The following are some materials that may help to build your own NN model.

- **Tools:**

  PyTorch,

  TensorFlow,

  MXNet,

  Keras,

  Matlab, etc.

- **Environment:**

  Anaconda, Jupyter Notebook, etc.

- **Installation:**

  PyTorch: https://pytorch.org/ **(official website)**

  https://blog.csdn.net/phykami/article/details/83311372

  video link: https://morvanzhou.github.io/tutorials/machine-learning/torch/1-2-install/

  TensorFlow: https://www.tensorflow.org/install

- **Demo:**

建立数据集

在这个例子中，我们创建一些仿真数据来模拟真实的情况，（也可以直接导入已有的数据），比如一个带噪声的一元二次函数：$y = ax^2 + b + \epsilon$,

```
import torch
import matplotlib.pyplot as plt

x = torch.unsqueeze(torch.linspace(-1, 1, 100), dim=1)  # x data (tensor), shape=(100, 1)
y = x.pow(2) + 0.2*torch.rand(x.size())                 # noisy y data (tensor), shape=(100, 1)

# 画图
plt.scatter(x.data.numpy(), y.data.numpy())
plt.show()
```

建立神经网络

我们建立一个神经网络我们可以直接运用 torch 中的体系，先定义所有的层属性（__init__()），然后再一层层搭建（forward(x)）层与层的关系连接，（本例只搭建了两层，层数等各种参数需你们自己设定）。建立关系的时候，我们会用到激励函数（activation function），这里我们用了 ReLu 函数（你可以选其他比如 sigmoid 等等，可以自己去了解他们的区别）。

```
import torch
import torch.nn.functional as F    # 激励函数都在这

class Net(torch.nn.Module):  # 继承 torch 的 Module
    def __init__(self, n_feature, n_hidden, n_output):
        super(Net, self).__init__()    # 继承 __init__ 功能
        # 定义每层用什么样的形式
        self.hidden = torch.nn.Linear(n_feature, n_hidden)  # 隐藏层线性输出
        self.predict = torch.nn.Linear(n_hidden, n_output)  # 输出层线性输出

    def forward(self, x):  # 这同时也是 Module 中的 forward 功能
        # 正向传播输入值, 神经网络分析出输出值
        x = F.relu(self.hidden(x))      # 激励函数(隐藏层的线性值)
        x = self.predict(x)             # 输出值
        return x

net = Net(n_feature=1, n_hidden=10, n_output=1)

print(net)  # net 的结构
"""
Net (
  (hidden): Linear (1 -> 10)
  (predict): Linear (10 -> 1)
)
"""
```

训练网络

训练的步骤很简单，如下：（optimizer 还可以选择很多比如 Adam, etc.）

```
# optimizer 是训练的工具
optimizer = torch.optim.SGD(net.parameters(), lr=0.2)  # 传入 net 的所有参数, 学习率
loss_func = torch.nn.MSELoss()      # 预测值和真实值的误差计算公式 (均方差)

for t in range(100):
    prediction = net(x)     # 喂给 net 训练数据 x, 输出预测值

    loss = loss_func(prediction, y)     # 计算两者的误差

    optimizer.zero_grad()   # 清空上一步的残余更新参数值
    loss.backward()         # 误差反向传播, 计算参数更新值
    optimizer.step()        # 将参数更新值施加到 net 的 parameters 上
```

可视化结果

一般用“matplotlib.pyplot”来可视化，下面是如何整个训练过程的可视化过程（一般 show 个结果图就好了。）

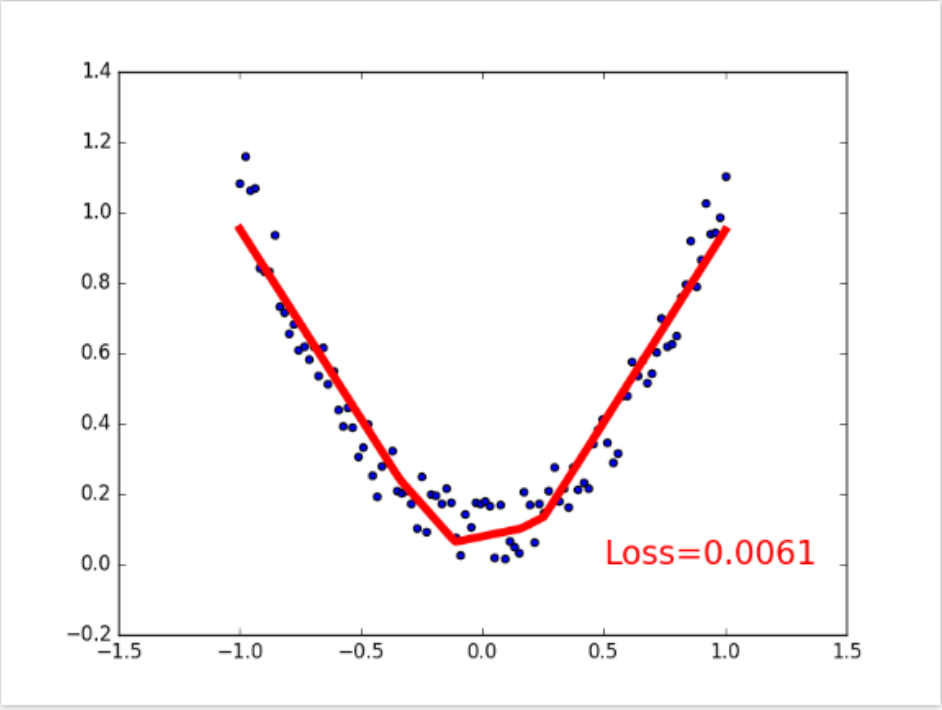
```
import matplotlib.pyplot as plt

plt.ion()   # 画图
plt.show()

for t in range(200):

    ...
    loss.backward()
    optimizer.step()

    # 接着上面来
    if t % 5 == 0:
        # plot and show learning process
        plt.cla()
        plt.scatter(x.data.numpy(), y.data.numpy())
        plt.plot(x.data.numpy(), prediction.data.numpy(), 'r-', lw=5)
        plt.text(0.5, 0, 'Loss=%.4f' % loss.data.numpy(), fontdict={'size': 20, 'color':  'red'})
        plt.pause(0.1)
```



- **Videos:**

  PyTorch: https://morvanzhou.github.io/tutorials/machine-learning/torch/

  TensorFlow: https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/

- **Document:**

1. Official Tutorials for PyTorch: https://pytorch.org/tutorials/
2. 动手学深度学习:

   MXnet:

   (Chinese version)

   http://zh.gluon.ai/chapter_deep-learning-basics/index.html#

   (English version)

   https://d2l.ai/

   PyTorch:

   http://tangshusen.me/Dive-into-DL-PyTorch/#/

3. 机器学习-吴恩达:

https://www.coursera.org/lecture/machine-learning/welcome-to-machine-learning-zcAuT