Problem 1.

① $k(u,v) = \alpha k_1(u,v) + \beta k_2(u,v)$ for $\alpha, \beta \geq 0$

proof:

$$\alpha k_1(u,v) = \langle \sqrt{\alpha}\phi_1(u), \sqrt{\alpha}\phi_1(v) \rangle, \quad \beta k_2(u,v) = \langle \sqrt{\beta}\phi_2(u), \sqrt{\beta}\phi_2(v) \rangle$$

$$\begin{aligned} k(u,v) &= \alpha k_1(u,v) + \beta k_2(u,v) \\ &= \langle \sqrt{\alpha}\phi_1(u), \sqrt{\alpha}\phi_1(v) \rangle + \langle \sqrt{\beta}\phi_2(u), \sqrt{\beta}\phi_2(v) \rangle \\ &= \langle [\sqrt{\alpha}\phi_1(u), \sqrt{\beta}\phi_2(u)], [\sqrt{\alpha}\phi_1(v), \sqrt{\beta}\phi_2(v)] \rangle \end{aligned}$$

Thus $k(u,v)$ can be expressed as an inner product.

② $k(u,v) = k_1(u,v)k_2(u,v)$

proof: Let $f_i(x)$ be the i^{th} feature value under the feature map ϕ_1, $g_i(x)$ be the i^{th} feature value under the feature map ϕ_2.

$$\begin{aligned} k(x,y) &= k_1(x,y)k_2(x,y) \\ &= (\phi_1(x)\cdot\phi_1(y))(\phi_2(x)\cdot\phi_2(y)) \\ &= \left(\sum_{i=0}^{\infty} f_i(x)f_i(y)\right)\left(\sum_{j=0}^{\infty} g_j(x)g_j(y)\right) \\ &= \sum_{i,j} f_i(x)f_i(y)g_j(x)g_j(y) \\ &= \sum_{i,j} (f_i(x)g_i(x))(f_i(x)g_j(y)) \\ &= \langle \phi_3(x), \phi_3(y) \rangle \end{aligned}$$

where ϕ_3 has feature $h_{i,j}(x) = f_i(x)g_j(x)$

③ $k(u,v) = k_1(f(u), f(v))$, where $f: \mathcal{X} \to \mathcal{X}$

Since each polynomial term is a product of kernels with a positive coefficient, the proof follows from part ① and ②.

④ $k(u,v) = g(u)g(v)$, where $g: \chi \to \mathbb{R}$

proof:

We can express the gram matrix K as the outer product of the vector $V = [g(x_1), \cdots, g(x_n)]'$. Hence, K is symmetric and positive semi-definite with rank 1. (It is positive semi-definite because the non-zero eigenvalue of VV' is the trace of VV' which is simply $V'V$ which is greater than or equal to 0).

⑤ $k(u,v) = f(k_1(u,v))$, where f is a polynomial with positive coeffecients

proof:

Since each polynomial term is a product of kernels with a positive coefficient, the proof follows by applying part ① and ②.

⑥ $k(u,v) = \exp(k_1(u,v))$

By Taylor expansion:

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Then the proof follows part ③.

① $k(u,v) = \exp\left(\frac{-\|u-v\|^2}{\sigma^2}\right)$

proof:

$$k(x,z) = \exp\left(\frac{-\|u-v\|^2}{\sigma^2}\right) = \exp\left(\frac{-\|u\|^2 - \|v\|^2 + 2u^Tv}{\sigma^2}\right)$$

$$= \left(\exp\left(\frac{-\|u\|^2}{\sigma^2}\right)\exp\left(\frac{-\|v\|^2}{\sigma^2}\right)\right)\exp\left(\frac{2u^Tv}{\sigma^2}\right)$$

$g(x)\,g(z)$ is a kernel according to 4, and $\exp(k_1(x,z))$ is a kernel according to part ⑥. According to part 2, the product

of two kernels is a valid kernel.

Note that the Gaussian kernel is translation invariant, since it only depends on $x - z$.

Problem 2

First, split the dataset in to two halves for cross validation. Then, we train SVM using linear kernel, polynomial kernel and RBF kernel with different parameters and costs. Testing errors are plotted in the following 3 graphs.

```
Execution time:  0.0 seconds
Status : □
|w0|^2    : 0.530721
Margin    : 2.745345
Sum alpha : 0.530720
Support Vectors : 14 (19.4%)

err =

     0

fx >>
```

Figure 1 the error of SVM using linear kernel

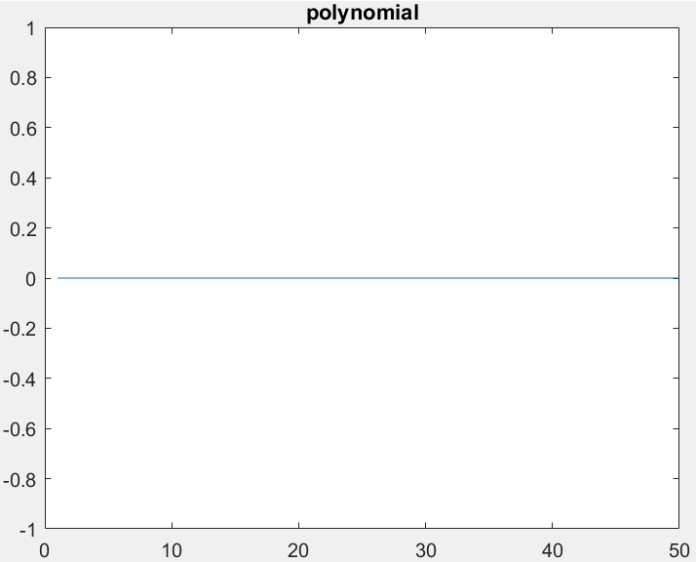

Figure 2 the error of SVM using polynomial kernel

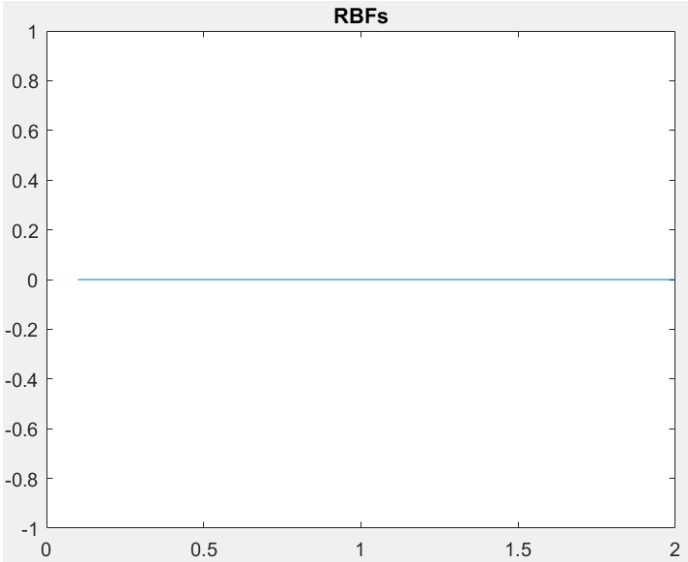

Figure 3 the error of SVM using RBF kernel

Code:

```
%% problem 2
load('shoesducks.mat');
% plot(X(12,:));

% split the dataset
trnX = X( 1:72,: );trnY = Y( 1:72,: );
tstX = X( 73:144,: );tstY = Y( 73:144,: );


% classify using linear
[nsv, alpha, b0] = svc(trnX,trnY);
predictedY = svcoutput(trnX,trnY,tstX,'linear',alpha,b0,0);
err = svcerror(trnX,trnY,tstX,tstY,'linear',alpha,b0)
```

```
global p1;
% classify using polynomials
d_max = 50;
err_d = zeros( 1,d_max );
for d=1:d_max
    p1 = d;
    % train
    [nsv,alpha,b0] = svc(trnX,trnY,'poly');
    % predict
    predictedY = svcoutput(trnX,trnY,tstX,'poly',alpha,b0);
    % compute test error
    err_d(d)= svcerror(trnX,trnY,tstX,tstY,'poly',alpha,b0);
end
% plot error vs polynomial degree
f = figure(1);
clf(f);
plot( 1:d_max,err_d );
title('polynomial');
print( f,'-depsc','poly.eps' );


% classify using rbfs
sigmas = .1:.1:2;
err_sigma = zeros( 1,numel(sigmas) );
for sigma_i=1:numel(sigmas)
    p1 = sigmas(sigma_i);
    % train
    [ nsv,alpha,bias ]=svc(trnX,trnY,'rbf',inf);
    % predict
    predictedY = svcoutput( trnX,trnY,tstX,'rbf',alpha,bias );
    % compute test error
    err_sigma(sigma_i)= svcerror( trnX,trnY,tstX,tstY,'rbf',alpha,bias );
end
% plot error vs polynomial degree
f2 = figure(2);
clf(f2);
plot(sigmas,err_sigma);
title('RBFs');
print(f2,'-depsc','rbf.eps');
```

Problem 3

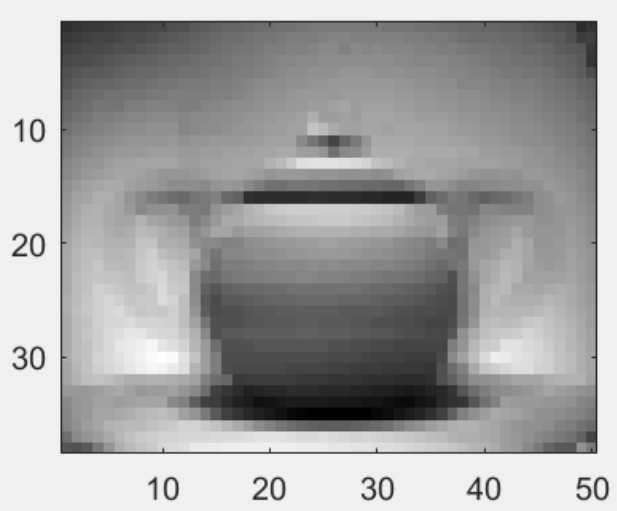

Figure 3, the data mean

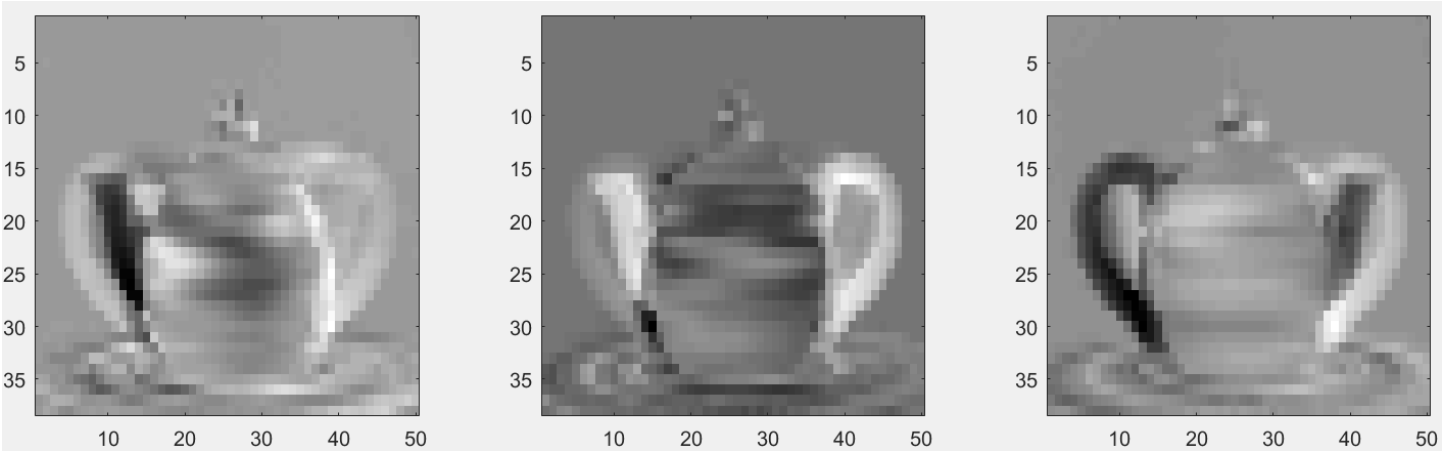

Figure 4, top 3 eigenvectors of the data covariance matrix

Figure 5, 10 different images before and after reconstruction

Discussion:

The reconstructed images can show the basic shape and main features of the teapot. But considering the original images show the teapot from different angles, it seems that the reconstructed images cannot contain this “angle” information. So, we can draw the conclusion that PCA can get the most of the information but not all the details.