COMP0083: Kernel assignment: advanced topics in machine learning

January 24, 2025

Contents

Question 1

Question 1.1

From the given figure, the dataset consists of a red class enclosed by a ring of blue points. A simple feature space that allows error-free linear classification for this dataset is defined as:

$$\phi_1(x_1, x_2) = x_1^2 + x_2^2. \tag{1}$$

In this feature space, one can separate two classes by choosing a threshold value r_0 such that

- For $\phi_1(x_1, x_2) < r_0$, classify as red,
- For $\phi_1(x_1, x_2) \geq r_0$, classify as blue.

Question 1.2

We can apply eigendecomposition of inner product matrix K. Let us assume $\lambda_1, \cdots, \lambda_m$ are eigenvalues of K and $v_1, \cdots, v_m$ are corresponding eigenvectors. Then, we can decompose K as

$$K = Q\Lambda Q^{-1} = Q\Lambda Q^T, \tag{2}$$

where Λ is a diagonal matrix of eigenvalues λ_i and Q is a matrix of eigenvectors $[v_1, \cdots, v_m]$. As K is positive semidefinite, we can split the original diagonal matrix apart and get

$$K = Q\Lambda^{\frac{1}{2}} \left(\Lambda^{\frac{1}{2}}\right)^T Q^T = \Phi\Phi^T, \tag{3}$$

where we have $\Phi = Q\Lambda^{\frac{1}{2}}$.

In the question, the matrix K can be express as inner product of feature space $\phi(x_i)$. Thus, we can deduce

$$\begin{aligned} K_{ij} &= \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} = \left(\Phi\Phi^T\right)_{ij}, \\ \Rightarrow &\phi(x_i) = \Phi_i, \\ \Rightarrow &\phi(x_i) = [\sqrt{\lambda_1} v_{i1}, \cdots, \sqrt{\lambda_m} v_{im}]. \end{aligned} \tag{4}$$

where we find the feature space representation in terms of eigenvalue and eigenvector of matrix K.

Question 2

Question 2.1 Incomplete Cholesky for efficient COCO

Let us calculate the exact cost of COCO first. The solution of COCO is given as

$$\begin{bmatrix} 0 & \frac{1}{n}\widetilde{K}\widetilde{L} \\ \frac{1}{n}\widetilde{L}\widetilde{K} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \gamma \begin{bmatrix} \widetilde{K} & 0 \\ 0 & \widetilde{L} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{5}$$

Thus, the computation cost is from solving the eigenvalue problem in above solution. It involves matrix multiplications. To compute general $n \times n$ matrix multiplications, we have

```
input A and B, both n by n matrices
initialize C to be an n by n matrix of all zeros
for i from 1 to n:
    for j from 1 to n:
        for k from 1 to n:
            C[i][j] = C[i][j] + A[i][k]*B[k][j]
output C (as A*B)
```

This needs n^3 multiplications and $n^3 - n^2$ additions. So the overall cost is in the order of $O(n^3)$.

To approximate our solution via incomplete Cholesky, we can have two different methods. Since both methods work well, we are going to introduce both.

Method 1: QR decomposition on matrix

In this method, we directly go to the final step of the problem. To solve Equation 5, we first simplify the equation as

$$Av = \gamma Bv, \tag{6}$$

where

$$A = \begin{bmatrix} 0 & \frac{1}{n}\widetilde{K}\widetilde{L} \\ \frac{1}{n}\widetilde{L}\widetilde{K} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \widetilde{K} & 0 \\ 0 & \widetilde{L} \end{bmatrix}. \tag{7}$$

Now, we apply QR decomposition on matrix B

$$B = D^\top C^\top C D = D^\top D. \tag{8}$$

Substitute this back to our problem, we can further derive

$$\begin{aligned} Av &= \gamma D^\top Dv, \\ \Rightarrow & AD^{-1}V = \gamma D^\top V, \\ \Rightarrow & \left(D^\top\right)^{-1} AD^{-1}V = \gamma V. \end{aligned} \tag{9}$$

Here, we rescale the vector from v to $V = Dv$, transforming the original problem into an eigenvalue problem of lower dimensionality. The computational cost can be analysed as follows. Applying the Incomplete Cholesky decomposition to the matrix B yields a $m \times n$ matrix D, where m is smaller than n. Subsequently, the eigenvalue problem is solved for a reduced $m \times m$ matrix. This process involves two primary computational costs: (1) the Incomplete Cholesky decomposition, which has a complexity of $O(nm^2)$, and (2) solving the eigenvalue problem, with a cost of $O(m^3)$. Together, these steps significantly reduce the computational burden compared to solving the original problem directly.

Method 2: Feature map projection

Instead of decompose matrix B, we apply Incomplete Cholesky on both K and L.

$$\begin{aligned} K = X^\top X = (QR)^\top QR = R^\top R, \\ L = Y^\top Y = (PS)^\top PS = S^\top S, \end{aligned} \tag{10}$$

where we let $X = QR$ and $Y = PS$. As we make projections for both X and Y, we can modify the original problem into a new basis. In that case, we will have a new equation for this problem. First of all, we define $\theta = RH\alpha$ and $\eta = SH\beta$. These changes will largely simplify our equations. Then, we can start to derive our eigenvalue equations. We write Equation 5 as

$$\begin{aligned} -\frac{1}{n}\widetilde{K}\widetilde{L}\beta + \lambda\widetilde{K}\alpha = 0, \\ -\frac{1}{n}\widetilde{L}\widetilde{K}\alpha + \gamma\widetilde{L}\beta = 0. \end{aligned} \tag{11}$$

which is easier for our substitutions. We have the following derivation.

$$\begin{aligned} -\frac{1}{n}\left(HR^\top RH\right)\left(HS^\top SH\right)\beta + \lambda\left(HR^\top RH\right)\alpha = 0, \\ -\frac{1}{n}\left(HS^\top SH\right)\left(HR^\top RH\right)\alpha + \gamma\left(HS^\top SH\right)\beta = 0. \end{aligned} \tag{12}$$

Multiply the first equation by $\alpha^\top$, and the second by $\beta^\top$. The first one is then written as

$$\begin{aligned} &-\frac{1}{n}\left(\alpha^\top HR^\top\right)\left(RHS^\top\right)\left(SH\beta\right) + \lambda\left(\alpha^\top HR^\top\right)\left(RH\alpha\right) = 0, \\ \Rightarrow \quad &-\frac{1}{n}\theta^\top RHS^\top\eta + \lambda\theta^\top\theta = 0. \end{aligned} \tag{13}$$

Similarly, we get the second equation as

$$\begin{aligned} &-\frac{1}{n}\left(\beta^\top HS^\top\right)\left(SHR^\top\right)\left(RH\alpha\right) + \gamma\left(\beta^\top HS^\top\right)\left(SH\beta\right) = 0, \\ &\Rightarrow -\frac{1}{n}\eta^\top SHR^\top\theta + \gamma\eta^\top\eta = 0. \end{aligned} \tag{14}$$

The solution can be expressed as

$$\begin{bmatrix} 0 & \frac{1}{n}RHS^\top \\ \frac{1}{n}SHR^\top & 0 \end{bmatrix}\begin{bmatrix} \theta \\ \eta \end{bmatrix} = \gamma\begin{bmatrix} \theta \\ \eta \end{bmatrix}. \tag{15}$$

The witness functions f and g are

$$\begin{aligned} f(x) &= \langle f, X\rangle = f^\top X = \theta^T R, \\ g &= \langle g, Y\rangle = g^\top Y = \eta^T S. \end{aligned} \tag{16}$$

In this case, we reduce the dimension of the problem again. We get the same order of computational cost as method 1. In summary, we reduce the computational cost by using Incomplete Cholesky from $O(n^3)$ to $O(nm^2)$.

Implement the incomplete Cholesky-based COCO

Here is the code for Gaussian kernel.

```
def gaussian_kernel(x, sigma):
    n = len(x)
    K = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            K[i, j] = np.exp(-((x[i] - x[j]) ** 2).sum() / sigma**2)
    return K
```

Gaussian Kernel

We follow the code provided in extra readings.

```
def incomplete_cholesky(K, eta):
    dim = K.shape[0]
    R = np.zeros((dim,dim))
    d = np.diag(K).copy()
    I = []
    j = 0
    while True:
        a = np.max(d)
        if a <= eta:
            break
        pivot_index = np.argmax(d)
        I.append(pivot_index)
        nu = np.sqrt(a)
        for i in range(dim):
            R[j, i] = (K[I[j], i] - R[:j, i] @ R[:j, I[j]]) / nu
            d[i] -= R[j, i] ** 2
        j += 1
    R = R[:j, :]
    return R
```

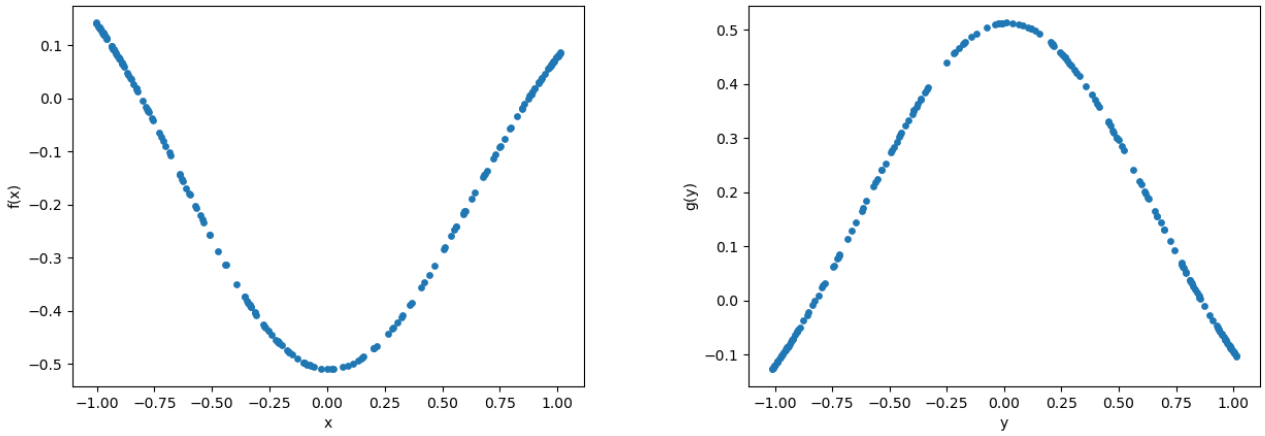
Incomplete cholesky

Here is code for first method. We apply incomplete cholesky on matrix B.

```
def coco_method1():
    # generate the data, with N samples
    N = 200
    t = np.random.uniform(0, 2 * np.pi, N)
    n1 = np.random.normal(0, 0.01, N)
    n2 = np.random.normal(0, 0.01, N)
    x = np.sin(t) + n1
    y = np.cos(t) + n2

    # define the parameters for gaussian kernel and incomplete cholesky
    sigma = 1
    eta = 1e-6

    # define K and L
    K = gaussian_kernel(x, sigma)
    L = gaussian_kernel(y, sigma)
    H = np.eye(N) - np.ones((N, N)) / N
    K_tilde = H @ K @ H
    L_tilde = H @ L @ H
    # define zero blocks
    z = np.zeros((N, N))
    # define matrix for eigen-problem
    # A v = gamma * B v
    A = np.block([[z, K_tilde @ L_tilde / N],
                  [L_tilde @ K_tilde / N, z]])
    B = np.block([[K_tilde, z],
                  [z, L_tilde]])

    # incomplete Cholesky solution
    # apply QR decomputation on B -> R^T R
    R = incomplete_cholesky(B, eta)
    # A v = gamma * R.T * R v, let u = Rv -> v = inv(R)u
    # A inv(R)u = gamma * R.T * u
    # inv(R.T) A inv(R) u = gamma u
    C = np.linalg.pinv(R.T) @ A @ np.linalg.pinv(R)
    eigvals, eigvecs = np.linalg.eig(C)
    coco = np.abs(np.real(eigvals[0]))
    eigvecs_v = np.linalg.pinv(R) @ eigvecs[:, 0]
    alpha = eigvecs_v[:N]
    beta = eigvecs_v[N:]
    # witness functions
    fx = K @ H @ alpha
    gx = L @ H @ beta
    # Corrections
    correlation = np.corrcoef(fx, gx)
```

Method 1: QR decomposition on matrix

Here are the plots via method 1.



(a) f(x) vs x

(b) g(y) vs y

Figure 1: COCO: Plot of f and g under method 1

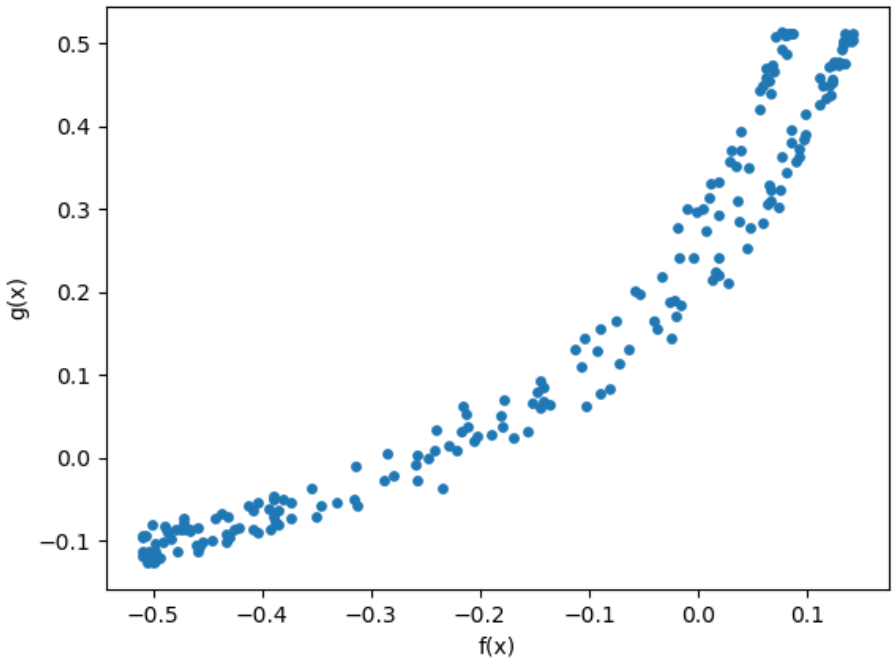


Figure 2: COCO: Plot of f vs g under method 1

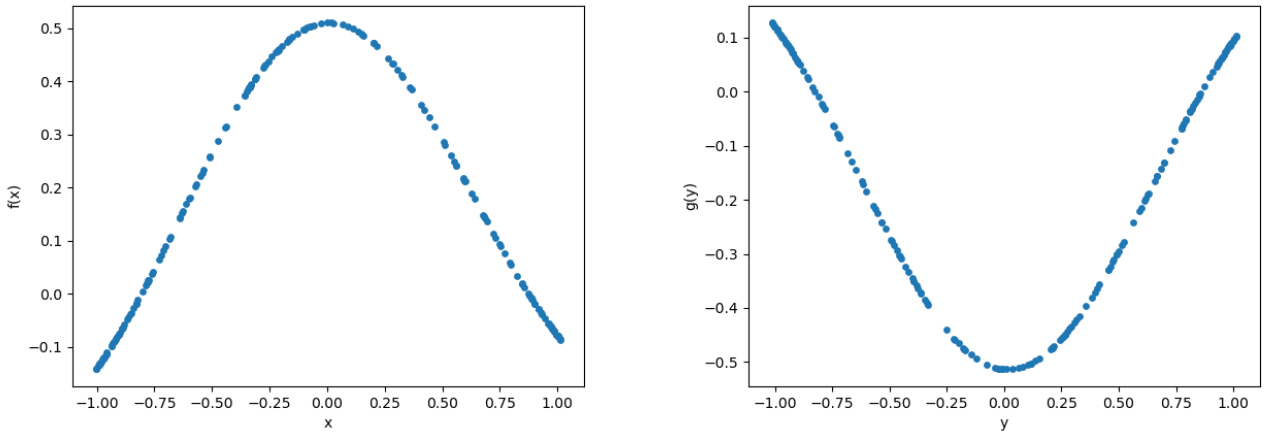
Here is code for method 2. To avoid repeated code, we only include the changed part.

```
def coco_method2():
    # same code as method 1
    # incomplete cholesky on both K and L
    R = incomplete_cholesky(K, eta)
    S = incomplete_cholesky(L, eta)
    # define zero blocks
    m_R = R.shape[0]
    m_S = S.shape[0]
    z_R = np.zeros((m_R, m_R))
    z_S = np.zeros((m_S, m_S))
    # define matrix for eigen-problem
    A = np.block([[z_R, R @ H @ S.T/ N],
                  [S @ H @ R.T / N, z_S]])

    eigvals, eigvecs = np.linalg.eig(A)
```

```
    N_max = np.argmax(eigvals)
    coco = np.abs(np.real(eigvals[N_max]))

    eigvecs_v = eigvecs[:, N_max]
    theta = eigvecs_v[:m_R]
    eta = eigvecs_v[m_R:]

    fx = theta.T @ R
    gx = eta.T @ S
    correlation = np.corrcoef(fx, gx)
```

Method 2: Feature map projection

Here are plots from method 2.



(a) f(x) vs x

(b) g(y) vs y

Figure 3: COCO: Plot of f and g under method 2

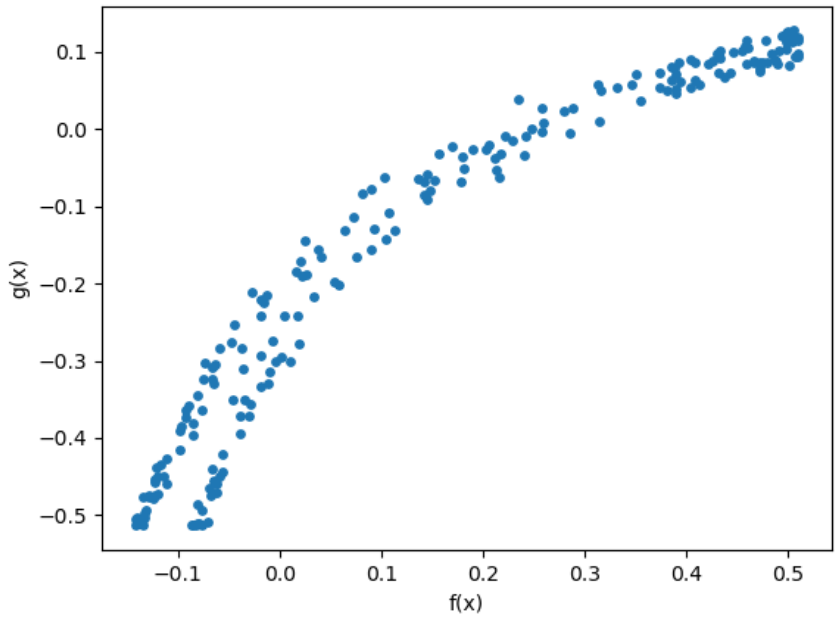


Figure 4: COCO: Plot of f vs g under method 2

In summary, we get same results from both method. The COCO value is 0.0949 and the correlation is 0.9503.

Question 2.2

We use the same method in the lecture notes. To maximize the canonical correlation with two constraints, we can write the corresponding Lagrangian:

$$\mathcal{L}(f, g, \lambda, \gamma) = -f^\top \widehat{C}_{XY} g + \frac{\lambda}{2}\left(f^\top \widehat{C}_{XX} f - 1\right) + \frac{\gamma}{2}\left(g^\top \widehat{C}_{YY} g - 1\right). \tag{17}$$

Let us write this in terms of the Gram matrices $\widetilde{K}$ and $\widetilde{L}$:

$$\begin{aligned} f^\top \widehat{C}_{XY} g &= \frac{1}{n}\alpha^\top H X^\top \left(XHY^\top\right) YH\beta = \frac{1}{n}\alpha^\top \widetilde{K}\widetilde{L}\beta, \\ f^\top \widehat{C}_{XX} f &= \frac{1}{n}\alpha^\top H X^\top \left(XHX^\top\right) XH\alpha = \frac{1}{n}\alpha^\top \widetilde{K}^2\alpha, \\ g^\top \widehat{C}_{YY} g &= \frac{1}{n}\beta^\top H Y^\top \left(YHY^\top\right) YH\beta = \frac{1}{n}\beta^\top \widetilde{L}^2\beta. \end{aligned} \tag{18}$$

Now, the Lagrangian is simplified as

$$\mathcal{L}(f, g, \lambda, \gamma) = -\frac{1}{n}\alpha^\top \widetilde{K}\widetilde{L}\beta + \frac{\lambda}{2}\left(\frac{1}{n}\alpha^\top \widetilde{K}^2\alpha - 1\right) + \frac{\gamma}{2}\left(\frac{1}{n}\beta^\top \widetilde{L}^2\beta - 1\right). \tag{19}$$

We must minimise the above Lagrangian with respect to α, β. Differentiating with respect to α and β and setting the resulting expressions to zero, we get

$$\begin{aligned} -\frac{1}{n}\widetilde{K}\widetilde{L}\beta + \frac{\lambda}{n}\widetilde{K}^2\alpha = 0, \\ -\frac{1}{n}\widetilde{L}\widetilde{K}\alpha + \frac{\gamma}{n}\widetilde{L}^2\beta = 0. \end{aligned} \tag{20}$$

Multiply the first equation by $\alpha^\top$, and the second by $\beta^\top$,

$$\begin{aligned} \alpha^\top \widetilde{K}\widetilde{L}\beta = \lambda\alpha^\top \widetilde{K}^2\alpha \\ \beta^\top \widetilde{L}\widetilde{K}\alpha = \gamma\beta^\top \widetilde{L}^2\beta \end{aligned} \tag{21}$$

This shows that we need to set $\lambda = \gamma$. Then, we need to solve following expression:

$$\begin{bmatrix} 0 & \widetilde{K}\widetilde{L} \\ \widetilde{L}\widetilde{K} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \gamma \begin{bmatrix} \widetilde{K}^2 & 0 \\ 0 & \widetilde{L}^2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{22}$$

When γ is at maximum, we obtain the solution for this eigenvalue problem.

Assume a Gaussian kernel, and that the points are also non-pathologically distributed so that K and L have full rank. This means that both $\widetilde{K}$ and $\widetilde{L}$ are invertible. Using $\lambda = \gamma$ in Equations 20, we multiply the first equation by $\widetilde{K}^\top$ and the second by $\widetilde{L}^\top$

$$\begin{aligned} \widetilde{L}\beta - \gamma\widetilde{K}\alpha = 0, \\ \widetilde{K}\alpha - \gamma\widetilde{L}\beta = 0. \end{aligned} \tag{23}$$

Then, we combine above two equations. we can multiply the second equation in Equation 23 by γ

$$\gamma\widetilde{K}\alpha - \gamma^2\widetilde{L}\beta = 0, \tag{24}$$

as $\gamma\widetilde{K}\alpha = \widetilde{L}\beta$, we can further deduce

$$\begin{aligned} &\widetilde{L}\beta - \gamma^2\widetilde{L}\beta = 0, \\ \Rightarrow &\left(1 - \gamma^2\right)\widetilde{L}\beta = 0, \\ \Rightarrow &1 - \gamma^2 = 0, \\ \Rightarrow &\gamma = \pm 1. \end{aligned} \tag{25}$$

Thus, we have a trivial solution for γ for any choice of α, β.

To solve this problem, one can add a regularization term into the constrains. The new constrains can be shown as:

$$\begin{aligned} \left\langle f, \widehat{C}_{XX} f \right\rangle_{\mathcal{F}} + \kappa\|f\|_{\mathcal{F}} = 1, \\ \left\langle g, \widehat{C}_{YY} g \right\rangle_{\mathcal{G}} + \kappa\|g\|_{\mathcal{G}} = 1. \end{aligned} \tag{26}$$

Combining the information from the question, we can write the new Lagrangian as

$$\mathcal{L}(f, g, \lambda, \gamma) = -\frac{1}{n}\alpha^\top \widetilde{K}\widetilde{L}\beta + \frac{\lambda}{2}\left(\alpha^\top \widetilde{K}^2\alpha + \kappa\alpha^\top \widetilde{K}\alpha - 1\right) + \frac{\gamma}{2}\left(\beta^\top \widetilde{L}^2\beta + \kappa\beta^\top \widetilde{L}\beta - 1\right), \tag{27}$$

where we remove $\frac{1}{n}$ factor in last two terms via rescaling. The following step is the same as above. Differentiating with respect to α and β and setting the resulting expressions to zero, we get

$$\begin{aligned} -\frac{1}{n}\widetilde{K}\widetilde{L}\beta + \lambda\left(\widetilde{K}^2\alpha + \kappa\widetilde{K}\alpha\right) &= 0, \\ -\frac{1}{n}\widetilde{L}\widetilde{K}\alpha + \gamma\left(\widetilde{L}^2\beta + \kappa\widetilde{L}\beta\right) &= 0. \end{aligned} \tag{28}$$

We multiply the first equation by $\alpha^\top$, and the second by $\beta^\top$,

$$\begin{aligned} \alpha^\top\frac{1}{n}\widetilde{K}\widetilde{L}\beta &= \lambda\alpha^\top\left(\widetilde{K}^2 + \kappa\widetilde{K}\right)\alpha, \\ \beta^\top\frac{1}{n}\widetilde{L}\widetilde{K}\alpha &= \gamma\beta^\top\left(\widetilde{L}^2 + \kappa\widetilde{L}\right)\beta. \end{aligned} \tag{29}$$

Then, we derive the form asked in the question

$$\begin{bmatrix} 0 & \frac{1}{n}\widetilde{K}\widetilde{L} \\ \frac{1}{n}\widetilde{L}\widetilde{K} & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda\begin{bmatrix} \widetilde{K}^2 + \kappa\widetilde{K} & 0 \\ 0 & \widetilde{L}^2 + \kappa\widetilde{L} \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{30}$$

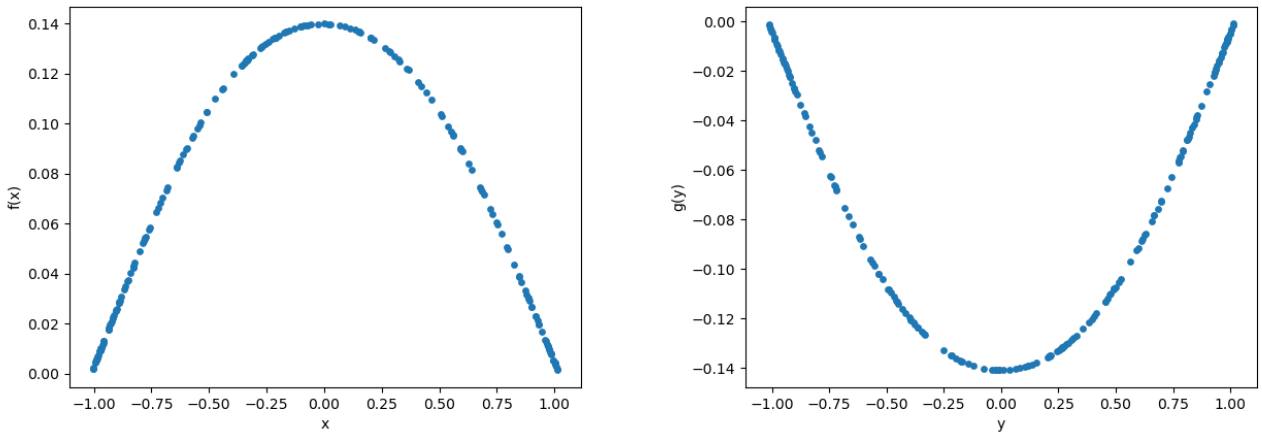
Implement the kernel CCA

In this section, we modify the previous code to incorporate the extra regularization term. Since the majority of the code remains unchanged, we will focus only on the modifications and explain the adjusted parts. For consistency with COCO, we will use two methods to implement the solution.

Let us begin with Method 1. This approach is straightforward to modify, as the changes are limited to the elements of the matrices. The rest of the code remains unchanged.

```
def cca():
    # only difference part are shown
    A = np.block([[z, K_tilde @ L_tilde / N],
                  [L_tilde @ K_tilde / N, z]])
    B = np.block([[K_tilde @ K_tilde + kapa * K_tilde, z],
                  [z, L_tilde @ L_tilde + kapa * L_tilde]])
```

CCA Method 1



(a) f(x) vs x

(b) g(y) vs y

Figure 5: CCA: Plot of f and g under method 1

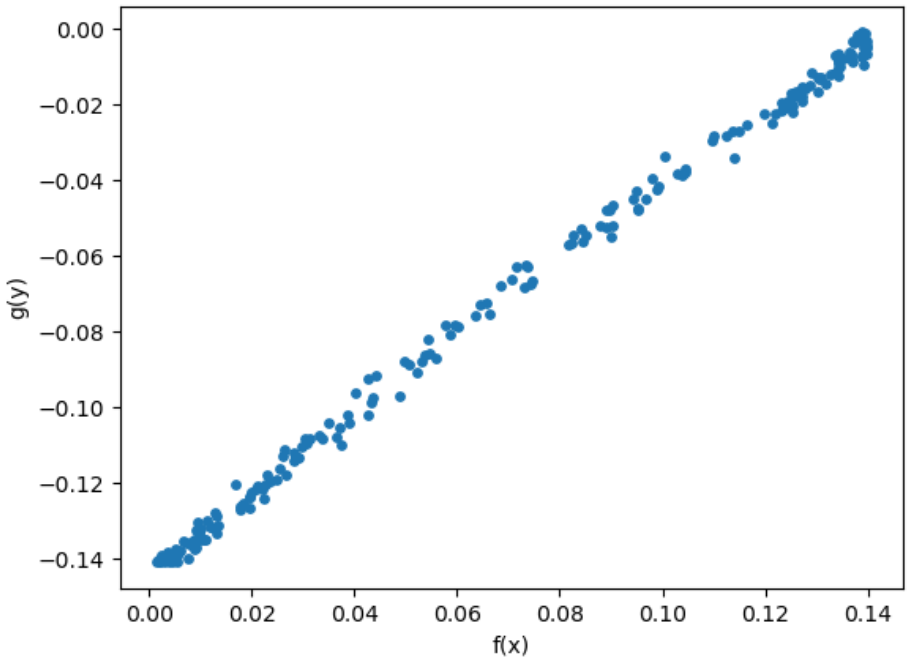


Figure 6: CCA: Plot of f vs g under method 1

For method 2, we need to rewrite our solution in new basis. We can start from Equation 29 and write it in terms of θ and η:

$$\begin{aligned} \frac{1}{n}\theta^\top RHS^\top \eta &= \lambda\theta^\top \left(RHR^\top + \kappa\right)\theta, \\ \frac{1}{n}\eta^\top SHR^\top \theta &= \gamma\eta^\top \left(SHS^\top + \kappa\right)\eta. \end{aligned} \tag{31}$$

Then, the solution is

$$\begin{bmatrix} 0 & \frac{1}{n}RHS^\top \\ \frac{1}{n}SHR^\top & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \eta \end{bmatrix} = \gamma \begin{bmatrix} RHR^\top + \kappa & 0 \\ 0 & SHS^\top + \kappa \end{bmatrix} \begin{bmatrix} \theta \\ \eta \end{bmatrix}. \tag{32}$$

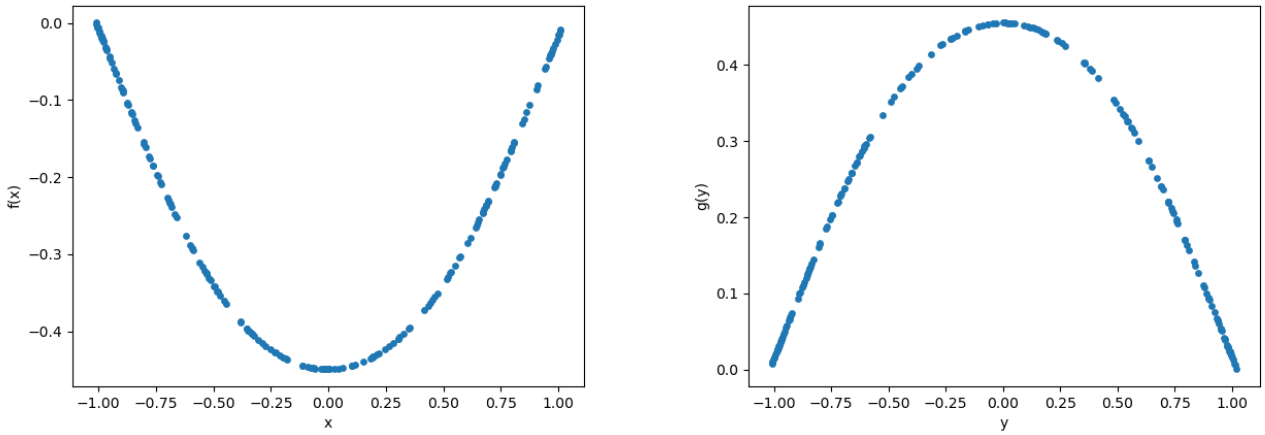
The corresponding code is

```
def cca_method2():
    # only different part shown here
    A = np.block([[z_R, R @ H @ S.T/ N],
                  [S @ H @ R.T / N, z_S]])

    B = np.block([[R @ H @ R.T + kapa*np.eye(m_R), z_RS],
                  [z_SR, S @ H @ S.T + kapa*np.eye(m_S)]])

    C = np.linalg.inv(B) @ A
    eigvals, eigvecs = np.linalg.eig(C)
```

CCA Method 2

Here are the plots.

The COCO value is 0.9876 and the correlation is 0.9980. Comparing with previous plots and value of correlations, we find that witness functions $f(x)$ and $g(y)$ derived from KCCA are smoother. Also, we have a higher correlation than COCO. This improvement comes from the regularization terms which makes our code more stable.



(a) f(x) vs x

(b) g(y) vs y

Figure 7: CCA: Plot of f and g under method 2

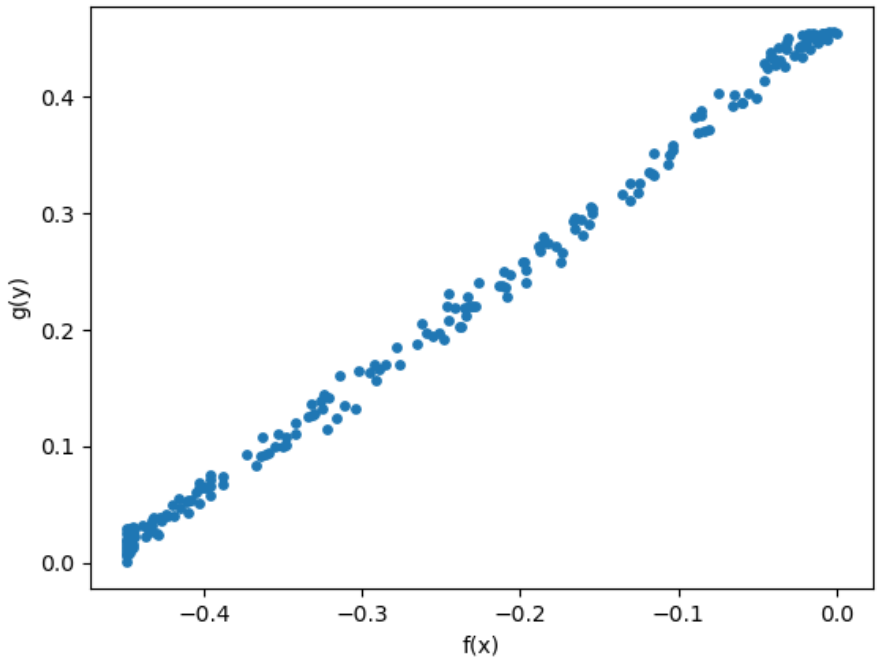


Figure 8: CCA: Plot of f vs g under method 2