# Stats 545: Homework 3

Due before class on Tuesday, Oct 1.
All plots should have labelled axes and titles.

**Important:Rcode, tables and figures should be part of a single .pdf or .html files from `R Markdown` and `knitr`. See the class reading lists for a short tutorial.Any derivations can also be in Markdown, in Latex or neatly written on paper which you can give to me.**

## 1 Problem 1: The K-means algorithm. [55 pts]

The MNIST dataset is a dataset of $28 \times 28$ images of hand-written digits. Download it from http://yann.lecun.com/exdb/mnist/ (you only really need the training images and labels though). To read these images in `R`, use the script from https://gist.github.com/brendano/39760. Make sure you understand this. Note that the `show_digit` command displays a particular digit.

1. Since the dataset is quite large, restrict yourself to the first 1000 training images, and their labels. Store these as variables called `digits` and `labels`. `digits` should be a $1000 \times 784$ matrix (or its transpose). Include `R` code. [5 pts]

2. Write a function `my_kmeans` to perform a k-means clustering of the 1000 images of `digits`. Use Euclidean distance as your distance measure between images (which can be viewed as vectors in a 784 dimensional space). Your function should take 3 arguments, the matrix `digits`, the number of clusters $K$ and the number of initializations $N$. You code should consist of 3 nested loops. The outermost (from 1 to $N$) cycles over random cluster initializations (i.e. you will call k-means $N$ times with different initializations). The second loop (this could be a `for` or `while` loop) is the actual k-means algorithm for that initialization, and cycles over the iterations of k-means. Inside this are the actual iterations of k-means. Each iteration can have 2 successive loops from 1 to $K$: the first assigns observations to each cluster and the second recalculates the means of each cluster. These should not require further loops. (You will probably encounter empty clusters. It is possible to deal with these in clever ways, but here it is sufficient to assign empty clusters a random mean (just like you initialized them)). Since your initializations are random, make your results repeatable by using the `set.seed()` command at the beginning (you can also make the seed value a fourth argument). Your function should return:

   (a) the cluster parameters and cluster assignments for the best solution

   (b) the sequence of values of the loss-function over k-means iterations for the best solution (this should be non-increasing) (recall from the slides that the k-means loss function is the sum of the squared distances of observations from their assigned means)

   (c) The set of $N$ terminal loss-function values for all initializations.

   Do not hardcode the number of images or their size. Include `R` code. [25 pts]

3. Explain briefly what stopping criteria you used (i.e. the details of the second loop). [2 pts]

4. Run your code on the 1000 `digits` for $K = 5, 10, 20$. Set $N$ to a largish number e.g. 25 (if this takes too long, use a smaller number).For each setting of $K$, plot the cluster means (using `show_digit`) as well as the evolution of the loss-function for the best solution (you can use a semi-log plot if that is clearer).

You do not have to print the other values returned by the function e.g. the cluster assignments, or the values of the cluster means etc., just plots is sufficient [20 pts]

5. For each setting of $K$, plot the distribution of terminal loss function values (using `ggplot`'s `geom_density()`).

6. Explain briefly how you might choose the number of clusters $K$. [3 pts]

# 2 Problem 2: Finite-state Hidden Markov models (HMMs) [45pts]

(Continued from the problem on Markov chains from the previous homework.)

Suppose now that we do not observe the state $S_t$ of the Markov chain. Instead, at time $t$ we observe $Y_t$. $Y_t$ can be anything: integers, reals, vectors, images. The only condition is that the probability distribution of $Y_t$ depends only on $S_t$ (and not e.g. on $S_{t-1}$). Write this as $P_Y(Y_t|S_t)$, with $P_Y(\cdot|S_t = i)$ giving the probability (or probability density) of $Y_t$ given $S_t = i$ (for simplicity, we let this same probability hold for all $t$). Also write $\mathbf{Y} = (Y_1, \dots Y_T)$.

Our HMM model defines a probability distribution over $(\mathbf{S}, \mathbf{Y})$.

1. Write down $P(S_1 = s_1, S_2 = s_2, \dots, S_T = s_T, Y_1 = y_1, \dots, Y_T = y_t)$ in terms of $\pi^1$, $A$ and $P_Y$. [3pts]

For the earlier 'Markov chain' problem, we could quite efficiently calculate $\pi_i^t = P(S_t = i)$, the marginal prior probability of $S_t$. We will now calculate the marginal posterior probabilities $P(S_t = i|\mathbf{Y})$. Look at the scanned notes for the Kalman filter for reference.

From now onwards, we will fix the values of $\mathbf{Y}$, since these are our observations. Then define an $N \times 1$ vector $B^t$, with $B_i^t = P(Y_t = y_t|S_t = i)$.

1. Is this a *probability* vector (i.e. is it a nonnegative, adding up to 1)? [1pts]

Define $\alpha$ and $\beta$ messages:

$$\alpha_i^t := P(S_t = i, Y_1 = y_1, \dots, Y_t = y_t) = P(S_t = i, \mathbf{Y}_{1:t})$$
$$\beta_i^t := P(Y_{t+1} = y_{t+1}, \dots, Y_T = y_T|S_t = i) = P(\mathbf{Y}_{t+1:T}|S_t = i)$$

Above := means 'which we define to be equal to', or 'which we will call'.

2. $\alpha^t$ and $\beta^t$ are both $N \times 1$ vectors. Are these vectors *probability* vectors? [1pts]

3. Write $P(S_t = i|\mathbf{Y})$ in terms of $\alpha^t$ and $\beta^t$. Include the normalization term (i.e summation of both sides over $i$ must give 1). In the scanned notes (e.g. top paragraph of page 2), we ignore the normalization constant, but we can easily calculate it since we know that probabilities must sum or integrate to 1. Use matrix (or vector) notation. You might have to use a transpose (e.g. $(\alpha^t)^\top$). Also, define $\mathbf{1}$ as an $N \times 1$ vector of ones, and note that $\sum_{i=1}^N a_i^t = \mathbf{1}^\top a^t$. Hint: First write it explicitly with summations (compare with the Kalman filter). [5pts]

4. Write $P(S_t = i, S_{t+1} = j|\mathbf{Y})$ in terms of $\alpha$, $\beta^t$, $A$ and $B$. Include the normalization term and use matrix (or vector) notation. [5pts]

5. Write $\alpha^t$ as a function of $\alpha^{t-1}$, $A$ and $B$. Use matrix notation. An operation you'll need here is the element-wise product of two vectors. In R this is easy, just write $V_1 * V_2$ for two vectors $V_1$ and $V_2$. In matrix notation, the simplest way to write this is as $\text{diag}(V_2) \cdot V_1$, where $\text{diag}(V_2)$ is an $N \times N$ matrix whose diagonal is $V_2$ and whose other elements are 0. Verify that $\text{diag}(V_2) \cdot V_1$ is a vector whose $i$th element is the product of the $i$th elements of $V_1$ and $V_2$. [10pts]

6. Write $\beta^t$ as a function of $\beta^{t+1}$, $A$ and $B$. Use matrix notation. [10pts]

7. How will you calculate the first $\alpha$ and $\beta$ at the beginning of the forward and backward pass?

8. Hopefully, you can now see a dynamic programming algorithm that sequentially calculates the $\alpha^t$'s, and then the $\beta^t$'s. These can then be combined to calculate $P(S_t|\mathbf{Y})$ for any $t$. The overall algorithm is called the Baum-Welch algorithm. Write down the cost in terms of $T$ and $N$. [5pts]

9. Imagine instead we wanted to calculate the most like sequence of states (rather than the marginal probabilities). One approach is to set $S_t = \text{argmax}P(S_t|\mathbf{Y})$ for all $t$. Why is this a bad idea? [5pts]