6.867 : Homework 3

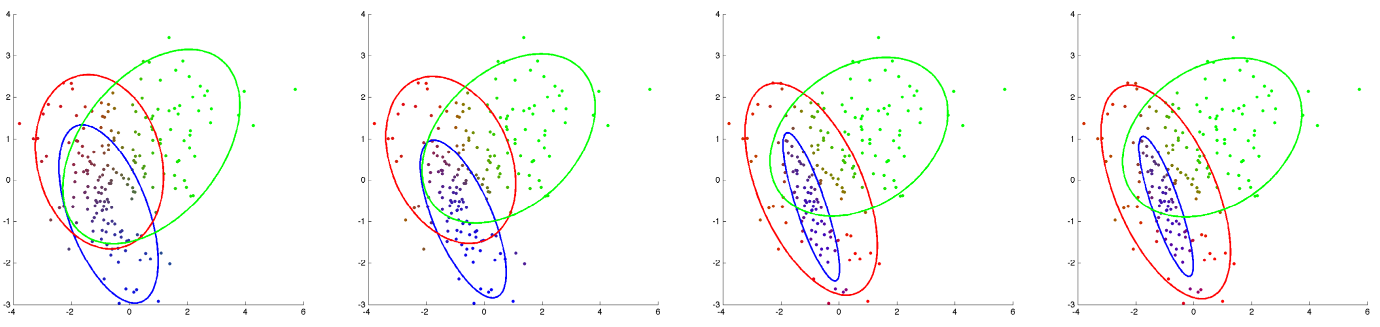
Anonynous authors

November 15, 2012

1 Expectation Maximization and its variations

1.1 Effects of the parameters of EM

We implemented the EM algorithm for Gaussian Mixtures Models as described in Bishop 9.2.2. The algorithm is somewhat long to describe so we won't report it here. In general, EM is an iterative algorithm to find an approximation to the maximum likelihood estimate for the parameters of a distribution that depends on some unobserved latent variables. In this paper we applied EM to estimate a Gaussian mixture model in which the latent unobserved variables are the assignment of each point to a particular component of the mixture. It can be shown that at each step the likelihood of the data, under the model learned by EM, increases. We assumed convergence when the difference between the log-likelihoods for two consecutive steps is below a threshold $t = 10^{-4}$ or when the number of iterations is greater than $it_{thresh} = 2.10^3$. We used the "logsumexp trick" in the computation of the log-likelihood. We ran our implementation for a few values of the convergence criterion. The algorithm usually converges in less than 200 steps and, in the vast majority of the cases we explored, before hitting the maximum number of iterations. Relaxing the convergence criterion on the log-likelihood, influences the tightness of the model fitting, and is akin to regularization. It naturally prevents from reaching higher likelihood, and obviously speeds up convergence. We report some plots of the model with 3 Gaussians on the *data_1_large* dataset on Fig 1 for several values of the convergence threshold[1]. All other results reported here are obtained with a value of $t = 10^{-4}$ which seemed to yield fairly repeatable results.



(a) $t = 1$, 2-*steps* (b) $t = 10^{-2}$, 8-*steps* (c) $t = 10^{-4}$, 48-*steps* (d) $t = 10^{-8}$, 121-*steps*

Figure 1: Effect of the convergence threshold t for dataset *data_1_large*. The caption indicates the number of EM steps required for convergence.

Changing the number of Gaussians in the mixture model (denoted by k) can strongly affect the distribution we learn with EM. We tried our implementation on the 3 datasets both *small* and *large* for $k = 1 \dots 5$. We initialized the EM algorithm with some arbitrary values of μ and set $\Sigma = I$. Fig. 2(a...e) shows the effects of increasing k on *dataset_1_large*. Fig. 2(f...j), shows the effect of varying the initial mixing coefficients $\pi = (\pi_1, \dots, \pi_k)$ on the same dataset.

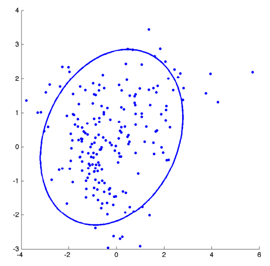
1.2 Evaluation of EM on the proposed datasets

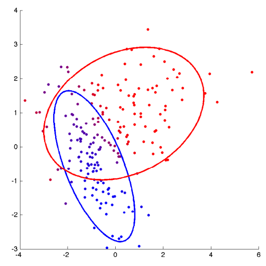
In order to evaluate the models learned with EM, we trained, on either the *small* or *large* set for datasets 1,2 and 3 and tested on the matching dataset of the opposite size. We scored our models using the average negative log-likelihood to be able to compare between test sets of different sizes. The resulting scores are reported in Table 1. The dataset name reported indicates the training set.

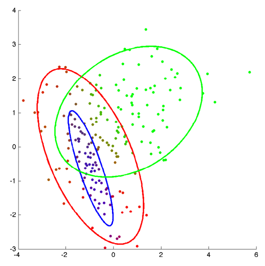
1.3 Variations on EM

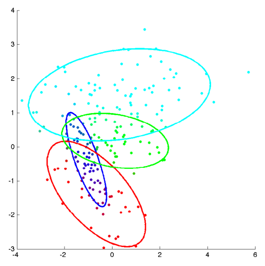
We explored two variations to the EM algorithm. First, we performed k-means clustering on the data and seeded the EM algorithm with the clusters' means for μ's and clusters' covariances for Σ. Second we constrained the covariance

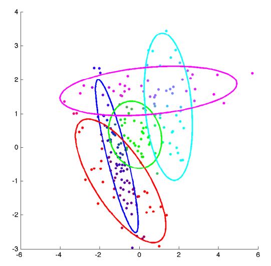
[1]The datapoints in all the following plots are colored with respect to their *responsibility* values as described by Bishop in Eq.(9.13).

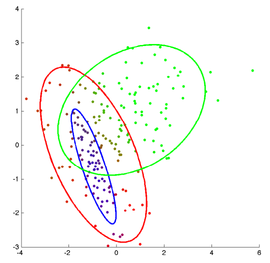

(a) $k = 1$

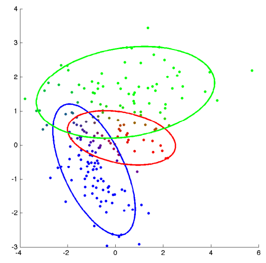

(b) $k = 2$

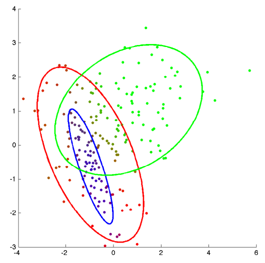

(c) $k = 3$

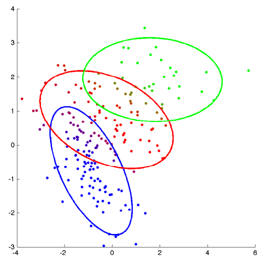

(d) $k = 4$


(e) $k = 5$


(f) $\pi = (\frac{1}{3}; \frac{1}{3}; \frac{1}{3})$


(g) *π randomly chosen*


(h) *π randomly chosen*


(i) *π randomly chosen*

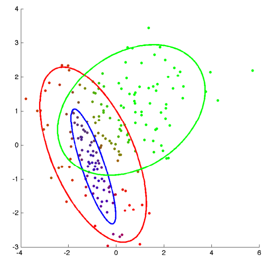
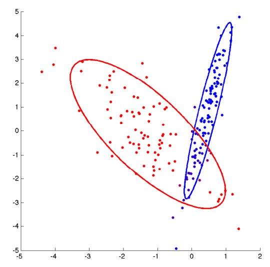
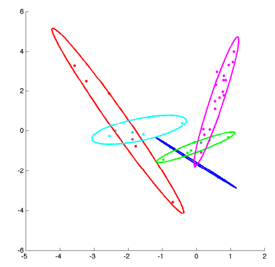
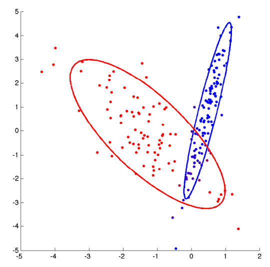

(j) *π randomly chosen*

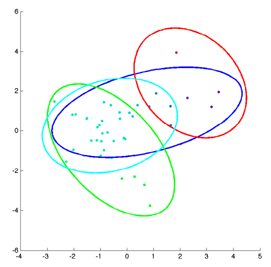
Figure 2: Varying the number of Gaussians and their mixing coefficients on the *data_1_large*. The top row is a change in k, the bottom row a random change in π for $k = 3$. The initializing μ, Σ are kept the same.

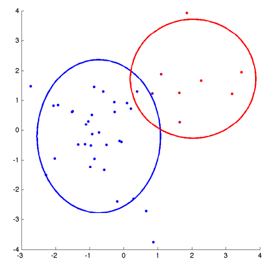
matrices to be diagonal. This restricts the class of GMM we can learn with EM to the class of GMMs in which each dimension is independent given the assignment to a component. In this class of models we have $\max_\Sigma p(X|\mu_k, \Sigma) = \max_{\sigma_1 \dots \sigma_d} \prod_{i=1}^{n} \prod_{i=1}^{d} \mathcal{N}(x_i^j|\mu_k^j, \sigma_k)$ where d is the dimensionality of the input variables x_i. In view of this, the maximum likelihood estimation (MLE) for the covariance matrix is the the diagonal matrix with each entry being the MLE of the variance for that particular dimension given the assignment. This discussion justifies modifying the EM algorithm by simply killing the off diagonal elements in the covariance estimation reported in Bishop (9.25) at every step. The complete performance of both methods is reported in Table 1. To help the reading of Table 1 we reported the best and worst models (evaluated as described above) for both arbitrary and k-means seeding in Fig. 3(a…d). In Fig 3(e…h) we show a few examples in which the Gaussians are constrained to be aligned with the main axes (diagonal covariances). Constraining the covariance matrices to be diagonal reduces the complexity of the model by lowering the effective number of parameters of the distribution, it also reduces the computational complexity making the algorithm faster. However, these constrained models compensate by favoring a higher number of Gaussians in the mixture model.

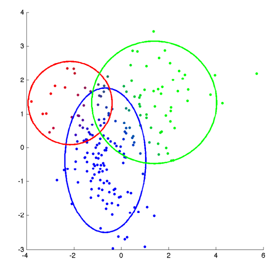

(a) *Arbitrary, Best,* $k = 2$, *data_2_large*

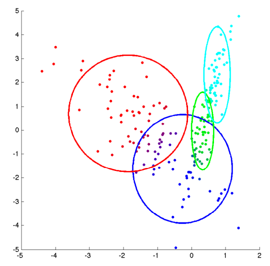

(b) *Arbitrary, Worst,* $k = 5$, *data_2_small*


(c) *K-means, Best,* $k = 5$, *data_2_large*


(d) *K-means, Worst,* $k = 4$, *data_1_small*


(e) *Diagonal, data1small,* $k = 2$


(f) *Diagonal, data1large,* $k = 3$.


(g) *Diagonal, data2large,* $k = 4$.

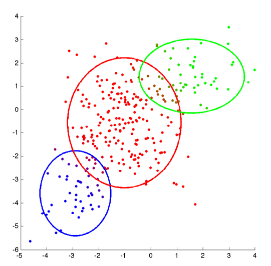

(h) *Diagonal, data3large,* $k = 3$.

Figure 3: Top row : best and worst performances achieved using a random then a k-means initialization. Bottom row : examples of models with diagonal covariance matrices.

Arbitrary seeding	k=1	k=2	k=3	k=4	k=5
large 1	3.5317	3.2731	**3.2517**	3.3541	3.2929
small 1	**3.4919**	3.5587	3.8749	4.1355	4.7795
large 2	3.6594	**2.7892**	2.8329	2.8027	2.9889
small 2	3.6063	**2.8705**	3.4865	3.5431	5.3917
large 3	3.7543	3.7133	**3.6731**	3.6917	3.7050
small 3	**3.7941**	3.8774	3.8652	4.0387	4.1069

k-means seeding	k=1	k=2	k=3	k=4	k=5
large 1	3.5317	3.2730	3.3802	3.3284	**3.2152**
small 1	**3.4919**	3.5046	3.7398	4.0879	4.0042
large 2	3.6589	**2.8238**	2.8461	2.8583	2.8651
small 2	3.6098	3.6085	3.2259	**3.2179**	3.3540
large 3	3.7543	3.7132	**3.6833**	3.6916	3.6887
small 3	3.8041	3.7847	**3.7660**	**3.7660**	**3.7660**

diagonal covariance	k=1	k=2	k=3	k=4	k=5
large 1	3.5679	3.4305	3.3821	3.3997	**3.3594**
small 1	3.5177	**3.4381**	3.4840	3.5026	3.7234
large 2	3.6675	3.2604	3.0571	2.9961	**2.9819**
small 2	3.6055	3.5709	**3.2748**	3.3087	3.4390
large 3	3.9297	3.8146	3.7113	3.7124	**3.7073**
small 3	3.9587	3.8741	**3.8258**	4.0209	4.0115

Table 1: Negative log-likelihood scores for all datasets using an arbitrary initialization and a k-means initialization. Lower is better. The dataset reported are the training datasets, we tested on the matching dataset of the opposite size.

2 Model Selection

The generalization performance of EM as applied on GMM depends on the number of components we allow in the distribution. In Table 2 we report the average negative log-likelihood of the training datasets under the model learned[2]. Just like we would expect, models with more components make the training set more likely essentially over fitting it. We can say this by comparing these results with the "small" rows of Table 1 where the "large" datasets are used for testing. To get a better sense of the generalization performance of our models, we implemented a n-folds cross-validation method.

Diagonal Covariances	k=1	k=2	k=3	k=4	k=5
small 1	3.5451	3.3436	3.1456	3.0252	**2.9478**
small 2	3.6598	3.4867	2.9216	2.7772	**2.6102**
small 3	3.9062	3.7546	3.5819	3.4808	**3.4348**

Full Covariances	k=1	k=2	k=3	k=4	k=5
small 1	3.5076	3.0978	2.9783	2.8595	**2.6411**
small 2	3.6484	2.6994	2.4808	2.3921	**2.2041**
small 3	3.5076	3.0978	2.9783	2.8595	**2.6411**

Table 2: Negative log-likelihood of the training set under the model learned by EM.

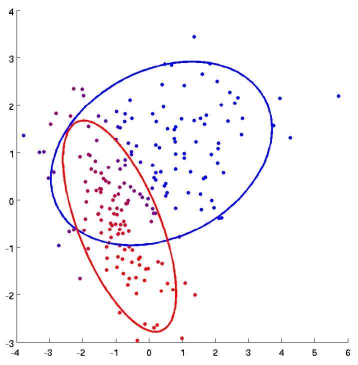
During training, we randomly reshuffle the input dataset and then split it into n equal parts. We use 1 of these parts as a validation set and the remaining $n-1$ as training set. For each *fold* we computed the average log-likelihood $\mathcal{L}$ on the validation set and then averaged over the *folds*. We repeated this procedure m times, changing the initial permutation of the dataset and kept as cross-validation score the best average $\mathcal{L}$ among the m reshuffles. We summarized our procedure in Algorithm 1.

We tried different values for n and finally settled for 10-fold cross-validation for the large datasets and leave-one-out ($n = |dataset|$) for the small training sets. This accounts for the respective sizes of the sets and gives repeatible results. We settled for $m = 1$ for the small datasets and $m = 100$ on the large datasets. We tested our cross-validation procedure by training and validating on either the *large* or *small* version of each dataset and testing on the other one. This enabled us to pick a best model for each dataset. Our findings are summarized on Fig 4 and Table 3. Compared to the naive ranking of the models, cross-validation provides a measure for generalization perfomance, and counterbalances overfitting. It favors moderately complex models that generalize better. We used unconstrained covariances for our cross-validation experiments.

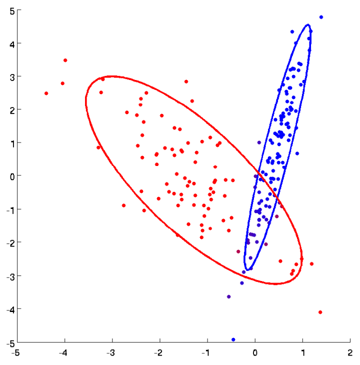
[2]In all the experiments in this section we seeded EM with k-means as described in Section 1.

```
foreach candidate model (i.e. choice of $k$) do
    for $m$ times do
        reshuffle the dataset
        foreach fold do
            v-fold $\leftarrow$ current fold
            train on remaining $(n-1)$ folds
            validate on v-fold recording the average log-likelihood $\mathcal{L}$
        end
        compute the average of $\mathcal{L}$ over the folds
    end
    record the maximum average $\mathcal{L}$ over the $m$ runs as score for this candidate model
end
choose the candidate model with the highest score
```

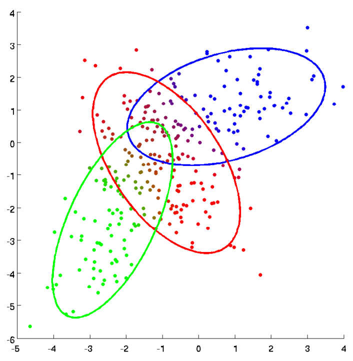
Algorithm 1: Our Cross-Validation procedure.



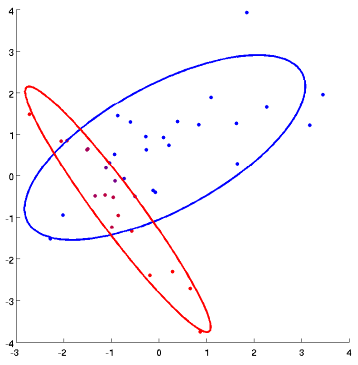
(a) *dataset_1_large*, $k = 2$



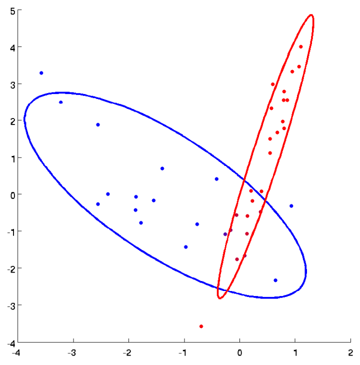
(b) *dataset_2_large*, $k = 2$



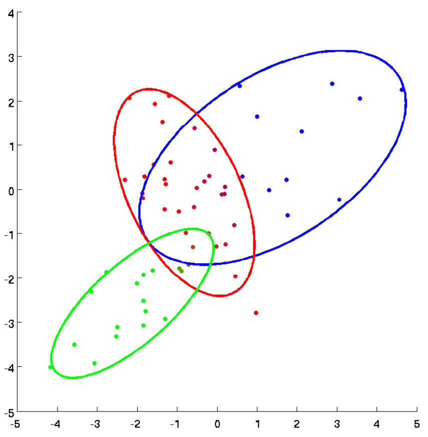
(c) *dataset_3_large*, $k = 3$



(d) *dataset_1_small*, $k = 2$



(e) *dataset_2_small*, $k = 2$



(f) *dataset_3_small*, $k = 3$

Figure 4: Best models determined by our cross-validation procedure on all given sets. The log likelihoods are reported in Table 3

Data Set	Large	Small
1	$k = 2$, $\mathcal{L} = 3.272723$	$k = 2$, $\mathcal{L} = 3.556298$
2	$k = 2$, $\mathcal{L} = 2.789503$	$k = 2$, $\mathcal{L} = 2.870081$
3	$k = 3$, $\mathcal{L} = 3.672748$	$k = 3$, $\mathcal{L} = 3.869479$

Table 3: Summary of the best performing models using our cross-validation procedure. The *Large* and *Small* columns indicate the dataset used for training and validation. The performance reported is the negative average log-likelihood for the matching dataset of opposite size. We used unconstrained covariances.

3 Predictions

In order to estimate the GMM on the *mystery* datasets, we used the cross-validation procedure described in Section 2 with 10-folds. We repeated this procedure twice: with used random initialization of the mixture parameters and with the k-means initialization described in Section 1. We used a regularization parameter of $\lambda = 10^{-2}$ for the covariance matrices i.e., at each step of EM we set $\Sigma = \Sigma + \lambda I$ this step limits the tightness of the Gaussians around the data points which makes sense given the small size of the training sets. We repeated this procedure for diagonal and unconstrained covariance matrices. For all three sets, the best scores were obtained by using a k-means initilialization and unconstrained covariances, they are reported in Table 4 for models with $k = 1, \ldots, 5$. The best fits are illustrated on Fig. 5 along with their respective parameters. Given the scores reported on Table 4, we discarded the diagonal models. Here are the final covariance matrices :

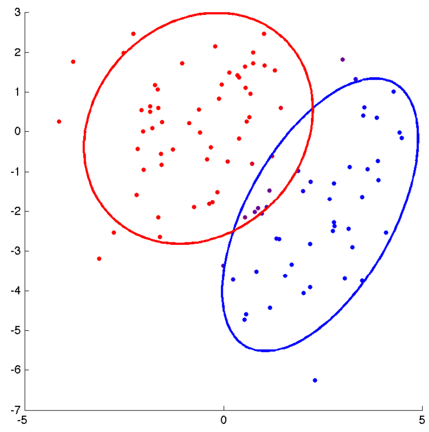
$$\textit{Mystery 1} \quad \Sigma_1 = \begin{pmatrix} 1.52 & 1.19 \\ 1.19 & 2.95 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 2.07 & 0.34 \\ 0.34 & 2.11 \end{pmatrix}$$

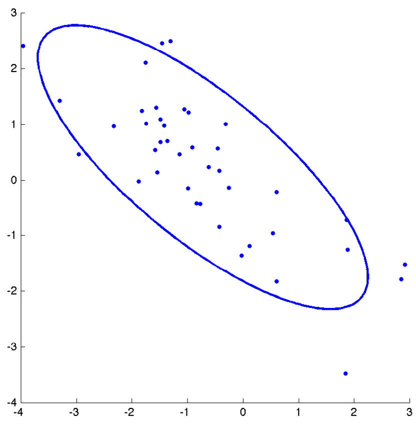
$$\textit{Mystery 2} \quad \Sigma_1 = \begin{pmatrix} 1.52 & 1.19 \\ 1.19 & 2.95 \end{pmatrix}$$

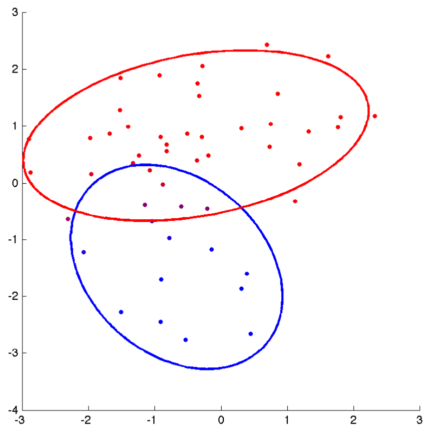
$$\textit{Mystery 3} \quad \Sigma_1 = \begin{pmatrix} 0.64 & -0.23 \\ -0.23 & 0.84 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 1.70 & 0.28 \\ 0.28 & 0.54 \end{pmatrix}$$

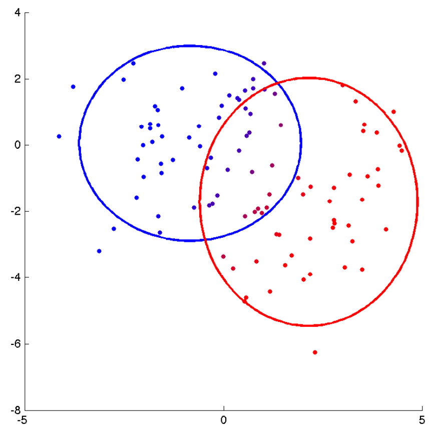
	k=1	k=2	k=3	k=4	k=5	**selected** k
Mystery 1	4.22	**4.19**	4.23	4.28	4.32	**2**
Mystery 1 - diag	4.24	**4.20**	4.24	4.27	4.23	**2**
Mystery 2	**3.14**	3.23	3.36	3.49	3.19	**1**
Mystery 2 - diag	3.63	**3.45**	3.52	3.99	3.96	**2**
Mystery 3	3.33	**3.25**	3.45	3.75	4.18	**2**
Mystery 3 - diag	3.34	**3.31**	3.41	3.57	3.64	**2**

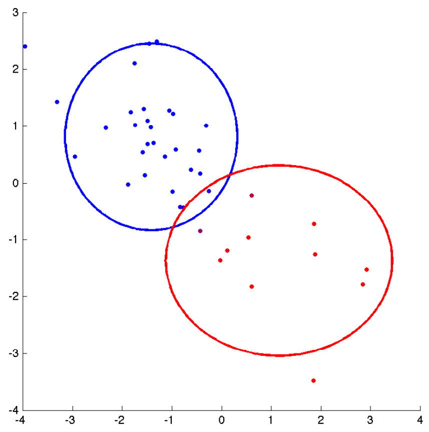
Table 4: Minimum negative log-likelihood-per-data-point scores for each model (lower is better), averaged on 10-folds cross-validation. The minimum value is taken on 100 trials using k-means initialization and 100 trials using random initialization. We picked our prediction based on these scores.


(a) *Mystery 1* - $\mu_1 = (2.43; -2.09)$, $\mu_2 = (-0.63; 0.09)$, $\pi = (0.43; 0.57)$


(b) *Mystery 2* - $\mu = (-0.720.23)$, $\pi = 1$


(c) *Mystery 3* - $\mu_1 = (-0.69; 0.85)$, $\mu_2 = (-0.38; -1.43)$, $\pi = (0.27; 0.73)$


(d) *Mystery 1 - diag*


(e) *Mystery 2 - diag*

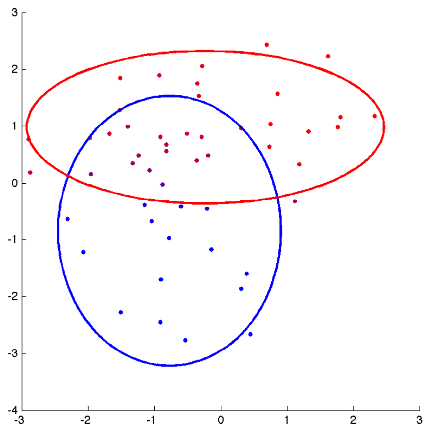

(f) *Mystery 3 - diag*

Figure 5: Our predictions on the mystery datasets. These correspond to the models with the best scores reported in Table 4. The Gaussian with index 1 is in blue. The diagonal covariances models are included in the bottom row, just for comparison. For $Mystery\_2$, we can see that a higher number of Gaussians compensate the lower complexity induced by diagonal covariances, in order to achieve similar likelihood.