Homework 5
Statistical Learning for Decision Making 2023

Li Ju
li.ju@it.uu.se

July 3, 2023

1 Problem 1

a). With the zero-one loss $\ell(y, a) = \mathbb{1}(y \neq a)$, the risk-minimizing predictor is derived as follows:

$$\begin{aligned}
\pi_\circ(\mathbf{x}) \coloneqq a_\circ &= \arg\min_{a\in\mathcal{Y}} L(\mathbf{x}, y, a) \\
&= \arg\min_{a\in\mathcal{Y}} \int \ell(y, a) p^\pi(\mathbf{x}, y, a) d\mathbf{x} dy da \\
&= \arg\min_{a\in\mathcal{Y}} \int \left[\sum_{a=0,1} \sum_{y=0,1} \ell(y, a) p(a|\mathbf{x}) p(y|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x} \\
&= \arg\min_{a\in\mathcal{Y}} \left[\mathbb{1}(a = 1) p(y = 0|\mathbf{x}) + \mathbb{1}(a = 0) p(y = 1|\mathbf{x}) \right] \\
&= \arg\min_{a\in\mathcal{Y}} p(y \neq a|\mathbf{x}) \\
&= \arg\max_{a\in\mathcal{Y}} p(y = a|\mathbf{x}) p(\mathbf{x}) = \arg\max_{a\in\mathcal{Y}} p(\mathbf{x}, y = a) \\
&= \arg\max_{a\in\mathcal{Y}} p(\mathbf{x}|y = a) p(y = a) \\
&= \mathbb{1}\left(\frac{p(\mathbf{x}|y = 1) p(y = 1)}{p(\mathbf{x}|y = 0) p(y = 0)} > 1\right)
\end{aligned}$$

b). Suppose the conditional covariate distributions are Gaussian $p(\mathbf{x}|y = k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and $p(y = 1) = q$ where q is a constant. With as-derived $\pi_\circ(\mathbf{x})$ in Question 1. a, we have

$$\begin{aligned}
\pi_\circ(\mathbf{x}) &= \mathbb{1}\left(\frac{p(\mathbf{x}|y = 1) p(y = 1)}{p(\mathbf{x}|y = 0) p(y = 0)}\right) \\
&= \mathbb{1}\left[\frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) q}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)(1 - q))} > 1\right] \\
&= \mathbb{1}\left[\ln \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \ln q - \ln \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) - \ln(1 - q) > 0\right] \\
&= \mathbb{1}\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_1^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) + \text{const.} > 0\right] \\
&= \mathbb{1}\left[\mathbf{x}^\top(\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1})\mathbf{x} + 2(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}_0^{-1})\mathbf{x} + \text{const.} > 0\right]
\end{aligned}$$

It is proved that the risk-minimizer predictor $\pi_\circ \in \Pi_{\text{quad}}$.

2 Problem 2

a). The plot of $\mathbf{x}$ with colors corresponding to $y = 0$ and $y = 1$ are shown as Figure 1. The code for the data generation and plotting is deferred to Appendix A.1.1.

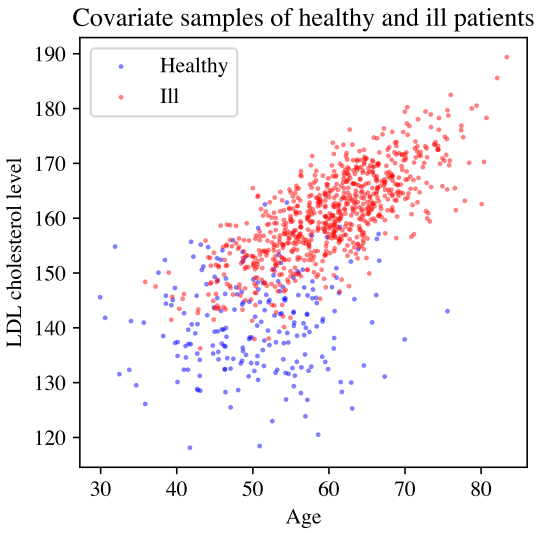

Figure 1: Plots of $\mathbf{x}$ conditioning on $y = 0$ (healthy) and $y = 1$ (ill)

b). The logistic model is to model logits with following form:

$$\ln \frac{p_{\boldsymbol{\theta}}(y=1|\mathbf{x})}{p_{\boldsymbol{\theta}}(y=0|\mathbf{x})} = \boldsymbol{\theta}^\top \phi(\mathbf{x})$$

Then we have

$$\begin{aligned} p_{\boldsymbol{\theta}}(y=1|\mathbf{x}) &= \frac{1}{1+\exp(-\boldsymbol{\theta}^\top \phi(\mathbf{x}))} \\ &= \sigma(\boldsymbol{\theta}^\top \phi(\mathbf{x})) \\ p_{\boldsymbol{\theta}}(y=0|\mathbf{x}) &= 1 - p_{\boldsymbol{\theta}}(y=1|\mathbf{x}) \\ &= \sigma(-\boldsymbol{\theta}^\top \phi(\mathbf{x})) \end{aligned}$$

where $\sigma(\cdot)$ denotes sigmoid function $\sigma(t) = \frac{1}{1+\exp(-t)}$

Using suprisal loss $\ell(\boldsymbol{x}, y) = -\ln p(y|\boldsymbol{x})$ for sample $(\boldsymbol{x}, y)$, the empiric risk minimizer is in following form with n samples:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell_{\boldsymbol{\theta}}(\boldsymbol{x}_i, y_i) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} -\ln p_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}_i) \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} [-y_i \ln p_{\boldsymbol{\theta}}(y=1|\boldsymbol{x}_i) - (1-y_i)\ln(p_{\boldsymbol{\theta}}(y=0|\boldsymbol{x}_i))] \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} [-y_i \ln(\sigma(\boldsymbol{\theta}^\top \phi(\mathbf{x}))) - (1-y_i)\ln(\sigma(-\boldsymbol{\theta}^\top \phi(\mathbf{x})))] \end{aligned}$$

With ϕ' and ϕ'', the results for the logistic regression are shown as Figure 2. The code for producing the figure is appended in Section A.1.2.

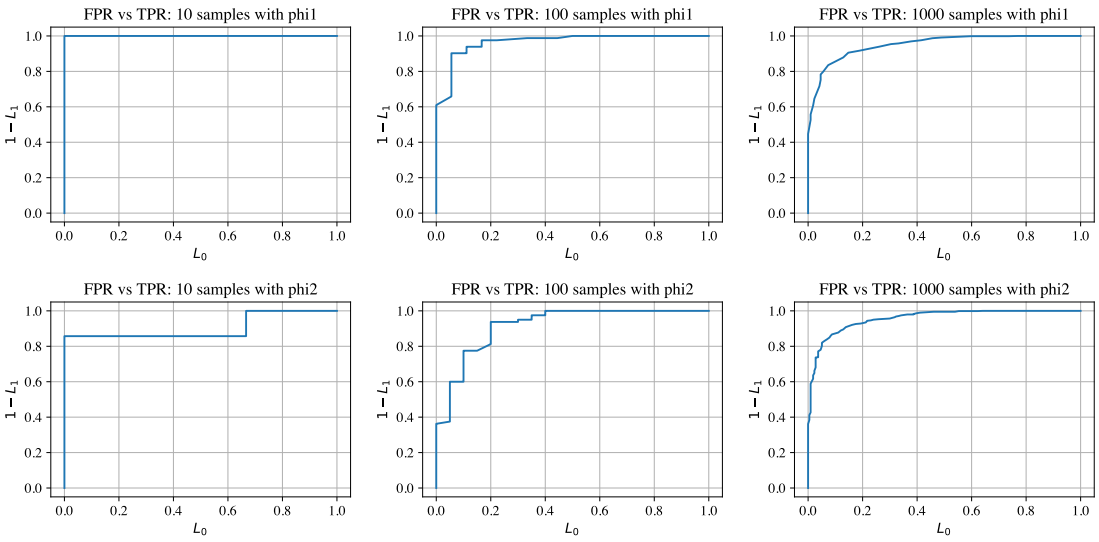

Figure 2: Plots of False Positive Rate against True Positive Rate with ϕ' and ϕ'' and different numbers of test samples.

c). With the model $p_{\theta_k}(\mathbf{x}|y=k) = \mathcal{N}(\mathbf{x}; \boldsymbol{m}_k, \boldsymbol{C}_k)$, we have

$$\begin{aligned} T(\mathbf{x};\boldsymbol{\theta}) &= \frac{p_{\theta_1}(\mathbf{x}|y=1)}{p_{\theta_0}(\mathbf{x}|y=0)} = \frac{\mathcal{N}(\mathbf{x};\boldsymbol{m}_0,\boldsymbol{C}_0)}{\mathcal{N}(\mathbf{x};\boldsymbol{m}_0,\boldsymbol{C}_0)} \\ &= \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{m}_1)^\top \boldsymbol{C}_1^{-1}(\mathbf{x}-\boldsymbol{m}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{m}_0)^\top \boldsymbol{C}_0^{-1}(\mathbf{x}-\boldsymbol{m}_0)\right] \cdot \frac{\det(\boldsymbol{C}_1)^{-\frac{1}{2}}}{\det(\boldsymbol{C}_0)^{-\frac{1}{2}}} \\ &= \exp\left[\frac{1}{2}(\mathbf{x}^\top(\boldsymbol{C}_1^{-1}-\boldsymbol{C}_0^{-1})\mathbf{x} + (\boldsymbol{m}_0^\top \boldsymbol{C}_0^{-1} - \boldsymbol{m}_1^\top \boldsymbol{C}_1^{-1})\mathbf{x} + \mathbf{x}^\top(\boldsymbol{C}_0^{-1}\boldsymbol{m}_0 - \boldsymbol{C}_1^{-1}\boldsymbol{m}_1))\right] \cdot A \end{aligned}$$

where A is a constant w.r.t $\mathbf{x}$. Then with τ with a fixed value, the policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{x})$ is

$$\begin{aligned} \pi_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathbb{1}(T(\mathbf{x};\boldsymbol{\theta}) > \tau) \\ &= \mathbb{1}(\exp\left[\frac{1}{2}(\mathbf{x}^\top(\boldsymbol{C}_1^{-1}-\boldsymbol{C}_0^{-1})\mathbf{x} + (\boldsymbol{m}_0^\top \boldsymbol{C}_0^{-1} - \boldsymbol{m}_1^\top \boldsymbol{C}_1^{-1})\mathbf{x} + \mathbf{x}^\top(\boldsymbol{C}_0^{-1}\boldsymbol{m}_0 - \boldsymbol{C}_1^{-1}\boldsymbol{m}_1))\right] > \frac{\tau}{A}) \\ &= \mathbb{1}(\underbrace{\mathbf{x}^\top(\boldsymbol{C}_1^{-1}-\boldsymbol{C}_0^{-1})\mathbf{x}}_{T} + (\boldsymbol{m}_0^\top \boldsymbol{C}_0^{-1} - \boldsymbol{m}_1^\top \boldsymbol{C}_1^{-1})\mathbf{x} + \mathbf{x}^\top(\boldsymbol{C}_0^{-1}\boldsymbol{m}_0 - \boldsymbol{C}_1^{-1}\boldsymbol{m}_1) > 2\ln\frac{\tau}{A}) \end{aligned}$$

It is observed that the policy is in a quadratic form, i.e. $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \Pi_{\text{quad.}}$. If the model covariances are equal $\boldsymbol{C}_0 = \boldsymbol{C}_1$, term T equals 0 and the policy is in a linear form, i.e. $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \Pi_{\text{lin.}}$.

d). With suprisal loss $\ell_\theta(\boldsymbol{x}, y) = -\ln p_\theta(\boldsymbol{x}|y)$ for sample $(\boldsymbol{x}, y)$, the empirical risk is given by

$$\begin{aligned} R_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) &= \sum_{i\in\{i|y_i=1\}} -\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|y=1) + \sum_{j\in\{j|y_j=0\}} -\ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_j|y=0) \\ &= \sum_{i\in\{i|y_i=1\}} \frac{1}{2}\ln(\det(\boldsymbol{C}_1)) + \frac{1}{2}(\boldsymbol{x}_i-\boldsymbol{m}_1)^\top \boldsymbol{C}_1^{-1}(\boldsymbol{x}_i-\boldsymbol{m}_1) + \\ &\quad \sum_{j\in\{j|y_j=0\}} \frac{1}{2}\ln(\det(\boldsymbol{C}_0)) + \frac{1}{2}(\boldsymbol{x}_j-\boldsymbol{m}_0)^\top \boldsymbol{C}_0^{-1}(\boldsymbol{x}_j-\boldsymbol{m}_0) \end{aligned}$$

The estimations of $\boldsymbol{m}_k, k \in \{0, 1\}$ is obtained if

$$\nabla_{\boldsymbol{m}_k} R_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}; \hat{\boldsymbol{m}_k}) = \sum_{i \in \{i|y_i=k\}} (\boldsymbol{x}_i - \hat{\boldsymbol{m}_k}) \boldsymbol{C}_k^{-1} = \mathbf{0}$$

Since we have $\boldsymbol{C}_k \neq \mathbf{0}$, we have

$$\hat{\boldsymbol{m}_k} = \sum_{i \in \{i|y_i=k\}} \boldsymbol{x}_i / n_k = \mathbb{E}_{n_k}[\mathbf{x}]$$

where $n_k \coloneqq |\{\boldsymbol{x}_i | y_i = k\}|$.

To estimate $\boldsymbol{C}_k, k \in \{0, 1\}$, we have two cases:

- If we have $\boldsymbol{C}_0 = \boldsymbol{C}_1 = \boldsymbol{C}$, the estimation of $\boldsymbol{C}$ is obtained if

$$\begin{aligned}\nabla_{\boldsymbol{C}} R_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}; \hat{\boldsymbol{C}}) = &\sum_{i \in \{i|y_i=1\}} \hat{\boldsymbol{C}}^{-1} - \hat{\boldsymbol{C}}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^\top \hat{\boldsymbol{C}}^{-1} + \\ &\sum_{j \in \{j|y_j=0\}} \hat{\boldsymbol{C}}^{-1} - \hat{\boldsymbol{C}}^{-1}(\boldsymbol{x}_j - \boldsymbol{m}_0)(\boldsymbol{x}_j - \boldsymbol{m}_0)^\top \hat{\boldsymbol{C}}^{-1} = \mathbf{0}\end{aligned}$$

  Then we have

$$\begin{aligned}n\hat{\boldsymbol{C}}^{-1} = &\sum_{i \in \{i|y_i=1\}} \hat{\boldsymbol{C}}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^\top \hat{\boldsymbol{C}}^{-1} + \\ &\sum_{j \in \{j|y_j=0\}} \hat{\boldsymbol{C}}^{-1}(\boldsymbol{x}_j - \boldsymbol{m}_0)(\boldsymbol{x}_j - \boldsymbol{m}_0)^\top \hat{\boldsymbol{C}}^{-1} \\ n\hat{\boldsymbol{C}} = &\sum_{i \in \{i|y_i=1\}} (\boldsymbol{x}_i - \boldsymbol{m}_1)(\boldsymbol{x}_i - \boldsymbol{m}_1)^\top + \sum_{j \in \{j|y_j=0\}} (\boldsymbol{x}_j - \boldsymbol{m}_0)(\boldsymbol{x}_j - \boldsymbol{m}_0)^\top \\ \hat{\boldsymbol{C}} = &\frac{n_0}{n} \mathbb{E}_{n_0}[(\mathbf{x} - \boldsymbol{m}_0)(\mathbf{x} - \boldsymbol{m}_0)^\top] + \frac{n_1}{n} \mathbb{E}_{n_1}[(\mathbf{x} - \boldsymbol{m}_1)(\mathbf{x} - \boldsymbol{m}_1)^\top]\end{aligned}$$

- If we have $\boldsymbol{C}_0 \neq \boldsymbol{C}_1$, the estimation of $\boldsymbol{C}_k, k \in \{0, 1\}$ is obtained if

$$\nabla_{\boldsymbol{C}_k} R_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}; \hat{\boldsymbol{C}}_k) = \sum_{i \in \{i|y_i=k\}} \hat{\boldsymbol{C}_k}^{-1} - \hat{\boldsymbol{C}_k}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}_k)(\boldsymbol{x}_i - \boldsymbol{m}_k)^\top \hat{\boldsymbol{C}_k}^{-1} = \mathbf{0}$$

  Then we have

$$\begin{aligned}n_k \hat{\boldsymbol{C}}_k = &\sum_{i \in \{i|y_i=k\}} (\boldsymbol{x}_i - \boldsymbol{m}_k)(\boldsymbol{x}_i - \boldsymbol{m}_k)^\top \\ \hat{\boldsymbol{C}}_k = &\mathbb{E}_{n_k}[(\mathbf{x} - \boldsymbol{m}_k)(\mathbf{x} - \boldsymbol{m}_k)^\top]\end{aligned}$$

e). The likelihood-ratio models of both linear and quadratic forms are implemented, and the results are plotted in Figure 3. The code is deferred to Appendix A.1.3.

Likelihood-ratio models requires the assumption for the underlying distribution of data. If the assumption matches the true distribution of data well, the model performs well. In contrast, inappropriate assumptions will lead to a poor-performed model. For logistic regression, we do not need to make assumptions for the distribution of data. Thus, if we do have prior knowledge about the distribution of data, likelihood-ratio models will outperforms logistic regression models, otherwise logistic regression is "safer" to avoid inappropriate assumptions.

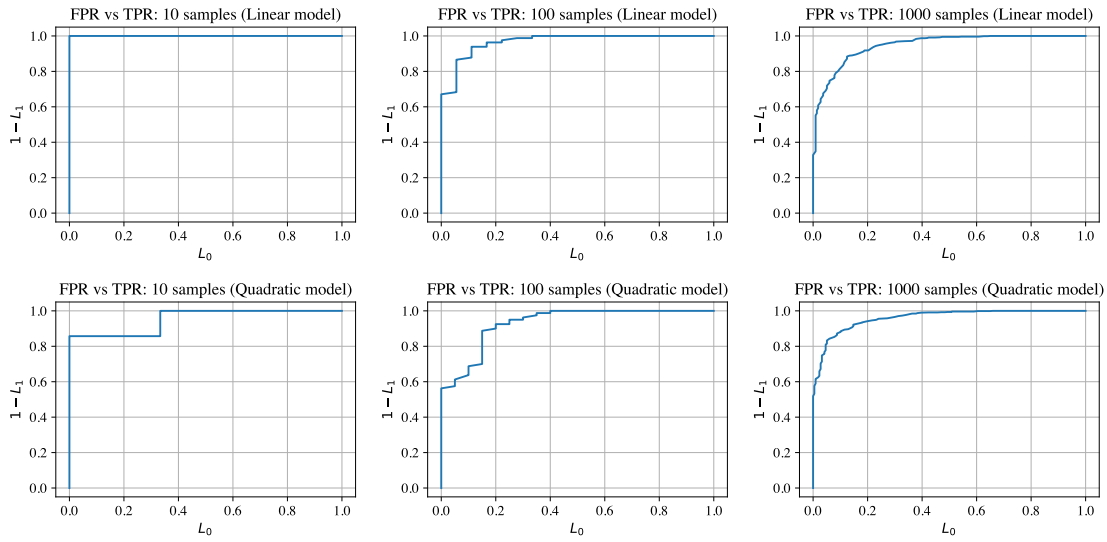

Figure 3: Plots of False Positive Rate against True Positive Rate with linear and quadratic policies and different numbers of test samples.

A Codes

A.1 Code for Problem 2

A.1.1 Problem 2a

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

m = 1000

mu0 = jnp.array([50, 140])
sigma0 = jnp.array([[64, 9], [9, 64]])

mu1 = jnp.array([60, 160])
sigma1 = jnp.array([[64, 49], [49, 64]])

key = jax.random.PRNGKey(0)

key, _ = jax.random.split(key)
y0_nums = jax.random.bernoulli(key, p=0.2, shape=(m,)).sum().item()

key, _ = jax.random.split(key)
x0s = jax.random.multivariate_normal(key, mu0, sigma0, shape=(y0_nums,))

key, _ = jax.random.split(key)
x1s = jax.random.multivariate_normal(
    key, mu1, sigma1, shape=(m-y0_nums,))

fig, ax = plt.subplots(figsize=(4, 4))
ax.grid(False)
ax.scatter(x0s[:, 0], x0s[:, 1], s=2,
           label="Healthy", c="blue", alpha=0.5)
```

```
ax.scatter(x1s[:, 0], x1s[:, 1], s=2,
           label="Ill", c="red", alpha=0.5)

ax.set_title("Covariate samples of healthy and ill patients")
ax.set_xlabel("Age")
ax.set_ylabel("LDL cholesterol level")

ax.legend()
fig.tight_layout()
fig.savefig("hw5_2a.pdf", dpi=500)
```

A.1.2 Problem 2b

```
import jax.numpy as jnp
import matplotlib.pyplot as plt
import jax
from sklearn.linear_model import LogisticRegression


m = 1000

# initialize mu0, sigma0, mu1, sigma1
mu0 = jnp.array([50, 140])
sigma0 = jnp.array([[64, 9], [9, 64]])

mu1 = jnp.array([60, 160])
sigma1 = jnp.array([[64, 49], [49, 64]])

# initialize key for random number generation
key = jax.random.PRNGKey(0)

# generate samples
key, _ = jax.random.split(key)
y0_nums = jax.random.bernoulli(key, p=0.2, shape=(m,)).sum().item()

key, _ = jax.random.split(key)
x0s = jax.random.multivariate_normal(key, mu0, sigma0, shape=(y0_nums,))

key, _ = jax.random.split(key)
x1s = jax.random.multivariate_normal(
    key, mu1, sigma1, shape=(m-y0_nums,))


# define transformation functions
phi1 = jax.vmap(lambda x: jnp.array([1, x[0], x[1]]),
                in_axes=(0,))

phi2 = jax.vmap(lambda x: jnp.array([1, x[0], x[1],
                                     x[0]**2, x[1]**2,
                                     x[0]*x[1]]),
                in_axes=(0,))

xs = jnp.concatenate((x0s, x1s))
ys = jnp.concatenate((jnp.zeros(y0_nums), jnp.ones(m-y0_nums)))
```

```
# compute false positive rate and false negative rate
def fp_fn_compute(tau, model, xs, ys):
    # compute log odds
    log_odds = model.intercept_ + jnp.dot(model.coef_, xs.T)
    # compute predictions
    predictions = (log_odds > tau).astype(int)
    # compute L0
    fp_rate = jnp.sum((predictions == 1) & (ys == 0)) / jnp.sum(ys == 0)
    # compute 1-L1
    fn_rate = jnp.sum((predictions == 1) & (ys == 1)) / jnp.sum(ys == 1)

    return fp_rate, fn_rate


nums_samples = [10, 100, 1000]
phis = {"phi1": phi1, "phi2": phi2}
for phi_name, phi in phis.items():
    # define logistic regression model
    model = LogisticRegression(solver='liblinear', random_state=0)
    model.fit(phi(xs), ys)

    # sample n points from xs
    for n in nums_samples:
        key, _ = jax.random.split(key)
        sample_indices = jax.random.choice(key, jnp.arange(m), shape=(n,),
                                           replace=False)
        sample_xs = phi(xs[sample_indices])
        sample_ys = ys[sample_indices]

        fps = []
        fns = []
        tau_range = jnp.linspace(-50, 50, 1000)
        for tau in tau_range:
            fp_rate, fn_rate = fp_fn_compute(tau, model,
                                             sample_xs, sample_ys)
            fps.append(fp_rate)
            fns.append(fn_rate)

        # print the results
        plt.plot(fps, fns)
        plt.xlabel(r'$L_0$')
        plt.ylabel(r'$1-L_1$')
        title = f'FPR vs TPR: {n} samples with {phi_name}'
        plt.title(title)
        plt.savefig(f'{title.replace(" ", "").replace(":", "")}.pdf', dpi=300)
        plt.close()
```

A.1.3 Problem 2e

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

# generate data
```

```
m = 1000

# initialize mu0, sigma0, mu1, sigma1
mu0 = jnp.array([50, 140])
sigma0 = jnp.array([[64, 9], [9, 64]])

mu1 = jnp.array([60, 160])
sigma1 = jnp.array([[64, 49], [49, 64]])

# initialize key for random number generation
key = jax.random.PRNGKey(0)

# generate samples
key, _ = jax.random.split(key)
y0_nums = jax.random.bernoulli(key, p=0.2, shape=(m,)).sum().item()

key, _ = jax.random.split(key)
x0s = jax.random.multivariate_normal(key, mu0, sigma0, shape=(y0_nums,))

key, _ = jax.random.split(key)
x1s = jax.random.multivariate_normal(
    key, mu1, sigma1, shape=(m-y0_nums,))

xs = jnp.concatenate((x0s, x1s))
ys = jnp.concatenate((jnp.zeros(y0_nums), jnp.ones(m-y0_nums)))


# define predict function
def predict(x, tau, mu0_hat, mu1_hat, sigma0_hat, sigma1_hat):
    # compute ratio of two likelihoods
    py1 = jax.scipy.stats.multivariate_normal.pdf(x, mu1_hat, sigma1_hat)
    py0 = jax.scipy.stats.multivariate_normal.pdf(x, mu0_hat, sigma0_hat)
    predictions = (jnp.log(py1/py0) > tau).astype(int)
    return predictions


# compute L0 and 1-L1
def fp_fn_compute(predictions, ys):
    # compute L0
    fp_rate = jnp.sum((predictions == 1) & (ys == 0)) / jnp.sum(ys == 0)
    # compute 1-L1
    fn_rate = jnp.sum((predictions == 1) & (ys == 1)) / jnp.sum(ys == 1)

    return fp_rate, fn_rate


# estimate mean values
mu0_hat = jnp.mean(x0s, axis=0)
mu1_hat = jnp.mean(x1s, axis=0)

# estimate the covariance matrix when assuming the same covariance matrix
sigma_hat = jnp.dot((x0s - mu0_hat).T, (x0s - mu0_hat))/m + \
    jnp.dot((x1s - mu1_hat).T, (x1s - mu1_hat))/m

# estimate the covariance matrix when assuming different covariance matrices
```

```
sigma0_hat = jnp.dot((x0s - mu0_hat).T, (x0s - mu0_hat))/m
sigma1_hat = jnp.dot((x1s - mu1_hat).T, (x1s - mu1_hat))/m

sigmas = {"Linear": (sigma_hat, sigma_hat),
          "Quadratic": (sigma0_hat, sigma1_hat)}


tau_range = jnp.linspace(-50, 50, 1000)
nums_samples = [10, 100, 1000]


for model_classs, sigma_values in sigmas.items():
    for n in nums_samples:
        key, _ = jax.random.split(key)
        sample_indices = jax.random.choice(key, jnp.arange(m), shape=(n,),
                                           replace=False)
        sample_xs = xs[sample_indices]
        sample_ys = ys[sample_indices]

        fps = []
        fns = []
        for tau in tau_range:
            predictions = predict(sample_xs, tau, mu0_hat, mu1_hat,
                                  sigma_values[0], sigma_values[1])
            fp_rate, fn_rate = fp_fn_compute(predictions, sample_ys)
            fps.append(fp_rate)
            fns.append(fn_rate)

        # print the results
        plt.plot(fps, fns)
        plt.xlabel(r'$L_0$')
        plt.ylabel(r'$1-L_1$')
        title = f'FPR vs TPR: {n} samples ({model_classs} model)'
        plt.title(title)
        plt.savefig(f'hw5/figure2e/{title.replace(" ", "").replace(":", "")}.pdf', dpi=300)
        plt.close()
```