class VAE():

 def \_\_init\_\_(self,a,st=0.1,data\_shape=[100,40],unit=[3,1,3],s=0.0001,first\_point=15):

 self.num=data\_shape[1]

 ##encoder/decoder parameter

 self.w1 = np.random.randn(data\_shape[0], unit[0]) \* st

 self.w2\_mu=np.random.randn(unit[0], unit[1]) \* st

 self.w2\_sigma=np.random.randn(unit[0], unit[1]) \* st

 self.w3 = np.random.randn(unit[1], unit[2]) \* st

 self.w4 = np.random.randn(unit[2], data\_shape[0])\* st

 self.b1 = np.zeros(unit[0], dtype=float)

 self.b2\_mu = np.zeros(unit[1], dtype=float)

 self.b2\_sigma = np.zeros(unit[1], dtype=float)

 self.b3 = np.zeros(unit[2], dtype=float)

 self.b4 = np.zeros(data\_shape[0], dtype=float)

 self.a=a

 self.sigma=np.ones(data\_shape[0])

 ##latent model parameter

　　　　self.z0=np.zeros(unit[1], dtype=float)+first\_point

 self.m=np.zeros(unit[1], dtype=float)

 self.s=np.zeros(unit[1], dtype=float)+s

 self.log\_s=np.log(s)

 def encoder(self, x):

 self.x=x

 self.layer0=x-self.a

 self.layer1 = np.tanh(np.dot(self.layer0, self.w1)+self.b1)

 self.layer\_mu = np.dot(self.layer1, self.w2\_mu)+self.b2\_mu

 self.layer\_sigma=np.dot(self.layer1, self.w2\_sigma)+self.b2\_sigma

 self.u=np.exp(self.layer\_sigma/2)

 return self.layer\_mu, self.u

 def reparameterization\_trick(self, random):

 self.z=self.layer\_mu+random\*self.u

 return self.z

 def decoder(self, z):

 self.layer3 = np.tanh(np.dot(z, self.w3)+self.b3)

 self.out = np.dot(self.layer3, self.w4)+self.b4+self.a

 return self.out

 def latent\_model(self,z):

 next\_z=z+self.m

 return next\_z