109550031_HW4

Part. 1, Coding

1. K-fold data partition

```
def cross_validation(sample_size, k=5):
    # calculate the size of each fold
    fold_size = np.ones(k , dtype=int) * (sample_size // k)
    fold_size[:sample_size % k] += 1 # if the sample size is not divisible by k
    # shuffle
    indexs = np.arange(sample_size)
    np.random.default_rng(seed=10).shuffle(indexs)
    # split indexs
    folds = np.array_split(indexs, k)
    folds = np.array(folds)
    # return training index and val index in each fold
    ret = []
    for i in range(k):
        train_fold_ind = np.delete(np.arange(k), i)
        train_fold = np.concatenate((folds[train_fold_ind]), axis=None)
        val_fold = folds[i]
        ret.append([train_fold, val_fold])
    return ret
```

cross_validation function

```
kfold_data = cross_validation(sample_size = x_train.shape[0], k=10)
# should contain 10 fold of data
assert len(kfold_data) == 10
# each element should contain train fold and validation fold
assert len(kfold_data[0]) == 2
# The number of data in each validation fold should equal to training data divieded by K
assert kfold_data[0][1].shape[0] == 700
✓ 0.1s
```

check if the size is correct

```
shape = 20
kfold_data_example = cross_validation(shape, k=5)
for i, (train_index, val_index) in enumerate(kfold_data_example):
    print("Split: %s, Training index: %s, Validation index: %s" % (i+1, train_index, val_index))
✓ 0.5s
Split: 1, Training index: [11  7  6  1  4 15 10 19  3 18  2 12 17  8  0 13], Validation index: [ 9  5 16 14]
Split: 2, Training index: [ 9  5 16 14  4 15 10 19  3 18  2 12 17  8  0 13], Validation index: [11  7  6  1]
Split: 3, Training index: [ 9  5 16 14 11  7  6  1  3 18  2 12 17  8  0 13], Validation index: [ 4 15 10 19]
Split: 4, Training index: [ 9  5 16 14 11  7  6  1  4 15 10 19 17  8  0 13], Validation index: [ 3 18  2 12]
Split: 5, Training index: [ 9  5 16 14 11  7  6  1  4 15 10 19  3 18  2 12], Validation index: [17  8  0 13]
```

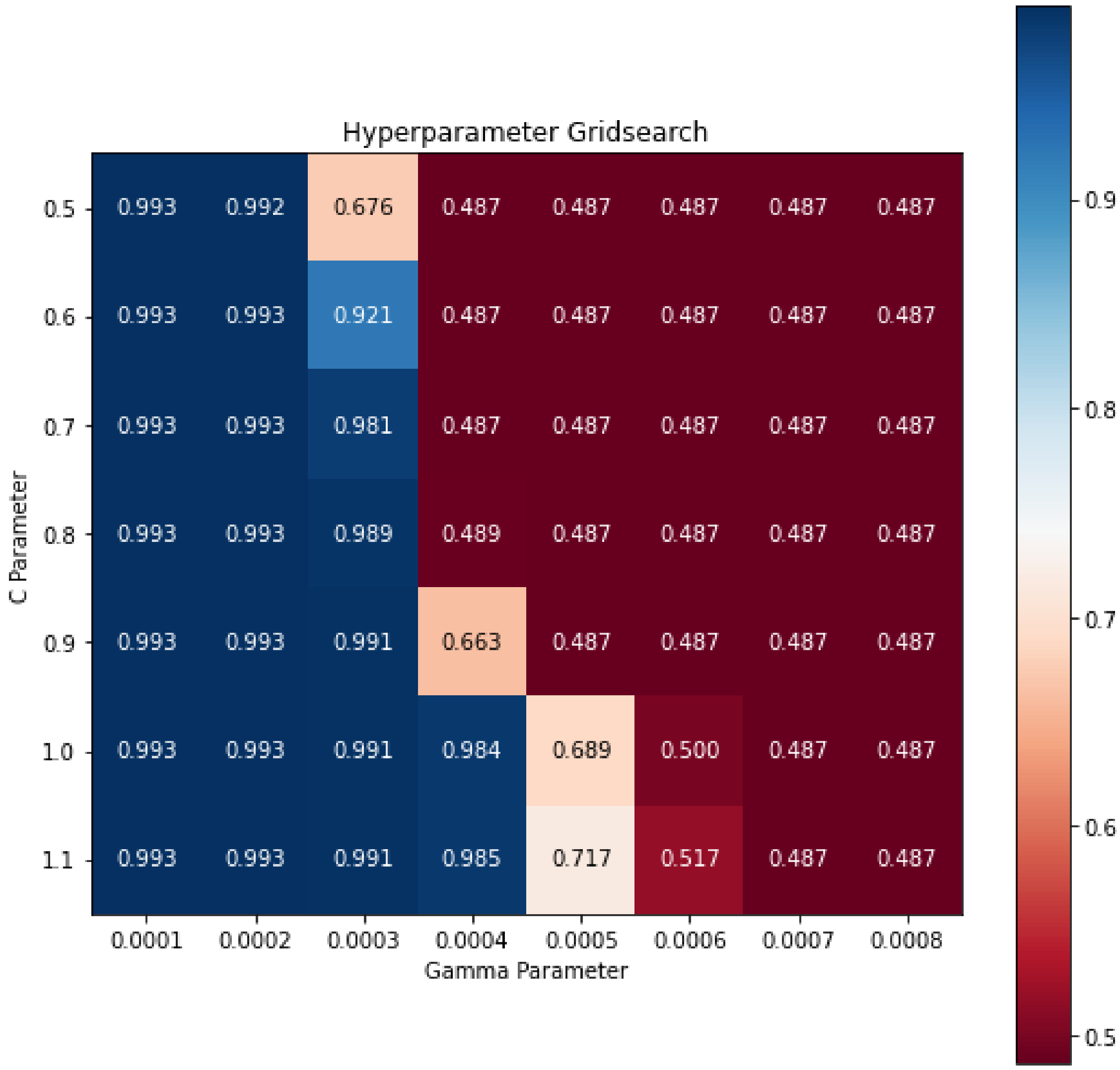
for example, x.shape[0]=20 and k=5

2. Grid Search & Cross-validation

```
print('best score: ', best_score)
print('best C: ', best_hyperparameters[0])
print('best gamma: ', best_hyperparameters[1])
✓ 0.1s
best score:  0.9932857142857143
best C:  0.7
best gamma:  0.0001
```

best score and best hyperparameters

3. Plot the grid search results of your SVM



Grid Search Result

Part. 2, Questions

1. Show that the kernel matrix $K = [k(x_n, x_m)]_{nm}$ should be positive semidefinite is the necessary and sufficient condition for $k(x, x')$ to be a valid kernel.

Assume a $N \times N$ gram matrix K, it is symmetric and semidefinite if and only if $K_{nm} = \phi(x_n)^T \phi(x_m)$, $K_{nm} = K_{mn}$ and $z \in \mathbb{R}^d$

Prove : positive semidefine $\Leftrightarrow$ $k(x, x')$ is valid

① K is positive semidefine $\Rightarrow$ $k(x, x')$ is valid

Since K is symmetric, $K = V \Lambda V^T$ where V is an orthonormal matrix V_t and the diagonal matrix Λ contains the eigenvalues λ_t of K. Moreover, K is positive semidefinite, so all eigenvalues are non-negative.

Next, consider the feature map : $\phi: x_i \mapsto (\sqrt{\lambda_t} V_{ti})_{t=1}^{n} \in \mathbb{R}^n$, we find that

$$\phi(x_i)^T \phi(x_j) = \sum_{t=1}^{n} \lambda_t V_{ti} V_{tj} = (V \Lambda V^T)_{ij} = K_{ij} = k(x_i, x_j) = k(x, x')$$

Thus, if K is positive semidefinite, $k(x, x')$ is a valid kernel. #

② $k(x, x')$ is a valid kernel $\Rightarrow$ K is positive semidefine

$$\begin{aligned}
z^T K z &= \sum_{n=1}^{N} \sum_{m=1}^{N} z_n k_{nm} z_m \\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} z_n \phi(x_n)^T \phi(x_m) z_m \\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} z_n \left(\sum_{k=1}^{N} \phi_k(x_n) \phi_k(x_m) \right) z_m \\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} \sum_{k=1}^{N} z_n \phi_k(x_n) \phi_k(x_m) z_m \\
&= \sum_{k=1}^{N} \left[\sum_{n=1}^{N} z_n \phi_k(x_n) \right]^2 \geq 0
\end{aligned}$$

Thus, if $k(x, x')$ is a valid kernel, K is positive semidefine #

2. Given a valid kernel $k_1(x, x')$, explain that $k(x, x') = exp(k_1(x, x'))$ is also a valid kernel. Your answer may mention some terms like ____ series or ____ expansion.

$k(x, x') = exp(k_1(x, x'))$

$exp(x) = \lim_{i \to \infty} \left(1 + x + \dots + \frac{x^i}{i!}\right)$

$\Rightarrow exp(k_1(x, x')) = 1 + k_1(x, x') + \frac{1}{2}[k_1(x, x')]^2 + \dots$

we can see that the exponential of a kernel is just an infinite series of multiplications and additions of the kernel

→ according to (6.13) $K = \alpha K_1$, $K = \beta K_2$ are valid kernels

→ according to (6.17) $K = \alpha K_1 + \beta K_2$ is a valid kernel (addition)

→ according to (6.18) $K = K_1 K_2$ (multiplication)

Thus, we can conclude that the exponential of a kernel is a valid kernel #

3. Given a valid kernel $k_1(x, x')$, prove that the following proposed functions are or are not valid kernels. If one is not a valid kernel, give an example of $k(x, x')$ that the corresponding K is not positive semidefinite and show its eigenvalues.

a. $k(x, x') = k_1(x, x') + 1$

First, prove that $k(x, x') = 1$ is a valid kernel

its kernel matrix must be $[1], \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \dots$

Let K is a $n \times n$ matrix, its possible eigenvalues are 0 and n

$\Rightarrow$ its a positive semidefinite matrix, so $k(x, x') = 1$ is a valid kernel

According to (6.17) $k(x, x') = k_1(x, x') + k_2(x, x')$ is a valid kernel

$k(x, x') = k_1(x, x') + 1$ is a valid kernel #

b. $k(x,x') = k_1(x,x') - 1$

First, prove that $k(x,x') = -1$ is a valid kernel

its kernel matrix must be $\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \dots$

Let K is a $n \times n$ matrix, its possible eigenvalues are 0 and $-n$ (negative)

$\Rightarrow K$ is not positive semidefinite

so, $k(x,x') = k_1(x,x') - 1$ is not a valid kernel #

c. $k(x,x') = k_1(x,x')^2 + exp(\|x\|^2) * exp(\|x'\|^2)$

① According to (6.18) $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ is a valid kernel

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

② According to (6.14) $k(x,x') = f(x) \cdot k_1(x,x') \cdot f(x')$ is a valid kernel

$\underbrace{exp(\|x\|^2)}_{f(x)} \times \underbrace{1}_{k_1(x,x')} \times \underbrace{exp(\|x'\|^2)}_{f(x')}$ is a valid kernel

↳ $k_1(x,x') = 1$ is a valid kernel proved in (a)

③ According to (6.17) $k(x,x') = k_1(x,x') + k_2(x,x')$ is a valid kernel

$k(x,x') = k_1(x,x')^2 + exp(\|x\|^2) \times exp(\|x'\|^2)$ is a valid kernel #

d. $k(x,x') = k_1(x,x')^2 + exp(k_1(x,x')) - 1$

① According to (6.18) $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ is a valid kernel

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

② According to (6.16) $k(x,x') = exp(k_1(x,x'))$ is a valid kernel

③ According to (6.17) $k(x,x') = k_1(x,x') + k_2(x,x')$ is a valid kernel

$k_1(x,x')^2 + exp(k_1(x,x'))$ is a valid kernel #

④ In (b), we know $k(x,x') = -1$ is not a valid kernel, and its possible eigenvalues are 0 and $-n$ #

so, $k(x,x') = k_1(x,x')^2 + exp(k_1(x,x')) - 1$ is not a valid kernel #

4. Consider the optimization problem

$$minimize \quad (x-2)^2 \quad subject \quad to \quad (x+3)(x-1) \leq 3$$

State the dual problem.

Lagrangian Function : $L = (x-2)^2 + \lambda\left[(x+3)(x-1)-3\right]$

$= x^2 - 4x + 4 + \lambda(x^2 + 2x - 6)$

$= (\lambda+1)x^2 + (2\lambda-4)x + (4-6\lambda)$

$\frac{L}{\partial x} = 2(\lambda+1)x + (2\lambda-4) = 0$

$\Rightarrow x = \frac{2-\lambda}{1+\lambda} \quad (\lambda \geq 0)$

$L = \frac{(2-\lambda)^2}{1+\lambda} + \frac{(2\lambda-4)(2-\lambda)}{1+\lambda} + (4-6\lambda)$

$= -\frac{(2-\lambda)^2}{1+\lambda} + (4-6\lambda)$

$= \frac{-(2-\lambda)^2 + (4-6\lambda)(1+\lambda)}{1+\lambda}$

$= \frac{-7\lambda^2 + 2\lambda}{1+\lambda}$

The dual problem : maximize $\frac{-7\lambda^2 + 2\lambda}{1+\lambda}$ subject to $\lambda \geq 0$ #