

Introduction to Data Science 1MS041

Group Assignment 2

Group 8

Course Co-ordinator

Benny Avelin

Group Members

Feruz Redi

Moutushi Sen

Samiha Akhter

Shaheryar

Tebogo Mitane



UPPSALA
UNIVERSITET

1. Consider a supervised learning problem where we assume that $Y | X$ is Poisson distributed. That is, the conditional density of $Y | X$ is given by

$$f_{Y|X}(y, x) = \frac{\lambda^y e^{-\lambda}}{y!}, \quad \lambda(x) = \exp(\alpha \cdot x + \beta).$$

Here α is a vector (slope) and β is a number (intercept). Follow the calculations from Section 4.2.1 to derive a loss that needs to be minimized with respect to α and β . Note: do we really need the factorial term?

Solution:

In this supervised learning problem, we assume that $Y | X$ follows a Poisson distribution, with the conditional probability mass function given by:

$$f_{Y|X}(y, x) = \frac{\lambda^y e^{-\lambda}}{y!}$$

where $\lambda(x) = \exp(\alpha \cdot x + \beta)$, with α being the slope (a vector) and β the intercept.

We aim to derive a loss function that needs to be minimized with respect to α and β . In this context, the natural approach is to use the negative log-likelihood of the Poisson distribution, which serves as the loss function.

Step 1: Log-likelihood function

For a dataset with n observations (x_i, y_i) , the likelihood function is:

$$L(\alpha, \beta) = \prod_{i=1}^n f_{Y|X}(y_i, x_i) = \prod_{i=1}^n \frac{\lambda(x_i)^{y_i} e^{-\lambda(x_i)}}{y_i!}$$

Where $\lambda(x_i) = \exp(\alpha \cdot x_i + \beta)$.

Taking the log of the likelihood function (log-likelihood):

$$\log L(\alpha, \beta) = \sum_{i=1}^n [y_i \log \lambda(x_i) - \lambda(x_i) - \log(y_i!)]$$

Substitute

$$\lambda(x_i) = \exp(\alpha \cdot x_i + \beta) \quad \log L(\alpha, \beta) = \sum_{i=1}^n [y_i(\alpha \cdot x_i + \beta) - \exp(\alpha \cdot x_i + \beta) - \log(y_i!)]$$

Step 2: Deriving the loss function

We minimize the negative log-likelihood to find the best estimates for α and β .

Ignoring the constant $\log(y_i!)$ term (because it doesn't depend on α and β doesn't affect the minimization), the negative log-likelihood (or the loss function) becomes:

$$L(\alpha, \beta) = - \log L(\alpha, \beta) = \sum_{i=1}^n [y_i(\alpha \cdot x_i + \beta) - \exp(\alpha \cdot x_i + \beta)]$$

Step 3: Final loss function

Thus, the loss function that needs to be minimized with respect to α and β is:

$$L(\alpha, \beta) = \sum_{i=1}^n [\exp(\alpha \cdot x_i + \beta) - y_i(\alpha \cdot x_i + \beta)]$$

The term $\log(y_i!)$ from the original log-likelihood comes from the normalization constant in the Poisson distribution. However, since it does not depend on the parameters α and β , it does not affect the optimization and can be safely ignored when deriving the loss function. Therefore, we do not need the factorial term in the final loss function.

Implementation:

```
import numpy as np
from scipy.optimize import minimize

# Define the loss function (negative log-likelihood for Poisson regression)
def poisson_loss(params, X, y):
    alpha = params[:-1] # Slope parameters (vector alpha)
    beta = params[-1]    # Intercept parameter (scalar beta)

    # Linear combination of input features with alpha and beta
    linear_term = np.dot(X, alpha) + beta

    # Poisson rate (lambda) is exp of the linear term
    lambdas = np.exp(linear_term)

    # Negative log-likelihood (ignoring the y! term as it's a constant)
    loss = np.sum(lambdas - y * linear_term)
    return loss
```

Dataset:

```
# Example dataset
# X is a matrix of features (n_samples, n_features)
X = np.array([
    [1, 2], # Example feature vectors
    [2, 1],
    [3, 4],
    [4, 3]
])

# y is the corresponding target values (n_samples)
y = np.array([3, 2, 6, 4])

# Initial guess for alpha (length n_features) and beta (intercept)
initial_params = np.zeros(X.shape[1] + 1)

# Use a minimization algorithm to find the optimal alpha and beta
result = minimize(poisson_loss, initial_params, args=(X, y), method='BFGS')

# Optimal parameters (alpha and beta)
optimal_params = result.x
alpha_opt = optimal_params[:-1]
beta_opt = optimal_params[-1]

print("Optimal alpha (slope):", alpha_opt)
print("Optimal beta (intercept):", beta_opt)
```

Output:

```
Optimal alpha (slope): [-0.02944575  0.37601937]
Optimal beta (intercept): 0.3760190612689197
```

2. Let X_1, \dots, X_n be IID from $\text{Uniform}(0, \theta)$. Let $\hat{\theta} = \max(X_1, \dots, X_n)$. First, find the distribution function of $\hat{\theta}$. Then compute the bias($\hat{\theta}$), se($\hat{\theta}$), and $\text{MSE}_n(\hat{\theta})$.

Solution:

1: Distribution Function of $\hat{\theta}$

The cumulative distribution function (CDF) of a uniform random variable $X_i \sim U(0, \theta)$ is:

$$F_{X_i}(x) = \left(\frac{x}{\theta}\right), 0 \leq x \leq \theta$$

The maximum of n IID uniform random variables $\hat{\theta} = \max(X_1, X_2, \dots, X_n)$ has the CDF:

$$F_{\hat{\theta}}(x) = P(\hat{\theta} \leq x) = P(X_1 \leq x, X_2 \leq x, \dots, X_n \leq x)$$

Since the variables are independent, this can be written as:

$$F_{\hat{\theta}}(x) = P(X_1 \leq x)^n = \left(\frac{x}{\theta}\right)^n, 0 \leq x \leq \theta$$

Thus the CDF of $\hat{\theta}$ is:

$$F_{\hat{\theta}}(x) = \left(\frac{x}{\theta}\right)^n, 0 \leq x \leq \theta$$

2. Computing the Bias of $\hat{\theta}$

$$\text{Bias}(\hat{\theta}) = E[\hat{\theta}] - \theta$$

To compute the bias, we first need to find $E[\hat{\theta}]$:

$$E[\hat{\theta}] = \int_0^{\theta} x f_{\hat{\theta}}(x) dx$$

We need to find PDF of $\hat{\theta}$:

$$f_{\hat{\theta}}(x) = \frac{d}{dx} \left(\frac{x}{\theta}\right)^n = \frac{nx^{n-1}}{\theta^n}, 0 \leq x \leq \theta$$

Now, we compute the expected value $E[\hat{\theta}]$:

$$\begin{aligned} E[\hat{\theta}] &= \int_0^{\theta} x f_{\hat{\theta}}(x) dx = \int_0^{\theta} x \frac{nx^{n-1}}{\theta^n} dx \\ E[\hat{\theta}] &= \frac{n}{\theta^n} \int_0^{\theta} x^n dx = \frac{n}{\theta^n} \cdot \left[\frac{x^{n+1}}{n+1} \right]_0^{\theta} = \frac{n}{\theta^n} \cdot \frac{\theta^{n+1}}{n+1} = \theta \frac{n}{n+1} \end{aligned}$$

The bias of $\hat{\theta}$ is:

$$\text{Bias}(\hat{\theta}) = E[\hat{\theta}] - \theta = \theta \frac{n}{n+1} - \theta = \theta \left(\frac{n}{n+1} - 1 \right) = \theta \left(\frac{n-(n+1)}{n+1} \right) = \theta \left(\frac{-1}{n+1} \right)$$

$$\text{Bias}(\hat{\theta}) = -\frac{\theta}{n+1}$$

3. Computing the Standard Error (SE) of $\hat{\theta}$

$$SE(\hat{\theta}) = \sqrt{Var(\hat{\theta})}$$

We first need to find $Var(\hat{\theta})$:

$$Var(\hat{\theta}) = E[\theta^2] - (E[\hat{\theta}])^2$$

$$E[\theta^2] = \int_0^{\theta} x^2 f_{\hat{\theta}}(x) dx = \int_0^{\theta} x^2 \frac{nx^{n-1}}{\theta^n} dx = \frac{n}{\theta^n} \int_0^{\theta} x^{n+1} dx = \frac{n}{\theta^n} \cdot \left[\frac{x^{n+2}}{n+2} \right]_0^{\theta} = \frac{n}{\theta^n} \cdot \frac{\theta^{n+2}}{n+2} = \frac{n\theta^2}{n+2}$$

$$Var(\hat{\theta}) = E[\theta^2] - (E[\hat{\theta}])^2 = \frac{n\theta^2}{n+2} - \left(\frac{n\theta}{n+1} \right)^2 = \frac{n\theta^2}{n+2} - \frac{n^2\theta^2}{(n+1)^2} = \theta^2 \left(\frac{n}{n+2} - \frac{n^2}{(n+1)^2} \right)$$

$$SE(\hat{\theta}) = \sqrt{\theta^2 \left(\frac{n}{n+2} - \frac{n^2}{(n+1)^2} \right)} = \theta \sqrt{\left(\frac{n}{n+2} - \frac{n^2}{(n+1)^2} \right)}$$

4. Computing the Mean Squared Error (MSE) of $\hat{\theta}$

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + (Bias(\hat{\theta}))^2$$

$$MSE(\hat{\theta}) = \theta^2 \left(\frac{n}{n+2} - \frac{n^2}{(n+1)^2} \right) + \left(-\frac{\theta}{n+1} \right)^2$$

$$MSE(\hat{\theta}) = \theta^2 \left[\left(\frac{n}{n+2} - \frac{n^2}{(n+1)^2} \right) + \frac{1}{(n+1)^2} \right] = \theta^2 \left(\frac{n}{n+2} - \frac{(n^2-1)}{(n+1)^2} \right)$$

$$MSE(\hat{\theta}) = \theta^2 \left(\frac{n}{n+2} - \frac{(n-1)(n+1)}{(n+1)^2} \right) = \theta^2 \left(\frac{n}{n+2} - \frac{(n-1)}{(n+1)} \right) = \theta^2 \left(\frac{(n^2+n) - (n^2+n-2)}{(n+2)(n+1)} \right)$$

$$MSE(\hat{\theta}) = \theta^2 \left(\frac{2}{(n+2)(n+1)} \right) = \frac{2\theta^2}{(n+2)(n+1)}$$

3. Consider the continuous distribution with density

$$p(x) = 1/2 \cos(x), \quad -\pi/2 < x < \pi/2.$$

- (a) Find the distribution function F. (b) Find the inverse distribution function F^{-1} . (c) To sample using an Accept-Reject sampler, Algorithm 1, we need to find a density g such that $p(x) \leq Mg(x)$ for some $M > 0$. Find such a density g and find the value of M.

Solution:

(a) Distribution function F (CDF), which is the integral of PDF p(x)

$$F(x) = \int_{-\pi/2}^x p(t)dt = \int_{-\pi/2}^x 1/2 \cos(t)dt$$

Integral computation:

$$\begin{aligned} F(x) &= 1/2 [\sin(x) - \sin(-\pi/2)] \\ &= 1/2 (\sin(x) + 1) \end{aligned}$$

$$\therefore \text{CDF is: } 1/2 (\sin(x) + 1)$$

(b) Inverse distribution function F^{-1}

$$\text{Let, } y = F(x)$$

$$\therefore x = F^{-1}(y)$$

Now from CDF,

$$y = 1/2(\sin(x) + 1)$$

$$2y = \sin(x) + 1$$

$$2y - 1 = \sin(x)$$

$$x = \sin^{-1}(2y - 1)$$

$$F^{-1}(y) = (2y - 1) \quad \text{where } 0 \leq y \leq 1$$

$$\therefore \text{Inverse distribution function is } F^{-1}(y) = (2y - 1)$$

(c) A simple density function g(x) and a constant M:

$$p(x) \leq M \cdot g(x) \quad -\pi/2 < x < \pi/2$$

Now, Uniform Distribution on $[-\pi/2, \pi/2]$ which has a constant density,

$$g(x) = 1/\pi \quad -\pi/2 < x < \pi/2$$

We know,

$$p(x) = 1/2 \cos(x)$$

$$\therefore \frac{p(x)}{g(x)} = \frac{1/2 \cos(x)}{1/\pi} = \pi/2 \cos(x)$$

The maximum value of $\cos(x)$ occurs when $x = 0$, where $\cos(0) = 1$

Thus, maximum value of $\pi/2 \cos(x)$ is: $M = \pi/2$

\therefore We can use $g(x) = 1/\pi$ and $M = \pi/2$ for the Accept-Reject Sampling.

4. Let Y_1, Y_2, \dots, Y_n be a sequence of IID discrete random variables, where

$P(Y_i = 0) = 0.1, P(Y_i = 1) = 0.3, P(Y_i = 2) = 0.2, P(Y_i = 3) = 0.4$. Let

$X_n = \max(Y_1, Y_2, \dots, Y_n)$. Let $X_0 = 0$ and verify that X_0, X_1, \dots, X_n is a Markov chain. Find the transition matrix **P**.

Solution:

Given a sequence of IID (independent and identically distributed) discrete random variables

Y_1, Y_2, \dots, Y_n where: $P(Y_i = 0) = 0.1, P(Y_i = 1) = 0.3, P(Y_i = 2) = 0.2, P(Y_i = 3) = 0.4$

We also define: $X_n = \max(Y_1, Y_2, \dots, Y_n)$ meaning X_n represents the maximum value among the first n observations of Y_i

$$X_0 = 0$$

Verify the Markov Property:

The Markov property means that the future state of the process (in this case, X_{n+1}) depends only on the current state (X_n) and not on any earlier states. We think about this in terms of the maximum function.

If $X_n = k$, the value of X_{n+1} can either remain k or increase depending on the value of Y_{n+1} .

- If $Y_{n+1} \leq k$, then $X_{n+1} = k$
- If $Y_{n+1} > k$, then $X_{n+1} = Y_{n+1}$.

Thus, X_{n+1} only depends on X_n and Y_{n+1} , which makes the process satisfy the Markov property.

Transition Probabilities

we calculate the transition probabilities for each possible value of X_n . The possible states of X_n are 0, 1, 2, and 3, because the values of Y_i can only be 0, 1, 2, or 3.

At: $X_n = 0$

If $X_n = 0$, then the maximum value so far is 0. The next state, X_{n+1} will depend on Y_{n+1} :

- $P(X_{n+1} = 0 \mid X_n = 0) = P(Y_{n+1} = 0) = 0.1$
- $P(X_{n+1} = 1 \mid X_n = 0) = P(Y_{n+1} = 1) = 0.3$
- $P(X_{n+1} = 2 \mid X_n = 0) = P(Y_{n+1} = 2) = 0.2$
- $P(X_{n+1} = 3 \mid X_n = 0) = P(Y_{n+1} = 3) = 0.4$

At: $X_n = 1$

If $X_n = 1$, the next state can either stay at 1 or increase. Thus:

- $P(X_{n+1} = 1 | X_n = 1) = P(Y_{n+1} \leq 1) = P(Y_{n+1} = 0) + P(Y_{n+1} = 1) = 0.1 + 0.3 = 0.4$
- $P(X_{n+1} = 2 | X_n = 1) = P(Y_{n+1} = 2) = 0.2$
- $P(X_{n+1} = 3 | X_n = 1) = P(Y_{n+1} = 3) = 0.4$

At: $X_n = 2$

If $X_n = 2$, the next state can either stay at 2 or increase. Thus:

- $P(X_{n+1} = 2 | X_n = 2) = P(Y_{n+1} \leq 2) = P(Y_{n+1} = 0) + P(Y_{n+1} = 1) + P(Y_{n+1} = 2) = 0.1 + 0.3 + 0.2 = 0.6$
- $P(X_{n+1} = 3 | X_n = 2) = P(Y_{n+1} = 3) = 0.4$

At: $X_n = 3$

If $X_n = 3$, then the maximum is already reached, and the next state can only stay at 3:

- $P(X_{n+1} = 3 | X_n = 3) = P(Y_{n+1} \leq 3) = 1$ (since all values of Y_{n+1} are less than or equal to 3).

The Transition Matrix

Now we construct the transition matrix P, where the rows represent the current state X_n and the columns represent the next state X_{n+1} :

$$P = \begin{bmatrix} 0.1 & 0.3 & 0.2 & 0.4 \\ 0.0 & 0.4 & 0.2 & 0.4 \\ 0.0 & 0.0 & 0.6 & 0.4 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

```
import numpy as np

# Transition probabilities
P_matrix = np.array([
    [0.1, 0.3, 0.2, 0.4], # X_n = 0
    [0.0, 0.4, 0.2, 0.4], # X_n = 1
    [0.0, 0.0, 0.6, 0.4], # X_n = 2
    [0.0, 0.0, 0.0, 1.0]  # X_n = 3
])
```


5. Let X_1, \dots, X_n be IID from some distribution F that is unknown. Let \widehat{F}_n be the empirical distribution function, use this to find an estimate of the quantile p of F . Use Theorem 5.28 to find a confidence interval for p .

Solution:

Firstly, we start by estimating the quantile p of F . We are given n IID observations from a distribution F . The empirical distribution function (EDF) \widehat{F}_n at a point x , which is defined as:

$$\widehat{F}_n(x) = \frac{1}{n} \sum_{i=1}^n I(X_i \leq x) \quad \because I=1 \text{ if } X_i \leq x \text{ and } 0 \text{ otherwise}$$

To find the p^{th} quantile of F , let's denote it as q_p , we have to find the smallest value x such that $\widehat{F}_n(x) \geq p$. The code implementation is as follows:

```
def empirical_quantile(data, p):
    """
    Calculate the empirical p-th quantile of data.
    """
    sorted_data = np.sort(data)
    index = int(np.ceil(p * len(data))) - 1
    return sorted_data[index]
```

Now for the second part, we use DKW inequality to find a confidence interval. The DKW Inequality provides a bound on the uniform convergence of the empirical distribution function to the true distribution function. For a given confidence level $1 - \alpha$, it states:

$$P(\sup_x |\widehat{F}_n(x) - F(x)| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

We solve for ϵ such that $2e^{-2n\epsilon^2} = \alpha$ to find the required confidence interval:

$$\epsilon = \sqrt{\frac{\log(2/\alpha)}{2n}}$$

This ϵ gives us the bounds around the quantile estimate we denoted by q_p . It would construct a confidence interval for p :

$$[q_p - \epsilon, q_p + \epsilon]$$

Here is the coded implementation of DKW confidence interval:

```
def dkw_confidence_interval(data, p, alpha=0.05):
    """
    Calculate the confidence interval for the p-th quantile using the DKW inequality with the corrected strict inequality.
    """
    n = len(data)
    epsilon = np.sqrt(np.log(2 / alpha) / (2 * n))
    q_p = empirical_quantile(data, p)

    #find indices for the quantile ± epsilon, ensuring they stay within bounds
    lower_index = max(0, int(np.floor((p - epsilon) * n)))
    upper_index = min(len(data) - 1, int(np.ceil((p + epsilon) * n)))

    sorted_data = np.sort(data)
    return sorted_data[lower_index], sorted_data[upper_index]
```

Example Implementation:

```
data = np.random.normal(0, 1, 1000) #simulate data from a Normal(0,1) distribution
p = 0.5 #our median
quantile_estimate = empirical_quantile(data, p)
confidence_interval = dkw_confidence_interval(data, p, alpha=0.05)

print(f"Estimated {p*100}% quantile: {quantile_estimate}")
print(f"Confidence interval for the quantile: {confidence_interval}")
```

Example's Output:

```
Estimated 50.0% quantile: 0.07291382223286276
Confidence interval for the quantile: (np.float64(-0.02433367759939726), np.float64(0.1804293189171668))
```