correct-q2

June 21, 2024

[11]:
```
import pandas as pd
import numpy as np
from scipy.stats import norm, truncnorm, uniform

# Load data from URL
url = "https://raw.githubusercontent.com/yadavhimanshu059/CGS698C/main/notes/
 ↪Data/word-recognition-times.csv"
dat = pd.read_csv(url)
dat = dat.iloc[:, 1:]  # Remove the first column

# Convert 'type' to numeric indicator
dat['type_indicator'] = np.where(dat['type'] == 'word', 0, 1)

# Extract data
y = dat['RT'].values
type_indicator = dat['type_indicator'].values

# Define log likelihood function
def log_likelihood(alpha, beta, sigma, RT, type_indicator):
    mu = alpha + beta * type_indicator
    return np.sum(norm.logpdf(RT, loc=mu, scale=sigma))

# Prior distributions
def prior_alpha(alpha):
    return norm.logpdf(alpha, loc=400, scale=50)

def prior_beta(beta):
    return np.log(truncnorm.pdf(beta, a=0, b=np.inf, loc=0, scale=50))

# MCMC settings
nsamp = 10000  # Number of samples
alpha_chain = np.zeros(nsamp)
beta_chain = np.zeros(nsamp)

# Initial values
alpha_chain[0] = np.random.normal(400, 50)
```

```
beta_chain[0] = truncnorm.rvs((0 - beta_chain[0]) / 0.1, (np.inf -␣
 ↪beta_chain[0]) / 0.1, loc=beta_chain[0], scale=0.1)

step_alpha = 0.1
step_beta = 0.1

i = 0
while i < nsamp - 1:
    # Propose new values
    proposal_alpha = np.random.normal(alpha_chain[i], step_alpha)
    proposal_beta = truncnorm.rvs((0 - beta_chain[i]) / step_beta, (np.inf -␣
 ↪beta_chain[i]) / step_beta, loc=beta_chain[i], scale=step_beta)

    # Calculate log posterior
    post_new = (log_likelihood(proposal_alpha, proposal_beta, 30, y,␣
 ↪type_indicator) +
                prior_alpha(proposal_alpha) +
                prior_beta(proposal_beta))

    post_prev = (log_likelihood(alpha_chain[i], beta_chain[i], 30, y,␣
 ↪type_indicator) +
                 prior_alpha(alpha_chain[i]) +
                 prior_beta(beta_chain[i]))

    # Hastings ratio
    Hastings_ratio = np.exp(post_new - post_prev)
    p_str = min(Hastings_ratio, 1)  # probability of acceptance

    if p_str > uniform.rvs():
        alpha_chain[i + 1] = proposal_alpha
        beta_chain[i + 1] = proposal_beta
        i += 1

# Calculate the 95% credible intervals
alpha_credible_interval = np.percentile(alpha_chain, [2.5, 97.5])
beta_credible_interval = np.percentile(beta_chain, [2.5, 97.5])

# Print results
print("95% credible interval for alpha:", alpha_credible_interval)
print("95% credible interval for beta:", beta_credible_interval)
```

```
95% credible interval for alpha: [367.53337729 420.71011892]
95% credible interval for beta: [ 8.65546426 53.39317177]
```

[ ]: