NYCU Pattern Recognition, Homework 4

Deadline: May 17, 23:59

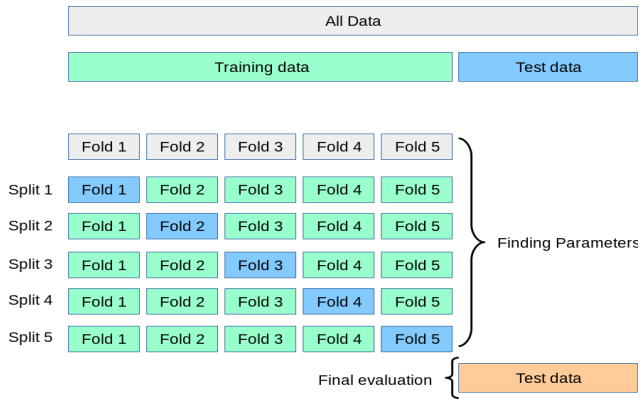
Part. 1, Coding (50%):

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):
    num_total = len(x_train)
    num_fold = int(num_total / k)
    num_remainder = num_total % k
    numList = [num_fold for i in range(k)]
    for i in range(num_remainder):
        numList[i] += 1

    indexList = []
    index = [i for i in range(num_total)]
    for i in range(k):
        fold_index = random.sample(index, numList[i])
        indexList.append(fold_index)
        index = list(set(index) - set(fold_index))

    fold_data = []
    index = [i for i in range(num_total)]
    for i in range(k):
        val_index = indexList[i]
        train_index = list(set(index) - set(val_index))
        val_index = np.array(val_index)
        train_index = np.array(train_index)
        fold_data.append([train_index, val_index])

    return fold_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
assert kfold_data[0][1].shape[0] == 700 # The number of data in each valida
```

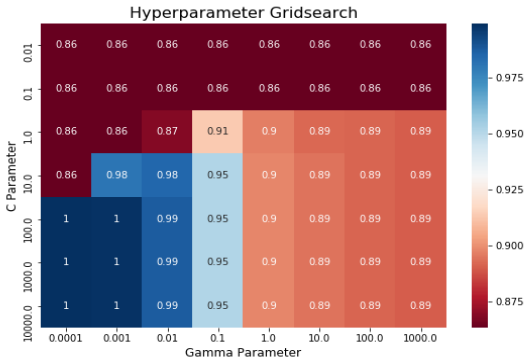
2. (10%) Set the kernel parameter to 'rbf' and do grid search on the hyperparameters **C** and **gamma** to find the best values through cross-validation. Print the best hyperparameters you found. Note that we suggest using K=5 for the cross-validation.

```python
def cross_val_score(clf, x_train, y_train, fold_data):
    scores = np.zeros(len(fold_data))
    for i in range(len(fold_data)):
        xtrain = x_train[fold_data[i][0], :]
        ytrain = y_train[fold_data[i][0]]
        xval = x_train[fold_data[i][1], :]
        yval = y_train[fold_data[i][1]]
        clf.fit(xtrain, ytrain)
        scores[i] = clf.score(xval, yval)
    mean_score = np.mean(scores)
    return mean_score

def Grid_Search(C_list, gamma_list, x_train, y_train, fold_data):
    Grid = np.zeros((len(C_list), len(gamma_list)))
    best_accuracy, best_C, best_gamma, best_model = 0, 0, 0, None
    for i, gamma in enumerate(gamma_list):
        for j, C in enumerate(C_list):
            clf = SVC(C=C, gamma=gamma, kernel='rbf')
            accuracy = cross_val_score(clf, x_train, y_train, fold_data)
            print(f"C = {C}\t  gamma = {gamma}\t  ----->  Accuracy = {accuracy}")
            Grid[j, i] = accuracy
            if accuracy > best_accuracy:
                best_accuracy = accuracy
                best_C = C
                best_gamma = gamma
                best_model = clf
        print("--------------------------------------------------------------------------------")
    return best_accuracy, best_C, best_gamma, Grid, best_model

# best_c, best_gamma = None, None
# # TODO HERE
# # k-Flod Cross Validation and Grid Search
# best_parameters=(best_c, best_gamma)
```

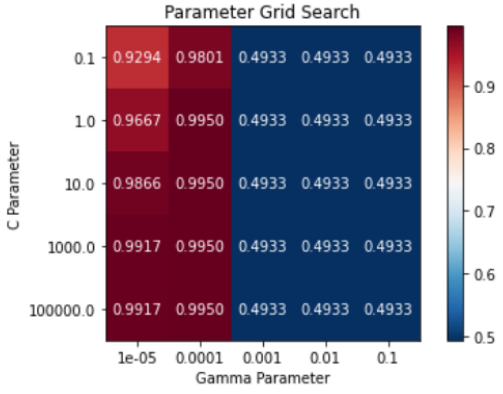
```python
kfold_data = cross_validation(x_train, y_train, k=5)
C_list = [0.1, 1.0, 1e1, 1e3, 1e5]
gamma_list = [0.00001, 0.0001, 0.001, 0.01, 0.1]
best_accuracy, best_C, best_gamma, Grid, best_model = Grid_Search(C_list, gamma_list, x_train, y_train, kfold_data)
best_parameters = (best_C, best_gamma)
print("(best_C, best_gamma) is ", best_parameters)
```

```
--------------------------------------------------------------------------------
(best_C, best_gamma) is  (1.0, 0.0001)
```

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.




```python
def heatmap(data, row_labels, col_labels, ax=None, cbar_kw=None, cbarlabel="", **kwargs):
    if ax is None:
        ax = plt.gca()
    if cbar_kw is None:
        cbar_kw = {}
    im = ax.imshow(data, **kwargs)                                  # Plot the heatmap
    cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)                 # Create colorbar
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")
    ax.set_xticks(np.arange(data.shape[1]))
    ax.set_yticks(np.arange(data.shape[0]))
    ax.set_xticklabels(labels=col_labels)
    ax.set_yticklabels(labels=row_labels)
    return im, cbar


def annotate_heatmap(im, data=None, valfmt="{x:.2f}", **textkw):
    if not isinstance(data, (list, np.ndarray)):
        data = im.get_array()
    kw = dict(horizontalalignment="center", verticalalignment="center")
    kw.update(textkw)
    if isinstance(valfmt, str):
        valfmt = matplotlib.ticker.StrMethodFormatter(valfmt)
    texts = []
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            kw.update(color="w")
            text = im.axes.text(j, i, valfmt(data[i, j], None), **kw)
            texts.append(text)
    return texts


plt.figure(figsize=(15,15))
fig, ax = plt.subplots()
im, cbar = heatmap(Grid, C_list, gamma_list, ax=ax, cmap="RdBu_r")
texts = annotate_heatmap(im, valfmt="{x:.4f}")
plt.title("Parameter Grid Search")
plt.ylabel("C Parameter")
plt.xlabel("Gamma Parameter")
fig.tight_layout()
plt.show()
```



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.

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

```
# 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:  0.995
```

Part. 2, Questions (50%):

1. (10%) Show that the kernel matrix $K = \left[k\left(x_n, x_m\right)\right]_{nm}$ should be positive semidefinite is the necessary and sufficient condition for $k(x, x')$ to be a valid kernel.

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.)

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. $k(x,\ x') = \ k_1(x,\ x') + \ x$ (There 's a typo, you may skip question 3.a)
   b. $k(x,\ x') = \ k_1(x,\ x') - 1$
   c. $k(x,\ x') = \ k_1(x, x')^2 + \ exp\left(\|x\|^2\right) * \ exp(\|x'\|^2)$
   d. $k(x,\ x') = \ k_1(x, x')^2 + \ exp\left(k_1(x, x')\right) - 1$
4. Consider the optimization problem

$$minimize\ (x - \ 2)^2$$
$$subject\ to\ (x + \ 4)(x - \ 1) \leq \ 3$$

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

$$L(x, \lambda) = \_\_\_\_\_$$

$$\nabla_x L(x, \lambda) = \_\_\_\_\_$$

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

$$x = \_\_\_\_\_$$

$$L(x, \lambda) = L(\lambda) = \_\_\_\_\_$$

(See the following pages.)

1.

1. (10%) Show that the kernel matrix $K = \left[k\left(x_n, x_m\right)\right]_{nm}$ should be positive semidefinite is the necessary and sufficient condition for $k(x, x')$ to be a valid kernel.

① K should be semidefinite $\Rightarrow$ $k(x, x')$ is a valid kernel

1° ∵ K is symmetric $\Rightarrow$ $K = V \Lambda V^T$

where V is an orthonormal matrix V_t

and Λ is an diagonal matrix containing eigenvalues λ_t of K

2° If K is positive semidefinite $\Rightarrow$ all eigenvalues ≥ 0

3° Consider $\phi: x_i \rightarrow \left(\sqrt{\lambda_t}\, V_{ti}\right)_{t=1}^{n} \in \mathbb{R}^n$

We find that $\phi(x_i)^T \phi(x_j) = \sum_{t=1}^{n} \lambda_t V_{ti} V_{tj}$

$= \left(V \Lambda V^T\right)_{ij} = K_{ij} = k(x_i, x_j)$

② $k(x, x')$ is a valid kernel $\Rightarrow$ K should be semidefinite

If $k(x, x')$ is a valid kernel, and K is the kernel matrix with $K_{ij} = k(x_i, x_j)$. Then:

$$c^T K c = \Sigma_i \Sigma_j c_i c_j K_{ij} = \Sigma_i \Sigma_j c_i c_j \phi(x_i) \phi(x_j)$$

$$= \Sigma_i c_i \phi(x_i) \cdot \Sigma_j c_j \phi(x_j) = \left\| \Sigma_j c_j \phi(x_j) \right\|^2 \geq 0$$

2.

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.)

① If $k(x, x')$ is a valid kernel, f is a polynomial with non-negative coefficients $\Rightarrow k'(x, x') = f(k(x, x'))$ is a valid kernel

pf: $k'(x, x') = a_1 k(x, x')^{e_1} + a_2 k(x, x')^{e_2} + \cdots$

1° All $k(x, x')^{e_i}$ are valid kernels by the product rule.

2° Thus all $a_i k(x, x')^{e_i}$ are valid kernels by the scaling rule.

3° Thus $k'(x, x')$ is a valid kernel by the sum rule.

② Consider that $\exp(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots + \frac{1}{i!}x^i + \cdots$

∴ By the limit of polynomials $\Rightarrow k(x, x') = \exp(k_1(x, x'))$ is a valid kernel #

3.

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. $k(x, x') = k_1(x, x') + x$~~

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

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

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

★ By theory:

$k_1(x, x')$ is a valid kernel

$\Leftrightarrow K = \begin{bmatrix} k_1(x_1, x_1) & \cdots & k_1(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k_1(x_n, x_1) & \cdots & k_1(x_n, x_n) \end{bmatrix}$ is positive semidefinite

$\Leftrightarrow \forall a \in \mathbb{R}^n,\ a^T K a \geq 0$

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

Suppose $k(x,x') = k_1(x,x') + k_2(x,x')$

where $k_1(x,x') = 0$ (valid, $\because K_1 = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$ is positive semidefinite)

$k_2(x,x') = -1$

Suppose $K = \begin{bmatrix} 0-1 & \cdots & 0-1 \\ \vdots & \ddots & \vdots \\ 0-1 & \cdots & 0-1 \end{bmatrix} = \begin{bmatrix} -1 & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & -1 \end{bmatrix}$ is positive semidefinite

But when we choose $a = [1\ 1 \cdots 1]$

$\Rightarrow a^T K a = -n^2 < 0$ ~~X~~

$\therefore k(x,x') = k_1(x,x') - 1$ is **not valid** #

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

1° By $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ — (6.18)

$k_1(x,x')^2$ is valid

2° Let $k_3(x,x') = 1$

$k_2(x,x') = \exp(\|x\|^2) \cdot k_3(x,x') \cdot \exp(\|x'\|^2)$

$\Rightarrow k_3(x,x') = 1$ is valid (proof in a.)

$\Rightarrow$ By $k(x,x') = f(x)\, k_1(x,x')\, f(x')$ — (6.14)

$k_2(x,x')$ is valid

3° $\therefore k(x,x') = k_1(x,x')^2 + k_2(x,x')$

By $k(x,x') = k_1(x,x') + k_2(x,x')$ — (6.15)

$\therefore k(x,x') = k_1(x,x')^2 + \exp(\|x\|^2) \cdot \exp(\|x'\|^2)$ is **valid** #

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

$k(x,x') = k_1(x,x')^2 - 1 + \left(1 + \frac{k_1(x,x')^2}{2!} + \frac{k_1(x,x')^3}{3!} + \dots + \frac{k_1(x,x')^n}{n!}\right)$

∵ By $k(x,x') = c\,k_1(x,x')$ — (6.13)

and $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ — (6.18)

⇒ each term in $k(x,x')$ is valid

∵ By $k(x,x') = k_1(x,x') + k_2(x,x')$

⇒ $k(x,x') = k_1(x,x')^2 + \exp(k_1(x,x')) - 1$ is valid #

4.

4. Consider the optimization problem

$$\begin{aligned} &\text{minimize } (x-2)^2 \\ &\text{subject to } (x+4)(x-1) \le 3 \end{aligned}$$

$3-(x+4)(x-1) \ge 0$

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

$L(x,\lambda) =$ _____

$\nabla_x L(x,\lambda) =$ _____

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

$x =$ _____

$L(x,\lambda) = L(\lambda) =$ _____

1° $L(x,\lambda) = (x-2)^2 + \lambda\left[(x+4)(x-1)-3\right]$

$= (x^2-4x+4) + \lambda(x^2+3x-7)$

where lagrange multiplier $\lambda \geq 0$

2° $\nabla_x L(x,\lambda) = 2x-4+\lambda(2x+3)$

$= 2x-4+2\lambda x+3\lambda$

$= (2\lambda+2)x + (3\lambda-4)$ #

3° When $\nabla_x L(x,\lambda) = 0 \Rightarrow \quad x = \frac{3\lambda-4}{2\lambda+2}$ #

4° By eliminating x from $L(x,\lambda)$:

$$L(\lambda) = \left(\frac{3\lambda-4}{2\lambda+2}-2\right)^2 + \lambda\left[\left(\frac{3\lambda-4}{2\lambda+2}\right)^2 + 3\left(\frac{3\lambda-4}{2\lambda+2}\right) - 7\right]$$

$$= (\lambda+1)\left(\frac{3\lambda-4}{2\lambda+2}\right)^2 + (3\lambda-4)\left(\frac{3\lambda-4}{2\lambda+2}\right) + (4-7\lambda)$$

$$= \frac{(\lambda+1)(3\lambda-4)^2 + (2\lambda+2)(3\lambda-4)^2}{(2\lambda+2)^2} + (4-7\lambda)$$

$$= \frac{3(\lambda+1)(3\lambda-4)^2}{4(\lambda+1)^2} + (4-7\lambda)$$

subject to $\lambda \geq 0$ #