Lab 2
CO902 – Probabilistic and statistical inference – 2012-13 Term 2
Lecturer: Tom Nichols

1. The *mean squared error or MSE* of an estimator is the expected value of its squared deviation from the true value of the parameter of interest. The MSE is a measure of the accuracy of the estimator that combines both bias and variance, and in fact can be show to be equal to the squared bias plus the variance of an estimator.
    a. Plot the MSE of the Bernoulli MLE against n for a few θ's (say, at least 0.01, 0.1 and 0.5). You may want to use the MATLAB function `binornd`, which simulates a Binomial random variable, i.e. the sum n Bernoulli random variables.
    b. Modify *bernoulli_MLE* to output also a second estimate, in which 1 is added to the numerator and 2 to the denominator: that is, \hat{theta} = (#successes + 1)/(n+2)
    c. Using the same simulation strategy as before, plot the MSE of both estimators (as a function of n) against each other. Which is better?

2. A *Markov chain* is a sequence of RVs X_i, having the property that the distribution of each RV given all the "previous" ones is equal to the distribution of the RV given only its immediate predecessor (*first order Markov assumption*). That is,
    $P(X_i \mid X_1 \ldots X_{i-1}) = P(X_i \mid X_{i-1})$.
   Consider the case where all RVs are binary, such that $X_i = \{0, 1\}$. Then, the conditional probabilities $P(X_i \mid X_{i-1})$ can be written as a *transition matrix* $\boldsymbol{T}$, with $T_{jk} = P(X_i = k \mid X_{i-1} = j)$. The marginal distribution $P(X_1)$ is called the *initial distribution* and is denoted by P_0. In this problem you will simulate, estimate and predict with this Markov chain model.
    a. Write a program *simulate_markov_chain* which given initial distribution P_0, transition matrix $\boldsymbol{T}$, and sample size n, generates n samples from the chain.
    b. Write a program *estimate_transition_matrix*, which given data $X = [X_1 \ldots X_n]$, $X_i = \{0, 1\}$ returns an estimate of the transition matrix $\boldsymbol{T}$. (It is OK to just return a subset of $\boldsymbol{T}$, as long as $\boldsymbol{T}$ can be uniquely determined from what you return).
    c. Contrast the ability of the Bernoulli MLE and *estimate_transition_matrix* to predict an unseen observation generated from a Markov chain. That is, at each of a range of sample sizes:
        i. Generate data using *simulate_markov_chain* (with uniform initial distribution P_0 and a symmetric transition matrix with entries $T_{jk} = 0.9$ for $j=k$, and 0.1 otherwise).
        ii. Apply the two estimators to *all but the last observation*.
        iii. Use your parameter/transition matrix estimates to *predict* this unseen observation.
        iv. Compare the prediction with the truth to see how often it's correct.

   Doing all this a large number of times will reveal the average prediction error of the two approaches. What happens to the error rate of the iid Bernoulli approach as sample size n grows? What happens to the Markov Chain method? Can you suggest why?

3. [*Please contact me before starting… we might save this for Lab 3*]
   Download the file spam_data.mat from the website. This contains a data vector X where each component corresponds to an email in a database. $X_i = 1$ signifies that the email in question contained the word "free". Each of these 1000 emails was hand-labelled by a human as spam or not-spam; a second vector "label" contains these labels, with $\text{label}_i = 1/0$ signifying that email i was determined to be spam/not-spam. Assuming P(spam) = P(non spam) = 0.5, what do you need to estimate from the dataset X in order to be able to make a prediction regarding the spam status of a new email?
    a. Write a program *learn_spam_params* that learns the required parameters.

b. Write a second program *predict_spam* that, given a data vector X_{new}, returns a yes/no prediction for each email message in X_{new}. In addition to a prediction, give a probability expressing the confidence of your classification.
c. Split the 1000 observations into “train” and “test” samples. Apply *learn_spam_params* to the training samples to learn the parameters, and then apply *predict_spam* to the test data and measure the prediction accuracy. For comparison, also run *predict_spam* with the training samples. Which has higher accuracy?