7.4: Generalized Linear Models

Instructor: Dr. GP Saggese - gsaggese@umd.edu

References:

- AIMA (Artificial Intelligence: a Modern Approach)
  - Chap 15: Probabilistic programming
- Martin, Bayesian Analysis with Python, 2018 (2e)

- ***Generalized Linear Models***
  - Simple Linear Model
  - Logistic Regression
  - Multiple Linear Regression

- Generalized Linear Models
  - ***Simple Linear Model***
  - Logistic Regression
  - Multiple Linear Regression

Linear Model

- Many problems can be formulated as **linear regression**:
  - X and Y: uni-dimensional continuous RVs
  - Dataset of paired observations: $\{(x_1, y_1), ..., (x_n, y_n)\}$
  - X: independent variables
  - Model/predict dependent variable Y
  - Assume linear relationship between Y and X
- E.g.,
  - Nobel laureates in a country vs amount of chocolate consumed
  - Sugar intake vs Blood glucose
  - Years of education vs Annual income
  - Advertising spend vs Sales
  - Study hours vs Exam score
  - Coffee consumption vs Productivity

Linear Model: Frequentist Approach

- A **linear model** is described by:

$$y = \alpha + \beta x$$

- **Frequentist approach**:
  - Find parameters α, β using least square fitting
    - Minimize average quadratic error between observed y and predicted $\hat{y}$

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\alpha + \beta x_i - \hat{y}_i)^2$$

  - Point-estimate from least squares corresponds to maximum a-posteriori with flat (uniform) priors on α and β
  - With certain assumptions, it's "optimal"
    - Residuals are Gaussian
    - Linearity in parameters
    - Independence
    - Homoscedasticity (constant variance of ε_i)

Linear Model: Bayesian Approach

- **Bayesian assumptions**:
  - Data y is Gaussian with mean $\alpha + \beta x$ and standard deviation

$$\sigma : y \sim N(\mu = \alpha + \beta x, \sigma)$$

  - α, β, σ are independent
  - Priors:

$$\begin{cases} \alpha \sim Normal(\mu_\alpha, \sigma_\alpha) \\ \beta \sim Normal(\mu_\beta, \sigma_\beta) \\ \sigma \sim HalfNormal(0, \sigma_\epsilon) \end{cases}$$

- Use **vague priors** to add problem knowledge:
  - Center intercept α around 0
  - Info on sign of β sometimes available
  - Use Half-Cauchy / exponential distribution for σ
  - Use uniform prior if parameter has hard boundaries

Linear Model: Synthetic Example

- Generate **random data from ground truth**
  - $\alpha = 2.5$
  - $\beta = 0.9$
  - Add noise

[122]:
```
np.random.seed(1)

# Number of samples.
N = 100

# Parameters.
alpha_real = 2.5
beta_real = 0.9
sigma_eps_real = 0.5

# Generate data.
x = np.random.normal(10, 1, N)
y_real = alpha_real + beta_real * x

# Add noise.
eps_real = np.random.normal(0, sigma_eps_real, size=N)
y = y_real + eps_real
```

[123]:
```
plt.scatter(x, y_real)
plt.scatter(x, y);
```

Linear Model: Synthetic Example

- Create **linear model in PyMC**
- **Use vague priors**
  - Prior of intercept α and β centered around 0
  - Half-Cauchy distribution for σ

[125]:
```
with pm.Model() as model_g:
    alpha = pm.Normal("alpha", mu=0, sigma=10)
    beta = pm.Normal("beta", mu=0, sigma=1)
    sigma = pm.HalfCauchy("sigma", 5)
    #
    mu = pm.Deterministic("mu", alpha + beta * x)
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sigma, observed=y)
    idata_g = pm.sample(2000, tune=1000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, sigma]
```

```
Sampling 4 chains, 0 divergences
```

```
Sampling 4 chains for 1_000 tune and 2_000 draw iterations (4_000 -
```

[127]:
```
pm.model_to_graphviz(model_g)
```

[127]:

Linear Model: Synthetic Example

- Run the sampler
- Ground truth
  - $\alpha = 2.5$
  - $\beta = 0.9$
- Recovered values
  - $\alpha = 2.12$
  - $\beta = 0.95$

[129]:
```
az.plot_trace(idata_g, var_names=["alpha", "beta", "sigma"]);
```

[128]:
```
az.summary(idata_g, var_names="alpha beta sigma".split(), kind="stats")
```

[128]:

	mean	sd	hdi_3%	hdi_97%
alpha	2.118	0.538	1.099	3.136
beta	0.946	0.053	0.845	1.046
sigma	0.476	0.034	0.412	0.538

SCIENCE ACADEMY

Linear Model: Bike Rental Example

- Model relationship between:
  - **Temperature** X
  - **Number of bikes rented** Y

- Intermediate variable is mean number of bikes rented for temperature X: $\mu = \alpha + \beta X$

$$Y \sim N(\mu, \sigma)$$

Linear Model: Bike Rental Example

- **Fit model** with PyMC:

$$\mu = 69 + 7.9X$$

- **Interpreting the model**
  - Temperature 0: expected rented bikes = 69
  - Each degree increase: expected rented bikes = +7.9
- **Parameters have uncertainty**
  - Posterior accounts for combined uncertainty
  - Bands represent quantiles [0.25, 0.75] and [0.03, 0.97] of prediction

- **Do you see any problem in the model?**

Linear Model: Bike Rental Example (Criticism)

- **Problems**
  1. Model outputs negative bike numbers
  2. Model predicts real numbers, but count is discrete
- **Root cause**: Using Gaussian likelihood:
  - Extends to negative numbers
  - Is continuous
- **Solutions**:
  - *Hack*: Clip predictions below 0 and discretize output
  - *Elegant*: Use model defined for discrete positive numbers

Count Data

- **Count data** when random variable is discrete, bounded at 0
  - E.g., number of rented bikes
  - Large numbers use continuous distribution
    - E.g., Gaussian
  - Sometimes modeled as discrete
    - E.g., Poisson, negative binomial

Generalized Liner Model (GLM)

- **Generalization of linear model** using different distributions for likelihood:

$$\begin{cases} \alpha \sim \textit{prior} \\ \beta \sim \textit{prior} \\ \mu = \alpha + \beta X \\ \theta \sim \textit{prior} \\ Y \sim \phi(f(\mu), \theta) \end{cases}$$

where:

- ϕ: arbitrary distribution
  - E.g., Normal, Student's t, negative binomial
- θ: auxiliary parameter for ϕ
  - E.g., σ for Normal
- $f()$: "inverse link function" transforming μ from real line to domain of ϕ

Poisson Distribution

- **Poisson distribution**
  - Models events in a fixed interval (time, space)
  - Assumes independent events at a constant rate
- **Examples**
  - *Call centers*: customers per hour
  - *Natural events*: earthquakes in a region
  - *Traffic flow*: cars through a toll booth per day
  - *Biology*: mutations in a DNA segment
- **Cons**
  - Assumes mean equals variance (can't model "overdispersion")

Negative Binomial Distribution

- **Negative binomial distribution**
  - Models failures/trials for fixed successes in IID Bernoulli trials
  - Generalizes geometric distribution, modeling trials to first success
  - Models overdispersion

$$\text{mean} \ll \text{variance}$$

- **E.g.,**
  - *Customer service*: unsuccessful interactions before success
  - *Sports*: games lost before winning fixed number of games

Overdispersion

- **Overdispersion** occurs when variance of count data is much larger than mean (e.g., 15-20x)
- **Examples**:
  - *Epidemiology*: modeling disease outbreak, number of new infections fluctuates widely
  - *Customer service*: large variation in daily complaints
- **How to model it?**
  - *Use Poisson*:
    - ❌: mean and variance must be equal
  - *Use Gaussian*:
    - Mismatch predicting negative rented bikes
    - Poor fit on positive side
  - *Use Negative Binomial*:
    - Better fit, though not perfect
    - Right tail predictions differ, but high demand probability is low
    - Second parameter controls variance
    - ✅: overall better than Normal model

Robust Regression

- **Outliers** pull regression line away from data
- **Robust regression** is a generalized linear model
  - Avoids feature transform/winsorization
- Student's t-distribution
  - Heavier tails than Normal
  - Reduces outlier impact

- Generalized Linear Models
  - Simple Linear Model
  - ***Logistic Regression***
  - Multiple Linear Regression

Logistic Regression

- **Logistic regression model**
  - Generalized linear model
  - Models binary response variable
    - E.g., ham/spam, safe/unsafe, cloudy/sunny, hotdog/not hotdog
- **Logistic model** is:

$$\begin{cases} \theta = logit(\alpha + \beta x) \\ y \sim Bernoulli(\theta) \end{cases}$$

- **Logistic function** (aka sigmoid)

$$logit(z) = \frac{1}{1 + \exp^{-z}}$$

- Convert real numbers from θ to [0, 1] for Bernoulli distribution

Iris Dataset

- Classical dataset of flower measurements from 3 Iris species
  - `setosa`, `virginica`, `versicolor`

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

- E.g., predict probability of a flower being `setosa` from `sepal_length`

Classification with Logistic Regression

- **Logistic regression**:
  - Computes probability of output being a certain value
  - Models $\theta = \Pr(Y = 1|X)$
  - Classifies output using decision rule, e.g.,

$$\Pr(Y = \texttt{versicolor} \mid \texttt{sepal_length}) > 0.5$$

- Same treatment as frequentist approach, has Bayesian flavor
  - Once Bayesian, generalize more
  - No gradient descent, solved with Bayesian inference
- **Classification with logistic regression might seem a misnomer**
  - **Regression** predicts continuous variable
  - **Classification** predicts discrete variable
  - Is "regression" since it computes probability of output being a certain value

Boundary Decision for a Classifier

- **Boundary decision** δ for a classifier are the values of independent variables making output probability 0.5
- E.g., for logistic regression δ:

$$
\begin{aligned}
0.5 &= \mathit{logit}(\alpha + \beta\delta) \\
0 &= \alpha + \beta\delta \\
\delta &= -\frac{\alpha}{\beta}
\end{aligned}
$$

  - Decision boundary uncertainty due to α and β uncertainty
- Probability threshold 0.5 for equal **misclassification risk**
  - Misclassification cost may not be symmetrical
  - Minimize false negatives
    - E.g., *"patient has disease, disease missed"*
  - Minimize false positives
    - E.g., *"patient doesn't have disease, predicted"*

Odds

- The **odds of event** "$y = 1$" is the ratio of favorable to unfavorable events

$$\text{odds} \triangleq \frac{\Pr(y = 1)}{1 - \Pr(y = 1)}$$

- Transformation between probability, odds, and log-odds
  - More intuitive in "gambling"
  - E.g., odds of rolling a 2 on a fair die are $\frac{1/6}{5/6} = \frac{1}{5} = 0.2$
    - One favorable event for 5 unfavorable events
- Interpreting **logistic regression in terms of odds**:
  - Since $\theta = \mathit{logit}(\alpha + \beta x)$ and $\theta = \Pr(y = 1)$:

$$\alpha + \beta x = \log\left(\frac{\Pr(y = 1)}{1 - \Pr(y = 1)}\right)$$

  - Logistic regression models log-odds linearly
  - β is the increase in log-odds for a unit change in x

Classification with Logistic Regression: Bayesian

```python
with pm.Model() as model_lrs:
    # Linear part.
    alpha = pm.Normal("alpha", mu=0, sigma=1)
    beta = pm.Normal("beta", mu=0, sigma=5)
    mu = alpha + x_c * beta
    # Sigmoid.
    theta = pm.Deterministic("theta", pm.math.sigmoid(mu))
    # Model.
    yl = pm.Bernoulli("yl", p=theta, observed=y_0)
    # Intercept?
    bd = pm.Deterministic("bd", - alpha / beta)
    #
    idata_lrs = pm.sample(random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta]
```

Sampling 4 chains, 0 divergences

Heteroskedasticity

- **Heteroskedasticity** is when variance of errors is not constant across observations
  - E.g., variance as a linear function of the dependent variable
- **Example**:
  - *"Baby height as a function of age"* is heteroskedastic
  - Mean μ as square root of a linear model
  - Standard deviation σ as a linear function of the predictor variable

Heteroskedasticity: Bayesian Model

```python
with pm.Model() as model_vv:
    # Create a shared variable so that the data can change after the model is created.
    x_shared = pm.Data("x_shared", data.month.values.astype(float))
    # Linear model for the mean is a function of sqrt(x).
    alpha = pm.Normal("alpha", sigma=10)
    beta = pm.Normal("beta", sigma=10)
    mu = pm.Deterministic("mu", alpha + beta * x_shared ** 0.5)
    # Linear model for the std dev.
    gamma = pm.HalfNormal("gamma", sigma=10)
    delta = pm.HalfNormal("delta", sigma=10)
    sigma = pm.Deterministic("sigma", gamma + delta * x_shared)
    # Fit.
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sigma, observed=data.length)
    #
    idata_vv = pm.sample(random_seed=123)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, gamma, delta]
```

- Generalized Linear Models
  - Simple Linear Model
  - Logistic Regression
  - ***Multiple Linear Regression***

Multiple Linear Regression

- In prediction problems, common to use **several independent variables**
  - E.g., student's grades = f(family income, mother's education, . . .)
- **Problem formulation**
  - k independent variables
  - N observations
  - Find a hyperplane of dimension k to explain data

$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k$$

  - Similar to polynomial regressions with independent variables

Multiple Regression: Synthetic Example 1/2

- Generate random data

```
np.random.seed(314)

N = 100
#N = 1000
alpha_real = 2.5
beta_real = [0.9, 1.5]
eps_stddev_real = 0.5
eps_real = np.random.normal(0, eps_stddev_real, size=N)

# Independent variables.
X = np.array([np.random.normal(i, j, N) for i, j in zip(
    # mean of gaussian.
    [10, 2],
    # std dev.
    [1, 1.5])]).T
X_mean = X.mean(axis=0, keepdims=True)
X_centered = X - X_mean

# Create samples.
y = alpha_real + np.dot(X, beta_real) + eps_real
```

- PyMC model

```
with pm.Model() as model_mlr:
    alpha_tmp = pm.Normal('alpha_tmp', mu=0, sigma=10)
    # Beta is a vector.
    beta = pm.Normal('beta', mu=0, sigma=1, shape=2)
    eps = pm.HalfCauchy('eps', 5)
    # mu.
    mu = alpha_tmp + pm.math.dot(X_centered, beta)
    # Extract alpha.
    alpha = pm.Deterministic('alpha', alpha_tmp - pm.math.dot(X_mean, beta))

    # Model.
    y_pred = pm.Normal('y_pred', mu=mu, sigma=eps, observed=y)

    idata_mlr = pm.sample(2000)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha_tmp, beta, eps]

Sampling 4 chains, 0 divergences
```

Multiple Regression: Synthetic Example 2/2

- Solve model

	mean	sd	hdi_3%	hdi_97%
alpha[0]	1.85	0.46	1.01	2.75
beta[0]	0.97	0.04	0.88	1.05
beta[1]	1.47	0.03	1.41	1.53
eps	0.47	0.03	0.41	0.54

Multiple Regression: Rented Bike Example 1/2

- **Assumption**: number of bike rented is function of temperature and hour of the day
- PyMC model

```
with pm.Model() as model_mlb:
    alpha = pm.Normal("alpha", mu=0, sigma=1)
    beta0 = pm.Normal("beta0", mu=0, sigma=10)
    beta1 = pm.Normal("beta1", mu=0, sigma=10)
    sigma = pm.HalfNormal("sigma", 10)
    mu = pm.Deterministic("mu", pm.math.exp(alpha + beta0 * bikes.temperature +
    beta1 * bikes.hour))
    _ = pm.NegativeBinomial("y_pred", mu=mu, alpha=sigma, observed=bikes.rented)
    #
    idata_mlb = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta0, beta1, sigma]

Sampling 4 chains, 0 divergences
```

Multiple Regression: Rented Bike Example 2/2

- Solve model

	mean	sd	hdi_3%	hdi_97%
alpha	3.55	0.13	3.32	3.82
beta0	0.04	0.01	0.03	0.05
beta1	0.08	0.01	0.06	0.10
sigma	1.11	0.08	0.97	1.25

Tutorial

- Generalized Linear Models