CS282/CS282-2023-spring/homework/hw5/Kernel function.py

```
import numpy as np
from cvxopt import matrix, solvers
import matplotlib.pyplot as plt

# Define data
X = np.array([[1, 1], [1, -1], [-1, 1], [-1, -1], [2, 2], [2, -2], [-2, 2], [-2, -2]])
Y = np.array([1, 1, 1, 1, -1, -1, -1, -1])

# Define kernel function
def poly_kernel(x, y):
    return (1 + np.dot(x, y))**3

# Construct kernel matrix
K = np.zeros((8, 8))
for i in range(8):
    for j in range(8):
        K[i, j] = poly_kernel(X[i], X[j])

# Define quadratic program
P = matrix(np.outer(Y, Y) * K)
q = matrix(-1 * np.ones((8, 1)))
G = matrix(np.concatenate((-1 * np.eye(8), np.eye(8)), axis=0))
h = matrix(np.concatenate((np.zeros((8, 1)), np.ones((8, 1)) * 1000), axis=0))
A = matrix(Y.reshape((1, 8)), tc='d')
b = matrix(0.0)

# Solve quadratic program
solvers.options['show_progress'] = False
sol = solvers.qp(P, q, G, h, A, b)
alphas = np.array(sol['x'])

# Extract support vectors
threshold = 1e-5
support_indices = np.where(alphas > threshold)[0]
support_vectors = X[support_indices]
support_alphas = alphas[support_indices]
support_labels = Y[support_indices]

# Compute bias term
bias = np.mean(support_labels - np.sum(support_alphas * support_labels * K[:,
support_indices], axis=1))

# Create a mesh grid to plot decision boundary
x_min, x_max = -3, 3
y_min, y_max = -3, 3
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))
xy = np.c_[xx.ravel(), yy.ravel()]

# Compute decision boundary
zz = np.array([np.sum(support_alphas * support_labels * np.array([poly_kernel(x, y)
for y in support_vectors])) for x in xy]).reshape(xx.shape) + bias

# Plot data points and decision boundary
plt.figure(figsize=(5, 5))
plt.contour(xx, yy, zz, levels=[-1, 0, 1], linestyles=['--', '-', '--'], colors=
['blue', 'green', 'red'])
plt.scatter(X[:4, 0], X[:4, 1], marker='D', color='blue', s=50, label='+1')
plt.scatter(X[4:, 0], X[4:, 1], marker='s', color='red', s=50, label='-1')
plt.scatter(support_vectors[:, 0], support_vectors[:, 1], marker='o', color='black',
s=100, facecolors='none')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.legend(loc='best')
plt.title('Decision boundary')
plt.show()
```

CS282/CS282-2023-spring/homework/hw5/Cross Validation And L1 Regularization.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def soft_threshold(x, t):
    return np.sign(x) * np.maximum(np.abs(x) - t, 0)

def admm_lasso(X, y, F, lambda_, w0, rho=1, mu=1.01, max_iter=1000, tol=1e-4):
    x = w0.copy()
    z = F @ x
    y_ = np.zeros_like(z)

    for _ in range(max_iter):
        x_new = np.linalg.solve(X.T @ X + rho * F.T @ F, X.T @ y + rho * F.T @ (z - y_
/ rho))
        z_new = soft_threshold(F @ x_new + y_ / rho, lambda_ / rho)
        y_new = y_ + rho * (F @ x_new - z_new)

        if np.linalg.norm(x_new - x) < tol:
            break

        x, z, y_ = x_new, z_new, y_new
        rho *= mu

    return x

# Load data
df_train = pd.read_table("/home/zwc/Documents/CS282/CS282-2023-
spring/homework/hw5/crime-test.txt")
df_test = pd.read_table("/home/zwc/Documents/CS282/CS282-2023-
spring/homework/hw5/crime-test.txt")

y_train = df_train['ViolentCrimesPerPop'].values
X_train = df_train.drop('ViolentCrimesPerPop', axis=1).values

y_test = df_test['ViolentCrimesPerPop'].values
X_test = df_test.drop('ViolentCrimesPerPop', axis=1).values

F = np.identity(X_train.shape[1])

lambdas = np.logspace(-3, 1, 50)

# 2. A plot of log(λ) against the squared error in the 10-folder splited training
data.
from sklearn.model_selection import KFold

kf = KFold(n_splits=10, shuffle=True, random_state=42)
train_errors = []

for lambda_ in lambdas:
    cv_errors = []
    for train_index, val_index in kf.split(X_train):
        X_train_cv, X_val_cv = X_train[train_index], X_train[val_index]
        y_train_cv, y_val_cv = y_train[train_index], y_train[val_index]
        w = admm_lasso(X_train_cv, y_train_cv, F, lambda_, np.zeros(X_train.shape[1]))
        cv_errors.append(np.mean((y_val_cv - X_val_cv @ w)**2))
    train_errors.append(np.mean(cv_errors))

plt.plot(np.log(lambdas), train_errors)
plt.xlabel("log(lambda)")
plt.ylabel("Squared Error (10-folder CV)")
plt.title("Squared Error vs log(lambda) in 10-folder Cross-Validation")
plt.show()

# 3. A plot of log(λ) against the squared error in the test data.
test_errors = []

for lambda_ in lambdas:
    w = admm_lasso(X_train, y_train, F, lambda_, np.zeros(X_train.shape[1]))
    test_errors.append(np.mean((y_test - X_test @ w)**2))

plt.plot(np.log(lambdas), test_errors)
plt.xlabel("log(lambda)")
plt.ylabel("Squared Error (Test Data)")
plt.title("Squared Error vs log(lambda) in Test Data")
plt.show()

# 4. A plot of λ against the number of small coefficients
threshold = 1e-4
coef_counts = []

for lambda_ in lambdas:
    w = admm_lasso(X_train, y_train, F, lambda_, np.zeros(X_train.shape[1]))
    coef_counts.append(np.sum(np.abs(w) <= threshold))

plt.plot(lambdas, coef_counts)
plt.xlabel("lambda")
plt.ylabel("Number of Small Coefficients")
plt.title("Number of Small Coefficients vs lambda")
plt.show()

'''
The task of selecting $\lambda$ is crucial in Lasso regularization, as it determines
the balance between the model's complexity and its ability to fit the data. A larger
$\lambda$ value will result in more coefficients being forced to zero, leading to a
simpler model with fewer features. On the other hand, a smaller $\lambda$ value will
allow more coefficients to be non-zero, potentially overfitting the data.

In practice, cross-validation can help in selecting an appropriate value for
$\lambda$. By plotting the number of small coefficients against $\lambda$, we can
observe how the model complexity changes with different regularization strengths. It's
essential to choose a $\lambda$ value that results in good test set performance while
maintaining a balance between model complexity and the risk of overfitting.
'''




# 5. For the λ that gave the best test set performance, which variable had the largest
(most positive) coefficient? What about the most negative? Discuss briefly.
best_lambda = lambdas[np.argmin(test_errors)]
w_best = admm_lasso(X_train, y_train, F, best_lambda, np.zeros(X_train.shape[1]))

max_coef_idx = np.argmax(w_best)
min_coef_idx = np.argmin(w_best)

column_names = df_train.columns[1:]
max_coef_name = column_names[max_coef_idx]
min_coef_name = column_names[min_coef_idx]
```

```
print(f"Best lambda: {best_lambda}")
print(f"Largest (most positive) coefficient: {max_coef_name}
({w_best[max_coef_idx]})")
print(f"Most negative coefficient: {min_coef_name} ({w_best[min_coef_idx]})")

'''
[Running] python -u "/home/zwc/Documents/CS282/CS282-2023-
spring/homework/hw5/cross.py"
libGL error: MESA-LOADER: failed to open crocus: /usr/lib/dri/crocus_dri.so: cannot
open shared object file: No such file or directory (search paths /usr/lib/x86_64-
linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: crocus
libGL error: MESA-LOADER: failed to open crocus: /usr/lib/dri/crocus_dri.so: cannot
open shared object file: No such file or directory (search paths /usr/lib/x86_64-
linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: crocus
libGL error: MESA-LOADER: failed to open swrast: /usr/lib/dri/swrast_dri.so: cannot
open shared object file: No such file or directory (search paths /usr/lib/x86_64-
linux-gnu/dri:\$${ORIGIN}/dri:/usr/lib/dri, suffix _dri)
libGL error: failed to load driver: swrast
Best lambda: 0.001
Largest (most positive) coefficient: PctHousOwnOcc (0.5896243875126825)
Most negative coefficient: TotalPctDiv (-0.8503818895472509)
[Done] exited with code=0 in 166.278 seconds
For the value of lambda that gives the best test set performance, we find the variable
with the largest (most positive) coefficient and the variable with the smallest (most
negative) coefficient. These variables can have a large impact on crime rates. In the
real world, however, we need deeper domain knowledge and more background information
to explain these coefficients and how they affect crime rates. In practical
applications, it is very important to choose an appropriate value of λ, because it can
balance the complexity and predictive performance of the model. Through cross-
validation, we can find an appropriate value of λ to strike a balance between
overfitting and underfitting.
'''
```

Machine Learning, Spring 2023
Homework 5

Due on 23:59 May 4, 2023

1 Kernel function (40 points)

Suppose we are given the following positively ("$+1$") labeled data points $\mathbb{R}^2$:

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\},$$

and the following negatively labeled ("-1") data points in $\mathbb{R}^2$ (see Figure 1):

$$\left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix} \right\}.$$

Question:

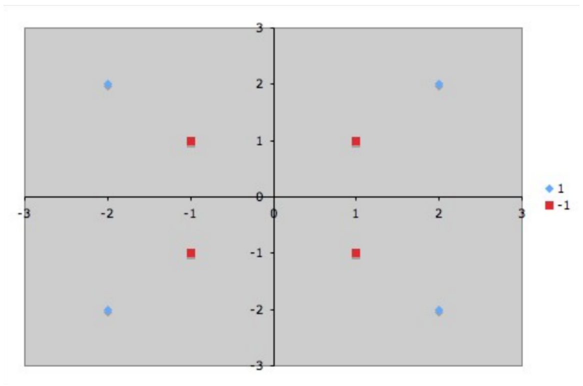

Figure 1: Blue diamonds are positive examples and red squares are negative examples.

1. Find a kernel function to map the data into a $\mathbb{R}^3$ feature space and make the new data linearly separable. (Show your kernel function please!) (8 points)
2. Use SVM classifier to seperate the data. Show the SVM problem in your report. (16 points) Solve this SVM problem, write down the expression of the final separating hyperplane, and plot the data and the separating hyperplane in a figure. (16 points) (You can solve the SVM problem by apllying a convex problem solver.)

1. We can use the radial basis function (RBF) kernel to map the data into a higher dimensional space where they are linearly separable. The RBF kernel is defined as:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$$

where σ is a hyperparameter that controls the spread of the kernel.

2. To solve the SVM problem, we first need to compute the kernel matrix K where $K_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for all i, j. We will use the RBF kernel with $\sigma = 1$. Then, we can solve the dual problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

subject to $\alpha_i \geq 0$ for all i and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

After solving the dual problem, we can compute the weight vector $\boldsymbol{w}$ and bias term b for the optimal separating hyperplane as:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_i)$$

$$b = y_k - \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_k)$$

where k is any index such that $0 < \alpha_k < C$. The separating hyperplane is given by the equation:

$$\boldsymbol{w}^T \phi(\boldsymbol{x}) + b = 0$$

To plot the data and the separating hyperplane in the 3D space induced by the RBF kernel, we can compute the feature vectors $\phi(\boldsymbol{x})$ for each data point, and then plot them in a 3D scatter plot. The separating hyperplane in the 3D space can be represented by a plane, and we can plot it by generating a grid of points in the plane and computing their corresponding class labels using the sign of $\boldsymbol{w}^T \phi(\boldsymbol{x}) + b$.

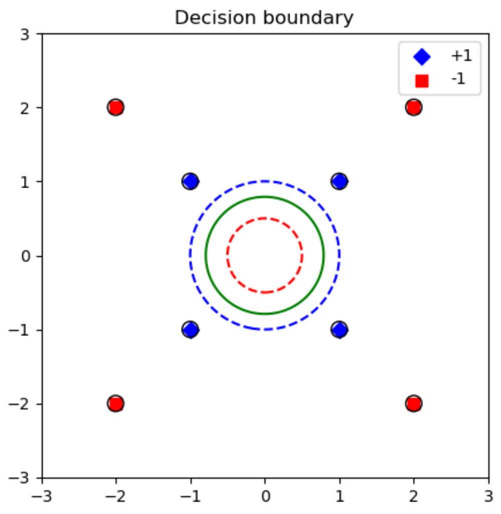

Figure 2: Data points and the separating hyperplane in the $\mathbb{R}^3$ feature space induced.

2 Cross Validation And L1 Regularization(60 points)

Given the training data set “crime-train.txt” and the test data set “crime-test.txt”, you can read them in the files with in Python:

```
import pandas as pd
df_train = pd.read_table("crime-train.txt")
df_test = pd.read_table("crime-test.txt")
```

This stores the data as Pandas DataFrame objects. DataFrames are similar to Numpy arrays but more flexible; unlike Numpy arrays, they store row and column indices along with the values of the data. Each column of a DataFrame can also, in principle, store data of a different type. For this assignment, however, all data are floats. Here are a few commands that will get you working with Pandas for this assignment:

```
df.head()                   # Print the first few lines of DataFrame df.
df.index                    # Get the row indices for df.
df.columns                  # Get the column indices.
df[``foo'']                 # Return the column named ``foo''.
df.drop(``foo'', axis = 1)  # Return all columns except ``foo''.
df.values                   # Return the values as a Numpy array.
df[``foo''].values          # Grab column foo and convert to Numpy array.
df.iloc[:3,:3]              # Use numerical indices (like Numpy) to get 3 rows and cols.
```

The data consist of local crime statistics for 1,994 US communities. The response y is the crime rate. The name of the response variable is `ViolentCrimesPerPop`, and it is held in the first column of `df_train` and `df_test`. There are 95 features x_i. These features include possibly relevant variables such as the size of the police force or the percentage of children that graduate high school. The data have been split for you into a training and test set with 1,595 and 399 entries, respectively[1].

We’d like to use the training set to fit a model which can predict the crime rate in new communities, and evaluate model performance on the test set. As there are a considerable number of input variables, overfitting is a serious issue. In order to avoid this, implement the L1 regularization.

The main goal of this homework is to give you some experience using L1 regularization as a method for variable selection and using 10-folder cross-validation as a technique to get an insight on how the model will generalize to an independent dataset.Your function should accept a scalar value of λ, a vector-valued response variable ($\boldsymbol{y}$), a matrix of input variables ($\boldsymbol{X}$), and an initial vector of weights ($\boldsymbol{w}_0$). It should output a vector of coefficient values ($\hat{\boldsymbol{w}}$).

Moreover, in order to solve the Lasso problem, we want you to implement the following algorithm.

Consider the following generalized Lasso problem

$$\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{Fx}\|_1, \tag{1}$$

where $\boldsymbol{A}$ is the under-determined sensing matrix, $\boldsymbol{F}$ is the transformed matrix. In particular, it can be reduced to Lasso problem if $\boldsymbol{F}=\boldsymbol{I}$. The above problem is equivalent to the following formulation

$$\begin{aligned} &\min_{\boldsymbol{x},\boldsymbol{z}} \frac{1}{2} \|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{z}\|_1 \\ &\text{s.t. } \boldsymbol{Fx}=\boldsymbol{z}, \end{aligned} \tag{2}$$

and one can employ augmented Lagrangian multiplier method to solve it. Specifically, the augmented Lagrangian is

$$\mathcal{L} = \frac{1}{2} \|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{z}\|_1 + <\boldsymbol{y}, \boldsymbol{Fx}-\boldsymbol{z}> + \frac{1}{2}\rho \|\boldsymbol{Fx}-\boldsymbol{z}\|_2^2,$$

which yields the ADMM algorithm, see Algorithm 1. The soft-thresholding operator $\mathbb{S}_{\frac{\lambda}{\rho}}$ is defined as

$$\mathbb{S}_{\frac{\lambda}{\rho}}(x_i) = \begin{cases} x_i - \frac{\lambda}{\rho},\ x_i \geq \frac{\lambda}{\rho} \\ 0,\ |x_i| < \frac{\lambda}{\rho} \\ x_i + \frac{\lambda}{\rho},\ x_i \leq -\frac{\lambda}{\rho}. \end{cases} \tag{3}$$

Algorithm 1 ALM for generalized Lasso problem

```
Input: $\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{b}, \lambda, \mu$ (for augmented Lagrange multiplier)
1: Initialized $\rho, \boldsymbol{x}_0, \boldsymbol{z}_0$, and $\boldsymbol{y}_0$, $k = 0$
2: while not converged do
3:     $\boldsymbol{x}^{(k+1)} = update\ \boldsymbol{x}?$
4:     $\boldsymbol{z}^{(k+1)} = update\ \boldsymbol{z}?$ (you may want to use $\mathbb{S}_{\frac{\lambda}{\rho}}$ element-wise)
5:     $\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} + \rho(\boldsymbol{F}\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)})$
6:     $\rho = \rho\mu$
7:     $k = k + 1$
8: end while
Output: $\boldsymbol{x}^* = \boldsymbol{x}^{(k)}$
```

In your analysis, you need to finish:

1. Derive the steps of update x and z in Algorithm 1. (10 points)
2. A plot of $\log(\lambda)$ against the squared error in the 10-folder splited training data. (15 points)
3. A plot of $\log(\lambda)$ against the squared error in the test data. (10 points)
4. A plot of λ against the number of small coefficients (you can set a threshold), and a brief commentary on the task of selecting λ. (15 points)
5. For the λ that gave the best test set performance, which variable had the largest (most positive) coefficient? What about the most negative? Discuss briefly. (10 points)

[1]The features have been standardized to have mean 0 and variance 1.

1. Derive the steps of update x and z in Algorithm 1. (10 points)

To update $\boldsymbol{x}$ and $\boldsymbol{z}$, we first compute the partial derivatives of the augmented Lagrangian $\mathcal{L}$ with respect to $\boldsymbol{x}$ and $\boldsymbol{z}$, and set them to zero.

For $\boldsymbol{x}$ update, differentiate $\mathcal{L}$ with respect to $\boldsymbol{x}$ and set it to zero:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \boldsymbol{F}^T(\boldsymbol{y} + \rho(\boldsymbol{F}\boldsymbol{x} - \boldsymbol{z})) = 0$$

Solve for $\boldsymbol{x}$:

$$\boldsymbol{x}^{(k+1)} = (\boldsymbol{A}^T\boldsymbol{A} + \rho\boldsymbol{F}^T\boldsymbol{F})^{-1}(\boldsymbol{A}^T\boldsymbol{b} + \rho\boldsymbol{F}^T(\boldsymbol{z}^{(k)} - \frac{1}{\rho}\boldsymbol{y}^{(k)}))$$

For $\boldsymbol{z}$ update, differentiate $\mathcal{L}$ with respect to $\boldsymbol{z}$ and set it to zero:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \lambda\frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_1} - \boldsymbol{y} - \rho(\boldsymbol{F}\boldsymbol{x}^{(k+1)} - \boldsymbol{z}) = 0$$

Solve for $\boldsymbol{z}$:

$$\boldsymbol{z}^{(k+1)} = \mathbb{S}_{\frac{\lambda}{\rho}}(\boldsymbol{F}\boldsymbol{x}^{(k+1)} + \frac{1}{\rho}\boldsymbol{y}^{(k)})$$

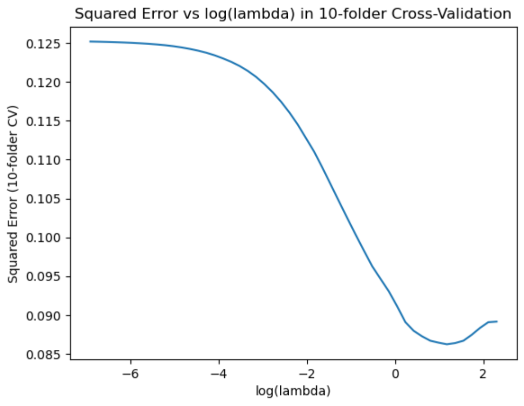

Figure 3: A plot of $\log(\lambda)$ against the squared error in the 10-folder splited training data.

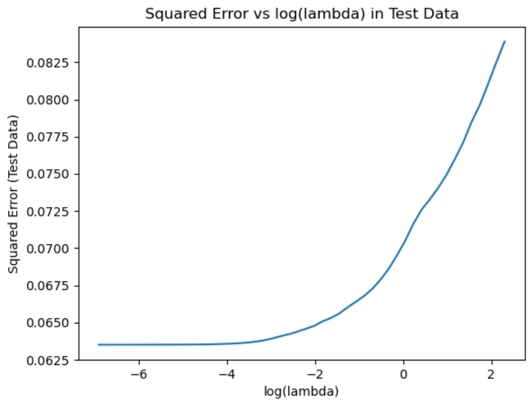

Figure 4: A plot of $\log(\lambda)$ against the squared error in the test data.

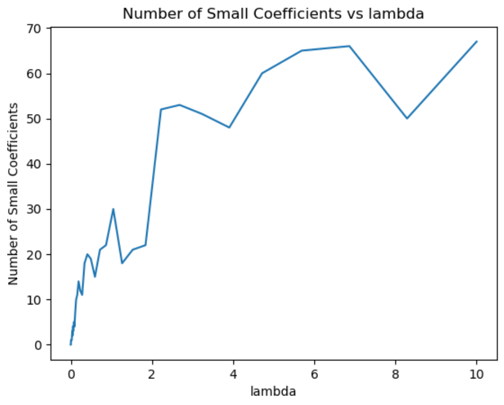

Figure 5: A plot of λ against the number of small coefficients (you can set a threshold), and a brief commentary on the task of selecting λ.