18040300058 顾浩然　　第一天作业：

一．深度学习基础知识

①深度学习发展历史

萌芽期（20 世纪 40 年代）

人工神经网络的研究最早可以追溯到人类开始研究自己的智能的时期；

1943 年，心理学家 McCulloch 和数学家 Pitts 建立了著名的阈值加权和模型，简称为 M-P 模型。发表于数学生物物理学会刊《Bulletin of Methematical Biophysics》；

1949 年，心理学家 D. O. Hebb 提出神经元之间突触联系是可变的假说——Hebb 学习律。

↓

第一次高潮期（1950~1968）

以 Marvin Minsky，Frank Rosenblatt，Bernard Widrow 等为代表人物，代表作是单级感知器（Perceptron）；

可用电子线路模拟；

人们乐观地认为几乎已经找到了智能的关键。许多部门都开始大批地投入此项研究，希望尽快占领制高点。

↓

反思期（1969~1982）

M. L. Minsky 和 S. Papert，《Perceptron》，MIT Press，1969 年；

“异或”运算不可表示；

二十世纪 70 年代和 80 年代早期的研究结果；

认识规律：认识——实践——再认识。

↓

第二次高潮期（1983~1990）

1982 年，J. Hopfield 提出循环网络：

建立 ANN 稳定性的判别依据；

阐明了 ANN 与动力学的关系；

用非线性动力学的方法来研究 ANN 的特性；

指出信息被存放在网络中神经元的联接上。

↓

再认识与应用研究期（1991~2011）

存在应用面还不够宽、结果不够精确等问题；

改进现有模型、算法，以提高网络的训练速度和运行的准确度；

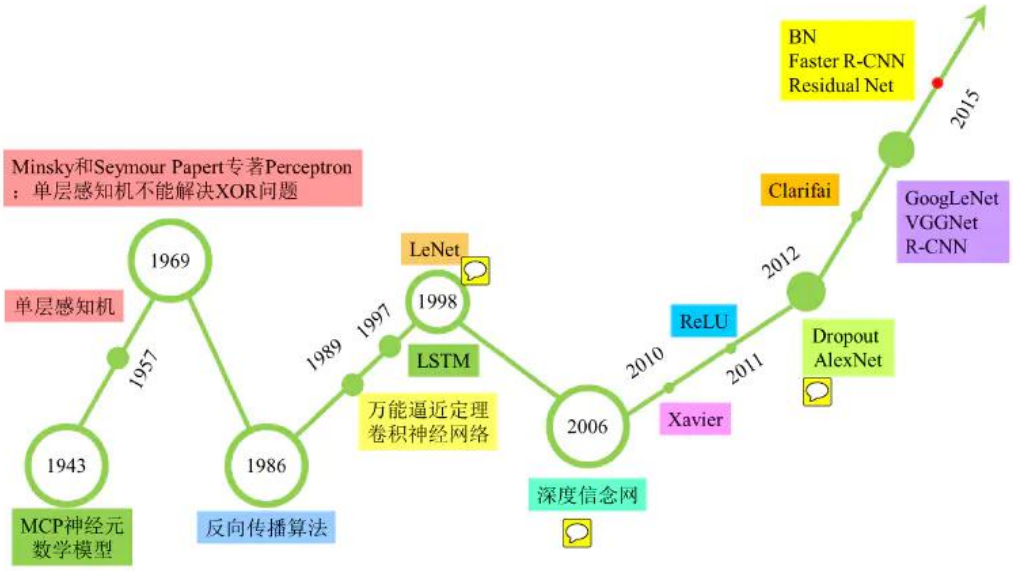
算法的集成；

希望在理论上寻找新的突破，建立新的专用/通用模型和算法；

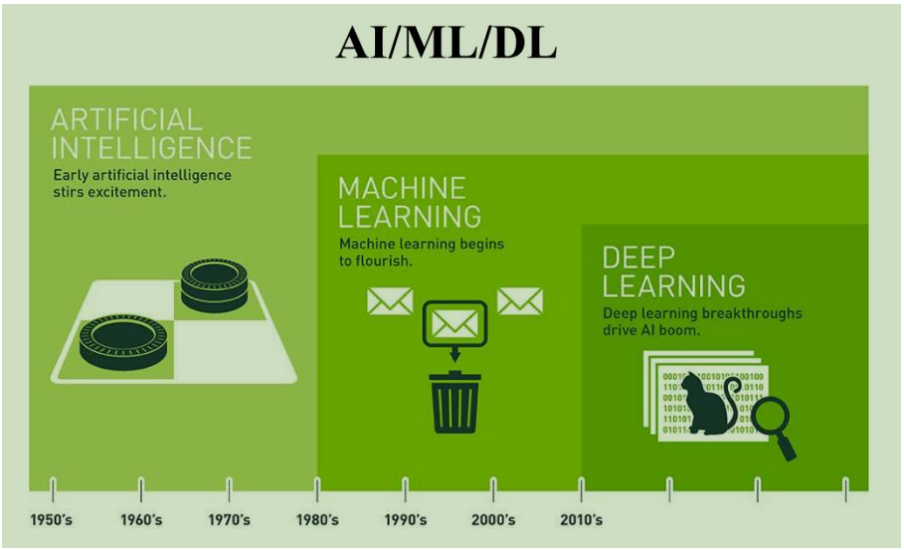
进一步对生物神经系统进行研究，不断地丰富对人脑的认识。

↓

爆发期（2012~现在）



②人工智能、机器学习、深度学习有什么区别和联系



如上图：人工智能是最早出现的，也是最大、最外侧的同心圆；其次是机器学习，稍晚一点；最内侧，是深度学习，当今人工智能大爆炸的核心驱动。

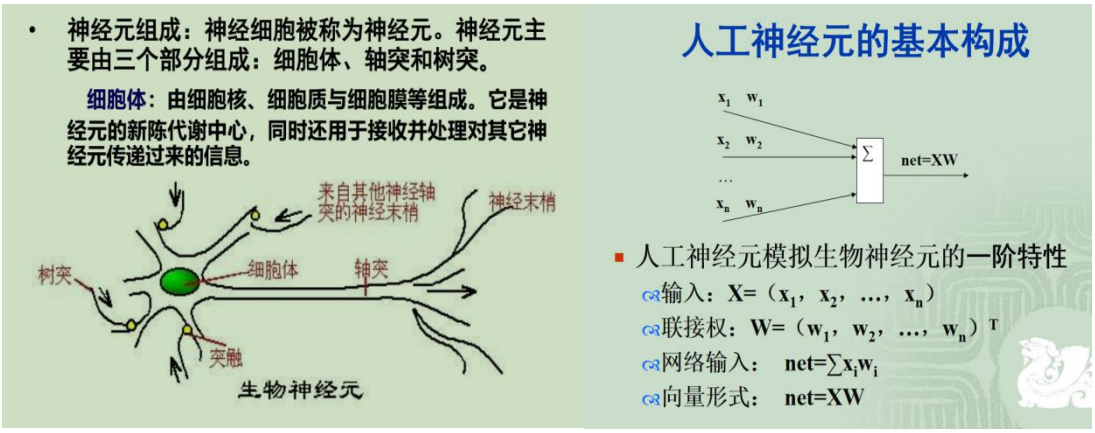
人工智能（Artificial Intelligence）——为机器赋予人的智能；

机器学习—— 一种实现人工智能的方法；

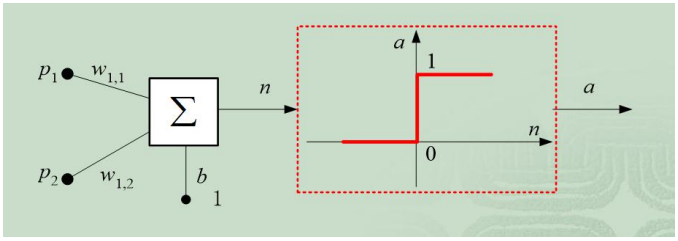
深度学习—— 一种实现机器学习的技术。

③神经元、单层感知机、多层感知机

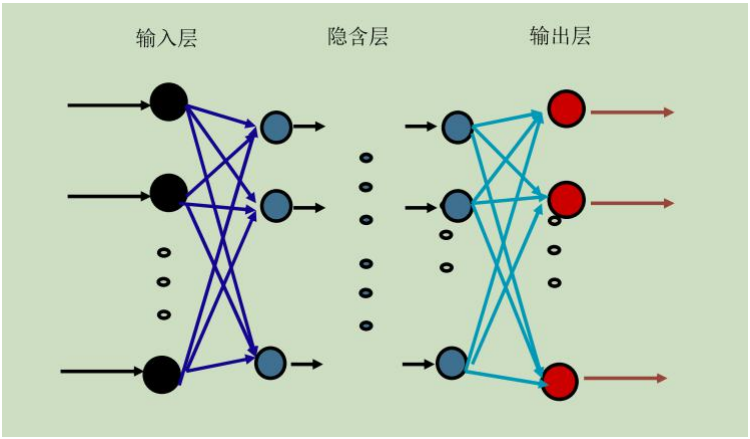
神经元



单层感知机

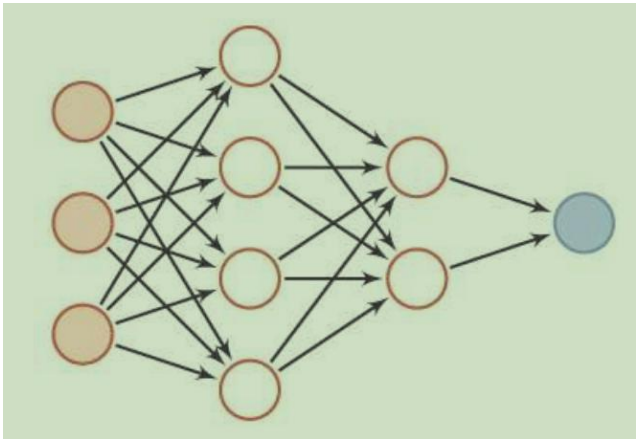


多层感知机



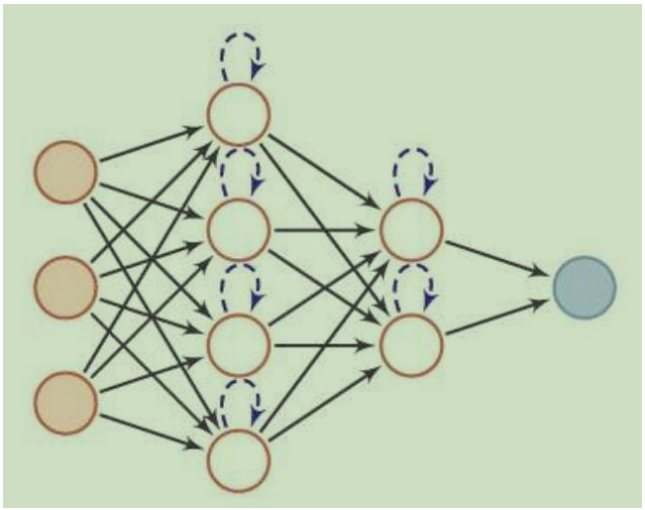
④前向传播

定义：前向传播，就是给网络输入一个样本向量，该样本向量的各元素，经过各隐藏层的逐级加权求和+非线性激活，最终由输出层输出一个预测向量的过程。



⑤反向传播

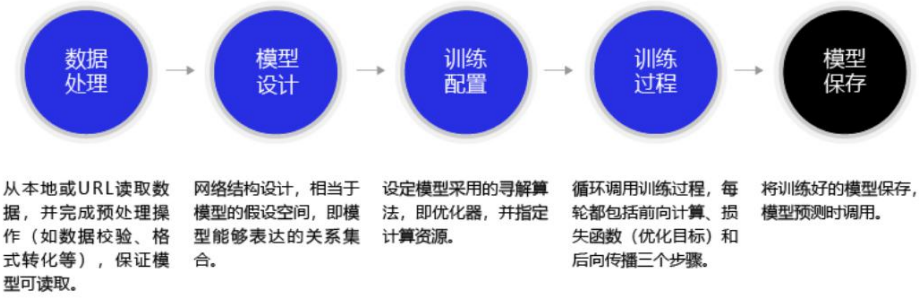
定义：反向传播算法(BP 算法)主要由两个环节(激励传播、权重更新)反复循环迭代，直到网络对输入的响应达到预定的目标范围为止。



二．代码实践

①使用 Python+numpy 实现具有两层全连接层的模型

②使用 paddle 实现具有两层全连接层的模型



1. 数据处理

进行数据的导入及归一化等操作。

2. 模型设计

模型定义的实质是定义线性回归的网络结构，通过创建 Python 类的方式完成模型网络的定义，该类需要继承 paddle.nn.Layer 父类，并且在类中定义 init 函数和 forward 函数。forward 函数是框架指定实现前向计算逻辑的函数，程序在调用模型实例时会自动执行 forward 方法。在 forward 函数中使用的网络层需要在 init 函数中声明。

实现过程分如下两步：

Ⅰ.定义 init 函数：在类的初始化函数中声明每一层网络的实现函数。

Ⅱ.定义 forward 函数：构建神经网络结构，实现前向计算过程，并返回预测结果，在本任务中返回的是房价预测结果。

3. 训练配置

声明定义好的回归模型 Regressor 实例，并将模型的状态设置为训练。

使用 load_data 函数加载训练数据和测试数据。

设置优化算法和学习率，优化算法采用随机梯度下降 SGD，学习率设置为 0.01。

4. 训练过程

训练过程采用二层循环嵌套方式：

内层循环： 负责整个数据集的一次遍历，采用分批次方式（batch）。假设数据集样本数量为 1000，一个批次有 10 个样本，则遍历一次数据集的批次数量是 1000/10=100，即内层循环需要执行 100 次。

```
for iter_id, mini_batch in enumerate(mini_batches):
```

外层循环： 定义遍历数据集的次数，通过参数 EPOCH_NUM 设置。

```
for epoch_id in range(EPOCH_NUM):
```

内循环过程

数据准备：将一个批次的数据先转换成 np.array 格式，再转换成 paddle 内置 tensor 格式。

前向计算：将一个批次的样本数据灌入网络中，计算输出结果。
计算损失函数：以前向计算结果和真实房价作为输入，通过损失函数 square_error_cost API 计算出损失函数值（Loss）。
反向传播：执行梯度反向传播 backward 函数，即从后到前逐层计算每一层的梯度，并根据设置的优化算法更新参数。

5. 保存并测试模型

保存模型
将模型当前的参数数据 model.state_dict()保存到文件中（通过参数指定保存的文件名 LR_model），以备预测或校验的程序调用。

测试模型
下面选择一条数据样本，测试下模型的预测效果。
本次作业，我选择的是倒数第 10 个样本进行房价的预测，对两个模型测试结果如下：
(1)python+numpy

```
Epoch  47 / iter   2, loss = 0.0737
Epoch  47 / iter   3, loss = 0.0811
Epoch  47 / iter   4, loss = 0.0964
Epoch  48 / iter   0, loss = 0.1426
Epoch  48 / iter   1, loss = 0.1198
Epoch  48 / iter   2, loss = 0.0794
Epoch  48 / iter   3, loss = 0.0802
Epoch  48 / iter   4, loss = 0.0011
Epoch  49 / iter   0, loss = 0.0721
Epoch  49 / iter   1, loss = 0.0474
Epoch  49 / iter   2, loss = 0.0509
Epoch  49 / iter   3, loss = 0.0565
Epoch  49 / iter   4, loss = 0.0491
取倒数第十行数据测试实际值为 19.7 预测值为 [21.3116794]
```

(2)paddle 框架

```
epoch: 0, iter: 0, loss is: [0.07902765]
epoch: 0, iter: 20, loss is: [0.0293816]
epoch: 0, iter: 40, loss is: [0.00801842]
epoch: 1, iter: 0, loss is: [0.12996557]
epoch: 1, iter: 20, loss is: [0.1096014]
epoch: 1, iter: 40, loss is: [0.01333511]
epoch: 2, iter: 0, loss is: [0.00940842]
epoch: 2, iter: 20, loss is: [0.03605303]
epoch: 2, iter: 40, loss is: [0.00727957]
epoch: 3, iter: 0, loss is: [0.06534393]
epoch: 3, iter: 20, loss is: [0.03968224]
epoch: 3, iter: 40, loss is: [0.11516473]
epoch: 4, iter: 0, loss is: [0.02689059]
epoch: 4, iter: 20, loss is: [0.09326521]
epoch: 4, iter: 40, loss is: [0.02105876]
epoch: 5, iter: 0, loss is: [0.06304868]
epoch: 5, iter: 20, loss is: [0.01863404]
epoch: 5, iter: 40, loss is: [0.17284994]
epoch: 6, iter: 0, loss is: [0.01640263]
epoch: 6, iter: 20, loss is: [0.01484649]
epoch: 6, iter: 40, loss is: [0.06604956]
epoch: 7, iter: 0, loss is: [0.02909173]
epoch: 7, iter: 20, loss is: [0.03317512]
epoch: 7, iter: 40, loss is: [0.06654827]
epoch: 8, iter: 0, loss is: [0.02663278]
epoch: 8, iter: 20, loss is: [0.03054448]
epoch: 8, iter: 40, loss is: [0.00934739]
epoch: 9, iter: 0, loss is: [0.01772374]
epoch: 9, iter: 20, loss is: [0.0228412]
epoch: 9, iter: 40, loss is: [0.00266121]
```

```
Inference result is [[19.85326]], the corresponding label is 19.700000762939453
```

对比发现：
①paddle 框架的损失函数值下降较快；
②两个模型对于数据的预测差异不大，且与实际房价差别很小。