Machine Learning HW4

B09902005 資工三盧冠綸

November 12, 2022

programming part

In the programming part, I discussed with B09502019 and B09901170.
In my files, model.pth is my best model, but it is generated by running the training part of HW4_sample_code.ipynb for many times (totally 2000 epochs, the first 1500 with `lr` $= 10^{-5}$, while the last 500 with `lr` $= 5 * 10^{-4}$).

problem1: (1.5%) 請說明 RNN、GRU、LSTM 等模型之間的異同。另外，如果你 Kaggle 上最佳的預測結果並不是使用上述三種模型產生的話，請額外說明你使用的 model 爲何，以及簡介其背後的原理/機制。

My model is generated by RNN. RNN is a neural network that former inputs may influence outputs of latter inputs. It's basic structure is as below (This is a picture from lecture's slide), where for any number i, x^i is the i-th input (In this homework, it is the i-th word in a sentence), y^i is the i-th output, and a^i is a temporary value.

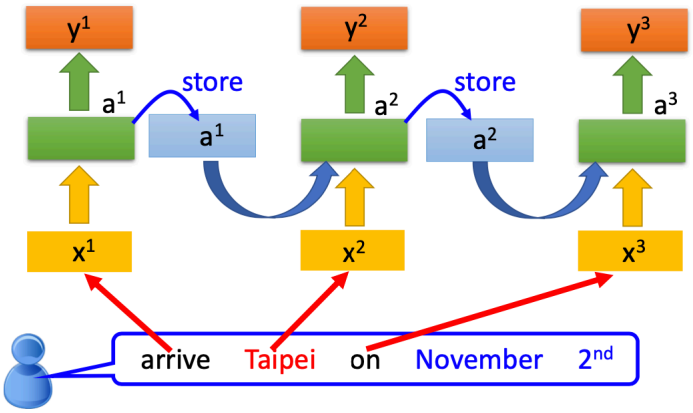

Figure 1: structure of RNN

In the picture, we can regard the green rectangle as a function that takes x^i and a^{i-1} as inputs and then outputs y^i and a^i. So, y^i will in fact be influenced by x^i and any inputs before x^i in RNN.

LSTM has three gates and a memory cell (the picture on the left). The three gates can decide which information is to be remembered or to be forgotten (The f function in the right picture can do this), so we can choose important information only by these gates. The memory cell is c^i in the right picture, which is similar to a^i in RNN. In the picture on the right, we can see that it takes z, z_i, z_f, and z_o as inputs, which are decided by input x and some matrices. Then, it outputs a. Sometimes choosing LSTM instead of RNN can be a good way to prevent from gradient vanishing.

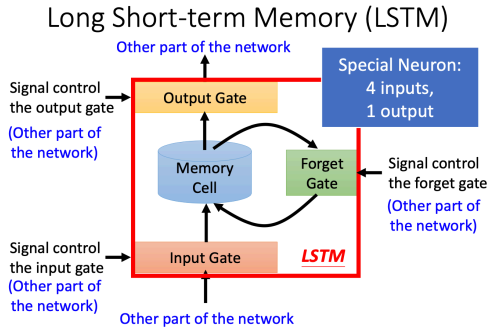

Figure 2: basic structure of LSTM

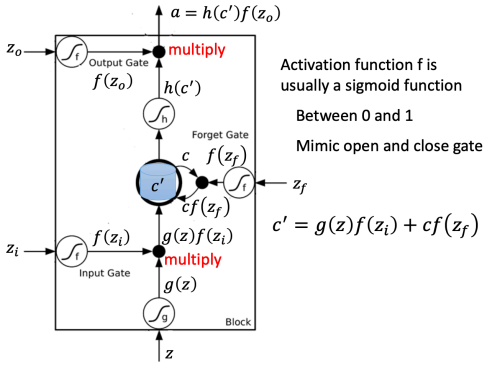

Figure 3: how to implement LSTM

However, LSTM has too much parameters, thus it may cost lots of time and space while training, so sometimes we may choose GRU in order to save time.
The basic concept of GRU is similar to RNN. It also takes x^i as input, takes a^{i-1} as temporary value, and then outputs y^i.
However, GRU combines input gate and forget gate together, named "reset gate", which decides value of $a^{i-1\prime}$ by a^{i-1} (In LSTM, value of c' is decided by c, input gate, and forget gate.). Then, the value of a^i is determined by "update gate", which takes $a^{i-1\prime}$ and a^{i-1} as input. So, update gate decides the value of a^i and y^i. (while LSTM does this by output gate)
So, GRU uses only 2 gates, while LSTM uses 3. Since parameter of GRU is relatively less than GRU, so GRU is a good choice when we want to train in a shorter time.

problem2: (0.5%) 請解釋爲何 RNN 模型會發生 gradient vanishing 以及 gradient exploding，以及這兩個現象對 training 可能會有什麼不良影響。

We can find the reasons of gradient vanishing or gradient exploding from Figure 1.

Note that in Figure 1, we can see that a^i is decided by a^{i-1} and x^i.
Due to chain rule, if there are totally n inputs (from x^1 to x^n), then

$$\frac{\partial y^n}{\partial x^1} = \frac{\partial y^n}{\partial a^{n-1}} \frac{\partial a^{n-1}}{\partial a^{n-2}} \cdots \frac{\partial a^2}{\partial a^1} \frac{\partial a^1}{\partial x^1}$$

So, if absolute value of $\frac{\partial a^i}{\partial a^{i-1}}$ tend to be smaller than 1 for all i, then $\frac{\partial y^n}{\partial x^1}$ will be very close to 0 if n is large enough. This means that if an input is still very far from the output, then it will hardly affect the value of the output, this is called gradient vanishing.
On the contrary, if absolute value of $\frac{\partial a^i}{\partial a^{i-1}}$ tend to be larger than 1 for all i, then $\frac{\partial y^n}{\partial x^1}$ will be very very large if n is large enough. This means that if an input is still very far from the output, then it will strongly affect the value of the output, this is called gradient exploding.

Below is a simple example of gradient vanishing and gradient exploding. Here, $x^1 = 1$, $y^1 = x^1$, $a^1 = y^1$, and for all $i \neq 1$, $x^i = 0$, $y^i = wa^{i-1} + x^i$, $a^i = y^i$.
(This example is also from lecture's slides)

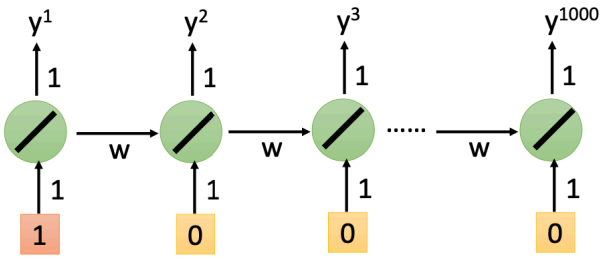

Figure 4: example of gradient vanishing and gradient exploding

So, from the structure above, $\frac{\partial y^n}{\partial x^1} = \frac{\partial y^n}{\partial a^{n-1}} \frac{\partial a^{n-1}}{\partial a^{n-2}} \cdots \frac{\partial a^2}{\partial a^1} \frac{\partial a^1}{\partial x^1} = w^{n-1}$. That is, if $w = 0.99$, then y^{1000} will be very close to 0, and $\frac{\partial y^{1000}}{\partial x^1}$ is also very close to 0. If $w = 1.01$, then both y^{1000} and $\frac{\partial y^{1000}}{\partial x^1}$ will be very large.

If gradient vanishing occurs, then inputs that are far from output will hardly affect the value of output. Thus, the model may not learn anything from those inputs. (for example, the model may think "Never gonna run around and desert you" and "I'm gonna run around and desert you" have the same meaning because "Never" and "I'm" are too far from the end of the sentence.)
If gradient exploding occurs, then the outputs may become exaggerating, and the model will be very unstable, so the model cannot learn effectively nor smoothly.

problem3: (1%) 相較於 **Sample Code** 來說，你做了哪些修改或嘗試 (如模型架構、資料前處理、後處理等)？請描述你做的嘗試以及其理由，如果你認爲你的做法帶來的進步與第一題的回答有關的話也請詳述之。(請注意若你的解釋太過不合理，則不論你在 **leader board** 上分數多高，這題都無法拿到滿分)

What I do can be divided into three parts:

1. **parameters:** First, I modify `EPOCH_NUM`. I found that my model's validation accuracy keeps increasing during multiple epochs, but it increases very slowly, so it needs many epochs to train. Thus, my `EPOCH_NUM` is more than 1500 and I seperate the training process into several days.

   After 1500 epochs, I found that the accuracy increases too slowly, so I modified learning rate from 10^{-5} into $5 * 10^{-4}$ instead. In fact, this helped me save a lot of time. Below is the relationship of the validation accuracy and epoches needed for my model, from this we can see that the accuracy increased much faster after the 1500-th epoch.

validation acc	70.0	70.5	71.0	71.5	72.0	72.5	73.0
epochs needed	578	731	873	1115	1432	1550	1678

   Then, I modified the dimension of the embedding matrix into 128. For the RNN net, I modified its hidden dimension into 64 and let it has 3 layers. I think more parameters in embedding matrix and in RNN net can make the accuracy better.

2. **text preprocessing:** First, for all words starting with '@', I regard them as "@" only, so the contents after @ don't matter for my model. I do this because I think the contents after @ won't affect the meanings of the sentence.

   Then, I separate punctuation, such as ',', '.', '!', '?', from the word it connects to. For example, if there is a word "Great!" in data, I'll take it for two words "Great" and "!". This way, the model may be more sensitive to those punctuation marks.

3. **else:** I tried to modify the structure of the classifier of the header, and tried to use GRU or LSTM instead of RNN to train my model, and use CBOW instead of skip-gram, but the performance doesn't seem to be better after these modification.

problem4: (0.5%) 請簡述你 leader board 上表現最好的實驗結果中使用的 embedding 爲何？如何產生？

My best model uses skip-gram to generate the embedding.
Skip-gram uses a word to predict words near it. For example, given a word ”RNN”, then the words ”LSTM” or ”GRU” may be more likely to appear near it than the words ”Cinderella” or ”Christmas”. Then, it uses its prediction and uses the dataset to transform these words into vectors, and uses gradient descent to update these vectors to make them more precise.
Compared with CBOW, skip-gram may cost more time because it has to make more estimations than CBOW. However, skip-gram may have better performance if the dataset is small or the word is relatively rare.

problem5:
Below is my code and output for dealing with problem 5. (The code is at the end of my model.)

```
tempmodel = torch.load("model.pth")
backbone = tempmodel["backbone"]
header = tempmodel["header"]

texts = ["You know the rules, and so do I.", "I know the rules, and so do You.", "Looks great, Zuck"]
temp_idx = [i for i in range (3)]
temp_text = [text.split(' ') for text in texts]
temp_text = parsing_text(temp_text, common_word)

temp_dataset = TwitterDataset(temp_idx, temp_text, None, preprocessor)
temp_loader = torch.utils.data.DataLoader(dataset = temp_dataset,
                                          batch_size = BATCH_SIZE,
                                          shuffle = False,
                                          collate_fn = test_dataset.collate_fn,
                                          num_workers = 8)
tempfile = "a.csv"
run_testing(temp_loader, backbone, header, device, tempfile)

print("----------Input sentences----------")
for text in texts:
  print(text)

print("----------Outputs of my model----------")
with open(tempfile, newline='') as csvfile:
  rows = csv.reader(csvfile)
  for row in rows:
    print(row)
```

```
----------Input sentences----------
You know the rules, and so do I.
I know the rules, and so do You.
Looks great, Zuck
----------Outputs of my model----------
['id', 'label']
['0', '0']
['1', '1']
['2', '1']
```

1. **(0.5%) 在本題中，s1、s2 互爲彼此的 valid permutation，若且唯若 s1、s2 兩句子的單字種類、數量相同、排列順序不同且各自皆爲有意義並且合乎文法的句子。例如，A student is a banana 是 A banana is a student 的 valid permutation。請找出一組互爲彼此的 valid permutation 且使你的 model 產生相反的 prediction 的 s1、s2。(s1、s2 須具備合乎邏輯且有實際生活意義的語意)**

   From the picture in the previous page, we can see that if `s1` is "You know the rules, and so do I.", and `s2` is "I know the rules, and so do You.", then my model will have different outputs on `s1` and `s2`.

2. **(1%，bonus) 請從網路上 (如 FB、IG、Twitter) 找出一則能夠讓 model 預測錯誤的「反串」酸留言，並將截圖附於 report 上**

   The left picture below is a post from Mark Zuckerberg in Facebook, while the picture on the right is the messages below the post.

Figure 5: The post

Figure 6: The messages below

There is a message "Looks great, Zuck", which is a message that looks like positive but is in fact negative. We can see most people replied "ha" instead of "thumb", which is an evidence that people know that this message is negative. However, my model outputs positive for this message.

mathematics part

problem 1: LSTM Cell (1%)

t	z	$g(z)$	z_i	$f(z_i)$	z_f	$f(z_f)$	c	c'	$h(c')$	z_o	$f(z_o)$	y
1	3	3	90	1	10	1	0	3	3	-10	0	0
2	-2	-2	90	1	10	1	3	1	1	90	1	1
3	4	4	190	1	-90	0	1	4	4	90	1	4
4	0	0	90	1	10	1	4	4	4	90	1	4

problem 2: Backpropagation through time via Simple RNN (1%)

First, note that

$$\frac{\partial L(y,\hat{y})}{\partial \hat{y}} = -\frac{\partial y\log(\hat{y}) + (1-y)\log(1-\hat{y})}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial \hat{y}}{\partial W_o} = \frac{\partial \sigma(W_o h_2)}{\partial W_o} = h_2 \frac{\partial \sigma(W_o h_2)}{\partial W_o h_2} = h_2 \hat{y}(1-\hat{y})$$

So,

$$\frac{\partial L(y,\hat{y})}{\partial W_o} = \frac{\partial L(y,\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial W_o} = (-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}})h_2\hat{y}(1-\hat{y}) = h_2(-y(1-\hat{y}) + (1-y)(\hat{y})) = h_2(\hat{y}-y)$$

Then, note that $\frac{\partial h_1}{\partial W_h} = \frac{\partial h_1}{\partial (W_i x_1 + W_h h_0)}\frac{\partial (W_i x_1 + W_h h_0)}{\partial W_h}$, but $\frac{\partial (W_i x_1 + W_h h_0)}{\partial W_h} = h_0 = 0$, so

$$\frac{\partial h_1}{\partial W_h} = 0$$

Then,

$$\frac{\partial h_2}{\partial W_h} = \frac{\partial h_2}{\partial (W_i x_2 + W_h h_1)}\frac{\partial (W_i x_2 + W_h h_1)}{\partial W_h}$$

$$= \frac{\partial h_2}{\partial (W_i x_2 + W_h h_1)}(\frac{h_1 \partial (W_h)}{\partial W_h} + \frac{W_h \partial (h_1)}{\partial W_h})$$

$$= \frac{1}{\cosh^2(W_i x_2 + W_h h_1)}(h_1 + 0)$$

$$= h_1(1-h_2^2)$$

And,

$$\frac{\partial L(y,\hat{y})}{\partial h_2} = \frac{\partial L(y,\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial h_2} = (-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}})W_o\hat{y}(1-\hat{y}) = W_o(-y(1-\hat{y}) + (1-y)(\hat{y})) = W_o(\hat{y}-y)$$

So,

$$\frac{\partial L(y,\hat{y})}{\partial W_h} = \frac{\partial L(y,\hat{y})}{\partial h_2}\frac{\partial h_2}{\partial W_h} = W_o(\hat{y}-y)h_1(1-h_2^2)$$

Then, note that

$$\frac{\partial h_1}{\partial W_i} = \frac{\partial h_1}{\partial(W_i x_1 + W_h h_0)}\frac{\partial(W_i x_1 + W_h h_0)}{\partial W_i} = \frac{x_1}{\cosh^2(W_i x_1 + W_h h_0)} = x_1(1-h_1^2)$$

Then,

$$\frac{\partial h_2}{\partial W_i} = \frac{\partial h_2}{\partial(W_i x_2 + W_h h_1)}\frac{\partial(W_i x_2 + W_h h_1)}{\partial W_i}$$

$$= \frac{\partial h_2}{\partial(W_i x_2 + W_h h_1)}(\frac{x_2\partial(W_i)}{\partial W_i} + \frac{W_h\partial(h_1)}{\partial W_i})$$

$$= \frac{1}{\cosh^2(W_i x_2 + W_h h_1)}(x_2 + W_h x_1(1-h_1^2))$$

$$= (1-h_2^2)(x_2 + W_h x_1(1-h_1^2))$$

So,

$$\frac{\partial L(y,\hat{y})}{\partial W_i} = \frac{\partial L(y,\hat{y})}{\partial h_2}\frac{\partial h_2}{\partial W_i} = W_o(\hat{y}-y)(1-h_2^2)(x_2 + W_h x_1(1-h_1^2))$$

problem 3: Multiclass AdaBoost (1%)

Note that

$$g_{T+1}^k(x) = g_T^k(x) + \alpha_{T-1}f_{T-1}^k(x)$$

So, to update $g_T^k(x)$ to derive $g_{T+1}^k(x)$ by gradient descent, we must let $\alpha_{T-1}f_{T-1}^k(x)$ and $\frac{\partial L(g_T^1,\ldots,g_T^K)}{\partial g_T^k(x)}$ to be in the same direction, where

$$\frac{\partial L(g_T^1,\ldots,g_T^K)}{\partial g_T^k(x)} = \frac{\partial\sum_{i=1}^m \exp(\frac{1}{K-1}\{\hat{y}_i\neq k\}g_T^k(x_i) - \{\hat{y}_i = k\}g_T^k(x_i))}{\partial g_T^k(x_i)}$$

$$= \sum_{i=1}^m(\frac{\{y_i\neq k\}}{K-1} - \{y_i = k\})\exp(\frac{1}{K-1}\{\hat{y}_i\neq k\}g_T^k(x_i) - \{\hat{y}_i = k\}g_T^k(x_i))$$

So, we want to find $f_T^k(x)$ to maximize

$$\sum_{i=1}^m(\frac{\{y_i\neq k\}}{K-1} - \{y_i = k\})f_T^k(x_i)\exp(\frac{1}{K-1}\{\hat{y}_i\neq k\}g_{T-1}^k(x_i) - \{\hat{y}_i = k\}g_{T-1}^k(x_i))$$

$$= \sum_{i=1}^{m} (\frac{\{y_i \neq k\}}{K-1} - \{y_i = k\}) f_T^k(x_i) \exp{(\frac{1}{K-1} \{\hat{y_i} \neq k\} \sum_{t=1}^{T-2} (\alpha_t f_t^k(x_i)) - \{\hat{y_i} = k\} \sum_{t=1}^{T-2} (\alpha_t f_t^k(x_i)))}$$

$$= \sum_{i=1}^{m} (\frac{\{y_i \neq k\}}{K-1} - \{y_i = k\}) f_T^k(x_i) \prod_{t=1}^{T-2} \exp{((\frac{\{\hat{y_i} \neq k\}}{K-1} - \{\hat{y_i} = k\})(\alpha_t f_t^k(x_i)))}$$

Then, we want to find $g_{T+1}^k(x) = g_T^k(x) + \alpha_T f_T^k(x)$ to minimize $L(g_T^1, ..., g_T^K)$, that is, we want to find α_T to minimize:

$$L(g_{T+1}^1, ..., g_{T+1}^K) = \sum_{i=1}^{m} \exp{(\frac{1}{K-1} \sum_{k \neq y_i} g_{T+1}^k(x_i) - g_{T+1}^{\hat{y_i}}(x_i))}$$

$$= \sum_{i=1}^{m} \exp{(\frac{1}{K-1} \sum_{k \neq y_i} (g_T^k(x_i) + \alpha_T f_T^k(x_i)) - (g_T^{\hat{y_i}}(x_i) + \alpha_T f_T^{\hat{y_i}}(x_i)))}$$

$$= \sum_{i=1}^{m} \exp{(\frac{1}{K-1} \sum_{k \neq y_i} (g_T^k(x_i)) - g_T^{\hat{y_i}}(x_i))} \exp{(\frac{1}{K-1} \sum_{k \neq y_i} \alpha_T f_T^k(x_i) - \alpha_T f_T^{\hat{y_i}}(x_i))}$$

Then, for convenience, we let $r_{x_i} = \frac{1}{K-1} \sum_{k \neq y_i} (g_T^k(x_i)) - g_T^{\hat{y_i}}(x_i)$, then,

$$\frac{\partial L(g_{T+1}^1, ..., g_{T+1}^K)}{\partial \alpha_T} = \frac{\partial \sum_{\hat{y_i} \neq f_T(x)} (\exp{(r_{x_i})} \exp{(\frac{\alpha_T}{K-1})}) + \sum_{\hat{y_i} = f_T(x)} (\exp{(r_{x_i})} \exp{(-\alpha_T)})}{\partial \alpha_T}$$

$$\frac{1}{K-1} \sum_{\hat{y_i} \neq f_T(x)} (\exp{(r_{x_i})} \exp{(\frac{\alpha_T}{K-1})}) - \sum_{\hat{y_i} = f_T(x)} (\exp{(r_{x_i})} \exp{(-\alpha_T)})$$

Note that $\frac{\partial L(g_{T+1}^1, ..., g_{T+1}^K)}{\partial \alpha_T} = 0$, so we can derive the conclusion:

$$\frac{\exp{(\frac{\alpha_T}{K-1})}}{\exp{(-\alpha_T)}} = \frac{\sum_{\hat{y_i} = f_T(x)} \exp{(r_{x_i})}}{\frac{1}{K-1} \sum_{\hat{y_i} \neq f_T(x)} \exp{(r_{x_i})}}$$

$$\exp{(\alpha_T (\frac{K}{K-1}))} = \frac{\sum_{\hat{y_i} = f_T(x)} \exp{(r_{x_i})}}{\frac{1}{K-1} \sum_{\hat{y_i} \neq f_T(x)} \exp{(r_{x_i})}}$$

$$\alpha_T = (\frac{K-1}{K}) \log{(\frac{\sum_{\hat{y_i} = f_T(x)} \exp{(r_{x_i})}}{\frac{1}{K-1} \sum_{\hat{y_i} \neq f_T(x)} \exp{(r_{x_i})}})}$$

problem 4: Expectation-Maximization for GMMs (1%)

1. First, we know that

$$\mathbb{P}[z_i = k|x_i;\theta^{(t)}] = \frac{\mathbb{P}[x_i, z_i = k;\theta^{(t)}]}{\mathbb{P}[x_i;\theta^{(t)}]} = \frac{\pi_k^{(t)} N(x_i;\mu_k^{(t)};\Sigma_k^{(t)})}{\sum_{j=1}^{K}\pi_j^{(t)} N(x_i;\mu_j^{(t)};\Sigma_j^{(t)})}$$

   Then, we let $\delta_{ik}^{(t)} = \mathbb{P}[z_i = k|x_i;\theta^{(t)}]$, then we can derive that

$$Q(\theta|\theta^{(t)}) = \sum_{Z} p(Z|X;\theta^{(t)}) \log(p(X,Z;\theta))$$

$$= \mathbb{E}_{Z|X;\theta^{(t)}} \log(\prod_{i=1}^{N} p(x_i, z_i;\theta))$$

$$= \sum_{i=1}^{N} \mathbb{E}_{Z|X;\theta^{(t)}} \log(p(x_i, z_i;\theta))$$

$$= \sum_{i=1}^{N}\sum_{k=1}^{K} \delta_{ik}^{(t)} \log(p(x_i, z_i = k;\theta))$$

$$= \sum_{i=1}^{N}\sum_{k=1}^{K} \delta_{ik}^{(t)}(-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k) + \log(\frac{\pi_k}{\sqrt{(2\pi)^m|\Sigma_k|}}))$$

2. First, we know that

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial \mu_k} = \frac{\partial Q(\theta|\theta^{(t)})}{\partial (x_i-\mu_k)}\frac{\partial (x_i-\mu_k)}{\partial \mu_k} = -\sum_{i=1}^{N}\delta_{ik}^{(t)}(-\frac{2\Sigma_k^{-1}(x_i-\mu_k)}{2}) = \Sigma_k^{-1}\sum_{i=1}^{N}\delta_{ik}^{(t)}(x_i-\mu_k)$$

   We have to let $\mu_k^{(t+1)} = \mu_k$ such that $\frac{\partial Q(\theta|\theta^{(t)})}{\partial \mu_k} = 0$, so

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N}\delta_{ik}^{(t)} x_i}{\sum_{i=1}^{N}\delta_{ik}^{(t)}}$$

   Then, if we let $\Sigma_k^{-1} = \begin{bmatrix} a_{11}^k & \cdots & a_{1m}^k \\ \vdots & \ddots & \vdots \\ a_{m1}^k & \cdots & a_{mm}^k \end{bmatrix}$, then from the math problems in HW2, we can know

$$\frac{\partial Q(\theta|\theta^{(t)})}{\partial a_{ij}^k} = \frac{\partial \sum_{n=1}^{N} \frac{\delta_{nk}^{(t)}}{2}(\log(|\Sigma_k^{-1}|) - \text{Trace}(\Sigma_k^{-1}(x_n-\mu_k)(x_n-\mu_k)^T))}{\partial a_{ij}^k}$$

$$= \frac{1}{2}\sum_{n=1}^{N} \delta_{nk}^{(t)}(e_j^T\Sigma_k e_i - e_j^T(x_n-\mu_k)(x_n-\mu_k)^T e_i)$$

$$= \frac{1}{2}\sum_{n=1}^{N} \delta_{nk}^{(t)} e_j^T(\Sigma_k - (x_n-\mu_k)(x_n-\mu_k)^T)e_i$$

We have to let $\Sigma_k^{(t+1)} = \Sigma_k$ such that $\frac{\partial Q(\theta|\theta^{(t)})}{\partial \Sigma_k} = 0$, so

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \delta_{ik}^{(t)}(x_i-\mu_k^{(t+1)})(x_i-\mu_k^{(t+1)})^T}{\sum_{i=1}^{N}\delta_{ik}^{(t)}}$$

Finally, consider the constraint $\sum_{k=1}^{K}\pi_k = 1$, so given λ, we have

$$\frac{\partial Q(\theta|\theta^{(t)}) - \lambda\sum_{p=1}^{K}\pi_p}{\partial \pi_k} = (\sum_{i=1}^{N}\frac{\delta_{ik}^{(t)}}{\pi_k}) - \lambda$$

So, to let $\pi_k^{(t+1)} = \pi_k$ such that $\frac{\partial Q(\theta|\theta^{(t)}) - \lambda\sum_{p=1}^{K}\pi_p}{\partial \pi_k} = 0$, the ratio between all $\pi_k^{(t+1)}$ for $k \in \{1, 2, ..., K\}$ must be as below:

$$\pi_1^{(t+1)} : \pi_2^{(t+1)} : ... : \pi_k^{(t+1)} = \sum_{i=1}^{N}\delta_{i1}^{(t)} : \sum_{i=1}^{N}\delta_{i2}^{(t)} : ... : \sum_{i=1}^{N}\delta_{iK}^{(t)}$$

And, due to the constraint that $\sum_{k=1}^{K}\pi_k = 1$, we can derive the conclusion:

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^{N}\delta_{ik}^{(t)}}{\sum_{i=1}^{N}\sum_{p=1}^{K}\delta_{ip}^{(t)}} = \frac{\sum_{i=1}^{N}\delta_{ik}^{(t)}}{\sum_{i=1}^{N}\sum_{p=1}^{K}\mathbb{P}[z_i = p|x_i;\theta^{(t)}]} = \frac{\sum_{i=1}^{N}\delta_{ik}^{(t)}}{\sum_{i=1}^{N}1} = \frac{1}{N}\sum_{i=1}^{N}\delta_{ik}^{(t)}$$