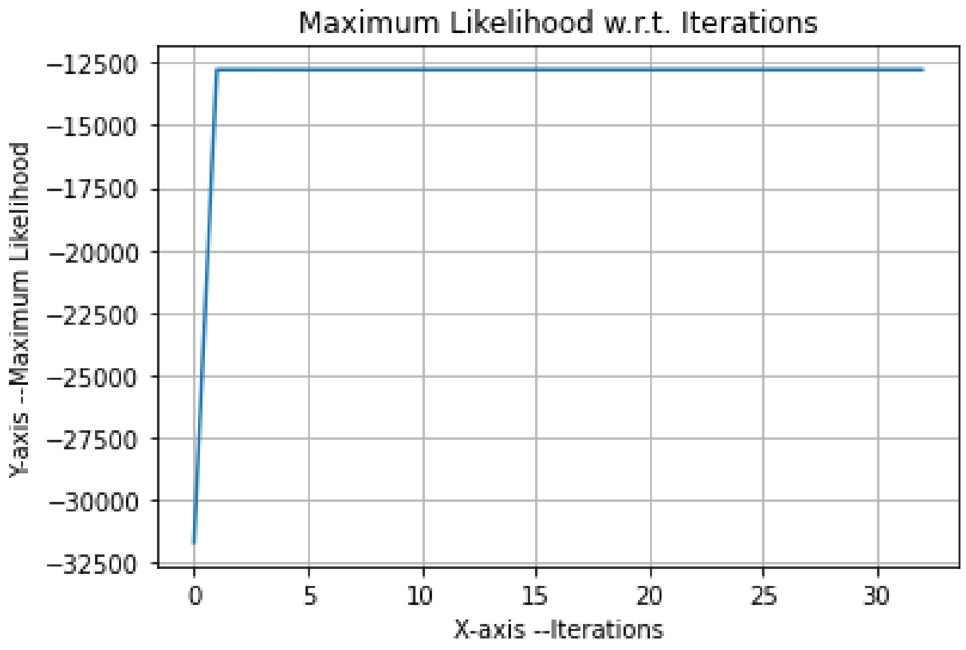
Pattern Recognition and Machine Learning - Assignment 2

Question 1. You are given a data-set with 400 data points in {0, 1} 50 generated from a mixture of some distribution in the file A2Q1.csv. (Hint: Each datapoint is a flattened version of a {0, 1} 10×5 matrix.)

Part i. Determine which probabilistic mixture could have generated this data (It is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Write a piece of code to implement the algorithm you derived by setting the number of mixtures K = 4. Plot the log-likelihood (averaged over 100 random initializations) as a function of iterations.

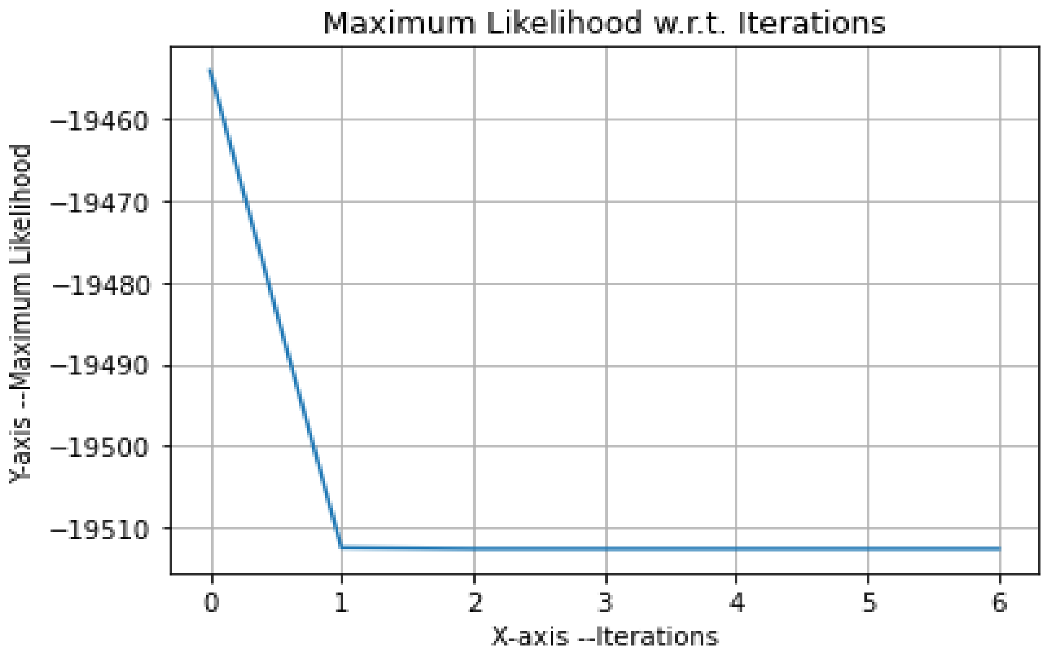
Solution: The plot obtained is as follows:



The data is assumed to be generated from bernoulli mixture model as the data provided is binary i.e 0 or 1. Hence there is a strong chance that it has been generated from an experiment involving coin tosses and thus the Bernoulli mixture model seems to be the model of choice.

Part ii. Assume that the same data was in fact generated from a mixture of Gaussians with 4 mixtures. Implement the EM algorithm and plot the log-likelihood (averaged over 100 random initializations of the parameters) as a function of iterations. How does the plot compare with the plot from part (i)? Provide insights that you draw from this experiment.

Solution: The Plot is as follows, due to some mistake of sign in the following the graph has been inverted. Kindly consider it generously.

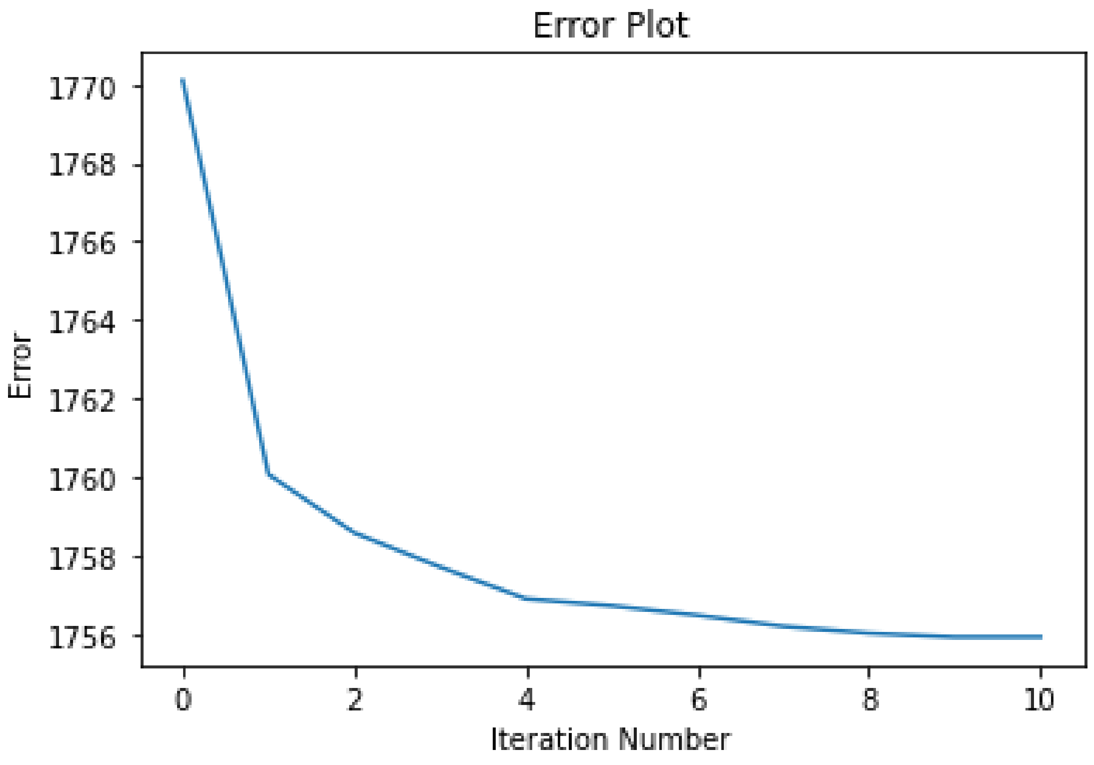


The Maximum Log Likelihood in bernoulli was in the range of -12786.6 but in gaussian the Maximum Log Likelihood value is in range of -19510.7. Thus it can be seen that the bernoulli gives a better estimate of the value and it seems to be more accurate data generating process as opposed to the gaussian mixture model.

Insights: Since the data is binary it seems that the data is most likely to be generated by a coin toss kind of experiment and hence bernoulli seems to be a better choice. The gaussian mixture model is preferred choice if the data was real value and continuous and not discrete as given in this dataset. Hence we can say it is better to use bernoulli for binary discrete data and it is better to use gaussian mixture model when the data is spread over a continuous range of values.

Part iii. Run the K-means algorithm with K = 4 on the same data. Plot the objective of K − means as a function of iterations.

Solution: The Error plot of the k means algorithm against 4 cluster is as follows:



The error is continuously decreasing with number of iterations and finally the k means with k=4 converges in 10 iterations.

Part iv. Among the three different algorithms implemented above, which do you think you would choose for this dataset and why?

Solution: Among the three different algorithms we can observe the following:

K - Means : This algorithm does not tell us anything about the data generating process it simply gives us the clusters of the data by putting the points appropriately thereby minimizing the objective loss function.

Bernoulli Mixture Model : In this model we try to estimate the data generating process by assuming that the experiment used to generate the data was coin tosses and since this data set is as points having binary features it makes a lot of sense to assume that the data generating process was about success and failure and thus consider discrete values. Also we can see the maximum log likelihood value for this is much better than that of the gaussian mixture model.

Gaussian Mixture Model : In this model we try to estimate the data generating process when the data is generally in a continuous interval which may contain possibly infinite values and hence it is preferred in such a setting. In the data set above since the data was binary and the gaussian mixture model could not beat the bernoulli mixture model.

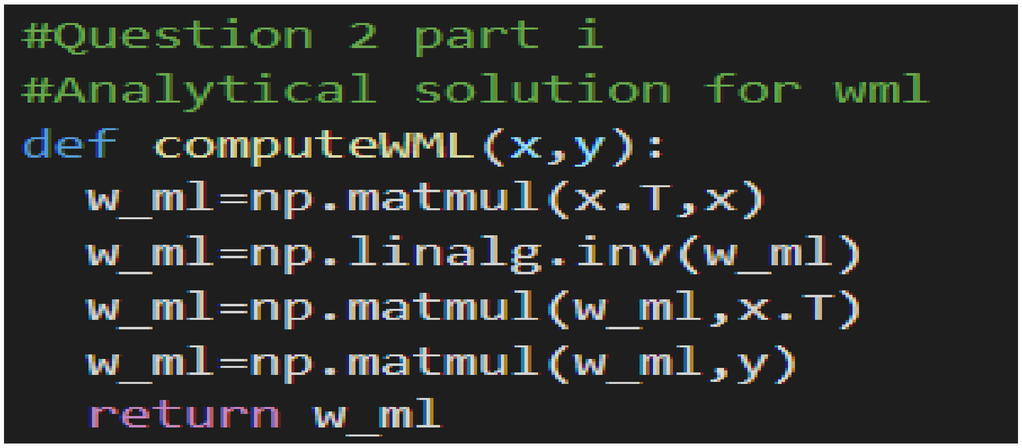
Thus for the above dataset the model of choice is **Bernoulli Mixture Model.**

Question 2. You are given a data-set in the file A2Q2Data train.csv with 10000 points in (R 100 , R) (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

Part i. Obtain the least squares solution wML to the regression problem using the analytical solution.

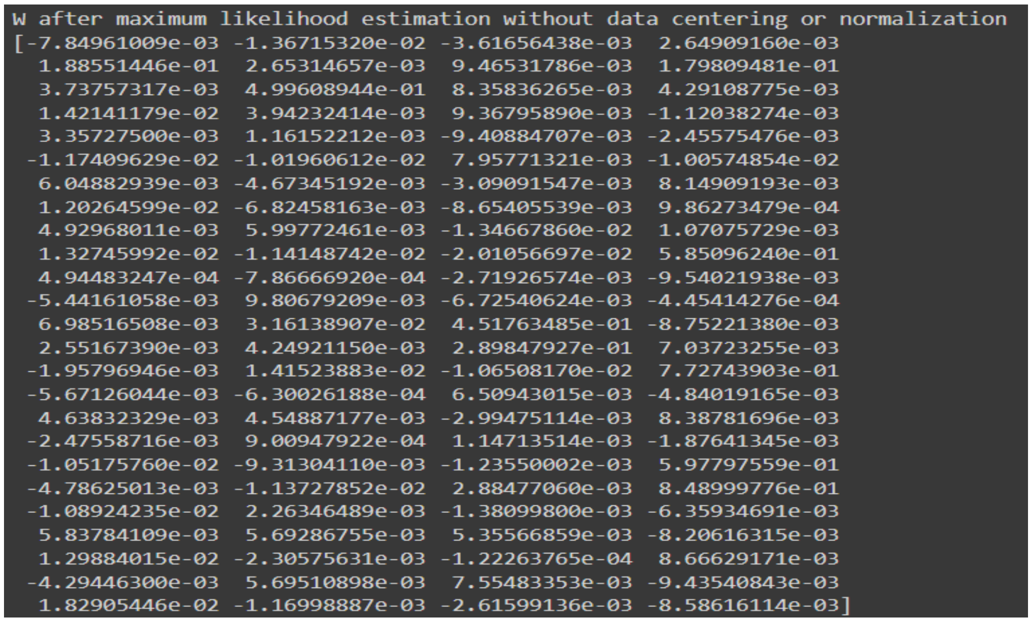
Solution: Calculating the analytical form of the solution using the closed form expression as derived in class is as follows:

Code:



```
#Question 2 part i
#Analytical solution for wml
def computeWML(x,y):
    w_ml=np.matmul(x.T,x)
    w_ml=np.linalg.inv(w_ml)
    w_ml=np.matmul(w_ml,x.T)
    w_ml=np.matmul(w_ml,y)
    return w_ml
```

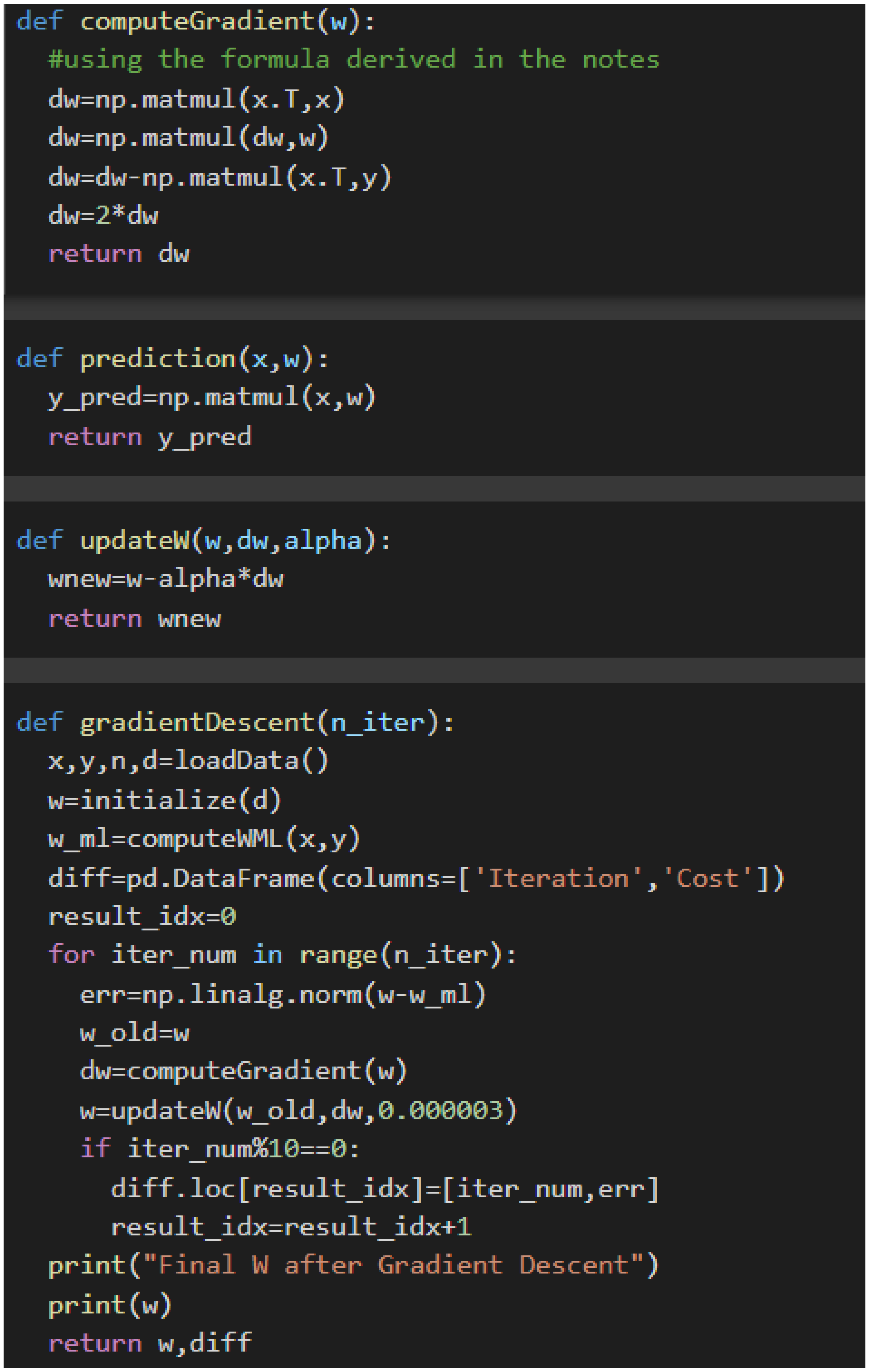
Final Solution:



```
W after maximum likelihood estimation without data centering or normalization
[-7.84961009e-03 -1.36715320e-02 -3.61656438e-03  2.64909160e-03
  1.88551446e-01  2.65314657e-03  9.46531786e-03  1.79809481e-01
  3.73757317e-03  4.99608944e-01  8.35836265e-03  4.29108775e-03
  1.42141179e-02  3.94232414e-03  9.36795890e-03 -1.12038274e-03
  3.35727500e-03  1.16152212e-03 -9.40884707e-03 -2.45575476e-03
 -1.17409629e-02 -1.01960612e-02  7.95771321e-03 -1.00574854e-02
  6.04882939e-03 -4.67345192e-03 -3.09091547e-03  8.14909193e-03
  1.20264599e-02 -6.82458163e-03 -8.65405539e-03  9.86273479e-04
  4.92968011e-03  5.99772461e-03 -1.34667860e-02  1.07075729e-03
  1.32745992e-02 -1.14148742e-02 -2.01056697e-02  5.85096240e-01
  4.94483247e-04 -7.86666920e-04 -2.71926574e-03 -9.54021938e-03
 -5.44161058e-03  9.80679209e-03 -6.72540624e-03 -4.45414276e-04
  6.98516508e-03  3.16138907e-02  4.51763485e-01 -8.75221380e-03
  2.55167390e-03  4.24921150e-03  2.89847927e-01  7.03723255e-03
 -1.95796946e-03  1.41523883e-02 -1.06508170e-02  7.72743903e-01
 -5.67126044e-03 -6.30026188e-04  6.50943015e-03 -4.84019165e-03
  4.63832329e-03  4.54887177e-03 -2.99475114e-03  8.38781696e-03
 -2.47558716e-03  9.00947922e-04  1.14713514e-03 -1.87641345e-03
 -1.05175760e-02 -9.31304110e-03 -1.23550002e-03  5.97797559e-01
 -4.78625013e-03 -1.13727852e-02  2.88477060e-03  8.48999776e-01
 -1.08924235e-02  2.26346489e-03 -1.38099800e-03 -6.35934691e-03
  5.83784109e-03  5.69286755e-03  5.35566859e-03 -8.20616315e-03
  1.29884015e-02 -2.30575631e-03 -1.22263765e-04  8.66629171e-03
 -4.29446300e-03  5.69510898e-03  7.55483353e-03 -9.43540843e-03
  1.82905446e-02 -1.16998887e-03 -2.61599136e-03 -8.58616114e-03]
```

Part ii. Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot kwt − wMLk2 as a function of t. What do you observe?

Solution: The snippets of the code are as attached:



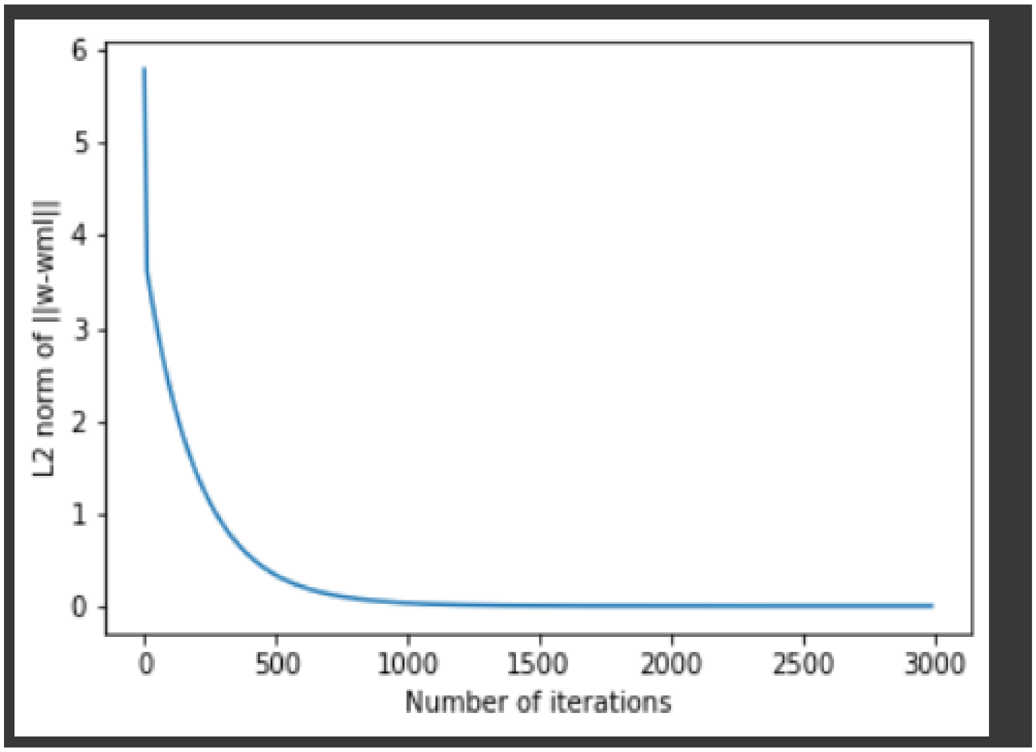
```python
def computeGradient(w):
  #using the formula derived in the notes
  dw=np.matmul(x.T,x)
  dw=np.matmul(dw,w)
  dw=dw-np.matmul(x.T,y)
  dw=2*dw
  return dw
```

```python
def prediction(x,w):
  y_pred=np.matmul(x,w)
  return y_pred
```

```python
def updateW(w,dw,alpha):
  wnew=w-alpha*dw
  return wnew
```

```python
def gradientDescent(n_iter):
  x,y,n,d=loadData()
  w=initialize(d)
  w_ml=computeWML(x,y)
  diff=pd.DataFrame(columns=['Iteration','Cost'])
  result_idx=0
  for iter_num in range(n_iter):
    err=np.linalg.norm(w-w_ml)
    w_old=w
    dw=computeGradient(w)
    w=updateW(w_old,dw,0.000003)
    if iter_num%10==0:
      diff.loc[result_idx]=[iter_num,err]
      result_idx=result_idx+1
  print("Final W after Gradient Descent")
  print(w)
  return w,diff
```

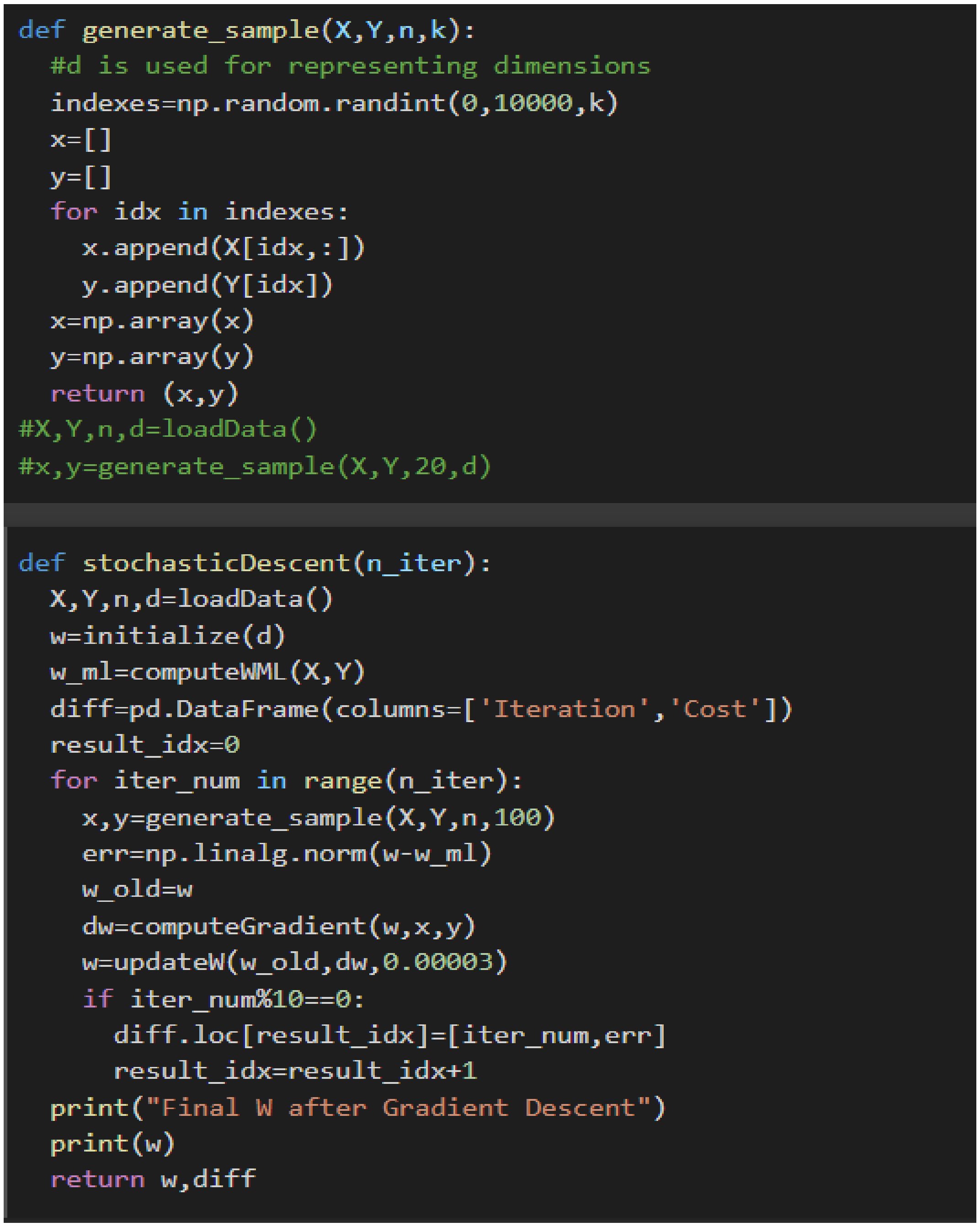
The variation of Wml and W is as follows:



Observation:
We start by a all one initialization of W which may be very far away from the optimal Wml and thus when we run the gradient descent in the start the gradient will be very steep and thus we will have large jumps. When the point of optimality is near then the gradient will become smaller and smaller and thus steps will be small and the difference between Wml and W will decrease slowly and finally it will become very narrow as the number of iterations will increase. Basically the graph first decreases steeply and then flattens out to nearly 0.

Part iii. Code the stochastic gradient descent algorithm using batch size of 100 and plot kwt − wMLk2 as a function of t. What are your observations?

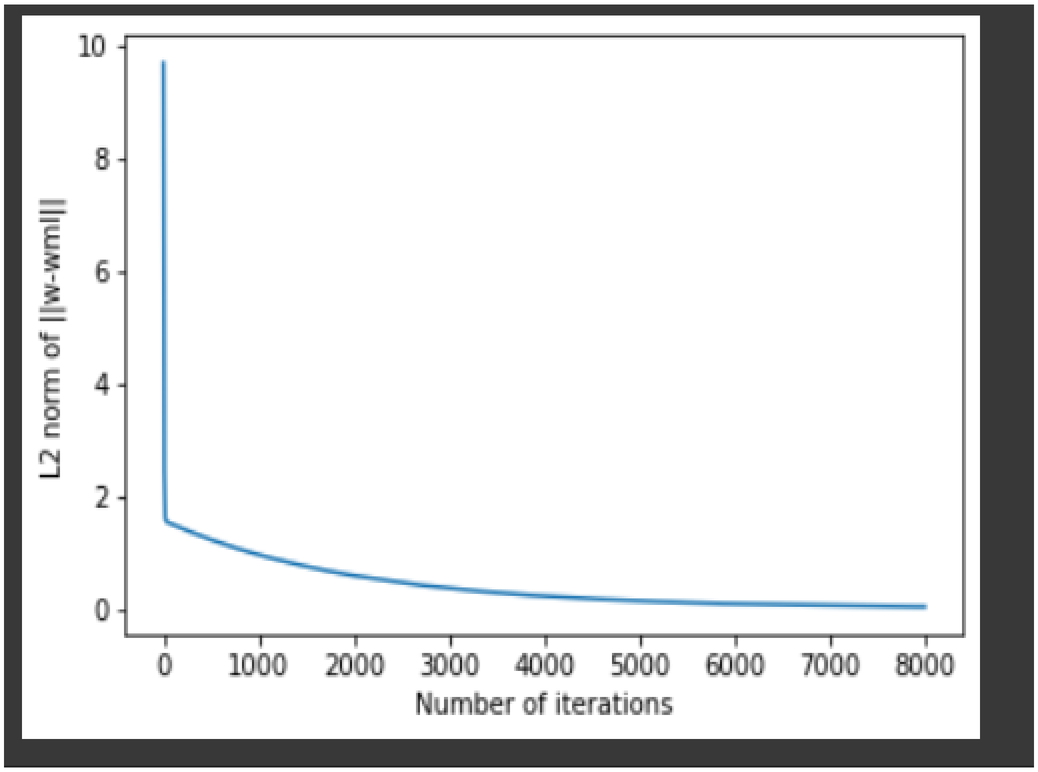
Solution: The code snippet are as attached:



```
def generate_sample(X,Y,n,k):
  #d is used for representing dimensions
  indexes=np.random.randint(0,10000,k)
  x=[]
  y=[]
  for idx in indexes:
    x.append(X[idx,:])
    y.append(Y[idx])
  x=np.array(x)
  y=np.array(y)
  return (x,y)
#X,Y,n,d=loadData()
#x,y=generate_sample(X,Y,20,d)
```

```
def stochasticDescent(n_iter):
  X,Y,n,d=loadData()
  w=initialize(d)
  w_ml=computeWML(X,Y)
  diff=pd.DataFrame(columns=['Iteration','Cost'])
  result_idx=0
  for iter_num in range(n_iter):
    x,y=generate_sample(X,Y,n,100)
    err=np.linalg.norm(w-w_ml)
    w_old=w
    dw=computeGradient(w,x,y)
    w=updateW(w_old,dw,0.00003)
    if iter_num%10==0:
      diff.loc[result_idx]=[iter_num,err]
      result_idx=result_idx+1
  print("Final W after Gradient Descent")
  print(w)
  return w,diff
```

The variation of Wml and W is as follows:



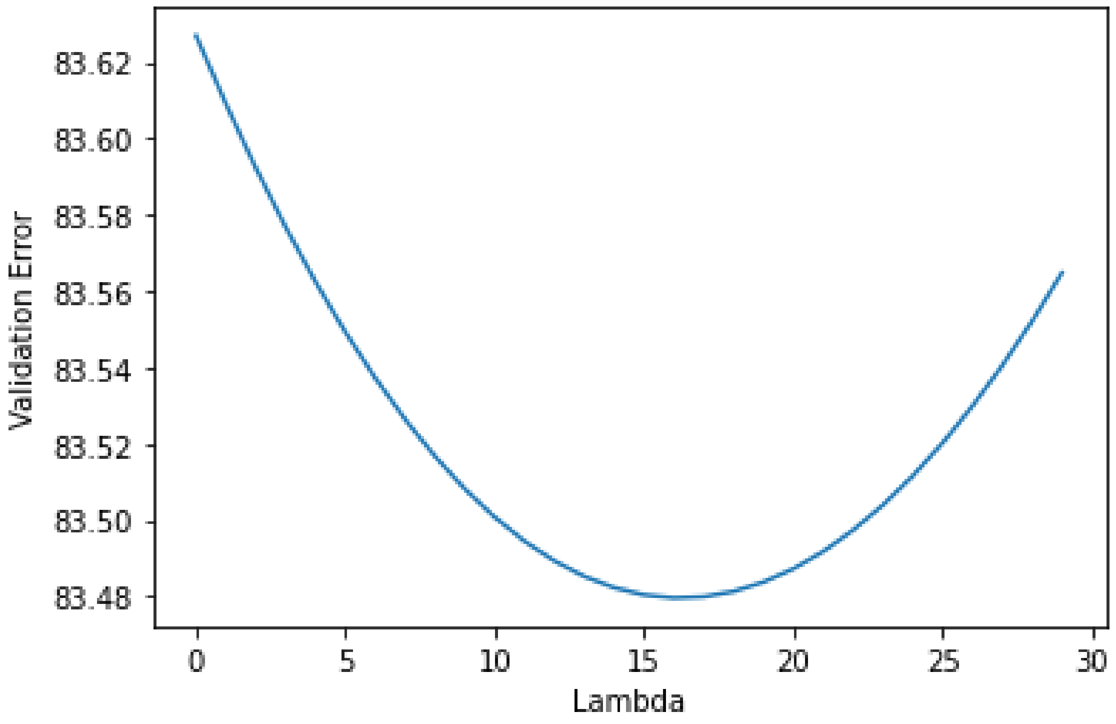
Observation: In this part we applied a stochastic gradient descent algorithm using the batch size of 100. It was easy to calculate the gradient as it was using only 100 data points as opposed to using the entire dataset for one epoch of the algorithm which is a major computational resource saving.

Additionally, since the gradient is computed over a subset of data sometimes it may so often happen that the batch taken is not a true representation of the data and we may move in a direction which is not the optimal direction but it is seen over large number of iteration we get lucky to converge close to the optimal solution.

Sometimes we may get fickle in the graph but the overall graph is smoothened out over a large number of iterations. Although it was observed that sometimes the error increased as compared to previous epochs, over a large number of iterations it was decreasing.

Part iv. Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of λ and plot the error in the validation set as a function of λ. For the best λ chosen, obtain wR. Compare the test error (for the test data in the file A2Q2Data test.csv) of wR with wML. Which is better and why?

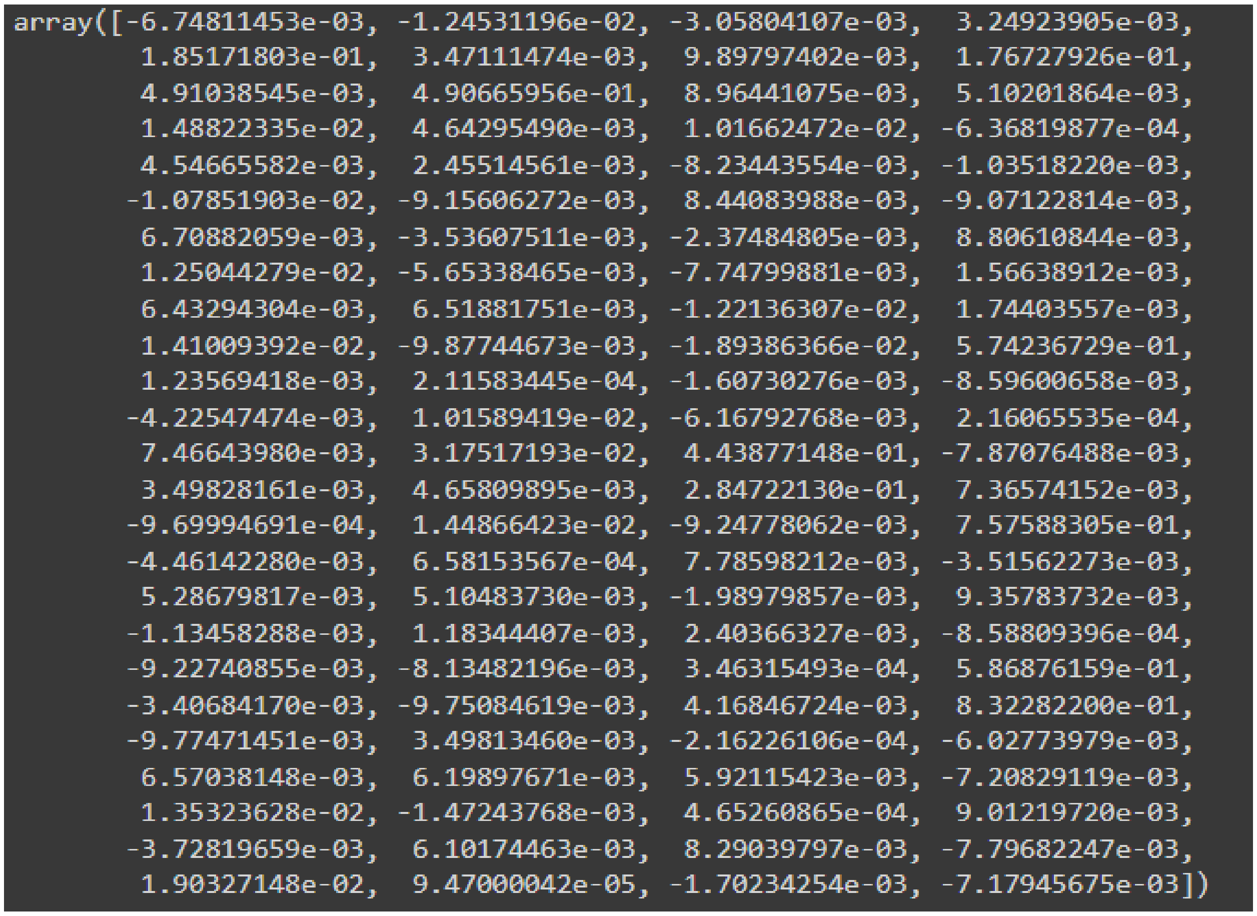
Solution: The plot of error against different values of lambda is as attached:



It was observed that the value of lambda where cross validation error was the lowest was recorded to be: 17.

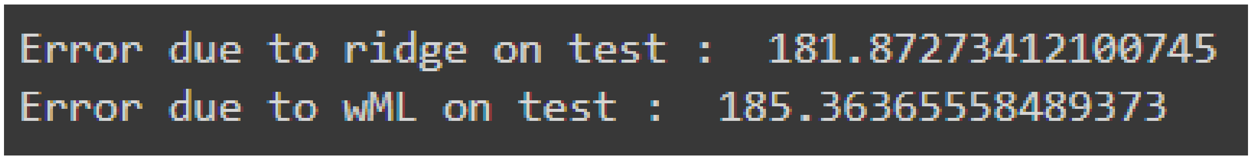
Thus lambda=17 for ridge regression.

The value of wR is as follows:



```
array([-6.74811453e-03, -1.24531196e-02, -3.05804107e-03,  3.24923905e-03,
        1.85171803e-01,  3.47111474e-03,  9.89797402e-03,  1.76727926e-01,
        4.91038545e-03,  4.90665956e-01,  8.96441075e-03,  5.10201864e-03,
        1.48822335e-02,  4.64295490e-03,  1.01662472e-02, -6.36819877e-04,
        4.54665582e-03,  2.45514561e-03, -8.23443554e-03, -1.03518220e-03,
       -1.07851903e-02, -9.15606272e-03,  8.44083988e-03, -9.07122814e-03,
        6.70882059e-03, -3.53607511e-03, -2.37484805e-03,  8.80610844e-03,
        1.25044279e-02, -5.65338465e-03, -7.74799881e-03,  1.56638912e-03,
        6.43294304e-03,  6.51881751e-03, -1.22136307e-02,  1.74403557e-03,
        1.41009392e-02, -9.87744673e-03, -1.89386366e-02,  5.74236729e-01,
        1.23569418e-03,  2.11583445e-04, -1.60730276e-03, -8.59600658e-03,
       -4.22547474e-03,  1.01589419e-02, -6.16792768e-03,  2.16065535e-04,
        7.46643980e-03,  3.17517193e-02,  4.43877148e-01, -7.87076488e-03,
        3.49828161e-03,  4.65809895e-03,  2.84722130e-01,  7.36574152e-03,
       -9.69994691e-04,  1.44866423e-02, -9.24778062e-03,  7.57588305e-01,
       -4.46142280e-03,  6.58153567e-04,  7.78598212e-03, -3.51562273e-03,
        5.28679817e-03,  5.10483730e-03, -1.98979857e-03,  9.35783732e-03,
       -1.13458288e-03,  1.18344407e-03,  2.40366327e-03, -8.58809396e-04,
       -9.22740855e-03, -8.13482196e-03,  3.46315493e-04,  5.86876159e-01,
       -3.40684170e-03, -9.75084619e-03,  4.16846724e-03,  8.32282200e-01,
       -9.77471451e-03,  3.49813460e-03, -2.16226106e-04, -6.02773979e-03,
        6.57038148e-03,  6.19897671e-03,  5.92115423e-03, -7.20829119e-03,
        1.35323628e-02, -1.47243768e-03,  4.65260865e-04,  9.01219720e-03,
       -3.72819659e-03,  6.10174463e-03,  8.29039797e-03, -7.79682247e-03,
        1.90327148e-02,  9.47000042e-05, -1.70234254e-03, -7.17945675e-03])
```

The following error on test data was recorded:



```
Error due to ridge on test :  181.87273412100745
Error due to wML on test :  185.36365558489373
```

Observation: When we run gradient descent on the training data sometimes what happens is that our model tries to minimize the error by sticking very hard to the training set and fails to capture the actual trends in the data. This situation is known as overfitting and thus we add some penalty to the loss function so that the model does not overfit and thus captures the trends in the data. The ridge regression uses l2 regularization to address overfitting. Lambda is a hyperparameter which controls the norm of w and keeps it in check so that we can reduce the parameter corresponding to the most correlated features in the dataset.
But we cannot put value of lambda as very high since at one time it will push all the parameters to zeros thereby not learning anything at all. Thus it is observed the error on the validation set first decreases on increasing the value of lambda and later it starts increasing. Thus we take the value where the bottom is achieved.
It is also seen that the ridge error decreases on the test data as opposed to the wMl error on the same data. Thus it can be seen that the ridge regression reduces overfitting and thus it serves its purpose.

Thank You