CS 542 Homework 2

Jingye Qiu

TOTAL POINTS

163 / 180

QUESTION 1

Written Problems 60 pts

1.1 Bishop 3.3: Weighted Sum-of-Squares Error 9 / 15

✓ **+ 4 pts** **Rewrite Error function in terms of Matrix products**

✓ **+ 4 pts** **Take gradient of Error function**

✓ **+ 1 pts** **Set equation to 0 and solve for w**

+ 1 pts The log likelihood function of target vector t with Parameters: data dependent variance r_n^{-1}, Design Matrix Phi, and Rn the diagonal matrix of weights.

+ 2 pts We can observe that maximizing the above log-likelihood with respect to w is the same as minimizing the weighted square error function as stated in the problem.

+ 3 pts Consider that we have two replicated data points, then we would add two same terms in the sum-of-square error function, thus, we could just multiply a weight, r_n = 2, in front of the original error term corresponding to the specific data point.

+ 3 pts Alternative, and correct interpretation of relationship between the weighted SSE function and data dependent variance

+ 0 pts No submission

+ 4 pts Solution given is equivalent to correct solution

- 0.5 pts (Phi R t) should be (Phi^T R t)

+ 0 pts All answers incorrect

Missing interpretations

1.2 Bishop 3.11: Uncertainty with Linear Regression 13 / 15

+ 3 pts Find S^{-1}_{N+1} in terms of S^{-1}_N

✓ **+ 5 pts** **Derive S_{N+1} from the Matrix identity**

✓ **+ 5 pts** **S_N - S_{N+1} is PSD**

✓ **+ 2 pts** **Conclude the uncertainty with N is less than or equal than the uncertainty with N+1**

+ 15 pts Alternative and correct Proof

+ 0 pts Incorrect or Missing Proof

+ 1 **Point adjustment**

+1 Show your work. How did you derive S^{-1}_{N+1}?

1.3 Bishop 3.14: Constructing New Basis Function 11 / 15

- 0 pts Correct

✓ **- 4 pts** **First part is partially wrong**

- 8 pts First part is wrong or missing

- 4 pts Second part is partially wrong

- 7 pts Second part is wrong

1.4 Bishop 3.21: Optimal Value of alpha 10 / 15

- 0 pts Correct

- 10 pts 1st part is missing or wrong

✓ **- 5 pts** **2nd part is missing or wrong**

- 3 pts 2nd part is partially wrong or incomplete

QUESTION 2

Programming 120 pts

2.1 Linear Regression: Predicting Homicide Rate 60 / 60

✓ **- 0 pts** **Correctly identified the third variable with explanation**

- 10 pts used the right reasoning but the answer is not correct

- **10 pts** Didn't use own implementation of regression function
- **10 pts** Minor mistakes in reasoning and analysis
- **15 pts** Not enough reasoning and analysis
- **20 pts** No result analysis provided in the report
- **30 pts** Method used is not justified
- **30 pts** No source code
- **30 pts** No programming report provided.
- **60 pts** Couldn't find programming report or source code

2.2 Nearest Neighbor i) 15 / 15

✓ **- 0 pts Correct**

- **5 pts** For categorical values, mean may not make sense in some cases.
- **10 pts** Didn't explain how missing data is imputed in reporting report
- **15 pts** No solution found

> Note that for categorical labels, since there is usually no order, you should use mode, i.e. most frequent label rather than median as stated in the problem sheet.

2.3 Nearest Neighbor ii) 15 / 15

✓ **- 0 pts Correct**

- **15 pts** No solution

2.4 Nearest Neighbor iii) 15 / 15

✓ **- 0 pts Correct**

- **10 pts** Couldn't locate table summarizing accuracy for different k's
- **15 pts** No solution found

2.5 Nearest Neighbor iv) 15 / 15

✓ **- 0 pts Correct**

- **15 pts** Didn't find source code submission for problem 2

gradescope

3.3 Let $\nabla \ln E_D(w) = 0$ $\qquad r_n(t_n - w^T\phi(x))$

$$\nabla \ln E_D(w) = \sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}\phi(x_n)^T$$

$$= \sum_{n=1}^{N} r_n t_n \phi(x_n)^T - w^T \sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T$$

$$\Rightarrow \quad t^T R \Phi - w^T \Phi^T R \Phi = 0$$

$$w^* = (\Phi^T R \Phi)^{-1} \Phi^T R t$$

3.11 $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T S_{N+1} \phi$ $\qquad \left(\phi(x)^T S_N \phi(x) \geq \phi(x)^T S_{N+1} \phi(x)\right.$

$S_{N+1} = (S_N^{-1} + \beta \phi_{N+1} \phi_{N+1}^T)^{-1}$ $\qquad \left.\Leftrightarrow \phi(x)^T (S_{N+1} - S_N)\phi(x) \leq 0\right)$

$$= S_N - \frac{(S_N \phi_{N+1} \beta^{\frac{1}{2}})(\beta^{\frac{1}{2}} \phi_{N+1}^T S_N)}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

$$= S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

i) $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T \cdot \left(S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}\right)\phi$

$$= \sigma_N^2(x) - \frac{\beta \phi^T S_N \phi_{N+1} \phi_{N+1}^T S_N \phi}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}} \leq \sigma_N^2(x)$$

↳ Since S_N is positive definate, this is larger than zero.

3.14 Since $\alpha = 0$: $S_N = (\beta \phi^T \phi)^{-1}$

Let A be a square matrix, $\psi(x) = A\phi(x)$

$\phi(x) = A^{-1}\psi(x)$

$\because \Psi = \Phi A^T \Rightarrow \Phi = (A^{-1})^T \Psi$

i) $S_N = \beta^{-1}(\Phi^T \Phi)^{-1}$

$= \beta^{-1}[(A^{-1})^T \Psi^T \Psi A^{-1}]^{-1}$

$= \beta^{-1} A^T A$

ii) $k(x, x') = \beta \phi(x)^T S_N \phi(x')$

$= \phi(x)^T A^T A \phi(x')$

$= \psi(x)^T \psi(x')$

$= \psi_1(x)\psi_1(x) + \ldots + \psi_m(x)\psi_m(x_n)$

$+ \ldots + \psi_1(x)\psi_1(x_N) + \ldots + \psi_m(x)\psi_m(x_N)$

$= \psi_1(x) \cdot \sum_{n=1}^{N} \psi_1(x_n) + \ldots + \psi_m(x) \cdot \sum_{n=1}^{N} \psi_m(x_n)$

$= 1$

If $j = k$: $\sum_{n=1}^{N} k(x, x_n) = \sum_{n=1}^{N} \psi_j(x)^T \psi_k(x_n) = I_{jk} = 1$

If $j = 1, \psi_1 \equiv 1, i = 1$: $\sum_{n=1}^{N} \psi_i(x_n) = 1$

1.1 Bishop 3.3: Weighted Sum-of-Squares Error 9 / 15

✓ **+ 4 pts** **Rewrite Error function in terms of Matrix products**

✓ **+ 4 pts** **Take gradient of Error function**

✓ **+ 1 pts** **Set equation to 0 and solve for w**

+ 1 pts The log likelihood function of target vector t with Parameters: data dependent variance r_n^{-1}, Design Matrix Phi, and Rn the diagonal matrix of weights.

+ 2 pts We can observe that maximizing the above log-likelihood with respect to w is the same as minimizing the weighted square error function as stated in the problem.

+ 3 pts Consider that we have two replicated data points, then we would add two same terms in the sum-of-square error function, thus, we could just multiply a weight, r_n = 2, in front of the original error term corresponding to the specific data point.

+ 3 pts Alternative, and correct interpretation of relationship between the weighted SSE function and data dependent variance

+ 0 pts No submission

+ 4 pts Solution given is equivalent to correct solution

- 0.5 pts (Phi R t) should be (Phi^T R t)

+ 0 pts All answers incorrect

Missing interpretations

gradescope

3.3 Let $\nabla \ln E_D(w) = 0$ $\qquad r_n(t_n - w^T\phi(x))$

$$\nabla \ln E_D(w) = \sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}\phi(x_n)^T$$

$$= \sum_{n=1}^{N} r_n t_n \phi(x_n)^T - w^T \sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T$$

$$\Rightarrow \quad t^T R \Phi - w^T \Phi^T R \Phi = 0$$

$$w^* = (\Phi^T R \Phi)^{-1} \Phi^T R t$$

3.11 $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T S_{N+1} \phi$ $\qquad \left(\phi(x)^T S_N \phi(x) \geq \phi(x)^T S_{N+1} \phi(x)\right.$

$S_{N+1} = (S_N^{-1} + \beta \phi_{N+1} \phi_{N+1}^T)^{-1}$ $\qquad \left.\Leftrightarrow \phi(x)^T (S_{N+1} - S_N)\phi(x) \leq 0\right)$

$$= S_N - \frac{(S_N \phi_{N+1} \beta^{\frac{1}{2}})(\beta^{\frac{1}{2}} \phi_{N+1}^T S_N)}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

$$= S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

i) $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T \cdot \left(S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}\right)\phi$

$$= \sigma_N^2(x) - \frac{\beta \phi^T S_N \phi_{N+1} \phi_{N+1}^T S_N \phi}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}} \leq \sigma_N^2(x)$$

↳ Since S_N is positive definate, this is larger than zero.

3.14 Since $\alpha = 0$: $S_N = (\beta \phi^T \phi)^{-1}$

Let A be a square matrix, $\psi(x) = A\phi(x)$

$\phi(x) = A^{-1}\psi(x)$

$\because \Psi = \Phi A^T \Rightarrow \Phi = (A^{-1})^T \Psi$

i) $S_N = \beta^{-1}(\Phi^T \Phi)^{-1}$

$= \beta^{-1}[(A^{-1})^T \Psi^T \Psi A^{-1}]^{-1}$

$= \beta^{-1} A^T A$

ii) $k(x, x') = \beta \phi(x)^T S_N \phi(x')$

$= \phi(x)^T A^T A \phi(x')$

$= \psi(x)^T \psi(x')$

$= \psi_1(x)\psi_1(x) + \ldots + \psi_m(x)\psi_m(x_n)$

$+ \ldots + \psi_1(x)\psi_1(x_N) + \ldots + \psi_m(x)\psi_m(x_N)$

$= \psi_1(x) \cdot \sum_{n=1}^{N} \psi_1(x_n) + \ldots + \psi_m(x) \cdot \sum_{n=1}^{N} \psi_m(x_n)$

$= 1$

If $j = k$: $\sum_{n=1}^{N} k(x, x_n) = \sum_{n=1}^{N} \psi_j(x)^T \psi_k(x_n) = I_{jk} = 1$

If $j = 1, \psi_1 \equiv 1, i = 1$: $\sum_{n=1}^{N} \psi_i(x_n) = 1$

1.2 Bishop 3.11: Uncertainty with Linear Regression **13 / 15**

+ **3 pts** Find S^{-1}_{N+1} in terms of S^{-1}_N

✓ **+ 5 pts** **Derive S_{N+1} from the Matrix identity**

✓ **+ 5 pts** **S_N - S_{N+1} is PSD**

✓ **+ 2 pts** **Conclude the uncertainty with N is less than or equal than the uncertainty with N+1**

+ **15 pts** Alternative and correct Proof

+ **0 pts** Incorrect or Missing Proof

+ 1 Point adjustment

+1 Show your work. How did you derive S^{-1}_{N+1}?

gradescope

3.3 Let $\nabla \ln E_D(w) = 0$ $\qquad r_n(t_n - w^T\phi(x))$

$$\nabla \ln E_D(w) = \sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}\phi(x_n)^T$$

$$= \sum_{n=1}^{N} r_n t_n \phi(x_n)^T - w^T \sum_{n=1}^{N} r_n \phi(x_n)\phi(x_n)^T$$

$$\Rightarrow \quad t^T R \Phi - w^T \Phi^T R \Phi = 0$$

$$w^* = (\Phi^T R \Phi)^{-1} \Phi^T R t$$

3.11 $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T S_{N+1} \phi$ $\qquad \left(\phi(x)^T S_N \phi(x) \geq \phi(x)^T S_{N+1} \phi(x)\right.$

$S_{N+1} = (S_N^{-1} + \beta \phi_{N+1} \phi_{N+1}^T)^{-1}$ $\qquad \left.\Leftrightarrow \phi(x)^T (S_{N+1} - S_N)\phi(x) \leq 0\right)$

$$= S_N - \frac{(S_N \phi_{N+1} \beta^{\frac{1}{2}})(\beta^{\frac{1}{2}} \phi_{N+1}^T S_N)}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

$$= S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}$$

i) $\sigma^2_{N+1}(x) = \beta^{-1} + \phi^T \cdot \left(S_N - \frac{\beta S_N \phi_{N+1} \phi_{N+1}^T S_N}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}}\right)\phi$

$$= \sigma_N^2(x) - \frac{\beta \phi^T S_N \phi_{N+1} \phi_{N+1}^T S_N \phi}{1 + \beta \phi_{N+1}^T S_N \phi_{N+1}} \leq \sigma_N^2(x)$$

↳ Since S_N is positive definate, this is larger than zero.

3.14 Since $\alpha = 0$: $S_N = (\beta \phi^T \phi)^{-1}$

Let A be a square matrix, $\psi(x) = A\phi(x)$

$\phi(x) = A^{-1}\psi(x)$

i) $\psi = \phi A^T \Rightarrow \phi = (A^{-1})^T \psi$

i) $S_N = \beta^{-1}(\phi^T \phi)^{-1}$

$= \beta^{-1}[(A^{-1})^T \psi^T \psi A^{-1}]^{-1}$

$= \beta^{-1} A^T A$

ii) $k(x, x') = \beta \phi(x)^T S_N \phi(x')$

$= \phi(x)^T A^T A \phi(x')$

$= \psi(x)^T \psi(x')$

$= \psi_1(x)\psi_1(x) + \ldots + \psi_m(x)\psi_m(x_n)$

$+ \ldots + \psi_1(x)\psi_1(x_N) + \ldots + \psi_m(x)\psi_m(x_N)$

$= \psi_1(x) \cdot \sum_{n=1}^{N} \psi_1(x_n) + \ldots + \psi_m(x) \cdot \sum_{n=1}^{N} \psi_m(x_n)$

$= 1$

If $j = k$: $\sum_{n=1}^{N} k(x, x_n) = \sum_{n=1}^{N} \psi_j(x)^T \psi_k(x_n) = I_{jk} = 1$

If $j = 1, \psi_1 \equiv 1, i = 1$: $\sum_{n=1}^{N} \psi_i(x_n) = 1$

1.3 Bishop 3.14: Constructing New Basis Function 11 / 15

- **0 pts** Correct

✓ **- 4 pts** **First part is partially wrong**

- **8 pts** First part is wrong or missing
- **4 pts** Second part is partially wrong
- **7 pts** Second part is wrong

gradescope

3.21 $Au_i = \lambda_i u_i \quad |A| = \prod_{i=1}^{M} \lambda_i, \quad Tr(A) = \sum_{i=1}^{M} \lambda_i$

$$\ln|A| = \ln\left(\prod_{i=1}^{m} \lambda_i\right) = \sum_{i=1}^{m} \ln(\lambda_i)$$

$$\frac{d}{d\alpha} \ln|A| = \sum_{i=1}^{m} \frac{1}{\lambda_i} \frac{d}{d\alpha} \lambda_i = Tr\left(A^{-1} \frac{dA}{d\alpha}\right)$$

$$A = \sum_{i=1}^{m} \lambda_i u_i u_i^T$$

$$\therefore A^{-1} = \sum_{i=1}^{m} \frac{1}{\lambda_i} u_i u_i^T$$

$$Tr\left(A^{-1} \frac{d}{d\alpha} A\right) = Tr\left(\sum_{i=1}^{m} \frac{i}{\lambda_i} u_i u_i^T \frac{d}{d\alpha} \sum_{j=1}^{m} \lambda_j u_j u_j^T\right)$$

$$= Tr\left\{\sum_{i=1}^{m} \frac{1}{\lambda_i} u_i u_i^T \left[\sum_{j=1}^{m} \frac{d\lambda_j}{d\alpha} u_j u_j^T + \lambda_j (b_j u_j^T + u_j b_j^T)\right]\right\}$$

$$= Tr\left(\sum_{i=1}^{m} \frac{1}{\lambda_i} u_i u_i^T \sum_{j=1}^{m} \frac{d\lambda_j}{d\alpha} u_j u_j^T\right) + Tr\left[\sum_{i=1}^{m} \frac{1}{\lambda_i} u_i u_i^T \sum_{j=1}^{m} \lambda_j (b_j u_j^T + u_j b_j^T)\right]$$

$$= Tr\left(\sum_{i=1}^{n} \frac{1}{\lambda_i} u_i u_i^T \sum_{j=1}^{n} \frac{d\lambda_j}{d\alpha} u_j u_j^T\right) + Tr\left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_i} u_i u_i^T u_j \frac{du_j^T}{d\alpha} + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_i} u_i u_i^T \frac{du_j}{d\alpha} u_j^T\right)$$

$$= Tr\left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{\lambda_i} \frac{d\lambda_j}{d\alpha} u_i u_i^T u_j u_j^T\right) = Tr\left(\frac{d}{d\alpha} \sum_{i=1}^{m} u_i u_i^T\right) = \sum_{i=1}^{n} \frac{1}{\lambda_i} \frac{d\lambda_i}{d\alpha} \cdot Tr(u_i u_i^T)$$

$u_i^T u_j = 0$ when $i \neq j$

$$\therefore \sum_{i=1}^{m} u_i u_i^T = I$$

$$\therefore Tr\left(A^{-1} \frac{d}{d\alpha} A\right) = \sum_{i=1}^{m} \frac{1}{\lambda_i} \frac{d\lambda_i}{d\alpha}$$

1.4 Bishop 3.21: Optimal Value of alpha **10 / 15**

- **0 pts** Correct
- **10 pts** 1st part is missing or wrong

✓ **- 5 pts** **2nd part is missing or wrong**

- **3 pts** 2nd part is partially wrong or incomplete

gradescope

Programming Report

(a) Linear Regression

Formulas:

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \tag{1}$$

where **Φ** is the design matrix which contains 3 vectors including FTP WE and the testing variable, **t** is the HOM column vector

$$w_0 = \overline{t} - \sum_{j=1}^{M-1} w_j \overline{\phi_j} \tag{2}$$

where t and Φ with bars are their means

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x}_n)\}^2. \tag{3}$$

Thus, we can get the errors for each variable. And the variable with the smallest error might be the third variable.

After coding and running, we can get the third variable which might be GR or LIC since they are very close to each other (GR is smaller). Below is the plot of errors for each variables .

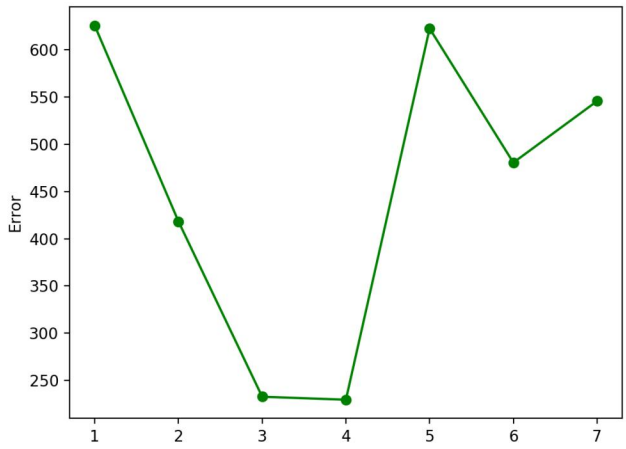

Figure 1: y-axis represents the error of each variable, x-axis represents variables as 1: UEMP, 2: MAN, 3: LIC, 4: GR, 5: NMAN, 6: GOV, 7: HE

Source Code

2.(a): Linear Regression

```
import pandas as pd
import numpy as np
import scipy.io as si
import matplotlib.pyplot as plt

def main():
    # Load ".mat" file, and create a DataFrame
    df = si.loadmat('/Users/qiujingye/Downloads/Homework2/detroit.mat')['data']
    df = pd.DataFrame(df)

    # Prepare the design matrix, combine FTP, WE and a test variable
    m = []
    for j in range(1, 8):
        temp = []
        for i in range(len(df)):
            row = [df[j][i],df[0][i],df[8][i]]
            temp.append(row)
        m.append(temp)

    # Test the rest 7 variables, and let t be the HOM column
    t = df[9]
    errors = []
    for i in range(7):
        # Create the design matrix p
        p = np.array(m[i])
        pb = range(3)
        tb = 0
        for j in range(len(p)):
            pb += p[j]
            tb += t[j]
        # Get the mean of Φ and t
        pb /= len(t)
        tb /= len(t)
        # Formula 1 on my report
        weights = (np.linalg.inv(p.T.dot(p))).dot(p.T).dot(t)
        # Formula 2 on my report
        wpb = 0
```

```
        for j in range(3):
            wpb += weights[j] * pb[j]
        w0 = tb - wpb
        # Formula 3 on my report
        e = 0
        for n in range(len(p)):
            wp = weights.T.dot(p[n])
            e += (t[n] - w0 - wp) ** 2
        e /= 2
        errors.append(e)

    print('\nThe errors of UEMP, MAN, LIC, GR, NMAN, GOV, HE are listed below:')
    print(errors)
    print('(The one with the smallest error is the third variable)')
    print('And the plot is as shown.')
    # Plot
    plt.plot(range(1, 8), errors, linestyle = '-', color = 'green', marker = 'o')
    plt.ylabel('Error')
    plt.show()

main()
```

2.1 Linear Regression: Predicting Homicide Rate 60 / 60

✓ - **0 pts** **Correctly identified the third variable with explanation**

- **10 pts** used the right reasoning but the answer is not correct
- **10 pts** Didn't use own implementation of regression function
- **10 pts** Minor mistakes in reasoning and analysis
- **15 pts** Not enough reasoning and analysis
- **20 pts** No result analysis provided in the report
- **30 pts** Method used is not justified
- **30 pts** No source code
- **30 pts** No programming report provided.
- **60 pts** Couldn't find programming report or source code

gradescope

(b) Nearest Neighbor

i.) Impute Missing Values:

For nominal features:

Replaced all "?" in feature 1 with b
Replaced all "?" in feature 4 with u
Replaced all "?" in feature 5 with g
Replaced all "?" in feature 6 with c
Replaced all "?" in feature 7 with v
Replaced all "?" in feature 9 with t
Replaced all "?" in feature 10 with f
Replaced all "?" in feature 12 with f
Replaced all "?" in feature 13 with g

For "+"labelled real-valued features:

Replaced all "?" in "+"labelled feature 2 with 33.72049180327869
Replaced all "?" in "+"labelled feature 3 with 5.904951140065147
Replaced all "?" in "+"labelled feature 8 with 3.427899022801302
Replaced all "?" in "+"labelled feature 11 with 4.605863192182411
Replaced all "?" in "+"labelled feature 14 with 164.421926910299
Replaced all "?" in "+"labelled feature 15 with 2038.85993485342

For "-"labelled real-valued features:

Replaced all "?" in "-"labelled feature 2 with 29.80823056300268
Replaced all "?" in "-"labelled feature 3 with 3.8399477806788513
Replaced all "?" in "-"labelled feature 8 with 1.2579242819843344
Replaced all "?" in "-"labelled feature 11 with 0.6318537859007833
Replaced all "?" in "-"labelled feature 14 with 199.6994680851064
Replaced all "?" in "-"labelled feature 15 with 198.60574412532637

Method:

Used Pandas module in Python to implement.

Output files:

crx.data.training.processed
crx.data.testing.processed

2.(b) i): Impute Missing Values

```
import pandas as pd

def main(filepath):
    # Load files and create DataFrame
    df = pd.read_csv(filepath, sep=',', header=None)
    training = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.training', sep=',',
header=None)
    testing = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing', sep=',',
header=None)

    # Combine training file and testing file for better accuracy
    frames = [training, testing]
    result = pd.concat(frames)

    # For text features, replace "?" with modes
    num=[0, 3, 4, 5, 6, 8, 9, 11, 12]
    for i in num:
        df[i].loc[df[i] == '?'] = result[i].loc[result[i] != '?'].mode()[0]
        print('Replaced all "?" of feature ' + str(i+1) + ' with ' +
str(result[i].loc[result[i] != '?'].mode()[0]))

    # Break the dataset into 2 parts ("+" and "-")
    df1 = result.loc[result[15] == '+']
    df2 = result.loc[result[15] == '-']

    # Change feature 2 and 14 to numerical
    df[1].loc[df[1]!='?'] = pd.to_numeric(df[1].loc[df[1]!='?'])
    df[13].loc[df[13] != '?'] = pd.to_numeric(df[13].loc[df[13] != '?'])

    # Replace "?" with mean for real-valued features
    df[1].loc[(df[1] == '?') & (df[15] == '+')] = pd.to_numeric(df1[1], errors='coerce').mean()
    df[2].loc[(df[2] == '?') & (df[15] == '+')] = df1[2].loc[df1[2]!= '?'].mean()
    df[7].loc[(df[7] == '?') & (df[15] == '+')] = df1[7].loc[df1[7]!= '?'].mean()
    df[10].loc[(df[10] == '?') & (df[15] == '+')] = df1[10].loc[df1[10]!= '?'].mean()
    df[13].loc[(df[13] == '?') & (df[15] == '+')] = pd.to_numeric(df1[13],
errors='coerce').mean()
    df[14].loc[(df[14] == '?') & (df[15] == '+')] = df1[14].loc[df1[14]!= '?'].mean()
    # Ones with "-"
    df[1].loc[(df[1] == '?') & (df[15] == '-')] = pd.to_numeric(df2[1], errors='coerce').mean()
```

```
    df[2].loc[(df[2] == '?') & (df[15] == '-')] = df2[2].loc[df2[2]!= '?'].mean()
    df[7].loc[(df[7] == '?') & (df[15] == '-')] = df2[7].loc[df2[7]!= '?'].mean()
    df[10].loc[(df[10] == '?') & (df[15] == '-')] = df2[10].loc[df2[10]!= '?'].mean()
    df[13].loc[(df[13] == '?') & (df[15] == '-')] = pd.to_numeric(df2[13],
errors='coerce').mean()
    df[14].loc[(df[14] == '?') & (df[15] == '-')] = df2[14].loc[df2[14]!= '?'].mean()

    # Print action
    print('Replaced all "?" in "+"labelled feature 2 with ' + str(pd.to_numeric(df1[1],
errors='coerce').mean()))
    print('Replaced all "?" in "+"labelled feature 3 with ' + str(df1[2].loc[df1[2]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 8 with ' + str(df1[7].loc[df1[7]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 11 with ' + str(df1[10].loc[df1[10]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 14 with ' + str(pd.to_numeric(df1[13],
errors='coerce').mean()))
    print('Replaced all "?" in "+"labelled feature 15 with ' + str(df1[14].loc[df1[14] !=
'?'].mean()))

    print('Replaced all "?" in "-"labelled feature 2 with ' + str(pd.to_numeric(df2[1],
errors='coerce').mean()))
    print('Replaced all "?" in "-"labelled feature 3 with ' + str(df2[2].loc[df2[2] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 8 with ' + str(df2[7].loc[df2[7] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 11 with ' + str(df2[10].loc[df2[10] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 14 with ' + str(pd.to_numeric(df2[13],
errors='coerce').mean()))
    print('Replaced all "?" in "-"labelled feature 15 with ' + str(df2[14].loc[df2[14] !=
'?'].mean()))

    # Save file
    p = filepath + '.processed'
    df.to_csv(p,index=False,header=False)
    print('\nSuccess!\n\nProcessed data saved to '+p)

main('/Users/qiujingye/Downloads/credit 2019/crx.data.training')
main('/Users/qiujingye/Downloads/credit 2019/crx.data.testing')
```

2.2 Nearest Neighbor i) 15 / 15

✓ **- 0 pts Correct**

- 5 pts For categorical values, mean may not make sense in some cases.

- 10 pts Didn't explain how missing data is imputed in reporting report

- 15 pts No solution found

Note that for categorical labels, since there is usually no order, you should use mode, i.e. most frequent label rather than median as stated in the problem sheet.

gradescope

ii.) k-NN Algorithm:

Output files:

Distance

LabelledTesting

iii.) k-NN Accuracy:

k-value	Accuracy
1	0.8115942028985508
2	0.8115942028985508
3	0.855072463768116
4	0.8623188405797102
5	0.8478260869565217
6	0.8623188405797102
7	0.855072463768116
8	0.8623188405797102
9	0.8478260869565217
10	0.8840579710144928
11	0.855072463768116
12	0.8695652173913043
13	0.8623188405797102
14	0.8985507246376812
15	0.8695652173913043
16	0.8840579710144928
17	0.8840579710144928
18	0.8913043478260869
19	0.8840579710144928
...	.....

Table1: Accuracy Table

Conclusion:

The best k-value would be 14. (I have tested k from 1 to 137, there is no accuracy of k that is larger than the accuracy of k = 14)

2.(b) ii): k-NN Algorithm

```
import pandas as pd
from math import sqrt

def main():
    # Load files
    df0 = pd.read_csv('/Users/qiujingye/Downloads/credit
2019/crx.data.training.processed', sep=',', header=None)
    df1 = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing.processed',
sep=',', header=None)
    # Combine for better accuracy
    frames = [df0, df1]
    result = pd.concat(frames)
    std = result.std()     # Standard deviation for z-scaling

    d = pd.DataFrame(index = [], columns = range(len(df1)))    # Create distance DataFrame

    # Sort
    values = [1, 2, 7, 10, 13, 14]
    strings = [0, 3, 4, 5, 6, 8, 9, 11, 12]
    # Calculate distance
    for i in range(len(df1)):
        temp = []
        for k in range(len(df0)):
            terms = 0
            for j in values:
                terms += ((df0[j][k] - df1[j][i])/std.loc[j]) ** 2
            for l in strings:
                if df0[l][k] != df1[l][i]:
                    terms += 1
            temp.append(sqrt(terms))
        d[i] =  temp
        print('\r' + str(i+1) + '/' + str(len(df1)) + ' finished',end='')
    # Save distance data
    d.to_csv('/Users/qiujingye/Downloads/credit 2019/Distance', index=False, header=False)

    print('\n\nDistance data saved.\n')

    # Calculate accuracy
    v = []
```

```
    for k in range(1, len(df1)):
        check = 0
        for i in range(len(df1)):

            num = d.nsmallest(k, i)
            temp = []
            for j in num.index.values:
                temp.append(df0[15][j])
            result = max(temp, key=temp.count)
            if result == df1[15][i]:
                check += 1
            accuracy = check / len(df1)
        v.append(accuracy)

        print('When k = ' + str(k) + ', the accuracy is ' + str(accuracy))
    v = pd.DataFrame(v)
    best = v[0].idxmax() + 1

    print('\nThe best k value is ' + str(best))

    # Save additional labelled data
    sign = []
    for i in range(len(df1)):
        num = d.nsmallest(k, i)
        temp = []
        for j in num.index.values:
            temp.append(df0[15][j])
        sign.append(max(temp, key=temp.count))
    df1[16] = sign

    df1.to_csv('/Users/qiujingye/Downloads/credit 2019/LabelledTesting', index=False,
header=False)

    print('\nLabelled testing data saved.')

main()
```

2.3 Nearest Neighbor ii) **15 / 15**

✓ **- 0 pts** **Correct**

- 15 pts No solution

gradescope

ii.) k-NN Algorithm:

Output files:

Distance

LabelledTesting

iii.) k-NN Accuracy:

k-value	Accuracy
1	0.8115942028985508
2	0.8115942028985508
3	0.855072463768116
4	0.8623188405797102
5	0.8478260869565217
6	0.8623188405797102
7	0.855072463768116
8	0.8623188405797102
9	0.8478260869565217
10	0.8840579710144928
11	0.855072463768116
12	0.8695652173913043
13	0.8623188405797102
14	0.8985507246376812
15	0.8695652173913043
16	0.8840579710144928
17	0.8840579710144928
18	0.8913043478260869
19	0.8840579710144928
...	.....

Table1: Accuracy Table

Conclusion:

The best k-value would be 14. (I have tested k from 1 to 137, there is no accuracy of k that is larger than the accuracy of k = 14)

2.(b) iii): k-NN Accuracy

```
import pandas as pd

def main():

    # Load files
    d = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/Distance', sep=',',
header=None)
    df0 = pd.read_csv('/Users/qiujingye/Downloads/credit
2019/crx.data.training.processed', sep=',', header=None)
    df1 = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing.processed',
sep=',', header=None)

    # Calculate accuracy
    v = []
    for k in range(1, len(df1)):
        check = 0
        for i in range(len(df1)):
            num = d.nsmallest(k, i)
            temp = []
            for j in num.index.values:
                temp.append(df0[15][j])
            result = max(temp, key=temp.count)
            if result == df1[15][i]:
                check += 1
            accuracy = check / len(df1)
        v.append(accuracy)

        print('When k = ' + str(k) + ', the accuracy is ' + str(accuracy))
    v = pd.DataFrame(v)
    best = v[0].idxmax() + 1

    print('\nThe best k value is ' + str(best))

    # Save additional labelled data
    sign = []
    for i in range(len(df1)):
        num = d.nsmallest(best, i)
        temp = []
        for j in num.index.values:
```

```
            temp.append(df0[15][j])
        sign.append(max(temp, key=temp.count))
    df1[16] = sign

    df1.to_csv('/Users/qiujingye/Downloads/credit 2019/LabelledTesting', index=False,
header=False)

    print('\nLabelled testing data saved.')

main()
```

2.4 Nearest Neighbor iii) **15 / 15**

✓ **- 0 pts** **Correct**

- 10 pts Couldn't locate table summarizing accuracy for different k's

- 15 pts No solution found

gradescope

```
            temp.append(df0[15][j])
        sign.append(max(temp, key=temp.count))
    df1[16] = sign

    df1.to_csv('/Users/qiujingye/Downloads/credit 2019/LabelledTesting', index=False,
header=False)

    print('\nLabelled testing data saved.')

main()
```

2.5 Nearest Neighbor iv) **15 / 15**

✓ **- 0 pts** **Correct**

- 15 pts Didn't find source code submission for problem 2

gradescope