Interaction Terms in Regression

General Principles

If you have a case where you believe the effect of one independent variable depends on the value of another independent variable, you can use regression analysis with interaction terms. In this approach, we extend the simple linear regression model to include an interaction term (a multiplication) between the two independent variables (see note on how this multiplication arises).

Considerations

Note

- We have the same assumptions as for Regression for continuous variable.
- We wish to model the relationship between a dependent variable, Y, and an independent variable, X_1, whose effect varies as a function of a second independent variable X_2. To do this, we explicitly model the hypothesis that the slope between Y and X_1 depends on (i.e., is conditional on) X_2.
- For continuous interactions with normalized data, the intercept becomes the grand mean of the outcome variable.
- The interpretation of slopes estimates is more complex. The coefficient for a non-interaction term reflects the expected change in Y when X_1 increases by one unit, holding X_2 constant at its average value. The coefficient for the interaction term represents how the effect of X_1 on Y changes depending on the value of X_2, and vice versa, showing how the relationship between the two variables influences the outcome Y.
- Triptych plots are very handy for understanding the impact of interactions, especially when more than two interactions are present.

Example

Below is example code demonstrating Bayesian regression with an interaction term between two continuous variables using the Bayesian Inference (BI) package. The data consist of three continuous variables (temperature, humidity, energy consumption), and the goal is to estimate the effect of the interaction between temperature and humidity on energy consumption. This example is based on McElreath (2018).

Python

```python
from BI import bi

# Setup device------------------------------------------------
m = bi(platform='cpu')

# Import Data & Data Manipulation ------------------------------------------------
# Import
from importlib.resources import files
data_path = m.load.tulips(only_path = True)
m.data(data_path, sep=';')
m.scale(['blooms', 'water', 'shade']) # Normalize


# Define model ------------------------------------------------
def model(blooms,shade, water):
    sigma = m.dist.exponential(1, name = 'sigma', shape = (1,))
    bws = m.dist.normal(0, 0.25, name = 'bws', shape = (1,))
    bs = m.dist.normal(0, 0.25, name = 'bs', shape = (1,))
    bw = m.dist.normal(0, 0.25, name = 'bw', shape = (1,))
    a = m.dist.normal(0.5, 0.25, name = 'a', shape = (1,))
    mu = a + bw*water + bs*shade + bws*water*shade
    m.dist.normal(mu, sigma, obs=blooms)

# Run mcmc ------------------------------------------------
m.fit(model) # Optimize model parameters through MCMC sampling

# Summary ------------------------------------------------
m.summary()
```

```
jax.local_device_count 16
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:01<26:16,  1.58s/
arviz - WARNING - Shape validation failed: input_shape: (1, 500), minimum_shape: (chains=2, d
```

	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a[0]	0.09	0.10	-0.05	0.25	0.0	0.01	560.88	176.01	NaN
bs[0]	-0.31	0.11	-0.49	-0.13	0.0	0.01	634.25	265.47	NaN
bw[0]	0.56	0.10	0.39	0.72	0.0	0.00	510.54	317.55	NaN
bws[0]	-0.32	0.11	-0.53	-0.17	0.0	0.01	504.61	382.08	NaN
sigma[0]	0.57	0.09	0.42	0.70	0.0	0.00	484.71	399.50	NaN

R

```r
library(BayesianInference)
m=importBI(platform='cpu')

# Load csv file
m$data(m$load$tulips(only_path = T), sep = ''), sep=';')
m$scale(list('blooms', 'water', 'shade')) # Normalize
m$data_to_model(list('blooms', 'water', 'shade')) # Send to model (convert to jax array)

# Define model ------------------------------------------------
model <- function(blooms, water,shade){
  # Parameter prior distributions
  alpha = bi.dist.normal( 0.5, 0.25, name = 'a')
  beta1 = bi.dist.normal( 0,  0.25, name = 'b1')
  beta2 = bi.dist.normal(  0,  0.25, name = 'b2')
  beta_interaction_ = bi.dist.normal(  0, 0.25, name = 'bint')
  sigma = bi.dist.normal(0, 50, name = 's')
  # Likelihood
  m$normal(alpha + beta1*water + beta2*shade + beta_interaction_*water*shade, sigma, obs=bloo
}

# Run mcmc ------------------------------------------------
m$fit(model) # Optimize model parameters through MCMC sampling

# Summary ------------------------------------------------
m$summary() # Get posterior distributions
```

Julia

```julia
using BayesianInference

# Setup device------------------------------------------------
m = importBI(platform="cpu")

# Import Data & Data Manipulation ------------------------------------------------
# Import
data_path = m.load.tulips(only_path = true)
m.data(data_path, sep=';')
m.scale(["blooms", "water", "shade"]) # Normalize
# Define model ------------------------------------------------
@BI function model(blooms,shade, water)
    sigma = m.dist.exponential(1, name = "sigma", shape = (1,))
    bws = m.dist.normal(0, 0.25, name = "bws", shape = (1,))
    bs = m.dist.normal(0, 0.25, name = "bs", shape = (1,))
    bw = m.dist.normal(0, 0.25, name = "bw", shape = (1,))
    a = m.dist.normal(0.5, 0.25, name = "a", shape = (1,))
    mu = a + bw*water + bs*shade + bws*water*shade
    m.dist.normal(mu, sigma, obs=blooms)
end

# Run mcmc ------------------------------------------------
m.fit(model)  # Optimize model parameters through MCMC sampling

# Summary ------------------------------------------------
m.summary() # Get posterior distributions
```

Mathematical Details

Frequentist formulation

We model the relationship between the input features (X_1 and X_2) and the target variable (Y) using the following equation:

$$Y_i = \alpha + \beta_1 X_{[1,i]} + \beta_2 X_{[2,i]} + \beta_3 X_{[1,i]} X_{[2,i]} + \epsilon_i$$

Where:

- Y_i is the dependent variable for observation i.

- α is the intercept term.
- $X_{[1,i]}$ and $X_{[2,i]}$ are the values of the two independent variables for observation i.
- β_1 and β_2 are the coefficients for X_1 and X_2, respectively, when the other variable has value 0.
- β_3 is the coefficient which controls the extent to which the coefficient on one variable depends on the value of the other.
- ϵ_i is the error term, assumed to be independent and normally distributed.

Bayesian formulation

In the Bayesian formulation, we define each parameter with priors . We can express the Bayesian regression model as follows:

$$Y_i \sim \text{Normal}(\alpha + \beta_1 X_{[1,i]} + \beta_2 X_{[2,i]} + \beta_3 X_{[1,i]} X_{[2,i]}, \sigma)$$

$$\alpha \sim \text{Normal}(0, 1)$$

$$\beta_1 \sim \text{Normal}(0, 1)$$

$$\beta_2 \sim \text{Normal}(0, 1)$$

$$\beta_3 \sim \text{Normal}(0, 1)$$

$$\sigma \sim \text{Exponential}(1)$$

Where:

- Y_i is the dependent variable for observation i.
- α is the intercept term, which in this case has a unit-normal prior.
- β_1 and β_2 are the coefficients for X_1 and X_2, respectively, when the other variable has value 0.
- β_3 is the coefficient which controls the extent to which the coefficient on one variable depends on the value of the other.

- $X_{[1,i]}$ and $X_{[2,i]}$ are the two values of the independent continuous variables for observation i.
- σ is a standard deviation parameter, which here has an Exponential prior that constrains it to be positive.

Notes

Note

The interaction term equation:

$$Y_i \sim Normal(\alpha + \beta_1 X_{[1,i]} + \beta_2 X_{[2,i]} + \beta_3 X_{[1,i]} X_{[2,i]}, \sigma)$$

can be re-written as:

$$Y_i \sim Normal(\alpha + (\beta_1 + \beta_3 X_{[2,i]}) X_{[1,i]} + \beta_2 X_{[2,i]}, \sigma)$$

simply by factoring the terms with $X_{[1,i]}$ in them. The result is that the coefficient on $X_{[1,i]}$ is written specifically as a linear regression model of $X_{[2,i]}$.

Reference(s)

McElreath, Richard. 2018. *Statistical Rethinking: A Bayesian course with examples in R and Stan*. Chapman; Hall/CRC.