In [1]:

```
# importing modules
import numpy as np
import numpy.linalg as LA
import math
from sklearn.naive_bayes import GaussianNB
```

In [2]:

```
f = open("/home/tandon/IIIT-H/3rd/SMAI/arcene/arcene_train.data")
X = np.zeros((100, 10000))
row = 0
col = 0
for line in f:
    for token in line.split():
        X[row, col] = int(token)
        col += 1
    col = 0
    row += 1
```

In [3]:

```
X[0,9999]
```

Out[3]:

524.0

In [5]:

```
# Creating the RBF Kernel matrix
def kernel_pca(X, gamma, components):
    n = X.shape[0]
    K = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            difference = X[i,:] - X[j,:]
            dif_norm = LA.norm(difference)
#             print('dif norm', dif_norm)
            prod = -1.0 * gamma * dif_norm
            K[i, j] = np.exp(prod)
#             print('k val', K[i, j])

    onen = np.ones((n, n))/n
    # Centering the Kernel matrix
    K = K - onen.dot(K) - K.dot(onen) + onen.dot(K).dot(onen)

    eigen_values, eigen_vectors = LA.eig(K)
#     print('Eigen vecs', eigen_values.shape)
    # Contains indices for largest to smallest eigen values
    idx = eigen_values.argsort()[::-1]
    alphas = eigen_vectors[:, idx[0:components]]
    lambdas = eigen_values[idx[0:components]]
    return alphas/lambdas


Q = kernel_pca(X, 0.025,3)
```

In [ ]:

```
def project_point(point, X, alphas, gamma):
    kvec = np.array[np.sum((point - row)**2) for row in X]
    kvec = np.exp(- gamma * kvec)
    print('Projected point')
```

In [53]:

```
asq = np.array([[1,2,3], [4,5,6], [7,8,9]])
asd = np.array([2,3,4])
print('Original\n', asq)
print('Broadcasted\n', asq*asd)
```

```
Original
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Broadcasted
 [[ 2  6 12]
 [ 8 15 24]
 [14 24 36]]
```

In [56]:

```
X.shape
np.exp(-146936)
```

Out[56]:

```
0.0
```

In [ ]: