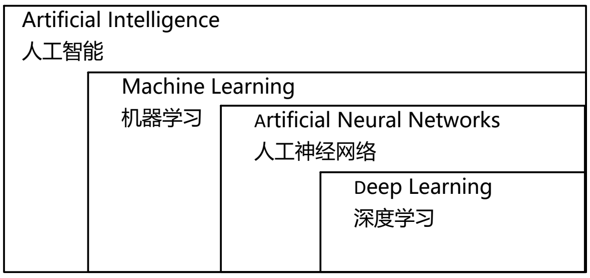
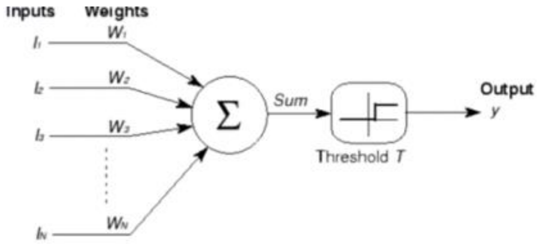
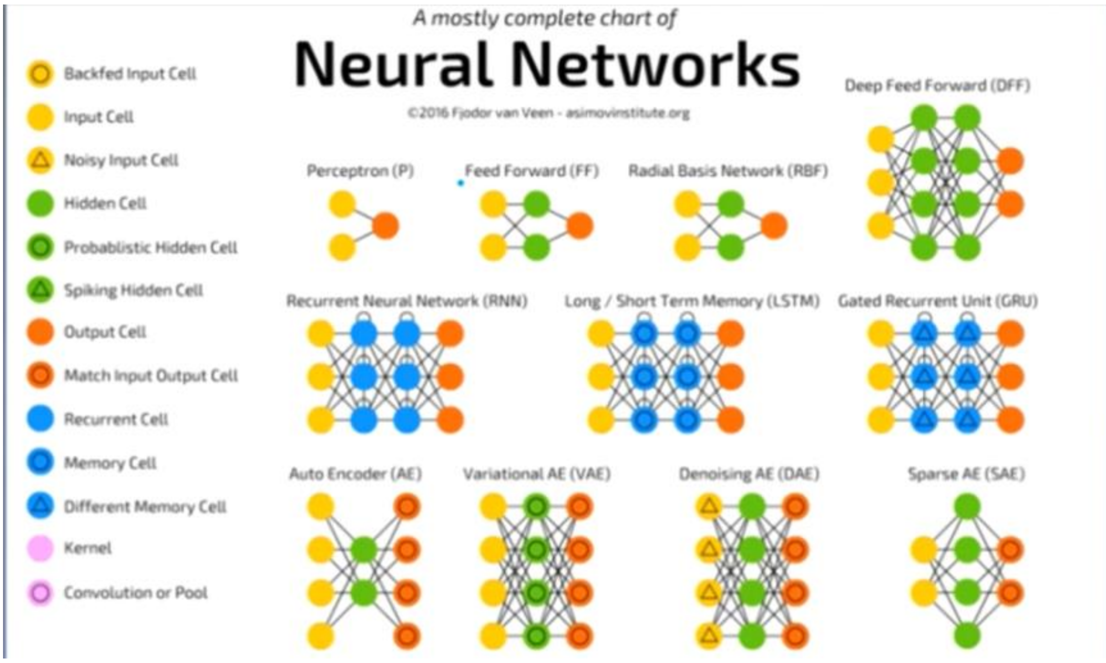
Basic knowledge of Neural Network

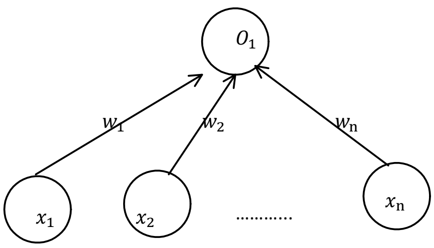
2023年7月31日 13:01





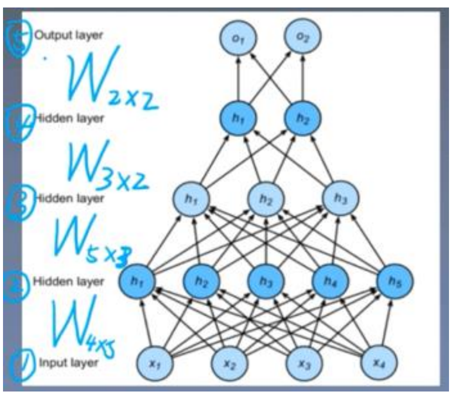


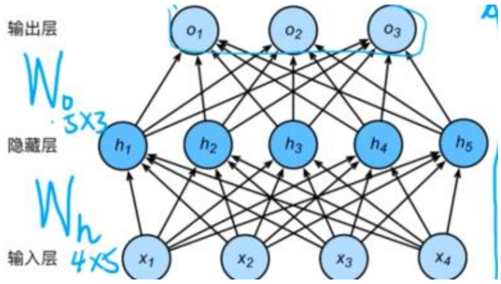
- 人工神经元：1958年Rosenblatt提出了Perception（感知器），致命缺点被Minsky在1969年证明，无法解决异或问题。



$$o = \sigma(< wx > +b)$$
$$\sigma(x) = \begin{cases} 1 & if\ x > 0 \\ 0 & o.w. \end{cases}$$

- 多层感知机（Multi Layer Perceptron, MLP)



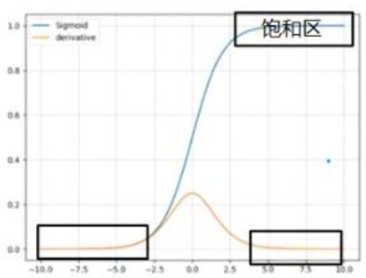


$\sigma(X_{1*4} \cdot W_h) = H_{1*5}$
$\sigma(H_{1*5} \cdot W_o) = O_{1*3}$

- 激活函数
  a. 让多层感知机成为真正的多层，否则等价于一层
  b. 引入非线性，使网络可以逼近任何非线性函数（万能逼近定理，universal approximator)

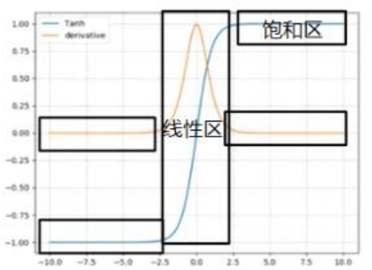
$$H = XW_h + b_h$$
$$O = HW_o + b_o = (XW_h + b_h)W_o + b_o = X\underbrace{W_hW_o}_{W} + \underbrace{b_hW_o + b_o}_{b} \rightarrow O = XW + b$$

  - 激活函数需要满足的性质：
    i. 连续并可导（允许少数点上不可导），便于利用数值优化的方法来学习网络参数
    ii. 激活函数及其导数要尽可能的简单，有利于提高运算效率
    iii. 激活函数的导函数的值域要在合适区间内，不能太大也不能太小，否则会影响训练效率及稳定性
  - 常见的激活函数：
    i. Sigmoid(S型）：
    弊端：如果神经元大量落入饱和区，梯度几乎为0很难继续优化



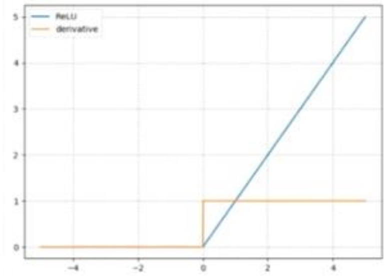
$$g(z) = \frac{1}{1+e^{-z}}$$
$$g'(z) = g(z) * (1 - g(z))$$

    ii. Tanh(双曲正切)
    弊端和Sigmoid相似



$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$g'(z) = 1 - (g(z))^2$$

    iii. ReLU(修正线性单元)
    非饱和函数



$$Relu = max(0, x)$$

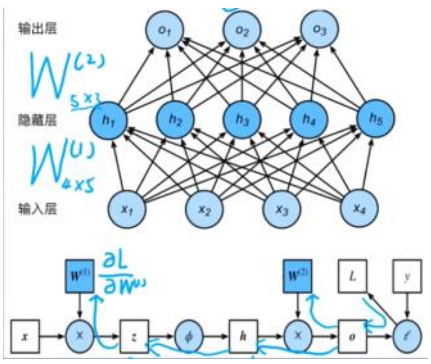
$$g^{'}(z) = \begin{cases} 1 & \text{if } z > 0 \\ undefined & \text{if } z = 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- 反向传播（back propagation)

  前向传播 : 输入层数据开始从前向后，数据逐步传递至输出层

  反向传播：损失函数开始从后向前，梯度逐步传递到第一层，用于权重更新，使网络输出更接近真实标签，原理是链式求导

  损失函数：衡量模型输出与真实标签之间的差距 $Loss = f(\hat{y}, y)$



$$Z = X \cdot W^{(1)} \cdots (1)$$
$$h = \Phi(Z) \quad \cdots (2)$$
$$O = h \cdot W^{(2)} \cdots (3)$$

$$\frac{\partial L}{\partial W^{(2)}} = prod\left(\frac{\partial L}{\partial O}, \frac{\partial O}{\partial W^{(2)}}\right) = \frac{\partial L}{\partial O} \cdot h^T$$
$$\frac{\partial L}{\partial h} = prod\left(\frac{\partial L}{\partial O}, \frac{\partial O}{\partial h}\right) = W^{(2)T} \cdot \frac{\partial L}{\partial O}$$
$$\frac{\partial L}{\partial Z} = prod\left(\frac{\partial L}{\partial O}, \frac{\partial O}{\partial h}, \frac{\partial h}{\partial Z}\right) = \frac{\partial L}{\partial h} \odot \Phi'(Z)$$
$$\frac{\partial L}{\partial W^{(1)}} = prod\left(\frac{\partial L}{\partial O}, \frac{\partial O}{\partial h}, \frac{\partial h}{\partial Z}, \frac{\partial Z}{\partial W^{(1)}}\right) = \frac{\partial L}{\partial Z} \cdot X^T$$

$prod(x, y)$表示x, y根据具体情况做相应的变换然后相乘

Gradient Decent : 权值沿着梯度负方向更新，使函数值减小

导数：函数在指定坐标轴上的变化率

方向导数：指定方向上的变化率

梯度：一个向量，方向为方向导数取得最大值的方向

Learning Rate : 控制更新步长

沿梯度负方向更新：$w_{i+1} = w_i - \varepsilon \cdot g(w_i)$，$\varepsilon < 1$

- 损失函数Loss Function:衡量模型输出与真实标签之间的差距
  - Loss Function 单样本：$Loss = f(\hat{y}, y)$
  - Cost Function 总体：$\text{Cost} = \frac{1}{N}\sum_i^N f(\hat{y}_i, y_i)$
  - Objective Function：$Obj = \text{Cost} + Regularization\ Term$*正则项控制模型复杂程度，减轻模型复杂程度*

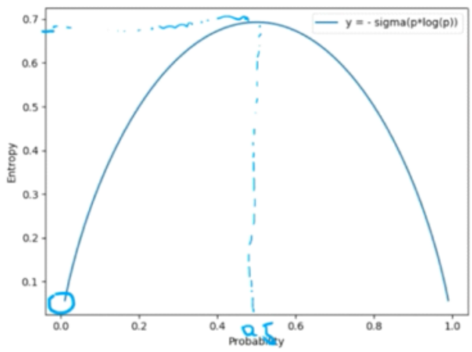
两种常见的Loss Function:

1） MSE(Mean Squared Error):输出与标签之差的平方的均值，一般用于回归任务 $MSE = \frac{1}{N}\sum_{i=1}^N (\hat{y}_i - y_i)^2$

2） CE(Cross Entropy交叉熵）：源自信息论，用于衡量两个分布的差异，常用于分类任务

$$H(p, q) = -\sum_{i=1}^{n} p(x_i)\log q(x_i)，p：真实，\quad q：模型$$

信息熵：描述信息的不确定度，越大越不确定，0.5的时候是最不确定的



自信息： $I(x) = -\log p(x)$，$p(x)$是某事件发生的概率

信息熵 = 所有可能取值的信息量的期望， $H（x）= E_{x\sim p}[I(x)] = -E[\log p(x)] = -\sum_{i=1}^{N} p_i \log(p_i)$ *是固定的，相当于一个常数*

相对熵：又称K-L散度，衡量两个分布之间的差异。

$$D_{KL}(P||Q) = E_{x\sim p}\left[\log\frac{P(x)}{Q(x)}\right] = E_{x\sim p}[\log P(x) - \log Q(x)]$$

$$= \sum_{i=1}^{N} P(x_i)\left(\log P(x_i) - \log Q(x_i)\right)$$

$\Rightarrow$ $H(p,q) = H(p) + (D_{KL}|p|q)$
即：交叉熵 = 信息熵（常数）+ 相对熵
优化交叉熵 ⇔优化相对熵

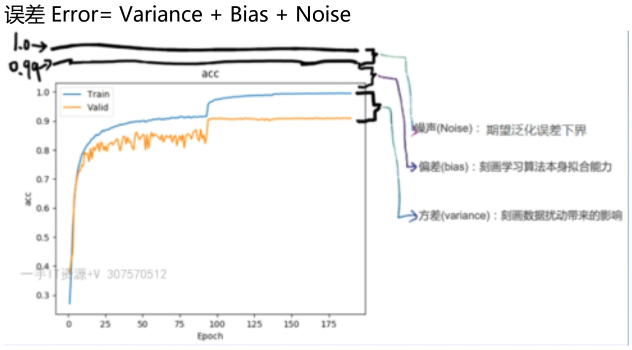
交叉熵：$H(p,q) = -\sum_{i=1}^{n} p(x_i)\log q(x_i)$，$p$：*真实(0 or 1)*， q：*模型输出(probability)*

由于交叉熵的输出q_i必须是一个概率（1.值为非负，2.概率之和为1）⇒ Softmax函数：将数据变换到符合概率分布的形式

i） Softmax函数：$y_i = S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, i = 1,2,\ldots,C$ *指数实现非负，除以指数之和实现和为1*

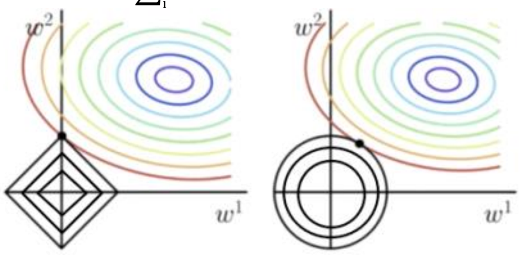
- 权值初始化Initialization: 训练前对权值参数赋值，良好的权值初始化有利于模型训练
  - 简单错误的方法：初始化全部为0， 会使得神经元全部退化为1个
  1) 随机初始化方法：
     I. 高斯分布随机初始化：从高斯分布中随机采样，对权重进行赋值，权重过大会落入饱和区导致梯度消失。比如：$N\sim(0\ \text{mean}，0.01\ \text{sd})$ 3σ准则：数值分布在$(\mu-3\sigma，\mu+3\sigma)$中的概率为99.73%
     II. 自适应标准差：自适应方法随机分布中的标注差
        i. Xavier初始化：$U\left(-\sqrt{\frac{6}{a+b}}，\sqrt{\frac{6}{a+b}}\right)$ a是输入神经元的个数，b是输出神经元的个数
        ii. Kaiming初始化（MSRA）
- Regularizaton正则化方法：减少方差的策略⇒减轻过拟合 Overfit: 方差过大，在训练集上表现良好，在测试集上表现不好
  - 偏差bias：度量了学习算法的期望预测与真实结果的偏离程度，即刻画了学习算法本身的拟合能力
  - 方差Variance: 度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响
  - 噪声noise：表达了在当前任务上任何学习算法所能达到的期望泛化误差的下限(不管是谁来完成这个任务，模型只能达到如此，不可能完美)

  误差 Error= Variance + Bias + Noise



常见的正则化方法：

1) L1 Regularization Term: $\sum_{i}^{N}|w_i|$ *权值退化（在w_2上时，w_1退为0）*

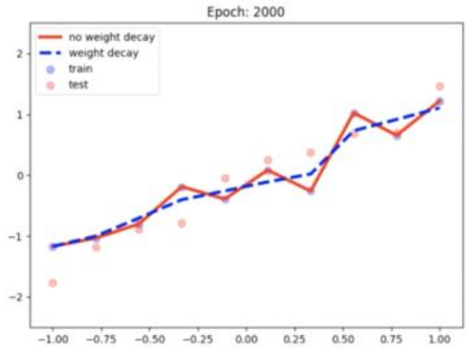


2) L2 Regularization Term: $\sum_{i}^{N} w_i^2$ *权值衰减 weight decay：减少对权值的依赖性*

$$Obj = Loss + \frac{\lambda}{2} * \sum_{i}^{N} w_i^2$$

无正则项：$w_{i+1} = w_i - \frac{\partial Obj}{\partial w_i} = w_i - \frac{\partial Loss}{\partial w_i}$

有正则项：$w_{i+1} = w_i - \left(\frac{\partial Loss}{\partial w_i} + \lambda * w_i\right) = w_i(1-\lambda) - \frac{\partial Loss}{\partial w_i}$



3) 随机失活 dropout:

pro: 避免过度依赖某个神经元，实现减轻过拟合

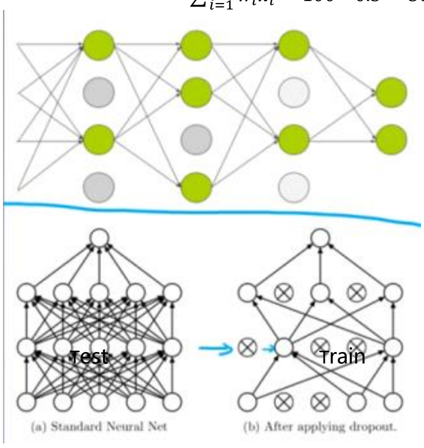
随机：dropout probability (e.g. p=0.5 就是失活一半)

失活：weight = 0

注意事项：训练和测试两个阶段的数据尺度变化

测试时，神经元输出值需要*p $\quad \sum_{i=1}^{100} w_i x_i = 100 * \mathrm{p} = 50 \rightarrow$ 测试

$\sum_{i=1}^{50} w_i x_i = 100 * 0.5 = 50 \rightarrow$ 训练



4) 其他正则化方法：Batch normalization (BN), Layer Normalization(LN),
Instance Normalization（IN)，Group Normalization(GN)