Clustering MNIST with Bernoulli Mixtures

Jinghan Yang

Task:

The task is to cluster images in the MNIST dataset. We wish to find one cluster for each digit in an unsupervised manner. For the purpose of simplicity, we only consider instances with label 2, 3, 4. So our task is to find 3 clusters, which correspond to 2, 3, 4 respectively, without looking at training labels. Here we use the Bernoulli mixtures as the clustering algorithm. Bernoulli mixtures method assumes feature values are binary. The original data are represented as pixels ranging from 0 to 255. So we first normalize features to numbers between 0 and 1 and then further make them into binary values by thresholding at 0.5.

Model

Bernoulli mixture assumes that each data point is generated by a hidden cluster, and features are independent in that cluster.

Bernoulli mixtures have the following density form:

$$p(x|\boldsymbol{\mu},\pi) = \sum_{k=1}^{K} \pi_k p(x|\mu_k)$$

where

$$p(x|\mu_k) = \prod_{\boldsymbol{d=1}}^{\boldsymbol{D}} \mu_{\boldsymbol{kd}}^{\boldsymbol{x_d}}(1 - \mu_{\boldsymbol{kd}})^{(\boldsymbol{1-x_d})}$$

Bernoulli mixtures can be trained by EM with the following update equation:

E step:

$$r(z_{nk}) = \frac{\pi_k * p(x_n|\mu_k)}{\sum_{j=1}^{k} \pi_j p(x_n|\mu_j)}$$

M step:

$$N_k = \sum_{n=1}^{N} r(z_{nk})$$

$$\mu_k = \frac{\sum_{n=1}^{N} r(z_{nk}) * x_n}{N_k}$$

$$\pi_k = \frac{N_k}{N}$$

Where z_{nk} is a hidden indicator variable, such that $z_{nk} = 1$, if x_n is generated from k^{th} cluster.

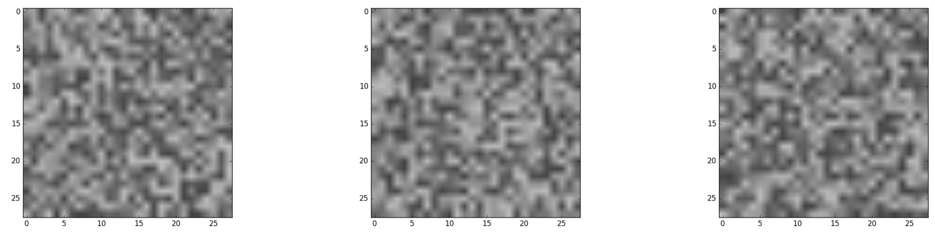
Derivation of update equations is in the appendix.

Experiments

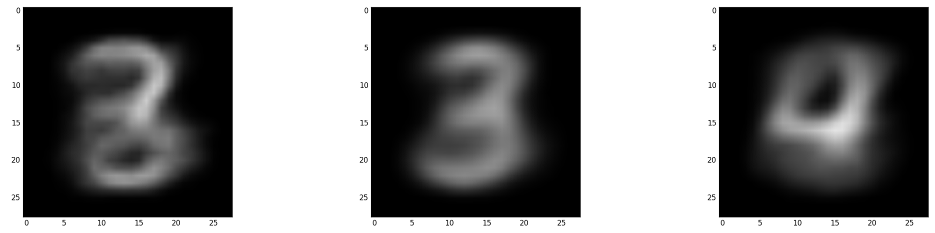
We train a Bernoulli mixture with EM on MNIST digits with labels 2,3,4. Each image is represented as a 784-bit vector and there are 20955 training images. We visualize our results by plotting the $\boldsymbol{\mu}$ of each cluster. Each $\boldsymbol{\mu}$ is a continuous valued vector. We treat these values as pixel values and draw a plot using them. each $\boldsymbol{\mu}$ can be interpreted as the centroid of the corresponding cluster. We show how these three centroids evolve from iteration to iteration. Parameters are initialized randomly.

1.

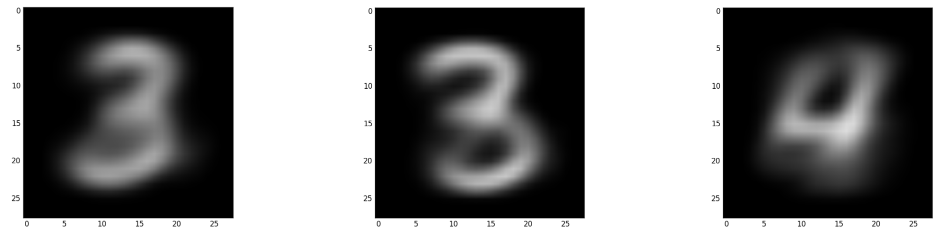
0 iteration (initialization)



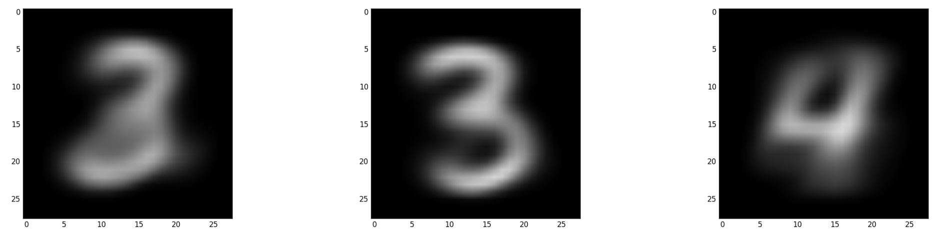
1 iteration



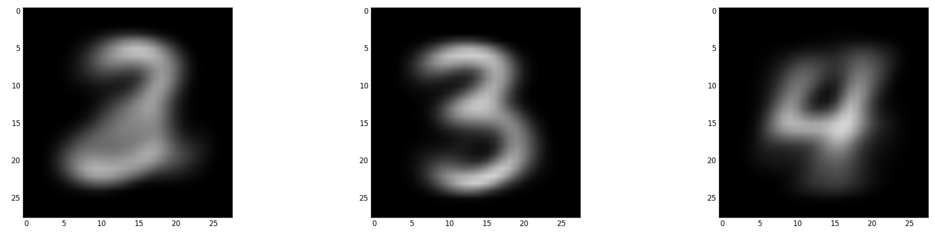
2 iterations



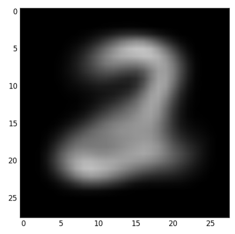
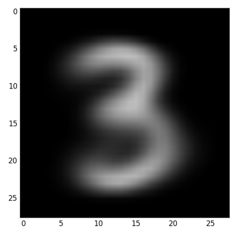
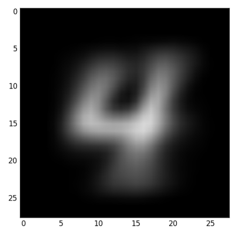
3 iterations



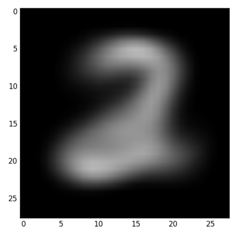
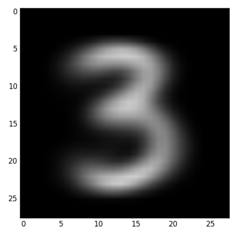
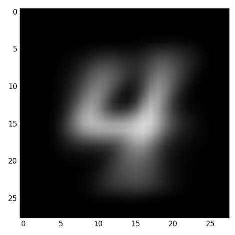
4 iterations



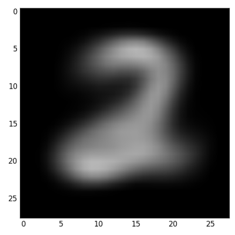
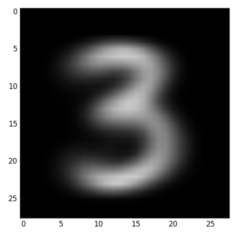
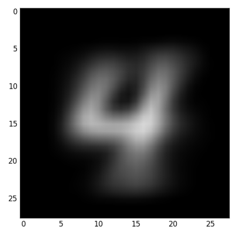
5 iterations

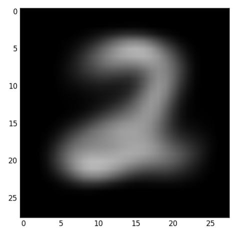
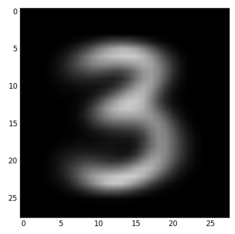
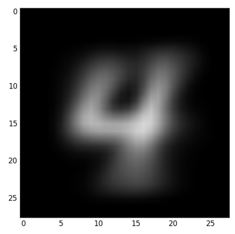
6 iterations

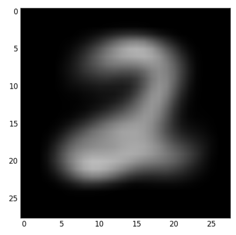
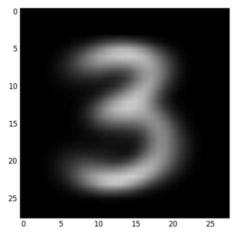
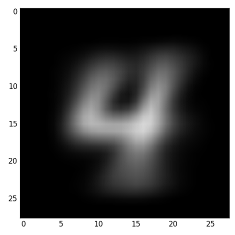
7 iterations

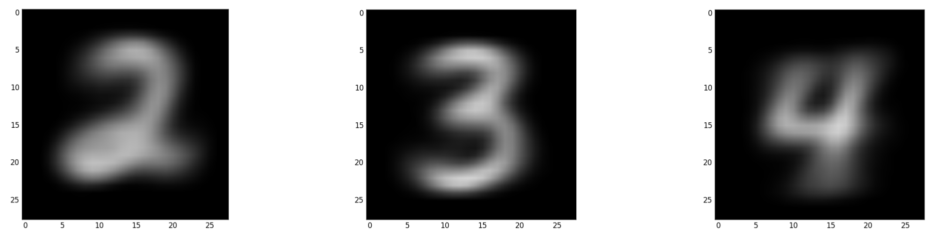
8 iterations

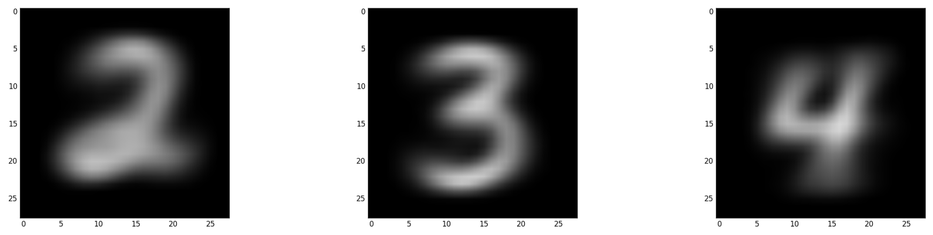
9 iterations

10 iterations



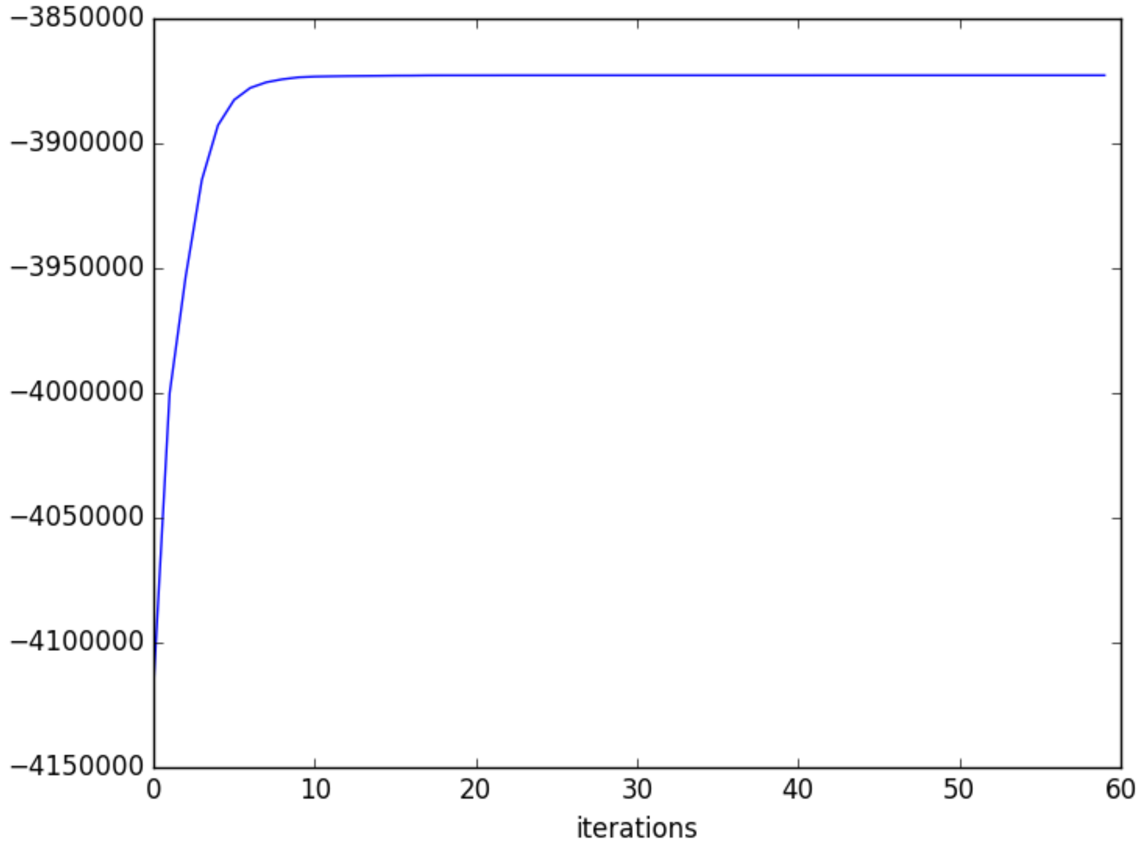
60 iterations



It is clear that Bernoulli mixture trained by EM correctly identifies the three clusters with weights [0.37284859 0.33855085 0.28860056] that correspond to three digits, 3, 4, 2.

Monitor objective function
To monitor the progress of EM, we evaluate the objective function of EM after each iteration. The actual objective function is
$$\ln(X|\boldsymbol{\mu}, \pi) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k p(x_n|\mu_k)\}$$

It is also common to evaluate the lower bound of objective function. But for this problem, evaluating the original objective is easier than evaluating the lower bound for numerical reasons.
The plot below shows the improvement of the original objective during training.



Generate sample digits from the learned model:
Since the learned Bernoulli mixture provides a density estimation for the given data, it is interesting to test whether we could sample digits from this density so that the sampled ones look like the given ones.

Firstly, we sample a cluster k based on π_1, π_2, π_3, and then we use corresponding μ_k to draw a digit. For each pixel independently, we sample the corresponding pixel value based on μ_{kd}.

```
    XX XXX
    XXXXXXX
   XXXX XX XX
   XXX XXXXXX
  X   X     XX
            XX
           XXX
            XX
            XXX
           XXXXX
           XXXX
           XXX
           XXX
  XX X   XXXX
 XXXXXXXXXXX
 XXXXXXXXXXX
XXX  XXXXXXXXXX
 XX XXXXXXXXXXX
XXXXXXXXX  XXXX
 XXXXX
```

```
             XX
      X  XX X     X
       XXXX X XX
     XXXX XX  XXX
      X X XXXXX   XX
     XX XX    X
   X     X    XX
             XXX
             X  X
         X   XX
             XX X
        XXXXX   X
     XXX XXXXX     X
  X X XX XXXXX X
  X X X  X  X
X XX XX  XX XXX  X
    XXXXXX X XX
   X XXXX X X  X    X
  XXX X XX X X    X
  X X  X   X X      X
   X
```

```
      XX XX
     XXXXXXXXX
     XXXXXXXXXX
     XXX   XXXX
             XX
            XXX
         XXXXX
       XXXXXXX
       XXXXXXXX
       XXXXXXXXX
       XXXXX   XX
       X       XX
               XX
              XXX
XX           XXX
 XX        XXXXX
XXXXX X   X XX
XXXXXXXXXXXXX
  XXXXXXXXXX
     XXXXX
```

```
       X    X
       XX XX X
 X XXXXXXXXX    X
   XX XXX X   X
   XX  XXX XX X
    X XX  X X
   X   X X XXX
        XX X  X
     XX XX  X
      X  X X
    XXXX XX X X
        XX X XX X
 X         X   X
      X      XXX X
       X   X   X
           XX X X
   XX X   XXX X
    X XX XX X
  X  XXXXXXXXXX
   XX XXXXX  X
          X X
X
```

```
    XX      X
    XX    XXX
   XX     XXX
  XXX    XXXX
  XXX     XX
 XXX    XXX
 XXX   XXXX
XXXXX XXXXX
 XXXXXXXXXXX
 XXXXXXXXXXX
  XXXXXXXX
     XXX
     XXX
   XXXXX
    XXX
   XXXX
  XXXXX
  XXXX
   XXX
  XXX
```

```
                X
 X X      XX
             XX  X
  XX   X      X
  XXX     XXX   X
   X     X XX
 XXX X        X
X XX X     XXXX
XX XXX  X
  XX X  XXXX X
 XXXXXX  XX XXX
 XXXX X XXXXX
   X X   XXXX
       XXXXX
      X
X         X
   X   X   XX
      X
   X  X
      X   X
```

We can compare the sampled numbers on the right side against the numbers from the training set on the left side. One can clearly see that the lack of smoothness of the sampled numbers. One possible reason is that, for this model, once we assign data points to a cluster, we assume that features (positions) are independent. Such assumption could result in some isolated points. In other words, this assumption reduces the smoothness of the graph.

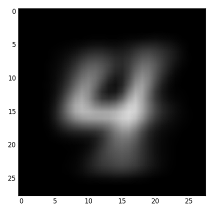
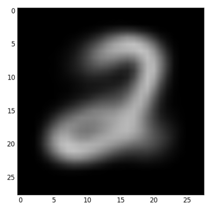
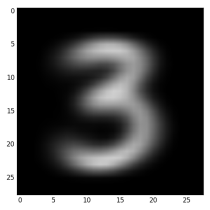
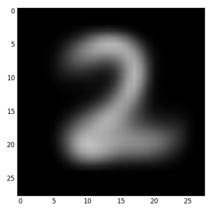
Classification

We could also use the trained Bernoulli mixture model for classification. The training is unsupervised. After training, we could get 3 centroids and weights corresponding to three digits. And we should look at the 3 centroids and know which cluster corresponds to which digit.

For a test data point x, we could compute $p(z = k|x) = \frac{p(z=k,x)}{p(x)}$, and pick whichever cluster k gives the max probability. Then for each x, we can predict the digit for cluster k. For example, I divided the dataset 60% for training and 40% for testing, and got 90.5% for accuracy. It is pretty satisfying given that we only label three images.

Clustering with four components

We can also try other k numbers. For example, below is result for $k = 4$, for 20 iterations, with weights of [0.34638679 0.17433678 0.29084584 0.18843059].

We can see that there are two writing types of 2. If we want to know more types of drawing, we could try different values for k.