General Linear Model  
Seqian Wang, In collaboration with Sulantha Mathotaarachchi and Maxime Parent

# 1) Familiarize yourself with GLMs

## a) Estimate w using a linear regression for a normal distribution

load exercise1-1.mat  
wLr = regress(y,X);  
scatter3(X(:,1),X(:,2),X\*w); hold on; scatter3(X(:,1),X(:,2),y,'g');

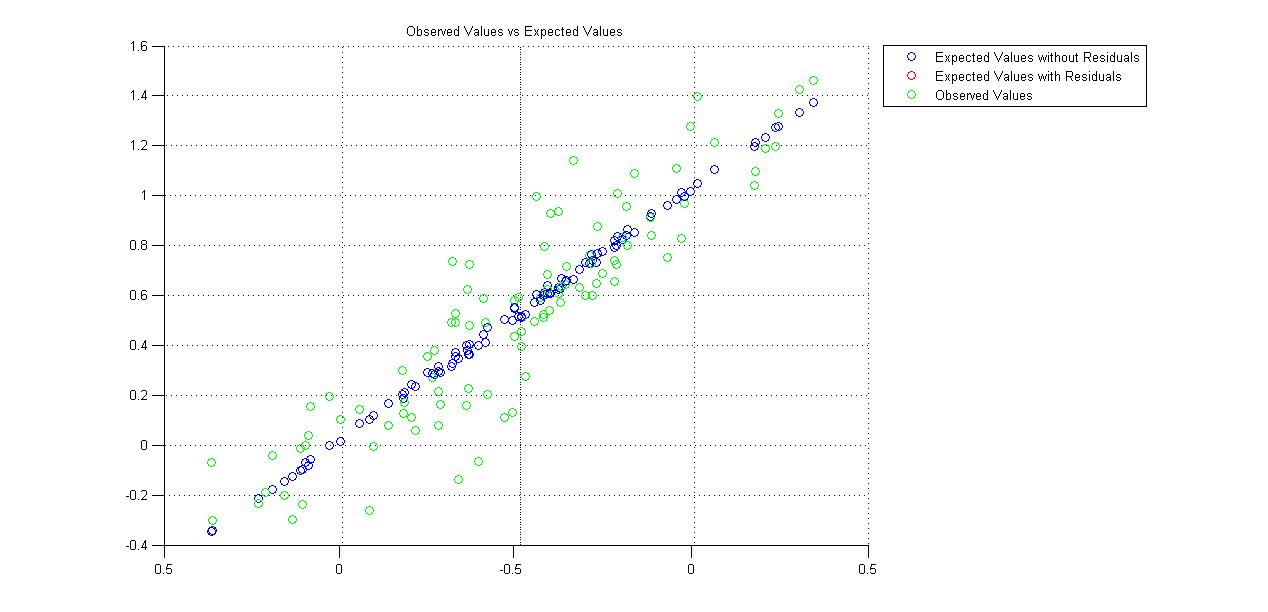


Figure 1: Predicted Values from a Matrix Multiplication between the Design Matrix (X) and the weights (w), vs Observed values (y). The predicted values follow a linear regression. w = [ 0.9838 -1.0331 0.5278].

## Estimate w using a GLM for a linear distribution

wGlm = glmfit(X,y,'normal','constant','off');

The data is continuous; therefore, we expect the GLM to have normal errors, an identity nonlinearity, and to be equivalent to linear regression. Indeed, the weights are the same (w = [ 0.9838 -1.0331 0.5278]).

## b) Estimate w using a linear regression for a binomial distribution

wLr = regress(y,X);  
wGlm = glmfit(X,y,'binomial','constant','off');

The weights obtained from both techniques differ: wLr = [ 0.8184 -1.0317 0.5835 ], while wGlm = [ 15.9435 -19.3639 1.9046 ]. In this case, a GLM for a binomial distribution is more appropriate because a linear regression is limited to modelling continuous data, whereas we are working with a binary dataset here.

scatter3(X(:,1),X(:,2),X\*wLr,'b'); hold on;  
scatter3(X(:,1),X(:,2),1./(1+exp(-X\*wGlm)),'r');  
scatter3(X(:,1),X(:,2),y,'g'); hold off;

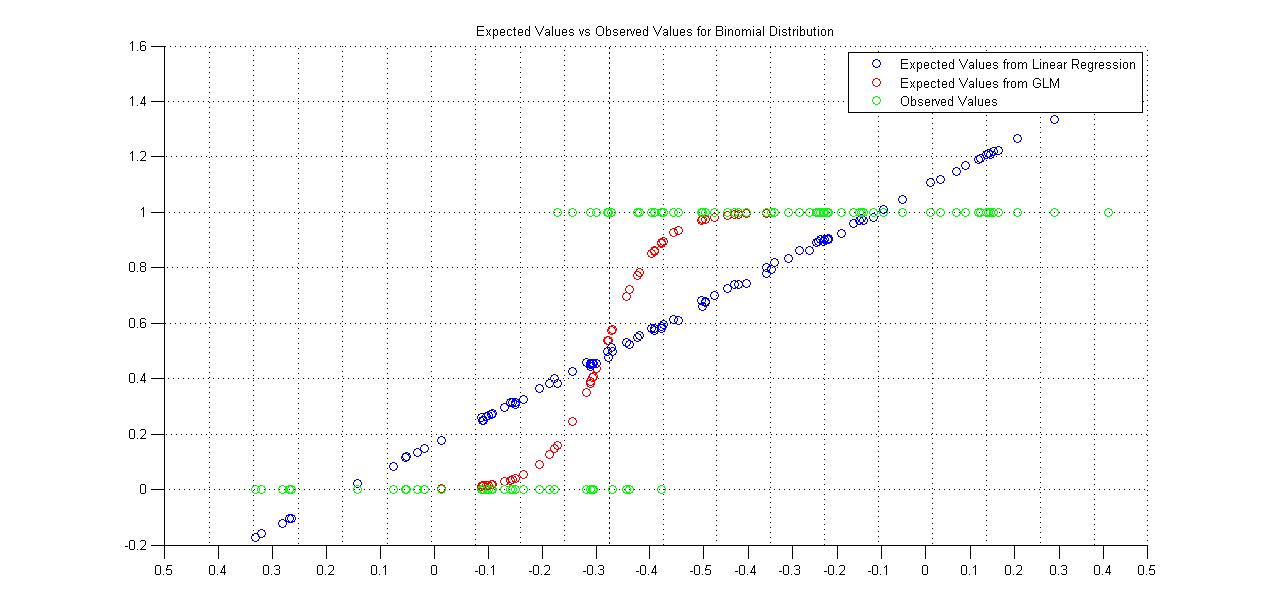


Figure 2: Predicted values from a linear regression, a GLM after nonlinearity, vs the observed values. The GLM after nonlinearity is a much better fit with observed values.

A binomial distribution requires a logistic regression to properly model it, and so we applied one for the GLM’s nonlinearity.

## c) Compute the model deviance and the p-values

[~,wDeviance, stats] = glmfit(X,y,'binomial','constant','off');  
[~,wDevianceOffset, statsOffset] = glmfit(X(:,3),y,'binomial','constant','off'); display(wDeviance, stats.p, wDevianceOffset, statsOffset.p);

The deviance is an objective function that informs us on the fitting of the model to observed values. The less the deviance is, the better fit it is.

Table 1: Model deviance and p-values, with all weights and only the offset

|  |  |
| --- | --- |
| wDeviance | 34.0569 |
| wDevianceOffset | 131.7911 |
|  | X(:,1) | X(:,2) | Offset |
| p-values | 0.0003 | 0.0002 | 0.0045 |
| p-values with Offset only |  |  | 0.0102 |

From Table 1, the deviance of the model with all the weights is lower than the model with only the offset, suggesting that the former model is a better fit to observed values. p-values for the offset is lower when in the model with two extra predictors, suggesting that it is more significant there. Likewise, the p-values of both predictors are highly significant, suggesting that adding the two extra predictors is worth it for a better fit.

# 2) Fit rat data with GLMs

## a) Spike Rate Histogram

hist([x(y == 2) ; x(y == 2) ; x(y == 1)]); % The inner array is a list of x’s (positions) where action potential occurred. If 2 action potentials were recorded for a given position, it is noted twice in the array.

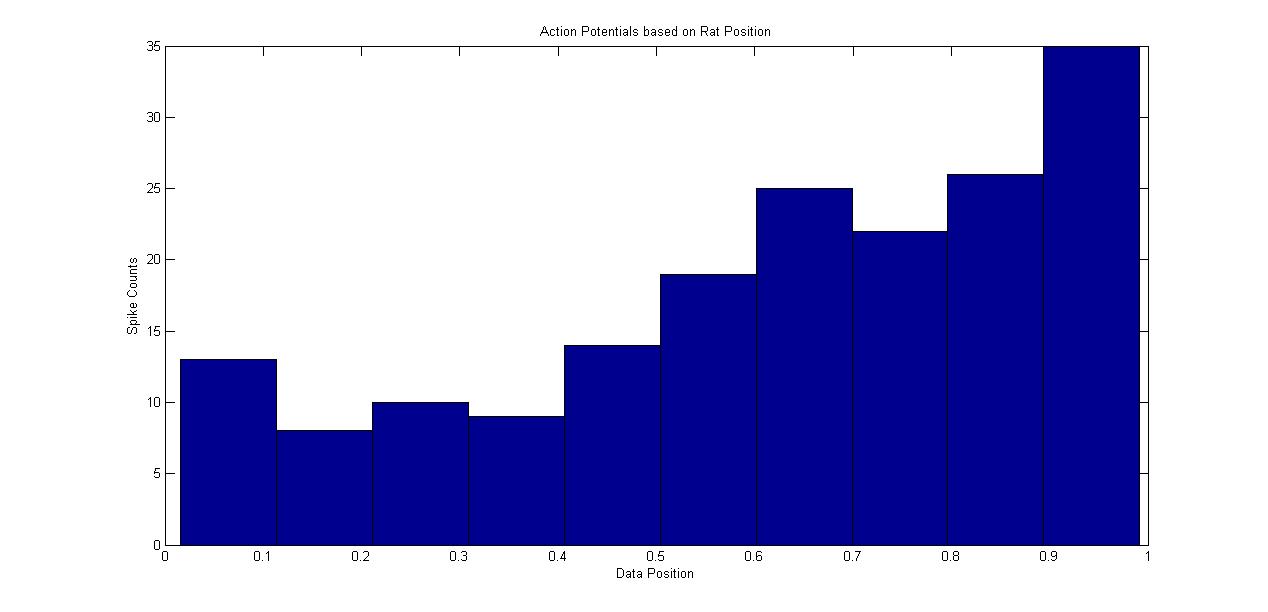


Figure 3: Spike Count as a function of the rat position

From the histogram, we can deduce that the recorded neuron prefers positions close to the end of the track, as its firing rate is higher there.

## b) Estimate w using a GLM for a Poisson distribution

Since we are dealing here with spike data (which is really a count of action potentials), a Poisson regression with exponential canonical nonlinearity is the most suitable.

X = [x ones(2000,1)]; % Spatial GLM with an offset in second column  
wGlm = glmfit(X,y,'poisson','constant','off');  
plot(X(:,1),exp(X\*wGlm));

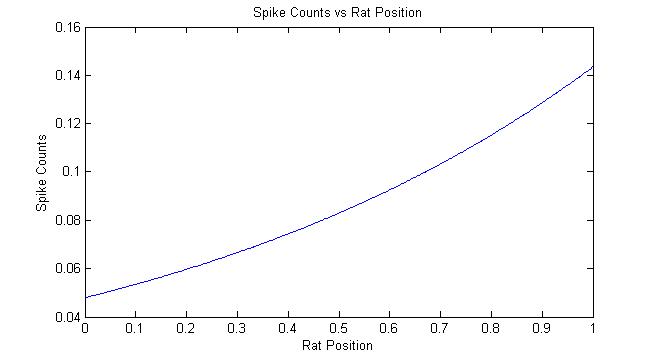


Figure 4: Spike Counts as a function of rat position

The GLM model predicts a higher spike count for rat positions towards the end of the maze, similar to what was found in the histogram.

## c) GLM model for the post-spike filter effect

times = 15;  
X = zeros(2000,times);  
for j = 1:1999  
 for k = 1:times  
 X(k+j+1, k) = y(j)>0;  
 end  
end  
X = [X ones(length(X),1)]; % Add an offset column  
wGlm = glmfit(X(1:2000,:),y,'poisson','constant','off');  
plot(wGlm(1:15));

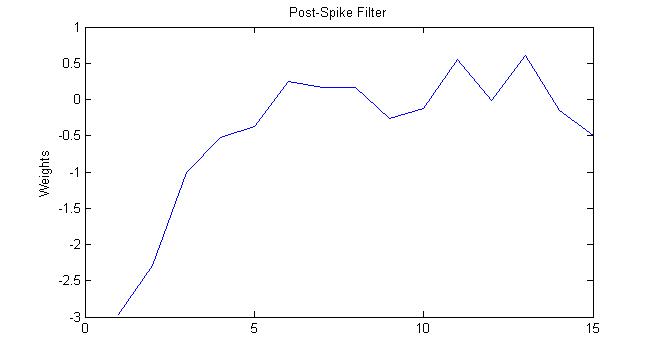


Figure 5: Post-Spike Filter from the GLM

The post-spike filter is the weights obtained from the GLM with a Poisson distribution.

## d) GLM model with the post-spike and spatial effects

X = [x X(1:2000,:)]; % Add a spatial effect column  
wGlm = glmfit(X,y,'poisson','constant','off');  
plot(wGlm(1:16));

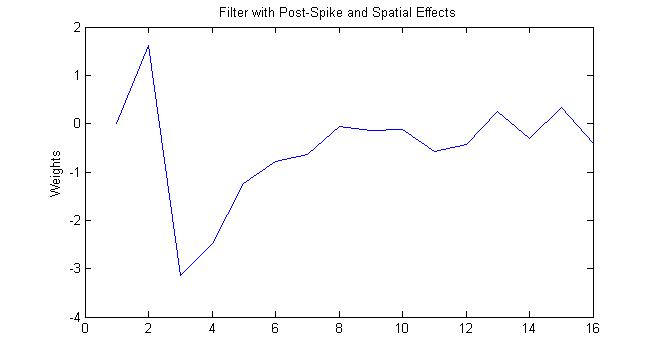


Figure 6: Filter with Post-spike and spatial effects

Compared to the previous filter with no spatial effect, the weights are different. This is because there is an additional consideration or effect that is involved into the model. With the addition of the spatial effect, the position of the rate of the maze will be considered into the glm, namely the difference in spike rate between the beginning and the end.

e) Model Deviance as a function of the number of time lags

deviances = [];

for times = 1:15

X = zeros(2000,times+1);

for j = 1:1999

for k = 1:times+1

X(k+j+1, k) = y(j)>0;

end

end

X = [X ones(length(X),1)]; % Add an offset column

[~,deviance] = glmfit(X(1:2000,:),y,'poisson','constant','off');

deviances(end+1) = deviance;

end

plot(1:15,deviances);

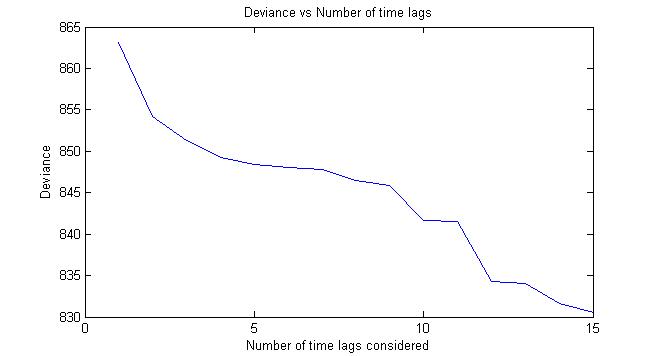


Figure 7: Deviance as a function of number of time lags

As mentioned previously, the lower the deviance, the better and more fit the model to observed values. The more effects or columns are considered, the lower will be the deviance, but the improvement will not be as sharp as initially. With Figure 7, it seems that the deviance is less affected after 13 time lags, and thus would be the best number of time lags to consider in the model (although it would be wise to check with more time lags to really see a decreasing rate of deviance change).