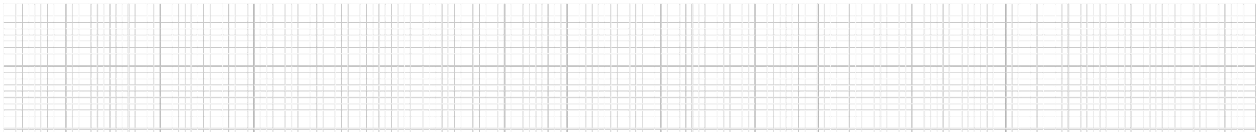
PREDICTING MLB GAMES USING A MULTILAYER PERCEPTRON NEURAL NETWORK

RYAN LEWIS

MAY 1, 2023



PROBLEM

- Predicting the outcomes of any sport match is a very difficult task. Coaches, managers, and especially sports bettors all try to create predictive models to find ways to win

- M.C. Purucker, in his paper “Neural Network Quarterbacking,” found that a supervised neural network with back-propagation was the most effective at predicting NFL games, and he achieved a 61% accuracy rate. This was enough to achieve long term profitability of sports bets

- My goal is to extend his work and achieve at least 60% accuracy in predicting MLB games, as well as try to achieve long term profitability in betting markets

DATASETS

- Schedules, batting statistics and pitching statistics for the entire 2022 season were downloaded from espn.com and stored in an sqlite3 database
- Each team's moneyline odds for each game were also downloaded
- Total of 2,396 games comprise the dataset, with 75% (1,797) of the games used to train, and 25% (599) used to test the network

Batting Splits

NAME	GP	W	L	AB	R	H	2B	3B	HR	RBI	AVG
Total	162	101	61	5509	789	1394	298	11	243	753	.253
Home	81	55	26	2656	395	680	137	5	125	378	.256
Away	81	46	35	2853	394	714	161	6	118	375	.250

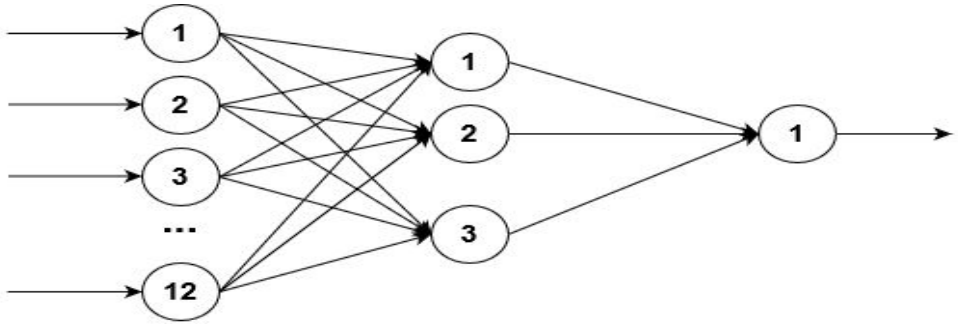
FEATURE SELECTION

- Six statistics were used as input for each team:
  - Runs scored per game
  - Runs allowed per game
  - Team batting average
  - Average number of runs allowed by the starting pitcher
  - Total win percentage
  - Home/away win percentage

- First 3 statistics were aggregated over a team's most recent 30 games
  - Accounts for a team's hot and cold streaks while still avoiding outlier games

THE NETWORK

- 6 input features for each team for a total of 12 values in the input vector
- Through testing, a hidden layer of 3 neurons was found to produce the best results
- Output is binary, with a 0 representing a home team win and 1 for an away team win



THE NETWORK (cont)

- The scikit-learn library was used to construct the network. It offers 3 different optimization functions for its MLPClassifier:
  - Stochastic gradient descent
  - Adam
  - L-BFGS
- All three were tested to find the most accurate results

```
# Trim and scale datasets
X_pd = pd.DataFrame(np.array(X))
X_train, X_test, y_train, y_test = train_test_split(X_pd, y, shuffle=False, test_size=0.25)
sc_X = StandardScaler()
X_trainscaled = sc_X.fit_transform(X_train)
X_testscaled = sc_X.transform(X_test)

# Train network
clf = MLPClassifier(hidden_layer_sizes=(3,), activation="relu", random_state=1, solver="sgd", max_iter=1000)\
    .fit(X_trainscaled, y_train)

# Predict test data
y_pred = clf.predict(X_testscaled)
```

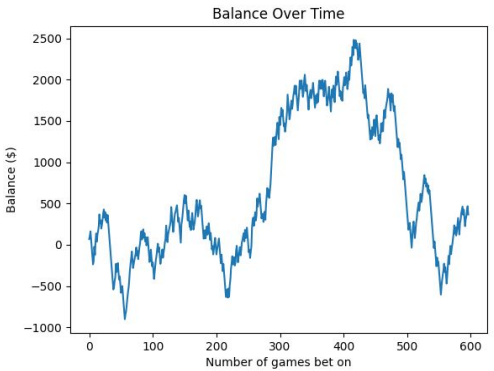
PERFORMANCE ANALYSIS

- The accuracy of each of the three optimization algorithms were recorded
- For each, A simulation was performed to evaluate the profitability in betting markets
    - Starting balance of $0
    - For each prediction, a $100 bet was “placed” on the winning team
    - Incorrect prediction: $100 subtracted from balance
    - Correct: Profit was calculated using moneyline odds and added to balance
    - Was done for each of the 599 games in the test dataset

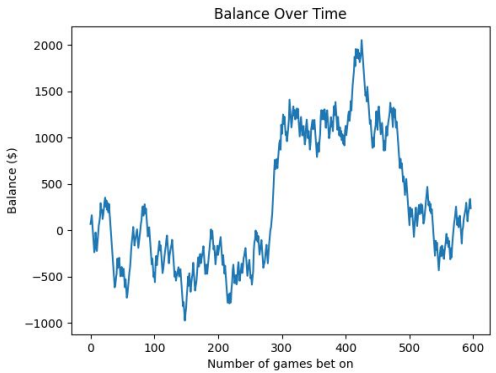
RESULTS

Algorithm	SGD	ADAM	L-BFGS
Accuracy	55.9%	56.9%	62.1%
Ending Balance	$367.05	$238.74	$3797.48

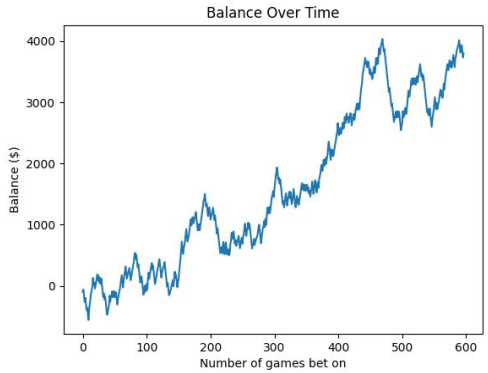
SGD



Adam



L-BFGS



LESSONS LEARNED

- MLP neural networks can be very effective in predicting the results of MLB games, even with only a few input features
- The optimization algorithm used can drastically change the results
- Even a very small accuracy difference can greatly improve profitability in betting markets, and just a 62% accuracy can be enough

FUTURE WORK

- Continue testing performance of this network over a larger sample size, possible that it just got lucky

- Conduct many more experiments on various other parameters:
  - Different statistics used as input
  - Aggregation size (other than 30 games)
  - Different structure/numbers of hidden nodes in the network
  - Different sizes of training/testing data