Exercise Sheet: Fisher Information Matrix

CDS120 - Uncertainty Quantification

Dr. Florian Herzog, Fachhochschule Graubünden

October 7, 2025

Instructions

This exercise sheet covers the theory and applications of the Fisher Information Matrix. Work through each problem systematically, showing all mathematical steps. Use the hints provided when you get stuck, but try to solve the problems independently first.

1 Distribution-Specific Examples

1.1 Exercise 1: Log-Normal Distribution

Let $X \sim \text{LogNormal}(\mu, \sigma^2)$, meaning $\log X \sim N(\mu, \sigma^2)$. The density is:

$$p(x; \mu, \sigma^2) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right), \quad x > 0$$

(a) Derive the log-likelihood function $\ell(\mu, \sigma^2; x)$.

(b) Compute the score vector $U(\mu, \sigma^2; x) = \left(\frac{\partial \ell}{\partial \mu}, \frac{\partial \ell}{\partial \sigma^2}\right)^T$.

(c) Calculate the Fisher Information Matrix $\mathcal{I}(\mu, \sigma^2)$ using either the outer form (covariance of score) or inner form (negative expected Hessian).

(d) What are the asymptotic standard errors of the MLE $(\hat{\mu}, \hat{\sigma}^2)$ for a sample of size n?

Hint: For the log-normal distribution, note that if $Y = \log X$, then $Y \sim N(\mu, \sigma^2)$. Use the transformation $y = \log x$ and the Jacobian $dx = x\, dy$. For the Fisher information, remember that $\mathbb{E}[\log X] = \mu$ and $\text{Var}(\log X) = \sigma^2$.

1.2 Exercise 2: Binomial Distribution

Let $X \sim \text{Binomial}(n, p)$ where n is known and $p \in (0, 1)$ is unknown. The PMF is:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n$$

(a) Write down the log-likelihood function $\ell(p; x)$ for a single observation.

(b) Derive the score function $U(p; x) = \frac{\partial \ell}{\partial p}$.

(c) Calculate the Fisher information $\mathcal{I}(p)$ using the outer form. Verify your result using the inner form.

(d) For m independent observations $X_1, \ldots, X_m$ from Binomial(n, p), what is the Fisher information $\mathcal{I}_m(p)$?

Hint: For the binomial distribution, recall that $\mathbb{E}[X] = np$ and $\text{Var}(X) = np(1-p)$. For the score function, you'll get terms involving X/p and $(n-X)/(1-p)$.

1.3 Exercise 3: Multi-Class Logistic Regression

Consider a multi-class logistic regression with $K = 4$ classes and p covariates. For observation i with covariates $\mathbf{x}_i \in \mathbb{R}^p$ and class label $y_i \in \{1, 2, 3, 4\}$, the model is:

$$P(Y_i = k|\mathbf{x}_i) = \frac{\exp(\mathbf{x}_i^T \boldsymbol{\beta}_k)}{\sum_{j=1}^4 \exp(\mathbf{x}_i^T \boldsymbol{\beta}_j)}$$

where $\boldsymbol{\beta}_k \in \mathbb{R}^p$ is the parameter vector for class k. For identifiability, set $\boldsymbol{\beta}_4 = \mathbf{0}$.
(a) Write the log-likelihood function for a single observation $(y_i, \mathbf{x}_i)$.
(b) Derive the score function (gradient) with respect to $\boldsymbol{\beta}_k$ for $k = 1, 2, 3$.
(c) Show that the Fisher Information Matrix has the block structure:

$$\mathcal{I}(\boldsymbol{\beta}) = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \otimes \mathbf{W}_i$$

where $\mathbf{W}_i$ is a 3×3 matrix and $\otimes$ denotes the Kronecker product.
(d) Derive the explicit form of the weight matrix $\mathbf{W}_i$ in terms of the predicted probabilities $\pi_{ik} = P(Y_i = k|\mathbf{x}_i)$.

Hint: For multi-class logistic regression, use indicator variables $I(y_i = k)$ in the log-likelihood. The score for $\boldsymbol{\beta}_k$ involves the difference between observed and predicted class probabilities. For the Fisher information, note that the Hessian involves second derivatives of the log-probabilities. The weight matrix $\mathbf{W}_i$ will be related to the covariance matrix of the multinomial distribution.

2 Python Implementation

2.1 Exercise 4: Log-Normal Distribution - Python/Jupyter Implementation

Implement the following analysis for the log-normal distribution using Python. Create a Jupyter notebook with the following structure:

Part A: Theoretical Implementation

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats, optimize
from scipy.stats import lognorm
import pandas as pd
from numdifftools import Hessian
import seaborn as sns

# Set random seed for reproducibility
np.random.seed(42)

# Define log-normal parameters
mu_true = 1.0
sigma_true = 0.7
```

```python
n_samples = 1000

# Generate sample data
# Your code here: generate log-normal data

# Define log-likelihood function
def log_likelihood(params, data):
    """
    Log-likelihood function for log-normal distribution
    params: [mu, sigma]
    data: observed data
    """
    # Your implementation here
    pass

# Define score function (gradient of log-likelihood)
def score_function(params, data):
    """
    Score function for log-normal distribution
    Returns gradient vector [d/dmu, d/dsigma]
    """
    # Your implementation here
    pass
```

Part B: Maximum Likelihood Estimation

```python
# Implement MLE estimation
def mle_lognormal(data):
    """
    Maximum likelihood estimation for log-normal distribution
    """
    # Your implementation here
    pass

# Compute MLE estimates
# Your code here

# Compare with analytical solution
# For log-normal: mu_hat = mean(log(X)), sigma_hat^2 = var(log(X))
```

Part C: Fisher Information Matrix

```python
# Analytical Fisher Information Matrix
def fisher_info_analytical(mu, sigma, n):
    """
    Analytical Fisher Information Matrix for log-normal distribution
    """
    # Your implementation here
    # I_11 = n/sigma^2
    # I_12 = I_21 = 0
    # I_22 = n/(2*sigma^4)
    pass

# Observed Information Matrix (negative Hessian)
def observed_information(params, data):
    """
    Compute observed information matrix using numerical Hessian
    """
    # Your implementation here using numdifftools
    pass

# Outer Product of Gradients (OPG)
def opg_estimator(params, data):
    """
    Compute OPG estimator of Fisher Information
    """
    # Your implementation here
    pass
```

Part D: Comparison and Visualization

```python
# Compare the three methods
# Your code here to compute and compare:
# 1. Analytical Fisher Information
# 2. Observed Information Matrix
# 3. OPG Estimator

# Compute standard errors
# SE = sqrt(diag(I^(-1)))

# Create visualization comparing the methods
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
# Your plotting code here
```

Part E: Monte Carlo Simulation

```python
# Monte Carlo simulation for coverage probability
def monte_carlo_simulation(n_sims=1000, n_samples=1000,
                           mu_true=1.0, sigma_true=0.7):
    """
    Monte Carlo simulation to verify coverage probability
    """
    # Your implementation here
    # For each simulation:
    # 1. Generate data
    # 2. Compute MLE
    # 3. Compute confidence intervals
    # 4. Check if true parameter is in CI
    pass

# Run simulation and report coverage
# Your code here
```

Questions to Answer in Your Implementation:

1. How do the three Fisher Information estimates compare? Which is most accurate?
2. What happens to the Fisher Information Matrix as σ increases? Interpret this result.
3. Verify that the coverage probability is approximately 95% for your confidence intervals.

4. Create plots showing the relationship between sample size and estimation precision.
5. Investigate the effect of different true parameter values on Fisher Information.

> **Hint:** For the log-normal distribution, the transformation $Y = \log X$ simplifies many calculations since $Y \sim N(\mu, \sigma^2)$. Use this relationship in your implementations. For numerical stability, work with log-likelihood rather than likelihood. The analytical Fisher Information Matrix for log-normal distribution is diagonal, which simplifies computations.

Submission Guidelines

1. Show all mathematical derivations clearly for Exercises 1-3
2. For Exercise 4, submit a complete Jupyter notebook with:
   - Well-commented Python code
   - Clear explanations of each step
   - Visualizations and interpretations
   - Answers to all questions
3. Provide interpretations of your numerical results
4. Due date: [To be announced in class]