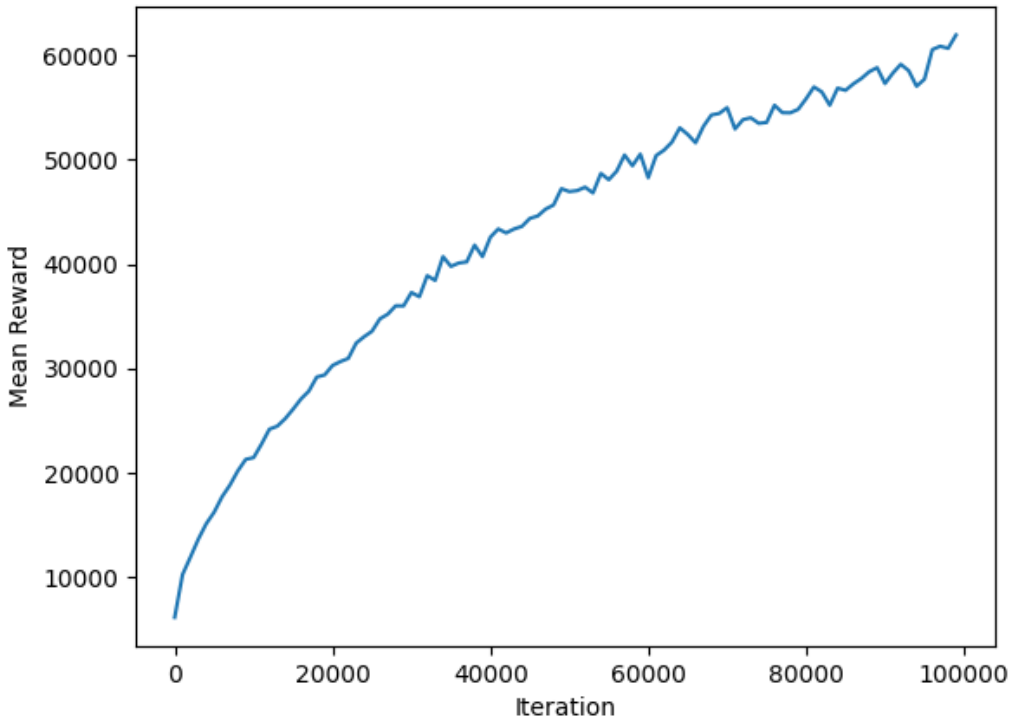
RL Topic HW1

1. Training Curve



2. Implementation and usage of n-tuple network
    a. Usage: n-tuple network would add all isomorphic pattern of each n-tuple of the current board to estimate the value.
    b. Implementation: First, add all value of the n-tuples, a 'feat' is a n-tuple pattern.

```
661     /**
662      * accumulate the total value of given state
663      */
664     float estimate(const board& b) const {
665         debug << "estimate " << std::endl << b;
666         float value = 0;
667         for (feature* feat : feats) {
668             value += feat->estimate(b);
669         }
670         return value;
671     }
```

and then query the summation of all 8 isomorphic of current n-tuple pattern.

```
/**
 * estimate the value of a given board
 */
virtual float estimate(const board& b) const {
    // TODO
    float val = 0.0;
    for(const auto& iso : const vector<int> &  : isomorphic) {
        size_t index = indexof( patt: iso, b);
        val += operator[]( i: index);
    }
    return val;
}
```

the ‘index_of’ would encode the numbers on board according to an isomorphic into a (4 * n) bits integer.

```
size_t indexof(const std::vector<int>& patt, const board& b) const {
    // TODO
    size_t index = 0;
    for(size_t i = 0; i < patt.size(); i++) {
        index |= b.at( i: patt[i]) << (4 * i);
    }
    return index;
}
```

3. Mechanism of TD(0)
    a. It would update the current value by an single step reward and next estimation, without the need of calculated the discounted total return.
4. Implementation detail
    a. Action selection:

For each valid action (valid means it would actually move the numbers), it will firstly find the empty spaces, then calculate the expectation of the board from generating a number (90% 2, 10% 4) on this empty space.

```
701             for (state* move = after; move != after + 4; move++) {
702                 if (move->assign(b)) {
703                     // TODO
704                     float total_value = 0.0;
705                     int total_space = 0;
706 
707                     for(int i = 0;i < 16;i++) {
708                         if(move->after_state().at(i) == 0) {
709                             total_space++;
710                             board temp = move->after_state();
711                             temp.set(i, t: 1);
712                             total_value += 0.9f * (float)estimate( b: temp);
713                             temp.set(i, t: 2);
714                             total_value += 0.1f * (float)estimate( b: temp);
715                         }
716                     }
```

After summing up all the expected values, take average of all the values, since all space would have a same probability to be selected to generate a number. Then we add the average with the action reward, then it would be the value regarding the action. Afterall, we select the action with the largest action value.

```
718                     total_value /= (float)total_space;
719                     total_value += (float)move->reward();
720 
721                     move->set_value( v: total_value);
722 
723                     if (move->value() > best->value())
724                         best = move;
```

b. TD-backup

We can get the TD-update by the formulation:

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

And the value of $r + V(s'') - V(s)$ can be easily calculated in a reversed way.

Since the update function would return the updated value, so when we have a path of $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_x$, then we can update s_2 by $\alpha\big(r_2 + V(s_x)[=0] - V(s_2)\big)$, then update s_1 by $\alpha\big(r_1 + V(s_2)[= update(s_2)] - V(s_1)\big)$.

```
747     void update_episode(std::vector<state>& path, float alpha = 0.1) const {
748         // TODO
749         float next_est = 0.0;
750         for(auto it1 : reverse_iterator<...>  = ++path.rbegin(); it1 != path.rend(); ++it1) {
751             float err = (float)it1->reward() + next_est - estimate( b: it1->before_state());
752             next_est = update( b: it1->before_state(), u: alpha * err);
753         }
754     }
```