Problem 1

In [1]:

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

In [26]:

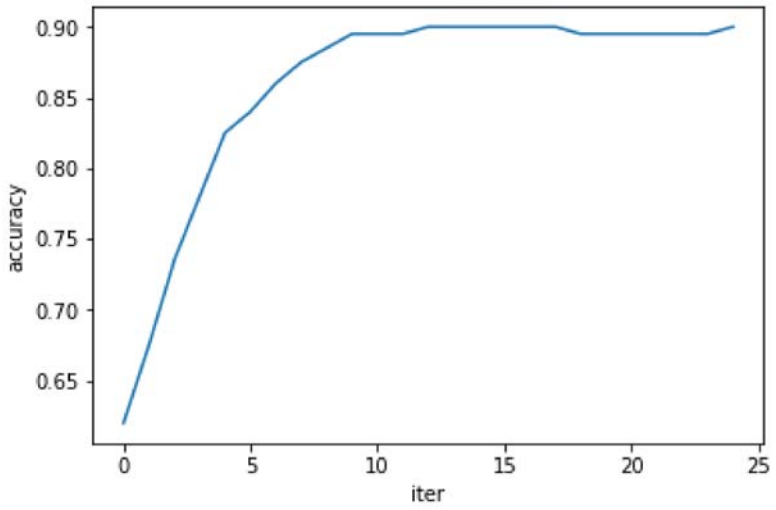
```
n=200
x=3*(np.random.rand(n,4)-0.5)
y=(2*x[:,1]-x[:,2]+0.5+0.5*np.random.randn(n))>0
y=2*y-1
```

In [15]:

```
#1. implement batch steepest gradient method
w=np.array([1.,1.,1.,1.])
lam=0.01
steepest_acc=[]
steepest_j=[]
s=0
for i in range(25):
    p=1/(1+np.exp(-y*np.dot(w,x.T)))
    j=np.sum(-y*(1-p)*x.T,axis=1)/n+2*lam*w
    jw=np.sum(np.log(1+np.exp(-y*np.dot(w,x.T))))/n+lam*np.dot(w,w.T)
    steepest_j.append(jw)
    w=w-j
    f=np.dot(w,x.T)>0
    f=2*f-1
    acc=0
    for i in range(n):
        if f[i]==y[i]:
            acc+=1
    steepest_acc.append(acc/n)
plt.plot(steepest_acc)
plt.xlabel('iter')
plt.ylabel('accuracy')
```

Out[15]:

```
Text(0, 0.5, 'accuracy')
```

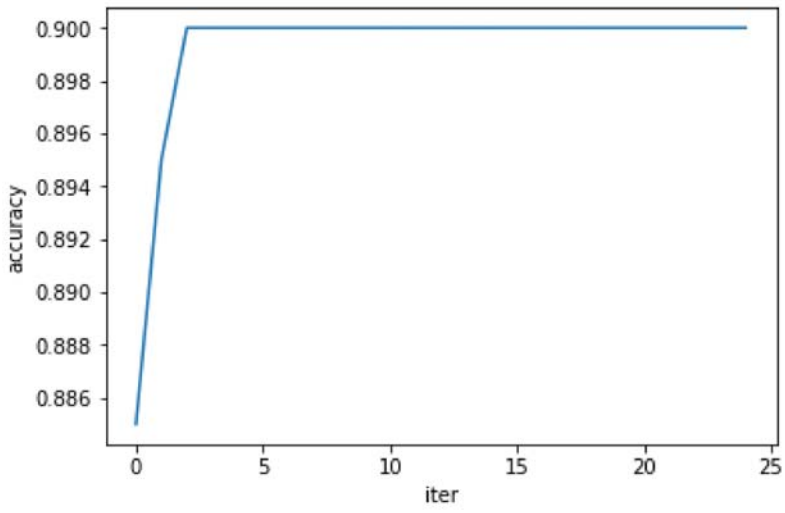


In [27]:

```
#2. implement Newton based method
w=np.array([1.,1.,1.,1.])
lam=0.01
newton_acc=[]
newton_j=[]
s=0
for i in range(25):
    p=1/(1+np.exp(-y*np.dot(w,x.T)))
    j=np.sum(-y*(1-p)*x.T,axis=1)/n+2*lam*w
    hess=np.dot((p*(1-p)*x.T),x)/n+2*lam
    jw=np.sum(np.log(1+np.exp(-y*np.dot(w,x.T))))/n+lam*np.dot(w,w.T)
    newton_j.append(jw)
    w=w-np.dot(np.linalg.inv(hess),j)
    f=np.dot(w,x.T)>0
    f=2*f-1
    acc=0
    for i in range(n):
        if f[i]==y[i]:
            acc+=1
    newton_acc.append(acc/n)
plt.plot(newton_acc)
plt.xlabel('iter')
plt.ylabel('accuracy')
```

Out[27]:

```
Text(0, 0.5, 'accuracy')
```

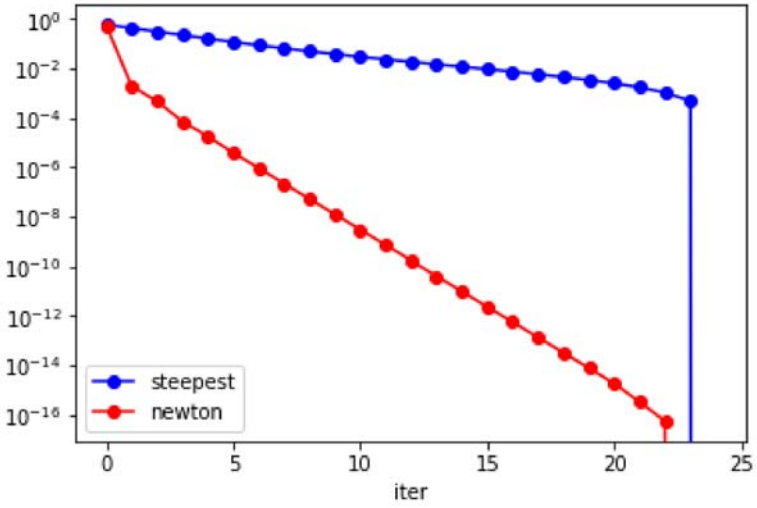


In [28]:

```
#3. compare the performance
plt.plot(np.abs(steepest_j-min(steepest_j)),'bo-',label='steepest')
plt.plot(np.abs(newton_j-min(newton_j)),'ro-',label='newton')
plt.xlabel('iter')
plt.legend()
plt.semilogy()
```

Out[28]:

```
[]
```



In [10]:

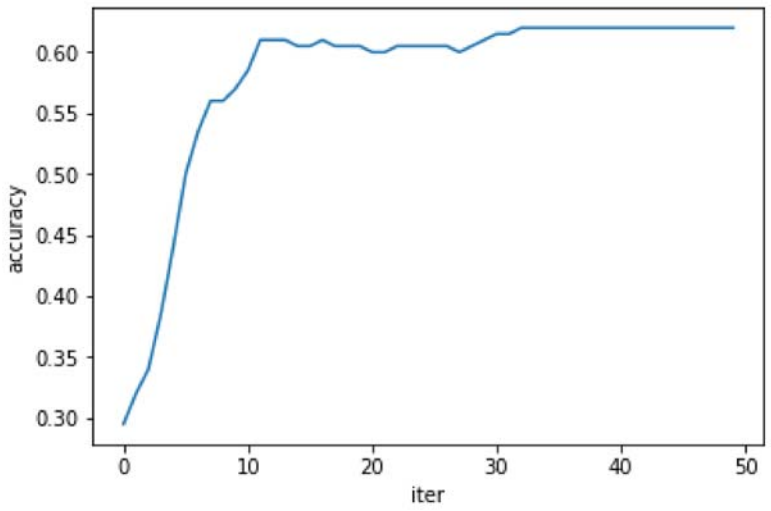
```
#4. multiclass version of logistic regression
x=3*(np.random.rand(n,3)-0.5)
w=np.array([2,-1,0.5,-3,2,1,1,2,3]).reshape(3,3)
y=np.dot(np.hstack((x[:,1:3],np.ones(n).reshape(n,1))),w)+0.5*np.random.randn(n,3)
y[y<2]=-1
y[y>=2]=1
```

In [11]:

```
# steepest
w=np.array([2,-1,0.5,-3,2,1,1,2,3]).reshape(3,3)
lam=0.01
steepest_acc=[]
steepest_j=[]
s=0
for i in range(50):
    p=1/(1+np.exp(-y.T*np.dot(w,x.T)))
    j=np.dot(-y.T*(1-p),x)/n+2*lam*w
    jw=np.sum(np.log(1+np.exp(-y.T*np.dot(w,x.T))),axis=1)/n+lam*np.dot(w,w.T)
    steepest_j.append(jw)
    w=w-j
    f=np.dot(w,x.T)>0
    f=2*f-1
    acc=0
    for i in range(n):
        if (f.T[i]==y[i]).all():
            acc+=1
    steepest_acc.append(acc/n)
plt.plot(steepest_acc)
plt.xlabel('iter')
plt.ylabel('accuracy')
```

Out[11]:

```
Text(0, 0.5, 'accuracy')
```

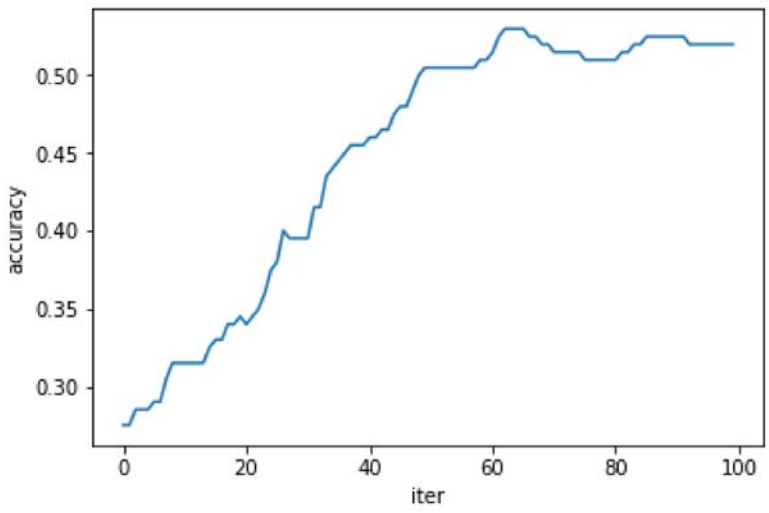


In [12]:

```
# newton
w=np.array([2,-1,0.5,-3,2,1,1,2,3]).reshape(3,3)
lam=0.01
newton_acc=[]
newton_j=[]
s=0
for i in range(100):
    p=1/(1+np.exp(-y.T*np.dot(w,x.T)))
    j=np.dot(-y.T*(1-p),x)/n+2*lam*w
    hess=np.dot(np.dot(np.dot(p,(1-p).T),x.T),x)/n+2*lam
    jw=np.sum(np.log(1+np.exp(-y.T*np.dot(w,x.T))))/n+lam*np.dot(w,w.T)
    newton_j.append(jw)
    w=w+np.dot(np.linalg.inv(hess),j)
    f=np.dot(w,x.T)>0
    f=2*f-1
    acc=0
    for i in range(n):
        if (f.T[i]==y[i]).all():
            acc+=1
    newton_acc.append(acc/n)
plt.plot(newton_acc)
plt.xlabel('iter')
plt.ylabel('accuracy')
```

Out[12]:

```
Text(0, 0.5, 'accuracy')
```



Problem 2

In [5]:

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

In [103]:

```
A=np.array([[3,0.5],[0.5,1]])
m=np.array([[1],[2]])
```

In [104]:

```
def st(m,q):
    w=np.zeros(m.shape)
    for i in range(len(w)):
        if m[i]>q:
            w[i]=m[i]-q
        elif np.abs(m[i])<q:
            w[i]=0
        else:
            w[i]=m[i]+q
    return w
```

In [105]:

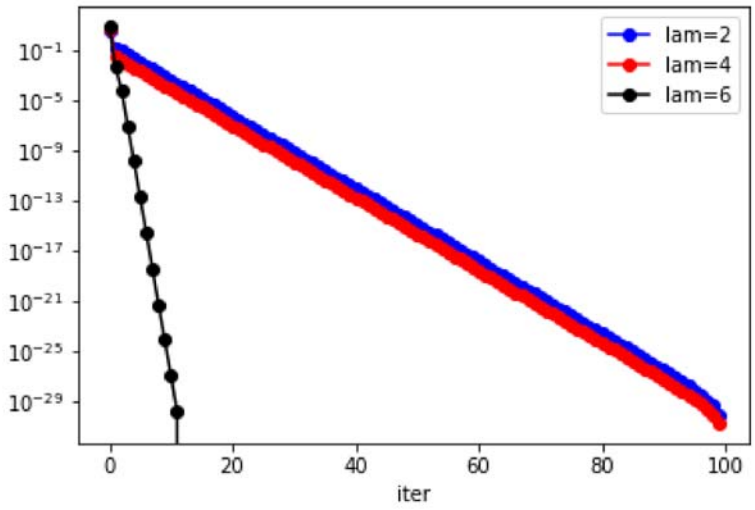
```
L=np.max(np.linalg.eig(2*A)[0])
for lam in [2,4,6]:
    w=np.array([[3],[1]])
    w_list=w.copy().T
    for i in range(100):
        grad=2*np.dot(A,w-m)
        w=st(w-grad/L,lam/L)
        w_list=np.vstack((w_list,w.T))
    globals()['w'+str(int(lam/2))]=w_list
    w_list=w_list-w_list[100]
    perf=[]
    for i in range(100):
        perf.append(np.dot(w_list[i],w_list[i].T))
    globals()['perf'+str(int(lam/2))]=perf
```

In [106]:

```
#1. show the result of PG
plt.plot(perf1,'bo-',label='lam=2')
plt.plot(perf2,'ro-',label='lam=4')
plt.plot(perf3,'ko-',label='lam=6')
plt.xlabel('iter')
plt.legend()
plt.semilogy()
```

Out[106]:

[]

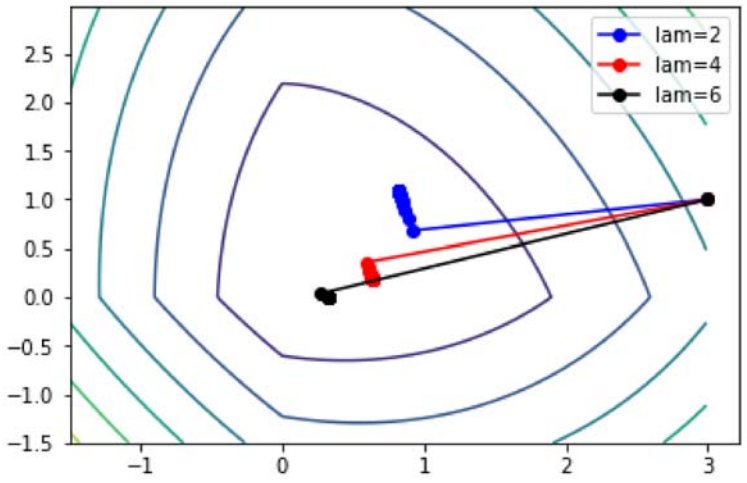


In [107]:

```
#2. trajectories
x_1=np.arange(-1.5,3,0.01)
x_2=np.arange(-1.5,3,0.02)
X1,X2=np.mgrid[-1.5:3:0.01,-1.5:3:0.02]
f=np.zeros((len(x_1),len(x_2)))
for i in range(len(x_1)):
    for j in range(len(x_2)):
        inr=np.vstack([x_1[i],x_2[j]])
        f[i,j]=np.dot(np.dot((inr-m).T, A), (inr- m)) + lam * (np.abs(x_1[i]) + np.abs(x_2[j]))
plt.contour(X1,X2,f)
plt.plot(w1[:,0],w1[:,1],'bo-',label='lam=2')
plt.plot(w2[:,0],w2[:,1],'ro-',label='lam=4')
plt.plot(w3[:,0],w3[:,1],'ko-',label='lam=6')
plt.legend()
```

Out[107]:

<matplotlib.legend.Legend at 0x1da0829d6d8>



Problem 3

1. 仮定

$$\xi_i = \max(0,\ 1 - y_i w^T x_i)$$

これにより

$$1^T \xi = \sum_{i=1}^{n} \max(0,\ 1 - y_i w^T x_i)$$

原問題は

$$\min \quad 1^T \xi + \lambda w^T w$$

$$\text{s.t.} \quad \xi_i \geq 1 - y_i w^T x_i, \quad i = 1, \dots, n$$

$$\xi \geq 0$$

となる。Lagrangian式は

$$L(w, \xi, \alpha, \beta)$$

$$= 1^T \xi + \lambda w^T w + \sum_i \alpha_i (1 - y_i w^T x_i - \xi_i) - \beta^T \xi$$

である。

$$\frac{\partial L}{\partial w} = 2\lambda w - \sum_i \alpha_i y_i x_i = 0 \Rightarrow \hat{w} = \frac{1}{2\lambda} \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial \xi} = 1 - \alpha - \beta = 0 \quad \Rightarrow \hat{\alpha} + \hat{\beta} = 1$$

$\omega, \xi, \alpha, \beta$ を Lagrangian 式に代入し，

$$L(\omega, \xi, \alpha, \beta)$$

$$= -\frac{1}{4\lambda}\alpha^T K \alpha + (1^T - \alpha^T - \beta^T)\xi + 1^T\alpha$$

$$= -\frac{1}{4\lambda}\alpha^T K \alpha + 1^T\alpha \quad (K = y_i y_j x_i^T x_j)$$

よって，原問題は

$$\max \quad -\frac{1}{4\lambda}\alpha^T K \alpha + \alpha^T 1$$

$$\text{s.t.} \quad 0 \leqq \alpha \leqq 1$$

となる

2. 問題1により

$$\frac{\partial L}{\partial w} = 2\lambda w - \sum_i \alpha_i y_i x_i = 0 \Rightarrow w = \frac{1}{2\lambda}\sum_i \alpha_i y_i x_i$$