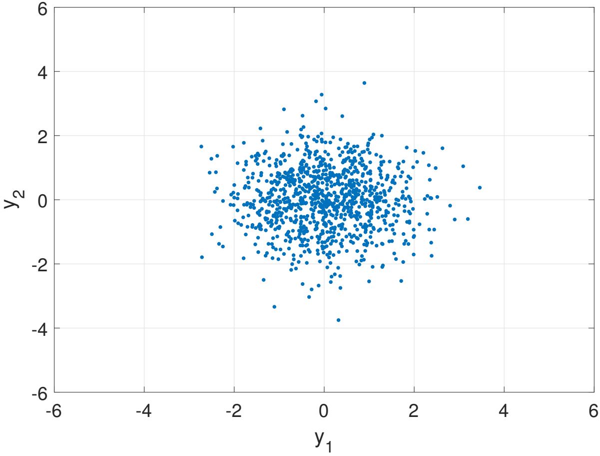
  then they are multivariate normal with means $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_q)$ and covariance matrix $\boldsymbol{\Sigma}$
- The entry μ_j is the mean for Y_j
- The entry

$$\Sigma_{i,j} = \text{cov}(Y_i, Y_j)$$

  is the covariance between Y_i and Y_j.

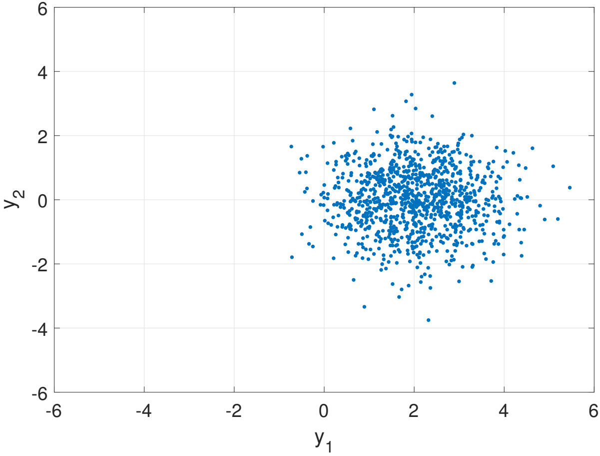
Multivariate Normal Distribution (3)

- Example, $\boldsymbol{\mu} = (0, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



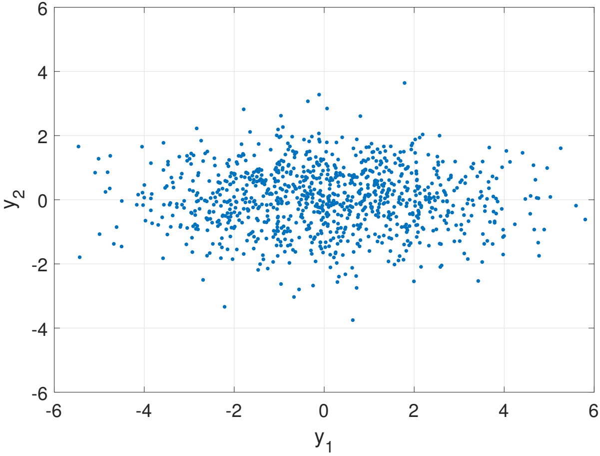
Multivariate Normal Distribution (4)

- Example, $\boldsymbol{\mu} = (2, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



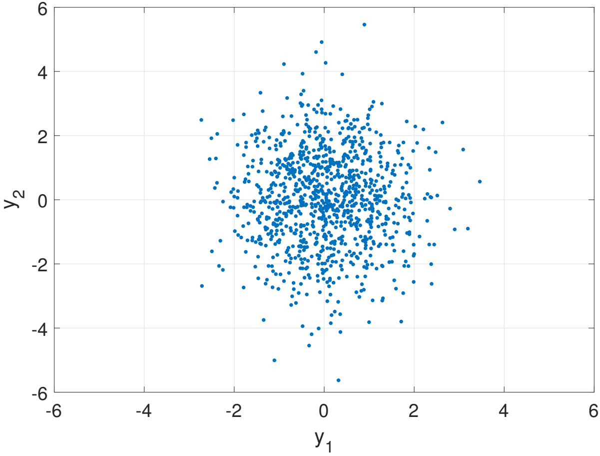
Multivariate Normal Distribution (5)

- Example, $\boldsymbol{\mu} = (0, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$



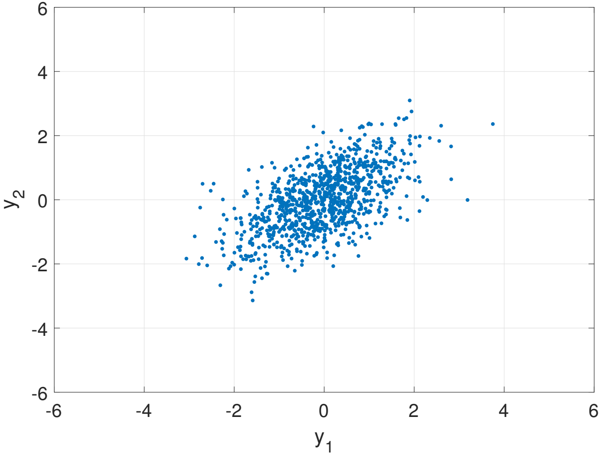
Multivariate Normal Distribution (6)

- Example, $\boldsymbol{\mu} = (0, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1.5 \end{pmatrix}$



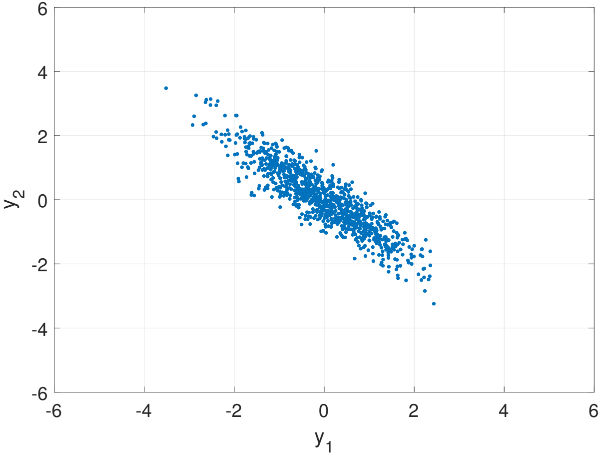
Multivariate Normal Distribution (7)

- Example, $\boldsymbol{\mu} = (0, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.6 \\ 0.6 & 1 \end{pmatrix}$



Multivariate Normal Distribution (8)

- Example, $\boldsymbol{\mu} = (0, 0)$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & -0.9 \\ -0.9 & 1 \end{pmatrix}$



Multivariate Normal Distribution (9)

- Multivariate normal generalises the univariate normal distribution
  - For $q = 1$, reduces to usual normal distribution
- Several different common covariance structures:
  - Diagonal $\boldsymbol{\Sigma}$, all variances the same (spherical)
  - Diagonal $\boldsymbol{\Sigma}$, variances differing
  - Arbitrary $\boldsymbol{\Sigma}$ (elliptical)
- Each structure has more parameters to estimate
  $\Longrightarrow$ more flexible, but more complex

Estimating Mixture Models (1)

- Given class memberships, the negative log-likelihood of data in class k is
$$-\sum_{i=1}^{n} r_{i,k} \sum_{j=1}^{q} \log p(y_{i,j} \,|\, \boldsymbol{\theta}_{k,j})$$
$\implies$ weighted negative log-likelihood
- Use expectation-maximisation (EM) algorithm
  1. Estimate parameters, $\boldsymbol{\theta}_{k,j}$, $(k = 1, \ldots, K)$, $(j = 1, \ldots, q)$ using weighted maximum likelihood
  2. Re-calculate class memberships $r_{i,k}$ based on new parameters
  3. If estimates have not stabilised, go to step (1)
- Initialise model with random class memberships
- Generalisation of k-means

Estimating Mixture Models (2)

- Find K by minimising a goodness-of-fit criterion
- Difficult, non-convex optimisation problem
  $\Longrightarrow$ Many local minima
- Each iteration, do the following
  - Remove classes with too few data points
  - Attempt to split all classes
  - Attempt to combine pairs of classes
  - Randomly assign data to classes, and re-estimate
- The mixture model with the smallest criterion score is retained, and the process is repeated

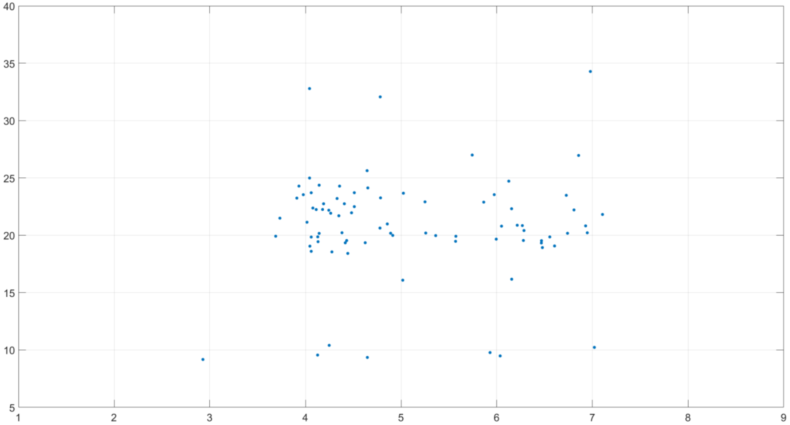
Estimating Mixture Models (3)

- Information Criteria goodness-of-fit criterion
  - Popular for learning mixture models
- Information criterion score is our yardstick, comprised of
  1. Goodness of fit of the mixture model to the data
  2. Model complexity penalty based on number of classes/parameters

  $\Longrightarrow$ choose model which balances complexity against fit
- Popular method is called minimum message length
  - Developed here at Monash by C.S.Wallace
  - Uses information theory interpretation of probability
  - Compress data using model; find model that leads to shortest compressed data

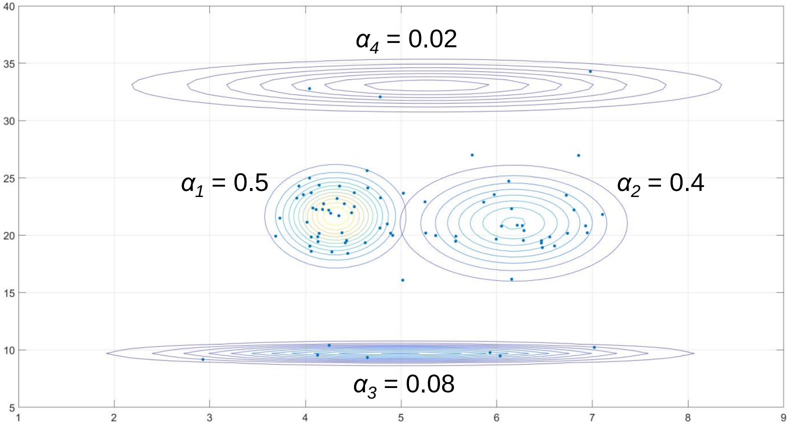
Example (1)

- Example: two dimensional dataset



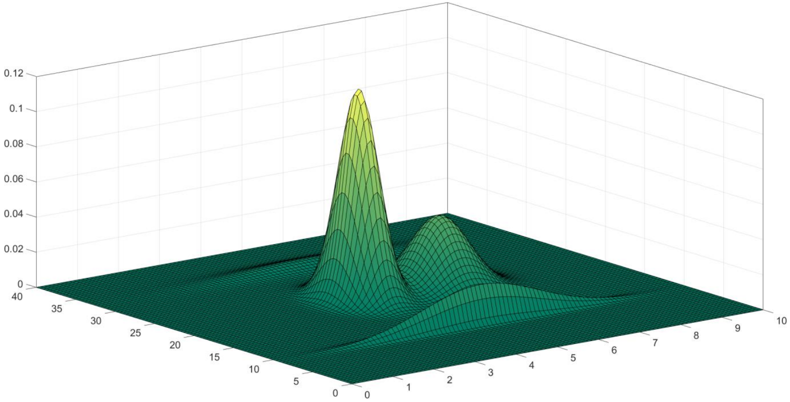
Example (2)

- Mixture modelling discovers $K = 4$ classes

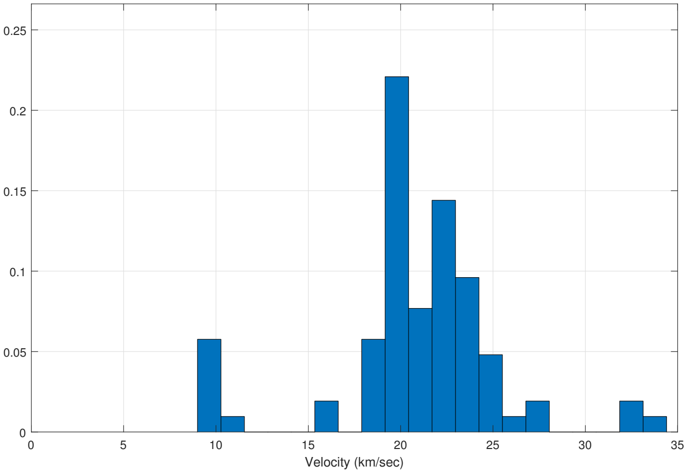


Example (3)

- Plot of the mixture model density

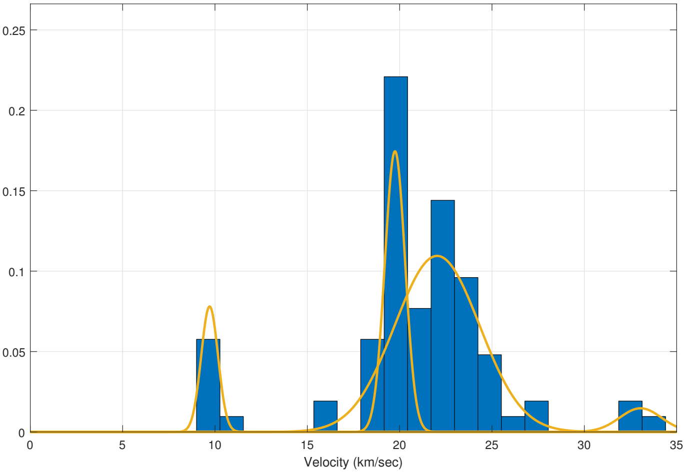


Example: Galaxy data (1)



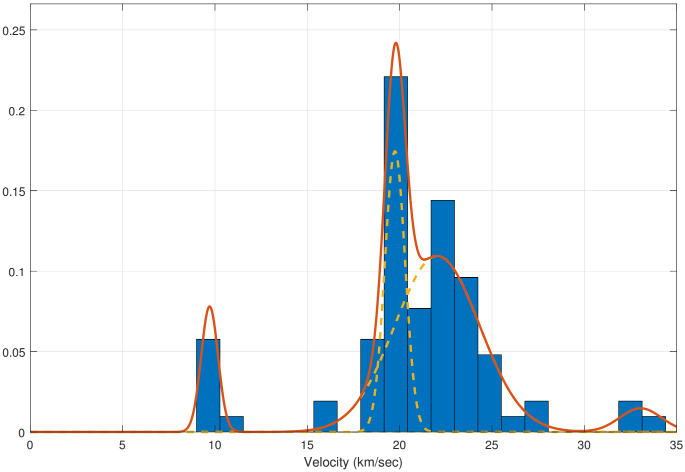
Data on $n = 82$ galaxies; each data point is the velocity of a galaxy.

Example: Galaxy data (2)



Mixture modelling finds $K = 4$ classes.
$8.9\% \, N(9.71, 0.2), \; 23\% \, N(19.74, 0.3), \; 62\% \, N(22, 5.25), \; 4\% \, N(33.04, 1.27)$

Example: Galaxy data (3)



Mixture modelling finds $K = 4$ classes.
$8.9\% \, N(9.71, 0.2), \; 23\% \, N(19.74, 0.3), \; 62\% \, N(22, 5.25), \; 4\% \, N(33.04, 1.27)$

Example: Multivariate Data Analysis (1)

- Well known diabetes dataset
  - 268 diabetics, 500 non-diabetics
  - 768 samples, with 8 predictors
  - 763 missing exposure measurements (12%)
- Outcome is diabetes in Pima indians (DIA)

Pima Indians Variables

Name	Mean	σ	Min	Max	% Missing
Number of Pregnancies (PREG)	4.5	3.2	1	17	14.4%
Plasma Glucose Concentration (PLAS)	121.6	30.5	44	199	0.6%
Diastolic Blood Pressure (BP)	72.4	12.4	24	122	4.5%
Triceps Skin Fold Thickness (SKIN)	29.1	10.5	7	99	29.5%
2-hour Serum Insulin (INS)	155.5	118.8	14	846	48.7%
Body Mass Index (BMI)	32.4	6.9	18.2	67.1	1.4%
Diabetes Pedigree Function (PED)	0.47	0.33	0.078	2.42	0%
Age (AGE)	33.2	11.7	21	81	0%

Example: Multivariate Data Analysis (2)

- Estimate mixture model for exposures and outcome
  - All predictors Gaussian, target (diabetes) is Bernoulli
  - $I_4 = 18{,}719.1$, $I_5 = 18{,}713.0$, $I_6 = 18{,}714.7$, $I_7 = 18{,}732.7$

Pima Indians Mixture Model (Means)

Class	$\hat{\alpha}_k$	PREG	PLAS	BP	SKIN	INS	BMI	PED	AGE	DIA
1	0.13	2.5	150	75	35	238	37	0.59	33	0.82
2	0.23	7.6	141	78	33	214	35	0.52	43	0.78
3	0.25	2.0	104	66	20	105	27	0.42	24	0.02
4	0.19	2.7	112	71	34	138	36	0.47	26	0.20
5	0.18	6.4	110	75	28	117	30	0.41	42	0.06

Outline

Matrix Completion Problem (1)

- We have a large matrix of data $\mathbf{Y}$
  - Rows of $\mathbf{Y}$ are individuals
  - Columns of $\mathbf{Y}$ are attributes of individuals
- Many entries of $\mathbf{Y}$ are missing
  - Usually they are unmeasured

- Matrix completion involves filling in the missing entries
- Assume individuals are independent, attributes are dependent
  - Use dependencies between attributes to estimate missing entries

Matrix Completion Problem (2)

- Some applications of matrix completion
  1. Imputation
     - Matrix of features for a supervised learning problem
     - Most supervised learning methods cannot handle missing data
     - Filling in missing entries lets us use entire matrix
  2. Recommender systems
     - Matrix is set of ratings/purchasers
     - Rows are individuals, columns are products
     - For example, Netflix challenge

Terminator	Love Actually	Aliens	Predator	Bridesmaids
2	4	2	1	5
4	1	5	4	1
4	–	4	–	–
–	4	–	–	–

     - Estimate missing ratings and recommend movies

Matrix Completion Problem (3)

- Univariate imputation
  - Simplest approach to imputation
  - Estimate a statistical model each column
  - Replace missing entries with suitable statistic
    - Mean/median for numeric variables
    - Mode for categorical variables
  - Ignores structure and relationships between variables
  - Is very fast
- Multivariate normal
  - Specify correlations between variables
  - Estimate missing entries using correlation info
  - Takes into account relationships between variables
  - Assumes data is clustered in one single cluster
    - Can use mixture modelling to extend this idea further

Imputation using k-Nearest Neighbours (1)

- Recall k-nearest neighbours algorithm
  - We have a set of n example predictor/target pairs
    - Predictor values $x_{i,1}, \ldots, x_{i,p}$ paired with target y_i
  - We want to predict target value for new individual with predictor values $x'_1, \ldots, x'_p$
- Find k individuals in our data "most similar" to the new individual
  - Use target values of these k individuals to predict target for our new individual
- Very weak assumptions
  - Individuals similar to each in other in terms of predictor values will be similar in terms of targets
- Use cross-validation to select neighbourhood size k

Imputation using k-Nearest Neighbours (2)

- Easily adapted to matrix completion
- When computing similarity, ignore missing values
- Then, for each column $j = 1, \dots, p$
  - Predict each missing entry in column j using all other columns as explanatory variables
- Sometimes called collaborative filtering
- Netflix example using $k = 1$

Terminator	Love Actually	Aliens	Predator	Bridesmaids
2	4	2	1	5
4	1	5	4	1
4	?	4	–	–
–	4	–	–	–

Imputation using k-Nearest Neighbours (3)

- Easily adapted to matrix completion
- When computing similarity, ignore missing values
- Then, for each column $j = 1, \ldots, p$
  - Predict each missing entry in column j using all other columns as explanatory variables
- Sometimes called collaborative filtering
- Netflix example using $k = 1$

Terminator	Love Actually	Aliens	Predator	Bridesmaids
2	4	2	1	5
4	1	5	4	1
4	$?$	4	$-$	$-$
$-$	4	$-$	$-$	$-$

Imputation using k-Nearest Neighbours (4)

- Easily adapted to matrix completion
- When computing similarity, ignore missing values
- Then, for each column $j = 1, \ldots, p$
  - Predict each missing entry in column j using all other columns as explanatory variables
- Sometimes called collaborative filtering
- Netflix example using $k = 1$

Terminator	Love Actually	Aliens	Predator	Bridesmaids
2	4	2	1	5
4	1	5	4	1
4	1	4	–	–
–	4	–	–	–

Reading/Terms to Revise

- Terms you should know:
  - Clustering
  - k-means algorithm
  - Mixture modelling
  - Matrix completion
  - Imputation
  - Collaborative filtering

- Next week: simulation based methods (bootstrapping, permutation tests, random number generation)