Intro to Machine Learning & Deep Learning

Karrol Zhang

(Wuhan University, School of Information Management)

Deep Learning历史

- 1958年 感知机 Perceptron（线性模型）
- 1969年 感知机存在限制，感知机只学会了一些与图片本质不相关的特征来区分图片内容，例如区分卡车和单车，它学会的可能是照片光线不一样
- 1980s Multi-layer perceptron 多层感知机，和今天的深度学习没有显著差别
- 1986 Backpropagation 反向传播, 通常超过3 layer的模型就得不到好的结果
- 1989 一个隐藏层就够好了，可以模拟任何的函数，为什么要深度学习？
- 2006 RBM initialization（突破），当时许多人说20世纪DL和以往的Multi-layer perceptron有什么区别，主要看梯度下降时有没有使用RBM initialization设置初始值（但实际上没有多少作用，但因为非常复杂，引起了学界重新对DL开展研究）

- 2009 GPU可以用于加速计算，让DL快速训练
- 2011 DL在语音识别中流行
- 2012 DL在图片识别中流行

概念梳理

机器学习

- 让机器具备找一个函数（Function）的能力就是Machine Learning
- 这个函数非常复杂，通常人类无法手写得出，需要让机器进行自动寻找，这件事称之为ML
- 从机器学习到的function来看，机器学习的类型
  - 回归(Regression)：函数输出是一个标量（scalar）数值
  - 分类(Classification)：在给定的预定选项中，替人类做出正确的选择，选项可以多个
  - 结构学习(Structure Learning)：除了输出数值、选择选项这种简单的行动，还要让机器学会创造东西，即让机器输出具有结构的东西，例如一篇文章，一段旋律

```
#图像识别
f(image)="cat"
#语音识别
f(voice)="How are u"
#AlphaGo 下棋机器人，下围棋就是一个包含19*19个选项的分类
f(棋局)="5-5"(next move)
```

模型 **model**

带有未知参数的函数，例如 $y = b + wx_1$

b(bias)和w(weight)是未知参数，需要从数据中学习得到。

Hyper-parameter & Parameter

- parameter是模型参数，需要learning from data，是机器学习出来的，用θ代表

关于模型中的未知参数，用θ来统称。θ是一维向量$\theta = \begin{vmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{vmatrix}$，该向量由模型中所有的未知参数拼接而成

- Hyper-parameter是非模型参数，由人设定的。常见的有：learning_rate，迭代次数，layer数量，隐含层的神经元数量等

损失函数 **Loss**

Loss是一个自变量为model的parameter的函数，可以写成$L(\theta)$。

损失函数通常通过预测值(y)与真实值(label $\hat{y}$)的差距来衡量模型的好坏，差距的衡量有多种方式。

示例：model $y = b + wx_1$

L(b,w)代表了预设模型未知参数θ取值为(b,w)时，这个模型有多好

Error surface

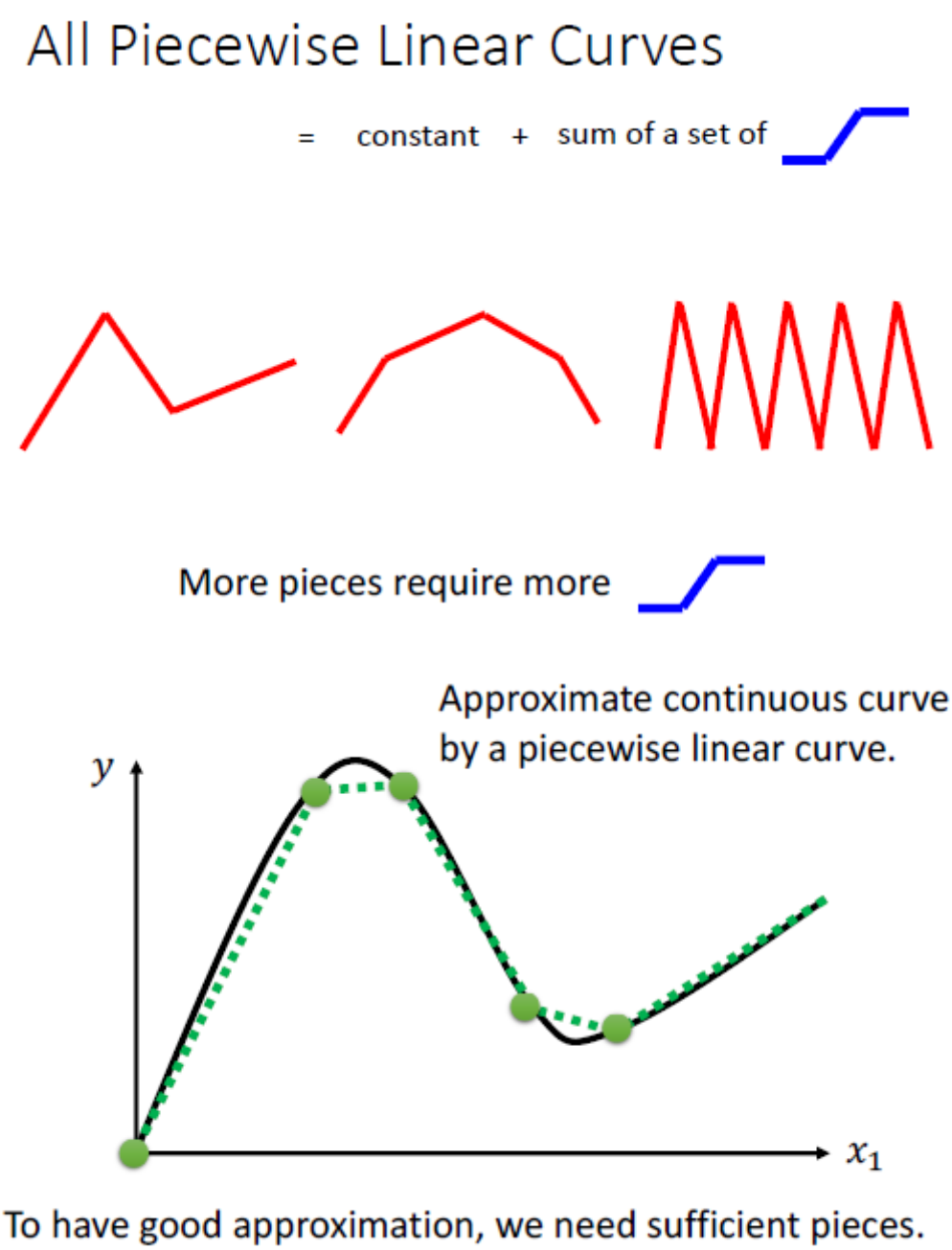
(b,w)各种取值下的Loss函数等高线图，颜色的深浅代表了损失函数值的大小，等高线上的损失相同

Model Bias

无法模拟真实现状的原因是model本身的限制，这个model形式本身与真实存在的model相差甚远，不管parameter取何值，都不会拟合出真实model的走势。

Piecewise Linear Curve

一类函数，这类函数由多段线段组成。并且这类函数可以通过(常数+n个激活函数)表示出来，或者说是逼近表示出来（任何一条曲线，只要取的点够多，就可以通过线段逼近表示）



Update & Epoch

- update动作指每更新一次parameter
- epoch动作指把所有的batch都看过一遍，batch是数据分组，通常把N笔资料（数据总体）分成多个组(batch)来进行梯度下降，加快运算速度

激活函数 **Activation function**

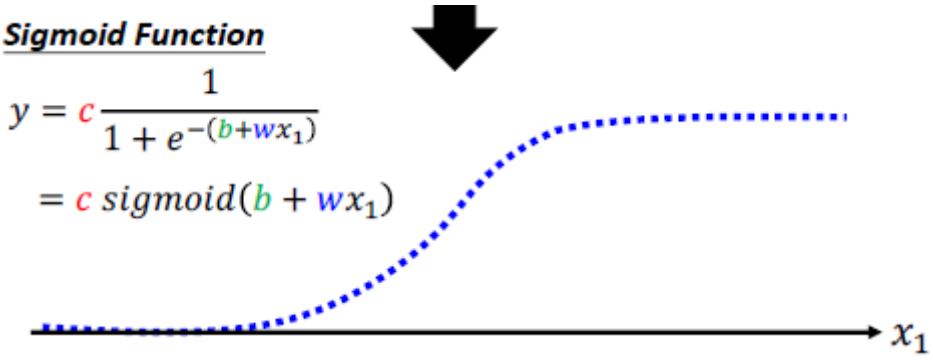
- 不论是relu还是sigmoid，这些用来逼近机器待学习的真实function的函数称为激活函数
- 多个激活函数能够弹性组合，从而实现逼近任意真实函数的能力

sigmoid 函数

一类函数，S型函数，函数曲线型如S。

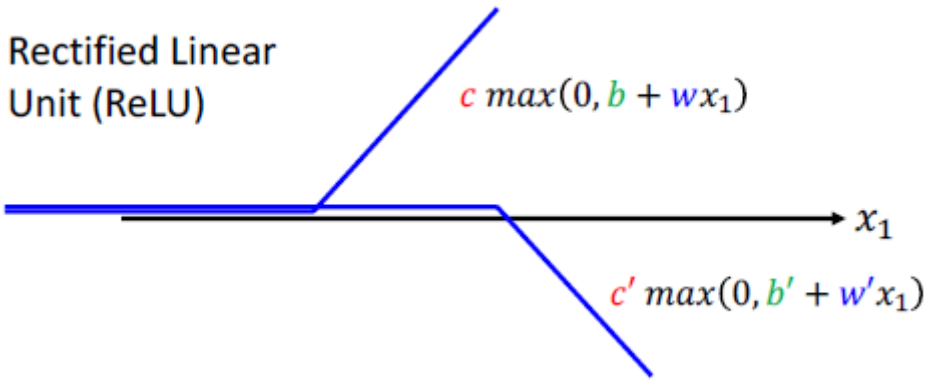
- $x_1 \to \infty$时，$csigmoid \to c, x_1 \to -\infty, csigmoid \to 0$
- 对于csigmoid函数，w改变的函数斜率，b改变函数左右位置，实现平移，c改变函数收敛高度

$$y = c\frac{1}{1+e^{b+wx_1}} = csigmoid(b+wx_1)$$

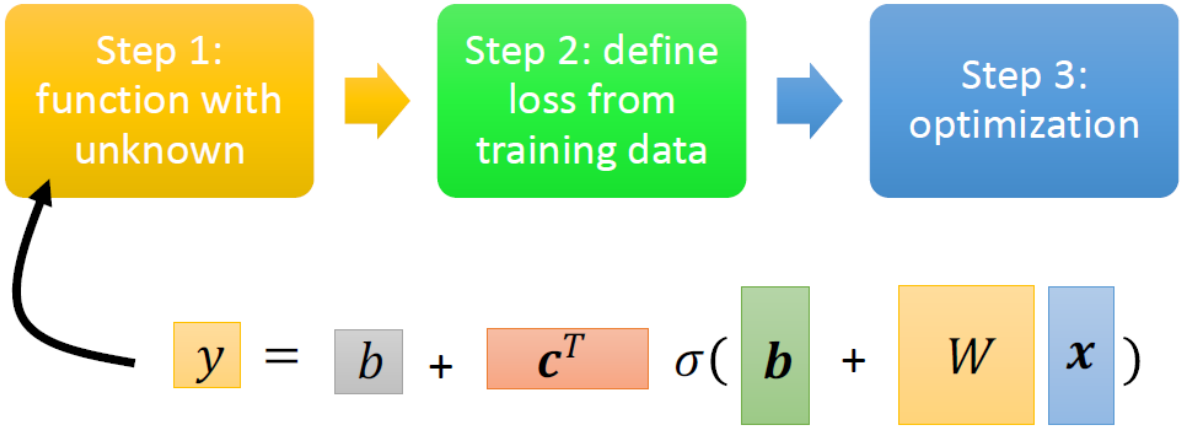


Relu函数

一类函数，函数由一条水平线加一个转折线



机器学习训练过程(General Steps)



step1 基于领域知识，设定关于自变量的可能函数模型

定义带有未知参数的piecewise函数，模型示例：

- 线性回归model：$y = b + wx_1$
- 深度学习model：
  $y = b + \sum_i \mathbf{c_i} sigmoid(\mathbf{b_i} + \sum_j w_{ij} x_j) = b + \mathbf{c^T}\sigma(\mathbf{r}) = b + \mathbf{c^T}\sigma(\mathbf{b} + \mathbf{Wx})$ [1]

在机器学习的解决方案中，模型使用哪一个形式是猜测的，需要有领域知识。

通过一个模型预测效果可以增强研究者对数据和问题的了解，从而重新选择和修改model，找到合适的model。

model不合适会产生较为严重的model bias。

step2 定义损失函数**Loss**

Loss人为设定，其表达可以多种多样，但要能贴近实际，能够衡量预测失误的风险。

常见的衡量模型好坏的损失函数的方法：

- MAE 期望绝对值误差 $e = |y - \hat{y}|$，$L = \frac{1}{N}\sum_n e_n$
- MSE 期望平方误差 $e = (y - \hat{y})^2$，$L = \frac{1}{N}\sum_n e_n$
- 概率误差损失函数计算方法：if y and $\hat{y}$ are both probability distributions →Cross-entropy

step3 最优化 **Optimization & Gradient Descent**

通过数据学习model中未知参数取何值能让model最优的过程，即找到一个最优化的θ，让$L(\theta)$最小。

$L(\theta)$取最小值时，$\theta = \theta^*$:

$$\theta^* = arg\min_{\theta} L(\theta)$$

最常使用最优化方法为梯度下降搜寻法（Gradient Descent），主要缺点是容易陷入局部最优（local minima），但局部最优是一个假问题？

Gradient Descent步骤如下：

- 设定初始的搜索起点θ^0，初始起点可以是随机选取的
- 计算损失函数$L(\theta)$的梯度，即微分（斜率），确定搜索方向。斜率像一个环顾四周的地势高低，并选择向低地势踏步

$$\frac{\partial L}{\partial \theta}\Big|_{\theta=\theta^0}$$

- 确定搜索步长

步长 = 学习率 × 斜率

学习率（η:learning_rate）是一个hyper-parameter，人为设定的常数

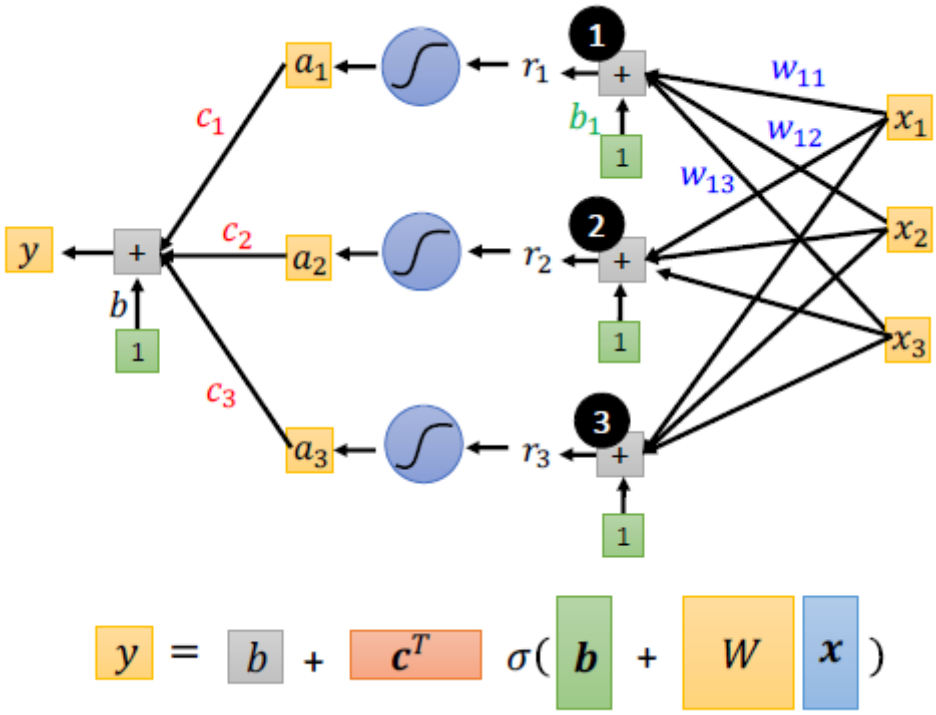
斜率大则大踏步，斜率小则小踏步，更新θ值

$$\theta^1 = \theta^0 - \eta\frac{\partial L}{\partial \theta}\Big|_{\theta=\theta^0}$$

- 判定是否结束搜索：
  - 失去搜索耐心，最多更新多少次θ，计算多少次梯度？由迭代次数决定，超出就停止搜索。
  - 斜率变0，判定找到最优，停止搜索。

step4 model让人满意了嘛？

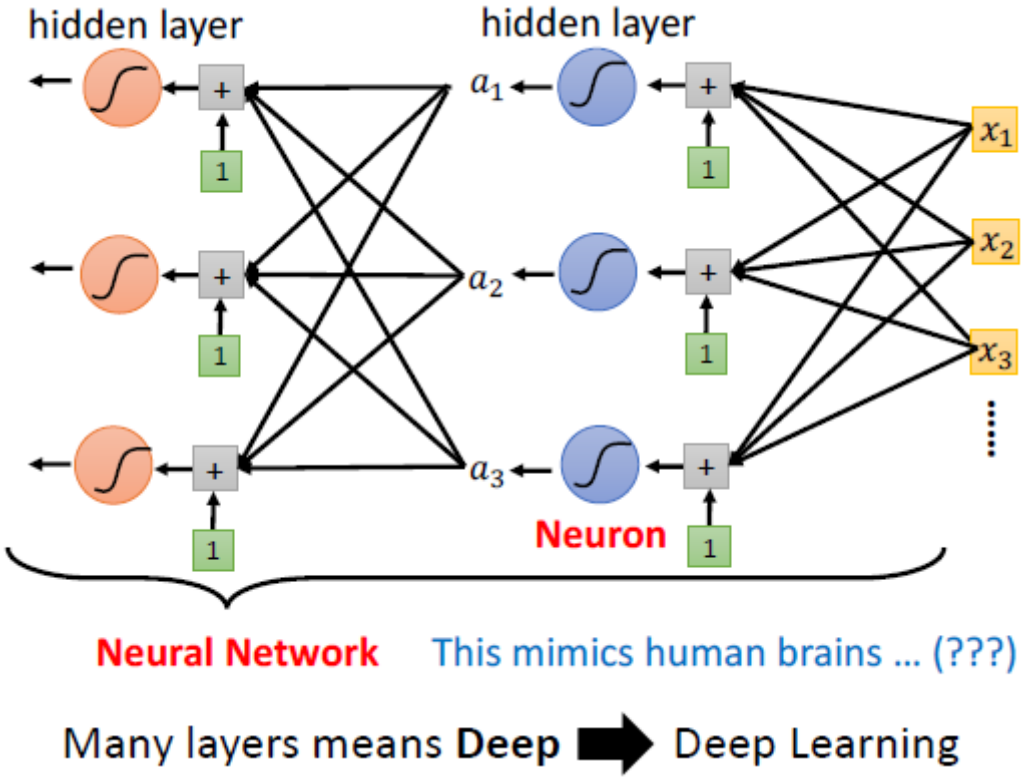
- model训练的时候Loss很小，但model用于预测时Loss很大？——过拟合情况？
- 特征（自变量）的选取是否足够？改变特征是否会带来训练和预测时Loss的大幅度变化？
- 模型的结构调整是否会带来模型的进步？



深度学习的训练过程同机器学习一致，在step1设定函数模型这里比较特殊。

上图显示了一个具有一层隐含层的神经网络结构：

- 向量**x**为输入层，输入做了线性变换$(\mathbf{b}+\mathbf{W}\mathbf{x})$
- 神经元（Neuron）由单个激活函数构成，接收输入(r_1)并产生输出(a_1)。图中只有一层隐含层，一层隐含层中包含3个神经元。



---

1. y=b+...的b是一个常数，不加粗，出现在∑中的b是向量，向量加粗 ↩