zd2221

P1

(a) $f(x_1, x_2, \cdots x_n) = \boxed{\prod_{i=1}^{N} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda}}$

(b) $l(x) = \log f(x_1, x_2, \cdots x_n) = \sum_{i=1}^{n} x_i \log\lambda - \log x_i! - n\lambda$

$\frac{\partial l(x)}{\partial\lambda} = 0 \qquad \frac{\sum_{i=1}^{n} x_i}{\lambda} - n = 0$

$\boxed{\lambda_{MLE} = \frac{\sum_{i=1}^{n} x_i}{n}}$

(c) $p(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda} \qquad \lambda \sim gamma(a, b)$

$\lambda_{map} = \arg\max \ln p(\lambda | x)$

$= \arg\max \ln \frac{p(x|\lambda) p(\lambda)}{p(x)}$

$= \arg\max(\ln(p(x|\lambda) + \ln\lambda - \ln(x))$

$\ln p(x|y) + \ln p(\lambda)$

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

$\frac{\partial(\ln p(x|y) + \ln p(\lambda))}{\partial\lambda} = \frac{\sum_{i=1}^{n} x_i}{\lambda} - n + \frac{a-1}{\lambda} - b = 0$

$\boxed{\lambda_{map} = \frac{a + \sum_{i=1}^{n} x_i - 1}{b+n}}$

(d) $p(\lambda|x) = p(x|\lambda) p(\lambda) = \frac{\prod_{i=1}^{n} \lambda^{x_i} e^{-\lambda}}{\prod_{i=1}^{n} x_i!} \times \frac{b^a \lambda^{a-1} e^{-b\lambda}}{\Gamma(a)}$

$= \frac{\lambda^{\sum_{i=1}^{n} x_i + a - 1} e^{-n\lambda - b\lambda} b^a}{\prod_{i=1}^{n} x_i! \; \Gamma(a)}$

$\boxed{gamma\left(\sum_{i=1}^{n} x_i + a, \; b+n\right)}$

(e)

$$P(\lambda|X) = \text{Gamma}\left(\sum_{i=1}^{n} x_i + a,\ b+n\right)$$

$$\text{mean} = \frac{\sum_{i=1}^{n} x_i + a}{b+n}$$

$$\text{Variance} = \frac{a + \sum_{i=1}^{n} x_i}{(b+n)^2}$$

Relation with λ_{ML} and λ_{MAP}:

(1) If we set $a=b=0$ mean of $P(\lambda|X)$ equals λ_{MLE}
The mean of posterior is highly related to MLE. but is modified by prior as well

(2) MAP is the mode of $P(\lambda|X)$ λ_{MAP} will maximize $P(\lambda|X)$ and the mode of $P(\lambda|X)$ is the value at which $P(\lambda|X)$ takes its maximum value

P2

- Ridge regression

$y \sim N(XW, \sigma^2 I) \quad w \sim N(0, \lambda^{-1} I)$

$W_{RR} = (\lambda I + X^T X)^{-1} X^T y$

$E[W_{RR}] = (\lambda I + X^T X)^{-1} X^T E[Y]$

$E[y] = XW$

$\boxed{E[W_{RR}] = (\lambda I + X^T X)^{-1} X^T X W}$

$Var[W_{RR}] = E[(W_{RR} - E[W_{RR}])(W_{RR} - E[W_{RR}])^T]$

$= E[W_{RR} W_{RR}^T] - E[W_{RR}] E[W_{RR}]^T$

$= E[(\lambda I + X^T X)^{-1} X^T y y^T X (X^T X + \lambda I)^{-1}] -$

$(\lambda I + X^T X)^{-1} X^T X W (XW)^T X (\lambda I + X^T X)^{-1}$

plug in $E[yy^T] = \sigma^2 I + X w w^T X^T$

- $Var[W_{RR}] = (\lambda I + X^T X)^{-1} X^T (\sigma^2 I + X w w^T X^T) X (X^T X + \lambda I)^{-1} -$

$(\lambda I + X^T X)^{-1} X^T X w w^T X^T X (\lambda I + X^T X)^{-1}$

$= \boxed{(\lambda I + X^T X)^{-1} X^T \sigma^2 I X (X^T X + \lambda I)^{-1}}$

MLhw1

February 13, 2019

In [89]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Import data
train_x=pd.read_csv("hw1-data/X_train.csv",header=None)
train_y=pd.read_csv("hw1-data/Y_train.csv",header=None)
test_x=pd.read_csv("hw1-data/X_test.csv",header=None)
test_y=pd.read_csv("hw1-data/Y_test.csv",header=None)
```

In [66]:
```
# Scale the data standadize normalizatoin ????
# we do not touch 1 because this is design matrix
#train_x.loc[:,:5]=(train_x.loc[:,:5]-train_x.loc[:,:5].mean())/train_x.loc[:,:5].std
#test_x.loc[:,:5]=(test_x.loc[:,:5]-train_x.loc[:,:5].mean())/train_x.loc[:,:5].std()
```

In [67]:
```
# calculate the svd of train_X matrix
(u, s, vh)=np.linalg.svd(train_x.values,full_matrices=False)
# calculate the df(lamda)
```

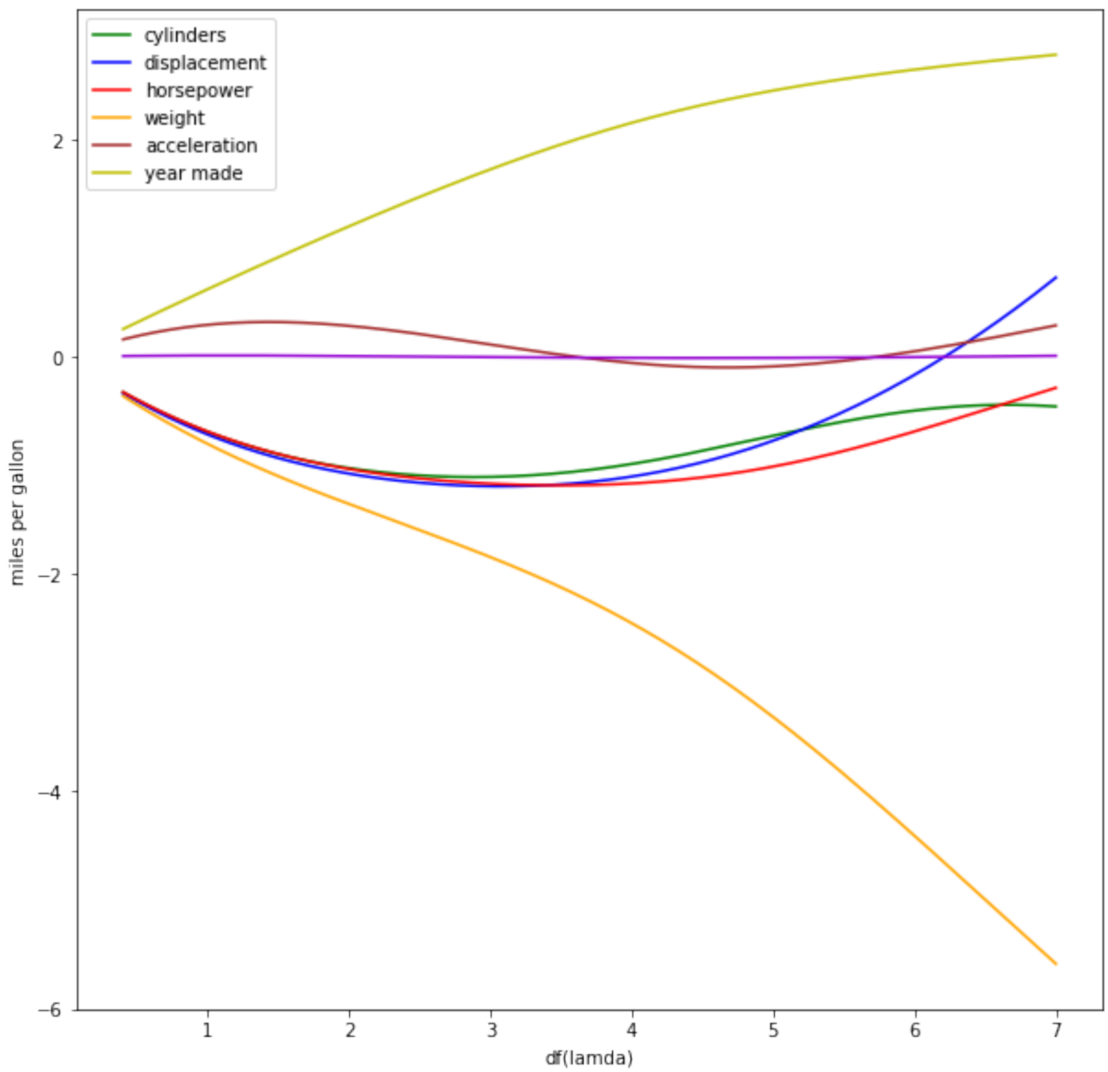
In [68]:
```
ans=[sum(np.square(s)/(np.square(s)))]
weight=np.linalg.inv(np.dot(train_x.transpose(),train_x)).dot(train_x.transpose()).do
for lamda in range(1,5000):
    ans.append(sum(np.square(s)/(np.square(s)+lamda)))
    weight=np.hstack((weight,np.linalg.inv(np.dot(train_x.transpose(),train_x)+lamda*
```

0.1 (a) For = 0,1,2,3,...,5000, solve for wRR. (Notice that when= 0, wRR=wLS.) In one figure, plot the 7 values inwRRas a function of df(). You will need to call a built in SVD function to do this as discussed in the slides. Be sure to label your 7 curves by their dimension in x2

In [69]:
```
plt.figure(figsize=(10,10))
line1,=plt.plot(ans,weight[0,:],'g')
line2,=plt.plot(ans,weight[1,:],'b')
line3,=plt.plot(ans,weight[2,:],'r')
line4,=plt.plot(ans,weight[3,:],'orange')
line5,=plt.plot(ans,weight[4,:],'brown')
line6,=plt.plot(ans,weight[5,:],'y')
line7,=plt.plot(ans,weight[6,:],'darkviolet')
```

```
plt.gca().legend((line1,line2,line3,line4,line5,line6),("cylinders","displacement","ho
plt.ylabel('miles per gallon')
plt.xlabel('df(lamda)')
#plt.show()
```

Out[69]: Text(0.5, 0, 'df(lamda)')



0.2 (b) Two dimensions clearly stand out over the others. Which ones are they and what information can we get from this?

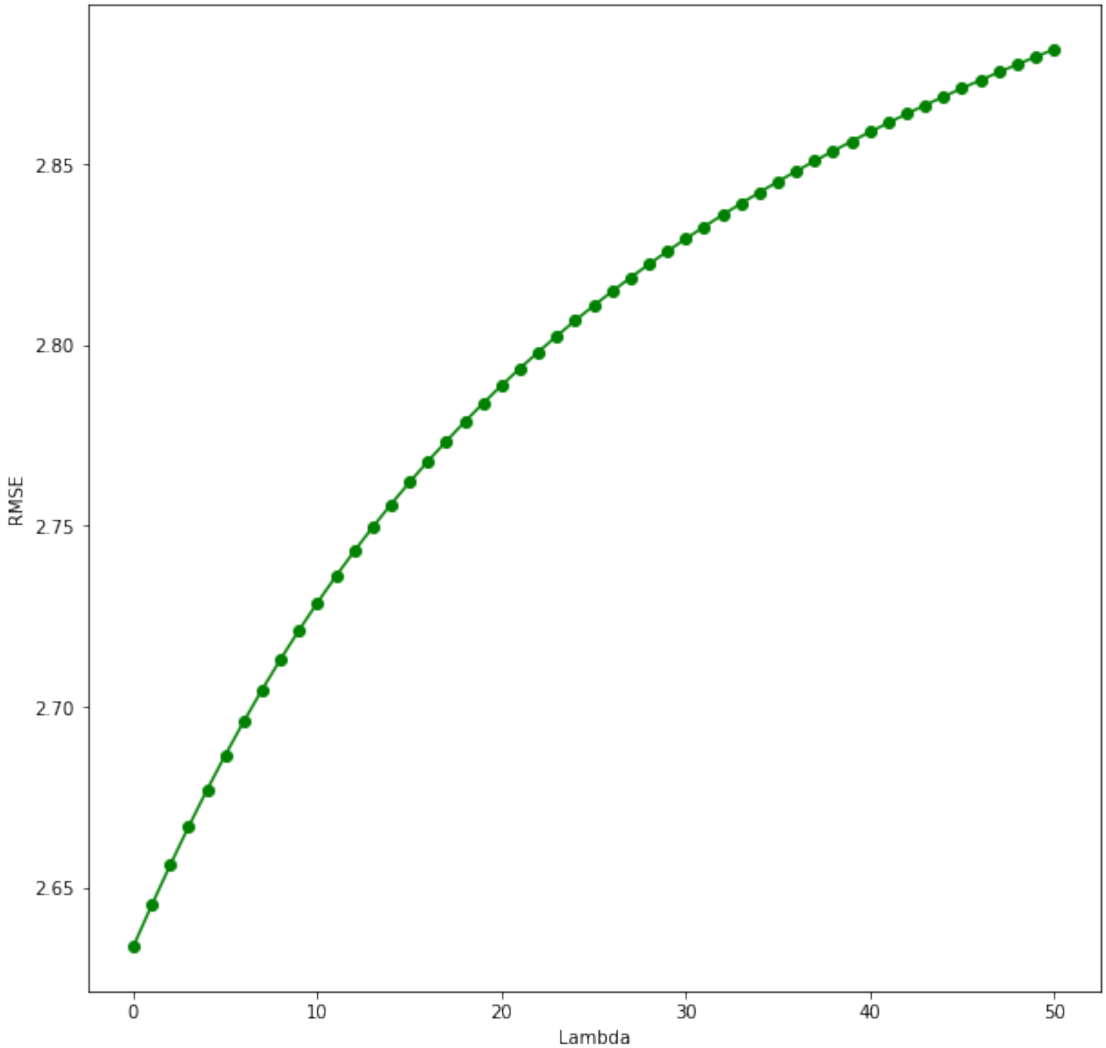
Feature "Year" and "Weight" stand out over the others, because they have higher weight and shink slower than the others. From the picture above, we guess the response "miles per gallon" is positively related to the age of cars, and negatively related to the weight of cars.

0.3 (c) For= 0,...,50, predict all 42 test cases. Plot the root mean squared error (RMSE)3 on the testset as a function of —not as a function of df(). What does this figure tell you when choosing for this problem (and when choosing between ridge regression and least squares)?

From the picture below, we can observe that the rmse increases as lamda increases, which means when we add more penalization to the weights, the accuracy drops. The model may suffer from high bias when adding to much penalty to the cost function. Thusm, we need to be careful with the lambda. If the number of features is not large, and the model is relatively simple, adding penalty will hurt the performance of the model.

```
In [70]: #For= 0,...,50, predict all 42 test cases. Plot the root mean squared error (RMSE)3on

         def RMSE(y_predict,y_true):
             return np.sqrt(np.mean(np.square(y_predict-y_true),axis=0))

         y_predict=np.dot(test_x,weight[:,:51])
         #y_predict-test_y.values
         R1=RMSE(y_predict,test_y.values)
         # plot RMSE OF Lamda
         plt.figure(figsize=(10,10))
         plt.ylabel('RMSE')
         plt.xlabel('Lambda')
         plt.plot(range(0,51),R1,'-go')
```

```
Out[70]: [<matplotlib.lines.Line2D at 0x21901a1d0>]
```



1 Part 2

In [73]:
```
# order=1
y_predict=np.dot(test_x,weight[:,:101])
#y_predict-test_y.values
R1=RMSE(y_predict,test_y.values)
```

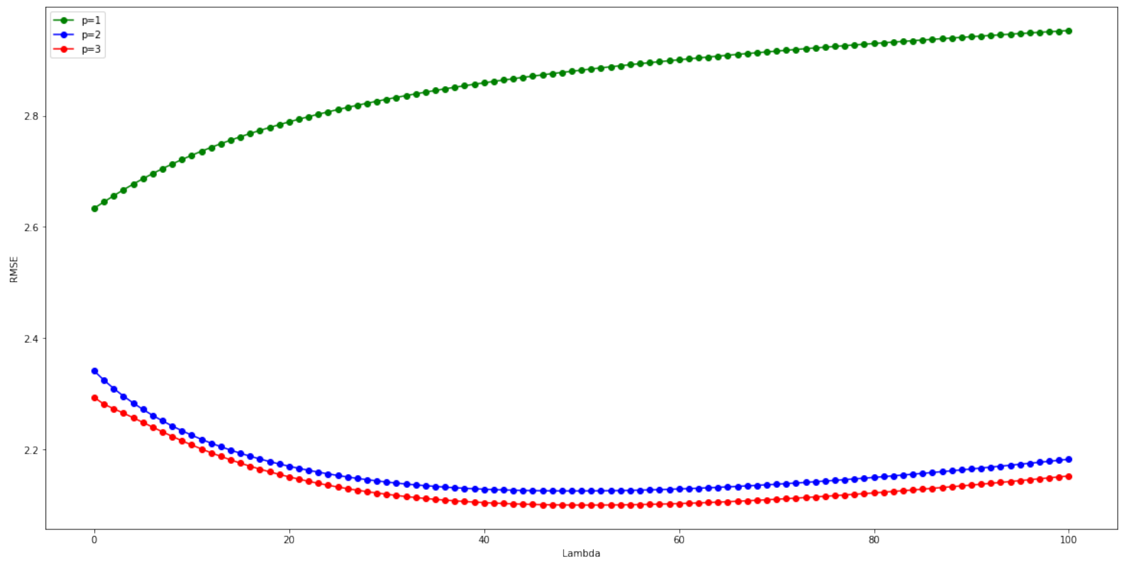
In [113]:
```
# Import data
train_x=pd.read_csv("hw1-data/X_train.csv",header=None)
train_y=pd.read_csv("hw1-data/Y_train.csv",header=None)
test_x=pd.read_csv("hw1-data/X_test.csv",header=None)
test_y=pd.read_csv("hw1-data/Y_test.csv",header=None)

# order=2
```

```
# add higher order features
for i in range(6,12):
    train_x.loc[:,i]=train_x.loc[:,i-6]**2
    test_x.loc[:,i]=test_x.loc[:,i-6]**2
# scale the data
mean2=train_x.loc[:,6:11].mean()
std2=train_x.loc[:,6:11].std()
train_x.loc[:,6:11]=(train_x.loc[:,6:11]-mean2)/std2
test_x.loc[:,6:11]=(test_x.loc[:,6:11]-mean2)/std2
train_x.loc[:,12]=1
test_x.loc[:,12]=1

weight2=np.linalg.inv(np.dot(train_x.transpose(),train_x)).dot(train_x.transpose()).c
for lamda in range(1,101):
    weight2=np.hstack((weight2,np.linalg.inv(np.dot(train_x.transpose(),train_x)+lamd

y_predict=np.dot(test_x,weight2)
#y_predict-test_y.values
R2=RMSE(y_predict,test_y.values)
```

In [114]:
```
# order=3


# add higher order features
for i in range(12,18):
    train_x.loc[:,i]=train_x.loc[:,i-12]**3
    test_x.loc[:,i]=test_x.loc[:,i-12]**3
# scale the data

mean3=train_x.loc[:,12:17].mean()
std3=train_x.loc[:,12:17].std()
train_x.loc[:,12:17]=(train_x.loc[:,12:17]-mean3)/std3
test_x.loc[:,12:17]=(test_x.loc[:,12:17]-mean3)/std3
train_x.loc[:,18]=1
test_x.loc[:,18]=1

weight3=np.linalg.inv(np.dot(train_x.transpose(),train_x)).dot(train_x.transpose()).c
for lamda in range(1,101):
    weight3=np.hstack((weight3,np.linalg.inv(np.dot(train_x.transpose(),train_x)+lamd

y_predict=np.dot(test_x,weight3)
#y_predict-test_y.values
R3=RMSE(y_predict,test_y.values)
```

In [115]:
```
#plt.figure(figsize=(10,10))
#plt.plot(range(0,101),R3,'go')

plt.figure(figsize=(20,10))
```

```
line1,=plt.plot(range(101),R1,'-go')
line2,=plt.plot(range(101),R2,'-bo')
line3,=plt.plot(range(101),R3,'-ro')
plt.gca().legend((line1,line2,line3),("p=1","p=2","p=3"))
plt.ylabel('RMSE')
plt.xlabel('Lambda')
plt.show()
```



1.1 (d) In one figure, plot the test RMSE as a function of = 0,...,100 for p= 1,2,3. Based on this plot, which value of p should you choose and why? How does your assessment of the ideal value of change for this problem?

I will choose p=3, because when p equals 3, the model has smallest rmse on test set. We also notice, the rmse has the smallest value around 25 when p equal 2 or 3. We can conclude when the number of features are large and the model is complex, adding penalty will improve the generalization of model.