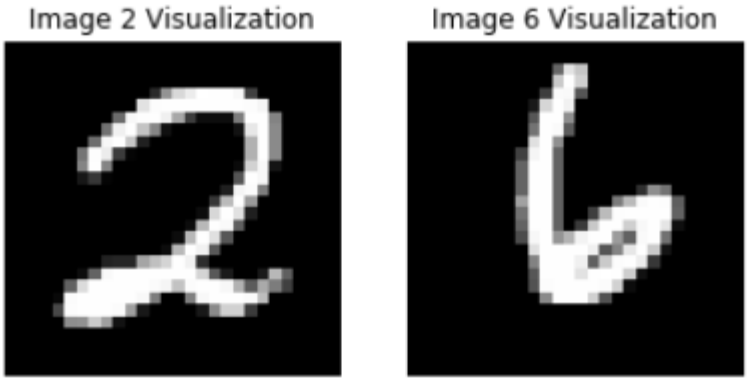
# 1) Implementing EM Algorithm for MNIST dataset (50 points)

Implement the EM algorithm formatting 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 _t GMM with C = 2. Use the data 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 _nd 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.

1a) (5 points) Select from data one raw image of \2" and \6" and visualize them, respectively.

## Solution 1(a)

As shown below, the raw image of 2 and 6 are displayed




---

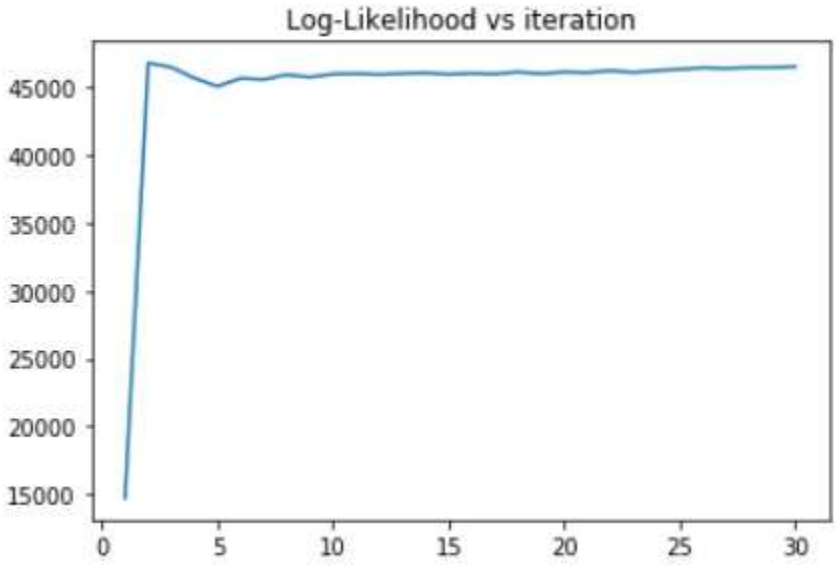
1b) (15 points) Use random Gaussian vector with zero mean as random initial means, and two identity matrices I as initial covariance matrices for the clusters. Plot the log-likelihood function versus the number of iterations to show your algorithm is converging.

## Solution 1(b)

GMM has been implemented using the below steps:

1. Clusters were initialized for the variables – pi, mu and sigma.
2. Random initial means and identity covariance matrix were used.
3. Expectation step was set up and implementation of speed up GMM to check Gaussian density on the data and mu values by taking the positive eigen values

4. Determinant was calculated.
5. Log-Likelihood was calculated.
6. Gmm was initialized and run –log for 30 iterations (tried different values) were calculated. Below is the plot:

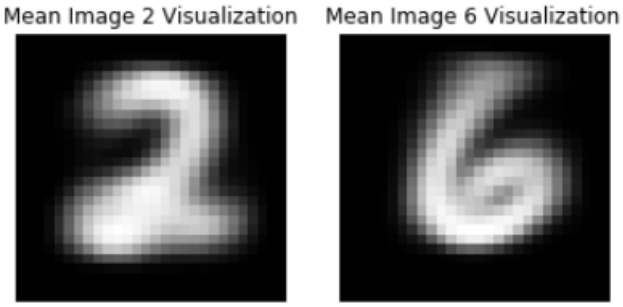



---

1c) (15 points points) Report, the _tting 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.

**Solution 1(c)**

Each cluster contains 2 and 6 separately. Reformatted the vectors into 28-by-28 images and used imshow to display the images.




---

(d) (15 points) Use the pic to infer the labels of the images, and compare with the true labels. Report the miss classi_cation 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 classi_cation rate for digits \2" and \6" respectively, and compare with GMM. Which one

achieves the better performance?

**Solution 1(d)**

Below table compares the miss classification percentage from K-Means vs GMM:

| Comparison | Digit 2 | Digit 6 |
|---|---|---|
| K-Means | 5.49 % | 7.17 % |
| GMM | 1.59 % | 1.08 % |

Clearly GMM performs better.

---

# 2) Basic optimization. (50 points.)

2a) (15 points) Derive the gradient of the cost function `(_) in (1) and write a pseudo-code for performing gradient descent to and the optimizer __. This is essentially what the training procedure does. pseudo-code means you will write down the procedure in steps for the algorithm, not necessarily any specific programming language.

**Solution 2(a)**

Cost Function is provided as below:

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

Now, to get the gradient of the cost function above we can take derivations with respect to $\theta$.

$$\frac{d}{d\theta}(y_i - 1)\theta x_i = (y_i - 1)x_i$$
$$\frac{d}{d\theta}\log(1 + \exp\{-\theta x_i\}) = \frac{x_i \exp\{-\theta x_i\}}{1 + \exp\{-\theta x_i\}}$$

So, the gradient of the cost function is as below:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{n} \left\{ (y_i - 1)x_i + \frac{x_i \exp\{-\theta x_i\}}{1 + \exp\{-\theta x_i\}} \right\}$$

Pseudo code for performing gradient descent:
Initialize parameter $\theta^0$
Do below:

$$\theta^{t+1} = \theta^t + \eta \sum_{i=1}^{n} \left\{ (y_i - 1) x_i + \frac{\exp(-\theta) x_i}{1 + \exp\{-\theta x_i\}} \right\}$$

While $\|\theta^{t+1} - \theta^t\| > \epsilon$

Where $\eta$ is the learning rate. After the last iteration the above algorithm gives the best values of $\theta$ for which $\max_{\theta} l(\theta)$.

---

2b) (15 points) Write down a stochastic gradient descent algorithm to solve the training of logistic regression problem (1).

## Solution 2(b)

Unlike the traditional algorithm which computes true gradient, in a stochastic gradient descent algorithm's iteration, the gradient is computed against a randomly sample small subset of the data.
The algorithm is as follows:
Initialize parameter $\theta^0$
Do below:

$$\theta^{t+1} = \theta^t + \eta \sum_{i \in S_k} \left\{ (y_i - 1) x_i + \frac{\exp(-\theta x_i) x_i}{1 + \exp(-\theta x_i)} \right\}$$

At each iteration, we randomly sample a small subset $S_k$ data point $(x_i, y_i), i \in S_k$ (in the extreme case, just one sample in $S_k$), and update using gradient estimated using a small subset of data.

While $\|\theta^{t+1} - \theta^t\| > \epsilon$

Using a different subset each time, so eventually loop through entire training data.
The data can be shuffled for each pass to prevent cycles.

---

(c) (20 points) Show that the training problem in basic logistic regression problem is concave. Derive the Hessian matrix of `(_) and based on this, show the training problem (1) is concave. Explain why the problem can be solved e_ciently and gradient descent will achieve a unique global optimizer, as we discussed in the lecture.

## Solution 2(c)

Since there is only one parameter $\theta$ in the cost function the shape of the Hessian matrix of the cost function will be 1 x 1.To derive the Hessian matrix let us take the second order derivatives of the cost function.

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

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

The cost function $\ell(\theta)$ is concave because all

$x_i^2,\ \exp(\theta x_i)$ and $(\exp(\theta x_i)+1)^2 > 0$ and the second order derivative of the cost function $\ell(\theta) < 0$

A local optimum is also a global optimum in case of a concave function.

Hence once the gradient descent converges to a local minimum optimal value, it can also be meant it reached a global minimum and this achieve a unique global optimizer.

---