Yilong Wang 483728
Linda Li 458029

Task 1

1. $w = (XX^T + \lambda I)^{-1} X y^T = \left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$= (1+4+9+\lambda)^{-1} \cdot 6$

$= \frac{6}{14.0001} \approx 0.429$

$g_1(x) = wx \approx 0.429x$

We have no β. $g = \begin{pmatrix} \beta \\ w \end{pmatrix}^T \begin{pmatrix} 1 \\ x \end{pmatrix}$ $\varphi(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}$

Results in poor fit is because all the data points have the same label $y = 1$. Ridge Regression gives larger weights to prevent overfitting, which leads to a compromised fit between identical labels and larger weight.

2. $X_2 = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix}$

$w = (XX^T + \lambda I)^{-1} X y^T$

$= \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 9 \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$= \left(\begin{bmatrix} 3 & 6 & 14 \\ 6 & 14 & 36 \\ 14 & 36 & 98 \end{bmatrix} + 0.0001 \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 3 \\ 6 \\ 14 \end{bmatrix}$

$= \begin{bmatrix} 3.0001 & 6 & 14 \\ 6 & 14.0001 & 36 \\ 14 & 36 & 98.0001 \end{bmatrix}^{-1} \begin{bmatrix} 3 \\ 6 \\ 14 \end{bmatrix}$

$= \begin{bmatrix} 0.9881 \\ 0.0021 \\ 0.0005 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

```
import numpy as np
A= np.array([[3.0001, 6, 14],
             [6, 14.0001, 36],
             [14, 36, 98.0001]])

A_inv = np.linalg.inv(A)

B = np.array([[3],
              [6],
              [14]])

result = np.dot(A_inv, B)

result
```

```
array([[ 9.98108233e-01],
       [ 2.09060667e-03],
       [-4.97725094e-04]])
```

$g_2(x) = w^T \cdot \varphi(x) = \begin{bmatrix} 0.9881 & 0.0021 & 0.0005 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$

$= 0.9881 + 0.0021x + 0.0005x^2$

This approximation is better, because this model has a constant term 0.9881.

3. a) Without regularization this model might be overfitting and lead to a poor generalization for new data.

b) $k_{i,j} = k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j) = \begin{bmatrix} 1 & x_i & x_i^2 \end{bmatrix} \begin{bmatrix} 1 \\ x_j \\ x_j^2 \end{bmatrix} = 1 + x_i x_j + x_i^2 x_j^2$

$k_{1,1} = \varphi(x_1)^T \varphi(x_1) = \begin{bmatrix} 1 & x_1 & x_1^2 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix} = 1 + x_1^2 + x_1^4 = 3$

$k_{2,2} = \varphi(x_2)^T \varphi(x_2) = \begin{bmatrix} 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \end{bmatrix} = 1 + 4 + 16 = 21$

$k_{3,3} = 1 + 9 + 81 = 91$

$k_{1,2} = k_{2,1} = 1 + 2 + 4 = 7$

$k_{1,3} = k_{3,1} = 1 + 3 + 9 = 13$

$k_{2,3} = k_{3,2} = 1 + 6 + 36 = 43$

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} = \begin{bmatrix} 3 & 7 & 13 \\ 7 & 21 & 43 \\ 13 & 43 & 91 \end{bmatrix}$$

$$\alpha = (K + \lambda I)^{-1} y^T = \left(\begin{bmatrix} 3 & 7 & 13 \\ 7 & 21 & 43 \\ 13 & 43 & 91 \end{bmatrix} + 0.0001\, I \right)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.99 \\ -2.99 \\ 0.995 \end{bmatrix}$$

```
import numpy as np
A= np.array([[3.0001, 7, 13],
             [7, 21.0001, 43],
             [13, 43, 91.0001]])

A_inv = np.linalg.inv(A)

B = np.array([[1],
              [1],
              [1]])

result = np.dot(A_inv, B)

result
```

```
array([[ 2.98884932],
       [-2.98546455],
       [ 0.99472346]])
```

$$g_3(x_{new}) = \sum_{i=1}^{3} \alpha_i k(x_i, x_{new}) = \alpha_1 k(x_1, x_{new}) + \alpha_2 k(x_2, x_{new}) + \alpha_3 k(x_3, x_{new})$$

$$= 2.99 \left[1 + x_{new} + x_{new}^2 \right] - 2.99 \left[1 + 2x_{new} + 4x_{new}^2 \right] + 0.995 \left[1 + 3x_{new} + 9x_{new}^2 \right]$$

$$= 0.99 - 0.02 x_{new} - 0.06 x_{new}^2$$

g_3 is same as g_2, because they represent the same model in different ways.

4. $k_{RBF}(x_i, x_j) = e^{\left(-\frac{\|x_i - x_j\|^2}{2\delta^2} \right)}$

for $\delta_1 = \sqrt{2}$:

$$K = \begin{bmatrix} 1 & e^{-\frac{1}{4}} & e^{-1} \\ e^{-\frac{1}{4}} & 1 & e^{-\frac{1}{4}} \\ e^{-1} & e^{-\frac{1}{4}} & 1 \end{bmatrix}$$

$$\alpha = (K + \lambda I)^{-1} y^T = \left(\begin{bmatrix} 1 & e^{-\frac{1}{4}} & e^{-1} \\ e^{-\frac{1}{4}} & 1 & e^{-\frac{1}{4}} \\ e^{-1} & e^{-\frac{1}{4}} & 1 \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.43 \\ -1.22 \\ 1.43 \end{bmatrix}$$

```
import math

import numpy as np
A= np.array([[1.0001, math.exp(-1/4), math.exp(-1)],
             [math.exp(-1/4), 1.0001, math.exp(-1/4)],
             [math.exp(-1), math.exp(-1/4), 1.0001]])

A_inv = np.linalg.inv(A)

B = np.array([[1],
              [1],
              [1]])

result = np.dot(A_inv, B)

result
```

```
array([[ 1.42723119],
       [-1.22293524],
       [ 1.42723119]])
```

$$g_{RBF1}(x) = \sum_{i=1}^{3} \alpha_i k(x_i, x)$$

$$= 1.43\, e^{-\frac{|1-x|^2}{4}} - 1.22\, e^{-\frac{|2-x|^2}{4}} + 1.43\, e^{-\frac{|3-x|^2}{4}}$$

for $\delta_2 = \frac{\sqrt{2}}{2}$:

$$K = \begin{bmatrix} 1 & e^{-1} & e^{-4} \\ e^{-1} & 1 & e^{-1} \\ e^{-4} & e^{-1} & 1 \end{bmatrix}$$

$$\alpha = (K + \lambda I)^{-1} y^T = \left(\begin{bmatrix} 1 & e^{-1} & e^{-4} \\ e^{-1} & 1 & e^{-1} \\ e^{-4} & e^{-1} & 1 \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

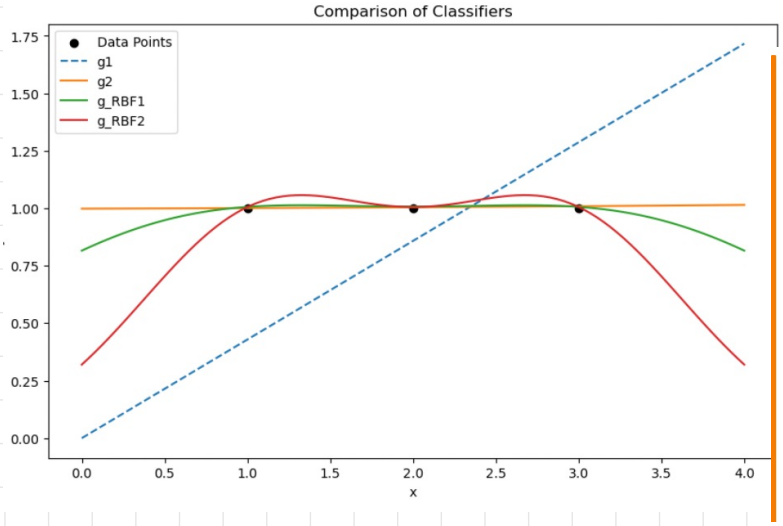
$$= \begin{bmatrix} 0.85 \\ 0.38 \\ 0.85 \end{bmatrix}$$

```
[7]: import math

     import numpy as np
     A= np.array([[1.0001, math.exp(-1), math.exp(-4)],
                  [math.exp(-1), 1.0001, math.exp(-1)],
                  [math.exp(-4), math.exp(-1), 1.0001]])

     A_inv = np.linalg.inv(A)

     B = np.array([[1],
                   [1],
                   [1]])

     result = np.dot(A_inv, B)

     result

[7]: array([[0.84538767],
            [0.37796072],
            [0.84538767]])
```

$$g_{RBF2}(x) = \sum_{i=1}^{3} \alpha_i k(x_i, x)$$

$$= 0.85\, e^{-|1-x|^2} + 0.38\, e^{-|2-x|^2} + 0.85\, e^{-|3-x|^2}$$

5.




```
[38]: import numpy as np
      import matplotlib.pyplot as plt

      X_plot = np.linspace(0, 4, 100)
      y_g1 = X_plot * 0.429

      def rbf_kernel_sigma(x1, x2, a):
          return np.exp(-np.linalg.norm(x1 - x2)**2 / (a))


      X2_plot = np.hstack((np.ones_like(X_plot)[:, np.newaxis], X_plot[:, np.newaxis], (X_plot**2)[:, np.newaxis]))
      y_g2_g3 = X2_plot.dot(np.array([[0.9981], [0.0021], [0.0005]]))

      alphas1 = np.array([1.43, -1.22, 1.43])
      alphas2 = np.array([0.85, 0.38, 0.85])

      X_original = np.array([1, 2, 3])
      y_original = np.array([1, 1, 1])

      g_RBF1_outputs = np.zeros_like(X_plot)
      g_RBF2_outputs = np.zeros_like(X_plot)

      for m in range(len(X_plot)):
          g_RBF1 = np.array([alphas1[i] * rbf_kernel_sigma(X_plot[m], X_original[i], 4) for i in range(len(X_original))])
          g_RBF2 = np.array([alphas2[i] * rbf_kernel_sigma(X_plot[m], X_original[i], 1) for i in range(len(X_original))])

          g_RBF1_outputs[m] = np.sum(g_RBF1)
          g_RBF2_outputs[m] = np.sum(g_RBF2)

      plt.figure(figsize=(10, 6))
      plt.scatter(X_original, y_original, color='black', label='Data Points')

      plt.plot(X_plot, y_g1, label='g1', linestyle='--')
      plt.plot(X_plot, y_g2_g3, label='g2 and g3')
      plt.plot(X_plot, g_RBF1_outputs, label='g_RBF1')
      plt.plot(X_plot, g_RBF2_outputs, label='g_RBF2')

      plt.xlabel('x')
      plt.ylabel('y')
      plt.legend()
      plt.show()
```

Task 2.

1. $k_{RBF}(x_i, x_j) = e^{\left(-\frac{\|x_i - x_j\|^2}{4}\right)}$

$$= e^{-\frac{1}{4}\left[(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \cdots + (x_{in}-x_{jn})^2\right]}$$

$$= e^{-\frac{1}{4}\sum_{m=1}^{n}(x_{im}-x_{jm})^2}$$

$$= \sum_{d=0}^{\infty} \frac{\left[-\frac{1}{4}\sum_{m=1}^{n}(x_{im}-x_{jm})^2\right]^d}{d!} \quad \Big| \quad e^x = \sum_{n=0}^{\infty}\frac{x^n}{n!}$$

$$= \sum_{d=0}^{\infty}\left[\frac{1}{d!}\cdot\left(-\frac{1}{4}\right)^d \cdot \left(\sum_{m=1}^{n}\left(x_{im}^2 - 2x_{im}x_{jm} + x_{jm}^2\right)\right)^d\right]$$

$$= \sum_{d=0}^{\infty}\left[\frac{1}{d!}\left(-\frac{1}{4}\right)^d \cdot \left(x_i^T \cdot x_i - 2x_i^T x_j + x_j^T x_j\right)^d\right]$$

$$= \sum_{d=0}^{\infty}\left[\frac{1}{d!}\left(-\frac{1}{4}\right)^d \cdot \left(k_1(x_i,x_i) - 2k_1(x_i,x_j) + k_1(x_j,x_j)\right)^d\right]$$

$$= e^{-\frac{1}{4}\left(k_1(x_i,x_i) - 2k_1(x_i,x_j) + k_1(x_j,x_j)\right)}$$

$$= e^{-\frac{1}{4}k_1(x_i,x_i)} \, e^{\frac{1}{2}k_1(x_i,x_j)} \, e^{-\frac{1}{4}k_1(x_j,x_j)}$$

$$= \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_1^d(x_i,x_i)\right) \cdot \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_1^d(x_i,x_j)\right) \cdot \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_1^d(x_j,x_j)\right)$$

$k_d(x_1,x_2) = (x_1^T x_2)^d = (k_1)^d$

$$= \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_d(x_i,x_i)\right) \cdot \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_d(x_i,x_j)\right) \cdot \sum_{d=0}^{\infty}\left(\frac{1}{d!}\left(-\frac{1}{4}\right)^d k_d(x_j,x_j)\right)$$

$$= e^{-\frac{1}{4}\sum_{i=1}^{D}\left(x_{1i}^2 - 2x_{1i}x_{2i} + x_{2i}^2\right)}$$

$$= e^{-\frac{1}{4}\left(\sum x_{1i}^2 - \frac{1}{2}\sum x_{1i}x_{2i} + \sum x_{2i}^2\right)}$$

$$= e^{-\frac{1}{4}\|x_1\|^2} \, e^{-\frac{1}{4}\|x_2\|^2} \cdot e^{\frac{1}{2}\sum x_{1i}x_{2i}}$$

$$= S\, e^{\frac{1}{2}\sum_{i=1}^{D} x_{1i}x_{2i}}$$

$$= S\sum_{d=0}^{\infty}\frac{\left(\frac{1}{2}\sum_{i=1}^{D} x_{1i}x_{2i}\right)^d}{d!} = S\sum_{d=0}^{\infty}\frac{\left(\sum_{i=1}^{D} x_{1i}x_{2i}\right)^d}{2^d d!}$$

2. a) assume C1 is english-class and C2 is german-class

We want to construct a w_φ, which has following characters:

$w_\varphi^T \varphi(x) > 0$ for all $x \in C1$

$w_\varphi^T \varphi(x) < 0$ for all $x \in C2$

By using one-hot encoding, we map words into a higher dimension. Each dimention corresponds with one word. This allow a linear classifier to operate in the higher dimension and seperate two language categories.

b) assume $\varphi(x) = \begin{bmatrix} 0 \\ \vdots \\ 1\ (x \text{ means yo/dude/What/Sexrus} \cdots) \\ \vdots \end{bmatrix}$

$$k(x,x') = \varphi(x)^T \cdot \varphi(x') = \begin{bmatrix} 0 & \cdots & 1(\text{for yo}) & \cdots & 1(\text{for dude}) & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1(\text{for yo}) \\ \vdots \\ 1(\text{for dude}) \\ \vdots \\ 0 \end{bmatrix} = 2$$

$k(x,x')$ means the number of the same words between x and x'

$\alpha = (K + \lambda I)^{-1} \varphi(y)$ with $K_{i,j} = k(x_i, x_j)$ and y is the label of train sets

assume x_1 = yo dude x_2 = What's up x_3 = Servus

$$y_1 = 1 \qquad y_2 = 1 \qquad y_3 = -1$$

$$K = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} = \begin{bmatrix} \varphi(x_1)^T\varphi(x_1) & \varphi(x_1)^T\varphi(x_2) & \varphi(x_1)^T\varphi(x_3) \\ \varphi(x_2)^T\varphi(x_1) & \varphi(x_2)^T\varphi(x_2) & \varphi(x_2)^T\varphi(x_3) \\ \varphi(x_3)^T\varphi(x_1) & \varphi(x_3)^T\varphi(x_2) & \varphi(x_3)^T\varphi(x_3) \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

then we calculate $\alpha = (K + \lambda I)^{-1} \varphi(y) = \begin{bmatrix} 2.0001 & 0 & 1 \\ 0 & 2.0001 & 0 \\ 1 & 0 & 1.0001 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$

$$= \begin{bmatrix} 2 \\ 0.5 \\ -3 \end{bmatrix}$$

```
2]: import numpy as np
    A= np.array([[2.0001, 0, 1],
                 [0, 2.0001, 0],
                 [1, 0, 1.0001]])

    A_inv = np.linalg.inv(A)

    B = np.array([[1],
                  [1],
                  [-1]])

    result = np.dot(A_inv, B)

    result
```

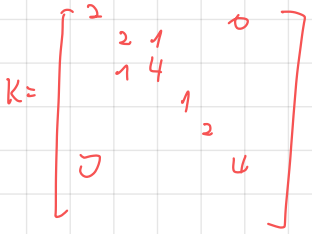
```
2]: array([[ 1.99950013],
           [ 0.499975  ],
           [-2.99920021]])
```

$$f_k(x) = \sum_{i=1}^{3} \alpha_i k(x, x_i)$$

$$= 2k(x, x_1) + 0.5k(x, x_2) - 3k(x, x_3)$$

test: f_k(hey there what's happening) $= 2 \cdot 0 + 0.5 \cdot 1 - 3 \cdot 0 = 0.5 > 0$

→ "hey there what's happening" is classified as english.

$K =$ 

$y^T = [1 \ 1 \ 1 \ -1 \ -1 \ -1]$

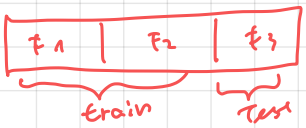
$\alpha = (K + \lambda I)^{-1} y^T$

$$= \begin{bmatrix} -1/3 \\ 3/2 \\ 11/9 \\ -1 \\ -2/3 \\ -1/4 \end{bmatrix}$$

$f_k(x_{new}) = sgn\left[\sum \alpha_i k(x_i, x_{new})\right]$ $\qquad \alpha = \begin{cases} 1 & x_i \text{: english} \\ -1 & x_i \text{: germany} \end{cases}$

Task 3

- Using a lot of data per folds (Fewer folds) Big Slices



Pros: Evaluation very accurate

1. Computational efficiency: Fewer folds mean fewer iterations of training and validation, which can be significantly faster, especially with large and complex models.

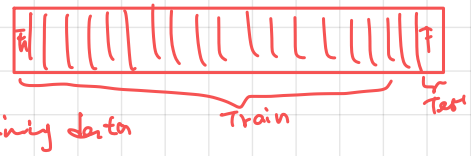
2. Lower Bias: Using larger data sets helps the model to learn and capture the real relationships in the data more effectively thereby reducing the models bias.

Cons: Model is not well trained

6.1 we have less data to train on, since we are using all of the data to train that is not used for testing -1

1. Increase in Variance: Fewer folds mean more overlap between each training set, which might make the model more sensitive to small fluctuation in the training data, thus increasing variance.

- Using little data per fold (More folds)



Pros: Model well-trained because so much training data

1. Low Variance: More folds mean the model is tested across a wider variety of training and test sets, which can reduce the variance in the performance estimate.

2. Fine-grained Evaluation: Having many folds allows for a more nuanced evaluation, as the model is assessed under a broader range of data scenarios.

Cons: testing is inaccurate

1. Higher Bias: With less data in each fold, there's a higher chance that the fold doesn't represent the overall dataset well. This could introduce more bias into the evaluation, thus leading to overfitting.

2. Computational Intensity: While each individual training might be faster due to less data, the overall process might be computationally intense because of a larger number of folds.

OLS: $w = (XX^T)^{-1} X y^T$

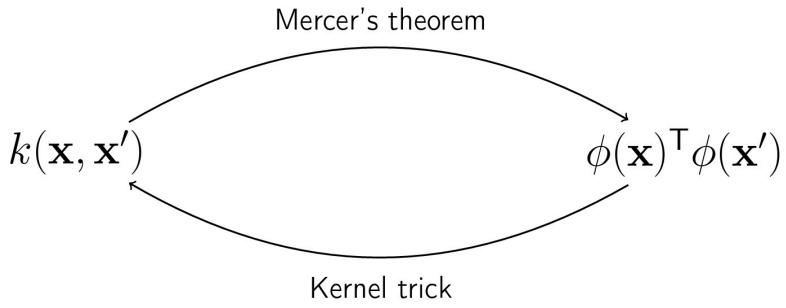
$f = w^T \cdot x$

$\alpha = (K + \lambda I)^{-1} y^T$

$O(n^2)$ $O(n^3)$

$f = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ $O(n)$

Summary



Kernels:

- Unknown (data in) feature space
- $D \gg n$
- Non-numeric data
- The entire dataset has to be stored

Feature maps: 不可能

- Feature map often infeasible to compute
- $n \gg D$