Machine Learning Homework 4

Eric Larsen, A02176917

March 2024

Collaboration

For this assignment I worked through issues at the TA's office hours and with Cody Grogan. I also used Chatgpt to understand more about concepts and help fix bugs in my code as I was developing it.

Problem 1: Deep Learning Paper

The basis of this paper is to go through an overview of machine learning methods and how they have evolved of the years to involve deep learning methods. The main focus of this paper, as with the machine learning community, was on supervised tasks. Unsupervised learning was left as a final comment at the conclusion of the paper as future work in the community. This paper starts out with explaining how representation learning can be a difficult task, and how machine learning has been able to learn its own representation from data presented. This paper mainly goes through and highlights the success of deep learning and how it has been built for different problems.
The problems stated in this paper range from image recognition, natural language processing, caption generation, data predicted etc.. The main focus of this paper seemed to be on image processing and the use of convolutional neural networks instead of favoring all methods individually. Explaining how data can be extracted using a number of filters designed to do different tasks (i.e. the kernels used). This extracted data is then pieced together in following layers to give a more complete understanding of the image. The paper also hints at how NLP representation of language can be advantageous when a word's vector representation is not known but can be learned through a deep learning method. Overall this paper goes through and gives a good high level overview of different learning tasks and the possible methods to solve those problems.

Problem 2: Conceptual Questions

Part A)

The main difference between LDA and PCA is in their end goals. LDA wants to maximize separation of data given different classifications preserving the initial known classes. While PCA wants to find the maximum variance amount the data set and maintain that in a new data representation. Another main difference between these two techniques is that LDA is a supervised task, and PCA is an unsupervised task.

Part B)

Multicollinearity isn't as big of an issue in PCA because the focus of PCA is finding the principle components that represent the variance in the data. Where the first principle component represents the maximum variance, and so on. Where these PCs are independent of each other, so the data itself might have multicollinearity but the PCs produced will not be dependent on each other. This shows that the variance explained by the first PC has no bearing on the variance explained by the second PC. So the original data can have multicollinearity and PCA will not correct that, but PCA will not have multicollinearity among its outputs.

Part C)

With PCA being focused on the variance of the data it is sensitive to outliers seeing them as a large contribution to the variance of the dataset. In order to overcome this issue there would either need to be a data pre-filtering process before PCA, or use of a method less sensitive to outliers like Robust PCA. Another limitation of PCA is knowing how many principle components to use to justify the variance of the original dataset, you can overcome this like we discussed in class by using a scree plot to see how much is retained. PCA will also have issues if the scales of the dataset are not the same where one feature will dominate over others. Overcoming this can be done by normalizing the data features so they are of similar scales.

Part D)

If we already have the GMM trained we would decide how many new data points we want and then predict these given what we know of the GMM and its sub components. This would then give us N new samples to the overall dataset. You then check these points against the known GMM distribution and how they match the expected distribution, and iterate if necessary by generating new samples that agree with the original GMM they were produced from.

Problem 6: Canonical Correlation Analysis (CCA)

Part A)

With the original optimization problem of

$$a, b = argmax_{a,b} \quad corr(Xa, Yb) \tag{1}$$

we can rewrite this optimization with the known equation for Corr(A,B) where

$$corr(A, B) = \frac{||A||||B||cos\theta}{\sigma(A)\sigma(B)} \tag{2}$$

where we know the $\sigma(A)$ and $\sigma(B)$ are the magnitude of the matrices given, or the collection of vectors given in matrix form. Since we know this we can then rewrite the former equation with our known matrices yielding

$$corr(A, B) = \frac{||A||||B||cos\theta}{||A||||B||} \tag{3}$$

where this then can be reduced to the cost between the two matrices, where we know the magnitude of these vectors must be 1 so we then reduce this equation to a simple dot product between the two matrices yielding

$$corr(Xa, Yb) =< Xa, Yb > \tag{4}$$

when performing this operation it then yields the following

$$corr(Xa, Yb) = a^T X^T Y b \tag{5}$$

Part B)

solving the optimization problem we can write the Lagrangian as

$$L(a, b) = a^T X^T Y b + \lambda_1(1 - ||Xa||) + \lambda_2(1 - ||Yb||) \tag{6}$$

$$L(a, b) = a^T X^T Y b + \lambda_1(1 - a^T X^T X a) + \lambda_2(1 - b^T Y^T Y b) \tag{7}$$

Taking the partials of this yields the following

$$\frac{\partial L}{\partial a} = X^T Y b - 2\lambda_1 X^T X a = 0 \tag{8}$$
$$\frac{\partial L}{\partial b} = Y^T X a - 2\lambda_2 Y^T Y b = 0 \tag{9}$$
$$\frac{\partial L}{\partial \lambda_1} = 1 - a^T X^T X a = 0 \tag{10}$$
$$\frac{\partial L}{\partial \lambda_2} = 1 - b^T Y^T Y B = 0 \tag{11}$$

Taking equations 8 and 9 we can multiply by a^T and b^T respectively to yield

$$a^T X^T Y b = 2\lambda_1 a^T X^T X a \tag{12}$$
$$b^T Y^T X a = 2\lambda_2 b^T Y^T Y b \tag{13}$$

Given our constraints such that

$$||Xa||^2 = ||Yb||^2 = 1 \tag{14}$$

we can see that equations 12 and 13 can be written as an equality to the λ_1, λ_2 values. Since the LHS of both equations yields a scalar we know the transpose of a scalar is the same scalar allows us to show that

$$a^T X^T Y b = 2\lambda_1 = \lambda = 2\lambda_2 = b^T Y^T X a \tag{15}$$

Therefore we can show that $\lambda_1 = \lambda_2 = \lambda$. If we sub the results of an arranged equation 8 back into the original optimization problem we get the following maximization problem

$$max_{a,b} \quad 2\lambda a^T X^T X a \tag{16}$$

where since we have the constraints that $||Xa||^2 = 1$ we can then show this problem is solving for the maximum eigenvalues of the matrix reducing this to

$$max_{\lambda} \quad 2\lambda \tag{17}$$

Part C)

Both of these methods seek to reduce the dimensionality of the dataset while still preserving the components that contribute the most to the information in the system. The main difference is how these methods retain this information. For PCA it tries to find the principal components (orthogonal axes) that retain the most information through the use of eigenvalues and eigenvectors. PCA looks to retain the most important components that explain the variance in the dataset. Where CCA looks to find linear combinations of the original datasets X with target Y that maximize the correlation between these combinations. CCA considers the dataset and the relationships between X and Y instead of just looking at the variance of X.

Part D)

Applying CCA to the desired dataset yielded the following results shown below. This code was implemented in python using the sklearn library to create the CCA method.

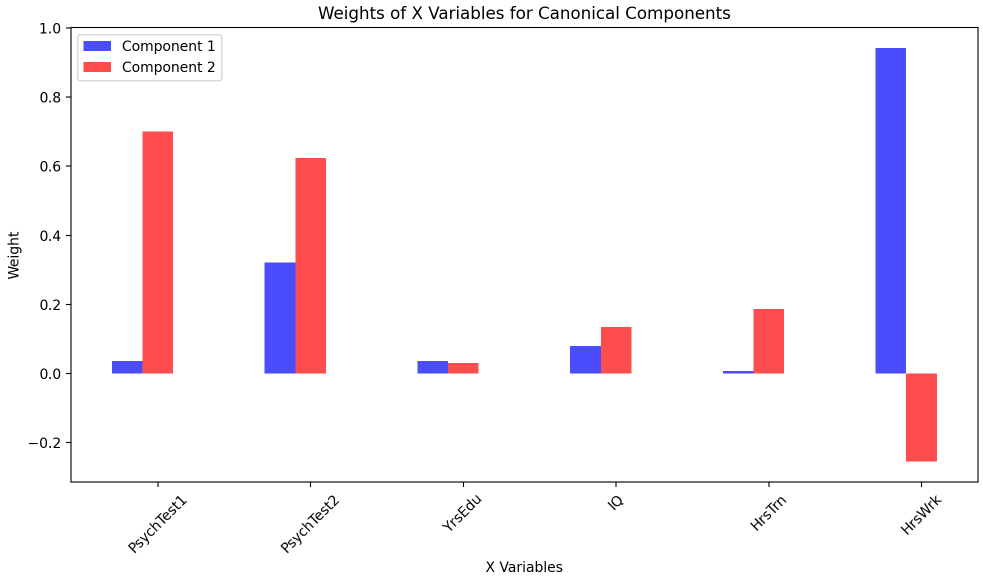

Figure 1: Results of CCA analysis on performance dataset given X

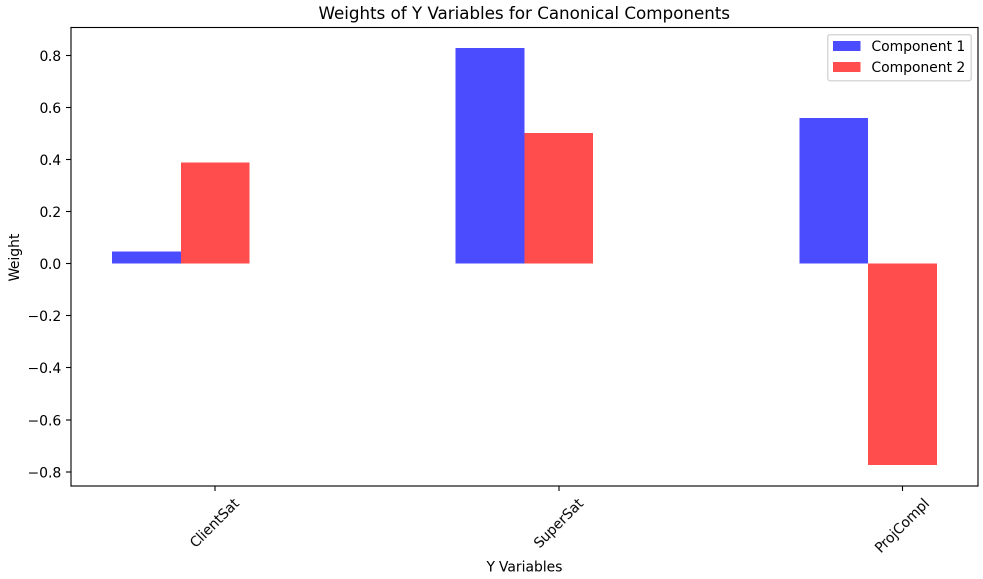

Figure 2: Results of CCA analysis on performance dataset given X

From these results we can see that the first two psych tests and the number of Hours Worked are the main components given x where the Y data. Where the psych tests have a positive correlation to client sat and super sat, while hours worked has a negative correlation. We also see that the hours worked has the highest correlation to super sat and projects completed. From these use of CCA we can see that there is a large correlation between specific aspects of the employee evaluation and the expected performance criteria.

Problem 7: Kernel PCA

Part A and B)

Taking what we know from equations 2 and 3 from the assignment text we can write there combination as follows

$$\frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T\sum_{j=1}^{N}a_{ij}\phi(x_i)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(x_n) \tag{18}$$

rearranging this equation will yield

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{N}a_{ij}\phi(x_n)\phi(x_n)^T\phi(x_i)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(x_n) \tag{19}$$

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{N}a_{ij}\phi(x_n)k(x_n,x_i)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(x_n) \tag{20}$$

Now we can multiply each side by $\phi(x_l)^T$ to yield to the following equation to answer part A of this problem

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{N}a_{ij}\phi(x_l)^T\phi(x_n)k(x_n,x_i)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(x_l)^T\phi(x_n) \tag{21}$$

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{N}a_{ij}k(x_l,x_n)k(x_n,x_i)=\lambda_i\sum_{n=1}^{N}a_{in}k(x_l,x_n) \tag{22}$$

Now we have this equation in terms of kernels allowing us to vectorize the equation to yield

$$\frac{1}{N}K^2a_i=\lambda_iKa_i \tag{23}$$

$$K^2a_i=N\lambda_iKa_i \tag{24}$$

we can then multiply by the inverse of K without effecting the result to then yield

$$Ka_i=N\lambda_ia_i \tag{25}$$

we can then redefine our λ_i as $\lambda_i=\lambda_iN$ to yield an equation of the form

$$Ka_i=\lambda_ia_i \tag{26}$$

Where this shows the eigenvalues of the Kernel matrix relate back to our projection from our original dataset

Part C)

Having a data matrix that is not centers requires us to find a new way to do the kernel trick. If we just add the offset to our original formulation we will not be able to use the kernel trick to help us. One way to solve this problem is to multiply the adjusted center data by another which complicates our kernel formulation. This looks like

$$\tilde{\phi}(x_l)\tilde{\phi}(x_n)=\left(\phi(x_l)-\frac{1}{N}\sum_{l=1}^{N}\phi(x_l)\right)^T\left(\phi(x_n)-\frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\right) \tag{27}$$

$$\tag{28}$$

expanding this our yields

$$\phi(x_l)^T\phi(x_n)-\frac{1}{N}\sum_{l=1}^{N}\phi(x_l)^T\phi(x_n)-\frac{1}{N}\sum_{n=1}^{N}\phi(x_l)^T\phi(x_n)+\frac{1}{N^2}\sum_{n,l=1}^{N}\phi(x_l)^T\phi(x_n) \tag{29}$$

where not each of these can be kernelized yielding

$$k(x_l, x_n) - \frac{1}{N}\sum_{l=1}^{N} k(x_l, x_n) - \frac{1}{N}\sum_{n=1}^{N} k(x_l, x_n) + \frac{1}{N^2}\sum_{n,l=1}^{N} k(x_l, x_n) \tag{30}$$

Using this we can then see our solution turn into with this result being shown in literature [1]

$$\tilde{K} = K - 1_n K - K 1_n + 1_n K 1_n \tag{31}$$

where 1_n is a matrix of entries $1/n$

Part D)

Implementation of this kernelized PCA yielded the following results shown in the figures below. The code for this problem can be seen in the submitted file Problem 7 code file submitted with this assignment.

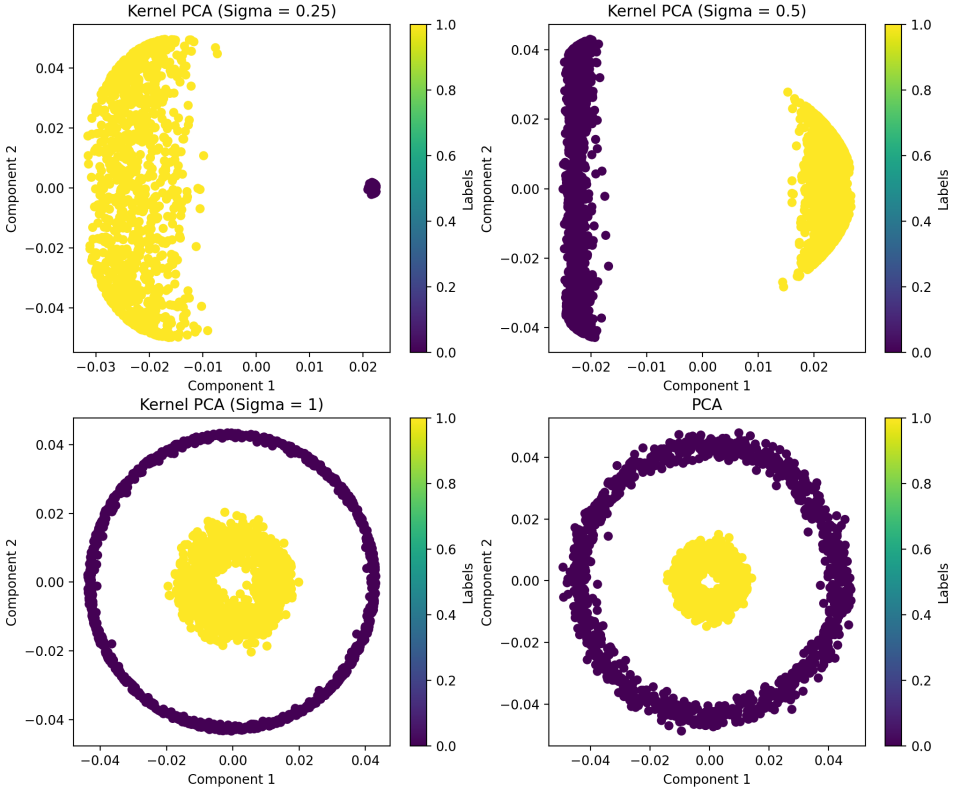

Figure 3: Kernel PCA on circle dataset given various sigma values

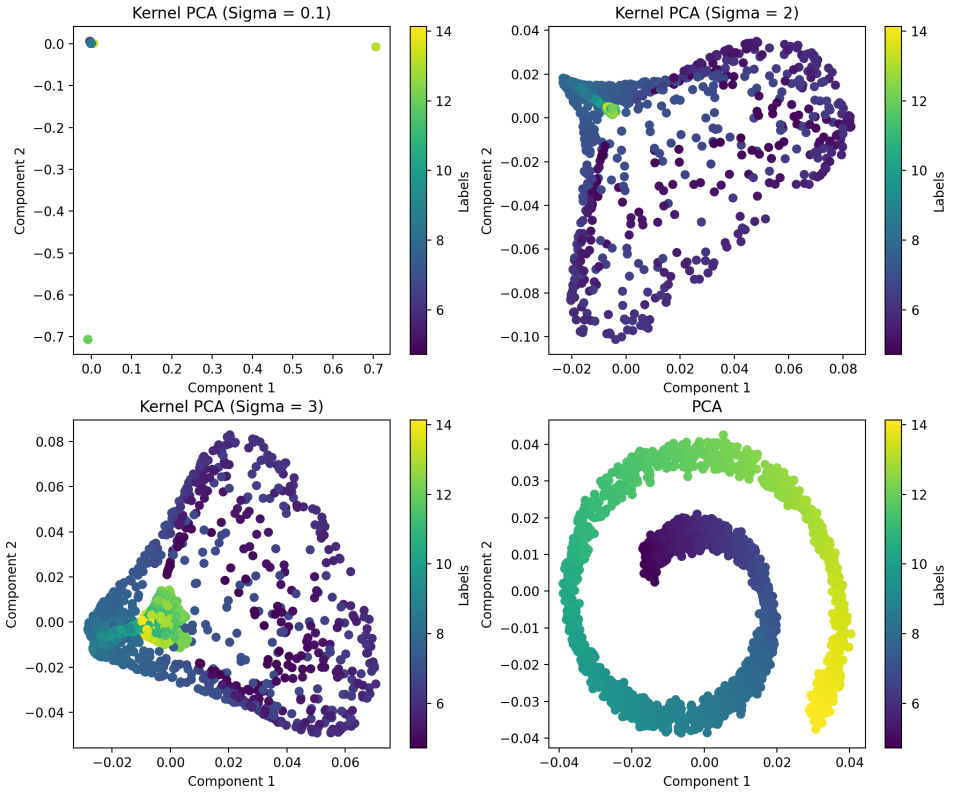

Figure 4: Kernel PCA on Swiss roll dataset given various sigma values

From these results we see that a low sigma value worked well or the circle dataset, while a higher sigma was required for the Swiss roll dataset. Where the low sigma value in the Swiss dataset produces no useful data results. During experimentation it was seen that if you go too high on the sigma for the Swiss roll set the data would give back a very close representation of the PCA method used.

Problem 8: EM Algorithm for Mixed Linear Regression

Part A)

Starting out with the log likelihood function we get the following

$$l(\theta; \underline{y}|\underline{x}) = \sum_{i=1}^{N} log\left(\sum_{k=1}^{K} \epsilon_k \phi(\underline{y}; w_k^T \underline{x} + b_k, \sigma_k^2)\right) \tag{32}$$

from this we can add our hidden variable where

$$\underline{z} = (\underline{y}, \underline{s}) \tag{33}$$

$$\tag{34}$$

where

$$\Delta_{i,k} = \begin{cases} 1 & \text{if } s_i = k \\ 0 & s_i \neq k \end{cases}$$

Adding this hidden variable to our log likelihood to create the complete-data log-likelihood yielding the following

$$l(\theta; \underline{y}, \underline{s}) = log(\Pi_{i=1}^{N} Pr(\underline{S} = \underline{s}; \theta) f(\underline{y}|\underline{s}; \theta)) \tag{35}$$

$$l(\theta; \underline{y}, \underline{s}) = \sum_{i=1}^{N} log(\epsilon_{s_i} \cdot \phi(y_i; w_k^T x_i + b_k, \sigma_k) \tag{36}$$

$$l(\theta; \underline{y}, \underline{s}) = \sum_{i=1}^{N} \sum_{k=1}^{K} log(\Delta_{i,k} \epsilon_k \cdot \phi(y_i; w_k^T x_i + b_k, \sigma_k) \tag{37}$$

giving the formula for $Q(\theta, \theta^{(j)})$ gives the following

$$Q(\theta, \theta^{(j)}) = \gamma_{i,k}^{(j)} \sum_{i=1}^{N} \sum_{k=1}^{K} log(\epsilon_k \cdot \phi(y_i; w_k^T x_i + b_k, \sigma_k) \tag{38}$$

calculating the E step yields the following

$$\gamma_{i,k}^{(j)} = E[\Delta_{i,k}|\underline{y}; \theta^{(j)}] \tag{39}$$

$$\gamma_{i,k}^{(j)} = Pr(\Delta_{i,k} = 1|\underline{y}; \theta^{(j)}) \tag{40}$$

$$\gamma_{i,k}^{(j)} = Pr(S_i = k|\underline{y}; \theta^{(j)}) \tag{41}$$

$$\gamma_{i,k}^{(j)} = \frac{Pr(S_i = k; \theta^{(j)}) f(y_i|S_i = k; \theta^{(j)})}{f(y_i|\theta^(j)} \tag{42}$$

$$\gamma_{i,k}^{(j)} = \frac{\epsilon_k \cdot \phi(y_i; w_k^T x_i + b_k, \sigma_k)}{\sum_{l=1}^{K} \epsilon_l \cdot \phi(y_l; w_k^T x_l + b_k, \sigma_l)} \tag{43}$$

Part B)

Solving for the M step we first start out with the log likelihood equation

$$l(\theta; \underline{y}|\underline{x}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \Delta_{i,k} \left[log(\epsilon_k) - \frac{1}{2} log(2\pi) - \frac{1}{2} log(\sigma_k^2) - \frac{1}{2\sigma_k^2} \left(y_i - w_k^T x_i - b_k \right)^2 \right] \tag{44}$$

Starting out we will use a lagrange multiplier to optimize for the ϵ portion giving us the following equation below using the known constraint that

$$1 = \sum_{k=1}^{K} \epsilon_k \tag{45}$$

$$L(\epsilon_k) = \sum_{i=1}^{N} \gamma_{i,k}^{(j)} log \epsilon_k + \lambda_1 (1 - \sum_{k=1}^{K} \epsilon_k) \tag{46}$$

taking the gradient of this yields

$$\frac{\partial L}{\partial \epsilon_k} = \sum_{i=1}^{N} \frac{\gamma_{i,k}}{\epsilon_k} - \lambda_1 = 0 \tag{47}$$

solving this equality yields

$$\epsilon_k = \frac{1}{\lambda_1} \sum_{i=1}^{N} \gamma_{i,k} \tag{48}$$

subbing this back into our constraint equation (eq 45) yields

$$\lambda_1 = \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{i,k} \tag{49}$$

$$\lambda_1 = N \tag{50}$$

plugging this back into our equation 48 gives use the next ϵ value

$$\epsilon_k^{(j+1)} = \frac{1}{N}\sum_{i=1}^{N}\gamma_{i,k}$$

Continuing to optimize our w_k, b_k, σ_k^2 values then yield the following, starting with the weight and bias together. Combining these into a single matrix w, can then adapting the x matrix

$$\tilde{w_k} = \begin{bmatrix} w_{1k} & \cdots & w_{nk} & b_k \end{bmatrix}^T \tag{51}$$

$$\tilde{x_i} = \begin{bmatrix} x_{1i} & x_{2i} & \cdots & x_{ni} & 1 \end{bmatrix} \tag{52}$$

$$\tilde{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & 1 \\ \vdots & & & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} & 1 \end{bmatrix} \tag{53}$$

$$\tilde{W} = \begin{bmatrix} w_{11} & \cdots & w_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ w_{k1} & \cdots & w_{kn} & b_k \end{bmatrix}^T \tag{54}$$

Using these we can then write lagrangian with respect to our new $\tilde{w}$ as follows holding σ_k^2 as a constant we can remove this as it can be multiplied out when doing the gradient of the function

$$L(\tilde{w_k}) = \sum_{i=1}^{K}\frac{-\gamma_{i,k}}{2\sigma_k^2}\left(y_i - \tilde{x_i}\tilde{w_k}\right)^2 \tag{55}$$

Now vectorizing this to remove the need of the summation requires us to use a diagonal matrix with the $\gamma_{i,k}$ values like shown below

$$\Gamma_k = \begin{bmatrix} \gamma_{1,k} & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \gamma_{n,k} \end{bmatrix} \tag{56}$$

using this we can rewrite the lagrangian as

$$L(\tilde{W_k}) = \left(Y - \tilde{X}\tilde{W_k}\right)^T \Gamma_k \left(Y - \tilde{X}\tilde{W_k}\right) = 0 \tag{57}$$

Now taking the gradient of this yields

$$\frac{\partial L}{\partial \tilde{W_k}} = -2\tilde{X}^T\Gamma_k Y + 2\tilde{X}^T\Gamma_k\tilde{X}\tilde{W_k} = 0 \tag{58}$$

solving this gives

$$\tilde{W_k}^{(j+1)} = (\tilde{X}^T\Gamma_k\tilde{X})^{-1}\tilde{X}^T\Gamma_k Y \tag{59}$$

Finally we can solve the required change in the σ_k value as shown below

$$L(\sigma_k^2) = \sum_{i=1}^{N}\frac{-\gamma_{i,k}}{2}log\sigma_k^2 - \sum_{i=1}^{N}\frac{\gamma_{i,k}}{2(\sigma_k^2)}\left(y_i - \tilde{x_i}\tilde{w_k}\right)^2 \tag{60}$$

$$\frac{\partial L}{\partial \sigma_k^2} = \sum_{i=1}^{N}\frac{-\gamma_{i,k}}{2\sigma_k^2} + \sum_{i=1}^{N}\frac{\gamma_{i,k}}{2(\sigma_k^2)^2}\left(y_i - \tilde{x_i}\tilde{w_k}\right)^2 = 0 \tag{61}$$

solving this equality for optimization yields

$$\sigma_k^2 = (y_i - \tilde{x}_i \tilde{w}_k)^2 \tag{62}$$

$$\sigma_k^{2(j+1)} = \left(Y - \tilde{X}\tilde{W}_k\right)^T \Gamma_k \left(Y - \tilde{X}\tilde{W}_k\right) \tag{63}$$

Where the leading coefficient is 1 divided by the sum of the column k requiring us to loop through each K value given our dataset

Part C)

Implementation of this EM method was done using python and the code is shown in the submitted files shown as problem 8. The results are shown below. In my code I was not able to get it to implement correctly. I double checked the matrix multiplications, but the steps converge to two of the same line. I checked my math with the TA and the E and M steps are correct but something is going wrong in the code. I have walked through it and find the problem. The following results are from the current state of the EM method giving the requested information

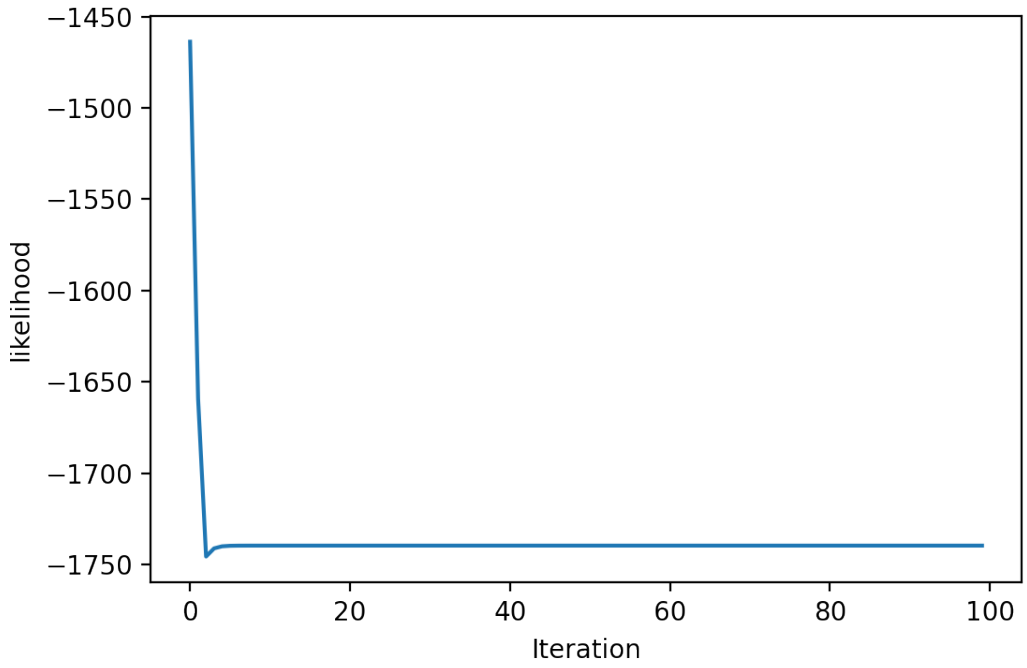

Figure 5: Log Likelihood over iteration

Where the method produces the following result for the end lines

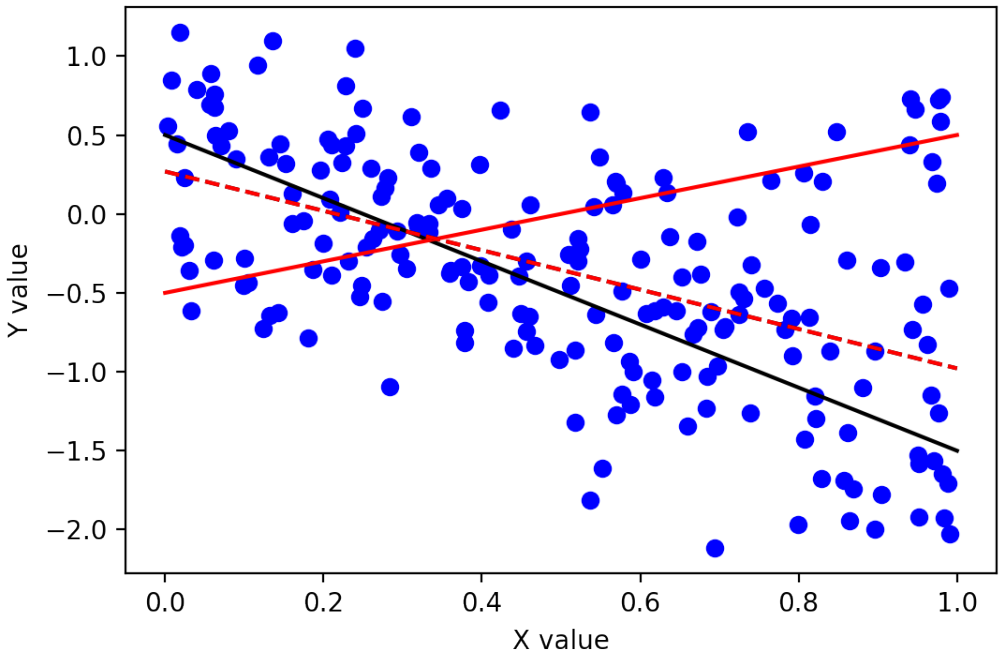

Figure 6: X Y with true lines and expected lines from EM method

We can see from these results it tries to split the difference between the lines. From the equations above we know the math is solid for the E and M steps but this code has a bug in it that I can not find

References

[1] B. Scholkopf, A. J. Smola, & K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, MaxPlanckInstitut fur biologische Kybernetik, 1996. Submitted to Neural Computation.