Training Minesweeper

Luis Gardea, Griffin Koontz, Ryan Silva

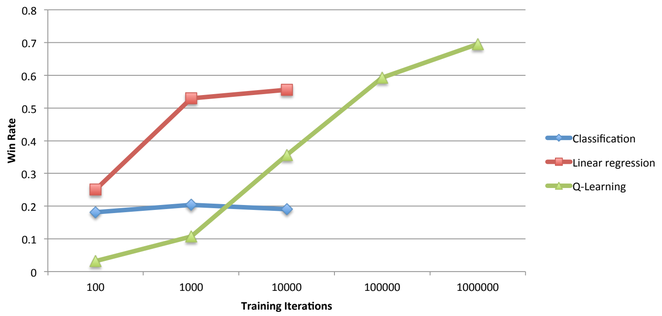
Introduction

Playing a game of Minesweeper involves uncovering tiles until the player uncovers a tile containing a mine or uncovers all of the tiles that do not contain mines. As the game progresses, the player is given limited information regarding the location of mines on the board. Using this information, a Minesweeper solver should determine which action to take at each stage of the game. While most games of Minesweeper will involve at least some level of guessing, running an analysis to determine the probability of a mine at each location can optimize performance despite this uncertainty. However, this probabilistic analysis is #P-Complete; it involves enumerating all possible solutions to an NP-Complete problem–determining mine locations–rather than simply finding a single solution. Given the difficulty of this algorithm, researchers have pursued a number of techniques to circumvent this explosive time complexity. In particular, we have devised and implemented four approaches: 1) a CSP that infers mine locations, 2) a Q-learning algorithm that involves modeling the game as an MDP 3) a localized machine learning approach that classifies individual tiles as containing a mine or not, and 4) a linear regression algorithm which predicts the probabilities of mine locations.

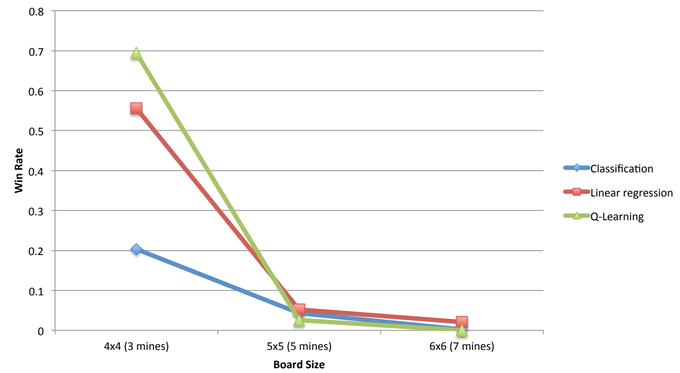
Data

- Data features corresponds to a single squares on the board: if the square has been uncovered, the feature value is the square's value. Otherwise, the feature takes on a value that demarcates the square as uncovered.
- The first machine learning implementation uses a feature extractor which obtains information about the surrounding squares of the square we wish to make a prediction on (i.e. whether the square contains a mine)
- This method has a simple but powerful interpretation: very often, the surrounding squares contain the information necessary to determine whether a square contains a mine with certainty. The model learns to make a prediction about a square based on local board configurations (a 3x3 or 4x4 region).
- A second approach looks at the global board state and tries to make predictions for the probability of every square containing a mine.
- The learning algorithm is trained using an existing Minesweeper solver, which calculates probabilities of every square containing a mine for a given board state.
- This method incorporates global information about the board- which implicitly includes the total number of mines in the board because this is used by the solver during training.
- The data for this approach can lie in a high dimension, and is somewhat sparse since many squares are not adjacent to any mines.

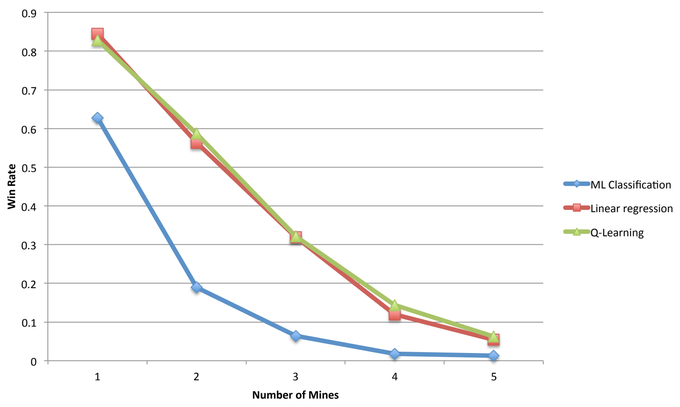
Number of Training Iterations on a 4x4 Board With Three Mines vs. Win Rate



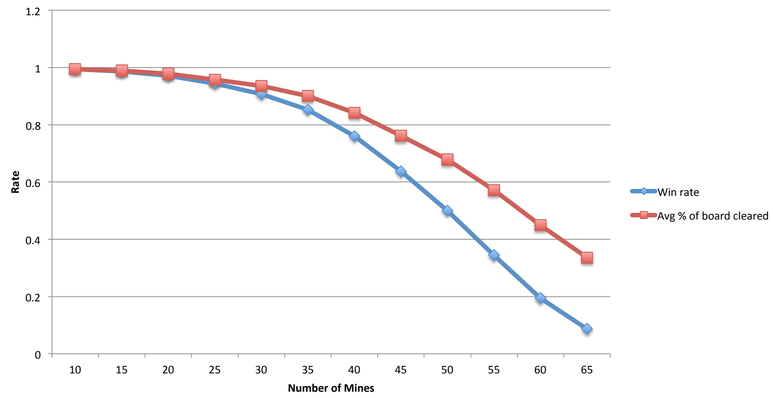
Board Size vs. Win Rate



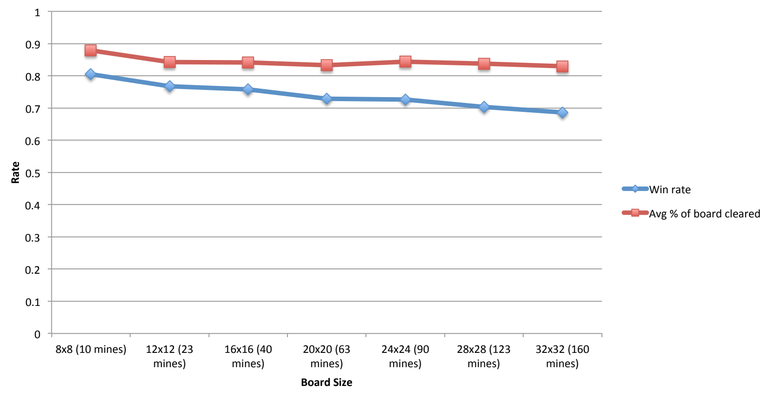
Number of Mines on 4x4 Board vs. Win Rate



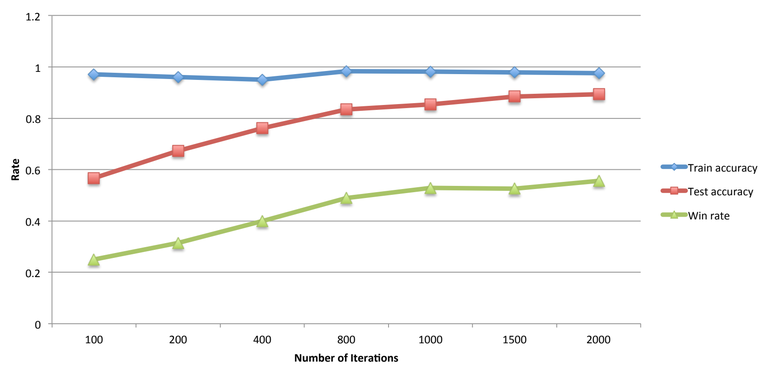
CSP: Number of Mines on a 16x16 Board vs. Success Rates



CSP: Board Size vs. Success Rates at Constant Mine Density



Linear Regression: Number of Iterations on a 4x4 Board With Three Mines vs. Success Rates



Methods

- The local model uses a binary classifier to predict whether a square contains a mine or not. We use logistic regression as our hypothesis model and preprocess the data with standardization.
- The number of parameters in our model depends on the size of the local area we wish to use. The model seems to suffer from high bias when only presented with information about the immediate neighbors. As the portion of the board looked at increases, the number of features in our also data increases. This corresponds to better training and testing accuracy.
- To use this model as an agent we predict the probability of each square in the perimeter (the uncovered squares that have a neighbor that is uncovered), and choose the square that has lowest predicted probability of containing a mine. This method may be generalizable to larger boards because local states are similar on any size board.
- The second method uses a regression approach to model the data. Specifically, we use a kernelized multiple ridge regression with cross validation for setting the meta-parameters.
- This model suffers from high variance and choosing the correct parameters for regularization is essential for good learning.
- To use this model as an agent for playing the game, we predict the probability of each square containing a mine, and choose the square that minimizes loss.
- Because we use a solver, we cannot train our algorithm on larger boards. Therefore, we optimize learning on smaller boards where learning the probabilities for every square is feasible.

Analysis

For the Machine Learning and Reinforcement learning approaches, we compared the results of different numbers of training iterations on a fixed board size to illustrate learning. We also compare varying board sizes, and varying numbers of mines for a fixed board size. When varying the number of training iterations, it is worth noting that the reinforcement learning takes many more iterations to reach the accuracy of the linear regression: this is due to the fact that it must visit each state multiple times to accurately learn the underlying MDP; varying iterations does not affect the classification approach. All three of these methods have decreased win rate when increasing the board size: this is due to the fact that these methods must string together increasingly long and complex series of correct predictions. Similarly, when increasing the number of mines, each guess has a higher likelihood of being incorrect. The CSP approach works better because it is able to solve for constraints and not make guesses. Increasing the density of mines for a fixed board size decreases the win rate mostly because the algorithm is forced to make more guesses, which typically occurs when mines are close together. Increasing the board size while fixing density does not seem to have a large impact on the win rate of the CSP.