Question 2

q2.1

ראשית, נשים לב שהפונקציה $1 - y_i(\langle w, x_i\rangle + b)$ היא פונק' ליניארית ולכן קמורה, והפעלת המקסימום בינה לבין 0 יוצרת פונקציה מונוטונית לא יורדת $f(x_i, y_i) = max\{0, 1 - y_i(\langle w, x_i\rangle + b)\}$, שעבורה מתקיים לכל שתי נק' $(x_1, y_1), (x_2, y_2) \in S$: אם נעביר מיתר בין $f(x_1, y_1)$ ל-$f(x_2, y_2)$, כל נק' על המיתר תהיה מעל לכל נק' על גרף הפונק'. הסבר נוסף הוא זה שהפעלת מקסימום על פונק' קמורות היא פונק' קמורה.

שנית, נתבונן בביטוי $\lambda||w||^2$. מאחר שנורמה היא פונק' אי שלילית וקמורה, לפי ההערה מהשאלה נוכל להסיק שהעלתה בריבוע קמורה גם כן (ומכפלתה בקבוע λ לא משנה את הקמירות).

לבסוף, נשים לב שהביטויים הקמורים נסכמים זה אל זה, ומכאן שפונק' המטרה המדוברת קמורה כסכום של פונקציות קמורות (וכפל הביטוי הסופי בקבוע $\frac{1}{m}$ לא משנה את הקמירות).

q2.2

תהי הנק' (x_i, y_i), מטעמי נוחות נסמן את פונק' ה-hingeloss שמתאימה לנק' זו ב-$l(w)$. נזכיר שעל מנת להוכיח שפונק' זו היא R-ליפשיצית, נצטרך להראות שמתקיים $||l(w_1) - l(w_2)|| \leqslant R||w_1 - w_2||$ לכל w_1, w_2.

נחלק למקרים:

מקרה 1, שתי הפונק' חיוביות ממש, $l(w_1), l(w_2) > 0$:

$$||l(w_1) - l(w_2)|| = ||1 - y_i\langle w_1, x_i\rangle - (1 - y_i\langle w_2, x_i\rangle)|| =$$
$$= ||y_i\langle w_1, x_i\rangle - y_i\langle w_2, x_i\rangle|| = ||\langle w_1 - w_2, y_i x_i\rangle||$$

וכעת נפעיל את אי שיוויון קושי-שוורץ ונקבל:

$$||l(w_1) - l(w_2)|| \leqslant ||w_1 - w_2||*||y_i x_i|| = ||w_1 - w_2||*||x_i|| \leqslant ||w_1 - w_2||*max_k||x_k||$$

כאשר המעבר השני נובע מכך שנורמה היא תמיד חיובית ולכן בין אם y_i הוא 1 או 1- התוצאה זהה, והמעבר האחרון נובע מכך שהנורמה של הנק' x_i תמיד קטנה/שווה לנורמה של הנק' בעלת הנורמה המקסימלית מבין הנקודות במדגם.

מקרה 2, אחת מהפונק' חיובית ממש והשנייה שווה לאפס, בה"כ $l(w_1) = 0,\ l(w_2) > 0$:

נשים לב שבמקרה שבו נחפש את הנורמה של $l(w_2) = 1 - y_i\langle w_2, x_i\rangle$, הביטוי שבתוך הנורמה יהיה חיובי ממש, ולכן נוכל להגדיל אותו בידיעה שתוצאת חישוב הנורמה תגדל גם היא. בין אם y_i הוא 1 או 1-, תמיד מתקיים ש-$1 \leqslant y_i\langle w_2, x_i\rangle$, ולכן מתקיים: $||l(w_1) - l(w_2)|| = ||1 - y_i\langle w_2, x_i\rangle|| = ||y_i\langle w_1, x_i\rangle - y_i\langle w_2, x_i\rangle||$. כעת, מאותם נימוקים כמו במקרה 1 (ותוך שימוש בקושי-שוורץ באותו אופן), נקבל ש-
$||l(w_1) - l(w_2)|| \leqslant ||w_1 - w_2||*max_k||x_k||$.

מקרה 3, שתי הפונק' שוות לאפס, $l(w_1), l(w_2) = 0$: באופן טריוויאלי מתקיים ש-
$||l(w_1) - l(w_2)|| = ||0 - 0|| \leqslant ||w_1 - w_2||*max_k||x_k||$.

$\Leftarrow$ עבור $R = max_k||x_k||$, כל המקרים מקיימים $||l(w_1) - l(w_2)|| \leqslant ||w_1 - w_2||*max_k||x_k||$, ולכן ה-hinge loss היא R-ליפשיצית.

q2.3

מאחר שפונק' המטרה הנתונה מכילה max שבוחר את הערך המקסימלי בין 0 לבין $1 - y_i(\langle w, x_i \rangle + b)$, נצטרך לחלק את הסאב-גרדיאנט למקרים.

הסאב-גרדיאנט לפי w יהיה:

$$\begin{cases} \frac{d}{dw}\lambda||w||^2 = 2\lambda w & ,1 - y_i(\langle w, x_i \rangle + b) \leqslant 0 \\ \frac{d}{dw}\left(1 - y_i(\langle w, x_i \rangle + b) + \lambda||w||^2\right) = -y_i x_i + 2\lambda w & ,1 - y_i(\langle w, x_i \rangle + b) > 0 \end{cases}$$

הסאב-גרדיאנט לפי b יהיה:

$$\begin{cases} \frac{d}{db}\lambda||w||^2 = 0 & ,1 - y_i(\langle w, x_i \rangle + b) \leqslant 0 \\ \frac{d}{db}\left(1 - y_i(\langle w, x_i \rangle + b) + \lambda||w||^2\right) = -y_i & ,1 - y_i(\langle w, x_i \rangle + b) > 0 \end{cases}$$

q2.4

In [ ]:
```
import numpy as np
from random import randint
```

In [ ]:
```
def gradient_descent_step(prev_value, l_rate, grad):
    return prev_value - l_rate * grad

def weights_and_bias_per_iter(x, label, w, b, l_rate, lam):
    """ Update weights and bias for each epoch of the SGD algorithm. """
    if (1 - label * (np.dot(w, x) + b)) <= 0:
        w_grad, b_grad = (2 * lam * w), 0
    else:
        w_grad, b_grad = (2 * lam * w - label * x), -label

    w = gradient_descent_step(w, l_rate, w_grad)
    b = gradient_descent_step(b, l_rate, b_grad)
    return w, b
```

In [ ]:
```
def svm_with_sgd(X, y, lam=0, epochs=1000, l_rate=0.01, sgd_type='practical'):
    """ Params:
        X: Matrix of the train data, where each row is a sample.
        y: Vector of the train labels, where each element is the label of the co
        lam: Regularization parameter, a non-negative value.
        epochs: Number of iterations to perform on the training data.
        l_rate: Learning rate, the size of the step to take in the direction of
        sgd_type: The type of the stochastic gradient descent to use, either 'pr
    """
    np.random.seed(2)
    num_samples, num_features = X.shape  # m, d
    w = np.random.uniform(size=num_features)
    b = np.random.uniform()

    if sgd_type == 'practical':
        for _ in range(epochs):
            permutation = np.random.permutation(num_samples)
            for i in permutation:
                w, b = weights_and_bias_per_iter(X[i], y[i], w, b, l_rate, lam)
        return w, b

    if sgd_type == 'theory':
        epochs *= num_samples
```

```
        w = [w]
        b = [b]
        for epoch in range(epochs):
            i = randint(0, num_samples - 1)
            w_and_b_per_iter = weights_and_bias_per_iter(X[i], y[i], w[epoch], b
            w.append(w_and_b_per_iter[0])
            b.append(w_and_b_per_iter[1])

        w_bar = sum(w) / len(w)
        b_bar = sum(b) / len(b)
        return w_bar, b_bar
```

q2.5

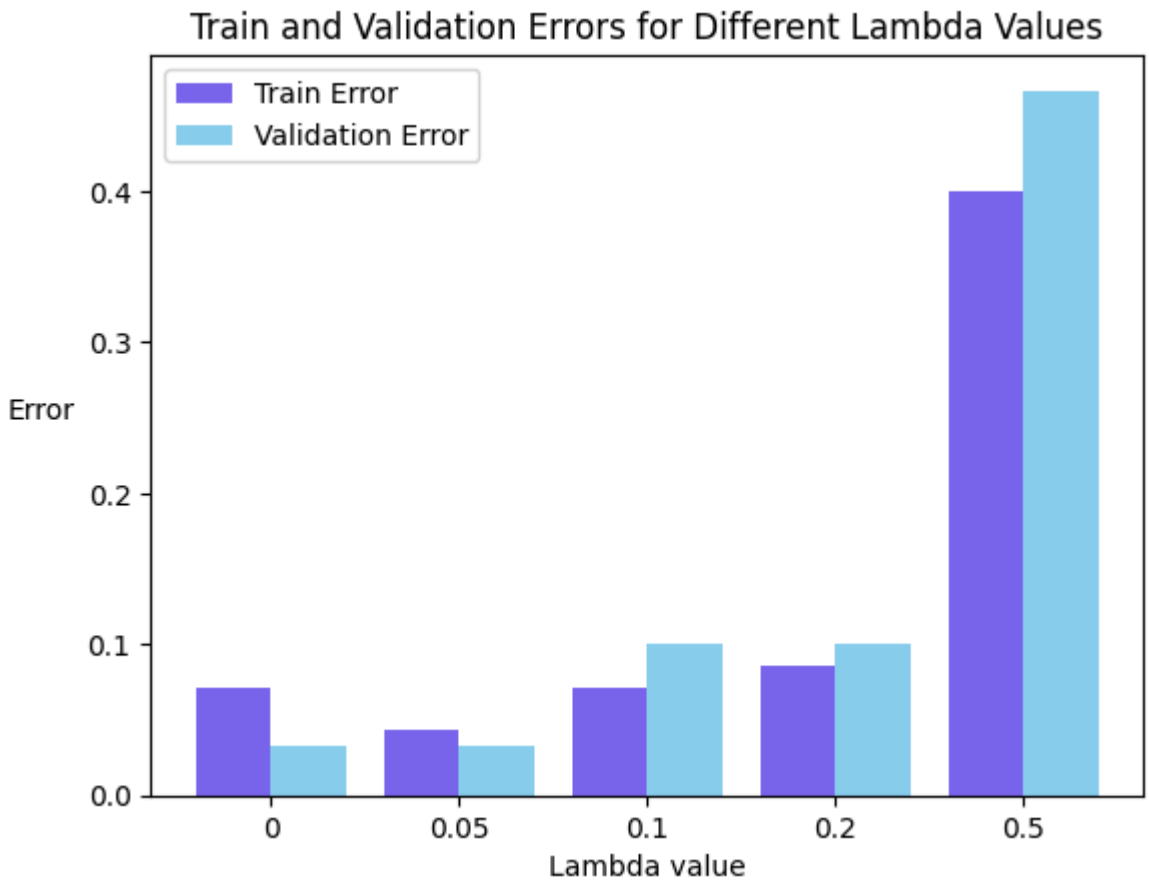
In [ ]:
```
def calculate_error(w, b, X, y):
    """ Params:
        w: The weights vector.
        b: The bias term.
        X: The dataset matrix, where each row is a sample.
        y: The labels vector, where each element is the label of the correspondi
    """
    y_pred = np.sign(np.dot(w, X.T) + b)
    return np.sum(y_pred != y) / len(y)
```

q2.6

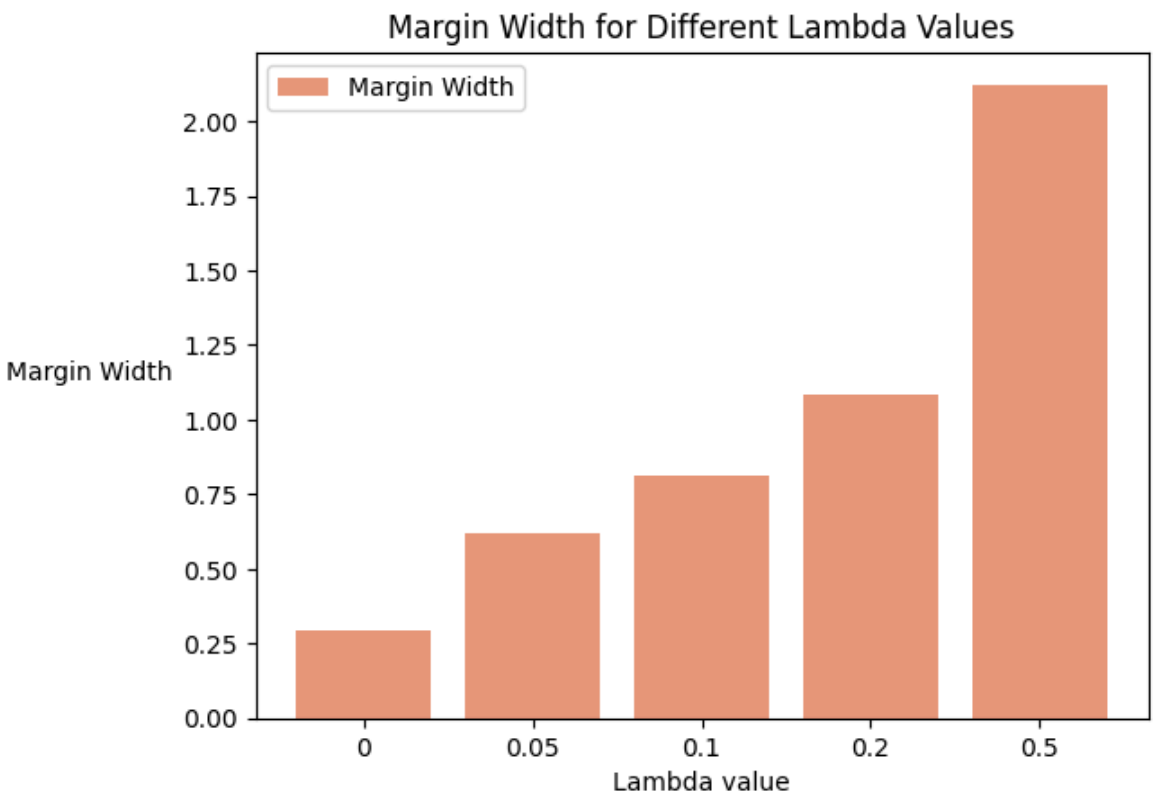
In [ ]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y=True)
non_zero_mask = y != 0
X = X[non_zero_mask]
y = y[non_zero_mask]
y[y == 2] = -1
X = X[:, 2:4]

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_st

lambdas = [0, 0.05, 0.1, 0.2, 0.5]
train_errs, val_errs, margin_widths = list(), list(), list()
for lam in lambdas:
    w, b = svm_with_sgd(X_train, y_train, lam=lam, sgd_type='practical')
    train_errs.append(calculate_error(w, b, X_train, y_train))
    val_errs.append(calculate_error(w, b, X_val, y_val))
    margin_widths.append(1 / np.linalg.norm(w))
```

In [ ]:
```
import matplotlib.pyplot as plt

x_axis = np.arange(len(lambdas))
bar_width = 0.4
plt.bar(x_axis - bar_width/2, train_errs, bar_width, label='Train Error', color=
plt.bar(x_axis + bar_width/2, val_errs, bar_width, label='Validation Error', col

plt.xticks(x_axis, lambdas)
plt.xlabel('Lambda value')
plt.ylabel('Error', rotation='horizontal', ha='right')
```

```
plt.legend()
plt.title('Train and Validation Errors for Different Lambda Values')
plt.show()
```



In [ ]:
```
plt.bar(x_axis, margin_widths, label='Margin Width', color='darksalmon')
plt.xticks(x_axis, lambdas)
plt.xlabel('Lambda value')
plt.ylabel('Margin Width', rotation='horizontal', ha='right')
plt.legend()
plt.title('Margin Width for Different Lambda Values')
plt.show()
```

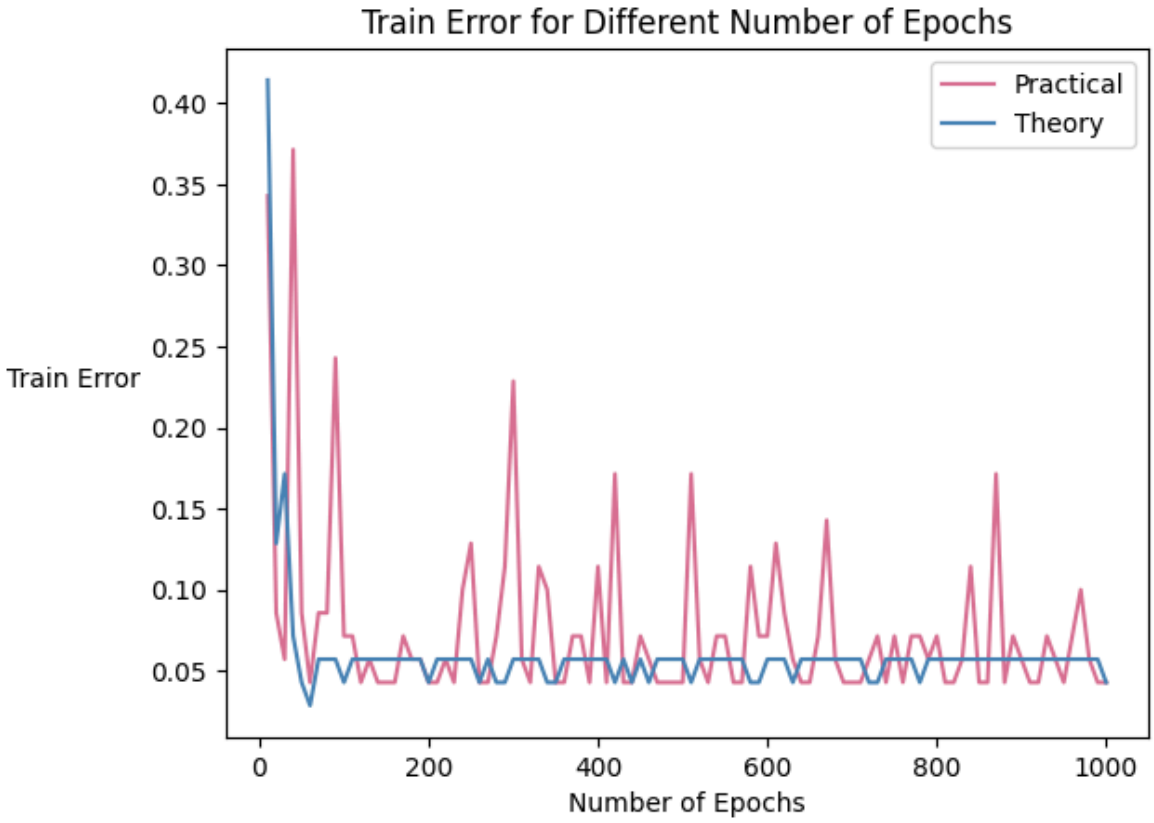


It seems that the best model is the one with lambda=0.05, because it minimizes the error on the test and train. The margin size is not too small, so we avoid overfitting, and it is also not too big, so we also avoid large error values.

q2.7

In [ ]:
```
def train_and_calculate_error(train_errs, val_errs, lam, num_epochs, sgd_type):
    """ Train the model and calculate the error for the given number of epochs a
    w, b = svm_with_sgd(X_train, y_train, lam=lam, epochs=num_epochs, sgd_type=s
    train_errs.append(calculate_error(w, b, X_train, y_train))
    val_errs.append(calculate_error(w, b, X_val, y_val))

lam = 0.05
num_epochs_list = list(range(10, 1001, 10))
train_errs_practical, train_errs_theory, val_errs_practical, val_errs_theory = l
for num_epochs in num_epochs_list:
    train_and_calculate_error(train_errs_practical, val_errs_practical, lam, num
    train_and_calculate_error(train_errs_theory, val_errs_theory, lam, num_epoch
```

In [ ]:
```
plt.plot(num_epochs_list, train_errs_practical, label='Practical', color='palevi
plt.plot(num_epochs_list, train_errs_theory, label='Theory', color='steelblue')
plt.xlabel('Number of Epochs')
plt.ylabel('Train Error', rotation='horizontal', ha='right')
plt.legend()
plt.title('Train Error for Different Number of Epochs')
plt.show()
```



In [ ]:
```
plt.plot(num_epochs_list, val_errs_practical, label='Practical', color='lightgre
plt.plot(num_epochs_list, val_errs_theory, label='Theory', color='crimson')
plt.xlabel('Number of Epochs')
plt.ylabel('Validation Error', rotation='horizontal', ha='right')
plt.legend()
plt.title('Validation Error for Different Number of Epochs')
plt.show()
```

In [ ]: