## Agent Type

* **T****DS**: TD Learning agent according to [Sutton&Bonde 1993] with a linear net or backprop net as approximator for the value function.
* **Mi****nimax**: game-tree search agent using Minimax strategy (Alpha-Beta-search). Realizes perfect play for TicTacToe
* **Ra****ndom**: random playing agent
* **CMA****-ES**: TD Learning agent using neuroevolution with CMA-ES as approximator for the value function
* **MC****TS**: Monte Carlo Tree Search
* **Hu****man**: Human player

## Play

Play a game with the agents currently selected as “X” or “O” in the [Agent Type](#AgentType) combo boxes. If the currently selected agent (“X” or “O”) is not yet trained or the trained agent differs from the one in the combo boxes, an error message is displayed. If one of the selected agents is [Human](#Human), the user has to fill in the appropriate moves.

The starting player is always “X”.

## InspectV

Inspect the value function V(**s’**) of the X-Player. The X-Player is

(a) before using the [Train X button](#Train_X): the MinimaxPlayer (as set in the constructor of class XArenaFuncs for m\_playAgent),   
(b) after using the [Train X button](#Train_X): the last trained X-Player.

Different after states **s** can be set by the user in the GameBoard board buttons (for a 2-player game: clicking a button moves it through the circle [ ] -> X -> O -> [ ] -> …). The values +V(**s’**) are displayed in the 3x3 yellow label field for each allowed successor **s’** of **s**.

## Params X, Params O

Display the multi-tabbed Params window and set parameter in any of the tabs ([TD params](#TD_params) and so on, see below). The parameter are fetched from this multi-tabbed window when one of the train buttons, Train X or Train O, is pressed

## Train X

Fetch the parameter settings from the multi-tabbed Params window. Set the X agent according to the X combo box and train it for “Train games” games.

During training, if [StopTest](#StopTest)>0 and [StopEval](#StopEval)>0, an Evaluator is called every StopTest training games. If the Evaluator signals “Training goal reached” (i.e. sufficient good play for a sufficient long period, see [Other params](#Other_params) for more details), the training is stopped prematurely.

After training, the trained player is evaluated (Evaluator.eval()). This is for example in the case TicTacToe:

“Success against random” = average success rate when playing 100 games against RandomPlayer, both as X and O (optimum: 1.0),  
“Success agains minimax” = success rate when playing an X- and an O-game against MinimaxPlayer (optimum: 0.0, i.e. always tie).

Note that the success rate becomes negative, when the other player predominantly wins.

## Train O

Same as [Train X](#Train_X), but for the O-player.

## MultiTrain

Same as [Train X](#Train_X), but perform “Agents trained” training runs and report the average success.

## Competition menu

This menu is only relevant for 2-player games.

### Single Compete

Make a competition “X vs O” consisting of “Games/Comp” games and report results.

### Swap Compete

Swap the roles of X and O, i. e. make a competition “O vs X” consisting of “Games/Comp” games and report results.

### Multi-Competition

Perform “Competitions” competitions “X vs O” and report results. The agents (if trainable) are trained anew before each competition.

## TD params

Parameter for Temporal Difference Learning (for [TDS](#TDS) player, [TDS-NTuple](#TDS_NTuple) agent, [CMA-ES](#CMAES) player and [ValIt](#ValIt) player):

* **Alpha init**: initial learning rate
* **Alpha final**: final learning rate
* **Epsilon init**: initial random move rate (clipped to allowed range [0, 1])
* **Epsilon final**: final random move rate (clipped to allowed range [0, 1])
* **Lambda**: eligibility trace parameter (currently only relevant for Agent Type [TDS](#TDS))
* **Gam****ma**: discount factor in range [0, 1]
* **Network type**:[linear] the output activation is either a linear function of the (generalized) input features or a backpropagation network with one hidden layer of size 15.
* **Output sigmoid**:[without]should the output unit be with a sigmoid? If“with”, then the Fermi function  
  is used as sigmoid.

If the value function is approximated by a neural network, the effective learning rate for the input-to-hidden weights is Alpha divided by the input-fan-in (size of input layer) and the learning rate for the hidden-to-output weights is Alpha divided by the hidden-fan-in (size of the hidden layer).

### Feature sets

Feature sets for [TDS](#TDS) player, [CMA-ES](#CMAES) player and [ValIt](#ValIt) player

* **0**: singlets/doublets/triplets for “self” and “opponent”
* **1**: singlets/doublets/triplets for “X” and “O”
* **2**: singlets/doublets + diversity + crosspoints for “X” and “O”
* **3**: same as **2** + the 9 “raw” board positions
* **4**: same as **2** + occupation midpoint, occupation corner
* **5**: same as **2** + …
* **9**: the 9 “raw” board positions

## MCTS params

Parameter for [MCTS](#MCTS) agent (class MCTSPlayer):

* **Iterations**: [1000] how many rollouts are performed
* **K[UCT]**: [sqrt(2)] balances exploitation and exploration in the UCT formula
* **Tree Depth**: [10] the maximum MCTS tree depth
* **Rollout Depth**: [10] the maximum rollout depth (how many plys)

## Other params

During or after training an agent, this agent can be evaluated by an evaluator (see below). If such an evaluator signals “success” (for a long enough training period of StopEval training games), then training might be stopped prematurely.

Settings:

* **NumEval**: [100] after every NumEval training games the performance of the trained agent is evaluated (success against [Minimax](#Minimax), success against [Random](#Random)) and the success against [Minimax](#Minimax) is plotted in a JFreeChart window. Choose higher values for NumEval to speed up training.
* **Sto****pTest**: [0] after every StopTest training games an Evaluator is called to see if we can stop training prematurely. If 0, this Evaluator is never called and so training is never stopped prematurely.
* **Sto****pEval**: [0] number of games an Evaluator has to signal “success” before training is stopped prematurely. If 0, training is never stopped prematurely.
* **Minimax Depth**: [10] the maximum tree depth (recursion depth) of the Minimax agent.

The available evaluators for TicTacToe are:

* Evaluator1: measure the success on a set of predefined states (deprecated)
* EvaluatorTTT with mode=1: make **Stop comp2** competitions against RandomPlayer (optimal success rate: 1.0) and one competition against MinimaxPlayer, both as X and O (optimum: 0.0). Return success against Minimax.
* EvaluatorTTT with mode=2: make one competition against MinimaxPlayer from 10 different start states, both as X and O (optimum: 0.0). Return average success

Currently, EvaluatorTTT with mode=2 is chosen in source code (class XArenaFuncs, method train). An evaluation of EvaluatorTTT is termed a “success”, if its return value is above the threshold m\_thresh (currently m\_thresh = -0.15 in source code)

## Help

Toggle the display of this help text in a HTML window.