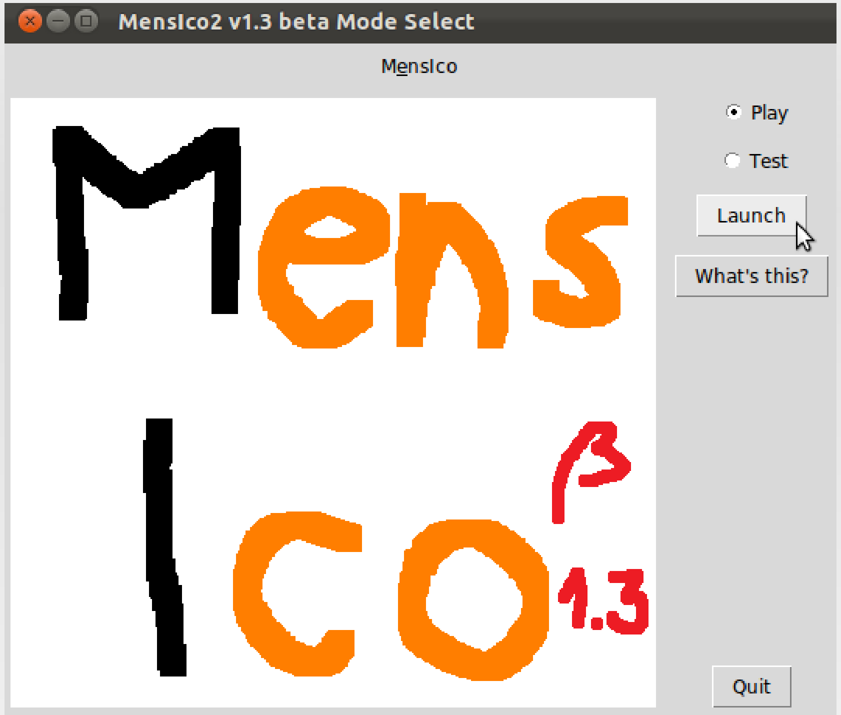
Machine Learning methods to enhance artificial opponents



9th Joint Conference on **Ma**thematics and **C**omputer **S**cience

February 9-12, 2012 Siófok, Hungary

András Fülöp
PhD student
University of Debrecen
fulibacsi@gmail.com

Overview

- Motivation
- The problem
- Our solutions
- Comparison of methods
- Further work

Motivation

- Who is a good opponent?
  - As strong/agile/smart as we are.
- Artificial opponents?
  - Too easy
  - Too hard
  - Just good - only in some cases
- Why? Because they are...
  - Heuristics driven
  - Scripted
  - Search based

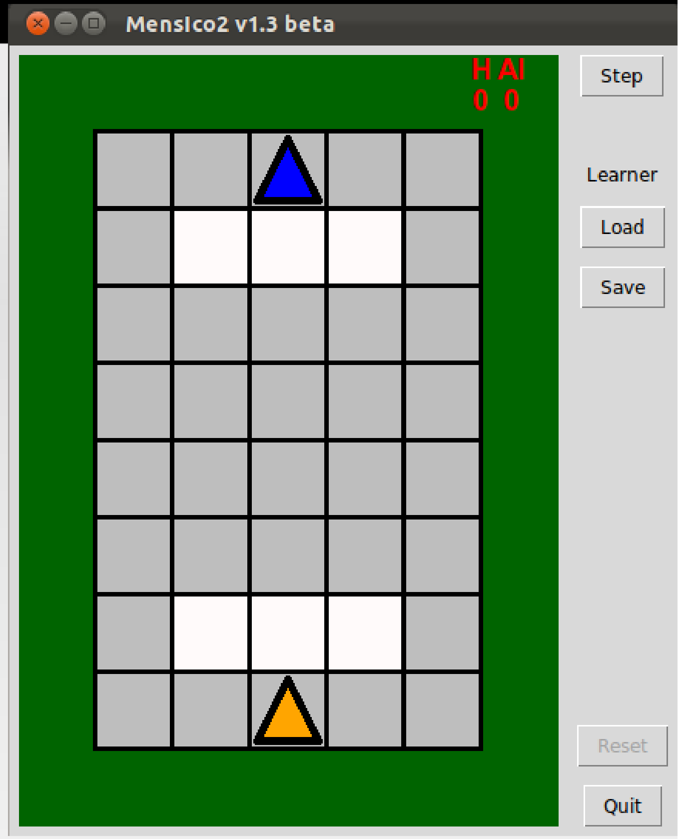
Motivation

- In some cases:
  - Stochastic
- Problem:
  - Perfect strategy is too good
  - How should we modify it? → learn it!

The problem

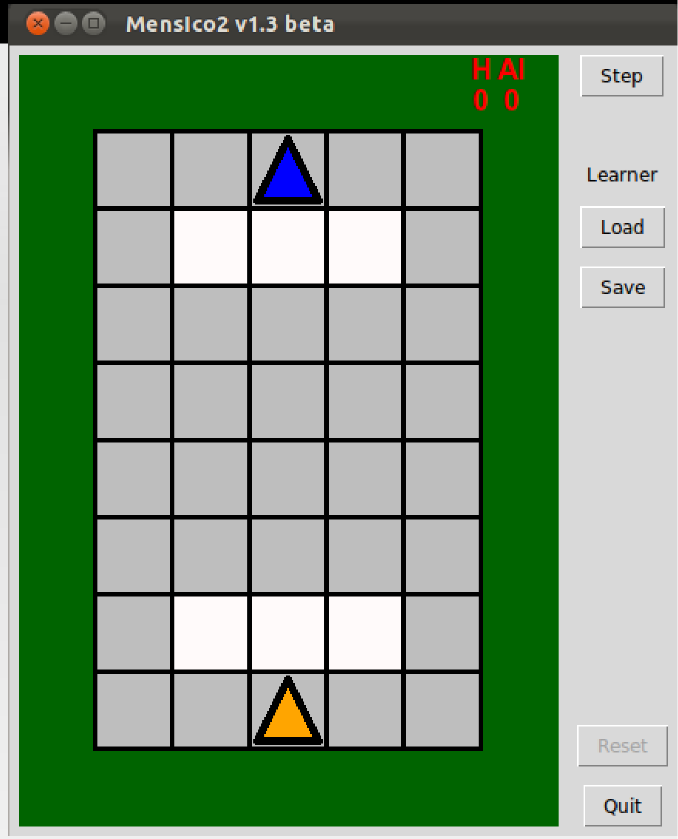
Test case:

- MensIco *(mens – mind, Ico – to hit)*
  - Discrete, two player, non-cooperative, symmetric, zero-sum, simultaneous, imperfect information game
  - Goal: reach the other side of the board
  - How: select where you want to step, guess where does the opponent will step
  - What if someone's guess was right? → opponent can't step



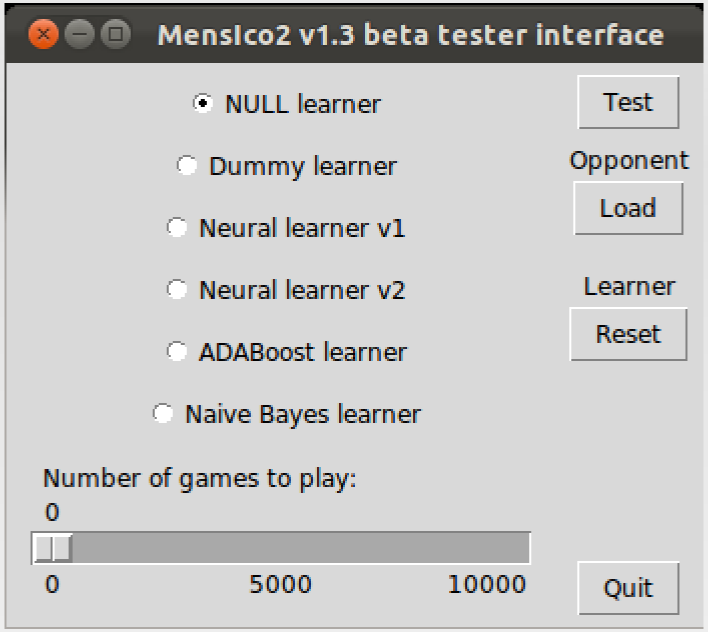
The problem

- Ideal strategy?
  - Every step has the same probability (uniform distribution)
- Is it fun playing against it?
  - It's the same as gambling, without the risk of losing money
- Solution?
  - Start from uniform distribution, but learn from the user's decisions



Our solutions

- Dummy learner
- Neural network based learner
- ADABoost based learner
- Naive Bayes method based learner



Our solution – Model

- Two matrix:
  - Step
  - Prediction
- Every tile on the board is equivalent to a matrix element in both of the matrices
- The values in the matrices are probabilities
- Decide based on these probabilities
- Modify these probabilities in the learning process
- Important constraint: $\sum_j w_t^{i,j} = 1.0$

Our solution – Model

- Notation

$w -$ weight / probability

$x -$ node

$i -$ line number

$j -$ index inside a line

$t -$ round

$\gamma -$ indicator of connection

$$\gamma(a, b) = \begin{cases} 1, \text{ if } a,\ b \text{ is connected} \\ 0, \text{ otherwise} \end{cases}$$

$\alpha -$ learning constant

$\varepsilon -$ error

Our solution – Dummy Learner

- Inspired by the common sense:
  - If something „was good”, increase probability
  - If something „went wrong”, decrease probability
- Formal:
  - Step:

$$w_{t+1}^{i}=\begin{cases} w_t^i * 1.1, & \text{if } i_{t-1} != i_t \\ w_t^i * 0.9, & \text{if } i_{t-1} = i_t \end{cases}$$

  - Prediction:

$$w_{t+1}^{i}=\begin{cases} w_t^i * 1.1, & \text{if } i_{t-1}^{opponent} = i_t^{opponent} \\ w_t^i * 0.9, & \text{if } i_{t-1}^{opponent} != i_t^{opponent} \end{cases}$$

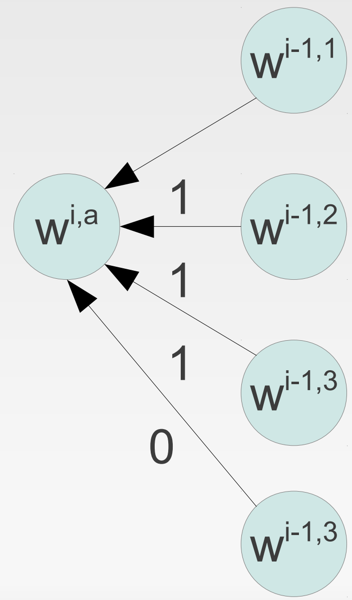
Our solution – Neural network

- Based on the neural network's backpropagation algorithm.
  - Consider every tile on the board to be a NN node.
  - With the following weight modification rule:

$$w_{t+1}^{i,a} = w_t^{i,a} + \alpha * \varepsilon * \sum_j \gamma\left(x_t^{i-1,j}, x_t^{i,a}\right) * w_t^{i-1,j}$$

  - where

$$\varepsilon = \begin{cases} -1, \text{ if } i_{t-1} != i_t \vee i_{t-1}^{opponent} != i_t^{opponent} \\ 1, \text{ if } i_{t-1} != i_t \vee i_{t-1}^{opponent} != i_t^{opponent}\ t \end{cases}$$



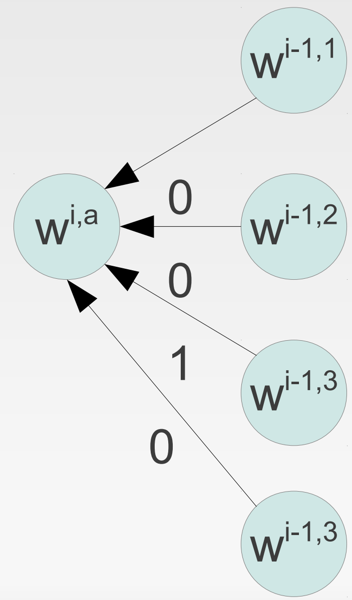
Our solution – Neural network v2

- Based on the neural network's backpropagation algorithm.
  - Consider every tile on the board to be a NN node.
  - With the following weight modification rule:

$$w_{t+1}^{i,a} = w_t^{i,a} + \alpha * \varepsilon * w_t^{i-1,previous}$$

  - where

$$\varepsilon = \begin{cases} -1, \text{ if } i_{t-1} != i_t \vee i_{t-1}^{opponent} != i_t^{opponent} \\ 1, \text{ if } i_{t-1} != i_t \vee i_{t-1}^{opponent} != i_t^{opponent} \; t \end{cases}$$



Our Solution – ADABoost

- Based on ADABoost weighting
- ADABoost is a Data Mining method that increase the efficiency of basic classifiers
- Learning rule:

$$w_{t+1}^{i,j} = w_t^{i,j} * \exp\left\{\frac{1}{2} * \ln\left(\frac{1-\varepsilon}{\varepsilon}\right)\right\}$$

$$\varepsilon = \begin{cases} \sum_j w_t^{i,j}, \text{ if } i_{t-1} != i_t \\ w_t^{i,\,guess} + w_t^{i,\,opponent\ step}, \text{ if } i_{t-1}^{opponent} = i_t^{opponent} \\ 0, \text{ otherwise} \end{cases}$$

Our solution – Naive Bayes

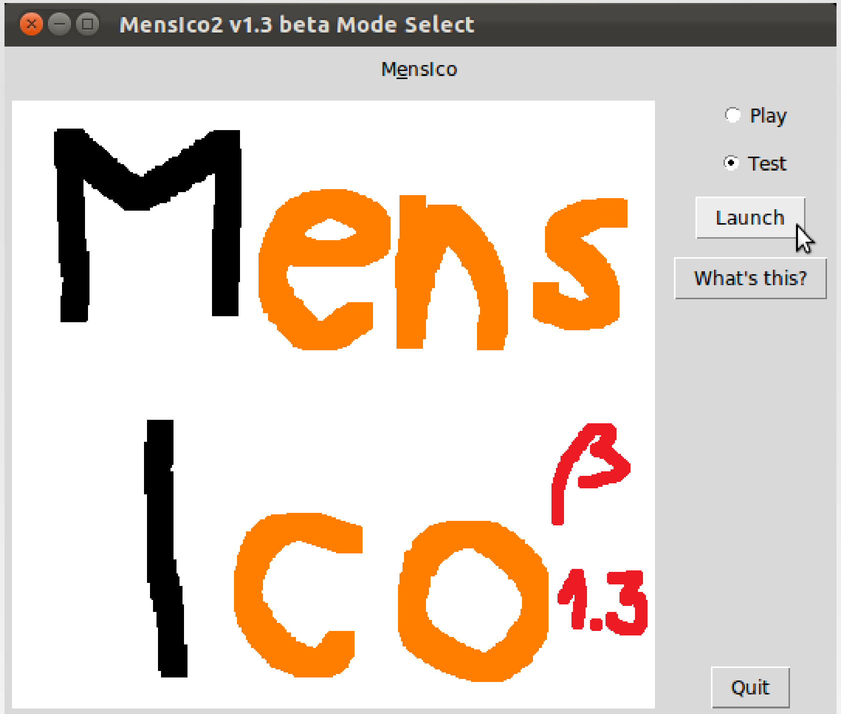
- Based on Naive Bayes method's μ approximation
- Estimate real distribution by sampling opponent's moves
- Start from uniform distribution, and modify plobabilities as following:

$$w_t^{i,j} = \frac{N^{i,j}}{N^i} \quad \text{where } N^{i,j} \text{ - number of visits in x\^{}\{i,j\}}$$

$$N^i \text{ - number of visits in line } i$$

$$w_{t+1}^{i,j} = \begin{cases} \dfrac{N^{i,j}+1}{N^i+1}, & \text{if } i_{t-1} != i_t \vee i_{t-1}^{opponent} = i_t^{opponent} \\ \dfrac{N^{i,j}}{N^i+1}, & \text{if } i_{t-1} = i_t \vee i_{t-1}^{opponent} != i_t^{opponent} \end{cases}$$

Comparison of methods



Comparison of methods – methodology

- To compare results we need metrics:
  - Win ratio

$$WinRatio = \frac{Win_{p1}}{Win_{p1} + Win_{p2}}$$

where

$$Win_p - \text{number of games won by player } p$$

Comparison of methods – methodology

- To compare results we need metrics:
  - Win ratio
  - Divergence between actual and ideal distribution
    - Kullback-Leibler divergence

$$\operatorname{div}_{KL}(P,Q)=\sum_j p_j * \exp\left(\frac{p_j}{q_j}\right)$$

    - Chi-square divergence

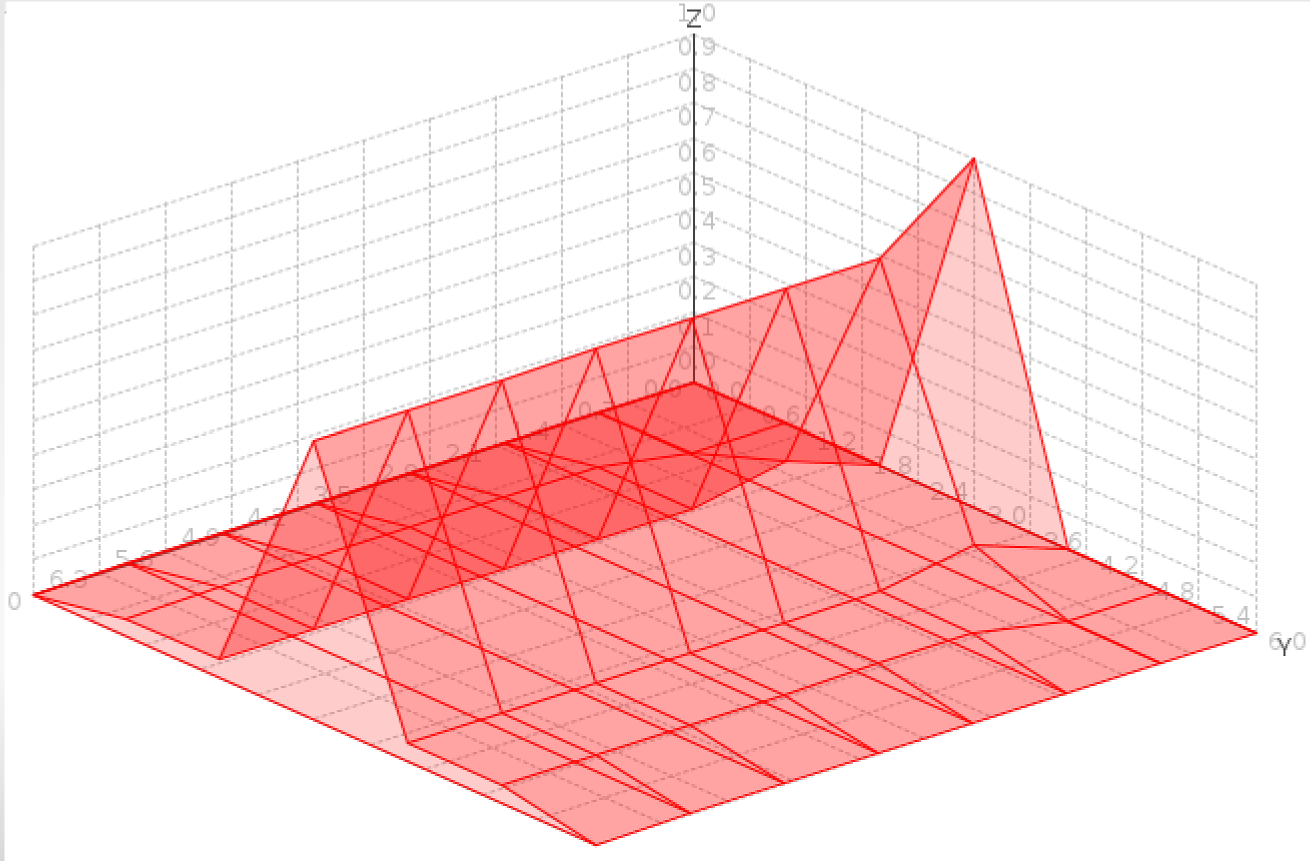
$$\operatorname{div}_{Chi^2}(P,Q)=\sum_j \left(\frac{(p_j-q_j)^2}{q_j}\right)$$

Comparison of methods – methodology

- To compare results we need metrics:
  - Win ratio
  - Divergence between actual and ideal distribution
    - Average of two parts:
      - $\mathrm{div}\left(P^{step}, 1-Q^{pred}\right)$
      - $\mathrm{div}\left(P^{pred}, Q^{step}\right)$

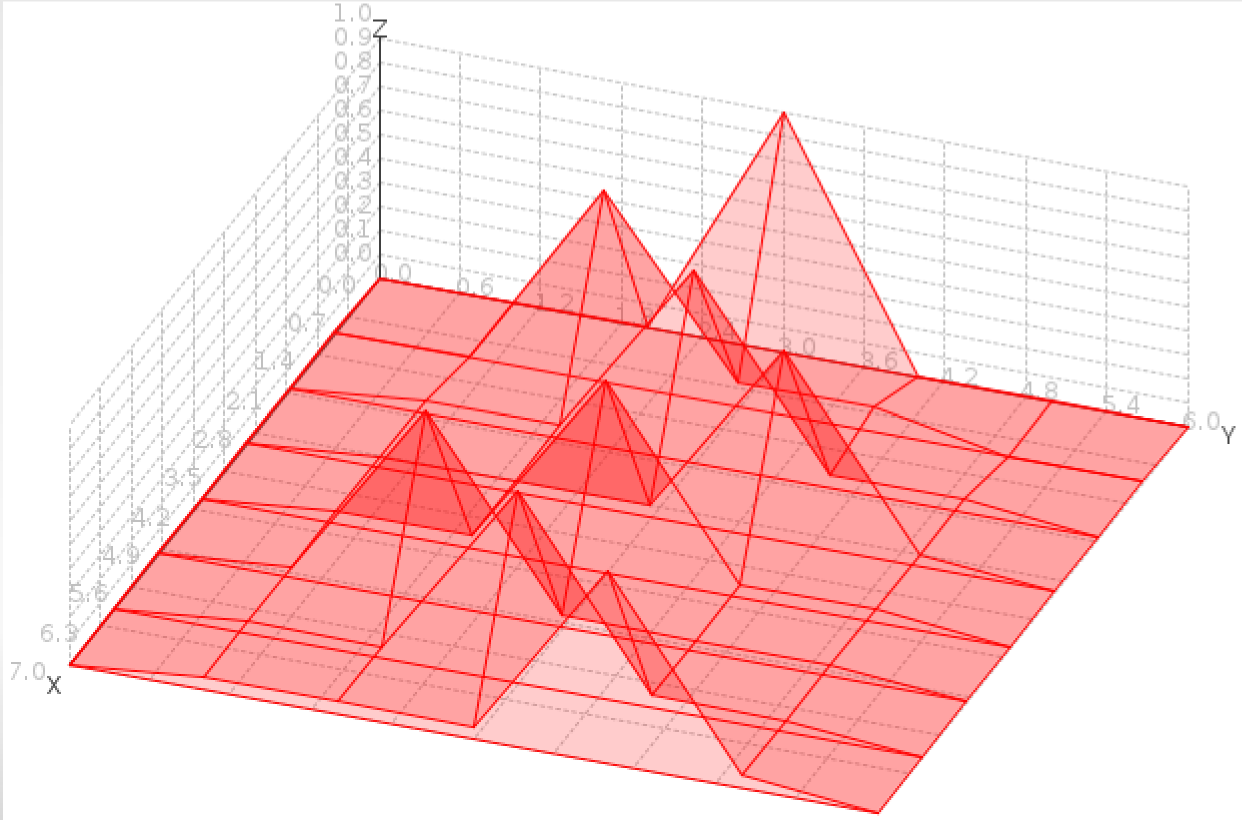
Comparison of methods – test strategies

- To compare results we also need test cases:
  - Pre-set strategies
    - „Straightforward”



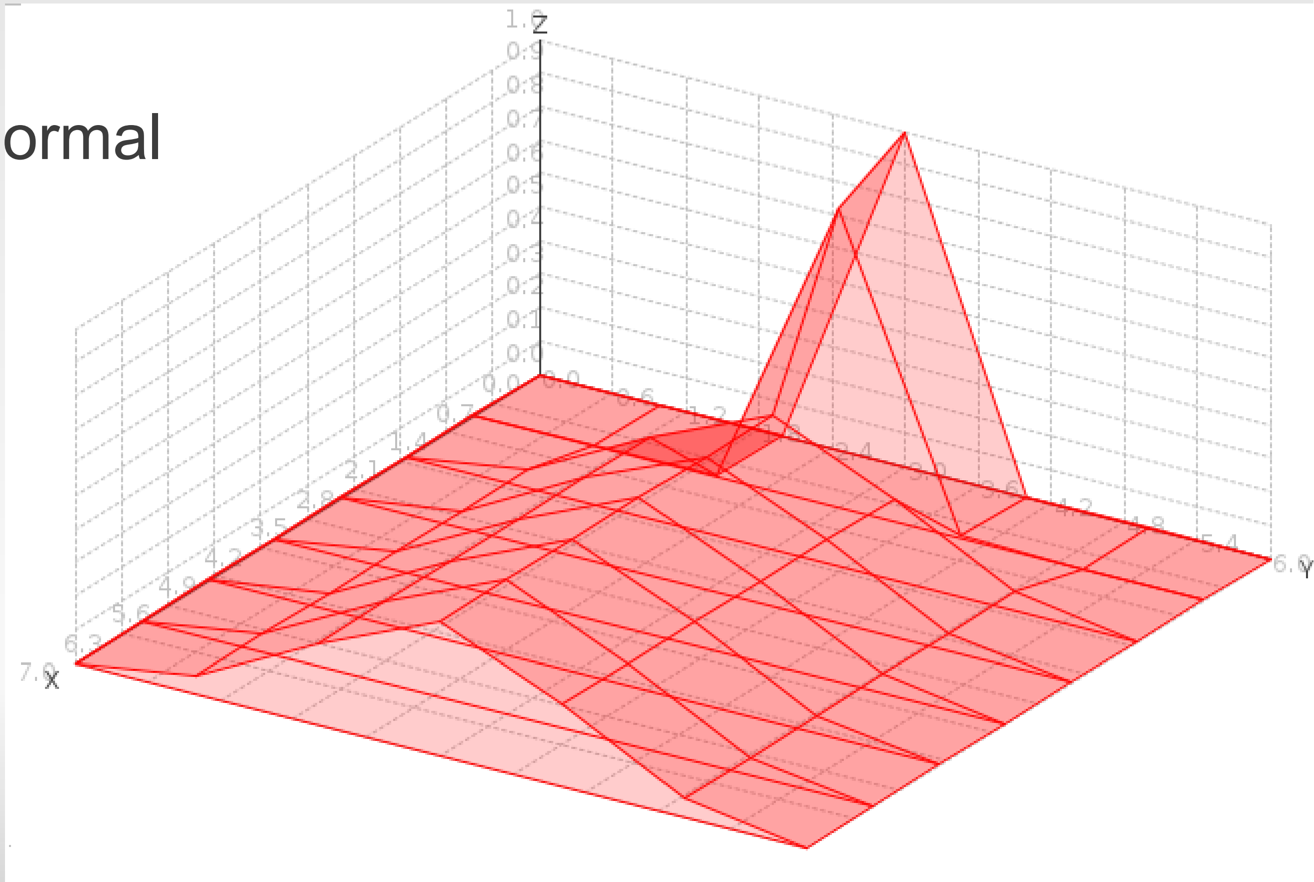
Comparison of methods – test strategies

- To compare results we also need test cases:
  - Pre-set strategies
    - „Straightforward”
    - „ZigZag”



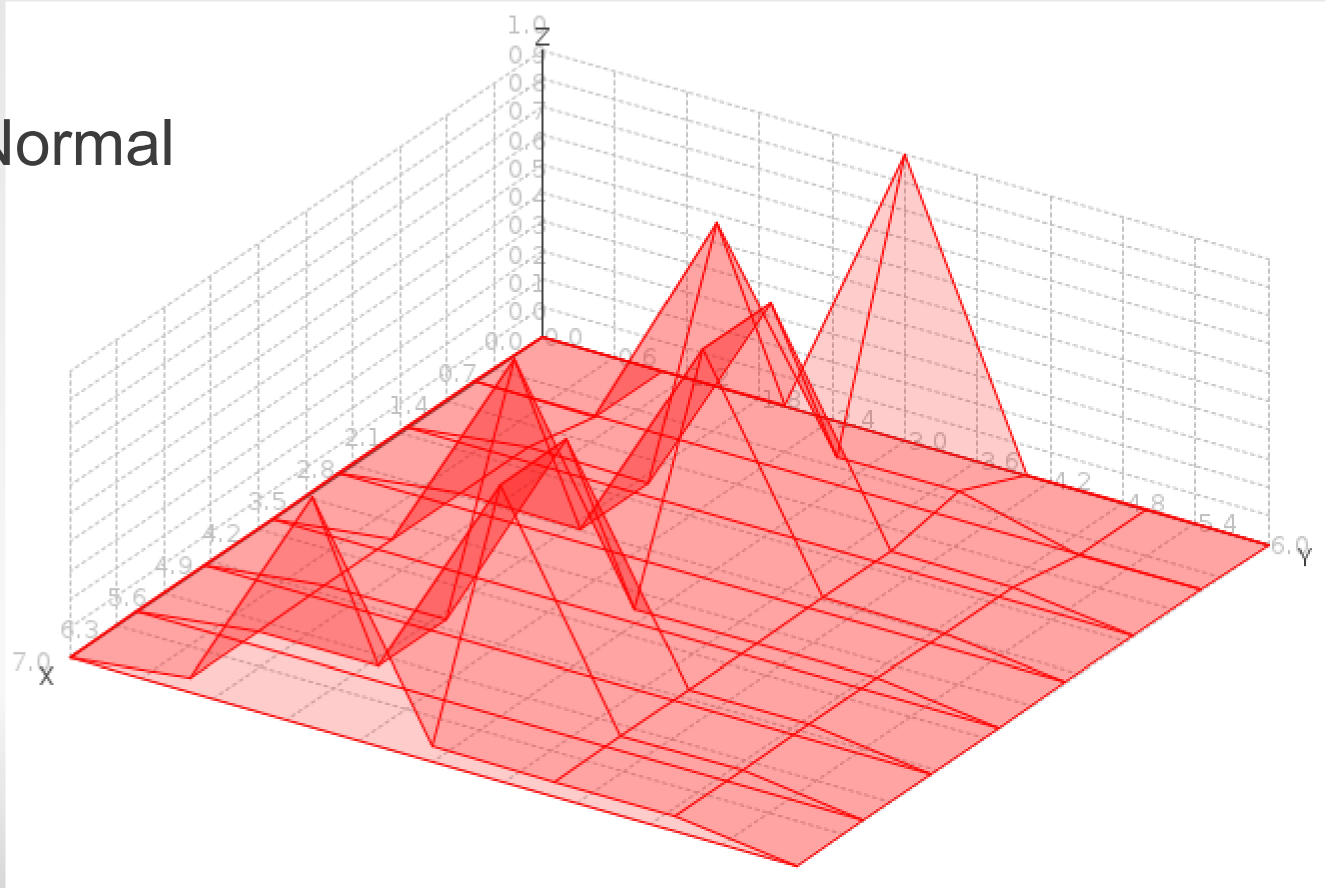
Comparison of methods – test strategies

- To compare results we also need test cases:
  - Pre-set strategies
    - „Straightforward”
    - „ZigZag”
    - Standard Normal



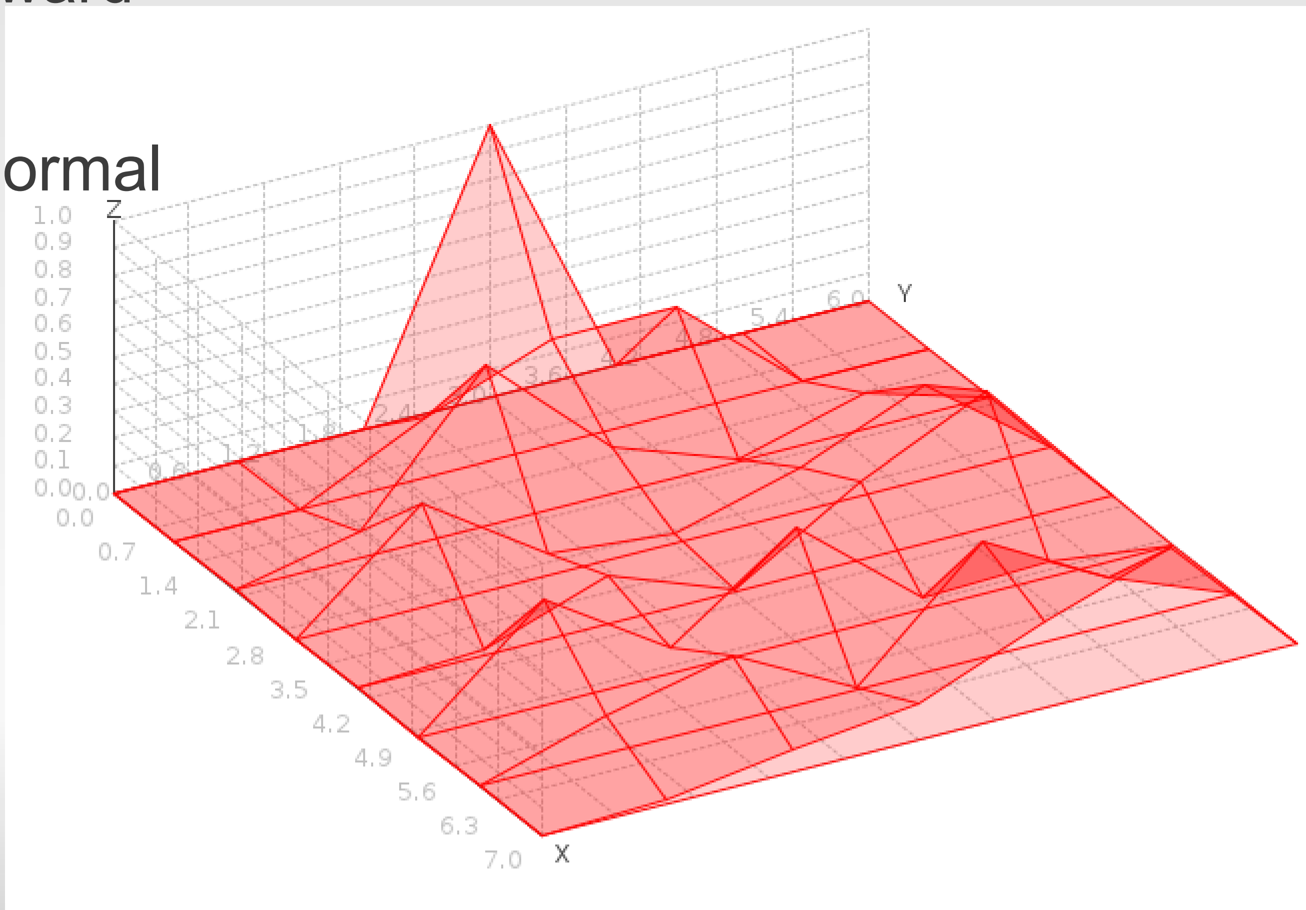
Comparison of methods – test strategies

- To compare results we also need test cases:
  - Pre-set strategies
    - „Straightforward”
    - „ZigZag”
    - Standard Normal
    - „Csardas”



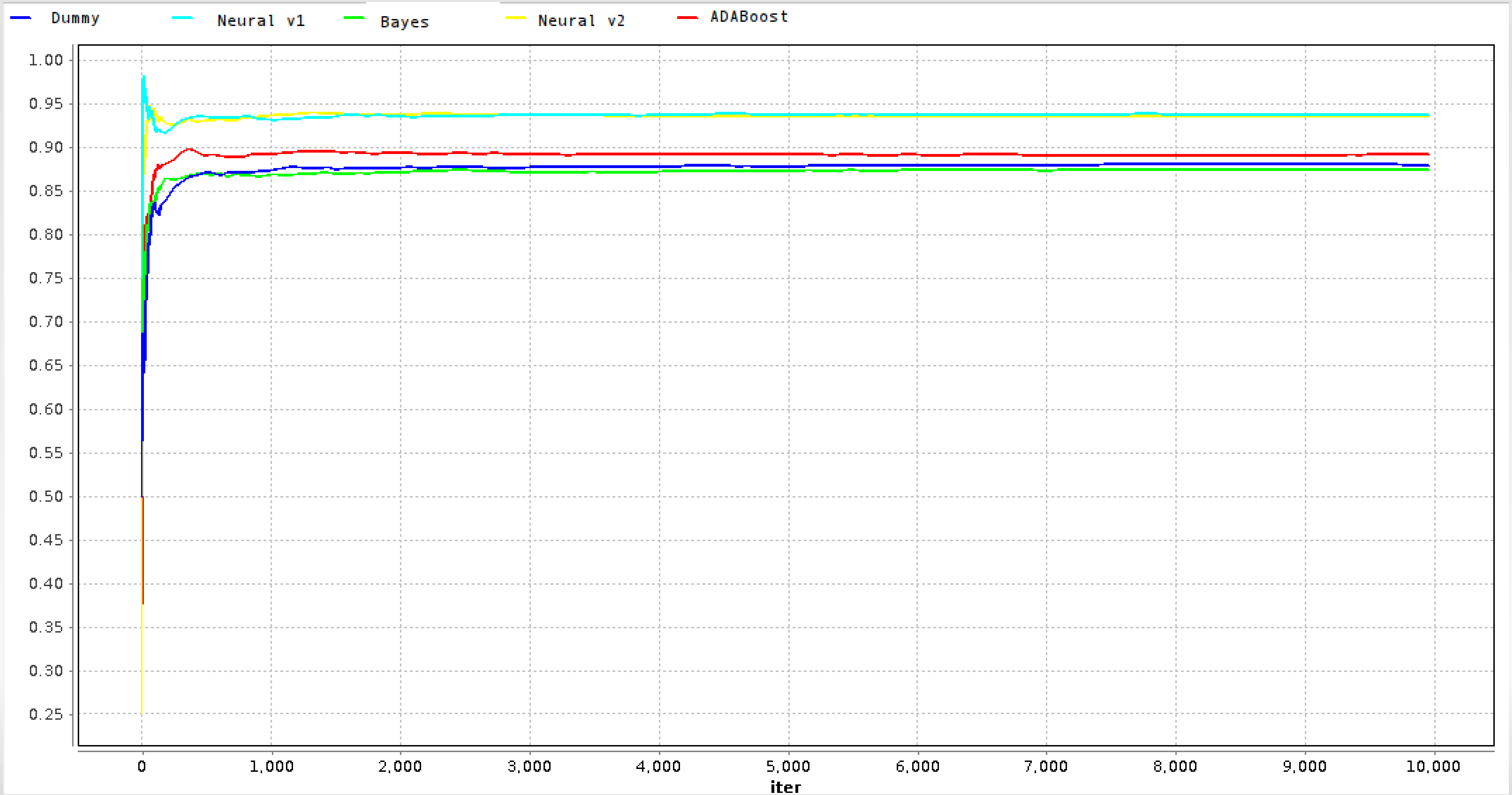
Comparison of methods – test strategies

- To compare results we also need test cases:
  - Pre-set strategies
    - „Straightforward"
    - „ZigZag"
    - Standard Normal
    - „Csardas"
  - Human players



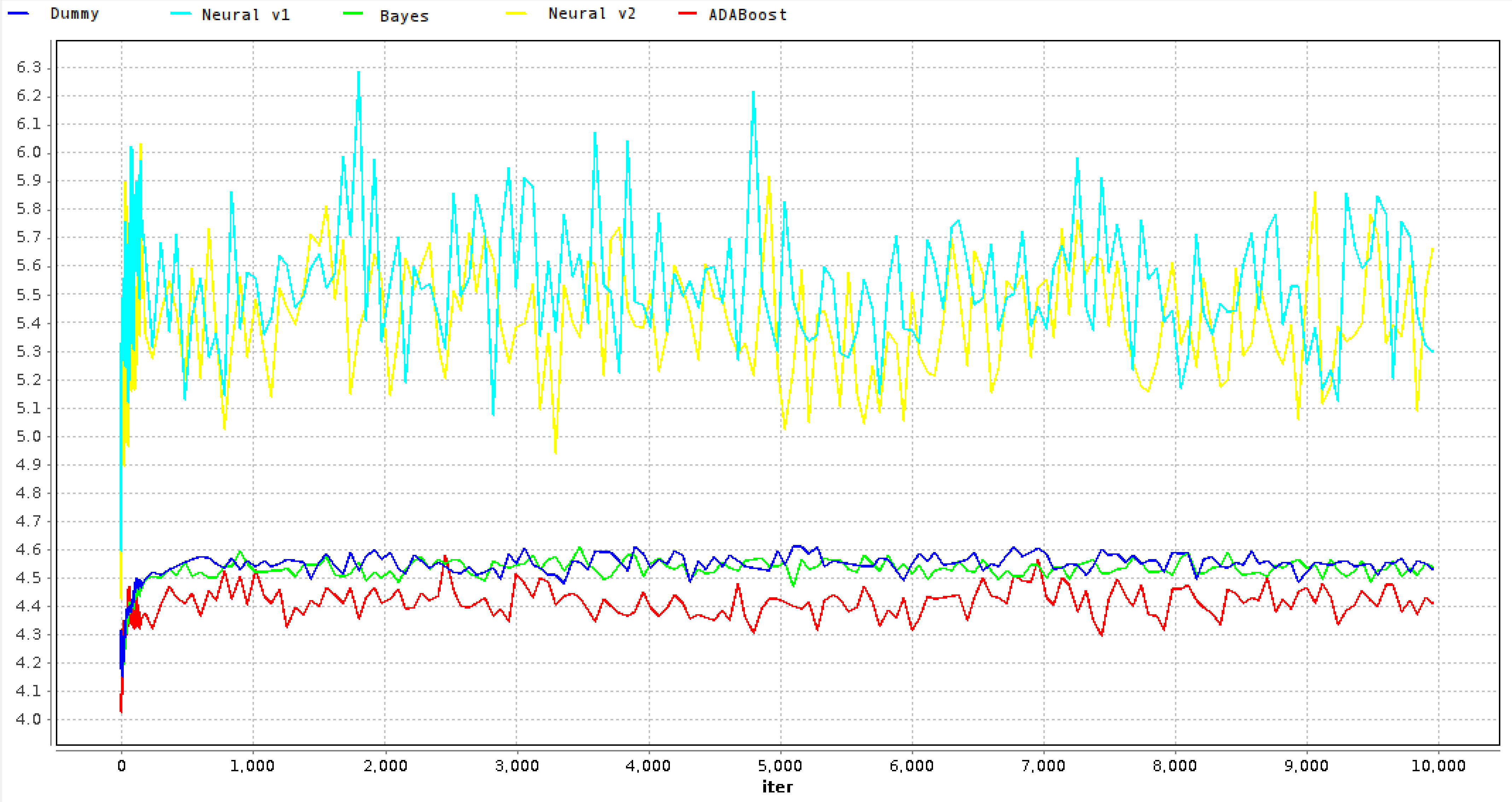
Comparison of methods – results

- Win ratio



Comparison of methods – results

- Error



Comparison of methods – conclusion

- Based on the previous results, we couldn't announce an obvious best choice
  - Based on win ratio, the NN methods are considerably better
  - Based on error rate, the ADABoost seems to be a better choice

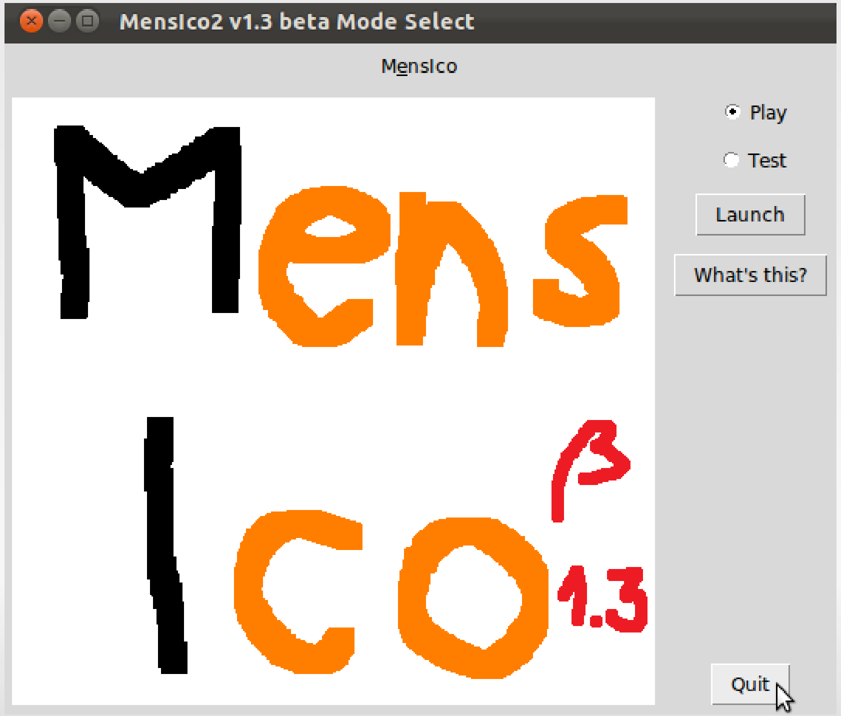
Further work

- After the closed alpha testing, we are planning to publish the beta version to play:
  - it online (in development)
  - as downloadable offline version (in development)
  - on Android operating system (future plan)
- Gether data and feedback to
  - further improve the software
  - research the distribution from which the individuals generate their random values
- Implementation of other learner methods.

Thank you for your kind attention!

If you have any remark, suggestion or question, please do not hesitate to ask it!

Also you are more than welcome to try out the actual version of the game!



András Fülöp
PhD student
University of Debrecen
fulibacsi@gmail.com