NYCU Introduction to Machine Learning, Homework 4

Part. 1, Coding (50%):

```
Accuracy of using linear kernel (C = 5.0):  0.83
Accuracy of using polynomial kernel (C = 1.0, degree = 3):  0.98
Accuracy of using rbf kernel (C = 10.0, gamma = 0.9):  0.99
```

Part. 2, Questions (50%):

1. (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 shows its eigenvalues.

   ** reference: (from Ch6 PPT p.15)

$$
\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= ck_1(\mathbf{x},\mathbf{x}') && (6.13)\\
k(\mathbf{x},\mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}') && (6.14)\\
k(\mathbf{x},\mathbf{x}') &= q\left(k_1(\mathbf{x},\mathbf{x}')\right) && (6.15)\\
k(\mathbf{x},\mathbf{x}') &= \exp\left(k_1(\mathbf{x},\mathbf{x}')\right) && (6.16)\\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}') && (6.17)\\
k(\mathbf{x},\mathbf{x}') &= k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}') && (6.18)\\
k(\mathbf{x},\mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x},\mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' && (6.20)\\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a) + k_b(\mathbf{x}_b,\mathbf{x}'_b) && (6.21)\\
k(\mathbf{x},\mathbf{x}') &= k_a(\mathbf{x}_a,\mathbf{x}'_a)k_b(\mathbf{x}_b,\mathbf{x}'_b) && (6.22)
\end{aligned}
$$

   a. $k(x, x') = k_1(x, x') + \exp(x^T x')$

      Since $x^T x'$ is the inner product of x and x', it is essentially a valid kernel.
      Assume $k_2(x, x') = \exp(x^T x')$; it is a valid kernel according to (6.16).
      Then, $k(x, x') = k_1(x, x') + k_2(x, x')$ is also a valid kernel according to (6.17).
      Proved that $k(x, x') = k_1(x, x') + \exp(x^T x')$ is a valid kernel.

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

      Assume $K_1 = \begin{bmatrix} k_1(x_1,x_1) & k_1(x_1,x_2) \\ k_1(x_2,x_1) & k_1(x_2,x_2) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

      → $(1-\lambda)^2 = 0$, so $\lambda_1 = \lambda_2 = 1$, both of which are positive

      Then $K = \begin{bmatrix} k_1(x_1,x_1)-1 & k_1(x_1,x_2)-1 \\ k_1(x_2,x_1)-1 & k_1(x_2,x_2)-1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

→ λ^2-1= 0, so λ_1 = 1 and λ_2 = -1,

since $\lambda_2 < 0$, K is not positive semidefinite

Proved that k(x, x') = k_1(x, x')-1 is not a valid kernel.

c. k(x, x') = exp(‖x-x'‖^2)

Assume x_1=1, x_2=0,

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) \\ k(x_2, x_1) & k(x_2, x_2) \end{bmatrix} = \begin{bmatrix} e^{(\|1-1\|^2)} & e^{(\|1-0\|^2)} \\ e^{(\|0-1\|^2)} & e^{(\|0-0\|^2)} \end{bmatrix} = \begin{bmatrix} 1 & e \\ e & 1 \end{bmatrix}$$

→ $(1-\lambda)^2-e^2 = 0$, so λ_1 = 1+e and λ_2 = 1-e,

since $\lambda_2 < 0$, K is not positive semidefinite

Proved that k(x, x') = exp(‖x-x'‖^2) is not a valid kernel.

d. k(x, x')= exp(k_1(x, x')) - k_1(x, x')

Using Taylor expansion around 0,

$$\exp(k_1(x, x')) = 1 + k_1(x, x') + \frac{k1(x,x')^2}{2!} + \frac{k1(x,x')^3}{3!} + \frac{k1(x,x')^4}{4!} + \ldots$$

Therefore, $\exp(k_1(x, x')) - k_1(x, x') = 1 + \frac{k1(x,x')^2}{2!} + \frac{k1(x,x')^3}{3!} + \frac{k1(x,x')^4}{4!} + \ldots$

Each element $\frac{k1(x,x')^n}{n!}$ is a valid kernel according to (6.13) and (6.18), and the summation of them is also a valid kernel according to (6.17).

Proved that k(x, x')= exp(k_1(x, x')) - k_1(x, x') is a valid kernel.

2. (15%) One way to construct kernels is to build them from simpler ones. Given three possible "construction rules": assuming K_1(x, x') and K_2(x, x')are kernels then so are

1. (scaling) f(x)K_1(x, x')f(x'), f(x)∈R
2. (sum) K_1(x, x')+K_2(x, x')
3. (product) K_1(x, x')K_2(x, x')

Use the construction rules to build a normalized cubic polynomial kernel:

$$K(x, x')=(1+(\frac{x}{\|x\|})^T(\frac{x'}{\|x'\|}))^3$$

You can assume that you already have a constant kernel K_0(x, x') = 1 and a linear kernel $K_1(x, x')=x^Tx'$. Identify which rules you are employing at each step.

I. Suppose f(x) = $(\frac{1}{\|x\|})$ and f(x') = $(\frac{1}{\|x'\|})$, we construct the first kernel

$K_f(x, x') = (\frac{x}{\|x\|})^T(\frac{x'}{\|x'\|}) = (\frac{1}{\|x\|})x^T x'\left(\frac{1}{\|x'\|}\right) = f(x)K_1(x, x')f(x')$ using "scaling" rule.

II. Construct the second kernel $K_s(x, x') = 1 + K_f(x, x') = K_0(x, x') + K_f(x, x')$ using "sum" rule.

III. Construct the final kernel $K(x, x') = K_s(x, x')^3 = (K_s(x, x')\, K_s(x, x'))\, K_s(x, x')$ using "product" rule.

3. (15%) A social media platform has posts with text and images spanning multiple topics like news, entertainment, tech, etc. They want to categorize posts into these topics using SVMs. Discuss two multi-class SVM formulations: `One-versus-one` and `One-versus-the-rest` for this task.

a. The formulation of the method [how many classifiers are required]

In `One-versus-one`, classifiers are trained by comparing each pair of classes i and j. The formulation involves creating $\frac{k(k-1)}{2}$ classifiers, and it will classify one test point according to which class has the highest number of votes.

In `One-versus-the-rest`, each classifier is trained to distinguish one class from the rest, so k classifiers are required. Prediction is then made by selecting the class with the highest score, that is, $y(x) = \max_k y_k(x)$.

b. Key tradeoffs involved (such as complexity and robustness).

The key tradeoffs involve complexity, robustness, and efficiency.

`One-versus-one` is robust to imbalanced training data since each classifier is trained on a balanced subset of the data (instances from two specific classes). However, since it needs to compute $\frac{k(k-1)}{2}$ classifiers, the complexity is $O(k^2)$, leading to long prediction times and lower efficiency.

In contrast, `One-versus-the-rest` has lower complexity, $O(k)$, resulting in better prediction efficiency. But the disadvantage is that it may face challenges with imbalanced training data since classifiers are trained on one class against the rest.

c. If the platform has limited computing resources for the application in the inference phase and requires a faster method for the service, which method is better.

`One-versus-the-rest` is the better method for this case. As mentioned earlier, it only creates k classifier, aligning with the requirement of limited resources. Additionally, its complexity is only O(k), ensuring better efficiency and faster service compared to `One-versus-One`, whose complexity is $O(k^2)$.