HW2_Problem_2

March 11, 2019

0.1 MCB 131 HW2 Problem 2 - Independent Component Analysis for Natural Image Reconstruction

0.1.1 Michael Cubeddu

0.1.2 2.1

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

        # read in mixed images
        with open('im1.csv', 'r') as im1:
            img1 = []
            csv_reader = csv.reader(im1, delimiter=',')
            for row in csv_reader:
                img1.append(list(map(lambda rgb: float(rgb), row)))

        with open('im2.csv', 'r') as im2:
            img2 = []
            csv_reader = csv.reader(im2, delimiter=',')
            for row in csv_reader:
                img2.append(list(map(lambda rgb: float(rgb), row)))

        with open('im3.csv', 'r') as im3:
            img3 = []
            csv_reader = csv.reader(im3, delimiter=',')
            for row in csv_reader:
                img3.append(list(map(lambda rgb: float(rgb), row)))

        img1 = np.array(img1)
        img2 = np.array(img2)
        img3 = np.array(img3)
        std_shape = img1.shape
        X = np.vstack([img1.flatten(), img2.flatten(), img3.flatten()])
        print(f'[MEAN] = {X.mean(1)}\n')
        print(f'[Variance]:\n\n{np.var(X, axis=1)}\n\n')
        print(f'[Skewness]:\n\n{stats.skew(X, axis=1)}\n\n')
        print(f'[Kurtosis]:\n\n{stats.kurtosis(X, axis=1)}\n\n')
```

```
[MEAN] = [273.94979095 147.22567749  44.36547852]

[Variance]:

[ 6255.12613353  3490.42898147 11827.2081117 ]


[Skewness]:

[0.50047648 0.76545431 0.05795406]


[Kurtosis]:

[-0.01609733  1.57688788  1.11255767]
```

0.1.3 2.2

```
In [2]: # center the data by subtracting mean vector
        centered = X.T - np.mean(X, axis=1)
        # get Cov matrix
        C = np.cov(centered.T)
        # get eigenvalues/vectors
        eig_vals, eig_vecs = np.linalg.eig(C)
        # decorrelate the images
        decorrelated = X.T.dot(eig_vecs)
        # avoid dividing by 0 by adding small epsilon to eigenvalues of 0
        whitened = decorrelated / np.sqrt(eig_vals + 1e-10)
        whitened.T.shape
```

```
Out[2]: (3, 131072)
```

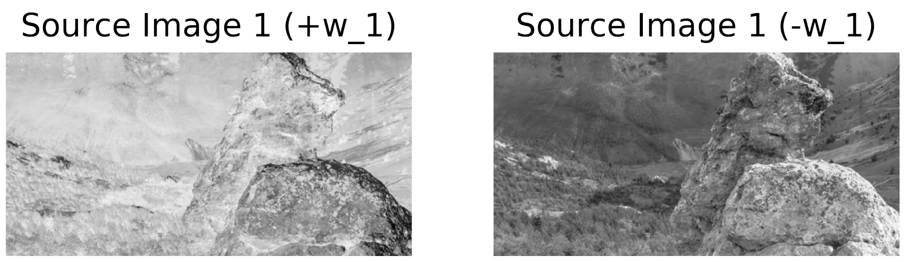
0.1.4 2.3

```
In [20]: # helper function calculating gradient of kurtosis
         def kurt_grad(w1, x_tilde):
             wT_x = w1.dot(x_tilde)
             p = len(x_tilde.T)
             A = np.sum(x_tilde.T.dot(w1))
             A_2 = np.sum(x_tilde.T.dot(w1)**2)
             A_3 = np.sum(x_tilde.T.dot(w1)**3)
             A_4 = np.sum(x_tilde.T.dot(w1)**4)
             D = 4*p*((A_3/(A_2**2)) - (A*A_4/(A_2**3)))
```

```
        x_mu = np.mean(x_tilde, axis=0)
        d_w1 = D*x_mu[0]/p
        d_w2 = D*x_mu[1]/p
        d_w3 = D*x_mu[2]/p
        res = np.array([d_w1, d_w2, d_w3])
        return res

    l_rate = 1e-7
    Wt = []
    # FIND w_1
    # w_next = np.random.normal(0,0.2,3)
    # I get better performance with fixed initialization
    w_next = np.array([-1e-6,1e-6,-1e-6])
    w_curr = np.array([.9,.9,.9])
    max_iter = 1e3
    ctr = 0
    while np.linalg.norm(w_next - w_curr) > 1e-5:
        w_curr = w_next
        ctr += 1
        if ctr >= max_iter:
            print('max iter reached')
            break
        w_next = w_curr + l_rate*kurt_grad(w_curr, whitened.T)
        w_next = w_next/np.linalg.norm(w_next)
    w_1 = w_next
    Wt.append(w_1)
    print(f'W_1 = {w_1}\n')
    plt.figure(dpi=500)
    plt.subplot(1,2,1)
    plt.title('Source Image 1 (+w_1)')
    plt.imshow(whitened.dot(w_1).reshape(std_shape), cmap='Greys')
    plt.axis('off')
    plt.subplot(1,2,2)
    plt.title('Source Image 1 (-w_1)')
    plt.axis('off')
    plt.imshow(whitened.dot(-w_1).reshape(std_shape), cmap='Greys')
    plt.savefig('source1.png', dpi=500)
    plt.show()
```

```
W_1 = [-0.68925907 -0.20392889 -0.69522295]
```



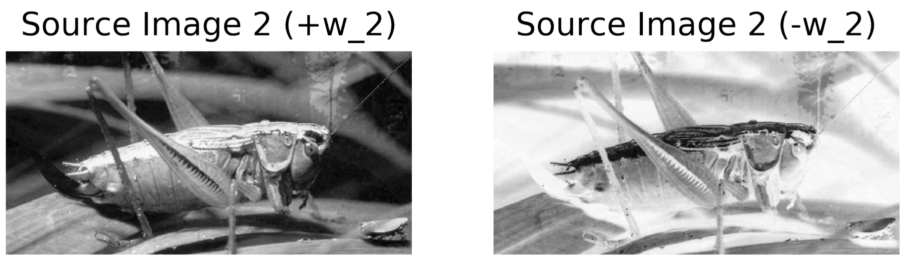
The reconstructed source image 1 from $\vec{w}_1$ and $-\vec{w}_1$ can be found here.

0.1.5 2.4

We need w_2 and w_1 to be orthogonal. Therefore, for each iteration update for w_2, we need to subtract $proj_{\vec{w}_1}\vec{w}_2$ from $w_2^{t-1} + \eta\Delta_{w_2}^{t-1}\text{kurtosis}(w_2^{t-1T}\tilde{x})$

```
In [22]: # FIND w_2
         # w_next = np.random.normal(0,0.2,3)
         w_next = np.array([1e-6,1e-6,-1e-6])
         w_curr = np.array([.9,.9,.9])
         max_iter = 1e3
         ctr = 0
         while np.linalg.norm(w_next - w_curr) > 1e-5:
             w_curr = w_next
             ctr += 1
             if ctr >= max_iter:
                 print('max iter reached')
                 break
             w_next = w_curr + l_rate*kurt_grad(w_curr, whitened.T)
             w1_proj = w_next.dot(w_1)
             w_next = (w_next - w_1*w1_proj)/np.linalg.norm(w_next)
         w_2 = w_next
         Wt.append(w_2)
         print(f'W_2 = {w_2}\n')
         plt.figure(dpi=500)
         plt.subplot(1,2,1)
         plt.title('Source Image 2 (+w_2)')
         plt.imshow(whitened.dot(w_2).reshape(std_shape), cmap='Greys')
         plt.axis('off')
         plt.subplot(1,2,2)
         plt.title('Source Image 2 (-w_2)')
         plt.axis('off')
         plt.imshow(whitened.dot(-w_2).reshape(std_shape), cmap='Greys')
         plt.savefig('source2.png', dpi=500)
         plt.show()
```

```
W_2 = [ 0.56561277  0.44821199 -0.69223421]
```



The second reconstructed source image from $\vec{w}_2$ and $-\vec{w}_2$ can be found here.

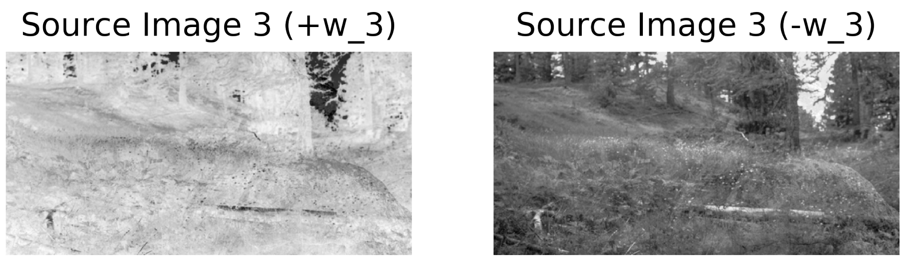
0.1.6 2.5

Again, we need all the columns of W to be orthogonal. Therefore, as we have already found w_1 and w_2, to get w_3, each iteration update we need to subtract the projection of w_3 onto the plane spanned by $w_1 \otimes w_2$. Thus, we calculate $proj_{\vec{w}_1}\vec{w}_3$ and $proj_{\vec{w}_2}\vec{w}_3$ and subtract the quantity $(proj_{\vec{w}_1}\vec{w}_3 + proj_{\vec{w}_1}\vec{w}_3)$ from our update equation. This will guarantee that w_3 is orthogonal to the plane $\vec{w}_1 \otimes \vec{w}_2$.

```
In [23]: # FIND w_3
         # w_next = np.random.normal(0,1e-4,3)
         w_next = np.array([-1e-6,1e-6,-1e-6])
         w_curr = np.array([.9,.9,.9])
         max_iter = 1e3
         ctr = 0
         while np.linalg.norm(w_next - w_curr) > 1e-5:
             w_curr = w_next
             ctr += 1
             if ctr >= max_iter:
                 print('max iter reached')
                 break
             w_next = w_curr + l_rate*kurt_grad(w_curr, whitened.T)
             w1_proj = w_next.dot(w_1)
             w2_proj = w_next.dot(w_2)
             w_next = (w_next - w_1*w1_proj - w_2*w2_proj)/np.linalg.norm(w_next)
         w_3 = w_next
         Wt.append(w_3)
         print(f'W_3 = {w_3}\n')
         plt.figure(dpi=500)
         plt.subplot(1,2,1)
         plt.title('Source Image 3 (+w_3)')
```

```
        plt.imshow(whitened.dot(w_3).reshape(std_shape), cmap='Greys')
        plt.axis('off')
        plt.subplot(1,2,2)
        plt.title('Source Image 3 (-w_3)')
        plt.axis('off')
        plt.imshow(whitened.dot(-w_3).reshape(std_shape), cmap='Greys')
        plt.savefig('source3.png', dpi=500)
        plt.show()
```

```
W_3 = [ 3.92569912e-13 -7.54994246e-13 -1.67957024e-13]
```



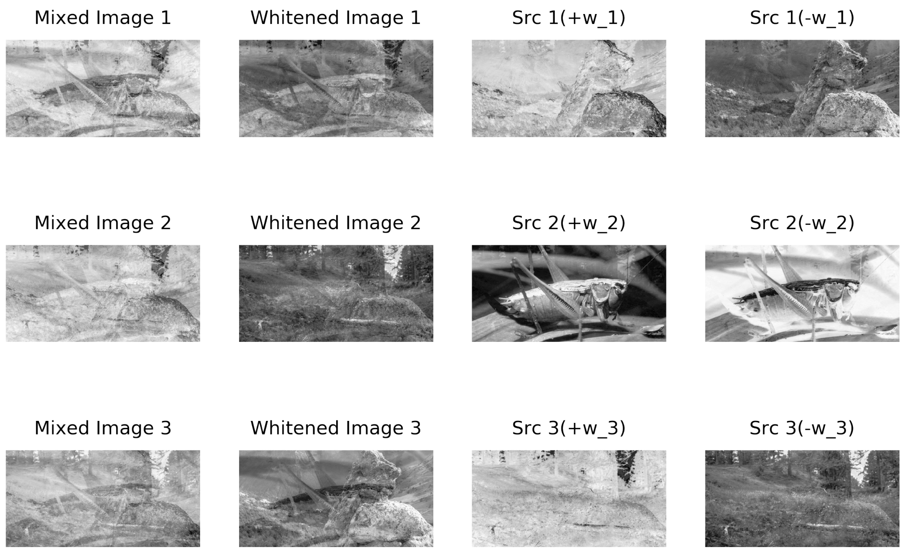
The third reconstructed source image from $\vec{w}_3$ and $-\vec{w}_3$ can be found here

0.1.7 2.6

```
In [24]: print('W = ')
         for col in Wt: print(col)
         plt.figure(dpi=500)
         plt.subplot(3,4,1)
         plt.imshow(img1, cmap='Greys')
         plt.title('Mixed Image 1', fontsize=7)
         plt.axis('off')
         plt.subplot(3,4,5)
         plt.imshow(img2, cmap='Greys')
         plt.title('Mixed Image 2', fontsize=7)
         plt.axis('off')
         plt.subplot(3,4,9)
         plt.imshow(img3, cmap='Greys')
         plt.title('Mixed Image 3', fontsize=7)
         plt.axis('off')
         plt.subplot(3,4,2)
         plt.imshow(whitened.T[0].reshape(std_shape), cmap='Greys')
         plt.title('Whitened Image 1', fontsize=7)
         plt.axis('off')
```

```
        plt.subplot(3,4,6)
        plt.imshow(whitened.T[1].reshape(std_shape), cmap='Greys')
        plt.title('Whitened Image 2', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,10)
        plt.imshow(whitened.T[2].reshape(std_shape), cmap='Greys')
        plt.title('Whitened Image 3', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,3)
        plt.imshow(whitened.dot(w_1).reshape(std_shape), cmap='Greys')
        plt.title('Src 1(+w_1)', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,7)
        plt.imshow(whitened.dot(w_2).reshape(std_shape), cmap='Greys')
        plt.title('Src 2(+w_2)', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,11)
        plt.imshow(whitened.dot(w_3).reshape(std_shape), cmap='Greys')
        plt.title('Src 3(+w_3)', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4, 4)
        plt.imshow(whitened.dot(-w_1).reshape(std_shape), cmap='Greys')
        plt.title('Src 1(-w_1)', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,8)
        plt.imshow(whitened.dot(-w_2).reshape(std_shape), cmap='Greys')
        plt.title('Src 2(-w_2)', fontsize=7)
        plt.axis('off')
        plt.subplot(3,4,12)
        plt.imshow(whitened.dot(-w_3).reshape(std_shape), cmap='Greys')
        plt.title('Src 3(-w_3)', fontsize=7)
        plt.axis('off')
        plt.savefig('images.png', dpi=500)
```

```
W =
[-0.68925907 -0.20392889 -0.69522295]
[ 0.56561277  0.44821199 -0.69223421]
[ 0.56561277  0.44821199 -0.69223421]
[ 3.92569912e-13 -7.54994246e-13 -1.67957024e-13]
```



The resulting Images figure can be found here. Although there is still some slight mixing, ICA has successfully recovered the three source images. Throughout my exploration of this problem, I found I was getting different source images based on the initial vectors w_i^0 for $i \in [1, 2, 3]$. Therefore, I decided to fix the initial vectors that gave me a visually acceptable source image reconstruction. Also, we can see that using negative vectors of W will give us the opposite intensity of the image (i.e. negative image). Thus, for these values of $\vec{w}_1, \vec{w}_2, \vec{w}_3$, the correct source image combination uses either $W = [\vec{w}_1, -\vec{w}_2, \vec{w}_3]$ or $W = [-\vec{w}_1, \vec{w}_2, -\vec{w}_3]$, with the latter being more visually satisfying.