hw3-ans

一、

(1) w初值为0。在算法的每一次更新过程中，w或是叠加上错误样本对应的$y_i x_i$，或是不做变化，因此最终的$w = \sum_{i\in[n]} \alpha_i y_i x_i$，其中$\alpha_i$为样本$i$在$T$轮循环中被分类错误的次数。

注意：该算法迭代T次，每次随机采样样本，而非遍历n个样本。因此回答$\alpha_i = 0$ or 1的同学扣除了少量分数。

(2) 由 (1)，$f(x) = \sum_{i\in[n]} \alpha_i y_i x_i^T x$。将$x_i^T x$替换成kernel形式即有：$f(x) = \sum_{i\in[n]} \alpha_i y_i \kappa(x_i, x)$。

(3) 将第三行中$w \leftarrow w + y_i x_i$替换成$\alpha_i \leftarrow \alpha_i + 1$即可。

二、

(1)

构造如下d维点集：$x_0 = (0,0,\dots,0)^T, x_1 = (1,0,\dots,0)^T, x_2 = (0,1,\dots,0)^T, \dots, x_d = (0,0,\dots,1)^T$。

考虑任意$[d]$的子集C（注意，不包括0）。

- 对于分类结果$y_i = +1$ if $i \in C$ else -1，$w = \sum_{i\in C} x_i, b = -0.1$的分类器可以正确分类。
- 对于分类结果$y_i = -1$ if $i \in C$ else $+1$，$w = -\sum_{i\in C} x_i, b = 0.1$的分类器可以正确分类。

以上两类包括了所有2^{d+1}种分类情况，因此$\mathcal{H}$可以打散$\mathbb{R}^d$中的$d+1$个点。

(2)

对于任意$d+2$个点$x_i \in \mathbb{R}^d, i \in [d+2]$，令$\hat{x}_i = [x_i^T, 1]^T \in \mathbb{R}^{d+1}$，则必存在一组不全为0的$\lambda_i, i \in [d+2]$，使得$\sum_{i\in[d+2]} \lambda_i \hat{x}_i = \mathbf{0}$。由于$\hat{x}_i$的最后一维是$1$，因此有$\sum_{i\in[d+2]} \lambda_i = 0$。因此可知，$\lambda_i, i \in [d+2]$中既存在正数，也存在负数。

令C为$[d+2]$的子集，使得$\lambda_i \geq 0 \iff i \in C$，有

$$\sum_{i\in C} \lambda_i \hat{x}_i = -\sum_{i\notin C} \lambda_i \hat{x}_i \tag{1}$$

$\mathcal{H}$中的线性分类器可以写作$h(x) = sign(\hat{w}^T \hat{x})$，其中$\hat{w} \in \mathbb{R}^{d+1}$。

考虑分类：$y_i = +1$ if $i \in C$ else -1。假设存在$\hat{w} \in \mathbb{R}^{d+1}$使该分类成立，将该$\hat{w}$乘到上式，有：

$$\sum_{i\in C} \lambda_i \hat{w}^T \hat{x}_i = -\sum_{i\notin C} \lambda_i \hat{w}^T \hat{x}_i \tag{2}$$

注意，根据分类结果，$\hat{w}^T \hat{x}_i > 0$ if $i \in C$ else < 0。结合$\lambda_i \geq 0$ if $i \in C$ else < 0，且$\lambda_i, i \in [d+2]$中存在正数，可以得知：上式左侧为正数，右侧为负数。因此矛盾，既不存在$\hat{w} \in \mathbb{R}^{d+1}$使该分类成立。由此证毕。

三、

(1) 根据VC generalization bound：

with a probability at least $1-\delta$，$\forall h \in \mathcal{H}$，$E_{out}(h) < E_{in}(h) + \sqrt{\frac{8}{n}\log\frac{2m_H(2n)}{\delta}}$

欲达到ϵ的泛化误差，即$\sqrt{\frac{8}{n}\log\frac{2m_H(2n)}{\delta}} \leqslant \epsilon$，因此$n \geqslant \frac{8}{\epsilon^2}\log\frac{2m_H(2n)}{\delta}$

(2) 令$m_H(2n) \approx (2n)^{d_{VC}}$，代入（1）计算得出的式子，有$n \geqslant \frac{8}{\epsilon^2}\log\frac{2\cdot(2n)^{d_{VC}}}{\delta}$

(3) 对（2）式子进行进一步简化：

$$n \geqslant \frac{8}{\epsilon^2}\log\frac{2\cdot(2n)^{d_{VC}}}{\delta} = \frac{8}{\epsilon^2}\log 2 + \frac{8d_{VC}}{\epsilon^2}\log 2n - \frac{8}{\epsilon^2}\log\delta \tag{3}$$

即：

$$f(n) = n - \frac{8}{\epsilon^2}\log 2 - \frac{8d_{VC}}{\epsilon^2}\log 2n + \frac{8}{\epsilon^2}\log\delta \geqslant 0 \tag{4}$$

```python
import numpy as np
```

✓ 0.0s

$f(n) = n - \frac{8}{\epsilon^2}\log 2 + \frac{8d_{VC}}{\epsilon^2}\log 2n - \frac{8}{\epsilon^2}\log\delta$ 是单调递增函数。

这里，采用二分法进行求解。low=0，high=100000（找到一个可以满足f(n)>0的值即可）

```python
def f(n,epsilon,delta,d_vc):
    f_n = n - 8 / (epsilon**2) * np.log(2) - 8 * d_vc / (epsilon**2) * np.log(2*n) + 8 / (epsilon**2) * np.log(delta)
    return f_n


def solve(epsilon,delta,d_vc):
    low = 0
    high = 1000000
    while low < high:
        mid = (low + high) / 2
        if f(mid,epsilon,delta,d_vc) < 0:
            low = mid + 1
        else:
            high = mid
    sample = np.ceil(high)
    assert f(sample,epsilon,delta,d_vc) >= 0
    print(f"d_vc:{d_vc}, sample size: {sample}")
```

✓ 0.0s

```python
epsilon = 0.1
delta = 0.1
solve(epsilon,delta,3)
solve(epsilon,delta,4)
solve(epsilon,delta,5)
```

✓ 0.0s

```
d_vc:3, sample size: 28695.0
d_vc:4, sample size: 38393.0
d_vc:5, sample size: 48312.0
```