**From PyTorch:**

**- Four Layers:**

1. FeatureSelector

2. GatingLayer

3. LinearLayer

4. MLPlayer

**- Three Models: (find the difference here between STG and L1RegressionModel)**

1. STGClassificationModel (MLPModel, ModelIOKeysMixin)

2. MLPModel (MLPLayer)

3. ModelIoKeysMixin (Object)

**- Model STGClassificationModel:**

0. layer: Get Input

1. layer: FeatureSelector

2. layer: Linear Hidde Layers (According to the hidden layers to append nn.Linear)

**- Loss Function:**

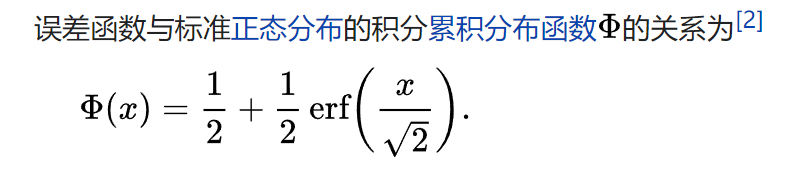
1. Loss part = nn.CrossEntropyLoss 🡪 for updating theta(normal parameters)

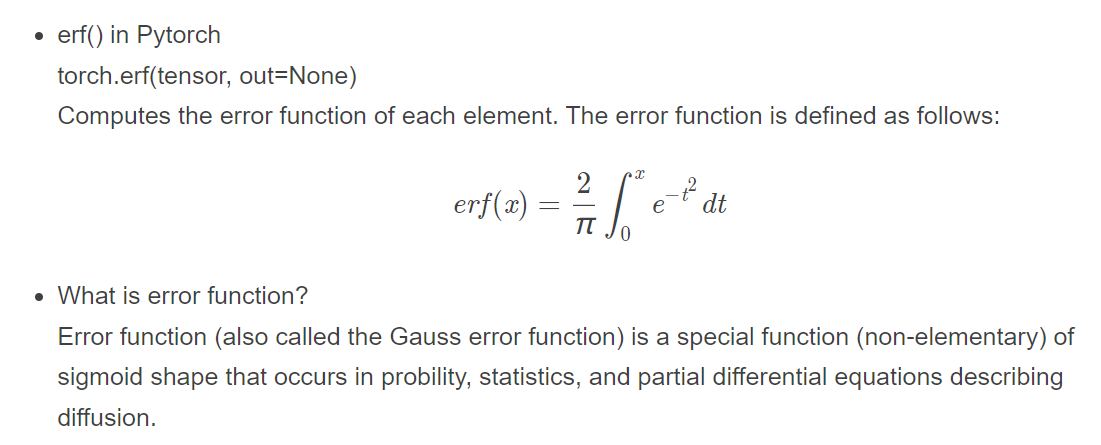
2. Regular part = torch.mean(self.***reg***((**mu** + 0.5)/ **sigma**)

***Reg:*** defined by layer FeatureSelector

Regularizer(self.x) 🡪 0.5\*(1+torch.erf(x / math.sqrt(2)))

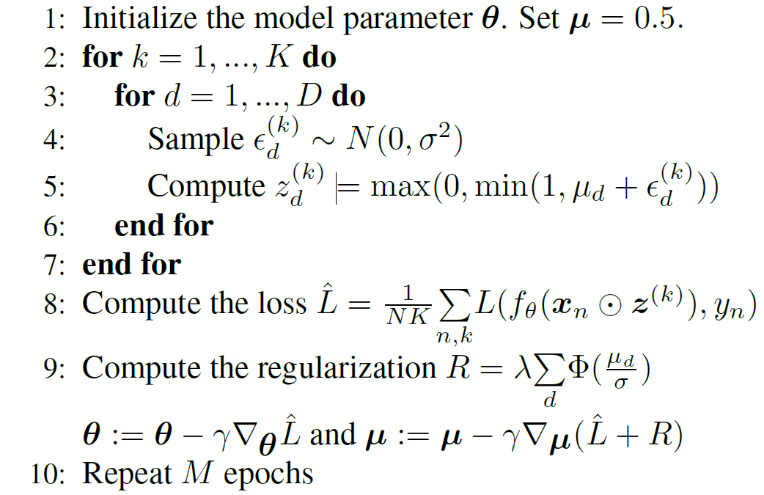
Torch.erf 🡪 gaussian error function





**mu,sigma**: are parameters which can be trained

Total Loss = 1.Loss part + 2. Regular part \* lam 🡪 for updating mu



**FeatureSelector:**

How to compute z?

**Step 1**

Z = self.**mu** + self.sigma \* self.**noise**.normal\_() \* self.training(==1)

Self.**mu** = torch.nn.Parameter(0.01 \* torch.randn(input\_dim, ), requires\_grad=True)

Self.**noise** = torch.randn(self.mu.size())

**Step 2**

Stochastic\_gate = self.hard\_sigmoid(z) (stochastic\_gate is the final z)

Hard\_sigmoid(z) = torch.clamp(z+0.5, 0.0, 1.0)

Torch.clamp(input, min, max, out=None) to restrict the value of input between[min, max]

**Step 3**

New\_x = prev\_x \* stochastic\_gate

Return new\_x