真-神经网络入门篇

作者: 骰子AI

2022-3

大纲

- 代码地址： rexrex9/basic_neural_networks_pytorch: 最入门的神经网络示例代码,pytorch版 (github.com)

1. 机器学习最简单的做法 (3 – 4)
2. 梯度下降(5 – 7)
3. 批量梯度下降(8 – 9)
4. 线性回归PyTorch版(10)
5. 波士顿房价预测(11 – 12)
6. 回归任务与分类任务 (13)
7. 多分类与二分类 (14)
8. 鸢尾花分类预测(15 – 16)
9. 神经网络的基础形态-多层感知机MLP (17 – 21)
10. 鸢尾花预测MLP版 (22)
11. 前向传播与后向传播 (23 – 24)
12. 欠拟合与过拟合 (25 – 26)

1.机器学习最简单的做法

1. 机器学习最简单的理解：给定一堆数据，从数据中寻找规律建立模型从而进行预测。
2. 大家可以思考一下下表“？”的值是多少？

X	1	2	3	4	5
Y	2	4	6	8	？

3. 如果是下表呢？

x1	3.4	2.5	-1.2	0.4	-2.7	2.1
x2	1.1	-1.2	3.2	2.1	2.5	0.6
Y	15.1	6.4	12.2	12.1	7.1	？

1.机器学习最简单的做法

第一步，设计一个公式，如下（在机器学习中,该公式被称为目标函数）：

$$\bar{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

第二步，随机初始化所有的w和b。

第三步，将一组数据中所有的x带入式子计算得到$\bar{y}$。

第四步，将$\hat{y}$与第三步中对应那一组数据中的y计算距离，公式如下：

$$dis = \bar{y} - y$$

第五步，对w和b进行更新，过程如下（其中lr指learning rate学习率，用来调整更新幅度）：

$$w_{new} = w_{old} - lr \times dis \times x$$

$$b_{new} = b_{old} - lr \times dis$$

第六步，重复三到五步的操作，直到指定的次数，或者y与$\hat{y}$的差距小于一定阈值。

2.梯度下降

- 梯度下降指的是将模型参数沿着梯度负方向迭代更新从而使损失函数逼近最小值。其中梯度指的是损失函数对于各个模型参数的偏导组成的向量。

- 假设损失函数表达为$f(\mathrm{x},y,z)$,则该损失函数对应各个参数的梯度可表达为：$\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y},\frac{\partial f}{\partial z}\right)$

- 损失函数指的是表达预测值与真实值之间距离的函数，优化损失函数指的就是将损失函数的值尽可能的趋近0，也就是减少预测值与真实值之间的距离。

2.1 目标函数与损失函数

- 目前所涉及的例子属于机器学习中最基础的线性回归问题，线性回归指的是通过数据拟合出一个线性函数表达出变量与标注之间的关系。
- 线性回归目标函数：

$$\bar{y} = \sum_{i=1}^{n} w_i x_i + b$$

- 平方差损失函数：

$$L = (\bar{y} - y)^2$$

2.2各偏导计算过程

- 将$\bar{y} = \sum_{i=1}^{n} w_i x_i + b$代入平方差损失函数得:

$$L = \left(\sum_{i=1}^{n} w_i x_i + b - y \right)^2$$

- 对w_i求偏导得:

$$\frac{\partial L}{\partial w_i} = 2x_i \left(\sum_{i=1}^{n} w_i x_i + b - y \right) \cong x_i(\bar{y} - y)$$

- 对b求偏导得:

$$\frac{\partial L}{\partial b} = 2x_i \left(\sum_{i=1}^{n} w_i x_i + b - y \right) \cong \bar{y} - y$$

3.批量梯度下降-前置知识

1. 矩阵点乘

   - 若: $\mathrm{m1}=\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, \mathrm{m2}=\begin{bmatrix} x & y \\ z & q \\ p & k \end{bmatrix}$。则$m1 \cdot m2=\begin{bmatrix} ax+bz+cp & \mathrm{a}y+bq+ck \\ dx+ez+fp & dy+eq+fk \end{bmatrix}$

2. 张量(Tensor)
   - 由多维数组表示的量
   - 一维张量是向量
   - 二维张量是矩阵

3.批量梯度下降

1. BGD

   批量梯度下降 Batch Gradient Descent，整批样本同时并行进行梯度下降，从而整批地迭代更新模型参数。

2. SGD

   随机梯度下降 Stochastic Gradient Descent，随机采样得到一批样本后，整体进行梯度下降并迭代更新模型参数。

3. MBGD

   小批量梯度下降 Mini-batch Gradient Descent，将整体的样本分成小批量进行多批次的梯度下降迭代更新模型参数。

4. MSGD

   小批量随机梯度下降 Mini-batch Stochastic Gradient Descent，将整体的样本随机打乱后，分成小批量进行多批次的梯度下降迭代更新模型参数。当下简称SGD。

4.线性回归PYTORCH版

```
class Linear_Reg( nn.Module ):
    def __init__( self, n_features ):
        super(Linear_Reg, self).__init__()
        self.linear = nn.Linear(n_features,1,bias=True)

    def forward( self, x ):
        y = self.linear(x)
        y = torch.squeeze( y )
        return y

def train( dates, epochs = 30, batchSize = 24, lr = 0.05 ):
    net = Linear_Reg(len(dates[0])-1) # 初始化线性回归模型
    criterion = torch.nn.MSELoss() #平方差损失函数
    optimizer = torch.optim.SGD( net.parameters(), lr=lr) #随机梯度下降
    for e in range(epochs):
        for dates in DataLoader(dates, batch_size = batchSize, shuffle = True):
            optimizer.zero_grad() #梯度归0
            X = datas[:,:-1] # 获取X
            y = datas[:,-1] # 获取y
            y_pred = net( X ) # 得到预测值y
            loss = criterion(y_pred, y) #将预测的y与真实的y带入损失函数计算损失值
            loss.backward() # 后向传播
            optimizer.step() #更新所有参数
```

5.波士顿房价预测

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
0	0.006	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
1	0.027	0	7.07	0	0.469	6.421	78.9	4.967	2	242	17.8	396.9	9.14	21.6
2	0.027	0	7.07	0	0.469	7.185	61.1	4.967	2	242	17.8	392.8	4.03	34.7
3	0.032	0	2.18	0	0.458	6.998	45.8	6.062	3	222	18.7	394.6	2.94	33.4
4	0.069	0	2.18	0	0.458	7.147	54.2	6.062	3	222	18.7	396.9	5.33	36.2
5	0.03	0	2.18	0	0.458	6.43	58.7	6.062	3	222	18.7	394.1	5.21	28.7
6	0.088	12.5	7.87	0	0.524	6.012	66.6	5.561	5	311	15.2	395.6	12.43	22.9
7	0.145	12.5	7.87	0	0.524	6.172	96.1	5.951	5	311	15.2	396.9	19.15	27.1
8	0.211	12.5	7.87	0	0.524	5.631	100	6.082	5	311	15.2	386.6	29.93	16.5
9	0.17	12.5	7.87	0	0.524	6.004	85.9	6.592	5	311	15.2	386.7	17.1	18.9
10	0.225	12.5	7.87	0	0.524	6.377	94.3	6.347	5	311	15.2	392.5	20.45	15
11	0.117	12.5	7.87	0	0.524	6.009	82.9	6.227	5	311	15.2	396.9	13.27	18.9
12	0.094	12.5	7.87	0	0.524	5.889	39	5.451	5	311	15.2	390.5	15.71	21.7
13	0.63	0	8.14	0	0.538	5.949	61.8	4.708	4	307	21	396.9	8.26	20.4
14	0.638	0	8.14	0	0.538	6.096	84.5	4.462	4	307	21	380	10.26	18.2
15	0.627	0	8.14	0	0.538	5.834	56.5	4.499	4	307	21	395.6	8.47	19.9
16	1.054	0	8.14	0	0.538	5.935	29.3	4.499	4	307	21	386.9	6.58	23.1
17	0.784	0	8.14	0	0.538	5.99	81.7	4.258	4	307	21	386.8	14.67	17.5
18	0.803	0	8.14	0	0.538	5.456	36.6	3.797	4	307	21	289	11.69	20.2
19	0.726	0	8.14	0	0.538	5.727	69.5	3.797	4	307	21	391	11.28	18.2
20	1.252	0	8.14	0	0.538	5.57	98.1	3.798	4	307	21	376.6	21.02	13.6
21	0.852	0	8.14	0	0.538	5.965	89.2	4.012	4	307	21	392.5	13.83	19.6
22	1.232	0	8.14	0	0.538	6.142	91.7	3.977	4	307	21	396.9	18.72	15.2
23	0.988	0	8.14	0	0.538	5.813	100	4.095	4	307	21	394.5	19.88	14.5
24	0.75	0	8.14	0	0.538	5.924	94.1	4.4	4	307	21	394.3	16.3	15.6
25	0.841	0	8.14	0	0.538	5.599	85.7	4.455	4	307	21	303.4	16.51	13.9
26	0.672	0	8.14	0	0.538	5.813	90.3	4.682	4	307	21	376.9	14.81	16.6
27	0.956	0	8.14	0	0.538	6.047	88.8	4.453	4	307	21	306.4	17.28	14.8
28	0.773	0	8.14	0	0.538	6.495	94.4	4.455	4	307	21	387.9	12.8	18.4

CRIM 各镇的人均犯罪率

ZN 划定为25,000平方英尺以上的住宅用地的比例

INDUS 每个镇的非零售商业亩数比例

CHAS 查尔斯河虚拟变量（=1，代表区块与河流相连；否则为0）。

NOX 氮氧化物浓度（每1000万份）。

RM 每个住宅的平均房间数

AGE 1940年以前建造的业主自用单元的比例

DIS 到波士顿五个就业中心的加权距离

RAD 辐射状高速公路的可达性指数

TAX 每10,000美元的全额财产税税率

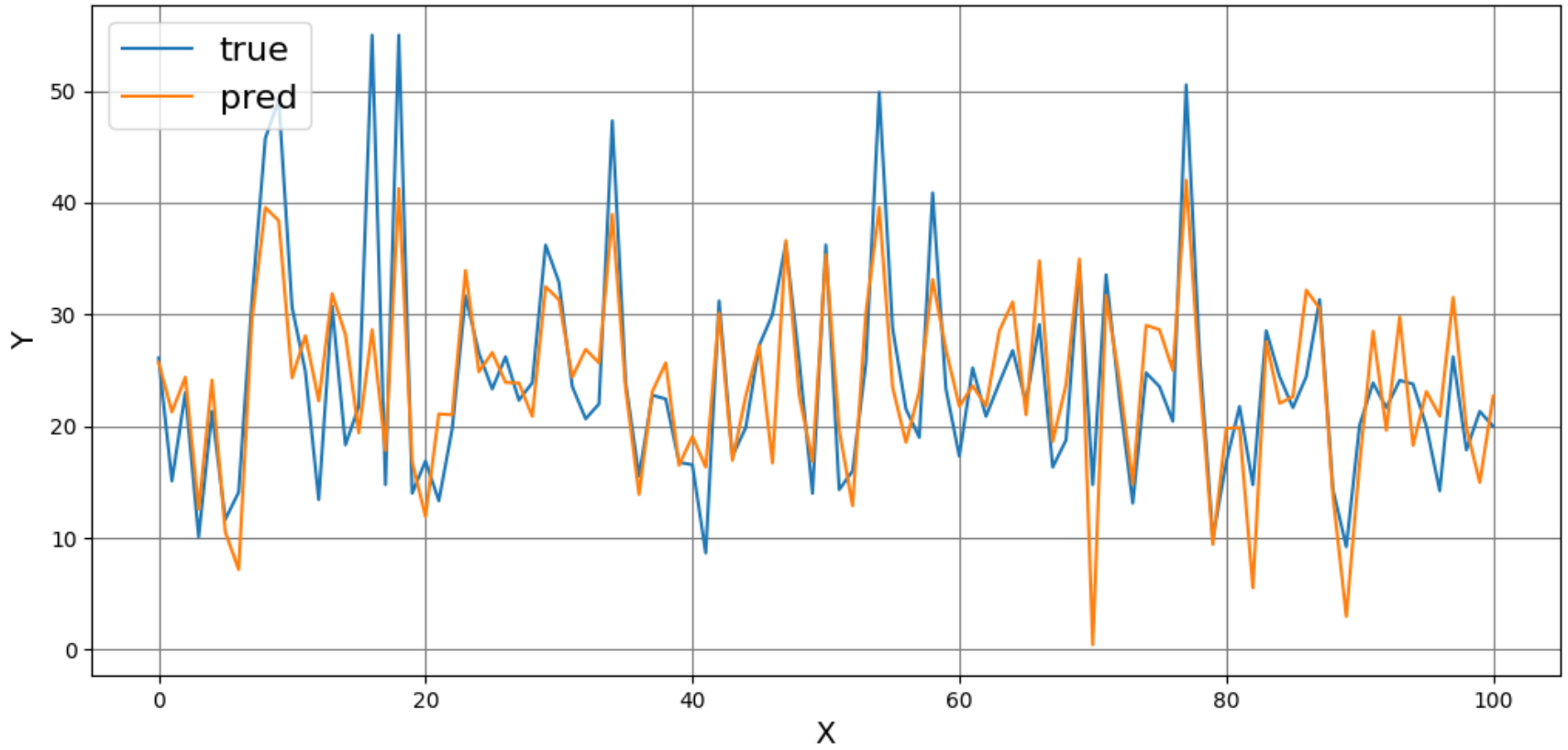
PTRATIO 各镇学生与教师的比例

B 1000(Bk - 0.63)^2 其中Bk是各镇的黑人比例

LSTAT 人口中地位较低的百分比

target 以1000美元为单位的房价

5.波士顿房价预测



6.回归任务与分类任务

- 机器学习可根据预测目标值是否离散分为回归任务与分类任务。

1. 回归任务，是对连续值进行预测，可以理解为预测某个目标为多少数字。
   - 例如：预测房价，预测年龄，预测明天温度。

2. 分类任务，是对离散值进行预测，可以理解为预测某个目标属于什么类别。
   - 例如：花的种类预测，人脸识别，预测明天天气是什么，预测明天是否会下雨。
   - 分类任务可分为多分类任务与二分类任务。

7.多分类与二分类

1. 多分类，通常模型输出的值是一个向量，向量中每一个值对应着每一个类别的概率。最终由Softmax函数归一化，损失函数通常采用交叉熵损失函数。
   - Softmax函数：

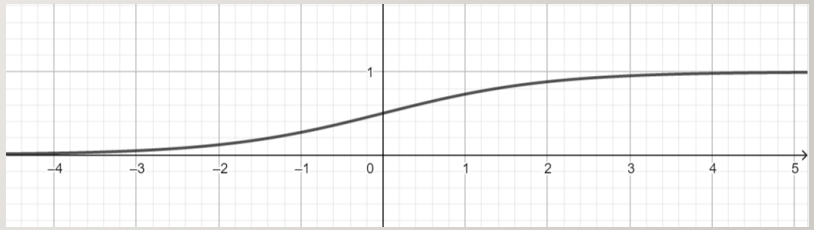
$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

   - 交叉熵损失函数(Cross Entropy Loss)：

$$CELoss = -\sum_{i=1}^{M} y_i \log(p_i) \cong -\log(p_i)$$

2. 二分类，可以看做类别数量为2的多分类预测，但更简单的方式是模型输出一个0~1的标量假设为p,该标量表示属于某一类别的概率，而$1-p$表示为另一个类别的概率。也可看做是一个True or False的布尔预测。若为不二预测，则最终由Sigmoid函数归一化，损失函数通常采用二分类交叉熵损失函数。
   - Sigmoid函数:

$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$



   - 二分类交叉熵损失函数(Binary Cross Entropy Loss):

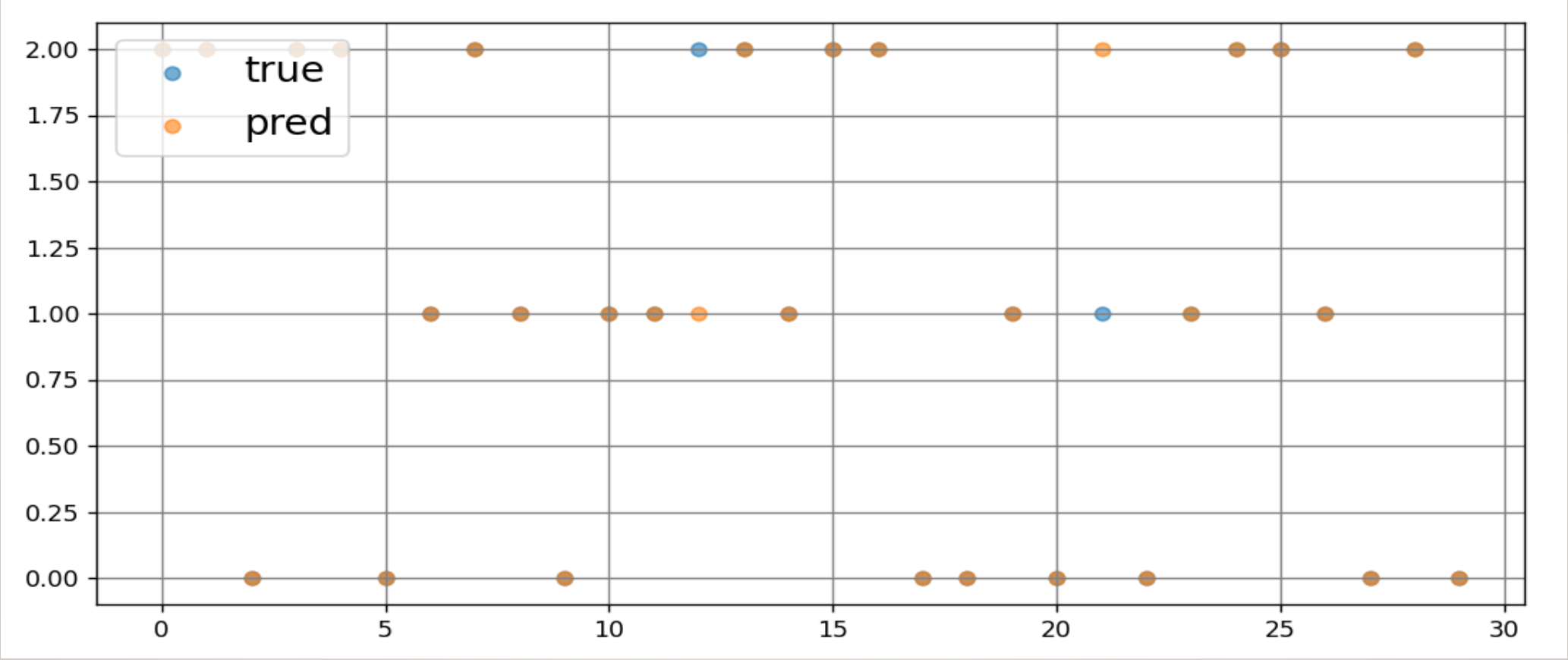
$$BCELoss = -(ylog(p) + (1-y)log(1-p)) \cong \begin{cases} -\log(p), & y = 1 \\ -\log(1-p), & y = 0 \end{cases}$$

8.鸢尾花分类预测

- 鸢尾花数据总共150组，3种类型，4个特征，具体如下：

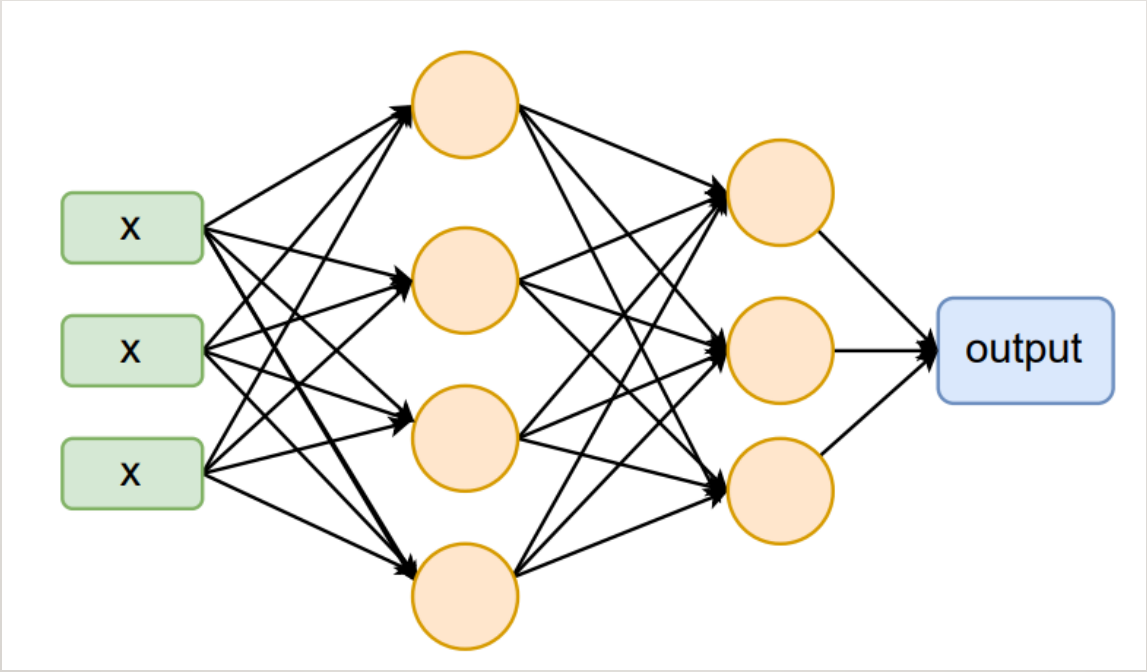
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3	1.4	0.1	0
13	4.3	3	1.1	0.1	0
14	5.8	4	1.2	0.2	0

8.鸢尾花分类预测

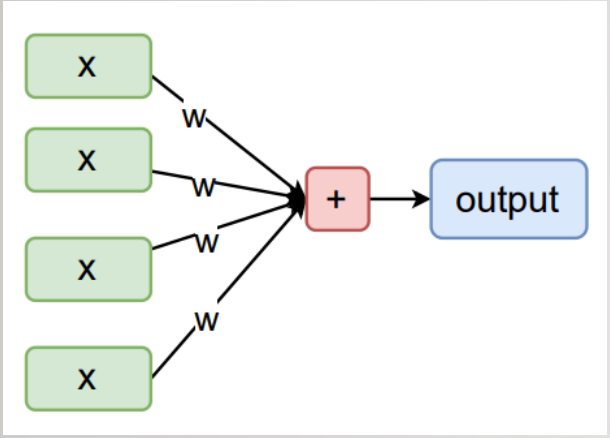


9.神经网络的基础形态-多层感知机MLP

- 多层感知机(Multilayer Perceptron, MLP)，也叫人工神经网络（ANN，Artificial Neural Network)。可简单理解为输入到输出之间包含了若干个隐藏层的结构。

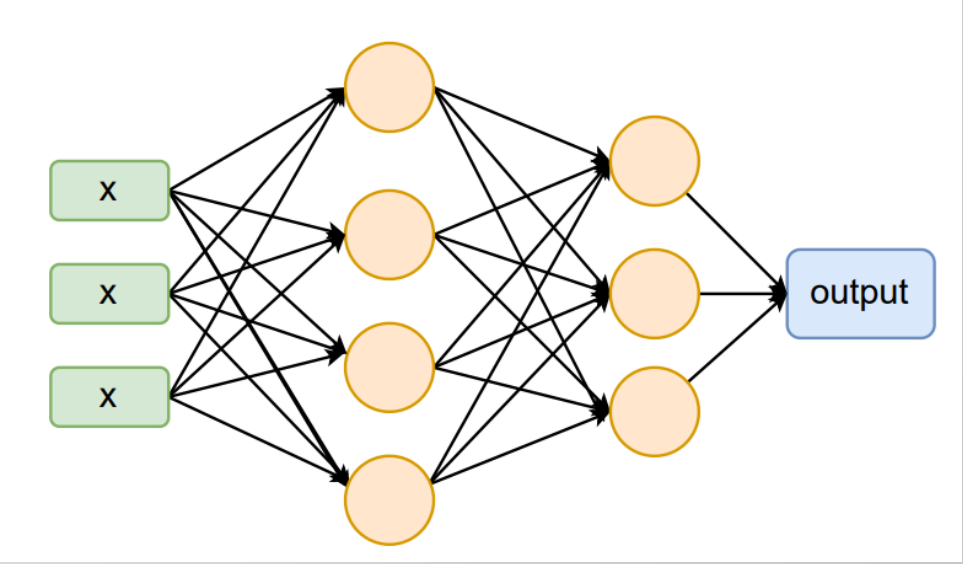


9.神经网络的基础形态-多层感知机MLP

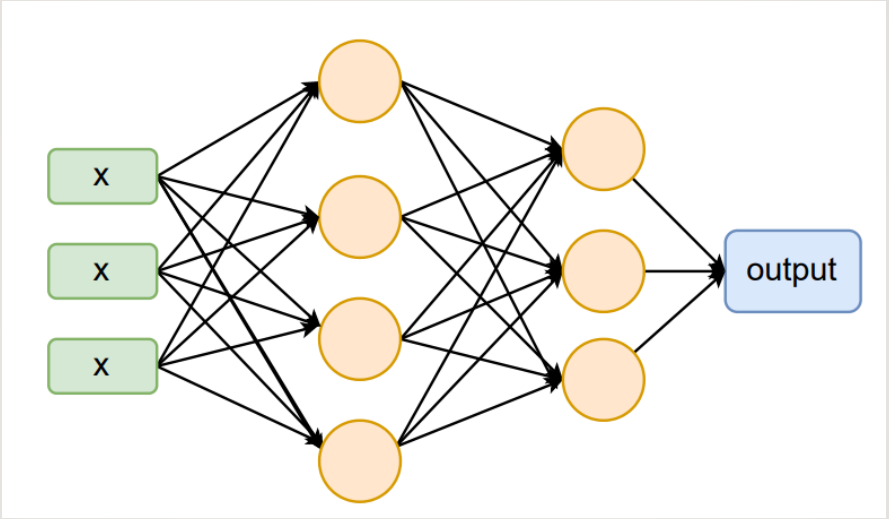


- 仅有输入与输出层的神经网络可理解为$y = wx + b$的单次线性变化。假设输入的x为有n个特征。则可初始化一个$n \times 1$的线性变化矩阵w来进行线性变化。例如$x = \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$，则$w = [w1 \quad w2 \quad w3]$。所以$w \cdot x = [w1x1 + w2x2 + w3x3]$。若有偏执项$b$，则最终的输出$y$仅需在$w \cdot x$的结果后加上$b$即可。

- 右图MLP中间隐藏层的圆圈可被认为是神经元，并没有物理意义，起到的是中间传递信息的作用。
- 假设输入的x为有n个特征，第一层隐藏层的神经元个数为k，则第一次的线性变化矩阵w形状需设定为$n \times k$。例如$x = \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$，$k$为2，则$w = \begin{bmatrix} w1 & w2 & w3 \\ w4 & w5 & w6 \end{bmatrix}$，则$w \cdot x = \begin{bmatrix} w1x1 + w2x2 + w3x3 \\ x1w4 + x2w4 + x3w5 \end{bmatrix}$。



9.神经网络的基础形态-多层感知机MLP

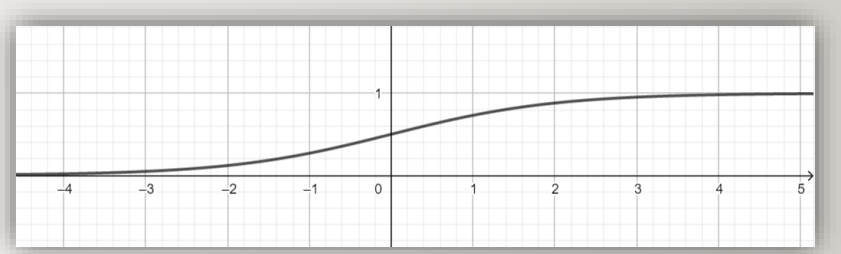


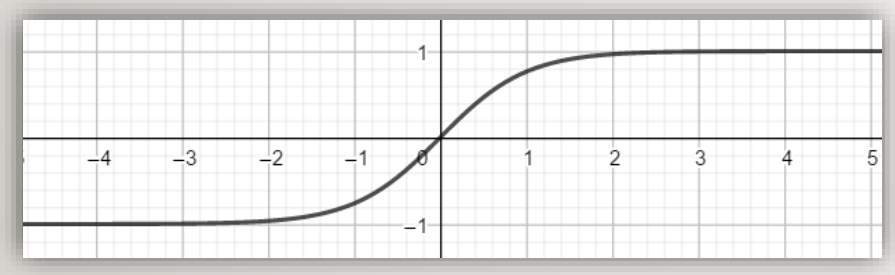
- 再将$w \cdot x$用激活函数计算一下则可被认为是该隐藏层的输出。所以一次线性变化加上一次激活函数组成MLP中完整的一个隐藏层，又称全连接层(Dense Layer)。
- 第L层的全连接层输出可被当做是第$L+1$层全连接层的输入，以此类推。可将最终一层全连接层的输出当做模型的输出。

9.1 激活函数

- 激活函数（Activation Function）。

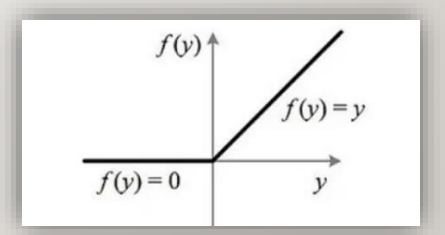
$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$





$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$ReLU = \max(0, x) = f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



9.2 激活函数的意义

- 如果没有非线性激活单元，则多层的线性层是没有意义的。假设有l个隐藏线性层，则x经过多层投影得到第l层输出的这一过程是可被描述为：

$$h^1 = \boldsymbol{w}^0 \cdot \boldsymbol{x}$$
$$h^2 = \boldsymbol{w}^1 \cdot \boldsymbol{h}^1$$
$$\cdots$$
$$h^l = \boldsymbol{w}^{l-1} \cdot \boldsymbol{h}^{l-1}$$

- 如果将此公式稍微改变一下形式可得：

$$h^l = \boldsymbol{w}^{l-1} \cdots \boldsymbol{w}^1 \boldsymbol{w}^0 \boldsymbol{x}$$

- 由此可见中间隐藏层的输出全部都没有意义，$\boldsymbol{w}^{l-1} \cdots \boldsymbol{w}^1 \boldsymbol{w}^0$这些计算最终只是输出一个第$l$层的权重$\boldsymbol{w}^l$, 因为永远是进行的线性变换，所以多次线性变换完全可由一次合适的线性变化代替。所以这个多层的神经网络与只初始化一个w^l的单层神经网络其实并无差别。而加入非线性激活单元就不一样了，通常由$\sigma(\cdot)$表示一次非线性激活计算。这样一来l个隐藏层的公式就必须写成：

$$h^1 = \sigma(\boldsymbol{w}^0 \cdot \boldsymbol{x})$$
$$h^2 = \sigma(\boldsymbol{w}^1 \cdot \boldsymbol{h}^1)$$
$$\cdots$$
$$h^l = \sigma\left(\boldsymbol{w}^{l-1} \cdot \boldsymbol{h}^{l-1}\right)$$

- 如此一来，多层的神经网络就会变得有意义。

10. 鸢尾花预测MLP版

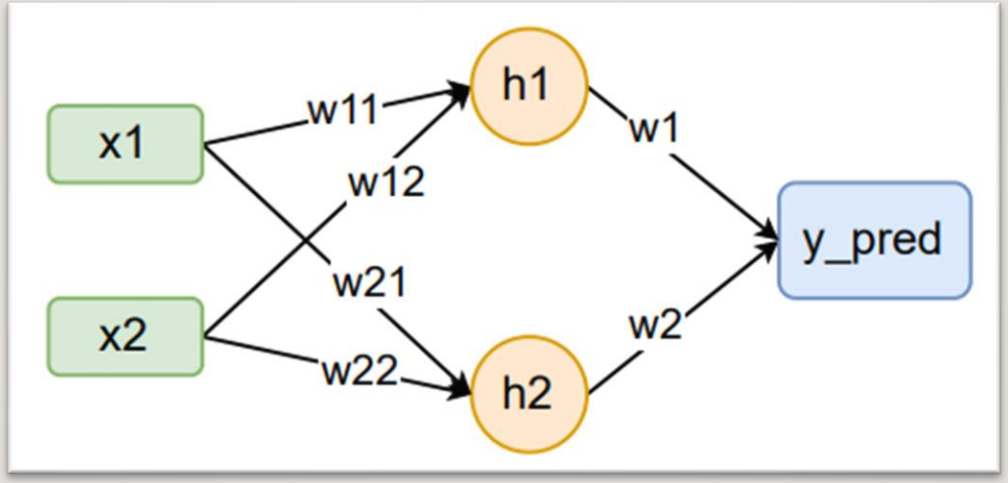
```
class Softmax_Reg_MLP( nn.Module ):
    def __init__( self, n_features,n_classes ):
        super(Softmax_Reg_MLP, self).__init__()
        self.mlp = nn.Sequential(
            nn.Linear( n_features, n_features*2),
            nn.Sigmoid(),
            nn.Linear( n_features*2, n_features ),
            nn.Sigmoid(),
            nn.Linear( n_features, n_classes),
            nn.Softmax()
        )

    def forward( self, x ):
        y = self.mlp(x)
        return y
```

- 普通的Sotfmax回归与MLP之间平均准确率的区别。

- 普通: 0.889
- MLP: 0.978

11. 前向传播与后向传播

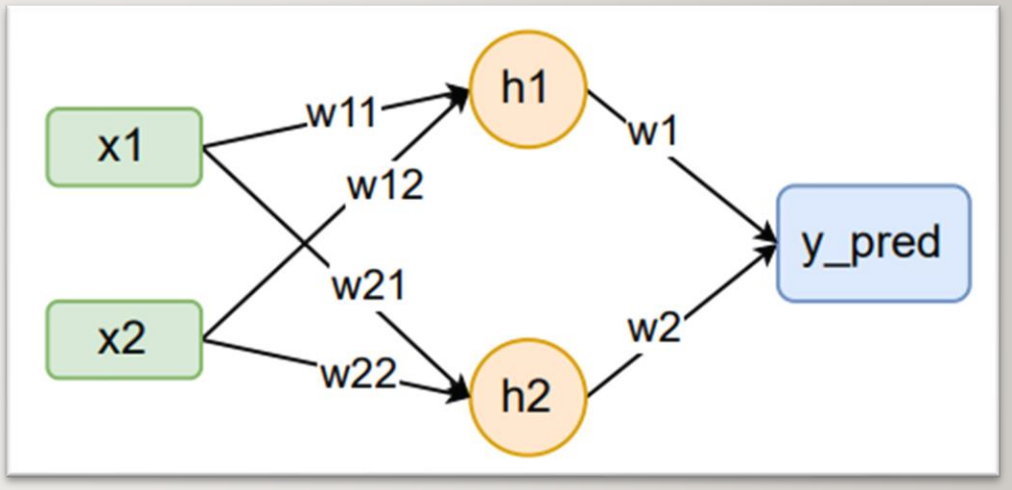


- 前向传播 (Forward)，既从输入到输出的计算过程。
- 后向传播 (Backward)，从输出根据链式求导法求损失函数对于各个模型参数对于的偏导，以更新各模型参数的过程。

11.1后向传播

- 假设目前神经网络的结构如右图所示。从输入到输出的公式组可表示为:

$$h1' = w11x1 + w12x2$$
$$h2' = w21x1 + w22x2$$
$$h1 = \sigma(h1')$$
$$h2 = \sigma(h2')$$
$$y_{pred} = w1h1 + w2h2$$
$$L = \frac{1}{2}(\mathrm{y_{pred}} - \mathrm{y_{ture}})^2$$



- 根据链式求导法则:

$$\frac{\partial L}{\partial w1} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial w1} = (y_{pred} - y_{true})h1$$

$$\frac{\partial L}{\partial w2} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial w2} = (y_{pred} - y_{true})h2$$

$$\frac{\partial L}{\partial w11} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial h1}\frac{\partial h1}{\partial h1'}\frac{\partial h1'}{\partial w11} = (y_{pred} - y_{true}) \cdot w1 \cdot \sigma'(h1') \cdot x1$$

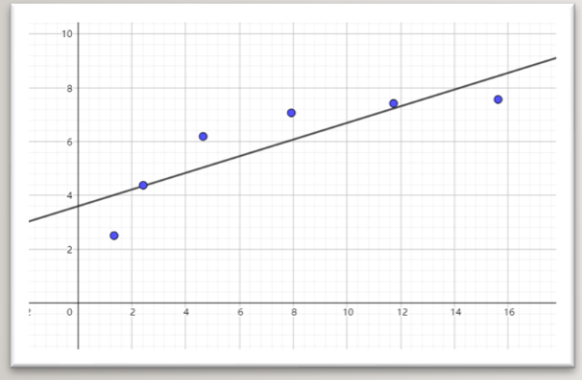
$$\frac{\partial L}{\partial w12} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial h1}\frac{\partial h1}{\partial h1'}\frac{\partial h1'}{\partial w12} = (y_{pred} - y_{true}) \cdot w1 \cdot \sigma'(h1') \cdot x2$$

$$\frac{\partial L}{\partial w21} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial h2}\frac{\partial h2}{\partial h2}\frac{\partial h2'}{\partial w21} = (y_{pred} - y_{true}) \cdot w2 \cdot \sigma'(h2') \cdot x1$$

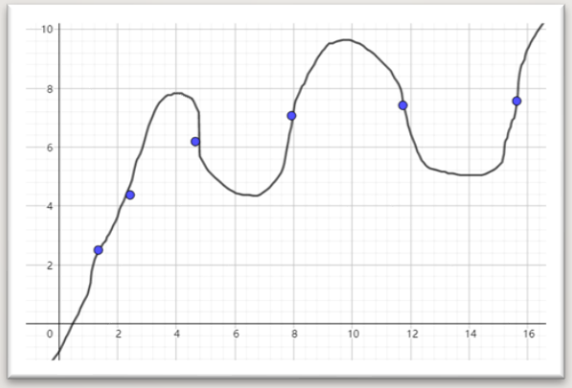
$$\frac{\partial L}{\partial w22} = \frac{\partial L}{\partial y_{pred}}\frac{\partial y_{pred}}{\partial h2}\frac{\partial h2}{\partial h2}\frac{\partial h2'}{\partial w22} = (y_{pred} - y_{true}) \cdot w2 \cdot \sigma'(h2') \cdot x2$$

12. 欠拟合与过拟合

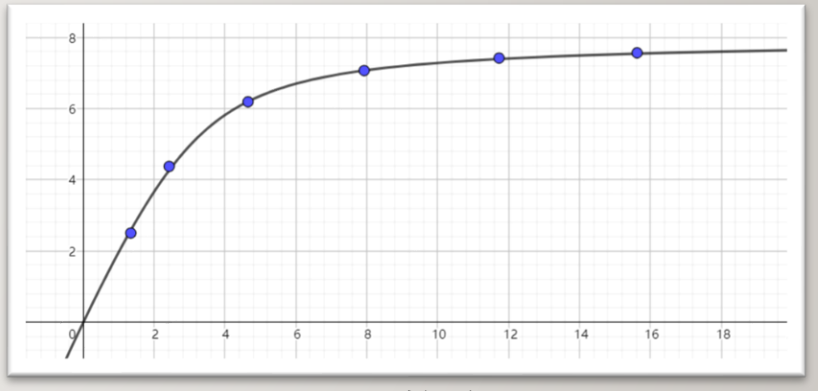
1. 欠拟合(Underfitting)：模型拟合能力不够。
2. 过拟合(Overfitting)：模型在训练集中拟合很好，但在测试集中无法很好的预测。



欠拟合



过拟合



正好拟合

12.欠拟合与过拟合

- 过拟合产生的原因。
  1. 训练集的数据量不足以支撑太复杂或模型参数过多的模型。
  2. 训练集中存在过多噪音数据，使得模型记住了噪音特征，反而忽略了真实有用的特征。
  3. 训练次数太多。
- 处理过拟合的一般方案。
  1. 增加训练集数据量。
  2. 降低模型复杂度。
  3. 数据增强。
  4. 特征筛选，降维。
  5. 减少训练次数。
  6. 正则化。
  7. 丢弃法(Dropout)。
  8. 批量归一化。(Batch Normalization)。

结束