Homework 2
Statistical Learning for Decision Making 2023

Li Ju
li.ju@it.uu.se

February 6, 2023

1 Problem 1

a). The asymptotic variance of the empirical risk minimizer $\hat{\tau}_n$ is given by

$$v_n = \dot{\boldsymbol{\tau}}_n^\top \mathbf{Q}_n^{-1} \mathbf{L}_n \mathbf{Q}_n^{-1} \dot{\boldsymbol{\tau}}_n$$

where $\mathbf{L}_n = \mathbb{E}_n[\dot{\boldsymbol{\ell}}\dot{\boldsymbol{\ell}}^\top]$, $\mathbf{Q}_n = \mathbb{E}_n[\ddot{\boldsymbol{\ell}}]$ and $\dot{\boldsymbol{\tau}}_n = \partial_\theta \tau(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_n}$. Then we have

$$\mathbf{Q}_n^{-1} \mathbf{L}_n \mathbf{Q}_n^{-1} = \mathbb{E}_n[(\hat{\boldsymbol{\theta}}_n - \mathbf{z})(\hat{\boldsymbol{\theta}}_n - \mathbf{z})^\top]$$
$$\dot{\boldsymbol{\tau}}_n = \frac{\hat{\boldsymbol{\theta}}_n - \mathbf{a}}{\|\hat{\boldsymbol{\theta}}_n - \mathbf{a}\|}$$

Then

$$v_n = \frac{(\hat{\boldsymbol{\theta}}_n - \mathbf{a})^\top}{\|\hat{\boldsymbol{\theta}}_n - \mathbf{a}\|^2} \mathbb{E}_n[(\hat{\boldsymbol{\theta}}_n - \mathbf{z})(\hat{\boldsymbol{\theta}}_n - \mathbf{z})^\top](\hat{\boldsymbol{\theta}}_n - \mathbf{a})$$

b). The reported confidence intervals $\mathcal{T}_\alpha^n$ are shown in Table 1. The code for the calculation is deferred to Appendix A.1.

n	$1-\alpha = 0.9$	$1-\alpha = 0.95$	$1-\alpha = 0.99$
100	[173.426, 180.065]	[172.790, 180.701]	[171.547, 181.944]
1000	[180.193, 182.373]	[179.984, 182.582]	[179.576, 182.990]

Table 1: Confidence intervals of $\hat{\tau}_n$ for different $1-\alpha$ and different n.

c). With $1-\alpha = 0.95$, by repeating the sampling $M = 10^4$ times, the coverage for the target quantity $\tau_\circ$ is 94.68% and 94.86% for sample size $n = 100$ and $n = 1000$ respectively. The code for the simulation is attached in Appendix A.2.

2 Problem 2

The Gaussian data model is given by

$$\begin{aligned} p_\theta(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{m}, v\mathbf{I}_d) \\ &= \frac{\exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{m})^\top v^{-1}\mathbf{I}_d(\mathbf{z}-\mathbf{m})\right)}{\sqrt{(2\pi)^d \det(v\mathbf{I}_d)}} \end{aligned}$$

The surprisal loss is given by

$$\begin{aligned}\ell_\theta(\mathbf{z}) &= -\ln p_\theta(\mathbf{z}) \\ &= \frac{1}{2}(\mathbf{z}-\mathbf{m})^\top v^{-1}\mathbf{I}_d(\mathbf{z}-\mathbf{m}) + \ln\sqrt{(2\pi)^d \det(v\mathbf{I}_d)} \\ &= \frac{\|\mathbf{z}-\mathbf{m}\|^2}{2v} + \frac{d}{2}\ln v + \underbrace{\frac{d}{2}\ln 2\pi}_{\text{constant}}\end{aligned}$$

The risk function is derived as follows:

$$\begin{aligned}L(\boldsymbol{\theta}) &= \mathbb{E}_\circ[\ell_\theta(\mathbf{z})] \\ &= \mathbb{E}_\circ[\frac{d}{2}\ln v + \frac{1}{2v}\|\mathbf{z}-\mathbf{m}\|^2] \\ &= \frac{d}{2}\ln v + \frac{1}{2v}(\mathrm{tr}(\mathbb{V}[\mathbf{z}]) + \mathbb{E}[\mathbf{z}]^\top\mathbb{E}[\mathbf{z}] + \mathbf{m}^\top\mathbf{m} - 2\mathbf{m}^\top\mathbb{E}[\mathbf{z}])\end{aligned}$$

The target parameters are given when the gradient of the risk function is $\mathbf{0}$ as follows:

$$\begin{aligned}\mathbf{0} = \frac{\partial L}{\partial \mathbf{m}}|_{\mathbf{m}=\mathbf{m}_\circ} &= \frac{1}{2v}(\mathbf{m}_\circ - 2\mathbb{E}_\circ[\mathbf{z}]) \\ \mathbf{m}_\circ &= \mathbb{E}_\circ[\mathbf{z}] \\ 0 = \frac{\partial L}{\partial v}|_{v=v_\circ} &= \frac{d}{2v_\circ} - \frac{1}{2v_\circ^2}(\mathrm{tr}(\mathbb{V}_\circ[\mathbf{z}]) + \underbrace{\mathbb{E}_\circ[\mathbf{z}]^\top\mathbb{E}_\circ[\mathbf{z}] + \mathbf{m}_\circ^\top\mathbf{m}_\circ - 2\mathbf{m}_\circ^\top\mathbb{E}_\circ[\mathbf{z}]}_{=0}) \\ v_\circ &= \frac{\mathrm{tr}(\mathbb{V}_\circ[\mathbf{z}])}{d}\end{aligned}$$

If we use the Gaussian model $\mathcal{N}(\mathbf{z};\mathbf{m}'+\boldsymbol{\delta}, v'\mathbf{I}_d)$ and $\boldsymbol{\theta} = [\begin{smallmatrix}\mathbf{m}' \\ \boldsymbol{\delta} \\ v'\end{smallmatrix}]$, we can prove that $\mathbf{m}_\circ = \mathbf{m}'_\circ + \boldsymbol{\delta}_\circ$ and $v_\circ = v'_\circ$, where $\boldsymbol{\Theta}'_\circ = [\begin{smallmatrix}\mathbf{m}'_\circ \\ \boldsymbol{\delta}_\circ \\ v'_\circ\end{smallmatrix}]$ denotes the new target parameters.

3 Problem 3

a). The surprisal loss function is derived as follows if we plug the parameterized distribution p_θ in:

$$\begin{aligned}\ell_\theta(\mathbf{z}) = -\ln p_\theta(\mathbf{z}) &= -(1-s)\ln(\mathrm{Poisson}(c;\lambda_0)) - s\ln(\mathrm{Poisson}(c;\lambda_1)) \\ &= -(1-s)(-\lambda_0 + c\ln\lambda_0 - \ln c!) - s(-\lambda_1 + c\ln\lambda_1 - \ln c!) \\ &= -(1-s)(-\lambda_0 + c\ln\lambda_0) - s(-\lambda_1 + c\ln\lambda_1)\end{aligned}$$

With the surprisal loss function, by taken the expectation over p_θ, we have the Risk function:

$$\begin{aligned}L(\theta) &= \mathbb{E}_{p_\theta}[\ell_\theta(\mathbf{z})] \\ &= \mathbb{E}_{p_\theta(c|s=0)}[\ell_\theta(c|s=0)]P(s=0) + \mathbb{E}_{p_\theta(c|s=1)}[\ell_\theta(c|s=1)]P(s=1) \\ &= \mathbb{E}_{p_\theta(c|s=0)}[(\lambda_0 - c\ln\lambda_0)] + \mathbb{E}_{p_\theta(c|s=1)}[(\lambda_1 - c\ln\lambda_1)] \\ &= \lambda_0 - \lambda_0\mathbb{E}[c|s=0] + \lambda_1 - \lambda_1\mathbb{E}[c|s=1]\end{aligned}$$

The target parameters $\boldsymbol{\theta}_\circ$ are taken when the gradient of the Risk function is 0:

$$\begin{aligned}\mathbf{0} &= [\frac{\partial L(\theta)}{\partial\lambda_0}, \frac{\partial L(\theta)}{\partial\lambda_1}]^\top \\ &= [1 - \frac{\mathbb{E}[c|s=0]}{\lambda_{\circ,0}}, 1 - \frac{\mathbb{E}[c|s=1]}{\lambda_{\circ,1}}]^\top \\ \boldsymbol{\theta}_\circ &= \begin{bmatrix}\mathbb{E}[c|s=0] \\ \mathbb{E}[c|s=1]\end{bmatrix}\end{aligned}$$

b). The asymptotic variance of $\hat{\tau}_n$ is given as follows:

$$\mathbb{V}[\hat{\tau}_n] = \dot{\tau}_n \mathbf{Q}_n^{-1} \mathbf{L}_n \mathbf{Q}_n^{-1} \dot{\tau}_n^\top$$

where

$$\begin{aligned}
\dot{\tau}_n &= \frac{\partial \tau}{\partial \theta}|_{\theta=\hat{\theta}_n} = [-1, 1]^\top \\
\mathbf{Q}_n &= \mathbb{E}_n[\frac{\partial^2 \ell}{\partial^2 \theta}|_{\theta=\hat{\theta}_n}] = \mathbb{E}_n \begin{bmatrix} \frac{c(1-s)}{\hat{\lambda}_{n,0}^2} & 0 \\ 0 & \frac{cs}{\hat{\lambda}_{n,1}^2} \end{bmatrix} \\
\mathbf{L}_n &= \mathbb{E}_n[\frac{\partial \ell}{\partial \theta}(\frac{\partial \ell}{\partial \theta})^\top|_{\theta=\hat{\theta}_n}] \\
&= \mathbb{E}_n \begin{bmatrix} (1-s)^2(1-\frac{c}{\lambda_{n,0}})^2 & 0 \\ 0 & s^2(1-\frac{c}{\lambda_{n,1}})^2 \end{bmatrix}
\end{aligned}$$

Thus, with the code appended in Appendix A.3, the intervals with different confidence levels of the experiment are listed in Table 2.

n	$1-\alpha = 0.9$	$1-\alpha = 0.95$	$1-\alpha = 0.99$
100	[17.070, 22.150]	[16.584, 22.636]	[15.633, 23.587]

Table 2: Confidence intervals of $\hat{\tau}_n$ for different $1-\alpha$.

4 Problem 4

a). The risk function of the problem is in the identical form of the empirical risk function in Problem 2 and is shown as follows:

$$\begin{aligned}
L(\theta)_n &= \mathbb{E}_n[\ell(\mathbf{z})_\theta] \\
&= \mathbb{E}_n[\frac{d}{2}\ln v + \frac{1}{2v}\|\mathbf{z}-\mathbf{m}\|^2] \text{ where } \mathbf{m} = 2c^{-1}\boldsymbol{\mu}(\mathbf{s})
\end{aligned}$$

The minimizer of $\mathbf{s}$ is given as follows:

$$\begin{aligned}
\hat{\mathbf{s}}_n &= \arg\min_{\mathbf{s}}(\mathbb{E}_n[\frac{d}{2}\ln v] + \mathbb{E}_n[\frac{1}{2v}\|\mathbf{z}-2c^{-1}\boldsymbol{\mu}(\mathbf{s})\|^2]) \\
&= \arg\min_{\mathbf{s}} \mathbb{E}_n[\|\mathbf{z}-2c^{-1}\boldsymbol{\mu}(\mathbf{s})\|^2]
\end{aligned}$$

b). The gradient of the empirical risk function is given as follows:

$$\begin{aligned}
\nabla L_n(\mathbf{s}) &= \frac{\partial L_n}{\partial \mathbf{m}}\frac{\partial \mathbf{m}}{\partial \mathbf{s}} \\
&= 2c^{-1}\mathbb{E}_n[2(2c^{-1}\boldsymbol{\mu}(\mathbf{s})-\mathbf{z}) \begin{bmatrix} \frac{(\mathbf{s}-\mathbf{a_1})^\top}{\|\mathbf{s}-\mathbf{a_1}\|} \\ \frac{(\mathbf{s}-\mathbf{a_2})^\top}{\|\mathbf{s}-\mathbf{a_2}\|} \\ \frac{(\mathbf{s}-\mathbf{a_3})^\top}{\|\mathbf{s}-\mathbf{a_3}\|} \end{bmatrix}]
\end{aligned}$$

With gradient descent, the estimated $\hat{s}_n$ from the empirical risk function is $(199.810, 199.776)$. The contour of the empirical risk function together with GD steps are shown in Figure 1. The code for the optimization is appended in Appendix A.4.

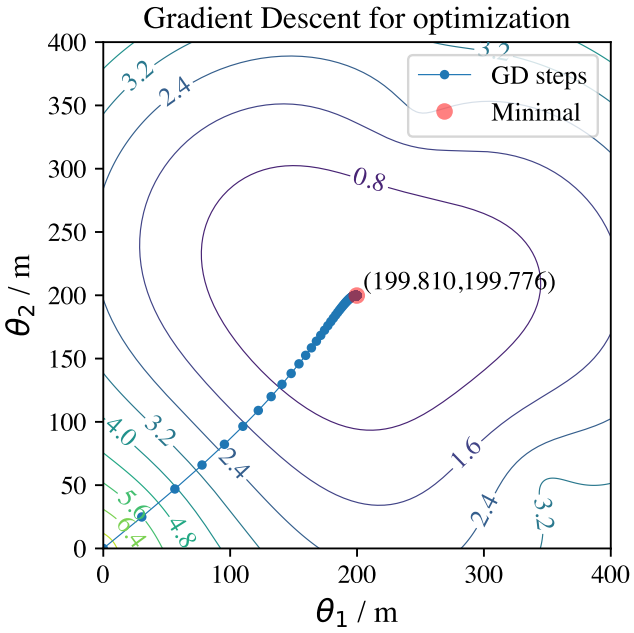

Figure 1: Contour of the empirical risk function and GD steps

A Problem 1

A.1 Code for 1.b

```
import numpy as np
import scipy.stats as st


a = np.array([50, 100])
ns = [100, 1000]
conf_levs = [0.9, 0.95, 0.99]

mu_z = [200, 200]
cov_z = [[400, 50], [50, 400]]

for n in ns:
    # sampling
    zs = np.random.multivariate_normal(mu_z, cov_z, n)
    # the point estimation of theta
    theta_n = np.sum(zs, 0)/zs.shape[0]
    # the variance of £\sqrt{n}(\hat{\theta} - \theta_\circ)£
    v_theta = (zs - theta_n).transpose() @ (zs - theta_n) / zs.shape[0]
    # the variance of £\sqrt{n}(\hat{\tau} - \tau_\circ)£
    v_n = 1/np.linalg.norm(theta_n - a) ** 2 * \
        (theta_n - a) @ v_theta @ (theta_n - a).transpose()
    for conf_lev in conf_levs:
        # calculate confidence interval with normal distribution
        intv = st.norm.interval(confidence=conf_lev,
                                loc=np.linalg.norm(theta_n - a),
                                scale=np.sqrt(v_n/n))
```

```
        print(f"{n} samples with confidence level {conf_lev}: "
              f"({intv[0]:.3f}, {intv[1]: .3f})")
```

A.2 Code for 1.c

```
import numpy as np
import scipy.stats as st


a = np.array([50, 100])
ns = [100, 1000]
conf_lev = 0.95

mu_z = [200, 200]
cov_z = [[400, 50], [50, 400]]

tau_o = np.linalg.norm(np.array(mu_z) - a)
M = int(1e4)
for n in ns:
    covered = 0
    for _ in range(M):
        # sampling
        zs = np.random.multivariate_normal(mu_z, cov_z, n)
        # the point estimation of theta
        theta_n = np.sum(zs, 0)/zs.shape[0]
        # the variance of £\sqrt{n}(\hat{\theta} - \theta_\circ)£
        v_theta = (zs - theta_n).transpose() @ (zs - theta_n) / zs.shape[0]
        # the variance of £\sqrt{n}(\hat{\tau} - \tau_\circ)£
        v_n = 1/np.linalg.norm(theta_n - a) ** 2 * \
            (theta_n - a) @ v_theta @ (theta_n - a).transpose()
            # calculate confidence interval with normal distribution
        intv = st.norm.interval(confidence=conf_lev,
                                loc=np.linalg.norm(theta_n - a),
                                scale=np.sqrt(v_n/n))
        if tau_o >= intv[0] and tau_o <= intv[1]:
            covered += 1
        # else:
        #     print("NOT IN!")
    print(f"Sample size {n}: Coverage: {covered/M}")
```

A.3 Code for 3.b

```
from numpy.random import poisson
import numpy as np
import scipy.stats as st

n = 200
m = n//2

lambda0 = 100
lambda1 = 120

s0_samples = poisson(lambda0, m)
s1_samples = poisson(lambda1, m)
```

```
mean0 = np.mean(s0_samples)
mean1 = np.mean(s1_samples)
tau_hat = mean1 - mean0

# inverse of Qn
Qninv = np.array([[n*(mean0**2)/np.sum(s0_samples), 0],
                  [0, n*(mean1**2)/np.sum(s1_samples)]])

# Ln
Ln = np.array([[np.sum((1-s0_samples/mean0)**2)/n, 0],
                  [0, np.sum((1-s1_samples/mean1)**2)/n]])

# tau_dot
taudot = np.array([-1, 1])

# variance of \sqrt{n}\hat{\tau}
v_n = taudot.transpose() @ Qninv @ Ln @ Qninv @ taudot

conf_levs = [0.9, 0.95, 0.99]
for conf_lev in conf_levs:
    intv = st.norm.interval(confidence=conf_lev,
                            loc=np.linalg.norm(tau_hat),
                            scale=np.sqrt(v_n/n))
    print(f"[{intv[0]: .3f}, {intv[1]: .3f}]"
          f" with confidence level {conf_lev}")
```

A.4 Code for 4.b

```
import numpy as np
import matplotlib.pyplot as plt

a1 = np.array([0, 0])
a2 = np.array([350, 50])
a3 = np.array([250, 350])
scirc = np.array([200, 200])
v = (1e-7)**2
c = 3e8
eps = 1e-7

n = 100

mean = 2/c*np.linalg.norm(scirc - np.array([a1, a2, a3]), axis=1)
cov = v * np.identity(3)

z = np.random.multivariate_normal(mean, cov, n)


def risk(z, s):
    mu = np.array([
        np.linalg.norm(s - a1),
        np.linalg.norm(s - a2),
        np.linalg.norm(s - a3),
    ])
    values = np.sum((z - 2/c*mu) ** 2, axis=1)
    return values.mean()
```

```
def grad(z, s):
    mu = np.array([
        np.linalg.norm(s - a1),
        np.linalg.norm(s - a2),
        np.linalg.norm(s - a3),
    ])
    # print(mu)
    gradient = 2*(2/c*mu - z)
    gradient = 2/c * gradient @ \
        np.array([(s - a1)/(mu[0] + eps),
                  (s - a2)/(mu[1] + eps),
                  (s - a3)/(mu[2] + eps)])
    gradient = np.mean(gradient, axis=0)
    return gradient


lr = 1e14
steps = 2000

theta = np.array([0, 0])
thetas = []
for i in range(steps):
    if not i % 10:
        thetas.append(theta)
    gradient = grad(z, theta)
    theta = theta - lr * gradient

thetas = np.array(thetas)
print(thetas)

# plot GD steps
minv, maxv = 0, 400
eval_nums = 400

s1, s2 = np.meshgrid(np.linspace(minv, maxv, eval_nums),
                     np.linspace(minv, maxv, eval_nums))

ss = np.array(list(zip(s1.flatten(), s2.flatten())))

values = np.array(list(map(lambda s: risk(z, s), ss))).reshape((eval_nums, -1))

fig, ax = plt.subplots(figsize=(4, 4))
ax.grid(False)
CS = ax.contour(s1, s2, values, levels=10, linewidths=0.5)
ax.plot(thetas[:-1, 0], thetas[:-1, 1], '-o',
        markersize=3, linewidth=0.5, label="GD steps")
opt_point = thetas[-1]
ax.annotate(f'({opt_point[0]:.3f},{opt_point[1]:.3f})', xy=opt_point+5)
ax.plot(opt_point[0], opt_point[1], 'ro', alpha=0.5, label="Minimal")
ax.set_title("Gradient Descent for optimization")
ax.set_xlabel(r"$\theta_1$ / m")
ax.set_ylabel(r"$\theta_2$ / m")
ax.clabel(CS)
```

```
ax.legend()
fig.tight_layout()
fig.savefig("hw2_4b.pdf", dpi=500)
```