Problem 1.

Q(1)

Suppose K is a gram matrix which $K_{ij} = K(x_i, x_j) \in R^{n \times n}$. For any vector $\alpha \in R^n$:

$\alpha^T K \alpha$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (K(x_i, x_j))$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (\sum_{k=1}^{m} w_k K_k(x_i, x_j))$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} w_k \alpha_i \alpha_j K_k(x_i, x_j)$

$= \sum_{k=1}^{m} w_k \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K_k(x_i, x_j)$

$\because \forall w \geq 0$ and $\forall K_k$ is valid kernel function, $k = 1, 2 \cdots m$

$= \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \Phi_k(x_i)^T \Phi_k(x_j)$

$\because \alpha_i$ and α_j are real numbers

$= \sum_{k=1}^{m} \sum_{i=1}^{n} \alpha_i \Phi_k(x_i)^T \sum_{j=1}^{n} \alpha_j \Phi_k(x_j)$

$= \sum_{k=1}^{m} \{\sum_{i=1}^{n} \alpha_i \Phi_k(x_i)\}^T \sum_{j=1}^{n} \alpha_j \Phi_k(x_j)$

$\because$ inner product always ≥ 0

$\therefore \alpha^T K \alpha \geq 0$ and K is positive semi-definite function.

For symmetry:

K^T

$= (\sum_{j=1}^{m} w_j K_j(x, x'))^T$

$= \sum_{j=1}^{m} w_j (K_j(x, x'))^T$

$= \sum_{j=1}^{m} w_j (K_j(x, x'))$

$= \mathrm{K}$

$\mathrm{K} = \sum_{j=1}^{m} w_j K_j(x, x')$ is a valid kernel function because it is symmetric and positive semi-definite function.

Q(2)

Suppose K is a gram matrix which $K_{ij} = K(x_i, x_j) \in R^{n \times n}$. For any vector $\alpha \in R^n$:

$\alpha^T K \alpha$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (K(x_i, x_j))$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (\sum_{k=1}^{2} K_k(x_i, x_j))$

$= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{2} \alpha_i \alpha_j K_k(x_i, x_j)$

$= \sum_{k=1}^{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K_k(x_i, x_j)$

$\because \forall K_k$ is valid kernel function, $k = 1, 2$

$= \sum_{k=1}^{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \Phi_k(x_i)^T \Phi_k(x_j)$

$\because \alpha_i$ and α_j are real numbers

$= \sum_{k=1}^{2} \sum_{i=1}^{n} \alpha_i \Phi_k(x_i)^T \sum_{j=1}^{n} \alpha_j \Phi_k(x_j)$

$= \sum_{k=1}^{2} \{\sum_{i=1}^{n} \alpha_i \Phi_k(x_i)\}^T \sum_{j=1}^{n} \alpha_j \Phi_k(x_j)$

$\because$ inner product always ≥ 0

$\therefore \alpha^T K \alpha \geq 0$ and K is positive semi-definite function.

For symmetry:

K^T

$= (\sum_{j=1}^{2} w_j K_j(x, x'))^T$

$= \sum_{j=1}^{2} w_j (K_j(x, x'))^T$

$= \sum_{j=1}^{2} w_j (K_j(x, x'))$

$= \mathrm{K}$

$\mathrm{K} = \sum_{j=1}^{2} K_j(x, x')$, which equals to K(x,x') = $K_1(x, x') + K_2(x, x')$ is a valid kernel function because it is symmetric and positive semi-definite function.

Q(2)

Problem 2

Algorithm:

In SVM, we introduce the concept of maximizing the interval. The interval refers to the minimum value of the distance between the data and the straight line, so maximizing this value reflects our model tendency. The segmented hyperplane can be written as:

$$0 = w^T x + b$$

Then maximize the interval (constrained as the requirement of the classification task):

$$argmax_{w,b}[\min_i \frac{|w^T x_i + b|}{||w||}] \ s.t. \ y_i(w^T x_i + b) > 0$$
$$\Longrightarrow argmax_{w,b}[\min_i \frac{y_i(w^T x_i + b)}{||w||}] \ s.t. \ y_i(w^T x_i + b) > 0$$

For this constraint $y_i(w^T x_i + b) > 0$, we fix $\min y_i(w^T x_i + b) = 1 > 0$. This is because the coefficients that separate the two types of hyperplanes will not change the plane after scaling, which is equivalent to constraining the coefficients of the hyperplane.The simplified formula can be expressed as

$$argmin_{w,b} \frac{1}{2} w^T w \ s.t. \ \min_i y_i(w^T x_i + b) = 1$$
$$\Rightarrow argmin_{w,b} \frac{1}{2} w^T w \ s.t. \ y_i(w^T x_i + b) \geq 1, i = 1, 2, \cdots, N$$

Hard-margin SVM is only solvable for separable data. If it is not separable, our basic idea is to add the possibility of misclassification to the loss function. The number of misclassifications can be written as:

$$error = \sum_{i=1}^{N} I\{y_i(w^T x_i + b) < 1\}$$

This function is not continuous and can be rewritten as:

$$error = \sum_{i=1}^{N} \max\{0, 1 - y_i(w^T x_i + b)\}$$

The formula in the summation symbol is also called Hinge Function.

Add this error to Hard-margin SVM, so:

$$\underset{w,b}{argmin} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \max\{0, 1 - y_i(w^T x_i + b)\} \ s.t. \ y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \cdots, N$$

In this formula, the constant C can be regarded as the allowable error level. At the same time, in order to further eliminate the max symbol, for each observation in the data set, we can consider that most of them satisfy the constraints. But some of them violate Constraints, so this part of the constraints becomes $y_i(w^T x + b) \geq 1 - \xi_i$, where $\xi_i = 1 - y_i(w^T x_i + b)$, further simplified:

$$\underset{w,b}{argmin} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \ s.t. \ y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, 2, \cdots, N$$

If the sample size or dimension is very high, it is difficult to solve directly or even the problem itself is unsolvable. So this problem needs to be dealt with further by introducing the Lagrange function:

$$L(w, b, \lambda, \xi, \alpha) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \lambda_i (1 - \xi_i - y_i(w^T x_i + b)) - \sum_{i=1}^{N} \alpha_i \xi_i$$

Our original problem is equivalent to:

$$\underset{w,b,\xi}{argmin} \max_{\lambda,\alpha} L(w, b, \lambda, \xi, \alpha) \ s.t. \ \lambda \geq 0$$

We exchange the signs of the minimum and maximum to get the dual problem:

$$\max_{\lambda,\alpha} \min_{w,b,\xi} L(w, b, \lambda, \xi, \alpha) \ s.t. \ \lambda \geq 0$$

The corresponding set of KKT conditions are given by:

$$\lambda_i (1 - \xi_i - y_i(w^T x_i + b)) = 0 (slackness \ complementary) \tag{1}$$
$$\lambda_i \geq 0 \tag{2}$$
$$\alpha_i \geq 0 \tag{3}$$
$$\xi_i \geq 0 \tag{4}$$
$$1 - \xi_i - y_i(w^T x_i + b) \leq 0 \tag{5}$$

Since the inequality constraint is an affine function, the dual problem is equivalent to the original problem:

* b: $\frac{\partial}{\partial b} L = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$

* w: First substitute b into the Lagrange function and then take derivative to w:

$$\frac{\partial}{\partial w} L = 0 \Rightarrow w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

* ξ_i: $\frac{\partial}{\partial \xi_i} L = 0 \Rightarrow \lambda_i = C - \alpha_i$

Using these results to eliminate w, b, and $\{\xi_i\}$ from the Lagrangian, we obtain the dual Lagrangian in the form:

$$L(w, b, \lambda, \xi, \alpha) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N}\lambda_i$$

Therefore, the dual problem is:

$$\max_{\lambda,\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N}\lambda_i,\ s.t.\ \sum_{i=1}^{N}\lambda_i y_i = 0, 0 \leq \lambda_i \leq C$$

From this equation, we could use cvxopt to solve the λ and obtain $\hat{w}$ and $\hat{b}$ using the following formula:

$\hat{w} = \sum_{i=1}^{N}\lambda_i y_i x_i$

$\hat{b} = \frac{1}{N_M}\sum_{i=1}^{M}(y_i - \sum_{j=1}^{S}\lambda_j y_j x_i^T x_j)$, where M and S denote the set of indices of data points having $0 \leq \lambda_i \leq$ C.

So the parameter w of this hyperplane is a linear combination of data points, and the final parameter value is a linear combination of vectors that partially satisfy $1 - \xi_i - y_i(w^T x_i + b) = 0$. These vectors are also Support vector.

Result:

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
Average_train_error_rate	0.519917	0.494167	0.488361	0.48575	0.488639	0.486639	0.485944	0.48475
Std_train_error_rate	0.015808	0.01117	0.005855	0.004278	0.010493	0.004097	0.00433	0.005319
Average_validation_error_rate	0.5325	0.5145	0.51175	0.50975	0.519	0.5165	0.50975	0.50325
Std_validation_error_rate	0.029155	0.022243	0.02385	0.027144	0.023854	0.025229	0.025628	0.020978

Figure 1: Summary of Linear SVM trained by 50 percentages of data

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
Average_train_error_rate	0.512611	0.503625	0.495222	0.494111	0.494681	0.493583	0.494431	0.493167
Std_train_error_rate	0.022294	0.014089	0.001855	0.002849	0.002729	0.00304	0.001968	0.003405
Average_validation_error_rate	0.5305	0.52075	0.513125	0.51525	0.511625	0.52225	0.51375	0.515125
Std_validation_error_rate	0.019986	0.019718	0.0122	0.009997	0.009683	0.016025	0.012918	0.015046

Figure 2: Summary of Linear SVM trained by all data

For Linear SVM trained by 50 percentages of data,which means the training samples would be 4000, we obtained C = 1000 and the error rate for the best model = 0.51. For Linear SVM trained by all data,which means the training samples would be 8000, we obtained C = 1 and the error rate for the best model = 0.5005.

The larger the C value, the less likely the classifier would allow classification errors ("outliers"). If the C value is too large, the classifier will try its

best to make fewer mistakes on the training data, which is actually impossible/meaningless, thus causing overfitting.

If the C value is too small, the classifier would not care too much about the classification error, thus causing underfitting and the generalization classification performance will be poor.

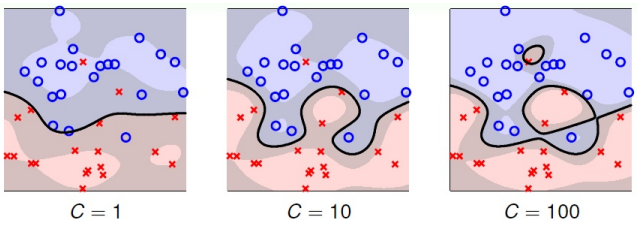

Figure 3: Machine Learning Techniques from Cousera

From the two trials, we can see the average train error rate is decreasing as the C value is increasing. Usually, when C keeps increasing, the average validation error rate would increase at some point. But from our two trials, the average validation error rate has a tendency of decreasing all the time. There are two reasons: First one is that the C is not large enough. Second one is that the linear SVM is totally useless in classifying this data set and the data is linear inseparable, which means not matter what the C is, the average validation error rate would always be around 0.5. For this problem, the second explanation would be more reasonable since all the average validation error rates and the average train error rates are around 0.5. Some fluctuations can be explained by the std.

Problem 3

Algorithm:

The kernel method can be applied to many problems. For strictly inseparable problems in classification problems, we introduce a feature conversion function to turn the original indivisible data set into a divisible data set, and then apply the existing model. After changing a data set in a low-dimensional space to a data set in a high-dimensional space, the data often becomes separable (the data becomes more sparse)

When applied in SVM, observe the above SVM dual problem:

$$\max_{\lambda,\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_{i=1}^{N}\lambda_i,\ s.t.\ \sum_{i=1}^{N}\lambda_i y_i = 0, 0 \le \lambda_i \le C$$

When solving, the inner product needs to be obtained, so the indivisible data needs to be transformed into the inner product after the feature transformation. It is often difficult to find the inner product of the transformation function. So we can use the transformation function of the inner product:

$$\forall x, x' \in \mathcal{X}, \exists \phi \in \mathcal{H} : x \to z\ s.t.\ k(x, x') = \phi(x)^T \phi(x)$$

Let $k(x, x')$ be a positive definite kernel function, where $\mathcal{H}$ is the Hilbert space (complete linear inner product space). If the condition of inner product is removed, we simply call it a kernel function.For example, $k(x, x') = \exp(-\frac{(x-x')^2}{2\sigma^2})$ is a kernel function.

The positive definite kernel function has the following equivalent definitions:

If the kernel function satisfies:

1. Symmetry

2. Positive Definiteness

Then this kernel function is a positive definite kernel function.

Proof:

1. Symmetry $\Leftrightarrow k(x, z) = k(z, x)$, obviously satisfies the definition of inner product.

2. Positive definite $\Leftrightarrow \forall N, x_1, x_2, \cdots, x_N \in \mathcal{X}$, the corresponding Gram Matrix $K = [k(x_i, x_j)]$ is positive semi-definite .

To prove: $k(x,z) = \phi(x)^T\phi(z) \Leftrightarrow K$ positive semi-definite + symmetry.

1. $\Rightarrow$: First of all, symmetry is obvious. For positive definiteness:

$$K = \begin{pmatrix} k(x_1,x_2) & \cdots & k(x_1,x_N) \\ \vdots & \vdots & \vdots \\ k(x_N,x_1) & \cdots & k(x_N,x_N) \end{pmatrix}$$

Take $\alpha \in \mathbb{R}^N$ arbitrarily, which means we need to prove $\alpha^T K\alpha \geq 0$:

$$\alpha^T K\alpha = \sum_{i,j} \alpha_i\alpha_j K_{ij} = \sum_{i,j} \alpha_i\phi^T(x_i)\phi(x_j)\alpha_j = \sum_i \alpha_i\phi^T(x_i)\sum_j \alpha_j\phi(x_j)$$

This formula is the form of inner product. Hilbert space satisfies linearity, so it is a proof of positive definiteness.

For Linear kernel, $k(x,x') = x^T x'$. For RBF kernel, $k(x,x') = \exp(-\frac{(x-x')^2}{2\sigma^2})$.

Result:

Linear:

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
Average_train_error_rate	0.496472	0.503028	0.486833	0.48625	0.487528	0.486333	0.486111	0.4865
Std_train_error_rate	0.022138	0.014035	0.003235	0.005322	0.004349	0.0042	0.002897	0.004831
Average_validation_error_rate	0.51675	0.52725	0.51	0.51775	0.51325	0.51575	0.51925	0.5055
Std_validation_error_rate	0.023158	0.025531	0.018303	0.027167	0.025667	0.016434	0.01743	0.029065

Figure 4: Summary of Linear Kernel SVM trained by 50 percentages of data

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
Average_train_error_rate	0.46375	0.487333	0.490569	0.488806	0.489736	0.487681	0.489778	0.489597
Std_train_error_rate	0.022294	0.006044	0.003145	0.003147	0.003549	0.004526	0.003481	0.004515
Average_validation_error_rate	0.477125	0.501	0.50025	0.5055	0.508625	0.5045	0.499875	0.500875
Std_validation_error_rate	0.024919	0.013084	0.009839	0.012503	0.01441	0.016117	0.017858	0.016724

Figure 5: Summary of Linear Kernel SVM trained by all data

RBF:

Average_train_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.49225	0.49225	0.442667	0	0	0	0	0
sigma = 0.1	0.49225	0.49225	0.4905	0	0	0	0	0
sigma = 1	0.4735	0.200306	0.114056	0.073667	0.035611	0.000389	0	0
sigma = 10	0.49225	0.49225	0.396278	0.158056	0.060083	0.062972	0.060667	0.065028
sigma = 100	0.49225	0.49225	0.49225	0.49225	0.502583	0.498333	0.454167	0.350667
sigma = 1000	0.49225	0.49225	0.49225	0.49225	0.49225	0.49225	0.492778	0.508917

Std_train_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.00293	0.002773	0.147565	0	0	0	0	0
sigma = 0.1	0.003039	0.0027	0.004658	0	0	0	0	0
sigma = 1	0.022865	0.03241	0.003336	0.001415	0.001604	0.000136	0	0
sigma = 10	0.001869	0.002544	0.084964	0.039031	0.002301	0.002045	0.002924	0.002157
sigma = 100	0.00258	0.003026	0.001603	0.002135	0.017876	0.011174	0.03528	0.101589
sigma = 1000	0.002319	0.003107	0.002032	0.003479	0.002553	0.003047	0.003909	0.019973

Average_validation_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.49225	0.49225	0.49225	0.49225	0.49225	0.49225	0.49225	0.49225
sigma = 0.1	0.49225	0.49225	0.49225	0.49225	0.49225	0.49175	0.492	0.49175
sigma = 1	0.47875	0.20925	0.123	0.095	0.08925	0.086	0.08625	0.08625
sigma = 10	0.49225	0.49225	0.404	0.17525	0.0605	0.06725	0.0665	0.0645
sigma = 100	0.49225	0.49225	0.49225	0.49225	0.505	0.5055	0.4735	0.367
sigma = 1000	0.49225	0.49225	0.49225	0.49225	0.49225	0.49225	0.5035	0.51825

Std_validation_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.026374	0.024961	0.01825	0.020629	0.015348	0.031314	0.023088	0.017694
sigma = 0.1	0.027351	0.024301	0.017445	0.019476	0.02212	0.016698	0.021442	0.020033
sigma = 1	0.039863	0.041564	0.015199	0.014274	0.010431	0.017219	0.018209	0.015742
sigma = 10	0.016824	0.022898	0.102727	0.042241	0.010416	0.016334	0.010677	0.010828
sigma = 100	0.023223	0.027236	0.014424	0.019218	0.021125	0.018534	0.058472	0.107307
sigma = 1000	0.02087	0.027961	0.018284	0.031314	0.02298	0.027419	0.024114	0.021996

Figure 6: Summary of RBF Kernel SVM trained by 50 percentages of data

For Linear Kernel SVM trained by 50 percentages of data,which means the training samples would be 4000, we obtained C = 1000 and the error rate for the best model = 0.503. For Linear Kernel SVM trained by all data,which means the training samples would be 8000, we obtained C = 0.0001 and the error rate for the best model = 0.4945.

Since Linear Kernel SVM is the same as the linear SVM, the explanation can be found in Problem 2.

For RBF Kernel SVM trained by 50 percentages of data,which means the training samples would be 4000, we obtained C = 1, $\sigma = 10$ and the error rate for the best model = 0.055.

The two hyperparameters we obtained, which are C and σ, are independent. If the sigma is small, the gaussian distribution would become tall and thin and the RBF kernel SVM will only act on the samples near the support vector samples, thus the classification for new sample would be very poor. Normally, we call it overfitting. If the sigma is too large, the underfitting will occur.

For this data set, we choose C = 1 and $\sigma = 10$, which are all in the middle range of the parameters that we tested. From the average train error rate and the average validation error rate, we can find that when σ is small, we can obtain the 0 train error rate but poor validation error rate. This proves the overfitting. When σ is large, we can obtain the high train error rate and high validation error rate. This proves the underfitting.

For average validation error rate, when C is very small, such as C = 0.0001 and C = 0.001, the change of σ would not improve the performance of the model since the hyperplane is too smooth.

For the C and sigma in middle range,if we fix either one of these two parameters, we can find a trend where the error rate first decreases and then increases.

This trend fits the convex optimization.

Problem 4

Algorithm:

In this problem, we use the one-against-all strategy to solve the multi-class classification using SVM. For every hyperparameter, we set the data of each of the ten class to be +1, and the rest class to be -1. Then we run SVM 10 times on each dataset to obtain w_j and b_j,$j = 0, 1, \cdots 9$. For the new data points, we predict the label as $argmax_j w_j^T x + b_j$,$j = 0, 1, \cdots 9$. The algorithm of most part would be similar as that of the problem 3.

Result:

Linear Kernel:

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
Average_train_error_rate	0.030972	0.009306	0.000556	0	0	0	0	0
Std_train_error_rate	0.002369	0.00125	0.000278	0	0	0	0	0
Average_validation_error_rate	0.036875	0.016875	0.013125	0.013125	0.015	0.01375	0.013125	0.015
Std_validation_error_rate	0.023626	0.009291	0.007099	0.010987	0.0075	0.008292	0.005896	0.008478

Figure 7: Summary of Linear Kernel SVM trained by all data

RBF Kernel:

Average_train_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.854583	0.874236	0.347639	0	0	0	0	0
sigma = 0.1	0.884444	0.875278	0.397569	0	0	0	0	0
sigma = 1	0.873264	0.895278	0.279653	0	0	0	0	0
sigma = 10	0.880347	0.622847	0.006042	0.000625	0	0	0	0
sigma = 100	0.875625	0.819236	0.208056	0.045556	0.03125	0.007639	0.000278	0
sigma = 1000	0.885069	0.884444	0.884514	0.797917	0.205486	0.048472	0.031736	0.008264

Std_train_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.06623	0.040626	0.180533	0	0	0	0	0
sigma = 0.1	0.030948	0.040424	0.227876	0	0	0	0	0
sigma = 1	0.040824	0.000867	0.204352	0	0	0	0	0
sigma = 10	0.033339	0.136736	0.005727	0.000208	0	0	0	0
sigma = 100	0.03973	0.070758	0.076939	0.004535	0.002282	0.000761	0.00034	0
sigma = 1000	0.030234	0.03048	0.029845	0.102951	0.111206	0.004466	0.001865	0.000725

Average_validation_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.92625	0.9275	0.926875	0.92875	0.925	0.92	0.92	0.923125
sigma = 0.1	0.926875	0.9175	0.929375	0.91875	0.916875	0.92	0.92375	0.924375
sigma = 1	0.9375	0.93125	0.921875	0.925	0.926875	0.924375	0.91375	0.9275
sigma = 10	0.919375	0.7025	0.06375	0.024375	0.014375	0.014375	0.015	0.01375
sigma = 100	0.910625	0.86375	0.248125	0.058125	0.03875	0.0175	0.01625	0.011875
sigma = 1000	0.91375	0.92125	0.924375	0.83875	0.24875	0.06	0.0375	0.02

Std_validation_error_rate								
	C=0.0001	C=0.001	C=0.01	C=0.1	C=1	C=10	C=100	C=1000
sigma = 0.001	0.01	0.016817	0.012824	0.016583	0.014524	0.015	0.01111	0.011541
sigma = 0.1	0.013707	0.011792	0.013989	0.018541	0.0122	0.014197	0.009186	0.016642
sigma = 1	0.01311	0.00927	0.016358	0.017455	0.011541	0.013244	0.016724	0.020387
sigma = 10	0.019455	0.137477	0.024335	0.012327	0.007421	0.010097	0.006374	0.00875
sigma = 100	0.037713	0.074624	0.103835	0.017689	0.019922	0.00875	0.009763	0.008592
sigma = 1000	0.032452	0.027128	0.030803	0.090173	0.136496	0.01375	0.015052	0.010753

Figure 8: Summary of RBF Kernel SVM trained by all data

For Linear Kernel SVM trained by all data,which means the training samples would be 1600, we obtained C = 0.01 and the error rate for the best model = 0.025.

For RBF Kernel SVM trained by all data,which means the training samples would be 1600, we obtained C = 1000, $\sigma = 100$ and the error rate for the best model = 0.025.

For this problem, the Linear Kernel SVM performs pretty well. We can obtain the best average validation error rate by setting the C to 0.01. The Linear Kernel SVM not only performs well when C equals to 0.01, but also performs well for all the C values. This good performance means that the data is linear separable and when the features of the data point are sparse, Linear Kernel would be a good choice.

For RBF Kernel SVM, most combinations of the C and the σ perform poorly, which means it is hard to find a boundary in the high dimension projection of this data set. From the comparison of the average train error rate and the average validation error rate, when C is large and σ is small, the overfitting problem occurs and when C is small and σ is large, the underfitting problem occurs. These observations prove the previous statements.

When C equals to 1000 and σ equals to 100, we could avoid the overfitting and the underfitting problem and obtain the best result.

Reference:

1. https://www.coursera.org/browse?source=deprecated spark cdp

2.Pattern Recognition and Machine Learning,CHRISTOPHER M.BISHOP

3.Lecture in Canvas,Nicholas Johnson