1. *Convolution:* This problem will consider the training and test data for the handwritten 0's and 9's from the famous MINST dataset. The `minst0_test.dat` and `minst0_train.dat` files (and associated files for the 9's are given on the Blackboard class site. These images all contain 28 x 28 pixels.

   (a) Consider the first image in the "9" training data. Use a convolution function and an appropriate 3 x 3 kernel to (a) "shift the image to the left", (b) "shift the image to the right", (c) "shift the image up", (d) "shift the image down", and (e) at least two different kernels to find edges. Show your kernel and your convolved image. Your convolutions should preserve the image size.

   (b) Experiment with the kernel dimension and report what you find.

   (c) Write a pooling function that can at least do max pooling and average pooling. Pool the image from the first part by finding the average in 4×4 blocks. Show the new 7×7 images. Do the same, but find the max in 7×7 blocks and show your pooled 4×4 image.

   (d) Use a Laplacian kernel to convolve the image, apply the "ReLU" function, and then follow this by max pooling to produce a 4×4 image.

   (e) *Building a Toy Convolution Neural Net Classifier:* You will build and train a VERY simple convolutional neural network to classify the 0 and 9 images. Follow the procedure outlined below:

      i. Construct the training data from at least the first 100 images from each of the training files (100 "9"s and 100 "0"s) (if your program is more efficient, use more than 100 images! Note, you should make a separate label file and image file.

      ii. Your convolutional neural network should include the following:

      - 5 trainable 3×3 kernels to convolve the image
      - application of the "ReLU" function
      - Max pooling each of the 5 transformed convolved images to a 2×2 image
      - Use the 4×5 values from the previous stage as inputs into a 10 hidden unit binary classification neural network
      - Train the kernel and α, β parameters in the neural network; **NOTE: this network should have 266 parameters (45 kernel parameters, 210 α parameters, and 11 β parameters); this is small enough that you could use a good non-gradient optimization algorithm if you don't want to program up the back-propagation; either way is ok, just make sure you say how you did it. The advantage of the non-gradient optimization is that you just need to form a generative model to consider in the optimization function. I used PSO to fit it because it was simple (and seemed to work).**

   (f) Show the within sample classification results and then apply the fitted model to the test data sets and report your classification results. Describe the sensitivity of your classification results to replication (choices of starting values, and, depending on your optimization, your optimization parameters). [If you want, you can compare this to gradient boosting or random forests (on the original images or on the convolved images]. Include your R code in an appendix.

2. This problem is concerned with simulating and fitting a simple Ising Markov Random Field (MRF) model.

   (a) Simulate an Ising model process on a 32×32 grid. Show results for $\beta = \{0.2, 0.5, 0.75, 0.9, 1.25\}$. Discuss your results.

   (b) Consider a simple model:

$$y_i = \beta_0 + \beta_1 + \epsilon_i, \;\; \epsilon_i \sim \; N(0, \sigma^2),$$

$$\boldsymbol{\beta} \sim \; N(\mathbf{0}, \boldsymbol{\Sigma}_\beta),$$

$$\sigma^2 \sim \; InvGamma(q, r).$$

   Your task is to choose some hyperparameters $(q, r, \boldsymbol{\Sigma}_\beta, \sigma^2)$ and simulate some observations, $y_i : i = 1, \ldots, n$. Then, use Approximate Bayes Computation (ABC) to obtain the posterior distribution of the parameters, $\boldsymbol{\beta}, \sigma^2$ given the simulated data. Describe the selections you use for the ABC procedure. [Optional: compare to an MCMC computed posterior with the SAME simulated dataset.]

   (c) Consider the black and white image on the Blackboard site, `myst_im.dat`. Use a hybrid MCMC/ABC Ising algorithm (as described in the lecture) to find the approximate posterior distribution of the parameters and to filter (denoise) the image. Describe your settings, show the posterior summary of the parameter distributions (β, σ^2), and show the posterior denoised image. Most importantly, take a guess at what the image is.