Mean Shift Clustering

The mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. Given n data points xi, i = 1,...,n on a d-dimensional space Rd, the multivariate kernel density estimate obtained with kernel K(x) and window radius h is

$$f(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right).$$

For radially symmetric kernels, it suffices to define the profile of the kernel k(x) satisfying

$$K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2)$$

where ck,d is a normalization constant which assures K(x) integrates to 1. The modes of the density function are located at the zeros of the gradient function ∇f(x) = 0. The gradient of the density estimator (1) is

$$\begin{aligned}\nabla f(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\\ &= \frac{2c_{k,d}}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}-\mathbf{x}\right].\end{aligned}$$

where g(s) = −k0(s). The first term is proportional to the density estimate at x computed with kernel G(x) = cg,dg(kxk2) and the second term

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}-\mathbf{x}$$

is the mean shift. The mean shift vector always points toward the direction of the maximum increase in the density. The mean shift procedure, obtained by successive • computation of the mean shift vector mh(xt), • translation of the window xt+1 = xt +mh(xt) is guaranteed to converge to a point where the gradient of density function is zero. Mean shift mode finding process is illustrated in Figure 1.

The mean shift clustering algorithm is a practical application of the mode finding procedure:

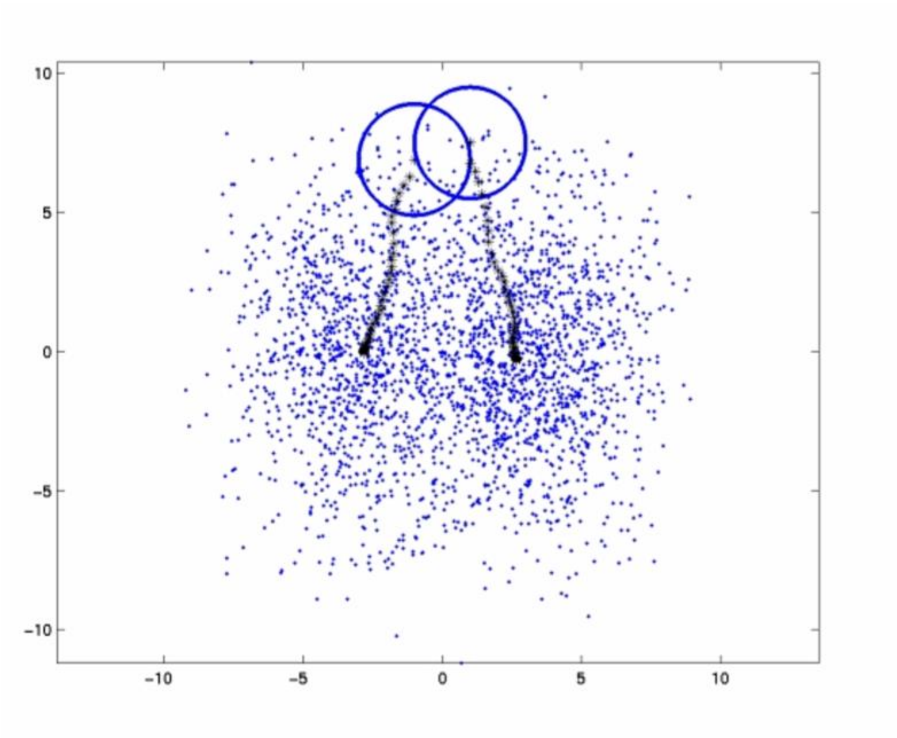


Figure 1: **Mean Shift Mode Finding**

• starting on the data points, run mean shift procedure to find the stationary points of the density function,

• prune these points by retaining only the local maxima.

The set of all locations that converge to the same mode defines the basin of attraction of that mode. The points which are in the same basin of attraction is associated with the same cluster. Figure 2 shows two examples of mean shift clustering on three dimensional data. More details on mean shift clustering on Lie Groups can be found in

Code

Weight computing :

```
def get_weight(x, m,a):
    new_wi=pow(np.subtract(x, m), 2)/(2*pow(a,2))
    # sum=[]
    # for item in new_wi:
    #     sum.append(np.exp(item))
    return np.exp(-pow(np.subtract(x, m), 2)/(2*pow(a,2)))
computing m :
```

```
def get_index(x, mu1):
    new_mu1 = mu1
    equal=False
    while not equal:
        num = np.zeros((1, x.shape[1]), float)
        den = np.zeros((1, x.shape[1]), float)
        for xi in x:
            wi = get_weight(xi, new_mu1,math.sqrt(x.shape[1] * (np.pi) * 2))
            num = num +(np.multiply(wi , xi))
            den = den+ wi
        i=new_mu1
        new_mu1 = np.divide(num ,den)
        subtract=np.exp(np.subtract(i,new_mu1))
        equal = (np.round(subtract,decimals=11)==1).all()

    return new_mu1
```

distance computing :

```
def mean_shift(x, m, a):
    new_x = np.array(pow(np.subtract(x, m), 2) / (2 * (a * a)))[0]
    new_x=np.exp(new_x)
    mean_shift_distance=np.dot(new_x, new_x.T)
    return math.sqrt(mean_shift_distance)
```

clustering:

```
def cluster_on_mean_shift(x, means: List[dict] = None, n=100):
    meanshift = []
    for mean in means:
        meanshift.append(mean_shift(x, list(mean.values())[0], math.sqrt(n * (np.pi)
* 2)))
        # mahal.append(mahalanobis(x, list(mean.values())[0]))
    min_dist = meanshift.index(np.min(meanshift))
    return list(means[min_dist].keys())[0]
```

resukt for x_train:

```
def classification(x, y):
    dist = []
    mean_shift_distance=[]
    new_X_train, new_X_test, new_y_train, new_y_test = train_test_split(x, y,
test_size=0.9, random_state=True)
    means, mean_shift = get_mean_for_each_class(new_X_train, new_y_train)
    print("indicator mean")
    len_x=len(new_X_test)
    for i in range(0, len_x):
        a = cluster_on_euclidean_distance(new_X_test[i], means=means)
        b=cluster_on_mean_shift(new_X_test[i],means=mean_shift,n=len_x)
        dist.append(a)
        mean_shift_distance.append(b)
    get_confusion_matrix(y_test=new_y_test, dist=dist)
    print("indicator find using meanshift distance")
    get_confusion_matrix(y_test=new_y_test, dist=mean_shift_distance)
```

indicator mean

Confusion Matrix :

```
[[158   0   0   0   1   1   0   0   0   0]
 [  0 108  10   0   0   2   3   0  22  13]
 [  1   9 133   3   0   0   0   1   5   2]
 [  0   0   0 140   0   1   0   6   8   6]
 [  1  10   0   0 151   0   0   3   3   0]
 [  0   0   0   4   1 127   2   0   0  31]
 [  0   4   0   0   0   0 157   0   1   0]
 [  0   1   3   0   3   1   0 155   8   0]
 [  0  18   1   2   0   1   1   0 125   7]
 [  0   2   0   7   0   4   0   7   4 140]]
Accuracy Score : 0.861557478368356
indicator find using meanshift distance
Confusion Matrix :
[[158   0   0   0   1   1   0   0   0   0]
 [  0 107  10   0   0   2   3   0  23  13]
 [  1   9 132   3   0   0   0   1   5   3]
 [  0   0   0 140   0   1   0   6   8   6]
 [  1   9   0   0 152   0   0   3   3   0]
 [  0   0   0   4   1 127   2   0   0  31]
 [  0   4   0   0   0   0 157   0   1   0]
 [  0   1   3   0   3   1   0 154   9   0]
 [  0  18   1   1   0   1   1   1 125   7]
 [  0   2   0   3   0   4   0   7   4 144]]
Accuracy Score : 0.8627935723114957
```