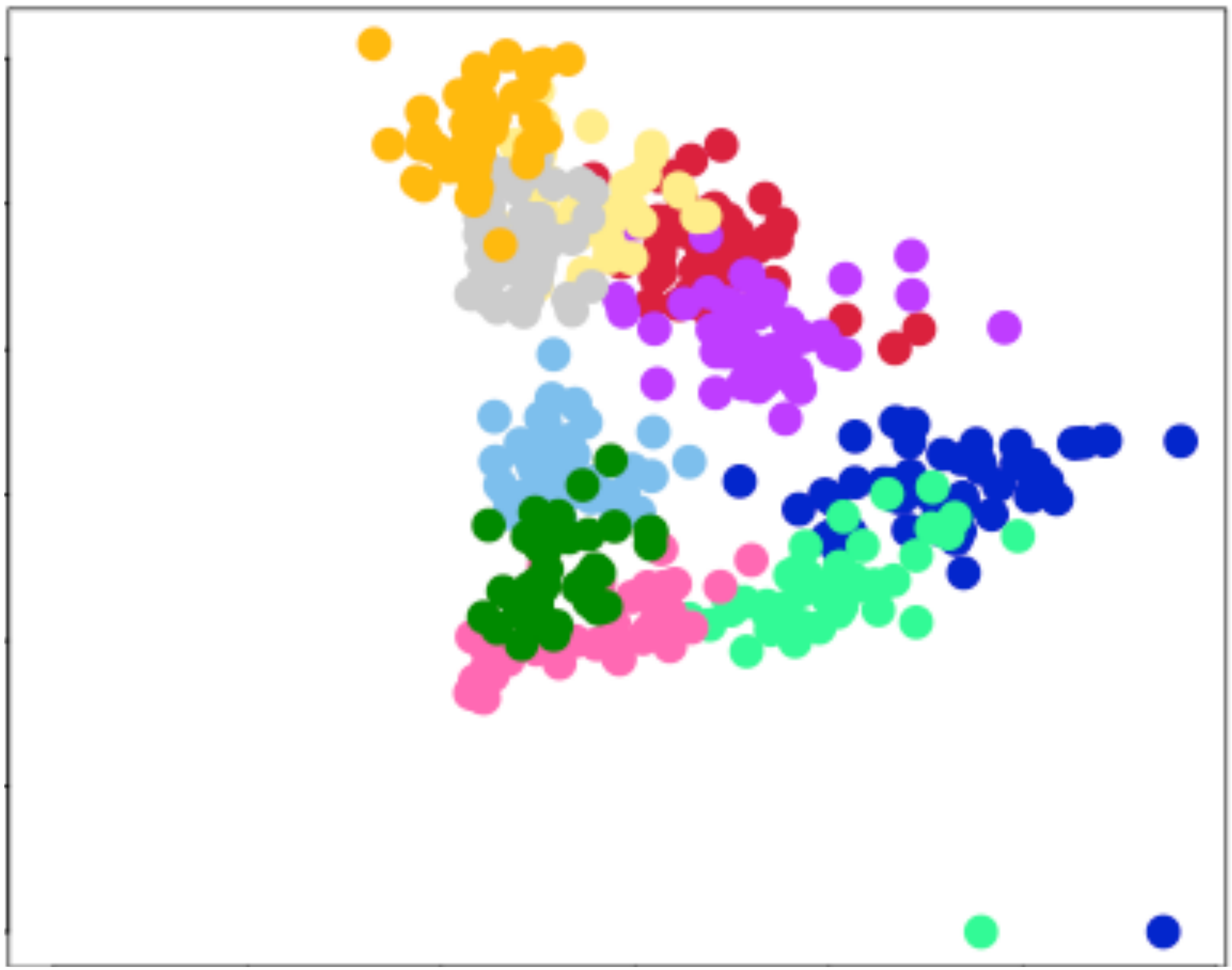
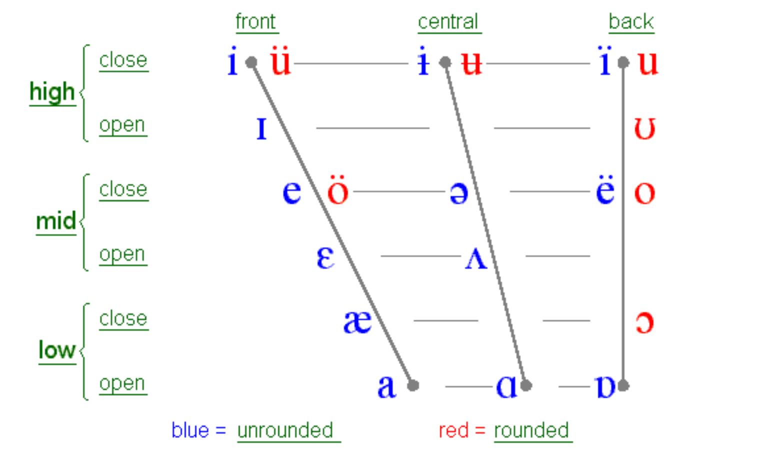
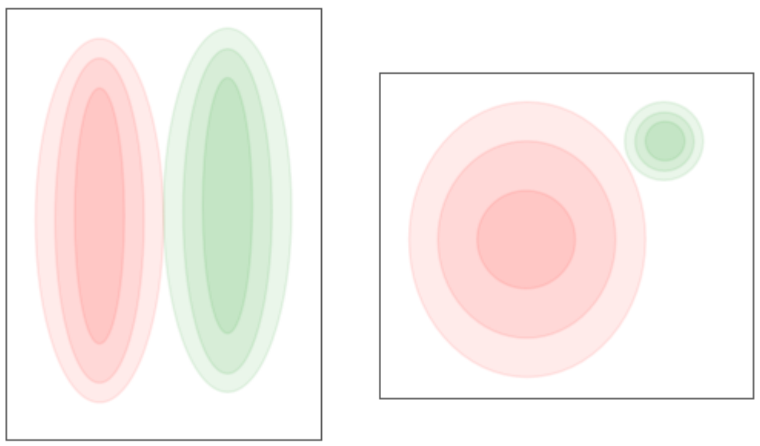
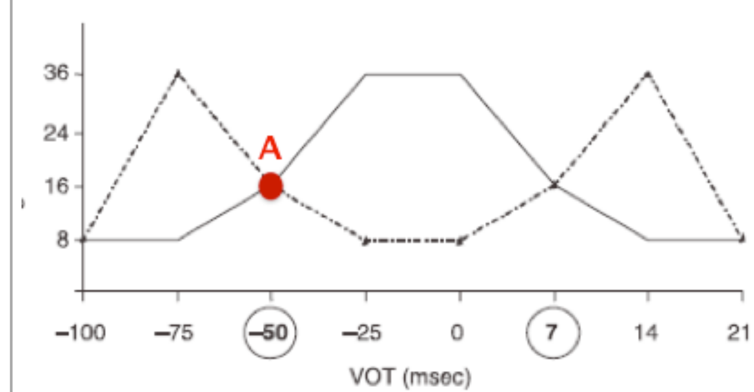
Computational Cognitive Science









Lecture 8: Unsupervised classification

Lecture outline

- Unsupervised classification
  - Case study: phoneme learning in language
- A first try: k-means clustering
  - Limitations and an extension
- Next try: Mixture of Gaussians
  - EM model for calculating
- Next: Semi-supervised classification

both fairly analogous
to prototype models

Lecture outline

- ➡ Unsupervised classification
  - Case study: phoneme learning in language
- A first try: k-means clustering
  - Limitations and an extension
- Next try: Mixture of Gaussians
  - EM model for calculating
- Next: Semi-supervised classification

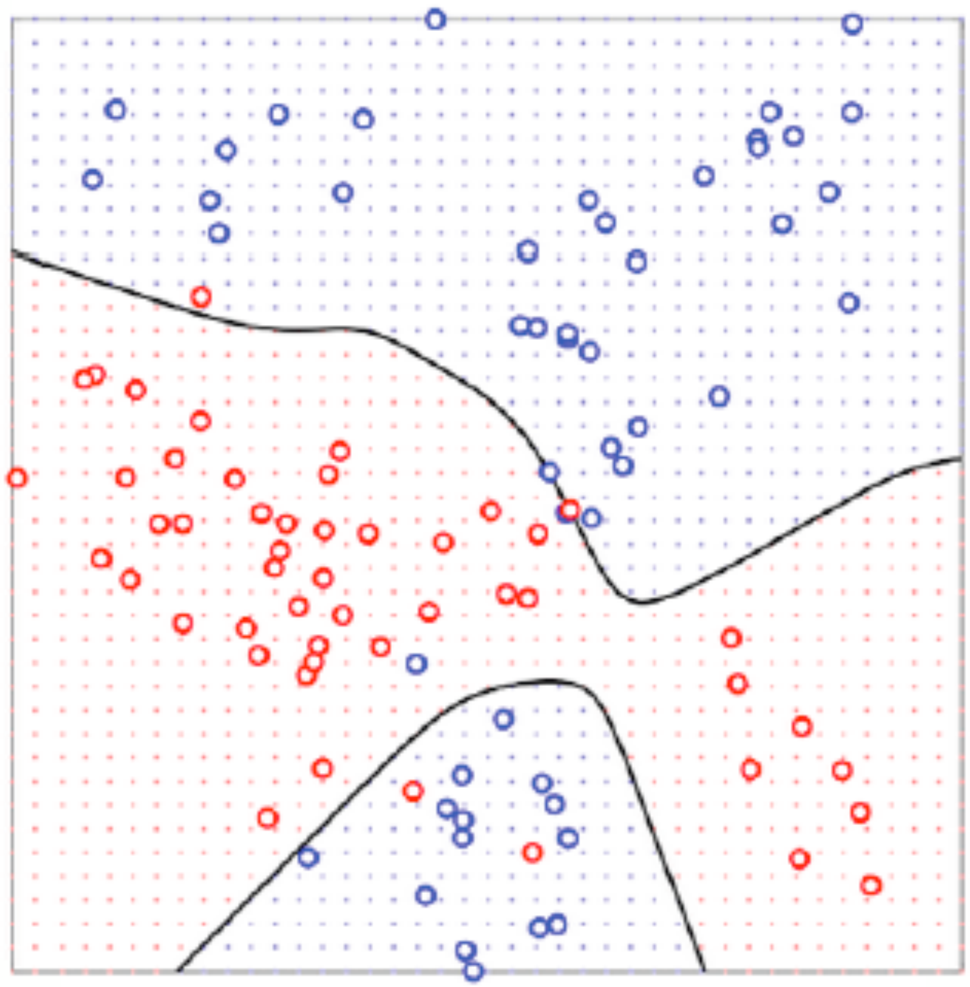
Where we stand so far

- So far we've been introduced to the problem of classification

observation x

label $\ell(x)$

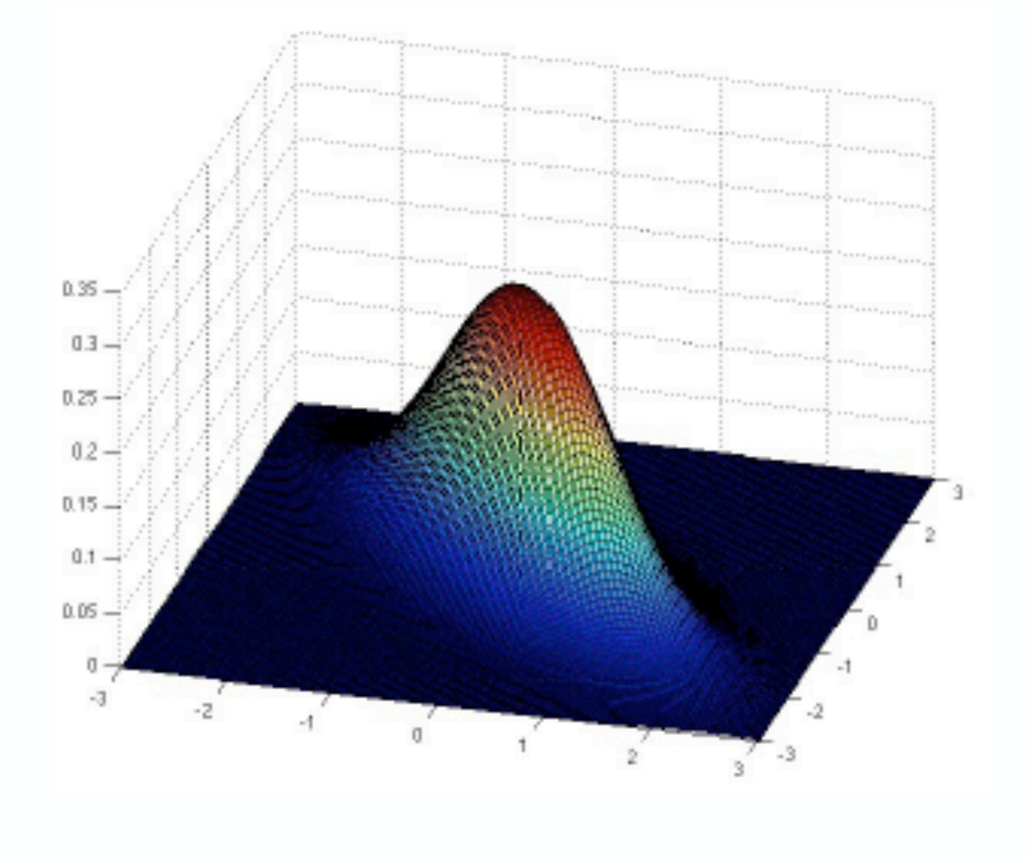
predict the label of other observations y



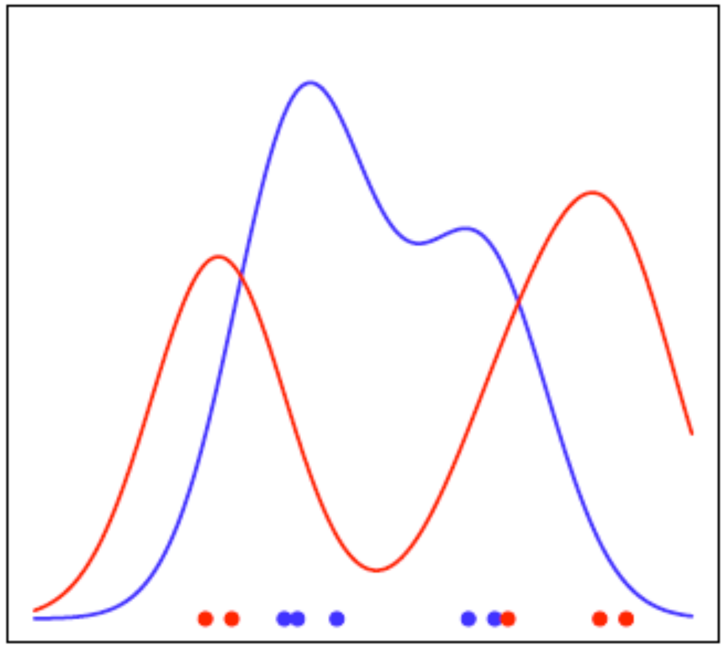
Where we stand so far

- So far we've been introduced to the problem of classification
- We've seen some possible models...

Categories are (Gaussian) probability distributions



Categories are estimated by combining "kernels" around each observation

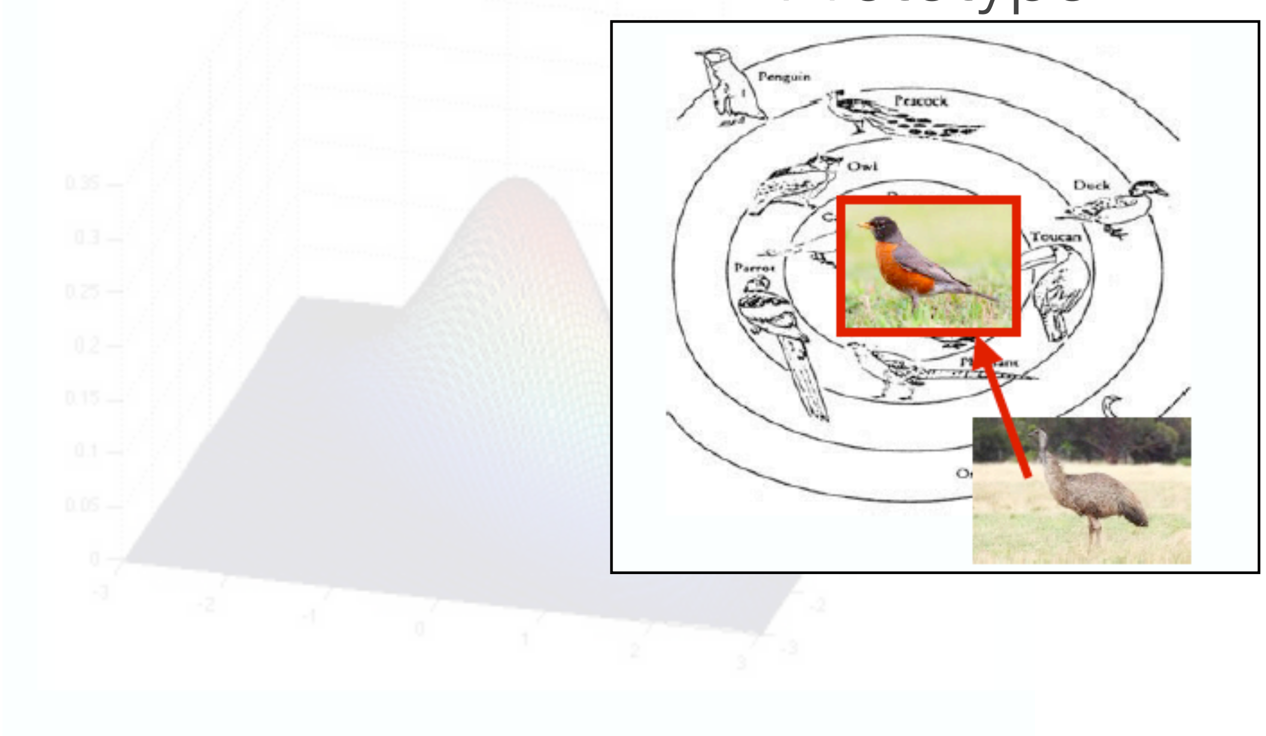


Where we stand so far

- So far we've been introduced to the problem of classification
- We've seen some possible models... and how they link to psychological theory

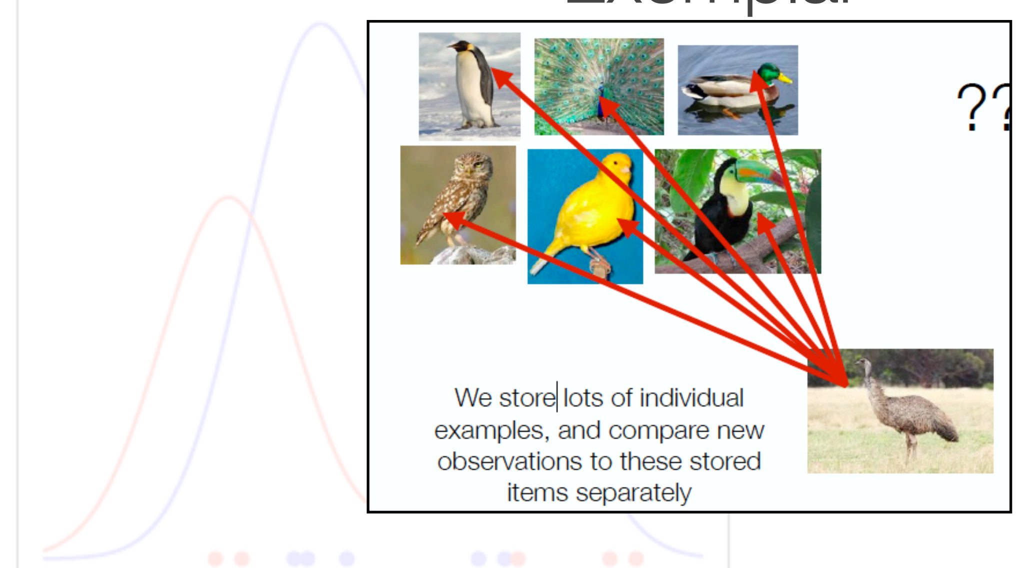
Categories are (Gaussian) probability distributions

Prototype



Categories are estimated by combining "kernels" around each observation

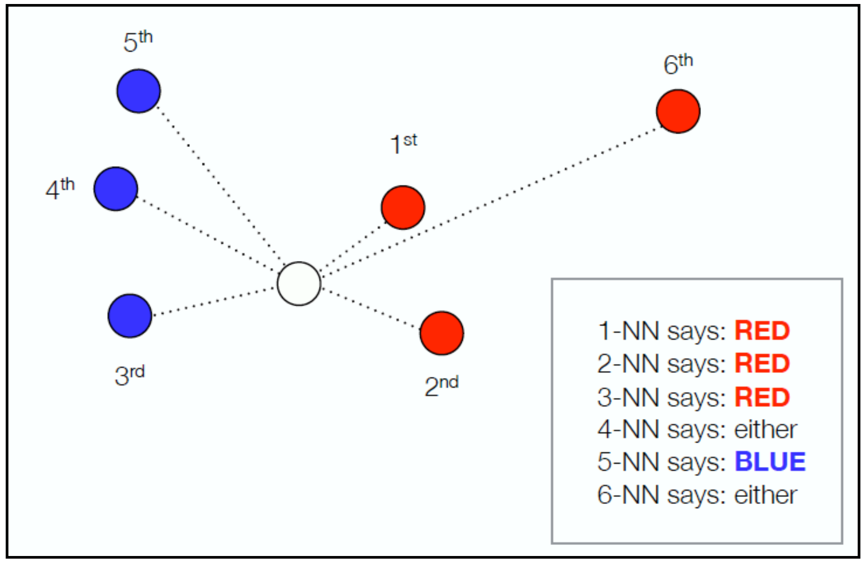
Exemplar

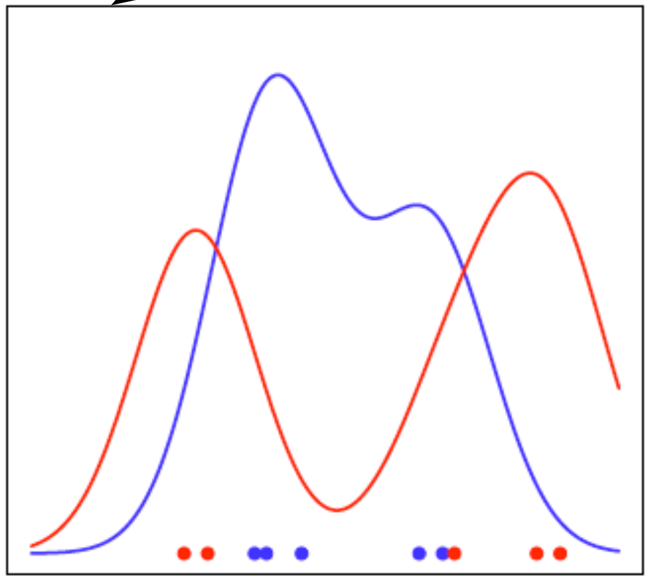


Where we stand so far

- So far we've been introduced to the problem of classification
- We've seen some possible models... and how they link to psychological theory
- But in all of these examples we've assumed that everything is labeled!

Indeed, many of the models require it!





A problem: This doesn't describe real life

- Most of the time things are at most semi-supervised; only some things are labelled

A problem: This doesn't describe real life

- And some things are never labelled (by definition!)
- For instance: sound categories in a language

Phonemes: units of sound (consonants or vowels) in a language. Shortest segment of speech that distinguishes two words

b

bed

h

head

t

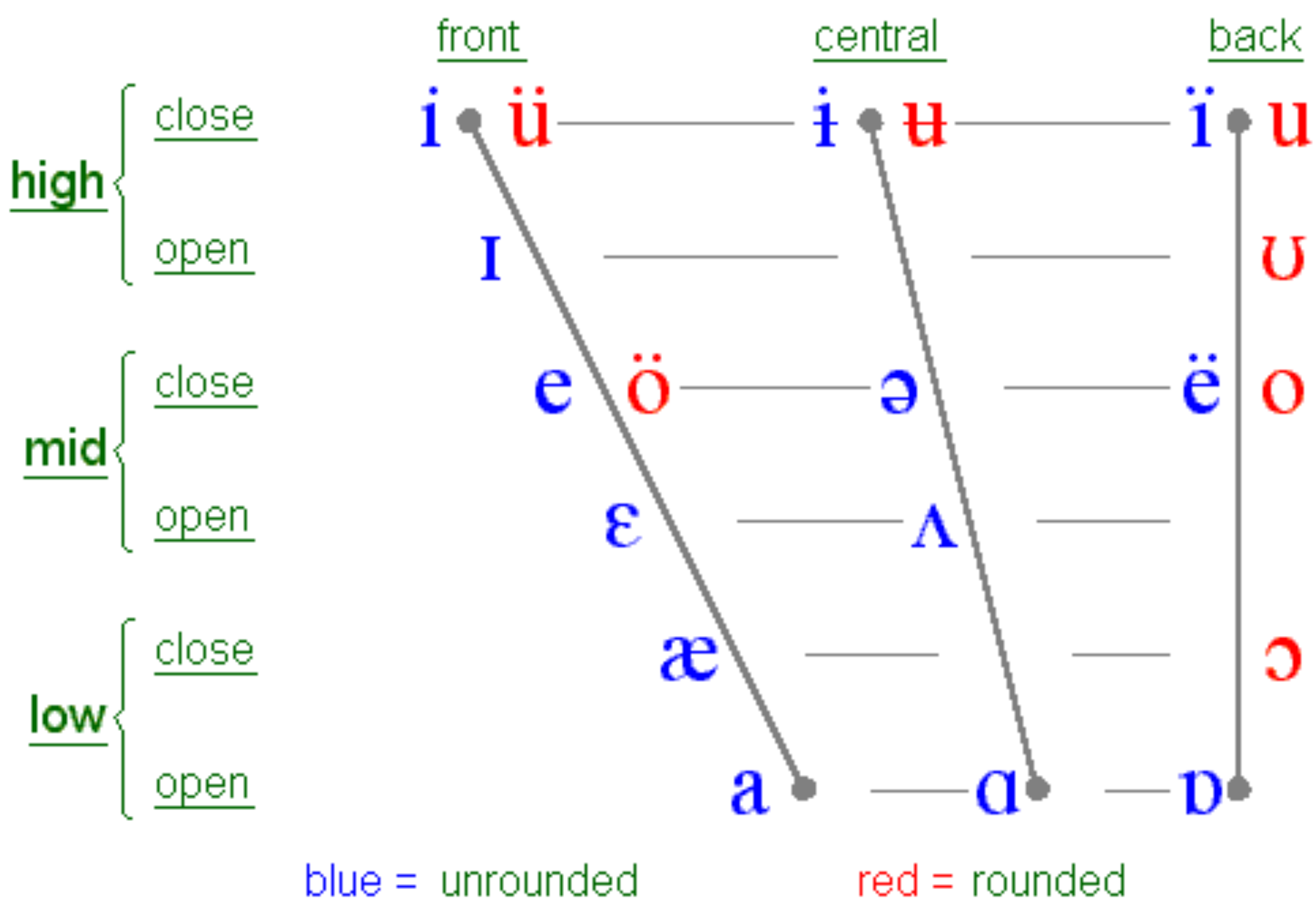
tower

sh

shower

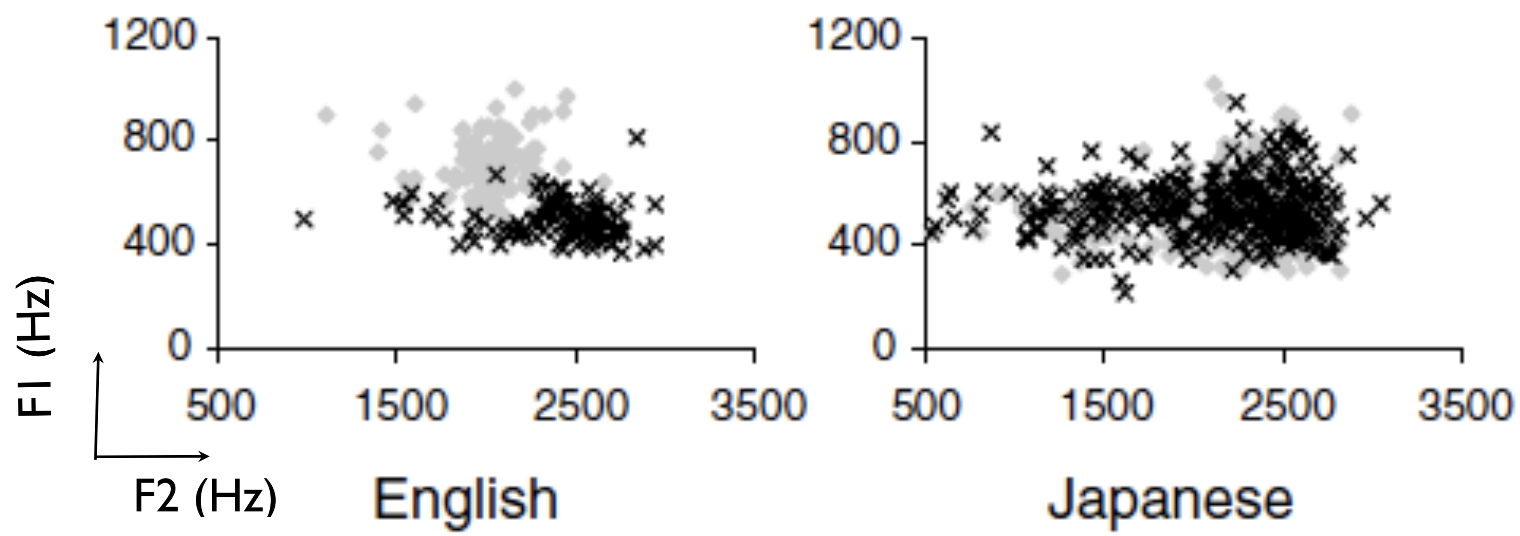
Unsupervised classification: Phonemes

Vowels: sounds where the air is not blocked,
classified by the shape of the mouth

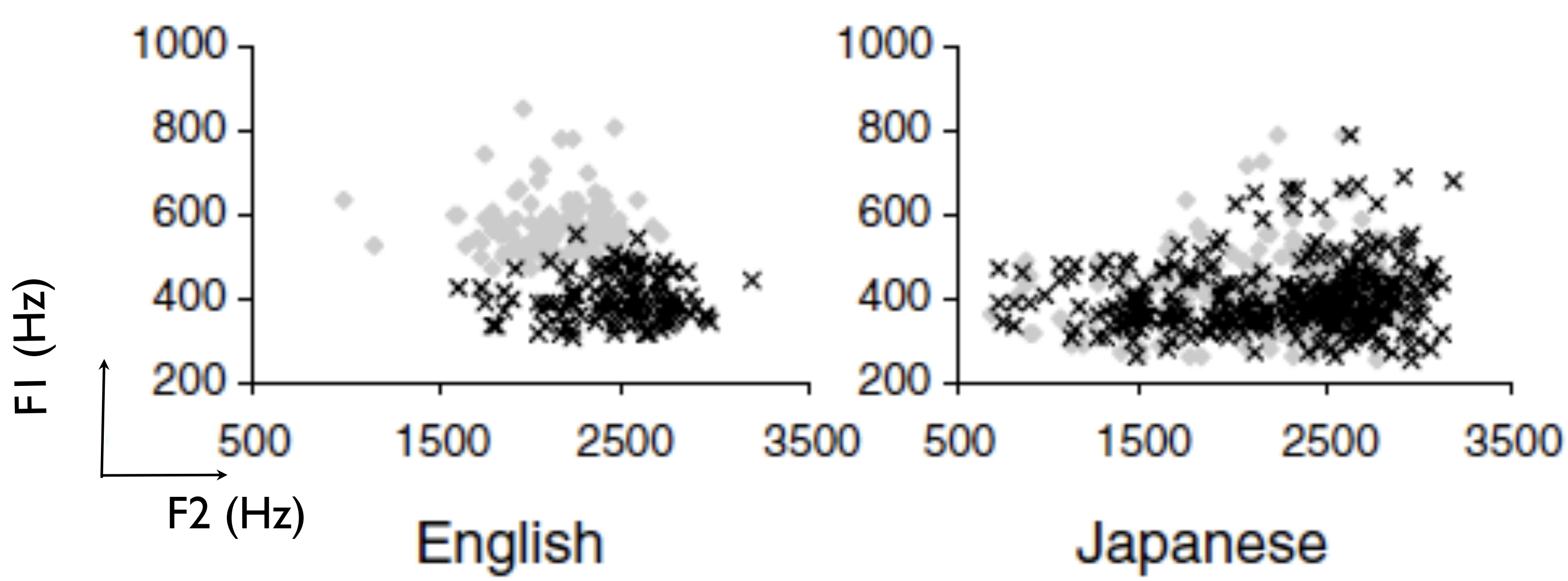


Unsupervised classification: Phonemes

Phoneme categories differ across languages



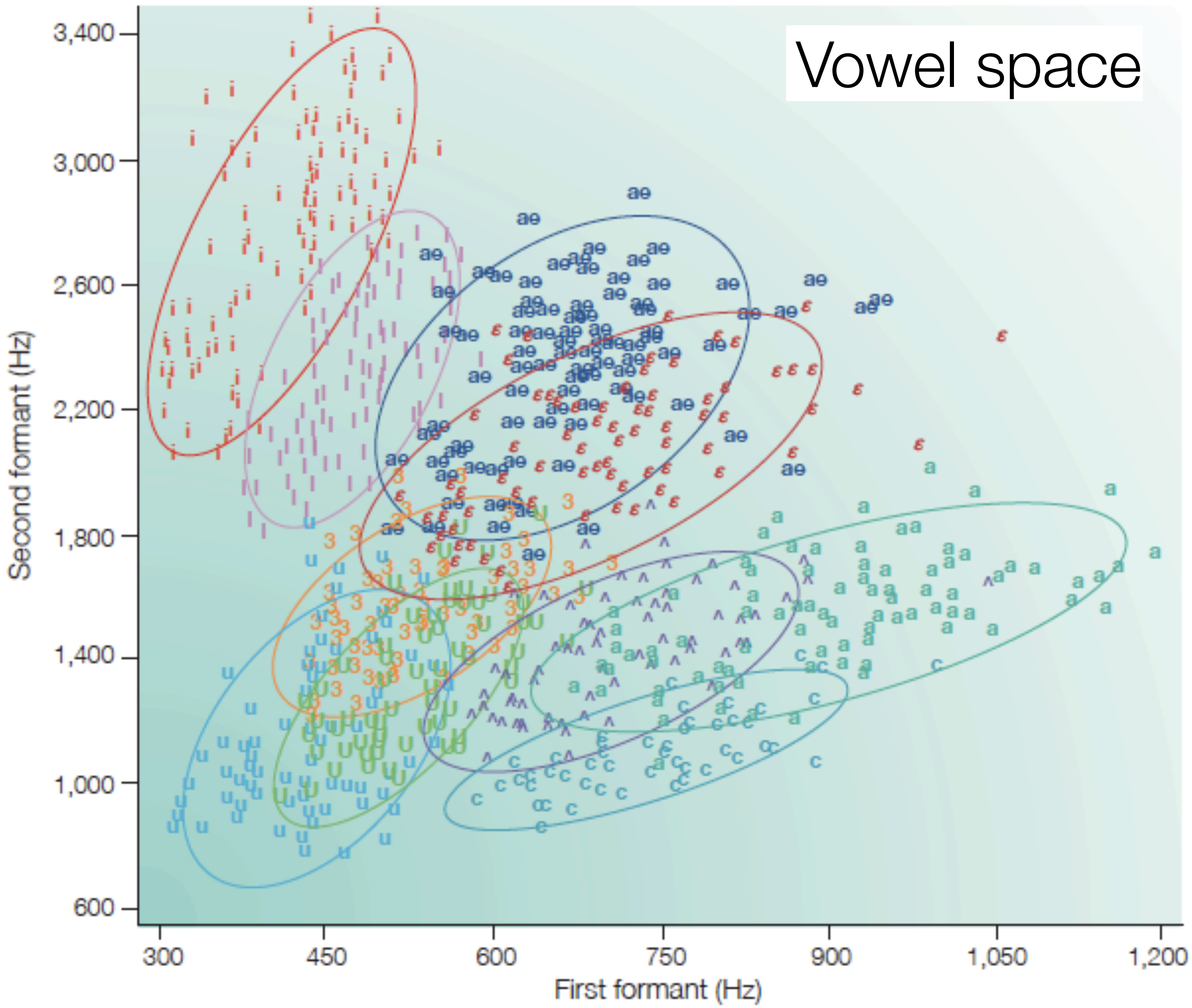
/e/ - /eɪ/
bet - bait



/ɪ/ - /ɪː/
bit - beat

Unsupervised classification: Phonemes

Phoneme categorisation (in any language) is very difficult!



High variability due to:

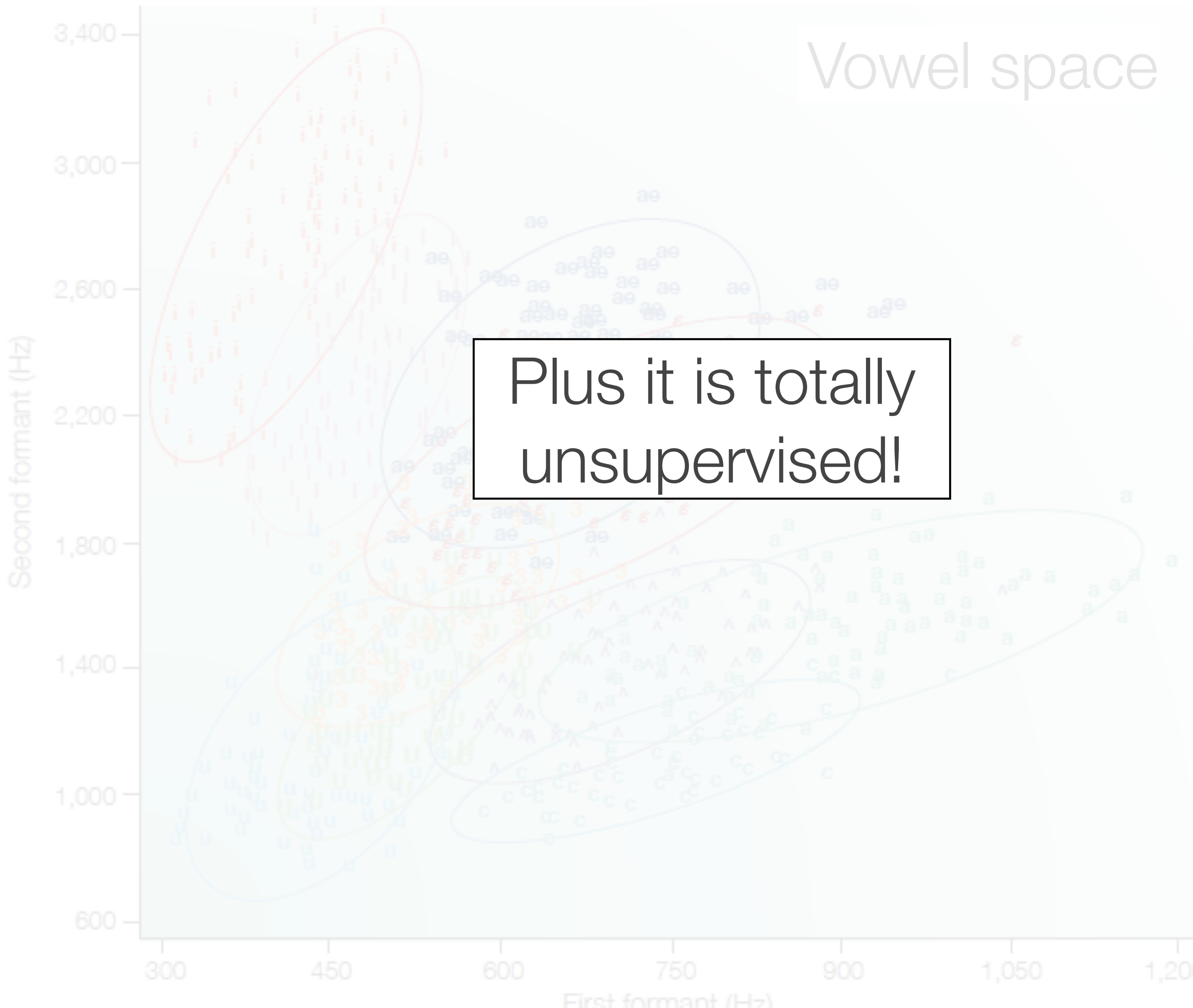
Speaker differences

Intonation

Context (surrounding phonemes)

Unsupervised classification: Phonemes

Phoneme categorisation (in any language) is very difficult!



Plus it is totally unsupervised!

High variability due to:

Speaker differences

Intonation

Context (surrounding phonemes)

Unsupervised classification: Phonemes

Phoneme categorisation (in any language) is very difficult!

Main question: How do people (children) learn the phoneme categories appropriate to their language?

Possible answer: They use *distributional information* (info about how the sounds are distributed)... which you can get just by listening (plus a reasonable mechanism for unsupervised learning)

Do people actually learn this way?

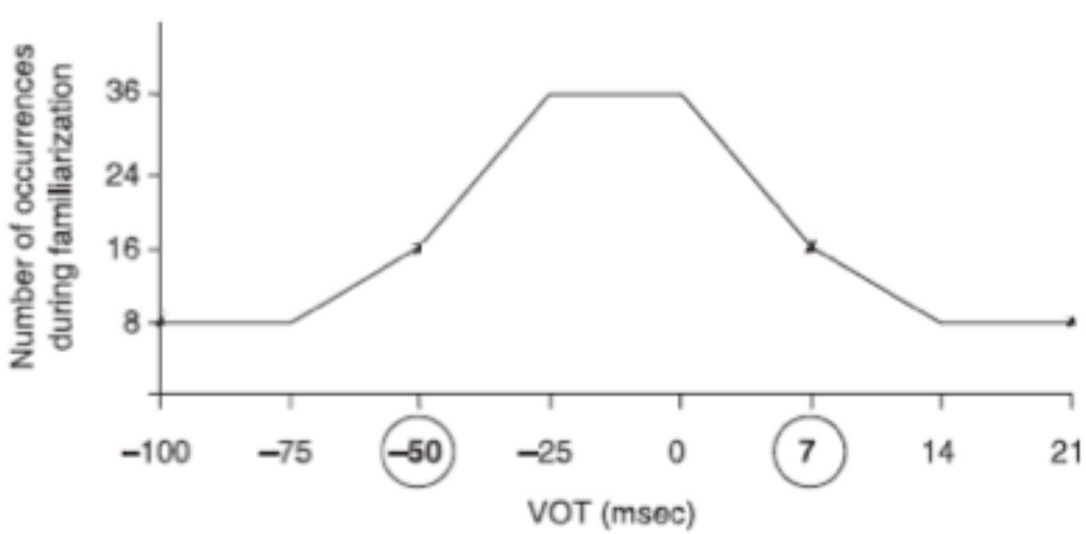
What algorithm might do this?

Do people actually learn distributionally?

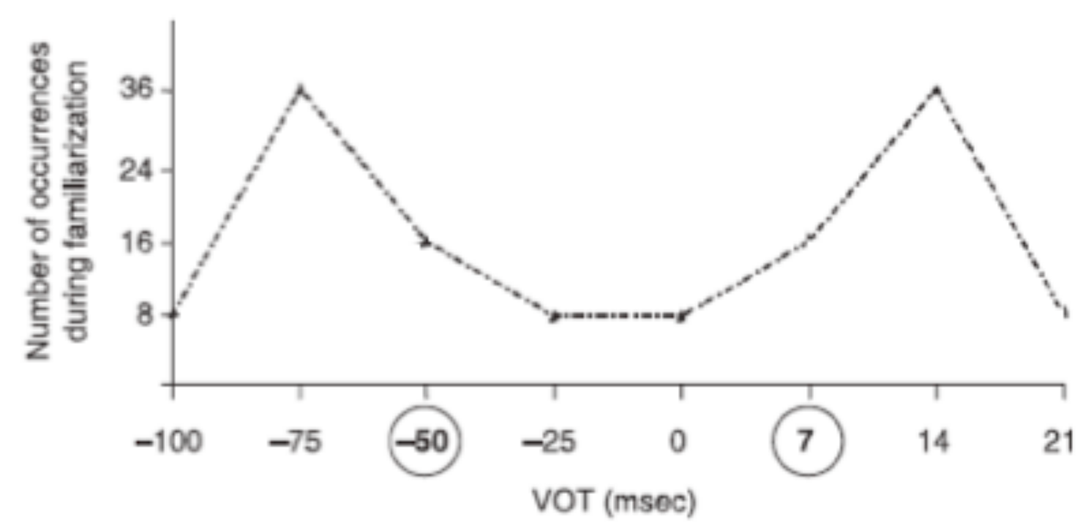
Yes, it seems they do.

Experimental test (infants): Present them with different distributions of sounds...

Unimodal: Should learn one phoneme



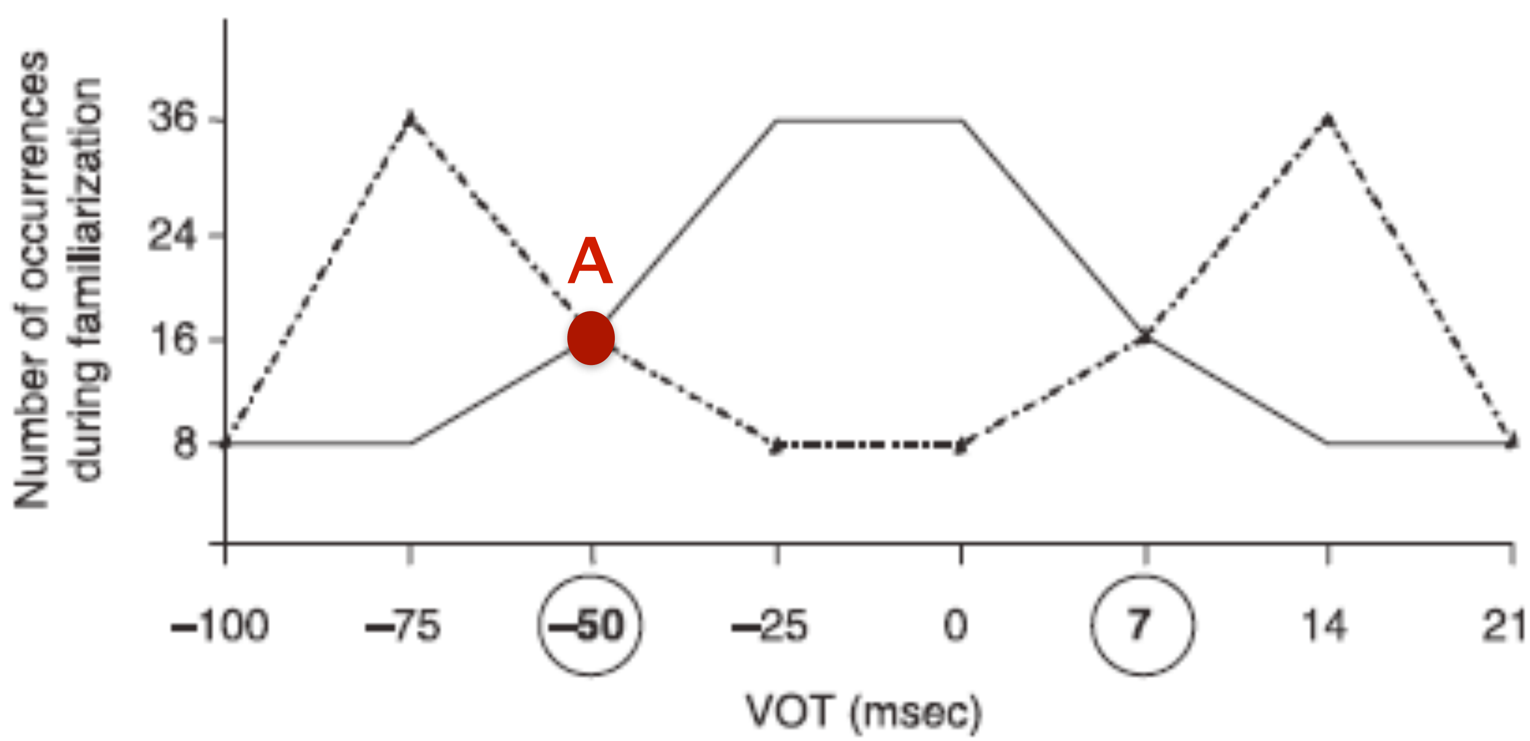
Bimodal: Should learn two phonemes



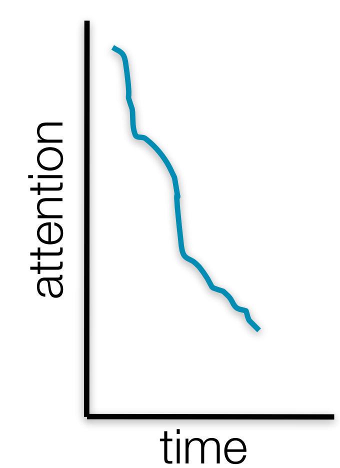
Do people actually learn distributionally?

Yes, it seems they do.

Experimental test (infants): Then, after they've heard that distribution, have them listen to one sound over and over until they get bored (habituated)



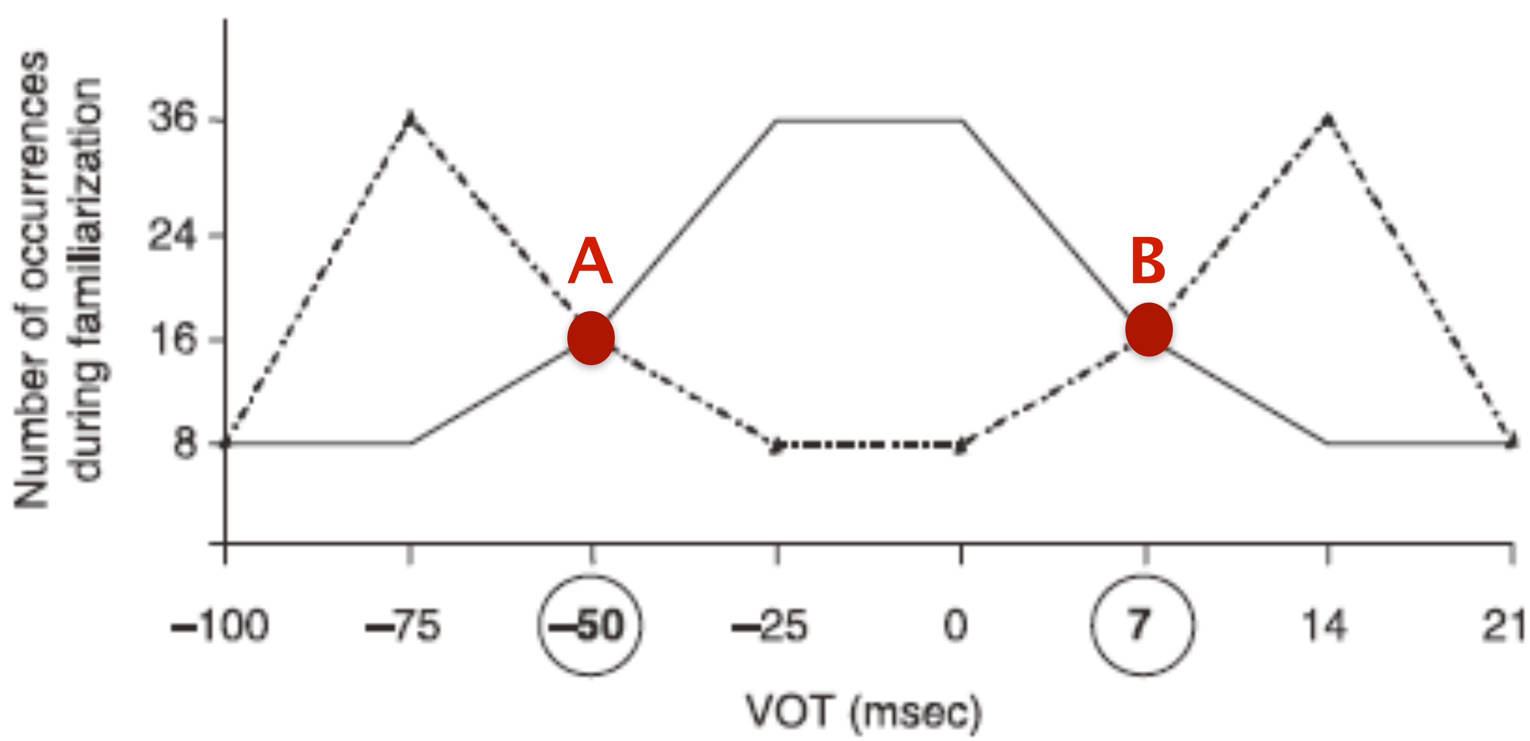
Habituation: play "A" repeatedly



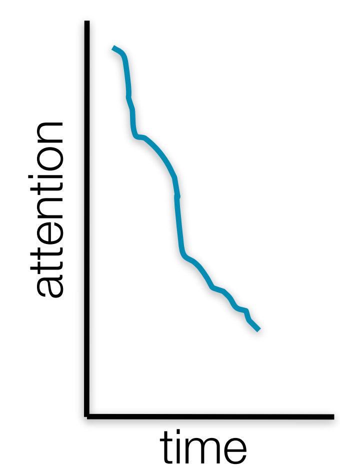
Do people actually learn distributionally?

Yes, it seems they do.

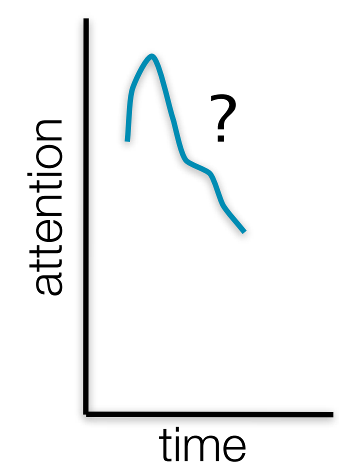
Experimental test (infants): Test on another sound. If they think it is one underlying category, they should remain bored. If they think it is two, they should get interested again.



Habituation: play "A" repeatedly



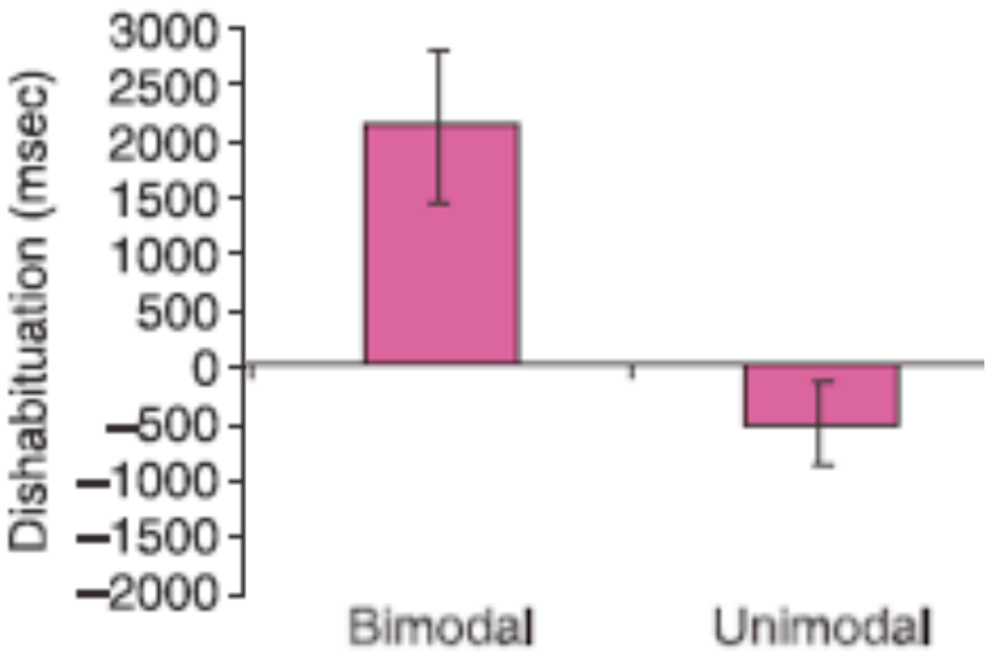
Test: play "B" repeatedly



Do people actually learn distributionally?

Yes, it seems they do.

Experimental result (infants): They *do* get interested again in the bimodal condition, but not the unimodal. So they must be learning the underlying distribution



Do people actually learn distributionally?

Yes, it seems they do.

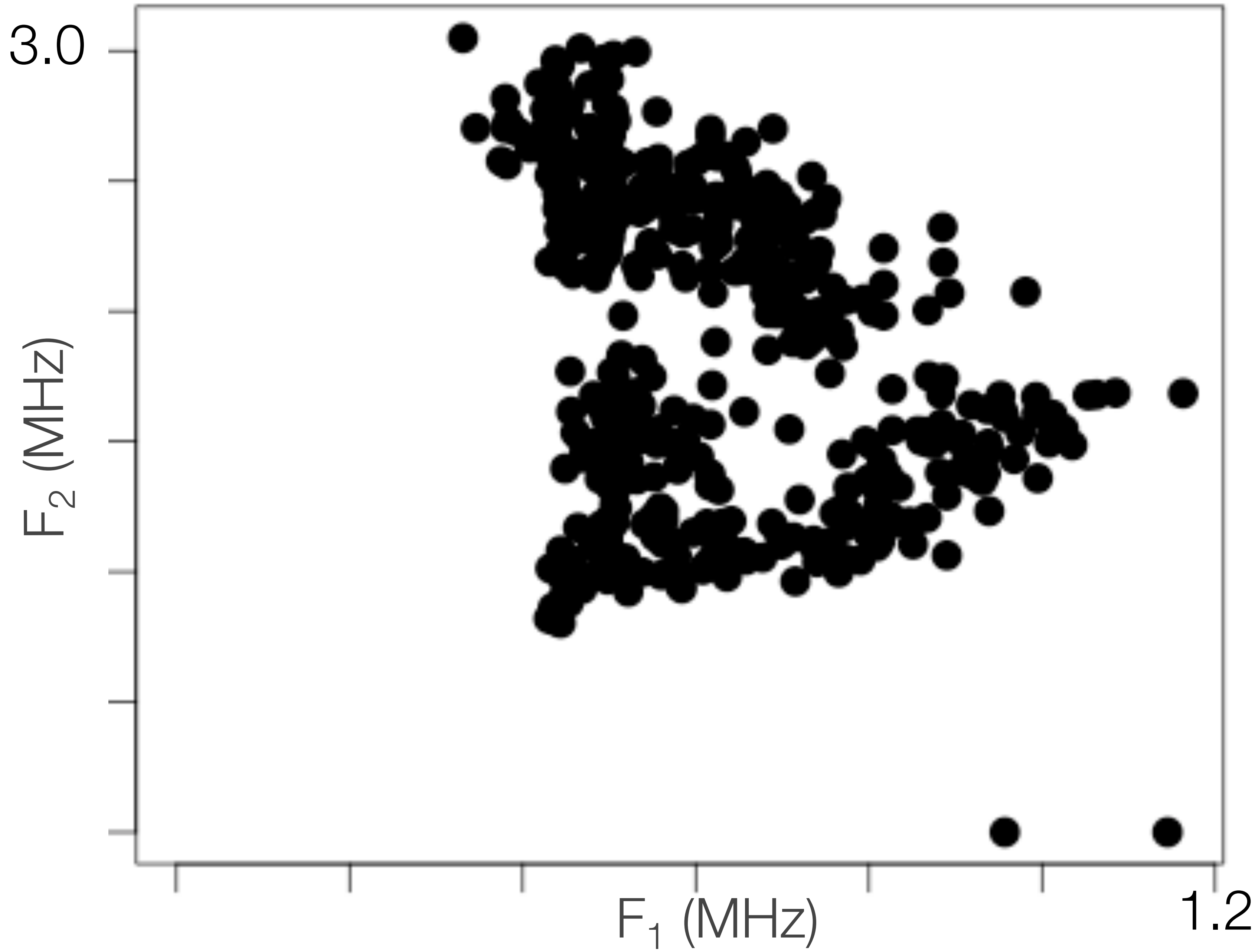
Experimental result (infants): They *do* get interested again in the bimodal condition, but not the unimodal. So they must be learning the underlying distribution

How?

Lecture outline

- Unsupervised classification
  - Case study: phoneme learning in language
- ➡ A first try: k-means clustering
  - Limitations and an extension
- Next try: Mixture of Gaussians
  - EM model for calculating
- Next: Semi-supervised classification

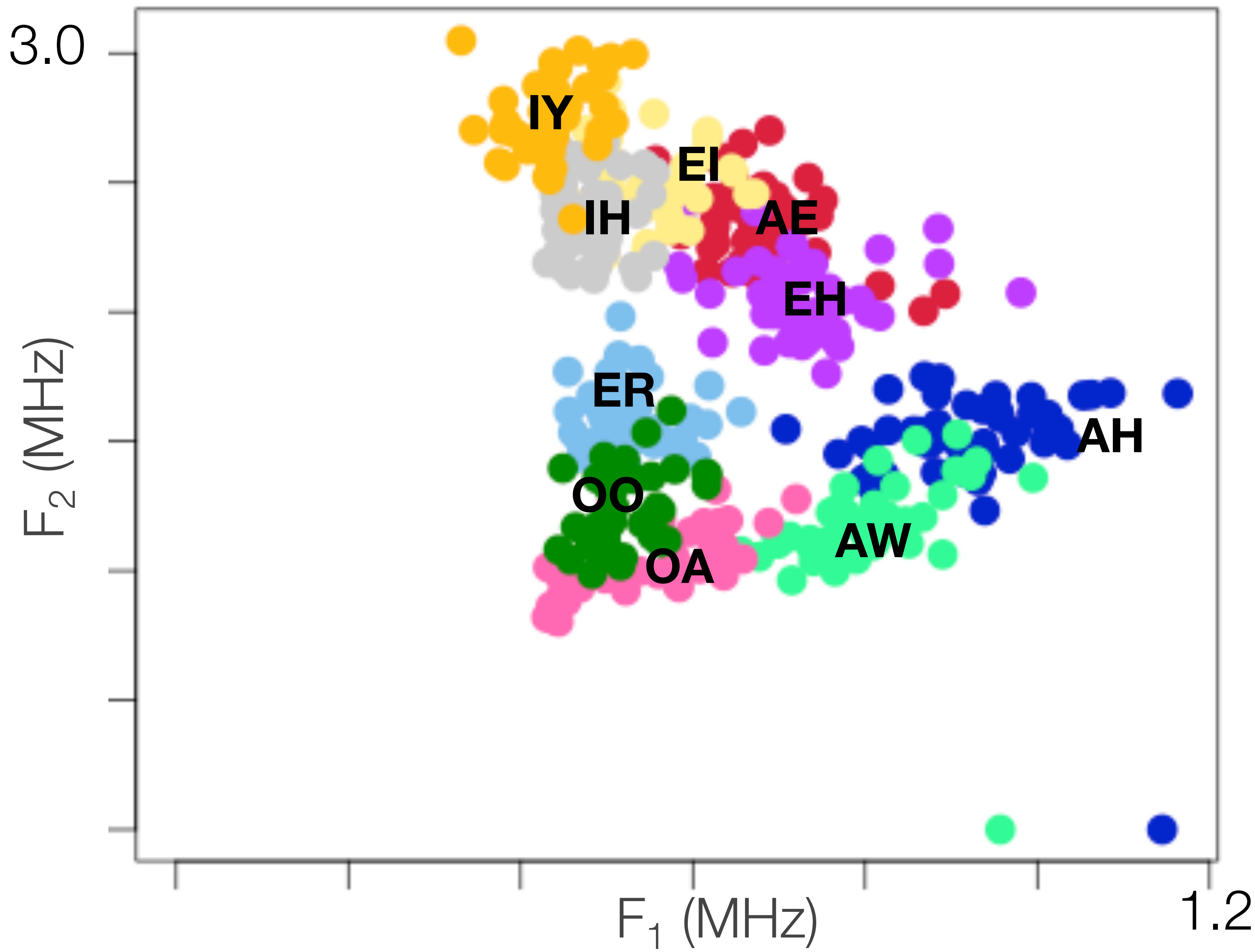
The computational problem



First 10 phonemes from Hillenbrand et al (1995), who recorded people saying vowel phonemes.

`load('phonemedata.RData')`

The computational problem

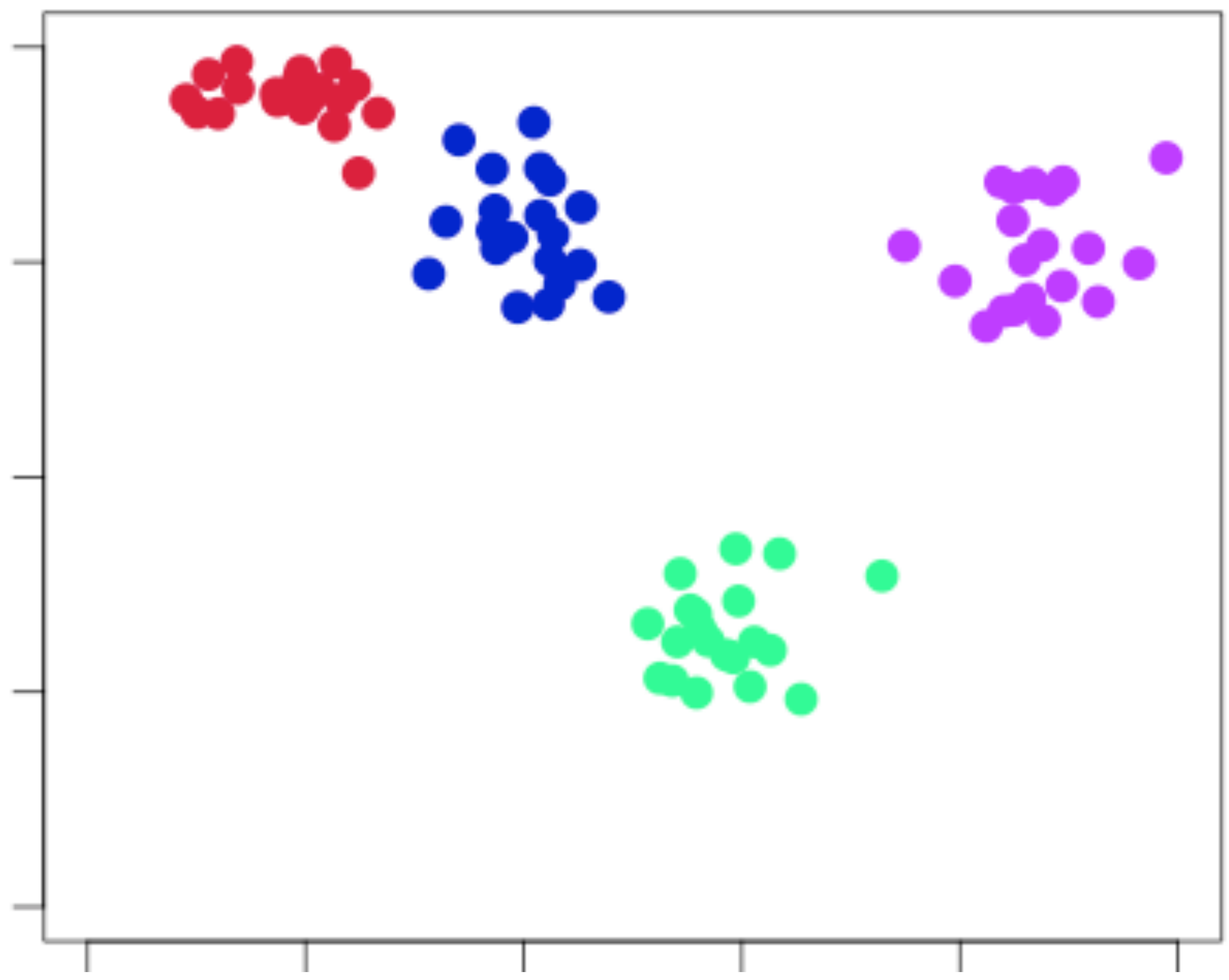


First 10 phonemes from Hillenbrand et al (1995), who recorded people saying vowel phonemes.

```
load('phonemedata.RData')
```

K-means clustering

Given a guess about how many clusters k there are, initialise the clusters randomly and then assign points to clusters in a way to minimise their distance from the centre of the clusters



```
load('sampledemo.RData')
kmeanscluster(d,4)
```

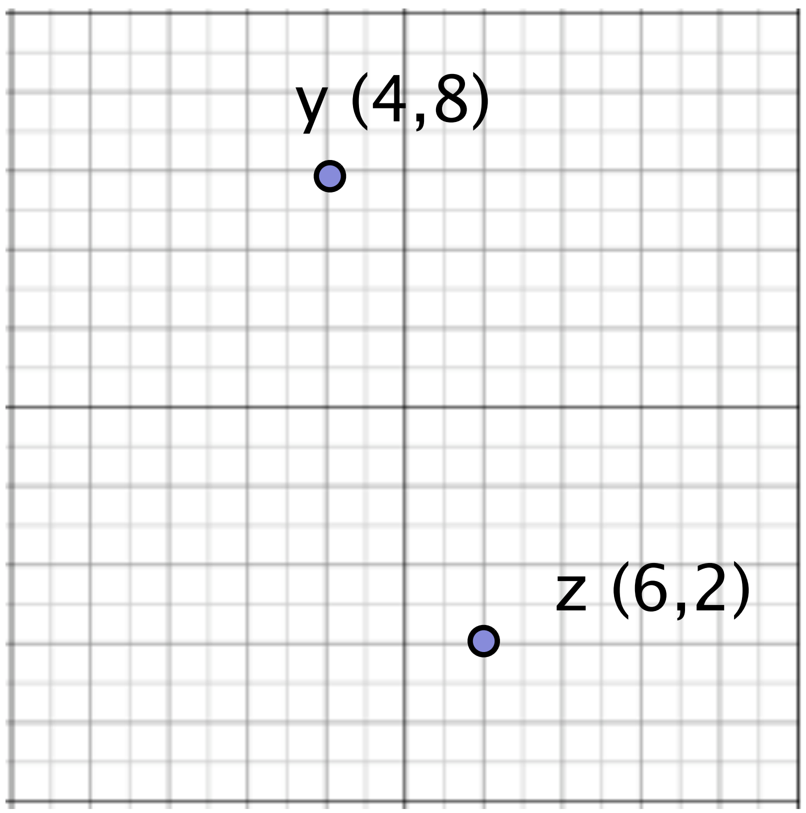
K-means clustering

Assumes that we can define a metric that measures the distance between two points in the space

For now, we'll use the following metric for any two points y and z:

$$d(\mathbf{y},\mathbf{z}) = \frac{1}{2}\sum_i (y_i - z_i)^2$$

$$\begin{aligned} d(\mathbf{y},\mathbf{z}) &= \frac{1}{2}\sum_i (y_i - z_i)^2 \\ &= \frac{1}{2}(4-6)^2 + \frac{1}{2}(8-2)^2 \\ &= \frac{1}{2}(4) + \frac{1}{2}(36) \\ &= 20 \end{aligned}$$



K-means clustering: Pseudocode

```
Initialization:
Set each mean to a random value*
Initialise “previous” responsibility matrix r_prev
Set up initial current responsibility matrix r_curr

While r_prev != r_curr

     Assignment step:
     Assign each datapoint to the closest mean

     Update step:
     Recalculate the means

  End
```

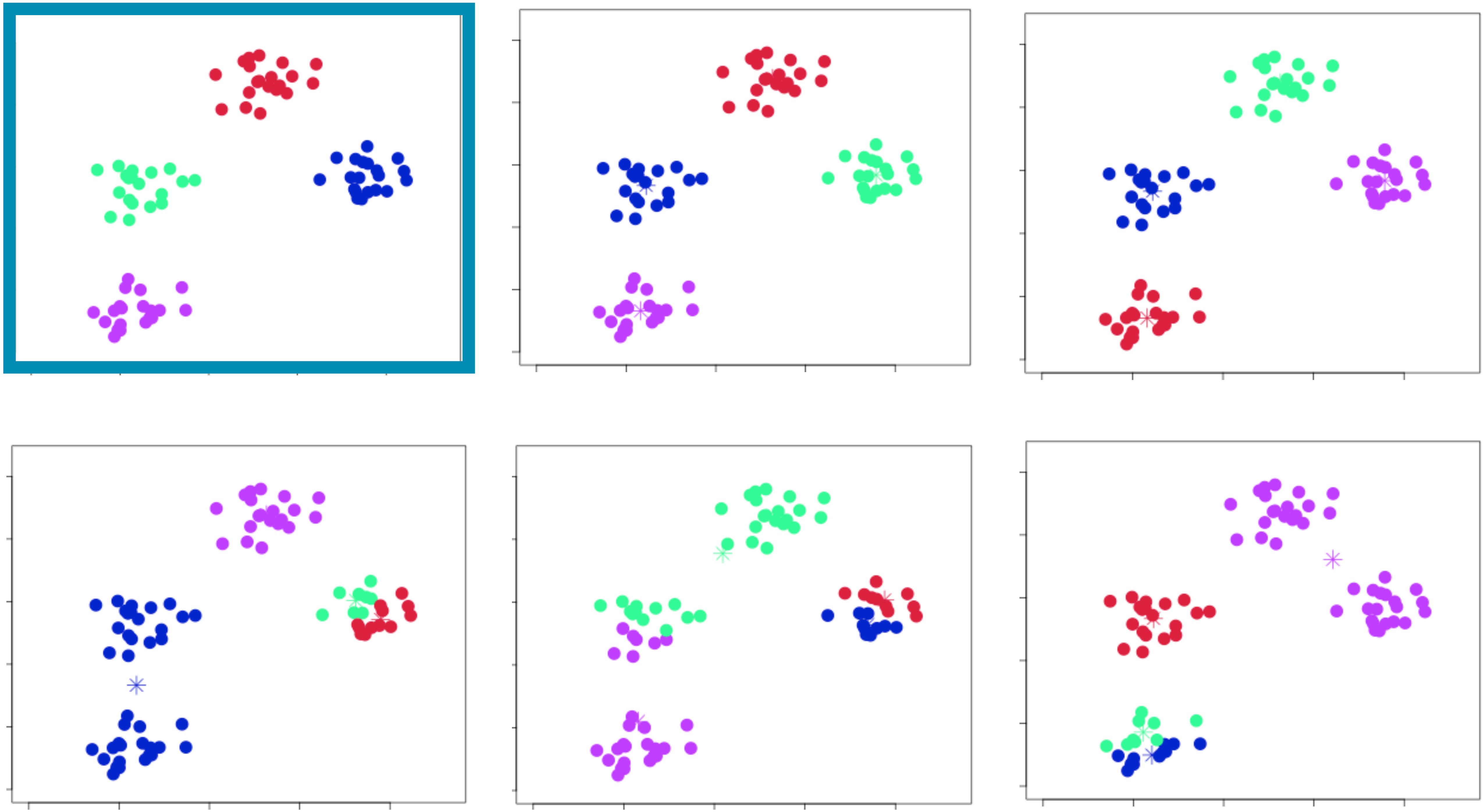
A “responsibility” matrix captures the assignment of datapoints to clusters

$$r_k^{(n)} = \begin{cases} 1 \text{ if } \mathbf{m}^{(k)} \text{ is the closest mean to datapoint } n \\ 0 \text{ otherwise} \end{cases}$$

$\mathbf{m}^{(k)}$ is the mean of the kth cluster

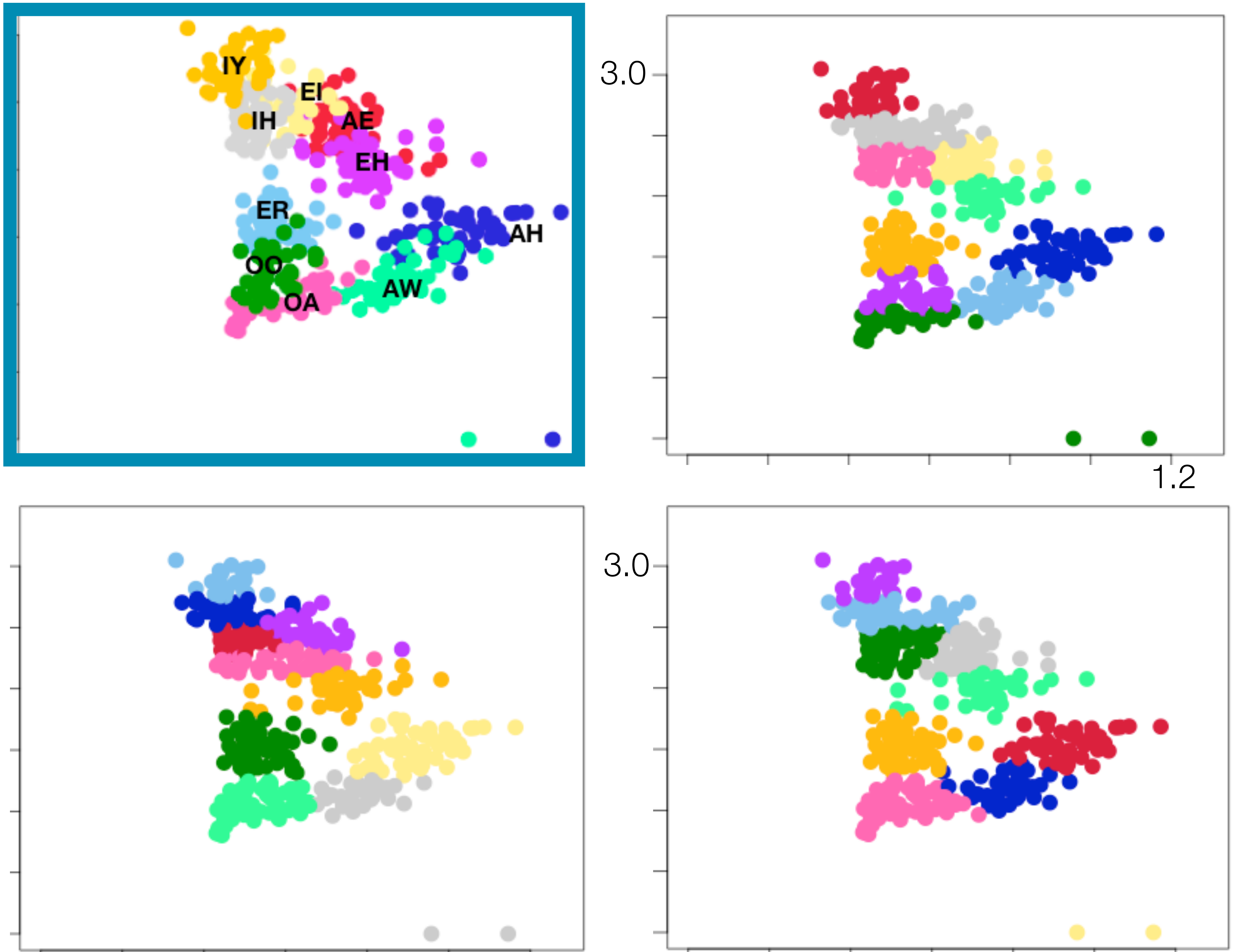
* To ensure that each cluster has at least one datapoint, set this to the value of a random datapoint

Example 1: Easy dataset



```
load('fakeeasydata.RData')
kmeanscluster(d,4)
```

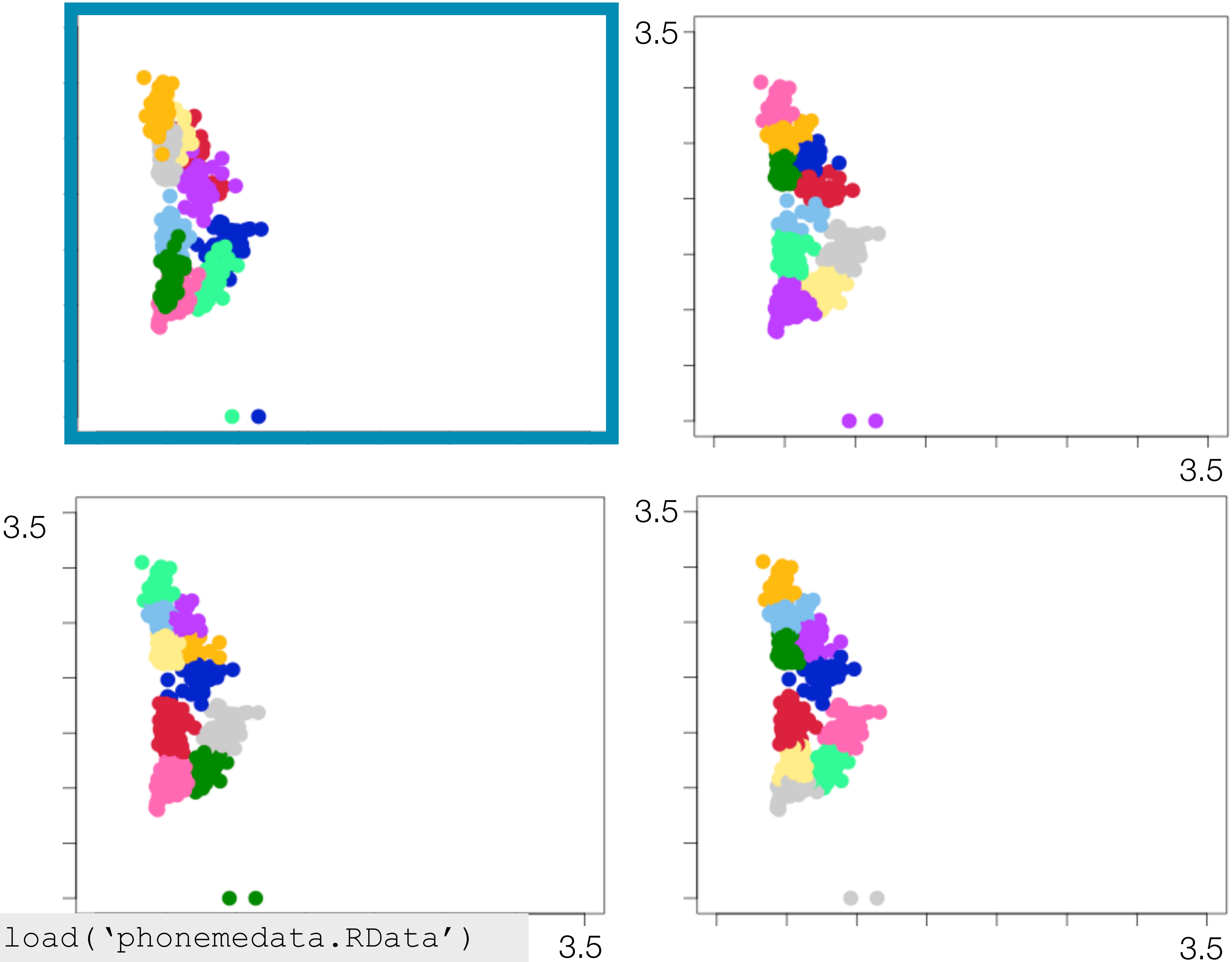
How well does it do on the phoneme data?



Note that the clusters appear elongated because the axes are on different scales.

```
load('phonemedata.RData')
kmeanscluster(d,4)
```

How well does it do on the phoneme data?



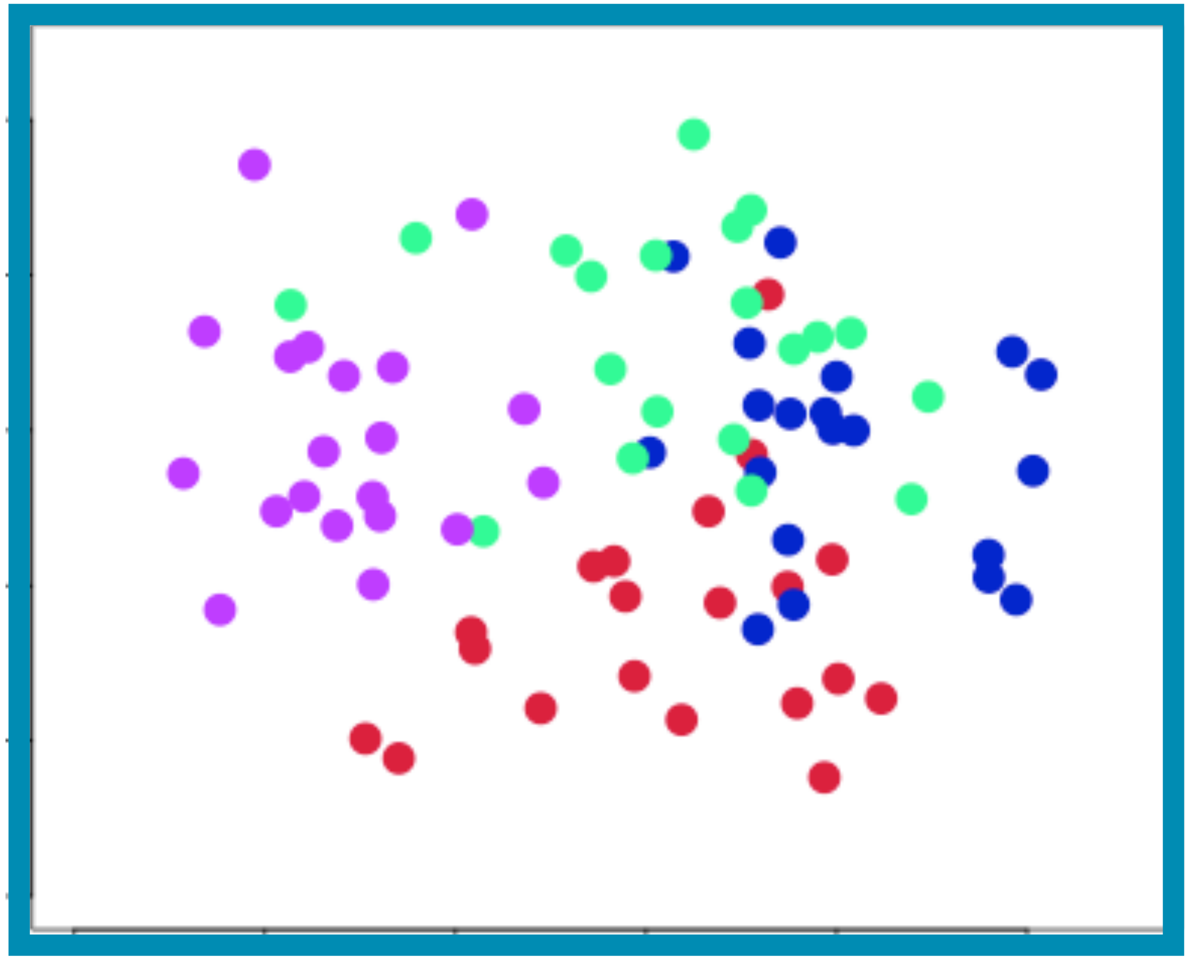

```
load('phonemedata.RData')
kmeanscluster(d,4)
```

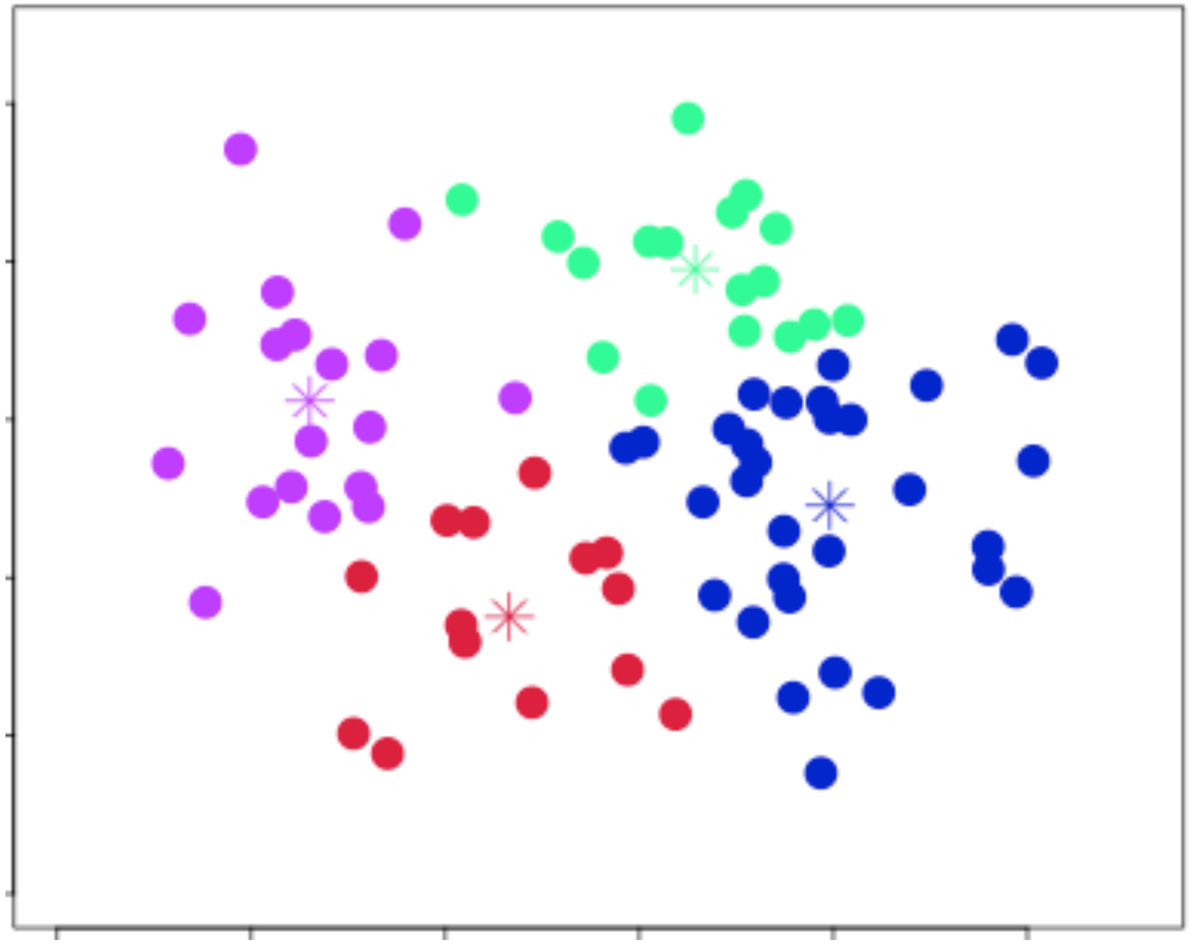
When on equal scales, the actual phonemes are elongated but the ones learned by k-means are not

Qualitative analysis of k-means

- Good things
  - Guaranteed to converge to a local maximum
  - Fast
- Bad things
  - Convergence is not to *global* maximum, so final result is very dependent on starting position
  - Particularly bad for certain kinds of datasets

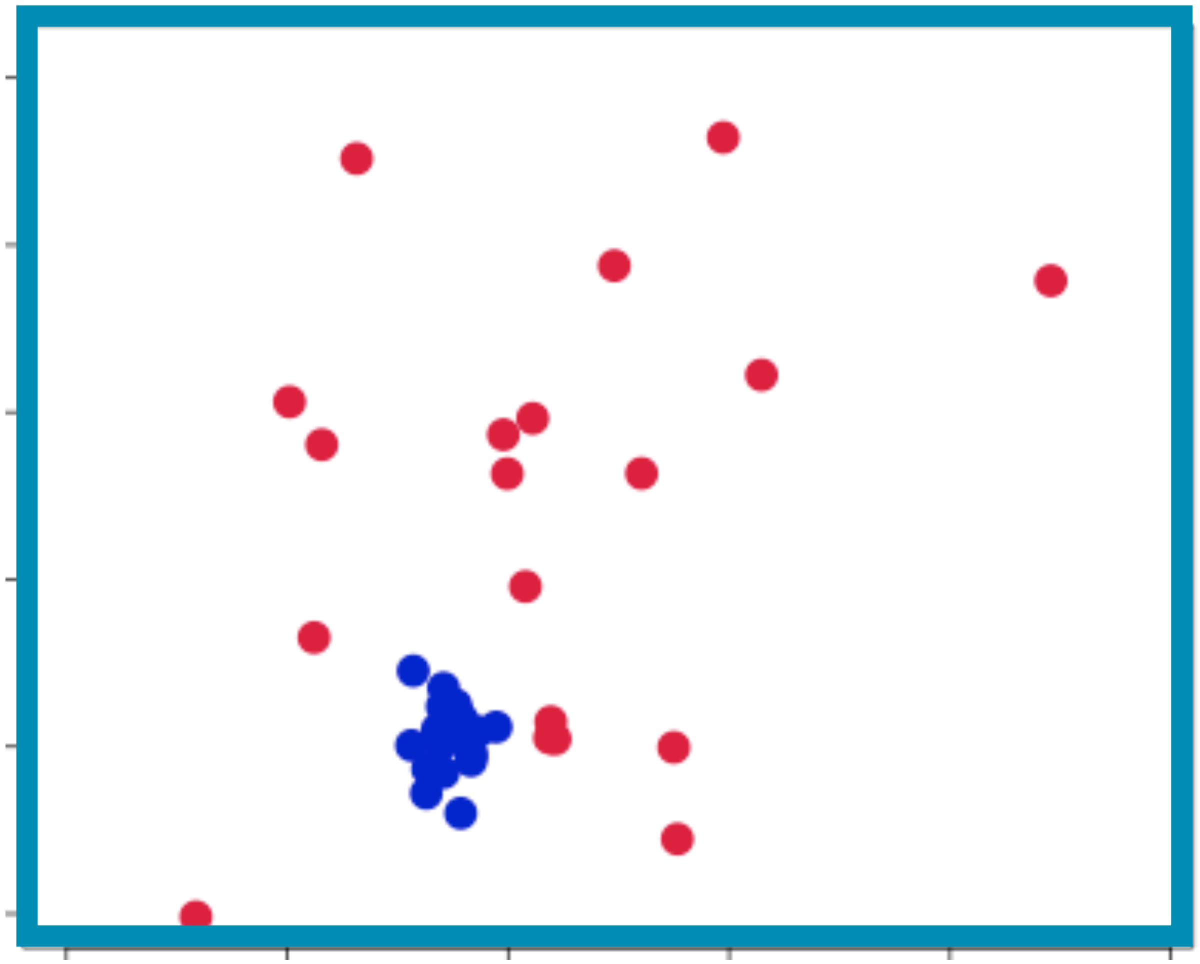
Bad dataset #1: Points overlap

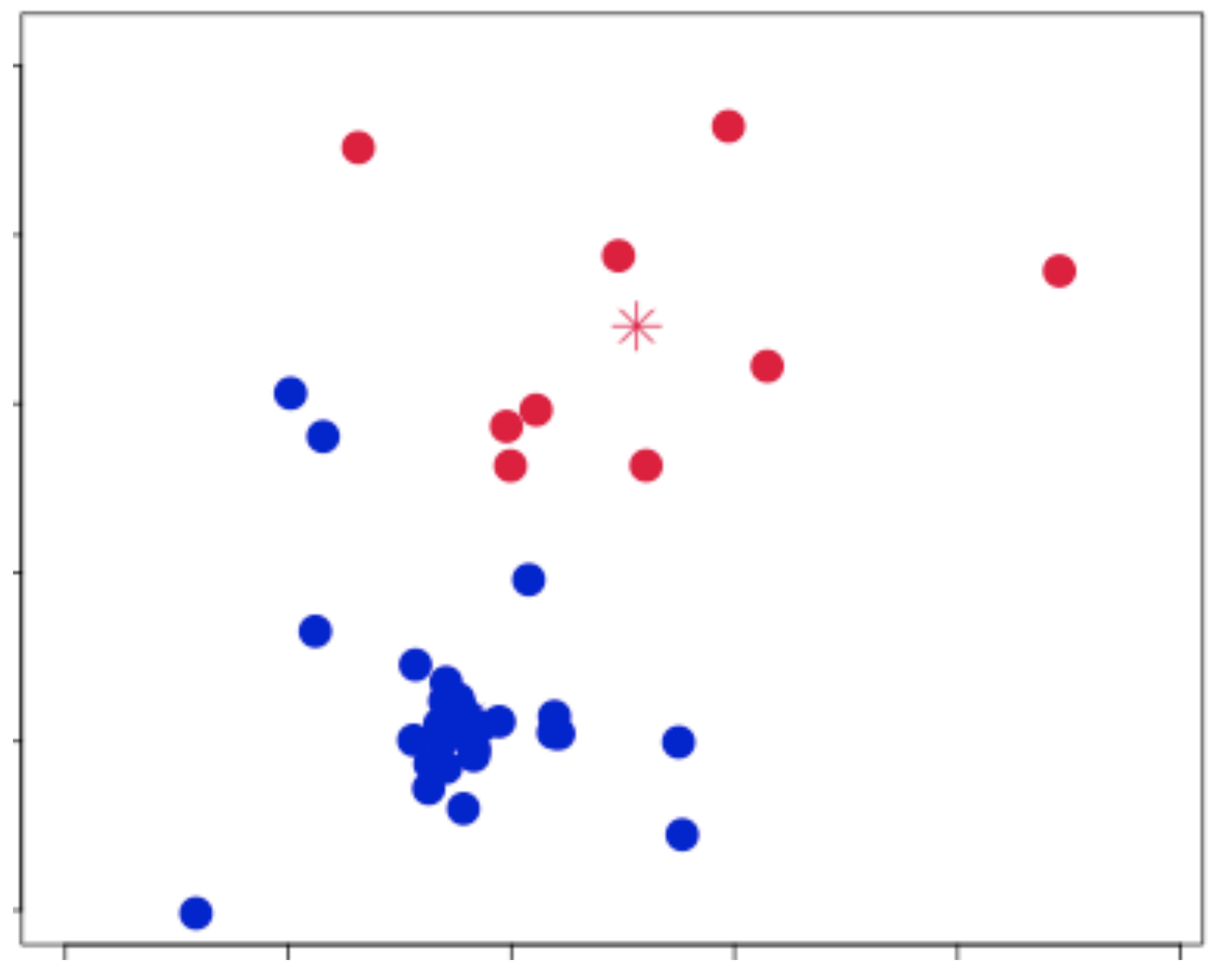




```
load('baddataset1.RData')
kmeanscluster(d,4)
```

Bad dataset #2: Differently sized clusters

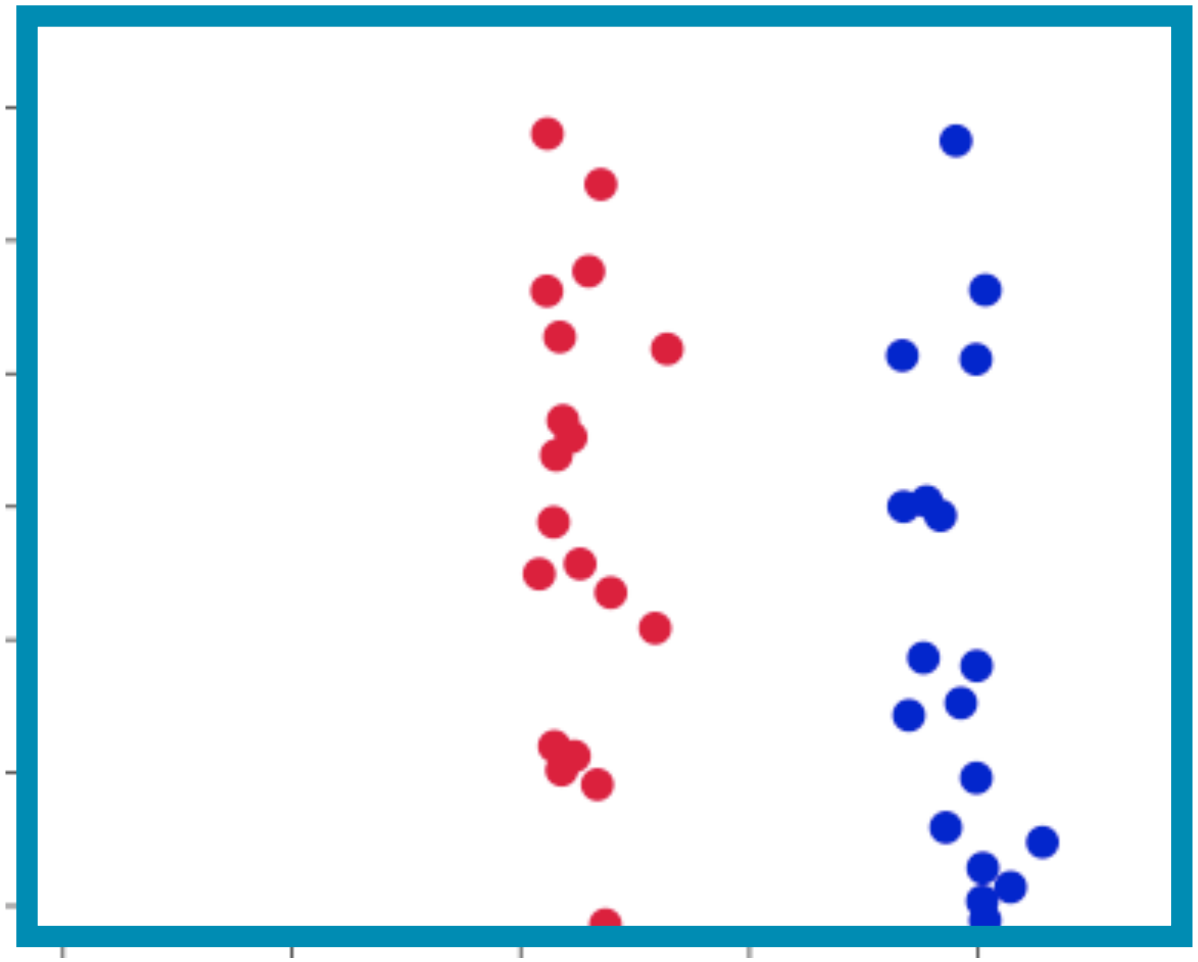


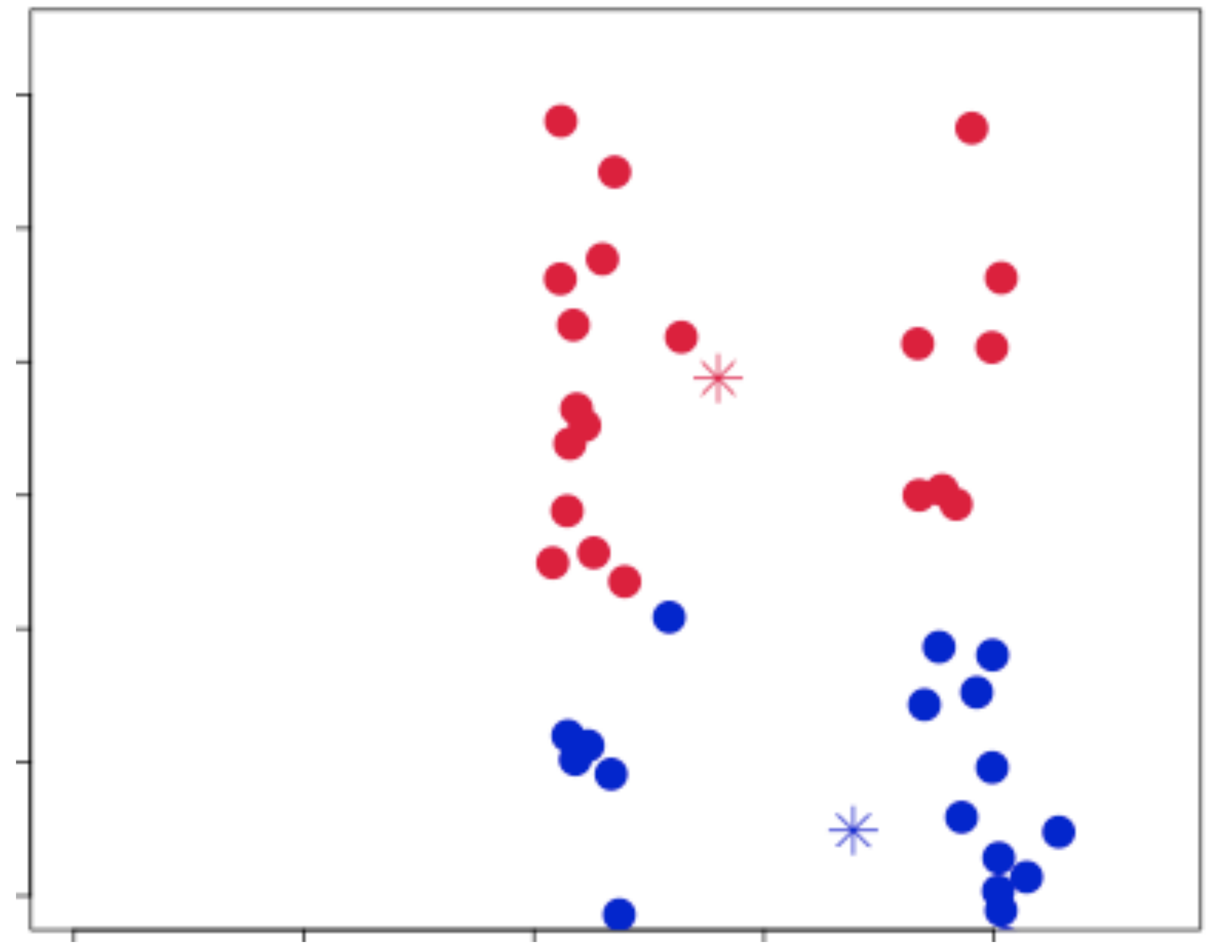


```
load('baddataset2.RData')
kmeanscluster(d,2)
```

Bad dataset #3: Elongated clusters

(like the phoneme data)

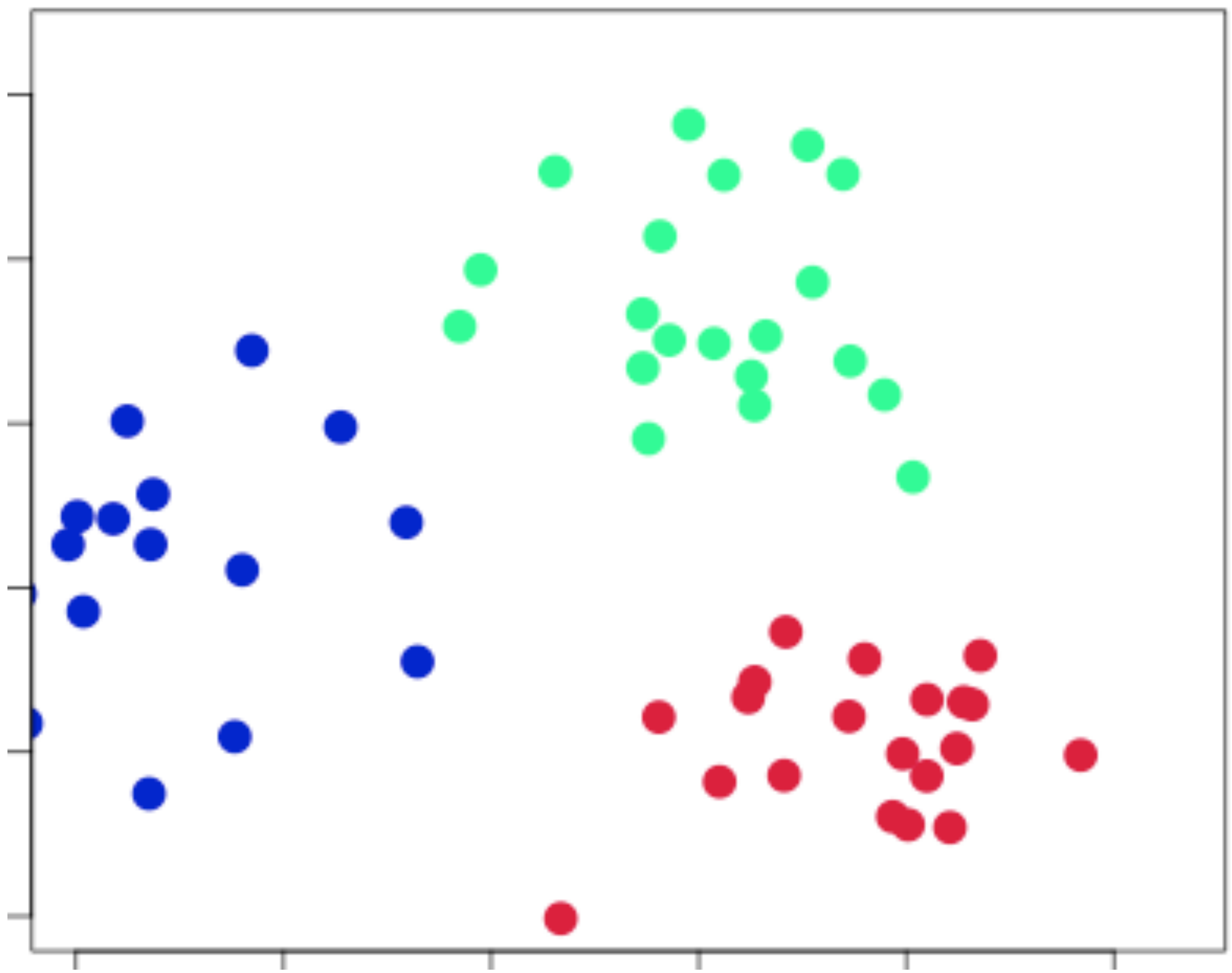




```
load('baddataset3.RData')
kmeanscluster(d,2)
```

Another general problem

Category assignments are hard. Points near the border should arguably affect the means of all nearby clusters.



Soft K-means clustering: Pseudocode

```
Initialization:
Set each mean to a random value*
Initialise "previous" responsibility matrix r_prev
Set up initial current responsibility matrix r_curr

While r_prev != r_curr

     Assignment step:
     Each datapoint is assigned to each mean probabilistically,
     proportional to its distance from the mean

     Update step:
     Recalculate the means

  End
```

Before: "responsibility" matrix captures the assignment of datapoints to clusters

$$r_k^{(n)} = \begin{cases} 1 \text{ if } \mathbf{m}^{(k)} \text{ is the closest mean to datapoint } n \\ 0 \text{ otherwise} \end{cases}$$

$\mathbf{m}^{(k)}$ is the mean of the kth cluster

* To ensure that each cluster has at least one datapoint, set this to the value of a random datapoint

Soft K-means clustering: Pseudocode

```
Initialization:
Set each mean to a random value*
Initialise "previous" responsibility matrix r_prev
Set up initial current responsibility matrix r_curr

While r_prev != r_curr

     Assignment step:
     Each datapoint is assigned to each mean probabilistically,
     proportional to its distance from the mean

     Update step:
     Recalculate the means

  End
```

Now: it captures a "soft" assignment of datapoints to clusters

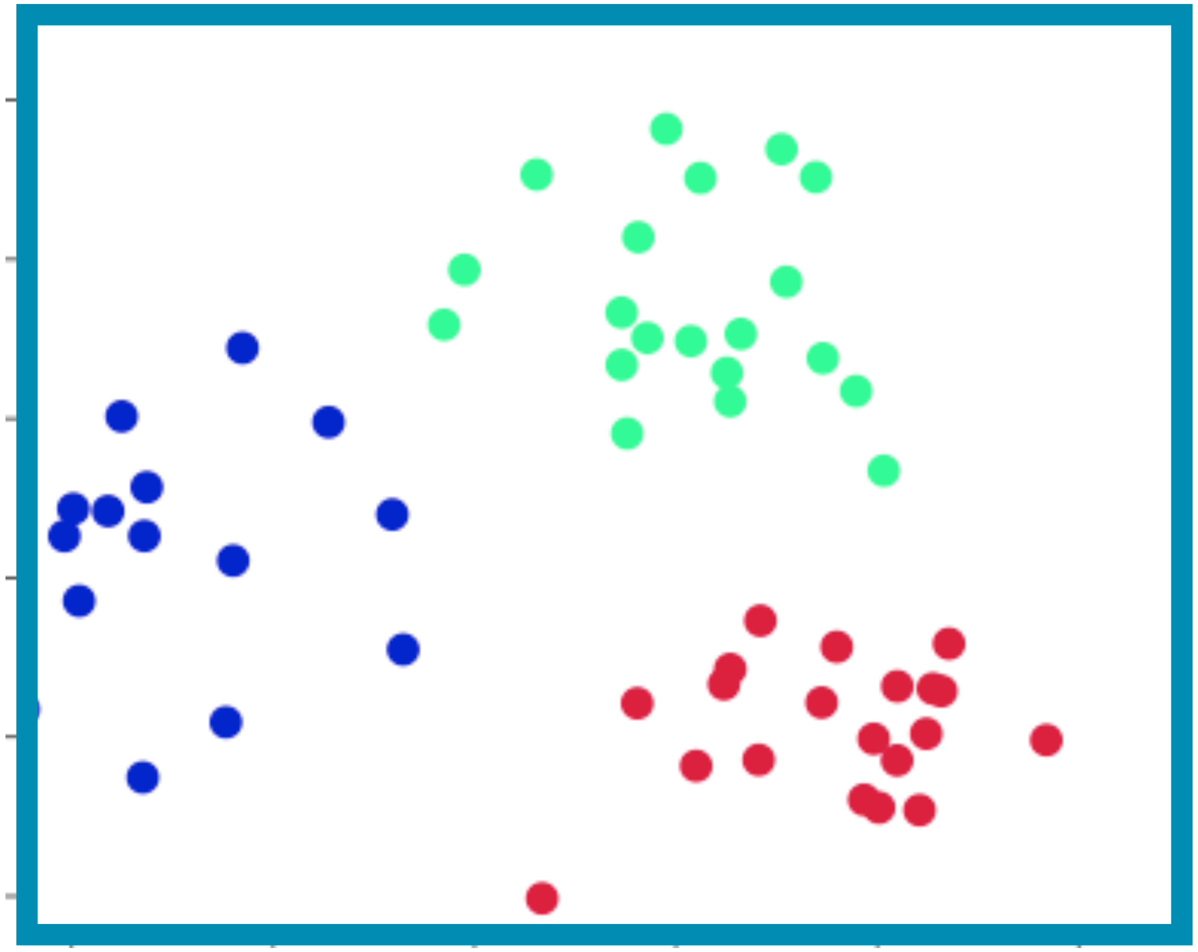
$$r_k^{(n)} = \frac{\exp(-\beta d(\mathbf{m}^{(k)},\mathbf{x}^{(n)}))}{\sum_{k'} \exp(-\beta d(\mathbf{m}^{(k')},\mathbf{x}^{(n)}))}$$

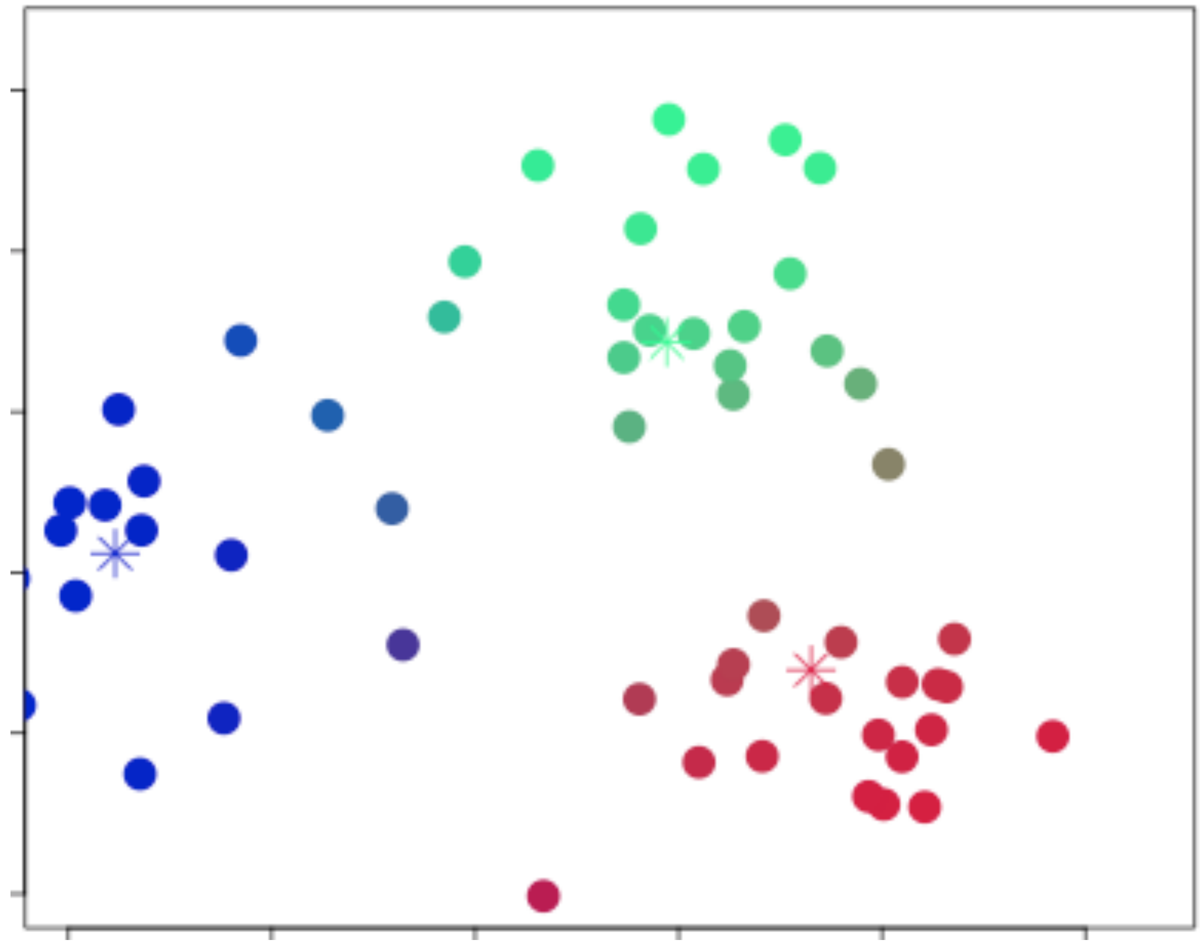
$\mathbf{m}^{(k)}$ is the mean of the kth cluster

β governs the "stiffness" of assignments

* To ensure that each cluster has at least one datapoint, set this to the value of a random datapoint

Soft K-means often performs sensibly



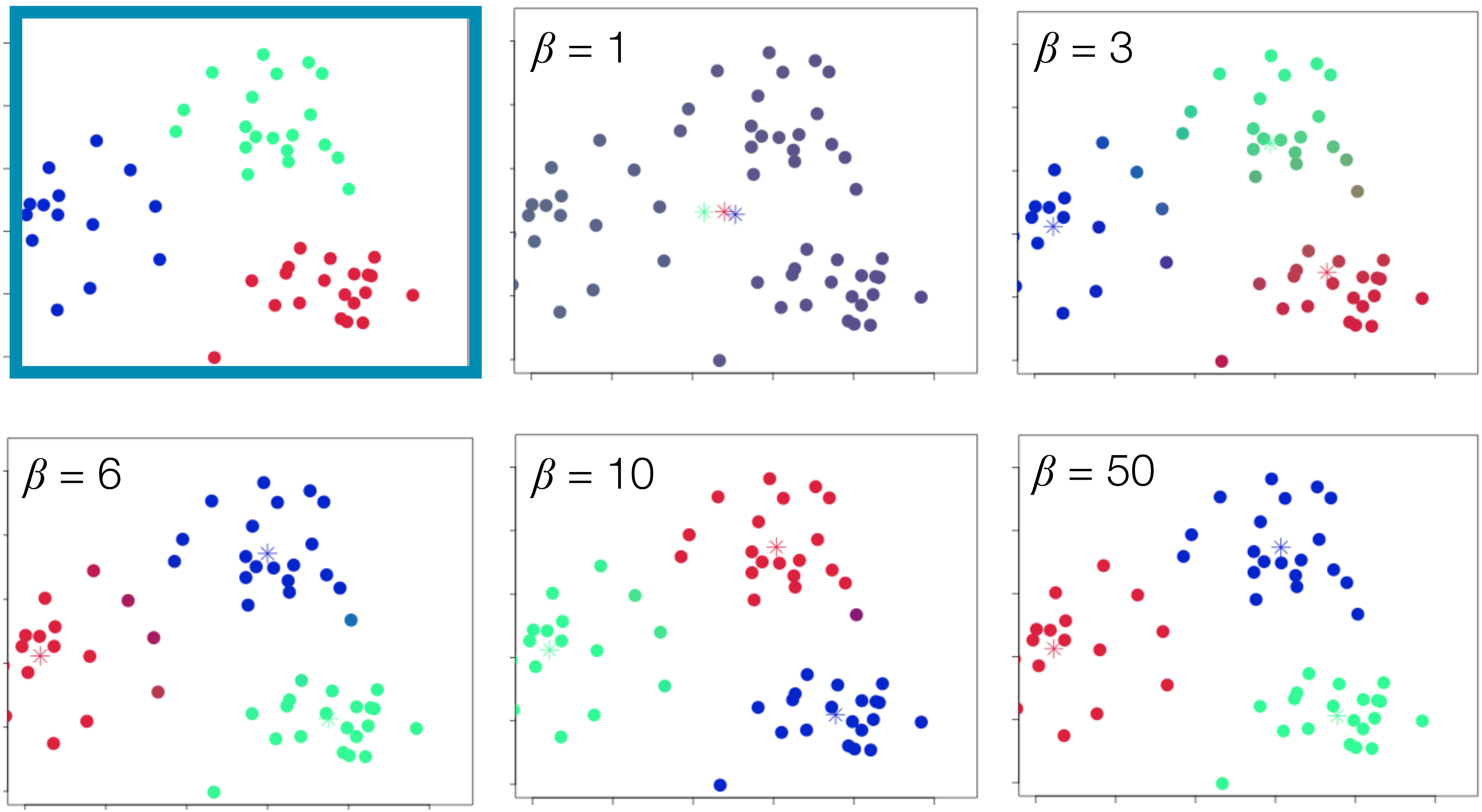


$\beta = 3$

```
load('softkmeansdemo.RData')
softkmeanscluster(d,3,3)
```

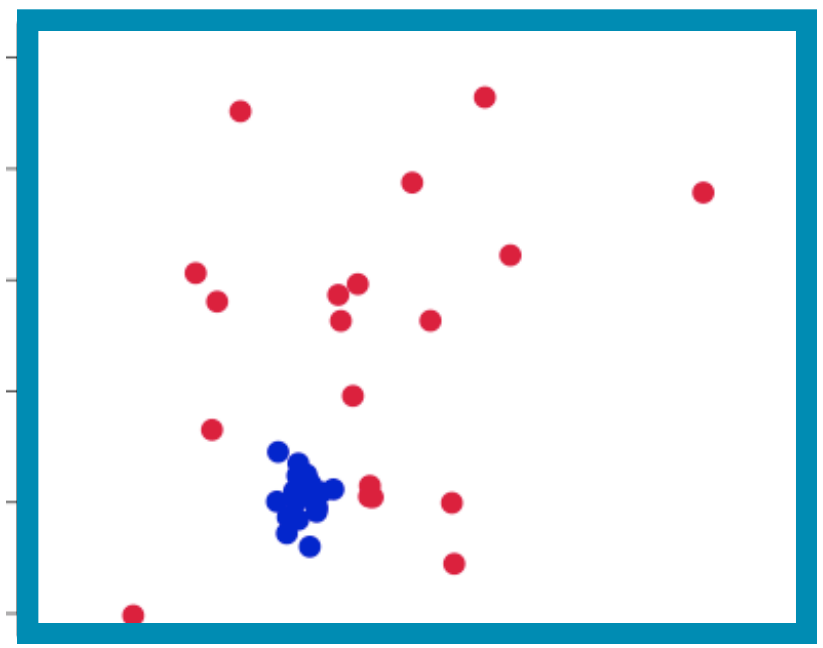
Soft K-means often performs sensibly

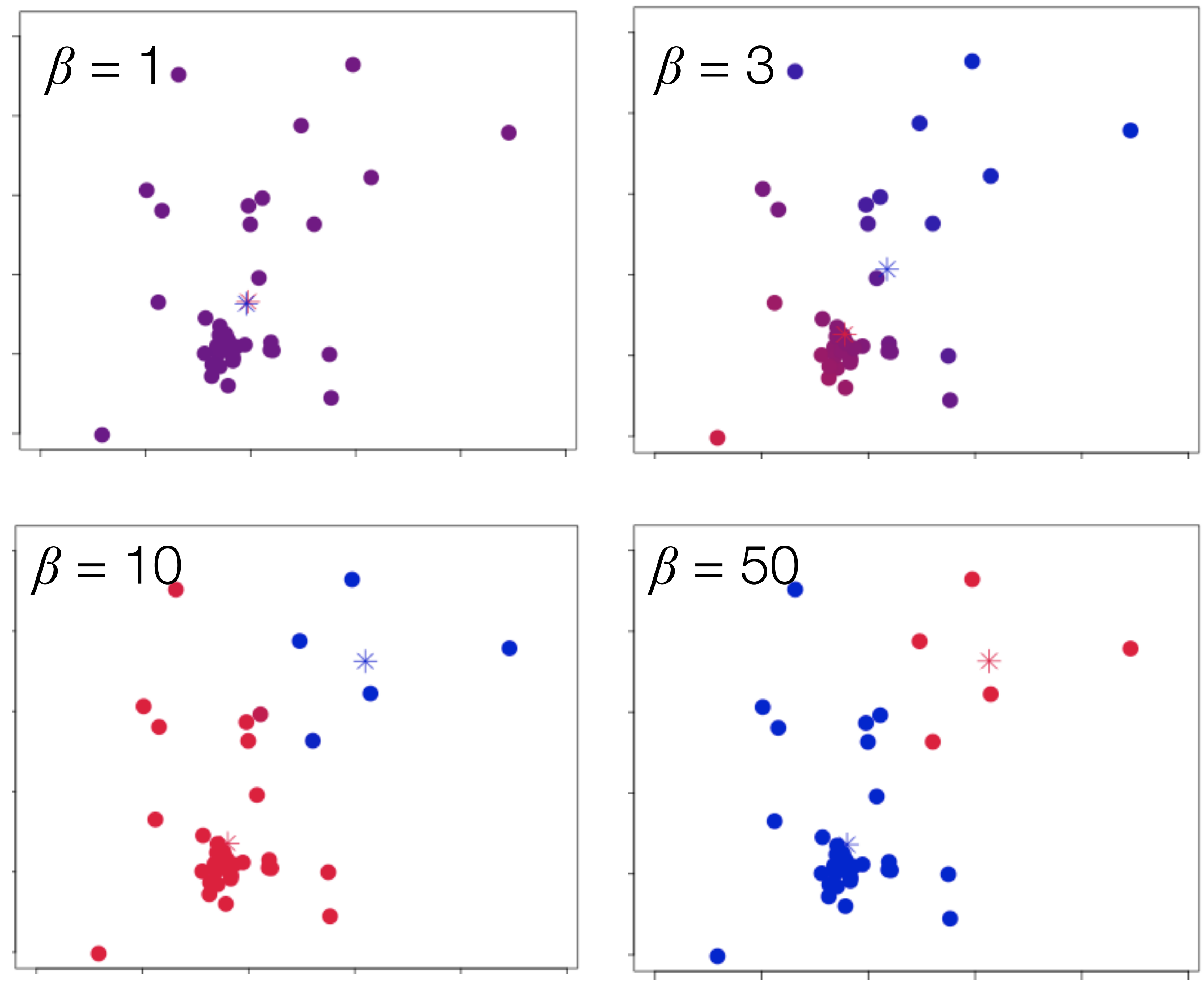
As β approaches infinity, it turns into hard k-means clustering



... but it still has many of the same problems

Can't handle clusters of different sizes

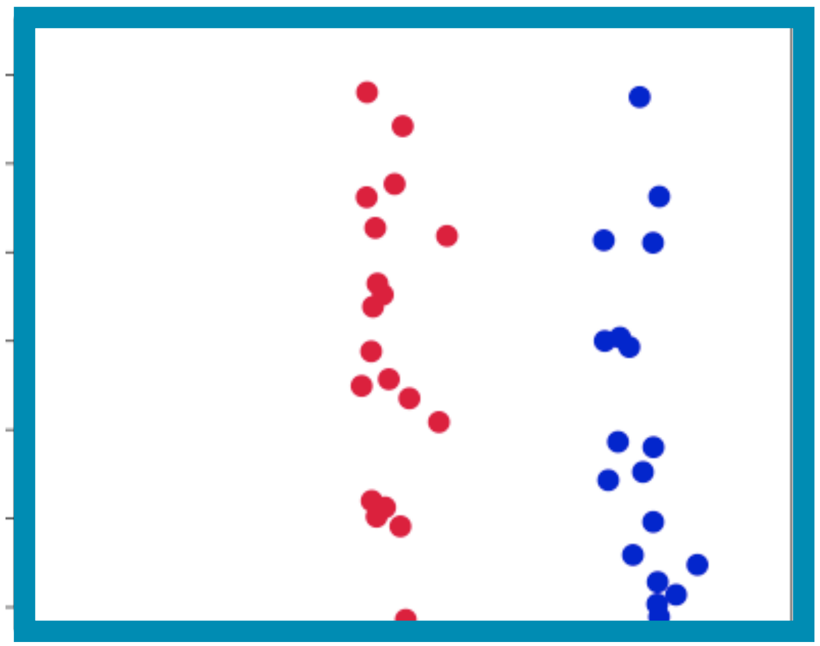


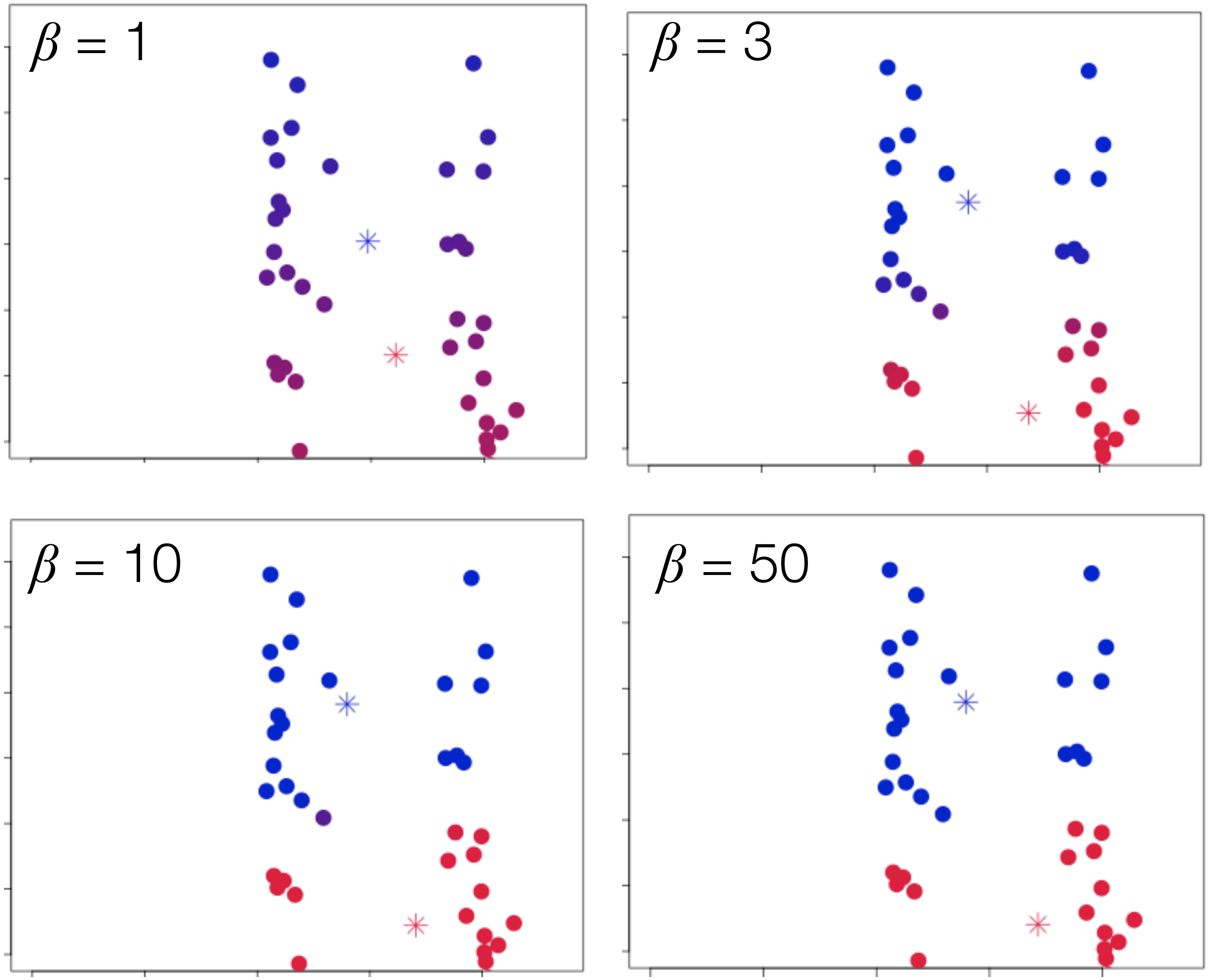



```
load('baddataset2.RData')
softkmeanscluster(d,2,beta)
```

... but it still has many of the same problems

Can't handle elongated clusters

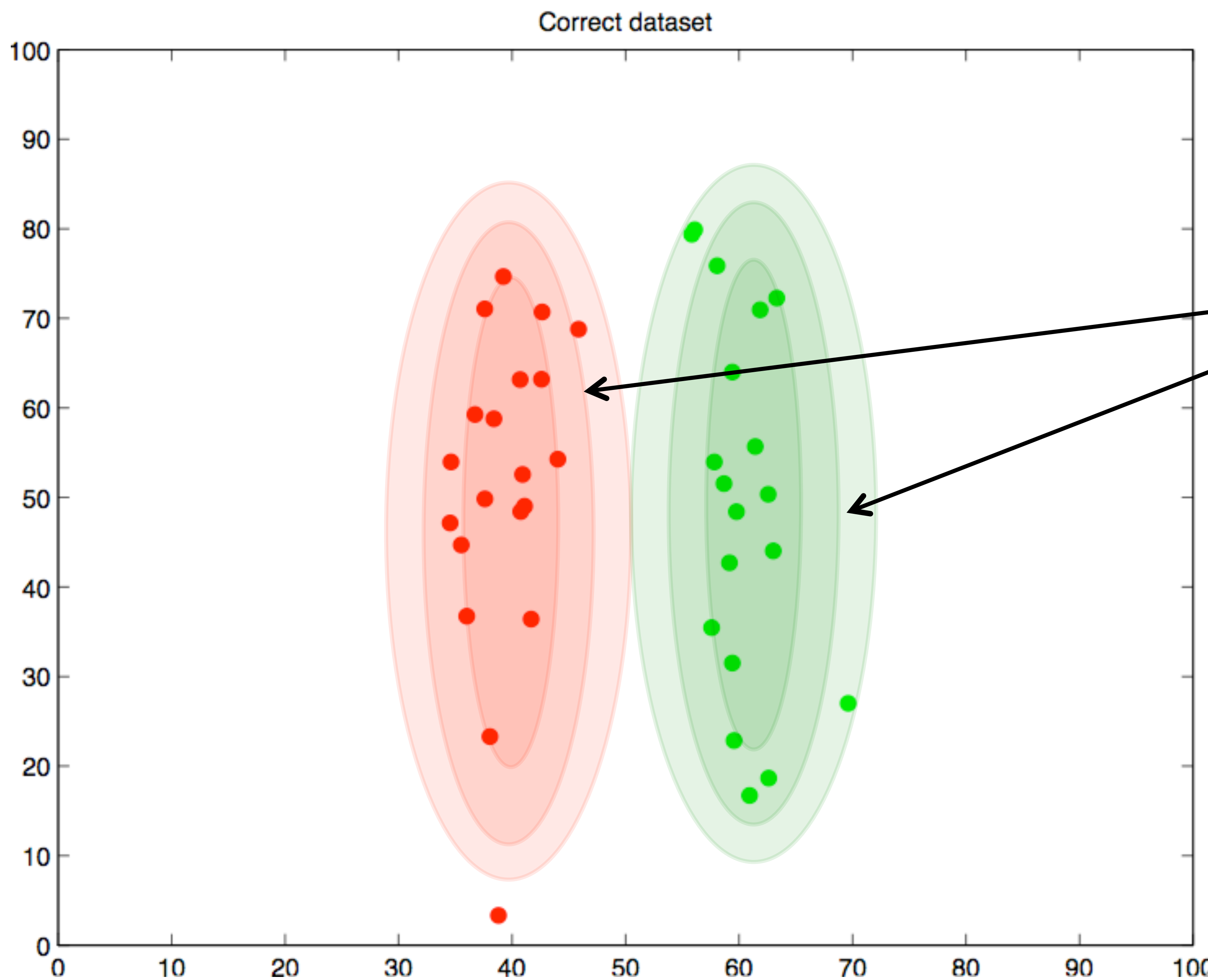





```
load('baddataset3.RData')
softkmeanscluster(d,2,beta)
```

What’s going on here?

Take a step back, first. How are data (like phonemes) probably generated?



Some sort of underlying process which imposes a distribution over data points

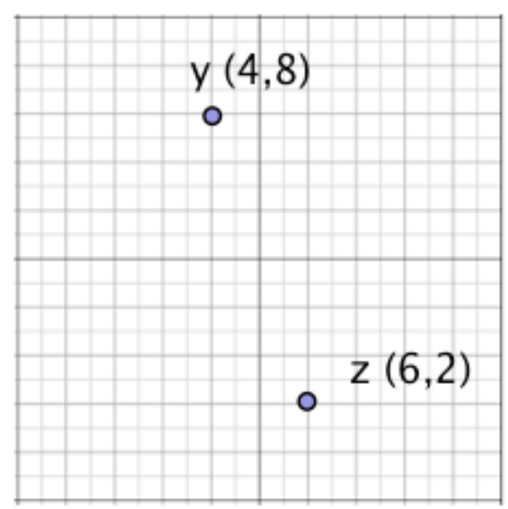
What’s going on here?

k-means clustering is making some implicit assumptions about the nature of that process

For now, we’ll use the following metric for any two points y and z:

$$d(\mathbf{y}, \mathbf{z}) = \frac{1}{2} \sum_i (y_i - z_i)^2$$

$$\begin{aligned} d(\mathbf{y}, \mathbf{z}) &= \frac{1}{2} \sum_i (y_i - z_i)^2 \\ &= \frac{1}{2}(4-6)^2 + \frac{1}{2}(8-2)^2 \\ &= \frac{1}{2}(4) + \frac{1}{2}(36) \\ &= 20 \end{aligned}$$

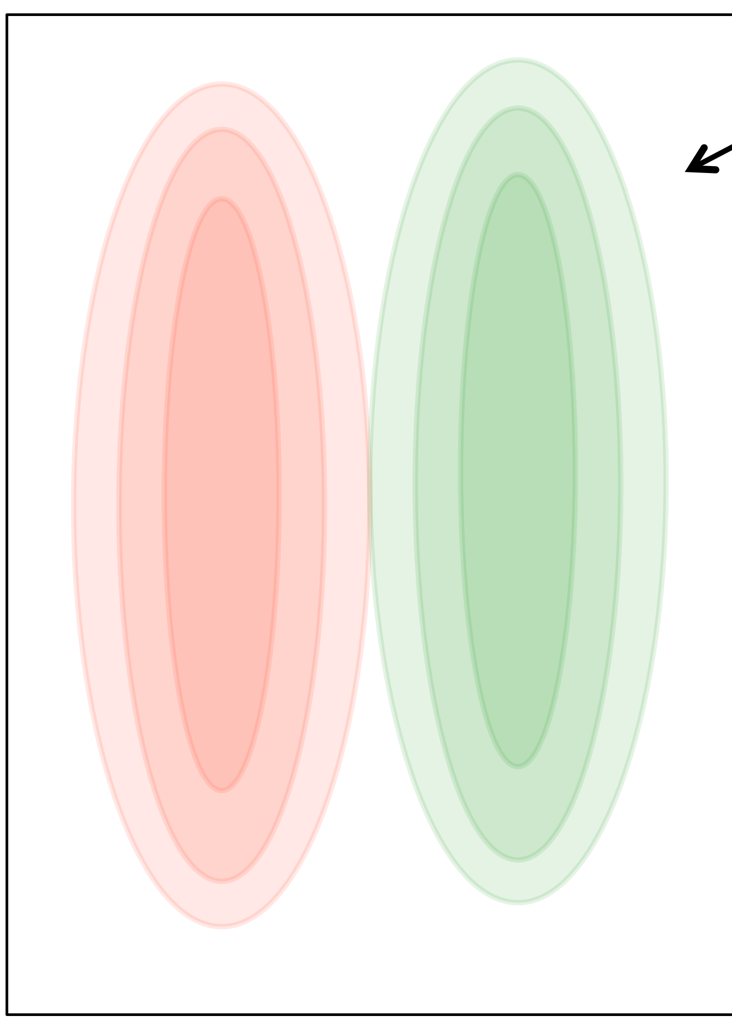


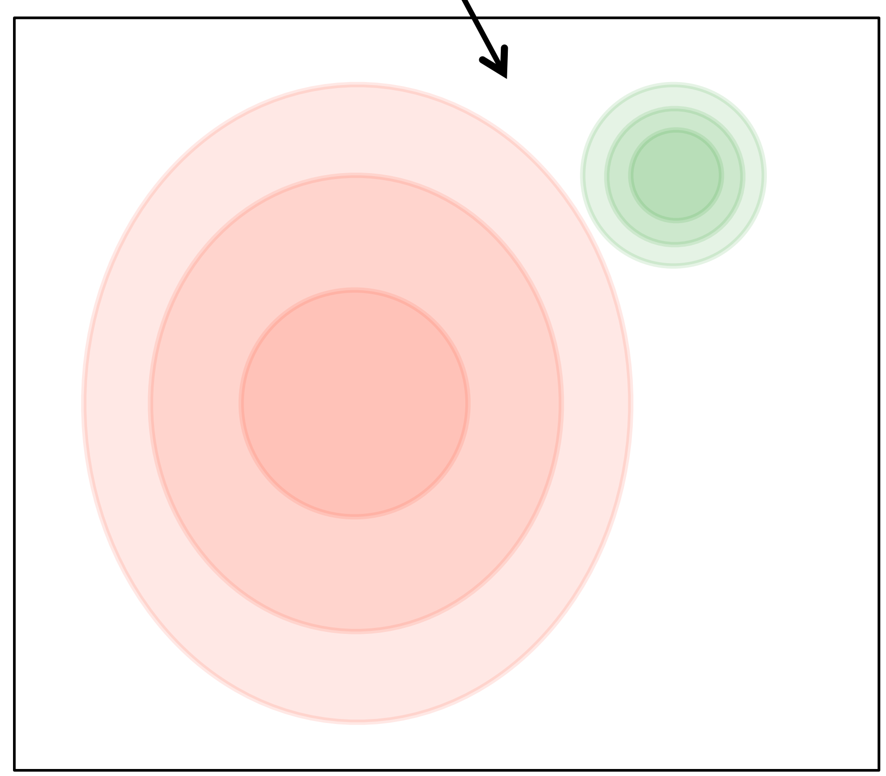
Distance metric is the same in every direction and for every cluster

What's going on here?

As a result, k-means assumes that all clusters are the same size as well as symmetric (circular)

Both of these are impossible to do well





The fix: change these assumptions

As a result, k-means assumes that all clusters are the same size as well as symmetric (circular)

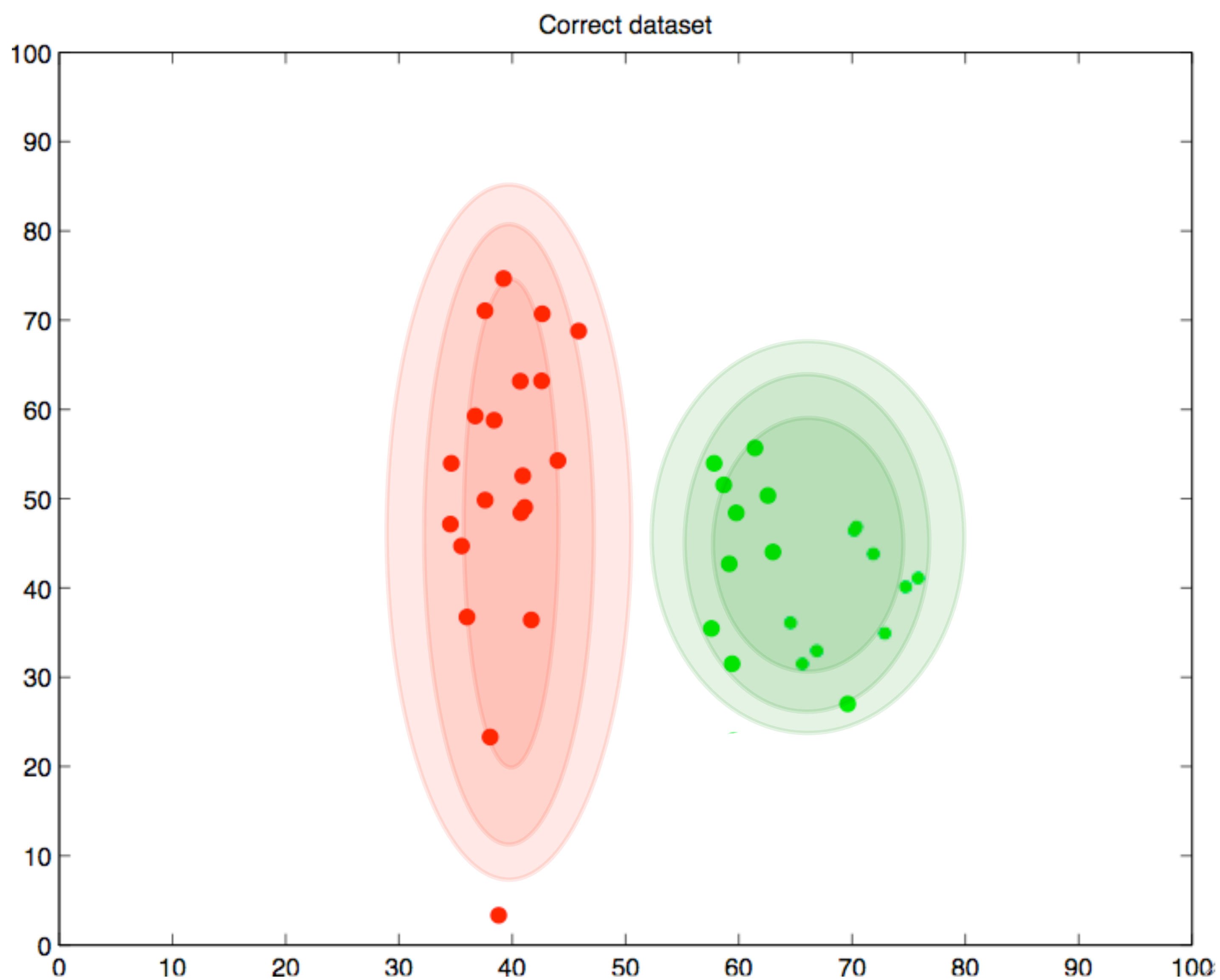
The result is an algorithm called *Mixture of Gaussians*

Lecture outline

- Unsupervised classification
  - Case study: phoneme learning in language
- A first try: k-means clustering
  - Limitations and an extension
- ➡ Next try: Mixture of Gaussians
  - EM model for calculating
- Next: Semi-supervised classification

Mixture of Gaussians

Assumes that the data are generated by Gaussians (normal distributions), possibly with different variances in different directions



The algorithm for calculating the best Gaussians is called the EM algorithm after the two steps involved

Mixture of Gaussians with EM

The Expectation step (or E-step) is a direct analogue of the assignment step previously: each datapoint is assigned probabilistically to each cluster

Responsibilities are:

$$r_k^{(n)} = \frac{w_k \frac{1}{\prod_{i=1}^{I} \sqrt{2\pi}\sigma_i^{(k)}} \exp\left(-\sum_{i=1}^{I}(m_i^{(k)}-x_i^{(n)})^2/2(\sigma_i^{(k)})^2\right)}{\sum_{k'} w_{k'} \frac{1}{\prod_{i=1}^{I} \sqrt{2\pi}\sigma_i^{(k')}} \exp\left(-\sum_{i=1}^{I}(m_i^{(k')}-x_i^{(n)})^2/2(\sigma_i^{(k')})^2\right)}$$

Equation for a Gaussian -
you've seen this in Dan's recent lectures!
(so this is exactly the same as calculating the likelihood of that point under the Gaussian distribution with parameters w, m, and σ)

Mixture of Gaussians with EM

The Expectation step (or E-step) is a direct analogue of the assignment step previously: each datapoint is assigned probabilistically to each cluster

Total number of dimensions i

Responsibilities are:

$$r_k^{(n)} = \frac{w_k \frac{1}{\prod_{i=1}^{I} \sqrt{2\pi}\sigma_i^{(k)}} \exp\left(-\sum_{i=1}^{I} (m_i^{(k)} - x_i^{(n)})^2 / 2(\sigma_i^{(k)})^2\right)}{\sum_{k'} w_{k'} \frac{1}{\prod_{i=1}^{I} \sqrt{2\pi}\sigma_i^{(k')}} \exp\left(-\sum_{i=1}^{I} (m_i^{(k')} - x_i^{(n)})^2 / 2(\sigma_i^{(k')})^2\right)}$$

weight for cluster k

standard deviation of cluster k

Mean along dimension i for cluster k

Point n at dimension i

Mixture of Gaussians with EM

The Maximisation step (or M-step) is an analogue of the update step previously, but in addition to the mean we need to update the weights and standard deviation

Means:

$$\mathbf{m}^{(k)} = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

Variance:

$$\sigma_k^2 = \frac{\sum_n r_k^{(n)} (x_i^{(n)} - m_i^{(k)})^2}{\sum_n r_k^{(n)}}$$

Weights:

$$w_k = \frac{\sum_n r_k^{(n)}}{\sum_k \sum_n r_k^{(n)}}$$

this is the same as the update step for soft k-means: the mean of all points weighted by the proportion to which they belong in the cluster

This is the average variance of the cluster where points are weighted by the proportion of their likelihood taken care of by that cluster

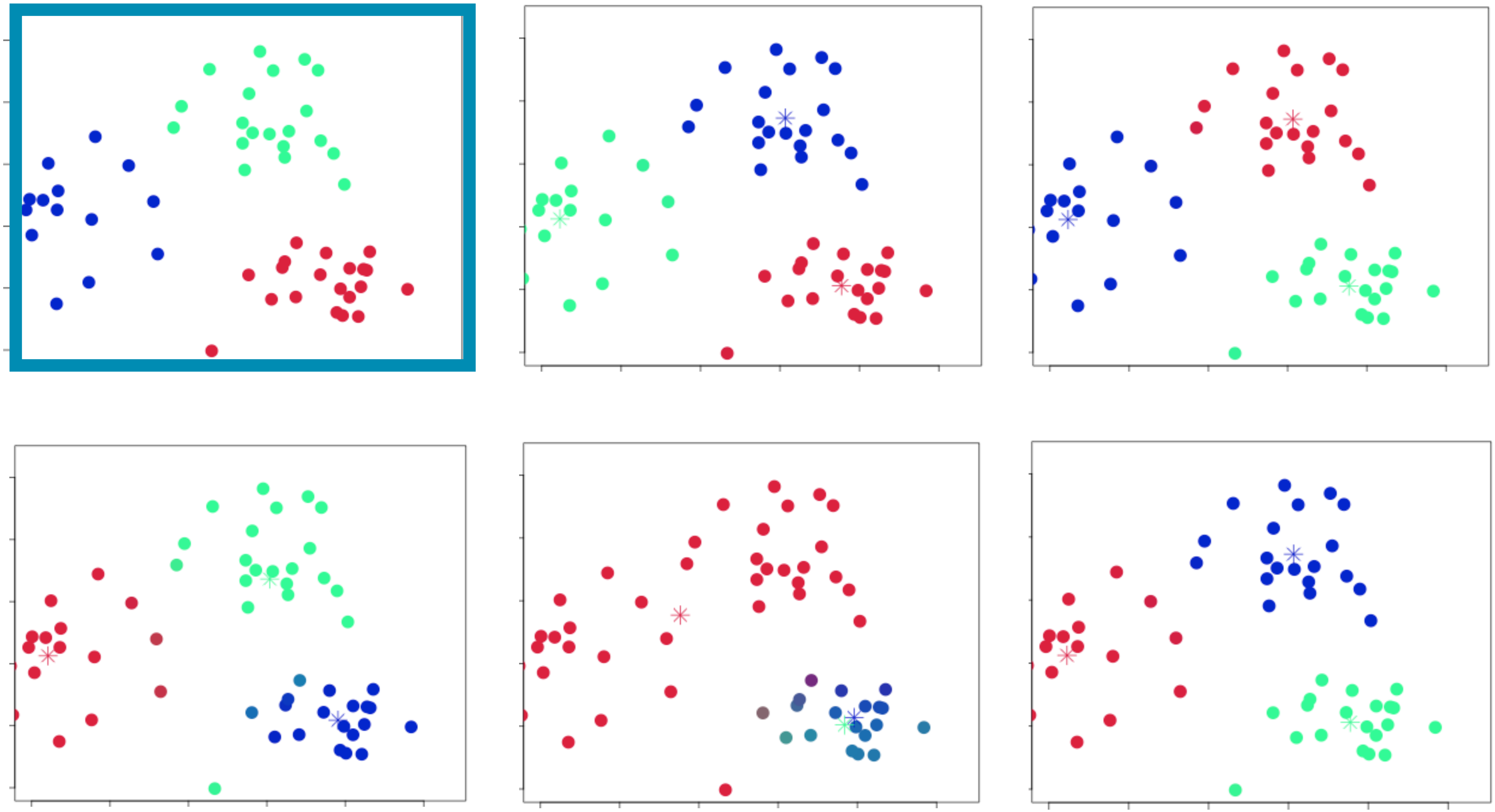
This is the sum of all responsibilities in that cluster (so clusters with more points have more weight)

Mixture of Gaussians: Pseudocode

```
Initialization:
Set each mean, standard deviation, and weight to a random value*
Initialise “previous” responsibility matrix r_prev
Set up initial current responsibility matrix r_curr

While r_prev != r_curr

     Assignment step (E-step):
     Calculate the likelihood of each datapoint in each cluster,
     assuming the cluster is a Gaussian with the current mean,
     standard deviation, and weight

     Update step:
     Recalculate the means
     Recalculate the standard deviations
     Recalculate the weights

  End
```

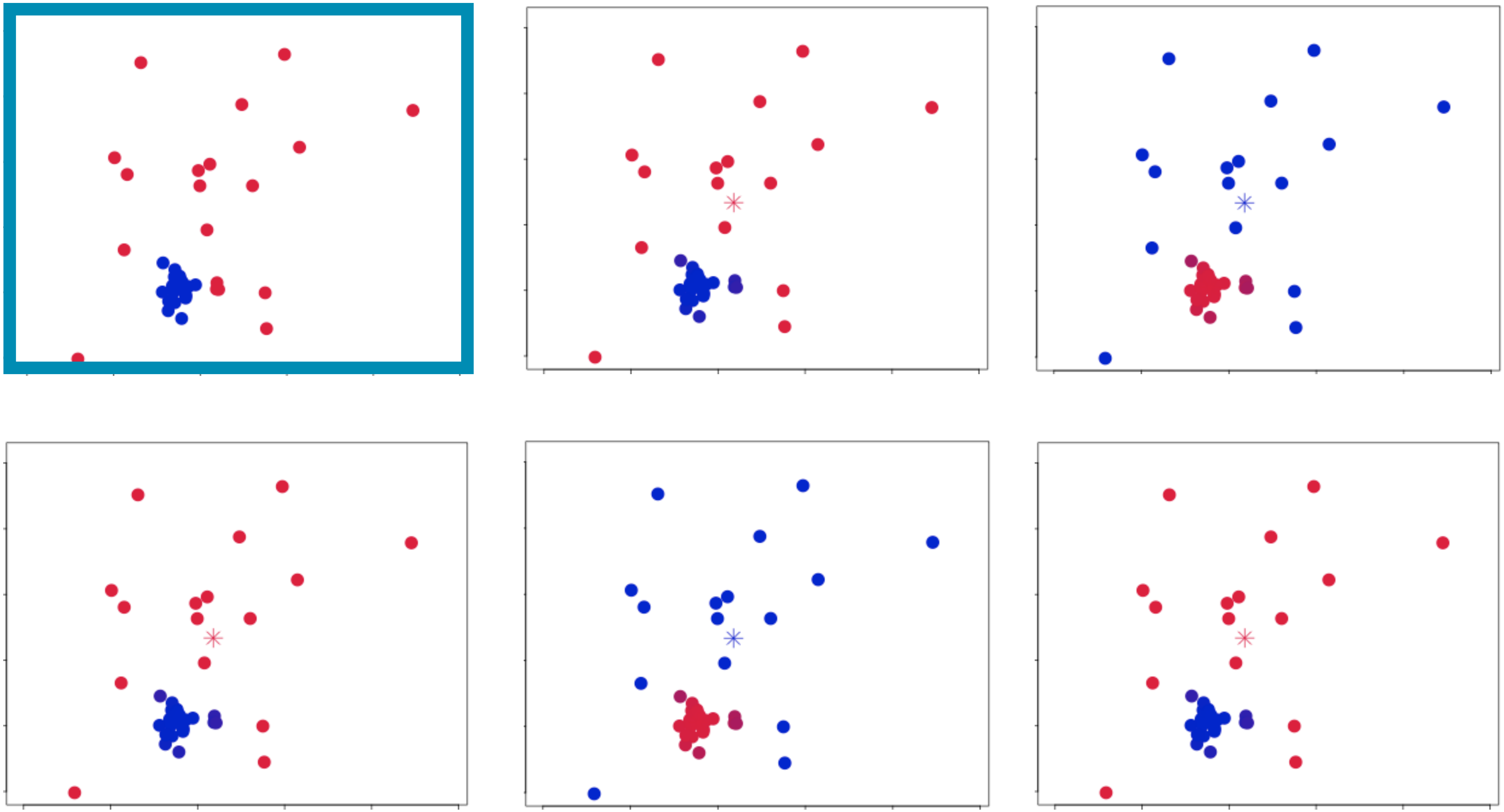
Corresponds to “version 3” algorithm on page 304 of MacKay (see readings)

How does MoG do?



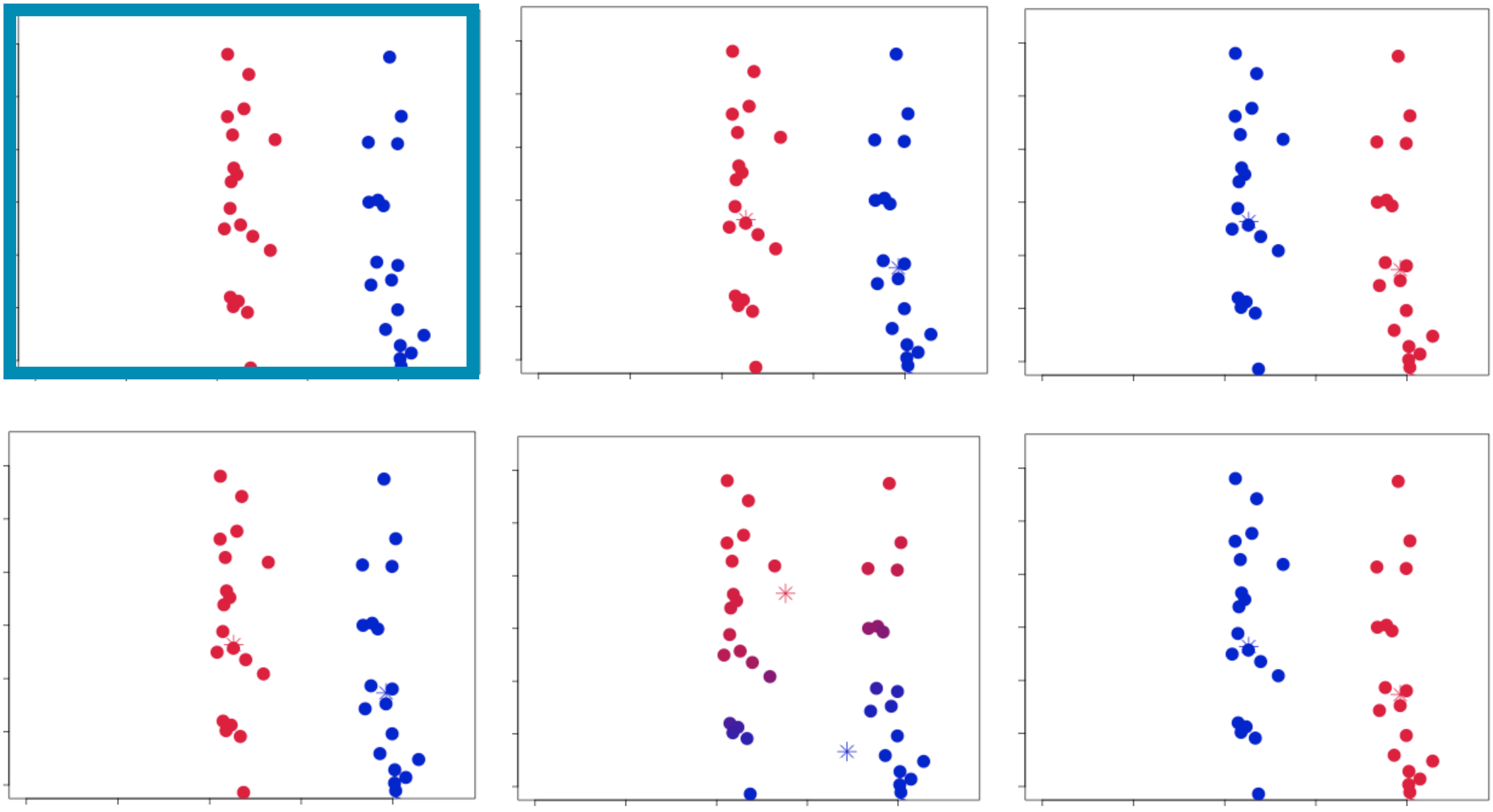
```
load('softkmeansdemo.RData')
mixtureofgaussians(d,3)
```

How does MoG do?



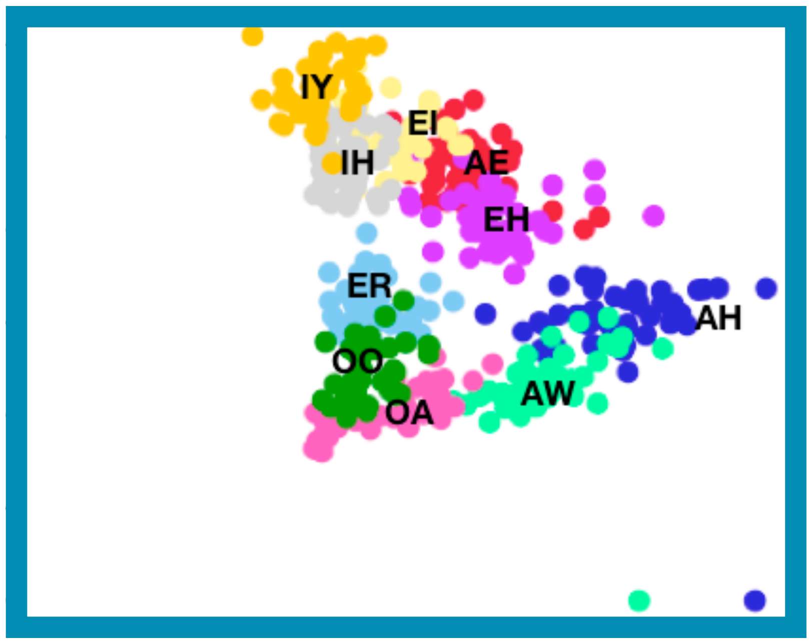
```
load('baddataset2.RData')
mixtureofgaussians(d,2)
```

How does MoG do?



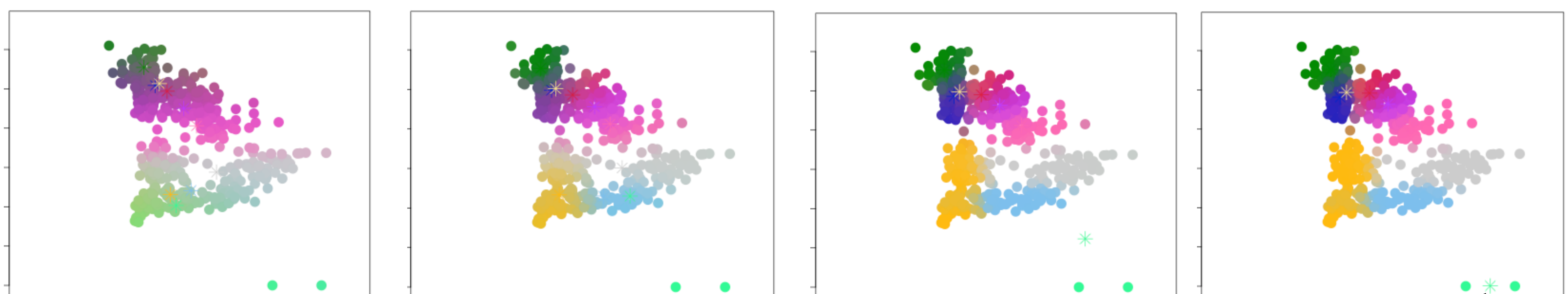
```
load('baddataset3.RData')
mixtureofgaussians(d,2)
```

How does MoG do on our phoneme data?



This error occurs because it is getting NaNs for the likelihood

```
> mixtureofgaussians(d,10)
Error in rgb(rgbvec) : color intensity nan, not in [0,1]
```

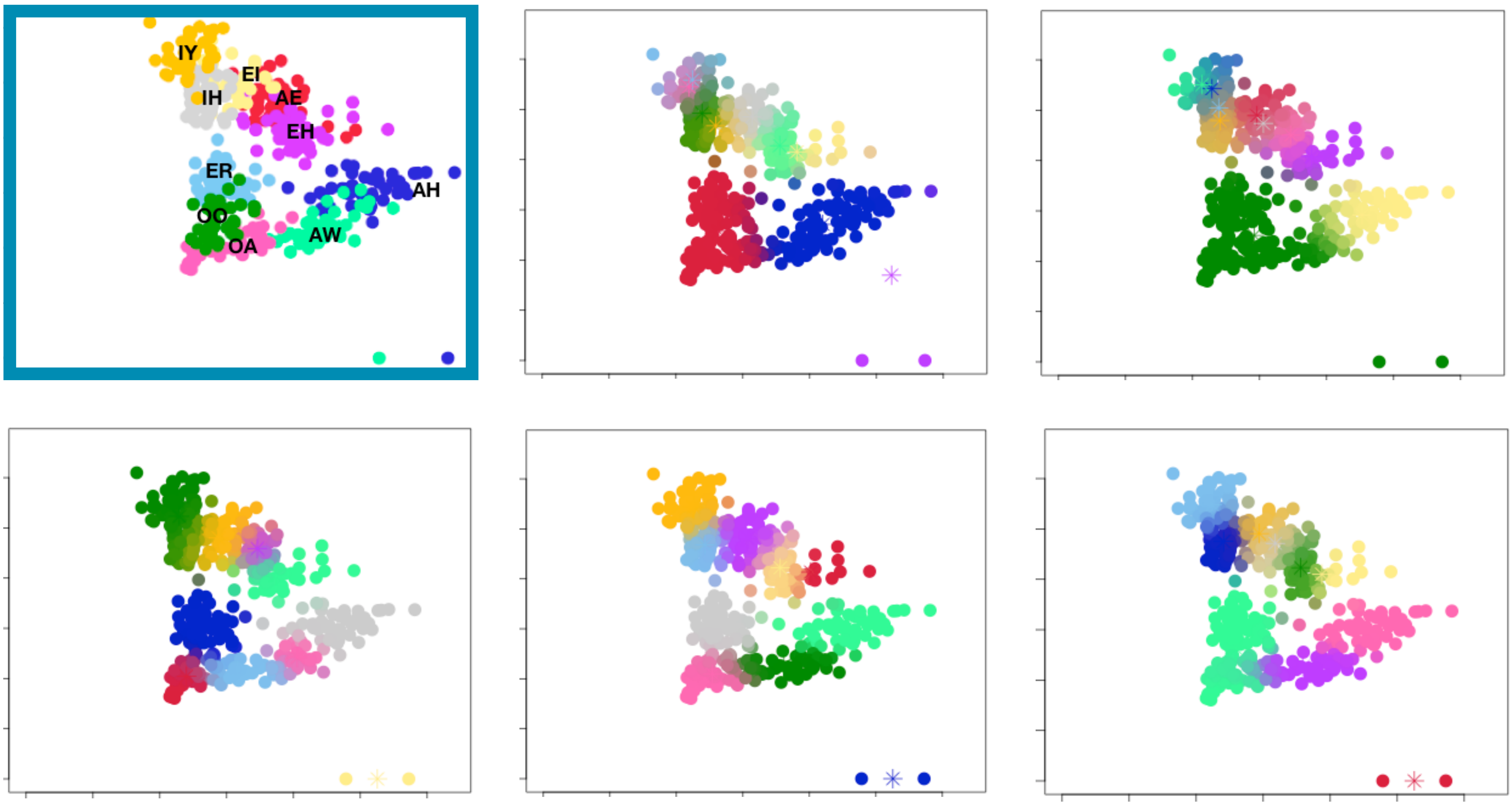


In trying to account for the two points at the bottom, it the variance in the y dimension to zero, resulting in infinite likelihood

```
load('phonemedata.RData')
mixtureofgaussians(d,10)
```

How does MoG do on our phoneme data?

kludge: just set it so the minimum variance can't go below some small constant (e.g., 0.001)



Good things about Mixture of Gaussians

- As with k-means, convergence to a local maximum is fast and guaranteed
- Performance is considerably better than k-means: can fit asymmetrc clusters of unequal variance
- Can handle soft assignment
- Interpretable probabilistically, in terms of maximising the likelihood of the dataset assuming the clusters are Gaussian

Bad things about Mixture of Gaussians

- As with k-means, not guaranteed to converge to a *global* maximum; still sensitive to initial conditions
  - You can especially see this if you set the initial variances too low or too high
- As with k-means, you have to tell it how many clusters there are
- Occasionally shows pathological behaviour in which (a) one cluster has infinitely small variance, or (b) all means are the same and all points shared among all clusters
  - Making it properly Bayesian by instead setting a prior on the variance can help here, and is more principled

A full model of phonetic learning

- MoG is vastly better, but still not great for a dataset as complicated as the phoneme data
- Existing models build on MoG in three ways:
  - Solving the local maximum problem: integrate over all possible solutions, don't just find a single best one given your starting point like EM
  - Solving the zero-variance problem: Set a prior over the means, variances, and weights
  - Learn how many categories would be appropriate through a special kind of prior on the # of categories; Dan will be talking about this in the next lecture!

Summary

- Although many things in life are supervised or semi-supervised, a number are completely unsupervised
- A very simple model of unsupervised clustering, k-means, is fast and okay but has several problems
  - Local maxima; sensitivity to starting conditions; can’t handle if the categories are not equal-sized and symmetric; have to tell it how many clusters; hard assignments
  - Adding soft assignments helps but doesn’t solve most problems
- Mixture of Gaussians with EM, which views the problem as finding the underlying Gaussian distributions, solves many of these problems, but not all
  - Local maxima; sensitivity to starting conditions; have to tell it how many clusters

Additional references (not required)

k-means clustering and mixture of Gaussians

- MacKay, D. (2003). *Information theory, inference, and learning algorithms.* Chapters 20 and 22.

Introduction to language / phonemes

- Kuhl, P. (2004). Early language acquisition: Cracking the speech code. *Nature Reviews Neuroscience 5*: 831-843.
- Chater, N., and Manning, C. (2006). Probabilistic models of language processing and acquisition. *Trends in Cognitive Science 10*(7): 335-344.