李宏毅机器学习笔记

**P1,P2--2022机器学习相关规定(相关基本概念)**

1. 什么是机器学习？寻找一个合适的函数，例如语音识别中采用一个函数将语音转换为一句话，图像识别中输入图片通过函数输出图像的类别，阿法狗是输入当前棋局的情况，通过函数让阿法狗选择合适的落子点。是拟合一种写不出的复杂函数。
2. 在机器学习中我们着重于深度学习，用类网络来表示复杂函数。对于网络的输入一般都是向量，矩阵，序列。输入一般是标量，或者几个中的类别，更复杂的输入文件或图片。
3. 有监督学习？收集大量的数据，数据都具有标签。
4. 自监督学习？通过无标签的数据在训练之前我们使用预训练方法学习基本的能力，例如不需要从0开始训练模型参数，减少学习时间，利用简单的任务让模型具有简单的能力，再处理复杂模型。
5. 现在都是大模型。Bert GPT4
6. 生成式对抗网络？存在一个生成器，一个判别器，让判别器是判别生成的模型是真还是假，通过训练，提高两个模型的泛化能力，以此达到可以生成以假乱真的图片。强化学习？阿法狗采用的技术。
7. 异常检测？例如在一个分类问题中，通过输入错误的数据(不属于任意一类的特征)，我们的模型可以检测到这种情况，而不会输出一个类别的标签。
8. 可解释AI？回答问题的同时可以告诉我们为什么。探究正确率高的背后原因。
9. 模型攻防？通过增加一个小噪音使图片分类发生变化。
10. 领域自适应？ 是同一种方法可以解决小范围变化的情况。
11. 模型压缩？压缩大模型的参数数量
12. 终生学习？让模型不断学习达到万能的AI。
13. 元学习？让模型在过往的基础发明自己的算法。

**P3,P4--机器学习的基本概念**

1. Regreesion: 线性回归输入一般是标量或者向量，输出是一个标量。
2. Classification：分类问题 输出是类别中的一类。
3. Structured Learning：让机器学会创造，输出图片，文章。
4. ML的训练步骤:将李宏毅youtube频道的日活记录作为线性回归的输入。
5. unknown parameters:首先尝试一次函数y=wx+b拟合求解明天得点阅次数，且只通过昨天的点阅次数作为输入，其中w和b就是我们未知参数是需要模型训练得到的参数。
6. Define loss: 由1)可知，我们需要合适的w,b来拟合函数，则通过loss(w,b)我们希望可以得到好的参数w,b。如何评价参数的好坏？通过代入计算y通过预测的和实际的y的差距loss=|y-|，如果越近则参数更好，在数学上则表示为模型的loss更低，我们可以通过不同天的累加loss\_acc作为损失，也可以求总loss\_acc的平均作为损失。一次方的loss通过绝对值来计算，也可以使用平方差来作为损失函数，或者多分类中的交叉损失函数。
7. Optimization：采用梯度下降(Gradient Descent)，例如在两维图像中，假设随机初始化的，按照梯度的定义，梯度时沿着变化最大方向，也可以简单的认为总是指向极值点，此时通过计算当前位置的梯度，将进行乘以学习率来确定此时朝向最好的W走多远，过大学习率导致导致在极值点边缘进行跳跃，过低学习率会导致需要训练多轮才得到W。学习率是我们模型中的超参数(hyperparameters)，是我们认为设置的参数，同样设置的超参数还有训练的轮数(epochs)。梯度下降好像会陷入局部最优解。优化过程是，通过微分更新w和b。

修改模型将考虑一周的点阅量，函数变为.

5.通过一系列函数的累加来模拟更复杂的函数。一个线性函数难以拟合复杂的函数。分段的一次函数可以认为常数项b组合不同的一次函数y。由此我们想到曲线如何拟合？我们可以才曲线上采点构造直线，只要采点够多，我可以化曲为直，采用够多的直线去拟合所有曲线。介绍了sigmoid函数。通过s函数进行拟合。

. 将变为将计算式子写出来，可以将计算简化为矩阵之间的计算.再使用sigmoid函数引入非线性来拟合曲线Step1:function with unknown

)

Step2:define loss from training data

Step1中的参数都是我们所需要训练的，对每一个参数都求微分计算梯度()，和前面一样，仅仅只是通过向量进行参数的更新。g来修改参数。在训练的情况中我们选择将数据分成小批量(batch\_size)通过小批量来更新参数，将所有数据使用一次叫做一次epoch。

介绍新的激活函数Relu()，实质是MAX(0,X)函数，两个Relu可以合成一个sigmoid。

通过修改激活函数或者修改层数来优化模型的能力。给神经网络改名字，增加层数，发现能力增强，但层数过多会出现过拟合(over fitting)，提出为什么进行深度学习而不是宽度学习？

Step3: optimization

**P6,P7--Pytorch基本操作和训练流程**

1. what is pytorch

一种ML的学习框架，特点是采用张量(tensor)进行计算，并且可以使用到GPU加速运算，还可以自动计算微分。

1. training &testing Neural Networks in pytorch

首先需要上传数据要用到4，训练步骤和P3一样三步走，反复训练验证，最后进行测试。

1. Dataset&Dataloader

torch.utils.data.Dataset存储各种数据包。

torch.utils.data.Dataloader将数据划分成batch\_size，迭代训练。包含特征和标签，参数shuffle可以设置是否打乱进行划分。

Dataset类

\_\_getitem\_\_用于获得某一数据

\_\_len\_\_获得数据的大小

1. Tensors

Pytorch中的高维矩阵，一维称作标量，两维向量，三位以上叫做张量(tensor)。

torch.shape()获得数据的维度。

torch.zeros()建立全零张量。

torch.ones()建立全一张量。

torch.tensor()建立张量或者转换张量，requires\_grad是否保留当前梯度信息

torch.transpose(x,y)交换张量的维度。

torch.unsqueeze(x)在dim=x增加一维。

torch.squeeze(x)在张量上减少一维 dim=x.

torch.cat([a,b],dim=x)将两个张量a,b在下标为x的维度进行拼接。

.to(device)将数据或者模型转移到cpu或者GPU上。

torch.cuda.is\_available()查看是否有GPU。

x.backward()计算梯度进行反向传播。

x.grad 显示保留的梯度信息。

5.torch.nn :model Loss Functions

nn.Linear(x,y) 输入x长度,输出y长度

layer.weight.shape 输出层的参数矩阵形式。 bias同理

nn.Sigmoid() nn.ReLU()激活函数

nn.Sequential()用于构造网络里面可以包含多层，定义你的网络的结构。

nn.MESLoss() 均方差。

nn.CrossEntropyloss() 交叉熵方差 用于多分类。

1. torch.optim: Optimization

torch.optim.SGD(model.params,lr)随机梯度下降更新参数，学习率是lr。

训练epoch

optimizer.zero\_grad() 每次训练前清零前一次训练保存的梯度

loss.backward()反向传播 存梯度

optimizer.step()更新参数

1. Save/load models

torch.save(model,path)将模型model存在路径path.

torch.load(path)从路径path上传模型

model.load\_state\_dict()上传存下的权重,可以使用在预训练。

Model.eval()测试模式不需要计算梯度，需要 with torch.no\_grad()。

**P11--2022作业**

same\_seed()获得相同的种子，保持结果的一致？

train\_valid\_split()分割训练集和测试集切成几比几。

predict()通过训练的模型和测试集获得预测输出列表。

COVID-19Dataset()自己定义的数据，先将数据转换为FloatTensor()

my\_model()构建一个16--8--1的三层线性层。

select\_feat() 选择是使用全部的特征还是前五个特征。

trainer() 和前面的一样采用均方差和随机梯度下降，采用summarywriter()，还实现了loss的可视化，采用early\_stop减少无用的训练次数。

Config={} 设置需要用到的超参数，例如种子seed,训批量大小batch\_size。

**P13--深度学习简介**

讲解深度学习发展历史，深度学习还是三步走，深度表示具有非常多的隐层。

Step1:Neural Network

讲解简单的2个输入3层的全连接层，网络由输入层，输出层，隐层组成，前一层的输出是后一层的输入。和前面一样我们可以将每层运算作为矩阵运算得到结果。可以认为网络计算就是经过了多次矩阵运算，不断地抽取杂糅特征，在多分类问题中通过最后特别好的特征进行softmax进行多分类。

简单的手写数字分类，就是输入是256的向量，输出是10的类别概率，但 是我们可以自己设置隐层的数量，和神经元的个数。

Step2:goodness of function

Step3:pick the best function

和机器学习一样通过loss的大小来评估模型的能力，计算总损失然后通过梯度下降继续更新权重。

为什么深度是好的？ 随着层数变多，参数变多发现模型损失减少，但是如果一层足够多的参数是否可以替代深度很深的模型？

宽度学习的代价远大于深度学习的代价，计算量是平方增长。

**P14--反向传播**

深度学习和前面机器学习不一样的就是参数多，为了有效的计算这么多参数，我们选择了反向传播，最重要的是就是数学中的链式法则(Chain Rule)，因为神经网络的某一参数可以认为是一条链，从后层往前层看认为是微分的累加，我们将所有需要求得参数都认为是一个关于我们可以通过对和计算，在整条链中我们需要借助中间输出量Z，在前向传播的过程我们发现是每层输入的X，为此我们计算梯度时需要保存所有前向传播的输出用于计算，非常占内存。对于后向传播中，发现这个积分由Z指向后层的的和得到，所以从输出向前传播，从最后一层不断往前指向的Z的所有经过的链子就是我们需要求。为此我们需要思考就和怎么求，如果此时是输出层我们直接可以使用和激活函数直接算出，如果不是输出层还是隐层，为此我们需要继续计算Z1和Z2后面的输出，所以反向传播就是从输出出发向前计算各层的损失。