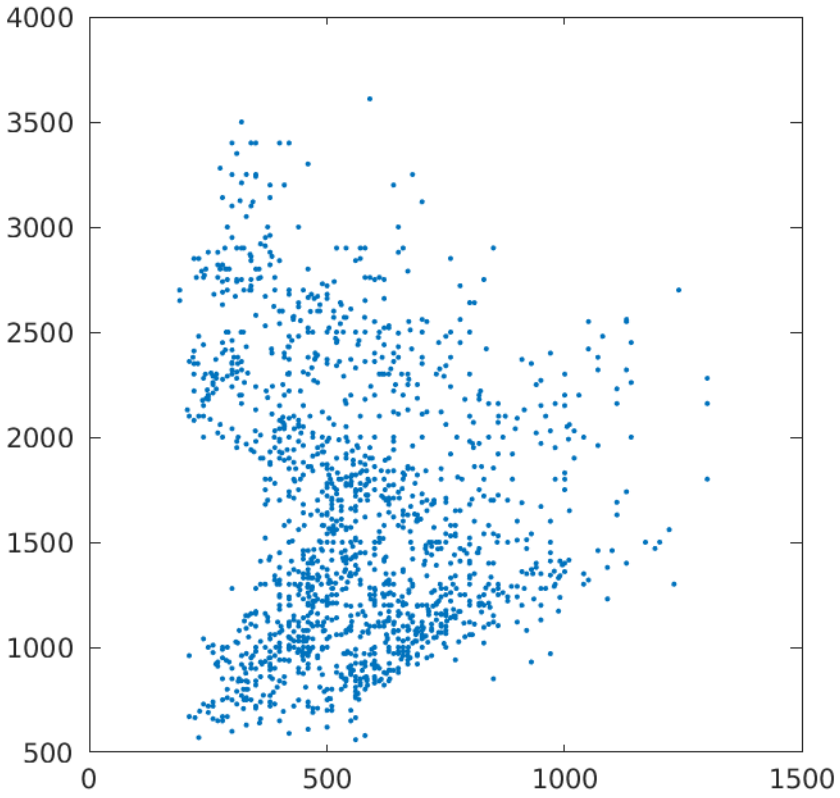
Task 1

The code for producing the plot is in its own script file task1.m



Task 2

The code for this task are in task2.m, the mog function was modified to export learned model parameters as variable s and to take in the number of components and data (x and k).

The learnt models are saved in p1_k3.m and p2_k3.m, the structure representing the model contains variables p representing the coefficient of the mixtures

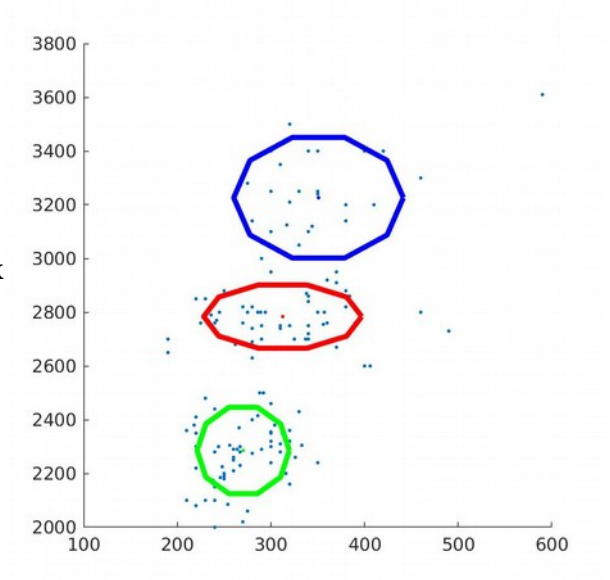


Illustration 1: gaussian components trained for phoneme 1

of gaussians , s2 the covariance matrices for the gaussian and mu the matrix of means for the gaussian components.

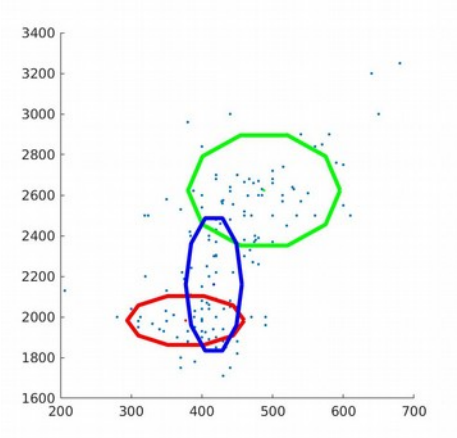


Illustration 2: Learned gaussian components for phoneme 2

Task 3

All the code for this task is in task3.m

The code first loads the filtered data supplied in `PB12.mat` , this dataset is partitioned into a training set for model 1 containing only the 80 % of points from X1 and the first 80% of points from X2 training set and a test set with the rest of the points.

initially accuracy varied because when simply selecting the first 80% of data depending on initialisation, but since the harmonics were ordered it was deemed better to take a random subset as a training set this is achieved by generating a random permutation of the indices for phoneme 1 data and phoneme 2 data and taking the first 80% (121 samples) as respective training data. This leaves the rest of the permutation indices are the data for the test set (62 samples) of which the first 31 are from phoneme 1 and the rest from phoneme 2.

This part can be seen in the first section **l1-18**

Models m1 and m2 were trained on training_set using mog.m in **l19-27**

As a convenience a new figure is generated to plot training data 2 and fitted Gaussian for illustration purposes in **line 22.**

The models were evaluated in the following way:

The Gaussian mixture model likelihood function (computing Sum of P(ck) X N(muk, Sk)) was extracted into **comp_mix_like.m**

for each sample in the first 31 samples of test set , we know this belongs to phoneme 1. hence comparing comp_mix_like(sample, m1) with comp_mix_like(sample,m2) yields a classification, if model 1 has a higher likelihood the confusion matrix is updated to reflect the result, and vice versa for samples 32-62 of test set. The results are very accurate (K=3) but this can vary slightly depending on random initialisation of training data (which also sets the covariance) and mus:

example 1 k = 3

confusion_mat =

30 0

1 31

mis_class_rate = 0.0161

example 2 k = 3

confusion_mat =

29 2

2 29

mis_class_rate =

0.0645

we can manually tweak line 20 and 26 to train a model with 6 components.

confusion_mat =

30 2

1 29

mis_class_rate =

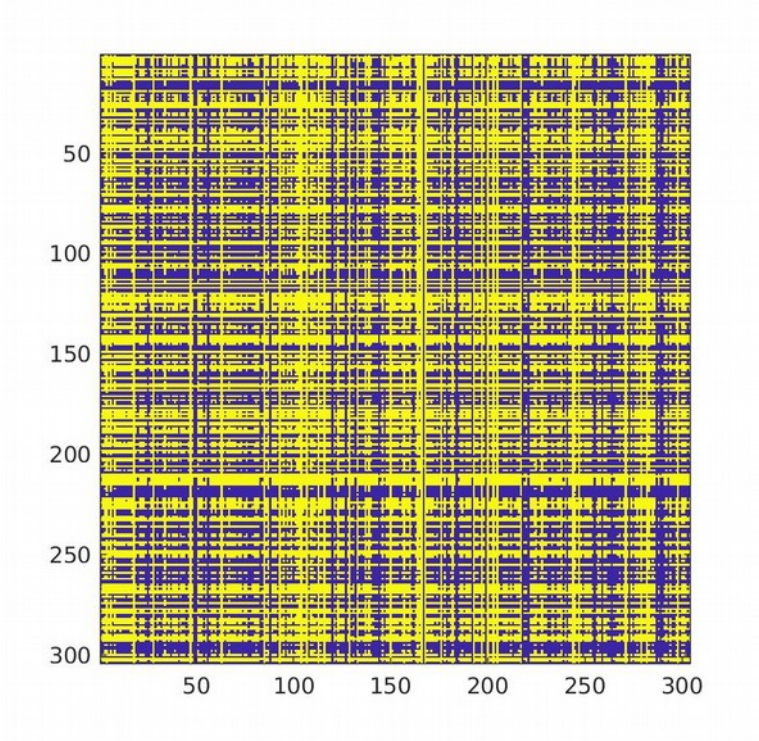
0.0484

Overall the accuracy is similar if not slightly better for k=3 but on more occasions the 6 component gaussian over-fits (singular gaussian are computed to working precision) which means that there are likely no more than 3-5 sub-processes underlying the model.

Task 4

The implementation of this task was done by loading the models learnt on full-data from task 2, a permutation of the full set of F1 column from phoneme 1 and phoneme 2 was compiled as x1 and the same was done with F2 for x2.. this generates random points that have the exact same values as previous datapoints but is a random machup of phoneme 1 and phoneme 2 data.

This data was then fed through the comp_mix_like function and the M classification matrix was created, here is a visual representation of it:



The datapoints that we have generated do not belong to any class, the two colours yellow and blue represent a 1 or 2 classification . Visually it appears that the classification is overall balanced in terms of total area of blue vs yellow. One striking aspect of the classification is that there are linear clusters that are more consistently classed as c2 vs c1. It is not possible to reason generally about this since the data is randomised except that for certain values one of dimension in the Gaussian is more distinctive between the two mixture models which leads to favouring one of the classifications along a horizontal line or vertical line.

Task 5

As instructed data J1 and J2 was constructed by adding a third co-dependent dimension to the data we're trying to fit. This is in task5.m **lines 2-5**

mog.m was adapted and all calls from other tasks to include a flag that determines if the full covariance matrix is used as opposed to the diagonal, it is set to false for all other tasks.

As a result training the mog on J1 and J2 rarely produced a model, the most common result of the computation is a singular gaussian (with 0 variance on the diagonal of covariance matrix).

One way to fix it is throughout the iterations of EM algorithm to ensure that a minimum value is set, this regularisation rate is a constant set as a minimum value for diagonal elements of covariance matrix.

After some trial and error it seems that only a very high value for the regulariser managed to converge well for the data.. this is because the initial covariance of the data is really high in the order of 10000, note the 10^4 factor in front of these initial Sigma matrices.

s2(:,:,1) =

1.0e+04 *

0.0666 0.0378 0.1044

0.0378 7.3570 7.3948

0.1044 7.3948 7.4993

s2(:,:,2) =

1.0e+04 *

0.1193 0.4037 0.5230

0.4037 4.6966 5.1004

0.5230 5.1004 5.6234

s2(:,:,3) =

1.0e+04 *

0.1193 0.4037 0.5230

0.4037 4.6966 5.1004

0.5230 5.1004 5.6234

The code for this step is in mog.m **lines 50-59**

One other way to reduce overfitting is to stop convergence earlier by detecting the decrease in changing rate of coeficients below a certain threshold.

The figure below shows the 3 3d gaussian mixture computed with regularisation , although the plot does not represent the 3d data.