Part 1:

Q1:

```python
def cross_validation(x_train, y_train, k=5):
    arr = np.arange(len(x_train))

    # random sequence
    np.random.shuffle(arr)

    #split into k-size arrays
    split = np.array_split(arr, k)

    lists = []
    for i in range(len(split)):
        current_list = []
        first_list = np.delete(arr, split[i])
        second_list = split[i]
        current_list.append(first_list)
        current_list.append(second_list)
        lists.append(current_list)
    return lists
```

Q2:

12 秒

```python
## your code
gammas = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]
Cs = [0.01, 0.1, 1, 10, 100, 1000, 10000]
result = []
datas = cross_validation(x_train, y_train, k=5)
best_accuracy = 0
best_parameters = [0, 0]
best_model = None
for C in Cs:
    tmp = []
    for gamma in gammas:
        clf = SVC(C=C, kernel='rbf', gamma=gamma)
        accuracies = []
        for data in datas:
            clf.fit(x_train[data[0]], y_train[data[0]])
            accuracies.append(clf.score(x_train[data[1]], y_train[data[1]]))
        current_accuracy = np.mean(accuracies)
        if current_accuracy>best_accuracy:
            best_parameters = [gamma, C]
            best_accuracy = current_accuracy
            best_model = clf
        tmp.append(np.mean(accuracies))
    result.append(tmp)
```

0 秒 [121]

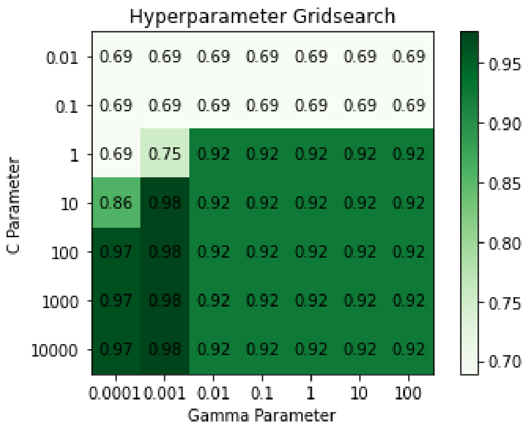
```python
print(best_parameters)
```

```
[0.001, 10]
```

Q3:

```
fig, ax = plt.subplots()
plt.title('Hyperparameter Gridsearch')
plt.xlabel('Gamma Parameter')
plt.ylabel('C Parameter')
data = result

ax.imshow(data, cmap='Greens')

# set the interval of the ticks
ax.set_xticks(np.arange(np.array(Cs).shape[0]), minor=False)
ax.set_yticks(np.arange(np.array(gammas).shape[0]), minor=False)

# show data value in the grid
for (i, j), z in np.ndenumerate(data):
        ax.text(j, i, '{:.2f}'.format(z), ha='center', va='center')

# show colorbar
fig.colorbar(ax.imshow(data, cmap='Greens', interpolation='none'))
ax.xaxis.tick_bottom()
ax.set_xticklabels(gammas)
ax.set_yticklabels(Cs)
```

```
[Text(0, 0, '0.01'),
 Text(0, 0, '0.1'),
 Text(0, 0, '1'),
 Text(0, 0, '10'),
 Text(0, 0, '100'),
 Text(0, 0, '1000'),
 Text(0, 0, '10000')]
```



Q4:

[123]

```
y_pred = best_model.predict(x_test)
print("Accuracy score: ", accuracy_score(y_pred, y_test))
```

```
Accuracy score:  0.9114583333333334
```

Part 2:

1.

a. Let $k_1(x,x') = t \Rightarrow k(x,x') = t^2 + (t+1)^2 = 2t^2 + 2t + 1$

which is a polynomial function of a valid kernel with positive coefficients $\Rightarrow$ valid. #

b. $(k_1(x,x'))^2 = k_1(x,x') \cdot k_1(x,x')$ is a kernel

$e^{|x|^2} \cdot e^{|x'|^2} = e^{x^Tx - 2x^Tx' + x'^Tx' + 2x^Tx'}$, $x^Tx - 2x^Tx' + x'^Tx'$ meets $f(x)k(x,x')f(x')$

$\Rightarrow \underbrace{(x^Tx - 2x^Tx' + x'^Tx')}_{\text{kernel}} + \underbrace{2x^Tx'}_{\text{kernel}}$ in the exp function is also a kernel $\Rightarrow$ valid #

2. Positive semidefinite means all the eigenvalues should be non-negative

to diagonalize the symmetric matrix $K = V \Lambda V^T$

$$\Rightarrow K_{ij} = (V \Lambda V^T)_{ij} = \sum_{k=1}^{n} \lambda^k \cdot V_{ki} \cdot V_{kj} = \phi(x_i)^T \cdot \phi(x_j)$$

Above is by mapping $x_i \rightarrow (\sqrt{\lambda_k} \cdot V_{ki})_{k=1}^{n}$

to make $\sqrt{\lambda_k}$ a real value, λ_k must be non-negative for all k

#

3. $a_n = -\frac{1}{\lambda}\{w^T\phi(x_n) - t_n\}$ $\quad$ (define $\sum_{i=1}^{m} w_i = W$)

$$= -\frac{1}{\lambda}\sum_{i=1}^{m}\left[w_i\phi(x_{ni}) - \frac{w_i}{W}\cdot t_n\right]$$

$$= \sum_{i=1}^{m} -\frac{w_i}{\lambda}\left(\phi(x_{ni}) - \frac{t_n}{W}\right)$$

#

4. $e^{\frac{-|x-x'|^2}{2\delta^2}} = e^{\frac{-(x^Tx - 2x^Tx' + x'^Tx')}{2\delta^2}}$

$$= e^{\left(\frac{-x^Tx}{2\delta^2} + \frac{x^Tx'}{\delta^2} + \frac{-x'^Tx'}{2\delta^2}\right)}$$

$\frac{x^Tx'}{\delta^2}$ is valid $\Rightarrow$ $\frac{-x^Tx}{2\delta^2} + \frac{x^Tx'}{\delta^2} + \frac{-x'^Tx'}{2\delta^2}$ is valid (meet $f(x)k(x,x')f(x')$)

$\Rightarrow e^{\left(\frac{-x^Tx}{2\delta^2} + \frac{x^Tx'}{\delta^2} + \frac{-x'^Tx'}{2\delta^2}\right)}$ is valid

$\Rightarrow = \phi(x)^T\phi(x')$

where $\phi(x) = e^{\frac{-x^Tx}{2\delta^2}} \cdot \begin{bmatrix} 1 \\ \sqrt{\frac{1}{1!\delta^2}}\,x \\ \sqrt{\frac{1}{2!\delta^4}}\,x^2 \\ \vdots \end{bmatrix}$ (by taylor expansion)

#