CS195f Homework 6

Mark Johnson and Erik Sudderth

Homework due at 2pm, 24th November 2009

We begin by considering the problem of clustering simple image data. We will again use a subset of the MNIST handwritten digit database, which in addition to being a benchmark for testing classification algorithms, is often used to validate unsupervised learning algorithms. The particular version of the data which we'll use is available here:

`/course/cs195f/asgn/clustering/mnist_all.mat`

In addition, a Matlab script `hw6_kmeans.m` is posted in the same directory, which contains some code to help you get started.

In the MNIST database, each training or test example is a 28-by-28 grayscale image. To ease programming of learning algorithms, these images have been converted to vectors of length $28^2 = 784$ by sorting the pixels in raster scan (row-by-row) order. The Matlab `reshape` command can be used to convert these vectors back to images for visualization. For example, we can plot the third training example of class 1 as follows:

```
>> imagesc(reshape(train1(3,:), 28, 28)');
```

Question 1:

In this question, we use the K-means algorithm to cluster the handwritten digit data. For all sections, we use $1,000$ *examples of each of the 10 digit classes, so that there are* $N = 10,000$ *data items in total. Letting* μ_k *denote the mean for cluster* k*, the K-means objective function can be written as*

$$J(y, \mu) = \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} ||x_i - \mu_k||^2$$

where $y_{ik} = 1$ *if example* i *is assigned to cluster* k*, and 0 otherwise. Note that since we are testing a clustering algorithm, the true MNIST digit class labels will be used only to evaluate hypothesized clusterings,* not *as part of the clustering algorithm.*

a) Implement and submit a function which runs the K-means algorithm to convergence for any K*, from an initialization specified via a set of starting cluster centers* $\{\mu_k\}_{k=1}^{K}$*. Your function should record and return the value of the K-means objective function* $J(y, \mu)$ *at each iteration.*

b) Run the K-means algorithm on the digit data with $K = 10$*, the true number of clusters. Randomly initialize K-means by choosing* $K = 10$ *of the observations at random, and*

setting the initial cluster centers to be these observations. Plot the K-means objective as a function of iteration, and verify that it monotonically decreases.

Hint: *Use* `randperm.m` *to generate random initializations, as in the example code.*

c) *Repeat part (b) for 10 different random initializations, running the K-means algorithm to convergence from each. Evaluate the consistency of each resulting clustering with the true digit labels by computing the Rand index. Make a scatter plot of the Rand index values, versus the corresponding values of the K-means objective function $J(y, \mu)$. Does the K-means objective provide a good predictor of cluster quality?*

d) *When clustering algorithms do not perform perfectly, there are two major sources of error: the objective function or model may not match the data well, or the algorithm used to optimize that objective may be stuck in local optima. We can sometimes separate these issues by "cheating". Consider a K-means initialization in which the cluster centers μ_k are set to the means of the 10 digit classes, as determined via the* true *class labels. Run K-means to convergence from this initialization, and compute the resulting Rand index and objective function values. How do these compare to those from part (c)? What does this suggest about how we should try to build a better clustering method for this data?*

e) *We conclude by considering how K-means performs on this data as the number of clusters, K, is varied. For each $K \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$, run K-means to convergence from a single random initialization determined as in part (b). Plot both the K-means objective function and the Rand index (computed using the true cluster labels) versus K. Which value of K gives the lowest objective function? Which gives the largest Rand index? How could we determine a good value for K without using the true class labels?*

This next question asks you to construct an EM clustering algorithm for multinomial mixtures. Here is a brief recap of the model:

- Each sentence x consists of a sequence of words $x = (u_1, \ldots, u_\ell)$.
- The probability of a sentence x and a topic $y \in \mathcal{Y} = \{1, \ldots, m\}$ is:

$$\begin{aligned} \mathrm{P}_{\pi,\varphi}(X = x, Y = y) &= \pi_y \prod_{j=1}^{\ell} \varphi_{u_j|y} \\ &= \pi_y \prod_{u \in \mathcal{U}} \varphi_{u|y}^{c_u(x)}, \text{ where:} \\ \pi_y &= \text{probability of generating class } y \\ \varphi_{u|y} &= \text{probability of generating word } u \text{ in class } y \\ c_u(x) &= \text{number of times word } u \text{ appears in sentence } x \end{aligned}$$

- The log likelihood of data $D = (x_1, \ldots, x_n)$ is:

$$\log L_D(\pi, \varphi) = \sum_{i=1}^{n} \log \sum_{y \in \mathcal{Y}} \pi_y \prod_{u \in \mathcal{U}} \varphi_{u|y}^{c_u(x_i)}$$

- The expectation-maximization algorithm for multinomial mixtures is:

  Guess initial values $\pi^{(0)}$ and $\varphi^{(0)}$

  For iterations $t = 1, 2, 3, \ldots$ do:

  E-step: calculate expected values of sufficient statistics

$$\begin{aligned} \mathrm{E}[n_y] &= \sum_{i=1}^{n} \mathrm{P}_{\pi^{(t-1)},\varphi^{(t-1)}}(Y = y \mid X = x_i) \\ \mathrm{E}[n_{u,y}] &= \sum_{i=1}^{n} c_u(x_i)\, \mathrm{P}_{\pi^{(t-1)},\varphi^{(t-1)}}(Y = y \mid X = x_i) \end{aligned}$$

  M-step: update model based on sufficient statistics

$$\begin{aligned} \pi_y^{(t)} &= \frac{\mathrm{E}[n_y]}{n} \\ \varphi_{u|y}^{(t)} &= \frac{\mathrm{E}[n_{u,y}]}{\sum_{u' \in \mathcal{U}} \mathrm{E}[n_{u',y}]} \end{aligned}$$

You will perform EM clustering on the sentences in `motherese.txt`, which contains about 60,000 sentences spoken by a mother talking to her child. The first few lines of this file are:

```
big drum ?
horse .
who is that ?
```

To simplify programming, we've reformatted the data as follows. The file `motherese.words.txt` associates each word with a positive integer called its *wordnumber*, which we will use to identify the word. Here are a few lines from somewhere in the middle of that file.

```
2954, festival
2955, fever
2956, few
```

The file `motherese.counts.txt` contains lines of the form

sentencenumber, wordnumber, count

which says that word number *wordnumber* appeared in the sentence *sentencenumber* a total of *count* times. The first few lines of this file are:

```
1, 97, 1
1, 1706, 1
1, 2721, 1
2, 92, 1
2, 3551, 1
3, 97, 1
3, 3670, 1
3, 5994, 1
3, 6504, 1
```

You can view these counts as a big matrix with *sentencenumber* and *wordnumber* as its indices. In fact, the expectations you need to calculate can be formulated fairly cleanly as matrix operations on this matrix, and we encourage you to do this. One thing to note is that this matrix is very *sparse*, i.e., most of its entries are zero. So you should use Matlab's sparse matrices to store these counts; if you do this the calculations will actually be quite efficient (orders of magnitude faster than if you use ordinary dense matrices). Code which reads in these text files and constructs corresponding data structures is available at

`/course/cs195f/asgn/clustering/hw6_em.m`

along with the various data files.

For this problem you should use $K = 10$ clusters. As we mentioned in class, while you can initialize π uniformly (i.e., $\pi_y^{(0)} = 1/K$), you'll need to initialize φ in such a way that breaks symmetry i.e., arranges for some words to be more strongly associated with some clusters than others. One way to do this is to set $\varphi_{u|y}^{(0)} = 1 + \epsilon$, where ϵ is a random number between 0 and 10^{-3}, and then renormalize $\varphi_{u|y}$ appropriately. 50 EM iterations should be sufficient for your model to converge.

As we mentioned in class, EM can converge to a local maximum of the likelihood, rather than a global maximum. There are a number of more or less ad hoc methods that attempt to deal with this – one of the simplest is just to run EM many times from different random initial starting points, and keep the model with the highest log likelihood. For this homework assignment you should run EM 10 times with different initial parameters and keep the model (π, φ) with the highest log likelihood.

Finally, there's the problem of understanding what the clusters actually mean. You can compute the most likely cluster for each sentence and sort sentences by cluster, but there are close to 60,000 of them so this is hard to understand. One way around this is to find the words (rather than the sentences) that are most closely associated with each cluster. Using the model with the highest log likelihood, for each cluster $y \in \mathcal{Y}$ you should report:

- the twenty words $u \in \mathcal{U}$ with the highest values of $\varphi_{u|y}$ (these are the most likely words to appear in cluster y), and
- the twenty words $u \in \mathcal{U}$ with the highest value of $(\mathrm{E}[n_{u,y}]+2)/(\sum_{y' \in \mathcal{Y}} \mathrm{E}[n_{u,y'}+20)$ (this is a smoothed estimate of $\mathrm{P}(y|u)$, which is a measure of how strongly u is associated with cluster y).

Question 2:

Write a program to perform EM clustering for multinomial mixtures and run it on the sentences in `motherese.txt` *(using the other data files if that is easier).*

a) Attach a plot of log likelihood versus iteration for each of the 10 runs you performed.

b) For the model with the highest log likelihood that you found, report for each cluster y the 20 words most likely to appear in cluster y, and the 20 words most strongly associated with cluster y.