hw06-WALB

June 8, 2016

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from scipy.io import wavfile
        from scipy.signal import periodogram
        %matplotlib inline
```

1 Machine Intelligence II

1.1 Exercise Sheet 06: Independent Component Analysis

1.1.1 Group WALB: Wichert, Alevi, Lang, Boelts

1.1.2 6.1

```
In [2]: # (a)
        rate = 8192
        p = 18000
        sound1 = np.loadtxt('sound1.dat')
        sound2 = np.loadtxt('sound2.dat')
        sounds = np.vstack([sound1, sound2])
        wavfile.write('original1.wav', rate, sound1)
        wavfile.write('original2.wav', rate, sound2)
```

```
In [3]: # (b)
        A = np.linalg.inv(np.random.random(size=(2,2)))
        X = A.dot(sounds)
```

```
In [4]: # (c)
        dist = X # for 6.3 (a) (ii)
        X = X.T
        np.random.shuffle(X)
        X = X.T
        dist_perm = X # for 6.3 (a) (ii)
```

```
In [5]: # (d)
        for i in range(2):
            for j in range(2):
                corr = np.cov(sounds[i,:], X[j,:]) / sounds[i,:].var() / X[j,:].var()
                print('correlation matrix between s_{} and x_{}:\n {}\n'.format(i, j, corr))
```

```
correlation matrix between s_0 and x_0:
 [[ 0.11978564 -0.00500343]
 [-0.00500343  1.0029702 ]]

correlation matrix between s_0 and x_1:
```

```
 [[  2.60444046e-01   7.91085491e-04]
 [  7.91085491e-04   1.00297020e+00]]

correlation matrix between s_1 and x_0:
 [[  1.19785638e-01   8.90394146e-04]
 [  8.90394146e-04   1.00235094e+00]]

correlation matrix between s_1 and x_1:
 [[ 0.26044405  0.00158761]
 [ 0.00158761  1.00235094]]
```

```
In [6]: # (e)
        X = X - X.mean(axis=1).reshape(2,1)
```

```
In [7]: # (f)
        W = np.linalg.inv(np.random.random(size=(2,2)))
        W_nat = np.linalg.inv(np.random.random(size=(2,2)))
```

1.1.3 6.2

```
In [8]: def f(y):
            return 1 / (1 + np.exp(-y))

        def phi(y):
            return 1 - 2 * f(y)
```

```
In [9]: # (a)
        def update_regular(W, x, eta):
            W_inv = np.linalg.inv(W)
            delta_W = W_inv.T + phi(W.dot(x)).reshape(2,1).dot(x.reshape(1,2))
            return W + eta * delta_W,  delta_W, eta * delta_W
```

```
In [10]: # (b)
         def update_natural(W, x, eta):
             phee = phi(W.dot(x)).reshape(2,1)
             delta_W = np.dot(phee.dot(np.dot(W, x).reshape(1,2)), W)
             delta_W = delta_W + W # multiplied out delta function
             delta_W[0,0] = 0 # Bell-Seijnowski solution
             delta_W[1,1] = 0 # Bell-Seijnowski solution
             return W + eta * delta_W,  delta_W, eta * delta_W
```

```
In [11]: # (c)
         eta_0 = 20
         delta_W_norms = []
         delta_W_norms_eta = []
         for t in range(X.shape[1]):
             eta = eta_0 / (t + 1)
             x = X[:,t]
             W, delta_W, delta_W_eta = update_regular(W, x, eta)
             if t % 1000 == 0:
                 delta_W_norms.append(np.sum(delta_W ** 2)) # for 6.3 (c)
                 delta_W_norms_eta.append(np.sum(delta_W_eta ** 2)) # for 6.3 (c)

         rec = W.dot(A.dot(sounds))

         eta_0 = .15
```

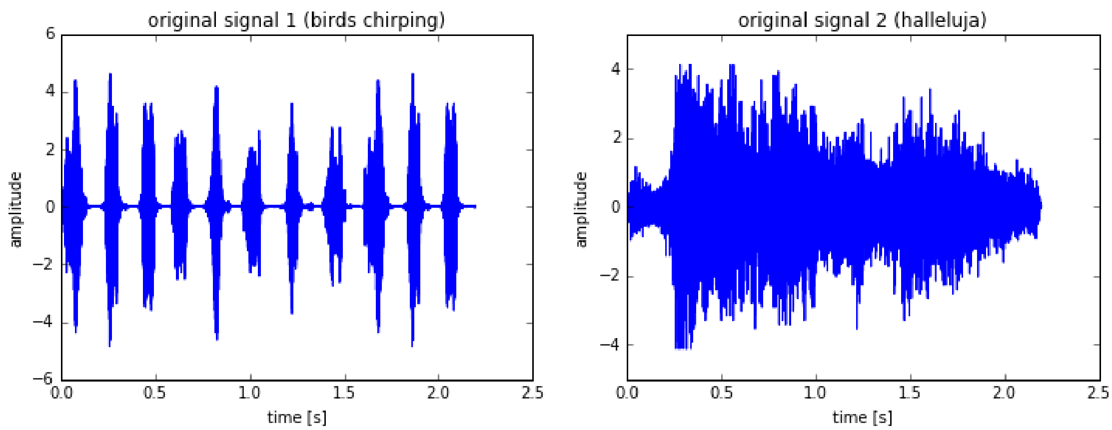
```
delta_W_norms_nat = []
delta_W_norms_nat_eta = []
W_nat = np.linalg.inv(np.random.random(size=(2,2)))
W_nat[0,0] = 1 # Bell-Seijnowski solution
W_nat[1,1] = 1 # Bell-Seijnowski solution
for t in range(X.shape[1]):
    eta = eta_0 / (t + 1)
    x = X[:,t]
    W_nat, delta_W_nat, delta_W_nat_eta = update_natural(W_nat, x, eta)
    assert not np.isnan(W_nat[0,0])
    if t % 1000 == 0:
        delta_W_norms_nat.append(np.sum(delta_W_nat ** 2)) # for 6.3 (c)
        delta_W_norms_nat_eta.append(np.sum(delta_W_nat_eta ** 2)) # for 6.3 (c)

rec_nat = W_nat.dot(A.dot(sounds))
```

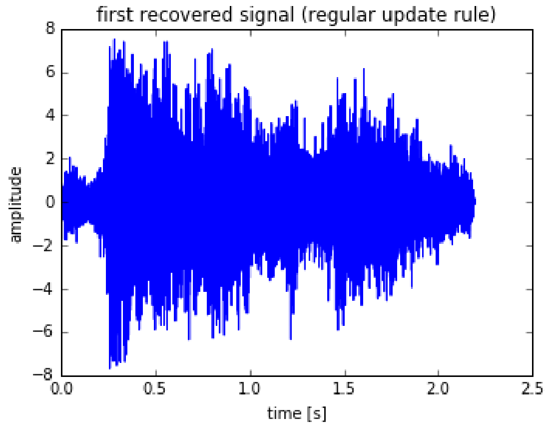
1.1.4 6.3

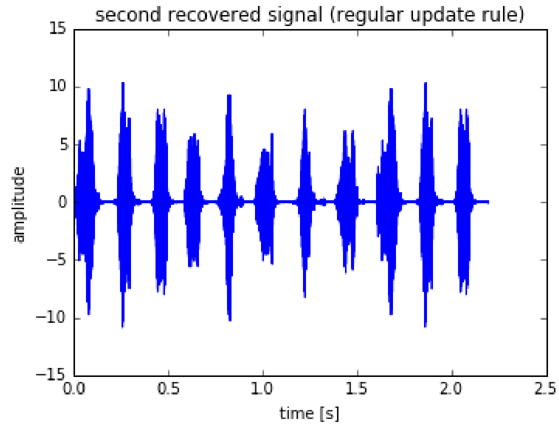
```
In [12]: # (a)

         wavfile.write('rec1.wav', rate, rec[0,:])
         wavfile.write('rec2.wav', rate, rec[1,:])
         wavfile.write('rec_nat1.wav', rate, rec_nat[0,:])
         wavfile.write('rec_nat2.wav', rate, rec_nat[1,:])
         wavfile.write('dist1.wav', rate, dist[0,:])
         wavfile.write('dist2.wav', rate, dist[1,:])
         wavfile.write('dist_perm1.wav', rate, dist_perm[0,:])
         wavfile.write('dist_perm2.wav', rate, dist_perm[1,:])
```

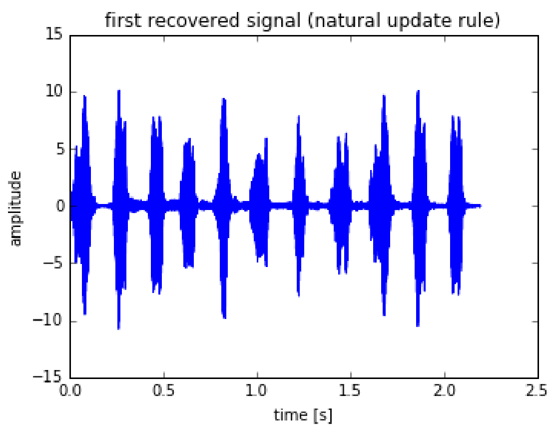
```
In [13]: time = np.arange(p) / rate

         plt.figure(figsize=(12,4))
         plt.subplot('121')
         plt.plot(time, sound1)
         plt.xlabel('time [s]')
         plt.ylabel('amplitude')
         plt.title('original signal 1 (birds chirping)')
         plt.subplot('122')
         plt.plot(time, sound2)
         plt.xlabel('time [s]')
         plt.ylabel('amplitude')
         plt.title('original signal 2 (halleluja)')

         plt.figure(figsize=(12,4))
         plt.subplot('121')
         plt.plot(time, rec[0,:])
         plt.xlabel('time [s]')
         plt.ylabel('amplitude')
         plt.title('first recovered signal (regular update rule)')
         plt.subplot('122')
         plt.plot(time, rec[1,:])
         plt.xlabel('time [s]')
         plt.ylabel('amplitude')
         plt.title('second recovered signal (regular update rule)')
```

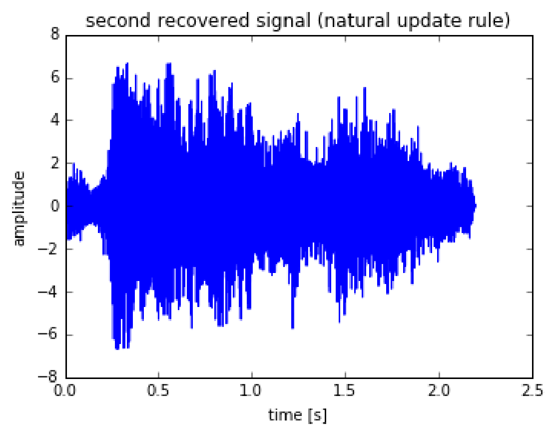
```
plt.figure(figsize=(12,4))
plt.subplot('121')
plt.plot(time, rec_nat[0,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('first recovered signal (natural update rule)')
plt.subplot('122')
plt.plot(time, rec_nat[1,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('second recovered signal (natural update rule)')

plt.figure(figsize=(12,4))
plt.subplot('121')
plt.plot(time, dist[0,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('mixed sources before permutation')
plt.subplot('122')
plt.plot(time, dist[1,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('mixed sources before permutation')

plt.figure(figsize=(12,4))
plt.subplot('121')
plt.plot(time, dist_perm[0,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('mixed sources after permutation')
plt.subplot('122')
plt.plot(time, dist_perm[1,:])
plt.xlabel('time [s]')
plt.ylabel('amplitude')
plt.title('mixed sources after permutation');
```

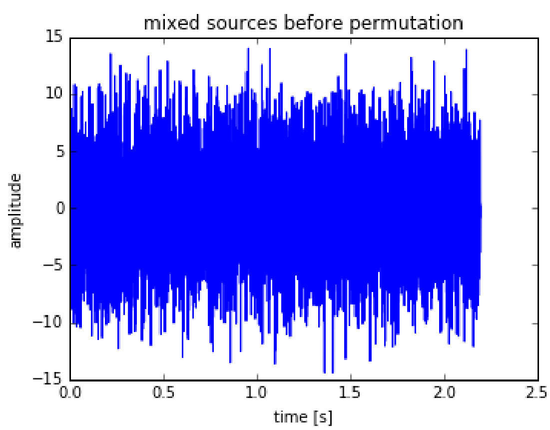


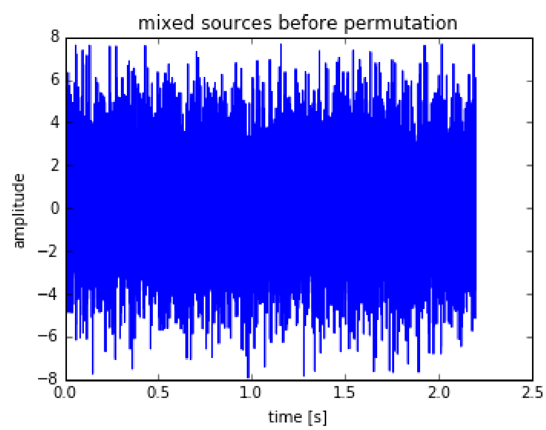

first recovered signal (regular update rule)
amplitude
time [s]
8 6 4 2 0 −2 −4 −6 −8
0.0 0.5 1.0 1.5 2.0 2.5

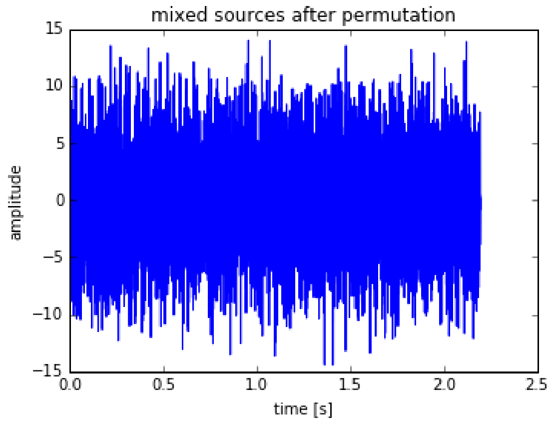

second recovered signal (regular update rule)
amplitude
time [s]
15 10 5 0 −5 −10 −15
0.0 0.5 1.0 1.5 2.0 2.5

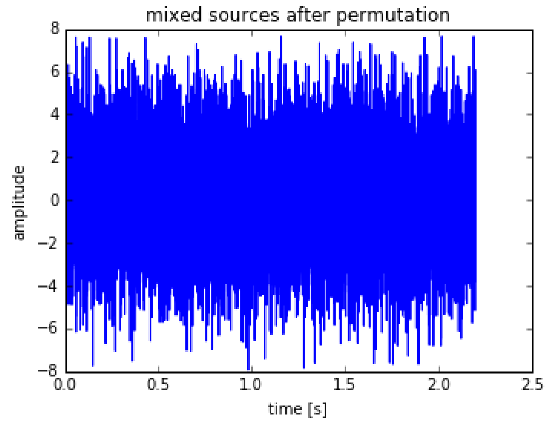

first recovered signal (natural update rule)
amplitude
time [s]
15 10 5 0 −5 −10 −15
0.0 0.5 1.0 1.5 2.0 2.5


second recovered signal (natural update rule)
amplitude
time [s]
8 6 4 2 0 −2 −4 −6 −8
0.0 0.5 1.0 1.5 2.0 2.5


mixed sources before permutation
amplitude
time [s]
15 10 5 0 −5 −10 −15
0.0 0.5 1.0 1.5 2.0 2.5


mixed sources before permutation
amplitude
time [s]
8 6 4 2 0 −2 −4 −6 −8
0.0 0.5 1.0 1.5 2.0 2.5



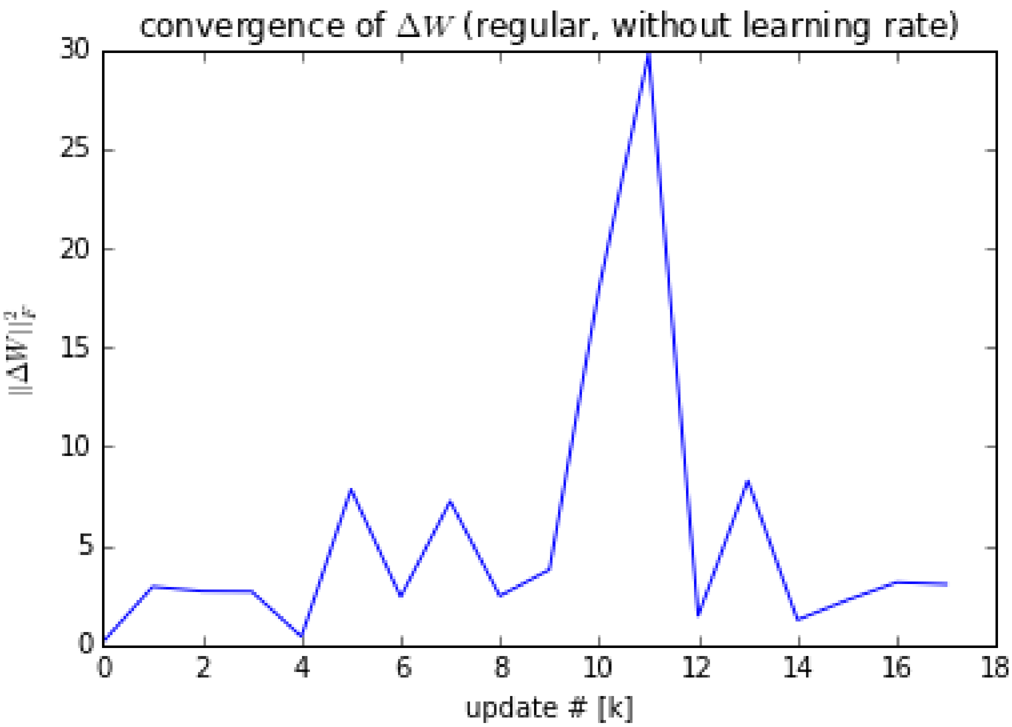


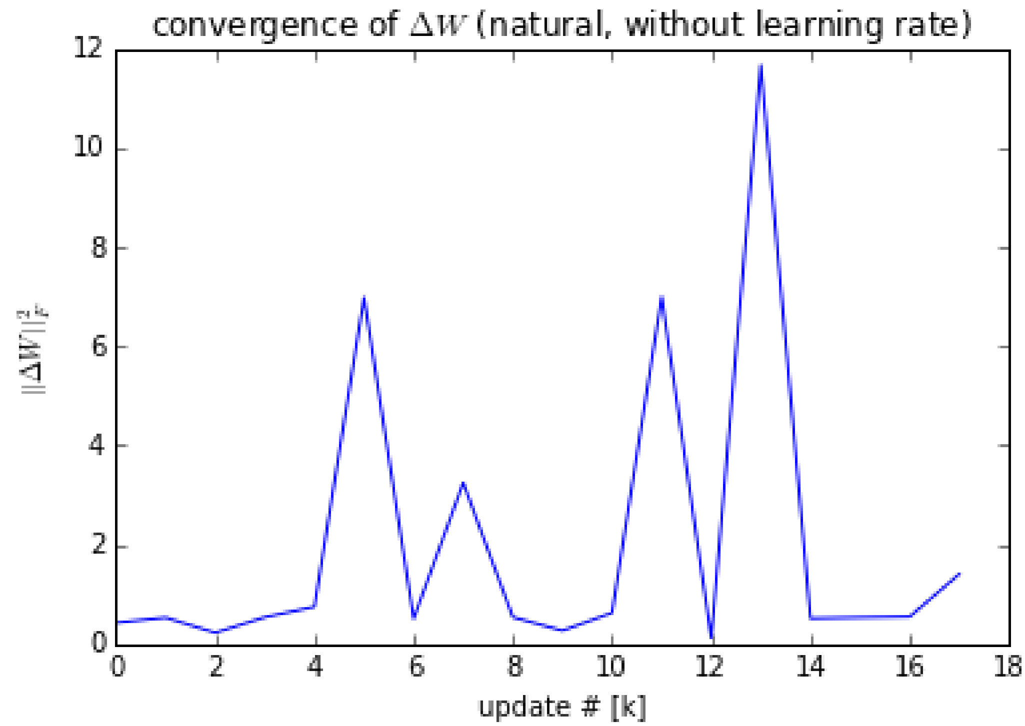
In [14]:
```
# (b)

for i in range(2):
    for j in range(2):
        corr = np.cov(sounds[i,:], rec[j,:]) / sounds[i,:].var() / rec[j,:].var()
        print('correlation matrix between s_{} and rec_{}:\n {}\n'.format(i, j, corr))
```

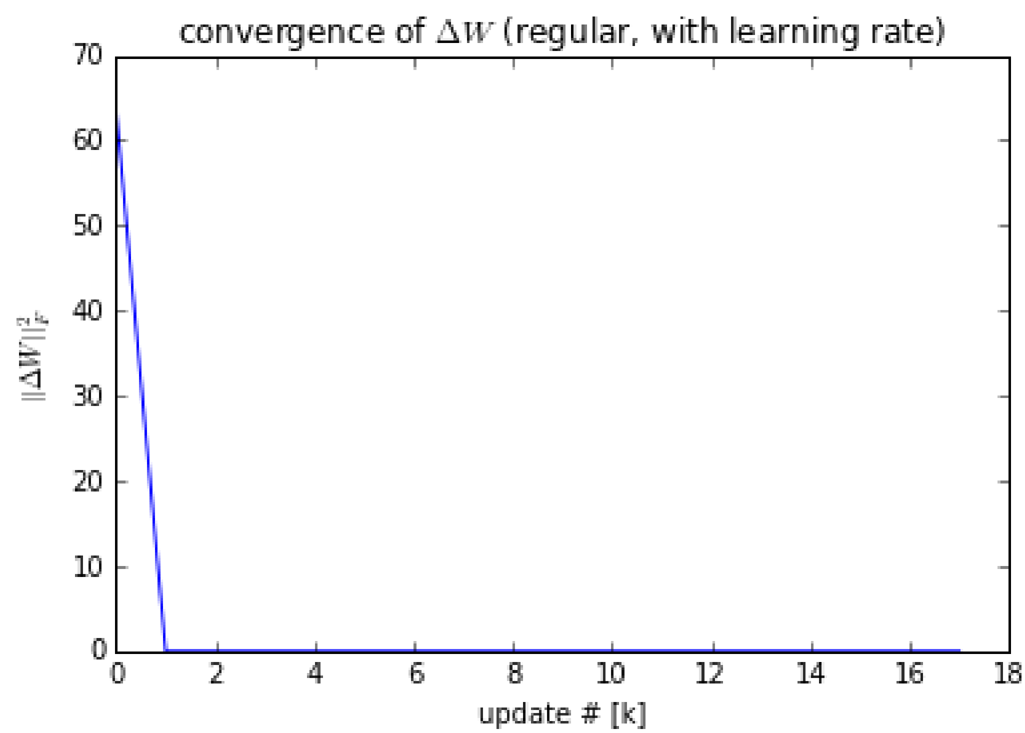
```
correlation matrix between s_0 and rec_0:
 [[ 0.30971707  0.02873851]
 [ 0.02873851  1.0029702 ]]

correlation matrix between s_0 and rec_1:
 [[ 0.20198591  0.45008551]
 [ 0.45008551  1.0029702 ]]

correlation matrix between s_1 and rec_0:
 [[ 0.30971707  0.55647067]
 [ 0.55647067  1.00235094]]

correlation matrix between s_1 and rec_1:
 [[ 0.20198591  0.00354693]
 [ 0.00354693  1.00235094]]
```

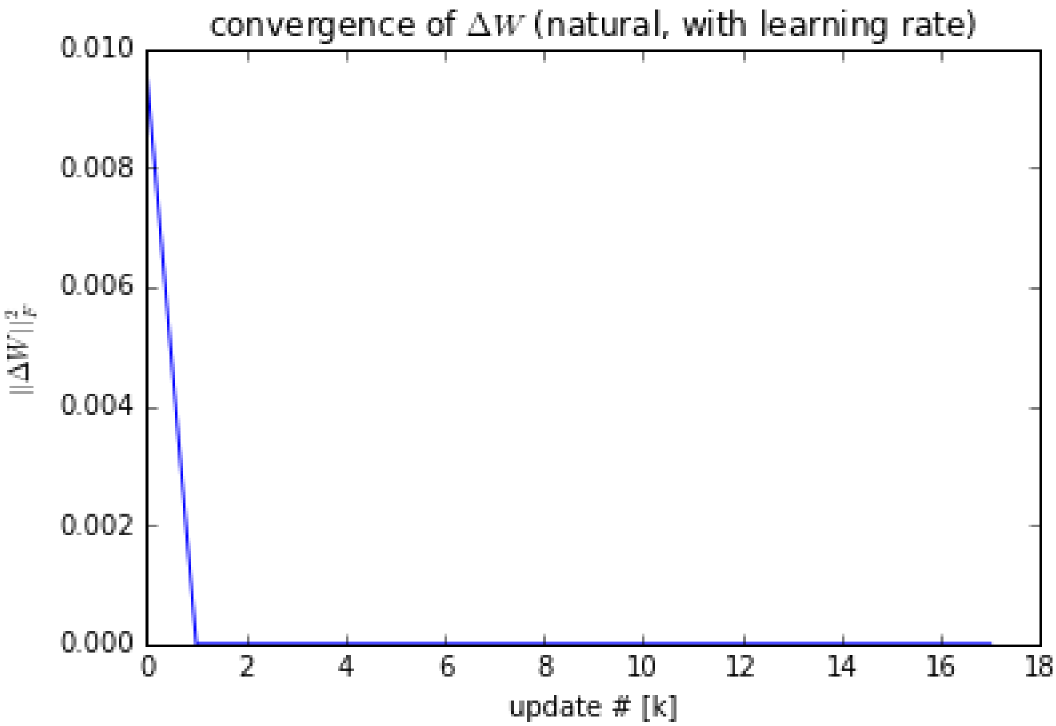
In [15]:
```
# (c)

plt.plot(delta_W_norms)
plt.title(r'convergence of $\Delta W$ (regular, without learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
plt.plot(delta_W_norms_nat)
plt.title(r'convergence of $\Delta W$ (natural, without learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
```

```
plt.plot(delta_W_norms_eta)
plt.title(r'convergence of $\Delta W$ (regular, with learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
plt.plot(delta_W_norms_nat_eta)
plt.title(r'convergence of $\Delta W$ (natural, with learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$');
```



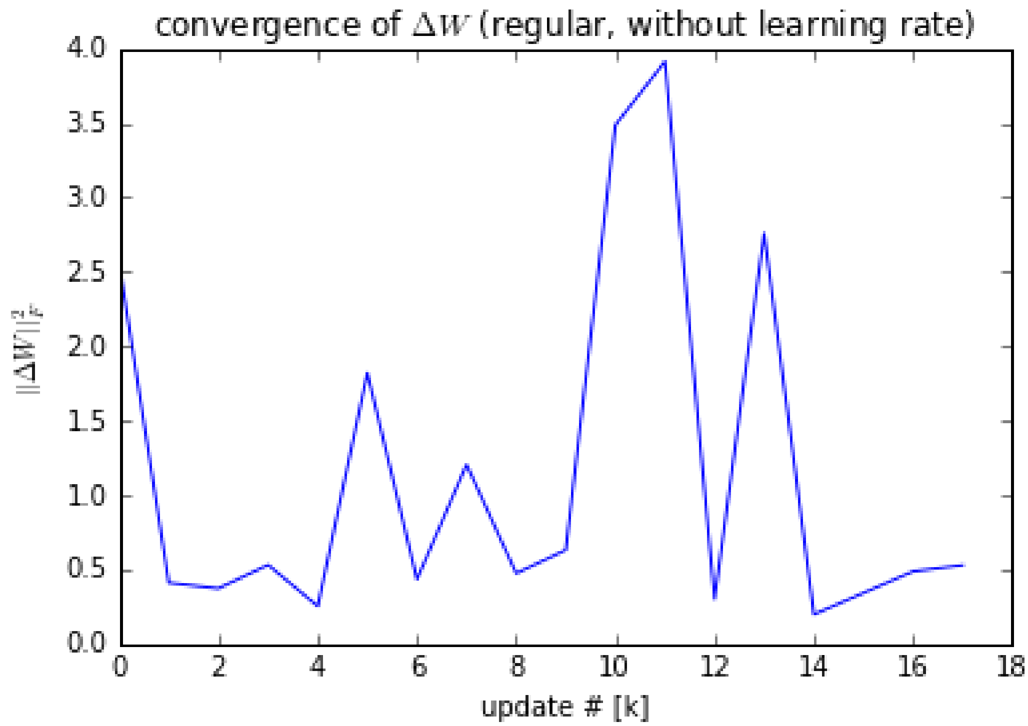

convergence of ΔW (natural, without learning rate)
$||\Delta W||_F^2$
update # [k]
0
2
4
6
8
10
12
14
16
18


convergence of ΔW (regular, with learning rate)
$||\Delta W||_F^2$
update # [k]
0
10
20
30
40
50
60
70
2
4
6
8
12
14
16
18

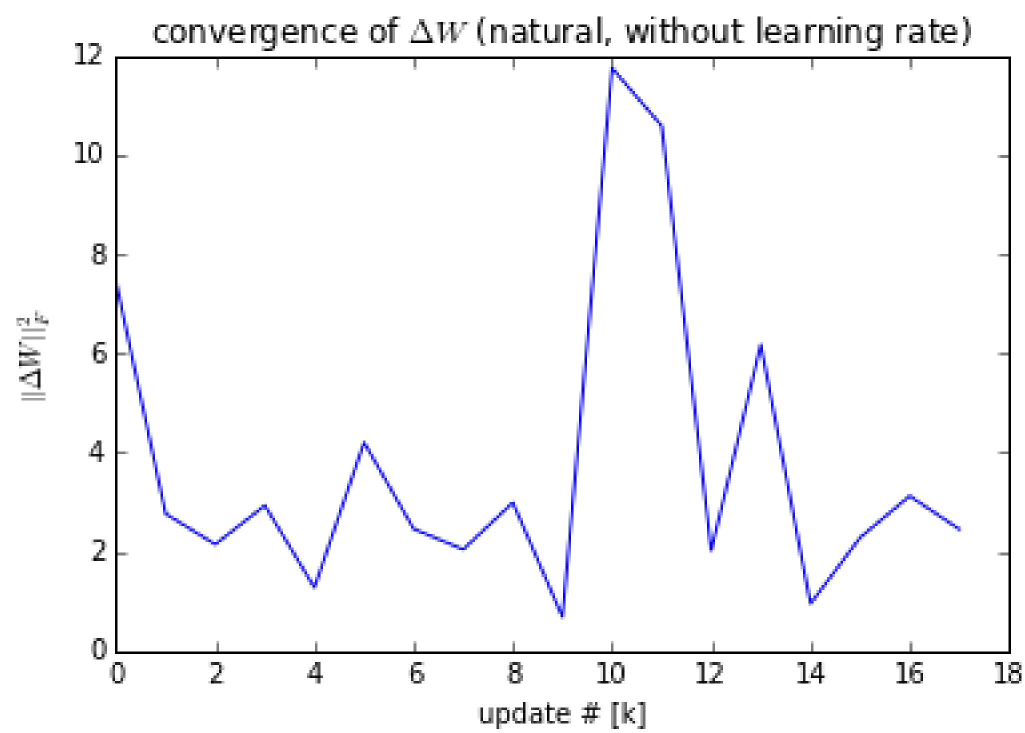



```
In [16]: # (c) (continued)

         def whiten_data(data):
             C = np.cov(data)
             w, V = np.linalg.eigh(C)
             D = np.diag(1 / np.sqrt(w))
             M = D.dot(V)
             return M.dot(data)

         X_white = whiten_data(X)

         eta_0 = 20
         delta_W_norms = []
         delta_W_norms_eta = []
         for t in range(X_white.shape[1]):
             eta = eta_0 / (t + 1)
             x = X_white[:,t]
             W, delta_W, delta_W_eta = update_regular(W, x, eta)
             if t % 1000 == 0:
                 delta_W_norms.append(np.sum(delta_W ** 2)) # for 6.3 (c)
                 delta_W_norms_eta.append(np.sum(delta_W_eta ** 2)) # for 6.3 (c)

         rec = W.dot(A.dot(sounds))

         eta_0 = .15
         delta_W_norms_nat = []
         delta_W_norms_nat_eta = []
```

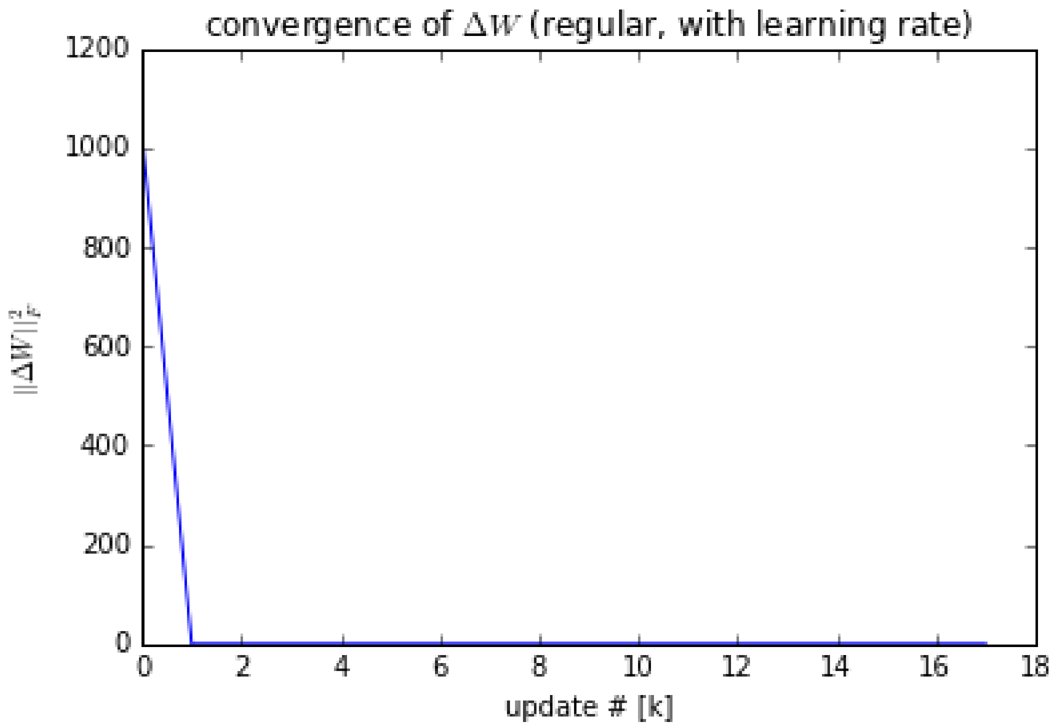
```python
W_nat = np.linalg.inv(np.random.random(size=(2,2)))
W_nat[0,0] = 1 # Bell-Seijnowski solution
W_nat[1,1] = 1 # Bell-Seijnowski solution
for t in range(X_white.shape[1]):
    eta = eta_0 / (t + 1)
    x = X_white[:,t]
    W_nat, delta_W_nat, delta_W_nat_eta = update_natural(W_nat, x, eta)
    assert not np.isnan(W_nat[0,0])
    if t % 1000 == 0:
        delta_W_norms_nat.append(np.sum(delta_W_nat ** 2)) # for 6.3 (c)
        delta_W_norms_nat_eta.append(np.sum(delta_W_nat_eta ** 2)) # for 6.3 (c)

rec_nat = W_nat.dot(A.dot(sounds))

plt.plot(delta_W_norms)
plt.title(r'convergence of $\Delta W$ (regular, without learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
plt.plot(delta_W_norms_nat)
plt.title(r'convergence of $\Delta W$ (natural, without learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
plt.plot(delta_W_norms_eta)
plt.title(r'convergence of $\Delta W$ (regular, with learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$')

plt.figure()
plt.plot(delta_W_norms_nat_eta)
plt.title(r'convergence of $\Delta W$ (natural, with learning rate)')
plt.xlabel('update # [k]')
plt.ylabel(r'$||\Delta W||^2_F$');
```

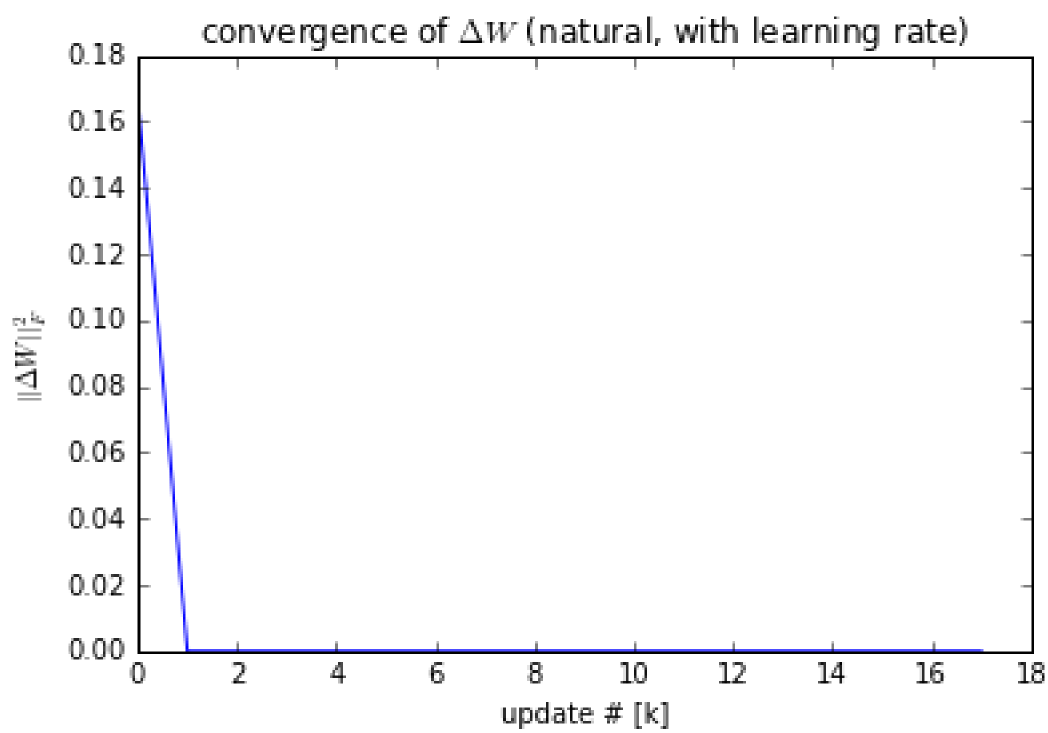


convergence of ΔW (regular, without learning rate)
$||\Delta W||_F^2$
update # [k]
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
2
4
6
8
10
12
14
16
18

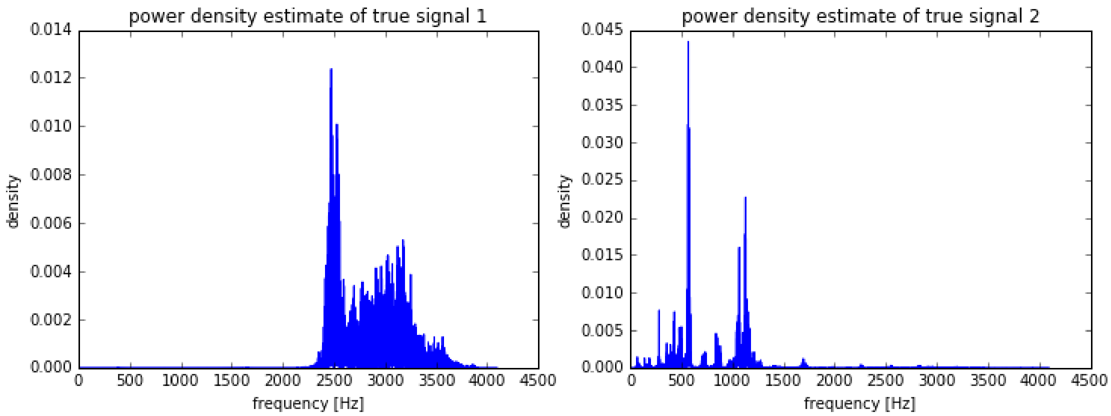


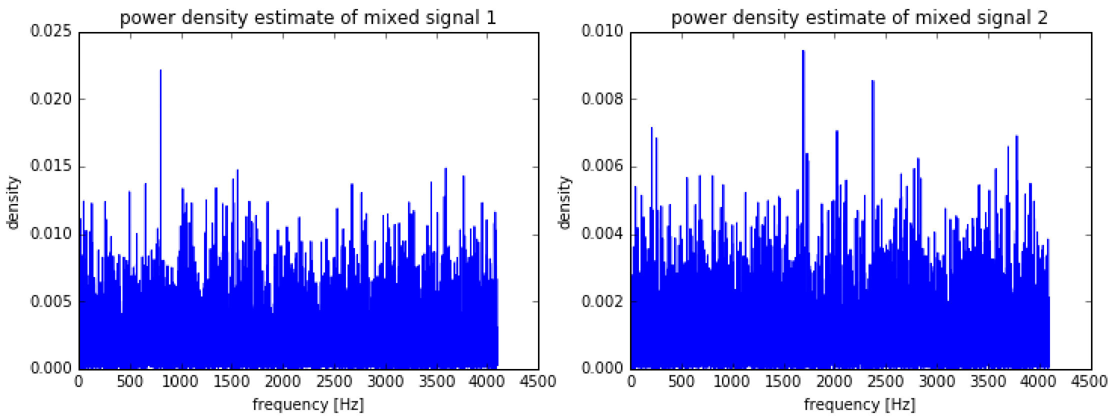
convergence of ΔW (natural, without learning rate)
$||\Delta W||_F^2$
update # [k]
0
2
4
6
8
10
12
0
2
4
6
8
10
12
14
16
18

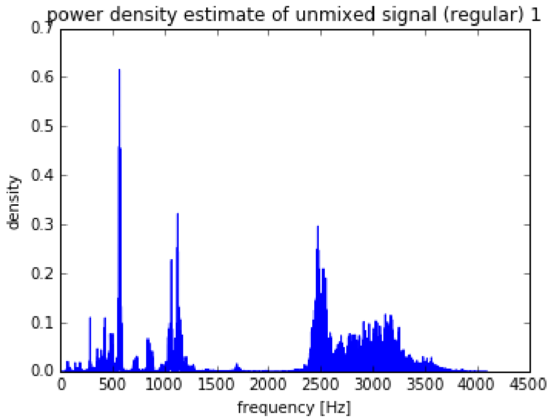

convergence of ΔW (regular, with learning rate)
$||\Delta W||_F^2$
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
14
16
18
update # [k]

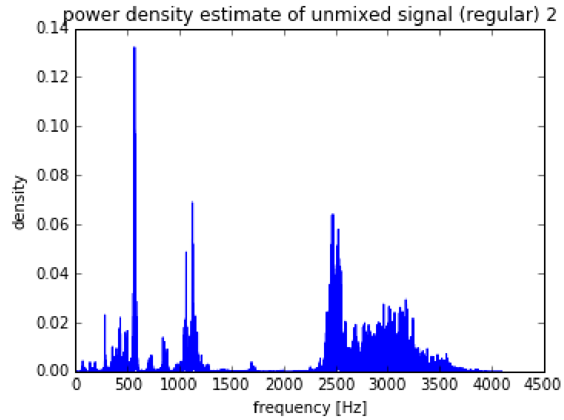

convergence of ΔW (natural, with learning rate)
$||\Delta W||_F^2$
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
14
16
18
update # [k]

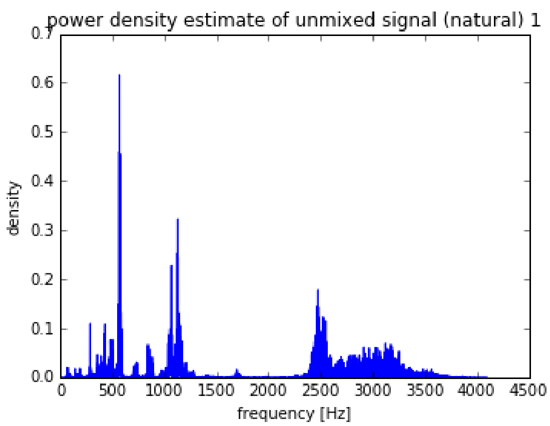
```
In [17]: # (d)

         def plot_density(signal1, signal2, name):
             f1, PXX_den1 = periodogram(signal1, rate)
             f2, PXX_den2 = periodogram(signal2, rate)
             plt.figure(figsize=(12,4))
             plt.subplot('121')
             plt.plot(f1, PXX_den1)
             plt.title('power density estimate of {} 1'.format(name))
             plt.xlabel('frequency [Hz]')
             plt.ylabel('density')
             plt.subplot('122')
             plt.plot(f2, PXX_den2)
             plt.title('power density estimate of {} 2'.format(name))
             plt.xlabel('frequency [Hz]')
             plt.ylabel('density')

         plot_density(sound1, sound2, 'true signal')
         plot_density(dist[0,:], dist[1,:], 'mixed signal')
         plot_density(rec[0,:], rec[1,:], 'unmixed signal (regular)')
         plot_density(rec_nat[0,:], rec_nat[1,:], 'unmixed signal (natural)')
```

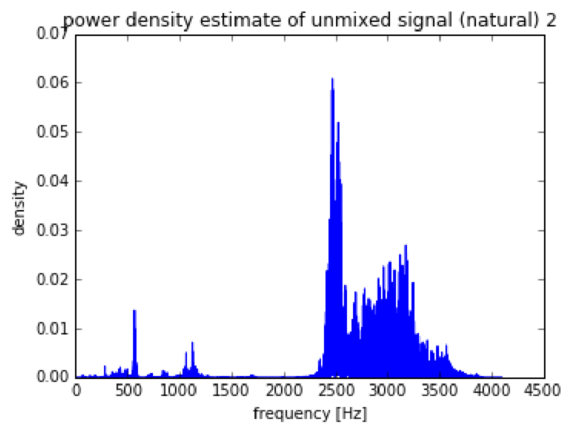





power density estimate of unmixed signal (regular) 1
density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0 500 1000 1500 2000 2500 3000 3500 4000 4500
frequency [Hz]


power density estimate of unmixed signal (regular) 2
density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0 500 1000 1500 2000 2500 3000 3500 4000 4500
frequency [Hz]


power density estimate of unmixed signal (natural) 1
density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0 500 1000 1500 2000 2500 3000 3500 4000 4500
frequency [Hz]


power density estimate of unmixed signal (natural) 2
density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0 500 1000 1500 2000 2500 3000 3500 4000 4500
frequency [Hz]