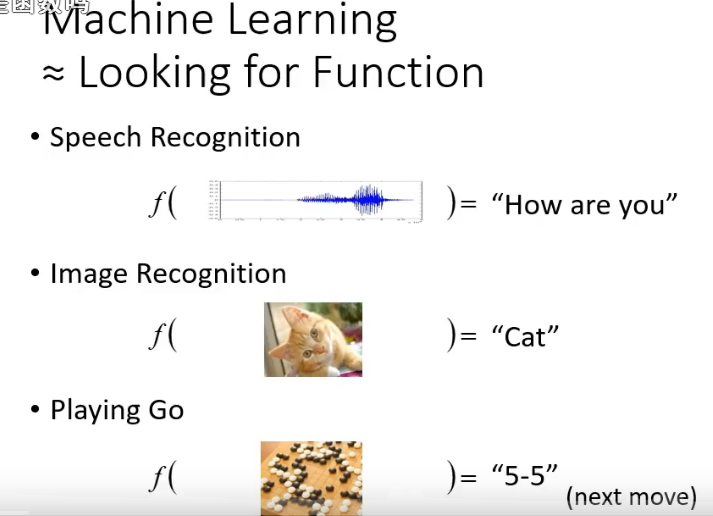
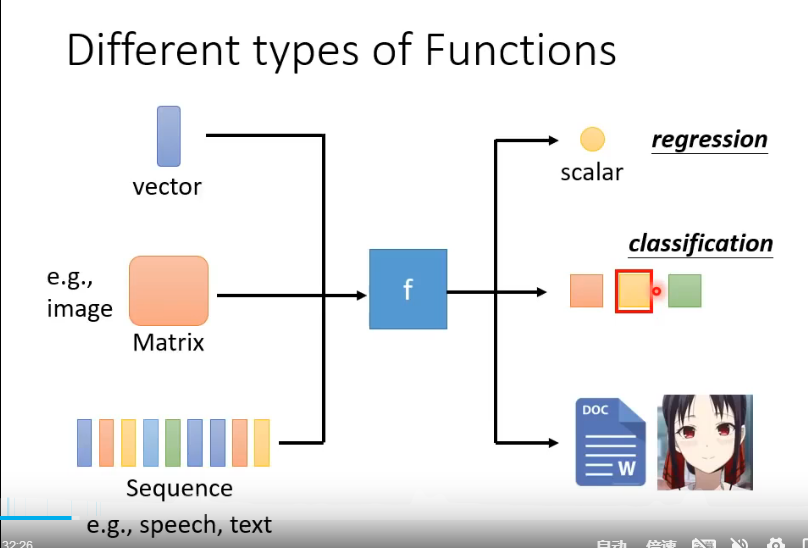
什么是机器学习？

机器学习是帮我们找一个复杂的函数来解决问题

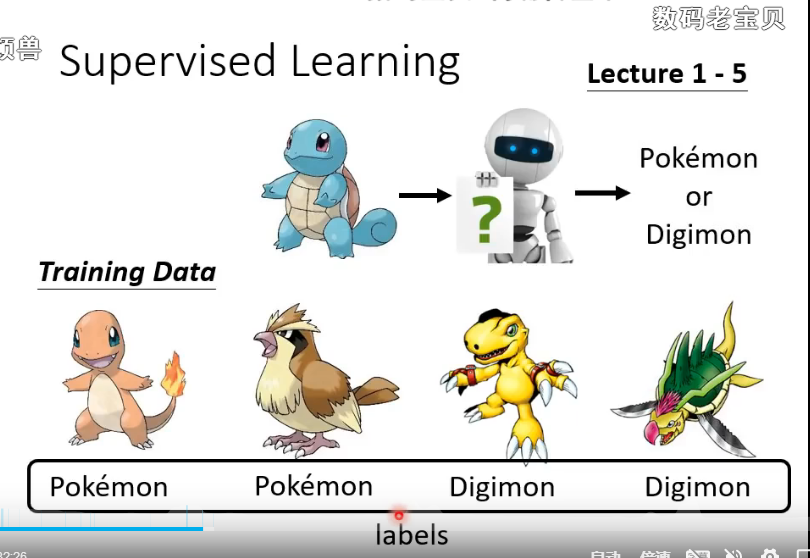


函数的输入可以是一个向量、矩阵(一个图片)、序列(声音或者一句话或者是一段文字)。

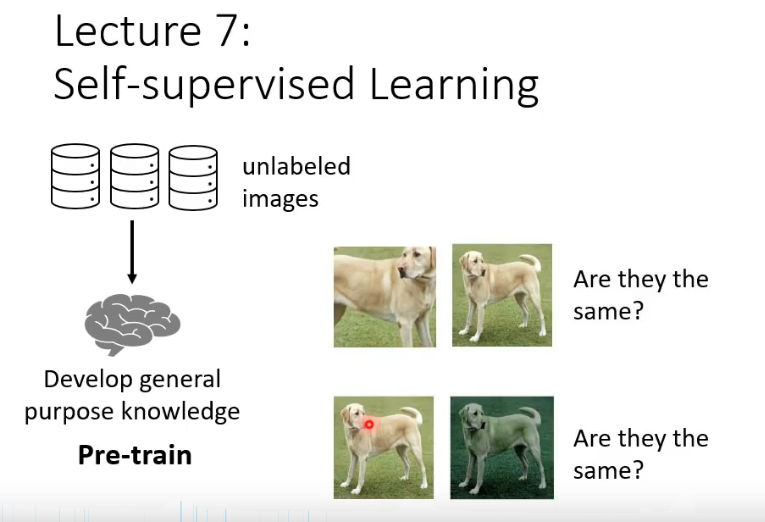
输出可以是一个数值(regression)、类别(classification)、文章或者图片。



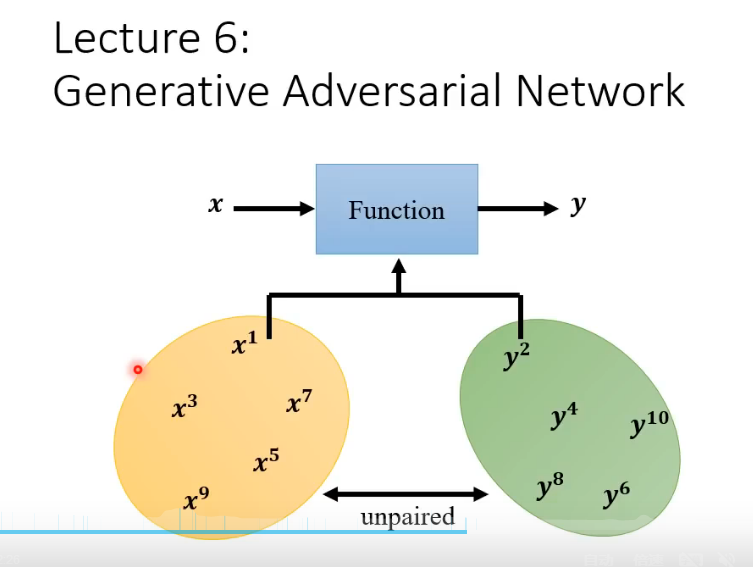
监督学习的做法就是要收集大量的训练资料，在收集过程中要给出对应的标签。在训练的过程中将图片放入网络中然后输出结果与其对应的标签做对比计算损失进而更新网络参数。



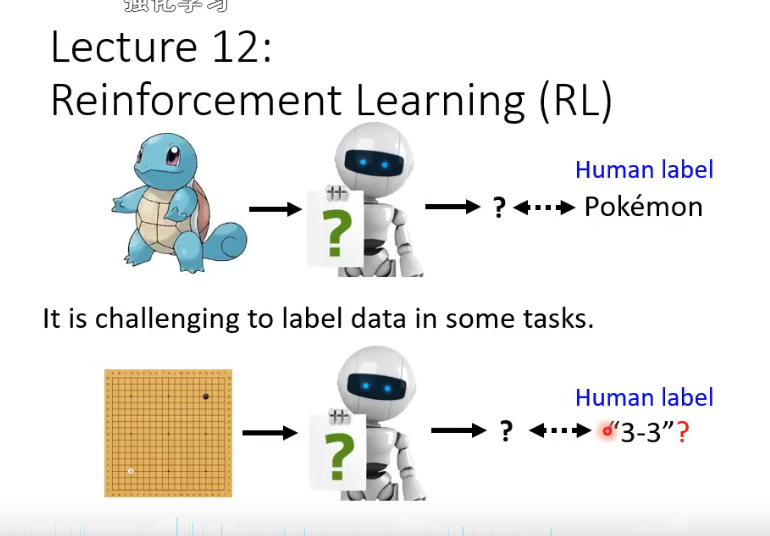
半监督学习就是在训练模型之前做预训练，让机器学会做各种不同的影像辨识相关任务的基本功，练好后就可以在新的任务上做好。



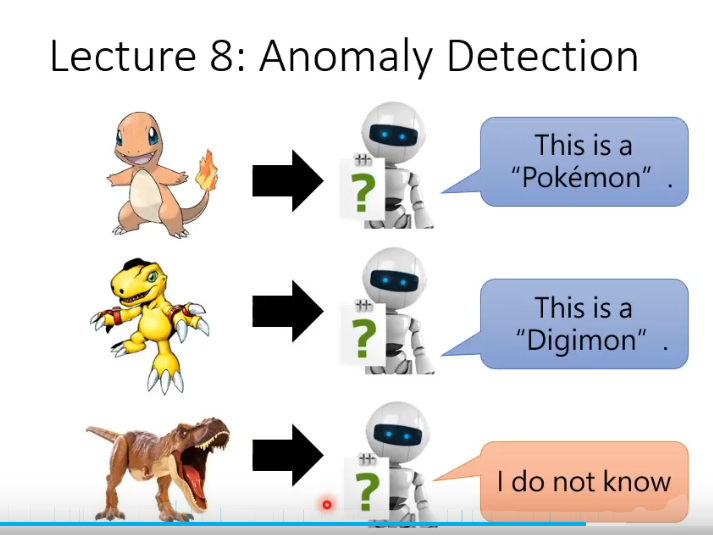
生成对抗网络只需要收集到大量的x和大量的y不需要它们之间成对的关系机器就可以自动把它们之间的关联找出来



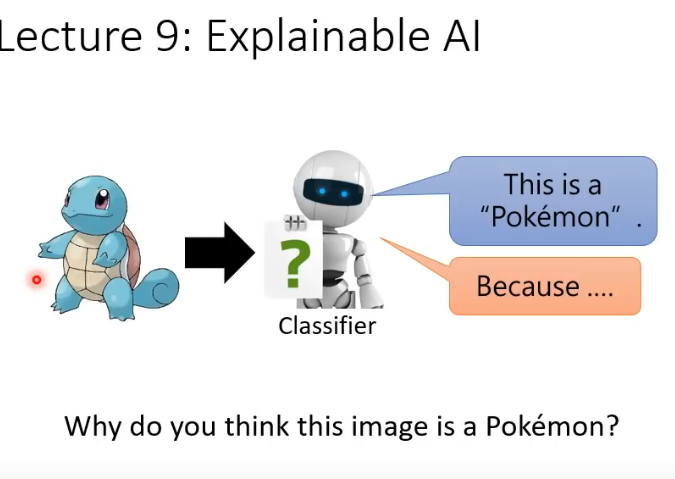
强化学习当不知道怎么标注资料但是可以判断好坏并且可以定义什么叫做成功的时候就可以使用强化学习来解决。



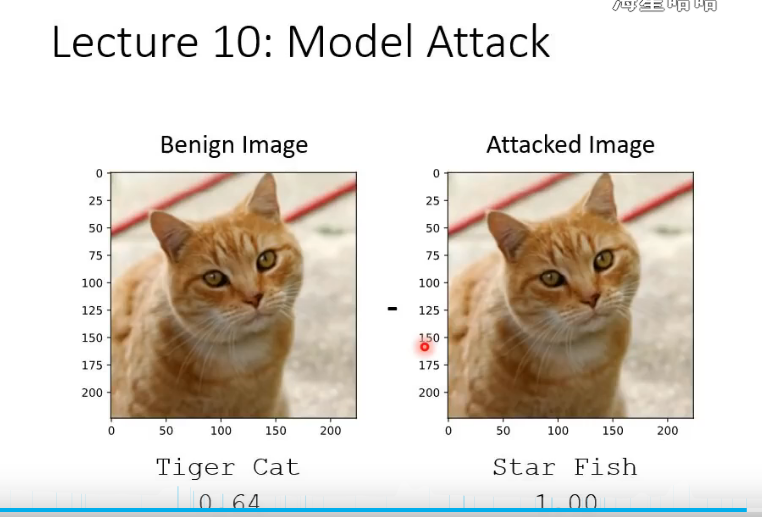
异常检测就是当出现一个类别是当前模型没有学习过的类别时需要模型输出我不知道。所以异常检测就是要让模型能够输出我不知道。



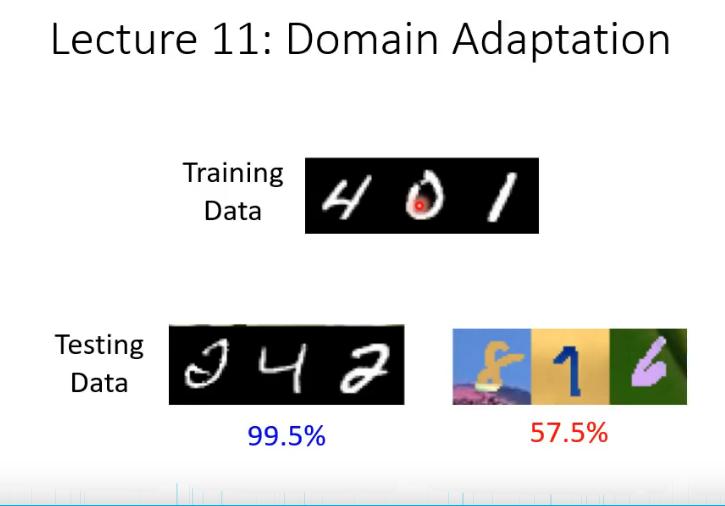
可解释性AI就是要让模型让我们知道为什么它能够知道答案是这个。



模型攻击就是通过在图片上添加一些人肉眼看不出来的噪音，使得模型识别错误



Domain Adaptation就是假设训练资料和测试资料的分布是类似的但是在实际应用的时候可能会与训练和测试的资料有差异，这种差异就有可能使得模型识别正确率暴跌，Domain Adaptation就是来解决这个问题。



机器学习的类别

Regression:模型的输出是一个数值。

例子：通过输入今天的PM2.5的值、平均温度和平均的臭氧气浓度等输出每天中午的MP2.5的值，找这个函数的任务就叫做Regression。

Classification：通过给出一些选项也就是类别来作为模型的输出。

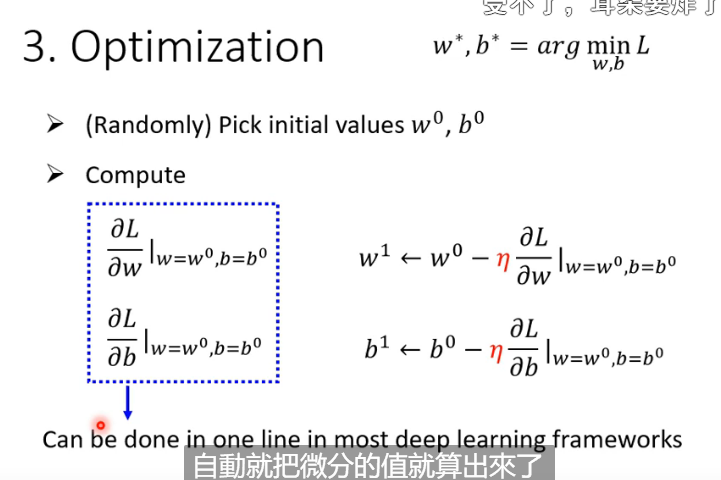
例子：通过输入一封邮件模型来判断并输出这封邮件是否是垃圾邮件。

Structured Learning：输出是一个有结果的东西

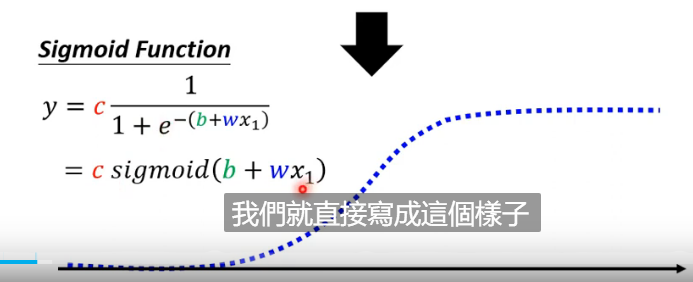
例子：机器画一张图、写一篇文章。

机器学习找出一个函数分为三个步骤

1. 写出一个带有未知参数的函数，为y=wx+b,y是要预测的值，w和b分别是权重和偏置，x则是一些已知的数值，w和b是要通过学习不断更新得到的参数。
2. 损失函数是通过输入模型的输出值和真实值来计算它们之间的差异，从而得知当前函数的参数学习的好坏。损失函数通常有MAE、MSE、Cross-entropy等。
3. Optimization就是去找出最佳的w和b使得损失值降到最低，通过梯度下降来调整w和b的值找到最小值点获得最佳的w和b。通过计算w和b的微分乘以学习率来获得新的w和b的值，通过设置epoch来确定学习的次数。



Sigmoid激活函数：当x非常大的时候y会趋于1，当x非常小的时候y会趋于0。



Loss就是计算当前参数值的准确度，通过将x放到参数中计算出y然后和真实的计算它们之间的差距，把所有的误差加起来就得的Loss。

Optimization of New Model

计算模型参数的梯度，然后根据梯度来更新参数，然后再算梯度，再更新参数，不断重复操作直到达到规定的次数或者直到无法更新参数为止。

数据读取

Dataset：通过输入数据集所在的路径来读取数据集。

Dataloader：将读取的数据集按照batch\_size分组，并且选择是否打乱，参数有dataset、batch\_size、shuffle。

MyDataset类中\_\_init\_\_方法在实例化MyDataset类中执行读取数据，\_\_getitem\_\_方法索引想要的数据，\_\_len\_\_方法获得数据的长度。需要重写这三个方法来。

Tensors就是一个高维度的阵列。

torch.from\_numpy函数是将numpy类型转换为tensor类型

torch.zeros函数是生成全为0的tensor类型

torch.ones函数是生成全为1的tensor类型

x.sum()是求和可以指定维度在指定维度求和

x.mean()是求平均值可以指定维度在指定维度求平均值

x.pow()是对x求高次幂

x.transpose()函数是将x的维度进行交换

x.shape获得x的维度大小

x.squeeze是将x的维度减少

x.unsqueeze是将x的维度增加

torch.cat([x,y,z],dim=1)函数是将x,y,z在1维进行拼接，并且x,y,z除了1维可以不一样外其他维度必须一样

x.to(‘cpu’)是将x放入cpu中

x.to(‘cuda’)是将x放入gpu中

z.backward()计算z的梯度

torch.nn.Linear(32,64)表示一个线性层，x的输入维度是32输出维度是64

nn.Sequential()是将里面的网络层封装起来

forward函数将数据放入网络里面并且输出

nn.MSELoss()是来计算和正确答案的差距一般是用在回归任务

nn.CrossEntropyLoss()是用在分类任务中

torch.optim.SGD()通过给模型参数学习率来进行梯度下降

optimizer.zero\_grad()将模型参数的梯度清0

loss.backward()计算模型参数的梯度

optimizer.step()优化模型的参数

model.train()将模型状态改为训练模式

model.eval()将模型状态改为验证模式

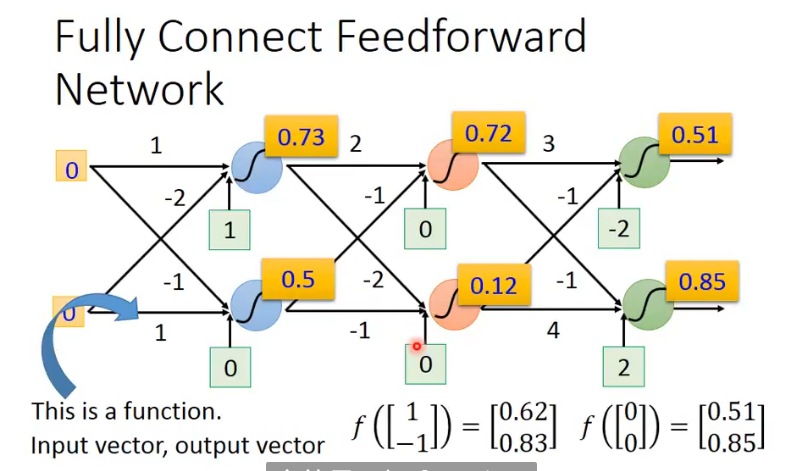
torch.save()保存模型参数

torch.load()读取模型参数

pd.read\_csv()读取csv文件数据

训练一个深度神经网络首先要有一个数据集然后要构建好神经网络，定义好损失函数还有优化器有了这些后就可以进行训练

前向传播是将数据x放入搭建好的网络中不断的计算最后得到结果的过程

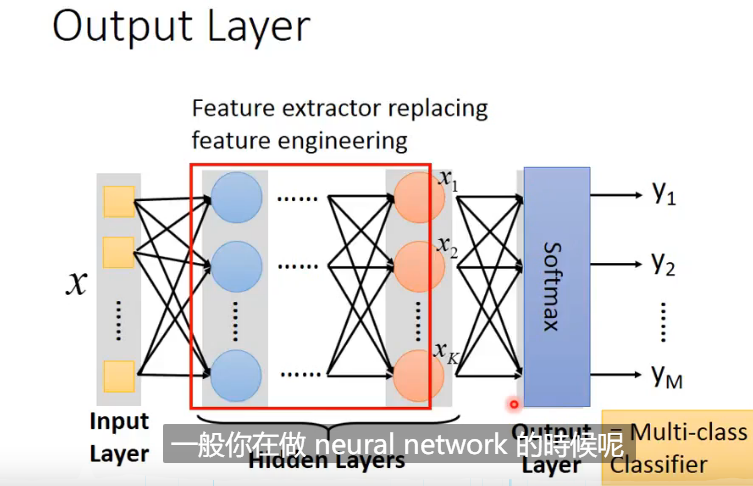


深度神经网络分为三层，Input Layer、Hidden Layers、Output Layer

Input Layer就是输入的数据

Hidden Layers 是所构建的神经网络

Output Layer 是通过Hidden Layers后得到的输出再过一个线性层然后便可以得到最后的结果，有几个结果就有几个输出结果。在分类的任务中会有一个SoftMax将结果归一化，获得对应的概率，选择概率最大的结果输出



反向传播

计算出所有参数的梯度，然后通过学习和梯度来更新计算参数

