NYCU Pattern Recognition, Homework 4

0810749, 張君實

Part. 1, Coding (50%):

Logistic Regression Model

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

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

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

    # TODO HERE
    k_fold_data = []

    #shuffle
    np.random.seed(0)
    train_size = np.shape(x_train)[0]
    shuffle_index = np.arange(train_size)
    np.random.shuffle(shuffle_index)
    for i in range(k):
        ##Calculate boundary
        left_bound = np.int64(i*train_size/k)
        right_bound = np.int64((i+1)*train_size/k)
        #Split the training data
        x_train_fold = np.concatenate((x_train[shuffle_index[0:left_bound]],x_train[shuffle_index[right_bound:train_size]]), axis=0)
        y_train_fold = np.reshape(np.concatenate((y_train[shuffle_index[0:left_bound]],y_train[shuffle_index[right_bound:train_size]]), axis=0),(-1,1))
        train_fold = np.concatenate((x_train_fold, y_train_fold), axis=1)
        x_val_fold = x_train[shuffle_index[left_bound: right_bound]]
        y_val_fold = np.reshape(y_train[shuffle_index[left_bound: right_bound]],(-1,1))
        val_fold = np.concatenate((x_val_fold, y_val_fold), axis=1)
        #Pack the data
        kth_data=[]
        kth_data.append(train_fold)
        kth_data.append(val_fold)
        k_fold_data.append(kth_data)
    return k_fold_data
```
✓ 0.0s

```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
```
[298] ✓ 0.2s

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
best_c, best_gamma = None, None

# TODO HERE
# k-Flod Cross Validation and Grid Search
#def accuracy_score(y, y_pred):
#    return (y_pred == y).sum()/np.shape(y)[0]

def grid_search( x_train, y_train, c, gamma, k):
    best_c, best_gamma, best_acc = None, None, 0
    kfold_data = cross_validation(x_train, y_train, k)
    total_acc = np.zeros((len(c),len(gamma)))
    for i_th ,c_ in enumerate(c):
        #print("i_th: ", i_th)
        for j_th, gamma_ in enumerate(gamma):
            #print("j_th: ", j_th)
            k_fold_acc = np.zeros(k)
            for k_th , k_th_data in enumerate(kfold_data):
                k_train = k_th_data[0]
                k_val = k_th_data[1]
                clf = SVC(C=c_, gamma=gamma_, kernel='rbf')
                clf.fit(k_train[:,0:-1], np.reshape(k_train[:,-1],-1))
                y_pred = clf.predict(k_val[:,0:-1])
                k_fold_acc[k_th] = accuracy_score(np.reshape(k_val[:,-1],-1), y_pred)
            total_acc[i_th][j_th] = np.average(k_fold_acc)
            #print("total_acc[i_th][j_th]: ", total_acc[i_th][j_th])
            if total_acc[i_th][j_th] >= best_acc:
                best_c = c_
                best_gamma = gamma_
                best_acc = total_acc[i_th][j_th]
    return best_c, best_gamma , total_acc

c = [0.01, 0.1, 1, 10, 100, 1000, 10000]
gamma = [1e-4, 1e-3, 0.01, 0.1, 1, 10, 100, 1000]
#c = [100, 1000]
#gamma = [10, 100]
k = 5
best_c, best_gamma , total_acc = grid_search(x_train, y_train, c, gamma, 5)
best_parameters=(best_c, best_gamma)
#print(total_acc)
```

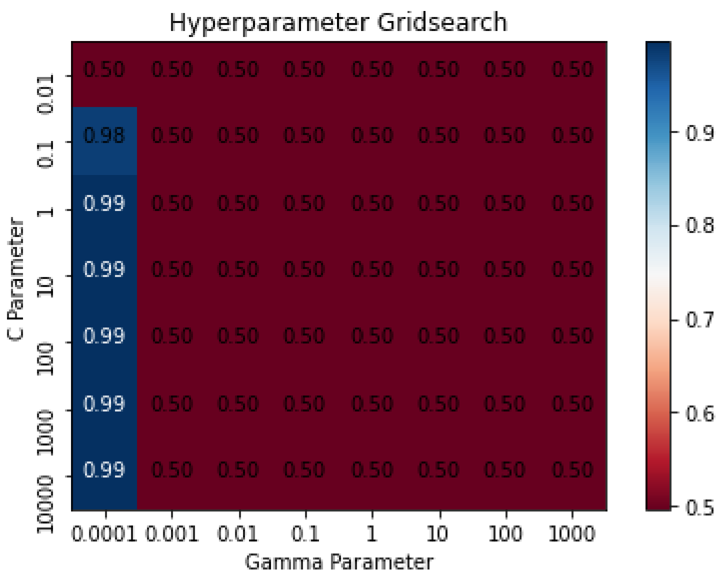
✓ 45m 20.8s

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

✓ 0.0s

```
(best_c, best_gamma) is  (10000, 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.**



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.**

Q4

```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).
```

[305] ✓ 11.6s

```
... Accuracy score:  0.995
```

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.

1.

If $k(x_n, x_m) = \phi(x_n)^T \phi(x_m) \Rightarrow$ Gram Matrix (kernel Matrix) K is positive semidefinite.

Prove:

$$k(x_n, x_m) = \phi(x_n)^T \phi(x_m) = \langle \phi(x_n), \phi(x_m) \rangle$$

$$= \langle \phi(x_m), \phi(x_n) \rangle = \phi(x_m)^T \phi(x_n)$$

$$= k(x_m, x_n)$$

We know the Gram Matrix K is the matrix such that $K_{mn} = k(x_m, x_n)$

$\Rightarrow K_{mn} = k(x_m, x_n) = k(x_n, x_m) = k_{nm} \Rightarrow$ Gram Matrix K is symmetric Matrix.

∵ K is symmetric $\Rightarrow x^T K x = x^T \Phi \Phi^T x = \|\Phi^T x\|^2 \geq 0$

If Gram Matrix K is symmetric and positive semidefinite $\Rightarrow k(x_n, x_m) = \phi(x_n)^T \phi(x_m)$

Prove: We know Gram Matrix K is symmetric

$\Rightarrow K_{mn} = k(x_n, x_m) = k(x_m, x_n) = K_{nm}$

If $k(x_n, x_m) = k(x_m, x_n)$, it means that we can find a basis ϕ that

$$k(x_n, x_m) = \phi(x_n)^T \phi(x_m) = \langle \phi(x_n), \phi(x_m) \rangle$$

$$\Rightarrow k(x_n, x_m) = \langle \phi(x_n), \phi(x_m) \rangle = \langle \phi(x_m), \phi(x_n) \rangle = k(x_m, x_n)$$

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.)

2、

We can use Taylor expansion:

If $k_1(x,x')$ is a valid kernel $=\langle\phi(x),\phi(x')\rangle$

$\Rightarrow e^{k_1(x,x')} = \lim_{N\to\infty} \sum_{n=0}^{N} \frac{1}{n!} k_1(x,x')^n$

Then, we need to use the power, scalings, sums properties.

If $k_1(x,y)$ and $k_2(x,y)$ are valid function:

Prove $a k_1(x,y)$ is valid kernel. (scalings)

Let $\phi'(x)=\sqrt{a}\,\phi(x)$

$\Rightarrow a k_1(x,y) = \langle \sqrt{a}\,\phi(x), \sqrt{a}\,\phi(y)\rangle = \langle\phi'(x),\phi'(y)\rangle$

Prove $k_1(x,y)+k_2(x,y)$ is valid kernel. (sum)

$\Rightarrow$ The Gram Matrix of $k_1(x,y)+k_2(x,y)$ is K_1+K_2

$\Rightarrow x^T(K_1+K_2)x = x^TK_1x + x^TK_2x \geq 0+0=0$

Prove $k_1(x,y)^n$ is valid kernel. (power)

$\Rightarrow$ We could use the property of "products".

The proof of products is below

$\Rightarrow$ Let Gram matrix K has kernel function $k_1(x,x')k_2(x,x')$

...... K_1 ...... $k_1(x,x')$

...... K_2 ...... $k_2(x,x')$

∵ K_1, K_2 are symmetric positive semi-definite

$$K_1 = \sum_{i=1}^{n} \lambda_i u_i u_i^T \quad , K_2 = \sum_{j=1}^{n} \mu_j v_j v_j^T$$

$$\Rightarrow K = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \mu_j (u_i u_i^T) \odot (v_j v_j^T)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \mu_j (u_i \odot v_j)(u_i \odot v_j)^T$$

$$= \sum_{k=1}^{n^2} r_k w_k w_k^T$$

$$\Rightarrow \forall a \in R^n , a^T K a = \sum_{k=1}^{n^2} r_k a^T w_k w_k^T a = \sum_{k=1}^{n^2} r_k (w_k^T a)^2 \geq 0$$

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. $\cancel{k(x, x') = k_1(x, x') + x}$ (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. $k(x, x') = k_1(x, x')^2 + exp\left(k_1(x, x')\right) - 1$

3.

(a) ~~It is not~~ a valid kernel,

$\Rightarrow \because \cancel{k(x,x')=k_1(x,x')+x \neq k_1(x',x)+x'=k(x',x)}$

There is a typo.

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

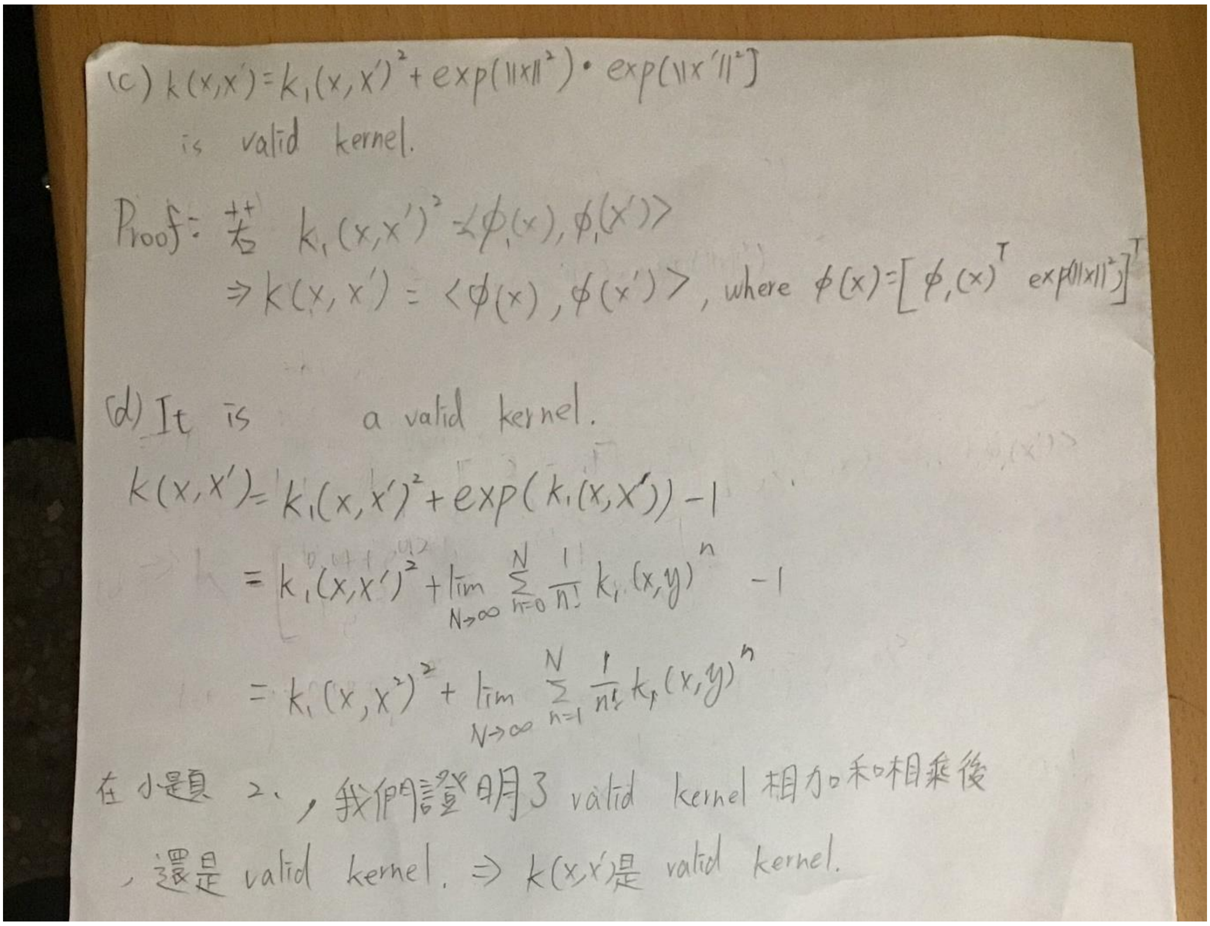
Proof:

Let $\phi_1(x_1)=\begin{bmatrix}1\\0\end{bmatrix}$ $\phi_1(x_2)=\begin{bmatrix}0\\1\end{bmatrix}$, $k_1(x,x')=\langle\phi_1(x),\phi_1(x')\rangle$

$$K_1=\begin{bmatrix}1&0\\0&1\end{bmatrix} \Rightarrow K=K_1-\begin{bmatrix}1&1\\1&1\end{bmatrix}=\begin{bmatrix}0&-1\\-1&0\end{bmatrix}$$

eigenvalue: $x^2-(-1)^2=0 \Rightarrow x=\pm1$.

$\Rightarrow$ It is not positive semidefinite.



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

is valid kernel.

Proof: 若 $k_1(x,x')^2 = \langle \phi_1(x), \phi_1(x') \rangle$

$\Rightarrow k(x,x') = \langle \phi(x), \phi(x') \rangle$, where $\phi(x) = [\phi_1(x)^T \ \exp(\|x\|^2)]^T$

(d) It is a valid kernel.

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

$$= k_1(x,x')^2 + \lim_{N\to\infty} \sum_{n=0}^{N} \frac{1}{n!} k_1(x,y)^n - 1$$

$$= k_1(x,x')^2 + \lim_{N\to\infty} \sum_{n=1}^{N} \frac{1}{n!} k_1(x,y)^n$$

在小題 2.，我們證明了 valid kernel 相加和相乘後，還是 valid kernel. $\Rightarrow k(x,x')$是 valid kernel.

4. Consider the optimization problem

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

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

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

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

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

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

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

4. consider the optimization problem:

minimize $(x-2)^2$

subject to $(x+4)(x-1) \le 3$

$\Rightarrow x^2+3x-4 \le 3$

$\Rightarrow x^2+3x-7 \le 0$

$$L(x,\lambda) = (x-2)^2 + \lambda(x^2+3x-7) = (1+\lambda)x^2 + (-4+3\lambda)x + (4-7\lambda)$$

$$\nabla_x L(x,\lambda) = 2(1+\lambda)x + (3\lambda-4)$$

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

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

dual problem:

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

subject $\lambda \ge 0$