Cover page for answers.pdf CSE512 Fall 2019 - Machine Learning - Homework 2

Your Name: Shubham S Bagi

Solar ID: 112672171

NetID : sbagi

email address: shubham.bagi@stonybrook.edu

Names of people whom you discussed the homework with: Gurusangama

3.1 Implement LOOCV Ridge Regression

Code:

```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, cross_val_score
from statistics import mean

#reading the csv files

trdata= pd.read_csv("/content/gdrive/My Drive/ML/HW2/trainData.csv")
trlabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/trainLabels.csv")
valdata = pd.read_csv("/content/gdrive/My Drive/ML/HW2/valData.csv")
vallabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/valLabels.csv")
testlabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/testData_new.csv")

# Removing the extra row at the first

trdata = trdata.iloc[:,1:]
trlabel = trlabel.iloc[:,1:]
valdata = valdata.iloc[:,1:]
vallabel = vallabel.iloc[:,1:]
testlabel = testlabel.iloc[:,1:]
inputX = np.transpose(trdata.values)
output = trlabel.values
wvalues_ridge = []

#Ridge Regression Function

def Ridge1(X,Y,Lambda):
  (k,N) = X.shape
  X2 = np.ones((1,N))
  Xbar = np.vstack((X,X2))
  Xbartran = np.transpose(Xbar)
  Product = Xbar.dot(Xbartran)
  Identity = np.identity(k)
  #print(Identity)
  X0 = np.zeros((k,1))
  X1 = np.zeros((1,k+1))
  IdenHor = np.hstack((Identity,X0))
  IdenVer = np.vstack((IdenHor, X1))
  Mul = np.dot(IdenVer,Lambda)
```

```
  Num = np.add(Mul,Product)
  Denom  = np.dot(Xbar,Y)
  Inve = np.linalg.inv(Num)
  #print('Ridge')
  Final = np.dot(Inve,Denom)
  #print('\n')
  #print('The scalar b value')
  b = Final[-1]
  #print('\n')
  #print(b)
  Wval = Final[:-1]
  #print(Wval)
  wtran = np.transpose(Wval)
  sum1=0
  for i in range(0,N):
    o=X[...,i].ravel()
    otr= np.transpose(o)
    prod= 0
    prod = wtran.dot(otr)
    sum1 = sum1 + prod
  caly= sum1+b[0]
  pred = (caly - Y[i])**2
  WFinal = sum(np.array(Wval)**2)*Lambda
  obj = WFinal + pred
  """print('\n')
  print("Objective Function value")
  print(obj)"""
  Errori=[]
  for i in range(N):
    numerator=  np.transpose(Final).dot(Xbar[...,i].ravel()) - Y[i][0]
    sam= np.transpose(Xbar[...,i].ravel()).dot(Inve)
    sample = sam.dot(Xbar[...,i].ravel())
    denominator = 1 -sample
    Errori.append(numerator/denominator)

  return [Wval,b,obj,Errori]
```

3.2 Predicting goodness points of a wine given its review

Code:

```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, cross_val_score
from statistics import mean

#Reading the csv files

trdata= pd.read_csv("/content/gdrive/My Drive/ML/HW2/trainData.csv")
trlabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/trainLabels.csv")
valdata = pd.read_csv("/content/gdrive/My Drive/ML/HW2/valData.csv")
vallabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/valLabels.csv")
testlabel = pd.read_csv("/content/gdrive/My Drive/ML/HW2/testData_new.csv")

# Removing the extra row at the first

trdata = trdata.iloc[:,1:]
trlabel = trlabel.iloc[:,1:]
valdata = valdata.iloc[:,1:]
vallabel = vallabel.iloc[:,1:]
testlabel = testlabel.iloc[:,1:]
inputX = np.transpose(trdata.values)
output = trlabel.values
wvalues_ridge = []

#Ridge Regression Function

def Ridge1(X,Y,Lambda):
  (k,N) = X.shape
  X2 = np.ones((1,N))
  Xbar = np.vstack((X,X2))
  Xbartran = np.transpose(Xbar)
  Product = Xbar.dot(Xbartran)
  Identity = np.identity(k)
  #print(Identity)
  X0 = np.zeros((k,1))
  X1 = np.zeros((1,k+1))
  IdenHor = np.hstack((Identity,X0))
  IdenVer = np.vstack((IdenHor, X1))
  Mul = np.dot(IdenVer,Lambda)
  Num = np.add(Mul,Product)
  Denom  = np.dot(Xbar,Y)
  Inve = np.linalg.inv(Num)
  #print('Ridge')
  Final = np.dot(Inve,Denom)
  #print('\n')
  #print('The scalar b value')
  b = Final[-1]
  #print('\n')
  #print(b)
  Wval = Final[:-1]
```

```
  #print(Wval)
  wtran = np.transpose(Wval)
  sum1=0
  for i in range(0,N):
    o=X[...,i].ravel()
    otr= np.transpose(o)
    prod= 0
    prod = wtran.dot(otr)
    sum1 = sum1 + prod
  caly= sum1+b[0]
  pred = (caly - Y[i])**2
  WFinal = sum(np.array(Wval)**2)*Lambda
  obj = WFinal + pred
  """print('\n')
  print("Objective Function value")
  print(obj)"""
  Errori=[]
  for i in range(N):
    numerator=  np.transpose(Final).dot(Xbar[...,i].ravel()) - Y[i][0]
    sam= np.transpose(Xbar[...,i].ravel()).dot(Inve)
    sample = sam.dot(Xbar[...,i].ravel())
    denominator = 1 -sample
    Errori.append(numerator/denominator)

  return [Wval,b,obj,Errori]
```

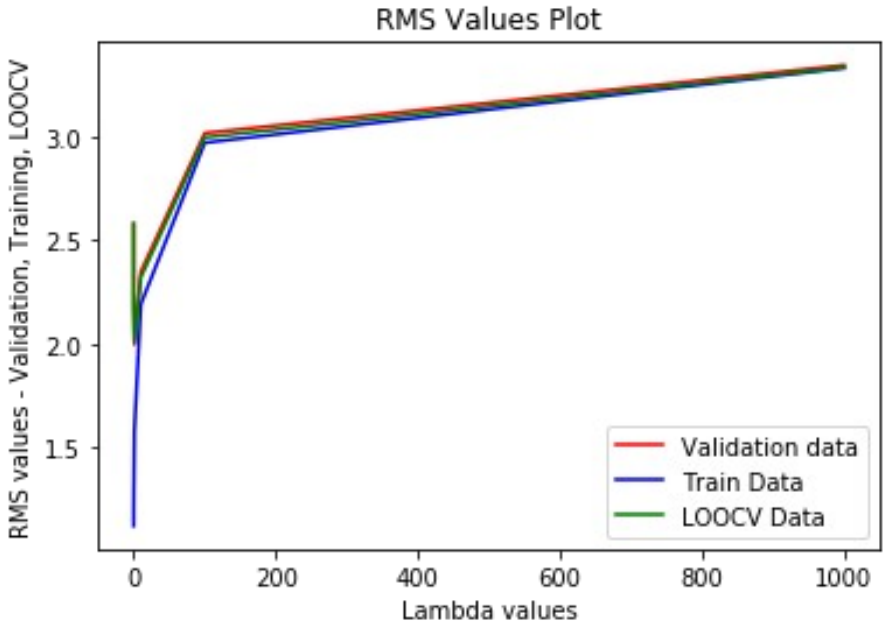
 # To calculate the LOOCV , Validation and Train Label RMS values

```
arrr = []
arrr1 = []
outer_arr = []
for i in [0.01, 0.1, 1, 10, 100, 1000]:
  r1=Ridge1(inputX,output,i)
  Ridge_Model_Coefficient=r1[0]
  Ridge_Model_bvalue=r1[1]
  Ridge_Model_obj = r1[2]
  Ridge_Model_Error = r1[3]
  Ridge_Model_Coefficient
  GenerateVal=[]
  for k in range(0,4748):
    PredictionVal=valdata.values[k].dot(Ridge_Model_Coefficient)+ Ridge_Model_bvalue
    GenerateVal.append(PredictionVal[0])
  RmsVal=[]
  for j in range(0,4748):
    RmsVal.append((GenerateVal[j]-vallabel.values[j])**2)
  arrr.append(np.sqrt(np.mean(RmsVal)))
  locerr = []
```

```
  for t in range(len(Ridge_Model_Error)):
    locerr.append((Ridge_Model_Error[t])**2)
  outer_arr.append(np.sqrt(np.mean(locerr)))
  Generate = []
  for k in range(0,4748):
    Prediction=trdata.values[k].dot(Ridge_Model_Coefficient)+ Ridge_Model_bvalue
    Generate.append(Prediction[0])
  Rms=[]
  for j in range(0,4748):
    Rms.append((Generate[j]-trlabel.values[j])**2)
  arrr1.append(np.sqrt(np.mean(RmsVal)))
arrr
```

#To plot the RMS values

```
fig,graph=plt.subplots()
value = [0.01, 0.1, 1, 10, 100,1000]
graph.plot(value, arrr,'r', value, arrr1,'b', value, outer_arr,'g')
graph.set_xlabel("Lambda values")
graph.set_ylabel("RMS values - Validation, Training, LOOCV")
graph.legend(labels=['Validation data','Train Data','LOOCV Data'])
graph.set_title("RMS Values Plot")
```



3.2.2 Best LOOCV performance

The RMS error values for LOOCV Validation for different Lambdas - [0.01, 0.1, 1, 10, 100, 1000]

```
[2.580745852489428, 2.182597036663108, 2.0097272593728728, 2.3202488051064845,
2.996761416199677, 3.335461543718505]
```

For Lambda = 1, we get the best LOOCV performance(RMSE value -2.0097272593728728)

Objective Function value - 3.36786768e+08

The sum of square errors(For Training Data) - 12450.96272903

```
Value of the regularization term(b- value) – 84.23997165
```

3.2.3

```
Least important of features from low to high :
1. Red Black
2. Medium Body,
3. Russian river valley
4. Chewy
5. Barbeque
6. Superripe
7. Meat
8. Bitter Chocolate
9. Pedigree
10 Judge.
```

Most important feature from high to low
1. Dry good
2. Market
3. Sites
4. Acidity
5. Volume
6. Liqueur
7. Wine Offers
8.Flavours pineapple
9. Drink Soon
10. Oak

```
Looking at the Correlation which is quite high with respect to the ratings of
the Wine, it can be seen that the features with more correlation value have a
higher rating.
```

3.2.4 Kaggle Submission

The Ridge model built has been used to predict the wine goodness points.
Please refer the ML_HW2.ipynb file for the complete code.

M.L - Homework - 2

(1) Parameter Estimation.

1.1 MLE.

The Poisson distribution with parameter λ is

$$P(X=k/\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \quad \text{where } k \in \{0,1,\dots\}$$

To find the log-likelihood of function.

$$P(x|\gamma) = \prod_{i=1}^{N} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} \quad \| \quad L = \ln(\lambda) = \log\lambda \sum x_i + n$$

$$L = \sum_{i=1}^{n} (x_i \log\lambda + (-\lambda) - \log x_i!)$$

$$L(\lambda) = \log\lambda \sum_{i=1}^{n} x_i + (-n\lambda) - \sum_{i=1}^{n} \log x_i!$$

as we want to find the maximum, $L'(\lambda) = 0$

Hence $L'(\lambda) = 0$

$$L'(\lambda) = \frac{\partial L}{\partial \lambda} = \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n - 0$$

$$\frac{1}{\lambda}\sum_{i=1}^{n} x_i - n = 0$$

$$\sum_{i=1}^{n} x_i = n\lambda$$

$$\lambda = \sum_{i=1}^{n} \frac{x_i}{n}$$

$$\underline{\underline{\lambda = \hat{X}.}}$$

1.1 Log likelihood function of X is.

$$L(\lambda) = \log\lambda \sum_{i=1}^{n} x_i + (-n\lambda) - \sum_{i=1}^{n} \log x_i!$$

1.2 MLE for λ in general case

$\lambda = \hat{X}$ = Mean of the distribution

1.3 MLE for λ using observed x

$$\lambda = \hat{x} = \text{Mean of distribution}$$

$$= \frac{4+12+3+5+6+9+10}{7}$$

$$= \frac{49}{7} = 7 //$$

$$\boxed{\lambda = 7}$$

(1.2) MAP.

Gamma distribution has pdf

$$P(\lambda|\alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}, \quad \lambda > 0$$

(1). $P(\lambda)$ - prior density of λ

$P(y|\lambda)$ - likelihood and $P(\lambda|y)$ - posterior

We know that $P(x|\lambda) = \prod_{i=1}^{N} \frac{e^{\lambda} \lambda^{x_i}}{x_i!}$

So to find $P(\lambda|x)$, we use Bayes Theorem.

$$P(\lambda|x) = \frac{P(x|\lambda)\, P(\lambda)}{P(x_1 \ldots x_n)}$$

To find max $P(\lambda|x)$

$$P(\lambda|x) = \text{argmax}\ P(x|\lambda) \cdot P(\lambda).$$

∴ Posterior distribution is.

$$P(\lambda|x) \propto \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}\right)\left(\sum e^{\lambda} \frac{\lambda^{x_i}}{x_i!}\right)$$

$$\propto \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}\right)\left(\frac{e^{-N\lambda} \lambda^{\sum_{i=1}^{N} x_i}}{\prod_{i=1}^{N} x_i!}\right)$$

$$\propto \left(\frac{\beta^{\alpha}}{\Gamma(\alpha) \cdot \prod_{i=1}^{N} x_i!}\right)\left(\lambda^{\alpha-1} e^{-\beta\lambda} \cdot e^{-N\lambda} \cdot \lambda^{\sum_{i=1}^{N} x_i}\right)$$

Here $\sum x_i = N\lambda$ (proved in the previous soln).

$$P(\lambda|y) \propto \underbrace{\left(\frac{\beta^{\alpha}}{\Gamma(\alpha)\prod_{i=1}^{N} x_i!}\right)}_{\text{constant}} \left(\lambda^{\alpha-1+\sum_{i=1}^{N} x_i} e^{-(N+\beta)\lambda}\right)$$

When we see the function we can see that the first bracket is a constant and second bracket is a Gamma distribution with $\hat{\alpha} = \alpha + \Sigma x_i$ and $\hat{\beta} = \beta + N$.

(1.3) Estimator

1. $\hat{\eta} = e^{-2\hat{\lambda}}$ - This is given.

Taking log on both sides

$$\log \hat{\eta} = \log(e^{-2\lambda})$$

$$\log \eta = -2\lambda (\log e) \Rightarrow \lambda = \frac{-1}{2} \log \eta$$

Likelihood distribution is given by

$$P(x|\lambda) = \frac{e^{-\lambda} \lambda^{x_i}}{x_i!} \quad \text{where } x_i = \{0, 1, 2, \ldots n\}.$$

Applying the value of 'λ'

$$P(x|\eta) = \prod_{i=1}^{n} \frac{\left(e^{\eta^{1/2}}\right)\left(\frac{-1}{2}\log \eta\right)^{x_i}}{x_i!}$$

Taking log on both sides

$$\log[P(x|\eta)] = \frac{n}{2}\log \eta + \sum_{i=1}^{n} x_i \log\left(\frac{-1}{2}\log \eta\right) - \log(\Sigma x_i!)$$

As, we want the maximum of loss function, we take the derivative, w.r.t η.

$$\frac{\partial L}{\partial \eta} = \frac{\partial}{\partial \eta}\left[\Sigma x_i \log\left(\frac{-1}{2}\right)\right] + \frac{\partial}{\partial \eta}\left(\sum_{i=1}^{n} x_i \ln(\log \eta)\right) + \frac{n}{2}\frac{\partial}{\partial \eta}\log \eta + \frac{\partial}{\partial \eta}(-\log \Sigma x_i!)$$

($\frac{\partial}{\partial \eta}\log\left(\frac{-1}{2}\right) = 0$, $\frac{\partial}{\partial \eta}(-\log \Sigma x_i!) = 0$)

$$\frac{\partial L}{\partial \eta} = \frac{1}{\log \eta}\left(\frac{1}{\eta}\right)\sum_{i=1}^{n} x_i + \frac{n}{2\eta}$$

$$\frac{1}{\log \eta}\left(\frac{1}{\eta}\right)\Sigma x_i + \frac{n}{2\eta} = 0.$$

$$\frac{1}{\log \eta}\left(\frac{1}{\eta}\right)\eta^{\bar{x}} = \frac{-1}{2}$$

$$\frac{\bar{x}}{\log \eta} = \frac{-1}{2}$$

$$\log \eta = -2\bar{x}$$

∴ Taking antilog

$$\eta = e^{-2\bar{x}}$$

(2) Here $P(x|\theta)$ – is the probability function and θ – estimator based on any observed data 'x'

$$\text{Bias}(\hat{\theta}) = E(\hat{\theta}) - \theta.$$

Here for η – estimator

$$\text{Bias} = E(\hat{\eta}) - \eta$$

Here $E(\eta) = \sum \eta f(\eta) = \sum \eta \cdot \frac{\lambda^x e^{-\lambda}}{x!}$

Here $\eta = e^{-2x}$

$$\therefore E(\eta) = \sum_{x=1}^{N} \frac{e^{-2x}\,\lambda^x\, e^{-\lambda}}{x!}$$

$$= e^{-\lambda} \sum_{x=1}^{N} \frac{e^{-2x}\,\lambda^x}{x!} \quad \text{—(1)}$$

We have to apply the Taylor Expansion.

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Hence (1) is re-written as

$$E(\eta) = e^{-\lambda}\, e^{-2\lambda} \;//$$

Now we find the bias.

$$\text{Bias} = E(\eta) - \eta$$

$$= e^{-\lambda} e^{+\frac{1}{e^2}\lambda} - \eta = e^{(1/e^2 - 1)\lambda} - \eta$$

$$= e^{(1/e^2 - 1)\lambda} - e^{-2\lambda} \quad //.$$

(3) For unbiased condition

$$\text{Bias} = 0$$

$$\therefore E(\eta) - \eta = 0$$

$$E(\eta) = \sum_x \eta \cdot P(x) = \sum \eta \cdot \frac{e^{-\lambda}\lambda^x}{x!}$$

As $\eta = -1$

$$= \sum_{x=0}^{N} (-1)^x \frac{e^{-\lambda}\lambda^x}{x!} = \sum_x \frac{(-1\lambda)^x e^{-\lambda}}{x!}$$

$$= e^{-\lambda} \sum \frac{(-\lambda)^x}{x!} \quad \left(\text{By Taylor expansion } e^x = \sum_{k=0} \frac{x^k}{k!}\right)$$

$$= e^{-\lambda} e^{-\lambda} = e^{-2\lambda} \quad //.$$

We know $\eta = e^{-2\lambda}$ Hence bias = 0 $(\because E(\eta) = \eta)$.

Bias $(-1)^x$ is the best estimator because we see that for every value of x from 0 to N, we see that the values of estimator are alternating. And hence the predictions are both negative and positive. So when we take it over the distribution, then we get the overall bias = 0.