TD-Learning for Connect Four

Help file on the usage of the C4GPF GUI

Last Update: May2015   -   Markus Thill, Wolfgang Konen

Contents

1   The Main Window

The **Connect-4 Game Playing Framework** (C4GPF) is a framework for loading

After the start of the program C4GPF (**Connect-4 Game Playing Framework**) the main window appears. The window can be divided in six sections:

1. The training- and game-panel on the right of the window
2. The menu in the upper part of the window
3. The Connect Four board in the left
4. The value-panel beneath the board and
5. Overall Results in the bottom-left corner of the window
6. The Status-Bar in the bottom-right corner of the window
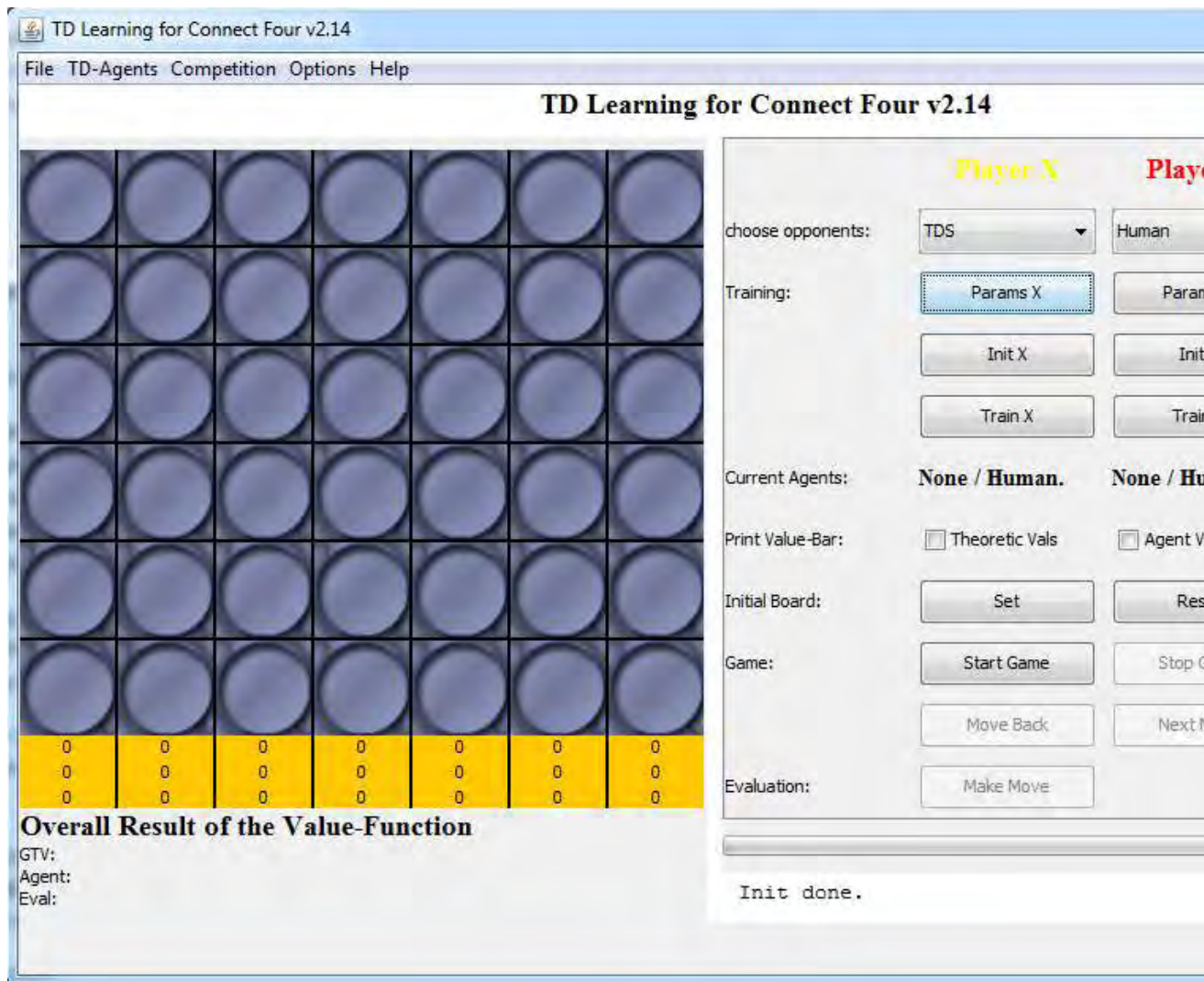
Figure 1 The main window

1.1 The Training- and Game-Panel

This panel is for initializing and training the available agents.

1.1.1 The Players

There are three different players:

1. Player X: This is the player with the yellow stones who begins the games. When playing perfect, yellow will always win.
2. Player O: This is the player with the red stones who always moves after the yellow one. Even if this player plays perfect, it will lose against a perfect playing opponent.
3. Evaluation: This player can be used to run simultaneous to other agents in a game to compare the values or to make single moves during a game. This can especially be useful for matches with two agents playing against each other. For instance Player X can be initialized with a Minimax-agent and Player Evaluation with a trained TD-agent. After each move of the Minimax-agent the user can use the "Make Move" -button to let the TD-Agent make the next move. Player O should not be initialized in this case.

1.1.2   Choosing the Opponents

For each player different kind of agents can be selected. This can be done in the line "choose opponents" with the three combo-boxes. If no agent is selected, then the user has to make the moves for that player during a game. Before the agents can work, they have to be initialized (and trained). This can be seen in the line "Current Agents".

1.1.3   Parameters for the Agents

Depending on which agents were chosen a click on the button "Params" will open a parameter-window for the specified agent. For a random agent or a human player there are no windows available and the button will do nothing. Parameters can currently be changed for Minimax-Agents and TD-agents.

1.1.4   Initialize the Agents

After the parameters for the agent have been set, the agent can be initialized. This has to be done for all agents. If parameters are changed, the agent has to be re-initialized for the changes to apply. Note that some agents also have to be trained after being initialized to work properly. If a agent shall be removed, simply select the human-player and re-init the selected player. If a player is not initialized during a game, then the moves are made by the user.

1.1.5   Training the Agents

Train the selected Agent. This has to be done for all TD-agents with a training-routine. If no training routine is found, then the agent will be re-initialized. Make sure that the agent is initialized or no training will be performed.

1.1.6   Current Agents

When a agent is initialized, this will be shown in the line "Current Agents".

1.1.7   Options for the Value-Bar

In the line "Print-Value-Bar" the user can choose which values are displayed in the "Value-Bar". The Value-Bar consists of three lines. First the user can choose if the values of Player X or Player O shall be displayed in the value-bar (in the second line). Then it can be chosen, if the game-theoretic-values (the real values for a perfect player) and the values for the player Evaluation shall be displayed. If the checkboxes are activated these values will be updated during a game.

1.1.8   Set/Reset the Initial Board

In some cases it can be helpful to set a initial board before starting a game. This can be done with the button "Set" in the line "Set initial board". The user then can make different moves in the Connect Four board in the left of the window. When finished, the user can press the same button to set the current board as initial one. The reset-button removes all pieces form the board.
The initial board is also used in some cases for competitions between two agents.

1.1.9   Playing a Game

Start a new match between the two players (X and O). If no agent is selected for the specified player, the moves are made by the user. This can simply be done by

selecting a column in the Connect Four board on the left of the window. When playing with agents note that these must be initialized (and trained) to make moves during the game. To make moves for a agent, simply select the Step-Button.

1.1.10 Previous and Next Boards

It is possible to take moves back or redo moves during a game. Take a move back during a game: If the opponents are two humans, then always ONE piece will be taken from the board. If one opponent is an agent, then always two pieces will be taken from the board, so that it's the human players move again. If two agents are playing against each other, then this button has no function.

1.1.11 Moves of the Player "Evaluation"

If the player "Evaluation" is an initialized/trained agent, then this player can make moves for the human player during a game. This can be used to study a trained agent.

1.2 The Menu

The menu contains some main functions for TD-Agents with N-Tuple-Systems and allows performing competitions etc.

1.2.1 The File Menu

Currently this menu only consists of an item to close the program.

1.2.2 The TD-Agents Menu

Allows operations on the players X, O and Evaluation. These operations will only work if the players are initialized with a TD-Agent. The same operations are provided for each player.

1.2.2.1 Load Agent

Open a COMPLETE TD-Agent with N-Tuples from a file. All lookup-tables and configurations of the agent will be loaded and assigned to the selected agent. Make sure that the selected file contains a TD-Agent. Otherwise nothing will be done. The agent should not be reinitialized or trained after loading from file, this would reset all weights of the LUTs.

1.2.2.2 Save Agent

Save a COMPLETE TD-Agent. All lookup-tables and configurations of the agent will be saved to HDD. Make sure, that enough HDD-memory is available (all lookup-tables together can need a lot of memory)

1.2.2.3 Import Agent Weights

Import the weights of an N-Tuple-TD-Agent from a ZIP file. The ZIP-file should contain two text files with the weights for Player X and Player O in the format described by Simon Lucas in http://algoval.essex.ac.uk:8080/othello/html/NTuple.html. (Note that we make a small extension to this format: In the first line we code not only T, the

number of N-tuples, but also P, the number of states for each cell. Each N-tuple of lengths N has P^n weights. In Connect-4 we use P=4 throughout.)

The ZIP file is usually created with Export Agent Weights. It is smaller than the agt.zip-file created by Save Agent (4 MB instead of 20 MB), but it takes longer to import or export.

1.2.2.4   Export Agent Weights
Export the weights of an N-Tuple-TD-Agent to a ZIP file. See Import Agent Weights for further details.

1.2.2.5   Show / Change N-Tuples
Show and change the N-Tuples of a TD-Agent. This operation doesn't work for other agent-types. The selected agent also must be initialized for this to work. If the used N-Tuples change, then the agent also must be initialized again!
A new window is opened that allows a fast selection of the single tuples. New sampling points can be added to a tuple simply by selecting empty fields on the Connect Four board. Existing sampling points can be removed by deselecting the occupied fields. The new window also allows the complete removal of N-Tuples and the creation of new ones. Make sure that all changes are saved and the agent is re-initialized afterwards.

1.2.2.6   Inspect LUT
Inspect the lookup-tables (LUTs) for the selected TD-Agent. For this purpose a new window will be opened.

1.2.2.7   Quick Evaluation
Allows a quick evaluation of the selected TD-agent. The TD-agent will play a certain number of matches against a perfect-playing Minimax-agent. Currently all boards with zero, one and two pieces will be used as initial boards for these matches. The TD-agent will always play for the theoretical winner (X = yellow or O = red). If the board leads to a draw, the TD-agent will play both players once. After each match it is checked if the TD-agent could reach the theoretical result.

After all these boards are tested, the most important part of the evaluation will start. The TD-Agent will still play against a perfect Alpha-Beta-agent from an empty board. But this time the Alpha-Beta-Agent will make more random moves, so that every match will be different. Mistakes of the TD-Agent will still be punished directly so that he loses the game. The Alpha-Beta-Agent will also prevent direct losses in his next ten moves, so that the TD-Agent has to hold steady and make correct moves. After the opening phase (currently 12 pieces) the Alpha-Beta-Agent will seek for the most distant loss so that the TD-agent has to play perfect to end to win the game. Currently 50 of these matches are performed between both agents.

All matches are used to determine the score. The score lies in the range 0 (TD-agent NEVER reached the possible theoretical best state) and +1 (TD-agent ALWAYS reached the possible theoretical best state).

1.2.2.8 Multi-Training

Allows a multiple number of training-processes for one TD-agent. During the training the strength of the agent is measured all X games by playing matches against another agent. The user can choose between a single match or multiple matches on different boards for every measuring interval. The Alpha-Beta-Agent plays randomly, so that every match will be different. Mistakes of the TD-Agent will still be punished directly so that he loses the game. The Alpha-Beta-Agent will also prevent direct losses during his next ten moves . All options can be changed in another window. The results are displayed in a separate window after the training is finished. For this purpose the score for every game-interval is calculated. It is also possible to show the results for the single game-intervals and training-procedures.

1.2.3 The Competition-Menu

Allows single- and multi-competitions between two agents and some tests for the value function of a single agent.

1.2.3.1 Competitions

The first and second player for single- and multi-competitions are selectable in the options. So if the first player is assigned with Player O, then player O will be the beginning player in the competitions, although he is only second in normal games. That's why also the player Evaluation can be selected too. For a competition to start both players have to be initialized with an agent. Multi-competitions just consists of several single-competitions. For every match of the multi-competition the same initial board will be used. But the initial board can be specified by the user before the competition begins. After the competition is over, the results are displayed in another window. For multi-competitions there is one overview-page printed first, but it is also possible to view the results of the single matches.

There is currently a small bug connected with multi-competition: If the agents behave deterministically, all games in the multi-competition will be the same. To make a more assessment of an agent's strength, please use Quick Evaluation which includes suitable random moves in a multi-game evaluation.

1.2.3.2 Testing the Value Function

Besides the competition it is also possible to check the value-function of a single agent. For this purpose a specific number of random boards is chosen and the values of the selected agent compared to the real ones. Since the values can't always be equal, the user can specify a spread-range around the real value for interpreting a agent-value as correct or at least in the correct range. After all options are specified, the test of the value-function can be started. The results are displayed in a new window.
A second functionality that this menu offers, is checking if the selected agent can find the best move in special situations. Therefore again a certain number of random boards are selected with the exception that only one move is the

correct one. In the test it will be checked if the agent is able to find these correct moves and the results will be displayed to a new window.

1.2.4 The Options Menu

The options-menu currently only contains one item. With this item the user can change some options for determining the game-theoretic-values (GTV) in the value-bar. There are three parameters that can be changed:

1. Set the Size of the Hash-Table for the Agent that determines the game-theoretic-values. If the Hash-Size is made smaller, more time is needed to calculate the score for a position.
2. Choose which databases shall be used for the Agent that determines the game-theoretic-values. The usage of the databases needs more memory but fastens the agent significantly.
3. Set the Search Depth for the Agent that determines the game-theoretic values. The search-depth should be the predefined value. If the depth is chosen smaller, than the values won't be GTVs, because a heuristic is used to evaluate positions at the leafs.

1.3 The Connect Four Board

The primary function of the board is the visualization of games. The user can make moves by simply selecting one of the fields in the desired column. The last move is always identified by the white corners in the specified field.

Another function of the board is displaying (Show N-Tuples) the N-Tuples of an TD-agent. The user can also change existing N-Tuples (Set / Change N-Tuples).

1.4 The Value Panel

The Value-Panel lies directly beneath the Connect Four board and consists of three lines. The first line is associated with the game-theoretic-values. These values are the "real" values for a perfect playing agent. The second line is associated with the agent-values (Player X or O). The last line is reserved for the evaluation-values of the player evaluation.

The lines again consist of 7 columns. Each of these labels represent the current value for the single columns of the board. The values lie in the range -100 .. +100. Negative values indicate an advantage for the second player O (red), positive values on the other hand indicate an advantage for the beginning player X (yellow).

Figure 2 Association of the Checkboxes to the lines of the value-panel

1.5   The Overall Results

Like the value-panel the overall results are divided in three lines for the game-theoretic-value, the agent- and the evaluation-value. In contrast to the value-panel, which shows the values for the single columns of the board, these values indicate the outcome of the game for the current board. The agent- value indicates the outcome of the game for the last move made by the agent.

1.6   The Status Bar

The Status-Bar consists of a progress-bar and a label which show the current state for longer lasting operations.

2   The Minimax-Parameters Window

When selecting the "Params"-button for a Minimax-player this window will be opened. Normally the option Use presetting should be activated. In this case an already existing standard-agent will be assigned to the player during the initialization. This can save a lot of memory.
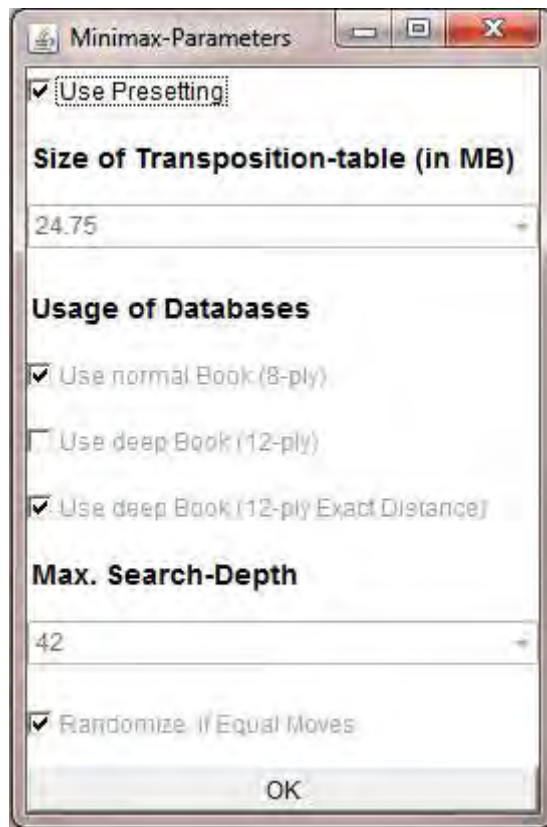
Figure 3 Options for a Minimax-player

There are four parameters that can be changed:

1. Set the Size of the Hash-Table for the selected player. If the Hash-Size is made smaller, more time is needed to calculate the score for a position.
2. Choose which databases shall be used for the selected player. The usage of the databases needs more memory but fastens the agent significantly. There are 3 books available. The 8-ply-database is the smallest with only about 100 KB and should always be selected. The 12-ply-database contains all positions with 12 pieces and needs about 7 MB memory. The 12-ply-database with exact distance is the biggest database with 21 MB. But this database allows to determine the exact distance to a win / loss - situation for perfect players. This can be helpful when a Minimax-agent is trying to find a near win or a distant loss.
3. Set the Search Depth for the selected player. If the depth is chosen smaller than the predefined values, then the values won't be GTVs (real values), because a heuristic is used to evaluate positions at the leaves. For a perfect playing agent the maximum search depth must be 42.
4. Should the agent find equal good moves by default, one of these moves will be selected randomly.

3   The TD-Parameters Window

When selecting the "Params"-button for a TD-player this window will be opened. This window is divided into the sections General, IDBD (or any other method for online adaptable learning

rates), TCL, N-Tuples and Index-Hashing. In the following we describe the parameters for all sections.

It is possible to save and load parameter files (.tdPar) with the buttons **Save** and **Open** at the bottom of this dialog.

Figure 4 Options for a TD-player

3.1 General

The general section contains basically all parameters – with exception of the Learning Rate Adaption algorithms – that are used in the classical TD-Learning algorithm.

Learning Rate Adaption: In the first combo-box a learning-rate adaption method can be chosen. Based on the selection certain elements in the window will be activated or deactivated. Until now, the following learning-rate adaption methods are supported:

- TCL: Temporal Coherence Learning. This method is described in [Beal1999] and [Beal2000]. We extended this method by introducing an exponential scheme [Bagheri14] and [Thill14]. Details on the parameter-settings for TCL can be found in section TCL.
- IDBD: Incremental Delta-Bar-Delta (IDBD) was first proposed by Sutton for linear value functions [Sutton92]. Recently, also non-linear versions of IDBD were introduced, such as Koop's non-linear IDBD ([Koop08] and [Sutton07]) and Konen's extension IDBD-WK [Konen14]. Koop's IDBD-nl is derived for the logistic sigmoid function, which we use in this case instead of tanh. Details can be found in IDBD (and others).
- AUTOSTEP: Also Mahmood's AUTOSTEP can be seen as an extension for IDBD [Mahmood10, Mahmood12], although it is limited to linear units. This method promises a low sensitivity on its meta step size parameter μ (when $\mu \ll 1$), which could also be shown for in several applications. Details can be found in IDBD (and others).
- K1: Sutton's K1 algorithm [Sutton92a] is a derived from the original IDBD algorithm using a normalized LMS. Details can be found in IDBD (and others).
- ELK1: Schraudolph [Schraudolph99] extended the K1 algorithm to ELK1 (extended, linearized K1). Details can be found in IDBD (and others).
- Alpha-Bounds: In contrast to the other algorithms Dabney's Alpha-Bounds [Dabney12] only computes one global step-size parameter α, which is used for all weights of the system. The original version of alpha-bounds only is derived for linear units. We extended the algorithm to non-linear units as well [Bagheri14].
- RProp: Several versions of the RProp-algorithm can be found in the literature. Currently, we only implemented iRProp+, as described in [Riedmiller93].

Load Best: With the button "Load Best" for every Learning Rate Adaptation algorithm the (so far) best known parameters (based on the criterion "time-to-learn") are selected.

Alpha init: Initial global learning rate

Alpha final: Final global learning rate

Epsilon Init: Initial probability for explorative moves

Epsilon final: Final probability for explorative moves

Lambda: Trace decay parameter. The value can be chosen in the range $0 \leq \lambda \leq 1$. Value $\lambda > 0$ activate eligibility traces and allow further options (check-boxes "Reset on random move" and "Replacing traces"). Details on the different options are described in [Thill14].

Gamma: Discount-rate parameter.

Reset on rand. move: Reset the trace vector, if the agent performs a random move.

Replacing Traces: Activate replacing traces. When activated, traces will not accumulate. Instead, every time a weight is activated, the corresponding trace will be replaced by the new value (see [Thill14] for details).

Epsilon Change: There are four methods to change the probability for explorative moves during the training:

1. Linear: Epsilon will change linearly from the initial value to the final value at the end of the training
2. Exponential: Epsilon will change exponential from the initial- to the final value

$$\varepsilon(n) = \left(\frac{\varepsilon_N}{\varepsilon_0}\right)^{\frac{n}{N}} \cdot \varepsilon_0$$

3. Usage of a sigmoid-function (tanh): The used function has following form:

$$\varepsilon(n) = (\varepsilon_0 - \varepsilon_N) \cdot \left[\frac{1 - \tanh\left(\frac{n - n_{wp}}{N}\right) m}{2}\right] + \varepsilon_N$$

with the total amount of games N and the point of inflection n_{wp}. The parameter m

is set fix to 10 but can be changed in the source. Note that the point of inflection is

set in the field "Extra Param". A typical graph for N = 3000 with n_{wp} =1500 could
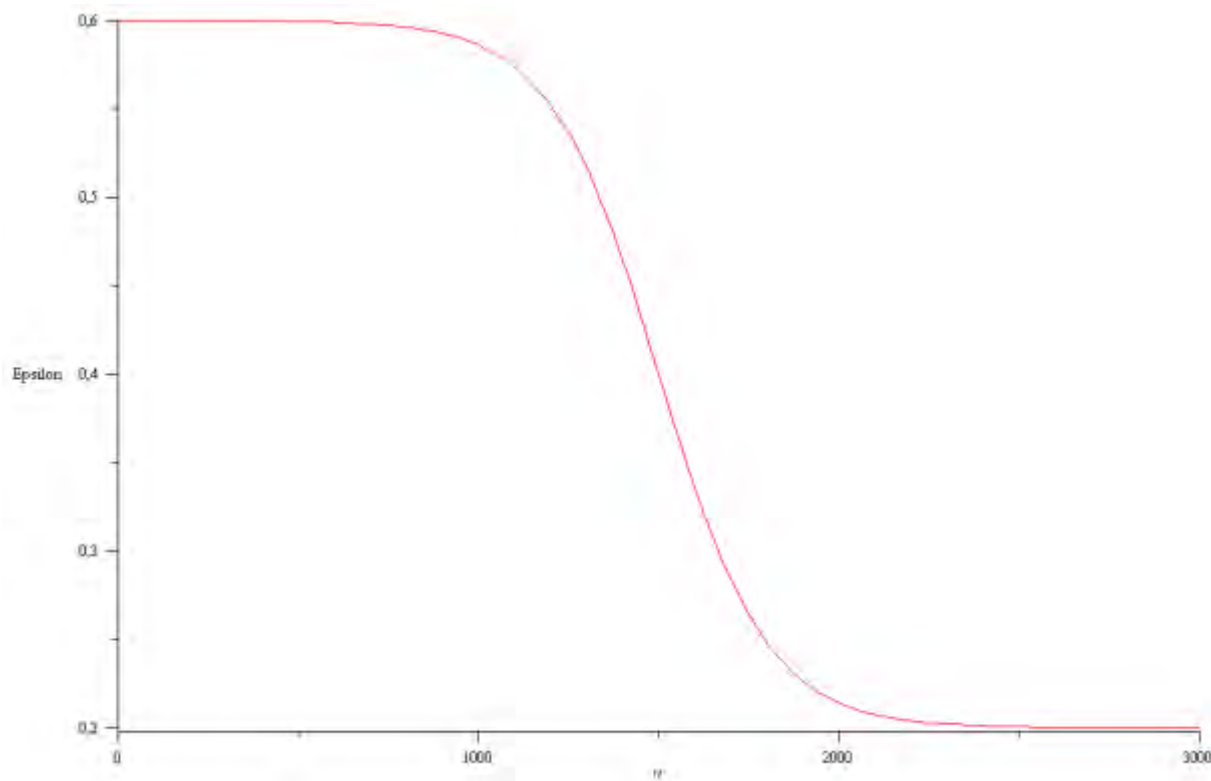
look like this:

Figure 5 Using a sigmoid-function for the adjustment of Epsilon

4. Usage of a simple Jump: At a certain point epsilon jumps form the initial value to the final one. This point is defined in the text field "Extra Param".

Extra Param: Extra Parameter for some adjusting-methods of epsilon (see above).

Activation: Choose an activation function, which can be $\tanh(y)$ or $\frac{1}{1+e^{-y}}$, with y as the output of the linear unit. If "linear" is selected, y remains unchanged.

Use Bias Weight: Use a single bias weight additionally to the n-tuple system. The input x_{bias} is always $x_{bias} = 1$.

n-Ply Look-ahead: During and after the training the agent can look ahead n moves. If n is assigned with 1 the agent will first try all possible moves and get the scores for these after-states. The best score (for the current player) will be selected from these and returned. This functionality corresponds to the simple Minimax-algorithm where the values at the leafs are fetched from the trained value-function (or environment) .

Game Number: Number of training games for one training-process

Reward after X moves: It is possible to train the opening-phase of the game. If a value X<42 is set, then all training-games will be stopped after X moves and the reward will be fetched from the environment.

Info-Interval: After X training games the current values for alpha and epsilon and printed to the console and the current strength of the trained agent is measured (see below).

Evaluation Matches: No. of evaluation matches against the perfect Minimax player at the end of each interval. The intervals for the evaluation of the agent can be changed with the button "Change Intervals".

Change Intervals: Opens a new Window in which the evaluation intervals of the TD agent can be defined. An interval is defined in a certain range of training games (start and end); the evaluation-points in this range are determined by a step-parameter. At every evaluation-point the strength of the TD agent is measured by playing against Minimax.
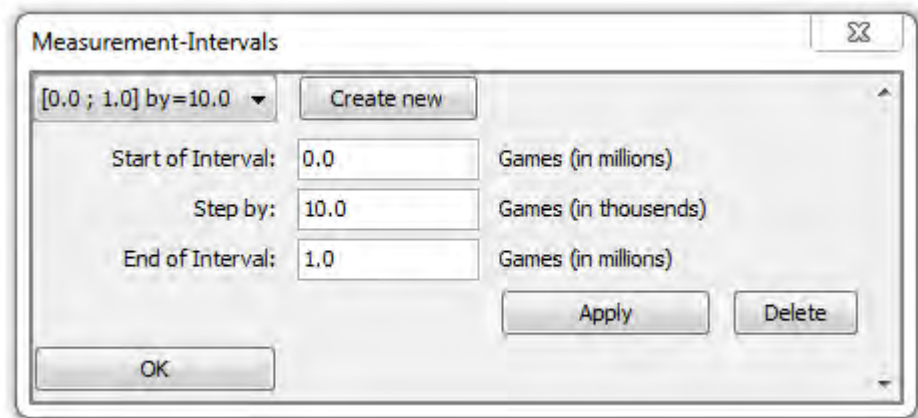

Figure 6 Window for defining different evaluation intervals.

3.1 IDBD (and others)

Almost all online adaptable learning rate algorithms require an initial value for the individual learning rates of the system. This value is defined by β_{init}. The corresponding initial learning rate is $\alpha_{init} = e^{\beta_{init}}$. Another relevant parameter is the meta step size parameter θ (μ in AUTOSTEP). The right choice of both parameters – β_{init} and θ – depend on many aspects, such as the number and the length of the n-tuples, the select learning rate algorithm (IDBD, AUTOSTEP, K1, …) and more. We provide for every method values that have proven to be suitable for the default n-tuple system (see "Load Best" in section General).   The IDBD version IDBD_WK has an additional parameter ω_k, as described in [Konen14].

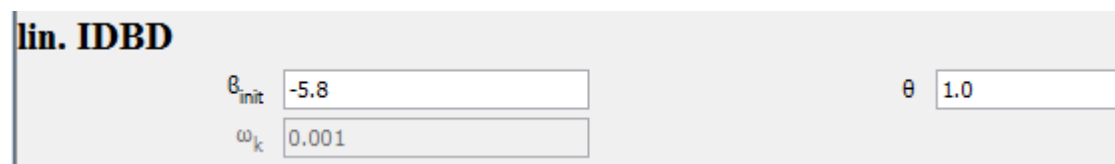Figure 7 Parameters for different online adaptable learning rate algorithms such as IDBD, AUTOSTEP, K1 and more.

3.2 TCL

Temporal Coherence Learning (TCL) has rather different parameters than the other learning rate algorithms. Therefore an own section is defined only for TCL. In the following we list the possible parameters. Details on different parameters can be found in [Beal99, Beal00, Bagheri14]:

Use Update Episodes: Activates update episodes, so that the weight updates and the learning rate adaptions are performed after an episode is completed. We define an episode as a certain number of moves.

Episode-Length: This input-field is activated when the usage of update-episodes is chosen. The episode-length defines the number of moves, after which the weights of the system and the corresponding individual learning rates are updated.

Mu-init: Decay $\mu_{init} \leq 1$ for the values A_i and N_i. In every update-step, all adapted A_i and N_i are multiplied by μ_{init} so that more recent events have higher impact on the learning rates. Typically the values should be in the range $0.9 \leq \mu_{init} \leq 1$, the default value is 1.

Weight-Factors – Weights vs. Weights – Weight-factors: Defines the operational order of an update-step. If the first option is selected, then the weight factors A_i and N_i are updated first and subsequently the corresponding weight w_i. Bei default the second option is selected, which delivered better results in our experiments and is the standard procedure mentioned in [Beal00, Beal99].

Use Error Signal vs. Use rec Weight Change: Allows to choose between the error signal δ_t and the recommended weight change $r_{i,t} = \delta_t \nabla_{w_i} V(w, s)$ to update the weight factors A_i and N_i. See [Bagheri14] for details.

Use exp. Scheme: Activates the exponential scheme for TCL, described in [Bagheri14].

Beta: An additional parameter needed when activating the exponential scheme for TCL.

3.3   N-Tuples

By default, an n-tuple system with 70 predefined 8-tuples is used for the training process. It is possible to randomly create a new n-tuple set as soon as the agent is initialized by activating the corresponding checkbox (Figure 9). There are different approaches for the generation process, described in the following:

Figure 9 Options for the n-tuple generation process.

N-Tuple-Type: There are currently four different types of N-Tuple-Systems available for the TD-Agents:

1. Random sample-points, length of all tuples equal
2. Random sample-points, length of all tuples random (2 … tupleLen)
3. Random Walk, length of all tuples equal
4. Random Walk, length of all tuples random (2 .. tupleLen)

Tuple-Num: Number of N-Tuples to be generated

Tuple-Length: (max.) N-Tuple-Length

Use Symmetry: If this checkbox is activated, the board will be mirrored at the center column in some operations. This should ensure a faster training of the agent.

Random weight init: If this checkbox is activated all weights in all LUTs (look-up-tables) will be initialized with small random numbers.

Pos. Vals / field: The user can choose between 3 or 4 possible values / field:

1. Three values: A field can be occupied by player X or O or simply be empty
2. Four Values: like three, but empty fields are divided in empty and directly reachable and empty and NOT reachable.

3.4   Index Hashing

During the training process the TD agent has to calculate the indexes for all LUTs of the n-tuple system for a certain board position many times. The indexing-process requires some computation time that can be saved if all indexes corresponding to a board position are saved in a hash-table. This needs some additional memory, but can speed up the training significantly.

Save: This allows the user to save the TD-Parameters inclusive the current N-Tuples (without the LUTs) to HDD. Not that only the parameters are saved. How to save whole agents can be read here.

Open: Open TD-Parameters from a file on HDD.

4   Competition Options Window

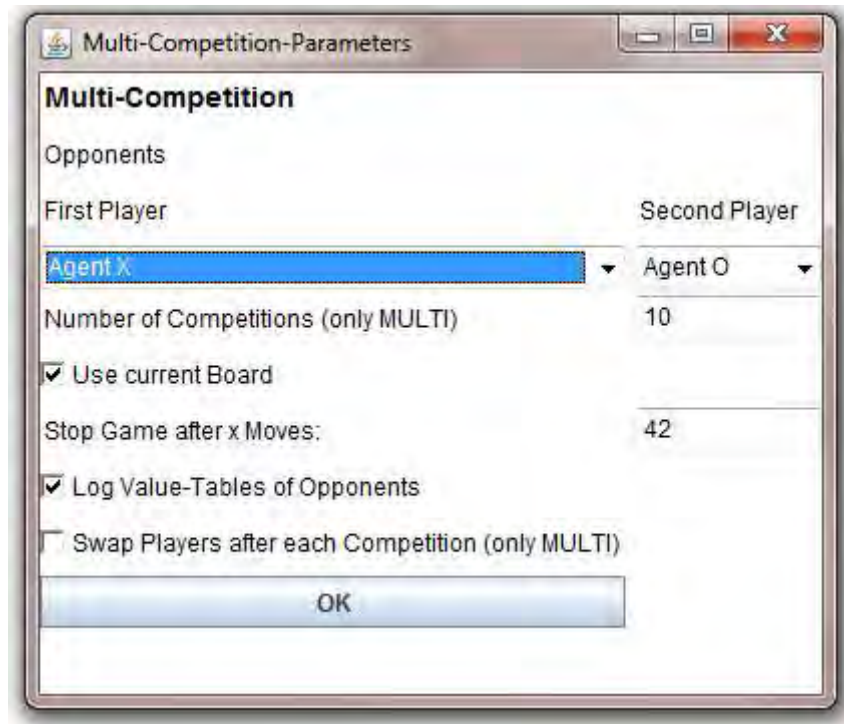When selecting the options-item in the "Competition-Submenu" this window will be opened:

Figure 10 Options for single- and multi-competitions

Opponents: The first and second player for single- and multi-competitions are selectable in the options. So if the first player is assigned with Player O, then player O will be the beginning player in the competitions, although he is only second in normal games. That's why also the player Evaluation can be selected too.

Number of competitions: This text field is only relevant for multi-competitions

Use current board: If this checkbox is selected the current Connect Four Board (How to set the initial board) will be used for all competitions.

Stop game after X moves: If one of the agents has only trained the opening-phase, then the competition can be aborted after X moves and a referee checks which agent would have theoretically won.

Log value-tables of the opponents: The values for all columns can be logged for both agents for every move made. This may cost more time, especially if the Minimax-agent is used.

Swap players after each competition: This checkbox is only relevant for multi-competitions. If this checkbox is activated the number of matches will be doubled because both opponents swap their role (beginning or second player) after each game.

5   Test-Value-Function Options Window

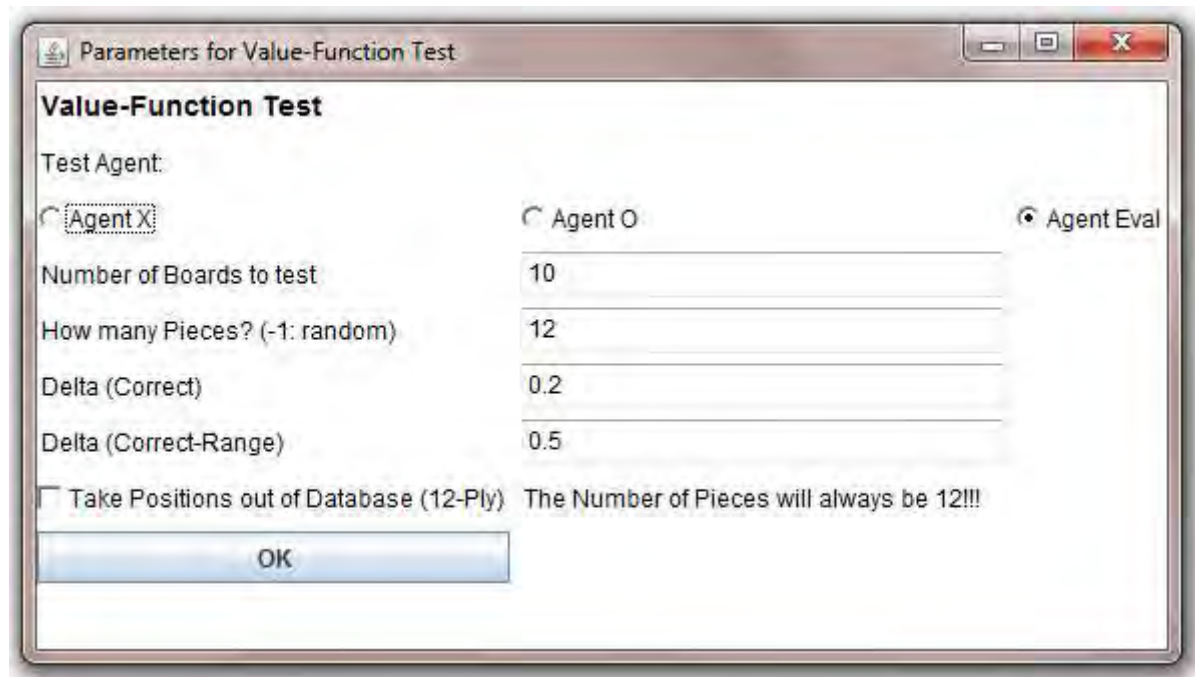When selecting the options-item in the "Test Value Function submenu" this window will be opened:

Figure 11 Options for testing the value-function

Test agent: Only one of the three players X, O and Evaluation can be tested.

Number of boards to test: A specific number of random boards is chosen for the tests

How many pieces: Number of pieces on the random boards.

Delta (Correct): Spread-range around the real value for interpreting an agent-value as correct

Delta (Correct range): Spread-range around the real value for interpreting an agent-value in the correct range

Take positions out of database (12-ply): If this checkbox is activated the testing-routine will always take positions from the 12-ply database. This can fasten the tests significantly, because no "referee" is needed to evaluate the results.

6   Inspecting the LUTs

This window can be used for debugging and analysis-purposes (only TD-Agents with an N-Tuple-System). There are two main modes which have completely different approaches. Both modes are explained below.
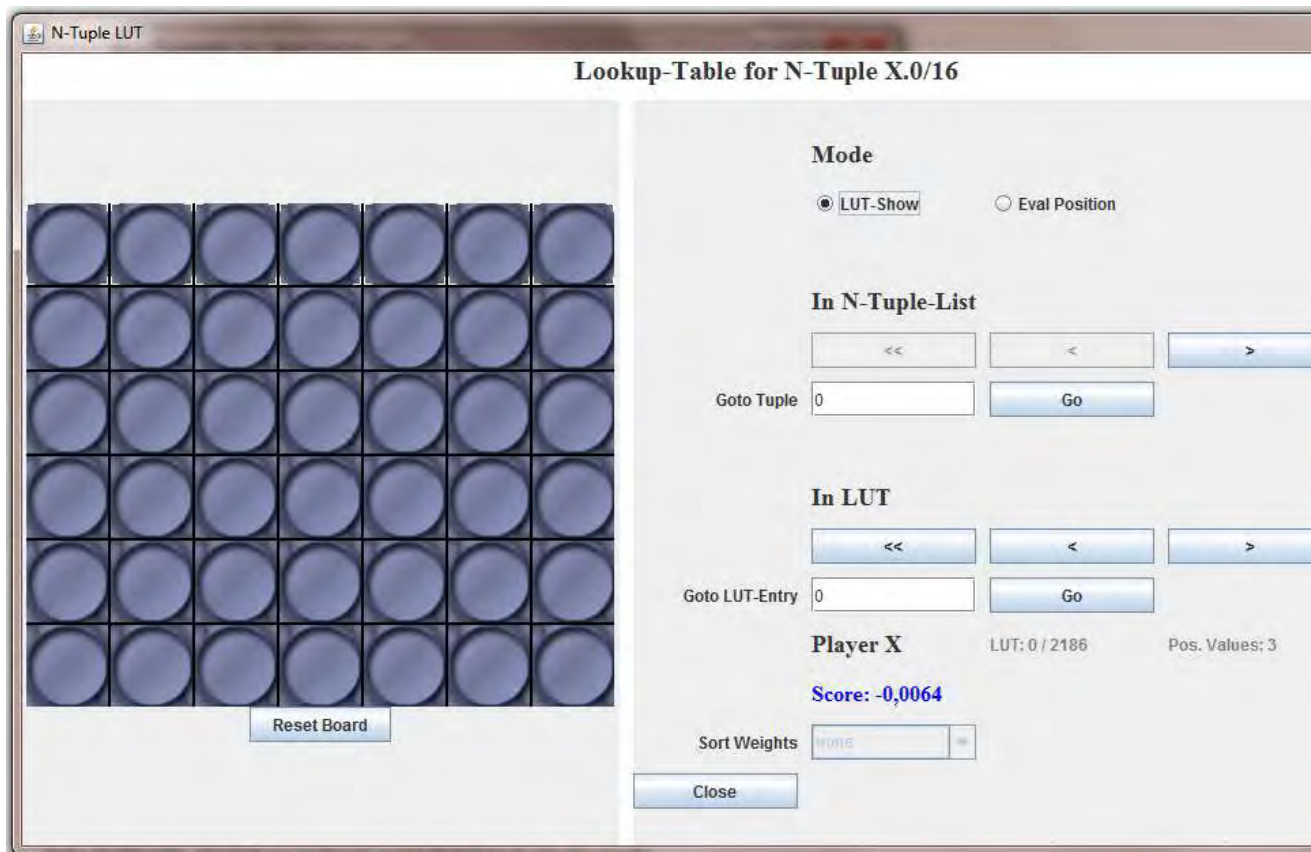
Figure 12 Main window for the LUT-Inspection

6.1 Mode: LUT-Show

In this mode it is possible for the user to view single entries in the different LUTs (lookup-tables). For every N-Tuple two LUTs are generated, one for player X (yellow) and one for player O. The user can see in the title which LUT is currently selected. For example a title "Lookup-Table for N-Tuple O.0/16" indicates that currently the first from 17 (34 for both players) LUTs for player O is selected. The user should always check which LUT (Player X or O) is selected for an N-Tuple.

The fields with the white-marked corners in the Connect Four board show the sampling points of the current N-Tuple. In this mode the user can only select and deselect these marked fields. With every selection of a sampling-point on the field the state of the field changes from empty to yellow, from yellow to red and from red to empty. If 4 possible values / field are used for the N-Tuple-System the state empty will turn to empty and reachable (indicated with a red bar at the bottom of the field).

The occupancy of the single sampling points results in a LUT-entry-Nr. This entry-Nr. (position in the LUT) is showed in the section "In LUT" on the right side of the window. The weight / score of this LUT-entry is displayed within the blue label. All weights of a LUT can be viewed by changing the states of the sampling points in the Connect Four board on the left or simply by moving through the LUT with the provided buttons. Another possibility is typing an entry number in a text field.
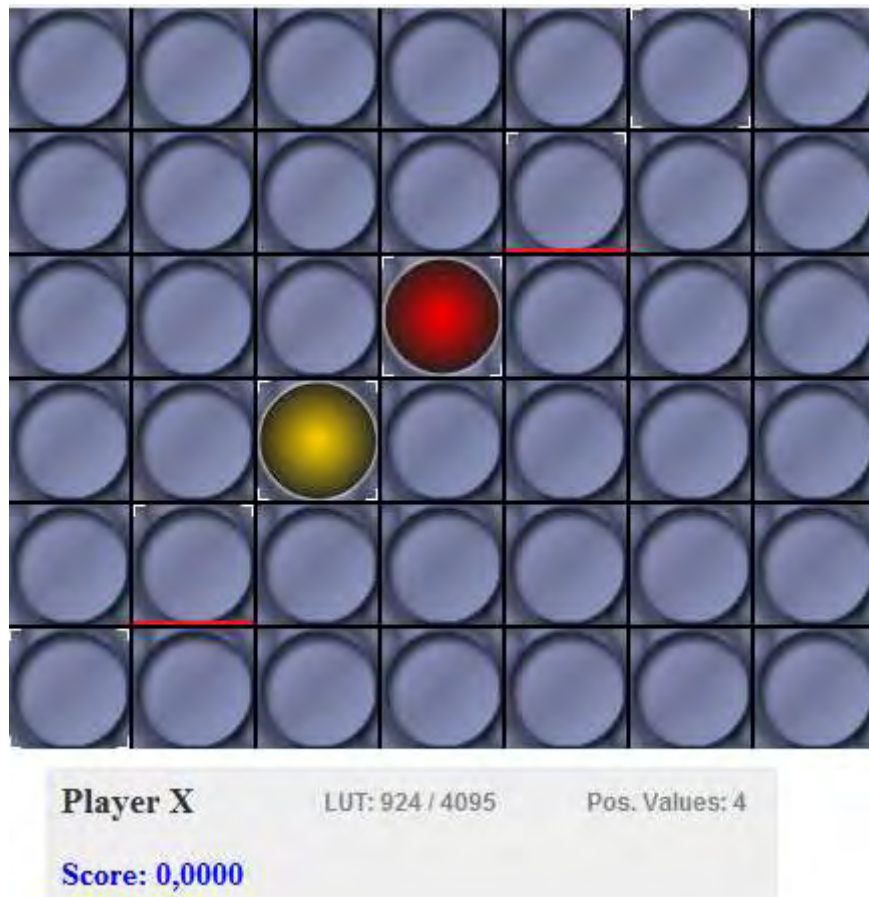
Figure 13 Example for a single LUT-entry. The red bars indicate empty but reachable fields

Another window allows fast switching between the single N-Tuples.
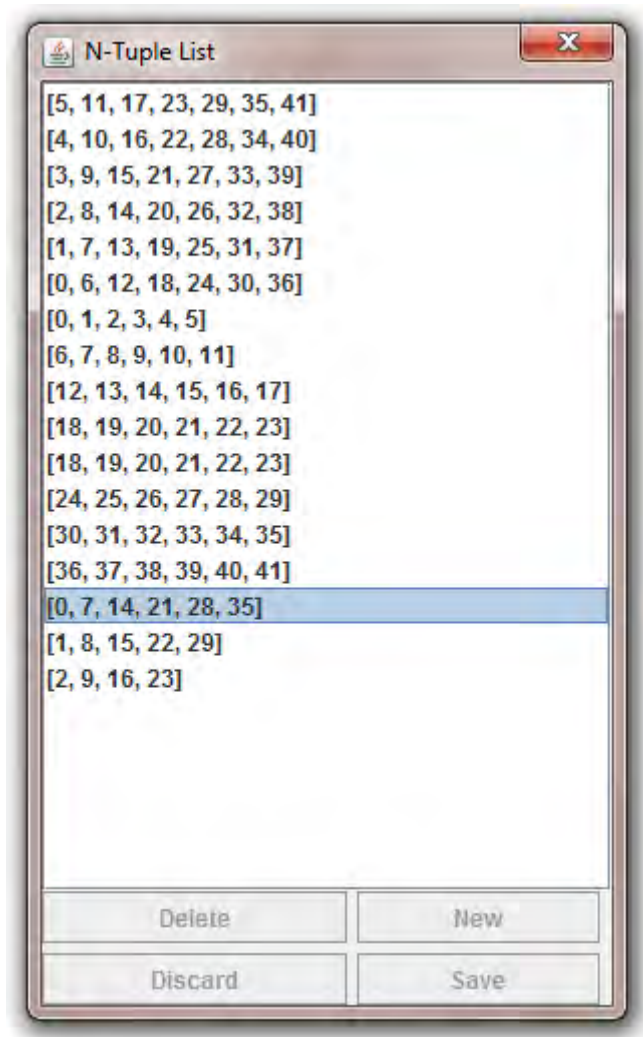
Figure 14 Simple switching between the single-NTuples

6.2 Mode: Eval Position

This mode allows the evaluation of all possible Connect Four positions. The user can set positions and check the result of the value-function. That's why all fields of the Connect Four board can be selected / deselected by the user. It can also be checked which LUT contributes which value for the current position. Note that the user has to ensure that these positions are legal. In this mode it is not possible to move inside the LUTs, because the current board determines the entry-Nr. for every single LUT. But the weights / scores for the selected LUT is still shown correctly in the main window. The fields with the white-marked corners in the Connect Four board show the sampling points of the current N-Tuple.

Always make sure, that the right LUT is inspected for a board (boards with an even number of pieces correspond to player X's LUTs).

Figure 15 Example for a position (winner: X->yellow)

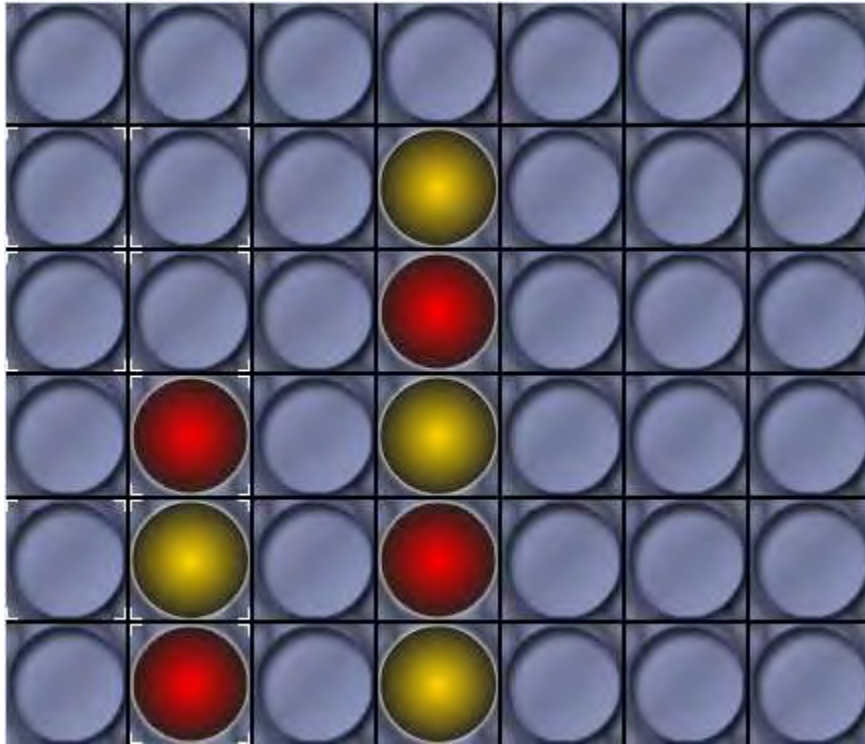The contributions of the single LUTs can be sighted in the other window. Note that the weights for the mirrored boards are listed too. The index within the LUT is listed for every N-Tuple.

Values for all N-Tuples (for current Board)

○ Player X   ○ Player O

Tuple-Nr.	Index	Score	Index (mir.)	Score (mir.)
X.0/69	41179:	-0,01642	41167:	+0,03905
X.1/69	612:	+0,07872	49764:	+0,08110
X.2/69	14720:	+0,17620	192:	-0,01767
X.3/69	51449:	-0,03842	35008:	+0,02412
X.4/69	48:	+0,09242	0:	+0,05024
X.5/69	896:	+0,01220	0:	+0,01144
X.6/69	56:	+0,02309	0:	-0,01075
X.7/69	20099:	-0,07159	19651:	-0,03316
X.8/69	0:	-0,02366	896:	-0,02350
X.9/69	32992:	-0,01097	32768:	+0,00351
X.10/69	17280:	-0,00078	16384:	+0,06230
X.11/69	12537:	+0,00114	14777:	-0,00678
X.12/69	224:	+0,02309	0:	-0,01075
X.13/69	6144:	-0,01250	6144:	-0,01250
X.14/69	1600:	-0,02032	1600:	-0,02032
X.15/69	205:	-0,03316	2445:	-0,04213
X.16/69	224:	-0,00184	12512:	-0,04190
X.17/69	2:	-0,00308	514:	+0,00210
X.18/69	769:	-0,01386	14849:	+0,11917
X.19/69	55552:	-0,09545	55552:	-0,09545
X.20/69	923:	+0,01054	783:	-0,00507
X.21/69	12:	+0,02768	3596:	+0,00378
X.22/69	475:	+0,08314	8655:	+0,04820
X.23/69	3584:	+0,04991	0:	+0,08077
X.24/69	0:	-0,00528	56:	+0,00411
X.25/69	128:	-0,04160	14464:	-0,11041

Values for all N-Tuples (for current Board)

○ Player X   ○ Player O

Tuple-Nr.	Index	Score	Index (mir.)	Score (mir.)
X.48/69	58368:	-0,00683	58368:	-0,00683
X.49/69	25648:	+0,02741	25600:	-0,01157
X.50/69	4:	-0,01799	2052:	-0,01742
X.51/69	771:	-0,01767	822:	-0,14018
X.52/69	48:	-0,01392	48:	-0,01392
X.53/69	0:	+0,03821	0:	+0,03821
X.54/69	100:	+0,06036	14436:	+0,01235
X.55/69	9:	-0,07006	2313:	-0,00135
X.56/69	13324:	+0,03893	13312:	+0,01080
X.57/69	0:	-0,01017	912:	-0,01420
X.58/69	1:	+0,14689	193:	+0,10517
X.59/69	272:	+0,00197	256:	+0,00851
X.60/69	53262:	+0,04094	53248:	+0,04439
X.61/69	1603:	+0,00652	34368:	-0,00751
X.62/69	14720:	+0,03447	192:	-0,00507
X.63/69	3328:	+0,03975	3328:	+0,03975
X.64/69	2435:	+0,01054	195:	-0,00507
X.65/69	1240:	+0,00415	1228:	-0,00520
X.66/69	18:	-0,02723	13328:	+0,01030
X.67/69	9:	+0,01718	3081:	-0,01782
X.68/69	33664:	-0,06100	32768:	-0,00377
X.69/69	51200:	+0,00982	52224:	-0,00507
-------	-------	-------	-------	-------
SUM		+0,59238		+0,21158
SCORE	+0,80396			
SCORE (Tanh)	+0,66624			

Figure 16 Bottom Part of the list for the above position
Figure 17 Upper part of the list for the above position

The received overall score of 0,8 seems to be quite appropriate for the above board in this case.

7 Developing Connect-4 Agents for the JAVA-Framework

If a new connect-4 agent is developed or an existing agent shall be plugged into this framework, a few points have to be considered in order to successfully use all the functionalities provided by the framework, such as agent-evaluation, agent-competitions and more:

- The interface "Agent.java" from the c4-package has to be implemented as a wrapper for the provided agent. This interface provides methods such as findBestMove or

getScore, which are typically needed to play matches against other agents or to evaluate the values of certain actions from different board-positions. For further information refer to the source of "Agent.java" and the corresponding java-doc.

- If the agent is used for Playing a Game and if the values of this agent are also shown in the Value-Panel, then it has to be ensured that the agent can handle multiple requests (finding the best move for a position and determining the values of all after-states) at the same time, since the game and the value-panel are running in different threads (although this does not have to be a problem for all agents). One possibility is to use semaphores, that only allow mutual exclusive access to the agent (this is also supported by the interface "Agent.java").

- After the source-code of a new agent is added to the framework, several Java-Files have to be adjusted currently:
  1. Add the name of the agent to the Array "String agentList[]" in "gui.C4Buttons.java"
  2. Add the initializations of the agent to the actionListeners of the buttons "bInitX", "bInitO", "bInitEval" in the Java-File "gui.Connect4Buttons.java"
  3. If the agent requires user-defined parameters, a Frame can be developed (extending the swing JFrame class) which allows the user to pass certain parameters to the agent. This JFrame then can be initialized in the method "openParams" in the file "gui.Connect4Buttons.java".

8 Pre-trained Agents and Predefined Parameter Sets

8.1 Pre-trained Agents

There are a few trained agents included in the distribution of C4GPF. They reside in the subdirectory CFour/agents/ and they can be accessed from the C4GPF GUI via the buttons Load Agent or the Import Agent Weights.

- **C4.TCL-EXP-lam0.5-et** (either .agt.zip or .txt.zip): This is (currently) our best agent, which is based on the TCL-EXP learning scheme and uses eligibility traces with λ=0.5. It is described in more detail in [Thil14].
- **C4.TCL-EXP-lam0.0.txt.zip**: This is an agent with TCL-EXP learning scheme without eligibility traces. It is described in more detail in [Bagheri14].
- **C4.KoopIDBD-lam0.0.txt.zip**: This is an agent with Koop's nonlinear IDBD learning scheme [Koop08]. It is described in more detail in [Bagheri14].

All agents were trained for 2 million training games and the resulting weights stored on file.

8.2 Predefined Parameter Sets

There are a few predefined parameter sets included in the distribution of C4GPF. They reside in the subdirectory CFour/tdPar/ and they can be accessed from the C4GPF GUI via the button "Open" in the TD-Parameters Window.

In addition there is the button "Load Best" in the TD-Parameters Window: With this button for every Learning Rate Adaptation algorithm the (so far) best known parameter set (based on the criterion "time-to-learn") is selected.

In any case, after the parameter set is loaded, the agent has to be initialized and trained. It is recommended to use 1 million or 2 million training games.

9   Literature

[Bagheri14] Bagheri, S.; Thill, M.; Koch, P.; Konen, W.: Online Adaptable Learning Rates for the Game Connect-4. In: Transactions on Computational Intelligence and AI in Games, (accepted for publication 11/2014), preprint available from Cologne Open Science Server: http://nbn-resolving.de/urn:nbn:de:hbz:832-cos-704, (2014)

[BagThill14] Bagheri, S.; Thill, M.: Temporal Coherence in TD-Learning for Strategic Board Games, Case Study Report, Cologne University of Applied Sciences, http://www.gm.fh-koeln.de/~konen/research/PaperPDF/CS-BagheriThill-2014.pdf (2014)

[Beal99] Beal, D.F., Smith, M.C.: Temporal coherence and prediction decay in TD learning. In: Dean, T. (ed.) IJCAI. pp. 564 – 569. Morgan Kaufmann (1999)

[Beal00] Beal, D.F., Smith, M.C.: Temporal difference learning for heuristic search and game playing. Information Sciences 122(1), 3 – 21 (2000)

[Dabney12] Dabney, W., Barto, A.G.: Adaptive step-size for online temporal difference learning. In: 26th AAAI Conference on Artificial Intelligence (2012)

[Konen14] Konen, W., Koch, P.: Adaptation in Nonlinear Learning Models for Nonstationary Tasks. In: PPSN'2014: 13th International Conference on Parallel Problem Solving From Nature, Ljubljana, Springer, Heidelberg, (2014)

[Koop08] Koop, A.: Investigating Experience: Temporal Coherence and Empirical Knowledge Representation. Canadian theses, Univ. of Alberta (Canada) (2008)

[Mahmood10] Mahmood, A.: Automatic step-size adaptation in incremental supervised learning. Ph.D. thesis, University of Alberta (2010)

[Mahmood12] Mahmood, A., Sutton, R., Degris, T., Pilarski, P.: Tuning-free step-size adaptation. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. pp. 2121 – 2124 (2012)

[Riedmiller93] Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: IEEE Int. Conf. on Neural Networks. pp. 586{591 (1993)

[Schraudolph99] Schraudolph, N.N.: Online learning with adaptive local step sizes. In: Neural Nets WIRN Vietri-99, pp. 151 – 156. Springer (1999)

[Sutton92] Sutton, R.S.: Adapting bias by gradient descent: An incremental version of delta-bar-delta. In: Swartout, W.R. (ed.) AAAI. pp. 171 – 176. AAAI Press / The MIT Press (1992)

[Sutton92a] Sutton, R.S.: Gain adaptation beats least squares. In: Proc. Yale Workshop on Adaptive and Learning Systems. pp. 161{166 (1992)

[Sutton07] Sutton, R.S., Koop, A., Silver, D.: On the role of tracking in stationary environments.

In: 24th Int. Conf. on Machine Learning. pp. 871 – 878. ACM (2007)

[Thill2012] Thill, M.; Koch, P.; Konen, W.: Reinforcement learning with n-tuples on the game Connect-4 In: Coello Coello, C.; Cutello, V; others, (Ed.): PPSN'2012: 12th International Conference on Parallel Problem Solving From Nature, Taormina, pp. 184–194, Springer, Heidelberg
http://maanvs03.gm.fh-koeln.de/webpub/CIOPReports.d/Thi12.d/Thil12.pdf (2012)

[Thill2012a] Thill, M.: Reinforcement Learning mit N-Tupel-Systemen für Vier Gewinnt (in German), Bachelor Thesis, Cologne University of Applied Sciences,
http://www.gm.fh-koeln.de/ciopwebpub/Thi12.d/Thil12.pdf (2012)

[Thill2014] Thill, M.; Bagheri, S.; Koch, P.; Konen, W.: Temporal Difference Learning with Eligibility Traces for the Game Connect-4 In: Preuss, M.; Rudolph, G. (Ed.): CIG'2014, International Conference on Computational Intelligence in Games, Dortmund,
http://www.gm.fh-koeln.de/~konen/Publikationen/ThillCIG2014.pdf (2014)