Torch nn Module

Tuesday, March 11, 2025 11:29 AM

Torch nn module is wide range of function its help to implement Nural network easy and efficent way.

pre-build layer - Activation function, optimizer, loss function and others.

Key Component in torch.nn:

common layer - liner, conv2d, Conv3d, nn.LSTM

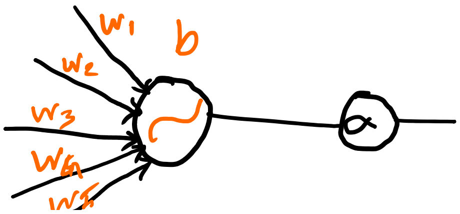
Activation fun: relu, softmax, sigmoid

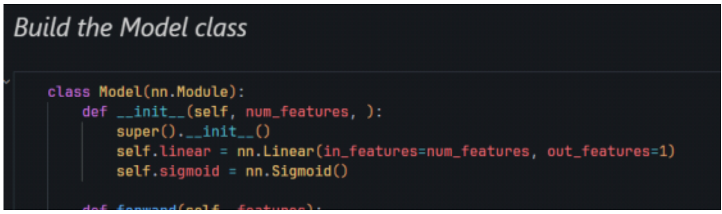
Loss function - nn.crossEntrophyloss, nn.MSELoss,

Container Module - Sequential module

Regularization And dropout: nn.Dropout()
nn.BatchNorm2d

→ Now build a simple neural network where only one Nuron. Binary classification problem.

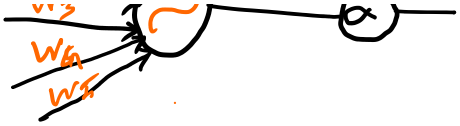


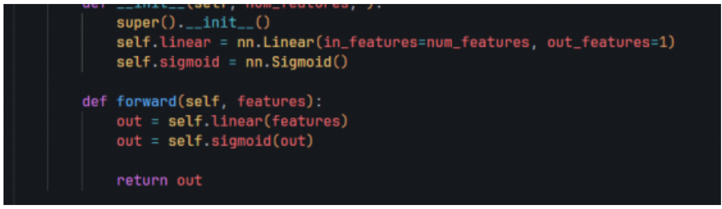


Build the Model class

```
class Model(nn.Module):
    def __init__(self, num_features, ):
        super().__init__()
        self.linear = nn.Linear(in_features=num_features, out_features=1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, features):
```





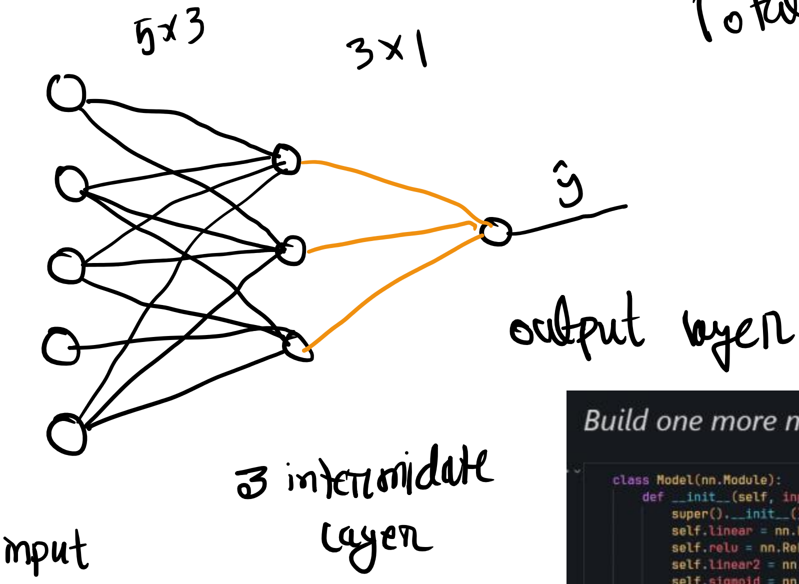
```
    def __init__(self, num_features, ):
        super().__init__()
        self.linear = nn.Linear(in_features=num_features, out_features=1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, features):
        out = self.linear(features)
        out = self.sigmoid(out)

        return out
```

Now build a bit complex Network.

Total Trainable paramiter

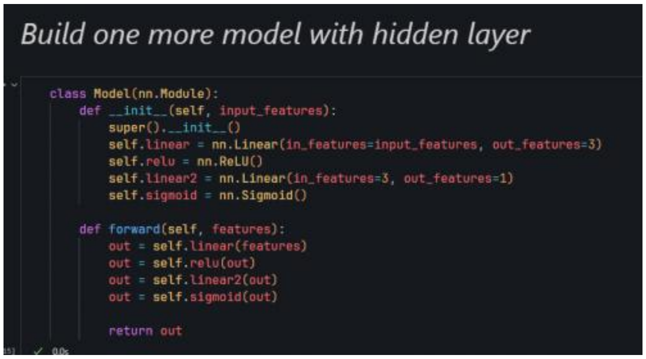


5x3 → 15

3x1 → 3

bias → 3+1

22



Build one more model with hidden layer

```
class Model(nn.Module):
    def __init__(self, input_features):
        super().__init__()
        self.linear = nn.Linear(in_features=input_features, out_features=3)
        self.relu = nn.ReLU()
        self.linear2 = nn.Linear(in_features=3, out_features=1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, features):
        out = self.linear(features)
        out = self.relu(out)
        out = self.linear2(out)
        out = self.sigmoid(out)

        return out
```

Torch.optim → SDG, Adam,

- Learning Schaduling. weight deay