Problem 1

Multi-label support vector machine (SVM).

data points: $(x_i, y_i) \in \mathbb{R}^n \times \{1, \dots, K\}$, $i = 1, \dots, m$.

K affine functions $f_k(x) = a_k^T x + b_k$, $k = 1, \dots, K$.

Given x we guess $\hat{y} = \operatorname{argmax}_k f_k(x)$.

To correctly classify the data examples, we need

$$f_{y_i}(x_i) > \max_{k \neq y_i} f_k(x_i) \quad \text{for all } i.$$

Then we have the loss function

$$L(A, b) = \sum_{i=1}^{m} \left(1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i)\right)_+$$

where $(u)_+ = \max\{u, 0\}$. The multi-label SVM chooses A and b to minimize

$$\min \; L(A, B) + \mu \|A\|_F^2,$$

$$\text{Subject to} \quad 1^T b = 0$$

$$\mu > 0.$$

(a) Want to show how to find A and b using convex optimization.

We can observe that the optimization problem is already convex in the problem description.

First we look at the loss function $L(A, b)$ with variables A, b.

$$\left(1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i)\right)_+$$

$f_k(x_i) = a_k^T x + b_k$ ← affine

$\max_{k \neq y_i} f_k(x_i)$ ← max of affine is convex

$f_{y_i}(x_i)$ ← linear

Therefore $L(A, b)$ is the sum of a constant, convex and linear function and is hence convex. Its also convex and nondecreasing so by composition it is convex.

Our only constraint $1^T b = 0$ is linear.

The regularization term $\mu \|A\|_F^2$ with $\mu > 0$ is the Frobenius norm $\mu \|A\|_F^2 = \mu \left[\sum_i^m \sum_j^n |a_{ij}|^2\right]$ and by definition is a convex quadratic term.

Therefore, our optimization problem is indeed convex as stated in the problem statement.

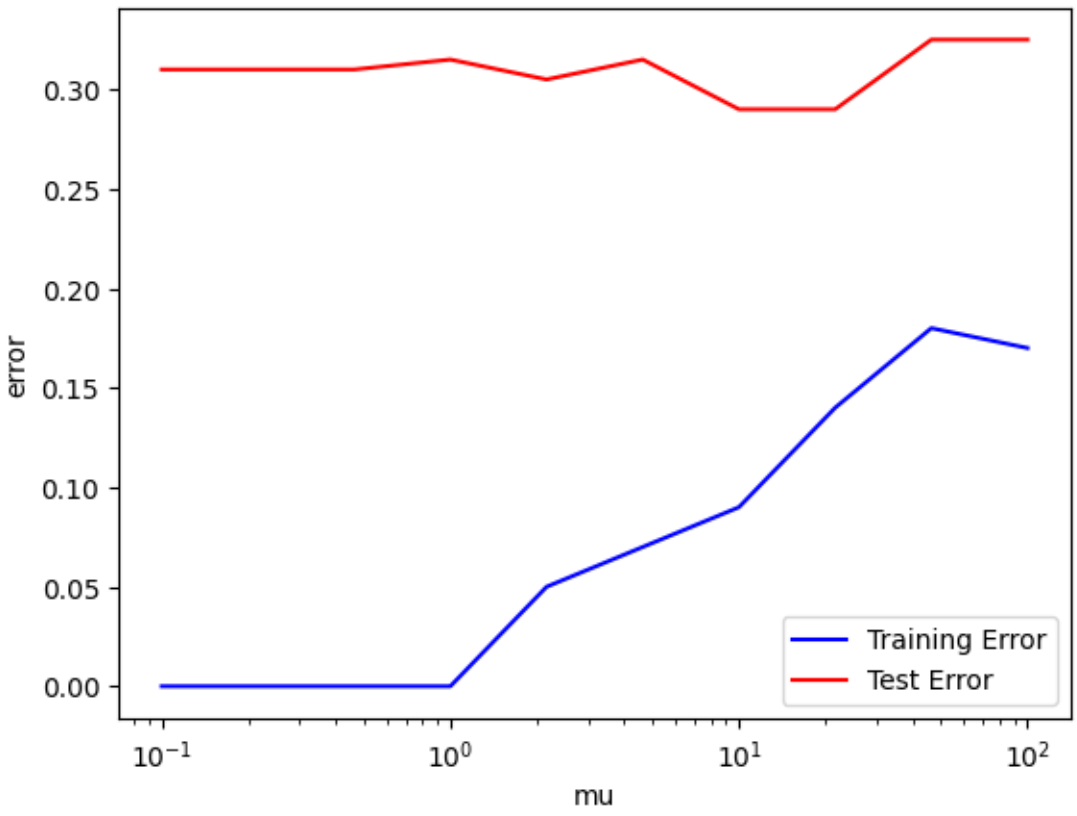
(b) Implement the multi-label SVM.

[16]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

[17]:
```
# Given training and test data

np.random.seed(10)

mTrain = 100 # size of training data
mTest  = 200 # size of test data
K = 5;       # number of categories
n = 20;      # number of features

A_true = np.random.normal(size=(K, n))
b_true = np.random.normal(size=(K, 1))
v = np.random.normal(size=(K, mTrain + mTest)) # noise
data = np.random.normal(size=(n, mTrain + mTest))
label = np.argmax(A_true @ data + np.tile(b_true, (1, mTrain + mTest)) + v,␣
 ↪axis=0)

# Training data
x = data[:, :mTrain]
y = label[:mTrain]
# Test data
xtest = data[:, mTrain:]
ytest = label[mTrain:]
```

[18]:
```
"""
Problem 1

part b
"""

errorTrain = []
errorTest = []

mus = np.logspace(-1, 2, 10)

for mu in mus:
    A = cp.Variable((K, n))
    b = cp.Variable(K)
    f = A @ x + b[:, np.newaxis]

    L = 0
```

```
    for k in range(K):
        # index is a list of every label k besides the current k in the for loop
        index = np.concatenate([np.arange(k), np.arange(k + 1, K)])
        L += cp.sum(cp.pos(1 + cp.max(f[index[:, np.newaxis], y == k], axis=0) -␣
↪f[k, y == k]))

    objective = cp.Minimize(L + mu * cp.square(cp.norm(A, "fro")))
    constraint = [cp.sum(b) == 0]

    problem = cp.Problem(objective, constraint)
    problem.solve()

    # check training error
    ypred_train = np.argmax((A @ x + b[:, np.newaxis]).value, axis=0)
    errorTrain.append(np.sum(ypred_train != y) / mTrain)
    # check test error
    ypred_test = np.argmax((A @ xtest + b[:, np.newaxis]).value, axis=0)
    errorTest.append(np.sum(ypred_test != ytest) / mTest)


# plot error curves
plt.semilogx(mus, errorTrain, 'b', label='Training Error')
plt.semilogx(mus, errorTest, 'r', label='Test Error')
plt.xlabel('mu')
plt.ylabel('error')
plt.legend()
plt.show()
```


0.30
0.25
0.20
0.15
0.10
0.05
0.00
error
10^{-1}
10^{0}
10^{1}
10^{2}
mu
Training Error
Test Error

[ ]:

Problem 2

For an $m \times n$ matrix A (with $m > n$) and an integer k between 1 and n, we define $f(A)$ as the sum of the largest k singular values of A:

$$f(A) = \sum_{i=1}^{k} \sigma_i(A),$$

where $\sigma_1(A), \ldots, \sigma_n(A)$ denote the singular values of A in nondecreasing order.

(a) Consider the following SDP:

$$\begin{aligned} \text{maximize} \quad & \operatorname{Tr}(A^T X) \\ \text{subject to} \quad & \begin{bmatrix} U & X \\ X^T & V \end{bmatrix} \succeq 0 \\ & U \preceq I \\ & V \preceq I \\ & \operatorname{Tr}(U) + \operatorname{Tr}(V) = 2k, \end{aligned}$$

with variable $X \in \mathbb{R}^{m \times n}$, $U \in S^m$, $V \in S^m$. Find the dual of this problem.

Lets start by writing out the dual where we replace A with Σ_A a matrix of singular values using the hint.

$$g(\lambda, v) = \inf_{Y,W,Z} L(Y, W, Z, \Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5, v)$$

$$= \inf_{Y,W,Z} \Big\{ -\mathrm{Tr}(\Sigma_A^T Y) + \mathrm{Tr}(\Lambda_4(W - I)) + \mathrm{Tr}(\Lambda_5(Z - I))$$

$$+ \mathrm{Tr}\left(\begin{bmatrix} \Lambda_1 & \Lambda_2^T \\ \Lambda_2 & \Lambda_3 \end{bmatrix} \begin{bmatrix} -W & -Y \\ -Y^T & Z \end{bmatrix}\right) + v(\mathrm{Tr}(W) + \mathrm{Tr}(Z) - 2k) \Big\}$$

$$= \inf_{Y,W,Z} \Big\{ \mathrm{Tr}(\Sigma_A^T Y) + (\mathrm{Tr}(\Lambda_4 W) - \mathrm{Tr}(\Lambda_4) + (\mathrm{Tr}(\Lambda_5 Z) - \mathrm{Tr}(\Lambda_5)$$

$$+ (\mathrm{Tr}(\Lambda_1 W) \cdot \mathrm{Tr}(\Lambda_2 Y) - (\Lambda_2^T Y^T) \cdot \mathrm{Tr}(\Lambda_3 Z))$$

$$+ v(\mathrm{Tr}(W) + \mathrm{Tr}(Z) - 2k) \Big\}$$

$$= \inf_Y \Big\{ -\mathrm{Tr}(\Sigma_A^T Y) - 2\mathrm{Tr}(\Lambda_2 Y) \Big\} +$$

$$\inf_W \Big\{ \mathrm{Tr}(\Lambda_4 W - \mathrm{Tr}(\Lambda_1 W) + v\,\mathrm{Tr}(W) \Big\} +$$

$$\inf_Z \Big\{ \mathrm{Tr}(\Lambda_5 Z) - \mathrm{Tr}(\Lambda_3 Z) + v\,\mathrm{Tr}(Z) \Big\}$$

$$- (\mathrm{Tr}(\Lambda_4) + \mathrm{Tr}(\Lambda_5) + 2vk)$$

$$= \inf_y \left[\mathrm{Tr}(-\Sigma_A - 2\Lambda_2^T Y) \right] + \inf_W \left\{ \mathrm{Tr}((\Lambda_4 - \Lambda_1 + \nu I) W) \right\}$$

$$+ \inf_Z \left\{ \mathrm{Tr}((-\Lambda_5 - \Lambda_3 + \nu I) Z) \right\}$$

$$- (\mathrm{Tr}(\Lambda_4) + \mathrm{Tr}(\Lambda_5) + 2\nu k)$$

Cleaning up, we can get our objective and constraints and write the dual problem

$$\max \; -(\mathrm{Tr}(\Lambda_4) + \mathrm{Tr}(\Lambda_5) + 2\nu k)$$

$$\text{s.t.} \quad \Sigma_A + 2\Lambda_2 = [0]$$

$$\Lambda_4 - \Lambda_1 + \nu I_m = [0]$$

$$\Lambda_5 - \Lambda_3 + \nu I_n = [0]$$

$$\Lambda_4, \Lambda_5 \succeq 0$$

$$\begin{bmatrix} \Lambda_1 & \Lambda_2^T \\ \Lambda_2 & \Lambda_3 \end{bmatrix} \succeq 0 .$$

(b) Show that $f(A)$ is the optimal value of problem (1).

Hint: since all 2×2 principal minors of a PSD matrix are PSD, we have

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \succeq 0 \Rightarrow B_{ii} \le \sqrt{A_{ii}C_{ii}} \le \frac{A_{ii}+C_{ii}}{2}.$$

Then, $$\begin{bmatrix} W & Y \\ Y^T & Z \end{bmatrix} \succeq 0 \Rightarrow Y_{ii} \le \sqrt{W_{ii}Z_{ii}} \le \frac{1}{2}(W_{ii}+Z_{ii})$$

$W \preceq I$

$Z \preceq I$

$$\sum_i Y_{ii} \le \sum_i \frac{1}{2}(W_{ii}+Z_{ii})$$

$$Tr(Y) \le \frac{1}{2}(\underbrace{Tr(W)+Tr(Z)}_{=2k})$$

$$Tr(Y) \le k$$

$$\sum_i Y_{ii} \le k$$

$$f(A) = \sum_{i=1}^{k} \sigma_i = \max_Y Tr(\mathcal{Q}_A Y)$$

$$\max_Y = \sum_{i=1}^{k} \sigma_i \le k\sigma_{max}$$

hence we have shown $f(A)$ is the optimal solution to the SDP problem.

(c) What does part (b) imply about convexity of $f(A)$?

It implies its convex because it is the solution to an SDP which is a convex optimization problem, the optimal value to an SDP is convex in the problem data.

(d) Use the dual problem to give an SDP formulation of the problem

$$\text{minimize} \quad f(A_0 + x_1 A_1 + \dots + x_p A_p)$$

with variable $x \in \mathbb{R}^p$, where $A_0, \dots, A_p$ are given $m \times n$ matrices.

Lets write our formulation from part (a):

$$\begin{aligned} \max_{\Lambda, \nu} \quad & -(\mathrm{Tr}(\Lambda_4) + \mathrm{Tr}(\Lambda_5) + 2\nu k) \\ \text{s.t.} \quad & \Sigma_A + 2\Lambda_2 = [0] \\ & \Lambda_4 - \Lambda_1 + \nu I_m = [0] \\ & \Lambda_5 - \Lambda_3 + \nu I_n = [0] \\ & \Lambda_4, \Lambda_5 \succeq 0 \end{aligned}$$

$$\begin{bmatrix} \Omega_1 & \Omega_2^T \\ \Omega_2 & \Omega_3 \end{bmatrix} \succeq 0$$

We want to substitute Σ_A first with A and then with $A_0 + X_1 A_1 + \dots + X_p A_p$. We will also rewrite the constraints. Lets go constraint by constraint

$$\Omega_4 - \Omega_1 + VI_m = [0] \rightarrow \Omega_1 = \Omega_4 + VI_m$$

$$\Omega_5 - \Omega_3 + VI_m = [0] \rightarrow \Omega_3 = \Omega_5 + VI_n$$

$$A + 2\Omega_2 = [0] \rightarrow \Omega_2 = -\frac{1}{2}A$$

$$\begin{bmatrix} \Omega_1 & \Omega_2^T \\ \Omega_2 & \Omega_3 \end{bmatrix} \succeq 0 \rightarrow \begin{bmatrix} \Omega_4 + VI_m & -\frac{1}{2}A^T \\ -\frac{1}{2}A & \Omega_5 + VI_n \end{bmatrix} \succeq 0$$

Where $A \rightarrow A_0 + \sum_{i=1}^{p} X_i A_i$

Now we have the new constraints

also before we had

max - objective

equiv. we will switch to

min + objective

So, with the new constraints and optimizing over λ as well now we have

$$\min_{X_p, \Omega_4, \Omega_5, V} \quad \mathrm{Tr}(\Omega_4) + \mathrm{Tr}(\Omega_5) + 2Vk$$

$$\text{s.t.} \quad \Omega_4, \Omega_5 \succeq 0$$

$$\begin{bmatrix} \Omega_4 + V I_m & -\frac{1}{2} A^T \\ -\frac{1}{2} A & \Omega_5 + V I_n \end{bmatrix} \succeq 0$$

where $A = A_0 + \sum_{i=1}^{p} X_i A_i$.

$\not\#$

Problem 3

Energy storage tradeoffs. We are using a storage device to reduce the total cost of electricity. The day is divided into T time periods, and p_t denotes the positive electricity price. Without a battery the total cost is $p^T u$. We let q_t denote the nonnegative energy stored in the battery in period t. So we have $q_{t+1} = q_t + c_t$, $t = 1, \ldots, T-1$, where c_t is the charging of the battery in period t. We will require that $q_1 = q_T + c_T$ i.e. we finish with the same battery charge that we start with. The total cost is then given as $p^T(u+c)$.

We have the capacity Q where $q_t \in Q$; the maximum charge rate as C where $c_t \leq C$ and the maximum discharge rate D where $c_t \geq -D$

(a) Explain how to find the charging profile $c \in \mathbb{R}^T$ and associated storage energy profile $q \in \mathbb{R}^T$ that minimizes the total cost, subject to the constraints.

To do this we will formulate this as a convex optimization problem. The objective and constraints were given explicitly.

Therefore we have the following formulation

$$\underset{c,q}{\text{minimize}} \quad p^T(u+c)$$

$$\text{subject to} \quad q_1 = q_T + c_T$$
$$q_{t+1} = q_t + c_t, \quad t = 1, \dots, T$$
$$u + c \geq 0$$
$$0 \leq q \leq Q1$$
$$-D1 \leq c \leq C1,$$

we notice that this is an LP since the objective and all of the constraints are linear.

(b) code

(c) Interpretation between trade off curves:

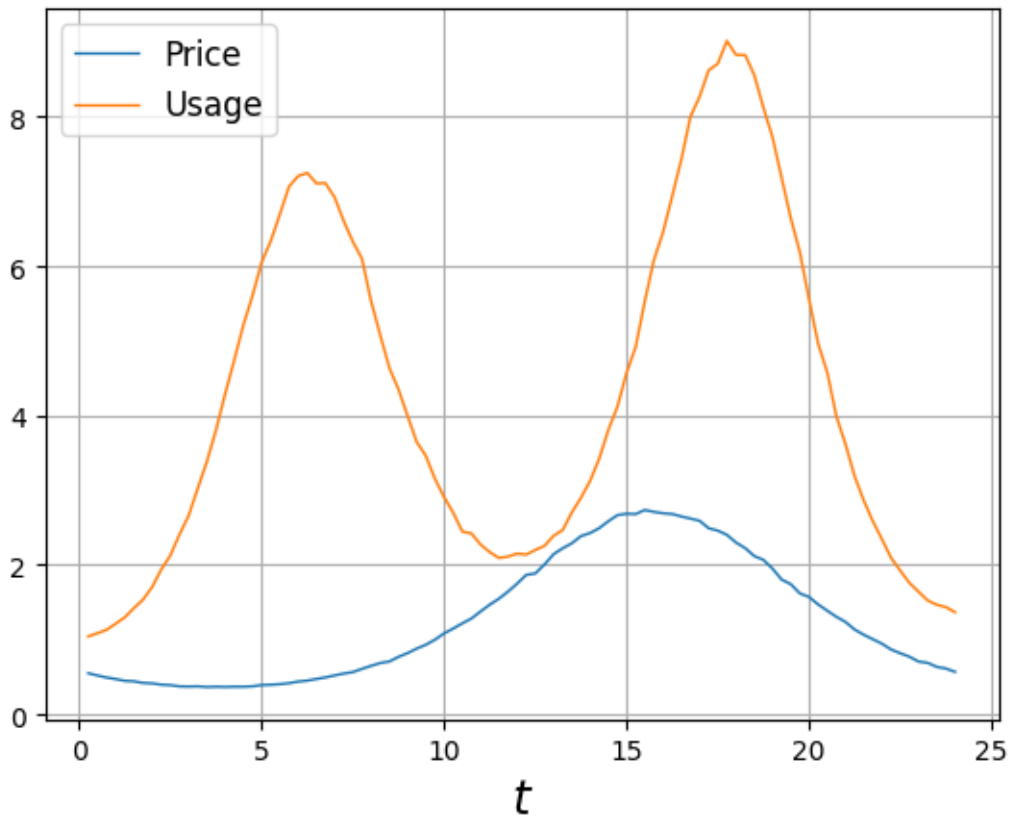
You can utilize larger storage if you have faster charge/discharge and in turn the cost lowers.

[101]:
```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

Storage Data

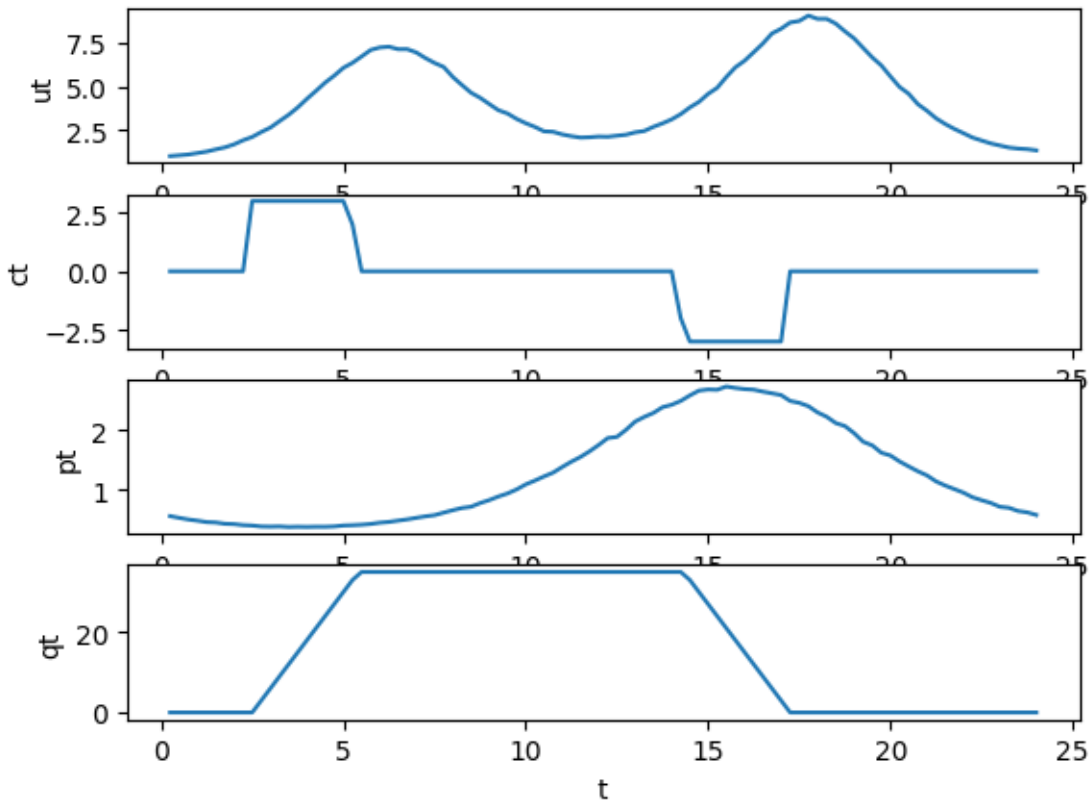
[102]:
```
T = 96
t = np.linspace(1, T, num=T)
p = np.array([0.550399, 0.520442, 0.493439, 0.473159, 0.449186, 0.442857, 0.
 ↪420869,
              0.415239, 0.398026, 0.391326, 0.375166, 0.372396, 0.375692, 0.
 ↪366320,
              0.370010, 0.367343, 0.370538, 0.369990, 0.375826, 0.391768, 0.
 ↪396798,
              0.404598, 0.417414, 0.439177, 0.452071, 0.473926, 0.495028, 0.
 ↪521115,
              0.546207, 0.566100, 0.608810, 0.649835, 0.687016, 0.708069, 0.
 ↪769852,
              0.820108, 0.880429, 0.931914, 0.999980, 1.084879, 1.149127, 1.
 ↪218684,
              1.283492, 1.378140, 1.468384, 1.547619, 1.643179, 1.750187, 1.
 ↪867013,
              1.886839, 2.005964, 2.144843, 2.223695, 2.290468, 2.387310, 2.
 ↪428065,
              2.493444, 2.585986, 2.667354, 2.686608, 2.681477, 2.736588, 2.
 ↪711546,
              2.692840, 2.683158, 2.650256, 2.621730, 2.592316, 2.492386, 2.
 ↪460559,
              2.404532, 2.300139, 2.226397, 2.119787, 2.069468, 1.951479, 1.
 ↪805108,
              1.743544, 1.619182, 1.572349, 1.472011, 1.387958, 1.306774, 1.
 ↪236908,
              1.135787, 1.066123, 1.006177, 0.945895, 0.867899, 0.818247, 0.
 ↪772271,
              0.707323, 0.689977, 0.636003, 0.613567, 0.569250])

u = np.array([1.043739, 1.086538, 1.131884, 1.214194, 1.298476, 1.422279, 1.
 ↪535897,
              1.705704, 1.940522, 2.128072, 2.409882, 2.661096, 3.028017, 3.
 ↪386349,
              3.807308, 4.290259, 4.743668, 5.213152, 5.608307, 6.052159, 6.
 ↪336147,
```

```
            6.696606, 7.066860, 7.208621, 7.250730, 7.110722, 7.113068, 6.
↪923818,
            6.605419, 6.329058, 6.102364, 5.534364, 5.078995, 4.640461, 4.
↪350952,
            3.997046, 3.648759, 3.460412, 3.153410, 2.908417, 2.699520, 2.
↪448209,
            2.422143, 2.273632, 2.172556, 2.094829, 2.112268, 2.152688, 2.
↪139740,
            2.200707, 2.255838, 2.391560, 2.467265, 2.702900, 2.899853, 3.
↪125037,
            3.424463, 3.806827, 4.119900, 4.572451, 4.919367, 5.532844, 6.
↪084863,
            6.455957, 6.937346, 7.427385, 8.002555, 8.269312, 8.625724, 8.
↪712766,
            9.019016, 8.831344, 8.829841, 8.561095, 8.127466, 7.727851, 7.
↪184223,
            6.646169, 6.185818, 5.562063, 4.965322, 4.579912, 3.992179, 3.
↪617470,
            3.188143, 2.864524, 2.585960, 2.343835, 2.094154, 1.924559, 1.
↪762417,
            1.643040, 1.526240, 1.467255, 1.435844, 1.367814])

fig, ax = plt.subplots()

ax.plot(t/4, p, '-', linewidth=1, label=r'Price')
ax.plot(t/4, u, '-', linewidth=1, label=r'Usage')

ax.grid(True)
ax.set_xlabel(r"$t$", fontsize='xx-large')
ax.legend(fontsize='large', loc='upper left')

plt.show()
```



[103]:
```
"""
problem 3
part b
"""
Q = 35
C = 3
D = 3

u = u.reshape((len(u),1))

q = cp.Variable((T, 1))
c = cp.Variable((T, 1))

objective = cp.Minimize(p.T @ (u + c))

constraints = [
    q[1:T] == q[0:T-1] + c[0:T-1],
    q[0] == q[T-1] + c[T-1],
    u + c >= 0,
    c >= -D,
```

```
    c <= C,
    q >= 0,
    q <= Q,]

problem = cp.Problem(objective, constraints)
problem.solve()

# plots
ts = np.arange(1, T+1) / 4

plt.figure()
plt.subplot(4, 1, 1)
plt.plot(ts, u)
plt.xlabel('t')
plt.ylabel('ut')

plt.subplot(4, 1, 2)
plt.plot(ts, c.value)
plt.ylabel('ct')
plt.xlabel('t')

plt.subplot(4, 1, 3)
plt.plot(ts, p)
plt.ylabel('pt')
plt.xlabel('t')

plt.subplot(4, 1, 4)
plt.plot(ts, q.value)
plt.ylabel('qt')
plt.xlabel('t')

plt.show()
```

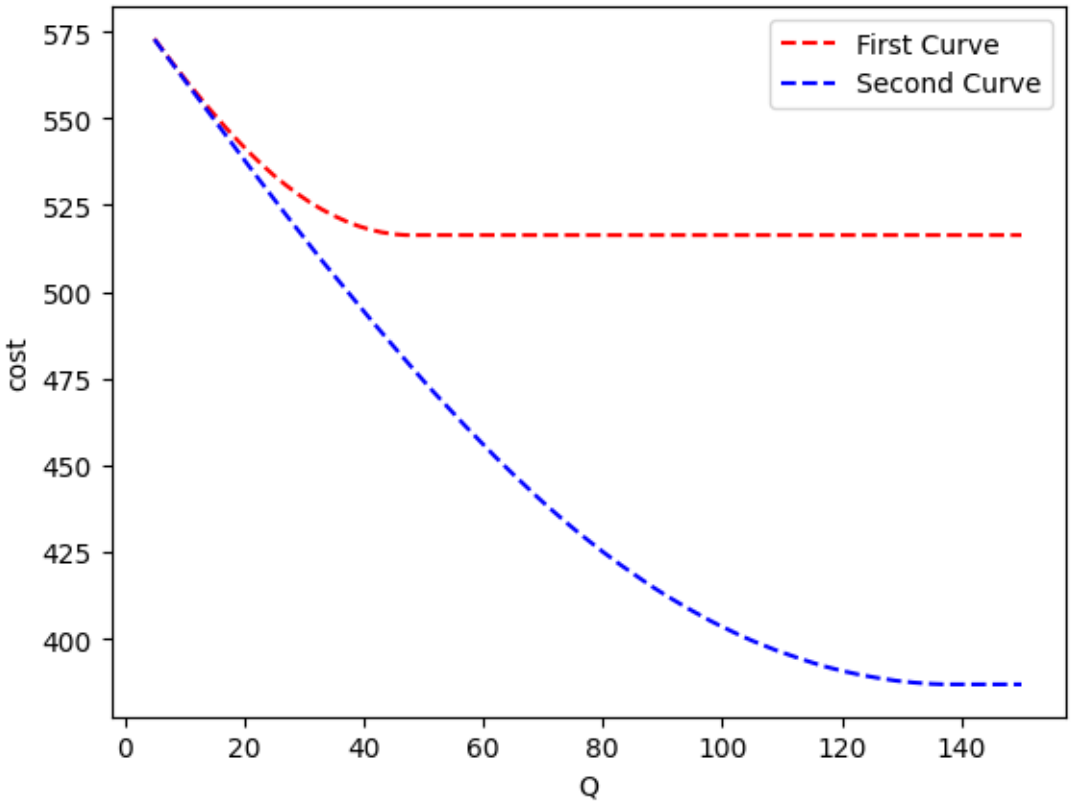


c

[104]:
```
"""
problem 3
part c
"""

# compute the tradeoff curves
N = 146
Qs = np.linspace(5, 150, num=N)
Vals3 = np.zeros(146)
Vals1 = np.zeros(146)

C = 1
D = 1

first_cost = np.zeros((N,1))
second_cost = np.zeros((N,1))

q = cp.Variable((T,1))
c = cp.Variable((T,1))
```

```
objective = cp.Minimize(p.T @ (u + c))

cons = []

for i in range(N):
    Q = Qs[i]

    cons = [c >= -D]
    cons += [c <= C]
    cons += [u+c >= 0]
    cons += [q <= Q]
    cons += [q >= 0]
    cons += [q[0] == q[T-1] + c[T-1]]
    cons += [q[1:] == q[:T-1] + c[:T-1]]

    problem = cp.Problem(objective, cons)
    p_optimal = problem.solve()
    first_cost[i] = p_optimal

C = 3
D = 3

cons = []

for i in range(N):
    Q = Qs[i]

    cons = [c >= -D]
    cons += [c <= C]
    cons += [u+c >= 0]
    cons += [q <= Q]
    cons += [q >= 0]
    cons += [q[0] == q[T-1] + c[T-1]]
    cons += [q[1:] == q[:T-1] + c[:T-1]]

    problem = cp.Problem(objective, cons)
    p_optimal = problem.solve()
    second_cost[i] = p_optimal

# plot tradeoff curves
plt.figure
plt.plot(Qs, first_cost, 'r--', label='First Curve');
plt.plot(Qs, second_cost, 'b--', label='Second Curve');
plt.xlabel('Q')
plt.ylabel('cost')
plt.legend()
```

```
plt.show()
```


First Curve
Second Curve
575
550
525
500
475
450
425
400
cost
0
20
40
60
80
100
120
140
Q

[ ]:

Problem 4

Feature elimination rule. We will derive a rule for discarding irrelevant features in a supervised learning problem with a convex loss function and sparsity inducing penalty. We consider the following supervised learning problem

$$\min_{w \in \mathbb{R}^m} \; F(w) := \sum_{i=1}^{n} f(a_i^T w + b_i) + \lambda \sum_{j=1}^{m} w_j$$

$$\text{s.t.} \quad w \geq 0.$$

We assume

- the loss function f is closed and convex, implying $f^{**} = f$.
- parameter λ exceeds 0.
- strong duality holds.

(a) Dual problem. Show that the problem of maximizing $-F(w)$ subject to $w \geq 0$ is dual to the problem

$$\min_{\theta} \; G(\theta) := -b^T \theta + \sum_{i=1}^{n} f^*(\theta_i)$$

$$\text{s.t.} \quad -A_j^T \theta \leq \lambda \quad \text{for } j = 1, 2, \ldots, m.$$

Since we know f is closed and convex meaning $f^{**} = f$, we can take the dual of the given dual and show that's equivalent to the original problem.

$$L(\theta, \lambda, \mu) = -b^T\theta + \sum_{i=1}^{n} f^*(\theta_i) - \mu\left(\sum_{j=1}^{m} -A_j^T\theta_i - \lambda\right)$$

$$g(\mu) = \inf_{\theta}\left\{-b^T\theta + \sum_{i=1}^{n} f^*(\theta_i) - \sum_{j=1}^{m} \mu^T A_j^T\theta_i - \mu\lambda\right\}$$

Write as stacked vectors and sum over $i = 1, \dots, n$.

$$[m \times 1]\left[m \times n \text{ (} i^{th} \text{ col)}\right]\left[n \times 1\right]$$

$$\mu \qquad A \qquad \theta$$

$$\mu^T A^T \theta \rightarrow$$

$$\Rightarrow g(\mu) = \inf_{\theta}\left\{-b^T\theta + \sum_{i=1}^{n} f^*(\theta_i) - \sum_{i=1}^{n} \mu_i^T a_i \theta_i - \mu\lambda\right\}$$

$$g(\mu) = \inf_{\theta}\left\{-\sum_{i=1}^{n} \mu_i a_i \theta_i - b^T\theta - \mu\lambda + \sum_{i=1}^{n} f^*(\theta_i)\right\}$$

$$g(\mu) = \sup_{\theta}\left\{\sum_{i=1}^{n} \mu_i a_i \theta_i + b^T\theta + \mu\lambda - \sum_{i=1}^{n} f^*(\theta_i)\right\}$$

Call this affine term v

Then we have,

$$\sup_{\theta} \{ v_i \theta_i - f^*(\theta_i) \} + u\lambda$$

$$= f^{**}(v_i) + u\lambda \quad \leftarrow \text{here } u = w_j$$

we know $f = f^{**}$

hence we have shown that the dual of the given dual gives us back the original problem:

$$\min_{w} \quad F(w) = \sum_{i=1}^{n} f(a_i^T w + b_i) + \lambda \sum_{j=1}^{m} w_j$$

$$\text{s.t.} \quad w \geq 0$$

#

(b) optimality conditions. Let w^* be any solution to the original problem, and let θ^* be any solution to the dual. Show that for any $j \in [m]$,

$$-A_j^T \theta^* < \lambda \Rightarrow w^*_j = 0$$

In other words, if $-A_j^T \theta^*$ is strictly less than λ then we must have $w^*_j = 0$.

It is given in the problem statement that strong duality holds so we can directly apply the complementary slackness condition which states

$$\lambda_i^* f_i(x^*) = 0 \quad \text{for } i = 1, \ldots, m$$

So $\lambda_i^* > 0 \Rightarrow f_i(x^*) = 0$, $f_i(x^*) < 0 \Rightarrow \lambda_i^* = 0$

in our problem we have that for any $j \in [m]$ we must have

$$-A_j^T \theta^* < \lambda \Rightarrow w_j^* = 0$$

where θ^* and w^* are optimal values to the dual and primal problems respectively. So this condition must hold by complementary slackness since $-A_j^T \theta^* \neq \lambda$.

(C) Ball that contains θ^*. For the remainder of this problem assume that $G(\theta)$ is differentiable and γ-strongly convex. That is for all $\theta_1, \theta_2 \in \text{dom } G$, we have

$$G(\theta_2) \geq G(\theta_1) + [\nabla G(\theta_1)]^T (\theta_2 - \theta_1) + \frac{\gamma}{2} \|\theta_2 - \theta_1\|_2^2$$

Give any vector $\tilde{w} \in \mathbb{R}^m_{\geq 0}$ and feasible $\tilde{\theta} \in \mathbb{R}^n$

(meaning $-A_j^T \tilde{\theta} \leq \lambda$ for all j), use strong convexity and properties of θ^* to show that the followings holds:

$$\theta^* \in \left\{ \theta : \|\theta - \tilde{\theta}\|_2 \leq \sqrt{\frac{2}{\gamma}\left(F(\tilde{w}) + G_1(\tilde{\theta})\right)} \right\} =: \Theta$$

That is, θ^* lies within a ball with center $\tilde{\theta}$ and known radius.

We are given:

$$G_1(\theta_2) \geq G_1(\theta_1) + \nabla G_1(\theta_1)^T(\theta_2 - \theta_1) + \frac{\gamma}{2}\|\theta_2 - \theta_1\|_2^2$$

plugging in θ^* (optimal) and $\tilde{\theta}$ (feasible)

$$\Rightarrow G_1(\tilde{\theta}) \geq G_1(\theta^*) + \nabla G_1(\theta^*)^T(\tilde{\theta} - \theta^*) + \frac{\gamma}{2}\|\tilde{\theta} - \theta^*\|_2^2$$

$$G_1(\tilde{\theta}) - G_1(\theta^*) \geq \underbrace{\nabla G_1(\theta^*)^T(\tilde{\theta} - \theta^*)}_{\geq 0} + \frac{\gamma}{2}\|\tilde{\theta} - \theta^*\|_2^2$$

note: we know for $\min f(x)$ s.t. $x \in C$

x^* is optimal iff it is feasible and

$$\nabla f(x^*)^T(x - y) \leq 0 \quad \text{for all } y \in C.$$

Therefore now we have

$$G_1(\tilde{\theta}) - G_1(\theta^*) \geq \frac{\gamma}{2}\|\tilde{\theta} - \theta^*\|_2^2$$

$$G(\theta^*) = -F(w^*) \geq -F(\tilde{w})$$

$$G(\tilde{\theta}) + F(\tilde{w}) \geq \frac{\gamma}{2} \| \tilde{\theta} - \theta^* \|_2^2$$

So trivially,

$$\boxed{\frac{2}{\gamma}(F(\tilde{w}) + G(\tilde{\theta}))} \geq \| \tilde{\theta} - \theta^* \|_2 .$$

QED.

(d) Elimination rule. Assume you are given a vector $\tilde{w} \in \mathbb{R}^m_{\geq 0}$ and feasible $\tilde{\theta} \in \mathbb{R}^n$. Using the results from (b) and (c) derive a sufficient condition that, if satisfied, guarentees $w_j^* = 0$ (for any $j \in [m]$). This sufficient condition should not involve w^* or θ^*.

we know $-A_j^T \theta^* \leq \lambda \Rightarrow w_j = 0$

and $\boxed{\frac{2}{\gamma}(G(\tilde{\theta}) + F(\tilde{w}))} \geq \| \tilde{\theta} - \theta^* \|_2^2$

$$\Rightarrow \quad -A_j^T \theta \leq \lambda$$

$$-A_j^T \theta + A_j^T \tilde{\theta} \leq \lambda + A_j^T \tilde{\theta}$$

$$A_j^T (\tilde{\theta} - \theta^*) \leq \lambda + A_j^T \tilde{\theta}$$

$$\| A_j^T (\tilde{\theta} - \theta^*) \|_2 \leq \| \lambda + A_j^T \tilde{\theta} \|_2$$

$$\|A_j^T(\tilde{\theta}-\theta^*)\|_2 \le \|A_j\|_2 \cdot \|\tilde{\theta}-\theta^*\|_2 \le \|\lambda + A_j^T\tilde{\theta}\|_2$$

$$\|A_j\|_2 \sqrt{\frac{2}{\gamma}(G(\tilde{\theta})+F(\tilde{w}))} < \|\lambda + A_j^T\tilde{\theta}\|_2 .$$

↑

QED.

Sufficient condition but not neccessary.

If the last inequality holds then $-A_j^T\theta^* < \lambda \Rightarrow w_j = 0$.

(e) Dual objective for Lasso. Let us now assume that $f(a_i^T w + b_i)$ is a squared loss data fitting term. That is, given a collection of n training instances $(x_1, y_1) \cdots (x_n, y_n)$ where $x_i \in \mathbb{R}^m$ is a feature vector and y_i is a corresponding label, we have

$$f(a_i^T w + b_i) = \frac{1}{2}(x_i^T w - y_i)^2.$$

Notice that $a_i = X$ and $b_i = -y_i$. Derive the dual objective, $G(\theta)$, for this problem.

We have the optimization

$$\min \sum_{i=1}^{n} \frac{1}{2}(x_i^T w - y_i)^2 + \lambda \sum_{i=1}^{r} w_i$$

$$\text{s.t. } w \ge 0$$

also written as

$$\min \ \frac{1}{2}\|z\|_2^2 + \lambda \sum_{i=1}^{c} w_i$$

$$\text{s.t.} \quad z_i = x_i^T w - y_i \ \rightarrow \ z = X^T w - y$$

$$-w \leq 0$$

write the Lagrangian

$$L(z, w, \mu, \theta) = \frac{1}{2}\|z\|_2^2 + \lambda \sum_{i=1}^{c} w_i - \mu^T w + \theta^T (X^T w - y - z)$$

write the dual

$$\inf_{z,w} \left\{ \frac{1}{2}\|z\|_2^2 + \overbrace{\lambda 1^T w}^{\lambda \sum_{i=1}^{c} w_i} - \mu^T w + \theta^T (X^T w - y - z) \right\}$$

$$\inf_z \left\{ \frac{1}{2}\|z\|_2^2 - \theta^T z \right\} + \inf_w \left\{ \lambda 1^T w - \mu^T w + \theta^T X w \right\} - \theta^T y$$

$$-\sup_z \left\{ \theta^T z - \underbrace{\frac{1}{2}\|z\|_2^2}_{\text{dual norm is itself}} \right\} + \inf_w \left\{ \underbrace{(\lambda 1 + \cancel{\mu} + X^T \theta)}_{= 0} {}^T w \right\} - \theta^T y$$

$$g(\theta) = -\frac{1}{2}\|\theta\|_2^2 - \theta^T y$$

$$\Rightarrow \quad G_1(\theta) = \frac{1}{2}\|\theta\|_2^2 + \theta^T y$$

the dual problem is

$$\max \quad \frac{1}{2}\|\theta\|_2^2 + \theta^T y$$

$$\text{s.t.} \quad \lambda 1 + \mu + X\theta = 0$$

$$\mu \geq 0$$

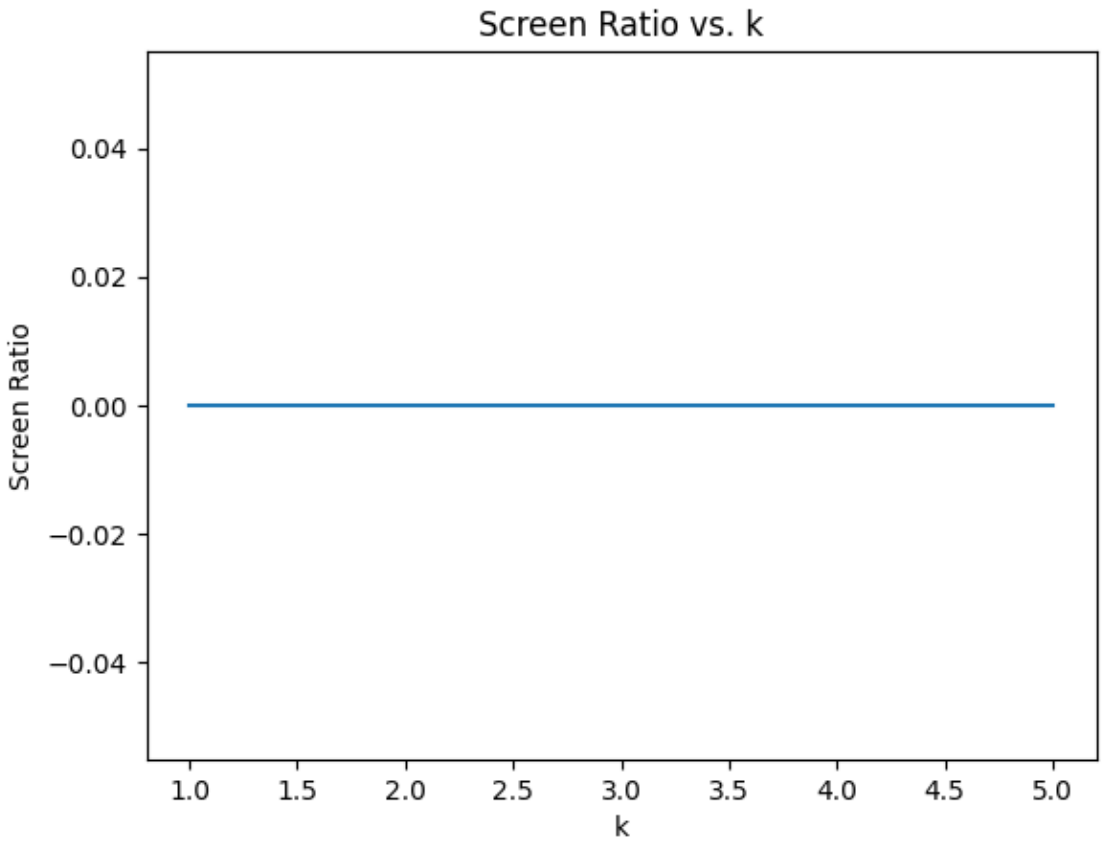
(f) I attempted parts i and ii with a significant amount of effort. I could not compute the screening ratios correctly even though I think my part (d) is correct. I printed out $\theta^*_{(0)}$ and $w^*_{(0)}$ which should have the correct values and left some comments in the code below.

[60]:
```
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt
import pickle

with open('restaurant_reviews.pkl', 'rb') as f:
    data = pickle.load(f)

X, y, name = data
X = np.array(X)
n, m = X.shape

lambda_0 = 2.5
r = 0.93
lambdas = [lambda_0 * r**k for k in range(6)]
gamma = 1  # assume strong convexity parameter is 1

# solve for w*(0) and theta*(0)
w = cp.Variable(m)

objective = 0.5 * cp.sum_squares(X @ w - y) + lambdas[0] * cp.norm(w, 1)
constraints = [w >= 0]

problem = cp.Problem(cp.Minimize(objective), constraints)
problem.solve()

w_star_0 = w.value
theta_star_0 = X @ w_star_0 - y

print("Theta_star_0: ", theta_star_0)
print("w_star_0: ", w_star_0)

def get_G(theta, y):
    return 0.5*(cp.square(cp.norm(theta))) + theta.T @ y

def get_F(w, x, y):
    return 0.5 * cp.sum_squares(X @ w - y)

# Apply feature elimination rule
screen_ratios = []

for k in range(1, 6):
    lambda_k = lambdas[k]
    theta_tilde = (lambda_k / lambda_0) * theta_star_0

    """
    If I were to implement part b:
```

```
    theta_tilde = (lambda_k / lambdas[k-1]) * theta_star_0
    w_tilde = w_star_0 / (lambda_k / lambdas[k-1])

    we are performing feature reduction so I'd expect the screen ratio to␣
↪decrease nearly uniformly at each k for part (i)

    for part ii I'd expect the monotonic decreasing fashion of k to stay the␣
↪same but with higher screen ratios like the ratios at each
    k are nearly reaching the same point.

    at each step more features are being eliminated as the optimization problem␣
↪progresses

    the theta_tilde and w_tilde are normalized factors less than one so as k␣
↪increases they are decreasing. As more features are eliminated,
    the screening ration decreasines because the proportion of remaining␣
↪features decreases in total.

    """

    G = get_G(theta_tilde,y)
    F = get_F(w_star_0,x,y)

    # Compute elimination condition
    elimination_condition = (cp.norm(X.T) * (cp.sqrt(2*(G + F)))) <= cp.
↪norm(lambda_k + X.T @ theta_tilde, axis=0)

    # Count features to be eliminated
    features_to_eliminate = np.sum(elimination_condition.value == True)
    screen_ratio = features_to_eliminate / m
    screen_ratios.append(screen_ratio)

    print(f"Screen Ratio for k={k}: {screen_ratio}")

# Plot screen ratio
plt.plot(range(1, 6), screen_ratios)
plt.xlabel("k")
plt.ylabel("Screen Ratio")
plt.title("Screen Ratio vs. k")
plt.show()
```

```
Theta_star_0:  [ 2.79429277  1.73116594  0.75485763 ...  1.66521198 -0.24037864
 -1.28251932]
w_star_0:  [-2.01732752e-08  1.02574477e-07 -3.03581145e-08 ...  3.12286701e-09
 -7.92038920e-07  3.86279571e+00]
Screen Ratio for k=1: 0.0
```

```
Screen Ratio for k=2: 0.0
Screen Ratio for k=3: 0.0
Screen Ratio for k=4: 0.0
Screen Ratio for k=5: 0.0
```



[ ]: