Generative Models: Clustering

Machine Learning

Unsupervised Learning I

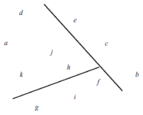
- So far, we have looked at constructing models from data D of the form $(\mathrm{x}_1, y_1), (\mathrm{x}_2, y_2), \ldots$ where the y_i are either numeric or nominal values, and the x_i are usually some vector of values
- The y_i are sometimes called *dependent* or *outcome* variables
- When the y_i are nominal (classes, for example), then the x_i's with the same y value can be thought of as a *cluster* or a group
    - When we constructed a set of rules for a predicting a class, we were identifying patterns that characterise that cluster
    - As we saw with classification models, the task of predicting y_i given the x_i can be seen as an application of discriminatory models (that is, we build a model for $P(Y|\mathrm{X})$)
    - We saw applications of this with logistic regression and class-probability trees
- But what if there were no y_i's, and we still want to find groups of similar instances

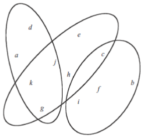
Unsupervised Learning II

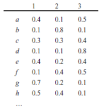
  - One way to do this is to use generative models (that is, build a model for $P(\mathrm{X})$)
  - We will see two examples of this approach using Naive Bayes and a mixture of Gaussians
  - All instances in a cluster can then be taken as having the same class value
  - Classification with missing (x_i, y_i) instances in which all the y_i are missing (cue: EM)
- BUT: why are there no y_i's?
  - Finding labels can be expensive: easier to get large numbers of x_i's
  - Even if some of the data were labelled, there might be *concept drift*
  - There may be no specific concept
- There are many different kinds of clustering models, but they are all essentially based on on a notion of *distance* between instances

Unsupervised Learning III

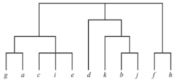
- Here are some different kinds of cluster representations:







	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
...			



From: Witten, Frank and Hall, *Data Mining* (2011)

Clustering

- The goal of clustering is to find sub-groups of N elements (instances)
- The sub-grpups can be partitions, but not always. Elements from same group (*cluster*) should have high similarity, and elements from different clusters low similarity
- Homogeneity and separation not well-defined. In practice, these concepts usually rely on some notion of distance between instances

Step Back: Distances I

- Minkowski distance: For instances from $\Re^d$, the *Minkowski distance* of order $p > 0$ is defined as:

$$Dis_p(\mathrm{x}, \mathrm{y}) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p}$$

- This is sometimes denoted L_p. We will often refer to Dis_p simply as the p-norm
- The 2-norm refers to the familiar *Euclidean distance*:

$$Dis_2(\mathrm{x}, \mathrm{y}) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} = \sqrt{(\mathrm{x} - \mathrm{y})^T (\mathrm{x} - \mathrm{y})}$$

Step Back: Distances II

- The 1-norm refers to the familiar *Manhattan distance*, (or *cityblock distance*):

$$Dis[_1](\mathrm{x},\mathrm{y}) = \sum_{j=1}^{d} |x_j - y_j|$$

- If we now let p grow larger, the distance is dominated by the largest coordinate-wise distance. $Dis[_\infty](\mathrm{x},\mathrm{y}) = \max_j |x_j - y_j|$ this called the *Chebyshev distance*.
- You will sometimes see references to the 0-norm, which counts the number of non-zero elements in a vector. The corresponding distance then counts the number of positions in which vectors x and y differ

Step Back: Distances III

- This is not strictly a Minkowski distance; however, we can define it as

$$Dis[_0](\mathrm{x}, \mathrm{y}) = \sum_{j=1}^{d} (x_j - y_j)^0 = \sum_{j=1}^{d} I(x_j = y_j)$$

  where $I(.)$ is an indicator function, that is 1 if its argument is and 0 otherwise
- If x and y are binary strings, this is also called the *Hamming distance*.
- Alternatively, we can see the Hamming distance as the number of bits that need to be flipped to change x into y.
- For non-binary strings of unequal length this can be generalised to the notion of *edit distance* or the *Levenshtein distance*

- Some desirable properties of distances
  - Distance between two points cannot be negative

Step Back: Distances IV

- Distance between a point to itselft is 0 (conversely, if the distance between a pair of points is 0, then the points are identical)
- Distance between points A and B is the same as between points B and A
- Distance between points A and C is less than or equal to the sum of the distances between points A and B; and B and C

- Distances that satisfy these properties are called *metrics*
- Given a set of points and a distance function, we can construct a *distance matrix* whose entry d_{ij} is the distance between x_i and x_j

Simple Distance-Based Clustering: Nearest Neighbour I

- The simplest kind of learning consists of looking up the table of instances seen so far:
    1. Given: A dataset of instances $\{x_1, x_2, \ldots, x_k\}$ where each x_i is actually a N-dimensional vector (in which one of the attributes may be the dependent or outcome variable y_i)
    2. Find: the instance x^* that most closely resembles a new instance x_{new}
    3. Update: x_{new} is added to the set of stored instances
- THEN WHAT? Depends: if you are interested in classifying x_{new}, or in numeric prediction then return y_i
- What if there is more than one x_{new} is reasonably similar to several x_i's? Return the (weighted) mean, mode, or median — whichever makes sense
- BUT: is this “learning”? Yes, in a manner (by memorisation, and using a distance measure)
    - Usually uses k closest instances (k-nearest-neighbour)

Simple Distance-Based Clustering: Nearest Neighbour II

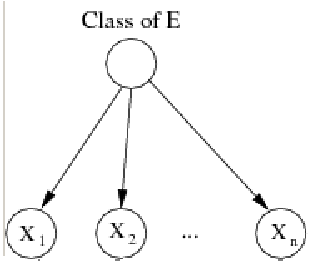
- Instance-based learning
  - Storage: may end up storing a lot (all?) the instances
  - Distance: Standard Euclidean distance is OK if all attributes are numeric. But what if they are not all numeric?
  - Pattern: no explicit pattern is learnt of similar instances, so we will always need all of the data
- Clustering methods are a natural extension of instance-based learning, but using more powerful representations for groups of instances

Probabilistic Clustering: Simple Bayes Model I

- Distance-based measures can be seen as special-cases or aproximations of a more general form of probability-based clustering it
  - Assume some joint-probability model
  - Estimate the parameters of the model that result in highest posterior probability (or highest likelihood, if using equi-probable priors)
- Before we look at more elaborate probabilistic models for clustering, we will look at a simple form of probabilistic clustering, using Naive Bayes as the joint probability model, that uses ideas from a method known as the EM algorithm

Probabilistic Clustering: Simple Bayes Model II

- Recall the network structure for the simple Bayes classifier:



- Usually, we are concerned with estimating parameters with this network given data for the random variables, along with a class value of this kind:

Probabilistic Clustering: Simple Bayes Model III

X_1	X_2	...	X_n	Y
1	0	...	0	$+$
1	1	...	0	$-$
0	1	...	1	$-$
...	...	...	...	...
...	...	...	...	...

We can easily compute the CPTs with this data (with beta priors if needed)

- We can treat the clustering problem as one for which the Y values are missing
  - How do we calculate the CPT entries now?
  - EM
- Let us look at an example with $N = 4$. Suppose the data are:

Probabilistic Clustering: Simple Bayes Model IV

X_1	X_2	X_3	X_4	Y
1	0	1	1	?
0	1	1	0	?
0	0	1	1	?
...	...	...	...	...

- The parameters that are needed for the Simple Bayes model are $P(Y), P(X_1|Y), P(X_2|Y), P(X_3|Y)$ and $P(X_4|Y)$
- As usual with EM, we will augment the data with a *count* column:

X_1	X_2	X_3	X_4	Y	*Count*
1	0	1	1	?	
0	1	1	0	?	
0	0	1	1	?	
...	...	...	...	...	

Probabilistic Clustering: Simple Bayes Model V

- Each cluster will act as a coin. We do not know which coin was used to obtain each data instance
- SO: Each data instance has a separate row for each coin, with initial counts assigned randomly:

X_1	X_2	X_3	X_4	Y	*Count*
1	0	1	1	1	0.4
1	0	1	1	2	0.1
1	0	1	1	3	0.5
0	1	1	0	1	0.2
0	1	1	0	2	0.4
0	1	1	0	3	0.4
0	0	1	1	1	0.3
0	0	1	1	2	0.3
0	0	1	1	3	0.4
...	...	...	...	...	...

Probabilistic Clustering: Simple Bayes Model VI

NOTE: The counts are intended to be "fractional instances". The total number of instances should sum to the number of instances of the combination. We will achieve this by rescaling appropriately

- In the E-step, counts are updated. In the M-step, parameters are obtained using Maximum Likelihood.
- Our model stores values for $P(C), P(X_1|C), P(X_2|C), P(X_3|C), P(X_4|C)$. Assume any particular combination r of X values happens m_r times

Probabilistic Clustering: Simple Bayes Model VII

Here is an algorithm sketch:

```
1. Assign $P(C), P^{(}X_j|C)$ randomly
2. while not done
    2.1 For each combination $r$ of $X_{1,r}, X_{2,r}, X_{3,r}, X_{4,r}$ values:
        2.1.1 Calculate $P(C = c_k|X_{1,r}, X_{2,r}, X_{3,r}, X_{4,r}$ using usual NB
              inference with current values of $P(C), P^{(}X_j|C)$ Call this
              $P(c_k|\mathrm{X}_r)$
        2.1.2 Update fractional counts.
    2.2 Re-estimate $P(C), P^{(}X_j|C)$ using new counts
```

The EM Algorithm: Coins with Missing Data I

- Suppose we have data from a sequence of Bernoulli trials, each with a probability of success p. Previously, we have seen how the maximum likelihood estimate of p was s/n where s was the number of successes observed in the data, and n were the total number of trials
- Each Bernoulli trial is like tossing a biased coin with probability p of landing *heads*. Maximum likelihood estimation can be used with more than 1 coin. If we have two coins A and B, then to obtain the maximum likelihood estimates of the parameters p_A and p_B, we repeatedly do the following:
  1. Randomly pick a coin (for the moment with equal probability)
  2. Toss the coin some number (say 10) times
  3. Record the experiment number, the number of *heads* observed; and which coin was chosen
  4. Repeat the experiment (that is, return to Step 1)

The EM Algorithm: Coins with Missing Data II

The maximum likelihood estimates of p_A and p_B will simply be the proportion of *heads* on tosses for which A and B were used

- Now, consider a harder problem. You still have 2 coins A and B, and you have data from say 5 repetitions:

  R1. *HTTTHHTHTH*
  R2. *HHHHTHHHHH*
  R3. *HTHHHHHTHH*
  R4. *HTHTTTHHTT*
  R5. *THHHTHHHTH*

  But there is no record of which coin was used in each case. How can we estimate p_A and p_B? (This is from *Nature Biotechnology*, Vol 26, No.8, pp 897–899.)

- One way is the following:
  1. Start with some guess about p_A and p_B. Call this ${p_A}^{(0)}$ and ${p_B}^{(0)}$

The EM Algorithm: Coins with Missing Data III

2. For each of the repeats R1–R5, calculate $P_A = P(D|{p_A}^{(0)})$ and $P_B = P(D|{p_B}^{(0)})$ (where D is the sequence of H's and T's on that repetition). If $P_A > P_B$, then assume A was used, otherwise assume B was used.
3. Now we have a complete table, and can calculate maximum likelihood estimates as before

We could iterate this entire procedure, using the maximum likelihood estimate as the guesses in Step (1).

- The "expectation-maximisation" algorithmm or EM algorithm is a refinement of this basic idea
- The current guesses for p_A and p_B are used to obtain new "weighted" instances.

The EM Algorithm: Coins with Missing Data IV

- The weights are proportional to $P(coin = A|D)$ and $P(coin = B|D)$. These are proportional to $P(D|coin = A)P(coin = A)$ and $P(D|coin = B)P(coin = B)$ Since the coins were known to be chosen with equal probability, then $P(coin = A|D) = \alpha P(D|coin = A)$ and $P(coin = B|D) = \alpha P(D|coin = B)$ We can calculate this using the current guess of p_A and p_B
- For example, with an initial guess of 0.60 for p_A and 0.50 for p_B, we find the following
  R1. D1 $= HTTTHHTHTH$. Then, $P(Coin = A|D1) = 0.45$ and $P(Coin = B|D1) = 0.55$. So, we create a new set of weighted instances generated by both coins:

$$CoinA : 2.2H \text{ and } 2.2T$$

$$CoinB : 2.8H \text{ and } 2.8T$$

(Both):] $5H$ and $5T$ (as before)

The EM Algorithm: Coins with Missing Data V

R2. D2 = $HHHHTHHHHH$ Then, $P(Coin = A|D2) = 0.80$ and $P(Coin = B|D2) = 0.2$, and get weighted instances:

$$CoinA : 7.2H \text{ and } 0.8T$$

$$CoinB : 1.8H \text{ and } 0.2T$$

(Both): $9H$ and $1T$ (as before)

and so on, for repetitions R3–R5. In each case, the number of *heads* in the weighted instances is the expected value of the number of *heads*, given the current estimate of the parameters. The totals obtained after all calculations are:

$$CoinA : 21.3H \text{ and } 8.6T$$

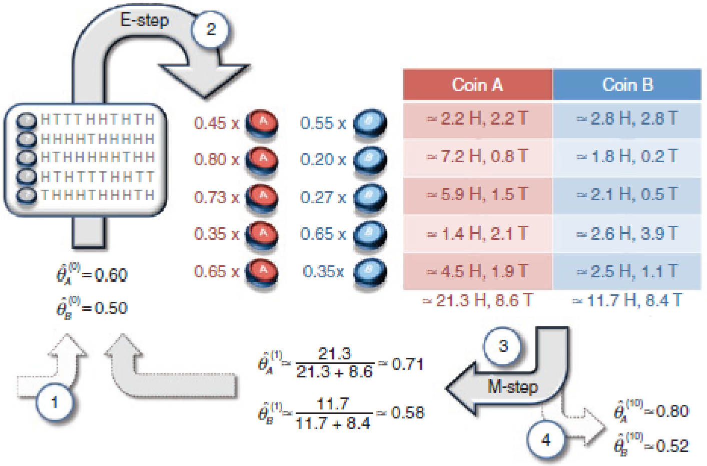
$$CoinB : 11.7H \text{ and } 8.4T$$

This gives a new estimate of p_A as $21.3/(21.3 + 8.6) = 0.71$ and p_B as 0.58

The EM Algorithm: Coins with Missing Data VI

- This can be used iteratively. The procedure we have just done is one iteration of the EM algorithm

The EM Algorithm: Coins with Missing Data VII



Coin A	Coin B
≈ 2.2 H, 2.2 T	≈ 2.8 H, 2.8 T
≈ 7.2 H, 0.8 T	≈ 1.8 H, 0.2 T
≈ 5.9 H, 1.5 T	≈ 2.1 H, 0.5 T
≈ 1.4 H, 2.1 T	≈ 2.6 H, 3.9 T
≈ 4.5 H, 1.9 T	≈ 2.5 H, 1.1 T
≈ 21.3 H, 8.6 T	≈ 11.7 H, 8.4 T

$$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71$$

$$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$$

Probabilistic Clustering: Mixture Models I

- Assume we have a model structure with some parameter Θ, and we want to work out which value of $\Theta = \theta$ is best for the data x (previously we used d) we have observed.
  - As before, θ and x will be values of random variables Θ and X (used to be D)
  - Both Θ and X are now continuous r.v.'s
- The only thing that changes is that both likelihood and priors are calculated using p.d.f's. That is:

Hypothesis.	$\Theta = \theta$
Data.	X in an interval dx around x
Prior.	$f(\theta)d\theta$
Likelihood.	$f(x\|\theta)dx$ (correctly, $f_{X\|\Theta}(x\|\theta)dx$)
Numerator.	$f(x\|\theta)dxf(\theta)d\theta$
Denominator.	$\int_0^1 f(x\|\theta)dxf(\theta)d\theta)\ d\theta$
Posterior.	$f(\theta\|x)d\theta$

Probabilistic Clustering: Mixture Models II

- Example
  - Suppose $X \sim N(\theta, 1)$ and $\Theta \sim N(2, 1)$
  - Then:

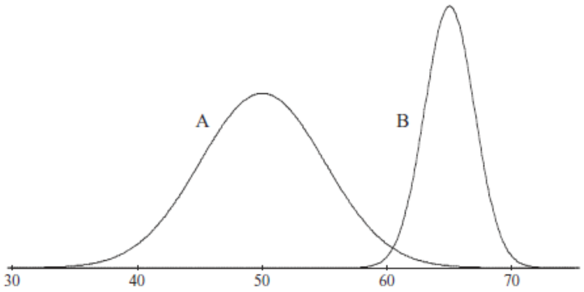
$$\begin{aligned}
\text{Prior.}\quad & c_1 e^{-(\theta-2)^2/2} d\theta \\
\text{Likelihood.}\quad & c_2 e^{-(5-\theta)^2/2} dx \\
\text{Numerator.}\quad & c_3 e^{-(2\theta^2-14\theta+29)/2} dx d\theta = \\
& c_3 e^{-((\theta^2-7/2)^2+9/4)} dx d\theta = c_4 e^{-(\theta^2-7/2)^2} dx d\theta \\
\text{Denominator.}\quad & \left(\int c_4 e^{-(\theta^2-7/2)^2} d\theta\right) dx = c_5 dx \\
\text{Posterior.}\quad & c_6 e^{-(\theta^2-7/2)^2} d\theta
\end{aligned}$$

  - The posterior p.d.f. is $N(7/2, \sigma)$ where $2\sigma^2 = 1$ and $c_6 = 1/(\sigma\sqrt{2\pi})$
- So a Gaussian is conjugate prior for Gaussian likelihood.

Mixture Models I

- A *mixture* is a set of k probability distributions, each representing a cluster
- Any particular instance x_i belongs to a cluster, but it is not known which one. So, each distribution j is used to compute the probability that x_i belongs to cluster j
- Simplest mixture model: there is one numeric attribute X and 2 or more clusters, each specified by a Gaussian with different means and variances

Mixture Models II



(From: Witten, Frank and Hall, (2012))

- Given a dataset, if we knew which cluster each instance came from, then estimating the means and s.d's of the Gaussians is easy:

$$\mu_j = \sum_{x_i \in C_j} \frac{x_i}{c_j} \quad {\sigma_j}^2 = \sum_{x_i \in C_j} \frac{(x_i - \mu_j)^2}{c_j - 1}$$

Mixture Models III

- Once we have the estimates of the parameters, finding the cluster for an instance x is also easy:

$$P(j|x) = \frac{P(x|j)P(j)}{P(x)} = \frac{N(x;\mu_j,\sigma_j)P(j)}{P(x)}$$

where:

$$N(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

(strictly speakimg, this is not correct of course, since the $N(.;.)$ is a p.d.f. But if we put in the $d\theta$'s, this is OK)

- BUT: we don't know the cluster to which instance belongs. So, what is to be done (CUE: EM)
  - Start with guesses for the μ_j, σ_j and $P(j)$ (for 2 clusters, only one guess is needed for $P(j)$).

Mixture Models IV

- Then calculate the fractional intances belonging to each cluster (these will be taken to be equal to the probability of instances belonging to a class)
- Re-calculate the means and s.d's:

$$\mu_j = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_i w_i}$$

$$\sigma_j^2 = \frac{\sum_{i=1}^{N} w_i (x_i - \mu_j)^2}{\sum_i w_i}$$

- The "best clustering" is one in which all instances are in the correct cluster. That is $P(j|x) = 1$ if $x \in j$. The value of $P(x|j)P(j)$ will be a maximum in this case
- The procedure iterates using a measure of goodness based on the weighted likelihood L of the data. The weight of cluster j is the weight of instances belonging to cluster j. The weight of an instance x_i in cluster j is taken as $P(j)P(x_i|j)$.

Mixture Models V

- Again, if we use $N(.;.)$ for $P(x_i|j)$, this is technically not correct, and L will not be between 0 and 1. But it is a good enough approximation for directing the EM procedure
  - In practice, $\log L$ will be used to avoid floating point errors
  - Typically, the EM procedure will run for about 10 iterations or so: usually $\log L$ will increase sharply and then subsequent changes will be small
- Extending from 2 to K classes is straightforward if we know K in advance
- If there are multiple numeric attributes, then each instance is really a vector x_i and either the usual Naive Bayes assumption is made:

$$P(\mathrm{x}_i|j) = \prod_{d=1}^{D} P(x_d|j)$$

Mixture Models VI

OR the clusters are treated as mixtures of multivariate Normal distributions. This allows the calculation of joint-probabilities of a set of correlated attributes.

- If there are missing values, then either impute using means, modes, medians, or add more machinery to the EM procedure
- If the Gaussian is inappropriate, use some other p.d.f (for example, a sigmoid function)
- Increasing the covariant attributes increases the number of parameters (for every pair, the $sigma_i$ is replaced by Σ_{ij} which has 4 entries rather than 1)
  - More parameters means greater risk of overfitting. More correlated attributes mean more parameters.

Approximate Mixture Modelling: K-Means I

- An approximation to the full EM-based Gaussian mixture modelling
  - Single parameter: means of clusters
  - Distance-based cluster membership rather than p.d.f-based cluster membership
  - Instance belongs to a single cluster, rather than fractional instances
- Procedure:
  1. Set value for K (the number of clusters)
  2. Randomly assign values for the centres (means) of the k clusters
  3. repeat
  4. For each instance x do:
     - 4.1 Calculate distance to each of the k-means
     - 4.2 Assign x to the closest cluster
  5. Recalculate the arithmetic means of each of the k-clusters
  6. until (local minimum)

Approximate Mixture Modelling: K-Means II

- BUT: when are we done? The procedure stops at a local minimum for:

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \|\mathrm{x}_n - \mu_k\|^2$$

  where $z_{nk} = 1$ if x_n is closest to cluster k and 0 otherwise; and $\| \cdot \|$ is the 2-norm
- Given a set of points, the arithmetic mean of the points minimises the sum of squared Euclidean distances to those points (Can you prove this?)
- Given a set of points C_k in a cluster k with a centre d_k, $\sum_{j \in C_k} \|\mathrm{x}_j - d_k\|^2$ will be minimum iff d_k is the arithmetic mean of the points in C_k

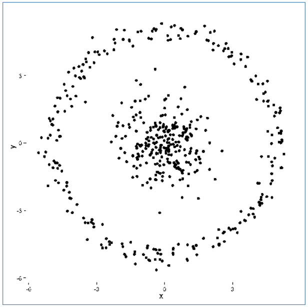
Approximate Mixture Modelling: K-Means III

- When the cluster assignment for each point does not change after an iteration (that is, if z_{nk} does not change), then by construction d_k will be the arithmetic mean of the points in C_k
- BUT: as the number of clusters increases, D becomes smaller (why?) In the worst case, it is possible to end up with $D = 0$ for $K = N$ (clusters with single points)
  - This is a problem

More Problems: When K-means Can Fail I

- K-means has some underlying assumptions:
  - Clusters are spheres centred at different points
  - Clusters are approximately of the same size (priors are the same)
- Each of these assumptions can be violated quite easily. Here is a case where we know the number of clusters exactly, and even their arithmetic means :

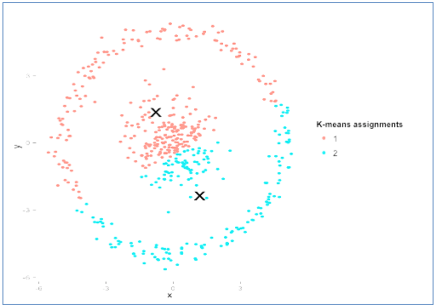
More Problems: When K-means Can Fail II



(from `stats.stackexchange.com`)

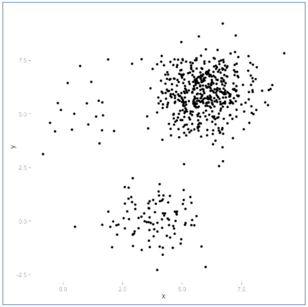
More Problems: When K-means Can Fail III

But the data do
not satisfy the "spherical data" assumption, and K-means fails:



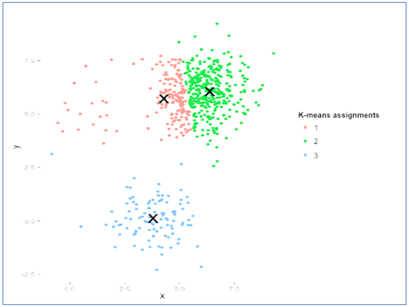
More Problems: When K-means Can Fail IV

- Here there are unequal clusters:



More Problems: When K-means Can Fail V

and again, the procedure finds sub-optimal clusters



- SO: what is to be done? There are other clustering techniques (that make other assumptions)

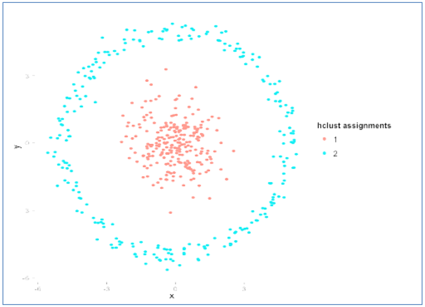
Hierarchical Clustering I

- Assign each instance to its own cluster. There are now N clusters
- Find the closest pair of clusters and merge them into a single cluster
  - Distances between clusters need not just be between the arithmetic mean of instances in the cluster
  - Each merge operation reduces the clusters by 1
- Distance between clusters
  - Single-linkage: nearest neighbour
  - Complete-linkage: furthest neighbour
  - Average-linkage: average of distances of all instance-pairs
- SO: pick a distance function, pick a linkage method, find distances between all cluster pairs, and merge the closest pair
  - This is *agglomerative* clustering (goes from N clusters to 1 cluster). The complexity is $O(N^2 \log N)$

Hierarchical Clustering II

- The other way is *divisive* clustering (goes from 1 cluster to N clusters). The complexity is $O(2^N)$, although greedy methods can do better

- Single-linkage agglomerative clustering (using Euclidean distances)



Summary

- There are several problems which are not about predicting a value y, given a data instance x. Instead, the task is simply to find subsets $S_1, S_2, \ldots, S_k$ of of a set of data instances $D = x_1, x_2, \ldots, x_N$, s.t. the S_i satisfy some properties
- Problems like these are collectively called *unsupervised learning* tasks, of which *clustering* problems are the best known (but not the only one) They are best tackled by thinking of them as generative models rather than discriminative models
- At the heart of most generative models for clustering is a distance measure
- One easy way of looking at clustering is to consider it as an application of using generative probabilistic models (like Naive Bayes), with a probability-based distance measure. Other clustering techniques exist that identify subsets differently, using other kinds of distance measures (e.g. hierarchical clustering)