ISYE 6740 Homework 4 Solution

Total 100 points

Ruyi Ding

1. **Implementing EM algorithm for MNIST dataset. (50 points)**

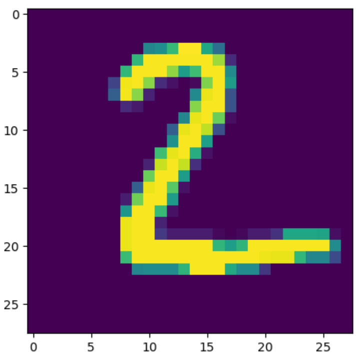
   Implement the EM algorithm for fitting a Gaussian mixture model for the MNIST dataset. We reduce the dataset to be only two cases, of digits "2" and "6" only. Thus, you will fit GMM with $C = 2$. Use the data file data.mat or data.dat on Canvas. True label of the data are also provided in label.mat and label.dat

   The matrix images is of size 784-by-1990, i.e., there are totally 1990 images, and each column of the matrix corresponds to one image of size 28-by-28 pixels (the image is vectorized; the original image can be recovered by map the vector into a matrix.)

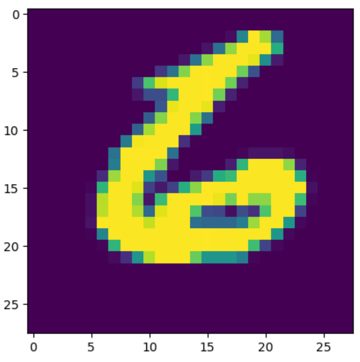
   Hint: You may find the notes speed-up-GMM.pdf useful, which explains how to evaluate the density of a multi-variate normal distribution. In this homework question, it is recommended you implement the evaluation of the Gaussian density this way, to avoid numerical issues.

   (a) (5 points) Select from data one raw image of "2" and "6" and visualize them, respectively.

   visualization of 2

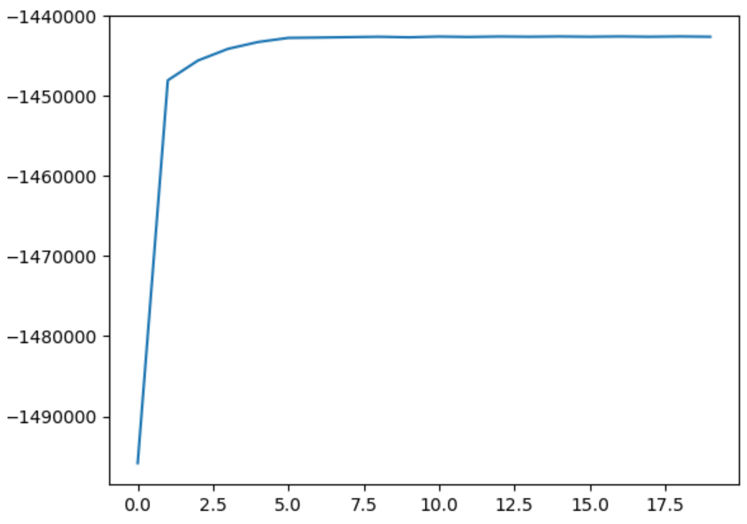
   

   visualization of 6

   clear visualize images of '2' and '6'. (4')
   numbers are not rotated. (1')

   (b) (15 points) Use random Gaussian vector with zero mean as random initial means, and two identity matrices I as initial covariance matrices for the clusters. Plot the log-likelihood function versus the number of iterations to show your algorithm is converging.

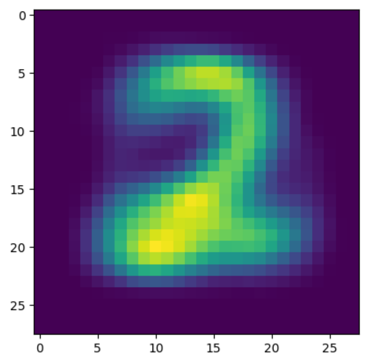


Actually we are plot **expection of loglikelihood** here, and this loglikelihood is estimated with the speed up algorithm.

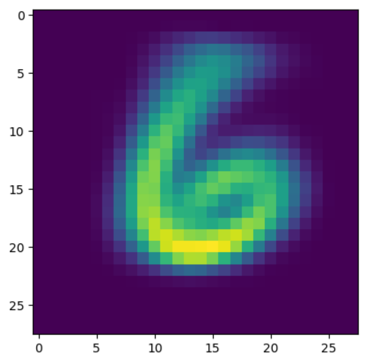
Try to write code of GMM. (5')
The plotted loglikelihood is converging. (5')
The convergence value of loglikelihood is close to -1440000 (5')

(c) (15 points points) Report, the fitting GMM model when EM has terminated in your algorithms, including the weights for each component and the mean vectors (please reformat the vectors into 28-by-28 images and show these images in your submission). Ideally, you should be able to see these means corresponds to "average" images. No need to report the covariance matrices.

average 2



average 6



weights 2	weights 6
0.524	0.476

Try to write code of GMM and evaluate the average image. (5')
The image is clear enough. (5')
The weights are in $[0.45, 0.55]$ (5')

(d) (15 points) Use the p_{ic} to infer the labels of the images, and compare with the true labels. Report the miss classification rate for digits "2" and "6" respectively. Perform K-means clustering with $K = 2$ (you may call a package or use the code from your previous homework). Find out the miss classification rate for digits "2" and "6" respectively, and compare with GMM. Which one achieves the better performance?

GMM has a better performance

Missing classification rate

Method	image 2	image 6
GMM	0.0126	0.0209
KMeans	0.0533	0.0710

State that GMM is better than KMeans (4')
Try to write code of GMM and evaluate the miss classification rate. (7')
The value of mcr (4')

Note: if student's answer is KMeans is better, but the code and values of MCR are reasonable, please only deduct 4 points for this question.

2. **Basic optimization.** (50 points.)

The background of logistic regression will be discussed in the next lecture. Here, we just focus on finding out the property of the optimization problem, related to training a logistic regression.

Consider a simplified logistic regression problem. Given n training samples (x_i, y_i), $i = 1, \ldots, n$. The data $x_i \in \mathbb{R}$, and $y_i \in \{0, 1\}$. To train/fit a logistic regression model for classification, we solve the following optimization problem, where $\theta \in \mathbb{R}$ is a parameter we aim to find:

$$\max_{\theta} \ell(\theta), \tag{1}$$

where the log-likelihood function

$$\ell(\theta) = \sum_{i=1}^{n} \{-\log(1 + \exp\{-\theta x_i\}) + (y_i - 1)\theta x_i\}.$$

(a) (15 points) Derive the gradient of the cost function $\ell(\theta)$ in (1) and write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does. pseudo-code means you will write down the steps of the algorithm, not necessarily any specific programming language.

The gradient here is essentially the derivative with respect to (wrt.) θ. Taking the derivative, we have:

$$\nabla l(\theta) = \sum_{i=1}^{n} \{\frac{exp(-\theta x_i) x_i}{1 + exp(-\theta x_i)} + x_i(y_i - 1)\} \tag{2}$$

Pseudo code:

Algorithm 1 GradientDescent

```
1: Initialize parameter $\theta^0$
2: while $t <$ number of iteration and $\epsilon >$ threshold do
3:     Calculate $\nabla\theta$ using (2)
4:     Update $\theta$ using $\theta^t = \theta^{t-1} - \eta\nabla\theta$
5:     Calculate $\epsilon = |\theta^{t-1} - \theta^t|$
6: end while
```

Note that, η is the learning rate, and $\eta > 0$.

Cost Function is correct (7')
The stop condition $\epsilon >$ threshold (3')
Basically step of Gradient Descent is correct (5')

Here $x_i \in \mathbb{R}^1$

(b) (15 points) Present a **stochastic gradient descent** algorithm to solve the training of logistic regression problem (1).

Similar to what we did in part (a), instead of using gradient descent, we use stochastic gradient descent (SGD) to improve the efficiency. Recall in the lecture, in SGD, we randomly sample from the data set without replacement and update θ with a subset of data. Therefore, with the same objective function (1), the gradient descent is performed on a subset of data we ramdomly sample from the full dataset. The gradient of θ becomes:

$$\nabla\theta = \sum_{i\in S_k}\{\frac{exp(-\theta x_i)x_i}{1+exp(-\theta x_i)} + x_i(y_i-1)\} \tag{3}$$

Eventually, we will loop through the entire training data, and start another round if the stopping criterion are not met. Below is a sample pseudo-code using 10% of the data in each iteration:

Algorithm 2 StochasticGradientDescent

```
1: Initialize parameter $\theta^0$ and the difference $\epsilon$
2: while $t <$ number of iteration and $\epsilon >$ threshold do
3:   Randomly divide the data $S$ into $\{S_1, S_2, \dots, S_n\}$
4:   for $S_k$ in $S$ do
5:     Calculate $\nabla\theta$ using (3)
6:     Update $\theta$ using $\theta^t = \theta^{t-1} - \eta\nabla\theta$
7:     Calculate $\epsilon = |\theta^{t-1} - \theta^t|$
8:     if $\epsilon <$ threshold then
9:       break
10:    end if
11:  end for
12: end while
```

Cost Function is correct (3')
Do Gradient Descent on each subset of data (4')
The algorithm will stop anytime when $\epsilon <$ threshold (4')
Other steps of SGD (4')

(c) (20 points) We will **show that the training problem in basic logistic regression problem is concave.** Derive the Hessian matrix of $\ell(\theta)$ and based on this, show the training problem (1) is concave. Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

As we are going to show the basic logistic regression problem, our data becomes $\boldsymbol{x} \in \mathbb{R}^d$, where d is the dimension of vector $\boldsymbol{x}$.

We know that for a given function $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^d$, the Hessian matrix can be derived by

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_d} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial^2 f(x)}{\partial x_d \partial x_1} & \frac{\partial^2 f(x)}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_d^2} \end{bmatrix}$$

To show the training problem is concave, we are going to show the Hessian Matrix of loglikelihood function is negative semidefinite.
The likelihood function of a basic logistic regression problem is:

$$l(\boldsymbol{\theta}) = log \prod_{i=1}^{n} P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - 1)\boldsymbol{\theta}^T \boldsymbol{x}_i - log(1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i))$$

Note: Here $\boldsymbol{x}_i$, $\boldsymbol{\theta}$ are d-dimensional vectors
Taking the first derivative :

$$\nabla l(\boldsymbol{\theta}) = \sum_{i=1}^{n} y_i \boldsymbol{x}_i - \boldsymbol{x_i} - (\frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)} \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i))) * \boldsymbol{x}_i = \sum_{i=1}^{n} (y_i - \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)})\boldsymbol{x}_i$$

Taking second derivative, we have:

$$\nabla^2 l(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)} (\frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)} - 1)\boldsymbol{x}_i \boldsymbol{x}_i^T$$

Write it into the format of matrix

$$H = \nabla^2 l(\boldsymbol{\theta}) = X^T D X$$

where $X \in \mathbb{R}^{n \times d}$, $D \in \mathbb{R}^{n \times n}$, and D is a diagnol matrix where $D_{ii} = \frac{1}{1+\exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)} (\frac{1}{1+\exp(-\boldsymbol{\theta}^T \boldsymbol{x}_i)} - 1)$
H is **negative-semidefinite**, which indicates the loglikelihood function is concave
Because the property of concavity, we know that $l(\boldsymbol{\theta})$ has no local maximum(only global optimal). As gradient descent always efficiently converges to the optimal (if the learning rate is reasonable), GD always reach the global maximum of logistic regression.

Correct Hessian matrix	(4')
Correct loglikelihood, where $\boldsymbol{x}$, $\boldsymbol{\theta}$ are vectors	(4')
Clear derive the second derivative of loglikelihood function	(4')
Clear declare the concavity comes from negative-semidefinite	(4')
Reasons for concavity to global maximum	(4')