# 1. 神经网络介绍

## 1.1 神经元

前向传播g(wx+b)

## 1.2 浅层/深层神经网络

Loss

Optimize

Gradient descent

Backpropagation

Overfitting

# 2. 深度学习的训练流程

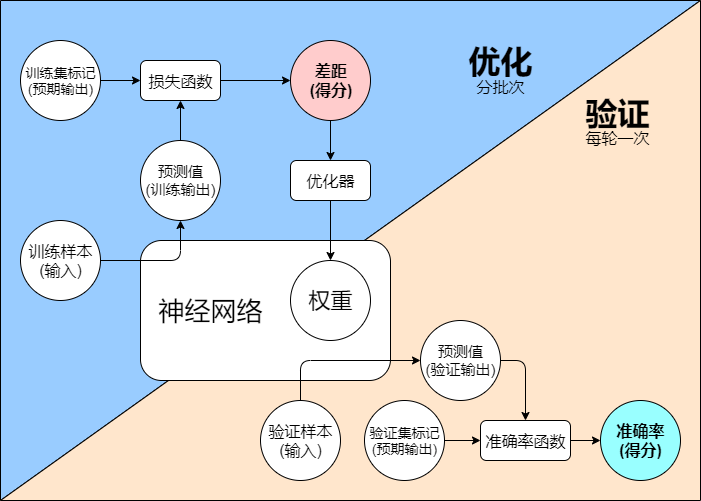
深度学习是学习神经网络中的权重的过程, 或者说是寻找最优权重值的过程.

训练是分多轮进行的, 一轮是指把所有训练样本都用一遍.

每轮包含两个部分, 优化和验证.

* 优化的任务是优化权重,
* 验证相当于做阶段性的总结, 以便对训练状态有个总的了解(模型准确率和过拟合情况).
* 一个很好的比喻: 训练是不断的做作业, 验证是一次阶段性的考试.

具体的流程如下图:



# 2. 深度学习的技术概念

## 轮次

轮次是深度学习训练过程中一种工作分段.

* 1轮 (Epoch) =所有的训练样本都用一遍.
* 对样本的使用包括:
  + 把样本输入网络得到预测值(这步称为前向传播)
  + 计算Loss,
  + 计算梯度(这步称为反向传播),
  + 优化权重.
* 相关超参数: 训练多少轮
  + 可以设定条件来自动终止训练(early stopping)
  + 可以提前终止(Ctrl+c), 也可以在结束后继续加练
    - 开启自动记录快照(每隔几轮记录一次)
    - 需要继续的时候, 指定从哪个快照上继续
    - 其中一种应用是停下来调整学习率, 然后继续

## 小批量

在训练的过程中, 会按批(batch)使用样本.

* 相对于训练样本总量, 每批样本的数量通常会比较小, 这样就可以快速调整权重, 让权重得到更多的调整机会, 让模型可以快速收敛.
* 就像我们学习投飞镖时, 会根据投掷的结果边练边调整姿势; 不会只管投而不看结果也不调整动作.
* 相关参数: 批量大小(batch size)
  + Kares里面默认是32
  + 设定时, 需要考虑训练样本总量的多少, 以及样本的特点

## 样本,标记和预测值

在监督学习中, 模型通过预测结果并对比标记好的数据, 来学习和改进神经网络的权重.

* 可以理解为不断的做题, 看看自己和参考答案差多远
* 样本 = 输入数据 + 标记(参考答案)
* 预测值是把输入数据通过神经网络计算后得出的结果

## 数据集的划分和洗牌

通常我们会把数据集划分为三个部分: 训练集, 验证集和测试集.

* 训练集相当于练习册, 验证集是模拟考试题, 测试集是真正的考试题
  + 训练册用于练习, 模拟考摸摸底, 最后真正的考试题来给最终的结论.
  + 数据集的划分可以在训练过程中提供客观评价, 防止过拟合
  + 过拟合就是刷过的题都会, 但是思路和能力不行, 见了新题型就懵
  + 三个集合应该是同分布的
* 数据集的划分的比例
  + 常见的是8:1:1
  + 和数据总量有关, 总量越大, 训练集的比例通常越大
* 每轮训练前通常会对数据进行洗牌, 以避免以下非随机因素干扰训练
  + 数据搜集导致的: 同质数据扎堆, 周期规律
  + 数据样本的固定顺序

## 训练效率

随着深度学习网络层数的加深, 再加上数据样本的数量巨大, 训练往往耗时费力,会产生大量的费用(比如电费, 和工作人员的时间). 所以训练效率也是非常重要的, 效率因素有很多, 包括(但不限于):

* 软硬件框架运行的效率, 比如:
  + 如何提高多GPU的并行工作效率
  + 根据模型和问题的特点选择合适的系统配置
* 模型执行的效率, 比如:
  + 通过预处理, 优化那些重复执行的部分
  + 选择合适的洗牌策略, 提高数据加载效率
  + 选择合适的批量大小
  + 选择合适 (甚至不断调整)的学习率

## 前向传播

### 激活函数

### 单次神经网络

### 深度神经网络

## 损失函数,准确度,评价指标

损失函数(loss)被用来用来衡量预测值和标记值之间的差距, 并用于通过反向传播来计算相关梯度, 以便调整优化神经网络中的权重.

* 损失函数有几个相关联的概念:目标函数,代价函数.
  + 代价函数就是损失函数, 衡量的是不一致性.
  + 目标函数是最终需要优化的函数，
    - 其中包括经验损失和结构损失。obj = loss+Ω.
    - 经验损失(loss)就是损失函数或者代价函数。
    - 结构损失(Ω) 就是正则项之类的来控制模型复杂程度的函数。

准确度(accuracy)

* 在分类问题中, 准确度=正确预测数量/总预测数量
* 此指标的局限在于:
  + 离散性
  + 过于简单
    - 正确预测量=TP+TN, 对分布不均衡的案例不友好

评价指标(metrics)

* metrics和loss类似, 但不用于模型训练,只是提供另一个角度的指标.

验证集(validation set)评价

* 如果训练方案中包含了验证集, 一轮结束后会提供验证集上的结果. (其数值的命名通常以val\_开头).

### 案例解析

以下是RNNoise的一轮训练记录,

* 样本数是250, 其中测试集和验证集分别是225/25, 批次大小为32.
* 其中msse是metrics. 在model.compile函数的参数列表中指定的.
* 模型的输出包括denoise\_output和vad\_output两部分
* 基于keras

Train...

Train on 225 samples, validate on 25 samples

Epoch 1/120

32/225 [===>..........................] - ETA: 1:39 - loss: 16.1515 - denoise\_output\_loss: 1.2328 - vad\_output\_loss: 7.6470 - denoise\_output\_msse: 0.2221

64/225 [=======>......................] - ETA: 59s - loss: 15.0375 - denoise\_output\_loss: 1.1134 - vad\_output\_loss: 7.8057 - denoise\_output\_msse: 0.2074

96/225 [===========>..................] - ETA: 41s - loss: 14.3796 - denoise\_output\_loss: 1.0548 - vad\_output\_loss: 7.6618 - denoise\_output\_msse: 0.2001

128/225 [================>.............] - ETA: 28s - loss: 13.8054 - denoise\_output\_loss: 0.9998 - vad\_output\_loss: 7.6138 - denoise\_output\_msse: 0.1936

160/225 [====================>.........] - ETA: 18s - loss: 13.4314 - denoise\_output\_loss: 0.9646 - vad\_output\_loss: 7.5697 - denoise\_output\_msse: 0.1917

192/225 [========================>.....] - ETA: 9s - loss: 12.9737 - denoise\_output\_loss: 0.9214 - vad\_output\_loss: 7.5179 - denoise\_output\_msse: 0.1873

224/225 [============================>.] - ETA: 0s - loss: 12.4702 - denoise\_output\_loss: 0.8729 - vad\_output\_loss: 7.4824 - denoise\_output\_msse: 0.1813

225/225 [==============================] - 66s 293ms/step

- loss: 12.4559 - denoise\_output\_loss: 0.8380 - vad\_output\_loss: 7.3704 - denoise\_output\_msse: 0.1812 - vad\_output\_msse: 0.3762

- val\_loss: 9.4757 - val\_denoise\_output\_loss: 0.5864 - val\_vad\_output\_loss: 7.2218 - val\_denoise\_output\_msse: 0.1504 - val\_vad\_output\_msse: 0.3534

从中我们可以看到:

* 在每个批次都输出了loss和msse(metrics)
* 本轮结束的时候, 汇总了各批次的数据, 并在验证集上计算了loss和msse

## 学习率和优化算法

学习率 (learning rate)，用于控制模型的学习进度.

* 在反向传播算法中, η即为步长: ωn←ωn−η∂L/∂ωn

优化算法, 被用于不断迭代模型参数以降低模型损失函数的值.

* 梯度下降是目前神经网络中使用最为广泛的优化算法之一。
* 为了弥补朴素梯度下降的种种缺陷，研究者们发明了一系列变种算法，从最初的 SGD (随机梯度下降) 逐步演进到 NAdam。
* Adam 是其中综合性能出色的一种算法，自适应学习率的引入使得损失函数可以快速收敛。然而，实践证明，虽然在训练早期 Adam 拥有出色的收敛速度，使用其训练的模型的最终泛化能力却并不如使用朴素 SGD 训练的好（体现在 Adam 训练的模型最终收敛时的 test error 更大）。

## 过拟合现象和应对

过拟合: 当模型的训练误差远小于它在测试数据集上的误差时, 我们称之为过拟合.

### 权重衰减和L2范数正则化

权重衰减等价于L2范数正则化（regularization）, 是应对应对过拟合问题的常用方法的常用方法.

从损失函数的角度来看, L2范数正则化, 会给loss加上L2范数惩罚项(权重每个元素的平方和 \* 一个正的常数)。

Loss += λ||w||2/2n

从权重优化的角度来看, L2范数正则化, 会先将权重缩小一点，再减去原梯度

w = (1- ηλ/|B|)w - ∆

### Dropout

### Early Stopping

3. RNNoise的

超参数列表

模型构造

多层神经网络

模型训练依据和策略

数据组织和整理