**Core – Senior Data Scientist Take Home Notes**

**Question 1:**

There are two scripts I’ve provided for this part.

1. *Bayesian NegBinom.*R: This contains the STAN model, which is run on simulated data. It takes about 40-45 minutes to run on my Macbook.

There is one improvement that could be made straight away: the STAN model in this part doesn’t incorporate the fact that the carryover effect after 15-20 weeks is pretty much zero. In this version of the model, the adstock\_rates matrix is a 500x2 matrix. We can set a *max\_lag* parameter, which is the number of timesteps after which we expect the carryover effect to be essentially zero. This would mean the adstock\_rates matrix will simply be a *max\_lag* x 2 matrix instead. The Adstock function would have to be edited for this.

Note that in the code, I am using the carryover rate, and not decay.

1. *nbinom\_adstock\_v2.*R: This contains the second version of the Adstock function in R which also takes in a *max\_lag* parameter. Unfortunately, I haven’t had time to implement it yet in STAN.

95% intervals for the model parameters are shown below, quite good estimates overall! The actual *mu* values were 4.177777, 5.465461 respectively.

The actual *phi* values were 0.2888798, 0.4428182 respectively.

Chart, scatter chart

Description automatically generated

**Diagnostics/Checks:**

1. **Trace Plot:** No obvious issues, chains seem to have converged. They exhibit randomness, stationarity, and good mixing. Additionally, all R-Hat values are 1.00.

**A picture containing arrow

Description automatically generated**

1. **Leave-one-out-validation using the *loo* library**

The results of the output are pasted below, with interpretation.

#########################################

Computed from 2000 by 500 log-likelihood matrix

Estimate SE

elpd\_loo 1597.9 16.4

p\_loo 11.1 2.7

looic -3195.8 32.8

------

Monte Carlo SE of elpd\_loo is NA.

Pareto k diagnostic values:

Count Pct. Min. n\_eff

(-Inf, 0.5] (good) 498 99.6% 607

(0.5, 0.7] (ok) 0 0.0% <NA>

(0.7, 1] (bad) 2 0.4% 64

(1, Inf) (very bad) 0 0.0% <NA>

######################################

* The first estimate ‘elpd\_loo’ gives the estimated log pointwise predictive density of the model, along with it’s SE. Higher value generally indicates a better fit.
* The second estimate, ‘p\_loo’ represents the effective number of parameters in the model; higher value indicates a more complex model usually.
* The looic value (-3195.8) is a corrected version of the AIC, which penalizes the model for its complexity. A lower looic value indicates a better model fit.

We’re mainly going to use these for comparing our models.

The pareto k diagnostic section shows the Pareto k diagnostic values for each observation in the dataset. The Pareto k value measures the degree of influence of each observation on the loo estimate, with values above 0.7 indicating potentially problematic observations.

99.6% of the observations have a Pareto k value below 0.5, which is considered good. However, two observations (0.4%) have a Pareto k value above 0.7, indicating that they may be influential and potentially problematic. The corresponding effective sample sizes (n\_eff) for these observations are relatively low (64), indicating that they may be poorly estimated by the model. This is also shown on the plot below:

Chart, scatter chart

Description automatically generated

Overfitting is a concern with this model, one way to test it would be to perform hold-out validation by splitting the dataset into training and test datasets. We will have to fit the model on the training set, and then evaluate its performance on the test set. If the performance on the test set is significantly worse than on the training set, it may be an indication of overfitting.

**Task 2:**

The graphs both illustrate the saturation/diminishing returns effect. Channel 2 has a higher value for *Kappa*, which is exhibited through the graph below.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

**Task 3**

The transformed parameters section of the code computes the predicted values as the sum of the intercept and the product of each predictor variable's adstocked spend and its corresponding ROI value..

The model was converted to a log-linear model by applying log transformations to the *predicted* variable.

The intercept estimate is drastically different for the multiplicative model. This is because in the additive model, the intercept represents the expected response when all predictors are equal to zero. However, in the log-linear model, the intercept represents the expected response when all predictors (in our case, the spend) are equal to one (since the predictors are on a log scale). As a result, it is common to use a prior that reflects prior knowledge or beliefs about this value.

The log-likelihood in the generated quantities block represents the log-likelihood of the observed data given the posterior samples of the model parameters. Specifically, for each data point, the log-likelihood is computed as the log-density of the normal distribution with mean equal to the predicted value of the response variable at that time point and standard deviation equal to the estimated sigma parameter. The *loo* library uses the log likelihood parameter.

Output from loo:

> loo\_mult

Estimate SE

elpd\_loo 985.9 32.7

p\_loo 18.7 8.8

looic -1971.8 65.4

------

Monte Carlo SE of elpd\_loo is NA.

Pareto k diagnostic values:

Count Pct. Min. n\_eff

(-Inf, 0.5] (good) 497 99.4% 327

(0.5, 0.7] (ok) 2 0.4% 114

(0.7, 1] (bad) 0 0.0% <NA>

(1, Inf) (very bad) 1 0.2% 4

See help('pareto-k-diagnostic') for details.

> loo\_add

Estimate SE

elpd\_loo 1562.2 15.2

p\_loo 7.9 0.9

looic -3124.4 30.3

------

Monte Carlo SE of elpd\_loo is NA.

Pareto k diagnostic values:

Count Pct. Min. n\_eff

(-Inf, 0.5] (good) 499 99.8% 656

(0.5, 0.7] (ok) 0 0.0% <NA>

(0.7, 1] (bad) 1 0.2% 118

(1, Inf) (very bad) 0 0.0% <NA>

The elpd\_loo column in both outputs represents the estimated expected log pointwise predictive density (ELPD), which measures the predictive accuracy of the model. Higher elpd\_loo values indicate better model performance. The p\_loo column represents the effective number of parameters in the model.

Based on these outputs, we can see that loo\_add has a higher elpd\_loo value (1562.2) compared to loo\_mult (985.9), indicating that loo\_add has better predictive performance. Additionally, loo\_add has a lower value for p\_loo (7.9) compared to loo\_mult (18.7), indicating that it is a more parsimonious model.

To compare the p\_loo and looic parameters from the two models, we can use the following guidelines:

1. For the p\_loo parameter, a lower value indicates a more parsimonious model that uses fewer parameters to achieve a similar level of predictive accuracy.
2. For the looic parameter, a lower value indicates a better-performing model.

We can see that loo\_add has a lower value for p\_loo (7.9) compared to loo\_mult (18.7), suggesting that it is a more parsimonious model. Additionally, loo\_add has a lower value for looic (-3124.4) compared to loo\_mult (-1971.8), indicating that it is a better-performing model.

**Task 4: Overfitting**

The most obvious change would be for different media channels to share parameters, specifically the half-saturation and mu/phi parameters in the negative binomial distribution (based on prior knowledge).

Cross validation could also be used to determine if the model was overfit. If the validation error is significantly higher than the training error, it indicates overfitting. Can use metrics such as MSE for this.

* If you have strong prior knowledge on the parameters, you should use narrower priors which reflect that knowledge instead of weakly informative priors.
* Regularisation techiniques such as L1/Lasso or L2/Ridge can also be applied, by adding a penalty term to the log-likelihood. The regularization strength can be controlled by a hyperparameter, which can be tuned using techniques such as cross-validation or by maximizing the marginal likelihood.
  + L1 regularization, also known as Lasso regression, involves adding a penalty term proportional to the absolute value of the model parameters. In Bayesian regression, this can be achieved by placing a Laplace prior distribution on the model parameters, which has a sharp peak at 0 and heavy tails. The Laplace prior encourages sparse solutions, where many of the model parameters are exactly 0, which can be useful for feature selection.
  + L2 regularization, also known as Ridge regression, involves adding a penalty term proportional to the square of the magnitude of the model parameters. In Bayesian regression, this can be achieved by placing a normal prior distribution on the model parameters with mean 0 and variance proportional to the inverse of the regularization strength.

For L1 regularization, you can add a lasso prior to the betas parameter. This can be done by adding the following line betas ~ double\_exponential(0, *lasso\_param*). ­­*lasso\_param* controls the regularisation strength.

Similarly for L2 regularization, you can add a ridge prior to the betas parameter. This can be done by specifying betas ~ normal(0, *ridge\_param*).

To tune the hyperparameters such as *lasso\_param* or *ridge\_param*, there are libraries in R that allow you to do so.

1. mlrMBO: uses model based optimisation (MBO) algorithms to for hyperparameter tuning.
2. Bayesopt:
   1. Split the data into training and test sets. We can use the Mean Squared Error as the validation error. The validation error is the **objective function** we want to minimise.
   2. Define a set of hyperparameters to optimize and their ranges or distributions, and constraints on those hyperparameters.
   3. BayesOpt will then try to find the value of the hyperparameters that minimises the objective function. Algorithms used include random search, Bayesian optimisation with Gaussian Processes etc. One algorithm that I can describe that is used is **simulated annealing**.

**Appendix: Deriving the negative binomial adstock function**

In this section, I’ll briefly try to explain the logic driving my implementation of the negative binomial adstock model.

For the sake of simplicity, assume that there is only one channel. The ‘spend’ matrix is then given as:

Where is the spend in the first week, is the spend in the second week and so on.

Separately, we also have a ‘rates’ matrix, which contains the adstock rates which we need to calculate the adstocked spend. The rates matrix is the same length as the spend matrix.

The adstocked spend in week 1 is simply

For week 2, the adstocked spend is

Similarly, for week 3, the adstocked spend is