COMP0083 Coursework 1

Adam Peace — 17008708

November 24, 2020

Abstract

My submission for part 1 of the COMP0083 coursework 1 (Questions 1 & 2).

1 Feature Spaces

1.1 Describe a simple feature space

The datasets in Figure 1 are not linearly seperable in $\mathbb{R}^2$. However, we can create a feature from the locus of each datapoint that will allow them to be linearly seperable.

$$\phi(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_1^2 + x_2^2 \end{bmatrix}^T$$

1.2 Eigendecomposition of K

Since $\mathbf{K}$ is psd and symmetric, its eigendecomposition is as follows:

$$\mathbf{K} = \mathbf{Q}\mathbf{\Delta}\mathbf{Q}', \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix}$$

$$\mathbf{\Delta} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix}$$

$$\begin{aligned} \mathbf{K} &= \mathbf{Q}\mathbf{\Delta}\mathbf{Q}' = \mathbf{Q}\sqrt{\mathbf{\Delta}}\sqrt{\mathbf{\Delta}}'\mathbf{Q} \\ &= \mathbf{Q}\sqrt{\mathbf{\Delta}}(\mathbf{Q}\sqrt{\mathbf{\Delta}})' \\ &= \mathbf{L}\mathbf{L}', \mathbf{L} = \begin{bmatrix} \sqrt{\lambda_1}\mathbf{u}_1 & \cdots & \sqrt{\lambda_m}\mathbf{u}_m \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \kappa_{i,j} &= \langle \Phi(x_i)\Phi(x_j) \rangle_{\mathcal{H}} = (\mathbf{L}\mathbf{L}')_{i,j} \\ &= \begin{bmatrix} \sqrt{\lambda_1}u_1^{(i)} & \cdots & \sqrt{\lambda_m}u_m^{(i)} \end{bmatrix} \begin{bmatrix} \sqrt{\lambda_1}u_1^{(j)} & \cdots & \sqrt{\lambda_m}u_m^{(j)} \end{bmatrix}' \\ &\implies \Phi(x_i) = \begin{bmatrix} \sqrt{\lambda_1}u_1^{(i)} & \cdots & \sqrt{\lambda_m}u_m^{(i)} \end{bmatrix} \end{aligned}$$

2 Kernel Dependence Detection

2.1 Incomplete Cholesky for Efficient COCO

To compare the computational time complexity of exact COCO to efficient COCO, we can determine the time complexity of both algorithms assuming we already have $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{L}}$ (which themselves cost $O(4n^3)$ to compute from $\boldsymbol{K}$, $\boldsymbol{L}$ & $\boldsymbol{H}$).

2.1.1 Exact COCO Time Complexity

To produce $\boldsymbol{A}$ and $\boldsymbol{B}$ block matrices from $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{L}}$ we have 2 sets of matrix multiplications $\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{K}}$ as well as occupying the $\boldsymbol{B}$ block matrix with 2 $n \times n$ matrices:

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

The cost of producing the solution to the generalized eigenvalue problem with two $2n \times 2n$ matrices A and B is:

$$O(8n^3)$$

Taking the maximum value from the eigenvalues gives us COCO, hence our total complexity is:

$$O(10n^3 + 2n^2)$$

2.1.2 Efficient COCO Time Complexity

For efficient COCO, we also have to compute the A and B block matrices with complexity:

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

Then we have to produce the solution R of the incomplete Cholesky algorithm with input matrix B. This costs us t loops of n operations, hence we have:

$$O(tn)$$

We then have the complexity of applying R to the A block matrix which is one $t \times 2n$ by $2n \times 2n$ matrix multiply and one $t \times 2n$ by $2n \times t$ matrix multiply. Since a $m \times n$ by $n \times p$ matrix multiply is complexity $O(mnp)$, we have:

$$O(4n^2t + 2t^2n)$$

Finally, we have to perform eigenvalue decomposition of the generalized eigenvalue problem with an A matrix and a B identity matrix of size $t \times t$, which costs:

$$O(t^3)$$

Once again, the maximum eigenvalue gives us COCO so overall, we undergo a total computational complexity for the efficient COCO of:

$$O(2n^3 + t^3 + (2 + 4t)n^2 + 2t^2n)$$

which is significantly cheaper than the $O(10n^3 + 2n^2)$ computational complexity we receive from the exact COCO computation. This observation also relies on the fact that for a high enough η, the value of t is often less than 10.

2.1.3 Implementation of Incomplete Cholesky-based COCO in Python

See Appendix A.

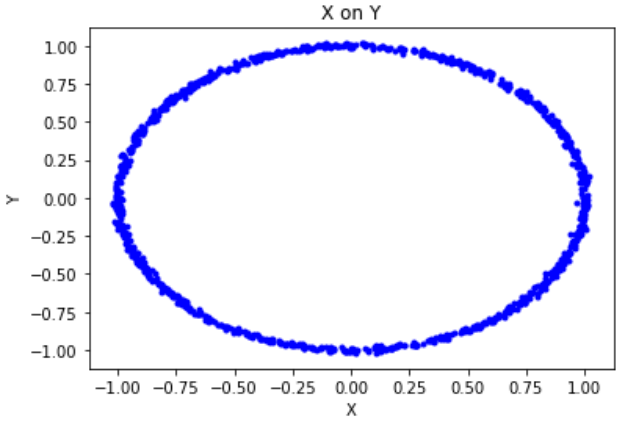

Figure 1: Visualization of the dataset.

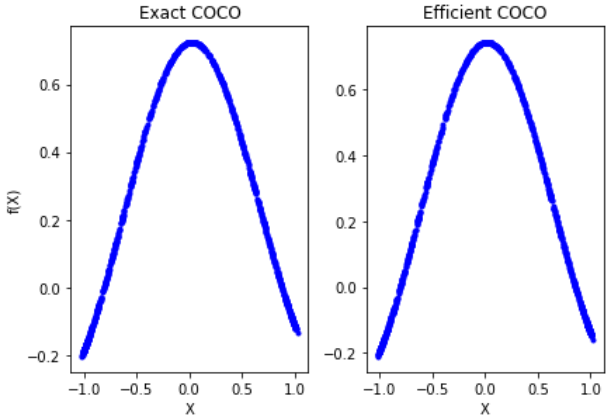

Figure 2: Plot of f with a Gaussian kernel.

2.1.4 Incomplete Cholesky-based COCO results

With $N = 1000$, the data is shown in Figure 1 and then the comparison between exact and efficient COCO shown in Figures 2, 3 & 4. The exact value for COCO was 0.0925856 where efficient COCO gave 0.0925857 with $\eta = 1 \times 10^{-}4$. It is clear that the efficient COCO has done a very good job of emulating the exact result, with some slight variation between the correlation of f(X) and g(Y) in Figure 4. Using the Pearson correlation coefficient between the resulting f and g resulted in 0.9344 with this data.

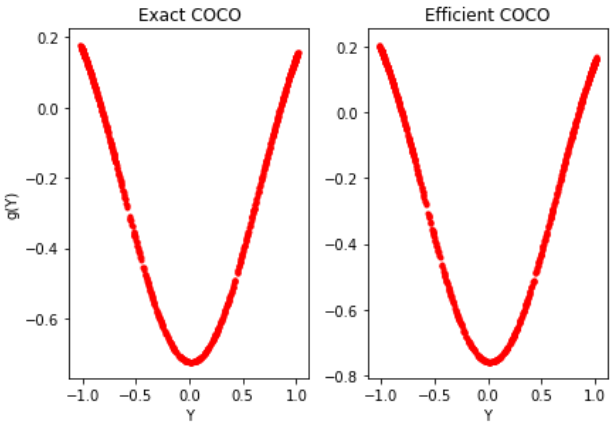

Figure 3: Plot of g with a Gaussian kernel.

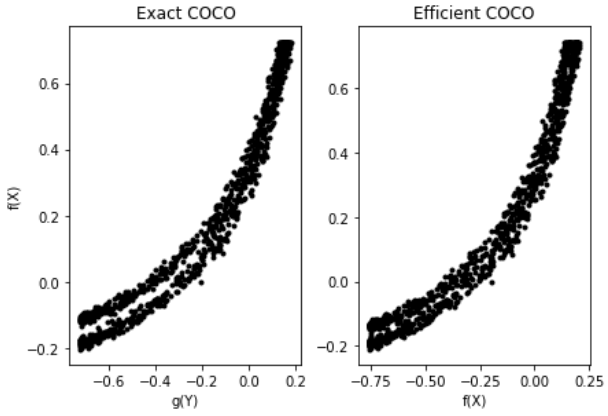

Figure 4: Plot of (x, y) via f and g. (Please excuse the typo on the efficient COCO chart. The axis label should be $g(Y)$.)

2.2 Kernel CCA

First let us define our Lagrangian based on the definitions and constraints in the assignment:

$$\mathcal{L}(f, g, \lambda, \gamma) = -\langle f, \boldsymbol{C_{XY}} g\rangle + \frac{\lambda}{2}(\langle f, \boldsymbol{C_{XX}} f\rangle - 1) + \frac{\gamma}{2}(\langle g, \boldsymbol{C_{YY}} g\rangle - 1)$$

As in the lecture notes, I have divided the Lagrangian multipliers by 2 and negated the covariance term.

$$\begin{aligned}
\langle f, \boldsymbol{C_{XY}} g\rangle &= (\mathbf{XH}\boldsymbol{\alpha})^{\mathrm{T}} \frac{1}{n}(\boldsymbol{XHY}^T)\boldsymbol{YH\beta} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{X}^T(\boldsymbol{XHY}^T)\boldsymbol{YH\beta} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\boldsymbol{HX}^T\boldsymbol{XHHY}^T\boldsymbol{YH\beta} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\boldsymbol{HKHHLH\beta} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta}
\end{aligned}$$

$$\begin{aligned}
\langle f, \boldsymbol{C_{XX}} f\rangle &= (\mathbf{XH}\boldsymbol{\alpha})^{\mathrm{T}} \frac{1}{n}(\boldsymbol{XHX}^T)\boldsymbol{XH\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\boldsymbol{H}^T\boldsymbol{X}^T(\boldsymbol{XHX}^T)\boldsymbol{XH\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\boldsymbol{H}(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{HH}(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{H\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T(\boldsymbol{HKH})(\boldsymbol{HKH})\boldsymbol{\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha}
\end{aligned}$$

$$\begin{aligned}
\langle g, \boldsymbol{C_{YY}} g\rangle &= (\mathbf{YH}\boldsymbol{\beta})^{\mathrm{T}} \frac{1}{n}(\boldsymbol{YHY}^T)\boldsymbol{YH\beta} \\
&= \frac{1}{n}\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{Y}^T(\boldsymbol{YHY}^T)\boldsymbol{YH\beta} \\
&= \frac{1}{n}\boldsymbol{\beta}^T(\boldsymbol{HY}^T\boldsymbol{YH})(\boldsymbol{HY}^T\boldsymbol{YH})\boldsymbol{\beta} \\
&= \frac{1}{n}\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta}
\end{aligned}$$

$$\mathcal{L}(f, g, \lambda, \gamma) = -\frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{\lambda}{2}(\frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} - 1) + \frac{\gamma}{2}(\frac{1}{n}\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} - 1)$$

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}} &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{1}{n}\frac{\lambda}{2}(\tilde{\boldsymbol{K}}^2 + (\tilde{\boldsymbol{K}}^2)^T)\boldsymbol{\alpha} \\
&= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{1}{n}\lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} + \frac{1}{n}\frac{\gamma}{2}(\tilde{\boldsymbol{L}}^2 + (\tilde{\boldsymbol{L}}^2)^T)\boldsymbol{\beta} \\
&= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} + \frac{1}{n}\gamma\tilde{\boldsymbol{L}}^2\boldsymbol{\beta}
\end{aligned}$$

Now set $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}}$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}}$ to 0:

$$\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \frac{1}{n}\lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} \tag{1}$$

$$\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \frac{1}{n}\gamma\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} \tag{2}$$

Now premultiply 1 by $\boldsymbol{\alpha}^T$ and 2 by $\boldsymbol{\beta}^T$:

$$\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \lambda\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} \tag{3}$$

$$\boldsymbol{\beta}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \gamma\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} \tag{4}$$

Subtracting 3 from 4 gives us:

$$\gamma\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} = \lambda\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha}$$

Now we know that if $\lambda \neq 0$ and $\gamma \neq 0$, then $\lambda = \gamma$. Apply this to 1 and 2:

$$\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} \tag{5}$$

$$\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \lambda\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} \tag{6}$$

Now we can formulate a generalized eigenvalue problem using block matrix form 5 and 6:

$$\begin{bmatrix} \mathbf{0} & \tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}} \\ \tilde{\boldsymbol{L}}\tilde{\boldsymbol{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} = \lambda \begin{bmatrix} \tilde{\boldsymbol{K}}^2 & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{L}}^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix} \tag{7}$$

Which can easily be solved in SciPy or Matlab.

What went wrong? Well, since we use a Guassian kernel, the centering of $\boldsymbol{K}$ and $\boldsymbol{L}$ means they are projected to identical column spaces with 0 orthogonal components, meaning they will always give a correlation score of 1. If we use the same method with a regularizer, this will no longer be an issue.

So let us add a regularizing term such that:

$$\text{var}(f(x)) = \langle f, \boldsymbol{C}_{\boldsymbol{XX}} f\rangle + \frac{1}{n}\varkappa||f||^2$$

$$\text{var}(g(y)) = \langle g, \boldsymbol{C}_{\boldsymbol{YY}} g\rangle + \frac{1}{n}\varkappa||g||^2$$

Now are Lagrangian is:

$$\mathcal{L}(f, g, \lambda, \gamma) = -\langle f, \boldsymbol{C}_{\boldsymbol{XY}} g\rangle + \frac{\lambda}{2}(\langle f, \boldsymbol{C}_{\boldsymbol{XX}} f\rangle + \frac{1}{n}\varkappa||f||^2 - 1) + \frac{\gamma}{2}(\langle g, \boldsymbol{C}_{\boldsymbol{YY}} g\rangle + \frac{1}{n}\varkappa||g||^2 - 1)$$

Now, we have the same Lagrangian function as before:

$$\langle f, \boldsymbol{C}_{\boldsymbol{XY}} g\rangle = \frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta}$$

and the constraints are now:

$$\begin{aligned}
\langle f, \boldsymbol{C}_{\boldsymbol{XX}} f\rangle + \frac{1}{n}\varkappa||f||^2 &= (\mathbf{X}\mathbf{H}\boldsymbol{\alpha})^{\mathrm{T}}\frac{1}{n}(\boldsymbol{X}\boldsymbol{H}\boldsymbol{X}^T)\boldsymbol{X}\boldsymbol{H}\boldsymbol{\alpha} + \frac{1}{n}\varkappa(\mathbf{X}\mathbf{H}\boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{X}\boldsymbol{H}\boldsymbol{\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\varkappa\boldsymbol{\alpha}^T(\boldsymbol{H}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{H})\boldsymbol{\alpha} \\
&= \frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\varkappa\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\boldsymbol{\alpha}
\end{aligned}$$

$$\begin{aligned}\langle g, \boldsymbol{C_{YY}} g\rangle + \varkappa||g||^2 &= (\mathbf{YH}\boldsymbol{\beta})^{\mathrm{T}}\frac{1}{n}(\boldsymbol{YHY}^T)\boldsymbol{YH}\boldsymbol{\beta} + \frac{1}{n}\varkappa(\mathbf{YH}\boldsymbol{\beta})^{\mathrm{T}}\boldsymbol{YH}\boldsymbol{\beta} \\ &= \frac{1}{n}\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\varkappa\boldsymbol{\beta}^T(\boldsymbol{HY}^T\boldsymbol{YH})\boldsymbol{\beta} \\ &= \frac{1}{n}\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\varkappa\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}\boldsymbol{\beta}\end{aligned}$$

So now when we substitute back into the Lagrangian equation we get:

$$\mathcal{L}(f,g,\lambda,\gamma) = -\frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{\lambda}{2}((\mathbf{XH}\boldsymbol{\alpha})^{\mathrm{T}}\frac{1}{n}\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\varkappa\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\boldsymbol{\alpha} - 1) + \frac{\gamma}{2}(\frac{1}{n}\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\boldsymbol{\beta}^T\varkappa\tilde{\boldsymbol{L}}\boldsymbol{\beta} - 1)$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}} &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{1}{n}\lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\frac{\lambda}{2}\varkappa(\tilde{\boldsymbol{K}} + \tilde{\boldsymbol{K}}^T)\boldsymbol{\alpha} \\ &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} + \frac{1}{n}\lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\lambda\varkappa\tilde{\boldsymbol{K}}\boldsymbol{\alpha}\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} + \frac{1}{n}\gamma\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\frac{\gamma}{2}\varkappa(\tilde{\boldsymbol{L}} + \tilde{\boldsymbol{L}}^T)\boldsymbol{\beta} \\ &= -\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} + \frac{1}{n}\gamma\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\gamma\varkappa\tilde{\boldsymbol{L}}\boldsymbol{\beta}\end{aligned}$$

Now set $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}}$ and $\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}}$ to 0:

$$\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \frac{1}{n}\lambda\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \frac{1}{n}\lambda\varkappa\tilde{\boldsymbol{K}}\boldsymbol{\alpha} \tag{8}$$

$$\frac{1}{n}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \frac{1}{n}\gamma\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \frac{1}{n}\gamma\varkappa\tilde{\boldsymbol{L}}\boldsymbol{\beta} \tag{9}$$

Now premultiply 8 by $\boldsymbol{\alpha}^T$ and 9 by $\boldsymbol{\beta}^T$:

$$\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \lambda\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \lambda\varkappa\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\boldsymbol{\alpha} \tag{10}$$

$$\boldsymbol{\beta}^T\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \gamma\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \gamma\varkappa\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}\boldsymbol{\beta} \tag{11}$$

Subtracting 10 from 11 gives us:

$$\gamma(\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \varkappa\boldsymbol{\beta}^T\tilde{\boldsymbol{L}}\boldsymbol{\beta}) = \lambda(\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \varkappa\boldsymbol{\alpha}^T\tilde{\boldsymbol{K}}\boldsymbol{\alpha})$$

Now we know that if $\lambda \neq 0$ and $\gamma \neq 0$, then $\lambda = \gamma$. Apply this to 9 and 9:

$$\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\beta} = \lambda(\tilde{\boldsymbol{K}}^2\boldsymbol{\alpha} + \varkappa\tilde{\boldsymbol{K}}\boldsymbol{\alpha}) \tag{12}$$

$$\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}\boldsymbol{\alpha} = \lambda(\tilde{\boldsymbol{L}}^2\boldsymbol{\beta} + \varkappa\tilde{\boldsymbol{K}}\boldsymbol{\beta}) \tag{13}$$

Now we can formulate a generalized eigenvalue problem using block matrix form of 12 and 13:

$$\begin{bmatrix}\mathbf{0} & \tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}} \\ \tilde{\boldsymbol{L}}\tilde{\boldsymbol{K}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{bmatrix} = \lambda\begin{bmatrix}\tilde{\boldsymbol{K}}^2 + \varkappa\tilde{\boldsymbol{K}} & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{L}}^2 + \varkappa\tilde{\boldsymbol{L}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{bmatrix} \tag{14}$$

Which doesn't have the aforementioned issues found in 7.

2.2.1 Implementation of Kernel CCA in Python

The implementation of Kernel CCA in Python can be found in Appendix B. The results are in Figures 5, 6 & 7, overlayed on the output from COCO. Note that the CCA functions have a larger range than those from COCO due to the different constraints on each process.

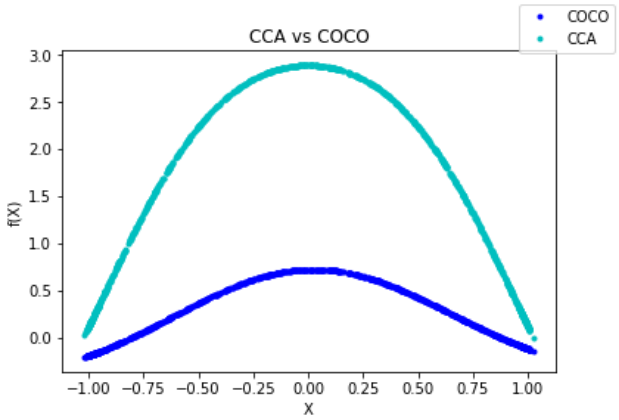

Figure 5: Plot of f with a Gaussian kernel for Kernel CCA and COCO.

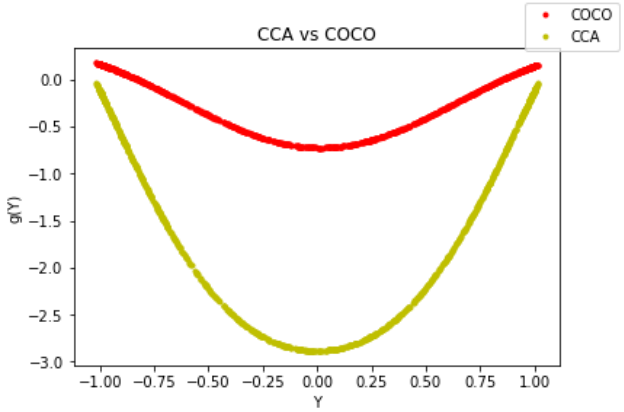

Figure 6: Plot of g with a Gaussian kernel for Kernel CCA and COCO.

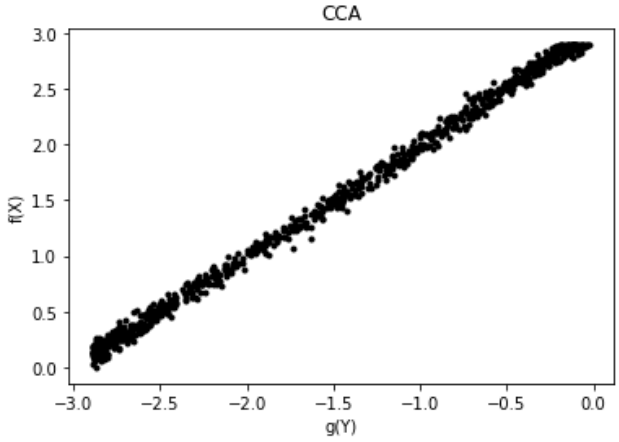

Figure 7: Plot of (x, y) via f and g for Kernel CCA.

Appendix A Python Efficient COCO

```
def get_gaussian_kernel(X, gamma):
    N = X.shape[0]
    K = np.zeros((N, N))
    for i in range(K.shape[0]):
        for j in range(K.shape[1]):
            K[i, j] = np.exp(-(1/gamma**2) * LA.norm(X[i] - X[j])**2)
    return K

def incomplete_cholesky_decomposition(K, eta):
    ell, ell = K.shape
    j, nu = 0, []
    R = np.zeros((ell, ell))
    d = K.diagonal().copy()
    a, I = d.max(), [d.argmax()] # [a,I(1)] = max(d);
    while a > eta:
        nu.append(np.sqrt(a))
        for i in range(ell):
            R[j, i] = (K[I[j], i] - R[:, i].T @ R[:, I[j]])/nu[j]
            d[i] = d[i] - R[j, i] ** 2
        a, I = d.max(), I + [d.argmax()] # [a,I(j+1)] = max(d);
        j += 1
    T = j
    return R[0:T, :]

N = 1000
sigma = 0.01
t = np.random.uniform(0, 2 * np.pi, size=(N, 1))
n1 = np.random.normal(0, sigma, (N, 1))
n2 = np.random.normal(0, sigma, (N, 1))
X = np.sin(t) + n1
Y = np.cos(t) + n2
plt.plot(X, Y, "b.")
gamma = 1
eta = 1e-4

K = get_gaussian_kernel(X, gamma)
L = get_gaussian_kernel(Y, gamma)
H = np.eye(N) - (1/N * np.ones(N))

Ktilde = H @ K @ H
Ltilde = H @ L @ H
Z = np.zeros((N, N))

A = np.block([
    [Z, (1/N * Ktilde @ Ltilde)],
    [(1/N * Ltilde @ Ktilde), Z]
])
B = np.block([
    [Ktilde, Z],
    [Z, Ltilde]
])
```

```
def get_efficient_COCO():
    N = X.shape[0]
    eta = 1e-5

    R = incomplete_cholesky_decomposition(B, eta)
    AB = LA.pinv(R.T) @ A @ LA.pinv(R)
    eigvals, eigvecs = LA.eigh(AB)

    coco = eigvals[-1]

    eigvecs_reg = LA.pinv(R) @ eigvecs[:, -1]
    alpha = np.sqrt(2) * eigvecs_reg[N:(2*N)]
    beta = np.sqrt(2) * eigvecs_reg[0:N]

    f = K @ H @ alpha
    g = L @ H @ beta

    return f, g, coco

f, g, coco = get_efficient_COCO()
```

Appendix B Python Kernel CCA

Using the function `get_gaussian_kernel()` described above in A:

```
N = 1000
sigma = 0.01
t = np.random.uniform(0, 2 * np.pi, size=(N, 1))
n1 = np.random.normal(0, sigma, (N, 1))
n2 = np.random.normal(0, sigma, (N, 1))
X = np.sin(t) + n1
Y = np.cos(t) + n2
plt.plot(X, Y, "b.")
gamma = 1
eta = 1e-4

K = get_gaussian_kernel(X, gamma)
L = get_gaussian_kernel(Y, gamma)
H = np.eye(N) - (1/N * np.ones(N))

Ktilde = H @ K @ H
Ltilde = H @ L @ H
Z = np.zeros((N, N))

def get_kernel_cca():
    N = X.shape[0]
    reg = 1e-1

    A = np.block([
        [Z, (1/N * Ktilde @ Ltilde)],
        [(1/N * Ltilde @ Ktilde), Z]
    ])
    B = np.block([
        [sqr(Ktilde) + reg * Ktilde, Z],
```

```
        [Z, sqr(Ltilde) + reg * Ltilde]
    ])

    eigvals, eigvecs = smart_eigh(A, B)
    cca = eigvals[-1] * N

    alpha = np.sqrt(2) * np.sqrt(N) * eigvecs[0:N, -1]
    beta = np.sqrt(2) * np.sqrt(N) * eigvecs[N:(2*N), -1]

    f = K @ H @ alpha
    g = L @ H @ beta

    return f, g, cca

f, g, cca = get_kernel_cca()
```