

5.4 统计分析函数

Markdown by 张涛涛

1. 均值和百分位数

导入模块，生成数据。

```
In [30]: import numpy as np
import matplotlib.pyplot as plt
if __name__ == '__main__':
    np.random.seed(3)
    time = np.arange(30)
    y = np.random.rand(30)*10-5+0.3*time
```

计算平均数和百分位数。

```
In [31]: dat = [y,
               np.mean(y,axis=0), ## 计算平均数。第一个参数为要计算的数据；axis参数指定轴。
               np.median(y,axis=0), ## 计算中位数。用法同上。
               np.percentile(y,90,axis=0), ##计算百分位数。用法同上，但第二个参数为要计算的百分位数。
               np.percentile(y,10,axis=0)]
print(dat[1:5])
```

[3.8050986699512066, 4.343131393906942, 7.704979080599599, -1.5119811389049338]

2. 方差和标准差

```
In [32]: var = np.var(y,axis=0) ## 计算方差。用法与np.mean()一致；
std = np.std(y,axis=0) ## 计算标准差，用法同上
print(var,std**2)
```

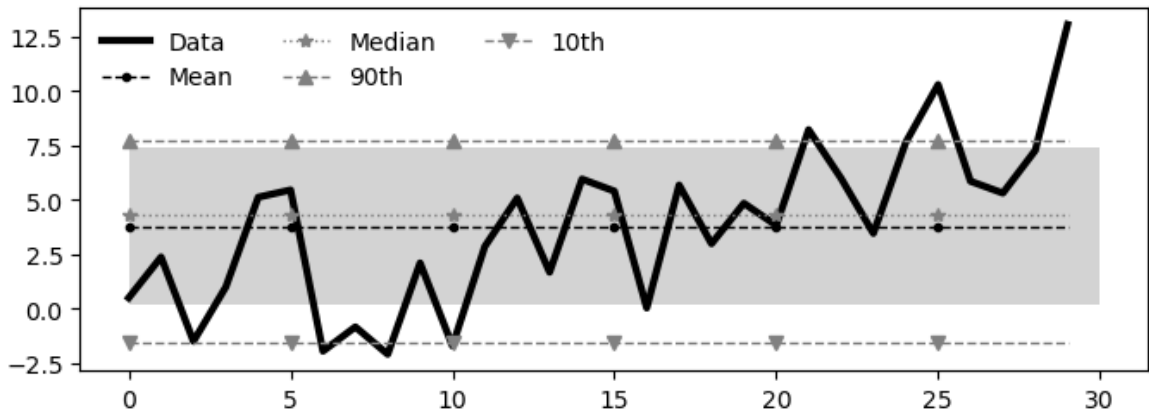
13.175594239681516 13.175594239681518

将以上数据的计算结果绘图。

```
In [33]: color = ['k','k','grey','gray','gray']
line = ['-', '--', ':', '--', '--']
wd = [3,1,1,1,1]
label=['Data', 'Mean', 'Median', '90th', '10th']
marker = ['', '.', '*', '^', 'v']
xf = [0,30,30,0,0]
yf = [-1*std, -1*std, std, std, -1*std]
yf = np.array(yf)+np.mean(y)
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(2,1,1)
L = []
for i,x in enumerate(dat):
    if i>0:
        x = np.tile(x,(len(time),))
        tmp1, = ax.plot(time,x,
                        color=color[i],label=label[i],
                        marker=marker[i],markevery=5,
                        linestyle=line[i],linewidth=wd[i])
        L.append(tmp1)
```

```
ax.fill(xf,yf, facecolor='lightgrey')
ax.legend(handles=L,ncol=3,loc='upper left',
          frameon=False)
```

Out[33]: <matplotlib.legend.Legend at 0x228641a2560>



3. 协方差和相关系数

```
In [34]: cov = np.cov(y,time) ## 协方差
print(cov)
r = np.corrcoef(y,time) ## 相关系数
print(r)
```

```
[[13.62992508 22.98957124]
 [22.98957124 77.5       ]]
[[1.         0.70734825]
 [0.70734825 1.         ]]
```

4. 线性回归和趋势

Numpy库提供的np.linalg.lstsq()函数可以用来作一元和多元线性回归。

```
In [36]: import numpy as np
import matplotlib.pyplot as plt
if __name__=='__main__':
    np.random.seed(3)
    time = np.arange(30)
    y = np.random.rand(30)*10-5+0.3*time ##生成带趋势的随机数据

    x0 = np.ones(len(time))
    x1 = time
    x2 = np.sin(4*2*time/30*np.pi)
    X = np.vstack([x0,x1,x2]).T
    b,*res = np.linalg.lstsq(X,y,rcond=None)
    print(X.shape,b.shape)
    yh = np.dot(X,b)
```

(30, 3) (3,)

对原始数据和回归结果绘图。

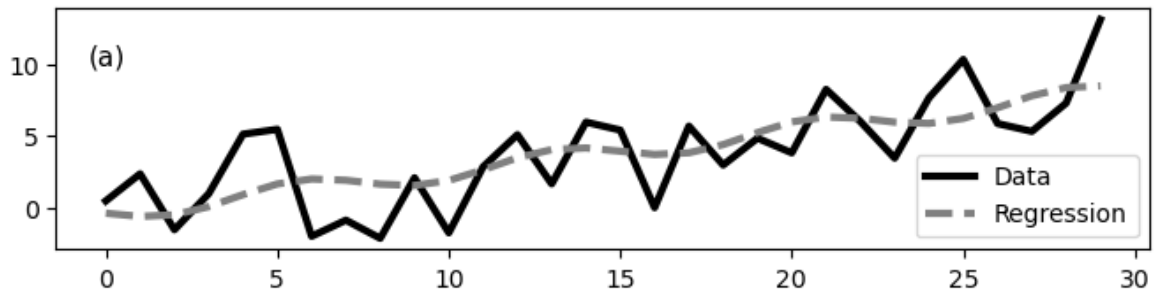
```
In [41]: fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(3,1,1)
ax.plot(y,color='k',label='Data',
        linewidth=3,linestyle='-')
ax.plot(yh,color='gray',label='Regression',
```

```

        linewidth=3,linestyle='--')
ax.legend(loc='lower right')
ax.text(-0.5,10,'(a)',fontsize=11)

```

Out[41]: Text(-0.5, 10, '(a)')



np.linalg.lstsq()函数做一元线性回归则得到线性趋势。此外，np.polyfit()和np.polyval()函数也可快速得到线性趋势。

```

In [44]: X = np.vstack([x0,x1]).T
b,*res = np.linalg.lstsq(X,y,rcond=None)
yh1 = np.dot(X,b)
pf = np.polyfit(time,y,1)
yt = np.polyval(pf,time)

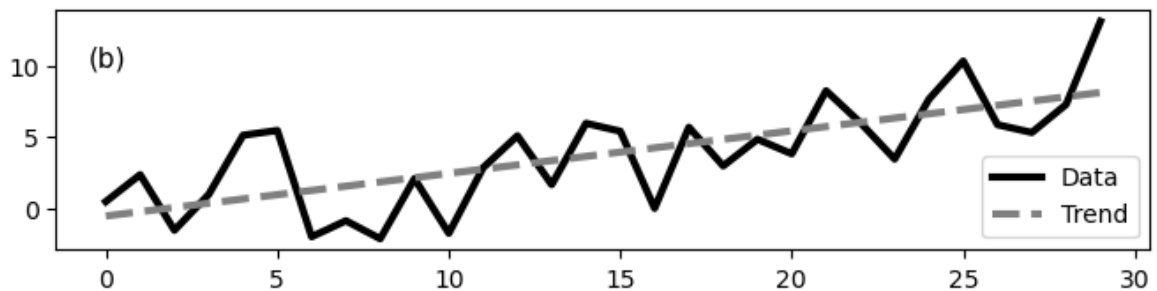
```

```

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(3,1,2)
ax.plot(y,color='k',label='Data',
        linewidth=3,linestyle='--')
ax.plot(yh1,color='gray',label='Trend',
        linewidth=3,linestyle='--')
ax.legend(loc='lower right')
ax.text(-0.5,10,'(b)',fontsize=11)

```

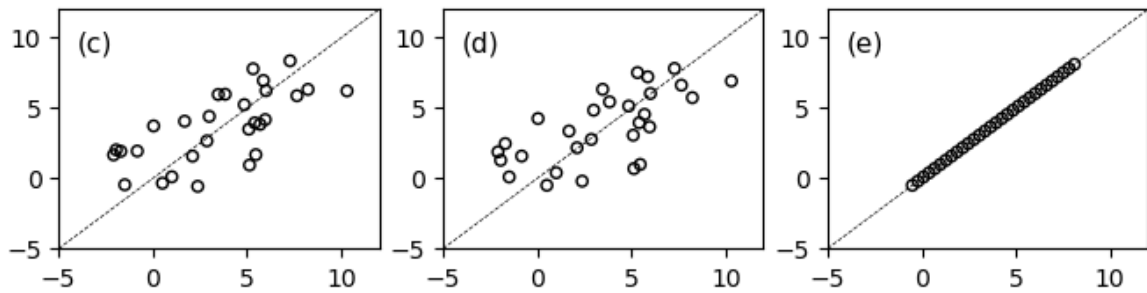
Out[44]: Text(-0.5, 10, '(b)')



```

In [47]: fig = plt.figure(figsize=(8,6))
for i,x in enumerate([[y,yh],
[y,yh1],
[yh1,yt]]):
    ax = fig.add_subplot(3,3,7+i)
    ax.scatter(x[0],x[1],20,marker='o',
               edgecolor='k',facecolor='none')
    ax.plot([-5,12],[-5,12], '--k',
            linewidth=0.5)
    ax.set_xlim([-5,12])
    ax.set_ylim([-5,12])
    ax.text(-4,9,(''+chr(ord('c')+i)+''),
            fontsize=11)

```



上图从左至右分别为：“原始数据与回归量”，“原始数据与趋势”，“线性回归与拟合”的数据结果。表明使用回归函数和拟合函数的结果是一致的。

通常在对气象数据进行处理时需要做“去趋势”，将原始数据 y 减去 yt 即可，也可使用 `scipy.signal.detrend()` 函数。

5. 显著性检验

对统计分析结果进行物理意义解释前，需要先进行显著性检验。气象上一般用Student's t 检验。也可以构建随机分布检验，需要用到Scipy.stats模块。

```
In [52]: import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

def ttest(dat, mu):
    m = np.mean(dat)
    t = (m - mu) / np.std(dat) * np.sqrt(len(dat))
    p = stats.t(len(dat) - 1).cdf(t)
    ta = stats.t(len(dat) - 1).ppf(0.975)
    print(t, ta, p)
    return t, p, ta

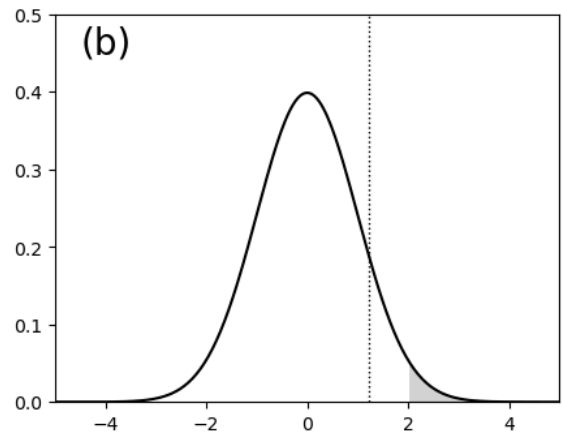
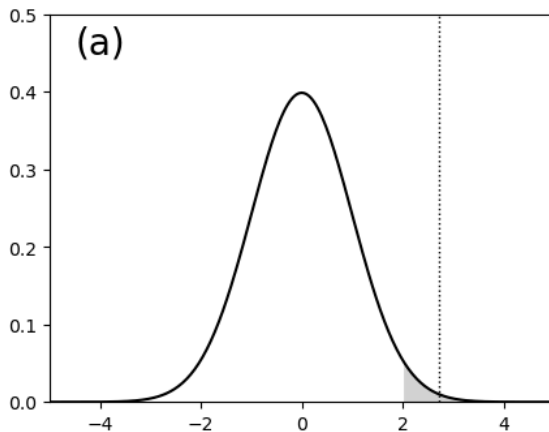
if __name__ == '__main__':
    np.random.seed(3)
    time = np.arange(30)
    dat = np.random.rand(30) * 10 - 5 + 0.3 * time
    t, p, ta = ttest(dat, 0)
    x = np.linspace(-10, 10, 501)
    y = stats.t(len(x) - 1).pdf(x)
    fig = plt.figure(figsize=(11, 8.5))

    for i, mu in enumerate([2, 3]):
        t, p, ta = ttest(dat, mu)
        ax = fig.add_subplot(2, 2, i + 1)
        ax.plot(x, y, '-k')
        msk = x >= ta
        pa = stats.t(len(y) - 1).pdf(ta)
        ax.fill(np.hstack((ta, ta, x[msk])),
                np.hstack((0, pa, y[msk])),
                color='lightgray')
        ax.plot([t, t], [0, 0.5], ':k', linewidth=1)
        ax.set_ylim([0, 0.5])
        ax.set_xlim([-5, 5])
        ax.text(-4.5, 0.45, '(' + chr(ord('a') + i) + ')',
                fontsize=20)
    plt.show()
```

```

5.741712524032624 2.045229642132703 0.9999983788365278
2.723807853452162 2.045229642132703 0.9945923325554527
1.2148555181619316 2.045229642132703 0.882889779282764

```



对相关系数做t检验

```

In [50]: import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

def ttest(r,n):
    t = r/np.sqrt(1-r**2)*np.sqrt(n-2)
    p = stats.t(n-2).cdf(t)
    ta = stats.t(n-2).ppf(0.975)
    return t,p,ta

if __name__=='__main__':
    np.random.seed(3)
    time = np.arange(30)
    dat = np.random.rand(30)*10-5+0.3*time
    x2 = np.sin(4*2*time/30*np.pi)
    for x in [time,x2]:
        r = np.corrcoef(dat,x)[0,1]
        t,p,ta = ttest(r,30)
        print('r=', '%6.3f' %(r),
              't=', '%6.3f' %(t),
              'p=', '%6.3f' %(p),
              'ta=', '%6.3f' %(ta))

```

```

r= 0.707 t= 5.295 p= 1.000 ta= 2.048
r= -0.259 t= -1.420 p= 0.083 ta= 2.048

```

当不知道相关系数的t分布公式，可以使用随机分布进行显著性检验

Monte Carlo检验

```

In [51]: R = []
for i in range(10000):
    x = np.random.rand(30)
    tmp = np.corrcoef(dat,x)[0,1]
    R.append(tmp)
R = np.array(R)
for r in [0.707,-0.259]:
    msk = R>=np.abs(r)
    p = np.sum(msk)/len(msk)

```

```
print('r=', '%6.3f' %(r),  
      'p=', '%6.3f' %(p))
```

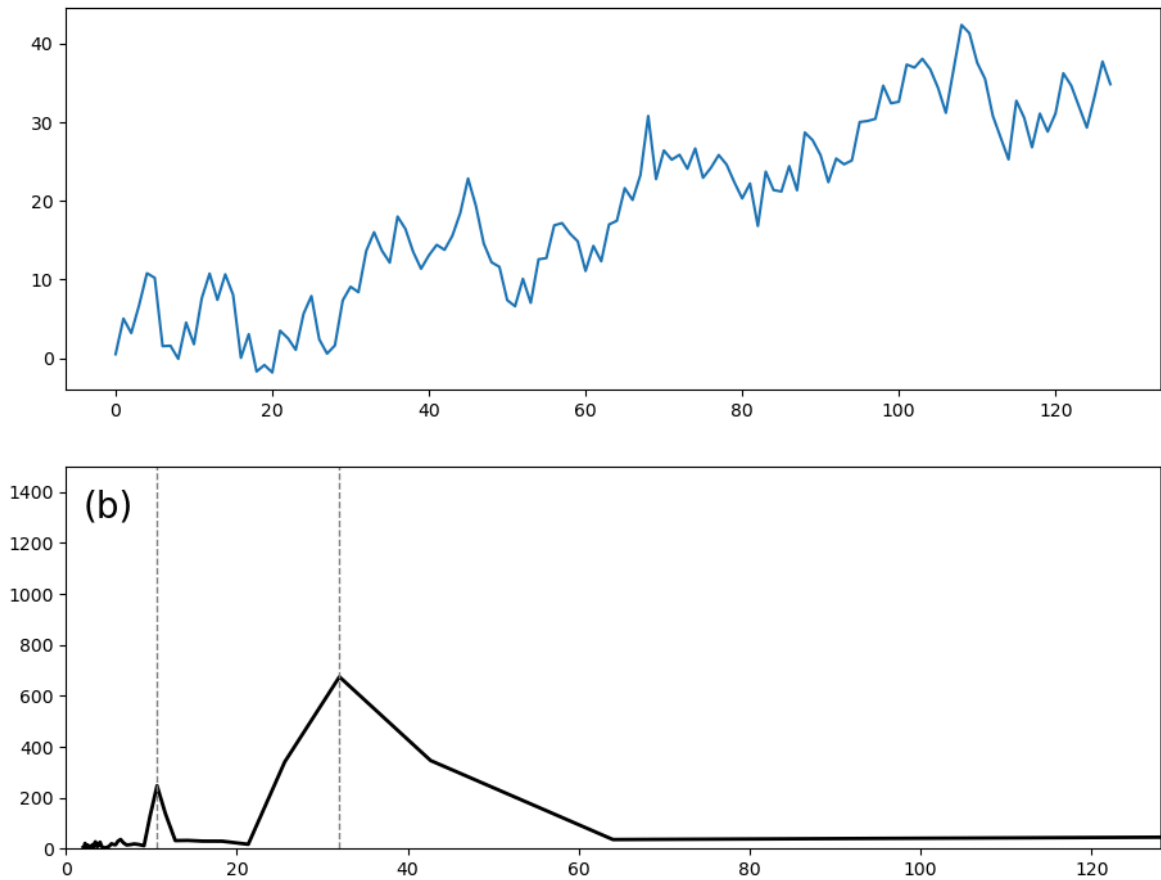
```
r= 0.707 p= 0.000  
r= -0.259 p= 0.085
```

6. 谱分析和滤波

6.1 谱分析 ——在气象上常用来提取数据的周期特征。

```
In [53]: import numpy as np  
from scipy.signal import detrend  
import matplotlib.pyplot as plt  
  
def specx_anal(dat, smooth=0):  
    dat = detrend(dat)                ## 子函数输入参数为一维序列dat和滑动平均窗口长  
    frq = np.fft.fftfreq(len(dat))    ## 对数据做去趋势  
    p = np.fft.fft(dat)              ## 计算频率  
    p = np.abs(p)**2                 ## 快速傅里叶变换，得到谱结果  
    n = len(p)                       ## p.real**2+p.imag**2  
    if smooth:  
        wgt = np.ones(smooth)  
        wgt[[0,-1]] = 0.5  
        p0 = p.copy()  
        half = smooth//2  
        for i in range(half, n-half):  
            p0[i] = np.average(  
                p[i-half:i+half+1],  
                weights=wgt)  
        p = p0  
    df = frq.copy()  
    df[1:] = frq[1:] - frq[:-1]  
    df[0] = df[1]  
    p = p*df*2  
    return frq, p  
  
if __name__ == '__main__':  
    ## 生成随机数据  
    np.random.seed(3)  
    n = 128  
    time = np.arange(n)  
    dat = np.random.rand(n)*10-5+0.3*time  
    x1 = np.sin(12*2*time/n*np.pi)  
    x2 = np.sin( 4*2*time/n*np.pi)  
    dat = dat+5*x2+3*x1  
    ## 调用谱分析  
    frq, p = specx_anal(dat, smooth=3)  
    msk = frq>0  
    frq = frq[msk]  
    p = p[msk]  
    fig = plt.figure(figsize=(11, 8.5))  
    ax = fig.add_subplot(2, 1, 1)  
    ax.plot(time, dat)  
    ax = fig.add_subplot(2, 1, 2)  
    ax.plot(1/frq, p, '-k', linewidth=2)  
    for k in [4, 12]:  
        ax.plot([n/k, n/k], [0, 1500],  
                linestyle='--',  
                color='gray', linewidth=1)
```

```
ax.set_xlim([0,128])
ax.set_ylim([0,1500])
ax.text(2,1300,'(b)',fontsize=20)
plt.show()
```



也可使用Scipy.signal.periodogram()函数进行谱分析

```
In [55]: import numpy as np
from scipy.signal import periodogram,detrend
from scipy import stats
import matplotlib.pyplot as plt

def specx_ci(dat,smooth=0,a=0.05):
    frq, p=periodogram(dat,fs=1,
                        detrend='linear',
                        scaling='density')
    n = len(p)
    df = 2
    if smooth:
        wgt = np.ones(smooth)
        wgt[[0,-1]] = 0.5
        wgt = wgt/np.sum(wgt)
        wgtsq = np.sum(wgt**2)
        df = df/wgtsq
        p0 = p.copy()
        half = smooth//2
        for i in range(half,n-half):
            p0[i]=np.average(p[i-half:i+half+1],
                             weights=wgt)
    p = p0
    dat = detrend(dat)
    phi = np.corrcoef(dat[1:],dat[:-1])[0,1]
```

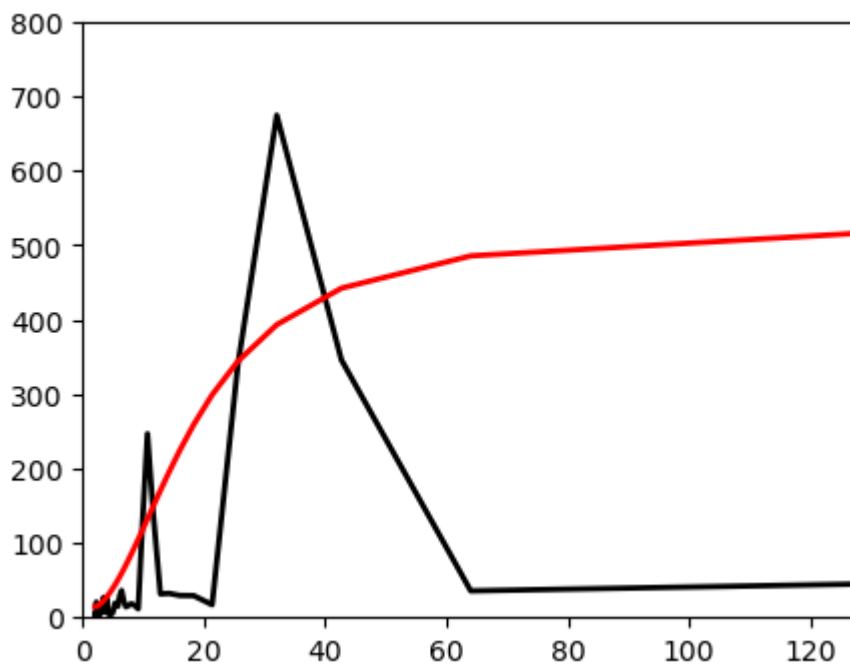
```

n = len(dat)
mkov = 1./(1+phi**2-2*phi*np.cos(2*np.pi*frq))
S = np.sum(p)/np.sum(mkov)*mkov
Sa = S/df*stats.chi2(df).ppf(1-a)
return frq,p,Sa

if __name__=='__main__':
    np.random.seed(3)
    n = 128
    time = np.arange(n)
    dat = np.random.rand(n)*10-5+0.3*time
    x1 = np.sin(12*2*time/n*np.pi)
    x2 = np.sin( 4*2*time/n*np.pi)
    dat = dat+5*x2+3*x1
    frq, p, Sa = specx_ci(dat,smooth=3, a=0.05)
    fig = plt.figure(figsize=(11,8.5))
    ax = fig.add_subplot(2,2,1)
    ax.plot(1/frq,p,'-k',linewidth=2)
    ax.plot(1/frq,Sa,'-r',linewidth=2)
    ax.set_xlim([0,128])
    ax.set_ylim([0,800])
    plt.show()

```

C:\Users\ttzhang\AppData\Local\Temp\ipykernel_14472\750116768.py:41: RuntimeWarning: divide by zero encountered in divide
 ax.plot(1/frq,p,'-k',linewidth=2)
C:\Users\ttzhang\AppData\Local\Temp\ipykernel_14472\750116768.py:42: RuntimeWarning: divide by zero encountered in divide
 ax.plot(1/frq,Sa,'-r',linewidth=2)



6.2 滤波

a) Scipy.signal.savgol_filter()函数

```

In [63]: from scipy import signal
import numpy as np
import matplotlib.pyplot as plt

if __name__=='__main__':

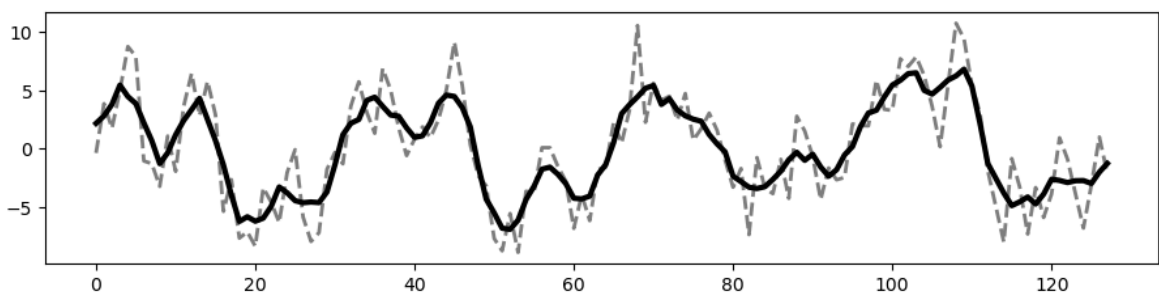
```



```

## 生成数据
np.random.seed(3)
n = 128
time = np.arange(n)
dat = np.random.rand(n)*10-5+0.3*time
x1 = np.sin(12*2*time/n*np.pi)
x2 = np.sin( 4*2*time/n*np.pi)
dat = dat+5*x2+3*x1
dat = signal.detrend(dat)
## 平滑 —savgol_filter()函数第一个参数为数据，参数2表示几点平滑，
## 参数3表示平滑阶数，参数mode为边界处理方式。
dats = signal.savgol_filter(dat,5,1, mode='mirror')
##绘图
fig = plt.figure(figsize=(11,8.5))
ax = fig.add_subplot(3,1,1)
ax.plot(time,dat,linestyle='--',color='gray',linewidth=2)
ax.plot(time,dats,color='k',linewidth=3)

```



b) Butterworth滤波器

In [66]:

```

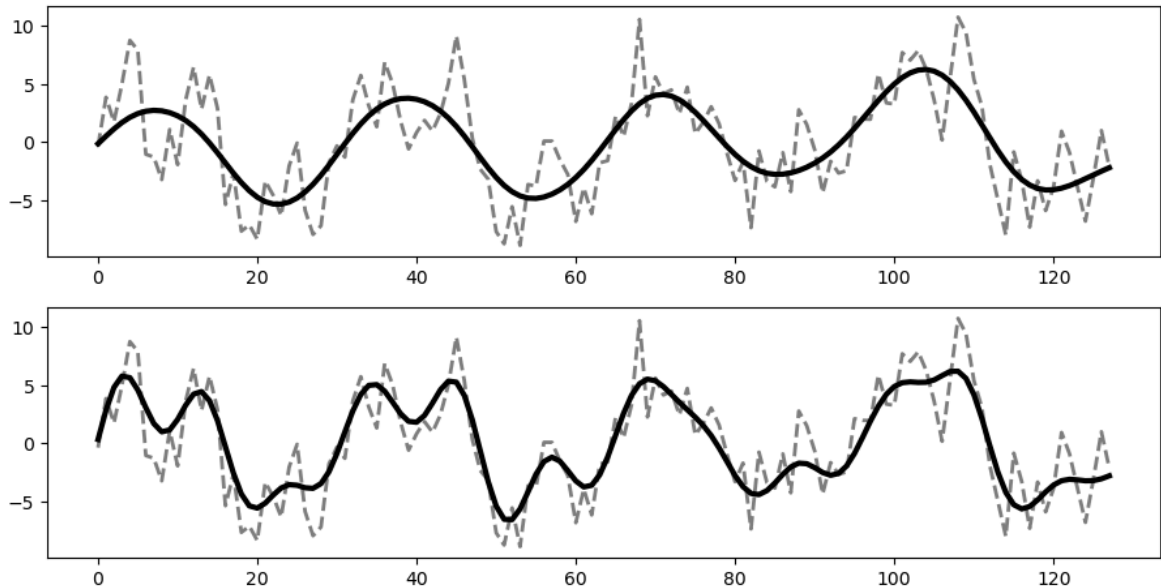
##构造滤波器
##参数1为滤波阶数，常用3，数越大滤波效果越强
##参数2为Nyquist频率，常用半周期的倒数表示
##参数3为滤波方式，有low,high,band
sos = signal.butter(3, 2/20, btype='low', output='sos')
##使用sosfiltfilt函数进行滤波
datf = signal.sosfiltfilt(sos,dat,axis=-1)

fig = plt.figure(figsize=(11,8.5))
ax = fig.add_subplot(3,1,2)
ax.plot(time,dat,linestyle='--',color='gray',linewidth=2)
ax.plot(time,datf,color='k', linewidth=3)

## 带通滤波
sos = signal.butter(3,[2/60,2/10],btype='band',output='sos')
datf = signal.sosfiltfilt(sos,dat,axis=-1)

ax = fig.add_subplot(3,1,3)
dat = signal.detrend(dat)
ax.plot(time,dat,linestyle='--', color='gray',linewidth=2)
ax.plot(time,datf,color='k', linewidth=3)
plt.show()

```



7. EOF分解

EOF常用来提取气象要素的时空变化特征，其核心思想就是求解矩阵的特征值和特征向量。

```
In [71]: import numpy as np
import xarray as xr
from scipy.stats import binned_statistic_2d
import matplotlib.pyplot as plt

## 生成数据的子函数
def crdat():
    lon = np.arange(0,360,2.5)
    lat = np.arange(-90,91,2.5)
    x,y = np.meshgrid(lon,lat)
    m, n = (1,2)
    A = 100*np.sin(n*y/180*np.pi)**2
    dat = []
    for i in range(100):
        tmp = A*np.sin(m*x/180*np.pi-2*np.pi*i/100)
        dat.append(tmp)
    dat = xr.DataArray(dat,dims=('time','lat','lon'),
        coords={'time':np.arange(100),
            'lat':lat,
            'lon':lon})
    return dat

## 计算EOF的子函数
def caleof(dat):
    m = dat.shape
    dat = dat - np.tile(np.mean(dat,0),(m[0],1,1)) ##求距平场
    X = dat.reshape(m[0],-1) ##将三维数据转成“时间×格点”的二维形状
    print(X.shape)
    A = np.dot(X,X.T) ##得到“时间×时间”形状的A矩阵
    lmd, u = np.linalg.eigh(A) ##计算A矩阵的特征值和特征向量
    lmd = lmd[::-1] ## 把上述结果从大到小排列
    u = u[:,::-1]
    V = np.dot(X.T,u)
    V = V/np.tile(np.sqrt(np.abs(lmd)),
        (X.shape[1],1))
```

```

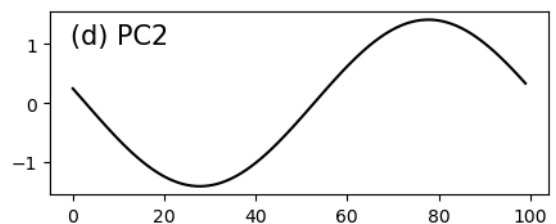
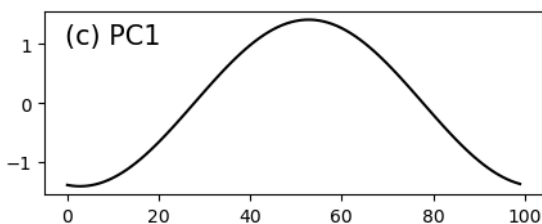
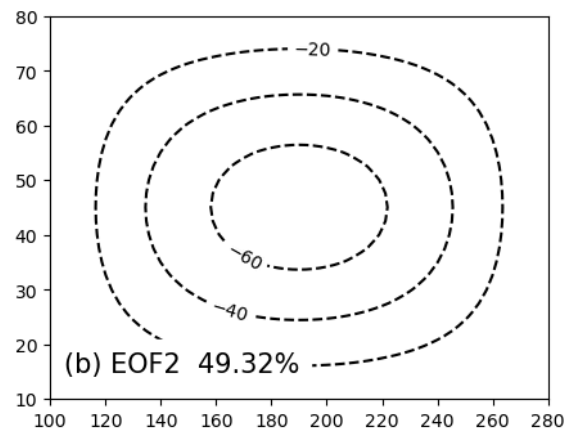
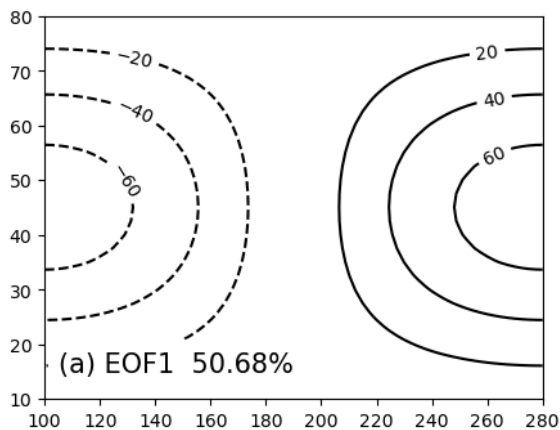
Z = np.dot(X,V) ## 求时间系数
V = V*np.tile(np.std(Z,0),(X.shape[1],1))##特征向量
Z = Z/np.tile(np.std(Z,0),(m[0],1))
V = V.reshape(m[1],m[2],-1)
ro = lmd/np.sum(lmd) ##计算解释方差
return ro,V,Z

if __name__=='__main__':
    dat = crdat()
    dat = dat.loc[:,10:80,100:280]
    ro,V,Z = caleof(dat.to_numpy())
    levels = list(range(-100,100,20))
    levels.remove(0)
    fig = plt.figure(figsize=(11,8.5))
    for i in range(2):
        ax = fig.add_subplot(2,2,i+1)
        c = ax.contour(dat['lon'],dat['lat'],
                       V[:, :, i], levels=levels,
                       colors='k')
        ax.text(105,15,
                '('+chr(ord('a')+i)+') '+' \
                'EOF'+str(i+1)+' '+' \
                '%6.2f' %(ro[i]*100)+'%',
                backgroundcolor='w',
                fontsize=15)
        ax.clabel(c)
        ax = fig.add_subplot(4,2,i+5)
        ax.plot(Z[:,i], '-k')
        ax.text(-0.5,1,
                '('+chr(ord('c')+i)+') '+' \
                'PC'+str(i+1),
                fontsize=15)

    plt.show()

```

(100, 2117)



8. SVD分解

SVD分解用于探讨两个气象要素场之间的联系。

```
In [72]: import numpy as np
import xarray as xr
from scipy.stats import binned_statistic_2d
import matplotlib.pyplot as plt

def crdat():
    lon = np.arange(0,360,2.5)
    lat = np.arange(-90,91,2.5)
    x,y = np.meshgrid(lon,lat)
    m, n = (1,2)
    A = 100*np.sin(n*y/180*np.pi)**2
    B = 150*np.cos(n*y/180*np.pi)**2
    dat = []
    for i in range(100):
        tmp = A*np.sin(m*x/180*np.pi-2*np.pi*i/100)+ \
            B*np.sin(np.random.sample(x.shape)*np.pi)
        dat.append(tmp)
    dat = xr.DataArray(dat,dims=('time','lat','lon'),
        coords={'time':np.arange(100),
            'lat':lat,
            'lon':lon})

    return dat

## 计算非齐次场
def calcor(dat,T):
    m = dat.shape
    T = np.tile(T,(m[1],1)).T
    fnc = lambda x: (x -
        np.tile(np.mean(x,0),(x.shape[0],1)))/ \
        np.tile(np.std(x,0) ,(x.shape[0],1))
    dat = fnc(dat)
    T = fnc(T)
    r = np.mean(dat*T,0)
    return r

## 计算 SVD
def calsvd(dat0,dat1):
    m = dat0.shape
    n = dat1.shape
    X = dat0.reshape(m[0],-1)
    Y = dat1.reshape(n[0],-1)
    A = np.dot(X.T,Y)
    u,s,v = np.linalg.svd(A,full_matrices=False) ##计算SVD
    Tu = np.dot(X,u) ## 左场时间序列
    Tv = np.dot(Y,v.T) ## 右场时间序列
    ro = s/np.sum(s) ## 各模态解释率
    rlft = calcor(X,Tv[:,0])
    rrgt = calcor(Y,Tu[:,0])
    rlft = rlft.reshape(m[1],m[2])
    rrgt = rrgt.reshape(m[1],m[2])
    r = np.corrcoef(Tv[:,0],Tu[:,0])[0,1]
    return r,ro,rlft,rrgt

def smth9(dat):
    tmp = dat.copy()
    tmp[1:-1,1:-1] = dat[1:-1,1:-1]*1.0+ \
        dat[:-2,1:-1]*0.5+dat[2:,1:-1]*0.5+ \
```

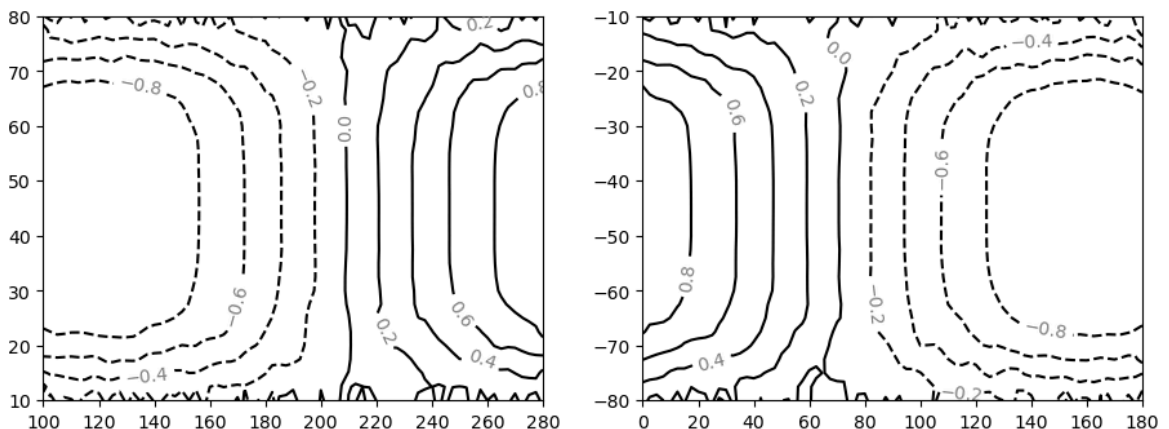
```

        dat[1:-1,:-2]*0.5+dat[1:-1,2:]*0.5+ \
        dat[:-2,:-2]*0.3+dat[2:, 2:]*0.3+ \
        dat[2:, :-2]*0.3+dat[: -2,2:]*0.3
    tmp[1:-1,1:-1] = tmp[1:-1,1:-1]/(1+0.5*4+0.3*4)
    return tmp

if __name__=='__main__':
    dat = crdat()
    dat0 = dat.loc[:,10:80,100:280]
    dat1 = dat.loc[:, -80:-10,0:180]
    r,ro,rlft, rrgt = calsvd(dat0.to_numpy(),
        dat1.to_numpy())
    print(r,ro[0])
    fig = plt.figure(figsize=(11,8.5))
    levels = list(np.arange(-1,1,0.2))
    rlft = smth9(rlft)
    rrgt = smth9(rrgt)
    ax = fig.add_subplot(2,2,1)
    c = ax.contour(dat0['lon'],dat0['lat'],rlft,
        levels=levels,colors='k')
    ax.clabel(c,colors='gray')
    ax = fig.add_subplot(2,2,2)
    c = ax.contour(dat1['lon'],dat1['lat'],rrgt,
        levels=levels,colors='k')
    ax.clabel(c,colors='gray')
    plt.show()

```

0.7504128176668776 0.46596437957082554



9. 聚类分析

聚类分析是气象上常用的统计方法之一，用于将具有相同特性的数值聚在一起，特别是在进行天气/气候的分型时使用非常方便，本小节仅介绍目前最常用的方法之一，K-means方法的使用。该方法仅需确定聚类的k值，即类别的个数。

```

In [3]: import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

def crdat():
    x0 = np.arange(100)*360/100
    y0 = np.arange(100)*360/100
    x, y = np.meshgrid(x0,y0)
    V = []
    for i in range(2):
        for j in range(2):

```

```

        c = np.sin(0.5*(j+1)*x/180*np.pi)* \
            np.sin(0.5*(i+1)*y/180*np.pi)
        c = c/np.sqrt(np.sum(c**2))
        V.append(c.flatten())

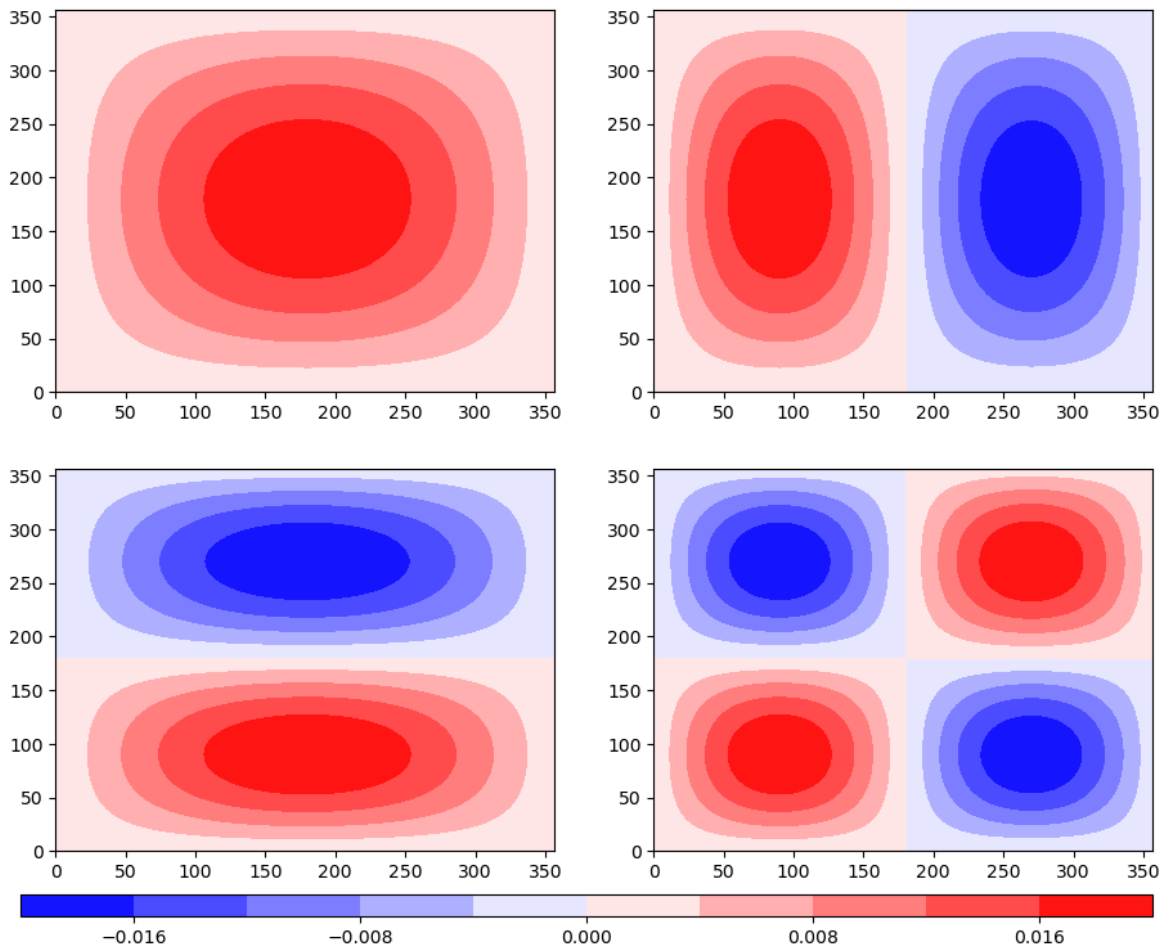
n = 250
X = []
for i in range(4):
    theta = np.random.rand(n)*2*np.pi
    z = 100*np.sin(theta)
    z = z.reshape(-1,1)
    v = V[i].reshape(1,-1)
    X.append(np.dot(z,v))
X = np.array(X)
X = X.reshape(-1,len(y0),len(x0))
I = np.arange(X.shape[0])
np.random.shuffle(I)
X = X[I]
V = np.reshape(V,(4,len(y0),len(x0)))
return x,y,X,V

```

```

In [10]: x,y,X,V = crdat()
fig = plt.figure(figsize=(11,8.5))
for i in range(4):
    ax = fig.add_subplot(2,2,i+1)
    h = ax.contourf(x,y,V[i],levels=np.linspace(-0.02,0.02,11),
                    cmap='bwr')
cax = fig.add_axes([0.1,0.05,0.8,0.02])
plt.colorbar(h,cax=cax,orientation='horizontal')
plt.show()

```



```
In [11]: F = X.reshape(X.shape[0],-1)
print(F.shape)
```

(1000, 10000)

```
In [14]: model = KMeans(n_clusters=8,init='k-means++')
kmeans = model.fit(F)
lb = kmeans.labels_
print(lb)
```

```
[2 4 2 5 4 1 1 3 4 0 6 1 1 1 6 4 1 1 5 4 1 4 1 1 6 1 7 1 3 4 7 1 2 1 1 7 3
0 4 5 5 1 1 1 6 6 2 4 2 1 1 6 2 6 1 5 1 6 7 0 1 2 1 0 2 7 3 0 5 1 7 3 2 1
0 5 5 3 1 1 5 5 3 1 1 0 3 7 2 1 1 1 3 1 2 3 2 5 4 6 1 6 5 4 5 1 3 1 1 7 0
0 2 1 5 3 7 0 1 7 1 0 1 2 1 7 6 4 1 0 6 1 4 2 1 1 5 5 7 0 0 5 0 0 7 1 3 1
0 1 2 1 1 4 0 1 2 7 0 1 3 2 1 2 0 0 1 4 7 0 7 4 1 0 2 5 5 1 3 0 6 1 2 4 3
5 1 5 6 1 7 0 5 1 1 7 1 1 6 1 6 7 2 4 2 7 3 3 1 3 4 0 2 7 1 0 3 2 1 1 7 3
2 7 0 4 5 1 3 4 0 5 1 7 5 0 0 3 1 2 3 6 1 1 3 1 0 0 0 1 4 5 3 7 4 4 2 5 4
0 4 1 7 6 1 6 0 4 0 7 1 4 1 1 1 0 0 4 3 3 3 0 1 6 0 1 5 4 1 0 6 4 7 5 1 1
3 5 1 7 7 6 7 1 0 5 7 2 1 0 7 1 1 7 1 3 6 6 1 1 3 4 1 1 3 2 1 3 7 3 2 0 5
5 2 2 2 1 2 5 0 2 4 5 1 4 1 6 0 1 7 1 6 4 5 1 4 1 7 6 4 1 1 1 2 3 0 1 0 7
5 1 1 6 1 6 1 4 1 4 3 1 3 1 1 6 3 6 1 1 1 1 6 1 4 6 1 0 3 2 1 6 4 2 1 1 7
7 3 6 4 2 1 1 3 6 1 1 2 0 7 0 5 1 7 1 0 0 3 2 1 7 6 5 7 1 1 6 7 2 4 1 2 4
1 6 1 5 6 1 0 5 0 1 4 3 6 6 0 1 2 4 1 2 1 6 1 4 2 6 4 1 1 5 1 1 5 2 1 5 5
4 3 2 1 1 3 2 1 4 1 1 1 4 2 1 1 2 1 1 1 7 5 1 0 2 5 6 3 4 4 0 2 4 5 2 1 4
4 7 1 0 1 1 4 0 3 5 0 6 4 0 1 7 0 7 0 3 1 4 2 1 7 1 3 1 3 1 2 5 7 7 6 1 5
1 3 5 6 1 7 4 1 6 1 2 0 5 1 1 1 4 2 1 1 5 6 7 1 5 6 6 1 4 7 5 1 0 7 2 1 0
6 2 7 4 1 5 3 0 4 5 0 0 3 4 4 2 1 7 6 1 1 1 2 1 1 3 1 0 2 4 0 1 3 4 6 5 1
4 4 7 5 0 5 7 1 2 6 1 1 5 1 1 7 0 7 1 1 1 5 5 1 2 1 5 3 1 1 3 1 2 0 1 7 1
0 1 7 2 1 5 5 5 1 5 5 6 7 4 2 1 7 1 2 6 0 1 7 2 3 1 1 4 3 2 6 4 7 4 1 1 0
1 3 2 6 1 0 4 3 0 5 2 6 1 1 6 7 6 1 3 4 7 1 2 4 1 1 4 6 4 7 2 7 1 1 7 5 2
6 6 4 2 1 1 6 4 1 5 5 6 6 1 3 1 7 6 4 2 3 7 4 1 1 3 4 7 6 2 1 5 3 6 1 5 0
2 1 0 1 7 1 1 4 1 4 3 2 5 3 1 1 6 1 5 1 5 3 2 3 6 1 1 5 1 5 2 1 1 1 6 1 5
7 6 1 2 0 7 6 1 1 1 7 1 2 5 7 7 3 3 1 1 7 1 1 0 1 2 2 6 1 2 1 7 0 1 0 1 1
1 5 1 0 4 7 1 7 1 0 1 7 5 2 4 4 1 1 3 1 1 2 1 1 2 1 6 5 0 5 5 1 2 1 5 7 7
3 1 0 3 4 5 2 1 2 6 1 2 3 1 3 1 1 1 3 6 5 2 3 7 0 1 1 4 4 1 1 1 1 6 4 1 5
5 5 7 1 1 0 1 6 7 1 1 5 5 1 6 5 2 7 0 2 0 1 7 2 2 2 2 1 3 7 1 1 2 0 4 5 3
1 1 0 0 7 4 5 6 1 0 3 5 1 1 7 1 1 1 0 4 2 7 7 3 0 6 1 5 1 6 4 1 2 6 1 3 7
0]
```

```
C:\ProgramData\Anaconda3\envs\mete\Lib\site-packages\sklearn\cluster\_kmeans.py:1
429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=4.
warnings.warn(
```

```
In [19]: fig = plt.figure(figsize=(11,8.5))
for i in range(8):
    msk = lb==i
    ax = fig.add_subplot(3,3,i+1)
    h = ax.contourf(x,y,np.mean(X[msk],0),levels=np.linspace(-2,2,21),
                    extend='both',cmap='bwr')
cax = fig.add_axes([0.1,0.05,0.8,0.02])
plt.colorbar(h,cax=cax,orientation='horizontal')
plt.show()
```

