Task 1

Write a MATLAB (M-file) program to compute the discriminant function, if one exists, for the following SVMs, using the training set provided.

(1) Hard margin SVM with the linear kernel:

$$K\ (x_1, x_2) = \ x_1^T x_2$$

(2) Hard margin SVM with a polynomial kernel:

$$K\ (x_1, x_2) = (x_1^T x_2 + 1)^p$$

(3) Soft margin with a polynomial kernel as given in Equation (2).

The step of computing the discriminate function can be summarized as follows:

a. Data preparation and pre-preprocess.

Here our training and testing dataset have 2,000 and 1,536 samples respectively. Each sample has 57 features. To ensure the data in internally consistent and enhance our training confidence, I firstly use standardization method to pre-process the data.

b. Gram matrix

With the given kernel, we need to calculate the gram matrix.

c. Mercer condition

Compute the eigenvalues of (b) gram matrix. We need to make sure the minimal eigenvalue is still non-negative. If not, the kernel candidate is not admissible. In Matlab, the computed eigenvalues are pretty much small. Here I set -1e-6 as the threshold. Only when the min eigenvalue is greater than the threshold, the training can move on.

d. Quadratic programing

Since the Mercer condition is already met, we have a global optimal value for this optimization problem. Following the Quadratic programming documentation in Matlab, we can calculate the optimal values.

e. Support vector determination

Set the threshold as 1e-6 to determine supper vectors in Matlab.

f. Weights calculation

For linear case:

$$\boldsymbol{w_0 = \sum_{i=1}^{N} \alpha_{0,i} d_i x_i;\ b_{0,i} = \frac{1}{d_i} - \ w_0^T x_i\ ;\ b_0 = \frac{\sum_{i=1}^{m} b_{0,i}}{m}}$$

For non-linear case:

$$\boldsymbol{g(x) = \sum_{i=1}^{N} \alpha_{0,i} d_i K(x, x_i) + b_0; b_0 = \frac{\sum_{i=1}^{m} b_{0,i}}{m}}$$

Task 2

Write a MATLAB (M-file) program to implement the SVMs with the discriminant functions obtained in Task 1.

Type of SVM	Training accuracy	Test accuracy
Hard margin with Linear kernel	0.938	0.92513

Hard margin with Polynomial kernel	p = 2	p = 3	p = 4	p = 5	p = 2	p = 3	p = 4	p = 5
	0.995	0.998	Non-convex	Non-convex	0.9014	0.89518	Non-convex	Non-convex

Soft margin with Polynomial kernel	C = 0.1	C = 0.6	C = 1.1	C = 2.1	C = 0.1	C = 0.6	C = 1.1	C = 2.1
p = 1	0.932	0.9405	0.938	0.937	0.92057	0.92708	0.92383	0.92383
p = 2	0.9805	0.994	0.995	0.9955	0.90951	0.89909	0.89909	0.89648
p = 3	0.996	0.998	0.998	0.998	0.90951	0.90104	0.89323	0.89063
p = 4	Non-convex				Non-convex			
p = 5								

Discussion

a. In terms of the hard margin with polynomial kernel, when p =4,5, the results are non-convex. The Mercer condition cannot be met in this situation.

b. In terms of the soft margin with polynomial kernel, when p = 4,5, the results are non-convex. The Mercer condition is not met no matter what c value is.

c. In terms of soft margin with polynomial kernel when p =2, we can find that with a larger tolerance of mis-classified samples (c value increases), the testing accuracy drops though the training accuracy is still consistent.

d. By comparing the hard margin with different kernel, although the training accuracy of linear kernel is lower than that of polynomial kernel, the testing accuracy of linear is higher.

e. Even though the training accuracy can reach almost 100% like p =2&3 with soft margin, the testing accuracy is still around 90%. It might be caused by overfitting.

Task 3

Design a SVM of your own.

In order to determine a better SVM classifier, we can start from two aspects based on previous experiment.

a. Kernel selection.

Here I tested two popular kernel besides linear and polynomial. They are Gaussian kernel and sigmoid kernel. After experiment, Gaussian kernel as a general-purpose kernel is used in this case. It works well when there is no prior knowledge about the data.

b. Soft margin parameters and Gaussian kernel parameters.

Type of SVM	Training accuracy				Test accuracy			
Soft margin with Gaussian kernel	C = 0.1	C = 1	C = 10	C = 20	C = 0.1	C = 1	C = 10	C = 20
sigma = 1	0.7645	0.992	0.9975	0.9975	0.64909	0.75911	0.76237	0.75911
sigma = 10	0.9105	0.935	0.9555	0.9625	0.9056	0.92383	0.9362	0.93685
sigma = 20	0.876	0.921	0.938	0.942	0.87956	0.91146	0.93034	0.93164
sigma = 50	0.8565	0.8915	0.922	0.927	0.86393	0.88932	0.91602	0.91667

Here we found when sigma = 10 and c = 20, both training and testing accuracy achieve highest values in our experiment. Thus, we chose these two as our own SVM classifier.