Intelligent Systems 2020: 6th practical assignment

Machine Learning Agents

Your name: Seeun Park

Your VUnetID: spk760

If you do not provide your name and VUnetID we will not accept your submission.

Preliminaries

At the end of this exercise you should be able to work with some basic Machine Learning concepts, and implement and evaluate a learning-based approach to playing Schnapsen. In this notebook we are going to create an adaptive bot. We will use the principle discussed in the machine learning lecture, but now in an agent setting. This comes down to using basic hill-climbing search, but learn the heuristic function rather than implementing it. This will require a few basic ingredients:

> Script that plays games between existing bots and creates a dataset to learn from. The dataset contains each observed state, labeled with the (eventual) winner of the game. See the script train-ml-bot.py.
> A function that translates a state object to a feature vector. See the function features(...) in ml.py
> An implementation with a hill-climbing bot that gets its heuristic from a machine learning model. See bots/ml/ml.py

Feature vectors were discussed in the lecture. Didn't get it, or working ahead? See https://brilliant.org/wiki/feature-vector/ https://www.youtube.com/watch?v=3Vy47dbI708

Practicalities

Follow this Notebook step-by-step. For this course it is necessary that you manipulate the python programmes we provide. You can do the exercises in any Programming Editor of your liking. Still, please fill in the questions in this notebook as usual. You can also run tournaments in it if you want, but running them in your editor or via the commandline seems much more convenient.

Please use your studentID+Assignment6.ipynb as the name of the Notebook, and fill in the missing cells.

Note: unlike the courses dedicated to programming we will not evaluate the style of the programs. But we will, however, test your programs on other data that we provide, and your program should give the correct output to the test-data as well.

As was mentioned, the assignment is graded as pass/fail. To pass you need to have either a full working code or an explanation of what you tried and what didn't work for the tasks that you were unable to complete (you can use multi-line comments or a text cell).

Train a Machine Learning Model

The plan is as follows: we run the train-ml-bot.py script, which creates a model for us, and places it in the bots/ml directory. All you need to do is to is to complete the feature extraction method in bots/ml/ml.py It returns a basic feature vector modelling the properties of the game state, or more precisely the bot's perspective of the game state (which means that in phase 1 of the game there are parts of the feature values unknown (for the cards that are either in the adversaries hands or in the pile).

To complete the function, you'll need to write some code which transforms information you get from state.py into integer values.

To run the bots using the commandline/terminal:

```
- If you want to play 2 bots against each other, e.g. rand and bully:
python play.py -1 rand -2 bully
- To see what other options there are: python play.py --help
- If you run python tournament.py it'll play a round-robin tournament
between bully, rand and rdeep where every pair of players play 10
matches. Run python tournament.py --help to see how you can change the
players, and the number of games played (if needed).
```

Task 1

Fill in the missing code (all the '???' lines) and run a number of games to check whether your agent "works". You can either run the play.py script in a command line, or copy the play code from one of the previous notebooks here (do not forget to import all the necessary modules and code).

Please copy you code in the following cell

In [ ]:
```
next_state = state.next(move)

    # Add player 1's points to feature set
    p1_points = state.get_points(1)

    # Add player 2's points to feature set
    p2_points = state.get_points(2)

    # Add player 1's pending points to feature set
    p1_pending_points = state.get_pending_points(1)

    # Add plauer 2's pending points to feature set
    p2_pending_points = state.get_pending_points(2)

    # Get trump suit
    trump_suit = state.get_trump_suit()
```

```
    # Add phase to feature set
    phase = state.get_phase()

    # Add stock size to feature set
    stock_size = state.get_stock_size()

    # Add leader to feature set
    leader = state.leader()

    # Add whose turn it is to feature set
    whose_turn = state.whose_turn()

    # Add opponent's played card to feature set
    opponents_played_card = state.get_opponents_played_card()
```

Run a tournament between rand, bully and ml, and copy the result of the tournament in the following cell.

In [1]:
```
import sys, random

from api import State, engine, util
```

In [8]:
```
botnames = []
verbose = False
myphase = 1
myrepeats = 10

# Create player 1
player1 = util.load_player("rand")
player2 = util.load_player("bully")
player3 = util.load_player("ml")

bots = [player1,player2,player3]

n = len(bots)
wins = [0] * len(bots)
matches = [(p1, p2) for p1 in range(n) for p2 in range(n) if p1 < p2]

totalgames = (n*n - n)/2 * myrepeats
playedgames = 0

print('Playing {} games:'.format(int(totalgames)))
for a, b in matches:
    for r in range(myrepeats):

        if random.choice([True, False]):
            p = [a, b]
        else:
            p = [b, a]

        # Generate a state with a random seed
        state = State.generate(phase=myphase)

        winner, score = engine.play(bots[p[0]], bots[p[1]], state, 1000, verbose, True)

        if winner is not None:
            winner = p[winner - 1]
            wins[winner] += score

        playedgames += 1
```

```
        print('Played {} out of {:.0f} games ({:.0f}%): {} \r'.format(playedgames, tota

print('Results:')
for i in range(len(bots)):
    print('    bot {}: {} points'.format(bots[i], wins[i]))
```

```
Traceback (most recent call last):
  File "C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\api\util.py",
line 73, in load_player
    player = cls() # Instantiate the class
  File "C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\bots\ml\ml.p
y", line 29, in __init__
    self.__model = joblib.load(model_file)
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\joblib\n
umpy_pickle.py", line 577, in load
    with open(filename, 'rb') as f:
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\seeun\\Desktop\\Inte
lligent Systems\\Practical\\Assignment 6\\bots\\ml/model.pkl'
ERROR:root:Internal Python error in the inspect module.
Below is the traceback from this internal error.

C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\bots\ml/model.pkl
ERROR: Could not load the class "Bot" Bot from file ./bots/ml/ml.py.
Traceback (most recent call last):
  File "C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\api\util.py",
line 73, in load_player
    player = cls() # Instantiate the class
  File "C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\bots\ml\ml.p
y", line 29, in __init__
    self.__model = joblib.load(model_file)
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\joblib\n
umpy_pickle.py", line 577, in load
    with open(filename, 'rb') as f:
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\seeun\\Desktop\\Inte
lligent Systems\\Practical\\Assignment 6\\bots\\ml/model.pkl'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython
\core\interactiveshell.py", line 3418, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-8-0ae074189e58>", line 9, in <module>
    player3 = util.load_player("ml")
  File "C:\Users\seeun\Desktop\Intelligent Systems\Practical\Assignment 6\api\util.py",
line 78, in load_player
    sys.exit()
SystemExit

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython
\core\ultratb.py", line 1170, in get_records
    return _fixed_getinnerframes(etb, number_of_lines_of_context, tb_offset)
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython
\core\ultratb.py", line 316, in wrapped
    return f(*args, **kwargs)
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython
\core\ultratb.py", line 350, in _fixed_getinnerframes
    records = fix_frame_records_filenames(inspect.getinnerframes(etb, context))
  File "c:\users\seeun\appdata\local\programs\python\python39\lib\inspect.py", line 152
9, in getinnerframes
```

```
    frameinfo = (tb.tb_frame,) + getframeinfo(tb, context)
AttributeError: 'tuple' object has no attribute 'tb_frame'
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\Desktop\Intelligent Systems\Practical\Assignment 6\api\util.py in load_player(name, cl
assname)
     72         cls = getattr(module, classname)
---> 73         player = cls() # Instantiate the class
     74         player.__init__()

~\Desktop\Intelligent Systems\Practical\Assignment 6\bots\ml\ml.py in __init__(self, ran
domize, model_file)
     28         # Load the model
---> 29         self.__model = joblib.load(model_file)
     30

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\joblib\numpy_pic
kle.py in load(filename, mmap_mode)
    576     else:
--> 577         with open(filename, 'rb') as f:
    578             with _read_fileobject(f, filename, mmap_mode) as fobj:

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\seeun\\Desktop\\Inte
lligent Systems\\Practical\\Assignment 6\\bots\\ml/model.pkl'

During handling of the above exception, another exception occurred:

SystemExit                                Traceback (most recent call last)
    [... skipping hidden 1 frame]

<ipython-input-8-0ae074189e58> in <module>
      8 player2 = util.load_player("bully")
----> 9 player3 = util.load_player("ml")
     10

~\Desktop\Intelligent Systems\Practical\Assignment 6\api\util.py in load_player(name, cl
assname)
     77         traceback.print_exc()
---> 78         sys.exit()
     79

SystemExit:

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
    [... skipping hidden 1 frame]

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\int
eractiveshell.py in showtraceback(self, exc_tuple, filename, tb_offset, exception_only,
 running_compiled_code)
   2036                     stb = ['An exception has occurred, use %tb to see '
   2037                            'the full traceback.\n']
-> 2038                     stb.extend(self.InteractiveTB.get_exception_only(etype,
   2039                                                                      value))
   2040                 else:

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ult
ratb.py in get_exception_only(self, etype, value)
    821         value : exception value
    822         """
--> 823         return ListTB.structured_traceback(self, etype, value)
    824
    825     def show_exception_only(self, etype, evalue):
```

```
c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in structured_traceback(self, etype, evalue, etb, tb_offset, context)
    696             chained_exceptions_tb_offset = 0
    697             out_list = (
--> 698                 self.structured_traceback(
    699                     etype, evalue, (etb, chained_exc_ids),
    700                     chained_exceptions_tb_offset, context)

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in structured_traceback(self, etype, value, tb, tb_offset, number_of_lines_of_context)
   1434         else:
   1435             self.tb = tb
-> 1436         return FormattedTB.structured_traceback(
   1437             self, etype, value, tb, tb_offset, number_of_lines_of_context)
   1438

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in structured_traceback(self, etype, value, tb, tb_offset, number_of_lines_of_context)
   1334         if mode in self.verbose_modes:
   1335             # Verbose modes need a full traceback
-> 1336             return VerboseTB.structured_traceback(
   1337                 self, etype, value, tb, tb_offset, number_of_lines_of_context
   1338             )

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in structured_traceback(self, etype, evalue, etb, tb_offset, number_of_lines_of_context)
   1191         """Return a nice text document describing the traceback."""
   1192
-> 1193         formatted_exception = self.format_exception_as_a_whole(etype, evalue, etb, number_of_lines_of_context,
   1194                                                                tb_offset)
   1195

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in format_exception_as_a_whole(self, etype, evalue, etb, number_of_lines_of_context, tb_offset)
   1149
   1150
-> 1151         last_unique, recursion_repeat = find_recursion(orig_etype, evalue, records)
   1152
   1153         frames = self.format_records(records, last_unique, recursion_repeat)

c:\users\seeun\appdata\local\programs\python\python39\lib\site-packages\IPython\core\ultratb.py in find_recursion(etype, value, records)
    449     # first frame (from in to out) that looks different.
    450     if not is_recursion_error(etype, value, records):
--> 451         return len(records), 0
    452
    453     # Select filename, lineno, func_name to track frames with

TypeError: object of type 'NoneType' has no len()
```

I assume that the bots will win in the following order; ml > bully > rand

Task 2:

The first thing we can do to improve the bot, is to improve the quality of the games it observes. Change the player in train-ml-bot.py to a kbbot and/or rdeep player, and retrain the model. You may wish to lower the number of games played in train-ml-bot.py if the games are taking a long

time. Please describe in the following cell what you can observe when running a tournament like before after training ml.

By retraining the bot with a better quality of a game, ml will be able to perform even better. Since kbbot and rdeep have stronger strategies, ml would be able to score higher and have the possibility to prevent the other bots to score any points.

Training in different phases

Using alphabeta for training might not be a good idea, since it has to start in phase 2 with perfect information. This may not translate so well to phase 1 gameplay. Nevertheless, it is a good idea to experiment. If you wish to do this, you have to specify in train-ml-bot.py that the training games start in phase 2.

Task 3

Re-run the tournament. Does the machine learning bot do better? Show the output, and mention which bot was used for training.

By training it against rdeep ml will be able to win with a higher score gap with rand and bully since rdeep is the strongest bot that exists.

Testing more than one ML agent

We will need a more robust way of testing different machine learning approaches against each other. Change the training script so that it doesn't overwrite the previous model. Now write a script that creates two ml players with different models and plays games between them. This might then look like this:

```
from bots.ml import ml
player1 = ml.Bot(model_file='./models/rand-model.pkl')
player2 = ml.Bot(model_file='./models/rdeep-model.pkl')
```

Read and train-ml-bot.py carefully for inspiration.

Task 4

Make three models: one by observing rand players, one by observing rdeep players, and one by observing one of the ml players you made earlier. Describe the experiments you run, and their results in the next cell.

Rand model would be the one with the lowest grade since it is the weakest out of the players. Rdeep would be in the middle since it is the strongest discluding ml. The one that observed the ml players would have been able to score the highest with the biggest gap amongst the two others since it includes both the rand and rdeep observed players.

Improving the set of features

You may not see a lot of improvement for clever tricks like this. This is because the game has a lot of belief states, i.e. has an extremely broad search tree. The machine learning model only sees as small proportion, and chances are that no “similar” game has been described in the training set. Maybe the card deck and number of won or lost points.

To improve this, we might need better features. Think of some simple additional features and add them to the features() method in ml.py.

> Note that this means the bot can no longer use your old models, since they rely on 4-dimensional feature vectors. You'll most likely want to create a copy of ml.py for every feature-extraction strategy you would like to try, or add different feature extractors as a parameter to the bot.

Feature extraction is an art. You want to translate the information in the state into numbers in a way that makes sense to a linear model. We'll discuss this in-depth in the lectures. To start with, just try and think of numbers you can compute from the state that are high if the state is good for player one and low if the state is bad. You might want to add combinations of important features to create a design matrix, as discussed in class.

Task 5

Add some simple features and show that the player improves. Describe the features you added, copy their code and copy the result of the tournement into the following cell.

I would add the feature to play all marriages when having the opportunity to as well as the royal marriage trick.

Feature engineering

Finally, since coming up with features is an ad-hoc business, you'll want to test features you come up with to see if they actually add to the performance. How would you go about this? Could there be features that depend on each other? Ie. add feature A or B separately and there's no improvement, but add them together and the bot gets better?

Task 6

As shown in the lecture, adding the product of existing features (a design matrix) is a simple way to increase the power of your method without changing models.

Try to add at least 2 combined features to your feature table and evaluate it in a number of tournaments. Describe the new features and copy the code in the next cell.

Also, copy the result of the experiments and an interpretation in your own words.

I would try to combine playing the marriage and in order to increase the possiblity to play a marriage, I would code it to keep the queen and kings and play the jack, 10 or ace instead unless the only cards left were queens and kings. In this case I would make sure that the queen or king played would not be from the trump suit. Moreover, I'd make sure that if the king or queen that was

needed in order to play a marriage it would be categorized to be able to play and not keep anymore.

Knock yourself out

Of course, there is a wealth of other things to explore. Here are some things you can try out which will be similar to the things to do for

1) Have a look at the sklearn documentation: http://scikit-learn.org/stable/modules/classes.html It's a bit complex, but maybe you can figure how to use different machine learning models. The logistic regression we used is a very simple starting point.

2) Evaluate your model on the dataset by cross validation. See if you can improve the performance by tweaking its parameters,

Have fun. To be continued in Project Intelligent Systems in Period 3.