Statistical machine learning HW1

Q1. Bias-variance decomposition

(a) $N(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

$P(x;\mu) \propto \exp(-\frac{1}{2}(x-\mu)^T I_p^{-1}(x-\mu))$; $x \sim N(\mu, I_p)$

If take out the constants,

I will get $P(x;\mu) \propto \exp(-\frac{1}{2}\|x-\mu\|^2)$

$\log P(x;\mu) \propto -\frac{1}{2}\|x-\mu\|^2$.

$\frac{\partial}{\partial\mu}\log P(x;\mu) \Rightarrow \hat{\mu}_{MLE} = x \in \mathbb{R}^{p\times 1}$

$Bias(\hat{\mu}_{MLE}) = E[\hat{\mu}_{MLE} - \mu] = E[x-\mu] = 0$

$Var(\hat{\mu}_{MLE}) = Var(x) = I_p$

Thus, mean squared error

$E[\|\hat{\mu}_{MLE} - \mu\|^2] = \|Bias(\hat{\mu}_{MLE})\|^2 + Trace(Var(\hat{\mu}_{MLE}))$

$= 0 + p = p$

(b) $\log P(x;\mu) - \lambda\|\mu\|^2 = -\frac{1}{2}\|x-\mu\|^2 - \lambda\|\mu\|^2$

$Loss = -(\log P(x;\mu) - \lambda\|\mu\|^2) = \frac{1}{2}\|x-\mu\|^2 + \lambda\|\mu\|^2$

$\frac{\partial Loss}{\partial\mu} = \frac{1}{2}\cdot 2(\mu - x) + 2\lambda\mu = (2\lambda+1)\mu - x$

$\hat{\mu}_{RMLE} = \frac{x}{2\lambda+1}$

$Bias(\hat{\mu}_{RMLE}) = E[\hat{\mu}_{RMLE}] - \mu = \frac{1}{2\lambda+1}E[x] - \mu$

$= -\frac{2\lambda\mu}{2\lambda+1}$

$Var(\hat{\mu}_{RMLE}) = Var(\frac{x}{2\lambda+1}) = \frac{1}{(2\lambda+1)^2}Var(x) = \frac{I_p}{(2\lambda+1)^2}$

Mean square error is thus

$E[\|\hat{\mu}_{RMLE} - \mu\|^2] = \|Bias(\hat{\mu}_{RMLE})\|^2 + Trace(Var(\hat{\mu}_{RMLE}))$

(c) $\hat{\mu}_{SCALE} = c^* x$

c^* is chosen to minimize $E[\|cx-\mu\|^2]$

$Loss = E[\|cx-\mu\|^2] = E[\|c(x-\mu) + (c-1)\mu\|^2]$

$= E[\|c(x-\mu)\|^2 + 2c(c-1)\mu^T(x-\mu) + (c-1)^2\|\mu\|^2]$

$= c^2 E[\|x-\mu\|^2] + 2c(c-1)\mu^T E[x-\mu] + (c-1)^2\|\mu\|^2$

$= c^2 Trace(Var(x)) + (c-1)^2\|\mu\|^2$

$= pc^2 + (c-1)^2\|\mu\|^2$

$\frac{\partial Loss}{\partial c} = 2pc + 2(c-1)\|\mu\|^2 \Rightarrow c^*$

$Bias(\hat{\mu}_{SCALE}) = -\frac{p\mu}{p+\|\mu\|^2}$

Corresponding minimum mean square error is

$E[\|\hat{\mu}_{SCALE} - \mu\|^2] = \frac{p\|\mu\|^2}{p+\|\mu\|^2}$

(d)

i. $\hat{\mu}_{JS} = (1 - \frac{p-2}{\|x\|^2})x$

$E[\|x-\mu\|^2] = 2E[(x-\mu)^T g(x)] + p$

ii. $E\|g(x)\|^2 = E[\|\frac{p-2}{\|x\|^2}x\|^2] = E[\frac{(p-2)^2}{\|x\|^4}\|x\|^2] = (p-2)^2 E[\|x\|^{-2}]$

iii. $E[(x-\mu)^T g(x)] = E[\sum_{j=1}^{p}\frac{\partial}{\partial x_j}g_j(x)]$

$g_j(x) = \frac{p-2}{\|x\|^2}x_j = \frac{p-2}{\sum_{j=1}^{p}x_j^2}\cdot x_j$

$\frac{\partial}{\partial x_j}g_j(x) = \frac{p-2}{\sum_{j=1}^{p}x_j^2} - \frac{(p-2)\cdot x_j\cdot(\frac{\partial}{\partial x_j}\sum_{j=1}^{p}x_j^2)}{(\sum_{j=1}^{p}x_j^2)^2}$ $\Rightarrow$ holds

iv. $E[\|\hat{\mu}_{js} - \mu\|^2] = p - (p-2)^2 E[\|X\|^{-2}]$

e. $E[\|\hat{\mu}_{MLE} - \mu\|^2] = p$

$E\|\hat{\mu}_{RIDGE} - \mu\|^2 = \frac{4\lambda^2\|\mu\|^2 + p}{4\lambda^2 + 4\lambda + 1}$

$E[\|\hat{\mu}_{SCALE} - \mu\|^2] = \frac{p\|\mu\|^2}{p + \|\mu\|^2}$

$E[\|\hat{\mu}_{js} - \mu\|^2] = p - (p-2)^2 E[\|X\|^{-2}]$

I can see that, only the MLE gives a mean squared error equals to p. All other measurements have value close to but smaller than p.

Which means that unbiasedness is given up to achieve for smaller mean squared error.

If I would choose, I would choose MLE estimator to estimate something new. p just restricts the maximum of these four estimators.

Q2. (a) $q = 0.5$ (left) one of l_q-regression encourages sparse estimates. Since $q=0$ and $q=1$ encourages sparse estimates. $q = 0.5$, which resides between 0 and 1, necessarily gives sparse estimates.

We can also see from the graphs that unregularized least squares solution has only one contact point with $l_{0.5}$ form

(b) For $q = 0.5$, X_3 is the point which would achieve the smallest cost under $L_{0.5}$-constrained least squares cost function. This is because X_3 could give a smallest $L_{0.5}$ form on the graph.

For $q=4$, X_4 is the point which would achieve the smallest cost under L_4-constrained least squares cost function. This is because X_4 could give a smallest L_4 form on the graph.

(c) For $q = 0.5$

Loss Function is

$\text{Loss} = \|y - \phi(x)\beta\|_2^2 + \lambda\|\beta\|_{0.5}$

For $q = 4$,

Loss Function

$\text{Loss} = \|y - \phi(x)\beta\|_2^2 + \lambda\|\beta\|_4$

$q=4$ is easier to calculate since $q=0.5$ has no closed form and is hard to solve

Q3 (a) prior $p(w) \propto \prod_{j=0}^{p} \exp(-\frac{1}{2}(w_j - u)^2), u \in \mathbb{R}$.

likelihood data, $i = 1, \cdots, n$.

$$L(w) = \prod_{i=1}^{n} (P[Y_i = 1 | X = x_i])^{Y_i} (P[Y_i = 0 | X = x_i])^{1-Y_i}$$

$$\propto \prod_{i=1}^{n} \left(\frac{\exp(w_0 + x_i^T w)}{\exp(w_0 + x_i^T w) + 1} \right)^{Y_i} \left(\frac{1}{\exp(w_0 + x_i^T w) + 1} \right)^{1-Y_i}$$

posterior: $P(w) L(w) = \prod_{j=0}^{p} \exp(-\frac{1}{2}(w_j - u)^2) \cdot \prod_{i=1}^{n} \frac{\exp(Y_i(w_0 + x_i^T w))}{\exp(w_0 + x_i^T w) + 1}$

log-posterior $\propto -\frac{1}{2}\sum_{j=0}^{p} (w_j - u)^2 + \sum_{i=1}^{n} Y_i(w_0 + x_i^T w) - \log(\exp(w_0 + x_i^T w) + 1)$

$\sum^{n}$ $\exp(w_0 + x_i^T w)$

(b) $\frac{\partial f}{\partial w_0} = -(w_0 - u) + \sum_{i=1} Y_i - \frac{1}{\exp(w_0 + X^T w) + 1}$

let $u - w_0 + \sum Y_i - P(Y=1|X=x) = 0$.

I can get $\hat{w_0} = \sum y_i + u - P(Y=1|X=x)$

$\frac{\partial f}{\partial w_i} = -(w_i - u) + \sum Y_i x_i - \frac{x_i \exp(w_0 + x_i^T w)}{\exp(w_0 + x_i^T w) + 1}$

let $u - w_i + \sum Y_i x_i - x_i P(Y=1|X=x) = 0$

$\hat{w_i} = \sum Y_i x_i - x_i P(Y=1|X=x) + u$

$\frac{\partial f}{\partial u} = \sum (w_j - u) = 0$. $\quad \hat{u} = \frac{1}{p} \sum w_i$

to get MAP, I need to set $\frac{\partial L}{\partial w_0}$ and $\frac{\partial f}{\partial w_i}$ for $i = \{1, 2, \dots n\} = 0$,

Q4 (a) $y_i = x_i^T B + \xi_i$, $\xi_i \sim N(0, 6^2)$, 6^2 is known.

$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^{n \times 1}$, $X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} \in \mathbb{R}^{n \times d}$, $\xi = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_N \end{bmatrix} \in \mathbb{R}^{n \times 1}$

$Y = XB + \xi \quad \xi = Y - XB \sim N(0, 6^2 I_N)$

$L(B) \propto \exp(-\frac{1}{26^2} \|Y - XB\|^2)$

$= \exp(-\frac{1}{26^2} \sum_{i=1}^{N} (y_i - x_i^T B)^2)$

(b) $L(B) \propto \exp(-\frac{1}{26^2} \|Y - XB\|^2)$

$\max L(B) \Leftrightarrow \max \log L(B) = \max(-\frac{1}{26^2} \|Y - XB\|^2)$

$\Leftrightarrow \min \frac{1}{26^2} \|Y - XB\|^2$

take out constant 6

$\Leftrightarrow \min \frac{1}{2} \|Y - XB\|^2$

(c) To get posterior: prior $p(B) \propto \exp(-\frac{\lambda}{6^2} \|B\|^2)$

likelihood $L(B) \propto \exp(-\frac{1}{26^2} \|Y - XB\|^2)$

posterior $\propto$ prior $\cdot$ likelihood $= \exp(\frac{-1}{6^2}(\frac{1}{2}\|Y - XB\|^2 + \lambda \|B\|^2))$

This is same as ridge regression

(d) $\hat{B}_{MLE} = (X^T X)^{-1} X^T Y$.

when $d >> N$, $X^T X$ is not invertible.

then nen have $\hat{B}_{MLE}$.

$\hat{B}_{MAP} = (X^T X + \lambda I)^{-1} X^T Y$

$X^T X + \lambda I$ is always invertible

So, always unique

In [1]:

```
import pandas as pd
import numpy as np
import sklearn
from sklearn import linear_model as lm
import matplotlib.pyplot as plt
```

In [2]:

```
X = pd.read_csv("hw1_Q5_X.txt", sep = " ")
```

In [3]:

```
Y = pd.read_csv("hw1_Q5_Y.txt")
```

In [4]:

```
X.shape
```

Out[4]:

```
(2303, 16384)
```

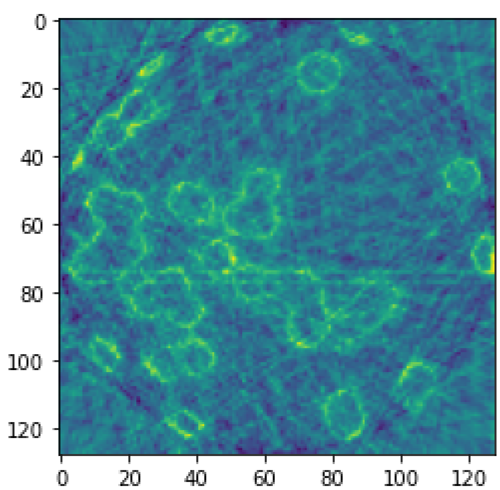
In [5]:

```
alpha = {0.000001, 0.0001, 0.01, 0.1, 1}
```

In [6]:

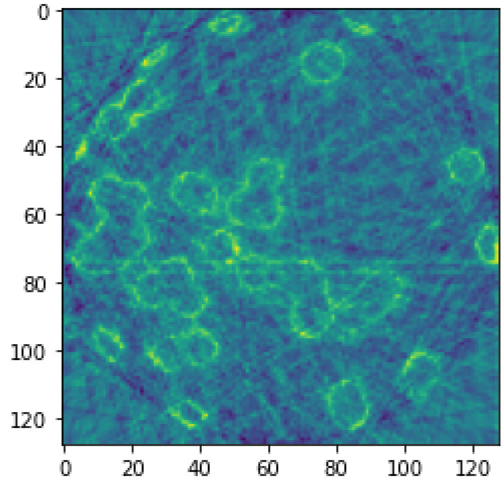
```
for i in alpha:
    reg = lm.Ridge(alpha = i)
    reg.fit(X, Y)
    print("alpha equals ",  i)
    print("coefficients equal to ", reg.coef_)
    print("intercept is ", reg.intercept_)
    print("This is graph for penalty value equals", i)
    plt.imshow(reg.coef_.reshape(128,128))
    plt.show()
```

```
alpha equals  0.1
coefficients equal to  [[ 0.01066459 -0.06157589 -0.13089499 ... -0.08670359 -0.0704923
   0.03089961]]
intercept is  [4.13412288]
This is graph for penalty value equals 0.1
```

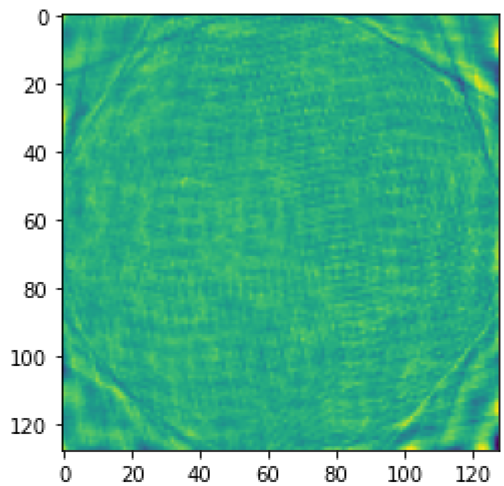



```
alpha equals  1
coefficients equal to  [[ 0.03331101 -0.04932843 -0.12765885 ... -0.090218   -0.07620072
   0.02076371]]
```

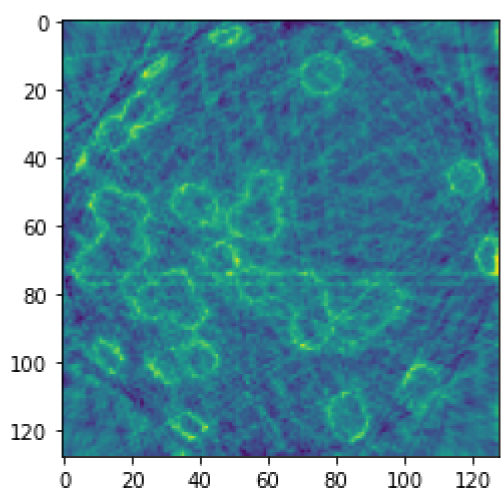
intercept is [4.19743864]
This is graph for penalty value equals 1



alpha equals 0.0001
coefficients equal to [[0.96972773 0.3988277 -0.04346479 ... 0.61388721 -0.71329611
  -2.40470515]]
intercept is [4.16473732]
This is graph for penalty value equals 0.0001

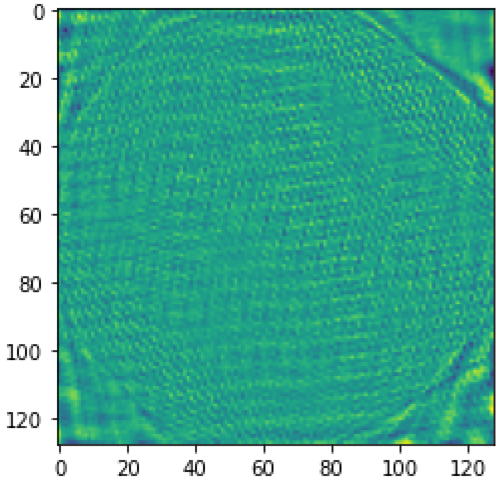


alpha equals 0.01
coefficients equal to [[-0.07456965 -0.07905594 -0.09351187 ... -0.05410599 -0.06472962
   0.00641405]]
intercept is [4.07484643]
This is graph for penalty value equals 0.01



alpha equals 1e-06

```
coefficients equal to  [[ 4.54100062 -5.73805794 -6.49401266 ...  3.71282832 -2.23877009
  -3.88228218]]
intercept is  [4.42025131]
This is graph for penalty value equals 1e-06
```



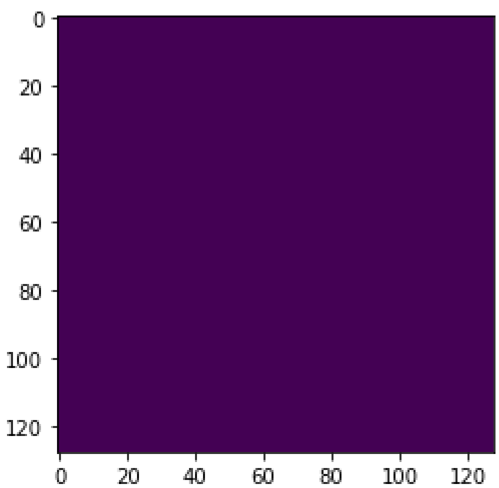
5.a

It seems that when alpha equals 0.01, 0.1 and 1, the plot will give better results than other penalty terms.

In [7]:

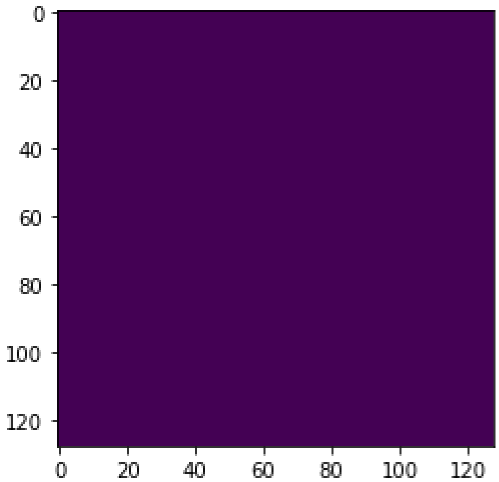
```
for i in alpha:
    reg = lm.Lasso(alpha = i)
    reg.fit(X, Y)
    print("alpha equals ",  i)
    print("coefficients equal to ", reg.coef_)
    print("intercept is ", reg.intercept_)
    print("This is graph for penalty value equals", i)
    plt.imshow(reg.coef_.reshape(128,128))
    plt.show()
```

```
alpha equals  0.1
coefficients equal to  [ 0. -0. -0. ... -0. -0. -0.]
intercept is  [5.79205569]
This is graph for penalty value equals 0.1
```

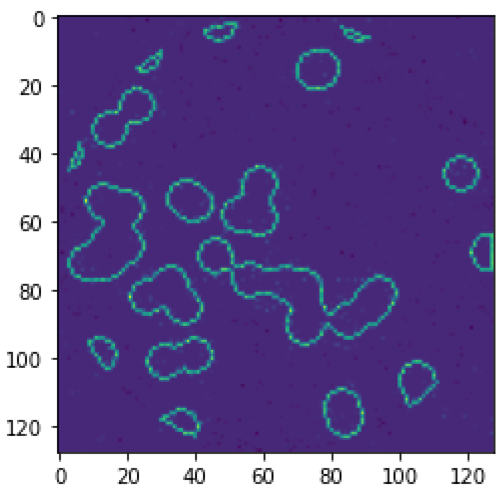


```
alpha equals  1
coefficients equal to  [ 0. -0. -0. ... -0. -0. -0.]
```

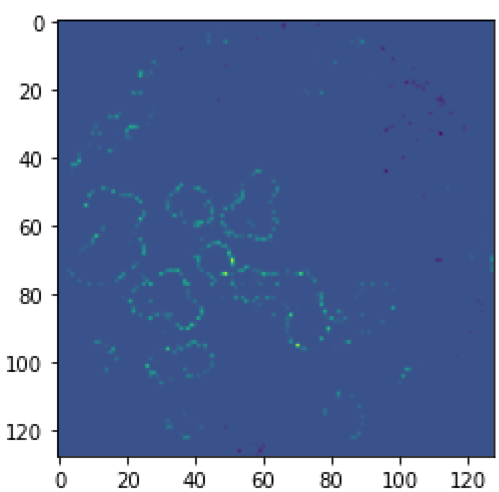
```
intercept is  [5.79205569]
This is graph for penalty value equals 1
```




```
alpha equals  0.0001
coefficients equal to  [-0. -0. -0. ... -0. -0. -0.]
intercept is  [0.38815123]
This is graph for penalty value equals 0.0001
```



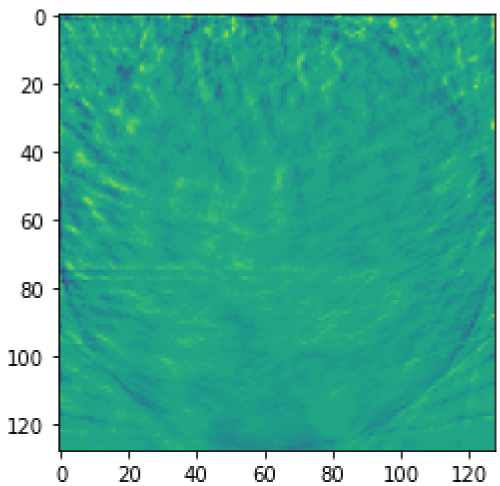

```
alpha equals  0.01
coefficients equal to  [ 0. -0. -0. ... -0. -0. -0.]
intercept is  [4.22962642]
This is graph for penalty value equals 0.01
```




```
alpha equals  1e-06
coefficients equal to  [ 0.56821171 -0.04214576 -0.76743992 ... -0.         -0.
 -0.        ]
```

```
intercept is  [11.26534154]
This is graph for penalty value equals 1e-06
```

```
C:\Users\edaoy\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:5
30: ConvergenceWarning: Objective did not converge. You might want to increase the numbe
r of iterations. Duality gap: 4.784204161011711, tolerance: 2.9338255001032407
  model = cd_fast.enet_coordinate_descent(
```



5.b

It seems that when peanlty term equals 0.0001, the plot yields the most results.

5.c

The results from Ridge regression seems to contain more points than Lasso regression. This makes sense because Lasso selects the most useful features and will produce more zero results tha Ridge regression