HW5-Q3

November 14, 2019

[1]:
```
# -*- coding: utf-8 -*-

from sklearn import preprocessing
from sklearn import linear_model

import numpy as np


class graphicalLasso:

    def __init__(self, rho=0.1, tol=1e-30):

        self.rho = rho
        self.tol = tol

    # graphical lasso
    def fit(self,X):
        criterion_dual = []
        criterion_prim = []
        n_samples,n_features = X.shape[0],X.shape[1]

        self.scaler=preprocessing.StandardScaler().fit(X)
        self.X=self.scaler.transform(X)

        S = self.X.T.dot(self.X)/n_samples



        invA = S + self.rho*np.eye(S.shape[0],S.shape[1])
        A = np.linalg.pinv(invA)
        A_old = A

        criterion_prim.append(-np.log(np.linalg.det(A)) + np.trace(np.dot(S,A))␣
↪+ self.rho*np.sum(np.abs(A)))
        criterion_dual.append(np.log(np.linalg.det(invA)) + n_features)

        while True:
```

```
            for j in range(n_features):
                R,s,sjj = self.__get(S,j)
                W = self.__get(invA,j)[0]

                beta_0 = np.zeros(W.shape[1])
                beta = self.coord_lasso_solve(W,s,self.tol,beta_0,self.rho)

                w = -W.dot(beta)

                invA = self.__put(W,w,sjj+self.rho,j)
                A = np.linalg.pinv(invA)

                criterion_prim.append(-np.log(np.linalg.det(A)) + np.trace(np.
↪dot(S,A)) + self.rho*np.sum(np.abs(A)))
                criterion_dual.append(np.log(np.linalg.det(invA)) + n_features)

            if np.linalg.norm(A-A_old,ord=2)<self.tol:
                break
            else:
                A_old = A

        self.covariance = S
        self.precision = A
        self.primal = criterion_prim
        self.dual = criterion_dual
        return self


    def soft_threshold(self,b,a):
        if (np.abs(a)<=b):
            return 0
        if(a<-b):
            return a+b
        if(a>b):
            return a-b

    def lasso_obj(self,beta,W,s,rho):
        return 0.5*np.dot(beta.T.dot(W),beta) + beta.T.dot(s) + rho*np.sum(np.
↪abs(beta))

    def coord_lasso_solve(self,W,s,tol,beta_0,rho):
        n = beta_0.shape[0]
        beta = beta_0.copy()
        obj_0 = self.lasso_obj(beta,W,s,rho)
        while True:
            for i in range(n):
```

```
                beta[i] =  self.soft_threshold(rho/W[i,i], beta[i]-(s[i]+np.
↪dot(W[i],beta))/W[i,i])
            obj_1 = self.lasso_obj(beta,W,s,rho)
            if (obj_0-obj_1<tol):
                break
            obj_0 = obj_1
        return beta


    def __get(self,S,idx):
        end = S.shape[0] - 1
        R = S.copy()
        R = np.delete(R,idx,0)
        R = np.delete(R,idx,1)
        s = S[idx].copy()
        s = np.delete(s,idx)
        sii = S[idx][idx]

        return [R,s,sii]

    def __put(self,R,s,sii,idx):
        X = R.copy()
        X = np.insert(X,idx,s,axis=1)
        s = np.insert(s,idx,sii)
        X = np.insert(X,idx,s,axis=0)
        return X
```

1 Experiment Setup

- Pick appropriate parameters: Here we set Data dimensions (100), Number of data samples (10000), Lasso penalty parameter (0.02).
- Generate data X using normal distribution
- Compare primal and dual criterion during optimization

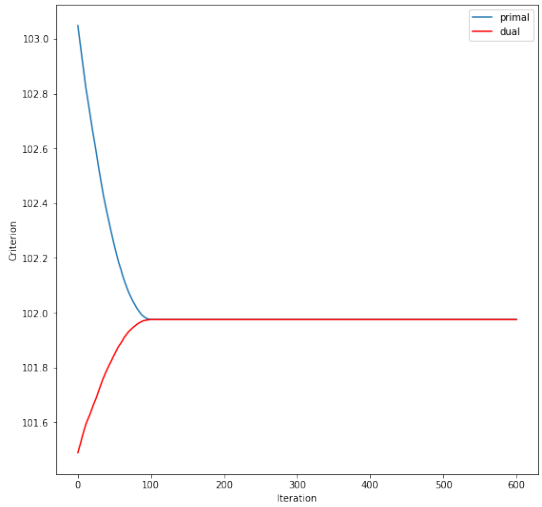
[261]:
```
X = np.random.rand(10000,100)
#X = np.random.standard_cauchy((10000,25))
g = graphicalLasso(rho=0.02)
R = g.fit(X)
```

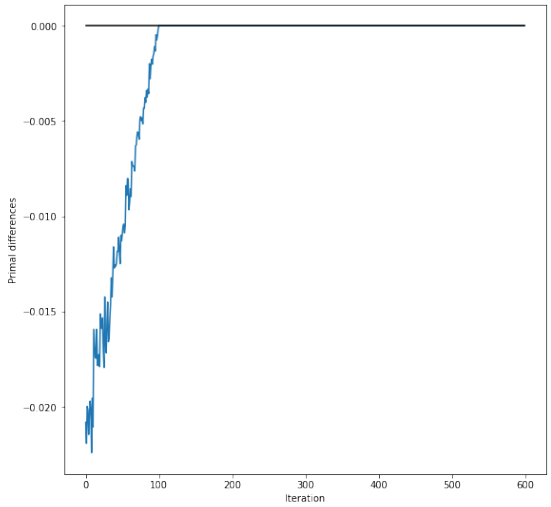
2 Figures

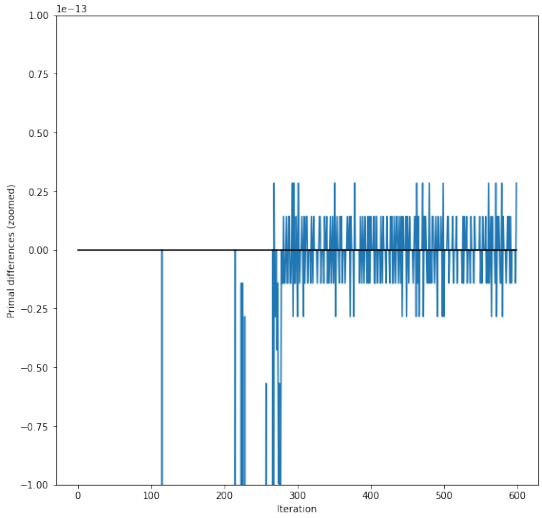
- Top-Left: The objective values of the primal criterion and the dual criterion corresponding to the covariance matrix produced by glasso algorithm as a function of the iteration index (each column/row update).

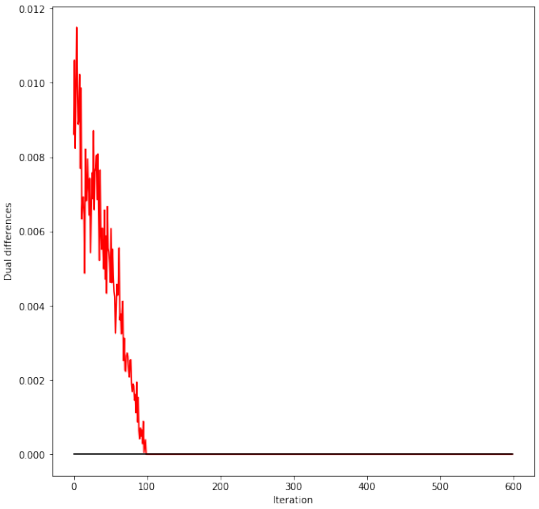
- Top-Right and Bottom-Left: The successive differences of the primal objective values
- Bottom-Right: The successive differences in the dual objective values

[262]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,20))

plt.subplot(2,2,1)
plt.plot(R.primal,label='primal')
plt.plot(R.dual,label='dual',color='red')
plt.legend()
plt.xlabel('Iteration')
plt.ylabel('Criterion')

plt.subplot(2,2,2)
plt.plot(np.array(R.primal[1:]) - np.array(R.primal[:-1]))
plt.plot(np.zeros(len(R.primal)-1),color='black')
plt.xlabel('Iteration')
plt.ylabel('Primal differences')

ax1 = plt.subplot(2,2,3)
plt.plot(np.array(R.primal[1:]) - np.array(R.primal[:-1]))
plt.plot(np.zeros(len(R.primal)-1),color='black')
ax1.set_ylim([-1e-13,1e-13])
plt.xlabel('Iteration')
plt.ylabel('Primal differences (zoomed)')

plt.subplot(2,2,4)
plt.plot(np.array(R.dual[1:]) - np.array(R.dual[:-1]),color='red')
plt.plot(np.zeros(len(R.dual)-1),color='black')
plt.xlabel('Iteration')
plt.ylabel('Dual differences')

plt.show()
```









[263]:
```
np.where(np.array(R.primal[1:]) - np.array(R.primal[:-1]) > 0)
```

[263]:
```
(array([268, 281, 285, 288, 289, 293, 295, 298, 301, 306, 309, 312, 313,
        319, 322, 330, 331, 337, 340, 345, 348, 351, 354, 358, 360, 368,
        371, 373, 378, 386, 389, 392, 396, 398, 400, 404, 407, 408, 414,
        417, 422, 427, 429, 432, 435, 436, 439, 442, 444, 448, 450, 453,
        460, 461, 463, 465, 471, 473, 475, 480, 484, 485, 489, 492, 496,
        499, 505, 506, 512, 517, 526, 528, 531, 537, 542, 549, 553, 557,
        561, 564, 566, 571, 573, 575, 579, 581, 587, 589, 591, 599]),)
```

[264]:
```
np.where(np.array(R.dual[1:]) - np.array(R.dual[:-1]) < 0)
```

[264]:
```
(array([], dtype=int64),)
```

3 Result Analysis

See figures above we have following findings: - Almost no duality gap between primal and dual - For the successive differences of the primal objective values — there are zero crossings which indicate non-monotonicity. - For the successive differences in the dual objective values — there are no zero crossings, indicating that glasso produces a monotone sequence of dual objective values.

The we can conclude:

- If the glasso algorithm were solving problem (3) by block coordinate-descent, we would not anticipate that the primal objective values is not monotone decreasing. Therefore, it's not a descent algorithm on the primal graphical lasso problem (3).
- The figure also illustrates that glasso is an ascent algorithm on the dual of the problem (3) , that is (5). The red curve in the Top-Left plot shows the dual objective rising monotonely, and the right-bottom plot shows that the increments are indeed positive.

4 Conclusion

Therefore, glasso is not a descent algorithm on the primal graphical lasso problem (3), but is indeed a descent algorithm on its equivalent dual (5).