NYCU Pattern Recognition, Homework 4

Deadline: May 17, 23:59

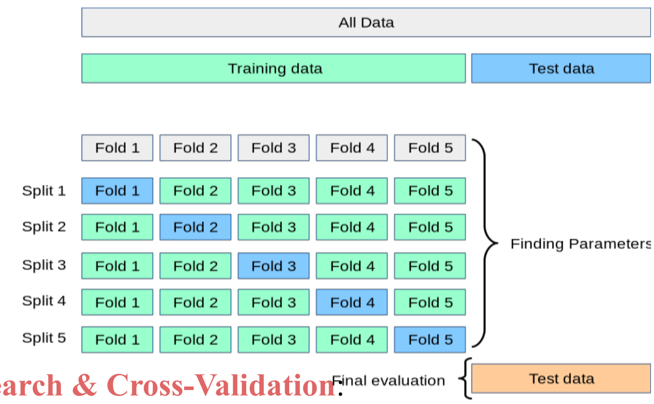
Part. 1, Coding (50%):

For this coding assignment, you are required to implement Cross-Validation and Grid Search using only NumPy. After that, you should train the SVM model from scikit-learn on the provided dataset and test the performance with the testing data. **You will get no points by simply calling sklearn.model_selection.GridSearchCV.**

(50%) K-Fold Cross-Validation & Grid Search

Requirements:

- Implement **K-Fold Cross-Validation** by creating a function that takes K as an argument and returns a list of K sublists.
    - Each sublist should contain two parts:
        - The first part contains the index of all training folds (index_x_train, index_y_train), for example, Fold 2 to Fold 5 in split 1.
        - The second part contains the index of the validation fold (index_x_val, index_y_val), for example, Fold 1 in split 1 .
    - You need to handle if the sample size is not divisible by K.
    - The first **n_samples % n_splits** folds should have a size of **n_samples // n_splits + 1**, and the other folds should have a size of **n_samples // n_splits**. Here, n_samples is the number of samples and n_splits is K.
    - Each of the samples should be used **exactly once** as the validation data.
    - Please **shuffle** your data before partition.



- Implement **Grid Search & Cross-Validation**:
    - Using sklearn.svm.SVC to train a classifier on the provided train set and perform **Grid Search** to find the best hyperparameters via cross-validation.

Criteria:

1. (10%) Implement K-fold data partitioning.

```python
def cross_validation(x_train, y_train, k=5, shuffle=True):

    # Do not modify the function name and always take 'x_train, y_train, k' as the inputs.
    # TODO HERE

    indices = np.arange(len(x_train))
    if shuffle :
        np.random.seed(42)
        np.random.shuffle(indices)

    fold_size = len(x_train) // k

    kfold_data = []

    for i in range(k):
        val_indices = indices[i*fold_size:(i+1)*fold_size]
        train_indices = np.concatenate([indices[:i*fold_size], indices[(i+1)*fold_size:]])

        kfold_data.append((train_indices, val_indices))

    kfold_data = np.array(kfold_data)

    return kfold_data
```

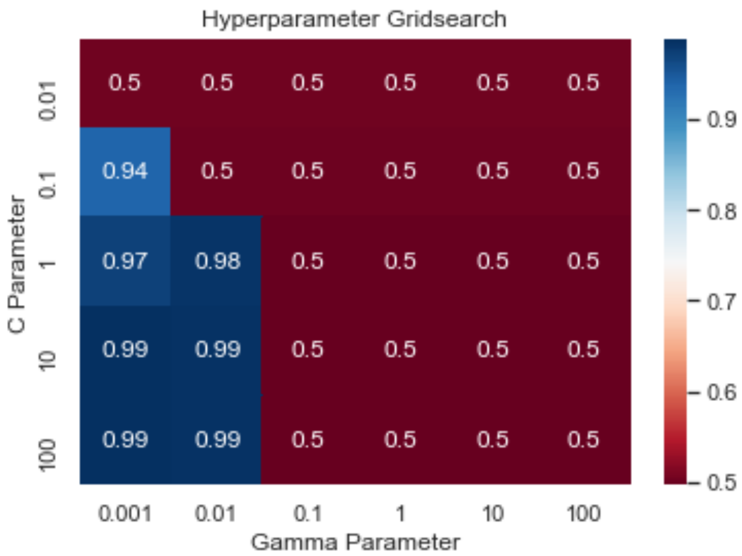
```python
kfold_data = cross_validation(x_train, y_train, k=10)
assert len(kfold_data) == 10 # should contain 10 fold of data
assert len(kfold_data[0]) == 2 # each element should contain train fold and validation fold
assert kfold_data[0][1].shape[0] == 700 # The number of data in each validation fold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

2. (10%) Set the kernel parameter to 'rbf' and do grid search on the hyperparameters **C** and **gamma** to find the best values through cross-validation. Print the best hyperparameters you found. Note that we suggest using K=5 for the cross-validation.

```python
print("(best_c, best_gamma) is ", best_parameters)
```

```
(best_c, best_gamma) is  (10, 0.001)
```

3. (10%) Plot the results of your SVM's grid search. Use "gamma" and "C" as the x and y axes, respectively, and represent the average validation score with color. Below image is just for reference.



4. (20%) Train your SVM model using the best hyperparameters found in Q2 on the entire training dataset, then evaluate its performance on the test set. Print your testing accuracy.

```python
# Do Not Modify Below

best_model = SVC(C=best_parameters[0], gamma=best_parameters[1], kernel='rbf')
best_model.fit(x_train, y_train)

y_pred = best_model.predict(x_test)

print("Accuracy score: ", accuracy_score(y_pred, y_test))

# If your accuracy here > 0.9 then you will get full credit (20 points).
```

```
Accuracy score:  0.988
```

Points	Testing Accuracy
20 points	acc > 0.9
10 points	0.85 <= acc <= 0.9
0 points	acc < 0.85

Part. 2, Questions (50%):

1. (10%) 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.

Suppose that $k(x,x')$ is a valid kernel, which means that there exists a feature space mapping $\phi(x)$ such that $k(x,x') = \phi(x)^T\phi(x')$

Then we can write a symmetric kernel K, whose elements are given by $k(x_n, x_m)$. That is, $K = k(x_n, x_m) = \phi(x_n)^T\phi(x_m)$

To show K is positive semidefinite, we need to show that for any vector $a = [a_1, a_2, \dots a_n]^T$, $a^TKa \geq 0$

$\Rightarrow a^TKa = a^T\phi(\cdot)\phi(\cdot)^Ta = (\phi(\cdot)^Ta)^T\phi(\cdot)^Ta = \underbrace{\|\phi(\cdot)^Ta\|^2}_{\text{Euclidean norm}} \geq 0$

$\because$ the Euclidean norm is non-negative

$\therefore$ K is positive semidefinite

2. (10%) Given a valid kernel $k_1(x, x')$, explain that $k(x, x') = exp(k_1(x, x'))$ is also a valid kernel. (Hint: Your answer may mention some terms like ____ series or ____ expansion.)

We express the exponential as a power series, yielding

$k(x,x') = exp(k_1(x,x')) = \sum_{m=0}^{\infty} \frac{(k_1(x,x'))^m}{m!}$

$\because$ this is a polynomial in $k_1(x,x')$ with positive coefficients,

It follows from $k(x,x') = g(k_1(x,x'))$, where $g(\cdot)$ function is a polynomial with non-negative coefficient and is also a valid kernel.

$\therefore$ $k(x,x') = exp(k_1(x,x'))$ is a valid kernel.

3. (20%) 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. $k(x, x') = k_1(x, x') + x$

This function is not a valid kernel.

If we choose $x = -1$, we have $k(x, x') = k_1(x, x') - 1$ and assume $k_1(x, x')$ is the standard Gaussian kernel

$\Rightarrow k_1(x, x') = \exp(-\|x - x'\|^2 / 2 \cdot \sigma^2)$ where σ is a positive constant, the matrix K is not positive semidefinite for some choices of $\{x_1, x_2\}$

when $x_1 = 0$ and $x_2 = 1$, $K = \begin{bmatrix} 0 & \exp(-1/2\sigma^2) & -1 \\ \exp(-1/2\sigma^2) & 0 & \exp(-1/2\sigma^2) \\ -1 & \exp(-1/2\sigma^2) & 0 \end{bmatrix}$

$\because$ the eigenvalues of the K are $[-0.61, -0.08, 0.69]$

$\therefore$ It has negative eigenvalues, $k(x, x')$ is not a valid kernel.

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

Let's choose $k_1(x, x') = (x^T x')^2$ which is a valid kernel,

then we have $k(x, x') = k_1(x, x') - 1 = (x^T x')^2 - 1$

If we use the following set of the inputs: $x_1 = [1, 0]$, $x_2 = [0, 1]$, $x_3 = [1, 1]$, so the kernel matrix K is

$K = [[0, -1, 0], [-1, 0, 0], [0, 0, 1]]$

$\Rightarrow \lambda = 1$ with multiplicity of 2, $\lambda = -1$

$\therefore$ The above result shows that K is not positive semidefinite

$\Rightarrow k(x, x') = k_1(x, x') - 1$ is not a valid kernel.

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

This function is a valid kernel

① $k_1(x, x')$ is a valid kernel, so $k_1(x, x')^2$ is also a valid kernel, this is the similar to the equation (6.18) of the course slide $\Rightarrow$ the first term is a valid kernel

② The second term is a positive constant term

$\therefore$ It is guaranteed to be positive semidefinite for any input x, x'

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

This function is not a valid kernel

$k_1(x,x') = x^T x' \Rightarrow k(x,x') = (x^T x')^2 + \exp(x^T x') - 1$

Let's take $x = (1,0)$ and $x' = (0,1)$, then $K = [[1,1],[1,1+e]]$

which is not positive semidefinite because it has negative eigenvalues.

$\therefore$ There exist vectors for K has negative eigenvalues

$\Rightarrow$ It's guaranteed that $k(x,x') = k_1(x,x')^2 + \exp(k_1(x,x')) - 1$ is not a valid kernel

4. Consider the optimization problem

$$\text{minimize } \underbrace{(x-2)^2}_{} = f(x)$$
$$\text{subject to } (x+4)(x-1) \le 3$$

$g(x) = (x+4)(x-1) - 3 \le 0$

State the dual problem. (Full points by completing the following equations)

$$L(x,\lambda) = f(x) + \lambda g(x) = (x-2)^2 + \lambda[(x+4)(x-1)-3]$$

$$\nabla_x L(x,\lambda) = 2(x-2) + \lambda(2x+3)$$

when $\nabla_x L(x,\lambda) = 0$,

$$x = \frac{4-3\lambda}{2+2\lambda}$$

$$L(x,\lambda) = L(\lambda) = \rule{5cm}{0.4pt}$$

$$L(\lambda) = \underbrace{\left(\frac{4-3\lambda}{2+2\lambda} - 2\right)^2}_{} + \lambda\underbrace{\left[\left(\frac{4-3\lambda}{2+2\lambda} + 4\right)\left(\frac{4-3\lambda}{2+2\lambda} - 1\right) - 3\right]}_{}$$

$$= \frac{49\lambda^2}{(2+2\lambda)^2} + \lambda\left[\frac{(5\lambda+12)(-5\lambda+2)}{(2+2\lambda)^2} - 3\right]$$

$$= \frac{49\lambda^2}{(2+2\lambda)^2} + \frac{\lambda(5\lambda+12)(-5\lambda+2)}{(2+2\lambda)^2} - 3\lambda$$

$$= -\frac{\lambda(25\lambda-24)}{4(\lambda+1)} - 3\lambda \quad \#$$