# 将人工智能应用于棋类游戏开发中的一般步骤

目录

[将人工智能应用于棋类游戏开发中的一般步骤 1](#_Toc6652175)

[**一、摘要：** 1](#_Toc6652176)

[**二、三子棋** 1](#_Toc6652177)

[**三、游戏界面编写** 2](#_Toc6652178)

[**四、人工智能** 3](#_Toc6652179)

[**五、机器学习** 3](#_Toc6652180)

[**1、定义** 4](#_Toc6652181)

[**2、设计** 4](#_Toc6652182)

[**2.1、选择训练经验** 5](#_Toc6652183)

[**2.2、选择目标函数** 5](#_Toc6652184)

[**2.3、选择目标函数的表示** 6](#_Toc6652185)

[**2.4、选择函数逼近算法** 12](#_Toc6652186)

[**六、应用程序架构** 12](#_Toc6652187)

[**1．域名：tictactoe.js.org** 12](#_Toc6652188)

[**2. 系统类型：纯静态网站** 12](#_Toc6652189)

[**2.1 ReactJs** 12](#_Toc6652190)

[**2.2 Ant Design** 13](#_Toc6652191)

[**3. 源代码** 13](#_Toc6652192)

[**3.1 目标函数的表示** 13](#_Toc6652193)

[**3.2 检查棋盘的边** 14](#_Toc6652194)

[**3.3 函数逼近算法** 15](#_Toc6652195)

[**4. CI/CD：** 15](#_Toc6652196)

[**七、评估以及后续工作：** 15](#_Toc6652197)

[**简化** 15](#_Toc6652198)

[**复杂化** 15](#_Toc6652199)

[**趣味性** 15](#_Toc6652200)

[**八、参考文献：** 16](#_Toc6652201)

**一、摘要：**

2016 年 3 月，AlphaGo 【1】击败世界围棋冠军李世石，使得人工智能引起人们的广泛讨论。如今，人工智能创业潮不断涌现，这个话题不仅活跃于科技教育界，甚至进入了街头巷尾，成为了人们的日常谈资。

各种人工智能框架的出现，也大大降低了人工智能编程的门槛。比如 tensorflow 不仅开源，还提供了一个在线训练人工神经网络的界面：[https://playground.tensorflow.org](https://playground.tensorflow.org" \t "_blank)【2、3】。这使得开发人员可以利用各种强大的工具，给自己的应用赋能，让传统的应用程序变得“聪明”。

通过使用这些工具，开发人员不需要理解人工智能的根本原理，知其然而不知其所以然。即如果需要从 0 开始，编写具有智能特性的程序，往往不知道如何下手。原因是这些框架本身是为了解决实际的应用问题而产生，并不是用来学习人工智能的知识和思想的。

所以本文尝试通过一个实际的例子（<https://tictactoe.js.org/>），展示人工智能程序开发的一般步骤。出于演示目的，本文选择了三子棋（tic-tac-toe）这个最简单的棋类游戏来举例。简化，也是学习根本原理的通用方法，通过简化的问题和模型排除不必要的干扰，让人看清原理。在掌握了原理之后，即可推而广之，解决更复杂的问题。

本文展示的一般步骤，不仅可以应用于棋类游戏开发，而且是所有人工智能程序开发的一般步骤。当然，复杂的程序步骤会更繁杂，但仍然会以本文列出的一般步骤为主干。

**二、三子棋**

三子棋，起源于古希腊【4】，也称为井字棋，或者圈圈叉叉。因为棋盘是一个九宫格，或者说像汉字中的“井”字，所以得名井字棋；又因为一个玩家通过打圈，另一个玩家打叉来玩这个游戏，所以得名为圈圈叉叉。本文称其为三子棋，是因为在这个游戏中，当玩家的棋子能够三子成线时，即为获胜，故名三子棋。

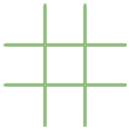


图 2.1：三子棋盘

https://uploader.shimo.im/f/RFHSIczyz6sXmvmN.png!thumbnail

图 2.2：一个 X 方取胜的例子

三子棋是最简单的棋类游戏，它只有 765 个可能局面，26830 个棋局。如果要实现一个人机对战的三子棋游戏，由于棋局的可能性并不多，完全可以使用穷举法做到。但是这样做，代码量并不少，而且对人类玩家来说，由于机器对手每次策略都一样，所以趣味性不高。

本文展示的人工智能程序的编写，不仅大大缩减了代码量（因为不需要存储取胜策略），而且由于程序的学习是一个动态过程，对于人类玩家的同样的走法，程序可能会给予不同的回应，这样下棋的趣味性更高。

**三、游戏界面编写**

游戏界面不涉及人工智能部分，但却是玩家必须要通过界面来与程序交互。本文选择了网页版界面，无需安装即可使用，而且便于传播。

要编写优秀的网页界面，有很多优秀的开发框架可供选择。本文采用了 ReactJs【5】，不仅因为它是目前最优秀的前端框架之一，而且它的官方教程恰好提供了一个详细的编写井字棋游戏的教程【6】。

按照教程编写出来的界面是一个可以运行的井字棋游戏，可以由两人交替走子：<https://codepen.io/gaearon/pen/gWWZgR?editors=0010>。

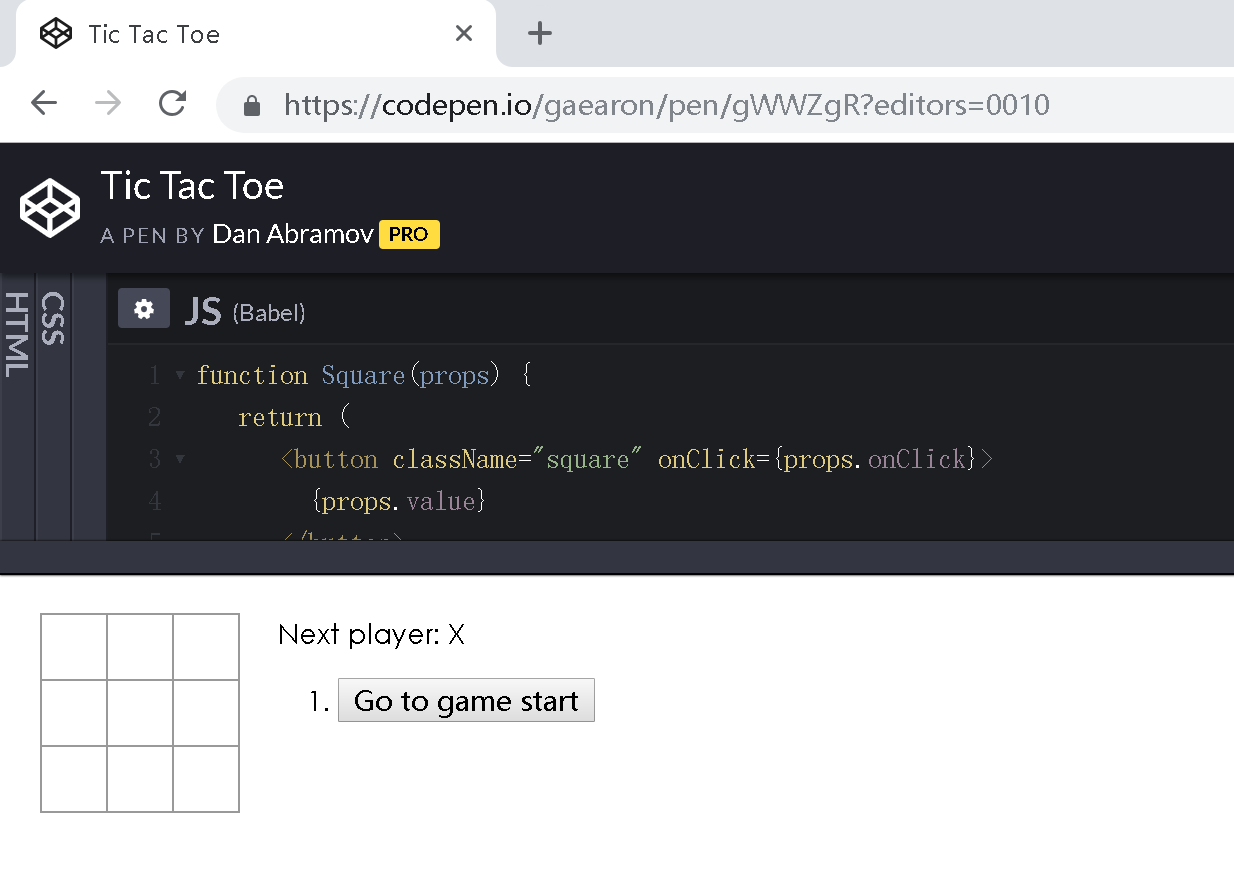


图3.1：按照 ReactJs 官方教程编写的游戏界面

**四、人工智能**

据维基百科【7】解释，人工智能（英语：Artificial Intelligence，缩写为AI）也称为机器智能，指由人制造出来的机器所表现出来有智能。通常人工智能是指通过普通计算机程序来呈现人类智能的技术。

本文聚焦在这个定义的后一段，即通过编写一个普通的计算机程序来呈现人类智能。

人工智能的研究课题非常之多，其中之一是机器学习，本文所展示的程序，更精确地说，是一个机器学习程序。本文所要展示的程序开发步骤，也即机器学习程序的开发步骤。

**五、机器学习**

机器学习是人工智能的一个分支【8】。人工智能的研究历史有着一条从以“推理”为重点，到以“知识”为重点，再到以“学习”为重点的自然、清晰的脉络 。显然，机器学习是实现人工智能的一个途径，即以机器学习为手段解决人工智能中的问题。

**1、定义**

通常有以下几种定义：

* 机器学习是一门人工智能的科学，该领域的主要研究对象是人工智能，特别是如何在经验学习中改善具体算法的性能。
* 机器学习是对能通过经验自动改进的计算机算法的研究。
* 机器学习是用数据或以往的经验，以优化计算机程序的性能标准。

一种更精确的教科书式的定义如下：

**对于某类任务 T 和性能度量 P，如果一个计算机程序在 T 上以 P 衡量的性能随着经验 E 而自我改善，那么我们称这个计算机程序在从经验 E 中学习。**

比如具体到我们的井字棋游戏，就是：对于学习下井字棋的计算机程序，它可以通过和自己下棋获取经验；它的任务是参与井字棋对弈；它的性能用它赢棋的能力来衡量。

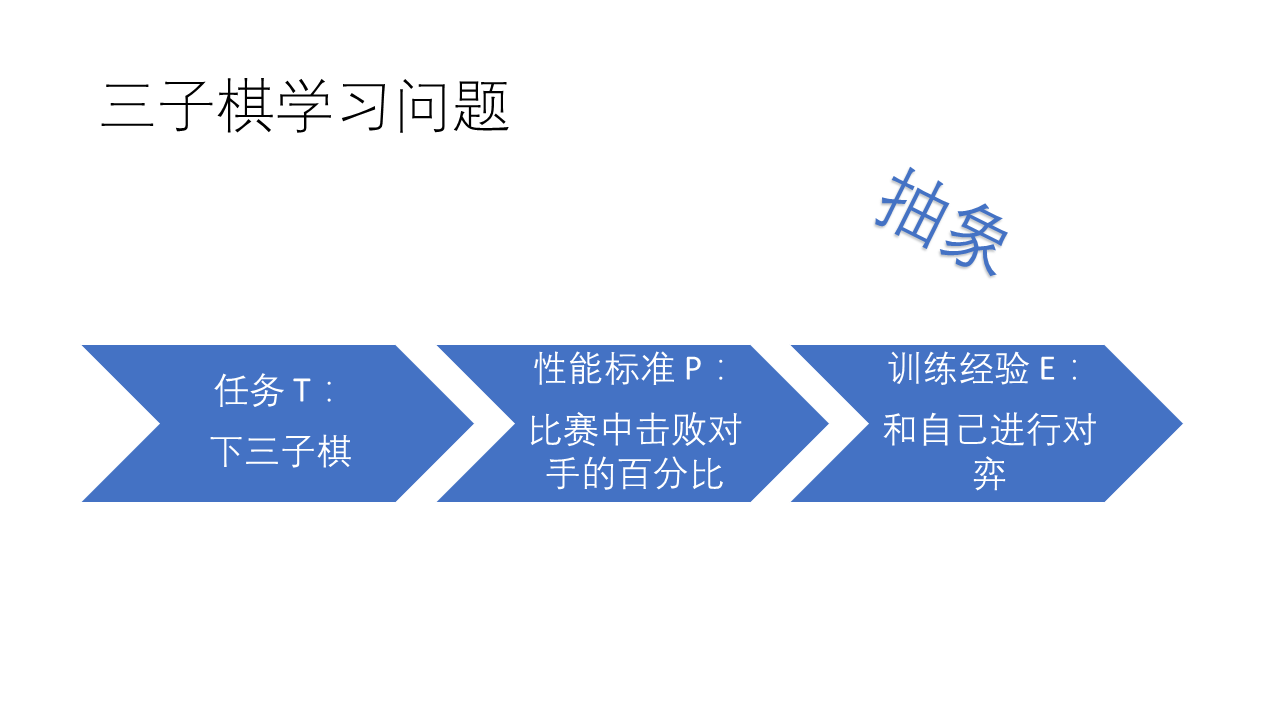


图 5.1：三子棋学习问题的抽象

**2、设计**

那么如何来设计一个机器学习系统呢？一般有以下几个步骤：



图 5.2：设计学习系统的一般步骤

**2.1、选择训练经验**

本程序选择了两种训练经验：

* 产生随机合法的棋局：在本程序对外公开访问之前，可以通过让程序与一些随机产生的合法棋局对弈，产生初级的智能，然后再公开给用户访问。这样比直接公开要好一些，因为直接公开，程序完全不具有智能，会让用户失去兴趣。
* 和玩家对弈：在程序对外公开后，随着玩家与程序的对弈，程序继续不断调整自身内部的评估函数（目标函数）参数的权值。玩家在与程序对弈的过程中，可能会感觉程序越来越聪明，从而更加激发玩家的好奇与兴趣。

当然，除了以上两种训练经验，还可以选择其他的训练经验，比如一个使用固定评估函数走子的棋局产生器。

**2.2、选择目标函数**

目标函数，也就是机器学习程序的评估函数，对程序的性能表现进行打分。大致地说，这个目标函数应该对好的表现打高分，对差的表现打低分。

具体到本三子棋程序，对于集合 B 中的任意棋局状态 b，我们定义如下目标函数 V(b)：

* 如果 b 是一最终的胜局，那么 V(b) = π/2；
* 如果 b 是一最终的负局，那么 V(b) = -π/2；
* 如果 b 是一最终的和局，那么 V(b) = 0；
* 如果 b 不是最终棋局，那么 V(b) = V(b')，其中 b' 是从 b 开始双方都采取最优对弈后可达到的终局。（这里出现一个递归逻辑， 所以需要后面的一步：函数逼近算法来计算这个函数值）

当然，你也可以选择其他的目标函数，比如，如果是最终胜局，令 V(b)=100；如果是负局，令 V(b)=-100 等。如果是和局，令 V(b)=0 显然是一个合适的选择。那么这里为什么选择了胜局时，令 V(b)=π/2 呢？

因为这里做了一个假设：即棋局的表现，在达到终局前，是介于胜局与负局之间的，如果用一个连续函数来表示这种趋势（向胜局或者向负局的趋势），应该是如下图所示的一个曲线：

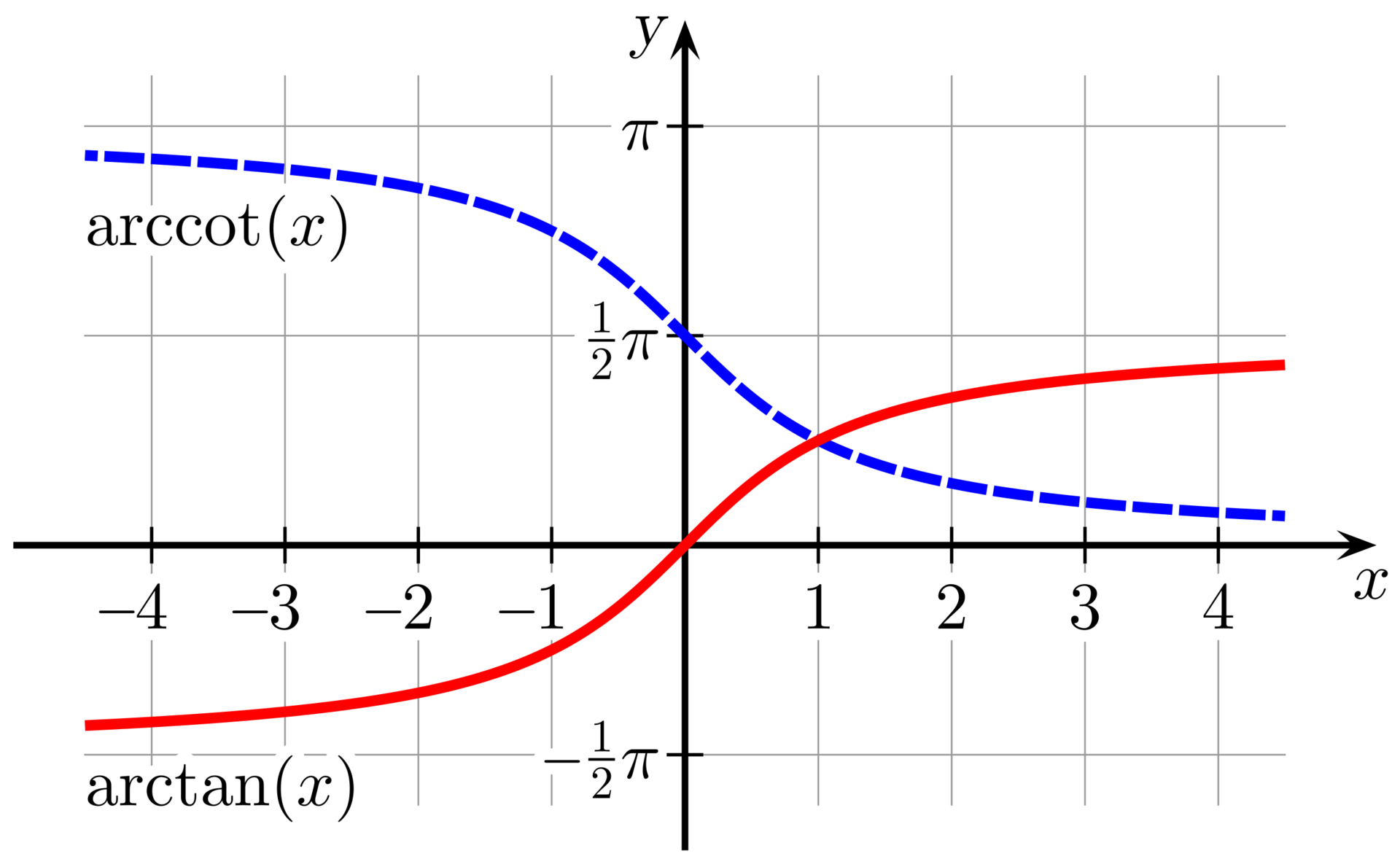


图 2.2：arctan 函数图形（红色曲线）

这很自然让人联想到 y=arctan(x) 这个反三角函数，而这个函数正好介于 -π/2 与 π/2 之间，故采用它们分别做为负局和胜局的表现得分，这样就省去了做函数变换的麻烦。

**2.3、选择目标函数的表示**

有了目标函数，如何将它表示出来，即如何将它表示成一系列自变量的函数形式呢？

首先，目标函数是受棋局的状态所影响的，所以自变量肯定是棋局上的状态。其次，不同的状态对于目标函数的影响是不同的，即不同的自变量，其权重不同。

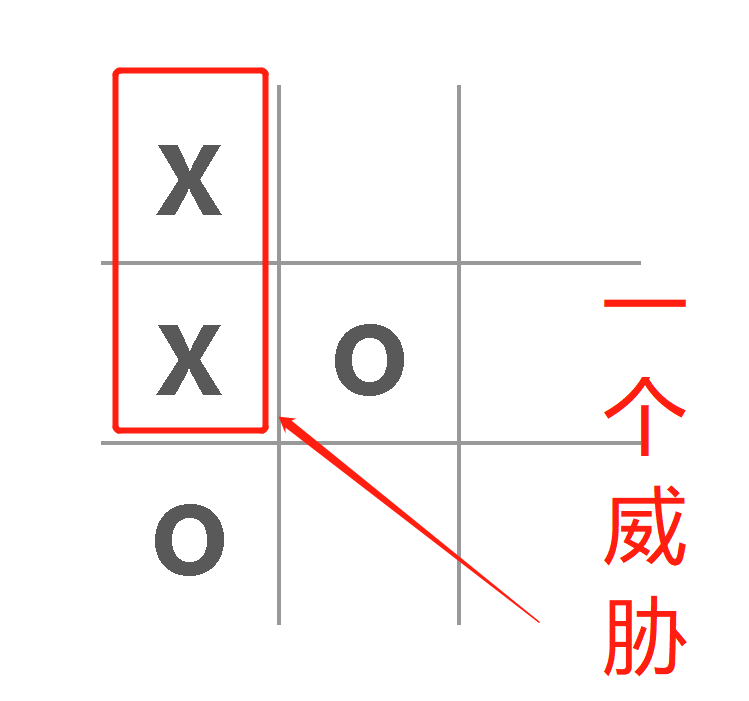
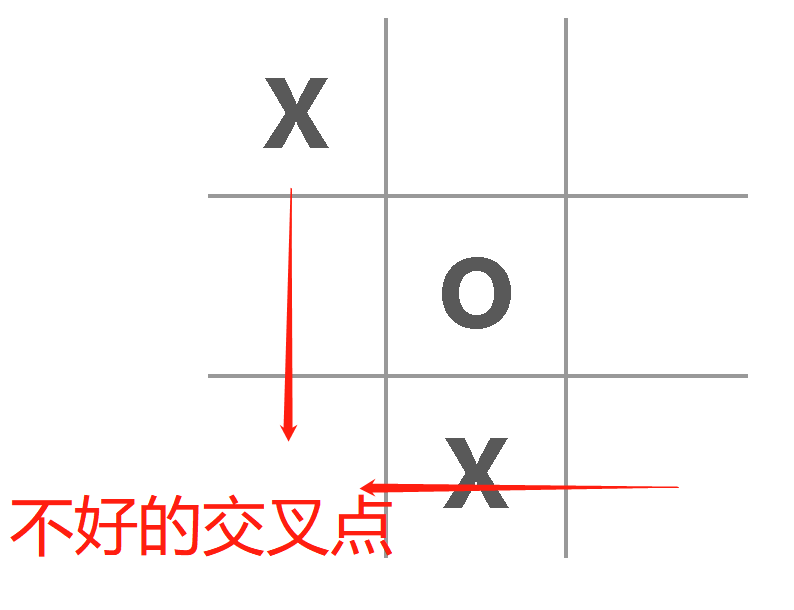
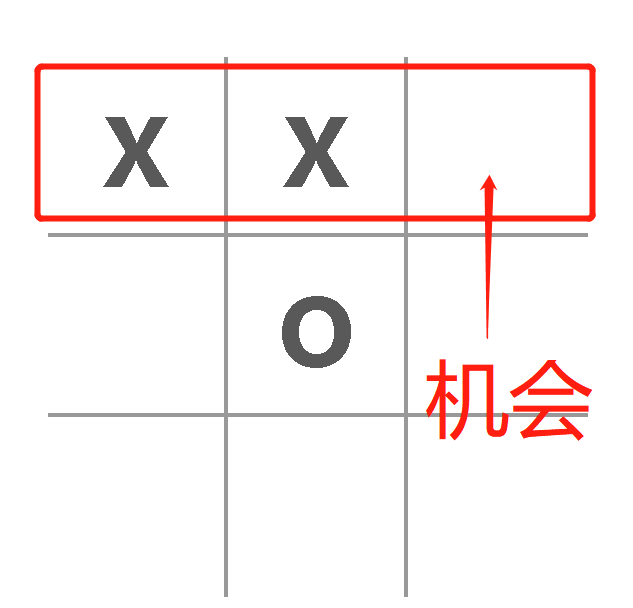
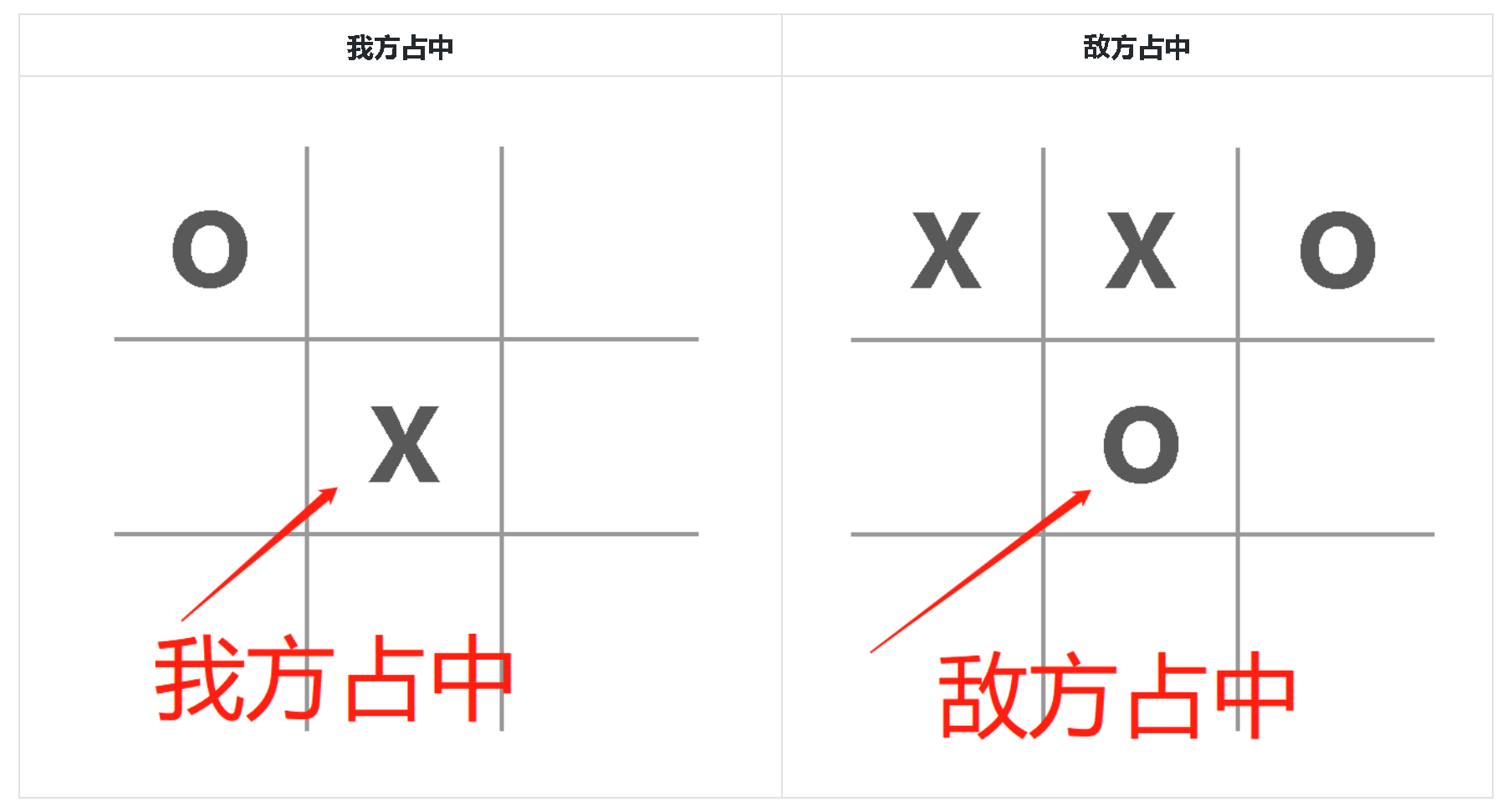
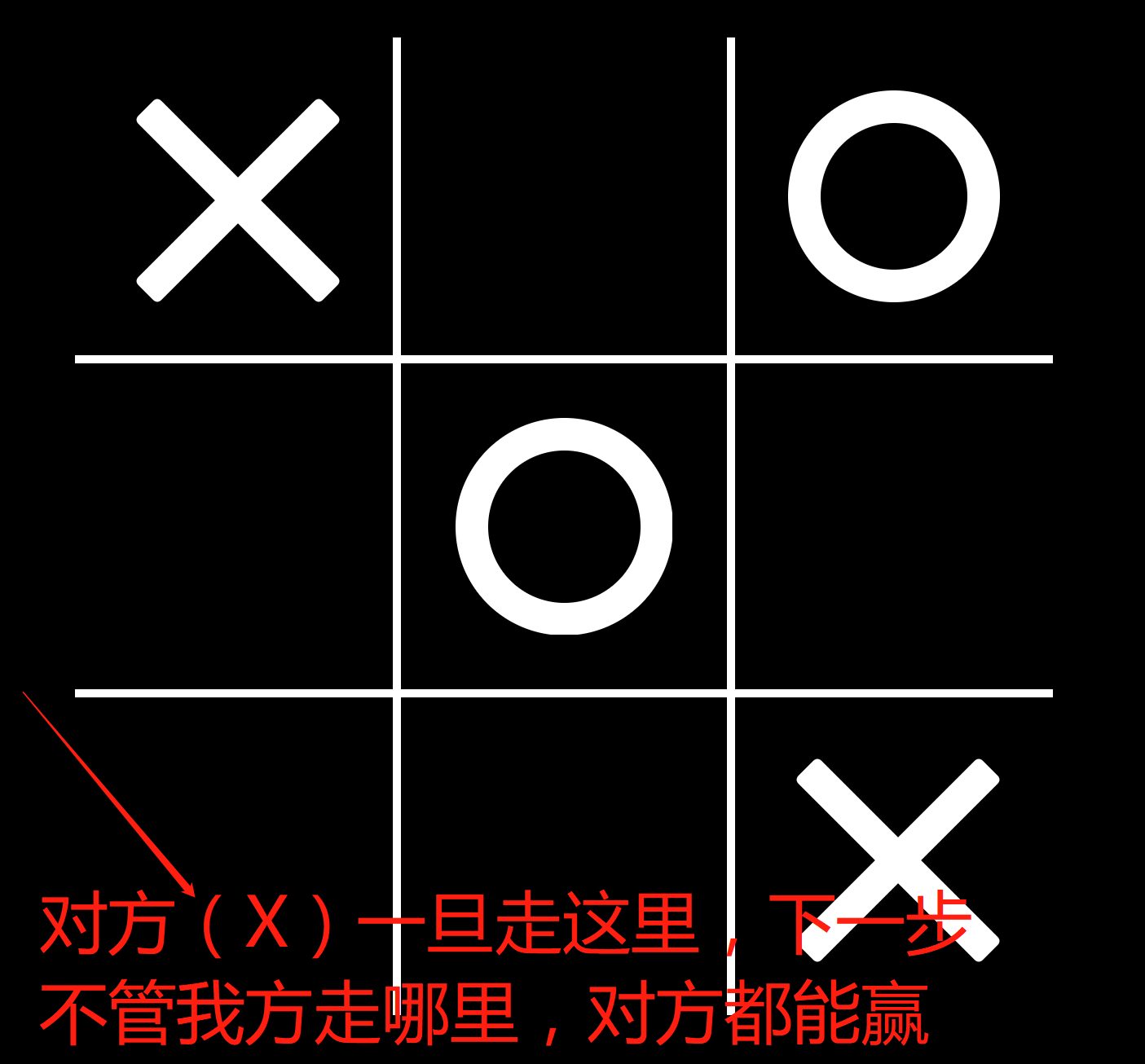
如何找到合适的自变量，以及将目标函数表示成这些自变量的什么函数，没有一定的标准。但是一旦确定了这个目标函数的表示，那么机器学习的过程，就转化成了一个函数调参的过程。

通过改变不同自变量的权重，将目标函数计算出来的值逼近给定的实验数据（即经验集合），即是机器学习的过程。

本学习程序把 V(b) 表示成对一个线性函数求反正切值（线性函数单调，但是无界。通过求反正切值，使得这个函数值满足了我们的有界性要求）： V(b)=arctan(w0+w1x1+w2x2+w3x3+w4x4+w5x5)，其中 w0 到 w5 为数字系数，或者叫做权，由学习算法来选择。而 x1 到 x5 为棋盘状态值，权 w1 到 w5 决定了不同的棋盘特征的相对重要性，而权 w0 为一个附加的棋盘状态值常量。

再次强调，自变量的选择并没有一定的标准，这里选择的 5 个自变量，是经过一些不同的实验之后，最后总结出来的可以达到很好效果但是数量又比较少的一个变量组合。有没有可能选择其他的变量组合来得到更好的效果，还值得进一步探讨。

这里的 5 个变量分别是：

* x1：棋盘上受到对方威胁的边数（一共 8 条边，如果一条边上出现两颗敌子并且没有我方棋子，那么这条边形成了对我方的一个威胁），取值范围是 0 到 8。 图 2.3.1： 棋盘上对我方形成的一个威胁示意图
* x2：坏交叉点数：如果两条边上各有一颗敌方棋子而没有我方棋子，并且这两条边交叉，那么这个交叉点称为坏交叉点（不好的交叉点）。 图 2.3.2： 坏交叉点示意图
* x3：机会边数：如果有一条边有我方两颗棋子而没有敌方棋子，那这条边是一条机会边。 图 2.3.3：机会边示意图
* x4：占中优势：如果中间格子是我方棋子，那么我方占据优势，x4 记为 1；如果是空，则记为 0；如果是敌方棋子，则记为 -1。对三子棋稍作了解就会知道，中间格子的棋子更容易与其他格子里的棋子连成一条线，故谁先占据中间格子，谁在一定程度上就占据了一些优势。 图 2.3.4：占中示意图
* x5：我方的赢面机会带来的威胁数。如果我方某边上已有两颗棋子，另一格子为空。如果我方占据这个空格就能赢，但是现在轮到对方走。如果对方占据这个空格，不仅阻碍了我方赢棋的可能，还同时让两条边以上拥有敌方的两颗棋子而另外一格是空的这种情况，那么下一步不管我方落子在哪格，对方总能赢，这就带来了威胁。 图 2.3.5：我方赢面机会所带来的潜在威胁

x1 到 x4 都很简单直观，而 x5 则是高手走法，通过前面的铺垫，在某一步，落一子而给对方造成两条边上的威胁。而在下一步中对方最多只能防一条边。

回顾：针对这个具体的三子棋学习任务，我们把它转换成了一个学习目标函数表示中的系数（或说参数）w0 到 w5 的值的问题（调参问题）。

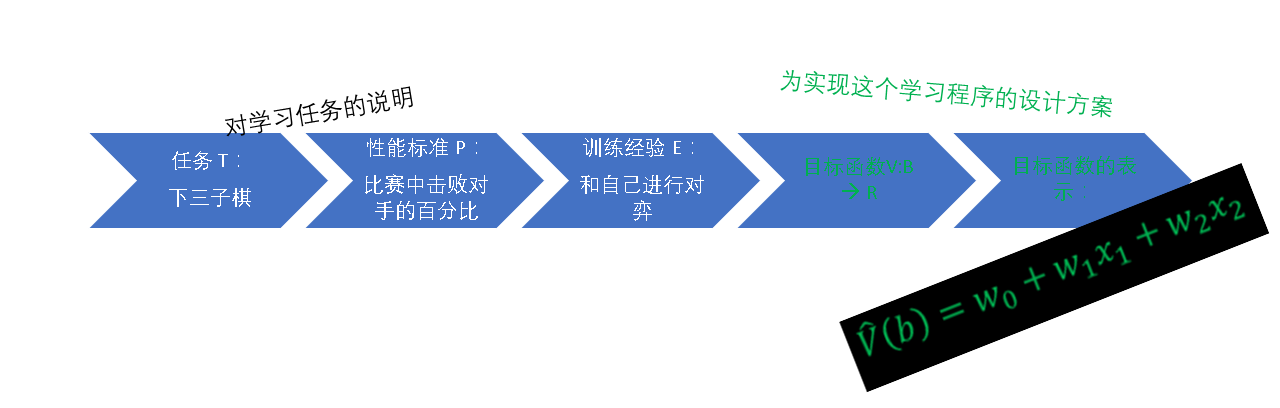


图 2.3.6：问题的转换

**2.4、选择函数逼近算法**

这个过程也被称作估计训练值，就是选择权重值的过程。我们可以从任意的权重值开始，然后不断调整权值，直到找到一个比较好的权重值组合。

本程序采用了 LMS 最小均方法，对于每一个训练样例，它能把权值向减小这个训练数据误差的方向略为调整。

详细地说，LMS 权值更新法则是这样的【9】：

对于 每一个训练样例<b, Vtrain(b)>

* 使用当前的权计算
* 对每个权值进行如下更新：

这里是一个小的常数（本示例程序采用的值是0.1），用来调整权值更新的幅度。直观地说，当误差为0时，权不会被改变。当为正时（例如，当太低时），每一个权值会根据其对应的特征值增加一定的比例。这会提升的值而减小误差。注意，如果某个参数为0，那么它的值不会因这个误差而改变，这样就使只有那些在训练样例的棋局中确实出现的特征的权值才被更新。在一定的条件下，这种简单的权值调整方法被证明可以收敛到值的最小误差平方逼近。

**六、应用程序架构**

**1．域名：tictactoe.js.org**

采用了 js.org 的二级域名 tictactoe，即三子棋的英文名称。Js.org 提供免费的二级域名，尤其乐意为 js 应用提供服务，而本应用正好是一个纯 js 应用。

**2. 系统类型：纯静态网站**

托管在 Github Pages 上。Github Pages 非常适合托管静态网站，而且是免费的。

**2.1 ReactJs**

第三章已经说明，本程序的游戏界面基于 ReactJs 的经典教程。ReactJs是一款用来构建用户界面的优秀的JavaScript库，是由Facebook出品的开源框架。

**2.2 Ant Design**

Ant Design是阿里推出的开源React组件库，包含了丰富的界面交互组件以及优美的设计风格。

**3. 源代码**

完整的源代码托管在 github 上：<https://github.com/Jeff-Tian/tic-tac-toe-ai>。这里重点介绍一下前面几章介绍的机器学习的关键步骤代码：

**3.1 目标函数的表示**

程序里使用目标函数来对棋局打分：

getBoardScore: **function** (bitmap, weights) {  
 weights = weights || Strategy.*getInitialWeights*();  
  
 **let** {lost, win, *factors*} = Strategy.*getBoardStatus*(bitmap);  
  
 **if** (lost) {  
 **return** {  
 factors: *factors*,  
 **namedFactors**: Strategy.*getNamedStrategy*(*factors*),  
 **total**: -***Math***.**PI** / 2  
 }  
 }  
  
 **if** (win) {  
 **return** {  
 factors: *factors*,  
 **namedFactors**: Strategy.*getNamedStrategy*(*factors*),  
 **total**: ***Math***.**PI** / 2  
 }  
 }  
  
 **let** score = ***Math***.atan(*factors*.map((s, i) => s \* weights[i]).reduce((prev, next) => prev + next, 0));  
  
 **return** {  
 factors: *factors*,  
 **namedFactors**: Strategy.*getNamedStrategy*(*factors*),  
 **total**: score  
 };  
}

求出的分数即 score，注意这个分数是x1到x5与权重的加权和再求arctan，从而让分数单调有界。2.3详细地说明了x1到x5这5个自变量，在代码里，将它们放在了一个factors数组里。这个factors 是由*getBoardStatus*来计算的，这里抽象成一个方法，好处是，如果要换自变量，那么这个打分的函数不用改变。如前所述，自变量的挑选，并不是确定的。

**3.2 检查棋盘的边**

上面的代码中有一个重要的函数：*getBoardStatus*，它的核心是去检查棋盘的边，得到棋盘的状态。在程序里，棋盘被表示成一个长度为9的数组，每个元素要么是1、要么是0、要么是-1。分别表示当前宫格里是我方的棋子，为空以及敌方的棋子。棋盘一共8条边，检查棋盘的边的函数是这样的：

**function** *checkSides*(bitmap) {  
 **let** d = 0;  
 **let** dead = 0;  
 **let** w = 0;  
 **let** c = 0;  
  
 **for** (**let** i = 0; i < sides.**length**; i++) {  
 **let** side = bitmap.filter((\_, j) => sides[i].indexOf(j) >= 0);  
  
 **let** negatives = side.filter(b => b === -1);  
 **let** zeros = side.filter(b => b === 0);  
 **let** ones = side.filter(b => b === 1);  
  
 **if** (negatives.**length** === 2 && zeros.**length** === 1) {  
 d++;  
 }  
  
 **if** (negatives.**length** === 3) {  
 dead++;  
 }  
  
 **if** (ones.**length** === 3) {  
 w++;  
 }  
  
 **if** (ones.**length** === 2 && zeros.**length** === 1) {  
 c++;  
 }  
 }  
  
 **return** {**danger**: d, **lost**: dead, **chance**: c, **win**: w};  
}

**3.3 函数逼近算法**

函数逼近的过程，也即学习的过程。这里表现为一个名字叫learn()的函数。这个函数接收两个参数，即上一棋局，以及当前棋局。注意之前2.2 提到，函数逼近算法，是不断用当前的权值，对新出现的棋局打分来估计前一棋局的分数。所以这个函数会使用同样的权重对前后两个棋局打分，并通过这个分数的差值，采用LMS方法来调整权重：

learn(lastSquares, currentSquares) {  
 **if** (!lastSquares) {  
 **return**;  
 }  
 **let** estimatedLastScore = Judger.getBoardScore(lastSquares, **this**.**weights**);  
 **let** actualScore = Judger.getBoardScore(currentSquares, **this**.**weights**);  
 **let** diff = actualScore.**total** - estimatedLastScore.**total**;  
  
 **for** (**let** i = 0; i < estimatedLastScore.factors.**length**; i++) {  
 **this**.**weights**[i] = **this**.**weights**[i] + 0.1 \* diff \* estimatedLastScore.factors[i];  
 }  
}

**4. CI/CD：**

采用了 Travis CI系统，只需要提交代码，Travis 会自动运行测试、分析代码。然后发布至 Github pages。

**七、评估以及后续工作：**

本程序很好地演示了一个最简单的人工智能程序的编写步骤，不需要任何复杂的AI框架，可以从0开始编写。不过，仍然存在优化的可能性，比如：

**简化**

对于目标函数的表示，有没有更好的或者更简洁的表示形式呢？是否可以进一步减少自变量的数量？

**复杂化**

除了简化，还可以朝另一个方向发展，即迭代更复杂的游戏。比如，扩展棋盘，或者扩展成五子棋等等。

**趣味性**

从游戏趣味性上扩展，可以考虑增加一个用户系统，开发出排行榜的功能，让纯静态网站变得动态等。

**八、参考文献：**

【1】<https://zh.wikipedia.org/wiki/AlphaGo>

【2】<https://www.tensorflow.org/>

【3】<https://playground.tensorflow.org>

【4】<https://zh.wikipedia.org/wiki/%E4%BA%95%E5%AD%97%E6%A3%8B>

【5】<https://zh-hans.reactjs.org/>

【6】<https://zh-hans.reactjs.org/tutorial/tutorial.html>

【7】<https://zh.m.wikipedia.org/zh-cn/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD>

【8】<https://zh.m.wikipedia.org/wiki/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0>

【9】[美]米歇尔(Mitchell, T. M.).机器学习[M]. 曾华军等译.北京：机械工业出版社，2003.