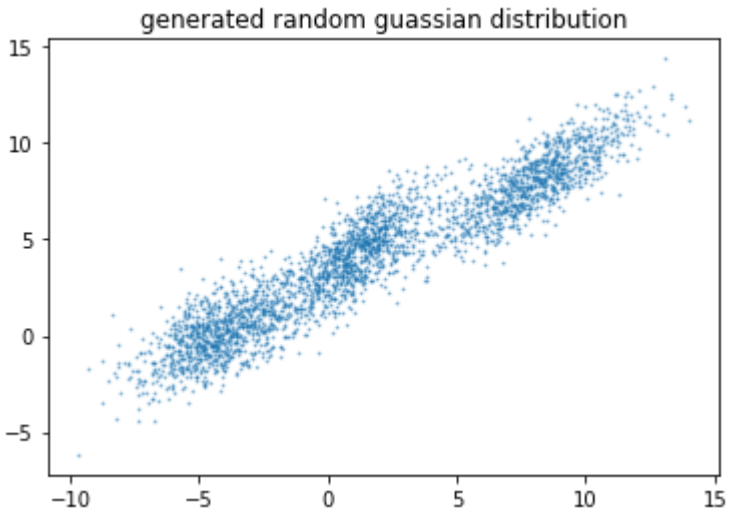
In [67]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from sklearn.cluster import KMeans as SklKMeans
```

In [68]:
```
def gen_data(k,d,ppc):
    np.random.seed(3)
    x = np.ndarray((k,ppc,d))
    mean = np.random.rand(k,d)*20-10
    for i in range(k):
        cov = np.random.rand(d,d+10)
        cov = cov@cov.T
        x[i] = np.random.multivariate_normal(mean[i],cov,ppc)

    x = x.reshape(-1,d)
    if(d==2):
        plt.figure()
        plt.title("generated random guassian distribution")
        plt.scatter(x[:,0],x[:,1],s=1,alpha=0.4)
        plt.show()
    return x
```

In [69]:
```
x = gen_data(3,2,1000)
```



In [70]:
```
m,n = x.shape
```

In [71]:
```
resp = np.zeros((m,3))
converged = False
log_like_trace = []
log_likelihood = 0
```

Initialize parameters

In [72]:
```
k = 3
```

```
weights = np.full(k,1/k)
np.random.seed(5)
means = x[np.random.choice(m, k , replace=False)]
c = np.cov(x,rowvar=False)
covs = np.full((k,n,n),c)

def init_mean(init_params,k,random_state=1):
    if init_params == "kmeans":
        global means,log_like_trace,log_likelihood,weights,covs
        log_like_trace = []
        log_likelihood = 0
        weights = np.full(k,1/k)
        c = np.cov(x,rowvar=False)
        covs = np.full((k,n,n),c)
        kmeans = SklKMeans(k,random_state=random_state)
        kmeans.fit(x)
        means = kmeans.cluster_centers_
    else:
        None
```

In [73]:

```
def do_estep():
    global resp, log_likelihood
    for i in range(k):
        resp[:,i] = weights[i]*multivariate_normal(means[i],covs[i]).pdf(x)
    px = np.sum(resp,axis=1,keepdims=1)
    resp = resp/px
    log_likelihood = np.sum(np.log(px))


def do_mstep():
    global means,covs,weights
    resp_weights = np.sum(resp,axis=0)
    weights = resp_weights/x.shape[0]
    means = resp.T @ x/resp_weights.reshape(-1,1)
    diff = x[:,np.newaxis,:] - means
    covs = np.einsum('ik,ikj,ikl->kjl',resp,diff,diff)/resp_weights.reshape(-1,1,1)
```

In [74]:

```
def draw(ax,title=""):
    ax.set_title(title)
    ax.scatter(x[:,0],x[:,1],s=1,alpha=0.2)
    delta = 0.05
    x1 = np.arange(*ax.get_xlim(), delta)
    y1 = np.arange(*ax.get_ylim(), delta)
    x1,y1 = np.meshgrid(x1,y1)
    col = ['C1', 'C2', 'C3']

    for i in range(k):
        mn, co = means[i],covs[i]
        z = multivariate_normal(mn, co).pdf(np.dstack([x1, y1]))
        ax.scatter(mn[0], mn[1], color=col[i])
        ax.contour(x1, y1, z,levels=[0.01],colors=col[i])
```

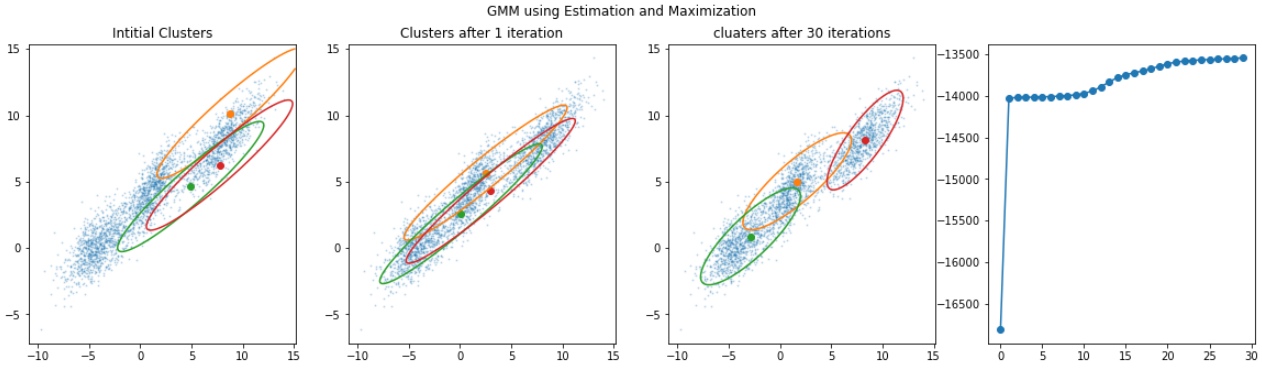
In [75]:

```
def gmm(n_iters,tol,init_params,random_state):
    init_mean(init_params,k,random_state=1)
```

```
    fig,ax = plt.subplots(1,4,figsize=(20,5))
    fig.suptitle("GMM using Estimation and Maximization")
    draw(ax[0],"Intitial Clusters")

    for i in range(n_iters):
        do_estep()
        do_mstep()
        log_like_trace.append(log_likelihood)

        if i == 0:
            draw(ax[1],"Clusters after 1 iteration")
        else:
            ol,nl = log_like_trace[-2:]
            if nl-ol <= tol:
                converged = True
                break

    draw(ax[2],f"cluaters after {i+1} iterations")
    ax[3].plot(log_like_trace,"-o")
    ax[3].set_title = "Log Likelihood"
    plt.show()
```

In [76]:

```
gmm(30,tol=10e-4,init_params="random",random_state=5)
```



In [77]:

```
gmm(30,tol=10e-4,init_params="kmeans",random_state=1)
```