《人工智能原理》设计报告题库

本文档为“人工智能原理-设计报告题库”，包括启发式搜索、对抗搜索、演化计算/群体智能、神经网络、生成对抗网络和强化学习等方向。每位同学可以从题库中任选 1 题。

评分规则：包括是否完成题目要求、工作量、提交时间、格式的规范等。鼓励“小题大做”：鼓励增加不同算法的对比、实验分析（包括大规模数据上的性能分析）、算法正确性分析、时间性能与空间性能分析、算法改进等。

报告模板：见附件“设计报告模板：2017217XXX-王二-《人工智能原理》上机作业报告.doc”

报告命名方式：学号-姓名-《人工智能原理》-设计报告。

作业的提交方式：在 educoder 平台（www.educoder.net），点击“图文作业”，选择并进入设计报告的提交页面，将设计报告作为附件上传即可。

一、 启发式搜索

一、实验目的：

熟悉和掌握启发式搜索的定义、估价函数和算法过程，理解求解流程和搜索顺序。

二、实验方法：

1.先熟悉启发式搜索算法；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

1.估价函数

在对问题的状态空间进行搜索时，为提高搜索效率需要和被解问题的解有关的大量控制性知识作为搜索的辅助性策略。这些控制信息反映在估价函数中。

估价函数的任务就是估计待搜索节点的重要程度，给这些节点排定次序。估价函数可以是任意一种函数，如有的定义它是节点 x 处于最佳路径的概率上，或是 x 节点和目标节点之间的距离等等。在此，我们把估价函数 $f(n)$定义为从初始节点经过 n 节点到达目标节点的最小代价路径的代价估计值，它的一般形式是：

$$f(n) = g(n) + h(n)$$

其中 $g(n)$是从初始节点到节点 n 的实际代价，$g(n)$可以根据生成的搜索树实际计算出来；$h(n)$是从 n 到目标节点的最佳路径的代价估计，$h(n)$主要体现了搜索的启发信息。

2. 启发式搜索过程的特性

（1）可采纳性

当一个搜索算法在最短路径存在的时候能保证能找到它，我们就称该算法是可采纳的。所有 A*算法都是可采纳的。

（2）单调性

一个启发函数 h 是单调的，如果

a） 对所有的状态 n_i 和 n_j，其中 n_j 是 n_i 的子孙，$h(n_i)- h(n_j)\leq cost(n_i,n_j)$，其中 $cost(n_i,n_j)$是从 n_i 到 n_j 实际代价。

b） 目标状态的启发函数值为 0，即 $h(Goal)=0$.

具有单调性的启发式搜索算法在对状态进行扩展时能保证所有被扩展的状态的f值是单调递增（不减）。

（3）信息性

比较两个启发策略h1和h2，如果对搜索空间中的任何一个状态n都有$h1(n) \leq h2(n)$，就说h2比h1具有更多的信息性。

一般而言，若搜索策略h2比h1有更多的信息性，则h2比h1考察的状态要少。但必须注意的是更多信息性需要更多的计算时间，从而有可能抵消减少搜索空间所带来的益处。

3.常用的启发式搜索算法

（1）局部择优搜索算法（瞎子爬山法）

瞎子爬山法是最简单的启发式算法之一。该算法在搜索过程中扩展当前节点并估价它的子节点。最优的子节点别选择并进一步扩展；该子节点的兄弟节点

和父节点都不再被保留。当搜索到达一种状态，该状态比它的所有子状态都要好，则搜索停止。因此，该算法的估价函数可表示为f(n) = h(n)。

在一个限定的环境下，瞎子爬山法可能会极大的提高搜索的效率，但是对整个搜索空间而言，可能得不到全局最优解。

（2）最好优先搜索法（有序搜索法）

该算法的估价函数采用f(n) = g(n) + h(n)，在搜索过程中算法使用OPEN表和CLOSE表来记录节点信息：OPEN表中保留所有已生成而未考察的节点；CLOSE表中保留所有已访问过的节点。算法在每一次搜索过程中都会对OPEN表中的节点按照每个节点的f值进行排序，选择f值最小节点进行扩展。算法的描述如下：

① 把起始节点S放到OPEN表中，计算f(S),并把其值与节点S联系起来。

② 若OPEN是个空表，则算法失败退出，无解。

③ 从OPEN表中选择一个f值最小的节点i。结果有几个节点合格，当其中有一个为目标节点时，则选择此目标节点，否则就选择其中任一个节点作为节点 i 。

④ 把节点i从OPEN表中移出，并把它放入到CLOSED的扩展节点表中。

⑤ 若节点i是个目标节点，则成功退出，求得一个解。

⑥ 扩展i，生成其全部后继节点。对i的每个后继节点j：

（a） 计算f(j)。

（b） 如果j既不在OPEN表中，也不在CLOSED表中，则用估价函数f将其添加到OPEN表。从j加一指向其父辈节点i的指针，以便一旦找到目标节点时记住一个解答路径。

（c） 如果j已则OPEN表中或CLOSED表中，则比较刚刚对j计算过的f值和前面计算过的该节点在表中的f的值。若新的f值较小，则

（i） 以此值取代旧值。

（ii） 从j指向i，而不是指向它的父辈节点。

（iii） 若节点j在CLOSED表中，则把它移回OPEN表。

⑦ 转向②。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）状态表示的数据结构

（2）状态扩展规则的表示

（3）搜索产生的状态空间图（样例）

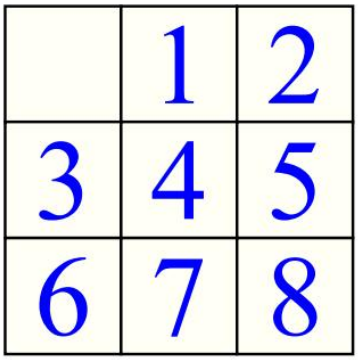
（4）OPEN表和CLOSE表变化过程（样例）

（5）衡量指标

（6）程序清单

（7）实验结果讨论

题目1.1：问题描述：用启发式搜索方法求解九宫问题，目标状态如下图



延伸（选做）：

a. 随机生成至少10个初始状态作为测试样例，通过实验比较分析A*算法与盲目搜索算法（深度优先和宽度优先）。

b. 随机生成至少10个初始状态作为测试样例，通过实验比较分析不同的启发式搜索策略。

c. 实现A*算法的不同改进，并通过实验比较分析。

d. 比较不同的启发式函数对于启发式搜索算法性能的影响。

e. 若问题的初始状态是随机产生的，你的实验程序应该如何改进？（给于给定的任意初始状态S1，如何设计算法，使得能判定状态S1是否可以到达目标状态S2？）

题目 1.2：一般的传教士和野人问题

目的：熟练掌握启发式搜索算法及其应用。

问题背景：一般的传教士和野人问题（Missionaries and Cannibals）是指：有 N 个传教士和 N 个野人来到河边准备渡河。河岸有一条船，每次至多可供 K 人乘渡。问传教士为了安全起见，应如何规划摆渡方案，使得任何时刻，在河的两岸以及船上的野人数目总是不超过传教士的数目，但允许在河的某一岸只有野人而没有传教士。

基本任务：假设 K<= 3，请编程实现传教士和野人问题的求解，并以不同的 N 为例给出摆渡方案验证算法的正确性。

延伸：

A. 有哪些不同的盲目搜索算法和启发式搜索算法可以求解该问题，请实现不同的对比算法，并通过实验结果分析各算法的区别。

B. 实现 A*算法的不同改进，并通过实验比较分析。

C. 比较不同的启发式函数对于启发式搜索算法性能的影响。

题 1.3：N 皇后问题

目的：熟练掌握搜索算法及其应用。

基本任务：在 N 行 N 列的国际象棋上摆放 N 个皇后，使其不能互相攻

击（即任意两个皇后都不能处于同一行、同一列或同一斜线上）。请问有多少种摆法，以及如何摆放这些皇后。请以八皇后问题为例给出摆放方案。

延伸：

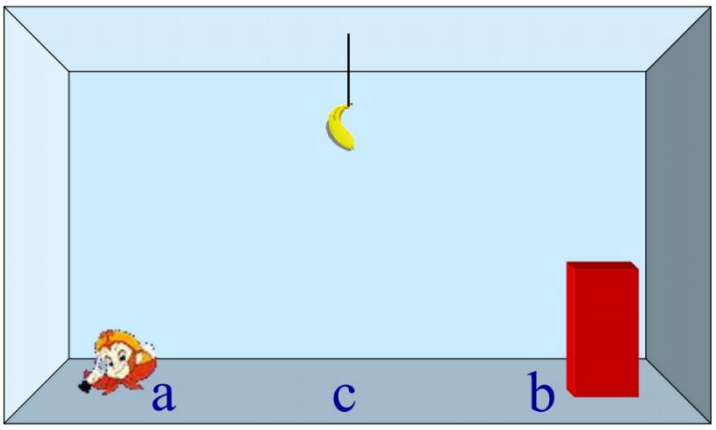
（1）有哪些算法可以求解该问题，请实现不同的算法，并从时间效率、空间效率、算法正确性等角度进行对比。

（2）从不同算法的对比结果中可以得出哪些有意义的结论？

（3）如何尽可能的提高搜索效率？

题 1.4 猴子香蕉问题

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意 活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可 被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为 A，箱子位置为 B，香蕉位置在 C），如何行动可摘取到香蕉。



目的：熟练掌握搜索算法及其应用。

基本任务：对于上图中的地图，请编程求解并给出猴子可摘到香蕉的行动方案。

延伸：

（1）若换一个房间地图，问题求解有哪些变化？例如：房间中没有箱子、有 m 种不同的砖块（猴子只能通过把若干个砖块叠起来才能够到香蕉），每种砖块的数量是 n_i。每种砖块都是长方体，第 i 种砖块的大小是（a_i，b_i，c_i），位置是(x_i，y_i)。砖块能够翻转，可以将任意两边当作底面，剩下的那边作为高。问猴子应如何规划行动方案？

（2）有哪些算法可以求解该问题，请实现不同的算法，并从时间效率、空间效率、算法正确性等角度进行对比。

二、 对抗搜索

一、实验目的：

熟悉和掌握对抗搜索的定义、估价函数和算法过程，理解求解流程和基本算法。

二、实验方法：

1.先熟悉对抗搜索算法；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

在之前的实验中，我们仅研究单个智能体相关联的搜索策略，该智能体旨在找到通常以一系列动作的形式表达的解决方案。但是，可能存在多个智能体在同一搜索空间中搜索解决方案的情况，这种情况通常发生在博弈中。具有多个智能体的环境被称为多智能体环境，其中每个智能体可能是其他智能体的对手并且彼此竞争。每个智能体都需要考虑其他智能体的操作以及该操作对其性能的影响。因此，搜索两个或更多具有相互冲突目标的玩家试图探索解决方案的相同搜索空间的搜索称为对抗搜索，通常称为博弈。搜索算法和启发式评估函数是有助于在 AI 中建模和解决博弈问题的两个主要因素。

本实验中考虑的博弈问题是：有完整信息的、确定性的、轮流行动的、两个游戏者的零和游戏（如象棋）。

确定性的：表示在任何时间点上，玩家之间都有有限的互动；

零和游戏：意味着游戏双方有着相反的目标，换句话说，在游戏的任何终结状态下，所有玩家获得的总和等于零，有时这样的游戏也被称为严格竞争博弈。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）状态表示的数据结构

（2）状态扩展规则的表示

（3）搜索产生的状态空间图（样例）

（4）OPEN表和CLOSE表变化过程（样例）

（5）衡量指标

（6）程序清单

（7）实验结果讨论

题目2.1 实现五子棋AI

目的：熟练掌握极大值极小值算法、Alpha Beta 剪枝算法等对抗搜索算法及其应用。

基本任务：请编程实现一个五子棋 AI，给出算法设计思路与实现方案。完整的一次对弈过程可通过录屏的方式展示。

延伸：

A. 启发式函数应如何设计？分析比较不同的启发式函数带来的影响。

B. 通过实验分析搜索深度这一参数对于实验结果的影响。

C. 有哪些提高搜索效率的方法？

题目2.2 实现井字棋AI

目的：熟练掌握极大值极小值算法、Alpha Beta 剪枝算法等对抗搜索算法及其应用。

基本任务：请编程实现一个井字棋 AI，给出算法设计思路与实现方案。完整的一次对弈过程可通过录屏的方式展示。

延伸：

A. 启发式函数应如何设计？分析比较不同的启发式函数带来的影响。

B. 通过实验分析搜索深度这一参数对于实验结果的影响。

C. 有哪些提高搜索效率的方法？

三、 演化计算/群体智能

一、实验目的：

熟悉和掌握演化计算/群体智能的基本思想和基本方法，通过实验培养利用演化计算/群体智能进行问题求解的基本技能，并且了解其他分支的基本思想和基本方法。

二、实验方法：

1.先熟悉进化计算中演化计算/群体智能的基本思想和流程；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

生物群体的生存过程普遍遵循达尔文的物竞天择、适者生存的进化准则。群体中的个体根据对环境的适应能力而被大自然所选择或淘汰。进化计算(evolutionary computation, 简称 EC；也称为演化计算)就是通过模拟自然物群进化的一类非常鲁棒的优化算法，它们模拟群体(其中组成群体的每个个体表示搜索问题解空间中的一个解)的集体学习过程，从任意初始群体出发，通过随机选择、变异和交叉过程，使群体进化学习到搜索空间中越来越好的区域。进化计算中选择过程使群体中适应性好的个体比适应性差的个体有更多的机会遗传到下一代中，交叉算子将父代信息结合在一起并他们遗传给下一代个体，而变异过程在群体中引入新的变种。进化计算包括遗传算法(evolutionary algorithm,EA)、进化策略(evolution strategy, ES)、进化编程(evolutionary programming, EP)、遗传编程(genetic programming, GP)等多个分支。

参考资料：

IEEE 动态优化社区主页：http://www.tech.dmu.ac.uk/~syang/IEEE_ECiDUE.html

IEEE CIS Task Force on Differential Evolution： http://labraj.feri.um.si/tf-cis-de/

IEEE CIS Task Force on Multi-Objective Evolutionary Algorithms:

http://www.is.ovgu.de/is_media/Research/IEEE_CIS_EMO_TF-p-1126.html

Suganthan (IEEE Fellow)：https://github.com/P-N-Suganthan/CODES

Yong Wang : http://ist.csu.edu.cn/YongWang.htm

Deb（IEEE Fellow）： http://www.egr.msu.edu/~kdeb/index.shtml

Xin Yao (IEEE Fellow)： https://www.cs.bham.ac.uk/~xin/

Shengxiang Yang http://www.tech.dmu.ac.uk/~syang/

Yaochu Jin (IEEE Fellow): http://www.soft-computing.de/

Michalewicz https://cs.adelaide.edu.au/~zbyszek/papers.html

Xiaodong Li：https://titan.csit.rmit.edu.au/~e46507/

PSO（部分开源代码）: http://www.adaptivebox.net/CILib/code/psocodes_link.html

全局优化算法（部分开源代码）：

http://www.mat.univie.ac.at/~neum/glopt/software_g.html

PSO（部分开源代码）: http://clerc.maurice.free.fr/pso/

多目标优化（部分开源代码）：

http://delta.cs.cinvestav.mx/~ccoello/EMOO/EMOOsoftware.html

http://www.macs.hw.ac.uk/~ml355/journals.htm

meta-heuristic 多种算法及 matlab 代码 https://yarpiz.com/

http://comopt.ifi.uni-heidelberg.de/software/

下面以简单的遗传算法为例介绍演化计算的基本求解过程。其他演化计算/群体智能算法可参考教材《计算智能导论》。

遗传算法是模仿生物遗传学和自然选择机理，通过人工方式构造的一类优化搜索算法，是对生物进化过程的一种数学仿真，是进化计算的一种最重要形式。遗传算法为那些那些难以找到传统数学模型的难题找出了一个解决方法。自从 Holland 于 1975 年在其著作《Adaptation in Natural and Artificial Systems》中首次提出遗传算法以来，经过近 30 年的研究，现在已发展到一个比较成熟的阶段，并且在实际中已经得到了很好的应用。

1.简单遗传算法的构成要素

（1）染色体编码和解码方法

在实现对一个问题用遗传算法之前，我们必须先对问题的解空间进行编码，以便使得它能够由遗传算法进行操作。解码就是把遗传算法操作的个体转换成原来问题结构的过程。常见的编码方法有：二进制编码、浮点数编码、格雷码等。以二进制为例：假设某一参数的取值范围是[A，B]，A＜B，我们有长度为 l 的二进制编码串来表示该参数，将[A，B]等分成 2^l-1 个子部分，每个等分的长度为δ，则它能够产生 2^l 种不同的编码。

$$
\begin{aligned}
&00000000000\ldots0000000000 = 0 \quad \rightarrow \quad A \\
&00000000000\ldots0000000001 = 1 \quad \rightarrow \quad A+\delta \\
&\qquad\qquad\qquad\vdots \\
&11111111111\ldots\ldots1111111111 = 2^l-1 \quad \rightarrow \quad B
\end{aligned}
$$

二进制编码的最大缺点是使得遗传算法操作的个体位串太长，这容易降低遗传算法的运行效率，很多问题采用其他编码方法可能更有利。如对于函数优化问题，浮点数编码方法就更有效，而二进制编码方法不但容易降低遗传算法的运行效率，而且会产生精度损失。浮点数编码方法是指个体的每个染色体用某一范围内的一个浮点数表示，个体的编码长度就等于问题变量的个数。

在实际运用遗传算法解决问题时，一般都需要根据具体的问题采用合适的编码方法，这样更有利于遗传算法搜索到问题的最优解。

（2）适应度函数

遗传算法在进化搜索中基本上不用外部信息，仅用目标函数也就是适应度函数为依据。适应度函数是用于度量问题中每一个染色体优劣程度的函数，体现了染色体的适应能力。遗传算法的目标函数不受连续可微的约束且定义域可以是任意集合。但对适应度函数有一个要求就是针对输入可以计算出的能加以比较的结果必须是非负的。

在具体应用中，适应度函数的设计要结合求解问题本身的要求而设计。因为适应度函数对个体的评估是遗传算法进行个体选择的唯一依据，因此适应度函数的设计直接影响到遗传算法的性能。对于很多问题，可以直接把求解问题的目

标函数作为适应度函数使用，但也存在很多问题需要进行一定的转换才能使得目标函数可以用作遗传算法的适应度函数。

（3）遗传算子

遗传算法主要有三种操作算子：选择(selection)、交叉(crossover)、变异(mutation)。

①选择算子

选择算子根据个体的适应度函数值所度量的优劣程度选择下一代个体。一般地，选择算子将使适应度函数值较大的个体有较大的机会被遗传到下一代，而适应度函数值较小的个体被遗传到下一代的机会较小。一般常采用的选择算子是赌轮选择机制。赌轮选择算子的基本原理如下。

令 $\sum f_i$ 表示种群的适应度值之总和，f_i 表示群体中第 i 个染色体的适应度值，则第 i 个个体产生后代的能力正好为其适应度值与群体适应度值的比值 $f_i/\sum f_i$。

赌轮选择算子在个体数不太多时，有可能出现不正确反映个体适应度的选择过程，也就是说适应度高的个体有可能反而被淘汰了。为了改进赌轮选择算子的这种缺点，有很多改进的交叉选择算子。如：最佳个体保存法、期望值方法、排序选择方法、联赛选择方法、排挤方法等。

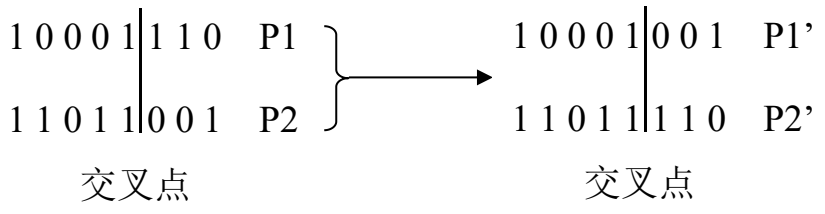
②交叉算子

在自然界生物进化过程中，起核心作用的是生物遗传基因的重组(加上变异)。同样，遗传算法中，起核心作用的是遗传操作的交叉算子。所谓交叉算子就是把两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃提高。

交叉算子设计一般与所求解的具体问题有关，但都应该考虑以下两点：Ⓐ设计的交叉算子必须能保证前一代中优秀个体的性状能在后一代的新个体中尽可能得到遗传和继承。Ⓑ交叉算子与问题的编码是相互联系的，因此，交叉算子设计和编码设计需协调操作，也就是所谓的编码—交叉设计。

对于字符串编码的遗传算法，交叉算子主要有一点交叉、两点交叉、多点交叉和一致交叉等。其中一点交叉的主要操作过程如下。假设两个配对个体为 P1、P2 分别如下所示。经过一点交叉后，得到两个新的个体 P1’、P2’。

对于实数编码的遗传算法，交叉算子主要是采用算术交叉算子。假设两个配对个体分别为 P1＝(x1,y1)和 P2=(x2,y2)。在 P1 和 P2 进行算术交叉后得到的两个新个体 P_1' 和 P_2' 分别可由下式计算得到。



$$P_1^{'} = (\lambda P_1 + (1-\lambda) P_2$$
$$P_2^{'} = (1-\lambda) P_1 + \lambda P_2$$

③变异算子

变异算子是改变数码串中某个位置上的数码。遗传算法中引入变异算子有两个目的：其一是使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算子已接近最优解领域时，利用变异算子的这种局部随机搜索能力可以加速遗传算法向最优解收敛。一般在这种情况下，遗传算法的变异概率应取较小的值，否则接近最优解的积木块会因为变异而遭到破坏。其二是使遗传算法可以维持较好的群体多样性，以防止遗传算法出现未成熟收敛现象。此时，遗传算法的变异概率应取较大的值。

遗传算法中，交叉算子具有很好的全局搜索能力。变异算子具有较好的局部搜索能力，是算法的辅助操作算子。遗传算法通过交叉和变异这一对相互配合又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力。变异算子除了基本的变异算子坏，还有很多有效的变异算子。如逆转变异算子、自适应变异算子等。对于字符串编码的遗传算法，基本变异算子就是随机的从个体中选择某些基因位，然后以一定的概率对这些基因位进行翻转变异，也就是把这些基因位中为 0 的基因位以概率 *Pm* 变为 1，为 1 的基因为以概率 *Pm* 变为 0。对于实数编码的遗传算法，一般采用随机变异的方式，使变异个体的每一个变量分量加上一个随机数，一般使用均为分布的随机数或者是高斯分别的随机数。

（4）基本遗传算法运行参数

N：群体大小，即群体中所含个体的数量，一般取 20～100

T：遗传算法的终止进化代数，一般取 100～500

pc：杂交概率，一般取 0.4～0.99

pm：变异概率，一般取 0.0001～0.1

pr：复制概率

2.算法过程：

简单遗传算法的基本流程如下：

(1)初始化群体：随机产生一个由确定长度的特征串组成的初始群体

(2)计算群体上每个个体的适应度值

(3)按由个体适应度值所决定的某个规则选择将进入下一代的个体，也就是选择算子操作。

(4)按概率 Pc 对配对池中的个体进行交叉算子操作。

(5)按概率 Pm 对交叉算子产生的所有新个体进行变异操作。

(6)若没有满足某种停止条件，则转(2)，否则进入下一步。

(7)输出群体中适应度值最优的染色体作为问题的满意解或最优解。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）问题的编码和解码表示

（2）适应度函数

（3）算法参数设置

算法流程图

（4）选择算子、交叉算子、变异算子的设计

（7）算法的结束条件

（8）程序清单

（9）根据实验内容，给出相应结果并分析

题目 3.1 选择一种演化算法/群体智能算法，求解函数 f(x)=x*sin(10л+x)+1.0 在区间[-1，2]的最大值。

延伸：

a. 什么是过早收敛和过慢结束？造成上述状况的原因有哪些？

b. 你的程序有没有要改进之处？如何改进？

题目 3.2 用遗传算法求解下列函数的最大值，设定求解精度到 15 位小数。

$$f(x,y)=\frac{6.452(x+0.125y)(\cos(x)-\cos(2y))^2}{\sqrt{0.8+(x-4.2)^2+2(y-7)^2}}+3.226y$$

$$x\in[0,10],y\in[0,10]$$

1）给出适应度函数（Fitness Function）的 M 文件（Matlab 中要求适应度函数最小化）。

2）设计及选择上述问题的编码、选择操作、交叉操作、变异操作以及控制参数等，填入表 1，给出最佳适应度(Best fitness)和最佳个体（Best individual）图。

表 1 遗传算法参数的选择

编码	编码方式（population type）	
种群参数	种群规模（population size）	
	初始种群的个体取值范围（Initial range）	
选择操作	个体选择概率分配策略（对应 Fitness scaling）	
	个体选择方法（Selection function）	
最佳个体保存	优良个体保存数量（Elite count）	
交叉操作	交叉概率（Crossover fraction）	
	交叉方式（Crossover function）	
变异操作	变异方式(Mutation function)	
停止参数	最大迭代步数（Generations）	
	最大运行时间限制（Time limit）	
	最小适应度限制（Fitness limit）	
	停滞代数（Stall generations）	
	停滞时间限制（Stall time limit）	

3）使用相同的初始种群（Use random state from previous run），设置不同的种群

规模（population size），例如 5、20 和 100，初始种群的个体取值范围（Initial range）为[0;1]，其他参数同表 1，然后求得相应的最佳适应度(Best fitness)、平均适应度（Mean fitness）和最佳个体（Best individual），填入下表 2，分析种群规模对算法性能的影响。

表 2 不同的种群规模的 GA 运行结果

种群规模	最佳适应度	平均适应度	最佳个体	
			x	*y*
5				
20				
100				

*4) 设置种群规模（population size）为 20，初始种群的个体取值范围（Initial range）为[0;10]，选择不同的选择操作、交叉操作和变异操作，其他参数同表 1，然后独立运行算法 10 次，完成下表 3，并分析比较采用不同的选择策略、交叉策略和变异策略的算法运行结果。

表 3 不同的选择策略、交叉策略和变异策略的算法运行结果

遗传算法参数设置（gaoptimset）			1	2	3	4
选择操作	个体选择概率分配 FitnessScalingFcn	Rank（排序）@fitscalingrank	√	√		√
		Proportional（比率）@fitscalingprop			√	
	个体选择 SelectionFcn	Roulette（轮盘赌选择）@selectionroulette	√	√		√
		Tournament（竞标赛选择）@selectiontournament			√	
交叉操作 CrossoverFcn	单点交叉 @crossoversinglepoint		√		√	√
	两点交叉 @crossovertwopoint			√		
变异操作 MutationFcn	Uniform（均匀变异）@mutationuniform		√	√	√	
	Gaussian（高斯变异）@mutationgaussian					√
最好适应度						
最差适应度						
平均适应度						

题目 3.3 用遗传算法求解下面一个 Rastrigin 函数的最小值，设定求解精度到 15 位小数。

$$f(x_1,x_2)=20+x_1^2+x_2^2-10(\cos 2\pi x_1+\cos 2\pi x_2)$$

1）给出适应度函数。

2）设计上述问题的编码、选择操作、交叉操作、变异操作以及控制参数等，填入表 4，并画出最佳适应度(Best fitness)和最佳个体（Best individual）图。

表 4 遗传算法参数的选择

编码	编码方式（population type）	
种群参数	种群规模（population size）	
	初始种群的个体取值范围（Initial range）	
选择操作	个体选择概率分配策略（对应 Fitness scaling）	
	个体选择方法（Selection function）	
最佳个体保存	优良个体保存数量（Elite count）	
交叉操作	交叉概率（Crossover fraction）	
	交叉方式（Crossover function）	
变异操作	变异方式(Mutation function)	
停止参数	最大迭代步数（Generations）	
	最大运行时间限制（Time limit）	
	最小适应度限制（Fitness limit）	
	停滞代数（Stall generations）	
	停滞时间限制（Stall time limit）	

3）设置种群的不同初始范围，例如[1;1.1]、[1;100]和[1;2]，画出相应的最佳适应度值(Best fitness)和平均距离（Distance）图，比较分析初始范围及种群多样性对遗传算法性能的影响。

4）设置不同的交叉概率（Crossover fraction=0、0.8、1），画出无变异的交叉（Crossover fraction=1）、无交叉的变异(Crossover fraction=0)以及交叉概率为 0.8 时最佳适应度值(Best fitness)和和平均距离（Distance）图，分析交叉和变异操作对算法性能的影响。

题目 3.4 针对 http://www.sfu.ca/~ssurjano/optimization.html 中列出的一个或多个测试问题，请实现至少一种演化算法/群体智能算法。

延伸：该问题还可以使用哪些算法进行求解？请实现不同的对比算法，通过实验结果分析各算法的区别，并分析参数对算法性能的影响。

题目 3.5 针对 http://www5.zzu.edu.cn/cilab/Benchmark/wysyhwtcsj.htm 中列出的一个或多个测试集，请实现至少一种演化算法/群体智能算法。

延伸：该问题还可以使用哪些算法进行求解？请实现不同的对比算法，通过实验结果分析各算法的区别，并分析参数对算法性能的影响。

题目 4 基于 MindSpore 框架的 MNIST 手写体识别

1.1 实验介绍

本实验拟实现一个简单的图片分类的功能，整体流程如下：

1、处理需要的数据集，这里使用了 MNIST 数据集。

2、定义一个网络，这里我们使用 LeNet 网络。

3、定义损失函数和优化器。

4、加载数据集并进行训练，训练完成后，查看结果及保存模型文件。

5、加载保存的模型，进行推理。

6、验证模型，加载测试数据集和训练后的模型，验证结果精度。

1.2 实验准备

在动手进行实践之前，确保已经正确安装了 MindSpore。如果没有，可以通过 MindSpore 官网安装页面：https://www.mindspore.cn/install/ ，将 MindSpore 安装在电脑当中。

同时希望你拥有 Python 编码基础和概率、矩阵等基础数学知识。

1.3 实验详细设计与实现

1.3.1 数据准备

我们用到的 MNIST 数据集是由 10 类 28*28 的灰度图片组成，训练数据集包含 60000 张图片，测试数据集包含 10000 张图片。

MNIST 数据集下载页面：http://yann.lecun.com/exdb/mnist/。页面提供 4 个数据集下载链接，其中前 2 个文件是训练数据需要，后 2 个文件是测试结果需要。

将数据集下载并解压到本地路径下，这里将数据集解压分别存放到工作区的./MNIST_Data/train、./MNIST_Data/test 路径下。

目录结构如下：

```
└─MNIST_Data
    ├─  test
    │       t10k-images.idx3-ubyte
    │       t10k-labels.idx1-ubyte
    │
    └─  train
            train-images.idx3-ubyte
            train-labels.idx1-ubyte
```

为了方便样例使用，我们可以在样例脚本中添加自动下载数据集的功能。

1.3.2 实验步骤

步骤一. 导入 Python 库&模块

在使用前，导入需要的 Python 库。

目前使用到 os 库，为方便理解，其他需要的库，我们在具体使用到时再说明。

```
import os
```

详细的 MindSpore 的模块说明，可以在 MindSpore API 页面中搜索查询。

步骤二. 配置运行信息

在正式编写代码前，需要了解 MindSpore 运行所需要的硬件、后端等基本信息。

可以通过 context.set_context 来配置运行需要的信息，譬如运行模式、后端信息、硬件等信息。

导入 context 模块，配置运行需要的信息。

```
import argparse
from mindspore import context

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='MindSpore LeNet Example')
    parser.add_argument('--device_target', type=str, default="CPU", choices=['Ascend', 'GPU',
'CPU'],
                        help='device where the code will be implemented (default: CPU)')
    args = parser.parse_args()
    context.set_context(mode=context.GRAPH_MODE, device_target=args.device_target)
    dataset_sink_mode = not args.device_target == "CPU"
    ...
```

在样例中我们配置样例运行使用图模式。根据实际情况配置硬件信息，譬如代码运行在 Ascend AI 处理器上，则--device_target 选择 Ascend，代码运行在 CPU、GPU 同理。详细参数说明，请参见 context.set_context 接口说明。

步骤三. 数据处理

数据集对于训练非常重要，好的数据集可以有效提高训练精度和效率。在加载数据集前，我们通常会对数据集进行一些处理。

定义数据集及数据操作

我们定义一个函数 create_dataset 来创建数据集。在这个函数中，我们定义好需要进行的数据增强和处理操作：

1. 定义数据集。
2. 定义进行数据增强和处理所需要的一些参数。
3. 根据参数，生成对应的数据增强操作。
4. 使用 map 映射函数，将数据操作应用到数据集。
5. 对生成的数据集进行处理。

其中，可以设置如下参数

batch_size：每组包含的数据个数，现设置每组包含 32 个数据。

repeat_size：数据集复制的数量。

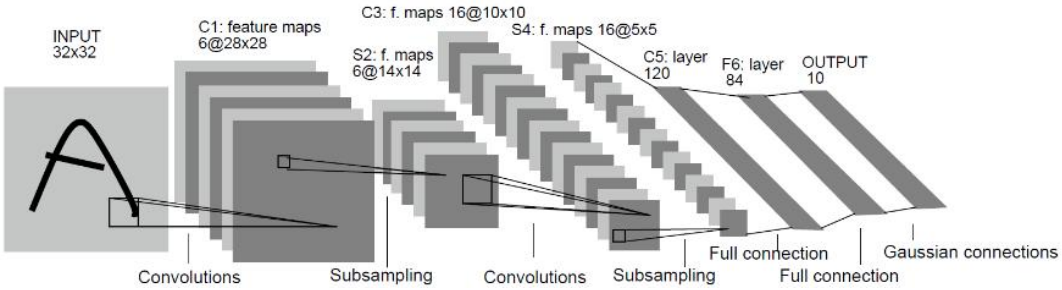
先进行 shuffle、batch 操作，再进行 repeat 操作，这样能保证 1 个 epoch 内数据不重复。

MindSpore 支持进行多种数据处理和增强的操作，各种操作往往组合使用，具体可以参考数据处理与数据增强章节。

步骤四. 定义网络

我们选择相对简单的 LeNet 网络。LeNet 网络不包括输入层的情况下，共有 7 层：2 个卷积层、2 个下采样层（池化层）、3 个全连接层。每层都包含不同数量的训练参数，如下图所示：

LeNet-5



更多的 LeNet 网络的介绍不在此赘述，希望详细了解 LeNet 网络，可以查询 http://yann.lecun.com/exdb/lenet/。

首先，我们需要对全连接层以及卷积层进行初始化。

TruncatedNormal：参数初始化方法，MindSpore 支持 TruncatedNormal、Normal、Uniform 等多种参数初始化方法，具体可以参考 MindSpore API 的 mindspore.common.initializer 模块说明。

使用 MindSpore 定义神经网络需要继承 mindspore.nn.cell.Cell。Cell 是所有神经网络（Conv2d 等）的基类。

神经网络的各层需要预先在__init__方法中定义，然后通过定义 construct 方法来完成神经网络的前向构造。

步骤五.　定义损失函数及优化器

基本概念

在进行定义之前，先简单介绍损失函数及优化器的概念。

损失函数：又叫目标函数，用于衡量预测值与实际值差异的程度。深度学习通过不停地迭代来缩小损失函数的值。定义一个好的损失函数，可以有效提高模型的性能。

优化器：用于最小化损失函数，从而在训练过程中改进模型。

定义了损失函数后，可以得到损失函数关于权重的梯度。梯度用于指示优化器优化权重的方向，以提高模型性能。

定义损失函数

MindSpore 支持的损失函数有 SoftmaxCrossEntropyWithLogits、L1Loss、MSELoss 等。这里使用 SoftmaxCrossEntropyWithLogits 损失函数。

```
from mindspore.nn.loss import SoftmaxCrossEntropyWithLogits
```

在__main__函数中调用定义好的损失函数：

```
if __name__ == "__main__":
    ...
    #define the loss function
    net_loss = SoftmaxCrossEntropyWithLogits(is_grad=False, sparse=True, reduction='mean')
    ...
```

定义优化器

MindSpore 支持的优化器有 Adam、AdamWeightDecay、Momentum 等。

这里使用流行的 Momentum 优化器。

步骤六. 训练网络

配置模型保存

MindSpore 提供了 callback 机制，可以在训练过程中执行自定义逻辑，这里使用框架提供的 ModelCheckpoint 为例。 ModelCheckpoint 可以保存网络模型和参数，以便进行后续的 fine-tuning（微调）操作。

步骤七. 配置训练网络

通过 MindSpore 提供的 model.train 接口可以方便地进行网络的训练。LossMonitor 可以监控训练过程中 loss 值的变化。 这里把 epoch_size 设置为 1，对数据集进行 1 个迭代的训练。

其中， 在 train_net 方法中，我们加载了之前下载的训练数据集，mnist_path 是 MNIST 数据集路径。

步骤八. 运行并查看结果

训练过程中会打印 loss 值，类似下图。loss 值会波动，但总体来说 loss 值会逐步减小，精度逐步提高。每个人运行的 loss 值有一定随机性，不一定完全相同。 训练过程中 loss 打印示例如下：

...

epoch: 1 step: 262, loss is 1.9212162

epoch: 1 step: 263, loss is 1.8498616

epoch: 1 step: 264, loss is 1.7990671

epoch: 1 step: 265, loss is 1.9492403

epoch: 1 step: 266, loss is 2.0305142

epoch: 1 step: 267, loss is 2.0657792

epoch: 1 step: 268, loss is 1.9582214

epoch: 1 step: 269, loss is 0.9459006

epoch: 1 step: 270, loss is 0.8167224

epoch: 1 step: 271, loss is 0.7432692

...

训练完后，即保存的模型文件，示例如下:

checkpoint_lenet-1_1875.ckpt

其中，

checkpoint_lenet-1_1875.ckpt：指保存的模型参数文件。名称具体含义 checkpoint_网络名称-第几个 epoch_第几个 step.ckpt。

步骤九. 验证模型

在得到模型文件后，通过模型运行测试数据集得到的结果，验证模型的泛化能力。

使用 model.eval 接口读入测试数据集。

使用保存后的模型参数进行推理。

其中，

load_checkpoint：通过该接口加载 CheckPoint 模型参数文件，返回一个参数字典。

checkpoint_lenet-1_1875.ckpt：之前保存的 CheckPoint 模型文件名称。

load_param_into_net：通过该接口把参数加载到网络中。

其中，

运行结果示例如下：

```
...
============== Starting Testing ==============
============== Accuracy:{'Accuracy': 0.9742588141025641} ==============
```

可以在打印信息中看出模型精度数据，示例中精度数据达到 97.4%，模型质量良好。

题目 5. 基于 MindSpore 框架的生成对抗网络

实验介绍

生成对抗网络是一种训练生成网络的框架，比如生成图片的深度卷积神经网络。构建一个用来生成图片的 GAN 模型需要同时具备判别模型和生成模型，判别模型，用来分析一张图片是真实的还是仿造的；生成模型使用反转的卷积层将输入转换为对应像素值的完整二维图像。本实验通过 GAN 模型的搭建，主要介绍 TensorFlow2.0 计算的基本流程，以及构建网络的基本要素。

实验目的

学习怎么定义和训练一个单独的判别模型，用来学习真假图片之间的不同点

学习怎么定义单独的生成模型，并训练同时具备生成和判别的复合模型

学习怎么评估 GAN 模型的表现，并使用最终得到的单独的生成模型生成图片

实验详细设计与实现

实验数据准备

数据集简介：

1). MNIST 数据集来自美国国家标准与技术研究所（National Institute of Standards and Technology ，简称 NIST)；

2). 该数据集由来自 250 个不同人手写的数字构成，其中 50%是高中学生，50%来自人口普查局的工组人员；

3). 数据集可在 http://yann.lecun.com/exdb/mnist/ 获取，它包含了四个部分:

- Training set images: train-images-idx3-ubyte.gz (9.9 MB，解压后 47 MB，包含 60,000 个样本)

- Training set labels: train-labels-idx1-ubyte.gz (29 KB，解压后 60 KB，包含 60,000 个标签)

- Test set images: t10k-images-idx3-ubyte.gz (1.6 MB，解压后 7.8 MB，包含 10,000 个样本)

- Test set labels: t10k-labels-idx1-ubyte.gz (5KB，解压后 10 KB，包含 10,000 个标签)

4). mnist 是一个入门级的计算机视觉数据集，它包含各种手写数字图片，下图展示了 mnist 中的一些图片：

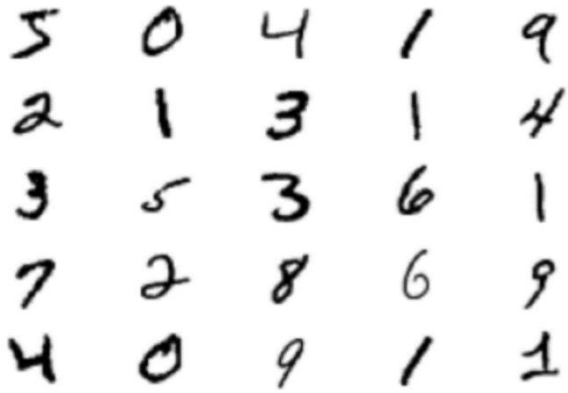


图 4-1 MNIST 数据集部分图片展示

数据集加载：

```
import numpy as np
# "中为放置 mnist.npz 的相对路径，改为自己数据集存放的路径
path = r'.\datasets\mnist.npz
f = np.load(path)                #加载数据集
trainX = f['x_train']          #训练集数据获取
trainy = f['y_train']          #训练集数据标签获取
testX = f['x_test']            #测试集数据获取
testy = f['y_test']            #测试集数据标签获取
print("Train", trainX.shape, trainy.shape)        #查看训练集数据形状及标签形状
print("Test", testX.shape, testy.shape)           #查看验证集数据形状及标签形状
```

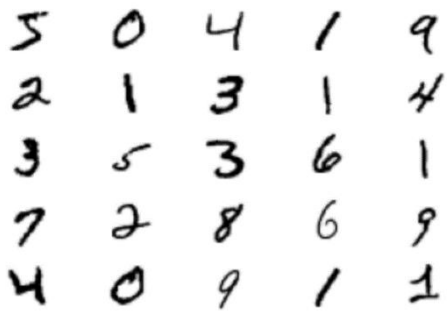
输出结果：

Train (60000, 28, 28) (60000,)

Test (10000, 28, 28) (10000,)

```
可视化训练集前 30 张图片:
from matplotlib import pyplot     #从 matplotlib 数据库导入 pyplot 模块
for i in range(30):                 # 使用 range()函数产生 0 到 29 这 30 个数，建立 for 循环
    #使用 subplot()构建子图，第一个参数 5 表示每列有 5 张图，
    第二个参数表示每行有 6 张图，1+i 表示第(1+i 张图)
    pyplot.subplot(5, 6, 1 + i)
    pyplot.axis('off')              # 关闭坐标轴("on"为显示坐标轴)
     #使用 imshow()函数画出训练集相应图片原始像素数据，参数 cmap = gray 表示黑底白字图像，这边使用"gray_r"表示白底黑字图
      pyplot.imshow(trainX[i], cmap='gray_r')
pyplot.show()                       # 固定写法，展示图片
```

输出结果：



GAN 模型的架构

GAN 模型的伪代码如下图所示：

```
<img src = "GAN 模型伪代码.PNG"  width=600>   #“”中为图片相对路径，改为自己图片存放的路径
```

输出结果：

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

导入相关的包

```
from numpy import expand_dims        #从 numpy 库导入 expand_dims 函数，用于数组增加维度
from numpy import zeros              #从 numpy 库导入 zeros 函数，用于生成元素都为 0 的数组
from numpy import ones               #从 numpy 库导入 ones 函数，用于生成元素都为 1 的数组
from numpy import vstack             #从 numpy 库导入 vstack 函数,用于在垂直方向(y 轴)堆叠数组
#从 numpy.random 模块导入 randn 函数，numpy.random.randn(d0, d1, …, dn)是
从标准正态分布中返回一个或多个样本值
from numpy.random import randn
#从 numpy.random 模块导入 randint 函数，用于生成整数，如 np.random.randint(1, 5)是随机生成 1，2，3，4 这四个整数，注意前闭后开
from numpy.random import randint
#从 keras.datasets.mnist 加载 load_data 函数，此次实验我们直接从本地加载图片，故不使用这个函数
# from keras.datasets.mnist import load_data
from tensorflow.keras.optimizers import Adam    #从 keras.optimizers 模块加载 Adam 优化器,用于训练模型
from tensorflow.keras.models import Sequential  #从 keras.models 库加载 Sequentil 序贯模型,用于线性堆叠多个网络层
from tensorflow.keras.layers import Dense       #从 keras.layers 库加载 Dense 层，用于全连接操作
#从 keras.layers 库加载 reshape 函数，用于将指定的矩阵变换成特定维数矩阵一种函数，且矩阵中元素个数不变，函数可以重新调整矩阵的行数、列数、维数。
from tensorflow.keras.layers import Reshape
```

```
#从 keras.layers 库加载 Flaten 函数，用来将输入“压平”，即把多维的输入一维化，常用在从卷积层到全连接层的过渡。Flaten 不影响 batch 的大小
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv2D                    #从 keras.layers 库加载 Conv2D 函数，用于卷积层操作
#从 keras.layers 库加载 Conv2DTranspore 函数，此函数的操作可以看作 Conv2D 函数的逆操作
from tensorflow.keras.layers import Conv2DTranspose
from tensorflow.keras.layers import LeakyReLU                 #从 keras.layers 库加载 LeakyReLU 激活函数
from tensorflow.keras.layers import Dropout                   #从 keras.layers 库加载 Dropout 函数
from matplotlib import pyplot                                 #从 matplotlib 库加载 pyplot 函数，用于显示图像
```

定义判别模型

模型从我们的数据集中获取样本图像作为输入，并输出关于样本是真实还是假的分类预测。

这是一个二分类问题：

输入：一个通道的图像，尺寸为 28×28 像素。 输出：二分类，样本是真实的（或假的）。

定义生成模型

生成器模型负责创建手写数字的新的，假的但合理的图像。

它通过从潜在空间中获取一个点作为输入并输出正方形灰度图像来实现此目的。

潜在空间是高斯分布值（例如 100 维）的任意定义的向量空间。它没有任何意义，但是通过在训练过程中从该空间中随机抽取点并将其提供给生成器模型，生成器模型将为潜点以及潜在空间分配含义，直到训练结束，潜矢量空间表示输出空间的 MNIST 图像的压缩表示，只有生成器才知道如何转换为看似可行的 MNIST 图像。

输入：潜在空间中的点，例如随机的一个服从高斯分布的 100 维向量。 输出：28×28 像素的二维正方形灰度图像，像素值范围[0,1]。 注意：我们不必非要使用 100 元素向量作为输入；它是一个整数并且被广泛使用，但是我们也同样可以去尝试 10、50 或 500 维的空间。

开发生成器模型要求我们将向量从具有 100 个维度的潜在空间转换为具有 28×28 或 784 个值的 2D 数组。

定义 GAN 网络

生成器模型中的权重将根据区分器模型的性能进行更新。当鉴别器善于检测假样本时，生成器会更新得更多；而当鉴别器模型相对较差或在检测假样本时会混淆时，生成器模型的更新会更少。这定义了这两个模型之间的零和或对抗关系。

我们采用一种较为简单的方法创建一个结合生成器模型和鉴别器模型的新模型。确切地说，我们不是在谈论或是创建新的第三个模型，而是一个新的逻辑模型（GAN 模型），该模型使用独立生成器和鉴别器模型中已经定义的层和权重。

生成器模型仅与鉴别器在假示例中的表现有关。因此，当鉴别器中的所有层是 GAN 模型的一部分时，我们都将其标记为不可训练，这样就不会在伪造的示例上对它们进行更新和过度训练。

通过此逻辑 GAN 模型训练生成器时，还有一个更重要的变化。我们希望判别器认为生成器输出的样本是真实的，而不是假的。因此，当将生成器训练为 GAN 模型的一部分时，我们会将生成的样本标记为真实（类别 1）。

生成真实样本

从 MINIST 数据集加载图片数据，并赋予标签

从潜在空间生成点作为生成模型的输入

使用生成模型生成虚假样本

保存生成的图片

评估判别模型，保存生成的图片，保存生成模型

对于 GAN 模型，没有一个特定的目标函数来评价这个模型的好坏，或者可以说是没有一个确切的指标评价来判定这个模型的好坏，所以我们会按一定周期去评估判别模型，同时保存此时刻生成器模型以及此时刻生成器生成的图片，通过人为地去评定图片生成的质量来确定某个时刻模型的好坏。

训练生成模型及判别模型

在 GAN 模型中，生成器模型和判别器模型会被同时训练，在同一步训练过程中，分为两个过程，首先，我们会使用一半批次数量的真实样本+一般批次数量的生成样本(由生成模型生成，标签为 0)训练判别模型；然后会生成一个批次的生成模型(由生成模型生成，标签为 1)训练 GAN 模型

题目 6：基于强化学习的迷宫寻宝

1. 实验介绍

该实验设置用字符表示的 6× 6 大小的迷宫环境，执行动作进行状态转移并且收益返回相应的结果，利用 Q-learning 算法训练智能体 agent 通过改变决策大小，使其不断对路径进行选择并实现其在迷宫中找到宝藏。

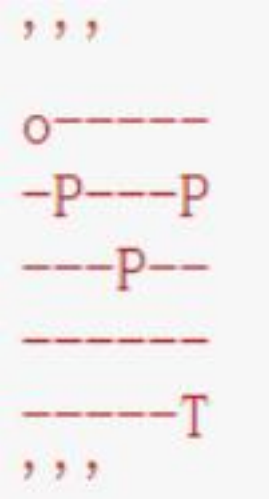
1.4 实验环境要求

1、python3.7

1.5 实验总体设计

使用强化学习 Q-Learning 算法设计一个可以在一定大小的迷宫中找到宝藏的 agent(智能体)。迷宫可以使用一个系列字符表示，字符"o"表示 agent 的位置，字符"T"表示宝藏的位置，该位置的收益为 1，且为游戏的终结状态，字符"P"表示陷阱的位置，该位置的收益为-1，且为游戏的终结状态，字符"_"表示收益为 0 的中间状态。初始情况下，迷宫如下图表示：

每次行动 agent 可以选择向不超过迷宫边界的相邻位置移动一格，通过 Q-Learning 算法希望 agent 最终找到一条通往宝藏的路径。



通过改变决策ϵ的大小，考察 agent 选择路径方式的变化。

1.6 背景知识

强化学习过程

- **智能体（agent）**：智能体是强化学习算法的主体，它能够根据经验做出主观判断并执行操作，是整个智能系统的核心。
- **环境（environment）**：智能体以外的一切统称为环境，环境在与智能体的交互中，能被智能体所采取的动作影响，同时环境也能向智能体反馈

状态和奖励。虽说智能体以外的一切都可视为环境，但在设计算法时常常会排除不相关的因素建立一个理想的环境模型来对算法功能进行模拟。

- **状态（state）**：状态可以理解为智能体对环境的一种理解和编码，通常包含了对智能体所采取决策产生影响的信息。
- **动作（action）**：动作是智能体所采取决策产生影响的方式，这里说的动作常常指概念上的动作，如果实在设计机器人时还需考虑动作的执行机构。
- **策略（policy）**：策略是智能体在所处状态下去执行某个动作的依据，即给定一个状态，智能体可根据一个策略来选择应该采取的动作。
- **奖励（reward）**：奖励是智能体序贯式采取一系列动作后从环境获得的利益。注意奖励概念是现实中奖励和惩罚的统合，一般用正值来代表实际奖励，用负值来代表实际惩罚。

如上图所示，人与环境交互的一种模型化表示，在每个时间点，大脑 agent 会从可以选择的

动作集合 A 中选择一个动作 at 执行。环境则根据 agent 的动作给 agent 反馈一个 reward，同时 agent 进入一个新的状态。根据图流程，强化学习的任务目标是获取尽可能多的 reward。reward 越多，就表示执行得越好。每个时间片，agent 根据当前的状态来确定下一步的动作。也就是说我们需要一个 state 找出一个 action，使得 reward 最大，从 state 到 action 的过程就称之为一个策略 Policy，一般用π 表示。

1.7 Q-Learning:

Q-Learning 是一个强化学习中一个很经典的算法，其出发点很简单，就是用一张表存储在各个状态下执行各种动作能够带来的 reward，如下表表示了有两个状态 s1，s2，每个状态下有三个动作 a1，a2， a3，表格里面的值表示 reward。

Reward	Action 1	Action 2	Action 3
State 1	-1	2	3
State 2	-4	2	-1

这个就是 Q-Table，里面的每个值定义为 Q(s,a)，表示在状态 s 下执行动作 a 所获取的 reward，那么选择的时候可以采用一个贪婪的做法，即选择价值最大的那个动作去执行。Q-Table 通过随机初始化来生成初始表格，然后通过不断执行动作获取环境的反馈并通过算法更新 Q-Table。更新规则如下：

$$Q'(s,a) = r + \gamma \max_{a'} Q(s',a')$$

贝尔曼(Bellman)方程也被称为动态规划方程(dynamic programming equation)，由理查德·贝尔曼(Richard Bellman)提出。贝尔曼方程描述了价值函数或是动作-价值函数的递推关系，是研究强化学习问题的重要手段。本实验代码中可以使用时序差分法(TD 法)公式求解更新参量，类似于监督学习中的梯度下降。