Latent Variable Models (WIP)

General Principles

In some scenarios, the observed data does not directly reflect the underlying structure or factors influencing the outcome. Instead, *latent variables*—variables that are not directly observed but are inferred from the data—can help model this hidden structure. These latent variables capture unobserved factors that affect the relationship between predictors (X) and the outcome (Y).

We model the relationship between the predictor variables (X) and the outcome variable (Y) with a latent variable (Z) as follows:

$$Y = f(X, Z) + \epsilon$$

Where: - Y is the observed outcome variable. - X is the observed predictor variable(s). - Z is the latent (unobserved) variable, which we aim to infer. - *f(X, Z)* is the function that relates X and Z to Y. - is the error term, typically assumed to be normally distributed with mean 0 and variance ^2.

The latent variable Z can represent various phenomena, such as group-level effects, time-varying trends, or individual-level factors, that are not captured by the observed predictors alone.

Considerations

In Bayesian regression with latent variables, we consider the uncertainty in both the observed and latent variables. We declare prior distributions for the latent variables, in addition to the usual priors for regression coefficients and intercepts. These latent variables are often modeled using Gaussian distributions (*Normal* priors) or more flexible distributions such as *Multivariate Normal* for correlations among the latent variables.

The goal is to infer the posterior distribution over both the parameters and the latent variables, given the observed data.

Example

Below is an example code snippet demonstrating Bayesian regression with latent variables using TensorFlow Probability:

```
from BI import bi
import numpy as np
import jax.numpy as jnp

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

# Data Simulation ----------------------------------------------
NY = 4  # Number of dependent variables or outcomes (e.g., dimensions for latent variables)
NV = 8  # Number of observations or individual-level data points (e.g., subjects)

# Initialize the matrix Y2 with shape (NV, NY) filled with NaN values, to be filled later
Y2 = np.full((NV, NY), np.nan)

# Generate the means and offsets for the data
# means: Generate random normal means for each of the NY outcomes
# offsets: Generate random normal offsets for each of the NV observations
means = m.dist.normal(0, 1, shape=(NY,), sample=True, seed=10)
offsets = m.dist.normal(0, 1, shape=(NV, 1), sample=True, seed=20)

# Fill the matrix Y2 with simulated data based on the generated means and offsets
# Each observation (i) is the sum of an individual-specific offset and an outcome-specific me
for i in range(NV):
    for k in range(NY):
        Y2[i, k] = means[k] + offsets[i]

# Simulate individual-level random effects (e.g., random slopes or intercepts)
# b_individual: A matrix of size (N, K) where N is the number of individuals and K is the num
b_individual = BI.distribution.normal(0, 1, shape=(N, K), sample=True, seed=0)

# mu: Add an additional effect 'a' to the individual-level random effects 'b_individual'
# 'a' could represent a population-level effect or a baseline
mu = b_individual + a

# Convert Y2 to a JAX array for further computation in a JAX-based framework
Y2 = jnp.array(Y2)
```

```
# Set data ------------------------------------------------
dat = dict(
    NY = NY,
    NV = NV,
    Y2 = Y2
)
m.data_on_model = dat

# Define model --------------------------------------------------
def model(NY, NV, Y2):
    means = m.dist.normal(0, 1, shape=(NY,), name='means')
    offset = m.dist.normal(0, 1, shape=(NV, 1), name='offset')
    sigma = m.dist.exponential(1, shape=(NY,), name='sigma')
    tmp = jnp.tile(means, (NV, 1)).reshape(NV, NY)
    mu_l = tmp + offset
    m.normal(mu_l, jnp.tile(sigma, [NV, 1]), obs=Y2)

# Run sampler ---------------------------------------------------
m.fit(model)

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

Mathematical Details

We can express the Bayesian latent variable model using probability distributions as follows:

$$
\begin{aligned}
p(Y|X, Z, W, \sigma) &= \text{Normal}(X \cdot W + Z, \sigma^2) \\
p(Z) &= \text{Normal}(0, \tau^2) \\
p(W) &= \text{Normal}(0, \alpha^2)
\end{aligned}
$$

Where: - *p(Y | X, Z, W,)* is the likelihood function for the observed outcome variable, which depends on both the observed predictor X and the latent variable Z. - *p(Z)* is the prior distribution for the latent variable Z, often modeled as *Normal* with a mean of 0 and variance *^2*. - *p(W)* is the prior distribution for the regression coefficient(s) W, typically assumed to follow a *Normal* distribution with mean 0 and variance *^2*.

The latent variable Z introduces additional flexibility to the model, capturing unobserved influences on the outcome Y.

Interpretation of Latent Variables

- **Latent Variable (Z)**: Represents hidden factors not captured by the observed variables, allowing the model to explain more of the variance in the outcome. For instance, in a psychological model, Z might represent a latent trait such as intelligence or anxiety that influences the outcome.
- **Posterior Inference**: The posterior distribution of the latent variable Z can give insights into how much the unobserved factors contribute to the outcome.

Use Cases

- **Latent Factors in Psychometrics**: In psychometric models, latent variables represent traits or abilities that are not directly observed, such as cognitive ability or personality traits.
- **Time-Varying Effects**: Latent variables can represent unobserved time trends or individual-specific effects in time-series or longitudinal models.
- **Mixed Models**: In hierarchical or mixed models, latent variables can represent group-specific intercepts or slopes.