Lab5: Temporal Difference Learning

系所：多媒體工程研究所

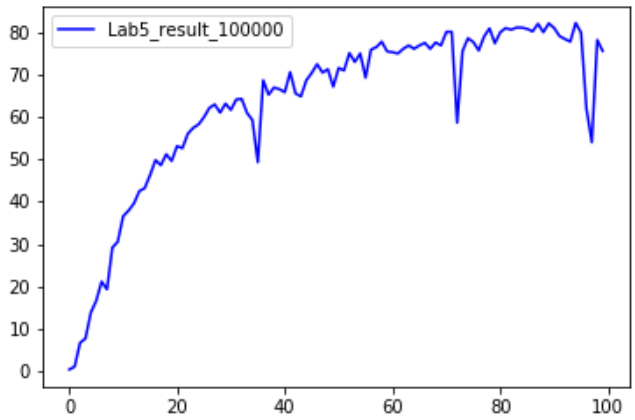
學號：0756616

姓名：周冠伶

1. Report (70%)

   - A plot shows episode scores of at least 100,000 training episodes (10%)

     第一次執行 100,000 次的結果：



   - Describe your implementation in detail. (10%)

     訓練過程：整體參數初始化完成後，首先判斷是否能夠繼續進行遊戲，如果可以繼續進行遊戲，評估各個動作的期望值並選擇最大值進行動作，再記錄最終結果作為回饋。

function PLAY GAME
  $score \leftarrow 0$
  $s \leftarrow$ INITIALIZE GAME STATE
  while IS NOT TERMINAL STATE(s) **do**
    $a \leftarrow \underset{a' \in A(s)}{\text{argmax}}$ EVALUATE(s, a')
    $r, s', s'' \leftarrow$ MAKE MOVE(s, a)
    SAVE RECORD(s, a, r, s', s'')
    $score \leftarrow score + r$
    $s \leftarrow s''$
  for (s, a, r, s', s'') FROM TERMINAL DOWNTO INITIAL **do**
    LEARN EVALUATION(s, a, r, s', s'')
  return $score$

     模型內容修改：

```cpp
float make_statistic(size_t n, const board &b, int score, int unit = 1000) {
  float thisEpochMax = 0.0;

  scores.push_back(score);
  maxtile.push_back(0);
  for (int i = 0; i < 16; i++) {
    maxtile.back() = std::max(maxtile.back(), b.at(i));
  }

  if (n % unit == 0) { // show the training process
    if (scores.size() != size_t(unit) || maxtile.size() != size_t(unit)) {
      error << "wrong statistic size for show statistics" << std::endl;
      std::exit(2);
    }
    int sum = std::accumulate(scores.begin(), scores.end(), 0);
    int max = *std::max_element(scores.begin(), scores.end());
    int stat[16] = {0};
    for (int i = 0; i < 16; i++) {
      stat[i] = std::count(maxtile.begin(), maxtile.end(), i);
    }
    float mean = float(sum) / unit;
    float coef = 100.0 / unit;
    info << n;
    info << "\t"
            "mean = "
         << mean;
    info << "\t"
            "max = "
         << max;
    info << std::endl;

    for (int t = 1, c = 0; c < unit; c += stat[t++]) {
      if (stat[t] == 0)
        continue;
      int accu = std::accumulate(stat + t, stat + 16, 0);
      info << "\t" << ((1 << t) & -2u) << "\t" << (accu * coef) << "%";
      info << "\t(" << (stat[t] * coef) << "%)" << std::endl;

      if ( ((1 << t) & -2u) == 2048 )
          thisEpochMax = (accu * coef);
    }

    scores.clear();
    maxtile.clear();
  }

  return thisEpochMax;
}
```

```cpp
int main(int argc, const char *argv[]) {
  info << "TDL2048-Demo" << std::endl;
  learning tdl;

  // set the learning parameters
  float alpha = 0.1;
  size_t total = 100000;
  unsigned seed;
  __asm__ __volatile__("rdtsc" : "=a"(seed));
  info << "alpha = " << alpha << std::endl;
  info << "total = " << total << std::endl;
  info << "seed = " << seed << std::endl;
  std::srand(seed);
  float trainEpochMax = 0.0;
  float thisEpochMax = 0.0;

  // initialize the features
  tdl.add_feature(new pattern({0, 1, 2, 3, 4, 5}));
  tdl.add_feature(new pattern({4, 5, 6, 7, 8, 9}));
  tdl.add_feature(new pattern({0, 1, 2, 4, 5, 6}));
  tdl.add_feature(new pattern({4, 5, 6, 8, 9, 10}));

  // restore the model from file
  tdl.load("Lab5_weight");
  std::ofstream outfile ("Lab5_result.txt");

  // train the model
  std::vector<state> path;
  path.reserve(20000);
  for (size_t n = 1; n <= total; n++) {
    board b;
    int score = 0;

    // play an episode
    debug << "begin episode" << std::endl;
    b.init();
    while (true) {
      debug << "state" << std::endl << b;
      state best = tdl.select_best_move(b);
      path.push_back(best);

      if (best.is_valid()) {
        debug << "best " << best;
        score += best.reward();
        b = best.after_state();
        b.popup();
      } else {
        break;
      }
    }
    debug << "end episode" << std::endl;

    // update by TD(0)
    tdl.update_episode(path, alpha);
    thisEpochMax = tdl.make_statistic(n, b, score);

    path.clear();

    if( trainEpochMax < thisEpochMax )
    {
        std::cout << "should save " << thisEpochMax << " > " << trainEpochMax << std::endl;
        trainEpochMax = thisEpochMax;
        tdl.save("Lab5_weight");
    }

    if( n%1000==0 )
        outfile << thisEpochMax << std::endl;
  }

  // store the model into file
  //tdl.save("Lab5_weight");
    outfile.close();

  return 0;
}
```

1. 回傳每 1000 unit 最高的勝率，並與目前訓練過程中最高的勝率相比：如果該次 unit 較高，則儲存模型權重；反之，不儲存。於每次訓練前讀取前次儲存之模型權重。
2. 修改儲存與讀取模型函數之變數。
3. 將每 1000 unit 之訓練結果儲存至自行建立的文字文件中。

```python
# custom function
def readFile():
    fileName = None

    root = tkinter.Tk()
    root.withdraw()
    root.wm_attributes("-topmost", 1)
    fileName = filedialog.askopenfilename(
        parent=root,
        title="Choose Training Data File",
        filetypes=[("Training Data", "*.txt")],
        multiple=False)

    return fileName


def getValue(fileName):
    file = open(fileName, "r")
    value = []
    for data in file.readlines():
        value.append(float(data.replace("\n", "")))

    return value


def plotValue(name, value):
    matplotlib.pyplot.plot(value, label=name, color='#0000FF')
    matplotlib.pyplot.legend(loc='best')
    matplotlib.pyplot.savefig(name)
    matplotlib.pyplot.show()
    matplotlib.pyplot.clf()
```

繪製結果：

1. 讀取於訓練過程中自行建立的勝率文字文件。
2. 使用 Matplotlib 繪製讀取之資料並儲存結果。

- Describe the implementation and the usage of nn-tuple network. (10%)

將盤面的排列組合以編碼方式呈現，其中會以特徵作為某種盤面的呈現與其得分計算方式，以節省所需記憶體空間以及加速計算。

程式碼：建立 N-tuple 特徵，並進行訓練。

```
931   // initialize the features
932   tdl.add_feature(new pattern({0, 1, 2, 3, 4, 5}));
933   tdl.add_feature(new pattern({4, 5, 6, 7, 8, 9}));
934   tdl.add_feature(new pattern({0, 1, 2, 4, 5, 6}));
935   tdl.add_feature(new pattern({4, 5, 6, 8, 9, 10}));
```

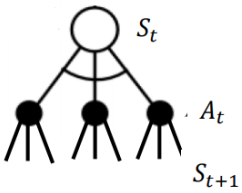
```
948     // play an episode
949     debug << "begin episode" << std::endl;
950     b.init();
951     while (true) {
952       debug << "state" << std::endl << b;
953       state best = tdl.select_best_move(b);
954       path.push_back(best);
955
956       if (best.is_valid()) {
957         debug << "best " << best;
958         score += best.reward();
959         b = best.after_state();
960         b.popup();
961       } else {
962         break;
963       }
964     }
965     debug << "end episode" << std::endl;
966
967     // update by TD(0)
968     tdl.update_episode(path, alpha);
969     thisEpochMax = tdl.make_statistic(n, b, score);
970
971     path.clear();
```

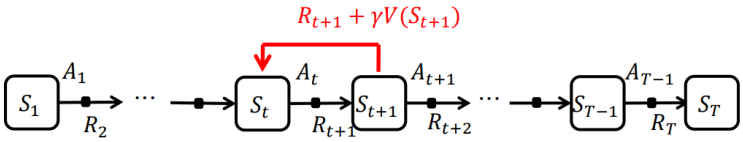
- Explain the TD-backup diagram of V(state). (5%) and explain the action selection of V(state) in a diagram. (5%)

$$\textbf{function } \text{LEARN EVALUATION}(s, a, r, s', s'')$$

$$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$$

使用當前的期望值更新次個狀態的期望值。





根據計算出的最大期望值決定動作。

- Explain the TD-backup diagram of V(after-state). (5%) and explain the action selection of V(after-state) in a diagram. (5%)

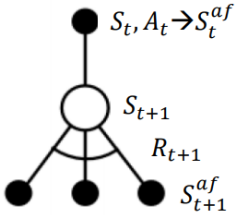
function LEARN EVALUATION(s, a, r, s', s'')

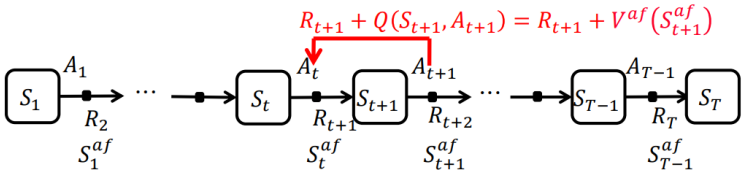
$$a_{next} \leftarrow \underset{a' \in A(s'')}{\operatorname{argmax}} EVALUATE(s'', a')$$

$$s'_{next}, r_{next} \leftarrow COMPUTE\ AFTERSTATE(s'', a_{next})$$

$$V(s') \leftarrow V(s') + \alpha(r_{next} + V(s'_{next}) - V(s'))$$

執行完動作後，利用重新計算後的期望值更新目前的期望值。





利用新的期望值更新前次的期望值。

- Explain the mechanism of temporal difference learning. (5%)

時間差學習是強化學習的一種，結合了動態規劃與蒙地卡羅的優點，主要機制是在每一次狀態的轉移時就進行學習。

與蒙地卡羅相比，時間差學習效率較好，因為時間差學習會在每一次轉移時就更新參數，不需要完成全部的訓練才能進行更新

- Explain whether the TD-update perform bootstrapping. (5%)

Bootstrapping 在強化學習中，表示在更新步驟中，使用一個或多個評估值來計算新的評估值。

在蒙地卡羅中，評估值政策使用 $V(S_t) \Leftarrow V(S_t) + \alpha[G_t - V(S_t)]$ ；在時間差學習中，是使用 $V(S_t) \Leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$ 來進行更新的。兩者都

是基於現有的評估值進行更新，但是兩者更新目標不同(G_t 、 $R_{t+1} + \gamma V(S_{t+1})$)。

- Explain whether your training is on-policy or off-policy. (5%)

  On-Policy 為行為策略與目標策略一致的方式，但是會出現陷入區域解的問題；為避免此問題，Off-Policy 就將兩者策略分離。

  2048 在強化學習訓練過程中，是利用評估結果進行動作、以回饋作為更新策略的，且不需要完成整個訓練即可更新參數，屬於 On-Policy。

- Other discussions or improvements. (5%)

  1. 雖然有儲存每次訓練後結果最好的模型，但是卻沒有儲存到該次訓練中最高的勝率。在下一次開始訓練時，可能會將較低的勝率覆蓋過前次訓練中最高勝率的模型權重。

  2. 不同數量的 n-tuple 設計：有對稱性的設計較能節省記憶體空間、相同數量但排列組合方式不同的設計會有不同的結果。

2. Performance (30%)

- The 2048-tile win rate in 1000 games, [winrate2048].

  訓練中最高勝率為 90.6%：

  ※90.5%為前一輪的最高勝率