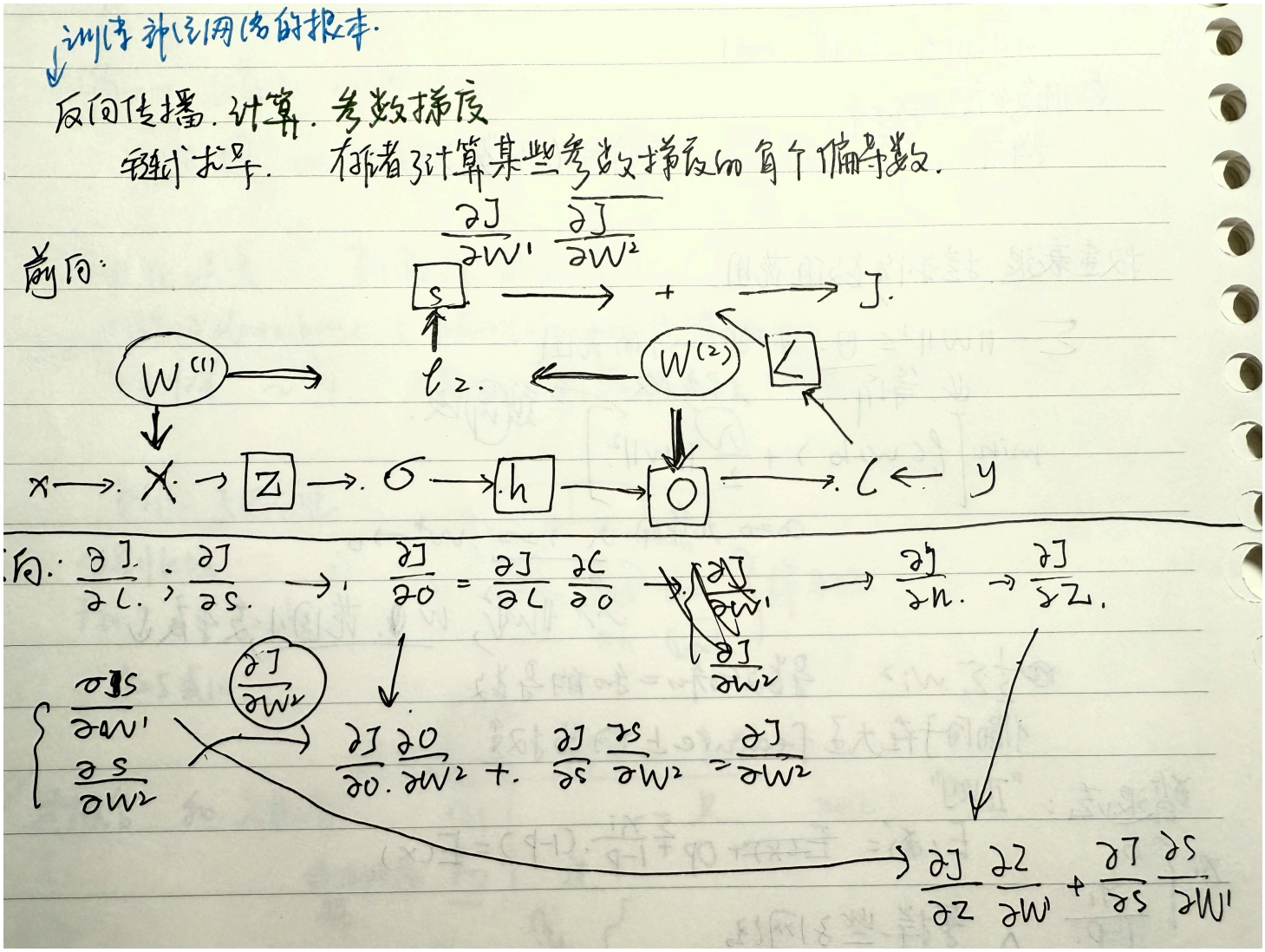
招新题解 2: 多层感知机理论

1.解释概念

1.

（1）隐藏层相当于神经中枢，是位于输入层和输出层之间的神经网络层，可设置大小和层数，通过激活函数对输入数据做非线性运算从而进行特征转换
（2）激活函数有点类似于判断神经冲动是否达到阈值的标准（Sigmoid 0，1）。是隐藏层（和输出层）每个神经元输出公式中的非线性函数。它引入了非线性运算，使得神经网络能够学习更为复杂的模型如图像识别
（3）全连接层相当于神经网络，其中每个神经元都与前一层的所有神经元相连接。也就是说它会对每一个输入输出的数据产生影响
2.
（1）过拟合是模型在训练数据上损失极低，但在测试数据上泛化误差很大。模型相对于数据过于复杂，甚至拟合出训练数据中的噪声和细节，而忽略了通用规律。
（2）欠拟合是模型在训练数据上损失很大，但在测试数据上泛化误差也很大。模型相对于复杂数据过于简单，无法捕捉数据中的通用规律。
（3）暂退法是一种解决过拟合的正则化技术。在训练过程中，它随机地将一部分隐藏层输出暂时设置为0，这可以防止模型对训练数据中的某些特定隐藏层产生过度依赖，每次训练时都在训练一个子网络，降低了模型的复杂程度以避免过拟合，在最后预测时仍是所有这些子网络的综合。
（4） 权重衰减是另一种解决过拟合的正则化技术。它在损失函数中添加一项惩罚项（k*权重平方，k越大权重越小，L2范数）再取最小值。这个惩罚项会让模型权重值的选择范围更小。较小的权重模型更加简单，泛化能力更强。
3.
（1）前向传播是数据从输入层开始，逐层经过隐藏层，最终到达输出层并产生预测结果的计算过程。按正序进行，存储的数据是进行反向传播的基础
（2）反向传播是在前向传播计算出预测结果和损失后，根据链式法则，将损失从输出层向输入层按倒序传播，以计算损失函数相对于每个参数的梯度并更新梯度。



（3）梯度消失是在反向传播过程中，梯度值变得越来越小，趋近于零。导致靠近参数几乎得不到更新，无法有效学习。原因可能是学习率过小或激活函数（主要是01的sigmoid函数）连续相乘会导致梯度指数级减小。

（4）梯度爆炸与梯度消失相反，梯度在反向传播中变得非常大（指数级增长）。导致参数更新幅度巨大，模型无法收敛。原因可能是学习率过大，参数选择范围过大

```python
    if epoch % 20 == 0:
        print(f'epoch {epoch }, loss: {loss:.5f}')
```

```
poch 0, loss: 25188.26795
poch 20, loss: 3711513778411785147379750379834545402977824229468760521616281928952514330848410891241210031045566319820 8.00000
poch 40, loss: 5469829189761272320079078602211101125721460676005818318807233596420466465351214873220218597548315416169254479251015740250386119137438506238859214680273562558322286985855381044487009214534693749721661440.00000
poch 60, loss: 8061139780536484303508638838869378388076122897369048404596940702159890457001901732261152903306983482984366071380761129682304123029703672983643916961294992859266865197383785989618545726610467551072996110371671647970816030427177647849513975749546957718884588687872926178332866243452990131760705887535 10.00000
poch 80, loss: inf
poch 100, loss: inf
poch 120, loss: inf
poch 140, loss: nan
poch 160, loss: nan
poch 180, loss: nan
```

2.回答问题

1.单层感知机是一个没有隐藏层仅有输入层和输出层的线性模型，只能通过一条直线来划分数据，XOR输出是由两个维度的输入决定的（相同为0不同为1），这就像找不到一条直线可以分离出第一三和二四象限一样。多层感知机增加了一个或多个隐藏层和非线性的激活函数，隐藏层可以学习到输入数据的中间特征，进行2次对逻辑的学习，输出层再将这些中间特征组合起来，从而得到XOR的结果。激活函数可以让直线边界弯曲以划分数据

2.

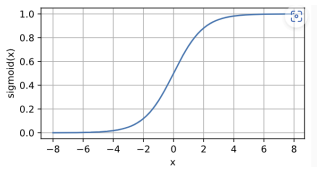
（1）h*(d + 1) + o*(h + 1)；h*d是输入层到隐藏层权重矩阵的维度，原理是矩阵乘法，+1是bias,*oh是隐藏层到输出层权重矩阵维度，同理bias

（2）Float32：4*参数量 B，INT8：1*参数量 B

（3）O(d × h + h × o),O(1)是激活函数的计算复杂度

3.

（1）对比

激活函数	公式	梯度	特点
ReLU	f(x) = max(0, x)	f'(x) = 1 if x>0 else 0	计算简单；在正区间梯度恒为1， 防止梯度消失
Sigmoid	f(x) = 1 / (1 + exp(-x))	f'(x) = f(x) * (1 - f(x))	由于不过原点， 每层方差和均值相差较大， 容易导致梯度消失。 当输入值很大或很小时，参数更新慢
Tanh	f(x) = (1 - exp(-2x)) / (1+exp(-2x))	f'(x) = 1 - (f(x))**2	"形状"与sigmoid相似，存在梯度消失， 但过原点且梯度在0是=1， 所以数值稳定性比Sigmoid好
			

（2）死ReLU问题

“死ReLU”是指当输入值小于0时，ReLU函数的输出和梯度都恒为0。导致在反向传播时权重无法被更新。原因可能是lr过高,可适当降低；bias为较小负数；或权重初始化有问题，可使用He初始化

（3）简化计算

指数运算exp(-x)导致了计算复杂性。 在前向传播中，可以采用数值近似法，预先计算并存储可能输入的 x 值的sigmoid(x) 值，通过查表来快速得到近似结果。在反向传播中，由于exp()的导数还是exp(),可重用前向传播的结果来求梯度。(想提高数据稳定性可以用变式：4*sigmoid(x)-2,过原点，梯度近似为1)

4.暂退法 (Dropout) 如何防止过拟合

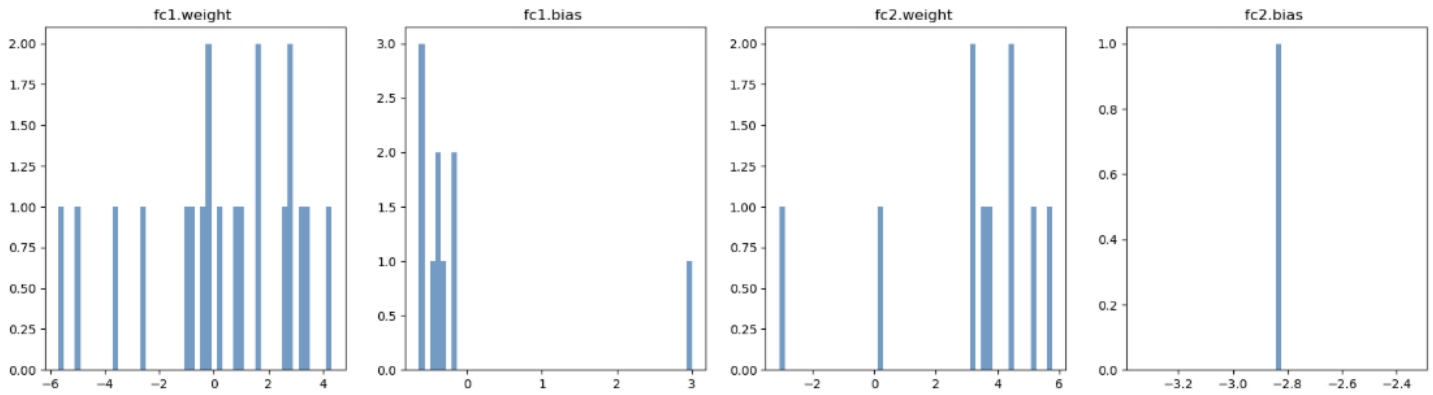
在训练过程中，模型随机地将一部分隐藏层输出暂时设置为0，相当于每次训练时都在训练一个子网络，减小对数据某特性的过度关注，降低了模型的复杂程度以避免过拟合，在最后预测时仍是所有这些子网络的平均综合。

3动手实现MLP基础架构

4简单的MLP实验

基本思路：定义MLP模型-定义激活函数-定义前向传播-生成数据集-划分训练测试集-定义训练模型（loss,参数更新）-绘制频率直方图（具体注释都在代码文件里）

```
Epoch [0], Loss: 0.8004, Accuracy: 0.6000
Epoch [10], Loss: 0.4225, Accuracy: 0.8150
Epoch [20], Loss: 0.3194, Accuracy: 0.8150
Epoch [30], Loss: 0.2658, Accuracy: 0.8800
Epoch [40], Loss: 0.2294, Accuracy: 0.9200
Epoch [50], Loss: 0.2012, Accuracy: 0.9450
Epoch [60], Loss: 0.1762, Accuracy: 0.9600
Epoch [70], Loss: 0.1552, Accuracy: 0.9800
Epoch [80], Loss: 0.1380, Accuracy: 0.9800
Epoch [90], Loss: 0.1241, Accuracy: 0.9800
Epoch [100], Loss: 0.1127, Accuracy: 0.9800
Epoch [110], Loss: 0.1035, Accuracy: 0.9800
Epoch [120], Loss: 0.0965, Accuracy: 0.9800
Epoch [130], Loss: 0.0900, Accuracy: 0.9800
Epoch [140], Loss: 0.0847, Accuracy: 0.9800
```



5让我的模型效果更好

1.深度与宽度

（1）增加深度有利于学习更复杂的模型，拟合出更复杂的非线性函数，但有过拟合的风险，训练起来更难，需要调整初始化和优化函数；增加宽度能学习到数据更多特征，但可能受到无关量如噪音的影响。

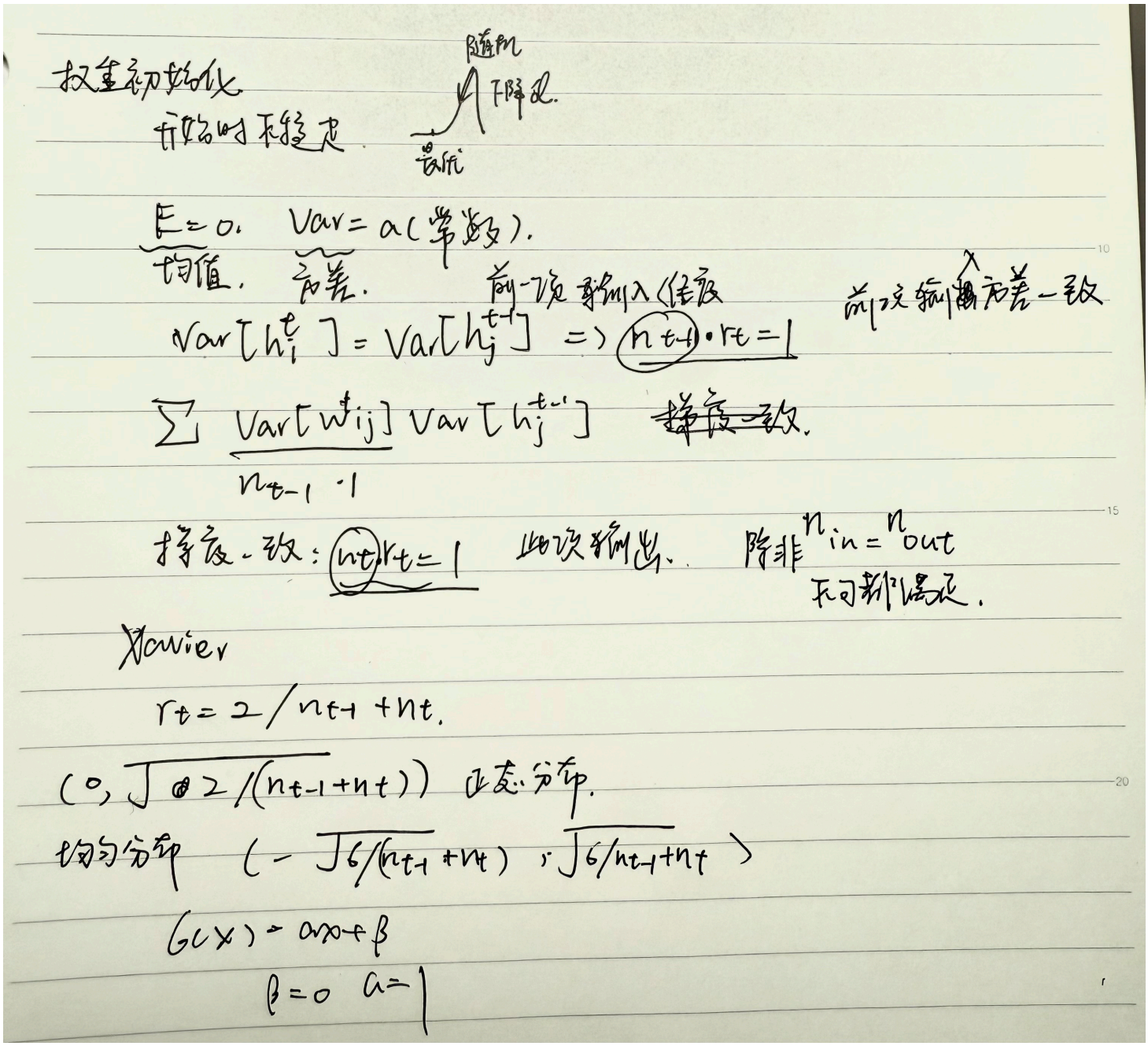
（2）计算效率上，增加深度比宽度效率高。原因：1.在一个很大的模型中，增加宽度时参数量随宽度平方级，增加深度时参数量随深度线性增长，求导显然增加深度时的速度更慢，而参数的增加意味着矩阵乘法运算的增加消耗了更多算力2.增加宽度相当于增加了存储模块，消耗了更多内存。综上，一般优先增加深度

2.初始化方法

（1）所有神经元初始值都为0，所有梯度更新一样，这意味着经过多次更新，这些权重始终保持相同的

值。同层的所有神经元学习了数据的相同特征，宽度失效了

（2）Xavier原理：根据输入和输出维度调整权重方差，（均值0，方差（2/input维度+output）**1/2）维度保持前项输入和反向传播中此项输出的方差基本一致



（3） He初始化是对Xavier初始化的一个改进，考虑到ReLU负值为0导致的方差减半，又乘了2倍，从而解决了死ReLU问题，在深层网络中能保持梯度方差稳定，缓解梯度消失

3.优化困境分析

（1） 局部最优与鞍点问题：优化器的目标是找到全局最优点，会遇到局部最优点和鞍点两大主要障碍。在一个局部邻域内，局部最优点的损失是最低的，但仍高于全局最优点。且在这一点上，所有方向梯度都为 0，也因此在高维模型中不常见；而在鞍点上，只有某些方向的梯度为0，更加常见。优化器很容易陷入梯度为0的鞍点或局部最优点区域，导致训练停滞

（2）学习率设置策略：1可以动手检验真理，多调多试先定范围在缩减到合适值2使用自适应优化器如

Adam，使每个参数拥有自己的学习率（依据它们的梯度定lr）

（3）批量大小的影响：

- 大批量:梯度估计基于更多数据梯度准确性高,GPU并行计算利用率高，训练效率高；
- 小批量：受局部最优点和鞍点的误导小，拟合能力更强，一个epoch需要更次参数更新，计算效率低，内存消耗低，按需求选择吧
- 加了He初始化和自适应优化器，然而效果没啥差，可能是因为纬度低用不着，loss本来就不大吧
-

```
losses, accuracies = train_model(model, x_train
```

```
Epoch [0], Loss: 0.8924, Accuracy: 0.5900
Epoch [1000], Loss: 0.0294, Accuracy: 0.9650
Epoch [2000], Loss: 0.0173, Accuracy: 0.9650
Epoch [3000], Loss: 0.0125, Accuracy: 0.9650
Epoch [4000], Loss: 0.0098, Accuracy: 0.9650
Epoch [5000], Loss: 0.0042, Accuracy: 0.9650
Epoch [6000], Loss: 0.0024, Accuracy: 0.9700
Epoch [7000], Loss: 0.0018, Accuracy: 0.9700
Epoch [8000], Loss: 0.0013, Accuracy: 0.9650
Epoch [9000], Loss: 0.0009, Accuracy: 0.9600
```