Mixture Models

UBCO MDS — DATA 573

Mixture Models

- Mixture models are just a combination of statistical distributions
- In general, for a random vector $\mathbf{X}$, a mixture distribution is defined by

$$f(\mathbf{x}) = \sum_{g=1}^{G} \pi_g p(\mathbf{x} \mid \theta_g)$$

- Where:
  - G is the number of groups
  - π_g are 'mixing proportions' — between 0 and 1, $\sum_{g=1}^{G} \pi_g = 1$
  - $p_g(\mathbf{x} \mid \theta_g)$ are parametric statistical distributions, with parameters θ_g.

Mixture Models

- In general, the $p_g(\mathbf{x} \mid \theta_g)$ could include a variety of distributions.
- Commonly, they are assumed to be the same, but with different θ_g.
- So, for example, all p_g are assumed multivariate normal, with mean vector $\boldsymbol{\mu}_g$ and covariance matrix $\boldsymbol{\Sigma}_g$.
- This is commonly referred to as a Gaussian mixture model.

Mixture Models

- Mixture models can be used in a variety of modelling scenarios.
- Perhaps their most natural application is in clustering — commonly referred to as model-based clustering.
- Why is this natural? Groups can be thought of as different populations, and therefore can be modelled with differing distributions.
- It is essentially discriminant analysis without the help of a response variable!

Problem

- So we want to fit separate distributions to unknown groups in the data.
- If groups are unknown, how can we estimate their distribution?
- Most commonly, we use the expectation-maximization (EM) algorithm, which helps us find parameter estimates with missing data (group memberships are missing).

EM Algorithm

- First, let's get specific about the missing data.
- We introduce a new variable that represents the group memberships: Z_{ig}. If $z_{ig} = 1$, then observation i belongs in group g. Otherwise, $z_{ig} = 0$.
- Now that we have that, let's look at a non-technical summary of the EM algorithm...

EM Algorithm: Summary

1. Start the algorithm with random values for $\hat{z}_{ig}$. (there are alternative starting options)

2. Assuming those $\hat{z}_{ig}$ are correct, estimate parameters $\boldsymbol{\mu}_g$ and $\boldsymbol{\Sigma}_g$ (via MLEs — hence, this is the maximization of EM)

3. Assuming those parameters are correct, find the expected value of group memberships

$$\hat{z}_{ig} = \frac{\pi_g \phi(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)}{\sum_{g=1}^{G} \pi_g \phi(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)}$$

   (this is the expectation of EM)

4. Repeat 2. and 3. until 'changes' are minimal. (The log-likelihood of the model is monitored for convergence)

Some Notes

- It may not be obvious, but the estimates $\hat{z}_{ig}$ are not going to be 0's and 1's when the algorithm converges.
- They are actually probabilities!
- So if $\mathbf{z}_i = (0.25, 0.6, 0.15)$?
- In comparison to k-means and hierarchical, this is a neat result. They are often called 'fuzzy' clustering results.
- But! We usually still want a 'best' guess as to which group the observation belongs to. Above example? Certainty and uncertainty?

Mixture Model Problem 1

- EM algorithm is a deterministic, monotonic optimization algorithm. It is (very) susceptible to local maxima!
- AKA, the solution is unlikely to be the 'best', and every time we run the algorithm with a different random start, we risk getting different clustering results.
- Strategy?

Mixture Model Problem 2

- Covariance matrix is $p \times p$, requiring $p(p+1)/2$ parameters to estimate...and we have G of them!!
- And for the EM, we are going to have to invert them as well.
- This is all quite computationally expensive...model-fitting can take some time.
- Strategy?

Mixture Model Benefits

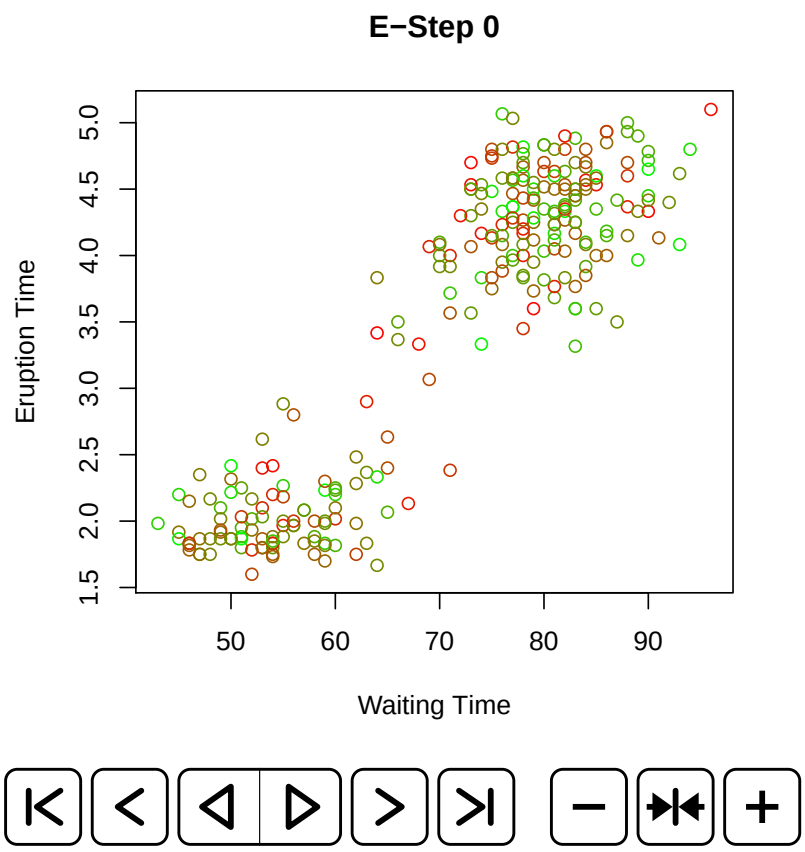
- What if we have skewed or outlier-ridden data (that is, clearly not normally distributed data)?
- Easy in theory — assume a more appropriate statistical distribution instead of MVN in the model. In practice? Could be tricky. If the software doesn't exist yet, you have a lot of mathematics and software to write.
- Parameter estimation/model-fitting is built-in to your clustering algorithm! Interested in the difference in covariance among the variables in each of your groups? Just take a look at them.
- We can look at hypothesis testing, standard model-fitting measures (AIC, BIC, etc) and other statistical inference like we do with all other statistical models.

Mixture Model Benefits

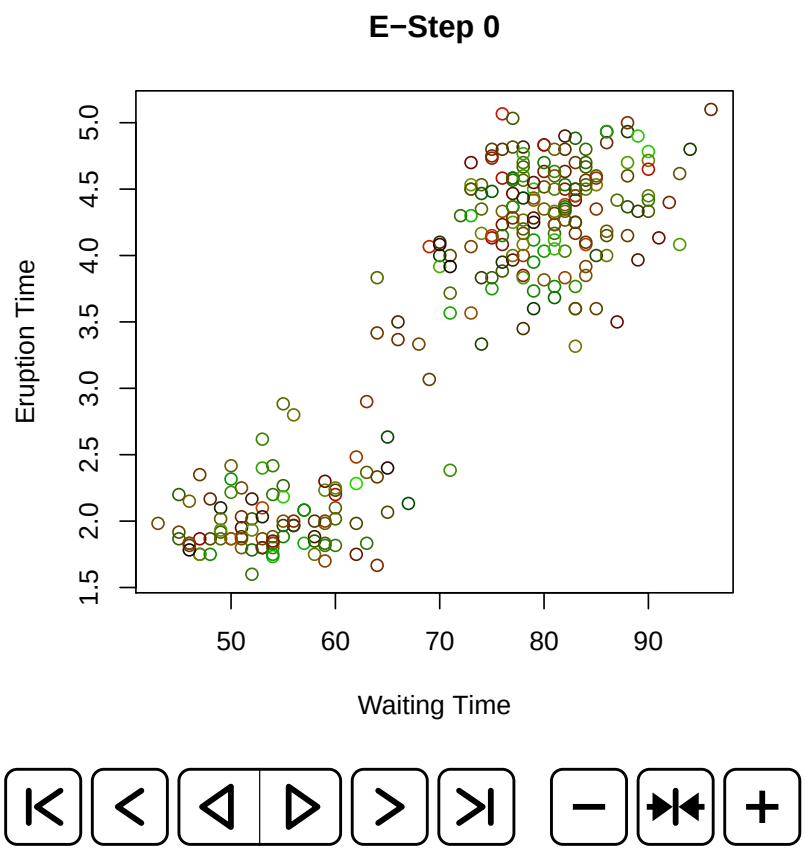
- That last point is perhaps a bigger deal than you realize...
- Mixture models provide us the first statistically viable option to answer the most basic clustering question: "how many groups are present in this data?"
- Most software use the BIC to answer this. So whichever model provides the largest BIC, that's what we go with.

Clustering Old Faithful

tEIGEN



MCLUST



Why Mixture Model-based Clustering? Summary

- Pros
  - Mathematically and statistically sound using standard theory
  - Parameter estimation is built-in, along with number of clusters via model selection
  - Hypothesis testing
  - Provides a probabilistic answer to the question "does observation i belong to group G?"
  - Data with outliers/skewness can be handled by non-Gaussian densities
- Cons (a.k.a. open avenues for research)
  - Computationally intensive (takes time to estimate and invert covariance matrices; not feasible for small n large p without some decomposition)
  - Model-selection method is still viewed as an open problem by many practitioners — BIC grudgingly accepted
  - Traditional parameter estimation via EM algorithm prone to local maxima, overconfidence in clustering probabilities, and several other issues

Mixture Model: Relationship with k-means

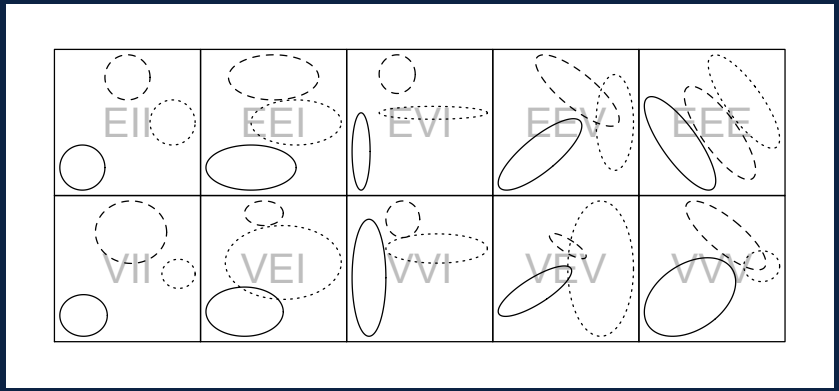
- Many fail to recognize that k-means is actually a highly restricted mixture model! We need to change two assumptions in mixture models to get back to the k-means algorithm...

1. $f(\mathbf{x}) = \sum_{g=1}^{G} \pi_g \phi(\mathbf{x} \mid \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g^*)$ where $\boldsymbol{\Sigma}_g^* = \lambda I$ and λ is a scalar (no covariance...under MVN assumptions, this means assuming independence among the variables)

2. During the EM algorithm, we must restrict our $\hat{z}_{ig}$ to only 0's and 1's.

Mixture Model Families

- To combat overparameterization, we often develop 'families' of mixture distributions via various decompositions of the covariance structure.
  - Facilitates parsimony and clustering flexibility
  - Occasionally gives interesting geometric properties (MCLUST)



- $\boldsymbol{\Sigma}_g = \lambda_g \mathbf{D}_g \mathbf{A}_g \mathbf{D}_g'$ (GPCM, MCLUST, tEIGEN)
- $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g' + \boldsymbol{\Psi}_g$ (MCFA, PGMM, MMtFA)

Example

- Let's use some software on a few benchmarking data sets
- First, the iris data set!

Example

```
> mrun <- Mclust(iris[,-5], G=1:5); t(mrun$BIC)
Bayesian Information Criterion (BIC):
            1          2         3         4         5
EII -1804.0854 -1123.4117 -878.7650 -893.6140 -782.6441
VII -1804.0854 -1012.2352 -853.8144 -812.6048 -742.6083
EEI -1522.1202 -1042.9679 -813.0504 -827.4036 -741.9185
VEI -1522.1202  -956.2823 -779.1566 -748.4529 -688.3463
EVI -1522.1202 -1007.3082 -797.8342 -837.5452 -766.8158
VVI -1522.1202  -857.5515 -744.6382 -751.0198 -711.4502
EEE  -829.9782  -688.0972 -632.9647 -646.0258 -604.8131
EVE  -829.9782  -657.2263 -666.5491 -705.5435 -723.7199
VEE  -829.9782  -656.3270 -605.3982 -604.8371        NA
VVE  -829.9782  -605.1841 -636.4259 -639.7078 -632.2056
EEV  -829.9782  -644.5997 -644.7810 -699.8684 -652.2959
VEV  -829.9782  -561.7285 -562.5522 -602.0104 -634.2890
EVV  -829.9782  -658.3306 -656.0359 -725.2925        NA
VVV  -829.9782  -574.0178 -580.8396 -630.6000 -676.6061

Top 3 models based on the BIC criterion:
    2,VEV     3,VEV     2,VVV
-561.7285 -562.5522 -574.0178
```

Example

```
> mruns <- Mclust(scale(iris[,-5]), G=1:5); t(mruns$BIC)
Bayesian Information Criterion (BIC):
            1          2          3          4          5
EII -1723.766 -1344.0530 -1214.5246 -1177.1625 -1135.6417
VII -1723.766 -1325.2731 -1222.8439 -1139.3923 -1107.7910
EEI -1738.798 -1259.6457 -1029.7330 -1044.1118  -958.6219
VEI -1738.798 -1172.9600  -995.8343  -965.1348  -905.0241
EVI -1738.798 -1223.9860 -1014.5146 -1054.2229  -983.5054
VVI -1738.798 -1074.2293  -961.3205  -967.7021  -928.1301
EEE -1046.656  -904.7750  -849.6448  -862.7323  -821.5159
EVE -1046.656  -906.9282  -862.9077  -902.9964  -901.9539
VEE -1046.656  -873.0048  -822.0760  -821.5149         NA
VVE -1046.656  -880.7933  -828.8731  -881.1270  -853.4148
EEV -1046.656  -873.6590  -875.3724  -930.1693  -877.2201
VEV -1046.656  -802.5253  -797.5483  -821.4057  -837.6111
EVV -1046.656  -875.0084  -872.7515  -941.9898         NA
VVV -1046.656  -790.6956  -797.5190  -847.2777  -893.2973

Top 3 models based on the BIC criterion:
    2,VVV     3,VVV     3,VEV
-790.6956 -797.5190 -797.5483
```

Example

```
> table(iris$Species, mrun$classification)

            1  2
  setosa     50  0
  versicolor  0 50
  virginica   0 50
> mrun3 <- Mclust(iris[,-5], 3)
> table(iris$Species, mrun3$classification)

            1  2  3
  setosa     50  0  0
  versicolor  0 45  5
  virginica   0  0 50
> mrun4 <- Mclust(iris[,-5], 4)
> table(iris$Species, mrun4$classification)

            1  2  3  4
  setosa     38 12  0  0
  versicolor  0  0 45  5
  virginica   0  0  0 50
```

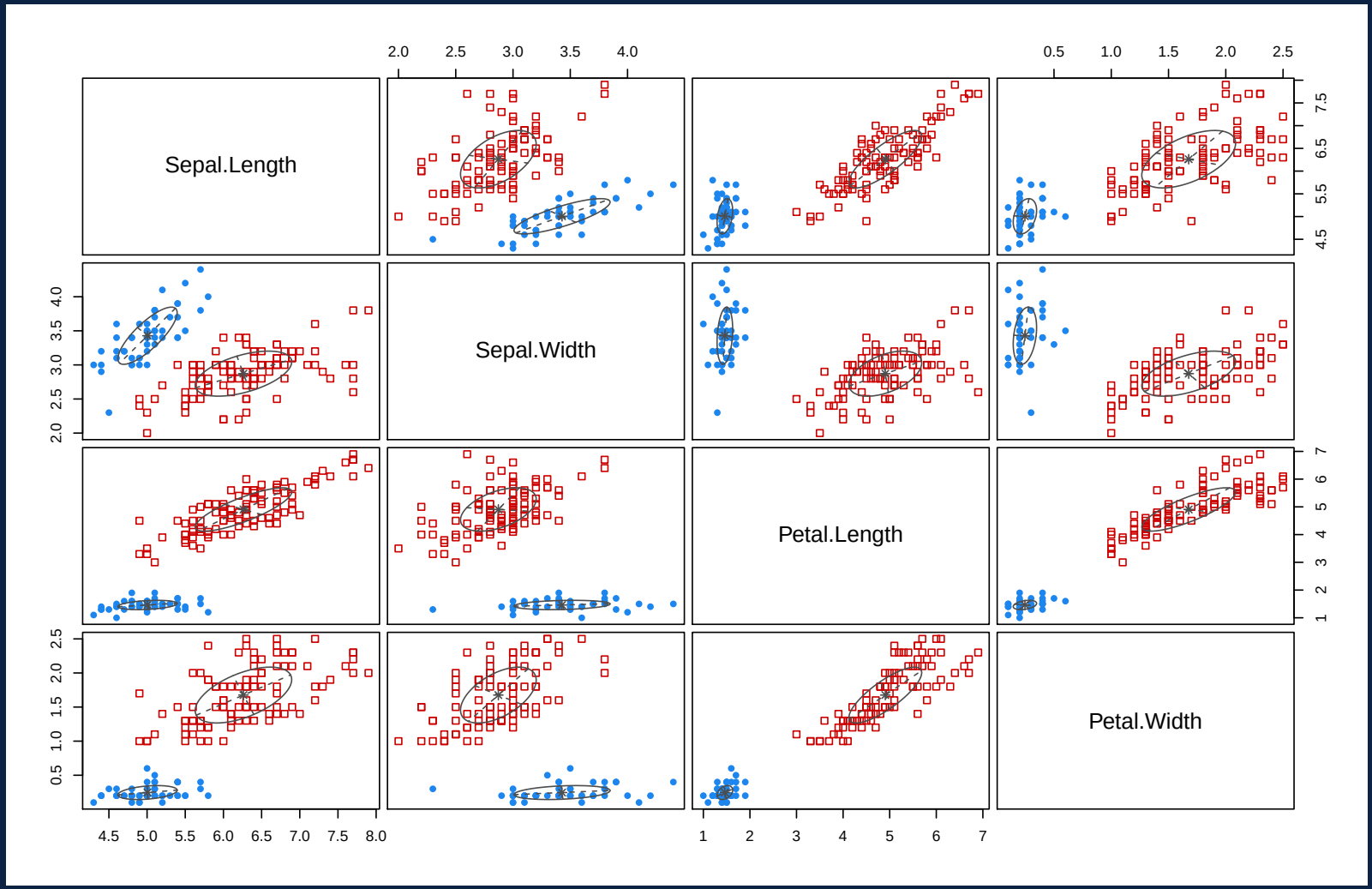
Example

```
> adjustedRandIndex(iris$Species, mrun$classification)
[1] 0.5681159
```

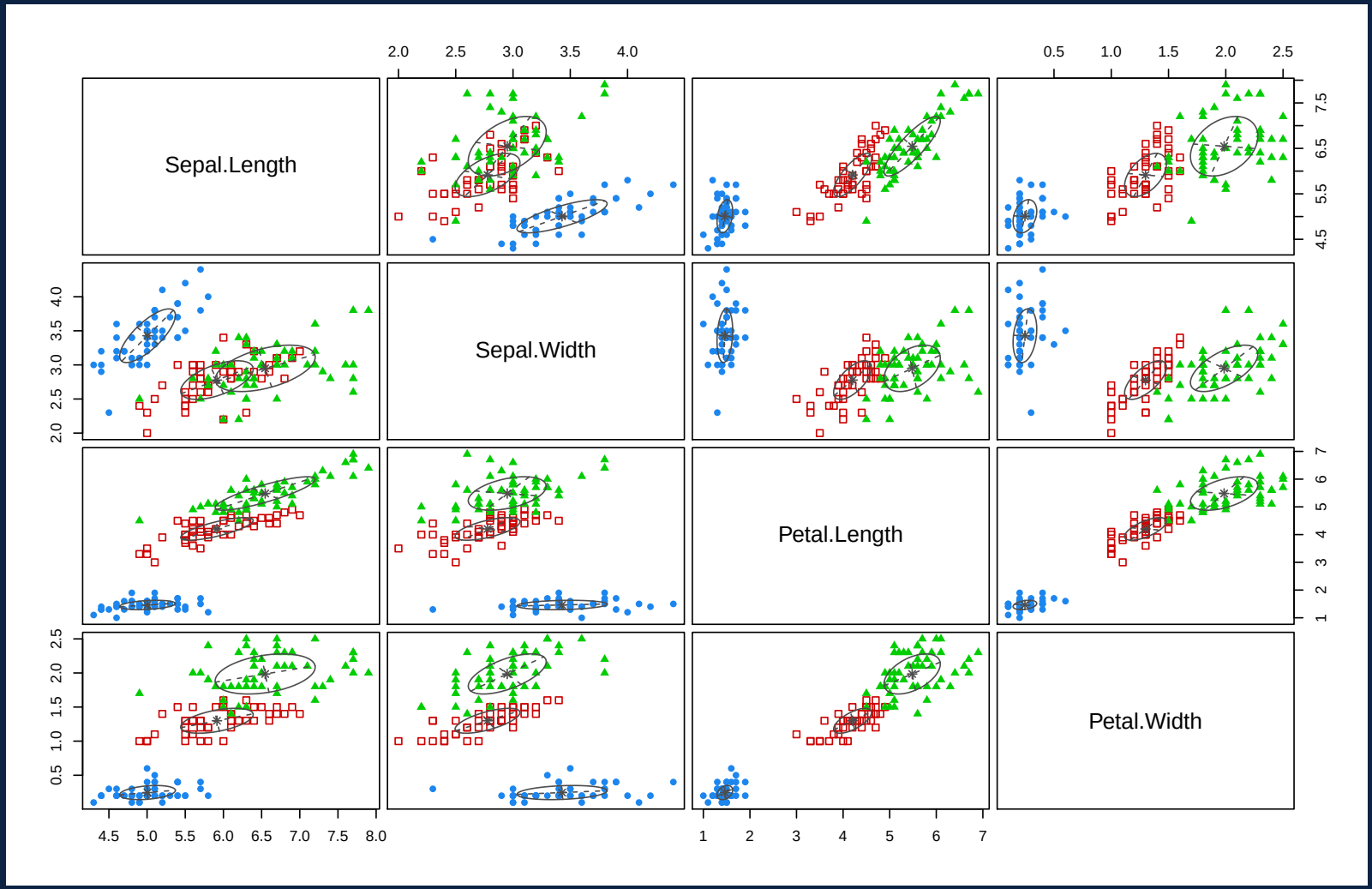
```
> adjustedRandIndex(iris$Species, mrun3$classification)
[1] 0.9038742
```

```
> adjustedRandIndex(iris$Species, mrun4$classification)
[1] 0.8054484
```

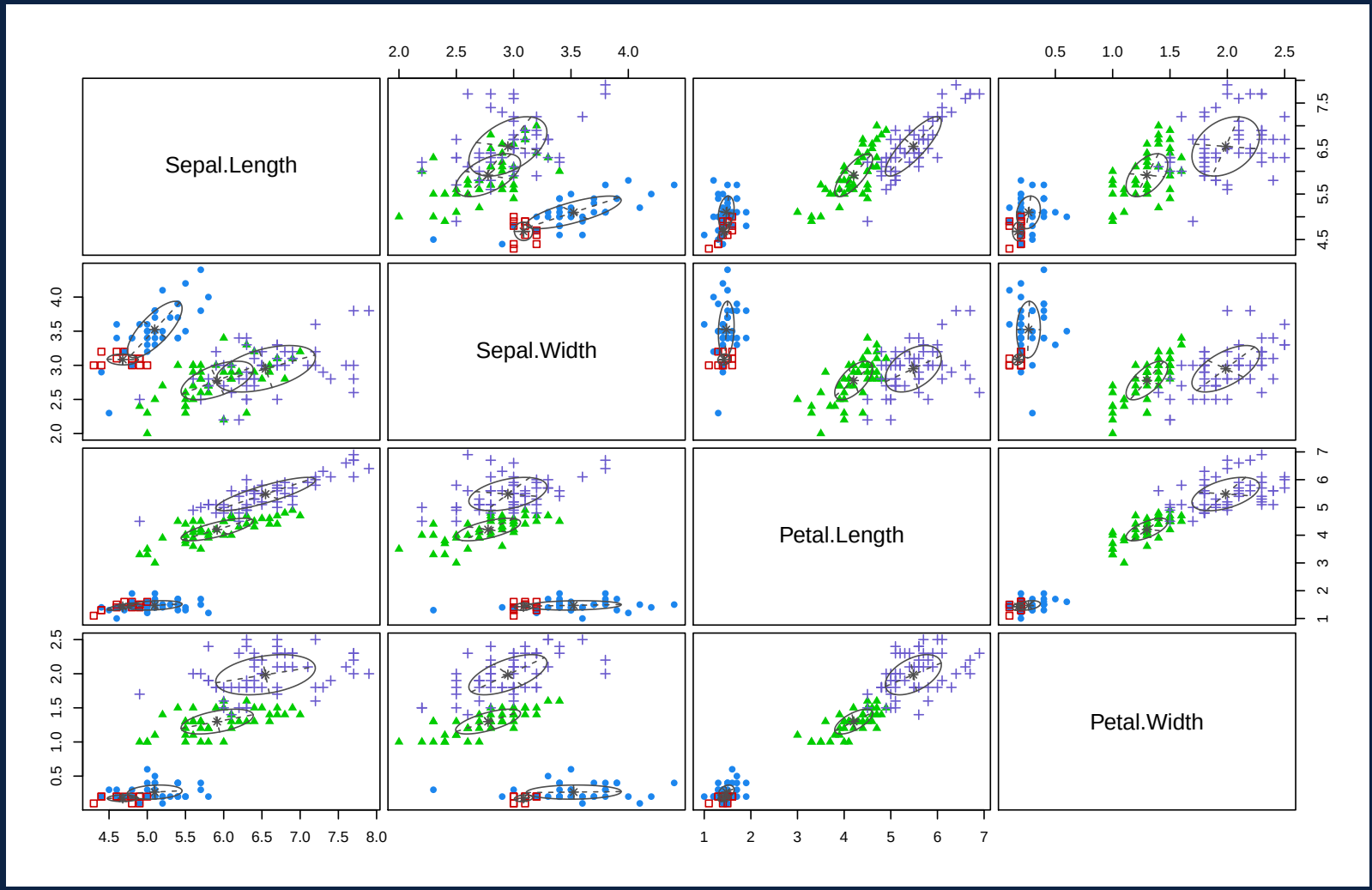
2 groups



3 groups


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 5.5 6.5 7.5
2.0 2.5 3.0 3.5 4.0
1 2 3 4 5 6 7
0.5 1.0 1.5 2.0 2.5
4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
1 2 3 4 5 6 7

4 groups


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 5.5 6.5 7.5
2.0 2.5 3.0 3.5 4.0
1 2 3 4 5 6 7
0.5 1.0 1.5 2.0 2.5
4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0
1 2 3 4 5 6 7

On unsupervised metrics

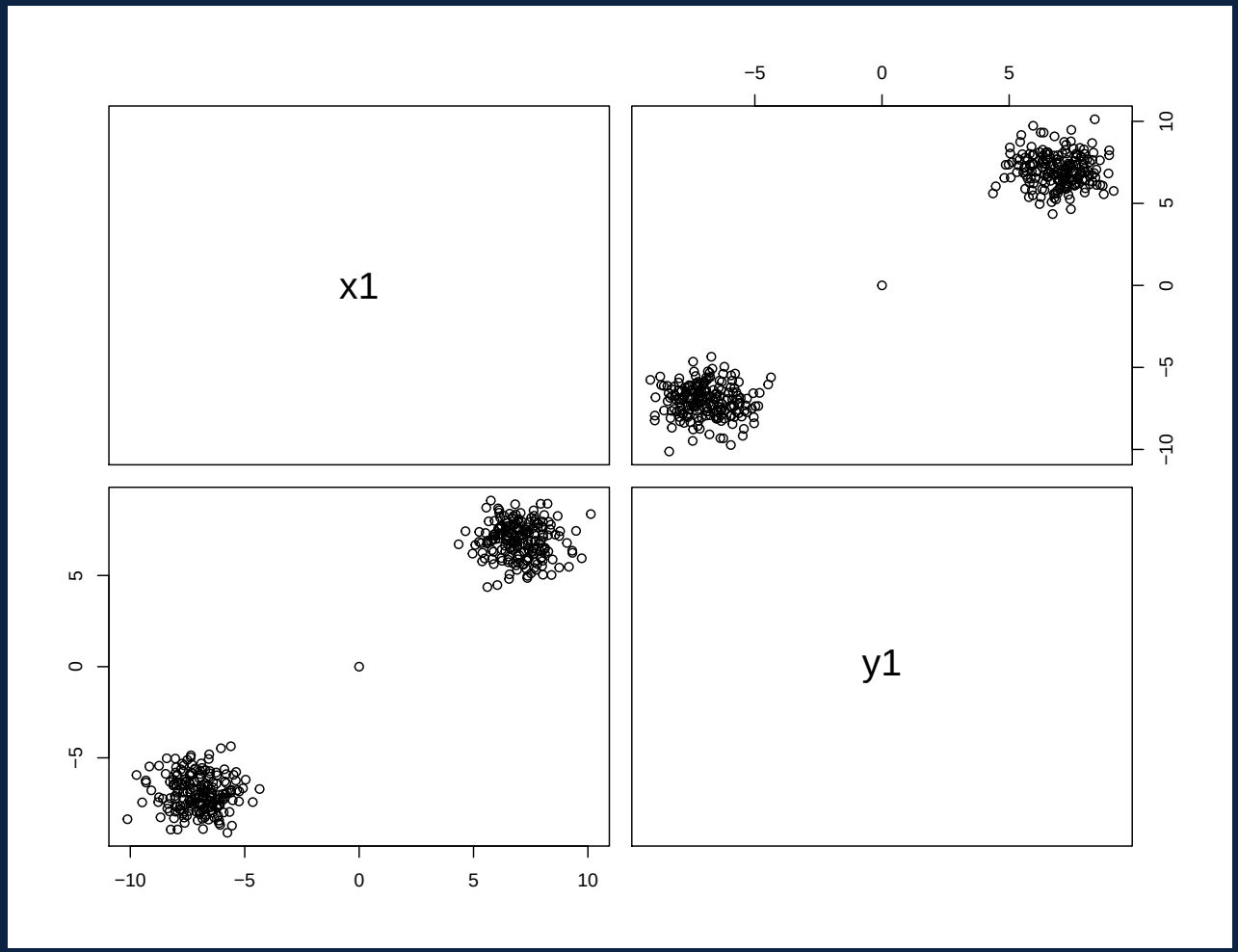
- Unsupervised learning is a funny modelling space.
- We have emphasized constantly the need for cross-validation in the supervised realm
- Since unsupervised methods are not optimizing using a specific response variable, it is certainly less of a concern — which means it is often overlooked.
- In other words, the argument is that since we haven't used the Iris species variable while fitting, it is unlikely that an overly flexible model will fit too closely to it and give us an overconfident view in recovering those groups.

On unsupervised metrics

- This is not to say that bias and variance are not a concern in unsupervised learning.
- High variance (flexible) models will tend to give drastically different groupings with new samples, and high bias (simple) models will tend to assume incorrect group structures.
- It's worth noting here that I personally do believe that overfitting is a problem in unsupervised learning[1], and that forms of cross-validation or resampling might serve as a fix.
- But it's harder to see, and in practice harder to know if it's going on. Let's look at a simulation (taken from that paper) that illustrates that it is going on...

[1] Andrews, JL (2018). "Addressing overfitting and underfitting in Gaussian model-based clustering". *Computational Statistics and Data Analysis*

Motivating Example - Bivariate

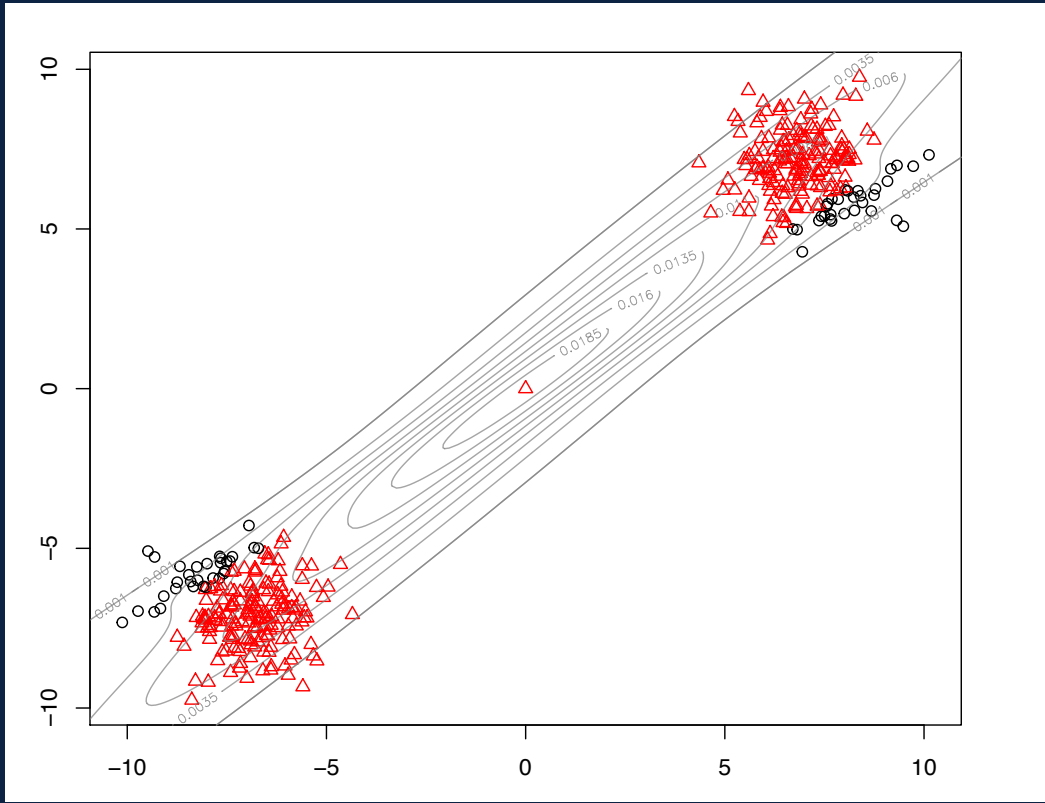


Classification at middle point, and certainty?

Bivariate

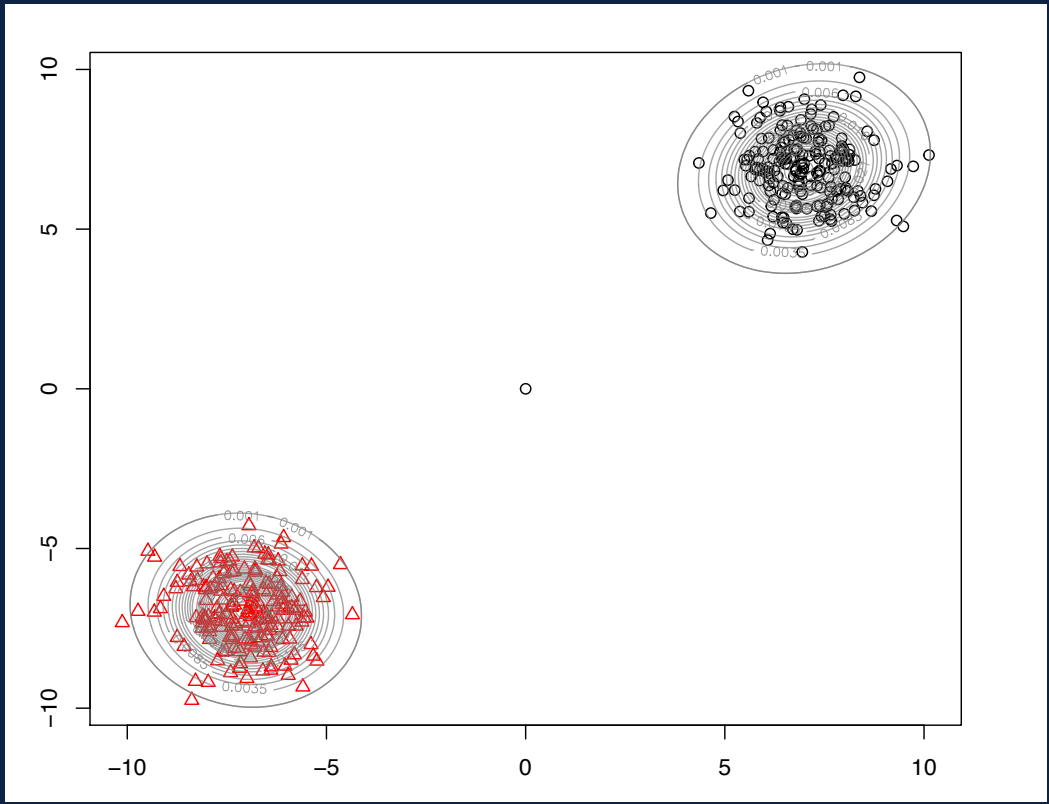
- The two groups of data are exact mirror images (along Y=-X) of one another, and the middle point placed precisely at (0,0).
- Thus, no sampling error here between the groups.
- So what happens when fitting a model on a dataset that has a point that should intuitively have (.5, .5) probabilities?
- Broadly, three possible results...

Solution 1 - EM Convergence to Bad Local Maxima



- Log-likelihood: -2068
- Common solution from random $\hat{z}_{ig}$ initialization

Solution 2 - EM Convergence to Global Maxima



- Log-likelihood: -1436
- Common solution from a 'good' initialization (like k-means, etc)

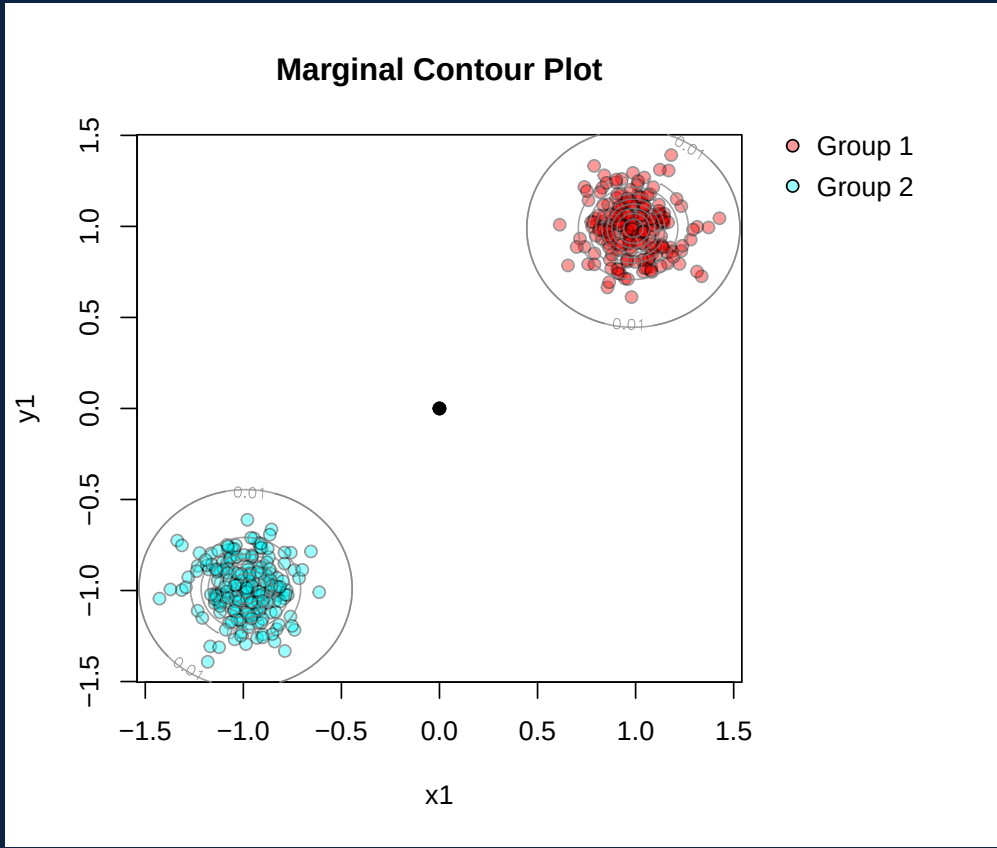
Problem with Global Max?

```
> mfit <- Mclust(dat2, G=2, modelNames="VVV")
> mfit$z[401,]
[1] 1.000000e+00 5.274169e-09
```

```
> tfit <- teigen(dat2, Gs=2, models="UUUU")
> tfit$fuzzy[401,]
[1] 9.999999e-01 8.674189e-08
```

- That is, essentially probability 1 of being in the first group, regardless of Gaussian vs t.

Solution 3 - EM Convergence to Good Local Maxima



- Log-likelihood: -1438
- Only found when initializing extremely close to final solution


THE UNIVERSITY OF BRITISH COLUMBIA