CS6301 MACHINE LEARNING LAB WEEK – 12 GMM and ICA

SRIHARI. S – 2018103601

Date: 03-05-2021 Monday

Aim: To implement Independent Component Analysis and Gaussian Mixture Model.

Gaussian mixture model (GMM) :

- GMM is also a type of clustering algorithm.
- Each cluster is modelled according to a different Gaussian distribution.
- This flexible and probabilistic approach to modelling the data means that rather than having hard assignments into clusters like k-means, we have soft assignments.

Algorithm:

• Write N datapoints xi = (x1i, x2i, . . . , xMi) as row vectors

• Put these vectors into a matrix X (which will have size N ×M)

• Centre the data by subtracting off the mean of each column, putting it into matrix B

• Compute the covariance matrix C = 1/N B^T B

• Compute the eigenvalues and eigenvectors of C, so V^{-1} CV = D, where V holds the eigenvectors of C and D is the M ×M diagonal eigenvalue matrix.

• Sort the columns of D into order of decreasing eigenvalues, and apply the same order to the columns of V

• Reject those with eigenvalue less than some n (eta), leaving L dimensions in the data

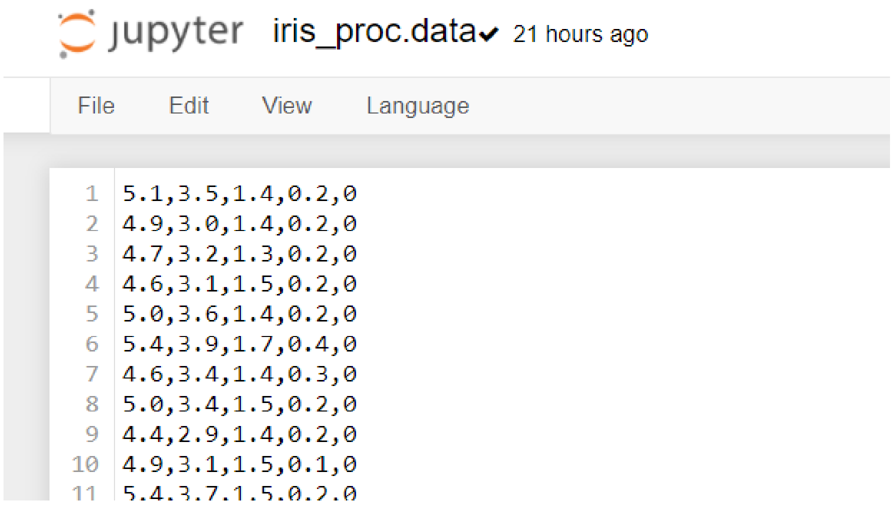
Dataset : Iris

Url: https://archive.ics.uci.edu/ml/datasets/iris

Description: The **Iris Dataset** contains four features (length and width of sepals and petals) of 50 samples of three species of **Iris** (**Iris** setosa, **Iris** virginica and **Iris** versicolor).

Input: The following 4 attributes

- sepal length in cm,
- sepal width in cm,
- petal length in cm,
- petal width in cm,



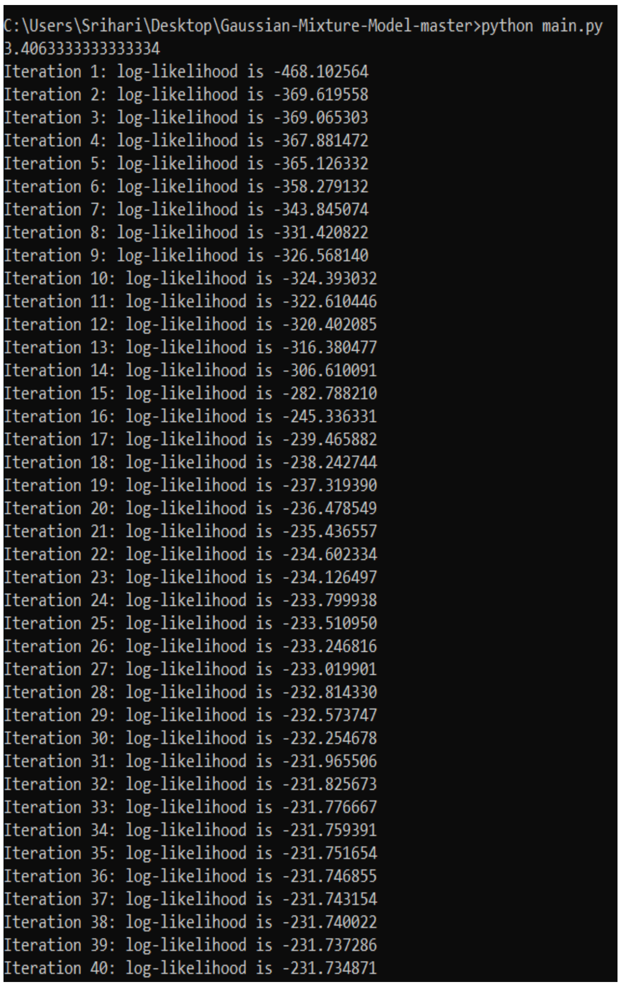
```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
4.6,3.1,1.5,0.2,0
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,0
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,0
4.9,3.1,1.5,0.1,0
5.4,3.7,1.5,0.2,0
```

Code:

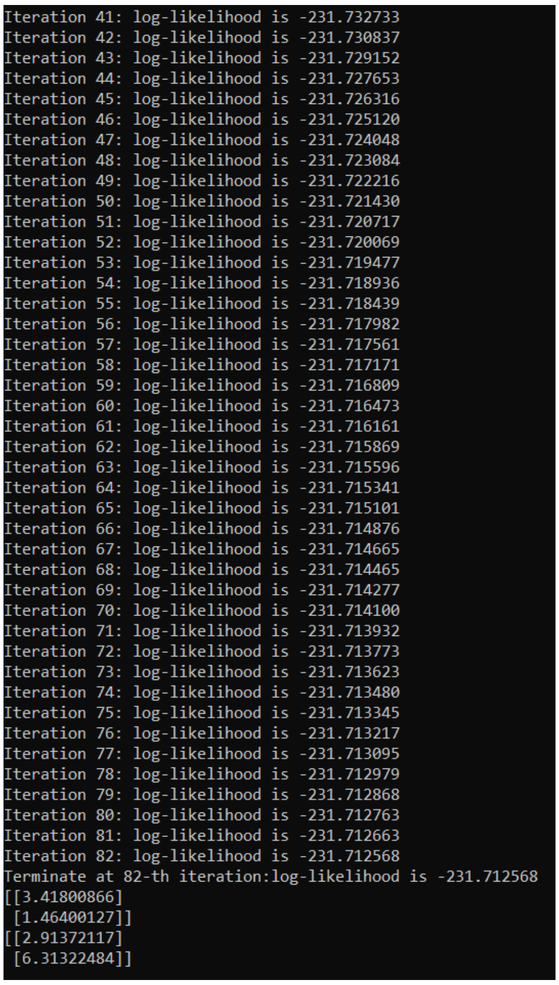
```
import pandas as pd
from GMM import *
import util as plot
from matplotlib import pyplot as plt

def For_Iris(features,No_Component=2):
   data = pd.read_csv("Data/Iris.csv", header = 0)
   data = data.reset_index()
   replace_map = {'Species': {'Iris-virginica': 1, 'Iris-versicolor': 2,'Iris-setosa':3}}
   data.replace(replace_map, inplace=True)
   label=data[['Species']]
   col=['SepalLengthCm','SepalWidthCm']
   x=data[col]
   x=np.array(x)
   gmm = GaussianMixModel(x,No_Component)
   gmm.fit()
   plot.plot_2D(gmm,x,col,label)

def main():
  For_Iris(2,3)
if __name__== "__main__":
  main()
```

```
import numpy as np
import scipy.stats as sp
class GaussianMixModel(object):
    def __init__(self, X, k=2):
        X = np.asarray(X)
        self.m, self.n = X.shape
        self.data = X.copy()
        print (np.mean(X))
        self.k = k

    def _init(self):
        self.mean_arr = np.asmatrix(np.random.random((self.k,
self.n))+np.mean(self.data))
        self.sigma_arr = np.array([np.asmatrix(np.identity(self.n)) for i in
range(self.k)])
        self.phi = np.ones(self.k)/self.k
        self.Z = np.asmatrix(np.empty((self.m, self.k), dtype=float))

    def fit(self, tol=1e-4):
        self._init()
        num_iters = 0
        logl = 1
        previous_logl = 0
        while(logl-previous_logl > tol):
            previous_logl = self.loglikelihood()
            self.e_step()
            self.m_step()
            num_iters += 1
            logl = self.loglikelihood()
            print('Iteration %d: log-likelihood is %.6f'%(num_iters, logl))
        print('Terminate at %d-th iteration:log-likelihood is %.6f'%(num_iters, logl))

    def loglikelihood(self):
        logl = 0
        for i in range(self.m):
            tmp = 0
            for j in range(self.k):
                tmp += sp.multivariate_normal.pdf(self.data[i, :],self.mean_arr[j,
:].A1,self.sigma_arr[j, :]) * self.phi[j]
```

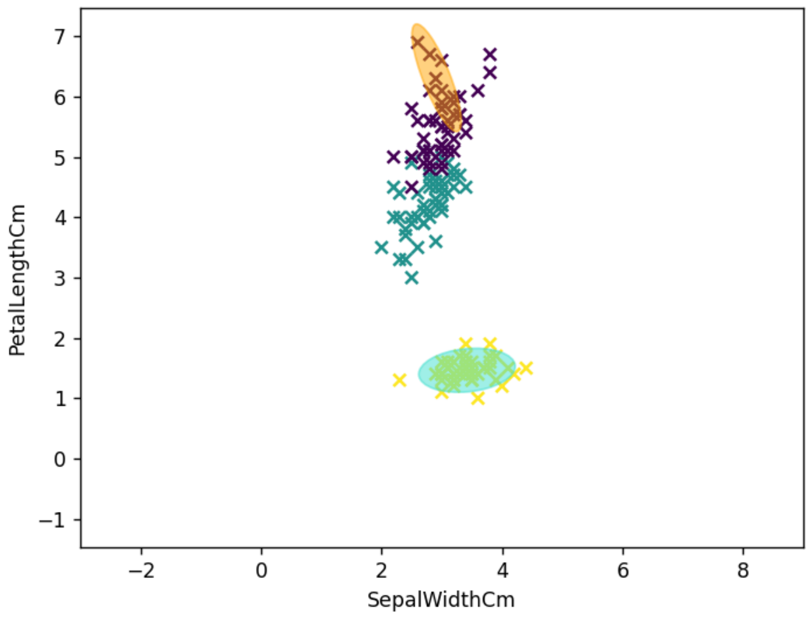
```
            logl += np.log(tmp)
        return logl

    def e_step(self):
        for i in range(self.m):
            den = 0
            for j in range(self.k):
                num = sp.multivariate_normal.pdf(self.data[i, :],
                                                 self.mean_arr[j].A1,
                                                 self.sigma_arr[j]) *\
                      self.phi[j]
                den += num
                self.Z[i, j] = num
            self.Z[i, :] /= den
            assert self.Z[i, :].sum() - 1 < 1e-4  # Program stop if this condition is false

    def m_step(self):
        for j in range(self.k):
            const = self.Z[:, j].sum()
            self.phi[j] = 1/self.m * const
            _mu_j = np.zeros(self.n)
            _sigma_j = np.zeros((self.n, self.n))
            for i in range(self.m):
                _mu_j += (self.data[i, :] * self.Z[i, j])
                _sigma_j += self.Z[i, j] * ((self.data[i, :] - self.mean_arr[j, :]).T *
(self.data[i, :] - self.mean_arr[j, :]))
            self.mean_arr[j] = _mu_j / const
            self.sigma_arr[j] = _sigma_j / const

import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
import matplotlib as mpl
import pandas as pd

def make_ellipses(gmm, ax):
    colors = ['turquoise', 'orange']
    for n, color in enumerate(colors):
        covariances = gmm.sigma_arr[n]
        v, w = np.linalg.eigh(covariances)
        u = w[0] / np.linalg.norm(w[0])
        angle = np.arctan2(u[1], u[0])
        angle = 180 * angle / np.pi  # convert to degrees
        v = 3. * np.sqrt(2.) * np.sqrt(v)
        mean=gmm.mean_arr[n]
        mean=mean.reshape(2,1)
        print(mean)
        ell = mpl.patches.Ellipse(mean, v[0], v[1],
                                  180 + angle, color=color)
        ell.set_clip_box(ax.bbox)
        ell.set_alpha(0.5)
        ax.add_artist(ell)
        ax.set_aspect('equal', 'datalim')
def plot_2D(gmm,x,col,label):
    h = plt.subplot(111, aspect='equal')
    make_ellipses(gmm, h)
    plt.scatter(x[:,0],x[:,1],c=label['Species'],marker='x')
    plt.xlim(-3, 9)
    plt.ylim(-3, 9)
    plt.xlabel(col[0])
    plt.ylabel(col[1])
    plt.show()
```



```
C:\Users\Srihari\Desktop\Gaussian-Mixture-Model-master>python main.py
3.4063333333333334
Iteration 1: log-likelihood is -468.102564
Iteration 2: log-likelihood is -369.619558
Iteration 3: log-likelihood is -369.065303
Iteration 4: log-likelihood is -367.881472
Iteration 5: log-likelihood is -365.126332
Iteration 6: log-likelihood is -358.279132
Iteration 7: log-likelihood is -343.845074
Iteration 8: log-likelihood is -331.420822
Iteration 9: log-likelihood is -326.568140
Iteration 10: log-likelihood is -324.393032
Iteration 11: log-likelihood is -322.610446
Iteration 12: log-likelihood is -320.402085
Iteration 13: log-likelihood is -316.380477
Iteration 14: log-likelihood is -306.610091
Iteration 15: log-likelihood is -282.788210
Iteration 16: log-likelihood is -245.336331
Iteration 17: log-likelihood is -239.465882
Iteration 18: log-likelihood is -238.242744
Iteration 19: log-likelihood is -237.319390
Iteration 20: log-likelihood is -236.478549
Iteration 21: log-likelihood is -235.436557
Iteration 22: log-likelihood is -234.602334
Iteration 23: log-likelihood is -234.126497
Iteration 24: log-likelihood is -233.799938
Iteration 25: log-likelihood is -233.510950
Iteration 26: log-likelihood is -233.246816
Iteration 27: log-likelihood is -233.019901
Iteration 28: log-likelihood is -232.814330
Iteration 29: log-likelihood is -232.573747
Iteration 30: log-likelihood is -232.254678
Iteration 31: log-likelihood is -231.965506
Iteration 32: log-likelihood is -231.825673
Iteration 33: log-likelihood is -231.776667
Iteration 34: log-likelihood is -231.759391
Iteration 35: log-likelihood is -231.751654
Iteration 36: log-likelihood is -231.746855
Iteration 37: log-likelihood is -231.743154
Iteration 38: log-likelihood is -231.740022
Iteration 39: log-likelihood is -231.737286
Iteration 40: log-likelihood is -231.734871
```



```
Iteration 41: log-likelihood is -231.732733
Iteration 42: log-likelihood is -231.730837
Iteration 43: log-likelihood is -231.729152
Iteration 44: log-likelihood is -231.727653
Iteration 45: log-likelihood is -231.726316
Iteration 46: log-likelihood is -231.725120
Iteration 47: log-likelihood is -231.724048
Iteration 48: log-likelihood is -231.723084
Iteration 49: log-likelihood is -231.722216
Iteration 50: log-likelihood is -231.721430
Iteration 51: log-likelihood is -231.720717
Iteration 52: log-likelihood is -231.720069
Iteration 53: log-likelihood is -231.719477
Iteration 54: log-likelihood is -231.718936
Iteration 55: log-likelihood is -231.718439
Iteration 56: log-likelihood is -231.717982
Iteration 57: log-likelihood is -231.717561
Iteration 58: log-likelihood is -231.717171
Iteration 59: log-likelihood is -231.716809
Iteration 60: log-likelihood is -231.716473
Iteration 61: log-likelihood is -231.716161
Iteration 62: log-likelihood is -231.715869
Iteration 63: log-likelihood is -231.715596
Iteration 64: log-likelihood is -231.715341
Iteration 65: log-likelihood is -231.715101
Iteration 66: log-likelihood is -231.714876
Iteration 67: log-likelihood is -231.714665
Iteration 68: log-likelihood is -231.714465
Iteration 69: log-likelihood is -231.714277
Iteration 70: log-likelihood is -231.714100
Iteration 71: log-likelihood is -231.713932
Iteration 72: log-likelihood is -231.713773
Iteration 73: log-likelihood is -231.713623
Iteration 74: log-likelihood is -231.713480
Iteration 75: log-likelihood is -231.713345
Iteration 76: log-likelihood is -231.713217
Iteration 77: log-likelihood is -231.713095
Iteration 78: log-likelihood is -231.712979
Iteration 79: log-likelihood is -231.712868
Iteration 80: log-likelihood is -231.712763
Iteration 81: log-likelihood is -231.712663
Iteration 82: log-likelihood is -231.712568
Terminate at 82-th iteration:log-likelihood is -231.712568
[[3.41800866]
 [1.46400127]]
[[2.91372117]
 [6.31322484]]
```



Independent Component Analysis:

- Independent Component Analysis is a signal processing method to separate independent sources linearly mixed in several sensors.
- ICA is a technique to separate linearly mixed sources.

Algorithm:

1. Center x by subtracting the mean
2. Whiten x

   To *whiten* a given signal means that we transform it in such a way that potential correlations between its components are removed (covariance equal to 0) and the variance of each component is equal to 1.

   The whitening process is simply a linear change of coordinate of the mixed data. Once the ICA solution is found in this “whitened” coordinate frame, we can easily reproject the ICA solution back into the original coordinate frame.

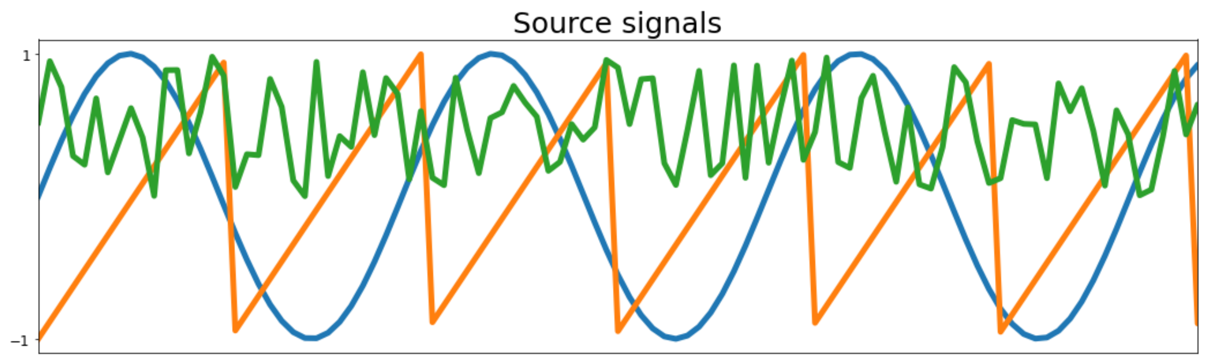
3. Choose a random initial value for the de-mixing matrix w
4. Calculate the new value for w
5. Normalize w
6. Check whether algorithm has converged and if it hasn’t, return to step 4
7. Take the dot product of w and x to get the independent source signals.

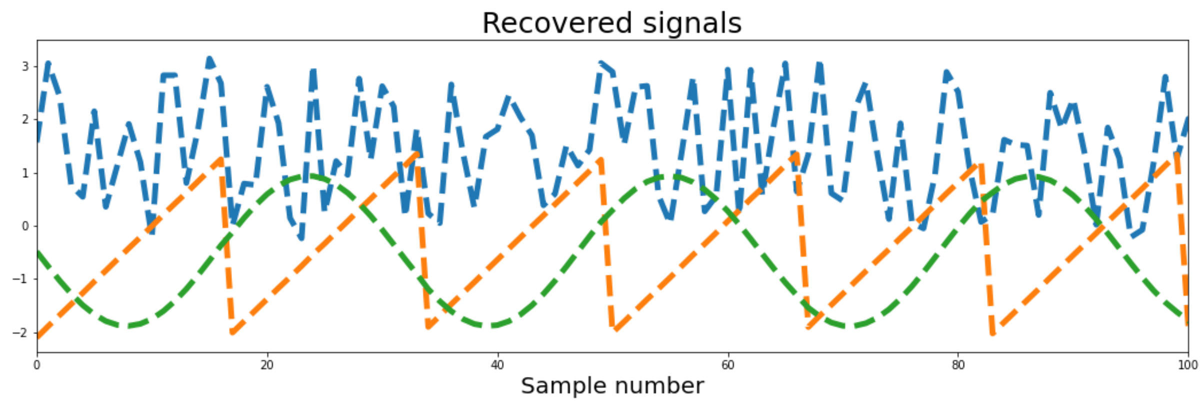
Code

```
import pandas as pd
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
```

```
%matplotlib inline
np.random.seed(23)
ns = np.linspace(0, 200, 1000)
S = np.array([np.sin(ns * 1), signal.sawtooth(ns * 1.9),
            np.random.random(len(ns))]).T
A = np.array([[0.5, 1, 0.2],
            [1, 0.5, 0.4],
            [0.5, 0.8, 1]])
X = S.dot(A).T
fig, ax = plt.subplots(1, 1, figsize=[18, 5])
ax.plot(ns, S, lw=5)
ax.set_xticks([])
ax.set_yticks([-1, 1])
ax.set_xlim(ns[0], ns[200])
ax.tick_params(labelsize=12)
ax.set_title('Independent sources', fontsize=25)
fig, ax = plt.subplots(3, 1, figsize=[18, 5], sharex=True)
ax[0].plot(ns, X[0], lw=5)
ax[0].set_title('Mixed signals', fontsize=25)
ax[0].tick_params(labelsize=12)
ax[1].plot(ns, X[1], lw=5)
ax[1].tick_params(labelsize=12)
ax[1].set_xlim(ns[0], ns[-1])
ax[2].plot(ns, X[2], lw=5)
ax[2].tick_params(labelsize=12)
ax[2].set_xlim(ns[0], ns[-1])
ax[2].set_xlabel('Sample number', fontsize=20)
ax[2].set_xlim(ns[0], ns[200])
plt.show()
def center(x):
    mean = np.mean(x, axis=1, keepdims=True)
    centered =  x - mean
    return centered, mean
```

```
def covariance(x):
    mean = np.mean(x, axis=1, keepdims=True)
    n = np.shape(x)[1] - 1
    m = x - mean
    return (m.dot(m.T))/n

def whiten(x):
    coVarM = covariance(X)
    U, S, V = np.linalg.svd(coVarM)
    d = np.diag(1.0 / np.sqrt(S))
    whiteM = np.dot(U, np.dot(d, U.T))
    Xw = np.dot(whiteM, X)
    return Xw, whiteM

def fastIca(signals,  alpha = 1, thresh=1e-8, iterations=5000):
    m, n = signals.shape
    W = np.random.rand(m, m)
    for c in range(m):
            w = W[c, :].copy().reshape(m, 1)
            w = w / np.sqrt((w ** 2).sum())
            i = 0
            lim = 100
            while ((lim > thresh) & (i < iterations)):
                ws = np.dot(w.T, signals)
                wg = np.tanh(ws * alpha).T
                wg_ = (1 - np.square(np.tanh(ws))) * alpha
                wNew = (signals * wg.T).mean(axis=1) - wg_.mean() *
w.squeeze()
                wNew = wNew - np.dot(np.dot(wNew, W[:c].T), W[:c])
                wNew = wNew / np.sqrt((wNew ** 2).sum())
                lim = np.abs(np.abs((wNew * w).sum()) - 1)
                w = wNew
                i += 1
            W[c, :] = w.T
    return W
Xc, meanX = center(X)
Xw, whiteM = whiten(Xc)
print(np.round(covariance(Xw)))
W = fastIca(Xw,  alpha=1)
unMixed = Xw.T.dot(W.T)
unMixed = (unMixed.T - meanX).T
fig, ax = plt.subplots(1, 1, figsize=[18, 5])
ax.plot(S, lw=5)
ax.tick_params(labelsize=12)
ax.set_xticks([])
ax.set_yticks([-1, 1])
ax.set_title('Source signals', fontsize=25)
ax.set_xlim(0, 100)
fig, ax = plt.subplots(1, 1, figsize=[18, 5])
ax.plot(unMixed, '--', label='Recovered signals', lw=5)
ax.set_xlabel('Sample number', fontsize=20)
ax.set_title('Recovered signals', fontsize=25)
ax.set_xlim(0, 100)
plt.show()
```

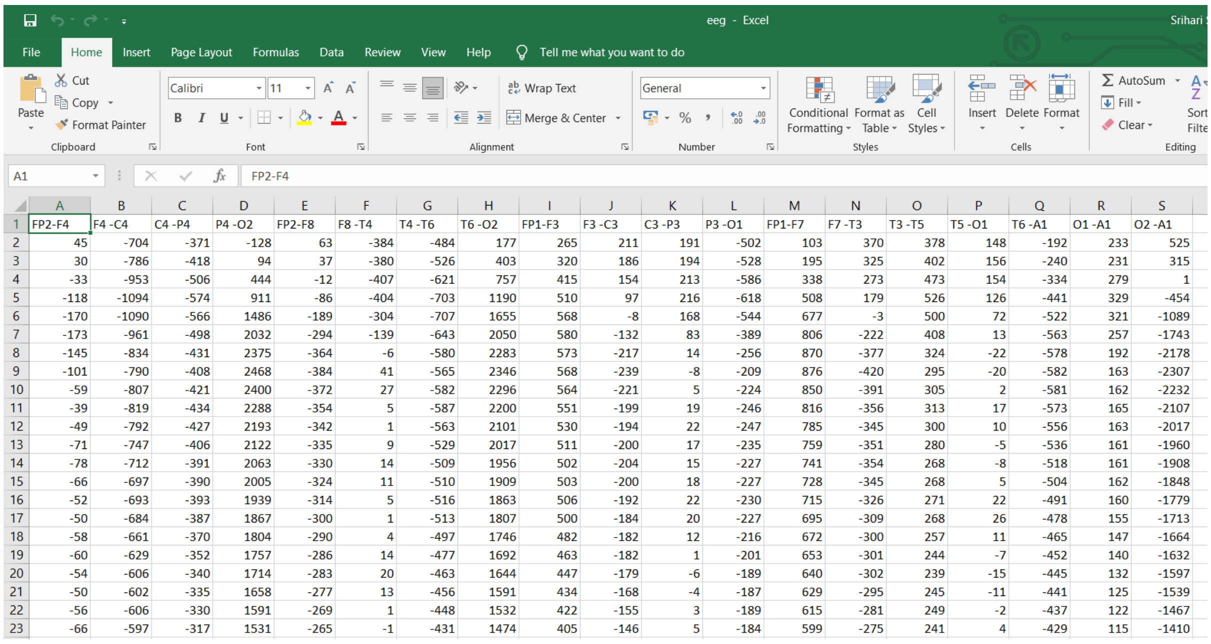




INDEPENDENT COMPONENT ANALYSIS ON EEG DATASET:

This data arises from a large study to examine EEG correlates of genetic predisposition to alcoholism. It contains measurements from 64 electrodes placed on subject's scalps which were sampled at 256 Hz (3.9-msec epoch) for 1 second. There were two groups of subjects: alcoholic and control. Each subject was exposed to either a single stimulus (S1) or to two stimuli (S1 and S2) which were pictures of objects chosen from the 1980 Snodgrass and Vanderwart picture set. When two stimuli were shown, they were presented in either a matched condition where S1 was identical to S2 or in a non-matched condition where S1 differed from S2.

Input:

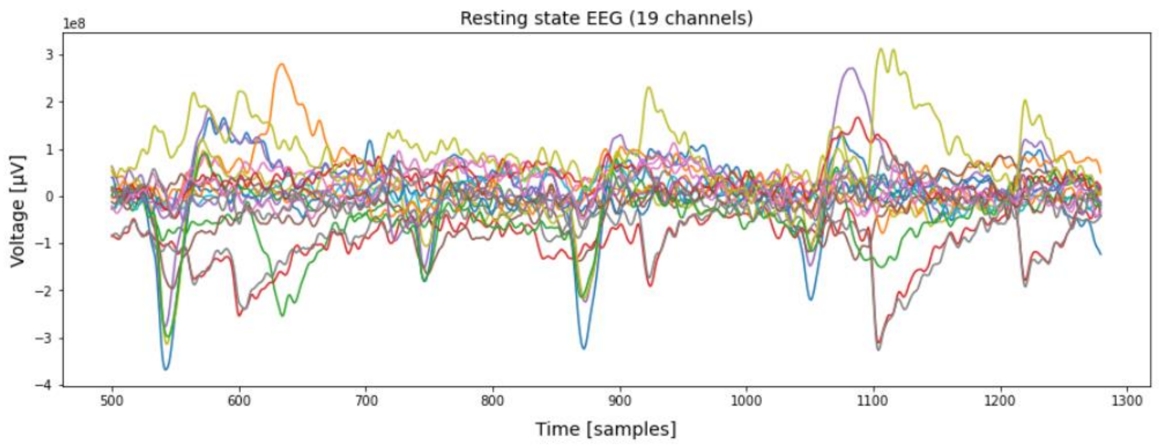


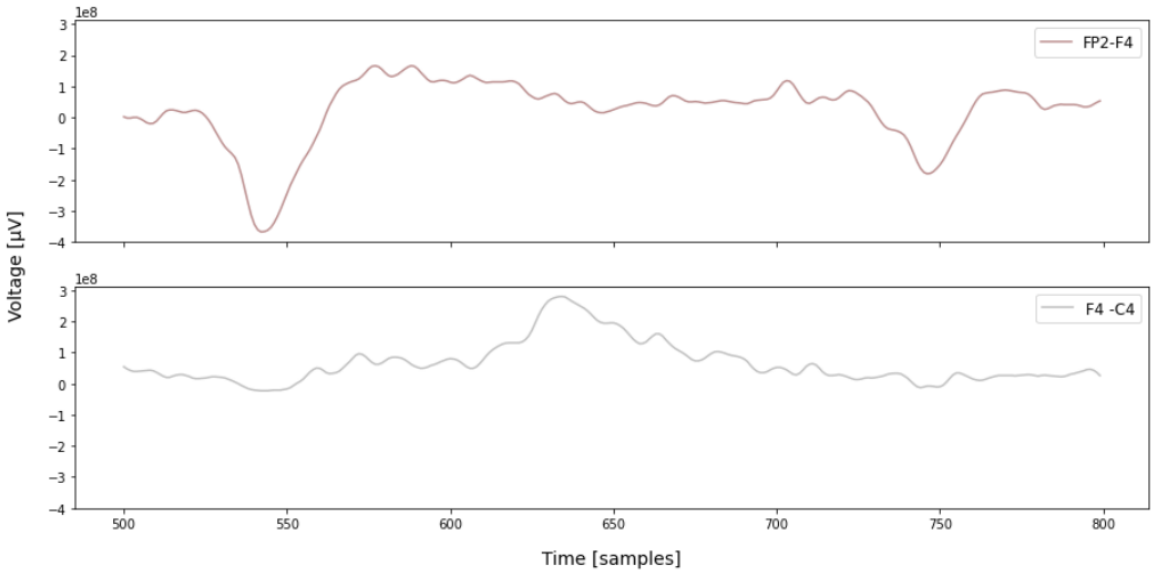
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	FP2-F4	F4 -C4	C4 -P4	P4 -O2	FP2-F8	F8 -T4	T4 -T6	T6 -O2	FP1-F3	F3 -C3	C3 -P3	P3 -O1	FP1-F7	F7 -T3	T3 -T5	T5 -O1	T6 -A1	O1 -A1	O2 -A1
2	45	-704	-371	-128	63	-384	-484	177	265	211	191	-502	103	370	378	148	-192	233	525
3	30	-786	-418	94	37	-380	-526	403	320	186	194	-528	195	325	402	156	-240	231	315
4	-33	-953	-506	444	-12	-407	-621	757	415	154	213	-586	338	273	473	154	-334	279	1
5	-118	-1094	-574	911	-86	-404	-703	1190	510	97	216	-618	508	179	526	126	-441	329	-454
6	-170	-1090	-566	1486	-189	-304	-707	1655	568	-8	168	-544	677	-3	500	72	-522	321	-1089
7	-173	-961	-498	2032	-294	-139	-643	2050	580	-132	83	-389	806	-222	408	13	-563	257	-1743
8	-145	-834	-431	2375	-364	-6	-580	2283	573	-217	14	-256	870	-377	324	-22	-578	192	-2178
9	-101	-790	-408	2468	-384	41	-565	2346	568	-239	-8	-209	876	-420	295	-20	-582	163	-2307
10	-59	-807	-421	2400	-372	27	-582	2296	564	-221	5	-224	850	-391	305	2	-581	162	-2232
11	-39	-819	-434	2288	-354	5	-587	2200	551	-199	19	-246	816	-356	313	17	-573	165	-2107
12	-49	-792	-427	2193	-342	1	-563	2101	530	-194	22	-247	785	-345	300	10	-556	163	-2017
13	-71	-747	-406	2122	-335	9	-529	2017	511	-200	17	-235	759	-351	280	-5	-536	161	-1960
14	-78	-712	-391	2063	-330	14	-509	1956	502	-204	15	-227	741	-354	268	-8	-518	161	-1908
15	-66	-697	-390	2005	-324	11	-510	1909	503	-200	18	-227	728	-345	268	5	-504	162	-1848
16	-52	-693	-393	1939	-314	5	-516	1863	506	-192	22	-230	715	-326	271	22	-491	160	-1779
17	-50	-684	-387	1867	-300	1	-513	1807	500	-184	20	-227	695	-309	268	26	-478	155	-1713
18	-58	-661	-370	1804	-290	4	-497	1746	482	-182	12	-216	672	-300	257	11	-465	147	-1664
19	-60	-629	-352	1757	-286	14	-477	1692	463	-182	1	-201	653	-301	244	-7	-452	140	-1632
20	-54	-606	-340	1714	-283	20	-463	1644	447	-179	-6	-189	640	-302	239	-15	-445	132	-1597
21	-50	-602	-335	1658	-277	13	-456	1591	434	-168	-4	-187	629	-295	245	-11	-441	125	-1539
22	-56	-606	-330	1591	-269	1	-448	1532	422	-155	3	-189	615	-281	249	-2	-437	122	-1467
23	-66	-597	-317	1531	-265	-1	-431	1474	405	-146	5	-184	599	-275	241	4	-429	115	-1410

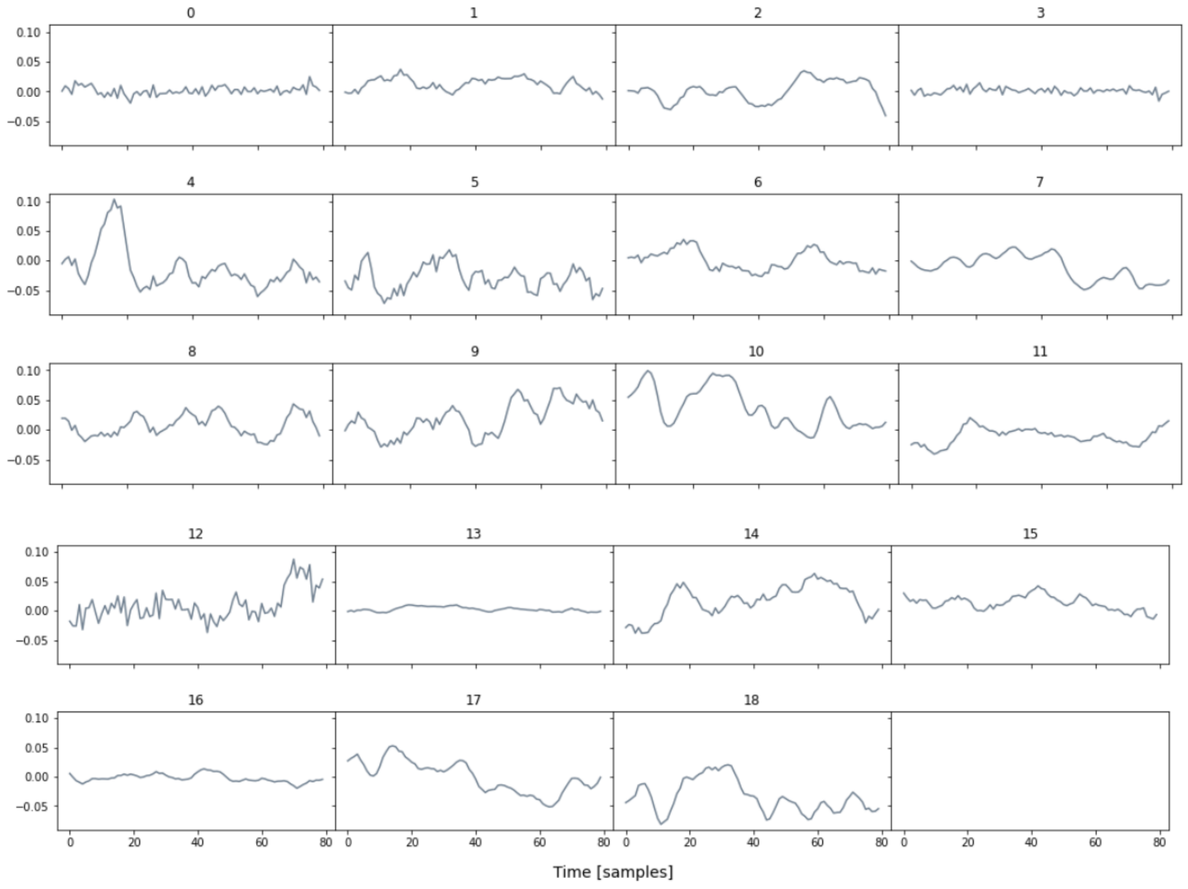
Code:

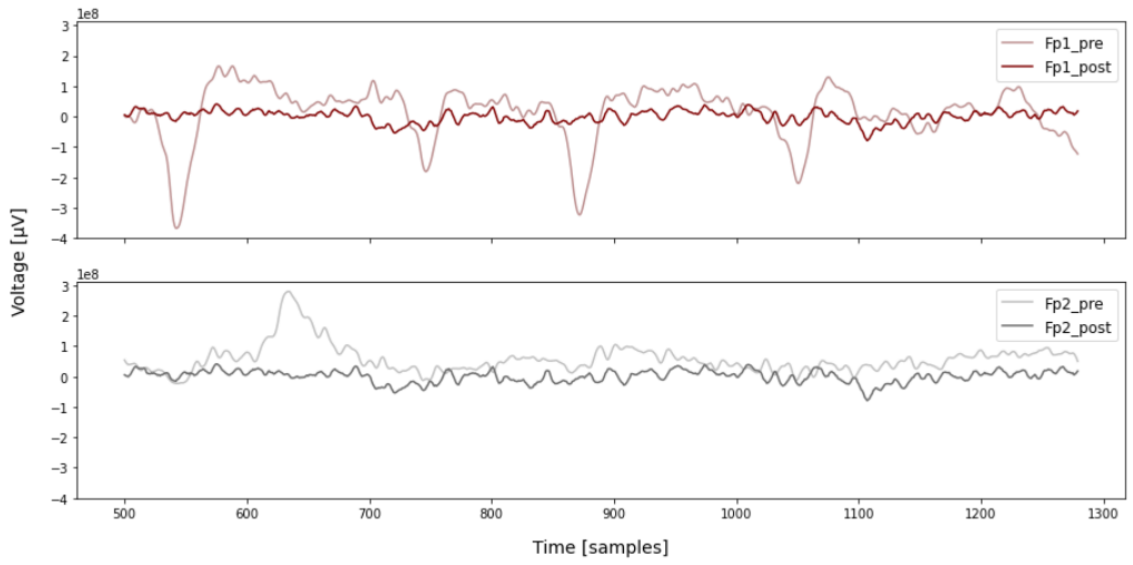
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import deque
from sklearn.decomposition import FastICA
eeg = pd.read_csv('eeg.csv')
eeg *= 10**6 # from V to uV
eeg.iloc[500:2500].plot(figsize=(15,5), legend=False)
plt.xlabel('Time [samples]', fontsize=14, labelpad=10)
plt.ylabel('Voltage [\u03BCV]', fontsize=14)
plt.title('Resting state EEG (19 channels)', fontsize=14)
plt.show()
fig, axs = plt.subplots(2,1, figsize=(15, 7), sharex=True, sharey=True)
axs = axs.ravel()
plt.margins(x=0.001)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', top=False, bottom=False, left=False,
right=False)
axs[0].plot(eeg.iloc[500:800,0], label='FP2-F4', color='rosybrown')
axs[0].legend(loc="upper right", fontsize=12)
axs[1].plot(eeg.iloc[500:800,1], label='F4 -C4', color='silver')
axs[1].legend(loc="upper right", fontsize=12)
plt.xlabel('Time [samples]', fontsize=14, labelpad=15)
plt.ylabel('Voltage [\u03BCV]', fontsize=14, labelpad=15)
plt.show()
ica = FastICA(n_components=19, random_state=0, tol=0.05)
comps = ica.fit_transform(eeg)
fig, axs = plt.subplots(5,4, figsize=(18, 13), sharex=True, sharey=True)
fig.subplots_adjust(hspace = .4, wspace=0)
axs = axs.ravel()

fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', top=False, bottom=False, left=False,
right=False)
plt.xlabel('Time [samples]', fontsize=14, labelpad=15)
```

```
for i in range(19):
   axs[i].plot(comps[1200:1600, i], color='slategrey')
   axs[i].set_title(str(i))

comps_restored = comps.copy()
comps_restored[:,[4,10]] = 0 # set artefact components to zero
restored = ica.inverse_transform(comps_restored)
fig, axs = plt.subplots(2,1, figsize=(15, 7), sharex=True, sharey=True)
axs = axs.ravel()
plt.margins(x=0.001)
fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none',     top=False,     bottom=False,     left=False,
right=False)
axs[0].plot(eeg.iloc[500:1300,0], label='Fp1_pre', color='rosybrown')
axs[0].plot(np.arange(500,1280),    restored[500:2500,    11],    label='Fp1_post',
color='maroon')
axs[0].legend(loc="upper right", fontsize=12)
axs[1].plot(eeg.iloc[500:1300,1], label='Fp2_pre', color='silver')
axs[1].plot(np.arange(500,1280),    restored[500:2500,    11],    label='Fp2_post',
color='dimgray')
axs[1].legend(loc="upper right", fontsize=12)
plt.xlabel('Time [samples]', fontsize=14, labelpad=15)
plt.ylabel('Voltage [\u03BCV]', fontsize=14, labelpad=15)
plt.show()
```




1e8
FP2-F4
F4 -C4
Voltage [µV]
Time [samples]


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Time [samples]



Inference:

Thus as we can see above the lighter lines indicate the signals captured before the distortions due to the blinking of the eye were removed. The darker version represents the final signals obtained after the eye blinking weren't considered. Thus the spikes in the signals are removed as the independent components 4 and 10 represents the contribution of the blinking of the eye.