Machine Learning HW2

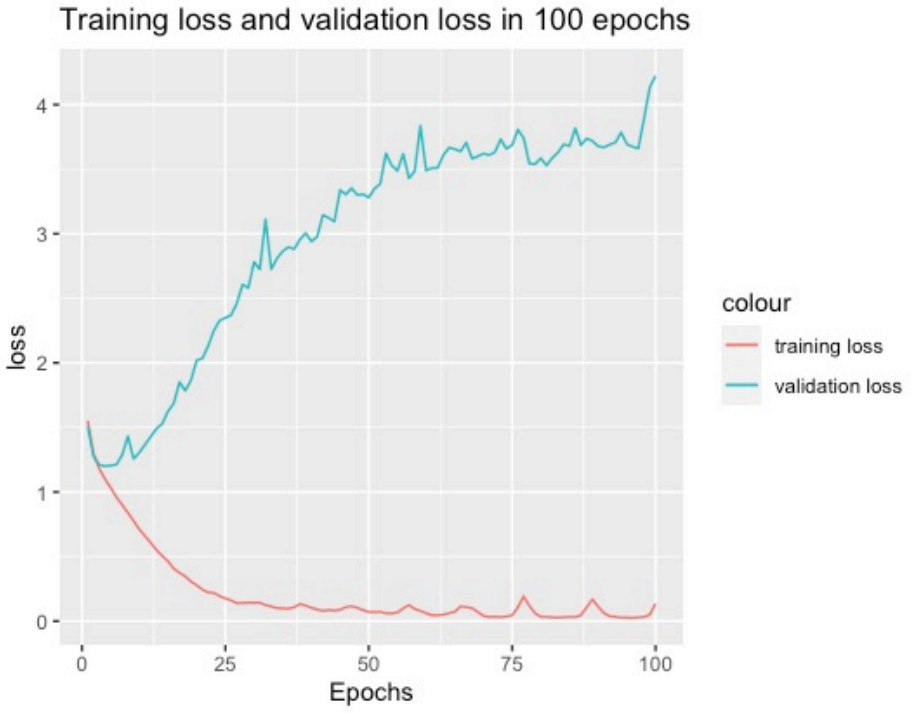
B09902005 資工三盧冠綸

October 19, 2022

programming part

In my folder, there are several files. `best_model.py` and `best_model.ipynb` (they both have the same code) contains the code that generate my best file, which is in `best_model.pth` (the one with public score 0.63333).
And, since the resource limit of colab may not be enough for `best_model.ipynb`, which has a larger data size, to implement without early-stopping, so I use `early_stop.py` and `early_stop.ipynb` (they both have the same code), which I didn't do any augmentation on the data, to answer problem 1.

problem 1: 實作 early-stopping，繪製 training, validation loss 的 learning curve，比較實作前後的差異，並說明 early-stopping 的運作機制。(1pts)
Below is the learning curve of `early_stop.ipynb` with 100 epochs.



Below is the pseudo code of how I implement early-stopping:

```
stop = 10
for epoch in range (NUM_EPOCH):
    stop -= 1
    if (new model accuracy is the max in acc_record):
        save_checkpoint(model)
        stop = min(10, stop+5)
    if (new model loss is smaller than the previous one):
        stop = min(10, stop+2)
    if (stop == 0):
        break
```

In fact, early-stopping is the process that may trace the tendency of loss or accuracy by using validation data. If the validation loss starts to increase and seems like that it'll keep increasing in the following epochs (or validation accuracy starts to decrease and seems to be decreasing in the following epochs), then we stop the training process to save time and prevent from over-fitting.
Without early-stopping, my code will run 100 epochs and the model it chose will be in the 9-th epoch with validation loss 1.2566 and accuracy 0.5580.
With early-stopping, my code will stop at the 21-st epoch, and it'll also choose the model in the 9-th epoch, which has the same output with that code without early-stopping. In fact, seeing the validation loss of the first 21 epochs, we may also speculate that the validation loss may keep increasing if we train more, so we decide to stop the training process at the 21-st epoch.
So, doing early-stopping can make our code complete in a shorter time and also prevent from the risks of over-fitting.

problem 2: 嘗試使用 augmentation，說明實作細節並比較有無該 trick 對結果表現的影響 (validation 或是 testing 擇一即可)，且需說明爲何使用這些 augmentation 的原因。(1pts)
I implement augmentation by 5 transformation functions (in my code, which is `transform1`, `transform2`, `transform3`, `transform4`, and `transform5`), which has different parameters to rotate the picture, crop the picture, horizontally flip the picture, and resize the picture to the original size.
Then, I generate 5 datasets (in my code, which is `train_dataset1`, `train_dataset2`, `train_dataset3`, `train_dataset4`, and `train_dataset5`), and combine them together (in my code, which is `train_dataset`). So, the dataset with augmentation is five times larger than that without augmentation.

Below is the performance of the model I trained with and without augmentation:

	validation accuracy	testing accuracy
augmentation	0.6233	0.62866
no augmentation	0.5580	0.57199

From the table, we do have strong evidence that this augmentation helps us improve our model.
The benefits of augmentation is, we can create more data for our model, which makes our data become more diverse and thus may prevent from over-fitting.
In my implementation, since pictures of people's face may be crooked by some angle, so I try to let machines see these pictures in different angles. Since size of faces in everyone's picture may not be always the same, so I crop the pictures with different parameters. To add symmetry of the picture, I horizontally flip the pictures in half probability. These augmentations can help our model train better.

problem 3: 畫出 confusion matrix 分析哪些類別的圖片容易使 model 搞混，找出模型出錯的例子，並分析可能的原因。(1pts)

predicted label Tune label	Angry	Disgust	Fear	Happy	Sad	Surprised	Neutral
Angry	0.55	0.01	0.08	0.05	0.17	0.03	0.12
Disgust	0.43	0.36	0.02	0.09	0.09	0.00	0.00
Fear	0.11	0.00	0.38	0.08	0.23	0.09	0.11
Happy	0.02	0.00	0.01	0.86	0.03	0.02	0.06
Sad	0.11	0.00	0.11	0.06	0.53	0.01	0.18
Surprised	0.04	0.01	0.09	0.04	0.04	0.73	0.04
Neutral	0.05	0.00	0.03	0.10	0.17	0.02	0.63

From the table, we can see that if an image is labeled with disgust or fear, then the probability that my model makes a mistake is larger than 50%. I think the model cannot precisely recognize pictures labeled with disgust because the amount of these pictures is too small (358 out of 22780), so the model cannot fit well. I think my model cannot predict fear well because fear is a relatively complex emotion for human among the 7 types of labels, so everyone's face when expressing fear may be various.

Figure 1: fear image 1 (10500.jpg)

Figure 2: fear image 2 (14526.jpg)

Here, I list two pictures labeled with fear which my model cannot recognize fear from them. The left picture is 10500.jpg, and the right one is 14526.jpg, which are in my validtion data. From the two pictures, we can see that the two people use different ways to express their fear.
The woman on the left uses her hands to cover her mouth while feeling fear, so the model may have difficulties guessing her emotions through her mouth. In fact, my model says she is feeling surprised, which seem also a reasonable prediction.
The baby on the right opens his eyes and mouth wide. This may seem easier for the model to observe his eyes and mouth than the left one. However, this is still confusing for the model. When we're feeling happy and we laugh out loud, we also open our mouth wide. Thus, a wide mouth may be a feature that indicates happy, so the model makes a mistake on this picture.

problem 4: 請統計訓練資料中不同類別的數量比例，並說明：對 **testing** 或是 **validation** 來說，不針對特定類別，直接選擇機率最大的類別會是最好的結果嗎？**(1pts)**
In the training data, there are 3139 labeled with angry, 358 labeled with disgust, 3296 labeled with fear, 5762 labeled with happy, 3785 labeled with sad, 2515 labeled with surprised, and 3925 labeled with neutral. We can see that pictures labeled with disgust is rare (about 1.6% among the training data), so we may have to deal with this.
One way to deal with this is to use a weighted loss function. And another way to deal with this is to resampling the dataset. We can use augmentation technique to imcrease the numbers of images labeled with disgust.

Here, I did some experiment of resampling dataset. I use five augmentation methods to add more images labeled with disgust in my dataset, so the amount of images labeled with disgust in the new dataset will be five times more than that in the original dataset.
Below is the comparison of the model trained by the new dataset and the model trained by the original dataset.

	valid loss	valid acc	test acc	valid acc of disgust
resampling	1.2321	0.5792	0.57133	0.5667
no resampling	1.2566	0.5580	0.57199	0.2500

We can see that the difference of loss and accuracy between the models are not very large, but the model trained with resampled dataset can correctly recognize more than half images labeled with disgust, which is even better than the model trained in `best_model.ipynb`.
So, besides directly choose the one with the largest probability, I think there may be another ways to train our model. We can check the proportion of all the labels first, and then decide whether to use another loss function or to use resampled dataset to deal with unbalanced data. This way, the model may have a better ability to classify some labels, and maybe the total loss and accuracy will be even better if we train it well.

mathematics part

problem 1: (1pts)
We want to find $\pi = (\pi_1, \pi_2, ..., \pi_K)$ with $\Sigma_{i=1}^{K} \pi_i = 1$, to maximize

$$\Pi_{i=1}^{K} \pi_i^{N_i}$$

That is, we want to maximize

$$\Sigma_{i=1}^{K} N_i \log \pi_i$$

.
In other words, we want to find $(\pi_1, \pi_2, ..., \pi_{K-1})$ to maximize

$$N_K \log(1 - \Sigma_{i=1}^{K-1} \pi_i) + \Sigma_{i=1}^{K-1} N_i \log \pi_i$$

If we denote the above function $L(\pi)$, then $\forall 1 \leq i \leq K-1$, we have

$$\frac{\partial L(\pi)}{\partial \pi_i} = -\frac{N_K}{1 - \Sigma_{i=1}^{K-1} \pi_i} + \frac{N_i}{\pi_i}$$

So, if we choose proper $(\pi_1, \pi_2, ..., \pi_{K-1})$ to maximize $L(\pi)$, then $\forall 1 \leq i \leq K-1$, we have

$$-\frac{N_K}{1-\Sigma_{i=1}^{K-1}\pi_i} + \frac{N_i}{\pi_i} = -\frac{N_K}{\pi_K} + \frac{N_i}{\pi_i} = 0$$

.

So, we have

$$\pi_1 : \pi_2 : ... : \pi_K = N_1 : N_2 : ... : N_K$$

And since $\Sigma_{i=1}^{K}\pi_i = 1$, thus we can conclude that $\forall 1 \leq i \leq K$,

$$\pi_i = \frac{N_i}{N}$$

problem 2: (1pts)

(a) First, consider $A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{bmatrix}$, and $w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$, then

$$\mathbf{w}^T\mathbf{A}\mathbf{w} = \Sigma_{i=1}^{m}\Sigma_{j=1}^{m} w_i w_j a_{ij}$$

Thus, $\forall 1 \leq i \leq m$,

$$\frac{\partial \mathbf{w}^T\mathbf{A}\mathbf{w}}{\partial w_i} = \Sigma_{j=1}^{m} w_j a_{ij} + \Sigma_{j=1}^{m} w_j a_{ji}$$

So, we can conclude that

$$\frac{\partial \mathbf{w}^T\mathbf{A}\mathbf{w}}{\partial \mathbf{w}} = \begin{bmatrix} \Sigma_{j=1}^{m} w_j a_{1j} + \Sigma_{j=1}^{m} w_j a_{j1} \\ \vdots \\ \Sigma_{j=1}^{m} w_j a_{mj} + \Sigma_{j=1}^{m} w_j a_{jm} \end{bmatrix} = \begin{bmatrix} \Sigma_{j=1}^{m} w_j a_{1j} \\ \vdots \\ \Sigma_{j=1}^{m} w_j a_{mj} \end{bmatrix} + \begin{bmatrix} \Sigma_{j=1}^{m} w_j a_{j1} \\ \vdots \\ \Sigma_{j=1}^{m} w_j a_{jm} \end{bmatrix} = \mathbf{A}\mathbf{w} + \mathbf{A}^T\mathbf{w}$$

(b) Note that

$$trace(AB) = \Sigma_{i=1}^{m}(\Sigma_{j=1}^{m} a_{ij} b_{ji})$$

From the equation above, we can easily conclude that

$$\frac{\partial trace(AB)}{\partial a_{ij}} = b_{ji}$$

(c) First, note that $\forall 1 \leq i \leq m$, we have

$$\frac{\partial \det(\Sigma)}{\partial \sigma_{ij}} = \frac{\partial \Sigma_{k=1}^{m} (-1)^{i+k} \sigma_{ik} \det(\Sigma_{ik})}{\partial \sigma_{ij}} = \Sigma_{k=1}^{m} (-1)^{i+j} \det(\Sigma_{ij})$$

So, we can know that

$$\frac{\partial \log(\det(\Sigma))}{\partial \sigma_{ij}} = \frac{\partial \log(\det(\Sigma))}{\partial \det(\Sigma)} \frac{\partial \det(\Sigma)}{\partial \sigma_{ij}} = \frac{(-1)^{i+j} \det(\Sigma_{ij})}{\det(\Sigma)}$$

Then, according to Cramer's Rule, if $\Sigma x = e_i$, then $x = \begin{bmatrix} \frac{(-1)^{i+1} \det(\Sigma_{i1})}{\det(\Sigma)} \\ \vdots \\ \frac{(-1)^{i+m} \det(\Sigma_{im})}{\det(\Sigma)} \end{bmatrix}$

So, if $\Sigma x = e_i$, then $e_j^T \Sigma^{-1} e_i = e_j^T \Sigma^{-1} \Sigma x = e_j^T x = \frac{(-1)^{i+j} \det(\Sigma_{ij})}{\det(\Sigma)}$.

From this, we can conclude that

$$\frac{\partial \log(\det(\Sigma))}{\partial \sigma_{ij}} = \frac{(-1)^{i+j} \det(\Sigma_{ij})}{\det(\Sigma)} = e_j^T \Sigma^{-1} e_i$$

problem 3: (1pts)

Consider the probability density function of D-dimensional Gaussian distribution given mean μ_k and covariance matrix Σ:

$$p(\mathbf{x}|\mu_k, \Sigma) = \frac{1}{(2\pi)^{\frac{2}{D}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma^{-1} (\mathbf{x}-\mu_k)}$$

Since for all data $(\mathbf{x}_i, t_i)$, we know $p(\mathbf{x}_i)$ will follow the distribution $p(\mathbf{x}_i|t_i^T \mu, \Sigma)$, where $\mu = [\mu_1, \mu_2, ..., \mu_K]$, so we want to maximize the function:

$$\Pi_{i=1}^{N} \frac{1}{(2\pi)^{\frac{2}{D}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_i-\mu t_i)^T \Sigma^{-1} (\mathbf{x}_i-\mu t_i)}$$

That is, we want to maximize:

$$\Sigma_{i=1}^{N} (-\frac{2}{D} \log(2\pi) - \frac{1}{2} \log|\Sigma| - \frac{1}{2} (\mathbf{x}_i - \mu t_i)^T \Sigma^{-1} (\mathbf{x}_i - \mu t_i))$$

In other words, we want to minimize the function below:

$$\Sigma_{i=1}^{N} (\log|\Sigma| + (\mathbf{x}_i - \mu t_i)^T \Sigma^{-1} (\mathbf{x}_i - \mu t_i))$$

We call the function above $L(\mu, \Sigma)$. Then, for all μ_k, we have:

$$\frac{\partial L(\mu, \Sigma)}{\partial \mu_k} = \frac{\partial \Sigma_{x_i \in C_k} (\mathbf{x}_i - \mu_k)^T \Sigma^{-1} (\mathbf{x}_i - \mu_k)}{\partial \mu_k}$$

Then, for convenience, we let $\mathbf{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,m} \end{bmatrix}$, $\mu_k = \begin{bmatrix} \mu_{k,1} \\ \vdots \\ \mu_{k,m} \end{bmatrix}$, $\Sigma^{-1} = \begin{bmatrix} \Sigma_{1,1}^{-1} \cdots \Sigma_{1,m}^{-1} \\ \vdots \ddots \vdots \\ \Sigma_{m,1}^{-1} \cdots \Sigma_{m,m}^{-1} \end{bmatrix}$,
then since $(\mathbf{x}_i - \mu_k)^T \Sigma^{-1} (\mathbf{x}_i - \mu_k) = \Sigma_{j=1}^{m} (x_{i,j} - \mu_{k,j})^2 \Sigma_{j,j}^{-1}$
So, we can derive that

$$\frac{\partial L(\mu, \Sigma)}{\partial \mu_{k,j}} = \frac{\partial \Sigma_{x_i \in C_k} \Sigma_{t=1}^{m} (x_{i,t} - \mu_{k,t})^2 \Sigma_{t,t}^{-1}}{\partial \mu_{k,j}} = \Sigma_{x_i \in C_k} (-2(x_{i,j} - \mu_{k,j}) \Sigma_{j,j}^{-1})$$

Consider the condition that $\frac{\partial L(\mu,\Sigma)}{\partial \mu_{k,j}} = 0$ for all k and j, we have to let
$\Sigma_{x_i \in C_k} (-2(x_{i,j} - \mu_{k,j}) \Sigma_{j,j}^{-1}) = 2\Sigma_{j,j}^{-1} (N_k \mu_{k,j} - \Sigma_{x_i \in C_k} x_{i,j}) = 0$ for all j and k. Which means,

$$\mu_{k,j} = \frac{1}{N_k} \Sigma_{x_i \in C_k} x_{i,j} = \frac{1}{N_k} \Sigma_{i=1}^{N} t_i^k x_{i,j}$$

So,

$$\mu_k = \frac{1}{N_k} \Sigma_{i=1}^{N} t_i^k x_i$$

After deriving the value of all μ_k, then we have to derive Σ to minimize $L(\mu, \Sigma)$ using the value of μ. Note that for all i and j, we have

$$\frac{\partial L(\mu, \Sigma)}{\partial \sigma_{kj}} = \frac{\partial \Sigma_{i=1}^{N} (\log |\Sigma| + (\mathbf{x}_i - \mu t_i)^T \Sigma^{-1} (\mathbf{x}_i - \mu t_i))}{\partial \sigma_{kj}}$$

Given that $\frac{\partial L(\mu,\Sigma)}{\partial \sigma_{kj}} = 0$ for all k and j, we should also have

$$\frac{\partial \Sigma_{i=1}^{N} (\log |\Sigma| + (\mathbf{x}_i - \mu t_i)^T \Sigma^{-1} (\mathbf{x}_i - \mu t_i))}{\partial \Sigma_{k,j}^{-1}} = 0$$

That is, we want to find Σ^{-1} first, so that we can also know Σ.
First, from the conclusion we have derived in problem 2.(c), we know

$$\frac{\partial \Sigma_{i=1}^{N} (\log |\Sigma|)}{\partial \Sigma_{k,j}^{-1}} = N \frac{\log |\Sigma|}{\partial \Sigma_{k,j}^{-1}} = N \frac{-\log |\Sigma^{-1}|}{\partial \Sigma_{k,j}^{-1}} = -N\sigma_{jk}$$

On the other hand, from the hint of this problem and the conclusion in problem 2. (b), we can also know:

$$\frac{\partial \Sigma_{i=1}^{N} (\mathbf{x}_i - \mu t_i)^T \Sigma^{-1} (\mathbf{x}_i - \mu t_i)}{\partial \Sigma_{k,j}^{-1}}$$

$$= \frac{\partial \Sigma_{i=1}^{N} trace(\Sigma^{-1} (\mathbf{x}_i - \mu t_i)(\mathbf{x}_i - \mu t_i)^T)}{\partial \Sigma_{k,j}^{-1}}$$

$$= \frac{\partial \Sigma_{l=1}^{K} \Sigma_{x_i \in C_l} trace(\Sigma^{-1}(\mathbf{x}_i - \mu_l)(\mathbf{x}_i - \mu_l)^T)}{\partial \Sigma_{k,j}^{-1}}$$

$$= \Sigma_{l=1}^{K} \Sigma_{x_i \in C_l} (\mathbf{x}_{i,j} - \mu_{l,j})(\mathbf{x}_{i,k} - \mu_{l,k})$$

From the calculation above, we can know that for all j and k,

$$-N\sigma_{jk} + \Sigma_{l=1}^{K} \Sigma_{x_i \in C_l} (\mathbf{x}_{i,j} - \mu_{l,j})(\mathbf{x}_{i,k} - \mu_{l,k}) = 0$$

So,

$$\Sigma = \begin{bmatrix} \sigma_{1,1} \cdots \sigma_{1,m} \\ \vdots \ddots \vdots \\ \sigma_{m,1} \cdots \sigma_{m,m} \end{bmatrix} = \frac{\Sigma_{k=1}^{K} \Sigma_{x_i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T}{N} = \frac{\Sigma_{k=1}^{K} \Sigma_{i=1}^{N} t_i^k (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T}{N}$$

So, from the equation above, if we set $\Sigma_k = \frac{1}{N_k} \Sigma_{i=1}^{N} t_i^k (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T$, then

$$\Sigma = \Sigma_{k=1}^{K} \frac{N_k}{N} \Sigma_k$$

problem 4: (1pts)

(a) Let $z_i = \begin{bmatrix} z_{i1} \\ \vdots \\ z_{id} \end{bmatrix}$, and $z = \begin{bmatrix} z(1) \\ \vdots \\ z(d) \end{bmatrix}$, then $\Sigma_{i=1}^{m} ||z_i - z||_2^2 = \Sigma_{i=1}^{m} \Sigma_{j=1}^{d} (z_{ij} - z(j))^2$
Then, for all j, we know

$$\frac{\partial \Sigma_{i=1}^{m} ||z_i - z||_2^2}{\partial z(j)} = \Sigma_{i=1}^{m} - 2(z_{ij} - z(j)) = 2m(z(j)) - 2\Sigma_{i=1}^{m} z_{ij}$$

So, to find $z(j)$ that minimizes $\Sigma_{i=1}^{m} ||z_i - z||_2^2$, we should let $z(j) = \frac{1}{m} \Sigma_{i=1}^{m} z_{ij}$.

So, to find z that minimizes $\Sigma_{i=1}^{m} ||z_i - z||_2^2$, we should let $z = \frac{1}{m} \Sigma_{i=1}^{m} z_i$.

Thus, if $\bar{z} = \frac{1}{m} \Sigma_{i=1}^{m} z_i$, then for any $z \in \mathbb{R}^d$, we have

$$\Sigma_{i=1}^{m} ||z_i - z||_2^2 \geq \Sigma_{i=1}^{m} ||z_i - \bar{z}||_2^2$$

(b) $L(C^{t+1}, \mu^t) = \Sigma_{i=1}^{n} ||x_i - \mu_{C^{t+1}(i)}^t||_2^2 \leq \Sigma_{i=1}^{n} ||x_i - \mu_{C^t(i)}^t||_2^2 = L(C^t, \mu^t)$

(c) Since $\mu_q^{t+1} = \frac{1}{N_q^{t+1}} \Sigma_{i:C^{t+1}(i)=q} x_i \forall q$, where N_q^{t+1} is the amount of points x_i that satisfies $C^{t+1}(i) = q$. So, from the conclusion we've made in (a), we can see
$L(C^{t+1}, \mu^{t+1}) = \Sigma_{q=1}^{k} \Sigma_{i:C^{t+1}(i)=q} ||x_i - \mu_q^{t+1}||_2^2 \leq \Sigma_{q=1}^{k} \Sigma_{i:C^{t+1}(i)=q} ||x_i - \mu_q^t||_2^2 = L(C^{t+1}, \mu^t)$

(d) From the formula of $L(C^t, \mu^t)$, we can easily see that $l_t = L(C^t, \mu^t) \geq 0$.

And, from the result of (b) and (c), we can conclude that for any t,

$$l_t = L(C^t, \mu^t) \geq L(C^{t+1}, \mu^t) \geq L(C^{t+1}, \mu^{t+1}) = l_{t+1} \geq 0$$

.

So, the loss of k-means clustering algorithm is monotonic decreasing.

(e) From the definition of C, we can see that there are at most n^k possible combinations of C. So, no matter what C^0 is, we must have the property: $\exists 1 \leq a < b \leq n^k + 1$, s.t. $C^a = C^b$.

Then, since C^{t+1} is determined by μ^t and all points x_i, and μ^t is determined by C^t and all points x_i, so if $C^a = C^b$, then $\mu^a = \mu^b$, then $C^{a+1} = C^{b+1}$.

By iteration, we can then easily see, if $C^a = C^b$, then $C^{a+i} = C^{b+i}$ for any $i \geq 0$. And thus, $C^a = C^b = C^{a+i(b-a)}$ for any $i \geq 0$.

Then, l^t is determined by C^t, μ^t, and all points x_i, and μ^t is determined by C^t and all points x_i, thus we can say that l^t is determined by C^t and all points x_i. Which means, if $C^a = C^b$, then $l_a = l_b$.

So, from the statements above, we may have a sense that
$\exists 1 \leq a < b \leq n^k + 1$, s.t. $l_a = l_{a+i(b-a)}$ for all $i \geq 0$.

Finally, since we've known that $l_t \geq l_{t+1}$ for all t, so we can improve our property: $\exists 1 \leq a < n^k + 1$, s.t. $l_a = l_{a+i}$ for all $i \geq 0$. This implies that k-means clustering algorithm converges in finitely many steps (at most n^k steps).