



# Machine Learning

## Quantum tic-tac-toe

- › Department of Artificial Intelligence
- › Eva van Viegen
- Peter Smit
- Coen Jonker
- Marc Volger



# Overview

- › Introduction
  - Problem description: Quantum-Tic-Tac-Toe
- › Method
  - Agents
  - Feedforward agent learning
- › Implementation
- › Experimental set-up
- › Preliminary Results
- › Discussion



# Introduction – Problem description

## › Quantum tic-tac-toe

$X_1 X_3$	$X_5 O_6$	$O_6$
1	2	3
$O_4 X_5$	$X_1 O_2$	
4	5	6
	$O_4$	$O_2 X_3$
7	8	9



# Introduction – Problem description

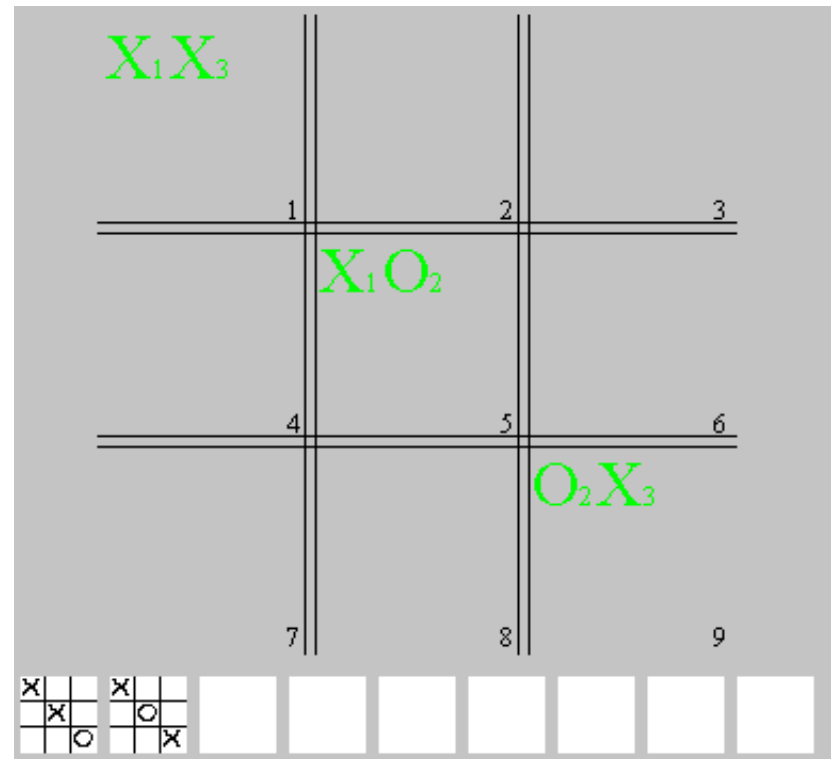
- › Quantum tic-tac-toe
- › Quantum move

$X_1$		
1	2	3
	$X_1 O_2$	
4	5	6
		$O_2$
7	8	9



# Introduction – Problem description

- › Quantum tic-tac-toe
- › Quantum move
- › Cycle detection





# Introduction – Problem description

- › Quantum tic-tac-toe
- › Quantum move
- › Cycle detection
- › Collapse move

$X_1 X_3$		
1	2	3
	$X_1 O_2$	
4	5	6
		$O_2 X_3$
7	8	9



# Introduction – Problem description

- › Quantum tic-tac-toe
- › Quantum move
- › Cycle detection
- › Collapse move
- › End game

$X_1 X_3$	$O_4 X_5$	$X_7$	1	2	3
$O_4 O_8$	$X_1 O_2$	$X_5 O_6$	4	5	6
$X_7$	$O_6 O_8$	$O_2 X_3$	7	8	9



## Introduction – Problem description

- › Quantum tic-tac-toe
- › Quantum move
- › Cycle detection
- › Collapse move
- › End game
  
- › Notice that X has an a priori chance of  $\pm 60\%$  of winning

$X_1 X_3$	$O_4 X_5$	$X_7$	1	2	3
$O_4 O_8$	$X_1 O_2$	$X_5 O_6$	4	5	6
$X_7$	$O_6 O_8$	$O_2 X_3$	7	8	9





## Method – Agents

- › Two agents playing X or O in the game
  - Random agent
  - Smarter random agent (Winning moves)
  - Feedforward agent (Feedforward neural network)
- › Agents use an internal gamemodel to play



## Method – Feedforward agent learning

- › Clients play batch: 500, 1000, 5000, or 10000 games
- › Server receives batches and trains network on them
- › Parameters of feedforward agent:
  - Threshold functions (tanh, sigmoid, no threshold)
  - Amount of levels of hidden neurons
  - Amount of hidden neurons per level
  - Learning rate
  - Initialization of the weights, always random
  - Epsilon, not adjustable, is 0.1



# Implementation

- › Software & Tools
  - Java Development Kit 1.6 (Netbeans 7.0)
  - Subversion 1.6.6
  - Java doc
  - junit tests (version 4)
- › Server & Client
  - Client plays games
  - Server trains networks



# Implementation (2)

The image shows two overlapping GUI windows. The top window, titled "Training on 1000 samples...", is divided into three main sections. The left section, outlined in pink, displays parameters for the X-Net: "X-Network: NO X-Network", "X-Net Learn Rate: 0.01", "X-Net Training Counter: 5790", and "X-Net Threshold: TANH". Below these are buttons for "Rename X-Net", "New X-net", "Open X-net", and "Save X-net". The middle section, outlined in red, shows "0 trainingsets waiting" and a progress bar. The right section, outlined in yellow, displays parameters for the O-Net: "O-Network: NO O-Network", "O-Net Learn Rate: 0.01", "O-Net Training Counter: 5259", and "O-Net Threshold: TANH", with buttons for "Rename O-Net", "New O-net", "Open O-net", and "Save O-net". Below these sections are fields for "Localhost = 0.0.0.0/0.0.0.0:1337" and "server port 1337", with "Stop" and "Exit" buttons.

The bottom window, titled "Q3T GUI client", is also divided into three sections. The left section, outlined in pink, shows "X-agent: Random" (dropdown), "X-Net name: NO X-Network", "O-agent: Neural Net O" (dropdown), and "O-Net name: NO O-Network". The middle section, outlined in red, displays statistics: "Games Played: 404", "X won: 204 (50.49 %)", "O won: 170 (42.07 %)", "ties: 30 (7.425 %)", and "Remembered sets: 8". The right section, outlined in cyan, shows "Server address: 0.0.0.0" and "Server port: 1337" with an "Interact!" button. At the bottom, a red-outlined section contains "Amount of games to play: 1000" (dropdown), "Stop automated run", "Stop playing", "Clear", "Save to CSV", and "Exit client" buttons.

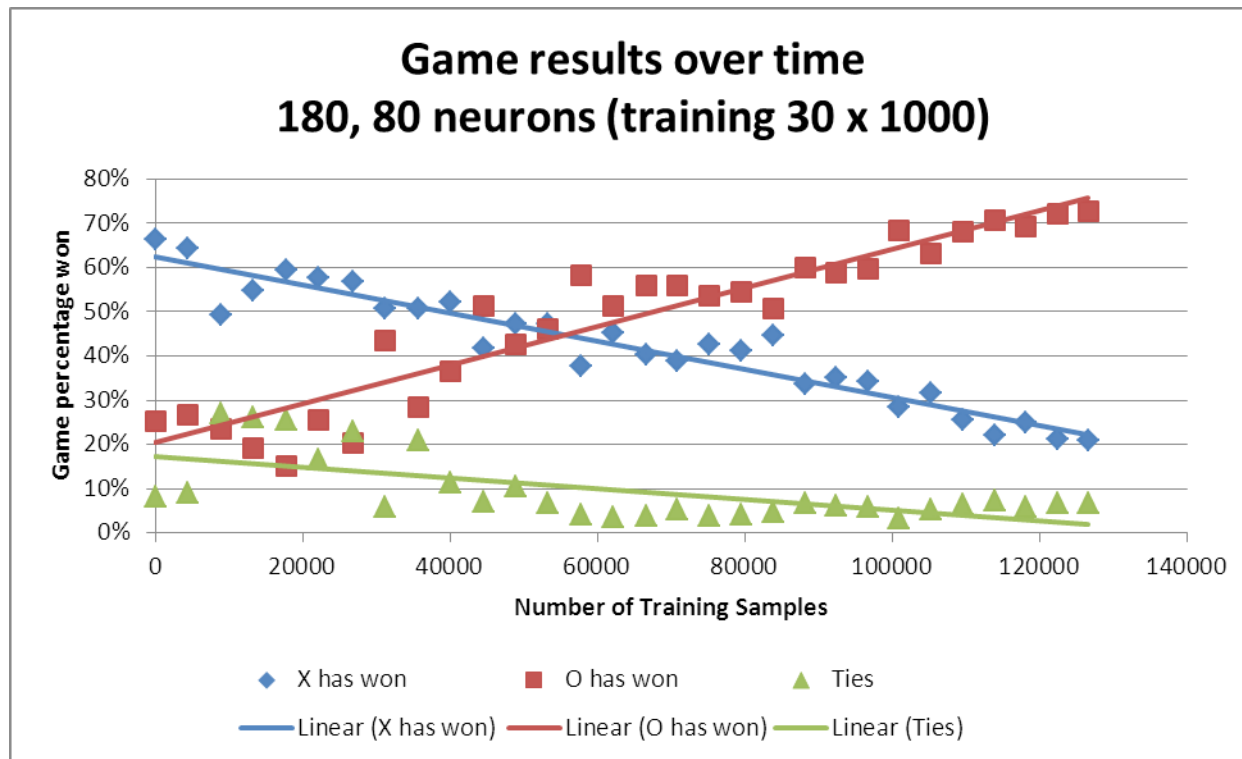


## Experimental set-up

- › X is played by random agent
- › O is played by feedforward agent
  - Epsilon = 0.1
  - Threshold function is 'tanh'
  - Learning rate = 0.01
  - Hidden neurons is varied (first, second layer):
    - 180, 80
    - 180
    - 80, 180

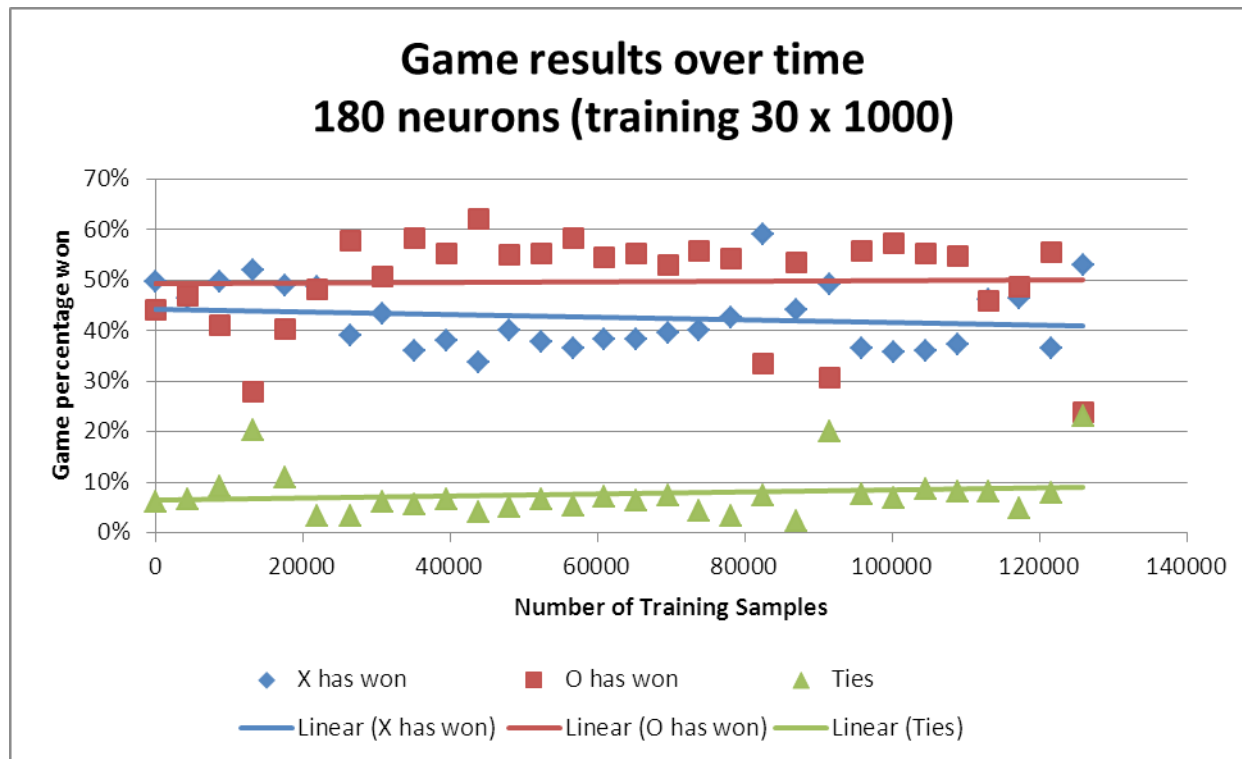


# Preliminary Results



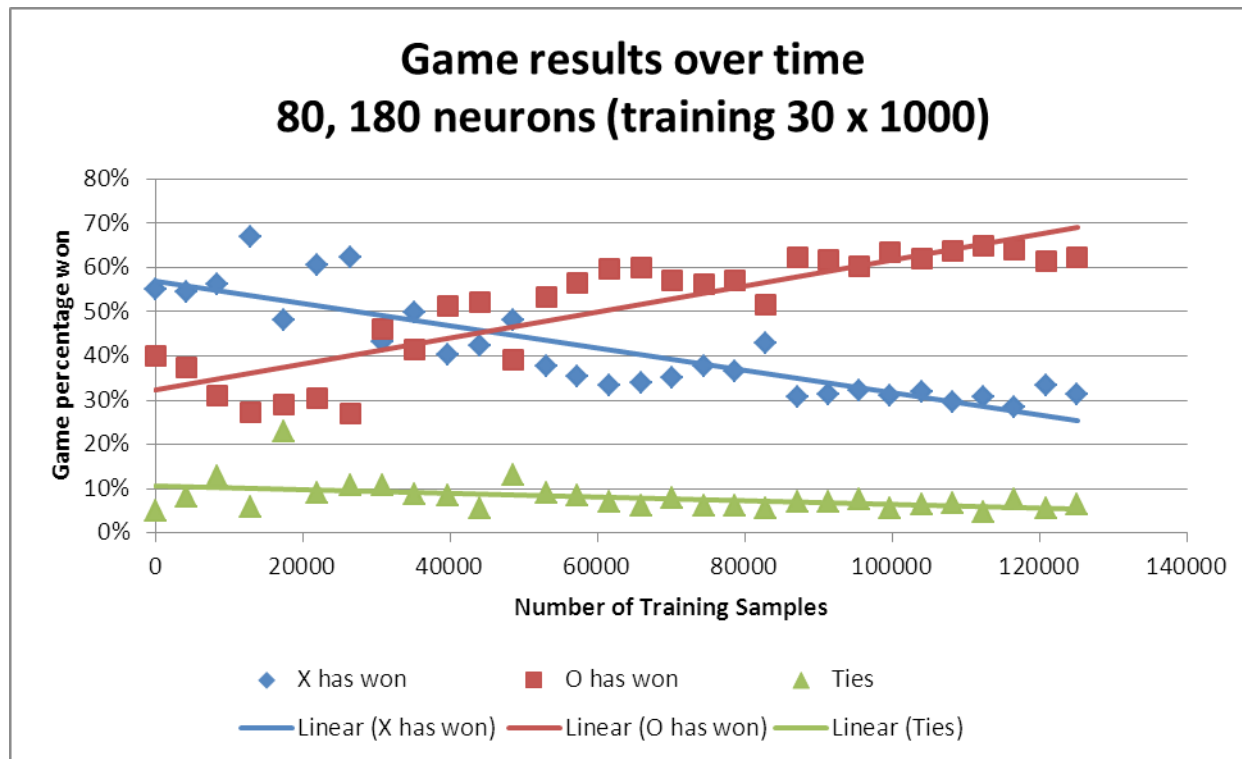


# Preliminary Results (2)





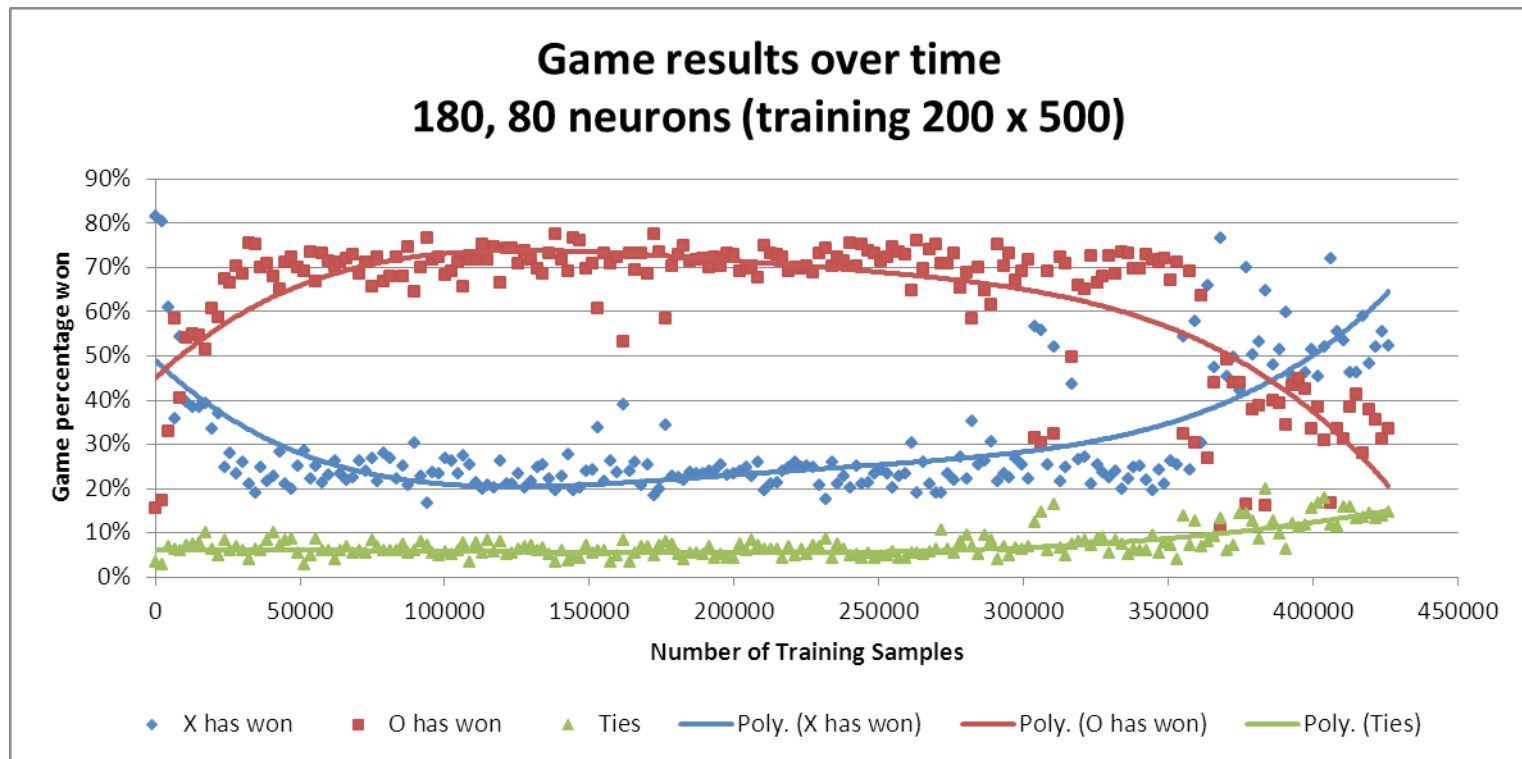
# Preliminary Results (3)







# Preliminary Results (4)





## Discussion

- › Algorithm shows a nice learning curve
- › But how to measure whether the results are good?
  - Random player is extremely bad
  - There are no human expert players (we know of)
- › Ideas about smarter random agent with win/block moves
- › Important: In what batch size is the server updated



## Discussion (2)

- › Looks like batch learning, but network actually trains on each game played
- › Entire sweep?
- › Upgrading to 4x4 tic-tac-toe



# Thank you for your attention

<http://www.codecup.nl>

<http://www.paradigmpuzzles.com/QT3Play.htm>

## Any questions?

