Problem 1

In [67]:
```
# EECS 545 FA21 HW5 - Kernel Logistic Regression
import numpy as np
from sklearn.metrics.pairwise import rbf_kernel, linear_kernel
```

In [68]:
```
# linear logistic regression
def linear_logistic_regression(x_train, y_train, x_test, y_test, step_size, reg_strength, num_iters):
    from sklearn.linear_model import LogisticRegression
    # only use sklearn's LogisticRegression
    clf = LogisticRegression(C=1/reg_strength)
    clf.fit(x_train, y_train)
    test_acc = clf.score(x_test, y_test)
    return test_acc
```

In [69]:
```
# kernal logistic regression
def kernel_logistic_regression(x_train, y_train, x_test, y_test, step_size, reg_strength, num_iters, kernel_parameter):
    """
    x_train - (n_train, d)
    y_train - (n_train,)
    x_test - (n_test, d)
    y_test - (n_test,)
    step_size: gamma in problem description
    reg_strength: lambda in problem description
    num_iters: how many iterations of gradient descent to perform

    Implement KLR with the Gaussian Kernel.
    The only allowed sklearn usage is the rbf_kernel, which has already been imported.
    """
    # TODO
    ntrain = x_train.shape[0]
    nfeatures = x_train.shape[1]
    ntest = x_test.shape[0]

    # create kernel matrices
    ker_train = rbf_kernel(x_train,x_train,gamma = kernel_parameter)
    ker_test = rbf_kernel(x_train,x_test,gamma = kernel_parameter)

#     sanity check puroposes
#     ker_train = linear_kernel(x_train,x_train)
#     ker_test = linear_kernel(x_train,x_test)

    ### do gradient descent
    # set initial parameter
    alp = np.zeros(ntrain)
    b = 1e-7
    for i in range(num_iters):
        update_mat = np.array([-y_train[j]/(1 + np.exp(y_train[j]*(np.dot(alp,ker_train[:,j]) + b))) for j in range(ntrain)])
        b -= step_size*(1/ntrain*np.sum(update_mat))
        alp -= step_size*(1/ntrain*update_mat + 2*reg_strength*alp)

    y_pred = np.ones(ntest)
    # apply classifier on test set
    eta = np.array([1/(1 + np.exp(-(np.dot(alp,ker_test[:,j]) + b))) for j in range(ntest)])
    y_pred[eta < 1/2] = -1
    test_acc = np.sum(y_pred == y_test)/ntest
    return test_acc
```

In [71]:
```
x_train = np.load("x_train.npy")    # shape (n_train, d)
x_test = np.load("x_test.npy")      # shape (n_test, d)

y_train = np.load("y_train.npy")    # shape (n_train,)
y_test = np.load("y_test.npy")        # shape (n_test,)

linear_acc = linear_logistic_regression(x_train, y_train, x_test, y_test, 1.0, 0.001, 200)
print("Linear LR accuracy:", linear_acc)

klr_acc = kernel_logistic_regression(x_train, y_train, x_test, y_test, 5.0, 0.001,200, 0.1)

# sanity check
# klr_acc = kernel_logistic_regression(x_train, y_train, x_test, y_test, 1.0, 0.001,200, 0.1)

print("Kernel LR accuracy:", klr_acc)
```

```
Linear LR accuracy: 0.769
Kernel LR accuracy: 0.796
```