CSC411 A2

Written by: Omri Myers, 1001902177

1.

Given the Gaussian mixture model:

$$p(x) = \sum_{k=1}^{K} \pi_k \, N(X|\mu_k, \Sigma_k)$$

We defined in lectures:

$$\gamma(z_k) \equiv p(z_k = 1|X) = \frac{p(z_k = 1)p(X|z_k = 1)}{\sum_{j=1}^{k} p(z_j = 1)p(X|z_j = 1)} = \frac{\pi_k N(X|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, N(X|\mu_j, \Sigma_j)}$$

Given X we want to maximize the likelihood of $\mu_k \; and \; \Sigma$, subject to $\Sigma_k \; = \; \Sigma \; for \; every \; k$:

$$p(X|\{\pi_k, \mu_k, \Sigma) = \prod_{n=1}^{N} \left[\sum_{k=1}^{K} \pi_k \, N\pi(X|\mu_k, \Sigma) \right]$$

$$l(X) = \ln p(X|\{\pi_k, \mu_k, \Sigma) = \sum_{n=1}^{N} ln \left[\sum_{k=1}^{K} \pi_k \, N\pi(X|\mu_k, \Sigma) \right]$$

We also know that:

$$N(X|\mu_k, \Sigma) = \frac{1}{\sqrt{2\pi^D \cdot |\Sigma|}} exp\left\{ -\frac{1}{2} (X^{(n)} - \mu_k)^T \Sigma^{-1} (X^{(n)} - \mu_k) \right\}$$

i) $\frac{\partial l(X)}{\partial \mu_k} = \frac{\partial}{\partial \mu_k} \sum_{n=1}^{N} ln\left[\sum_{k=1}^{K} \pi_k \, N\pi(X|\mu_k, \Sigma) \right] = -\frac{1}{2} \sum_{n=1}^{N} \frac{\pi_k N(X|\mu_k, \Sigma)}{\sum_{j=1}^{K} \pi_j N(X|\mu_j, \Sigma)} \Sigma^{-1} \left(\mu_k - X^{(n)} \right) = -\frac{1}{2} \Sigma^{-1} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mu_k - X^{(n)} \right)$

For $\gamma(z_{nk}) = \frac{\pi_k N(X|\mu_k, \Sigma)}{\sum_{j=1}^{K} \pi_j N(X|\mu_j, \Sigma)} = \frac{\pi_k exp\left\{ -\frac{1}{2} (X^{(n)} - \mu_k)^T \Sigma^{-1} (X^{(n)} - \mu_k) \right\}}{\sum_{j=1}^{K} \pi_j exp\left\{ -\frac{1}{2} (X^{(n)} - \mu_k)^T \Sigma^{-1} (X^{(n)} - \mu_k) \right\}}$

To find the $\mu_k \; that \; maximize \; the \; likelihood \; we \; set \; the \; derivative \; to \; 0$:

$$= -\frac{1}{2} \Sigma^{-1} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mu_k - X^{(n)} \right) = 0$$

$$than: \mu_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) X^{(n)}}{\sum_{n=1}^{N} \gamma(z_{nk})} = \frac{1}{N_k} \sum_{n=1}^{N} \frac{\pi_k N(X|\mu_k, \Sigma)}{\sum_{j=1}^{K} \pi_j N(X|\mu_j, \Sigma)} X^{(n)} =$$

$$\frac{1}{N_k} \sum_{n=1}^{N} \frac{\pi_k \frac{1}{\sqrt{2\pi^D \cdot |\Sigma|}} exp\left\{ -\frac{1}{2} (X^{(n)} - \mu_k)^T \Sigma^{-1} (X^{(n)} - \mu_k) \right\}}{\sum_{j=1}^{K} \pi_j \frac{1}{\sqrt{2\pi^D \cdot |\Sigma|}} exp\left\{ -\frac{1}{2} (X^{(n)} - \mu_k)^T \Sigma^{-1} (X^{(n)} - \mu_k) \right\}} X^{(n)}$$

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\frac{\pi_k exp\left\{-\frac{1}{2}(X^{(n)}-\mu_k)^T\Sigma^{-1}(X^{(n)}-\mu_k)\right\}}{\sum_{j=1}^{K}\pi_j\, exp\left\{-\frac{1}{2}(X^{(n)}-\mu_k)^T\Sigma^{-1}(X^{(n)}-\mu_k)\right\}}X^{(n)}$$

$$for\ N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

ii) $\frac{\partial l(X)}{\partial \Sigma} = \frac{\partial}{\partial \Sigma}\sum_{n=1}^{N} ln\left[\sum_{k=1}^{K}\pi_k\, N(X|\mu_k,\Sigma)\right] =$

$$\sum_{n=1}^{N}\frac{\sum_{k=1}^{K}\pi_k\, N(X|\mu_k,\Sigma)}{\sum_{j=1}^{K}\pi_j\, N\left(X|\mu_j,\Sigma\right)}\frac{\partial}{\partial\Sigma}\left(-\frac{1}{2}(X^{(n)}-\mu_k)^T\Sigma^{-1}(X^{(n)}-\mu_k)\right)$$

$$= -\frac{1}{2}\sum_{n=1}^{N}\frac{\sum_{k=1}^{K}\pi_k\, N(X|\mu_k,\Sigma)}{\sum_{j=1}^{K}\pi_j\, N\left(X|\mu_j,\Sigma\right)}\left((\Sigma^{-1})^T-\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\Sigma^{-1}\right)^T\right)$$

$$= -\frac{1}{2}(\Sigma^{-1})^T\sum_{n=1}^{N}\frac{\sum_{k=1}^{K}\pi_k\, N(X|\mu_k,\Sigma)}{\sum_{j=1}^{K}\pi_j\, N\left(X|\mu_j,\Sigma\right)}\left(1-\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T\right)$$

$$= -\frac{1}{2}(\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(1-\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T\right)$$

Set the derivative to zero:

$$-\frac{1}{2}(\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(1-\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T\right)=0$$

$$(\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk}) - (\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T=0$$

$$(\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk}) = (\Sigma^{-1})^T\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T$$

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\Sigma^{-1}\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)^T$$

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk}) = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\Sigma^{-1^T}\right)$$

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\,\Sigma^T = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)$$

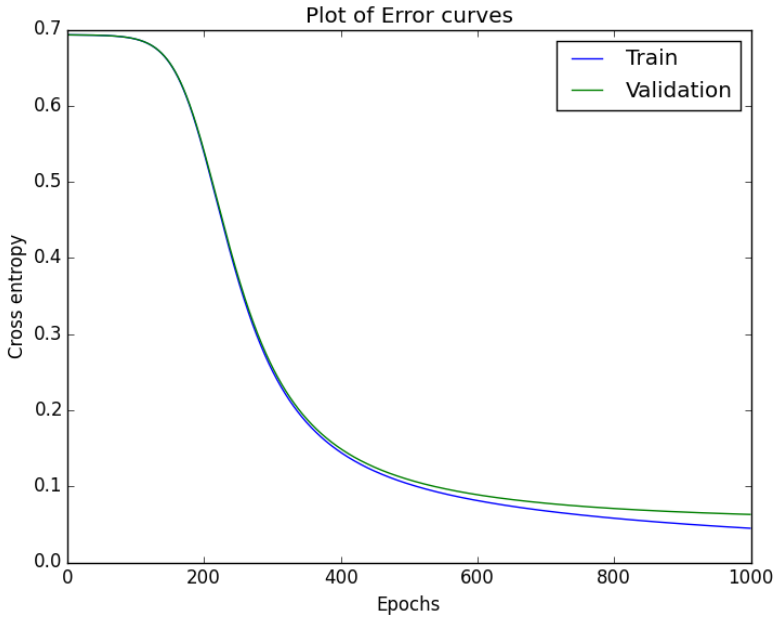
$$\Sigma^T = \Sigma = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)}{\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})}$$

$$\Sigma^T = \Sigma = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left(\left(X^{(n)}-\mu_k\right)\left(X^{(n)}-\mu_k\right)^T\right)}{\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})}$$

Since we sum over all k's the Σ is not depend on one k and equal for every k.

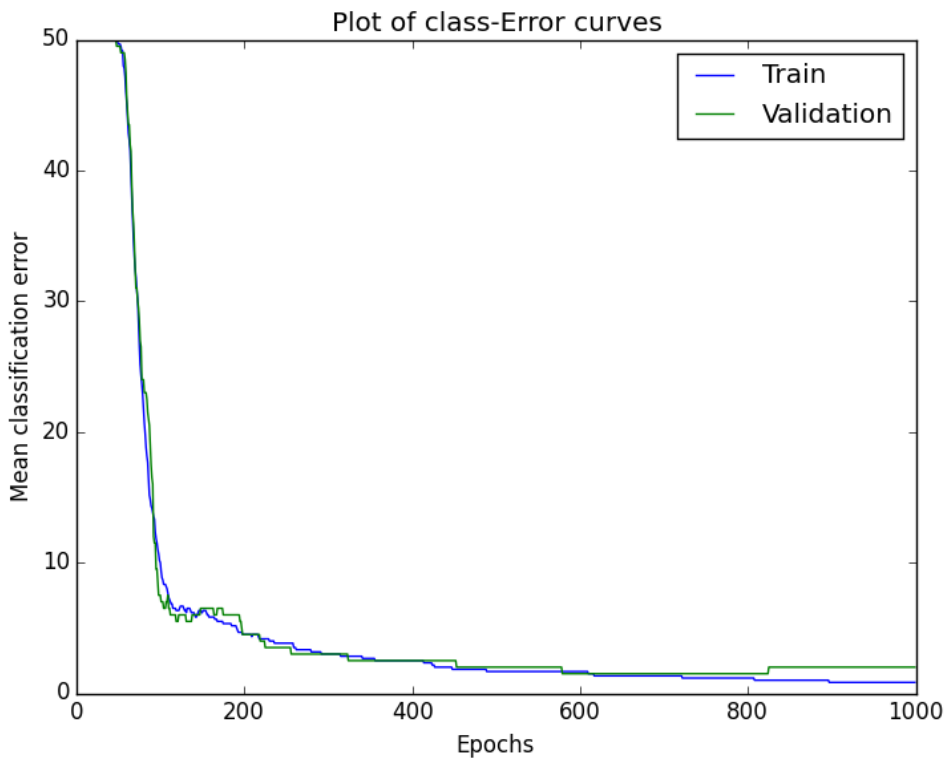
Question 2:

2.1 Basic generalization

Step 0 Train CE 0.69313 Validation CE 0.69312 Step 100 Train CE 0.68754 Validation CE 0.68761 Step 200 Train CE 0.53842 Validation CE 0.54082 Step 300 Train CE 0.25188 Validation CE 0.25673 Step 400 Train CE 0.14461 Validation CE 0.14935 Step 500 Train CE 0.10285 Validation CE 0.10861 Step 600 Train CE 0.08123 Validation CE 0.08888 Step 700 Train CE 0.06763 Validation CE 0.07773 Step 800 Train CE 0.05799 Validation CE 0.07079 Step 900 Train CE 0.05064 Validation CE 0.06618 Step 999 Train CE 0.04485 Validation CE 0.06301 Error: Train 0.04480 Validation 0.06301 Test 0.07330


From the statistics and the error curves we can learn that the network performance on the training and validation are almost identical during the first 400 epochs. After that we can see little difference and as expected the performance on the training set are slightly better. Finally we can see that the performance of the NN on the test set are reasonably good but still lower than the performance on the validation set and off course on the training set. That outcome makes sense since we update the waits according to the training set and therefore we expect the predictions on that set to be more accurate. We can infer that we the generalization can be improved by using different hyper parameters.

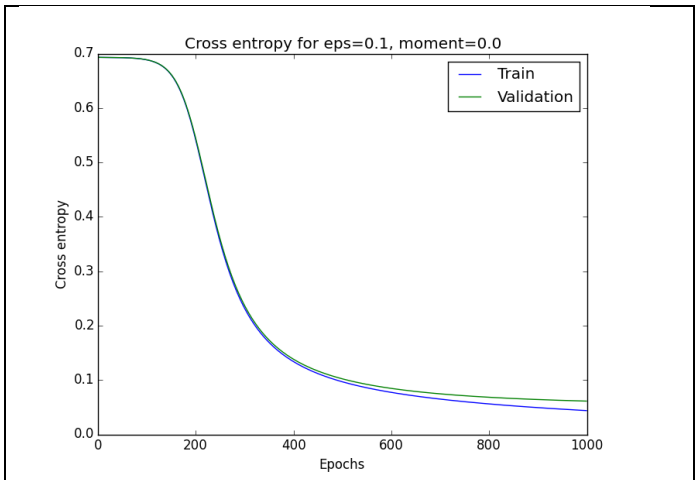
2.2. Classification error

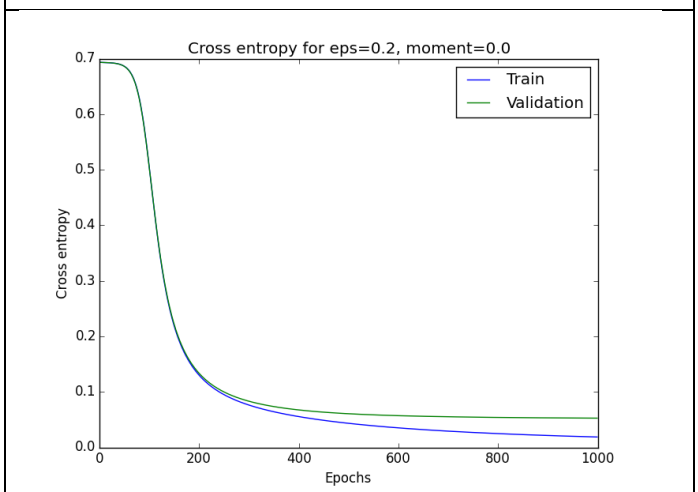


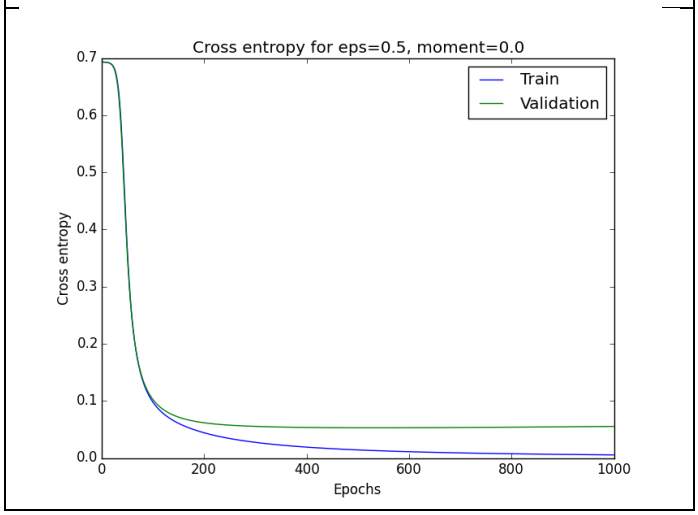
Step 0 Train MCE 50.00000 Validation MCE 50.00000
Step 100 Train MCE 10.00000 Validation MCE 7.50000
Step 200 Train MCE 4.50000 Validation MCE 4.50000
Step 300 Train MCE 3.00000 Validation MCE 3.00000
Step 400 Train MCE 2.50000 Validation MCE 2.50000
Step 500 Train MCE 1.66667 Validation MCE 2.00000
Step 600 Train MCE 1.66667 Validation MCE 1.50000
Step 700 Train MCE 1.33333 Validation MCE 1.50000
Step 800 Train MCE 1.16667 Validation MCE 1.50000
Step 900 Train MCE 0.83333 Validation MCE 2.00000
Step 999 Train MCE 0.83333 Validation MCE 2.00000
MCE: Train 0.83333 Validation 2.00000 Test 2.75000

2.3 Learning rate

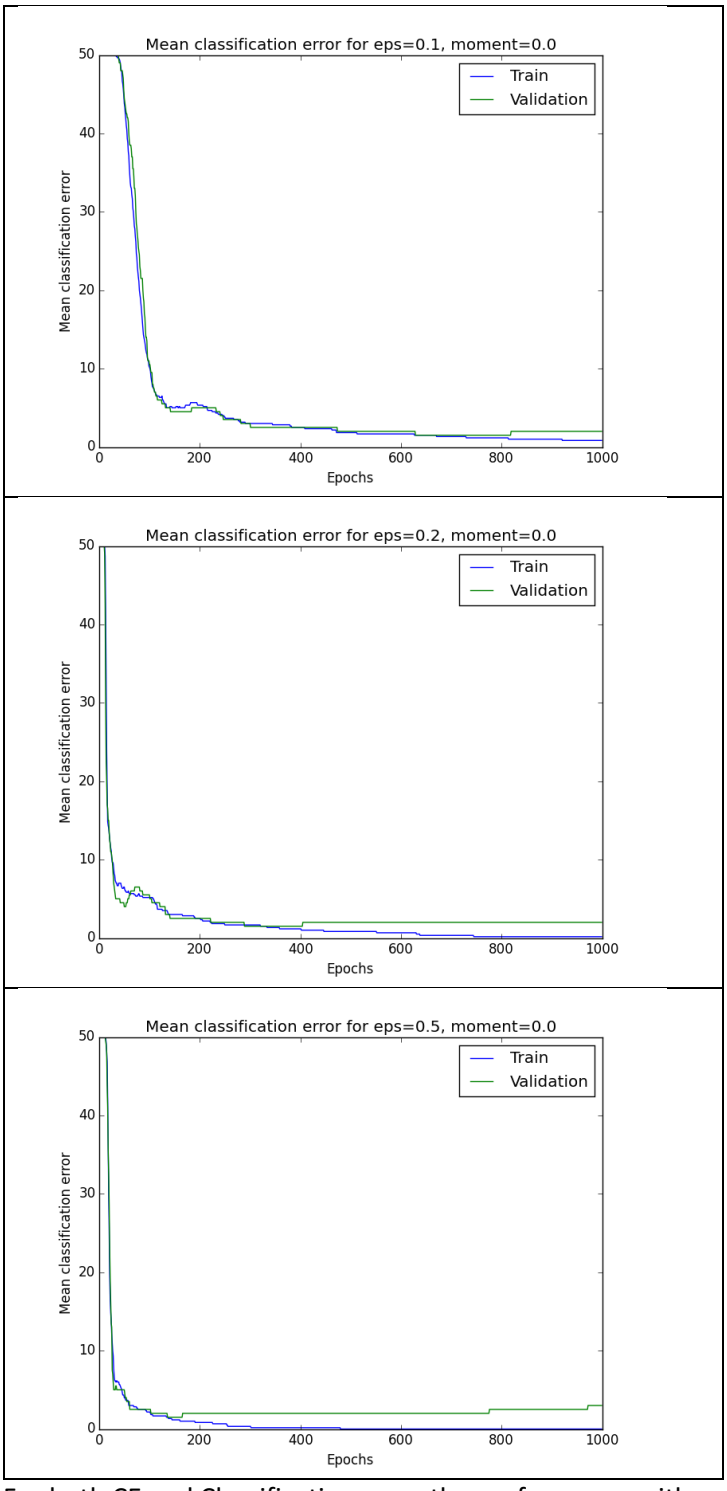
i) First case 1000 epochs, eps in {0.01,0.1,0.2,0.5} and constant momentum=0.0 I got the lowest CE for eps =0.2. But the fastest convergence rate was for eps=0.5 (after only 100 epochs the CE on both Training and validation sets was under 0.1.





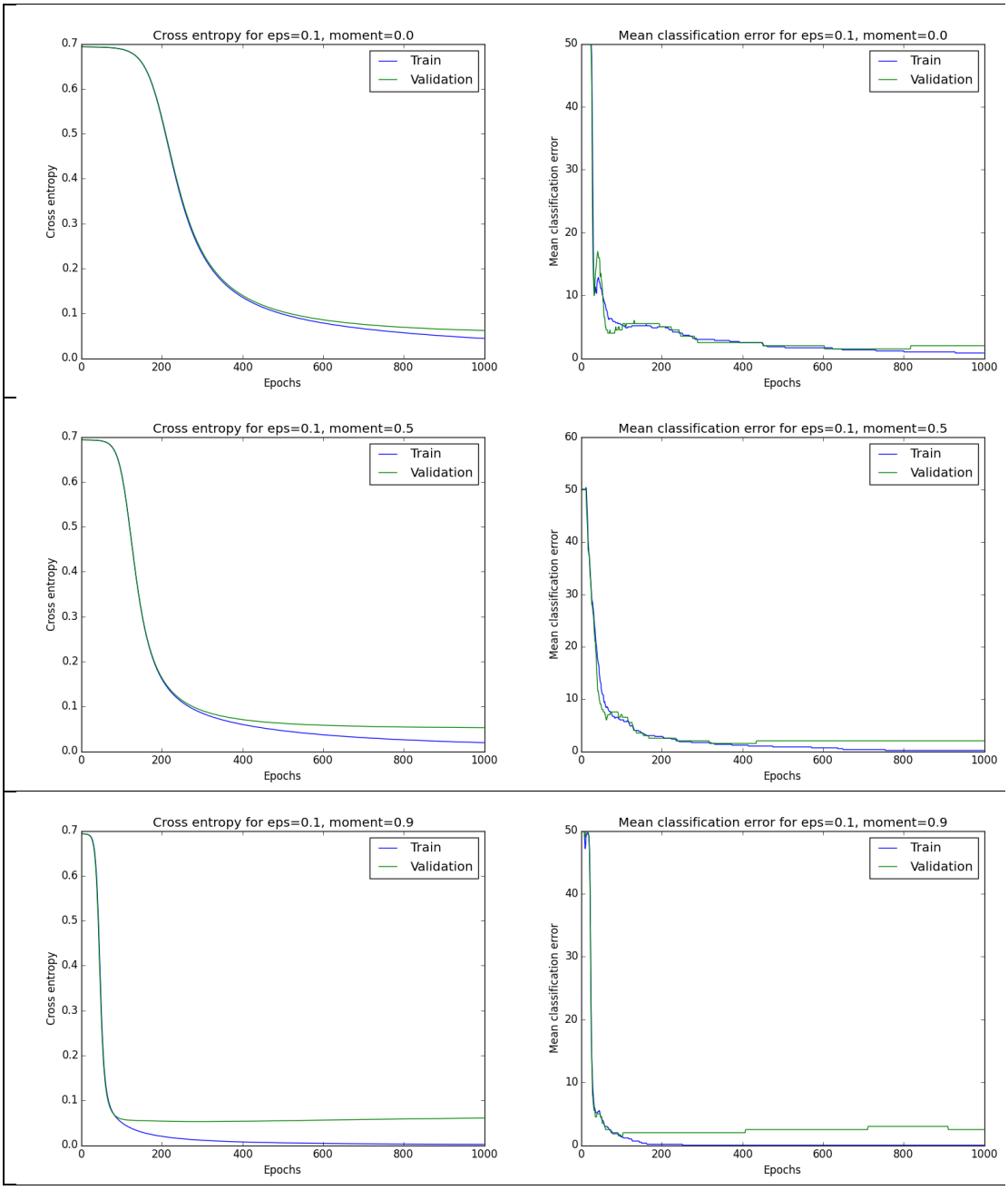


On the other hand the lowest classification error was for eps= 0.1, but as we expect the learning rate was faster for eps=0.5 and after less than 150 epochs we spot the best classification rate on the validation set. After that we can see growth in the classification error- probably due to overfitting to the training set.



For both CE and Classification error the performance with eps=0.01 was significantly lower than for the other value the algorithm did not converge in that case.

ii) Now I will refer to constant eps=0.1 and momentum in {0.0, 0.5, 0.9}:
The lowest CE achieved for momentum=0.5 and the lowest class error was for momentum=0.0
As expected the convergence rate for specific learning rate was grater for bigger momentum.



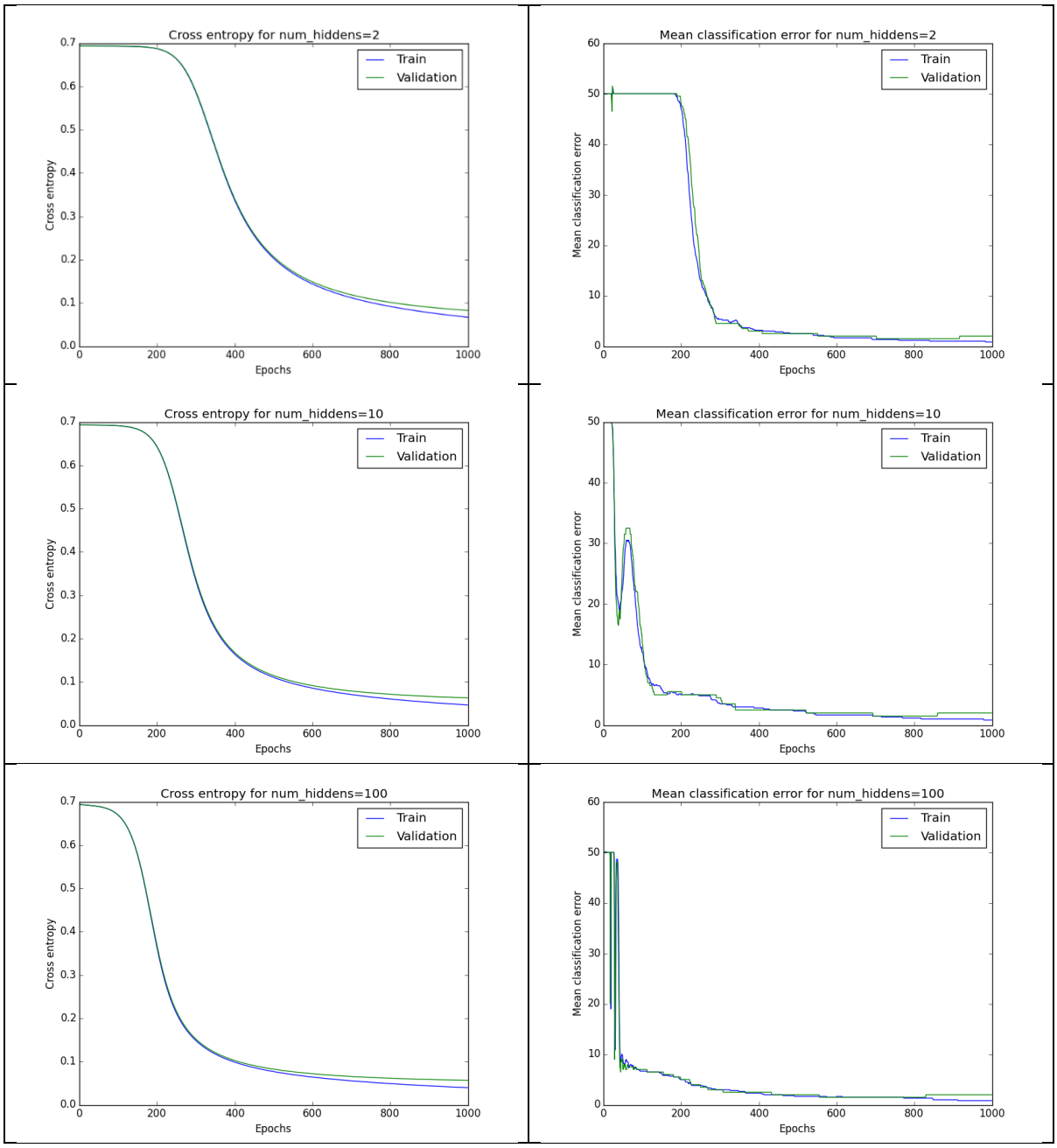
The best combination of learning rate and momentum was:

The min cross Entropy on the validation set is: 0.0421, achieved for eps 0.5, and momentum 0.9

The min Class Error on the validation set is: 1.5, achieved for eps 0.01, and momentum 0.9

2.4 Number of hidden units

After I Found the combination of eps and momentum in 2.3 I ran the algorithm on different number of hidden units. The classification error was the same in all cases, while the cross entropy was slightly lower for higher number of hidden units. When measuring the convergence rate of the algorithm for cross entropy we find out that it is higher for 100 hidden units. Moreover the generalization was better since we see lowest CE value for this setting. For classification error it is almost the same case.



2.5 Compare k-NN and Neural Networks

I compare the classification error for both models (no convergence for knn and no cross entropy). for knn algorithm I chose k=7 as I found the best in assignment 1. I compared it with combination of hyperparmeters that yield the best results for Neural-Networks

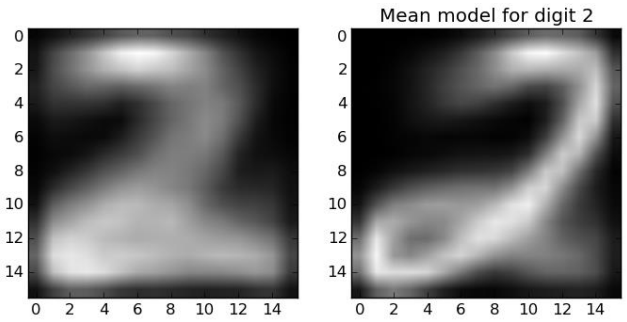
In both cases the best- lowest classification error was 1.5% again we can see the advantages of the simple model of k-nn for classification in this case. I would guess that if we had more data (training set) the Neural Network would outperform the knn. But since our data is limited knn the performance are the same but knn is much faster.

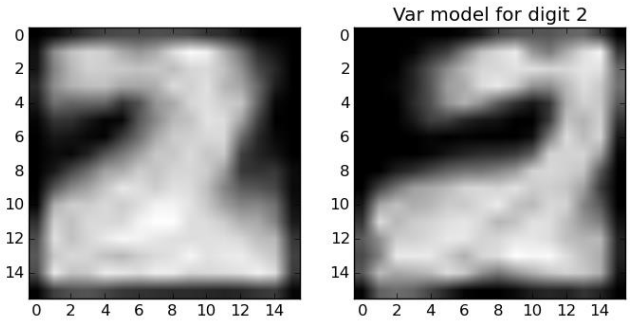
3.1 Done

3.2

After testing few combination of the parameters I found out that there is no big difference for different values of rand. I finally show both models for iters=10, minVary=0.01, k=2, crand=1.

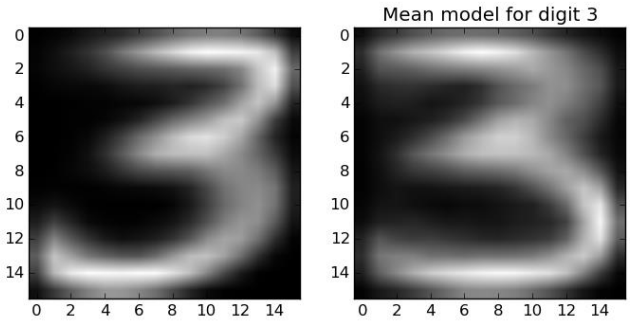
Model of 2:

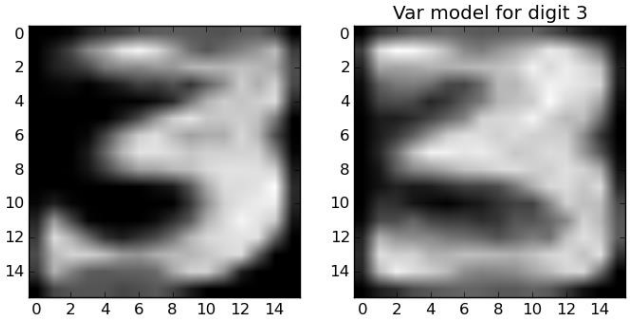




the mixing proportion for 2 model is:
[[0.46666666]
[0.53333334]]
and the logProb is:
[-3890.79258653]

Model for 3:





the mixing proportion for 3 model is:
[[0.52986667]
[0.47013333]]
and the logProb is:
[2294.22172789]

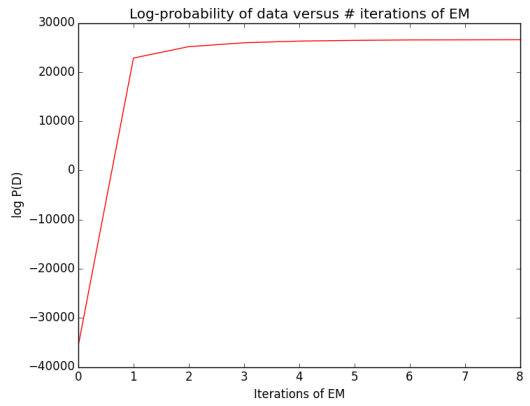
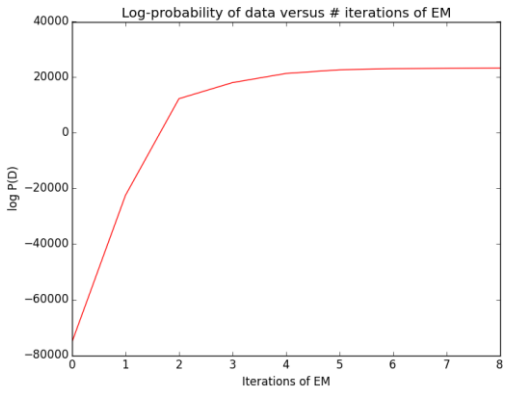
3.3

For normal (random) initialization we get the following results:

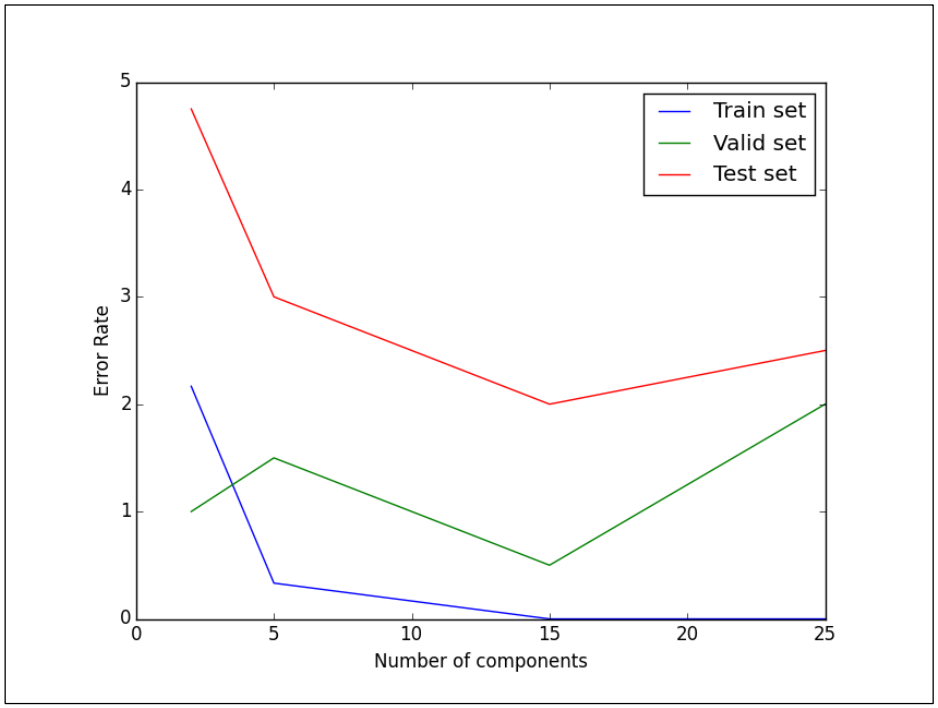
iteration	log probability random initialization	log probability k-means initialization
1	-75836.27593	-35631.63625
2	-26885.93079	23916.31756
3	8504.54988	26836.00872
4	19086.34150	27850.06363
5	22654.87912	28179.64580
6	23865.47933	28288.84619
7	24443.57532	28331.05099
8	24900.40420	28384.27117
8	25077.89251	28425.07544
10	25152.34816	28425.08531

We can see that the convergence rate with random initialization is significantly lower than the rate with k-means initialization. We can see that the algorithm that was initialized with k-means converged in the last iteration while the algorithm that was initialized randomly did not converge at all. Moreover even when I tried to run it over 20 iterations it did not converge.

We can see that the final log-prob on the algorithm that was initialized with k means is much higher than the log-prob of the randomly initialized algorithm. In that case we can say that each data point fits better to its cluster.

K-means initialization	Random initialization
	

3.4 Classification using MoGs:



i) The classification error is decreasing since the classifier become more specific for the training set. For each component we have less data points and that way we can achieve more accuracy over the training set.

ii) The test set is generally decreasing, but for more than 15 components is rise. We would expect that trend for big number of components we would have over fitting to the training set. The clusters that the algorithm find are too specific in that case and therefore new data that is not in found in the training set might be misclassified.

iii) I would choose the model that minimize the error on the test set. In our case that is 15 because the test error goes down to 2 and after that rise. We want to be able to classify wide variety of inputs but more complicated model can cause over fitting.