Awarewolf – Solving the game of Werewolf with neural networks

Stefan Swaans sswaans@stanford.edu

Introduction

- Werewolf is a multiplayer game in which players attempt to solve who among them are secretly werewolves and vote to hang them.
- Human players rely on observation of other players' dialogue, votes, and deaths in order to make their deductions.
- Using a dataset of Werewolf games and a neural net, this project attempts to accurately guess the werewolves' identities using only players' votes and deaths.
- Goal is to see if there is a learnable voting and killing pattern that will identify werewolves semi-reliably.

Data

- We used an existing dataset called the Mafiascum dataset [1] to train our neural net.
- The Mafiascum dataset consists of about 750 games of Werewolf played in online chatrooms.
- We processed the data with standard language parsing techniques to create feature vectors of player votes, deaths, and werewolf identities for each game.
- We used these vectors as inputs and expected outputs for the model.

References

[1] Bob de Ruiter, George Kachergis. The Mafiascum Dataset: A Large Text Corpus for Deception Detection.

[2] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer. Automatic Differentiation in PyTorch.

Implementation

- We used PyTorch [2] to build a neural network to analyze sample games and predict which players were werewolves.
- The input to the model is a feature vector containing all players' **votes and deaths**. For player 1 out of 12, this may look like:

[0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]

Voted to hang players 2 and 3

Was eaten by werewolf, not hanged

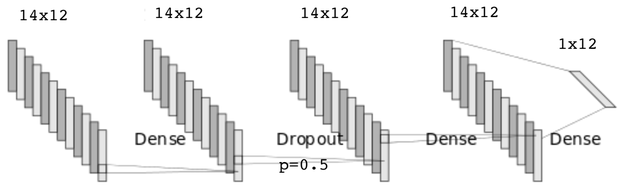
- The output of the model is a **vector of probabilities** where each probability represents the likelihood that the corresponding player is a werewolf. For a given game, this may look like:

[0.1, 0.2, 0.1, 0.5, 0.3, 0.2, 0.1, 0.4, 0.2, 0.2, 0.1, 0.1]

Highest prob, model predicts as first werewolf

Second highest prob, model predicts as second werewolf

- The model consists of a fully connected layer, a dropout layer, and two more fully connected layers.



- The network uses **binary cross entropy loss** to classify each of twelve players.
- We did not tune hyperparameters; they are set to initial values (Adam optimizer, learning rate of 0.001, betas of 0.9 and 0.999)

Results

- We used an 90/10 training/test split over eight epochs.
- **Individual accuracy** refers to how accurate the model was at guessing *individual* werewolves.
- **Team accuracy** refers to how accurate the model was at guessing entire werewolf *teams*.

	Individual Accuracy	Team Accuracy
Train	0.91	0.73
Test	0.55	0.25

Discussion/Error Analysis

- We tried several techniques to reduce **overfitting**, including tweaking the train/test split, number of epochs, adding the dropout layer, and experimenting with feature subsets.
- These techniques only slightly increased test accuracy, leading us to believe that the overfitting is primarily due to our **small dataset size**.
- However, an individual test accuracy of **0.55** is significantly better than chance and above the baseline of a novice human player, so we consider this experiment somewhat successful.
- **Most false positives** are players who were **hanged** (all villager victories require hanging werewolves)
- **No false positives** are players who were **eaten** (model learned that werewolves can't be eaten)
- A good deal of **false negatives** are werewolves who **voted to hang others scarcely** (uncommon play-style)
- **All mispredictions** are games that **werewolves won** (shorter game, less data, werewolves tend to win when data is scarce)