Ziyang Jiao **475589**
Flora Sun **475178**

1 Problem 1

a) We can prove this using induction:
First, when t = 0:

$$\because \vec{w}^{(0)} = \vec{0}$$
$$\therefore \vec{w}^{(0)} = \sum_{i=1}^{n} \alpha_i^{(0)} \vec{x_i} \qquad \forall\ \alpha_i = 0$$

Then, assume $\vec{w}^{(k)} = \sum_{i=1}^{n} \alpha_i^{(k)} \vec{x_i}$, then we need to prove that:

$$\vec{w}^{(k+1)} = \sum_{i=1}^{n} \alpha_i^{(k+1)} \vec{x_i}$$

According to **hw1 p1(d)**, we have:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - c\left(C \sum_{i=1}^{n} I(1 - y_i \vec{w}^{(t)T} \vec{x_i} > 0)(-y_i\vec{x_i}) + 2\vec{w}^{(t)} \right)$$

Substitute $\vec{w}^{(t)}$ with $\sum_{i=1}^{n} \alpha_i^{(k+1)} \vec{x_i}$, we got:

$$\begin{aligned}
\vec{w}^{(t+1)} &= \sum_{i=1}^{n} \alpha_i^{(k)} \vec{x_i} - c\left(C \sum_{i=1}^{n} I(1 - y_i(\sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}^T)\vec{x_i} > 0)(-y_i\vec{x_i}) + 2\sum_{i=1}^{n} \alpha_i^{(k)} \vec{x_i} \right) \\
&= \sum_{i=1}^{n} \alpha_i^{(k)} \vec{x_i} + \sum_{i=1}^{n} \left(c * C * y_i * I(1 - y_i(\sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}^T)\vec{x_i} > 0) * \vec{x_i} \right) - \sum_{i=1}^{n} 2c\alpha_i^{(k)} \vec{x_i} \\
&= \sum_{i=1}^{n} \left(\alpha_i^{(k)} + c * C * y_i * I(1 - y_i(\sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}^T)\vec{x_i} > 0) - 2c\alpha_i^{(k)} \right) \vec{x_i} \\
&= \sum_{i=1}^{n} \alpha_i^{(k+1)} \vec{x_i} \\
where\ \ \alpha_i^{(k+1)} &= \alpha_i^{(k)} + c * C * y_i * I\left(1 - y_i(\sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}^T)\vec{x_i} > 0 \right) - 2c\alpha_i^{(k)}
\end{aligned}$$

In conclusion, after t gradient steps,

$$\vec{w}^{(t)} = \sum_{i=1}^{n} \alpha_i^{(t)} \vec{x_i} \qquad for\ some\ value\ \alpha_i^{(t)}$$

b) As the loss function of SVM is:

$$L(\vec{w}) = C\sum_{i=1}^{n} max\{1 - y_i \vec{w}^T \vec{x_i}, 0\} + \|\vec{w}\|_2^2$$

Substitute $\vec{w} = \sum_{i=1}^{n} \alpha_i \vec{x_i}$, we will get:

$$\begin{aligned}
L(\vec{\alpha}) &= C \sum_{i=1}^{n} max\{1 - y_i \sum_{j=1}^{n} \alpha_j \vec{x_j}^T \vec{x_i}, 0\} + (\sum_{i=1}^{n} \alpha_i \vec{x_i})^T (\sum_{j=1}^{n} \alpha_j \vec{x_j}) \\
&= C \sum_{i=1}^{n} max\{1 - y_i \sum_{j=1}^{n} \alpha_j \vec{x_j}^T \vec{x_i}, 0\} + \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \vec{x_i}^T \vec{x_j} \\
&= C \sum_{i=1}^{n} max\{1 - y_i \sum_{j=1}^{n} \alpha_j K_{ji}, 0\} + \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K_{ij}
\end{aligned}$$

c)

$$\begin{aligned}
\because \frac{\partial L(\vec{\alpha})}{\partial \alpha_m} &= C \sum_{i=1}^{n} I(1 - y_i \sum_{j=1}^{n} \alpha_j K_{ji} > 0)(-y_i K_{mi}) + \sum_{i=1}^{n} \alpha_i K_{im} + \sum_{j=1}^{n} \alpha_j K_{mj} \\
(\because K_{ij} = K_{ji}) \qquad &= C \sum_{i=1}^{n} I(1 - y_i \sum_{j=1}^{n} \alpha_j K_{ji} > 0)(-y_i K_{mi}) + 2 \sum_{i=1}^{n} \alpha_i K_{mi}
\end{aligned}$$

So the gradient descent update rule for SVM with respect to α_i is:

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + c \left(C \sum_{k=1}^{n} I(1 - y_k \sum_{j=1}^{n} \alpha_j^{(t)} K_{jk} > 0)(-y_k K_{ik}) + 2 \sum_{k=1}^{n} \alpha_k^{(t)} K_{ik} \right)$$

2 Problem 2

a)

$$G = X^T Z$$

In the equation above,

$$\begin{aligned} G_{ij} &= row\, i\, in\, X * column\, j\, in\, Z \\ &= x_i^T z_j \end{aligned}$$

b)

$$\begin{aligned} D_{ij}^2 &= (\vec{x_i} - \vec{z_j})^T (\vec{x_i} - \vec{z_j}) \\ &= \vec{x_i}^T \vec{x_i} - 2\vec{x_i}\vec{z_j} + \vec{z_j}^T \vec{z_j} \\ &= S_{ij} - 2G_{ij} + R_{ij} \end{aligned}$$

$$\therefore D = S - 2G + R$$

Where S and R can be computed with the code shown below using numpy:

```
import numpy as np

[n, m] = X.shape
# find x_i^T x_i in the diagonal of X^T X
Sdiag = X.T.dot(X).diagonal()
# repeat x_i^T x_i m times, so all elements in i-th row of S is x_i^T x_i
S = Sdiag.repeat(m).reshape(n,m)

# find z_j^T z_j in the diagonal of Z^T Z
Rdiag = Z.T.dot(R).diagonal()
# repeat z_j^T z_j n times and transpose, so all elements in j-th column of R is z_j^T z_j
R = Rdiag.repeat(n).reshape(m,n).T
```

c) I do not see any chances that D computed by

$$D = S - 2G + R$$

with S and R computed by the code above will contain negative values. But if there is, we can settle them into 0 by:

```
D[D < 0] = 0
```

3 Problem 3

a)

$$\begin{aligned}\langle\phi(\vec{x_i}),\phi(\vec{x_j})\rangle &= 1+2(\vec{x_i})_1(\vec{x_j})_1+2(\vec{x_i})_2(\vec{x_j})_2+2(\vec{x_i})_3(\vec{x_j})_3+(\vec{x_i})_1^2(\vec{x_j})_1^2+(\vec{x_i})_2^2(\vec{x_j})_2^2+(\vec{x_i})_3^2(\vec{x_j})_3^2+\\ &\quad (\vec{x_i})_1(\vec{x_i})_2(\vec{x_j})_1(\vec{x_j})_2+(\vec{x_i})_1(\vec{x_i})_3(\vec{x_j})_1(\vec{x_j})_3+(\vec{x_i})_2(\vec{x_i})_3(\vec{x_j})_2(\vec{x_j})_3 \qquad (1)\\ &= ((\vec{x_i})_1(\vec{x_j})_1+(\vec{x_i})_2(\vec{x_j})_2+(\vec{x_i})_3(\vec{x_j})_3+1)^2 \qquad (2)\\ &= (\vec{x_i}^T\vec{x_j}+1)^2 \qquad (3)\end{aligned}$$

$D = 10$
Because when there is only inner product of $\vec{x}$ involves in the objective fuction, considering the inner products directly (formula (3)) will take 8 operations, while explicitly computing the feature mapping (formula (2)) will take 19 operations. Thus, considering the inner products directly will be computationally cheaper but can achieve the same result as explicitly computing the feature mapping, so considering the inner products directly is more efficient.

b) (1) A1 is positive semidefinite:
Solve

$$det(A1-\lambda I)=\begin{vmatrix}1-\lambda & 1\\ 1 & 1-\lambda\end{vmatrix}=0$$

The eigenvalues of A1 are :

$$\lambda_1=2,\ \lambda_2=0$$
$$\because \lambda_1>0,\ \lambda_2=0$$
$$\therefore A1 \text{ is positive semidefinite.}$$

(2) A2 is strictly positive definite:
Solve

$$det(A2-\lambda I)=\begin{vmatrix}2-\lambda & 1 & 1\\ 1 & 2-\lambda & 0\\ 1 & 0 & 2-\lambda\end{vmatrix}=0$$

The eigenvalues of A2 are :

$$\lambda_1=0.5858,\ \lambda_2=3.4142,\ \lambda_3=2$$
$$\because \lambda_1>0,\ \lambda_2>0,\ \lambda_3>0$$
$$\therefore A2 \text{ is strictly positive definite.}$$

(3) A3 is neither strictly positive definite nor positive semidefinite:
Solve

$$det(A3-\lambda I)=\begin{vmatrix}2-\lambda & 1 & -1\\ 1 & 1-\lambda & 1\\ -1 & 1 & 2-\lambda\end{vmatrix}=0$$

The eigenvalues of A3 are :

$$\begin{aligned}&\lambda_1 = -0.4142,\ \lambda_2 = 3,\ \lambda_3 = 2.4142\\ &\because \lambda_1 < 0,\ \lambda_2 > 0,\ \lambda_3 > 0\\ &\therefore A3 \text{ is neither strictly positive definite nor positive semidefinite.}\end{aligned}$$

c)

$$\begin{aligned}K(\vec{x},\vec{x'}) &= e^{-\frac{1}{2l^2}||\vec{x}-\vec{x'}||^2}\\ &= e^{-\frac{1}{2l^2}\vec{x}^T\vec{x}}e^{\frac{1}{l^2}\vec{x}^T\vec{x'}}e^{-\frac{1}{2l^2}\vec{x'}^T\vec{x'}}\\ &= \left(\sum_{n=0}^{\infty}\frac{(-\frac{\vec{x}^T\vec{x}}{2l^2})^n}{n!}\right)\left(\sum_{m=0}^{\infty}\frac{(\frac{\vec{x}^T\vec{x'}}{l^2})^m}{m!}\right)\left(\sum_{p=0}^{\infty}\frac{(-\frac{\vec{x'}^T\vec{x'}}{2l^2})^p}{p!}\right)\\ &= \left(\sum_{n=0}^{\infty}\frac{(-\frac{\sum_{i=1}^d x_i^2}{2l^2})^n}{n!}\right)\left(\sum_{m=0}^{\infty}\frac{(\frac{\sum_{j=1}^d x_jx'_j}{l^2})^m}{m!}\right)\left(\sum_{p=0}^{\infty}\frac{(-\frac{\sum_{k=1}^d x_k^2}{2l^2})^p}{p!}\right)\\ &= \sum_{n=0}^{\infty}\sum_{m=0}^{\infty}\sum_{p=0}^{\infty}\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d}\alpha_{nmp}{x_i}^{2n}{x_j}^m{x'_j}^m{x'_k}^{2p}\end{aligned}$$

where α_{nmp} is some constant related to n, m and p. As we can see in the last line of the equation listed above, we can transfer the feature into any order (${x_i}^{2n}{x_j}^m$ where $m, n \in N$), and compute the inner product with another transformed vector. So RBF Kernel corresponds to an inner product in an infinite dimensional space.

d)

$$\begin{aligned}\tilde{k}(\vec{x},\vec{x'}) &= ck_1(\vec{x},\vec{x'})\\ &= \langle\sqrt{c}\phi(\vec{x}),\sqrt{c}\phi(\vec{x'})\rangle \qquad (\because c \geq 0)\\ &= \langle\tilde{\phi}(\vec{x}),\tilde{\phi}(\vec{x'})\rangle\end{aligned}$$

where $\tilde{\phi}(\vec{x}) = \sqrt{c}\phi(\vec{x})$ and $k_1(\vec{x},\vec{x'})$ is a valid kernel, so $\tilde{k}(\vec{x},\vec{x'})$ can be represented as an inner product of a fixed (non-linear) feature space mapping $\tilde{\phi}(\vec{x})$.
Also, as $k_1(\vec{x},\vec{x'})$ is a valid kernel, $k(\cdot,\cdot) : \mathcal{X}\times\mathcal{X}\to R$, $\tilde{k}(\cdot,\cdot) = ck(\cdot,\cdot)$ is also a map : $\mathcal{X}\times\mathcal{X}\to R$

e) 1) $k(\vec{x},\vec{x'}) = \vec{x}^T\vec{x'}$
Let $\phi(\vec{x}) = \vec{x}$, then $k(\vec{x},\vec{x'}) = \langle\phi(\vec{x}),\phi(\vec{x'})\rangle$. Thus $k(\vec{x},\vec{x'}) = \vec{x}^T\vec{x'}$ is a valid kernel.

2) $k(\vec{x},\vec{x'}) = ck_1(\vec{x},\vec{x'})$
proved in d)

3) $k(\vec{x}, \vec{x'}) = k_1(\vec{x}, \vec{x'}) + k_2(\vec{x}, \vec{x'})$

$$\begin{aligned} k(\vec{x}, \vec{x'}) &= k_1(\vec{x}, \vec{x'}) + k_2(\vec{x}, \vec{x'}) \\ &= \sum_{i=1}^{\infty} \lambda_{1i}\phi_{1i}(\vec{x})\phi_{1i}(\vec{x'}) + \sum_{i=1}^{\infty} \lambda_{2i}\phi_{2i}(\vec{x})\phi_{2i}(\vec{x'}) \\ &= \sum_{i=1}^{\infty} (\lambda_{1i} + \lambda_{2i})\phi_i(\vec{x})\phi_i(\vec{x'}) \end{aligned}$$

where $\phi_i(\vec{x})$ is a union of $\phi_{1i}(\vec{x})$ and $\phi_{2i}(\vec{x})$
And set λ_{1i} for the term that exists in $\phi_{2i}(\vec{x})$ but not in $\phi_{1i}(\vec{x})$ to 0. Set λ_{2i} for the term that exists in $\phi_{1i}(\vec{x})$ but not in $\phi_{2i}(\vec{x})$ to 0.

$$\begin{aligned} &\because\ k_1(\vec{x}, \vec{x'}) \text{ and } k_2(\vec{x}, \vec{x'})\text{are valid kernels.} \\ &\therefore\ \lambda_{1i} \geq 0,\ \lambda_{2i} \geq 0 \\ &\therefore\ \lambda_{1i} + \lambda_{2i} \geq 0 \\ &\therefore k(\vec{x}, \vec{x'}) = \sum_{i=1}^{\infty} \lambda_i\phi_i(\vec{x})\phi_i(\vec{x'}) \qquad \text{where } \lambda_i \geq 0 \\ &\therefore k(\vec{x}, \vec{x'}) \text{ is also a valid kernel.} \end{aligned}$$

4) $k(\vec{x}, \vec{x'}) = g(k_1(\vec{x}, \vec{x'}))$ where g is a polynomial with positive coefficients
Because of 2) 3) 5), $k(\vec{x}, \vec{x'})$ is a valid kernel.

5) $k(\vec{x}, \vec{x'}) = k_1(\vec{x}, \vec{x'}) \times k_2(\vec{x}, \vec{x'})$

6) $k(\vec{x}, \vec{x'}) = f(\vec{x})k_1(\vec{x}, \vec{x'})f(\vec{x'})$ where $f\ :\ R^d \to R$

7) $k(\vec{x}, \vec{x'}) = e^{k_1(\vec{x}, \vec{x'})}$
According to Taylor expansion,

$$\begin{aligned} k(\vec{x}, \vec{x'}) &= e^{k_1(\vec{x}, \vec{x'})} \\ &= \sum_{n=0}^{\infty} \frac{(k_1(\vec{x}, \vec{x'}))^n}{n!} \end{aligned}$$

Then according to 4), $k(\vec{x}, \vec{x'})$ is a valid kernel.

8) $k(\vec{x}, \vec{x'}) = \vec{x}^T A\vec{x'}$ where A is PSD.

$$\begin{aligned} &\because A = Q\Lambda Q^{-1} \\ &\therefore k(\vec{x}, \vec{x'}) = \vec{x}^T A\vec{x'} = \vec{x}^T Q\Lambda Q^{-1}\vec{x'} \end{aligned}$$

As A is PSD, then $\forall\lambda$ in $\Lambda \geq 0$
Let $\phi(\vec{x}) = Q^T\vec{x}$, then:

$$k(\vec{x}, \vec{x'}) = \sum_{i=1}^{\infty} \lambda_i\phi_i(\vec{x})\phi_i(\vec{x'}) \text{ with } \forall\lambda_i \geq 0$$

Therefore, $k(\vec{x}, \vec{x'})$ is a valid kernel.

4 Problem 4

a) We can prove this using induction:

1) base case:

$$\because \vec{w}^{(0)} = 0$$
$$\therefore \vec{w}^{(0)} = \sum_{i=1}^{n} \alpha_i^{(0)} \vec{x_i} \qquad with\, \forall \alpha_i^{(0)} = 0$$

2) Assume that $\vec{w}^{(k)} = \sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}$, then:
If there does not exist any misclassified data. Then $\vec{w}^{(k)}$ is our final $\vec{w}$ and it can be represented as:

$$\vec{w} = \sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j}$$

If there exists a misclassified data $(\vec{x_i}, y_i)$. Then:

$$\begin{aligned}\vec{w}^{(k+1)} &= \vec{w}^{(k)} + y_i \vec{x_i} \\ &= \sum_{j=1}^{n} \alpha_j^{(k)} \vec{x_j} + y_i \vec{x_i} \\ &= \sum_{j=1}^{n} \alpha_j^{(k+1)} \vec{x_j}\end{aligned}$$

where

$$\alpha_j^{(k+1)} = \begin{cases} \alpha_j^{(k)} & j \neq i \\ \alpha_j^{(k)} + y_i & j = i \end{cases}$$

In conclusion,

$$\vec{w} = \sum_{i=1}^{n} \alpha_i \vec{x_i}$$

b)

$$\begin{aligned}h(\vec{x}) &= sign(\vec{w}^T \vec{x}) \\ &= sign(\sum_{i=1}^{n} \alpha_i \vec{x_i}^T \vec{x})\end{aligned}$$

c)

Algorithm 1 kernelized learning perceptron

```
1: Initialize $\mathbf{w} \leftarrow 0$
2: repeat
3:     Pick $(\mathbf{x}_i, y_i)$ randomly from D
4:     if $y_i \sum_{j=1}^{n}(\alpha_j \mathbf{x}_j^T)\mathbf{x}_i \leq 0$ then
5:         $\alpha_i \leftarrow \alpha_i + y_i$
6: until convergence
```