Homework 6
Statistical Learning for Decision Making 2023

Li Ju
li.ju@it.uu.se

August 9, 2023

1 Problem 1

With the policy class Π_w indexed by w, the risk of the decision process $p^\pi(\mathbf{x}, a, y)$ is then defined as

$$L(\pi, \phi) = \iiint \ell(\mathbf{x}, a, y) p^\pi(\mathbf{x}, a, y) d\mathbf{x}\, dy\, da$$

The directed acyclic graph (DAG) of the decision process is shown as Figure 1. Thus, the risk of the decision process can be causally factorized as

$$\begin{aligned} L(\pi, \phi) &= \iiint \ell(\mathbf{x}, a, y) p(y|a, \mathbf{x}) p^\pi(a|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}\, dy\, da \\ &= \iiint y \cdot p(y|a, \mathbf{x}) p^\pi(a|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}\, dy\, da \end{aligned}$$

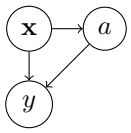

Figure 1: Directed acyclic graph of the decision process $p^\pi(\mathbf{x}, a, y)$

With $m_0 = 1000$ samples, the risk of different policies indexed by different values of w can be estimated. By plotting the risks of different policies against the values of w, Figure 2 is obtained. The code for the problem is deferred to Appendix A.1. It is observed that in the policy class Π_w, there exists one optimal policy with the least risk.

2 Problem 2

a). The directed acyclic graph (DAG) of the decision process is shown as Figure 3. To use data from randomized sampling process to estimate $L(\pi, \phi)$ with point-identifiability, following assumpions are required:

- $p(x) = \tilde{p}(\mathbf{x})$
- $p(y|a, \mathbf{x}) = \tilde{p}(y|a, \mathbf{x})$

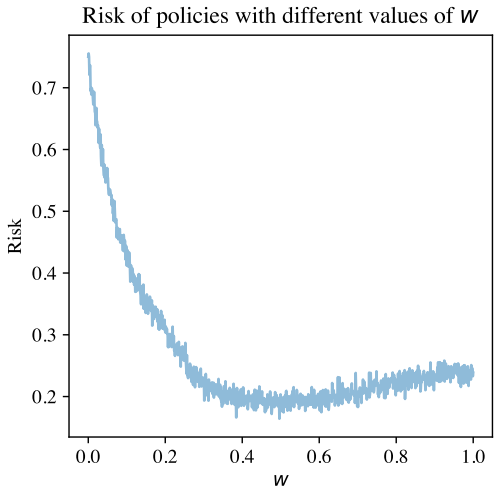

Figure 2: Plots of risk of different policies indexed by different values of w.

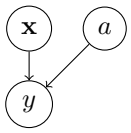

Figure 3: Directed acyclic graph of the randomized trail process $p^{\pi}(\mathbf{x}, a, y)$

b). With the two assumptions, we have following derivations:

$$\begin{aligned} L(\pi_w, \phi) &= \iiint \ell(\mathbf{x}, a, y) \cdot p^{\pi}(\mathbf{x}, a, y) d\mathbf{x}\, dy\, da \\ &= \iiint \ell(\mathbf{x}, a, y) \cdot \tilde{p}(\mathbf{x}) p(a|\mathbf{x}) \tilde{p}(y|a, \mathbf{x}) d\mathbf{x}\, dy\, da \\ &= \iiint \ell(\mathbf{x}, a, y) \frac{p(a|\mathbf{x})}{\tilde{p}(a)} \cdot \tilde{p}(\mathbf{x}) \tilde{p}(a) \tilde{p}(y|a, \mathbf{x}) d\mathbf{x}\, dy\, da \\ &\approx \sum_{i=1}^{n} y_i \frac{p(a_i|\mathbf{x}_i)}{\tilde{p}(a_i)} \end{aligned}$$

With different number of samples $n \in \{10, 100, 1000\}$, we estimate the risk of different policies indexed by different values of w. The implementation is deferred to Appendix A.2.1.

Compared with Figure 2, it can be observed that with randomized trial data, risks of different policies can be estimated. However, with small sample size, the estimation has large variation. With the increase of sample sizes, the risk estimation become more accurate.

c). With a different trial population $\tilde{p}(\mathbf{x}) : \mathbf{x} \sim \mathcal{U}(0, 2/3)^2$, the code from Problem 2b is repeated and the risk estimation is plotted in Figure 5.

Compared with Figure 2, it is observed that the estimated risks are biased. This is caused by the violation of the assumption $p(\mathbf{x}) = \tilde{p}(\mathbf{x})$.

3 Problem 3

a). To use observational data to estimate $L(\pi, \phi)$ with point-identifiability, following assumpions are required:

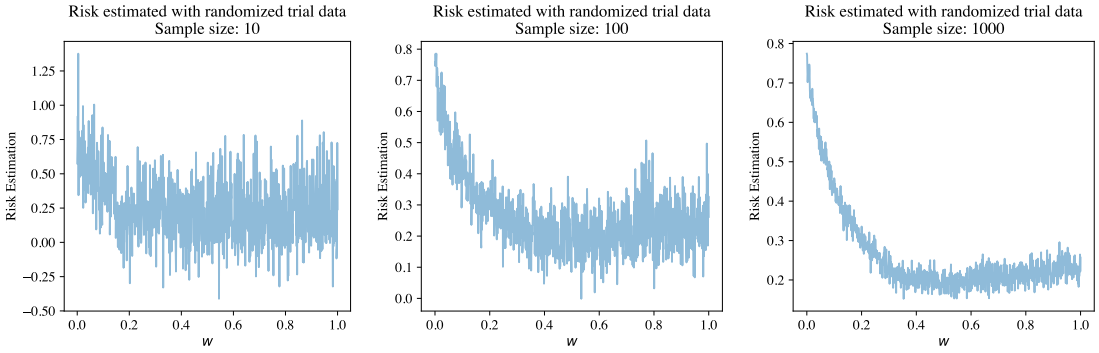

Figure 4: Risk estimated with randomized trial data with different sample sizes. The trial samples follows $\tilde{p}(\mathbf{x}) : \mathbf{x} \sim \mathcal{U}(0,1)^2$.

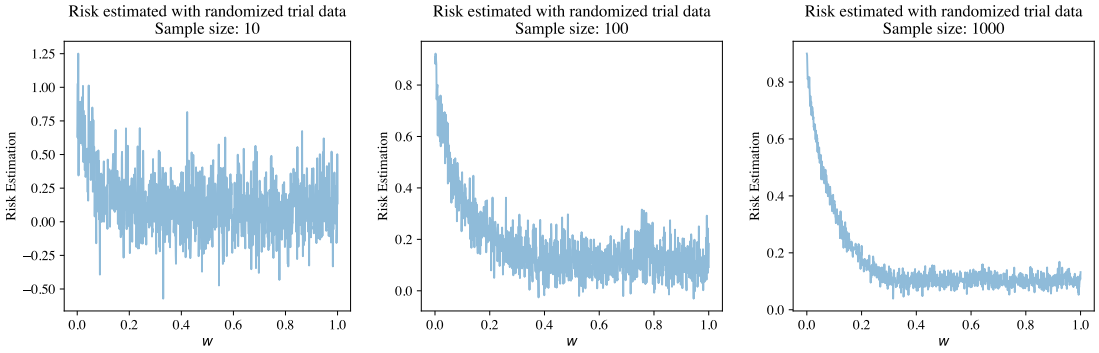

Figure 5: Risk estimated with randomized trial data with different sample sizes. The trial samples follows $\tilde{p}(\mathbf{x}) : \mathbf{x} \sim \mathcal{U}(0,2/3)^2$.

- $p(x) = \tilde{p}(\mathbf{x})$
- $p(y|a,\mathbf{x}) = \tilde{p}(y|a,\mathbf{x})$

b). With the two assumptions, we have $L(\pi_w;\phi) = \tilde{\mathbb{E}}[\frac{p^\pi(a|\mathbf{x})}{\tilde{p}(a|\mathbf{x})}y]$.

With the logistic actions

$$\tilde{p}(a|\mathbf{x}) = \begin{cases} s(\frac{1}{2}(x_1x_2+1)) & a=0 \\ 1-s(\frac{1}{2}(x_1x_2+1)) & a=1 \end{cases}$$

we can estimate risks of different policies indexed by w with sample sizes of $n \in \{100, 1000\}$. The estimated risks are plotted in Figure 6. It is observed that with logistic actions, the risks can be estimated accurate with large sample size. However, it should be noticed that current practice restrict to low-dimensional action space $\mathcal{A}$ and covariate space $\mathcal{X}$. The implementation code for the problem is deferred in Appendix A.3.1.

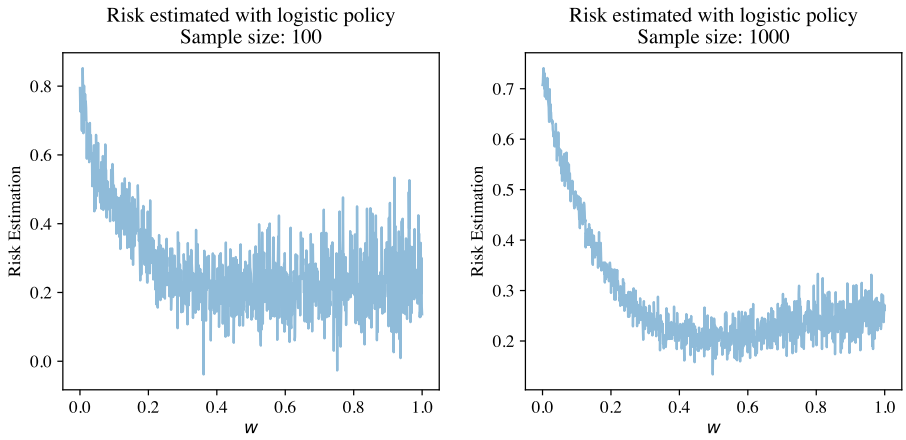

Figure 6: Risk estimated with logistic actions with different sample sizes. The trial samples follows $\tilde{p}(\mathbf{x}) : \mathbf{x} \sim \mathcal{U}(0,1)^2$.

A Codes

A.1 Code for Problem 1

```
import jax.numpy as jnp
import matplotlib.pyplot as plt
import jax

# sample size
m = 1000

key = jax.random.PRNGKey(0)

key, _ = jax.random.split(key)

# generate random x
xs = jax.random.uniform(key, shape=(m, 2))


# policy function
def get_a(x, w):
    return x[0] * x[1] < w


# vectorize the policy function
get_as = jax.vmap(get_a, in_axes=(0, None))


# function to sample y condition on x and a from gaussian noise
def sample_y(x, a, noise):
    offset = jax.lax.cond(a, lambda _: x[0]*x[1], lambda _: 1-x[0]*x[1], None)
    return noise * jnp.sqrt(0.1) + offset


# vectorize the function
sample_ys = jax.vmap(sample_y, in_axes=(0, 0, 0))
```

```
w_space = jnp.linspace(0, 1, 1000)

risks = []
for w in w_space:
    # sample actions
    actions = get_as(xs, w)

    # get random key to generate noise
    key, _ = jax.random.split(key)

    noises = jax.random.normal(key, shape=(m,))
    # sample ys
    ys = sample_ys(xs, actions, noises)

    # compute risk
    risks.append(ys.mean())

# plot the results
fig, ax = plt.subplots(figsize=(4, 4))
ax.grid(False)
ax.plot(w_space, risks, alpha=0.5)

ax.set_title(r"Risk of policies with different values of $w$")
ax.set_xlabel(r"$w$")
ax.set_ylabel("Risk")

# ax.legend()
fig.savefig("hw6_1.pdf", dpi=500)
```

A.2 Code for Problem 2

A.2.1 Problem 2b

```
import jax.numpy as jnp
import matplotlib.pyplot as plt
import jax

# sample size
ns = [10, 100, 1000]
action_prob = 0.3

key = jax.random.PRNGKey(0)

key, _ = jax.random.split(key)


# function to sample y condition on x and a from gaussian noise
def sample_y(x, a, noise):
    offset = jax.lax.cond(a, lambda _: x[0]*x[1], lambda _: 1-x[0]*x[1], None)
    return noise * jnp.sqrt(0.1) + offset


# vectorize the function
```

```
sample_ys = jax.vmap(sample_y, in_axes=(0, 0, 0))


# risk function
def risk(x, w, a, y):
    pax = jax.lax.cond((x[0]*x[1] < w) == a, lambda _: 1., lambda _: 0., None)
    pa = jax.lax.cond(a, lambda _: action_prob, lambda _: 1-action_prob, None)
    return y * pax / pa


# vectorize the function
risk_vec = jax.vmap(risk, in_axes=(0, None, 0, 0))

for n in ns:

    # generate random x
    xs = jax.random.uniform(key, shape=(n, 2))

    w_space = jnp.linspace(0, 1, 1000)

    risks = []
    for w in w_space:
        # sample actions
        key, _ = jax.random.split(key)
        actions = jax.random.bernoulli(key, p=action_prob, shape=(n,))

        # get random key to generate noise
        key, _ = jax.random.split(key)
        noises = jax.random.normal(key, shape=(n,))
        # sample ys
        ys = sample_ys(xs, actions, noises)

        # compute risk
        risks.append(risk_vec(xs, w, actions, ys).mean())

    # plot the results
    fig, ax = plt.subplots(figsize=(4, 4))
    ax.grid(False)
    ax.plot(w_space, risks, alpha=0.5)

    ax.set_title(f"Risk estimated with randomized trial data\nSample size: {n}")
    ax.set_xlabel(r"$w$")
    ax.set_ylabel("Risk Estimation")

    # ax.legend()
    fig.savefig(f"hw6_2_{n}.pdf", dpi=500)
    plt.close(fig)
```

A.3 Code for Problem 3

A.3.1 Problem 3b

```
import jax.numpy as jnp
import matplotlib.pyplot as plt
import jax
```

```python
# sample size
ns = [100, 1000]

key = jax.random.PRNGKey(0)

key, _ = jax.random.split(key)


def get_action(x, key):
    prob = 1 - jax.nn.sigmoid(x[0] * x[1] + 1)
    return jax.random.bernoulli(key, p=prob)


get_actions = jax.vmap(get_action, in_axes=(0, 0))


# function to sample y condition on x and a from gaussian noise
def sample_y(x, a, noise):
    offset = jax.lax.cond(a, lambda _: x[0]*x[1], lambda _: 1-x[0]*x[1], None)
    return noise * jnp.sqrt(0.1) + offset


# vectorize the function
sample_ys = jax.vmap(sample_y, in_axes=(0, 0, 0))


# risk function
def risk(x, w, a, y):
    pax = jax.lax.cond((x[0]*x[1] < w) == a, lambda _: 1., lambda _: 0., None)
    a0_prob = jax.nn.sigmoid(x[0] * x[1] + 1)
    tilde_pax = jax.lax.cond(a, lambda _: 1-a0_prob, lambda _: a0_prob, None)
    return y * pax / tilde_pax


# vectorize the function
risk_vec = jax.vmap(risk, in_axes=(0, None, 0, 0))

for n in ns:

    # generate random x
    xs = jax.random.uniform(key, shape=(n, 2))

    w_space = jnp.linspace(0, 1, 1000)

    risks = []
    for w in w_space:
        # sample actions
        keys = jax.random.split(key, n+1)
        key = keys[-1]
        actions = get_actions(xs, keys[:n])

        # get random key to generate noise
        key, _ = jax.random.split(key)
        noises = jax.random.normal(key, shape=(n,))
```

```
    # sample ys
    ys = sample_ys(xs, actions, noises)

    # compute risk
    risks.append(risk_vec(xs, w, actions, ys).mean())

# plot the results
fig, ax = plt.subplots(figsize=(4, 4))
ax.grid(False)
ax.plot(w_space, risks, alpha=0.5)

ax.set_title(f"Risk estimated with logistic policy\nSample size: {n}")
ax.set_xlabel(r"$w$")
ax.set_ylabel("Risk Estimation")

# ax.legend()
fig.savefig(f"hw6_3_{n}.pdf", dpi=500)
plt.close(fig)
```