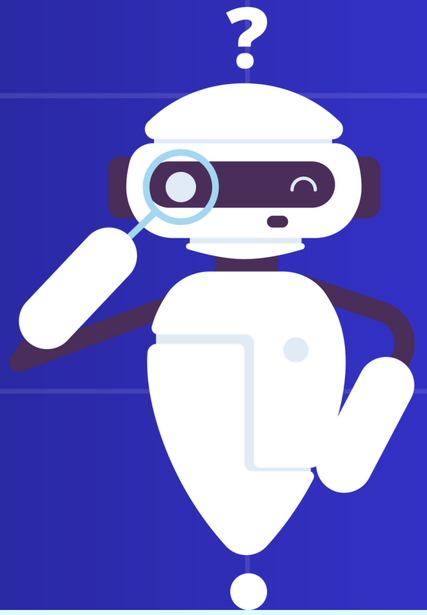
Sliced Wasserstein Distance for Learning Gaussian Mixture Models

Sarthak Kapoor CSE Roll No - 220101116
Srinjoy Som MnC Roll No - 220123074
Surankan De CSE Roll No - 220101121
Kashish Aggarwal CSE Roll No - 220101050
Aditya Kumar CSE Roll No - 220101005
Akansh Khandelwal CSE Roll No - 220101120

Introduction

Gaussian Mixture Models (GMMs) are widely used in machine learning and computer vision. Typically, Expectation Maximization (EM) estimates GMM parameters, but it may converge to suboptimal solutions due to the log-likelihood landscape. A better approach is based on minimizing the sliced Wasserstein distance between the GMM and the data distribution. The sliced Wasserstein distance has a smoother landscape, making it ideal for stochastic gradient descent and also shows increased robustness to initialization and captures high-dimensional data distributions more effectively than EM.



Wasserstein distance

In a metric space (Ω , d) say there are two probability measure ρ and ν. p-Wassertine distance is the minimum cost of any transformation map from ρ to ν.

```python
def pWasserstein(pdf1,pdf2,p):
    offset=1e-7
    pdf1=pdf1+offset
    pdf2=pdf2+offset
    pdf1=pdf1/pdf1.sum()
    pdf2=pdf2/pdf2.sum()
    CDF1=np.cumsum(pdf1)
    CDF2=np.cumsum(pdf2)
    inds=np.asarray(range(len(pdf1)))
    datpts=np.linspace(0,1,len(pdf1))
    Xpdf1 = np.interp(datpts,CDF1, inds)
    Xpdf2 = np.interp(datpts,CDF2, inds)
    u = np.interp(inds,Xpdf1,Xpdf1-Xpdf2)
    f = inds-u
    phi= np.cumsum(u/(len(pdf1)))
    phi-=phi.mean()
    Wp=(((abs(u)**p)*pdf1).mean())**(1.0/p)
    return f,phi, Wp
```

$$W_p(\rho,\nu) = \left(\inf_{\gamma\in\Gamma(\rho,\nu)}\int_{X\times Y} d^p(x,y)d\gamma(x,y)\right)^{\frac{1}{p}}, \quad (1)$$

where $\Gamma(\rho,\nu)$ is the set of all transportation plans, $\gamma \in \Gamma(\rho,\nu)$, and satisfy the following:

$$\gamma(A\times Y)=\rho(A) \quad \text{for any Borel subset } A\subseteq X$$
$$\gamma(X\times B)=\nu(B) \quad \text{for any Borel subset } B\subseteq Y$$

For 1D probability density function the formula is given by

$$W_p(\rho,\nu) = \left(\int_X d^p(J_y^{-1}(J_x(x)),x)d\rho(x)\right)^{\frac{1}{p}}$$
$$= \left(\int_0^1 d^p(J_y^{-1}(z),J_x^{-1}(z))dz\right)^{\frac{1}{p}}$$

**Where J is the CDF of the two distributions

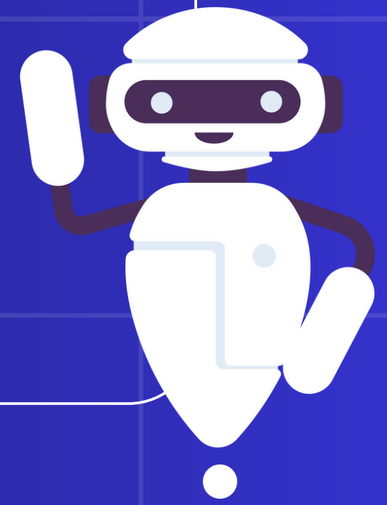
Radon Transform

```python
def radon_transform(L, d):
    T=np.zeros((L, d))
    t=np.random.rand(1, d)
    T[0,:]=t/ np.sqrt((t**2).sum())
    for i in range(1, L):
        t=np.random.randn(1, d)
        t=t / np.sqrt((t**2).sum())
        m=abs(np.matmul(T[:i, :], t.T)).max()
        while m > thresold:
            t=np.random.randn(1, d)
            t=t / np.sqrt((t**2).sum())
            m=abs(np.matmul(T[:i, :], t.T)).max()
        T[i, :] = t
    return T
```

The d-dimensional Radon transform R , maps a function f : R^d -> R given its integration over the d dimentional space is finite to a set of its integrals over hyperplanes of the space and defined as

$$R_f(\theta, p) = \int_{x \in \mathbb{R}^d} f(\mathbf{x})\, \delta(\theta \cdot \mathbf{x} - p)\, d\mathbf{x}$$

- where θ is a unit vector representing normal to a hyperplane
- p is the signed distance from the origin to the hyperplane
- δ is the dirac delta function which restricts
- integration to the hyperplane θ . x = p .



Sliced p-Wasserstien

The sliced pWasserstein distance approach involves first deriving a set of one-dimensional marginal distributions from a higher-dimensional probability distribution through linear projections by Radon transform. The distance between two distributions is then calculated as a functional based on the p-Wasserstein distance of these marginal distributions. In effect, this process entails solving multiple one-dimensional optimal transport problems, which have explicit solutions. Specifically, the Sliced Wasserstein distance between Ix and Iy is formulated as follows

$$SW_p(I_x, I_y) = \left(\int_{\mathbb{S}^{d-1}} W_p^p(\mathcal{R}I_x(., \theta), \mathcal{R}I_y(., \theta))d\theta\right)^{\frac{1}{p}}$$

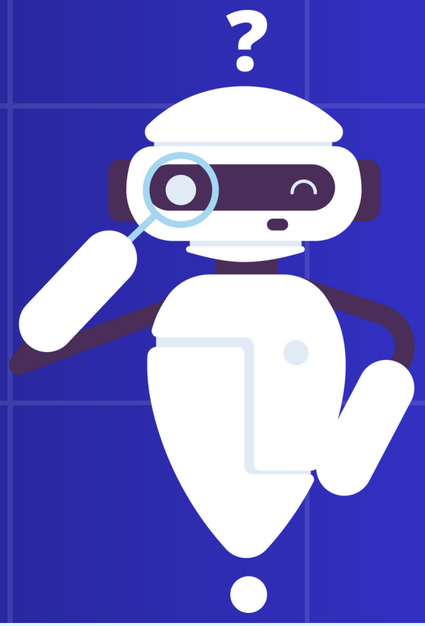
Gaussian Mixture Model

In context of GMM we have used p-wasserstein distance to calculate the minimum loss transport model i.e optimal transport model and use the transport potential of that sceme to calculate the modification of weight and means and standard daviation .Which performs better than negtive log-likelihood function and expectation maximization method.

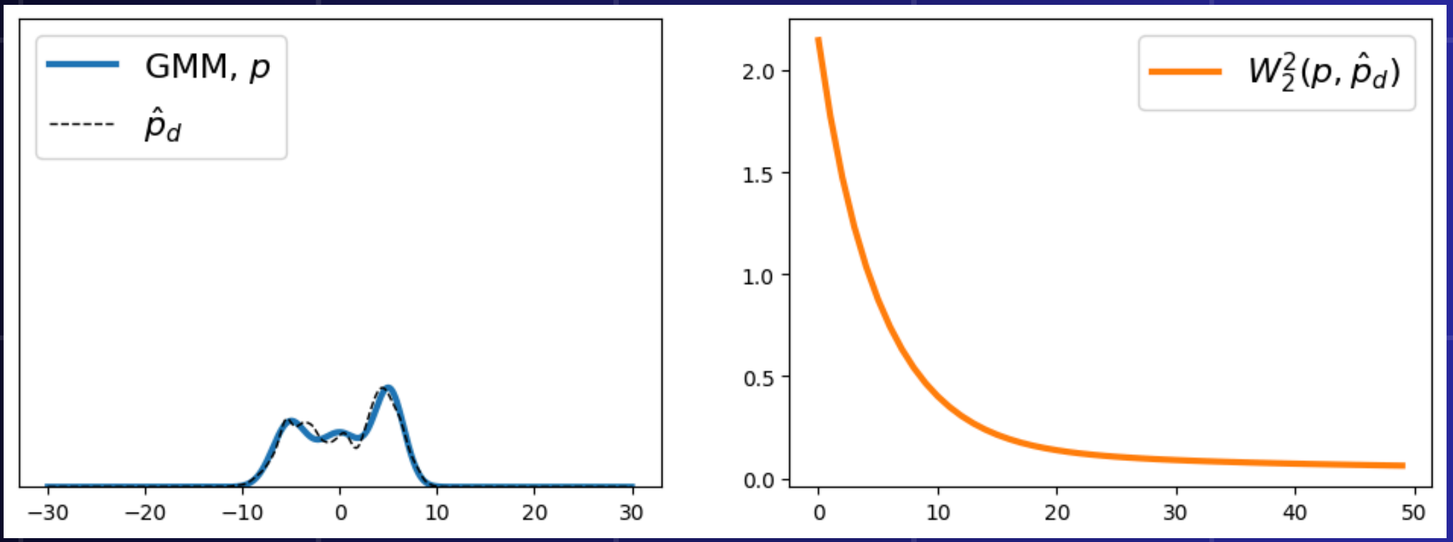
```
for i in range(K):
    dmu = alphas[i]*((phi)*dev_mu(t,means[i],sigmas[i])).mean()
    dsigma = alphas[i]*((phi)*dev_sig(t,means[i],sigmas[i])).mean()
    dalpha = ((phi)*normal_pdf(t,means[i],sigmas[i])).mean()

    means[i] = means[i]-lr*dmu
    sigmas[i] = max(sigmas[i]-lr*dsigma,0.)
    alphas[i] = max(alphas[i]-lra*dalpha,0.)

alpha_sum=np.sum(alphas)
alphas= [a/alpha_sum for a in alphas]
```



Results and Applications



The results after K iterations are shown below (K is according to the loop in the previous slide) as we can see the wasserstein distance reduces steeply at first and then stabilises , whose figure can be seen at the left

Some applications of sliced wasserstein distance can be in image comparison where the probability densities become the intensity of pixels

Resources used :
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8578459
https://github.com/skolouri/swgmm

Thank you