HW2 - REPORT

Fabien Pesquerel*, Thibault Lahire*
*ENS Paris-Saclay, 94230 Cachan, FRANCE
fabien.pesquerel@ens.fr
thibault.lahire@student.isae-supaero.fr

Notation
We note $\mathcal{N}(x|m, \Sigma)$ the density function of a normal random variable of parameters (m, Σ) evaluated at point x.

---

1 Classification: K-means, and the EM algorithm

Consider a mixture model, with K components, where datapoints X_i, $i = 1, ..., n$ have a probability p_k to be in component k: $P(Z_i = k) = p_k$, and, conditional on $Z_i = k$, $X_i \sim N(\mu_k, D_k)$, a multivariate Gaussian distribution with mean μ_k, and diagonal (not full) covariance matrix D_k.

1. Derive the expressions of the parameters (p_k, μ_k, D_k) at each iteration of the corresponding EM algorithm. (Explain your derivations.)

Before diving into computation one can make a few preliminary remarks. For the more general setting, it is easy and very intuitive that the EM steps for a full mixture of Gaussian can be written thanks to the following formula. The **E step** estimates how likely it is that a given data point belongs to a cluster that is represented thanks to a latent variable. To the contrary of MLE methods that we studied in the previous homework, one can see that EM keeps a bit of information about all points in all cluster but that the kept information decreases according its density of a Gaussian distribution. Thus, for points that are in between two clusters, EM will not be really confident about the latent variable of that point while it is the case for more categorical methods. Here θ_t is the set of estimators of the parameters of the model at step t. f_θ is the probability distribution of the mixture model.
E step:

$$f_{\theta_t}(z = k \mid x) = \frac{\mathcal{N}\left(x \mid \mu_k^t, \Sigma_k^t\right) p_k^t}{\sum_{k=1}^K p_k^t \mathcal{N}\left(x \mid \mu_k^t, \Sigma_k^t\right)}$$

E-step can be interpreted simply as a weighted sum of Gaussian densities weighted by the prior on the cluster. An MLE method would have estimate this parameters as the arg-maximum over the Gaussian densities and we would have lost information. Thus this quantity can be seen as a relaxed version of the belonging of a point to a cluster. This is a Bayesian point of view where we update our prior of the class of a point without getting rid of all others possibilities.
M step: Computations from the M-step are also easy to interpret. The new estimate for the probability of belonging to a cluster (the p_j) is a relaxed version of the frequency of points belonging to a cluster. A point in between clusters would weight half-half for each clusters for instance. The centers of Gaussians are estimated as the barycenters of all points where the weights are soft-belonging to the cluster. This is the same interpretation for covariance matrices.

$$
\begin{aligned}
p_j^{t+1} &= \frac{1}{n}\sum_{i=1}^{n} f_{\theta_t}(z=j \mid x_i) && \forall j \\
\mu_j^{t+1} &= \frac{\sum_{i=1}^{n} f_{\theta_t}(z=j \mid x_i) x_i}{\sum_{i=1}^{n} f_{\theta_t}(z=j \mid x_i)} && \forall j \\
\Sigma_j^{t+1} &= \frac{\sum_{i=1}^{n} f_{\theta_t}(z=j \mid x_i)(x_i-\mu_j^{t+1})(x_i-\mu_j^{t+1})^T}{\sum_{i=1}^{n} f_{\theta_t}(z=j \mid x_i)} && \forall j
\end{aligned}
$$

However, numerically computing those covariance matrices and inverting them can be slow (cubic complexity in the general setting). That is why we may be interested in a simplified model where we consider diagonal covariance matrices. Below, we do the full computation but we can remark that, given the general setting, it is possible to avoid the computation by seeing that the maximization step with respect to the covariance matrix can be written as a convex optimization problem, *i.e.*, the problem is convex with respect to the covariance matrix. The set of PSD matrices is a convex cone and the set of PSD diagonal matrices is a subset of the PSD matrices that is also a convex cone. Therefore, one use the projected gradient algorithm and immediatly say that the solution that will be found by the upcomming computation would simply be the projection of the full covariance matrix over the diagonal one, that is to say, we will only keep the update of diagonal elements. We also see that despite the fact that the problem is constrained on the subset of PSD matrices, this constraint is never active and we can avoid using a Lagrangian function for this constraint in the following derivation. Now, let's dive into a bit of calculus.

We consider a mixture model with K components. There are n datapoints $X_i \in \mathbb{R}^d$. Each datapoint X_i has a probablility p_k to be in component k: $\mathbb{P}(Z_i = k) = p_k$ and we have $X_i|Z_i = k \sim \mathcal{N}(\mu_k, D_k)$.

$$\text{We write} \quad p(x|Z_i=k) = \frac{1}{(2\pi)^{d/2}|D_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^\top D_k^{-1}(x-\mu_k)\right)$$

We consider the vector of parameters $\theta = (p_1, ..., p_K, \mu_1, ..., \mu_K, D_1, ..., D_K)$.

NB : To be precise, we don't need to put p_K in the vector θ, because p_K can be found thanks to $p_1, ..., p_{K-1}$ from $\sum_{k=1}^{K} p_k = 1$.

We write the likelihood :

$$l_{(x_i)_i}(\theta) = \prod_{i=1}^{n} p(x_i;\theta) = \prod_{i=1}^{n}\sum_{k=1}^{K} p(x_i, z_i=k;\theta) = \prod_{i=1}^{n}\sum_{k=1}^{K} p(z_i=k)p(x_i|z_i=k;\theta)$$

We can therefore compute the log-likelihood :

$$L_{(x_i)_i}(\theta) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} p_k \mathcal{N}(x_i|\mu_k, D_k)\right)$$

To maximize the log-likelihood (in order to find the MLE of θ), we use the EM algorithm. We introduce $\mathcal{L}$ a lower bound on $\log\left(\sum_{k=1}^{K} p_k \mathcal{N}(x_i|\mu_k, D_k)\right)$ for a given i.

$$\mathcal{L}(i, q, \theta^{(t)}) = \sum_{k=1}^{K} q(z_i=k)\log p(z_i=k|x_i;\theta^{(t)}) - \sum_{k=1}^{K} q(z_i=k)\log q(z_i=k)$$

1 - **E-step**: Given $\theta^{(t)}$, we know that $q(z_i = k) = p(z_i = k|x_i; \theta^{(t)})$ corresponds to maximizing $\mathcal{L}$ with respect to q. We introduce :

$$q_{k,i}^{(t+1)} = p(z_i = k|x_i; \theta^{(t)}) = \frac{p(z_i = k; \theta^{(t)})p(x_i|z_i = k; \theta^{(t)})}{p(x_i; \theta^{(t)})} = \frac{p(z_i = k; \theta^{(t)})p(x_i|z_i = k; \theta^{(t)})}{\sum_{l=1}^{K} p_l^{(t)} \mathcal{N}(x_i|\mu_l^{(t)}, D_l^{(t)})}$$

We note $w_k^{(t+1)} = \sum_{i=1}^{n} q_{k,i}^{(t+1)}$, then $\sum_{k=1}^{K} w_k^{(t+1)} = n$

2- **M-step**:

$$\begin{aligned}
\theta^{(t+1)} &= \arg\max_{\theta} \sum_{i=1}^{n} \mathcal{L}(i, q^{(t+1)}, \theta) \\
&= \arg\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} q_{k,i}^{(t+1)} \log p(z_i = k, x_i; \theta) \\
&= \arg\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} q_{k,i}^{(t+1)} \left(\log p_k + \log \mathcal{N}(x_i|\mu_k, D_k)\right) \\
&= \arg\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} q_{k,i}^{(t+1)} \left(\log p_k - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|D_k| - \frac{1}{2}(x_i - \mu_k)^\top D_k^{-1}(x_i - \mu_k)\right)
\end{aligned}$$

We have to optimize the function under the $\arg\max_\theta$ under the constraint $\sum_{k=1}^{K} p_k = 1$ (associated to the Lagrange multiplier ν). We write the Lagrangian :

$$\begin{aligned}
Lag(\theta, \nu) &= Lag((p_k), (\mu_k), (D_k), \nu) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} q_{k,i}^{(t+1)} \left(\log p_k - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|D_k| - \frac{1}{2}(x_i - \mu_k)^\top D_k^{-1}(x_i - \mu_k)\right) + \nu\left(1 - \sum_{k=1}^{K} p_k\right)
\end{aligned}$$

Now, we set the gradient of Lag to zero. We obtain for p_k :

$$0 = \frac{1}{p_k} w_k^{(t+1)} - \nu \qquad \forall\ (p_k)_{k\in[\![1;K]\!]}$$

However,

$$\nu = \nu \sum_{k=1}^{K} p_k = \sum_{k=1}^{K} \nu p_k = \sum_{k=1}^{K} w_k^{(t+1)} = n$$

Hence,

$$\forall\ (p_k)_{k\in[\![1;K]\!]},\ p_k^{(t+1)} = \frac{1}{n} w_k^{(t+1)}$$

Now, we set the gradient of Lag to zero with respect to (μ_k) :

$$\forall\ (\mu_k)_{k\in[\![1;K]\!]},\ 0 = -\sum_{i=1}^{n} q_{k,i}^{(t+1)} D_k^{-1}(\mu_k - x_i) = -w_k^{(t+1)} D_k^{-1}\mu_k + D_k^{-1}\sum_{i=1}^{n} q_{k,i}^{(t+1)} x_i$$

Hence,

$$\forall\ (\mu_k)_{k\in[\![1;K]\!]},\ \mu_k^{(t+1)} = \frac{1}{w_k^{(t+1)}} \sum_{i=1}^{n} q_{k,i}^{(t+1)} x_i$$

We now introduce the notations $D_k = \text{diag}(\delta_{k,1}, ..., \delta_{k,d})$ for all k, and the vector $g_{k,i} = (x_i - \mu_k)$. The j-th component of vector $g_{k,i}$ is denoted: $g_{k,i,j} = (x_i - \mu_k)_j$. Remark :

$$(x_i - \mu_k)^\top D_k^{-1}(x_i - \mu_k) = g_{k,i}^\top D_k^{-1} g_{k,i} = \sum_{j=1}^{d} g_{k,i,j}^2/\delta_{k,j} \qquad \text{and} \qquad |D_k| = \prod_{j=1}^{d} \delta_{k,j}$$

We then have $Lag(\theta, \nu) = Lag((p_k), (\mu_k), (D_k), \nu) = Lag((p_k), (\mu_k), (\delta_{k,j}), \nu)$. Hence:

$$0 = \sum_{i=1}^{n} q_{k,i}^{(t+1)} \left(-\frac{1}{2}\frac{1}{\delta_{k,j}} + \frac{1}{2}\frac{1}{\delta_{k,j}^2} g_{k,i,j}^2 \right)$$

Therefore:

$$\delta_{k,j}^{(t+1)} = \frac{1}{w_k^{(t+1)}} \sum_{i=1}^{n} q_{k,i}^{(t+1)} (x_i - \mu_k)_j^2$$

2. What may be the advantage of such a model, compared to the more standard Gaussian mixture model, where covariance matrices are full?

As we stated above, inverting a matrix has a cubic complexity with the size of the matrix while inverting a PSD diagonal matrix is linear in the size of the matrix. This is a much faster and tractable problem, especially for highly dimensional problem. In terms of computational complexity, there is a clear advantage toward this more restricted Gaussian mixture model than the general one.

Furthermore, in the more general setting, it might be that a covariance matrix degenerates so as to consider only one point. Indeed, the solutions to the problem we aim at solving with the EM algorithm are at the boundary of the domain and if we do not take care, we might get really bad solutions from the more general setting. This more constrained Gaussian mixture can still converge with some bad behaviour but is less prone to this kind of behaviour and can thus be seen as an extreme regularization of the problem. Furthermore, in high dimension, all the points can be seen as far from each others and it might therefore be better to over constrained the model than propagate bad gradients from outliers.

As we can see in the pictures of (Q3) (see Fig. 10 and Fig. 11), when the covariance matrix is diagonal, the semi-major axis and the semi-minor axis of the confidence ellipsis are aligned with the canonical base of the space (in the 2D-case the classical x- and y-axis). When the covariance matrix is full, the confidence ellipsis can have any shape (any eigenvectors and (positive) eigenvalues for the covariance matrix).

As a consequence, if we know that our data are structured, in particular, if we know that the different classes we are dealing with are orthogonal, then using a diagonal covariance matrix is a good idea. Indeed, if the classes are orthogonal but not align with the classical axes, we can preprocess our data to align the different classes with the classical axes. Moreover, using a diagonal covariance matrix means a set of parameters of smaller cardinality, which may means a better (= a lower) AIC score if the classes are orthogonal.

The AIC score is computed as follows : $AIC = 2\dim(\theta) - 2\ln L$, with L the likelihood. In the case of a diagonal covariance matrix, we have $\dim(\theta) = K + Kd + Kd = K(2d+1)$. In the case of a full covariance matrix, we have $\dim(\theta) = K+Kd+K(d(d+1)/2) = K(1+d+d(d+1)/2)$.

In the case of the IRIS dataset (Q3), we obtain a good result with a diagonal covariance matrix, and a better result with a full covariance matrix, both in terms of purity (i.e. quality of the clusters found) and in terms of AIC scores (the GM with full diagonal matrix provides a lower AIC score despite the higher number of parameters to tune. In this case, the parameters outside the diagonal made the likelihood increase significantly, resulting in a better AIC score).

3. Implement the algorithm, and compare the results with:

- **K-means**
- **EM for a standard Gaussian mixture (full covariance)**

for the IRIS data set, and $K = 2, 3, 4$. Note that the data set is available in scikit-learn and that these two methods are also implemented in that module. (EM for a Gaussian mixture with diagonal covariance matrices is also implemented, but please do your own implementation!) Please provide meaningful plots to help the comparison, i.e. scatter plots for every pair of dimensions, with clusters represented with different colors, and even better with a sur-imposed ellipsis. (What should the ellipses represent?)

In the presented figures, the color assigned to the datapoints represents the class inferred by the clustering algorithm. The color does not refer to the true label.

The crosses represent the centers of the clusters and the ellipses show 95% confidence areas.

Table 1 shows the AIC score (when it exists...) and the purity of clusters found by the 3 algorithms in the cases $K = 2, 3, 4$.

Table 1: Comparisons of the three algorithms in terms of time

ALGO:	K-Means	EM (full)	EM (diag)
$K = 2$, purity	0.67	0.67	0.67
$K = 2$, AIC score	None	487	808
$K = 3$, purity	0.89	0.97	0.91
$K = 3$, AIC score	None	448	668
$K = 4$, purity	0.88	0.96	0.90
$K = 4$, AIC score	None	452	602

See Fig. 1 to 9.

4. In which situations K-means is going to be significantly outperformed by the two EM algorithms discussed above? (Think about the shape of the clusters for instance.) Construct a synthetic dataset to illustrate this point (show that K-means fails to capture some of the clusters found by EM).

As written some lines above, the two EM algorithms are equivalent when the classes are aligned with the x- and y- axis.

K-means is efficient when the clusters are symmetric with respect to the chosen metric for the space in which the data points are embedded. For the L2 norm, the K-means is clearly outperformed when classes are not shaped like balls or cannot be captured inside non-intersecting balls. Typically, when the classes are generated by Gaussians with diagonal covariance matrices where one of the diagonal element is almost 0, the two EM algorithms perfectly work whereas K-means fails. See Fig. 10 and 11.

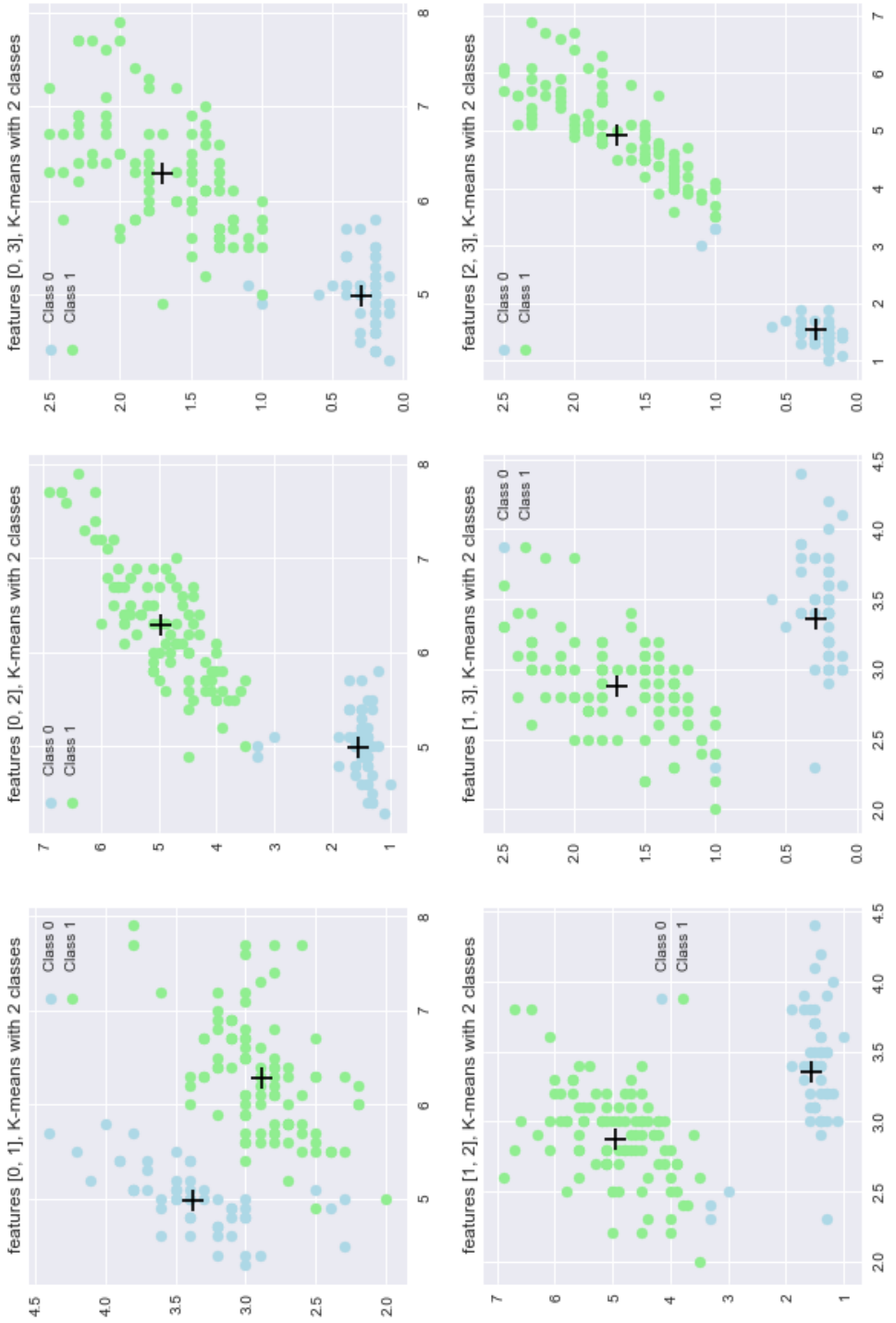

Figure 1: K-Means on iris dataset with K = 2

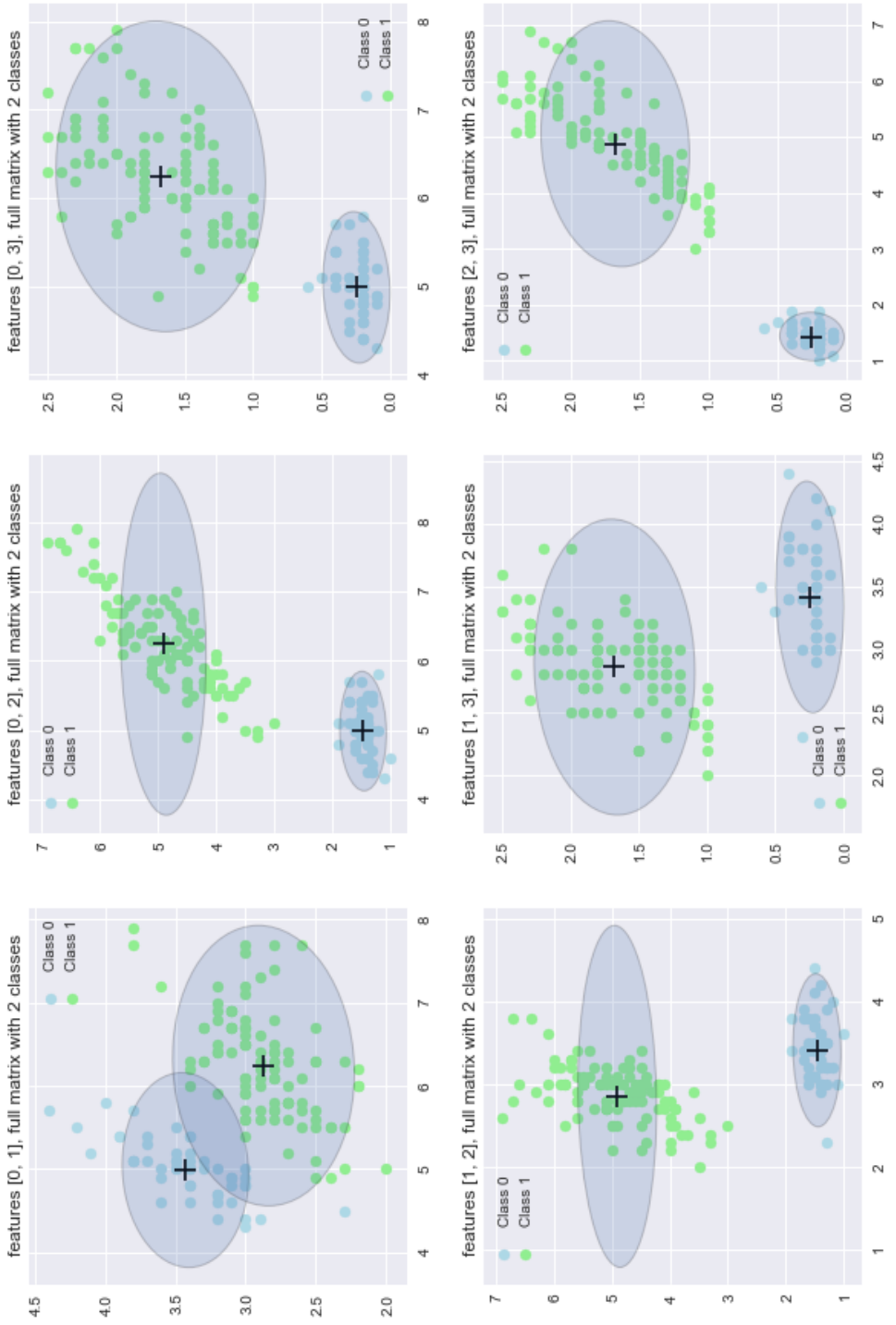

Figure 2: EM with full covariance matrix on iris dataset with K = 2

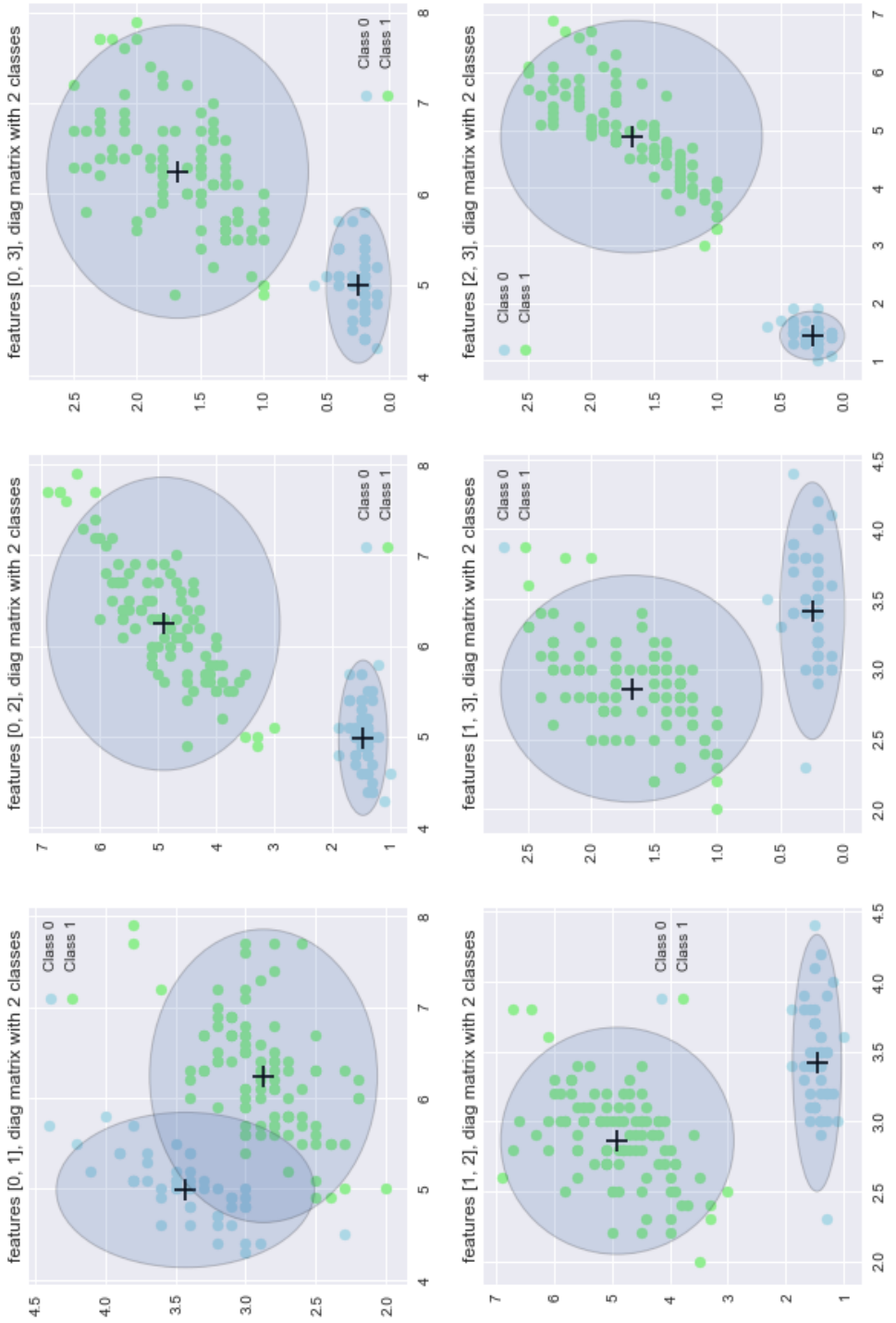

Figure 3: EM with diagonal covariance matrix on iris dataset with K = 2

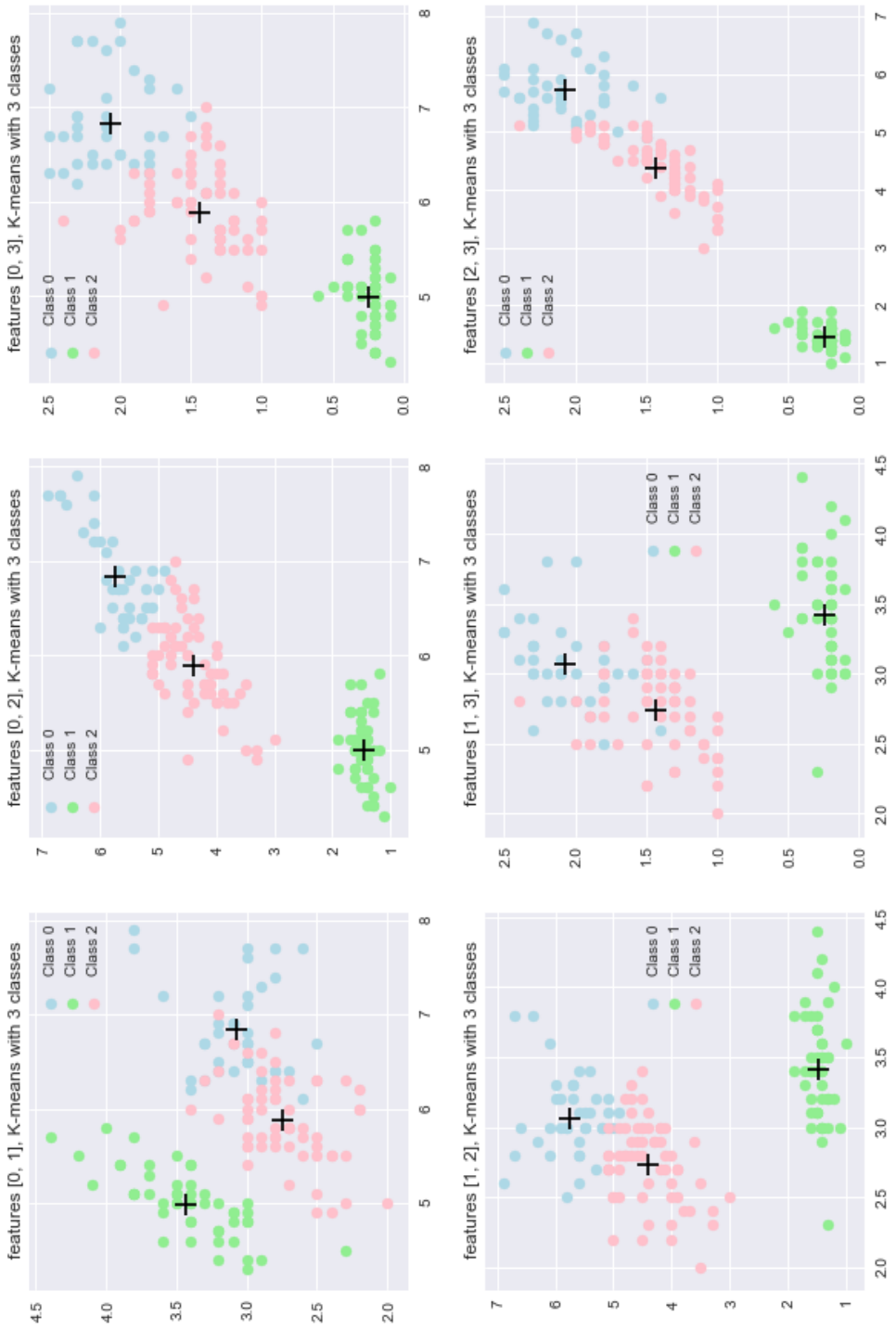

Figure 4: K-Means on iris dataset with K = 3

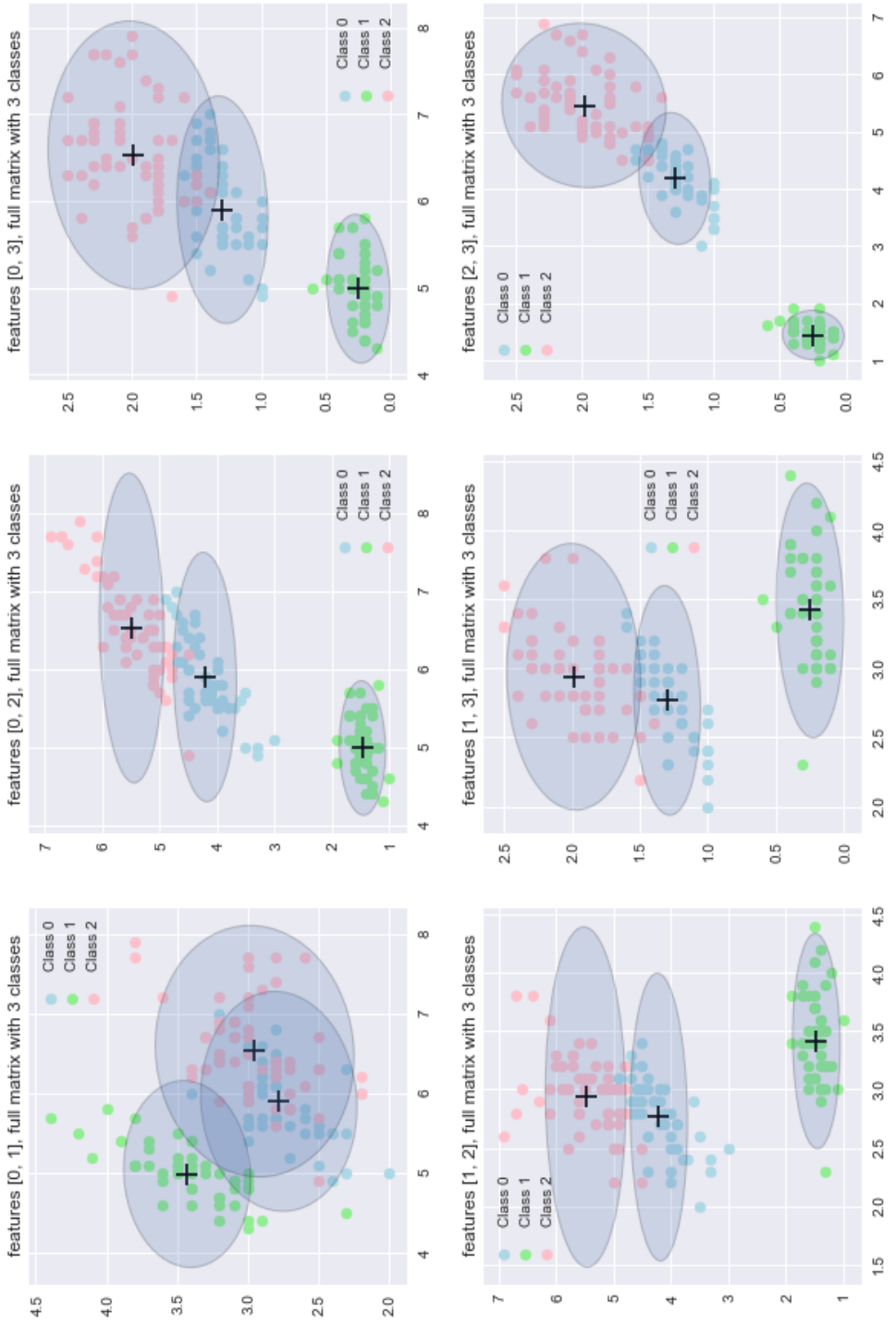

Figure 5: EM with full covariance matrix on iris dataset with K = 3

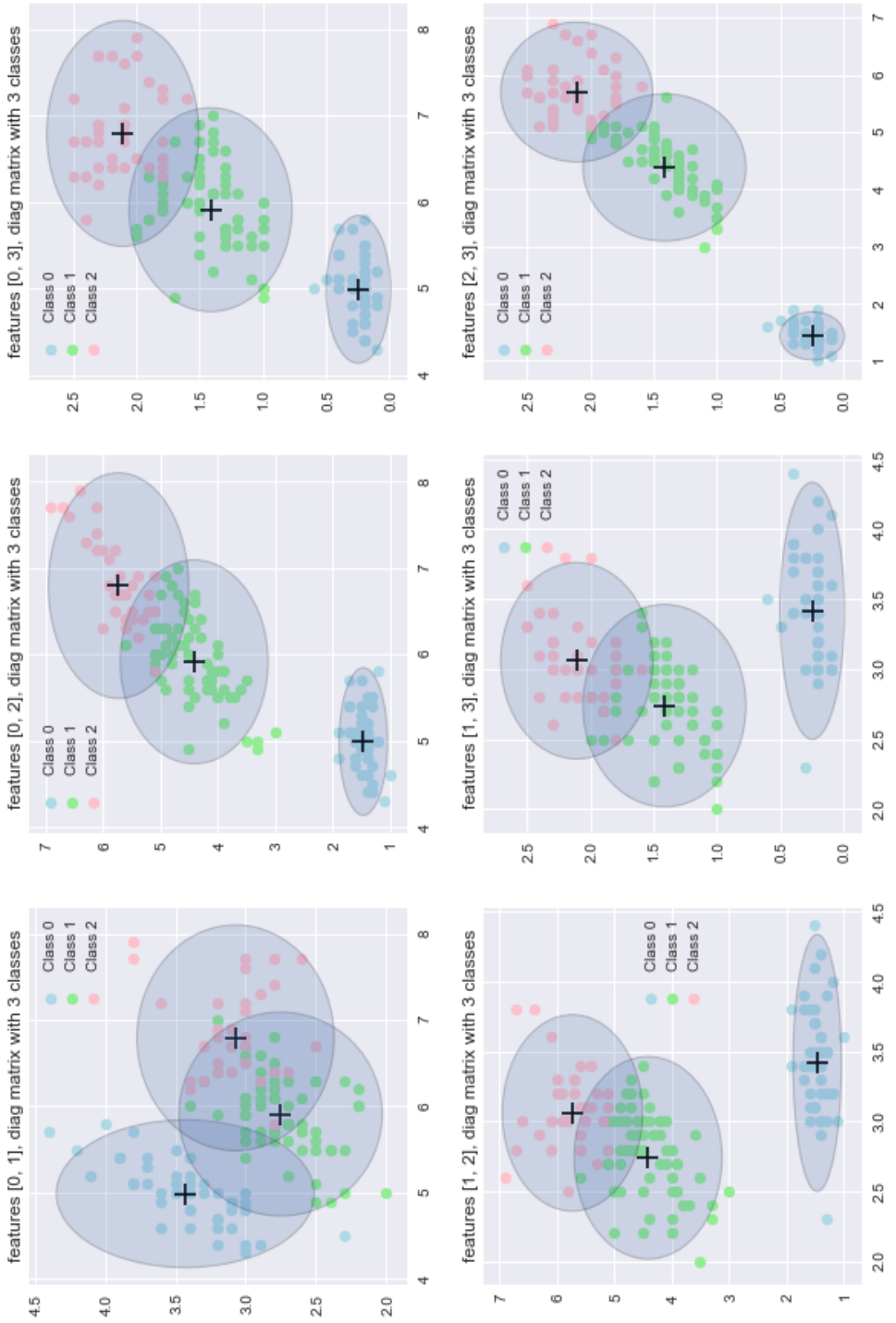

Figure 6: EM with diagonal covariance matrix on iris dataset with K = 3

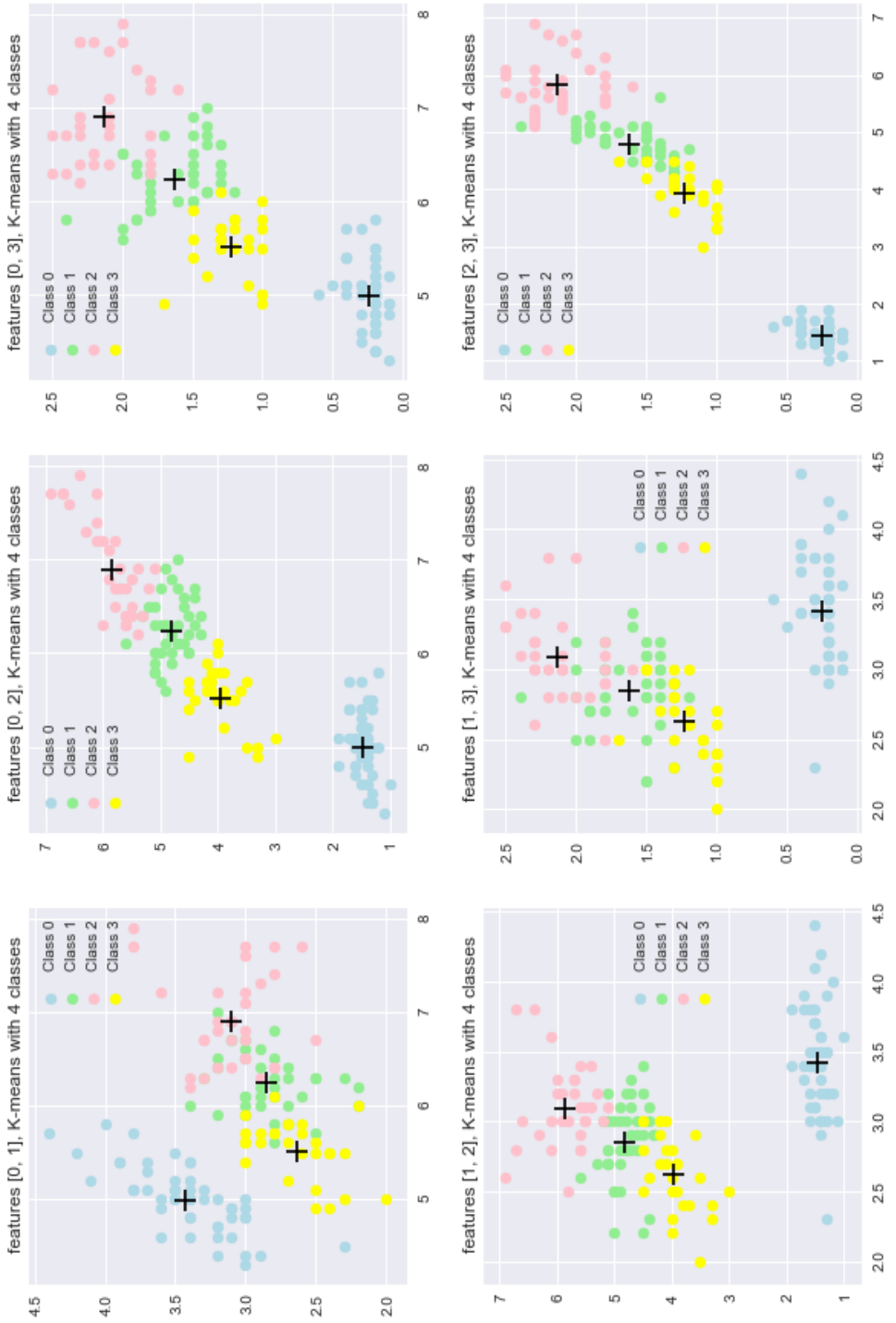

Figure 7: K-Means on iris dataset with K = 4

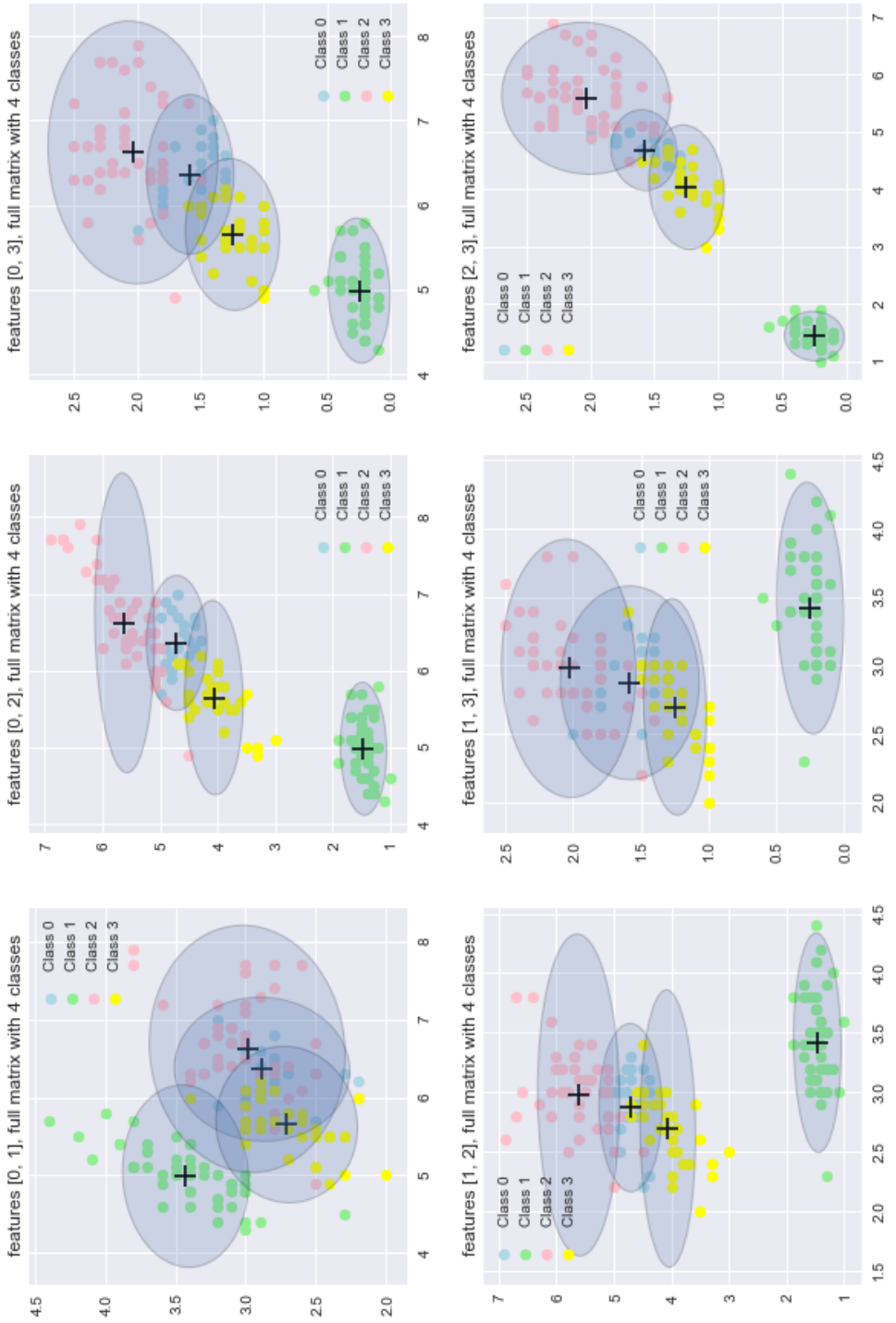

Figure 8: EM with full covariance matrix on iris dataset with K = 4

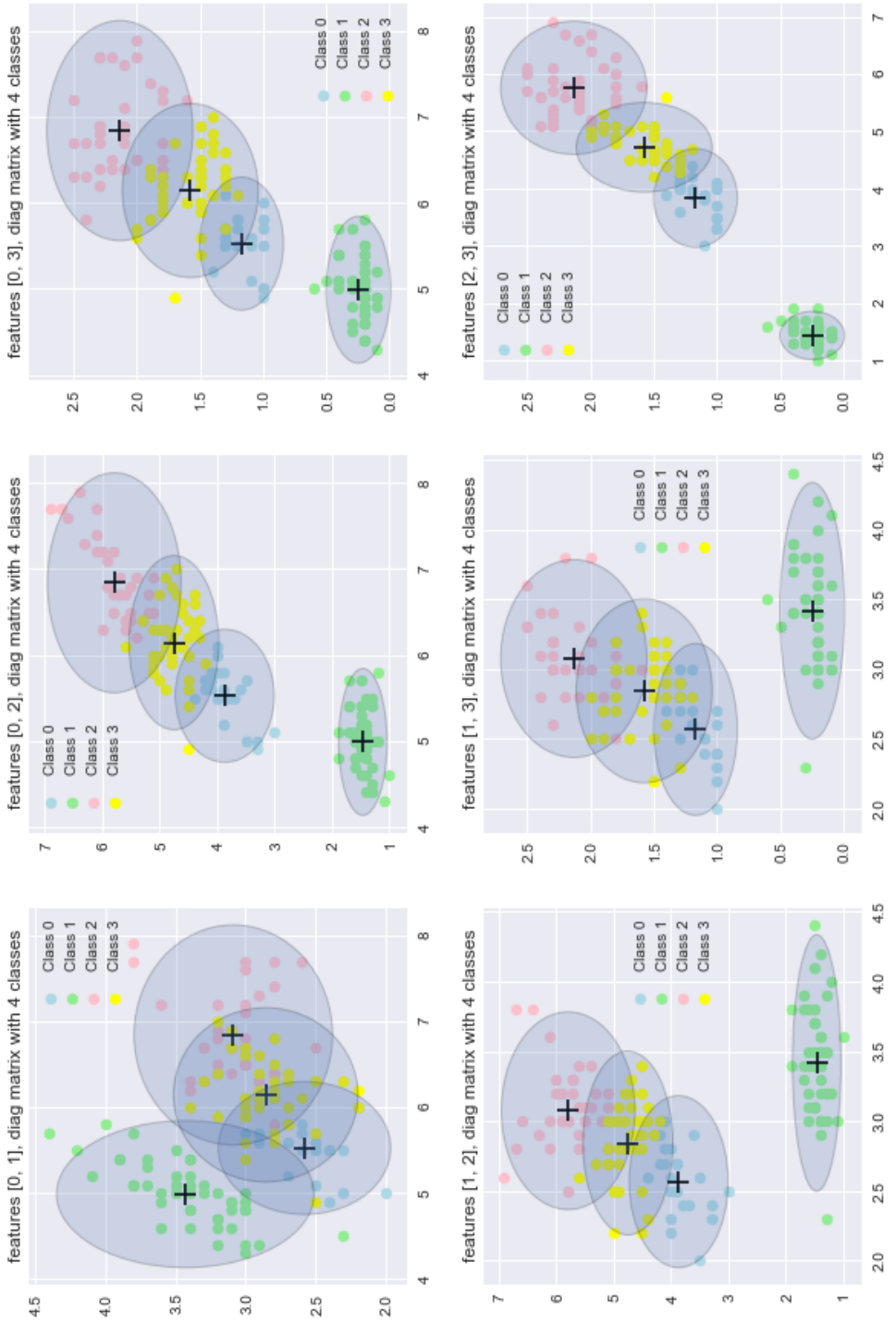

Figure 9: EM with diagonal covariance matrix on iris dataset with K = 4

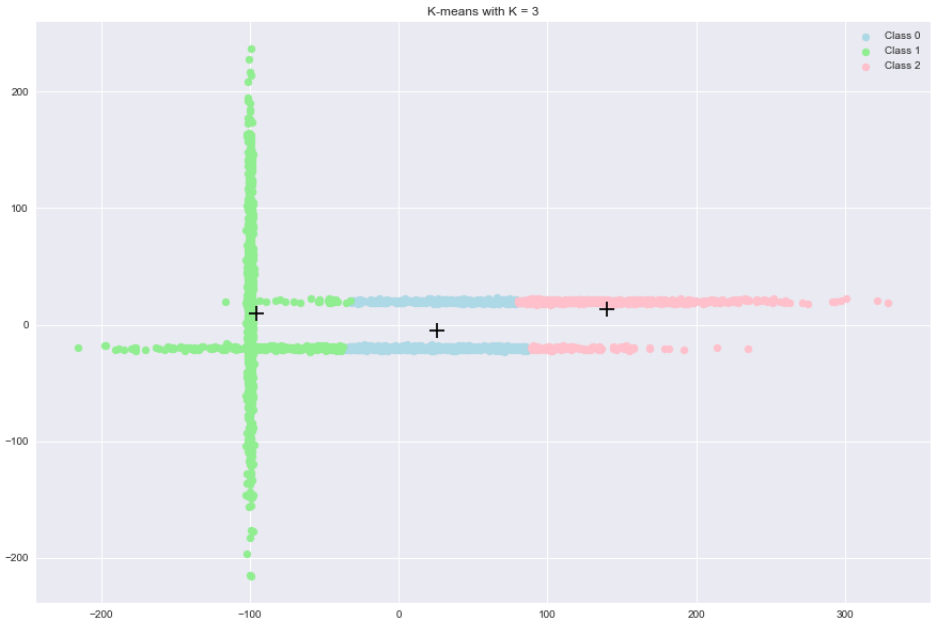

Figure 10: K-Means with K = 3

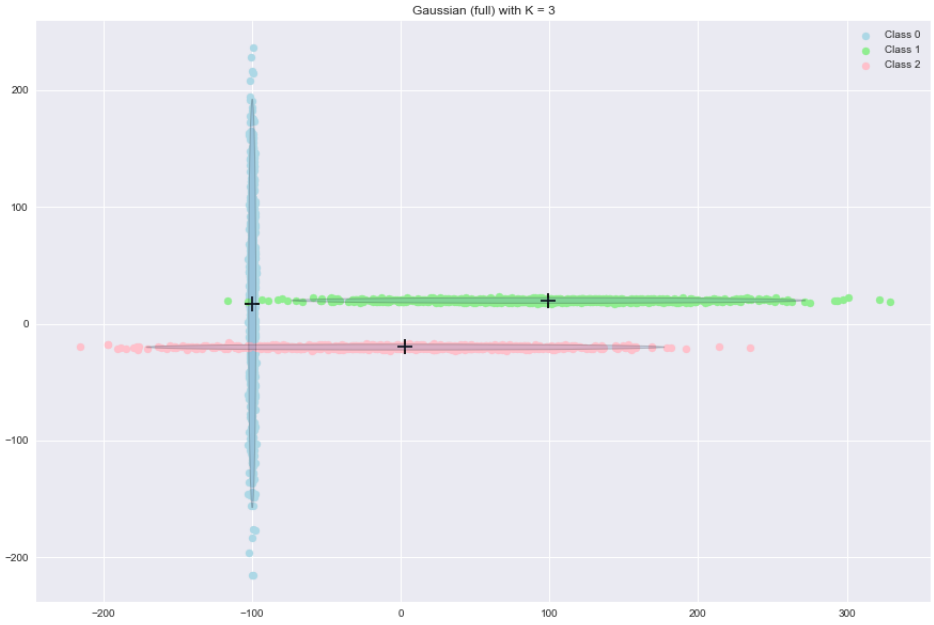

Figure 11: EM algorithm (full covariance matrix) with K = 3