作业2：线性模型、决策树和强化学习

李堃秀 2022012110 Likx22@mails.tsinghua.edu.cn

1 简答题

1. **什么样的场景适合使用K 折交叉验证（K-foldCross Validation）？参数 K 越大越好吗?**
   适合使用K 折交叉验证的场景：
   (1)数据量有限、直接划分训练集和测试集容易导致模型性能评估不稳定的场景；
   (2)希望对不同模型进行性能评估，并在多个模型之间进行选择的场景；
   (3)希望进行参数评估及调优从而避免过拟合的场景。
   参数 K 并不是越大越好：增大参数K可以有效提升评估精度，但也会带来较大的计算开销，因此需要根据实际需要评估参数K的具体取值。

2. **线性模型中为什么要引入基函数？基函数越复杂，学习的效果一定会越好吗?**
   在线性模型中引入基函数是为了增强模型的表达能力，使得原本无法用线性模型表达的非线性关系也能被模型学习。
   基函数越复杂，学习的效果不一定会越好，因为这将增加计算开销，且有可能会导致过拟合。

3. **随机森林使用哪些方法增加单棵决策树的多样性?**
   随机森林主要通过以下两种方法增加单棵决策树的多样性：
   (1) 随机抽样数据：对训练数据进行有放回的随机抽样；
   (2) 随机选择特征：在每次分裂节点时，从所有特征中随机选择一部分特征进行分裂。

4. **为什么说相比使用蒙特卡洛（`Monte-Carlo`）采样的策略梯度（`PolicyGradient`）方法，`Actor critic` 方法的方差更小?**
   原因：
   (1) `Actor critic` 方法相比于策略梯度方法，额外引入了价值函数。价值函数可以为策略更新提供基准，使得策略更新时更加稳定；
   (2) 蒙特卡洛方法依赖于从环境中获得的整个轨迹的回报，这通常具有较高的方差，而`Actor critic`方法通过使用策略函数对状态价值进行估计估计，可以有效减小这种波动，从而能够更加平稳地更新策略；

5. **为什么使用神经网络（`NeuralNetwork`）的 `Q-Learning` 不能保证收敛到最优状态动作值函数 Q^*？使用基于策略的方法（`Policy-based`）方法能保证收敛到全局最优策略吗?**
   Q-Learning 获得目标值依赖于当前策略，而神经网络的更新可能导致策略频繁变化，而不完善的探索策略可能导致某些状态空间没有被充分探索，从而影响收敛。同时，神经网络用于函数逼近可能会引入误差，导致 Q 值更新不准确。
   使用基于策略的方法不能保证收敛到全局最优策略，原因包括：
   (1) 策略空间非常复杂时，优化过程不一定能够完全探索所有可能的策略，因此优化过程可能陷入局部最优，无法找到全局最优策略；
   (2) 基于策略的方法需要大量的样本才能更为准确地估计梯度，样本不足时可能影响收敛效果。

2 ID3 算法的次优性

考虑以下训练数据，其中 $X = \{0,1\}^3, Y = \{0,1\}$:

$$\begin{aligned}(x_1,y_1)&=((1,1,1),1)\\(x_2,y_2)&=((1,0,0),1)\\(x_3,y_3)&=((1,1,0),0)\\(x_4,y_4)&=((0,0,1),0)\end{aligned}$$

1.写出基于信息增益准则的ID3算法构造最大深度不超过2的决策树的过程，包括信息增益的计算过程(当两个属性信息增益相同时，任选其一进行分裂);并说明训练误差将至少为1/4即至少有一个训练样本分类错误)。

2.写出一棵深度为2的决策树，其训练误差为0。

1. 构造过程：

1) 计算初始信息熵：

由数据集中有2个正例(y=1)和两个反例(y=0)，有

$$\begin{aligned}H(D)&=-\sum_i p_i\log_2\ (p_i)\\&=-\left(\frac{2}{4}\log_2\frac{2}{4}+\frac{2}{4}\log_2\frac{2}{4}\right)\\&=1\end{aligned}$$

2) 计算每个属性的条件熵和信息增益：

a. 对于x_1

$$\begin{aligned}H(D|x_1)&=\frac{3}{4}H(D|x_1=1)+\frac{1}{4}H(D|x_1=0)\\&=-\left(\frac{2}{3}\log_2\frac{2}{3}+\frac{1}{3}\log_2\frac{1}{3}\right)+0\\&\approx 0.688\end{aligned}$$

故信息增益为：

$$IG(D,x_1)=H(D)-H(D\mid x_1)=1-0.688=0.312$$

b. 对于x_2

$$\begin{aligned}H(D|x_2)&=\frac{2}{4}H(D|x_2=1)+\frac{2}{4}H(D|x_2=0)\\&=-\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)-\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)\\&=1\end{aligned}$$

故信息增益为：

$$IG(D,x_2)=H(D)-H(D\mid x_2)=1-1=0$$

c. 对于x_3

$$\begin{aligned}H(D|x_3)&=\frac{2}{4}H(D|x_3=1)+\frac{2}{4}H(D|x_3=0)\\&=-\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)-\left(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}\right)\\&=1\end{aligned}$$

故信息增益为：

$$IG(D,x_3)=H(D)-H(D\mid x_3)=1-1=0$$

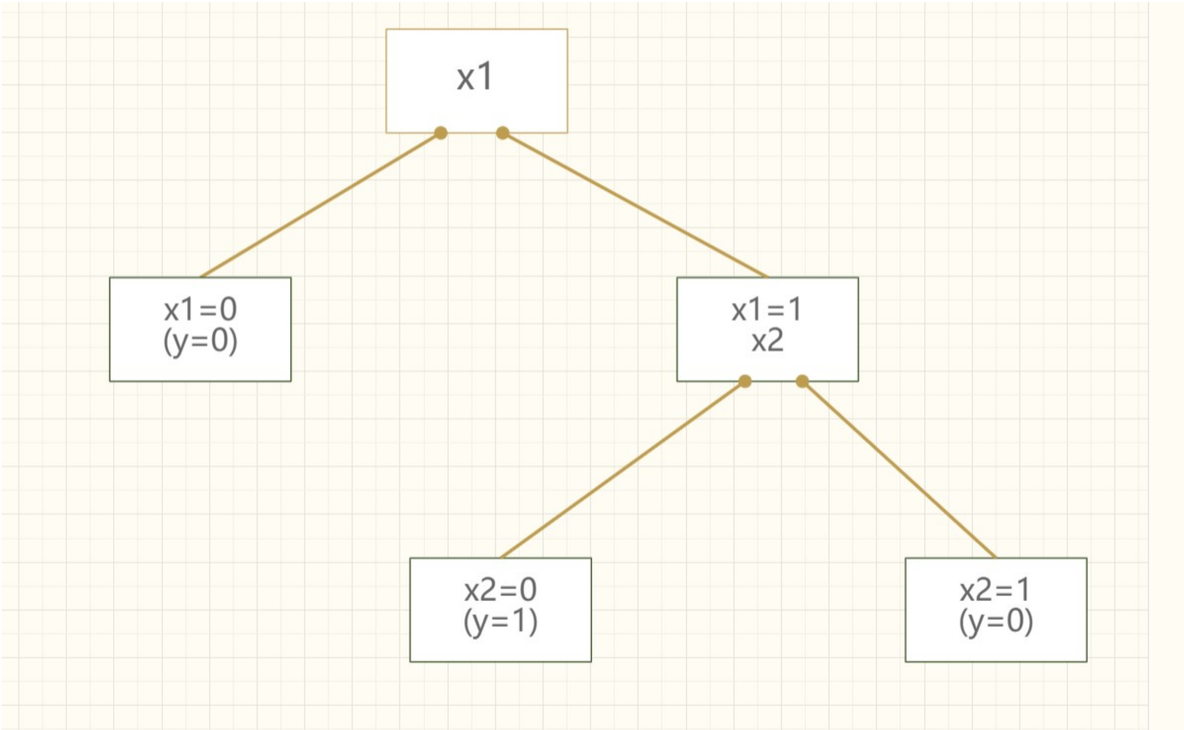
3) 构造决策树：

由于 x_1 拥有最高的信息增益，将其选做根节点

若 $x_1=1$ ，则 $(x_2,x_3,y)=(1,1,1),(0,0,1),(1,0,0)$

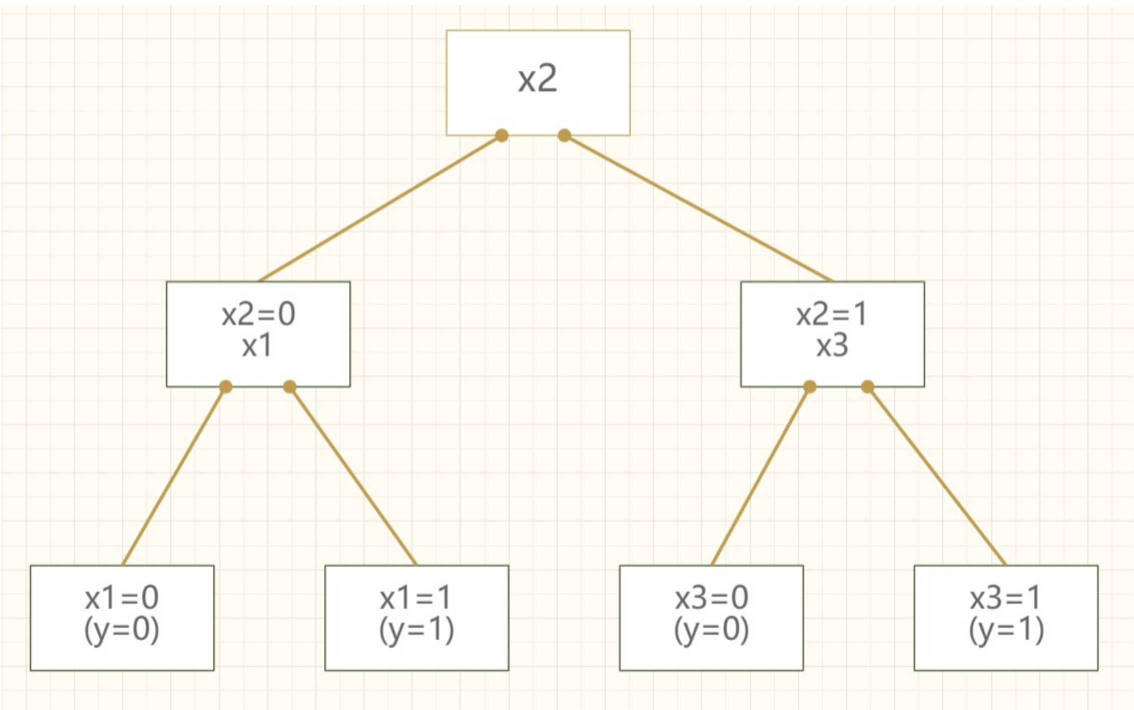
若 $x_1=0$ ，则 $(x_2,x_3,y)=(0,1,0)$

由于 x_2 和 x_3 具有相同的信息增益，可任选一节点，此处选择 x_2 作为下一个节点，则有：



由于需要基于信息增益准则的ID3算法构造决策树，该决策树的根节点固定为 x_1 ，因此根节点的一个节点（ $x_1=0$ ）无法继续分裂，即该深度为2的决策树最多只能分裂出3个叶子节点。对于4个样本的数据集，易知至少有一个训练样本分类错误。

2.



3 线性模型与AlphaGoZero

1. **计算对于一组B 条训练样本，价值模型的损失函数L_{value} 对参数w_v, b_v 的梯度（写出计算过程）**
   考虑损失函数 L 中的值损失 L_{value} ：

$$L_{value} = (x - v)^2 = \frac{1}{B}(z - v)^T(z - v)$$

设$v = tanh(Xw_v + b_v 1_B)$，对w_v计算梯度：

$$\begin{aligned}\frac{\partial L_{value}}{\partial w_v} &= \frac{\partial}{\partial w_v}\left[\frac{1}{B}\sum_{i=1}^{B}(z_i - v_i)^2\right] \\ &= -\frac{2}{B}\sum_{i=1}^{B}(z_i - v_i)\frac{\partial v_i}{\partial w_v}\end{aligned}$$

其中 $v_i = tanh(X_I w_v + b_v)$ ，根据链式法则 $\frac{\partial v_i}{\partial w_v} = (1 - v_i^2)X_i$ ，有：

$$\frac{\partial L_{value}}{\partial w_v} = -\frac{2}{B}\sum_{i=1}^{B}(z_i - v_i)(1 - v_i^2)X_i$$

对于 b_v ，同理有：

$$\begin{aligned}\frac{\partial L_{value}}{\partial b_v} &= \frac{\partial}{\partial b_v}\left[\frac{1}{B}\sum_{i=1}^{B}(z_i - v_i)^2\right] \\ &= -\frac{2}{B}\sum_{i=1}^{B}(z_i - v_i)\frac{\partial v_i}{\partial b_v} \\ &= -\frac{2}{B}\sum_{i=1}^{B}(z_i - v_i)(1 - v_i^2)\end{aligned}$$

2. **参考策略模型的相关实现，补全 model/linear_model.py 中空缺的价值模型损失函数和梯 度计算，用 check_grad 函数验证梯度计算是否正确。**
   具体见 .code
   验证结果如下：

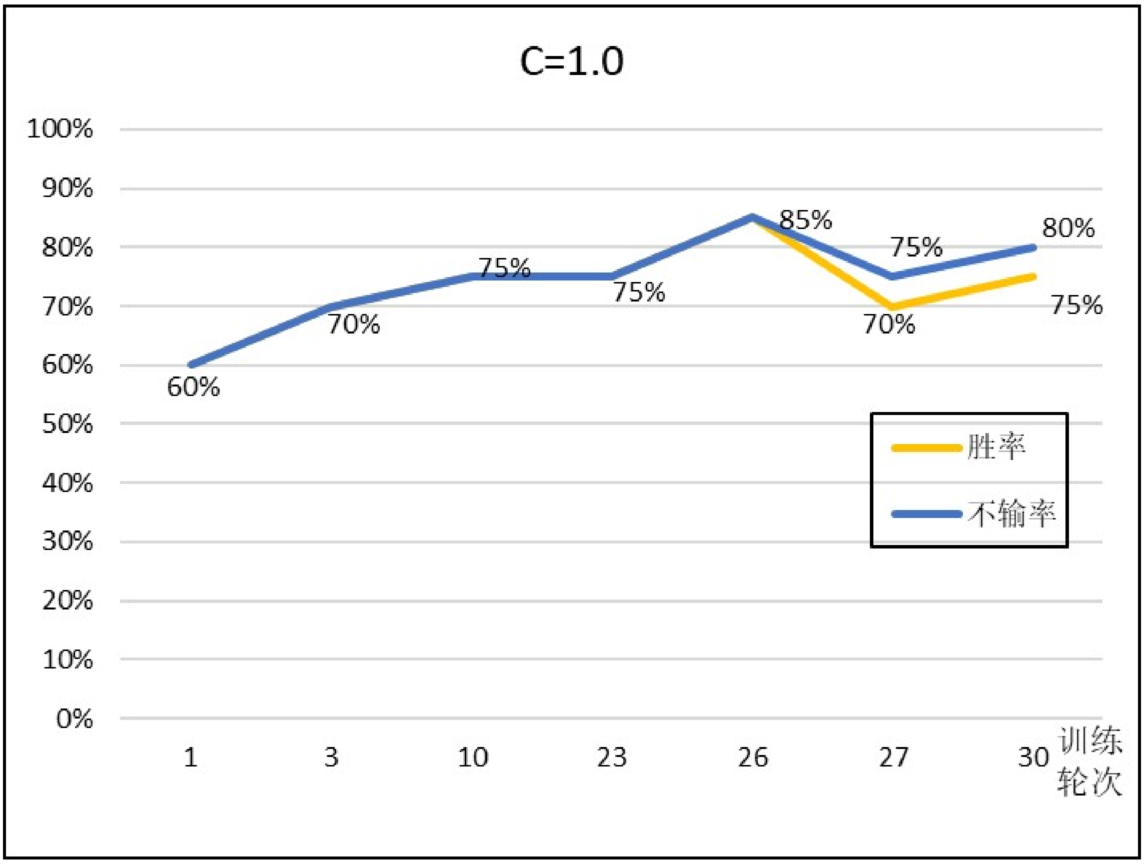
```
C:\Users\lkx20\Desktop\课程\大三下\人智导\AI2025_HW2\code>python -m model.linear_model
[Gradient of w_v] shape:(247,)        Mean Error: 0.000001, Max Error: 0.000001
[Gradient of b_v] shape:(1,)          Mean Error: 0.000000, Max Error: 0.000000
[Gradient of w_pi] shape:(247, 49)    Mean Error: 0.000000, Max Error: 0.000000
[Gradient of b_pi] shape:(49,)        Mean Error: 0.000000, Max Error: 0.000000
```

3. **补全 mcts/puct_mcts.py 中空缺的内容，实现使用 PUCB 的 MCTS 算法。**
   具体见 .code
4. **补全 alphazero.py 中空缺的内容，并选取适当的参数，进行AlphaGoZero训练，绘制训练过程中对Random Player 的胜率/不输率变化的折线图，并汇报你选取的参数。**
   补全alphazero.py空缺内容具体见.code
   选取训练参数如下：

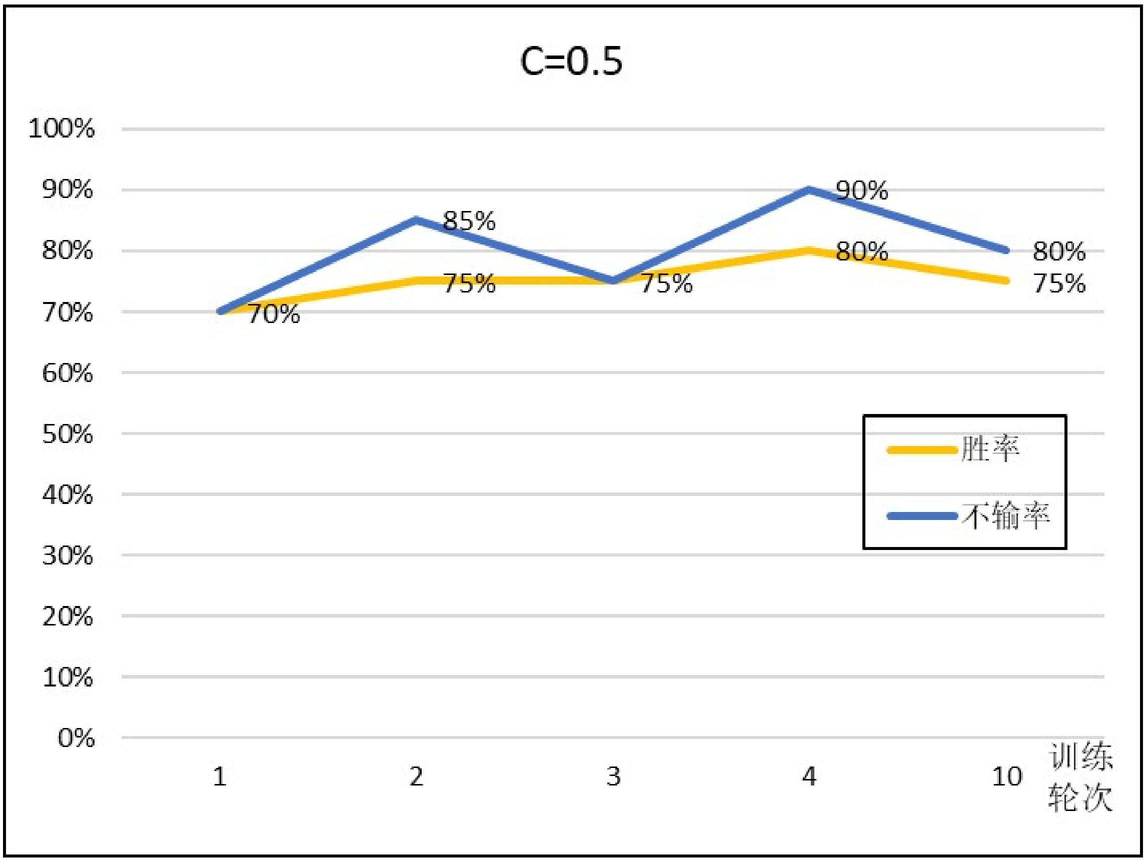
```
    config = AlphaZeroConfig(
        n_train_iter=30,
        n_match_train=10,
        n_match_update=20,
        n_match_eval=20,
        max_queue_length=160000,
        update_threshold=0.501,
        n_search=200,
```

```
        temperature=1.0,
        C=1.0,
        checkpoint_path="checkpoint/linear_7x7_exfeat_norm_p1"
    )
    model_training_config = NumpyModelTrainingConfig(
        epochs=5,
        batch_size=126,
        lr=0.0001,
        weight_decay=0.01
    )
```

对`Random Player`的胜率/不输率变化的折线图（已去除`REJECT`轮次）

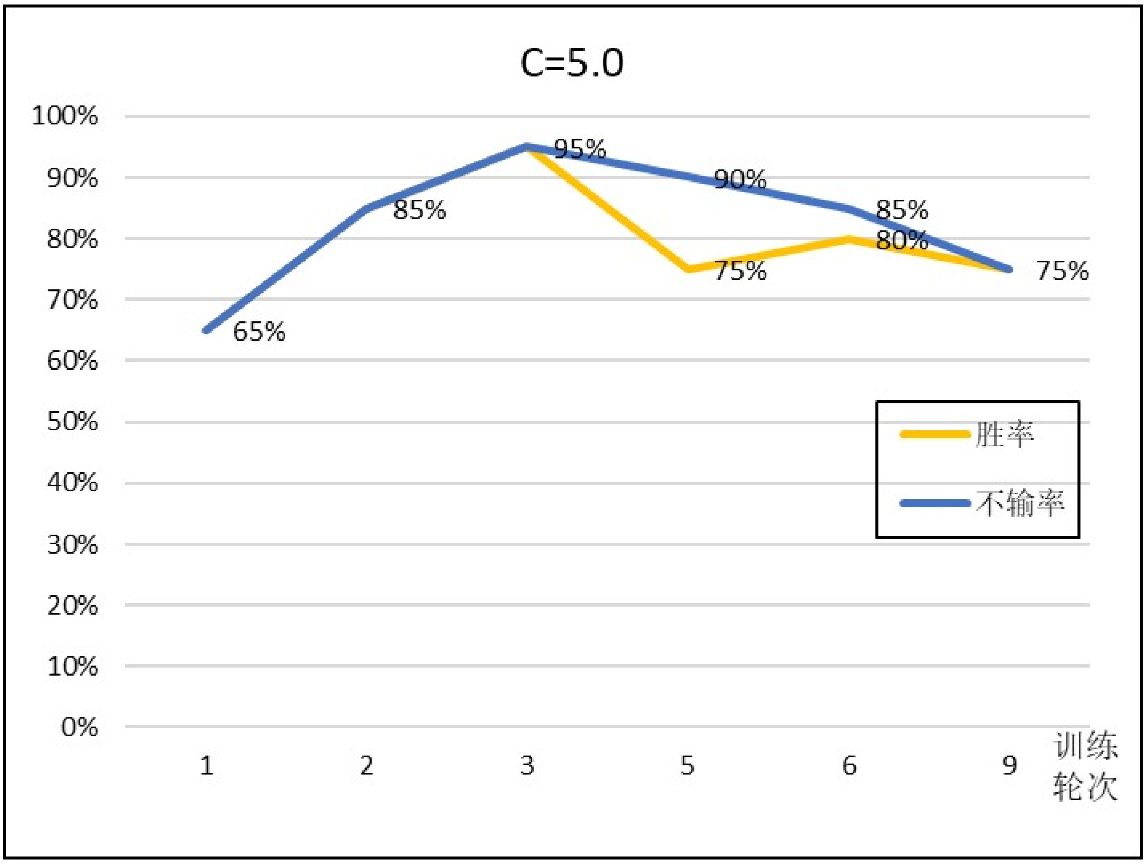


5. **改变`C`或`n_search`的值，同样绘制对`RandomPlayer`的胜率/不输率变化的折线图，分析该参数是如何影响模型训练和最终胜率的。（二选一完成即可）。**
   改变`C`的值，在`C=0.5`和`C=5.0`时分别进行训练：
   1) `C=0.5`，其余参数同4：
   减小C值会使得模型倾向于选择已知的高回报路径，而避免过多探索未知路径。因此，训练中会快速收敛（本实验中第一轮训练即得到了较好结果，后续胜率变化不显著），但也有可能会陷入局部最优，无法探索到全局最优策略，使得最终胜率较C值大者低（例如与本实验中C=5.0比较）。



2) `C=5.0`，其余参数同4：

增大C值会使得模型更倾向于探索所有的潜在可能。因此，训练过程将会更慢收敛，且训练过程将会更加不稳定，但更可能找到全局最优解，得到更高的最终胜率。



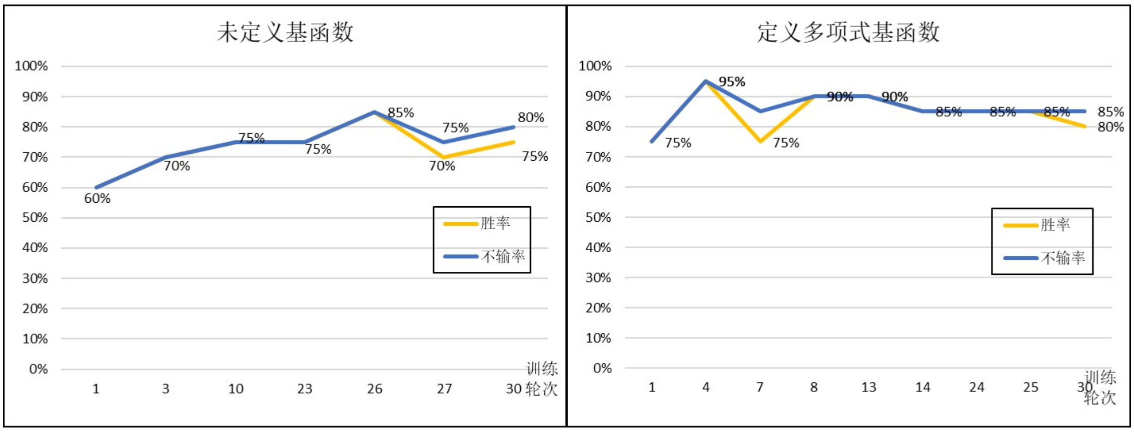
6. **[附加题]给线性模型增加一个基函数，分析你选取的基函数对于模型学习棋盘局面的策略和价值有什么好处，并通过实验验证其是否有效。**

   由于线性模型只能捕捉线性关系，增加多项式基函数可以帮助模型捕捉围棋局面中的非线性关系，提高模型的表达、泛化能力从而更精确地拟合复杂的棋盘布局，有效提高策略预测的准确性并避免

过拟合。
修改linear_model.py文件，增加多项式基函数定义并使用，基函数定义代码如下：

```python
# 定义多项式基函数
def polynomial_base_function(X, degree=2): # 本次测试多项式次数为2
    X_poly = [X]
    for i in range(2, degree + 1):
        X_poly.append(X ** i)
    return np.hstack(X_poly)
```

同样参数条件下，对比训练数据如下：



通过实验表明，新增多项式基函数后模型表现更好（最终胜率更高），且在训练过程中收敛更快、训练初期表现出较好的学习效果，说明多项式基函数能够帮助模型更有效地学习棋盘局面