Exercise 1: Familiarize yourself with GLMs

a) Load exercise1-1.mat. This .mat file contains a design matrix X and observed data y. The data was generated by the linear model y = X*w + epsilon, where epsilon is random Gaussian noise. The design matrix X has 3 columns, with the last one being a constant offset. From this dataset, estimate w using linear regression methods. You can use the *regress* function for this purpose. Verify that your results make sense by plotting the function you have fit, for example using *plot3* and *mesh*.
   Now fit this same data using a GLM. Remember, a GLM with *normal* errors and identity nonlinearity is equivalent to linear regression. You can use the *glmfit* function in Matlab for this purpose. Since we are explicitly including an offset in our model, call glmfit with the parameter 'constant' set to 'off', otherwise it won't work. Compare the weights given by *glmfit* to the ones you observed in (a). Are they the same?
b) Load exercise1-2.mat. This .mat file is like the previous one, except that here the data is generated by y ~ Binomial(Logistic(X*w)), where Logistic(z) = 1/(1+exp(-z)). Fit with linear regression, and also with the appropriate GLM. Note that by default *glmfit* assumes that the nonlinearity is the logistic for the binomial distribution, so you just need to tell it to use a binomial distribution. Plot both sets of results. In the case of the GLM, make sure to plot the result *after* the nonlinearity. How do the results differ? Explain why a GLM is more appropriate than linear regression for this dataset.
c) Use glmfit to compute the deviance of the model in (b), and also the deviance of a model with only an offset (only the third column of X). Based on this information, is it worth it to add the two extra predictors (the two first columns of X)? Confirm this by determining the p-values associated with each coefficient (hint: read the documentation for the *glmfit* function).

Exercise 2: Fit rat data with GLM

a) Load exercise2.mat. This contains simulated data from a rat doing a single run in a linear maze. y contain the spike counts, while x contains the position data. Does this cell prefers positions close to the beginning (x=0) or to the end (x=1) of the track? Check this by binning the data into epochs and figuring at what position the spike rate is highest (histogram).
b) Now fit a purely spatial GLM to this dataset; that is, a GLM with two factors, one for position and one for the offset. This is spike data, so use the correct distribution. You can use the default nonlinearity assumed by *glmfit* which is the exponential. Plot the expected spike rate as a function of position based on your fitted model. Is this consistent with the result you got with (a)?
c) Fit a GLM to this data accounting only for the post-spike filter effect, and not the spatial effect. Create a time lagged version of y (15 time lags; see slide 43 for what this matrix should look like), and use this (plus an offset) as the design matrix of a GLM. Plot the corresponding post-spike filter.
d) Fit a GLM including both a post-spike filter and the spatial effect. Compare the post-spike filter obtained with the one obtained in (c). What are the differences that you see? Try to explain why you obtain different time filters in (c) and (d).
e) Compute the deviance of the model in (d) assuming a post-spike filter which includes 1 time lag, 2 time lags and so on up to 15 time lags (see slide 62). Hint: read the documentation for *glmfit*. Based on this, what is the "best" number of time lags to include in the model? Remember, more parameters always lead to better deviance, but the improvement might not be significant.

Exercise 3: Fit rat data with GAM (extra credit: 2 points)

a) Familiarize yourself with splines. Choose a one-dimensional function that you like (sinusoid, Gabor, etc.) and generate a curve over a predefined x range. Use the function getSplineBasisFromKnots to get a basis of cubic splines. The knot sequence that you choose should cover more than the x range of your curve. Start with 10 knots. To fit your curve with splines, simply use the basis from getSplineBasisFromKnots as the design matrix in a linear regression. Compare your function and the spline approximation. Experiment with different knot positions. How does the approximation vary as you increase the number of knots?
b) Load exercise3.mat. This is much like the data in exercise 2, but this cell's receptive field is localized away from the edges. Again, using graphical methods, try to eyeball the position of the receptive field. Generate a spline basis with 10-15 knots covering the [0,1] interval. Fit a GLM with the spline basis as the design matrix. Plot the recovered place field. Is this consistent with your graphical estimate? Add a post-spike filter effect to the design matrix and refit the model. Did the shape of the estimated receptive field change?

Deliverables: In a .doc file, answer each question. Use plots to illustrate your points. Include the relevant snippets of your code. No question should take more than ~10 lines of code to answer. You don't have to do question 3 but if your class project has something to do with GLMs or you have a background in machine learning or stats you should definitely try it.