HTML

Homework 3

B07502166 魏子翔

1. For training K-class classifier, we can choose two of them to train a binary classifier, and repeat for all possible combination. So the new data set size is $\frac{2N}{K}$, with repeating time $\binom{K}{2} = \frac{K(K-1)}{2}$. The final answer is $a(K-1)N$

2. With the union of quadratic, linear, and constant hypotheses, which is same as map the inputs to $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$, we have 6 parameters to decide, and have 6 data to help us construct 6 equations. So for any y, we should be able to find the only solution to the 6 parameters.

3. Calculate the $\Phi(x)$ for all x, then dot with w, w_1 and w_4 is correct.

4. Because $\Phi(x) = \Gamma x$, and X is consist of x^T, so the transformed X should be $X\Gamma^T$.
   $w_{lin} = (X^TX)^{-1}X^Ty$
   $\tilde{w} = ((X\Gamma^T)^T(X\Gamma^T))^{-1}(X\Gamma^T)^Ty = (\Gamma X^TX\Gamma^T)^{-1}\Gamma X^Ty = (\Gamma^T)^{-1}(X^TX)^{-1}X^Ty$
   $\Rightarrow \ w_{lin} = \Gamma^T\tilde{w}$
   $E_{in}(w_{lin}) = \frac{1}{N}||Xw_{lin} - y||^2$
   $E_{in}(\tilde{w}) = \frac{1}{N}||X\Gamma^T\tilde{w} - y||^2 = \frac{1}{N}||X\Gamma^T(\Gamma^T)^{-1}w_{lin} - y||^2 = \frac{1}{N}||Xw_{lin} - y||^2 = E_{in}(w_{lin})$

5. For a single $\mathcal{H}_k$, the growth function is $2N$, so for the union of $\mathcal{H}$, the growth function is $2Nd$.
   Since the meaning of d_{vc} is to bound the growth function, so d_{vc} is equal to the maximal value of N that satisfied $2Nd = 2^N$.
   $2Nd \le 2^N \ \Rightarrow \ 1 + \log N + \log d \le N \ \Rightarrow \ 1 + \log d \le N - \log N \le N - \frac{N}{2} = \frac{N}{2}$
   $\Rightarrow \ N \ge 2(1 + \log d)$
   For N larger than $2(1 + \log d)$, the growth function is smaller than 2^N, so the VC dimension is bounded by $2(1 + \log d)$.

6. If we replace x_n with $2x_n$, then the right right hand side of the definition of Φ will be false, which makes the output of $g(2x_n)$ be 0, so it may not equal to $2y_n$.

7. Brute-force checking all possible values within (-2, 2) with step=0.01.

8. Brute-force checking all possible values within (0, 4) with step=0.01.

9. $\nabla E_{aug}(w) = \nabla E_{in}(w) + \frac{2\lambda}{N} w$
   $\Rightarrow\ w_{t+1} \leftarrow w_t - \eta \nabla E_{aug}(w) = \left(1 - \frac{2\eta\lambda}{N}\right) w_t - \eta \nabla E_{in}(w)$

10. As the reduction process in class, the original equation can be written as:
    $\min \frac{1}{N+K}(||Xw - y||^2 + ||\tilde{X}w - \tilde{y}||^2$
    So we only need to reduce $||\tilde{X}w - \tilde{y}||^2$ into $\lambda||w||^2$, which comes out as $\tilde{X} = \sqrt{\lambda} I_{d+1}, \tilde{y} = 0$

11. For any $[i, i]$ entry in $E(X_h^T X_h)$, $E(X_h^T X_h)_{i,i} = E\left(\sum_{j=1}^{N} x_{j,i}^2 + \tilde{x}_{j,i}^2\right) = E\left(\sum_{j=1}^{N} 2x_{j,i}^2 + 2\epsilon x_{j,i} + \epsilon^2\right)$
    Since $E(\epsilon) = 0$, $E(\epsilon^2) = \int_{-r}^{r} \epsilon^2 \frac{1}{r-(-r)} d\epsilon = \frac{r^2}{3}$, so, $E(X_h^T X_h)_{i,i} = \sum_{j=1}^{N} (2x_{j,i}^2) + \frac{Nr^2}{3}$
    For those $[i,\ j]$, i != j, the different ϵ from the different entry cause the $E(\epsilon^2)$ become $E(\epsilon)E(\epsilon')$, which is zero, so when i != j, all ϵ-related item become 0, the last answer is $2X^T X + \frac{N}{3} r^2 I_{d+1}$

12. $\left(\frac{1}{N}\sum_{n=1}^{N}(y - y_n)^2 + \frac{\lambda}{N}\Omega(y)\right)' = \frac{1}{N}\sum_{n=1}^{N} 2(y - y_n) + \frac{\lambda}{N}\Omega'(y) = 0$
    $\Omega'(y) = \frac{2}{\lambda}\left(\sum_{n=1}^{N}(y_n) - Ny\right) = \frac{2}{\lambda}\left(\sum_{n=1}^{N} y_n - \frac{N}{N+2K}\left((\sum_{n=1}^{N} y_n) + K\right)\right) = \frac{2K}{\lambda}\frac{1}{N+2K}\left(2\sum_{n=1}^{N} y_n - N\right)$
    $= \frac{4K}{\lambda}(y - 0.5)$
    $\Rightarrow\ \Omega(y) = \frac{2K}{\lambda}(y - 0.5)^2$

13.
```python
import numpy as np

def read_data(filename):
    X, y = [], []
    with open(filename, "r") as f:
        for line in f:
            line = line[:-1]
            line = list(map(float, line.split()))
            X.append(line[:-1])
            y.append(int(line[-1]))
    return np.array(X), np.array(y)

def P13(X, y, X_test, y_test):
    eta = 0.001
    X = np.concatenate((np.ones((len(X), 1)), X), axis=1)
    X_test = np.concatenate((np.ones((len(X_test), 1)), X_test), axis=1)
    w_13 = np.linalg.pinv(X) @ y
    err_13 = np.sum((X@w_13-y)**2)/len(X)
    print(err_13)
    err_18 = np.sum(np.sign(X@w_13) != y)/len(X)
    err_out_18 = np.sum(np.sign(X_test@w_13) != y_test)/len(X_test)
    err_list_14 = []
    err_list_15 = []
    err_list_16 = []
    err_list_17 = []
    for i in range(1000):
        np.random.seed(i)
        w_14 = np.zeros((len(X[0])))
        w_15 = np.zeros((len(X[0])))
        w_16 = w_13
        w_17 = w_13
        for _ in range(800):
            pick = np.random.randint(0, len(X))
            w_14 += eta*2*(y[pick] - np.dot(w_14, X[pick]))*X[pick]
            w_15 += eta * \
                (1/(1+np.exp(y[pick]*np.dot(w_15, X[pick]))))*y[pick]*X[pick]
            w_16 += eta * \
                (1/(1+np.exp(y[pick]*np.dot(w_16, X[pick]))))*y[pick]*X[pick]
            w_17 += eta * \
                (1/(1+np.exp(y[pick]*np.dot(w_17, X[pick]))))*y[pick]*X[pick]
        err_14 = np.sum((X@w_14-y)**2)/len(X)
        err_list_14.append(err_14)
        err_15 = np.sum(np.log(1+np.exp(-np.multiply(X@w_15, y))))/len(X)
        err_list_15.append(err_15)
        err_16 = np.sum(np.log(1+np.exp(-np.multiply(X@w_16, y))))/len(X)
```

```
            err_list_16.append(err_16)
            err_17 = np.sum(np.sign(X@w_17) != y)/len(X)
            err_out_17 = np.sum(np.sign(X_test@w_17) != y_test)/len(X_test)
            err_list_17.append(abs(err_17-err_out_17))

        print(np.mean(err_list_14))
        print(np.mean(err_list_15))
        print(np.mean(err_list_16))
        print(np.mean(err_list_17))
        print(abs(err_18-err_out_18))

def P19(X, y, X_test, y_test):
    Q_list = [2, 8]
    for Q in Q_list:
        X_new = np.ones((len(X), 1))
        X_test_new = np.ones((len(X_test), 1))
        for q in range(Q):
            X_new = np.concatenate((X_new, X**(q+1)), axis=1)
            X_test_new = np.concatenate((X_test_new, X_test**(q+1)), axis=1)
        w = np.linalg.pinv(X_new) @ y
        err = np.sum(np.sign(X_new@w) != y)/len(X_new)
        err_out = np.sum(np.sign(X_test_new@w) != y_test)/len(X_test_new)
        print(abs(err-err_out))

if __name__ == "__main__":
    X, y = read_data("hw3_train.dat")
    X_test, y_test = read_data("hw3_test.dat")
    P13(X, y, X_test, y_test)
    P19(X, y, X_test, y_test)
```