

KAIST Include 동아리 스터디

AlphaGo와 AlphaGo Zero를 만들며 익히는 딥러닝 및 강화학습

Chris Ohk

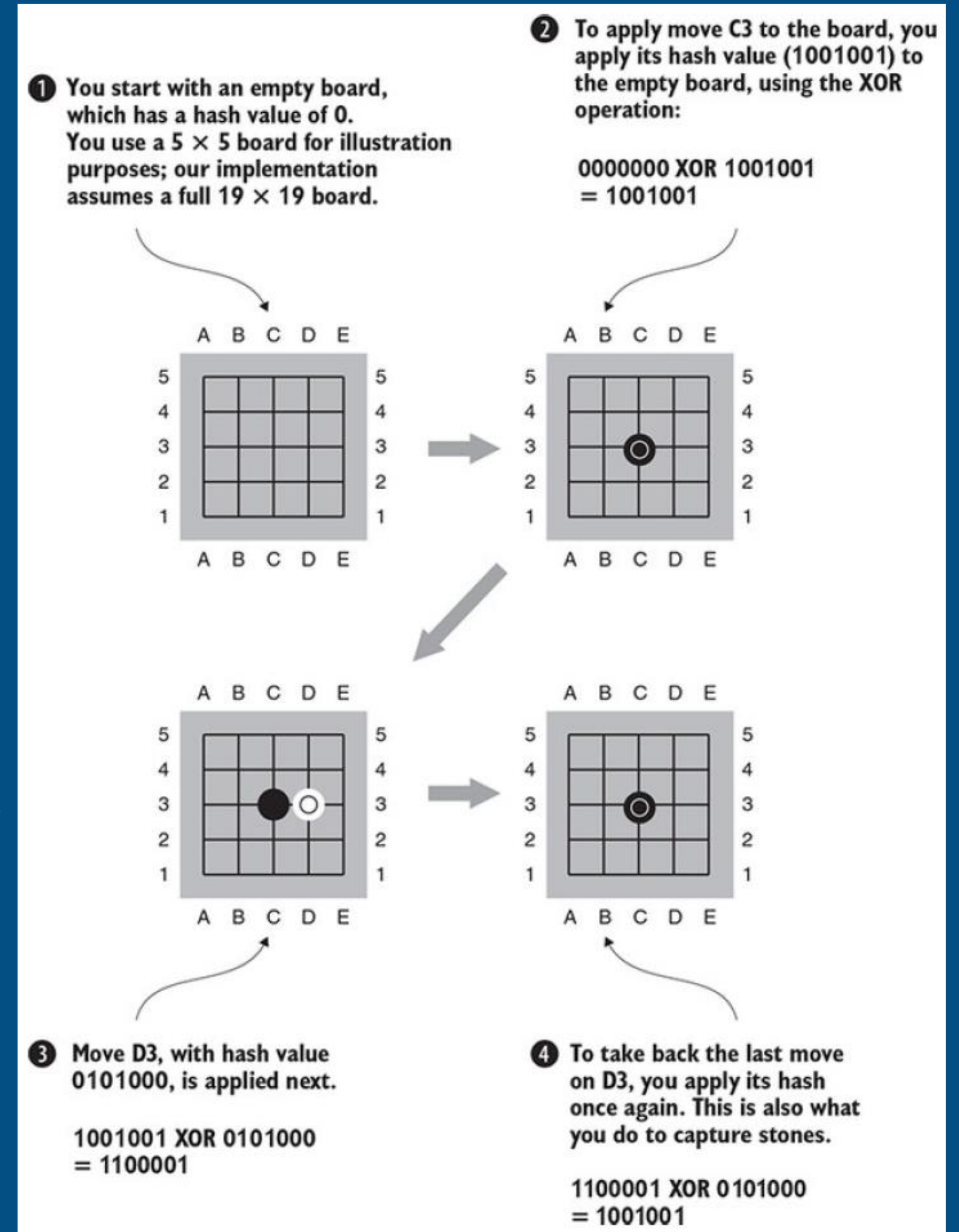
utilForever@gmail.com

- Zobrist 해싱을 사용한 대국 속도 향상
- 봇과 대국하기

- 동형반복 상황을 떠올려보자.
 - 현재 위치가 이전과 같은 경우였는지 확인하기 위해 게임 전체의 기록을 훑어야 했었다.
 - 자원 소모가 매우 심한 작업이다. 이 문제를 해결하려면 어떻게 해야 할까?
 - 바둑판에 놓인 돌의 모든 위치 상태값을 한꺼번에 저장하는 대신 훨씬 작은 해시값을 간단히 저장하면 된다.
 - 체스 같은 게임 분야에 널리 사용되는 해싱 기법 중 하나로 Zobrist 해싱이 있다.

- Zobrist 해싱에서는 바둑판의 가능한 수마다 해시값을 부여한다.
- 가장 좋은 방법은 각 해시값을 임의로 선택하는 것이다.
- 바둑에서 각 수는 흑과 백으로 이루어지므로 19×19 바둑판에서 Zobrist 해시 테이블 전체는 $2 \times 19 \times 19 = 722$ 개의 해시값으로 이루어진다.
- 이 722가지 해시값으로 바둑판에서 벌어지는 모든 상황을 표현할 수 있다.

- Zobrist 해시를 이용해 수를 나타내고
대국 상태를 효율적으로 저장하는 방법
- 처음에는 간단히 하기 위해 0으로 지정한 빈 바둑판의
해시값으로 경기를 시작한다.
- 첫 수는 해시값이 되므로 이 값을 바둑판의 값에 XOR
연산을 해서 추가한다. 이를 해시 적용이라고 한다.
- 해시 연산을 재적용하면 수를 이전으로 되돌릴 수 있다.
이를 해시 비적용이라고 한다. (XOR 연산의 편리함)



- Zobrist 해시 생성

```
import random

from dlgo.gotypes import Player, Point

def to_python(player_state):
    if player_state is None:
        return 'None'
    if player_state == Player.black:
        return Player.black
    return Player.white

MAX63 = 0x7fffffffffffffff

table = {}
empty_board = 0
for row in range(1, 20):
    for col in range(1, 20):
        for state in (Player.black, Player.white):
            code = random.randint(0, MAX63)
            table[Point(row, col), state] = code
```

```
print('from .gotypes import Player, Point')
print('')
print("__all__ = ['HASH_CODE', 'EMPTY_BOARD']")
print('')
print('HASH_CODE = {')
for (pt, state), hash_code in table.items():
    print(' (%r, %s): %r,' % (pt, to_python(state), hash_code))
print('}')
print('')
print('EMPTY_BOARD = %d' % (empty_board,))
```


- 돌과 활로에 대한 불변 집합형을 사용한 GoString 인스턴스

```
class GoString:
    def __init__(self, color, stones, liberties):
        self.color = color
        self.stones = frozenset(stones)
        self.liberties = frozenset(liberties)

    def without_liberty(self, point):
        new_liberties = self.liberties - set([point])
        return GoString(self.color, self.stones, new_liberties)

    def with_liberty(self, point):
        new_liberties = self.liberties | set([point])
        return GoString(self.color, self.stones, new_liberties)
```

- 빈 바둑판에 _hash 값을 넣어 바둑판을 인스턴스화하기

```
from dlgo import zobrist

class Board:
    def __init__(self, num_rows, num_cols):
        self.num_rows = num_rows
        self.num_cols = num_cols
        self._grid = {}
        self._hash = zobrist.EMPTY_BOARD
```


- 돌을 놓는 것 = 해당 돌의 해시값을 적용하는 것

```
new_string = GoString(player, [point], liberties)

for same_color_string in adjacent_same_color:
    new_string = new_string.merged_with(same_color_string)
for new_string_point in new_string.stones:
    self._grid[new_string_point] = new_string

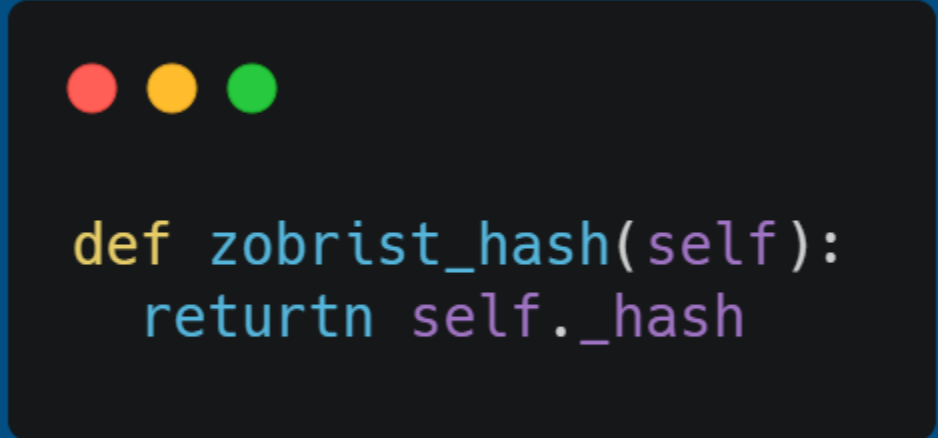
self._hash ^= zobrist.HASH_CODE[point, player]

for other_color_string in adjacent_opposite_color:
    replacement = other_color_string.without_liberty(point)
    if replacement.num_liberties:
        self._replace_string(other_color_string.without_liberty(point))
    else:
        self._remove_string(other_color_string)
```

- 돌을 제거하는 것 = 돌의 해시값을 비적용하는 것

```
def _replace_string(self, new_string):  
    for point in new_string.stones:  
        self._grid[point] = new_string  
  
def _remove_string(self, string):  
    for point in string.stones:  
        for neighbor in point.neighbors():  
            neighbor_string = self._grid.get(neighbor)  
            if neighbor_string is None:  
                continue  
            if neighbor_string is not string:  
                self._replace_string(neighbor_string.with_liberty(point))  
        self._grid[point] = None  
  
    self._hash ^= zobrist.HASH_CODE[point, string.color]
```

- 판의 현재 Zobrist 해시 값을 반환

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. It contains a Python function definition for `zobrist_hash`.

```
def zobrist_hash(self):  
    return self._hash
```

- Zobrist 해시로 경기 상태 초기화

```
class GameState:
    def __init__(self, board, next_player, previous, move):
        self.board = board
        self.next_player = next_player
        self.previous_state = previous
        if self.previous_state is None:
            self.previous_states = frozenset()
        else:
            self.previous_states = frozenset(
                previous.previous_states |
                {(previous.next_player, previous.board.zobrist_hash())})
        self.last_move = move
```

- Zobrist 해시로 패 상태를 빠르게 확인하기

```
def does_move_violate_ko(self, player, move):  
    if not move.is_play:  
        return False  
    next_board = copy.deepcopy(self.board)  
    next_board.place_stone(player, move.point)  
    next_situation = (player.other, next_board.zobrist_hash())  
    return next_situation in self.previous_states
```

- 바둑판을 더 빠르게 만드는 방법
 - goboard_slow.py : 2주차 스터디 때 만들었던 바둑판 파일
 - goboard.py : Zobrist 해싱을 사용해 빠르게 만든 바둑판 파일
 - goboard_fast.py : 객체를 새로 만들지 않고 복제해서 더 빠르게 만든 바둑판 파일
- goboard_fast.py 파일을 분석하고 goboard.py 파일과 비교하면서 어떻게 더 빠르게 만들었는지 확인해 보자.

- 2주차 스터디에서 스스로 자체 대국을 치루는 약한 봇을 만들었다.
이제 사람과 봇이 바둑을 둘 수 있는 프로그램을 만들어 보자.
- 모두 작성한 뒤 다음 명령을 입력해 잘 동작하는지 확인해 보자.

```
python human_v_bot.py
```


- 사람의 입력값을 좌표로 변환해 바둑판에 표기

```
def point_from_coords(coords):  
    col = COLS.index(coords[0]) + 1  
    row = int(coords[1:])  
    return gotypes.Point(row=row, col=col)
```

- 봇과 직접 게임할 수 있는 스크립트 만들기

```
from dlgo import agent
from dlgo import goboard_slow as goboard
from dlgo import gotypes
from dlgo.utils import print_board, print_move, point_from_coords
from six.moves import input

def main():
    board_size = 9
    game = goboard.GameState.new_game(board_size)
    bot = agent.RandomBot()

    while not game.is_over():
        print(chr(27) + "[2J")
        print_board(game.board)
        if game.next_player == gotypes.Player.black:
            human_move = input('-- ')
            point = point_from_coords(human_move.strip())
            move = goboard.Move.play(point)
        else:
            move = bot.select_move(game)
        print_move(game.next_player, move)
        game = game.apply_move(move)

if __name__ == '__main__':
    main()
```

감사합니다!

스터디 듣느라 고생 많았습니다.