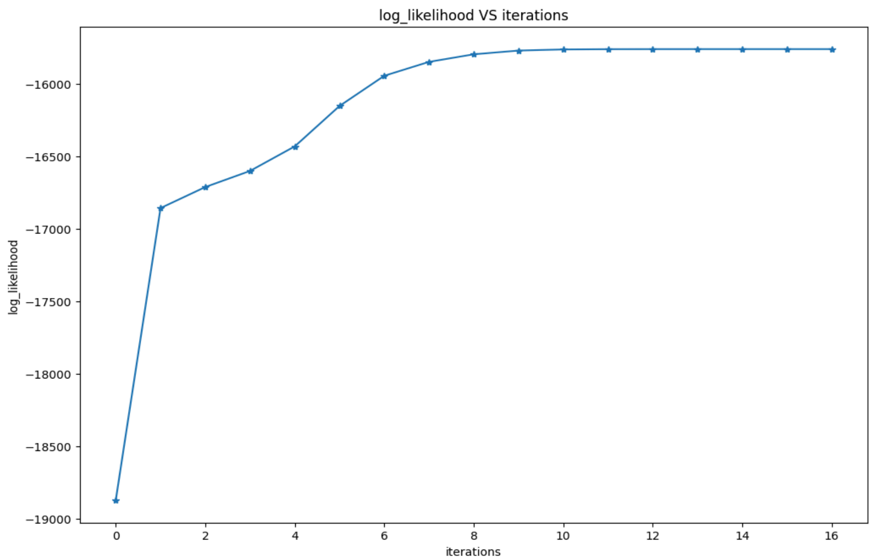
Homework 3

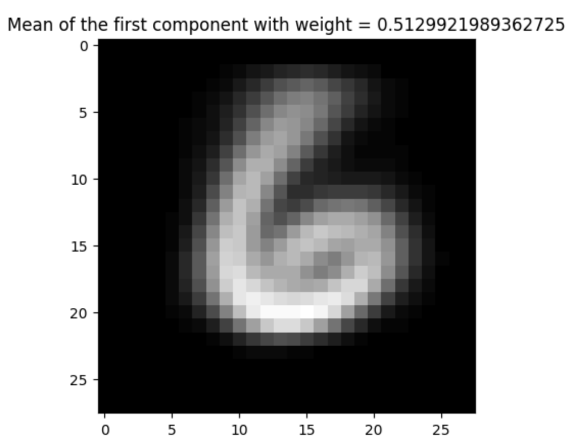
Xiaofan Jiao

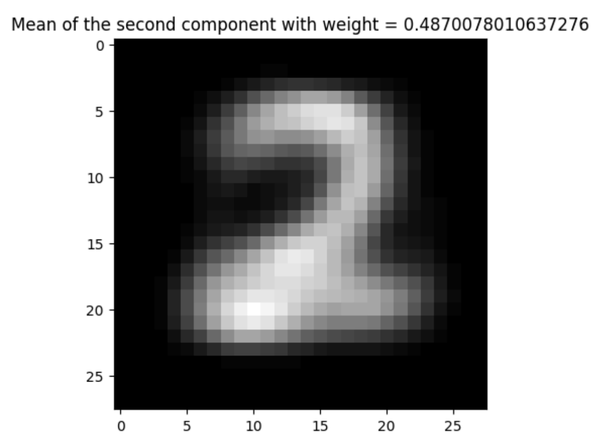
Question 1. Implementing EM for MNIST dataset.

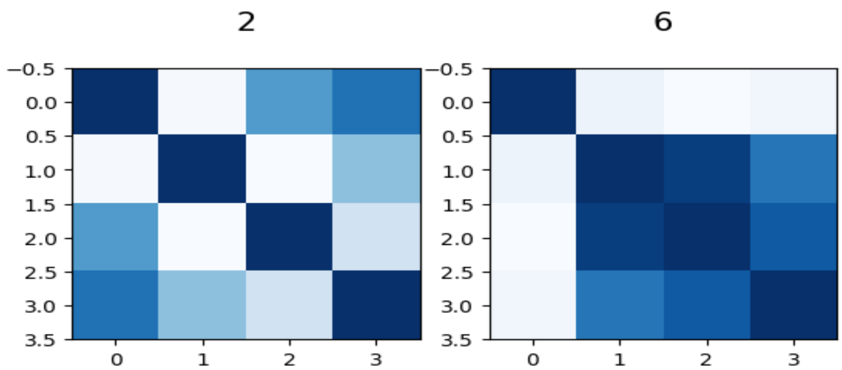
1). The log-likelihood increases with each iteration, indicating that the EM algorithm is successfully improving the model parameters to better fit the data.



2). After performing the GMM algorithm, the weights for 6’ and 2’ were 0.52 and 0.49 respectively.







3). GMM achieves better performance overall.

- For digit '2', both GMM and K-means have relatively similar misclassification rates, with GMM being slightly worse (6.4922%) compared to K-means (6.1047%).
- For digit '6', GMM performs significantly better with a misclassification rate of 0.9395%, whereas K-means has a higher misclassification rate of 7.9332%.
- The misclassification rates are as follows:

	'2'	'6'
GMM	6.4922%	0.9395%
K-means	6.1047%	7.9332%

Question 2. Optimization

1).

HW 3. Question 2. Optimization

1. $\ell(\theta) = \sum_{i=1}^{m} \left[y^i \theta^T x^i - \log\left(1 + \exp(\theta^T x^i)\right) \right]$

① differentiate $\ell(\theta)$ with respect to θ

$$\frac{d\ell(\theta)}{d\theta} = \sum_{i=1}^{m} \left[y^i x^i - \frac{\exp(\theta^T x^i)}{1 + \exp(\theta^T x^i)} x^i \right]$$

② Simplify term using $\sigma(z) = \frac{1}{1 + \exp(-z)}$

$$\frac{\exp(\theta^T x^i)}{1 + \exp(\theta^T x^i)} = \sigma(\theta^T x^i)$$

③ Thus, the gradient becomes:

$$\frac{d\ell(\theta)}{d\theta} = \sum_{i=1}^{m} \left[(y^i - \sigma(\theta^T x^i)) x^i \right]$$

2).

2. ① Initialize:

Set the initial value of θ, define learning rate. $\theta = \theta_0$

② Iterate:

Until convergence (the change in θ is smaller than a threshold or a max number of iterations)

Compute the gradient of the cost function with respect to θ

Update θ by taking a step in the direction of the gradient.

Initialize $\theta = \theta_0$

Set learning rate α

Repeat until convergence

{ gradient = 0

for $i = 1$ to m { gradient = gradient + $(y^i - \sigma(\theta^T x^i))x^i$ }

$\theta = \theta + \alpha \times$ gradient }

3).

3. The Stochastic Gradient Desent (SGD) updates the parameters for each training example instead of the whole dataset.

① Initialize:

Initialize $\theta = \theta_0$

Set learning rate α

② Repeat until convergence

- For each training example $i = 1$ to m
  - Compute the gradient

    $\text{gradient} = (y^i - \sigma(\theta^T x^i))x^i$
  - Update the parameters:

    $\theta = \theta + \alpha \cdot \text{gradient}$

- Gradient Desent uses the entire dataset to compute the gradient, leading to more stable covergence but can be computationally expensive. For larger datasets, it will require more computing power.
- Stochastic Gradient Desent updates the gradient for each training example, making it faster and able to handle large datasets, but the path to convergence may be noiser. When we have large dataset, SGD is often more computationally efficient when compared with GD.

4).

4. From previous questions we know that

Log-like Function:

$$l(\theta) = \sum_{i=1}^{m} \left[y^i \theta^T x^i - \log(1 + \exp(\theta^T x^i)) \right]$$

Gradient of $l(\theta)$:

$$\frac{\partial l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left[(y^i - \sigma(\theta^T x^i)) x^i \right]$$

where $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$

Hessian Matrix: second-order derivative of $l(\theta)$

$$H(\theta) = \frac{\partial^2 l(\theta)}{\partial \theta^2}$$

First Layer derivative

$$\frac{\partial l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left(y^i - \frac{1}{1 + \exp(\theta^T x^i)} \right) x^i$$

$$H(\theta) \quad \frac{\partial^2 l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left(\frac{x_i^T \exp(\theta^T x^i)}{(1 + \exp(\theta^T x^i))^2} \right) x^i$$

$$\frac{\partial^2 l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left(\frac{\exp(\theta^T x^i)}{(1 + \exp(\theta^T x^i))^2} \right)$$

$$\frac{\partial^2 l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \frac{1}{(1 + \exp(\theta^T x^i))} \times \frac{\exp(\theta^T x^i)}{(1 + \exp(\theta^T x^i))}$$

$$\frac{\partial^2 l(\theta)}{\partial \theta} = \sum_{i=1}^{m} \left(\frac{1}{(1 + \exp(\theta^T x^i))} \left(\frac{1}{1 + \exp(\theta^T x^i)} - 1 \right) \right)$$

Since $0 < \dfrac{1}{(1 + \exp(\theta^T x^i))} < 1$

Thus $\displaystyle\sum_{i=1}^{m} \left(\frac{1}{(1 + \exp(\theta^T x^i))} \left(\frac{1}{1 + \exp(\theta^T x^i)} - 1 \right) \right) < 0$

This ensures that $H(\theta)$ is negative semi-definite and $l(\theta)$ is concave. The concavity of $l(\theta)$ makes sure that there is a unique global maximum. This property is crucial because it means that optimization algorithms like gradient decent will converge to this unique global optimizer. Gradient descent is a reliable algorithm for finding the maximum of $l(\theta)$ because the concavity ensures no local maxima exist apart from the global maximum. The negative semi-definite Hessian means the function curves downwards, facilitating efficient optimization

Question 3. Bayes Classifier for spam filtering

1).

HW3 Question 3

1. • Total messages $m = 7$.
• No. of spam $n_{spam} = 3$.
• No. of non-spam $n_{non\text{-}spam} = 4$.

Thus $P(y=0) = \frac{n_{spam}}{m} = \frac{3}{7} \approx 0.429$

$P(y=1) = \frac{n_{non\text{-}spam}}{m} = \frac{4}{7} \approx 0.571$

2).

2. Given $m = 7$ log-likelihood function:

$$l(\theta) = \sum_{i=1}^{7} \sum_{k=1}^{15} x_k^{(i)} \log \theta_{y^{(i)},k}$$

We introduce Lagrangian multipliers.

$$\sum_{k=1}^{d} \theta_{c,k} = 1$$

$$\mathcal{L} = l(\theta) + \lambda_0 \left(1 - \sum_{k=1}^{15} \theta_{0,k}\right) + \lambda_1 \left(1 - \sum_{k=1}^{15} \theta_{1,k}\right)$$

① Compute Partial Derivatives & Set to 0

$\theta_{0,k}$: $\frac{\partial \mathcal{L}}{\partial \theta_{0,k}} = \sum_{i:y^{(i)}=0} \frac{x_k^{(i)}}{\theta_{0,k}} - \lambda_0 = 0$ $\quad \sum_{i:y^{(i)}=0} x_k^{(i)} = \lambda_0 \theta_0 k$

$\theta_{1,k}$: $\frac{\partial \mathcal{L}}{\partial \theta_{1,k}} = \sum_{i:y^{(i)}=1} \frac{x_k^{(i)}}{\theta_{1,k}} - \lambda_1 = 0$ $\quad \sum_{i:y^{(i)}=1} x_k^{(i)} = \lambda_1 \theta_1 k$

$\theta_0 k = \frac{\sum_{i:y^{(i)}=0} x_k^{(i)}}{\lambda_0}$ $\qquad \theta_1 k = \frac{\sum_{i:y^{(i)}=1} x_k^{(i)}}{\lambda_1}$

② Use constrains $=1$

λ_0: $\sum_{k=1}^{15} \frac{\sum_{i:y^{(i)}=0} x_k^{(i)}}{\lambda_0} = 1$ $\qquad \lambda_0 = \sum_{k=1}^{15} \sum_{i:y^{(i)}=0} x_k^{(i)}$

$\lambda_1 = \sum_{k=1}^{15} \sum_{i:y^{(i)}=1} x_k^{(i)}$

$\theta_{0,1} = 0.3$ $\qquad \theta_{1,1} \approx 0.059$

$\theta_{0,7} = 0.2$ $\qquad \theta_{1,15} \approx 0.059$

3).

3. From Prior we know that

$P(y=0) = \frac{3}{7}$ $\quad$ $P(y=1) = \frac{4}{7}$

Spam $\quad$ Non-Spam

today is secrect

$X_{test} = \{1,0,0,0,0,0,1,0,0,0,1,0,0,0,0\}$

$$\begin{bmatrix} \theta_{0,k} = 0.3 & \theta_{0,7} = 0.2 & \theta_{0,11} = 0.1 \\ \theta_{1,1} \approx 0.059 & \theta_{1,7} \approx 0.176 & \theta_{1,11} \approx 0.118 \end{bmatrix}$$

For $y=0$

$$P(X_{test} \mid y=0) \propto \theta_{0,1}^{x_1} \cdot \theta_{0,7}^{x_7} \cdot \theta_{0,11}^{x_{11}} = 0.3^1 \cdot 0.2^1 \cdot 0.1^1 = 0.006$$

For $y=1$

$$P(X_{test} \mid y=1) \propto \theta_{1,1}^{x_1} \cdot \theta_{1,7}^{x_7} \cdot \theta_{1,11}^{x_{11}} = 0.059^1 \cdot 0.176^1 \cdot 0.118^1 = 0.001226$$

$$P(X_{test}) = 0.006 \cdot \frac{3}{7} + 0.001226 \cdot \frac{4}{7} \approx 0.00257454$$

Posterior Probabilities:

For $y=0$

$$P(y=0 \mid X_{test}) = \frac{0.006 \times 0.429}{0.00257454} \approx \underline{0.999} > 0.272$$

For $y=1$

$$P(y=1 \mid X_{test}) = \frac{0.001226 \times 0.571}{0.00257454} \approx 0.272$$

$\therefore$ 'today is secret' is Classified as <u>spam</u> because $y=0 > y=1$

```
theta_0_1 (secret|spam) = 0.273
theta_0_7 (today|spam) = 0.182
theta_1_1 (secret|non-spam) = 0.059
theta_1_15 (pizza|non-spam) = 0.059
```

Question 4. Comparing classifiers: Divorce classification/prediction

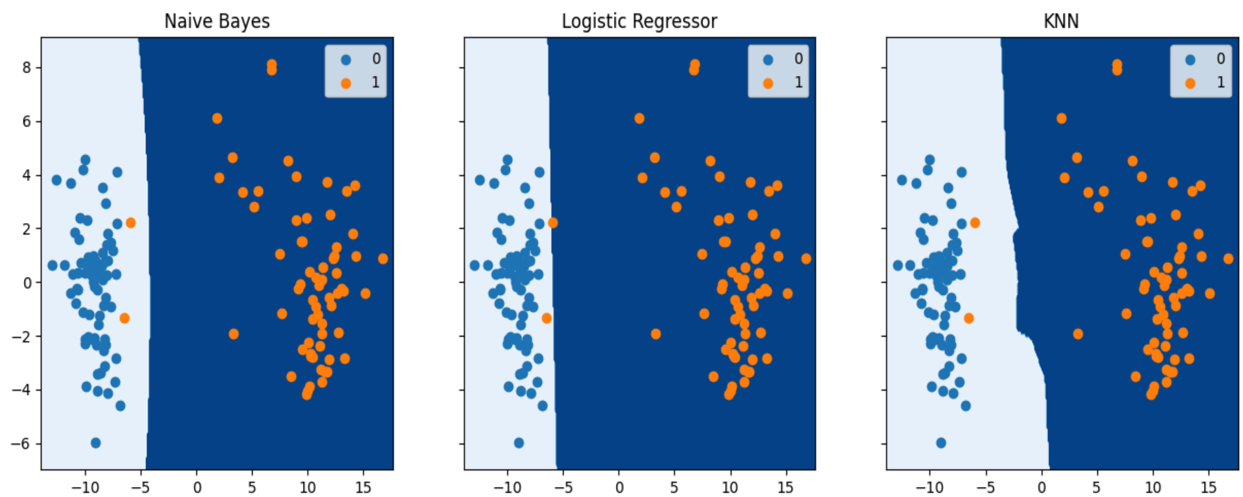
1).

	Model Name	Training Accuracy	Testing Accuracy
0	Naive Bayes	98.529412	94.117647
1	Logistic Regressor	100.000000	91.176471
2	KNN	98.529412	94.117647

The best classifiers are both KNN and Naive Bayes. Because of its ease of use and reliance on the concept of conditional independence, the Naive Bayes classifier is beneficial and appears to perform well on this dataset. By utilizing the local data structure, KNN also exhibits remarkably high performance. Even though logistic regression achieves flawless training accuracy, overfitting causes testing accuracy to decline. Consequently, Naive Bayes and KNN are the best models; their fair treatment of training and testing accuracy makes them appropriate for this specific classification job.

2).

```
Naive Bayes (PCA) training accuracy: 98.53%
Naive Bayes (PCA) testing accuracy: 94.12%
------------------------------------------------------------------------
Logistic Regressor (PCA) training accuracy: 99.26%
Logistic Regressor (PCA) testing accuracy: 100.00%
------------------------------------------------------------------------
KNN (PCA) training accuracy: 98.53%
KNN (PCA) testing accuracy: 94.12%
------------------------------------------------------------------------
```



For this dataset, the Logistic Regressor performs best; this is probably because the PCA modification made the space where logistic regression works best linearly separable. Naive Bayes and KNN also exhibit strong performance, when reduced to two dimensions using PCA. Logistic Regression shows perfect testing accuracy, making it the best model for this dataset.