COURSE 02457

Non-Linear Signal Processing: Answers to Exercise 7

Checkpoint 7.1

Choosing K and initialization.

In this checkpoint we are going to perform a K-means algorithm to estimate some clusters based on synthetic two-dimensional data with created by three "true" clusters.

The K-means algorithm is an iterative optimization algorithm consisting of two step. After an initialization of the desired number of clusters, the algorithm follows the procedure :

```
while it < maxit:

1) Compute the distance from all data points to all cluster centers.

   Assign each data point to the nearest cluster

2) Compute the clusters centers as the mean of the data points

   that was assigned to it.

end while
```

The clusters centers are initialized to the mean of the training data plus a small perturbation. The size of the perturbation is controlled by $initial_width$. There are K clusters.

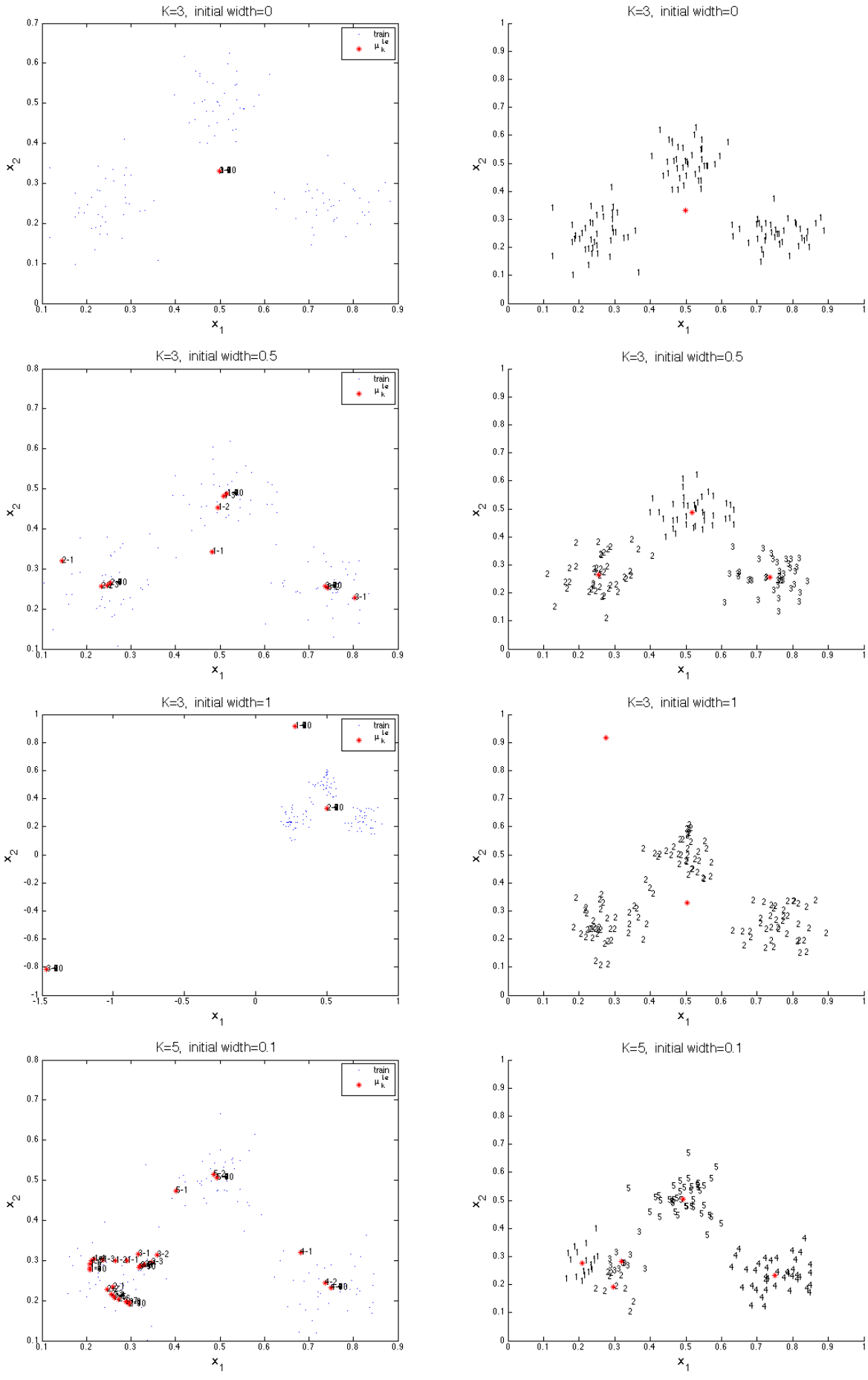

Figure 1: The K-means algorithm. **Left:** The evolution of the cluster centers. **Right:** The final labels of the data.

From Fig. 1 we see that the two parameters K and *initialwidth* plays an important part in the success of the algorithm.

If the *initialwidth* is too small (0), all the cluster centers are initialized to be the mean of the data. Consequently, the are all equally far away from the data points, and therefore the all the data points are assigned to the cluster that is first in the list/vector of clusters. Since the center of the first cluster that just got assigned all the points already is located in the mean of them, it doesn't move. The other cluster did not get any points assigned to them, and can therefore not get updated, so they don not move either.

If *initialwidth* is set to a small number less than 1 and the number of cluster is 3 (same as the data), the algorithm works perfectly.

If *initialwidth* is too large, some of the clusters may be initialized far from the data points, and the rest of the clusters that were more lucky with the initialization claim the points.

If there are too few or too many clusters, the misclassification rate gets high.

Checkpoint 7.2

The program **main7b.m** uses the Expectation-Maximization algorithm to adapt a Gaussian mixture model. Here we investigate the training and test error as a function of the number of clusters, and these can be seen in Figs. 2-3.

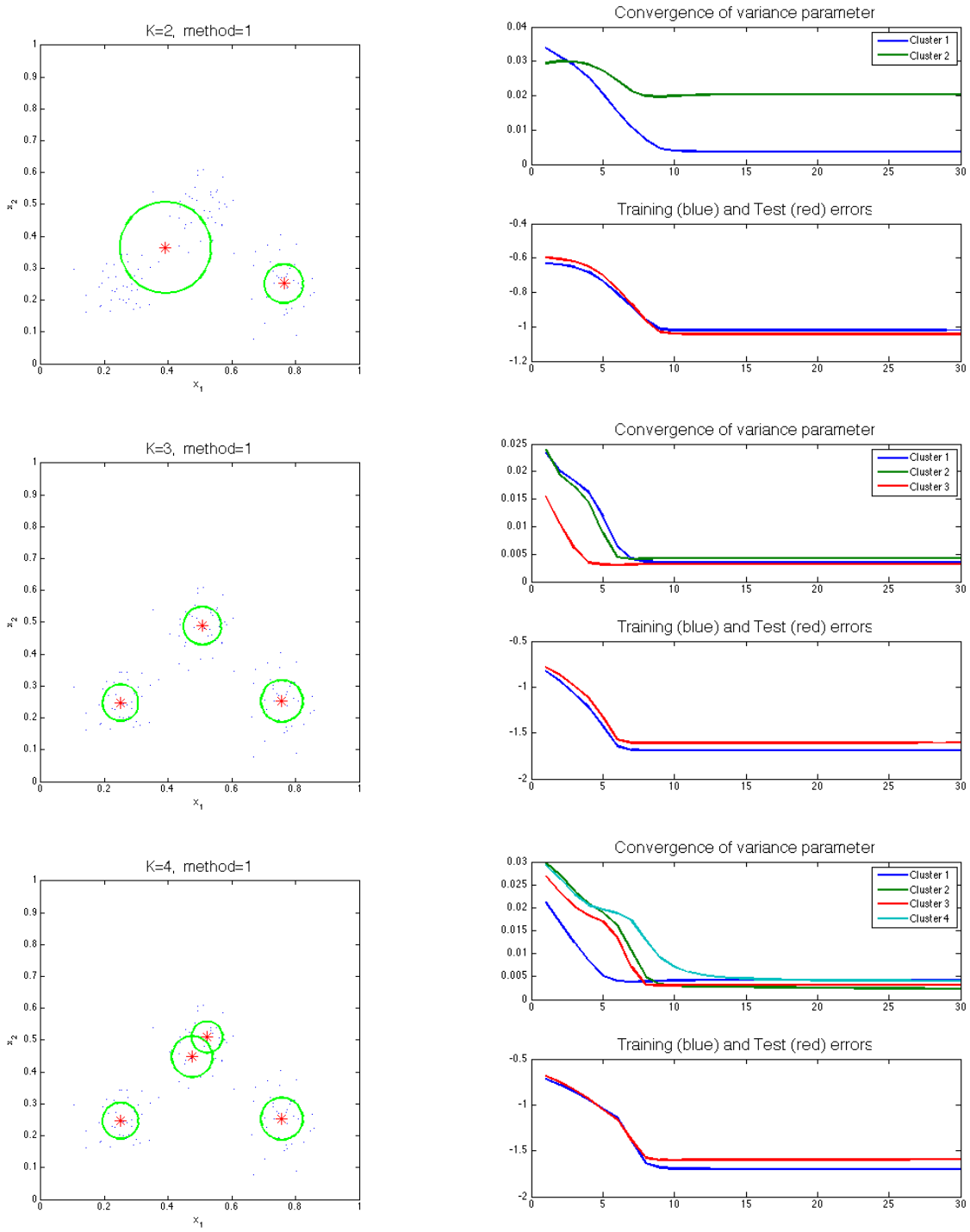

Figure 2: **Left:** The final gaussian mixture model. **Right:** The standard deviation of the clusters and the test and training errors.

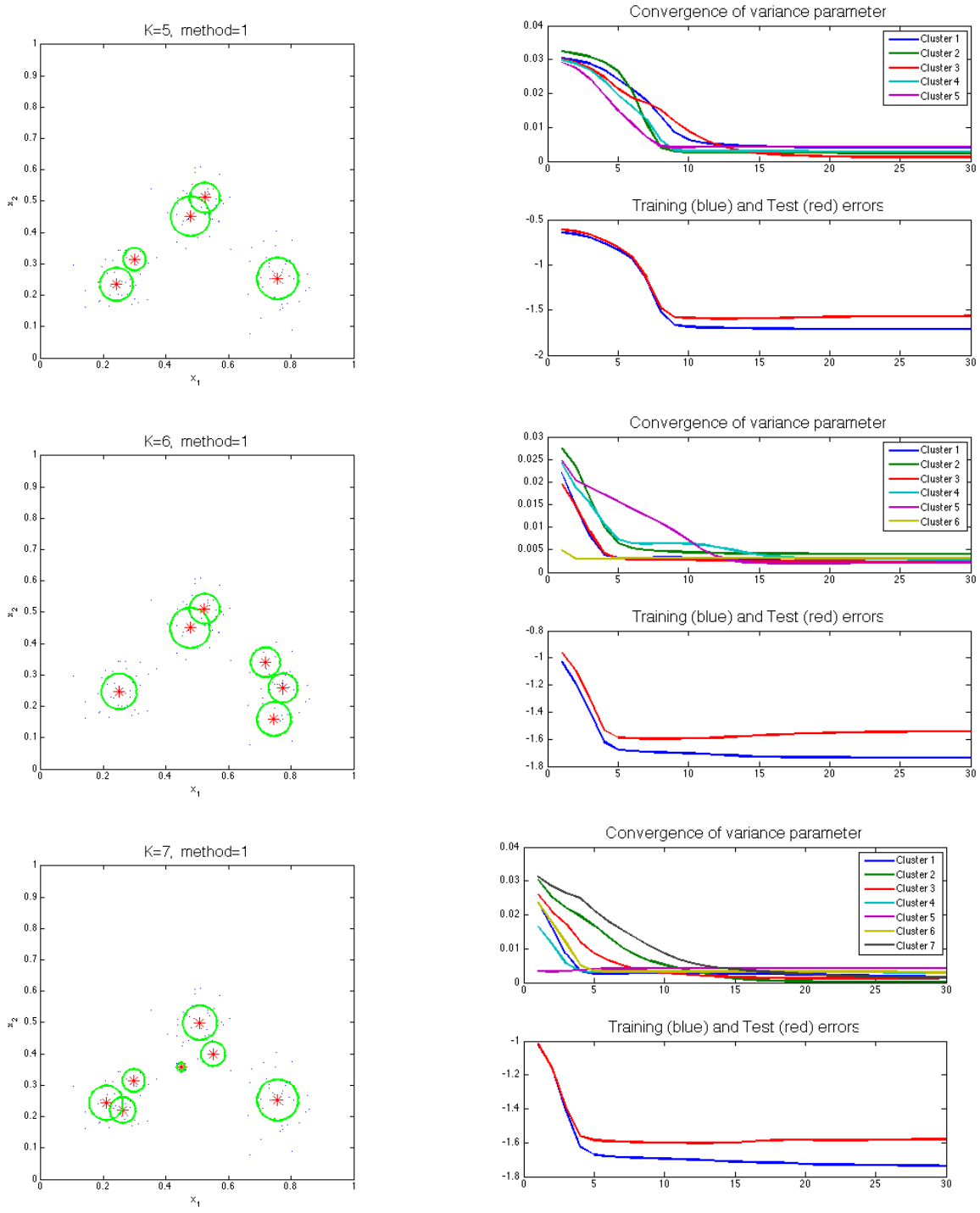

Figure 3: **Left:** The final gaussian mixture model. **Right:** The standard deviation of the clusters and the test and training errors.

We see that if we have too few clusters (2), we get a relatively high training and test error. As we increase the number of clusters, the errors decrease. However, when K gets large enough (5,6), the test error start to increase, which indicates over fitting.

Checkpoint 7.3

In **main7b.m** we can choose three different strategies for initializing the cluster centers and initial variances. Method number 1 and 3 are the same, the cluster centers are uniformly randomly chosen to be one of the data points, but the two methods have different variance. Method 3 does not choose the centers to be equal to some of the data points, but draw random coordinates (numbers) from a normal distribution having the same mean and variance as the total training data set.

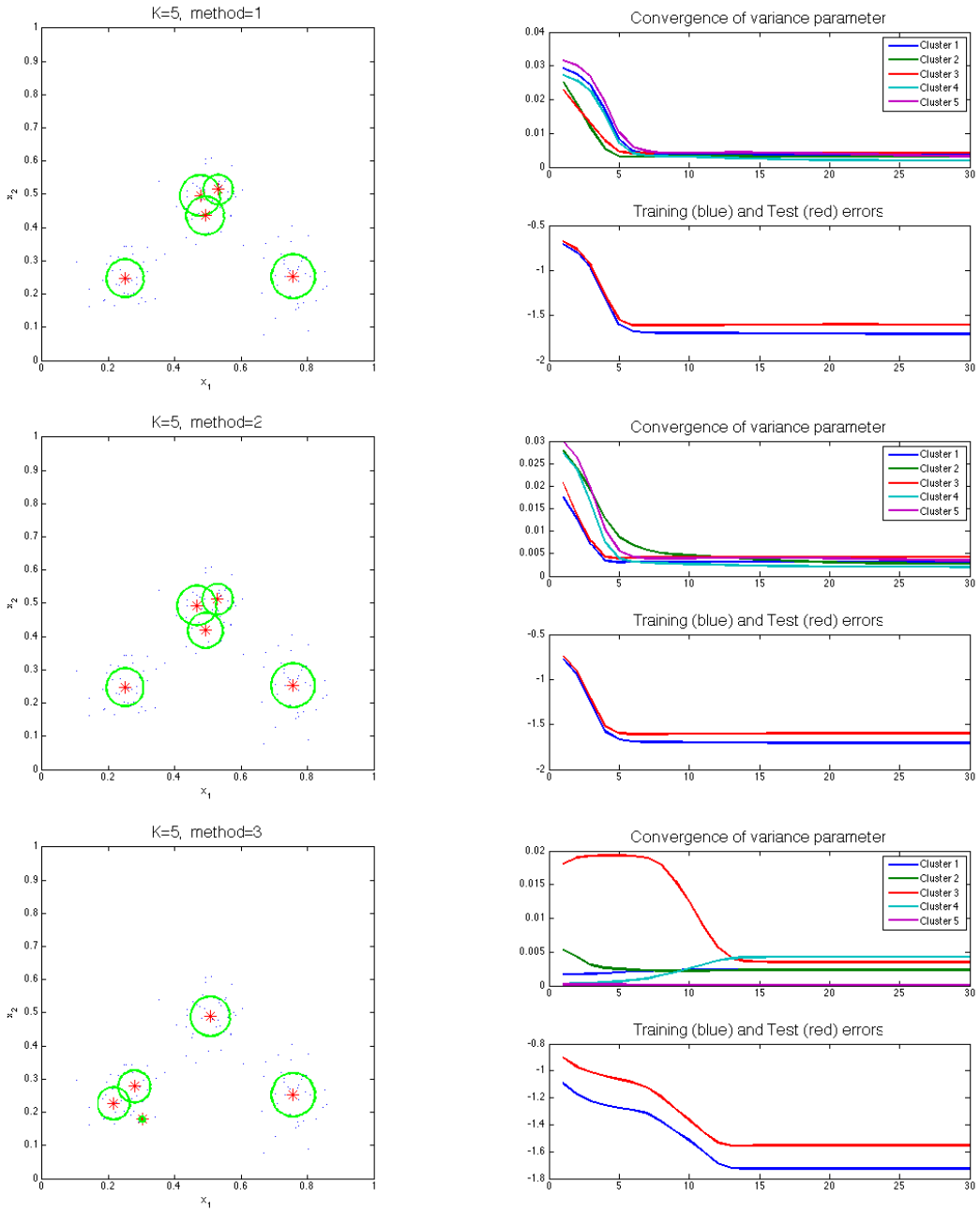

Figure 4: **Left:** Final gaussian components. **Right:** Training and test error and evolution of the variance of the gaussians.

From Fig.4 we see that for 5 clusters method 1 and 2 have similar performance, but method number 3 illustrates one of the weaknesses of the maximum likelihood approach. When the cluster centers are chosen to be on top of some of the data points and the initial variance is very small, some clusters risk not getting assigned other points than the point being it's center, so the variance shrinks to include only this point and goes to zero.

This contribution of this gaussian to the log likelihood is

$$\mathcal{N}(x_n|x_n, \sigma_j^2 I) = \frac{1}{(2\pi)^2}\frac{1}{\sigma_j} \tag{1}$$

which goes to infinity when σ_j goes to zero.

Intuitively, this is not a good solution because the small gaussian in the mixture model would not be able to model test data, but the likelihood is still very high, and that is all the maximum likelihood maximization cares about. This is also confirmed by the large test error for method 3.

To avoid this king of behavior, one could include a prior and instead model the MAP estimate to incur rage the solution to avoid very small variances.