PyStan #1

Ref: https://towardsdatascience.com/painless-introduction-to-applied-bayesian-inference-using-py-stan-36b503a4cd80

起手式:

- What is my likelihood
- What should my model be? interaction?
- What are my parameter & hyperparameter?
- What kind of priors I should choose?
- Are there any clustering, time or spatial dependence?

EDA –
- 可以先畫不同國家之間の household expenditure 之 density 圖
  - food vs non-food expenditure density 圖
- check micro model 中，y 變數 & X 之間有沒有相關性(畫散佈圖)

Data前處理:
- 類別轉dummy → pd.get_dummies(X, prefix_sep="_", drop_first=False)
- Training / Testing Split → X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=...)

Data Specification (在python中要用 dictionary):

```
cars_data = {"N": X_train.shape[0], "N_new": X_test.shape[0],
             "K": X_train.shape[1], "y_obs": y_train.values.tolist(),
             "X": np.array(X_train), "X_new": np.array(X_test)}
```

Modeling: Stan Model 分几个功能塊要 specify (其中 data, parameter & model 是必要)

```
cars_code = """
data { int<lower=1> N;
       int<lower=0> K;
       matrix[N, K] X;
       vector[N] y_obs;
       int<lower=1> N_new;
       matrix[N_new, K] X_new; }
parameter { real<lower=0> sigma;
            real alpha;
            vector[K] beta; }
transformed parameter { vector[N] theta;
                        theta = alpha + X * beta; }
model { sigma ~ exponential(1);
        alpha ~ normal(0, 6);
        beta ~ multi_normal(rep_vector(0, K), diag_matrix(rep_vector(1, K)));
        y_obs ~ normal(theta, sigma);
        y_new ~ normal(alpha + X_new * beta, sigma); } """
```

(附) 如果你有階層模型 放這 (→ parameter)

如果你很單純 只想討論某個參數 與前後的關係就用這塊 (→ transformed parameter)

prior & likelihood 放這 (→ model) $p(y|X, \alpha, \beta, \sigma)$

我們可以在Stan模型中 用類似此處 new 代號の方式 建立預測值

v.s.

theta 可以想為 training data fit出模型の參數估計值 此處是mean，代表最好 data最好の代表 也可用來做預測

跑模型：sm = pystan.StanModel(model_code = cars_code)

fit = sm.sampling(data = cars_data, iter=6000, chains=8)

(儲存模型（就不用每回跑一次建一次）：with open("bayes-cars.pkl", "wb") as f:
pickle.dump(sm, f, protocol = pickle.HIGHEST_PROTOCOL))

Result：① la = fit.extract(permuted = True) → 印出每了受估參数在每次iteration的估計值

② print(fit) → 給你一了fit出来的sumary table（含mean, stderror, credible interval, n_eff, Rhat.

Diagnosis：① Chain mixing

↳ az.plot_trace(fit, var_names=["θ","σ"])

可省

note：若想看density可用

az_data = az.from_pystan(posterior=fit)

az.plot_density(az_data, var_names=["θ"])

Visualization

② posterior credible intervals

↳ az.plot_forest(az_data, kind="forestplot", var_names=[ ], combined=True)

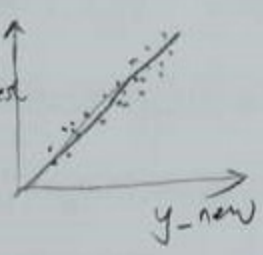
可省 默认也可

Prediction：$P(y_{new} \mid D_{old}, X_{new}) = \int P(y_{new} \mid \theta, D_{old}, X_{new}) P(\theta \mid D_{old}) d\theta$

即待预测值

↳ ① 求Bayes test MSE：metrics.mean_squared_error(y_test, la['y_new'].mean())

② 畫兩者的joint scatter plot：



手y_new和y_test做比較

③ 畫某个变数對y_new的预测图：az.plot_hpd(某变数, la["y_new"], plot_kwargs={"ls":"--"})

可省

	Train MSE	Test MSE
Bayesian	10.829	10.968
Frequentist/MLE	10.747	10.558

→ 更好的衡量做法是用bootstrap做train-test split ≥ 30次看test MSE的95% confidence interval

model performance

note：貝氏不見得比傳統Maximum Likelihood (Frequentist)好，特別在有大量的dataset