Student Name: Irene Chang

Collaboration Statement:

Total hours spent: 15 hours

I consulted the following resources:

- TA Kapil, Ike
- Piazza
- Kelsey

Contents

HW-1a: Problem Statement

Prove that the mean of vector x under the mixture distribution is given by:

$$\mathbb{E}_{p^{\text{mix}}(x)}[x] = \sum_{k=1}^{K} \pi_k \mu_k \tag{1}$$

HW-1a: Solution

$\mathbb{E}_{(f_k(x|\mu_k,\Sigma_k)}[x] = \mu_k \Rightarrow \sum_{k=1}^{K} x f_k(x|\mu_k, \Sigma_k) = \mu_k$

Thus we have:

$$\begin{aligned}
\mathbb{E}_{p^{\text{mix}}(x)}[x] &= \sum_{k=1}^{K} x p^{\text{mix}}(x) \\
&= \sum_{k=1}^{K} x \sum_{k=1}^{K} \pi_k f_k(x|\mu_k, \Sigma_k) \\
&= \sum_{k=1}^{K} \pi_k \left(\sum_{k=1}^{K} x f_k(x|\mu_k, \Sigma_k) \right) \\
&= \sum_{k=1}^{K} \pi_k \mu_k
\end{aligned}$$

HW-1b: Problem Statement

Given: $m = \mathbb{E}_{p^{\text{mix}}(x)}[x]$ (as in 1a). Prove that the covariance of vector x is:

$$\text{Cov}_{p^{\text{mix}}(x)}[x] = \sum_{k=1}^{K} \pi_k(\Sigma_k + \mu_k \mu_k^T) - mm^T \tag{2}$$

HW-1b: Solution

We know the formula for covariance matrix: $\Sigma_X = \mathbb{E}[XX^T] - \mathbb{E}[X]\mathbb{E}[X]^T$
$\Rightarrow \mathbb{E}[XX^T] = \Sigma_X + \mathbb{E}[X]\mathbb{E}[X]^T$

$$\begin{aligned}
\text{Cov}_{p^{\text{mix}}(x)}[x] &= \mathbb{E}_{p^{\text{mix}}(x)}[(x - \mathbb{E}_{p^{\text{mix}}(x)}[x])(x - \mathbb{E}_{p^{\text{mix}}(x)}[x])^T] \\
&= \mathbb{E}_{p^{\text{mix}}(x)}[xx^T] - \left(\mathbb{E}_{p^{\text{mix}}(x)}[x]\right)\left(\mathbb{E}_{p^{\text{mix}}(x)}[x]\right)^T \\
&= \mathbb{E}_{p^{\text{mix}}(x)}[xx^T] - mm^T \\
&= \sum_{k=1}^{K} \pi_k \mathbb{E}_{f_k}[xx^T] - mm^T \\
&= \sum_{k=1}^{K} \pi_k(\Sigma_k + \mathbb{E}_{f_k}[x]\mathbb{E}_{f_k}[x]^T) - mm^T \\
&= \sum_{k=1}^{K} \pi_k(\Sigma_k + \mu_k \mu_k^T) - mm^T \ (\mathbb{E}_{f_k}[x] = \mu_k)
\end{aligned}$$

HW-2a: Problem Statement

Show (with math) that using the parameter settings defined above, the general formula for γ_{nk} will simplify to the following (inspired by PRML Eq. 9.42):

$$\gamma_{nk} = \frac{e^{-\frac{1}{2}\frac{1}{\epsilon}||x_n - \mu_k||^2}}{\sum_{j=1}^{K} e^{-\frac{1}{2}\frac{1}{\epsilon}||x_n - \mu_j||^2}} \tag{3}$$

HW-2a: Solution

We have:

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon}||x - \mu_k||^2\right\}$$

$$\begin{aligned}
\gamma_{nk} &= \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \\
&= \frac{\pi_k \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_k||^2\right\}}{\sum_{j=1}^{K} \pi_j \frac{1}{(2\pi\epsilon)^{1/2}} \exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_i||^2\right\}} \\
&= \frac{\pi_k \exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_k||^2\right\}}{\sum_{j=1}^{K} \pi_j \exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_i||^2\right\}}
\end{aligned}$$

$\pi_{1:K}$ follows the uniform distribution, π_i is the same across all $i's$, so:

$$\gamma_{nk} = \frac{\exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_k||^2\right\}}{\sum_{j=1}^{K} \exp\left\{-\frac{1}{2\epsilon}||x_n - \mu_i||^2\right\}}$$

HW-2b: Problem Statement

HW-2b: Solution

Using $\epsilon = 1.0000$, we obtain γ as:

```
[[1.000 0.000 0.000]
 [0.999 0.001 0.001]
 [0.909 0.084 0.006]
 [0.991 0.007 0.001]
 [0.081 0.870 0.049]
 [0.000 0.651 0.349]
 [0.000 0.349 0.651]]
```

HW-2c: Problem Statement

HW-2c: Solution

Using $\epsilon = 0.1$, we see the γ values get more extreme:

```
[[1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [1.000 0.000 0.000]
 [0.000 1.000 0.000]
 [0.000 0.998 0.002]
 [0.000 0.002 0.998]]
```

HW-2d: Problem Statement

You can regard the outputs of the k-Means algorithm as a one-hot vector of dimension K. For example, if a sample x_n is assigned to a cluster k through k-Means, the corresponding one-hot vector will have value 1 at its k^{th} element. We can view this assignment by k-Means, for each example x_n, as a valid probability distribution, as the entries of a one-hot vector sums to 1.

What will happen to the value of γ as $\epsilon \to 0$? How is this related to the K-means one-hot assignment?

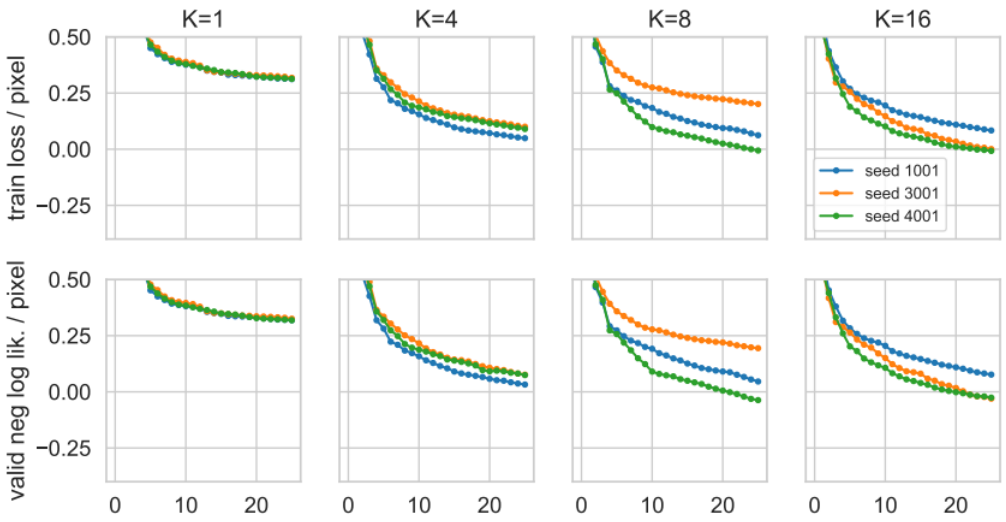
HW-2d: Solution

As $\epsilon \to 0$, the denominators go to 0 and the term for which $||x_n - \mu_k||^2$ is smallest go to 0 most slowly, and so γ_{nk} go to 0 for all terms k except for this j-th term. Thus, we will obtain the assignment a hard assignment of data points to clusters, just like with K-means, since $\gamma_{nk} \to 1$ if $||x_n - \mu_k||^2$ is smallest (the data belongs to the cluster) and $\gamma_{nk} \to 0$ otherwise.

CP-3a: Problem Statement

LBFGS Trace plots of Training Loss vs Iteration

Solution:

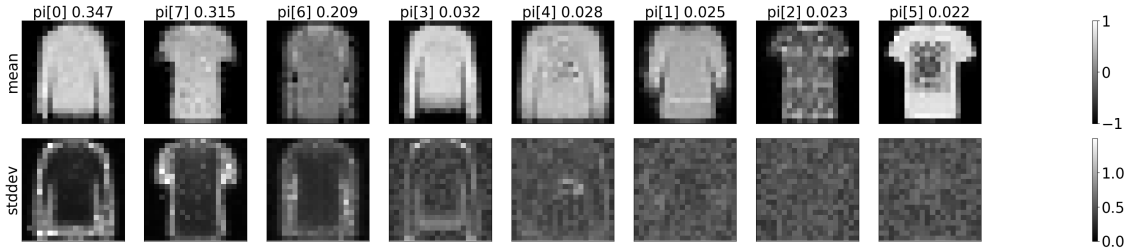


The training loss and validation set's negative log likelihood go down overall as we have more iterations. The loss/negative log likelihood go down a lot when we have 4 clusters instead of 1, but if we add even more clusters, the results seem to fluctuate or stay around the same results, which indicates that $k = 4$ is likely the best number of clusters here. The performance on the validation set reflects the pattern of the performance on the training set. However, the training with different seeds don't converge to the same results as we have multiple clusters within 25 iterations. Since the loss optimization problem is not convex, this shows that the LBFGS algorithm fails to agree on the best result under non-convex context with different starting points.

CP-3b: Problem Statement

LBFGS best model with K=08

Solution:



The means and standard deviations are organized in the order of decreasing probability of being in the given cluster. Observing the mean, we see that many of the clusters are not visibly different from each other. As the probability decreases, the standard deviation of that particular cluster becomes higher, so it makes sense that the image gets more noisy and towards the end of the row, there isn't any distinguishable item pictured in the frame. This also agrees with the results we observed from the plot above: since the algorithm doesn't seem to converge to the optimal solution yet, we don't get the definite shape of the items in half of the clusters (this in turn means that a lot of items won't distinctly be deemed to belong to one cluster, so the classification will also not be precise). Furthermore, the results not converging yet (the parameters are still being learned) is also shown in the fact that the items shown in the mean are still depicted with a lot of details, the shapes have not been generalized to represent the bigger cluster (unlike in GM as we will see below).

CP-3c: Problem Statement

LBFGS Scores vs Number of Clusters K

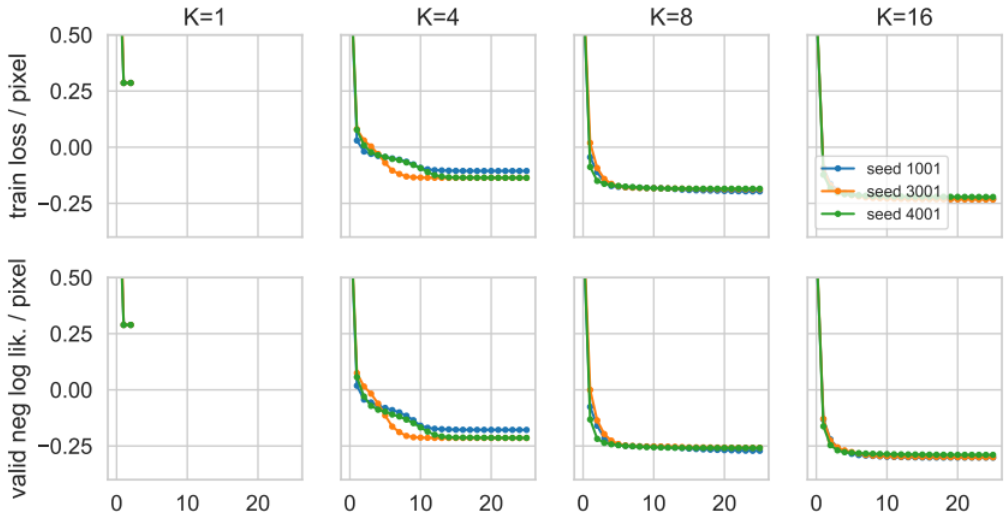
Solution:

k	Validation negative log likelihood per pixel
1	0.318792699418883
4	0.0324184803247638
8	-0.0369397162188361
16	-0.0303198879761775

CP-4a: Problem Statement

EM Trace plots of Training Loss vs Iteration

Solution:

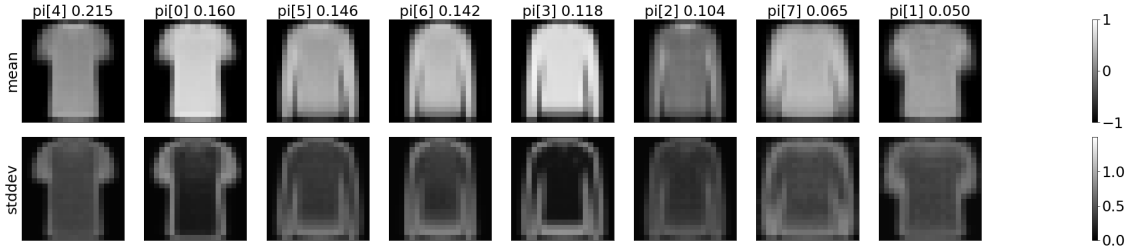


The result shows that the training loss and validation set's negative log likelihood go down overall as we have more iterations. The loss/negative log likelihood go down a lot when we have 4 clusters instead of 1, but if we add even more clusters, the results seem to stabilize (not go down as much), also indicating that $k = 4$ is likely the best number of clusters here. The performance on the validation set reflects the pattern of the performance on the training set. The model appears to converge much quicker compared to LBFGS as well. The training with different seeds converge to the same results as we have multiple clusters, (even though with fewer iterations the results still appear to diverge) which shows that in the context of non-convexity of the loss optimization, EM algorithm with different starting points arrive at the optimal result within 25 iterations.

CP-4b Problem Statement

EM best model with K=08

Solution



The means and standard deviations are also organized in the order of decreasing probability of being in the given cluster. Observing the mean, we see that many of the clusters are not visibly different from each other. However, unlike the result for LBFGS above, the traces of the items in the stddev plots are still visible (i.e we can see the actual piece of clothing here, even with the cluster at the end of the row). This aligns with the graph in 4b because we know that the EM algorithm actually converges to an optimal result, so we obtain the definite shape of the items in each cluster, also making the clustering result more accurate (lower loss). Another indication of the algorithm reaching convergence is the fact that the shapes (in the mean) are generalized, which makes sense cause they are the representation of that whole cluster, and in the stddev, the areas of the actual shirt are lighter, especially in the borders, which make sense because the part that is definitely not the shirt (borders of the images) shouldn't have much variation, whereas most variation should be in the area of the shirts (different textures, printed images, slight differences in location and width/length of arms and body).

CP-4c Problem Statement

EM Scores vs Number of Clusters K

Solution:

k	Validation negative log likelihood per pixel
1	0.28889111387949
4	-0.214584203851517
8	-0.271315836753881
16	-0.301758125000127

CP-4d Problem Statement

Interpret the results of the table in 4c.

Solution:

K = 4 is particularly better since we see a big decrease in the negative log likelihood. After k=4, the log likelihood increase very little. Also considering the fact that there are actually only 3 categories of items in the dataset, adding more cluster doesn't make the result much better. However, since the values shown are of validation set, technically the lower score we get, the better. However, in terms of interpretability, as we observed for k = 8, the clusters are not distinctly different from each other, so with k = 16, it is also very hard to interpret what types of items are in each cluster, compared to k = 4. So I think, depending on what we place more importance upon (better accuracy or better interpretation), we can stay with k=4, or go with more clusters.

CP-4e Problem Statement

Reflect on differences between EM and LBFGS

Solution: The loss we observe for EM is much lower than that of LBFGS, resulting in better clustering result of the images (clearer images). One reason for this is because LBFGS takes longer and more iterations to converge to the optimal result whereas EM takes very few iterations. EM algorithm always decreases loss (while gradient descent might if your step size is

large) and as we see above, with different starting points, the algorithm still reaches the best result, so it's more stable than gradient descent algorithms, especially in the case of non-convex loss optimization problems. EM is usually useful when it's hard to calculate the gradient of the log-likelihood, the reason is EM doesn't impose that we maximize at each descent like LBFGS, but as long as the loss "improves", we can move on to the next iteration, and this still guarantees convergence to the best result.