ML Homework 3 Solution

Han Wang (hw2435@nyu.edu)

October 13, 2019

Problem 1

Prove that in all these 7 cases we obtain valid kernels where k1 and k2 are valid (symmetric, positive definite) kernels on X.

By definition, $k(u, v) = u^T v$, where $u, v \in \mathbb{R}^D$

Case 1

$$k(u, v) = \alpha k_1(u, v) + \beta k_2(u, v) \quad for\ \alpha, \beta > 0 \tag{1-1}$$

Proof.

$$\begin{aligned} k(u, v) &= \alpha k_1(u, v) + \beta k_2(u, v) \\ &= \langle \sqrt{\alpha}\Phi_1(u), \sqrt{\alpha}\Phi_1(v) \rangle + \langle \sqrt{\beta}\Phi_2(u), \sqrt{\beta}\Phi_2(v) \rangle \\ &= \langle [\sqrt{\alpha}\Phi_1(u)\ \sqrt{\beta}\Phi_2(u)], [\sqrt{\alpha}\Phi_1(v)\ \sqrt{\beta}\Phi_2(v)] \rangle \end{aligned} \tag{1-2}$$

Hence, the new $k(u, v)$ is a valid kernel function on $[\sqrt{\alpha}\Phi_1(X)\ \sqrt{\beta}\Phi_2(X)]$

Case 2

$$k(u, v) = k_1(u, v)k_2(u, v) \tag{2-1}$$

Proof.

$$\begin{aligned} k(u, v) &= k_1(u, v)k_2(u, v) \\ &= \langle \Phi_1(u), \Phi_1(v) \rangle \langle \Phi_2(u), \Phi_2(v) \rangle \\ &= \langle \Phi_1(u)\Phi_2(u), \Phi_1(v)\Phi_2(v) \rangle \end{aligned} \tag{2-2}$$

Since $\langle \Phi_1(u), \Phi_1(v) \rangle, \langle \Phi_2(u), \Phi_2(v) \rangle$ are symmetric, positive definite matrix. Hence, $\langle \Phi_1(u)\Phi_2(u), \Phi_1(v)\Phi_2(v) \rangle$ is a valid kernel function on $\Phi_1(X)\Phi_2(X)$.

Case 3

$$k(u, v) = k_1(f(u), f(v)) \tag{3-1}$$

Proof.

Since $f(X)$ is a transformation function in the same domain.

Denote $\Phi(f(X)) \equiv \Phi_f(X)$.

$$\begin{aligned} k(u, v) &= k_1(f(u), f(v)) \\ &= \langle \Phi_1(f(u)), \Phi_1(f(v)) \rangle \\ &= \langle \Phi_f(u), \Phi_f(v) \rangle \end{aligned} \tag{3-2}$$

Hence, $k(u, v) = \langle \Phi_f(u), \Phi_f(v) \rangle$ is a valid kernel function on $\Phi_f(X)$.

Case 4

$$k(u, v) = g(u)g(v) \tag{4-1}$$

Proof.

Construct $V = [g(x_1)\ g(x_2)...g(x_n)]$, $k(u, v) = V'V$, which is a symmetric matrix and semi-definite with rank 1 for it have non-zero eignevalues.

Hence, $k(u, v)$ is a valid kernel function on $g(X)$.

Case 5

$$k(u, v) = f(k_1(u, v)) \tag{5-1}$$

Proof. Assume $f(u, v) = \sum c_{ij}\Phi(u)^i\Phi(v)^j$, while $c_{ij}\Phi(u)^i\Phi(v)^j$ is a valid kernel function according to proof of case 2 and the sum of $\sum c_{ij}\Phi(u)^i\Phi(v)^j$ is a valid kernel function according to proof of case 1. Hence, $f(u, v)$ is a valid kernel function.

Case 6

$$k(u, v) = exp(k_1(u, v)) \tag{6-1}$$

Proof.

Since,

$$exp(x) = \lim_{i \to \infty}(1 + x + ... + \frac{x^i}{i!}) \tag{6-2}$$

Which according to case 5, $k(u, v)$ is a valid kernel function.

Case 7

$$k(u, v) = exp(\frac{-||u - v||^2}{\sigma^2}) \tag{7-1}$$

Proof.

$$\begin{aligned} k(u, v) &= exp(\frac{-||u - v||^2}{\sigma^2}) \\ &= exp(\frac{2u'v - ||u||^2 - ||v||^2}{\sigma^2}) \\ &= exp(\frac{2u'v}{\sigma^2})exp(\frac{-||u||^2}{\sigma^2})exp(\frac{-||v||^2}{\sigma^2}) \end{aligned} \tag{1}$$

According to case 6, $exp(\frac{2u'v}{\sigma^2})$ is a valid kernel which can be rewritten into $exp(k'(u, v))$.

$exp(\frac{-||X||^2}{\sigma^2})$ is a function on X which can be rewritten into $g'(X)$, according to case 4, $exp(\frac{-||u||^2}{\sigma^2})exp(\frac{-||v||^2}{\sigma^2}) = g'(u)g'(v)$ is a valid kernel function.

And finally, according to case 2, $k(u, v) = exp(\frac{2u'v}{\sigma^2})exp(\frac{-||u||^2}{\sigma^2})exp(\frac{-||v||^2}{\sigma^2}) = exp(k'(u, v))g'(u)g'(v)$ is a valid kernel function.

Problem 2

In this problem, I randomly split the training set and the testing set at the ratio of 1:1. I perform SVC to training set with a different settings of hyperparameters each time, and test the accuracy of the classifier on test set. Meanwhile, I record the performance of the SVC with each experiment to help me do cross-validation to find the best settings of hyperparameters.

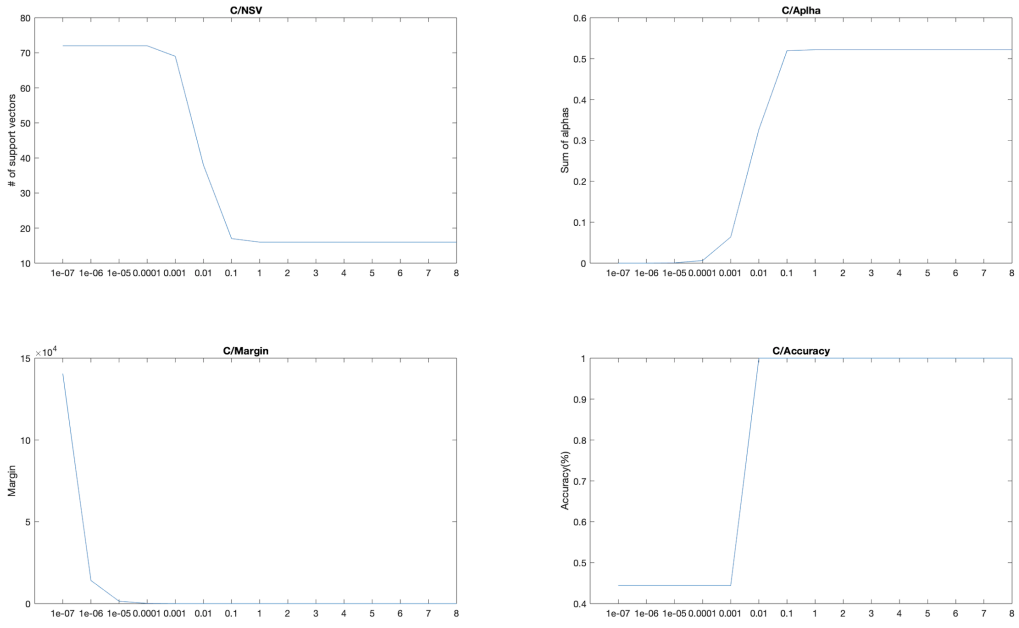
Case 1: Linear Kernel

When working with a linear kernel, the only hyperparameter is the regularization parameter C, which restrains the value of $0 < \alpha_i < C$. In order to dicuss how this parameter affects the performance of SVM, I vary C from **1e-7 to 1** with a growing ratio of **10** and **1 to 8** with an interval of **1**.

A SVC utility is called each with C value in **[1e-7 1e-6 1e-5 1e-4 1e-3 1e-2 1e-1 1 2 3 4 5 6 7 8]**.

Evaluation of the performance of a SVM is based on 4 aspects, which are **Number of Support Vectors(NSV), Sum of Alpha values, Functional Margin, Classification Accuracy.**

The performances of SVM with a linear kernel are plotted as follows:



First, take a look at the 'C/Accuracy' figure, accuracy reached and kept at 100% when $C \geq 0.01$.

When $C \geq 0.01$, the four figures indicate no significant changes in the Y-axis as C increases. Especially, when we look at the 'C/Alpha' figure, with the constraint of alpha which is the value of C increase, the sum of alpha values don't change as much, meaning the SVM problem optimize with this setting of α.

It would let us draw a conclusion that to choose the value of C **anywhere that** $C \geq 0.01$ **would be considered fair.**

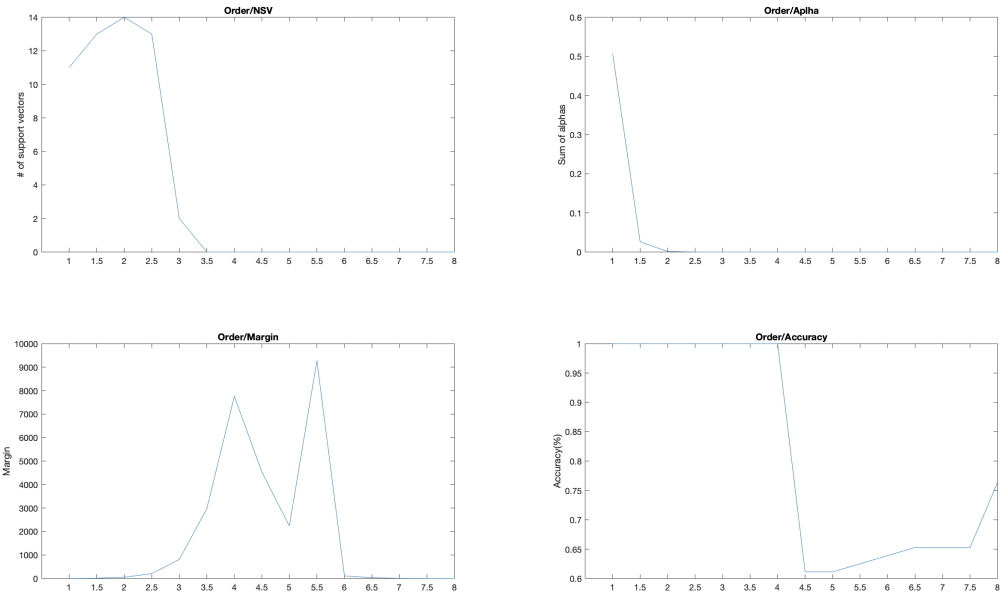
Case 2: Polynomial Kernel

With a polynomial kernel, there are two hyperparameter, the value of C same as in case 1 and the order of the polynomial kernel k.

C is picked in **[1e-7 1e-6 1e-5 1e-4 1e-3 1e-2 1e-1 1 2 3 4 5 6 7 8]**. While the value of k is picked between **[1 2 3 4 5 6 7 8]**, saving $k = 1$ just for comparsion.

Firstly, I fix the one hyperparameter by setting the value of C to 1, which is normally the default value in many SVM libarays, and I vary the value of the polynomial order.

The performances of SVM with a polynomial kernel with fixed C value is plotted as follows:



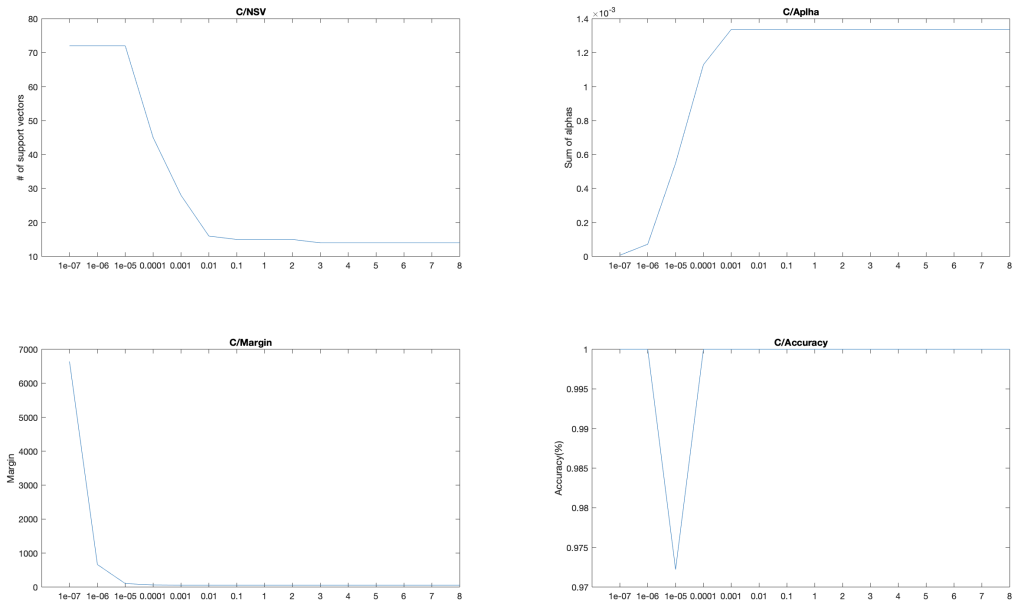
From the 'C/Accuracy' figure, we can see that when the polynomial order k is between $(1, 4)$ the accuracy is 100%, and after that the accuracy dropped. Although when as k increases to 8, the accuracy slightly increases, I don't resider k at this point is a valid choice for two reasons. One is that higher order of polynomial would cause way too many unnecessay computation comparing to that lower order already got the best performance. Another one is that the number of support vectors after order=4 is **0**, meaning the kernel functions don't fit the training data well.

Another interesting point is that when order dropped from linear to 2,3 and 4, with almost the same amount of support vectors, the sum of alpha values dropped drastically. It means with even more strict constraint C on alphas, it wouldn't harm its performance. From another point of view, with higher value of C (looser constraint), it wouldn't affect the selection of support vectors.

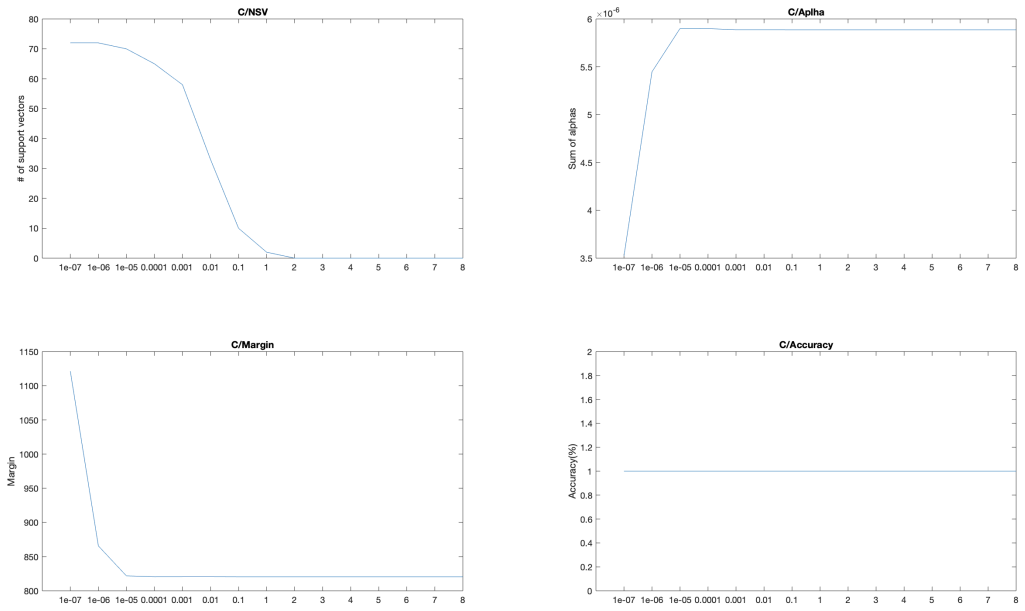
In conclusion, to choose of the polynomial order k between 1 to 4 is optimal.

In the next step, I fix the value of the polynomial order k. I choose two setting of k, $k = 2$ and $k = 3$.

When k=2, the performances of the SVM classifier is plotted below:



When k=3, the performances of the SVM classifier is plotted below:



From these two sets of figures, we can see that the difference between $k = 2$ and $k = 3$ is minor. The accuracy bascially stays at 100%, except for a little trembling when k=2.

The 'C/Accuracy' figure also provides solid evidence to support my theory with the observation with fixed C value that it wouldn't harm the classifier's performance with more strict C value.

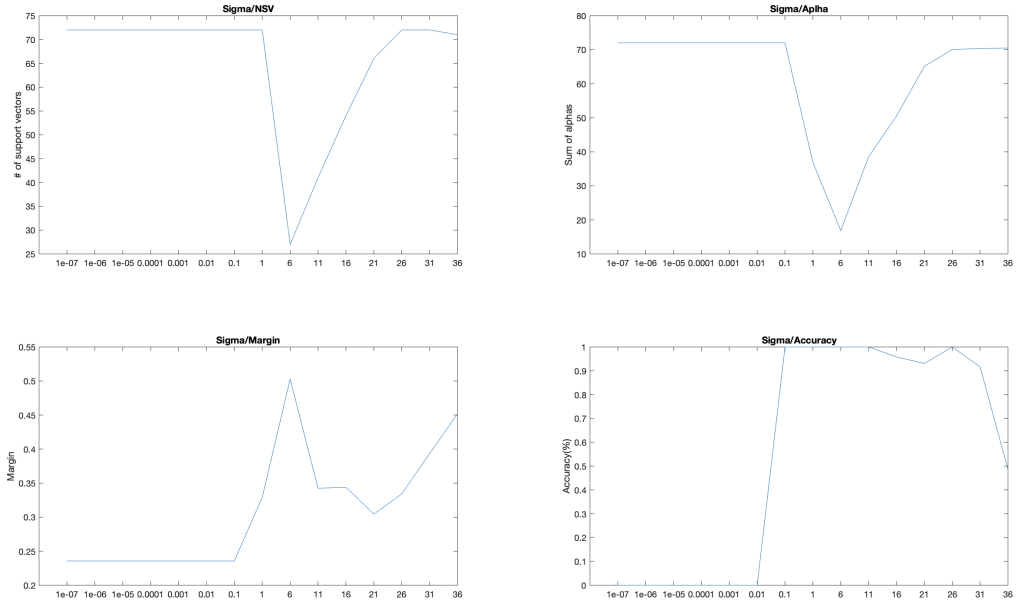
From the 'C/Alpha' figure, we can see that when $C \geq 0.01$, the sum of alpha values become stable, it wouldn't change with constraint, meaning it reaches its convergence. **Hence, the choice of C greater or equal to 0.01 would be optimal.**

Case 3: Kernel Choice-rbf

Simialiar to case 2, there are two hyperparameters C and σ.

The value of σ is picked in **[1e-7 1e-6 1e-5 1e-4 0.001 0.01 0.1 1 6 11 16 21 26 31 36]**.

Firstly, fix one hyperparameter by setting C to 1, we get the performance plots below:



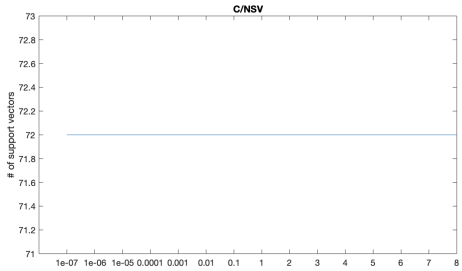
According to 'Sigma/Accuracy' figure, when $0.1 \leq \sigma \geq 26$, it gives out optimal accuracy with this setting of C value.

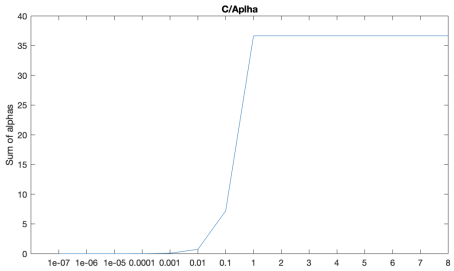
According to 'Sigma/NSV' figure, when $\sigma \leq 1$ and $\sigma \geq 26$, all the training samples are listed as support vectors, which is not good for the classifier.

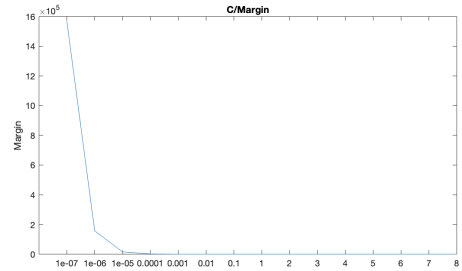
From the above two observation, **to choose σ from anywhere between (1,25) should be fine.**

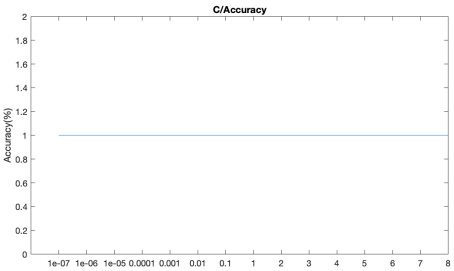
Next, fix σ to 1, 5, 25, vary the value of C in **[1e-7 1e-6 1e-5 1e-4 1e-3 1e-2 1e-1 1 2 3 4 5 6 7 8]**.

When $\sigma = 1$, the performances plots is as follows:

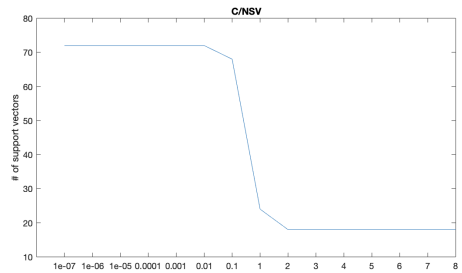


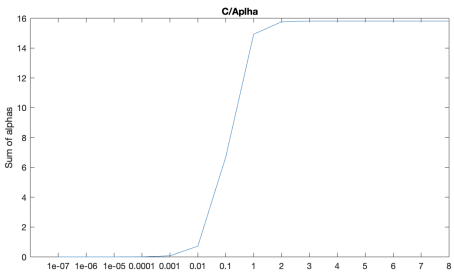


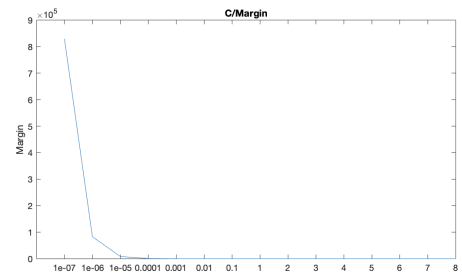


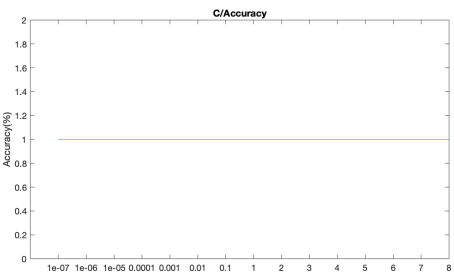


When $\sigma = 5$, the performances plots is as follows:

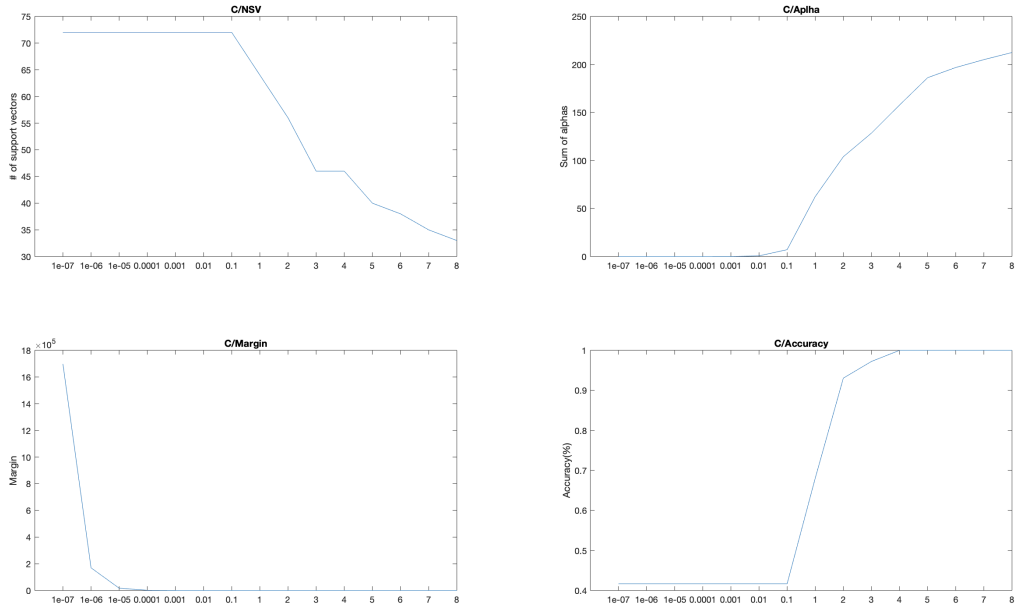








When $\sigma = 25$, the performances plots is as follows:



With each σ value, the accuracy performance are all excellent.

But from the 'C/Alpha' figure, we can see that when $C \geq 0.01$, the sum of alphas become stable. Also, we can notice a reasonable drop in number of support vectors which can be a good sign.

In conclusion, to choose C that $C \geq 0.01$ is optimal.

Problem 3

In this problem, we apply to PCA on image data to reconstruct the image. In detail, we find top-3 most significant components of the image data, and encode them with the mean of the data to reconstruct.

Step 0: Decenter the data

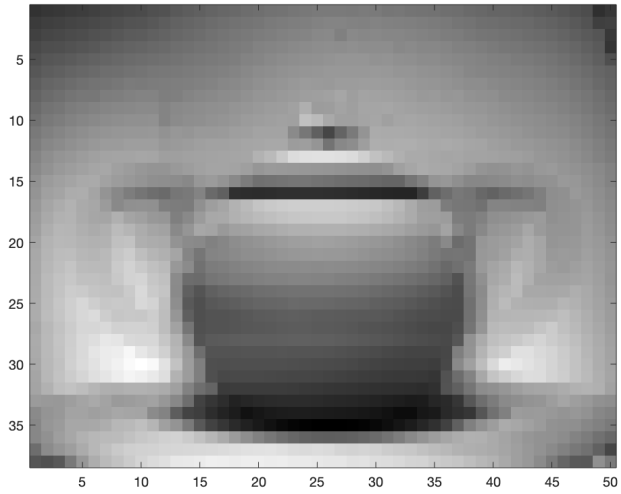
Compute the mean of the image data of shape (100, 1900). And then substract the mean from all data points to get the decentered data.

$$\mu = mean(\mathbf{X}) \tag{3-0}$$

And get $\mathbf{x}$ from

$$\mathbf{x} = \mathbf{X} - \mu \tag{3-1}$$

The mean of all image data is depicted below:



Step 1: Compute the covariance matrix

It is simply compute the covariance matrix of the data of the shape (100, 1900).

$$\mathbf{C} = cov(\mathbf{x}) = \mathbf{x}^T\mathbf{x} \tag{3-3}$$

Which has shape (1900,1900).

Step 2: Apply Eigenvalue Decomposition to the covariance matrix

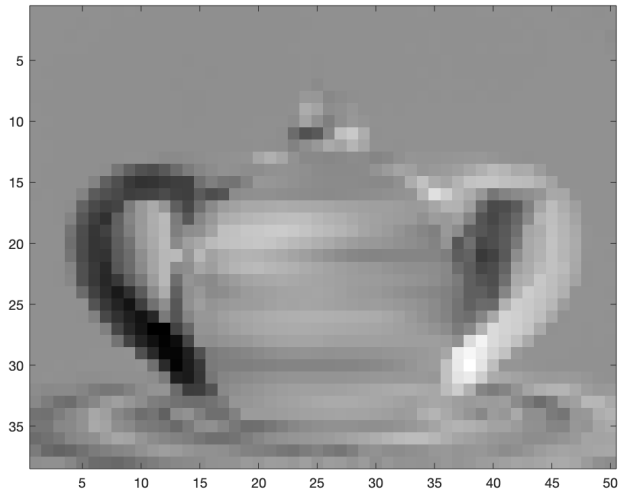
It would find the eigenvector matrix $\mathbf{V}$ and the matrix with eigenvalue on the diagonal $\mathbf{\Lambda}$

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{3-4}$$

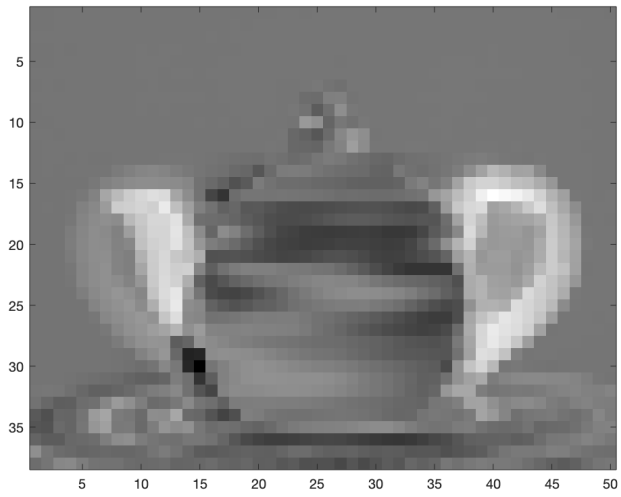
Step 3: Find the most significant value in eigenvalues

Because the covariance matrix is symmetric and positive-definite, the eigenvalue are all non-negative. In this problem, we find the top-3 most significant eigenvalue, which are [4.2150, 3.0168, 2.0993]. Each with a corresponding eigenvector.

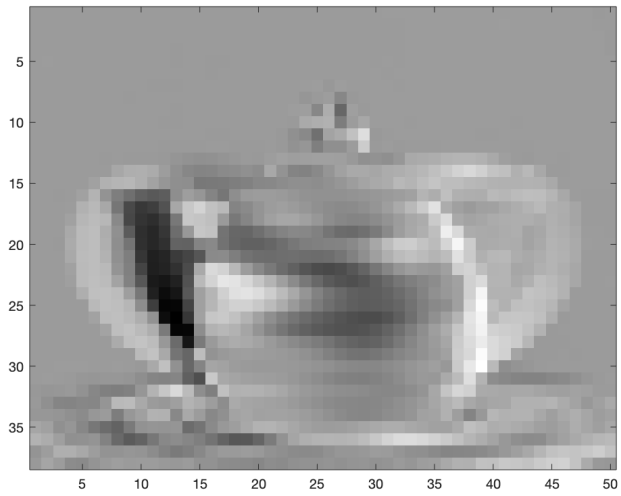
Each eigenvector can be depicted as a image. They are shown below:



eigenvalue=4.2150



eigenvalue=3.0168



eigenvalue=2.0993

Step 4: Encode by computing the coefficient matrix

The coefficient matrix can be obtained by $c_{ij} = (X_i - \mu)^T \mathbf{V}'_j = x_i^T \mathbf{V}'_j$ which is simply the inner product of **x** and the top-3 eigenvectors.

$$c = \mathbf{x}\mathbf{V}' \tag{3-5}$$

Step 5: Decode and construct the new image $\hat{x}$

We compute the $\hat{X}_i$ by

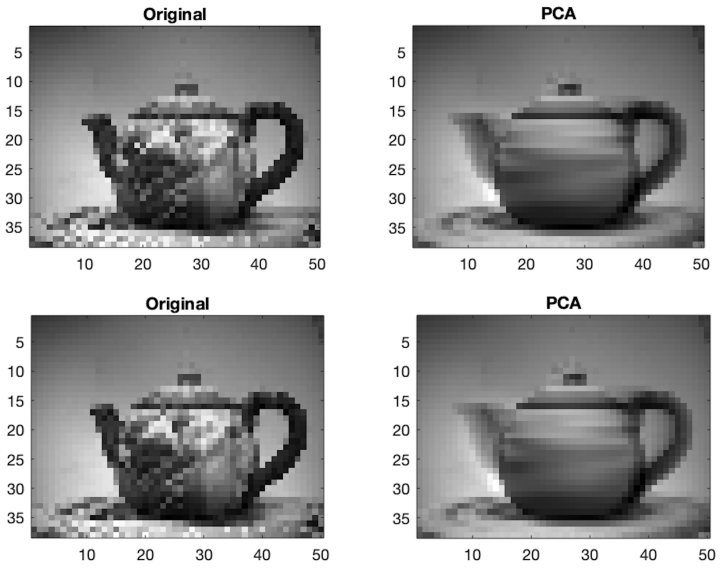
$$\hat{X}_i = \mu + \sum_{j=1}^{C} c_{ij} V'_j \tag{3-6}$$

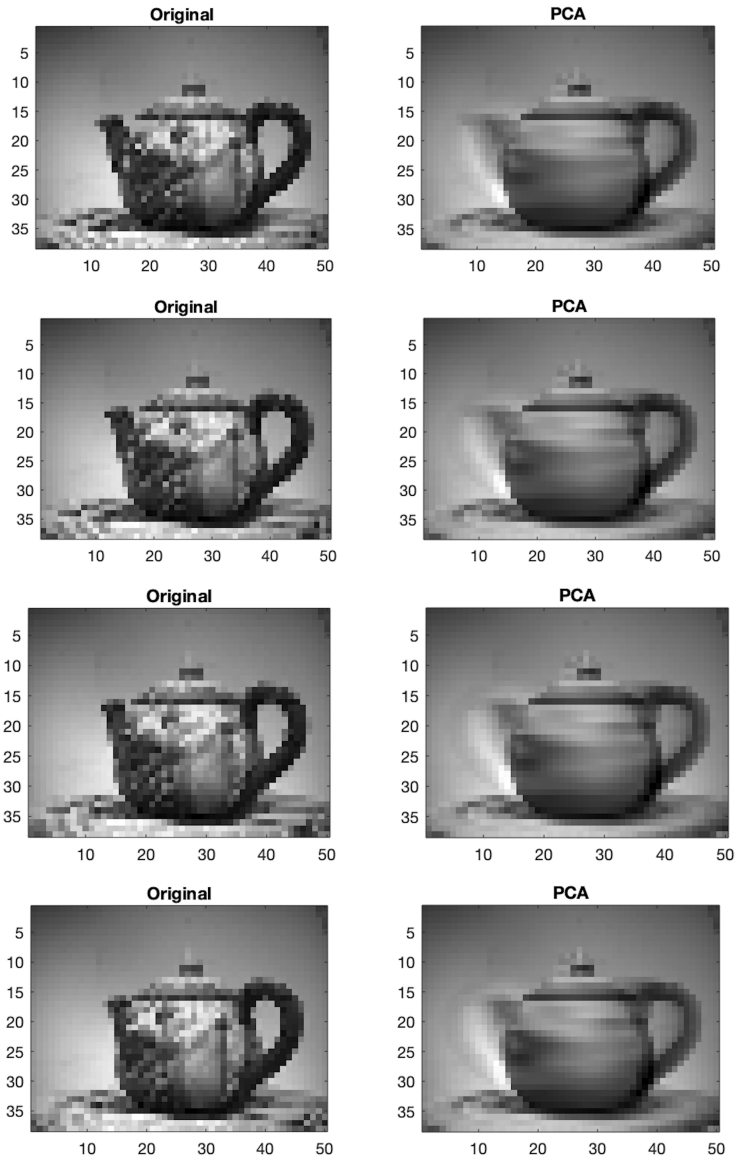
Which is simply just

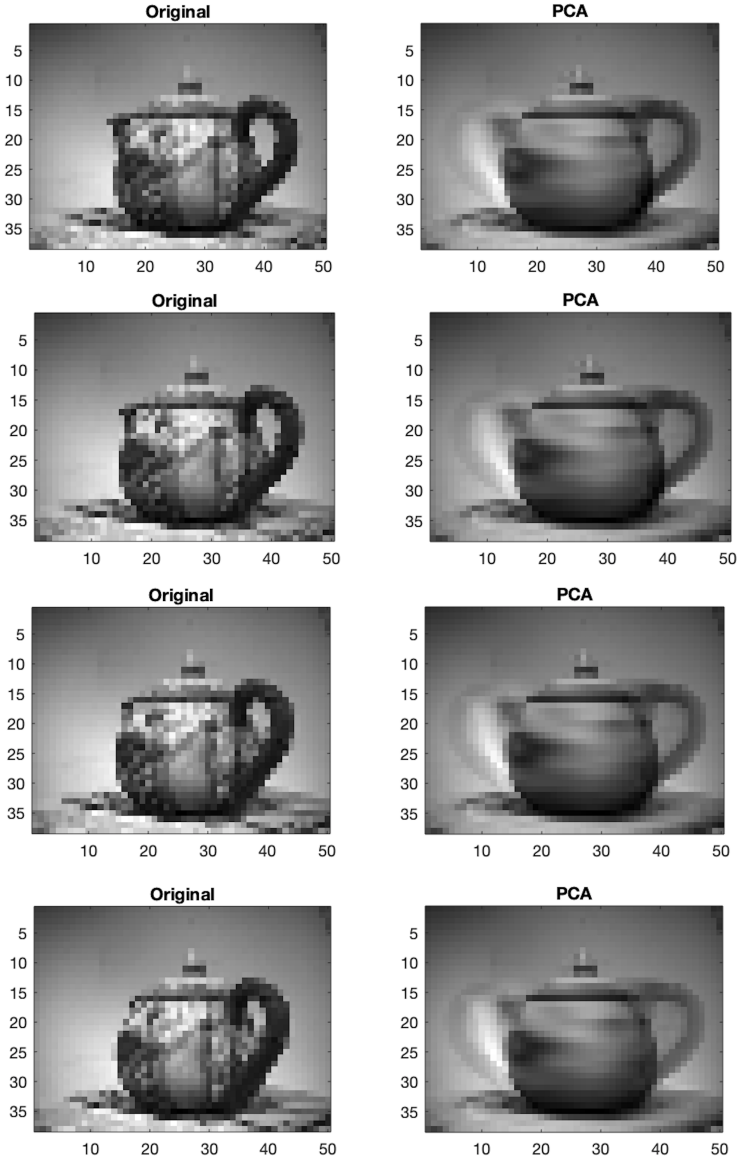
$$\hat{X} = \mu + c\mathbf{V}' \tag{2}$$

Performance

I randomly selected 10 representative images to reconstruct using PCA. Here are the results I got.




Original
PCA
Original
PCA
Original
PCA
Original
PCA
5
10
15
20
25
30
35
10
20
30
40
50

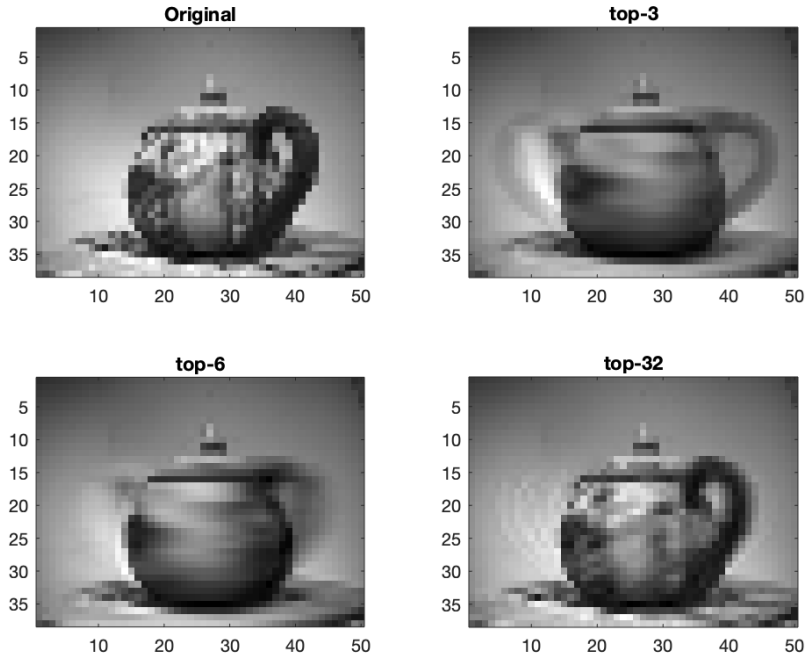


We can see that some of them is well reconstructed and distinguishable while some others may considered reconstructed poorly. I would say that only with top-3 components are pretty aggressive. By the result of eigenvalue decomposition, there are 32 value are greater than 0.1, with 6 values greater than 1. It might get a more comprehensive reconstruction by decoding with more components.

We evaluate the performance of PCA encoder by computing the L2-norm of $\mathbf{X}$ and $\hat{\mathbf{X}}$, the result is **13.6262**. By chosing the top-6 components, the norm reduce

to **9.8303**. With top-32 components, the norm further reduce to **3.1382**.

We get a sample from the image data, encode and decode respectively with top-3, top6 and top-32 most significant components. Here are the result we get.



It is clear that more components we encode into the reconstruction the better result and the more details we get. I wouldn't say to choose the top-3 most significant components is a bad idea. It all depends on the granularity and how to define main feature of the data we want to perserve.