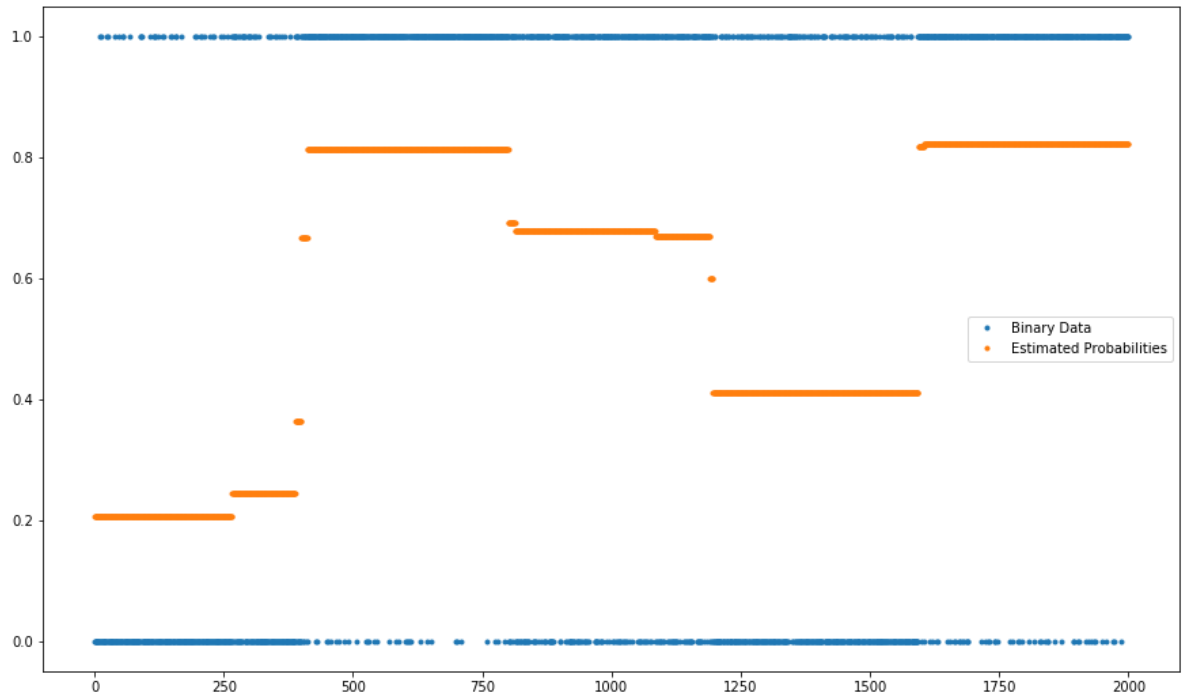
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

with open('binseq.txt', 'r') as f:
    binseq = f.readlines()
binseq = binseq[0].split(',')
binseq[-1] = '1'
binseq = [[float(y) for y in x] for x in binseq]
binseq = np.array(binseq)
binseq = binseq.squeeze()
```

In [3]:
```
from prox import prox_dp

import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
def g(theta,y):
    return np.sum(-y*theta+np.log(1+np.exp(theta)))

def h(theta,lam):
    return lam*np.sum(np.abs((theta - np.roll(theta,-1))[:-1]))

def gGrad(theta,y):
    return -y+(np.exp(theta)/(1+np.exp(theta)))

def obj(y,theta,lam):
    return g(y,theta)+h(theta,lam)

def Gt(theta,t,y):
    theta_0 = theta.copy()
    prox_dp(n=theta.shape[0], y=theta_0-t*gGrad(theta_0,y), lam=20*t,
theta=theta)
    return (theta_0 - theta)/t

def gd(beta,y):
    return np.sum(np.log(1+np.exp(-y*beta)))

def hd(D,beta,lam):
    return lam*np.linalg.norm(np.dot(D,beta),1)
```

In [5]:
```
#theta = np.random.randn(*binseq.shape)
theta = np.zeros_like(binseq)
y = binseq.copy()

bta = 0.8
epi = 1e-6
lam = 20
n = binseq.shape[0]
cot = 0

obj_0 = obj(y,theta,lam)
theta_0 = theta.copy()

while(True):
    t = 1
    G = Gt(theta,t,y)
    while (g(theta-t*G,y) > g(theta,y)-t*np.dot(gGrad(theta,y),G)+0.5*
t*(np.linalg.norm(G)**2)):
        t = bta*t
        cot = cot+1
    cot = cot+1
    prox_dp(n=n, y=theta-t*gGrad(theta,y), lam=lam*t, theta=theta)
    obj_1 = obj(y,theta,lam)

    if (obj_0 - obj_1) < epi:
         break

    obj_0 = obj_1


plt.figure(figsize=(15,9))
plt.plot(binseq,'.',label='Binary Data')
plt.plot((np.exp(theta)/(1+np.exp(theta))),'.',label='Estimated Probab
ilities')
plt.legend()
plt.show()
print('Total iterations:',cot)
# print(theta)
# print(obj_1)
```




```
Total iterations: 68
```

In [6]:

```
z = 2*y-1

#Form in 5
print('Loss in (5):',g(theta,y))
print('Penalty in (5):',h(theta,lam))

#Form in 3

Dx = -np.eye(z.shape[0]-1)
Dx = np.insert(Dx, 0, 0, axis=1)
Dx = np.insert(Dx, z.shape[0]-1, 0, axis=0)

D = np.eye(z.shape[0])+Dx
D = np.delete(D,z.shape[0]-1,axis=0)
print('D is :\n',D)

print('Loss in (3):',gd(theta,z))
print('Penalty in (3):',hd(D,theta,lam))
```

```
Loss in (5): 983.4964472358754
Penalty in (5): 130.84182100678208
D is :
 [[ 1. -1.  0. ...  0.  0.  0.]
 [ 0.  1. -1. ...  0.  0.  0.]
 [ 0.  0.  1. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ... -1.  0.  0.]
 [ 0.  0.  0. ...  1. -1.  0.]
 [ 0.  0.  0. ...  0.  1. -1.]]
Loss in (3): 983.4964472358755
Penalty in (3): 130.84182100678208
```

In [7]:
```
from cvxopt import matrix, solvers

lam = 20
delta = 0.01

n = z.shape[0]
m = D.shape[0]
A = (z*D).T
A = np.concatenate((A,-A,np.eye(m),-np.eye(m)))
b1 = (1-delta)*np.ones(n)
b2 = -delta*np.ones(n)
b3 = (lam-delta)*np.ones(m)
b = np.concatenate((b1,b2,b3,b3))
c = np.random.rand(m)

A = matrix(A)
b = matrix(b)
c = matrix(c)

sol=solvers.lp(c,A,b)
u0 = np.array(sol['x'])
u0 = u0.squeeze()
```

```
     pcost       dcost       gap    pres   dres   k/t
 0: -6.7595e-01 -1.1830e+05  1e+05  7e-02  5e-16  1e+00
 1: -5.7822e+03 -3.3770e+04  3e+04  2e-02  6e-15  4e-01
 2: -1.0385e+04 -3.1406e+04  2e+04  1e-02  1e-14  6e-01
 3: -1.5181e+04 -2.6442e+04  1e+04  7e-03  2e-14  5e-01
 4: -1.7261e+04 -2.3827e+04  7e+03  4e-03  2e-14  4e-01
 5: -1.8542e+04 -2.1934e+04  4e+03  2e-03  1e-14  2e-01
 6: -1.9340e+04 -2.0596e+04  1e+03  8e-04  1e-14  1e-01
 7: -1.9560e+04 -2.0255e+04  7e+02  4e-04  7e-15  7e-02
 8: -1.9704e+04 -1.9983e+04  3e+02  2e-04  6e-15  3e-02
 9: -1.9735e+04 -1.9914e+04  2e+02  1e-04  5e-14  2e-02
10: -1.9782e+04 -1.9820e+04  4e+01  2e-05  8e-14  5e-03
11: -1.9787e+04 -1.9799e+04  1e+01  7e-06  4e-13  1e-03
12: -1.9789e+04 -1.9791e+04  2e+00  1e-06  5e-13  3e-04
13: -1.9789e+04 -1.9790e+04  5e-01  3e-07  5e-13  7e-05
14: -1.9789e+04 -1.9790e+04  7e-02  4e-08  6e-13  9e-06
15: -1.9790e+04 -1.9790e+04  1e-02  8e-09  6e-13  2e-06
Optimal solution found.
```

In [8]:

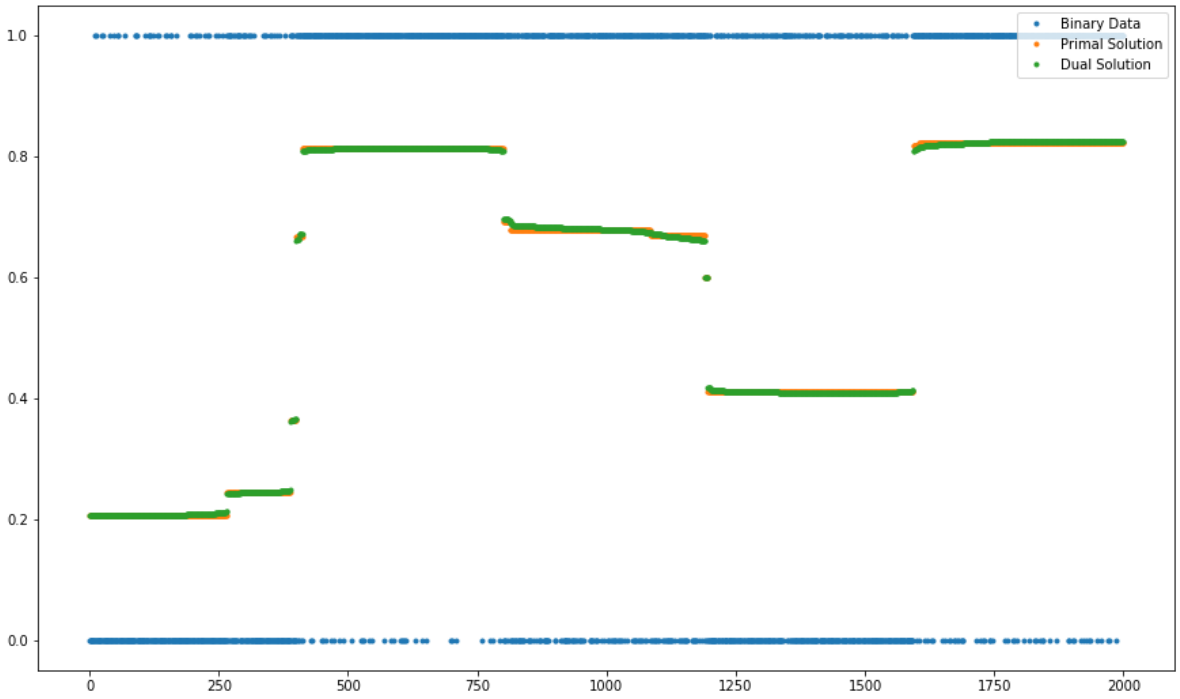
```
def g1(X):
    return np.sum(X*np.log(X)+(1-X)*np.log(1-X))

def h1(X,u,lam,tau):
    return -tau*(np.sum(np.log(X)+np.log(1-X))+np.sum(np.log(lam-u)+np
.log(u+lam)))

def obj1(Dt,u,y,lam,tau):
    X = y*Dt.dot(u)
    return g1(X)+h1(X,u,lam,tau)
    #return h1(Dt,u,y,lam,tau)

def g1Grad(X,D,y):
    G = np.log(X/(1-X))
    return np.asarray(np.sum((D.multiply(G*y)),axis=1)).T.squeeze()

def h1Grad(X,D,u,y,lam,tau):
    H1 = np.asarray(np.sum((D.multiply((1/X)*y)-D.multiply((1/(1-X))*y
)),axis=1)).T.squeeze()
    H2 = -2*u*(1/(lam-u))*(1/(lam+u))
    return -tau*(H1+H2)

def obj1Grad(Dt,D,u,y,lam,tau):
    X = y*Dt.dot(u)
    return g1Grad(X,D,y)+h1Grad(X,D,u,y,lam,tau)


def opr(x):
    return np.diag(x[:-1]) + np.diag(x[1:]) + np.diag(-x[1:-1],k=1) +
np.diag(-x[1:-1],k=-1)

def g2Grad(Dt,y,u):
    return opr((1 / ((y*Dt.dot(u))*(1-y*Dt.dot(u)))))

def h2Grad(Dt,u,y,lam,tau):
    return tau*(2*np.diag((lam**2+u**2) / ((lam**2-u**2)**2)) + opr((1
/(Dt.dot(u)))**2 + (1/(1-y*Dt.dot(u)))**2))

def obj2Grad(Dt,u,y,lam,tau):
    return g2Grad(Dt,y,u)+h2Grad(Dt,u,y,lam,tau)
```

In [9]:

```
# 3.(a)

from time import time
from scipy.sparse import dia_matrix
Ds = dia_matrix(D)
Dts = dia_matrix(D.T)

lam = 20
tau = 1/(1e3)
bta = 0.8
kMax = 50000
epi = 1e-6
alpha = 0.5
u = u0.copy()
#u = np.load("u_5e4.npy")
obj_0 = obj1(Dts,u,z,lam,tau)

for k in range(kMax):
    t = 1
    print('Iteration',k,':')
    print(obj_0)

    grad1 = obj1Grad(Dts,Ds,u,z,lam,tau)
    v = np.dot(-np.linalg.inv(obj2Grad(Dts,u,z,lam,tau)),grad1)
    c1 = obj1(Dts,u+t*v,z,lam,tau)
    while(np.isnan(c1)):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
    c2 = obj1(Dts,u,z,lam,tau)
    c3 = np.dot(grad1,v)
    while(c1>c2+alpha*t*c3):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)

    u = u+t*v
    obj_1 = c1
    if(obj_0-obj_1<epi):
        break
    obj_0 = obj_1
```

```
Iteration 0 :
-141.27836992050143
Iteration 1 :
-181.49164259637905
Iteration 2 :
-251.18651268043118
Iteration 3 :
-398.48288005253715
Iteration 4 :
-680.3406389786306
Iteration 5 :
-924.6573859640898
Iteration 6 :
-1006.4962616958954
Iteration 7 :
-1014.2751177334259
Iteration 8 :
-1031.577183293788
Iteration 9 :
-1038.6055580464583
Iteration 10 :
-1050.1858189442246
Iteration 11 :
-1058.7375748845534
Iteration 12 :
-1074.773106818741
Iteration 13 :
-1089.1450373240573
Iteration 14 :
-1092.49985180363
Iteration 15 :
-1093.3999933947102
Iteration 16 :
-1100.4423572724736
Iteration 17 :
-1107.5381487351497
Iteration 18 :
-1108.3005258797007
Iteration 19 :
-1113.0052090399686
Iteration 20 :
-1116.0607185570354
Iteration 21 :
-1121.0813649761394
Iteration 22 :
-1121.2889560557064
Iteration 23 :
-1121.2961122555666
Iteration 24 :
-1121.296406785194
```

In [10]:
```
# 4.(d)
X = Dts.dot(u)
beta = -z*np.log(X/(z-X))
plt.figure(figsize=(15,9))
plt.plot(binseq,'.',label='Binary Data')
plt.plot((np.exp(theta)/(1+np.exp(theta))),'.',label='Primal Solution'
)
plt.plot((np.exp(beta)/(1+np.exp(beta))),'.',label='Dual Solution')
plt.legend(loc='upper right')
plt.show()
# print('Estimated probabilites are closed to those in part(b). ')
# print('Primal Value:',np.sum(np.log(1+np.exp(-z*beta)))+lam*np.linal
g.norm(Ds.dot(beta),1))
# print('Primal Value in (b):',gd(theta,z)+hd(D,theta,lam))
# print('Primal Value in (b) is lower.')
```



In [93]:
```
# 3.(b)


from cvxopt import matrix, solvers

lam = 20
delta = 0.01

n = z.shape[0]
m = D.shape[0]
A = (z*D).T
A = np.concatenate((A,-A,np.eye(m),-np.eye(m)))
b1 = (1-delta)*np.ones(n)
b2 = -delta*np.ones(n)
b3 = (lam-delta)*np.ones(m)
b = np.concatenate((b1,b2,b3,b3))
c = np.random.rand(m)

A = matrix(A)
b = matrix(b)
c = matrix(c)

sol=solvers.lp(c,A,b)
u0 = np.array(sol['x'])
u0 = u0.squeeze()

y = binseq.copy()
objall = []


bta = 0.8
epi = 0
lam = 20
n = binseq.shape[0]
cot = 0
kMax = 100
theta = np.zeros_like(binseq)
obj_0 = obj(y,theta,lam)

for k in range(kMax):
    t = 1
    G = Gt(theta,t,y)
    while (g(theta-t*G,y) > g(theta,y)-t*np.dot(gGrad(theta,y),G)+0.5*
t*(np.linalg.norm(G)**2)):
        t = bta*t
        cot = cot+1
    cot = cot+1
    prox_dp(n=n, y=theta-t*gGrad(theta,y), lam=lam*t, theta=theta)
    obj_1 = obj(y,theta,lam)

#     if (obj_0 - obj_1) < epi:
#          break

    obj_0 = obj_1
    objall.append(obj_0)
```

```
lam = 20
tau = 1/(1e3)
bta = 0.8
kMax = 100
epi = 0
alpha = 0.5
u = u0.copy()

# X = Dts.dot(u)
# beta = -z*np.log(X/(z-X))
# obj_0 = obj(y,beta,lam)
obj_0 = obj1(Dts,u,z,lam,tau)


for k in range(kMax):
    t = 1

    grad1 = obj1Grad(Dts,Ds,u,z,lam,tau)
    v = np.dot(-np.linalg.inv(obj2Grad(Dts,u,z,lam,tau)),grad1)
    c1 = obj1(Dts,u+t*v,z,lam,tau)
    while(np.isnan(c1)):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
    c2 = obj1(Dts,u,z,lam,tau)
    c3 = np.dot(grad1,v)
    while(c1>c2+alpha*t*c3):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)

    u = u+t*v
    obj_1 = c1
#     if(obj_0-obj_1<epi):
#         break
    obj_0 = obj_1

    X = Dts.dot(u)
    beta = -z*np.log(X/(z-X))
    objall.append(obj(y,beta,lam))

f_star = min(objall)-1e-6
```

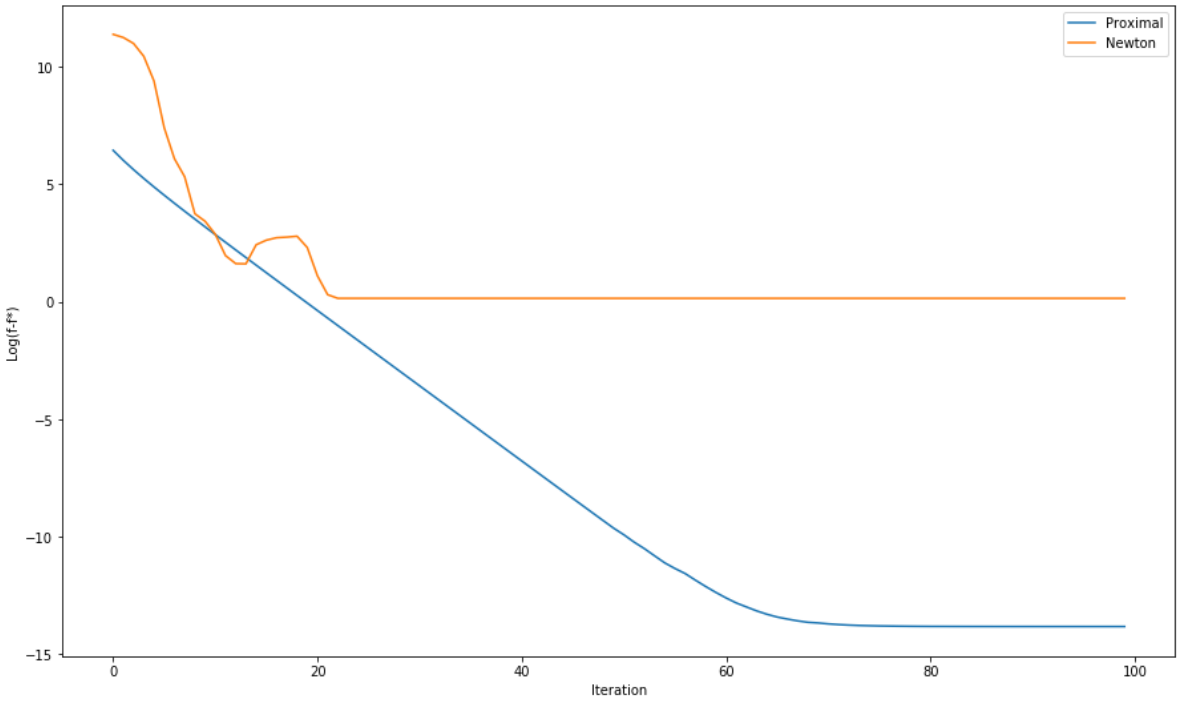
```
     pcost       dcost       gap    pres   dres   k/t
 0: -1.8423e+00 -1.1829e+05  1e+05  7e-02  5e-16  1e+00
 1: -5.8647e+03 -3.0937e+04  3e+04  2e-02  6e-15  3e-01
 2: -1.1005e+04 -2.8645e+04  2e+04  1e-02  2e-14  5e-01
 3: -1.4187e+04 -2.6221e+04  1e+04  7e-03  2e-14  5e-01
 4: -1.6797e+04 -2.3537e+04  7e+03  4e-03  2e-14  3e-01
 5: -1.8165e+04 -2.1834e+04  4e+03  2e-03  1e-14  2e-01
 6: -1.9160e+04 -2.0412e+04  1e+03  8e-04  1e-14  1e-01
 7: -1.9333e+04 -2.0191e+04  9e+02  5e-04  1e-14  8e-02
 8: -1.9541e+04 -1.9851e+04  3e+02  2e-04  9e-15  3e-02
 9: -1.9554e+04 -1.9820e+04  3e+02  2e-04  4e-14  3e-02
10: -1.9626e+04 -1.9693e+04  7e+01  4e-05  5e-14  8e-03
11: -1.9638e+04 -1.9654e+04  2e+01  1e-05  3e-13  2e-03
12: -1.9640e+04 -1.9646e+04  6e+00  3e-06  4e-13  7e-04
13: -1.9641e+04 -1.9642e+04  1e+00  7e-07  4e-13  1e-04
14: -1.9641e+04 -1.9642e+04  2e-01  1e-07  4e-13  3e-05
15: -1.9642e+04 -1.9642e+04  1e-02  6e-09  6e-13  1e-06
Optimal solution found.
```

In [99]:

```
y = binseq.copy()
gap_prox = []

bta = 0.8
epi = 0
lam = 20
n = binseq.shape[0]
cot = 0
kMax = 100
theta = np.zeros_like(binseq)
obj_0 = obj(y,theta,lam)

for k in range(kMax):
    t = 1
    G = Gt(theta,t,y)
    while (g(theta-t*G,y) > g(theta,y)-t*np.dot(gGrad(theta,y),G)+0.5*
t*(np.linalg.norm(G)**2)):
        t = bta*t
        cot = cot+1
    cot = cot+1
    prox_dp(n=n, y=theta-t*gGrad(theta,y), lam=lam*t, theta=theta)
    obj_1 = obj(y,theta,lam)

#     if (obj_0 - obj_1) < epi:
#          break

    obj_0 = obj_1
    gap_prox.append(obj_0-f_star)

lam = 20
tau = 1/(1e3)
bta = 0.8
kMax = 100
epi = 0
alpha = 0.5
u = u0.copy()
obj_0 = obj1(Dts,u,z,lam,tau)

gap_newton=[]

for k in range(kMax):
    t = 1

    grad1 = obj1Grad(Dts,Ds,u,z,lam,tau)
    v = np.dot(-np.linalg.inv(obj2Grad(Dts,u,z,lam,tau)),grad1)
    c1 = obj1(Dts,u+t*v,z,lam,tau)
    while(np.isnan(c1)):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
    c2 = obj1(Dts,u,z,lam,tau)
    c3 = np.dot(grad1,v)
    while(c1>c2+alpha*t*c3):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)

    u = u+t*v
```

```
    obj_1 = c1
#     if(obj_0-obj_1<epi):
#         break
    obj_0 = obj_1
    X = Dts.dot(u)
    beta = -z*np.log(X/(z-X))
    gap_newton.append(obj(y,beta,lam)-f_star)
```

In [116]:
```
plt.figure(figsize=(15,9))
plt.plot(np.log(gap_prox),label='Proximal')
plt.plot(np.log(gap_newton),label='Newton')
plt.xlabel('Iteration')
plt.ylabel('Log(f-f*)')
plt.legend(loc='upper right')
plt.show()
```



In [118]:

```
from cvxopt import matrix, solvers

lam = 0.02
delta = 1e-8

n = z.shape[0]
m = D.shape[0]
A = (z*D).T
A = np.concatenate((A,-A,np.eye(m),-np.eye(m)))
b1 = (1-delta)*np.ones(n)
b2 = -delta*np.ones(n)
b3 = (lam-delta)*np.ones(m)
b = np.concatenate((b1,b2,b3,b3))
c = np.random.rand(m)

A = matrix(A)
b = matrix(b)
c = matrix(c)

sol=solvers.lp(c,A,b)
u0 = np.array(sol['x'])
u0 = u0.squeeze()

#theta = np.random.randn(*binseq.shape)
#theta_0 = theta.copy()
theta = np.zeros_like(binseq)
y = binseq.copy()

bta = 0.8
epi = 0
lam = 0.02
n = binseq.shape[0]
cot = 0
kMax = 100
obj_0 = obj(y,theta,lam)
objall = []

for k in range(kMax):
    t = 1
    G = Gt(theta,t,y)
    while (g(theta-t*G,y) > g(theta,y)-t*np.dot(gGrad(theta,y),G)+0.5*
t*(np.linalg.norm(G)**2)):
        t = bta*t
        cot = cot+1
    cot = cot+1
    prox_dp(n=n, y=theta-t*gGrad(theta,y), lam=lam*t, theta=theta)
    obj_1 = obj(y,theta,lam)

#     if (obj_0 - obj_1) < epi:
#         break

    obj_0 = obj_1
    objall.append(obj_0)

from scipy.sparse import dia_matrix
Ds = dia_matrix(D)
```

```
Dts = dia_matrix(D.T)

lam = 0.02
tau = 1/(1e3)
bta = 0.8
kMax = 100
epi = 0
alpha = 0.5
u = u0.copy()
obj_0 = obj1(Dts,u,z,lam,tau)

for k in range(kMax):
    t = 1

    grad1 = obj1Grad(Dts,Ds,u,z,lam,tau)
    v = np.dot(-np.linalg.inv(obj2Grad(Dts,u,z,lam,tau)),grad1)
    c1 = obj1(Dts,u+t*v,z,lam,tau)
    while(np.isnan(c1)):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
    c2 = obj1(Dts,u,z,lam,tau)
    c3 = np.dot(grad1,v)
    while(c1>c2+alpha*t*c3):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)

    u = u+t*v
    obj_1 = c1
#     if(obj_0-obj_1<epi):
#         break
    obj_0 = obj_1
    X = Dts.dot(u)
    beta = -z*np.log(X/(z-X))
    objall.append(obj(y,beta,lam))

f_star = min(objall)-1e-6
```

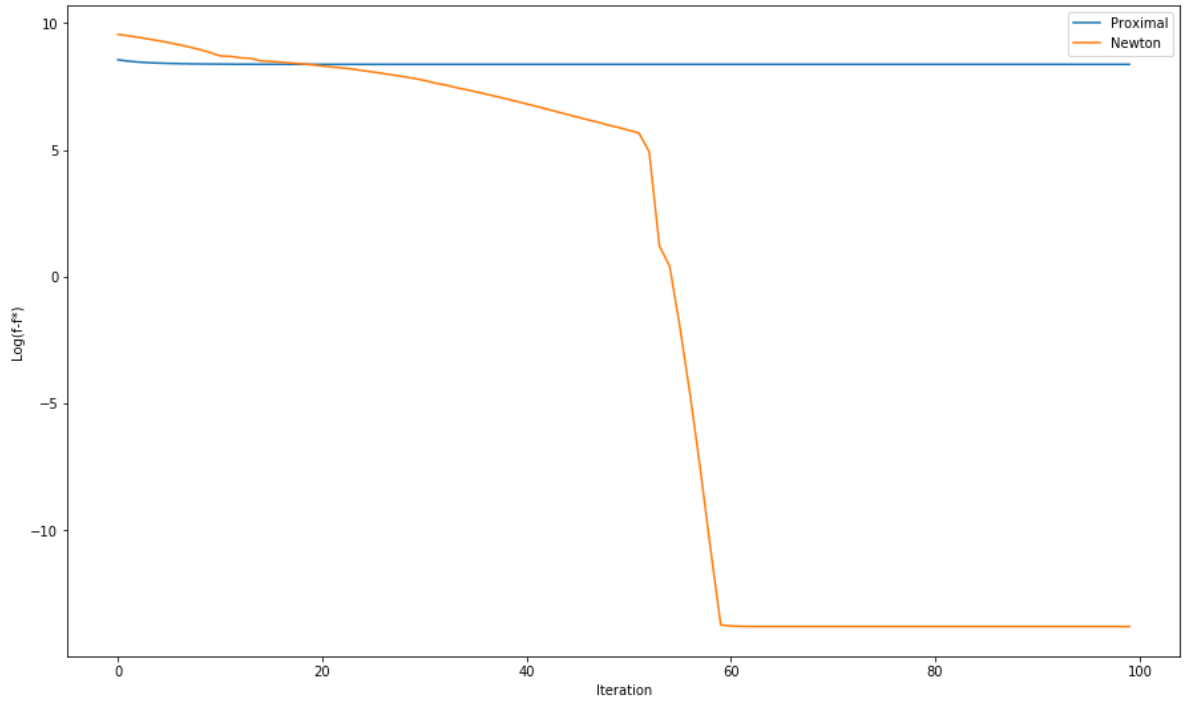
```
     pcost       dcost       gap    pres   dres   k/t
 0:  1.0425e+00 -2.9842e+03  2e+04  3e+00  6e-16  1e+00
 1:  4.1440e+01 -5.3376e+02  2e+03  5e-01  7e-16  1e+00
 2: -7.9858e+00 -6.6103e+01  2e+02  6e-02  5e-16  1e-01
 3: -1.5141e+01 -3.7948e+01  5e+01  2e-02  5e-16  5e-02
 4: -1.8482e+01 -2.5470e+01  1e+01  7e-03  4e-16  1e-02
 5: -1.9267e+01 -2.1921e+01  5e+00  3e-03  3e-16  3e-03
 6: -1.9468e+01 -2.0057e+01  1e+00  6e-04  4e-16  4e-04
 7: -1.9508e+01 -1.9632e+01  2e-01  1e-04  3e-16  5e-05
 8: -1.9517e+01 -1.9539e+01  4e-02  2e-05  4e-16  5e-06
 9: -1.9520e+01 -1.9520e+01  4e-04  2e-07  4e-16  6e-08
10: -1.9520e+01 -1.9520e+01  5e-05  3e-08  2e-12  7e-09
11: -1.9520e+01 -1.9520e+01  6e-05  3e-08  3e-10  8e-09
12: -1.9520e+01 -1.9520e+01  7e-05  3e-08  6e-10  8e-09
13: -1.9520e+01 -1.9520e+01  7e-05  2e-08  6e-10  8e-09
14: -1.9520e+01 -1.9520e+01  6e-05  2e-08  6e-10  7e-09
15: -1.9520e+01 -1.9520e+01  4e-05  1e-08  9e-10  4e-09
16: -1.9520e+01 -1.9520e+01  2e-05  5e-09  2e-09  2e-09
Optimal solution found.
```

In [121]:

```
y = binseq.copy()
gap_prox = []


bta = 0.8
epi = 0
lam = 0.02
n = binseq.shape[0]
cot = 0
kMax = 100
theta = np.zeros_like(binseq)
obj_0 = obj(y,theta,lam)

for k in range(kMax):
    t = 1
    G = Gt(theta,t,y)
    while (g(theta-t*G,y) > g(theta,y)-t*np.dot(gGrad(theta,y),G)+0.5*
t*(np.linalg.norm(G)**2)):
        t = bta*t
        cot = cot+1
    cot = cot+1
    prox_dp(n=n, y=theta-t*gGrad(theta,y), lam=lam*t, theta=theta)
    obj_1 = obj(y,theta,lam)

#     if (obj_0 - obj_1) < epi:
#          break

    obj_0 = obj_1
    gap_prox.append(obj_0-f_star)

lam = 0.02
tau = 1/(1e3)
bta = 0.8
kMax = 100
epi = 0
alpha = 0.5
u = u0.copy()
obj_0 = obj1(Dts,u,z,lam,tau)

gap_newton=[]

for k in range(kMax):
    t = 1

    grad1 = obj1Grad(Dts,Ds,u,z,lam,tau)
    v = np.dot(-np.linalg.inv(obj2Grad(Dts,u,z,lam,tau)),grad1)
    c1 = obj1(Dts,u+t*v,z,lam,tau)
    while(np.isnan(c1)):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
    c2 = obj1(Dts,u,z,lam,tau)
    c3 = np.dot(grad1,v)
    while(c1>c2+alpha*t*c3):
        t = bta*t
        c1 = obj1(Dts,u+t*v,z,lam,tau)
```

```
    u = u+t*v
    obj_1 = c1
#     if(obj_0-obj_1<epi):
#         break
    obj_0 = obj_1
    X = Dts.dot(u)
    beta = -z*np.log(X/(z-X))
    gap_newton.append(obj(y,beta,lam)-f_star)
```

In [136]:
```
plt.figure(figsize=(15,9))
plt.plot(np.log(gap_prox),label='Proximal')
plt.plot(np.log(gap_newton),label='Newton')
plt.xlabel('Iteration')
plt.ylabel('Log(f-f*)')
plt.legend(loc='upper right')
plt.show()
```



In [ ]: