Neural Network Prediction of NFL Football Games

Joshua Kahn

ECE539 – Fall 2003
December 19, 2003

Table of Contents

Introduction

Over the past decade, football has truly become America's game. Now, with millions of people watching from their easy chairs every Sunday, the National Football League has become a multi-billion dollar business. There are countless internet sites that claim that they "know" the outcomes of future games. These sites come in many varieties, from the trustworthy (such as ESPN.com) to the downright seedy (just do a search for NFL football betting). With a plethora of different opinions out there, the question becomes, which ones are actually correct?

Most of the available prognostications are based on human opinion, which invariably leads to some degree of bias. During this project, a completely objective system was designed to predict the outcome of future NFL games, purely for academic purposes, of course. The problem in designing such a system, however, is the high number of "intangible" aspects in every game. For example, towards the end of the season a team with a losing record may play harder to save a favorite coach from being fired. Such a scenario would be difficult to predict from a purely objective standpoint. In general, however, it would seem plausible that it is possible to predict the outcome of a game based only on the statistics of the competitors.

The NFL keeps a very large set of statistics for each game played, much beyond those found in the box scores of a typical newspaper. Many prognosticators study "splits," or statistics broken down into extremely specific categories. For instance, data exists on the performance of every skill position player (quarterback, running back, wide receiver) when they play in cold versus warm weather, indoors versus outdoors, and even their performance on a given down with a given amount of yardage necessary to get a first down. The amount of available data becomes overwhelming when this information is considered; therefore, some small subset of data must be chosen that accurately predicts the outcome. This, of course, is easier said than done.

For this project, it was determined that the data used in making a prediction would be available from a standard box score, thus cutting down on the amount of available data substantially. Still, box scores provide a great deal of information on each game and the importance or relevance of each statistic must be determined prior to making a prediction (this will be discussed further in the next section).

A final consideration here is that winning occurs in a variety of ways. A quick glance at the total yardage row in a box score does not correctly identify the winner of the game, although it does lend some insight. In fact, there are countless examples of games in which the team with more total yardage actually lost. This indicates that there is no linear mapping method in which a winner can be chosen based solely on a group of statistics. Instead, a neural network could be used to perform non-linear mapping based on a variety of statistics.

Work Performed

In order to build a neural network that was capable of accurately predicting the outcome of a NFL football game, several steps were required.

Data Collection

As mentioned earlier, it was determined early on that all attempts would be made to ensure that the statistics used for prediction would be available from a typical NFL box score. Also, as previously mentioned, winning occurs in a variety of ways. An easily visible pattern does not exist that indicates which team will win (if it did, this problem would be easily solved); therefore, a large amount of data must be provided. A large data set would include a variety of modes that win football games.

To create the data set used in this project, box scores for every NFL football game for the 2003 season up to week 14 were collected. This provided a total of 208 data sets that could be used for training and testing purposes. In addition, the outcome of every game was recorded so that wins and losses could be mapped to the appropriate statistical information. Box scores were collected from NFL.com (http://www.nfl.com/), the official web site of the NFL. Microsoft Excel was used to capture the data as the newest versions include a web query feature that can be used to import data from any web table. This made data collection slow, as it was a manually process, but extremely accurate as no human error could be introduced to the recording process. Box scores were grouped based on week and stored in individual Excel workbooks, with one worksheet per box score. An example box score can be found in Appendix 1.

After the box scores were collected, additional data was required in order to make accurate predictions. Team averages at the end of week 13, 14, and 15 were acquired as described above. This data provided information on the average weekly performance of a team in a number of key statistical categories. During the season, each NFL team has a week in which it does not play. By the end of week 13, however, all of the teams had already had their bye weeks, so all had played a total of twelve games. Finally, a baseline was created based on predictions made by ESPN sportscaster.

Data Extraction

Once all of the data had been collected, it became necessary to create data sets that could be used by the neural network. A Perl script was written that was capable of parsing entire Excel workbooks to extract the desired data. Essentially, the script would read each worksheet (one game) in every workbook (one workbook contained all data for one week) in a given directory, thus separating data based on the week and the game. This provided a great deal of control in terms of the information that was collected from each box score and the manner in which it was presented to the neural network. The script, *processStatistics.pl*, can be found in it final form in Appendix 2.

Preliminary Study

Armed with a Perl script that made it possible to extract any desired data from each of the box scores, it became necessary to determine which data was most useful in predicting the outcome of NFL games. First, a prediction methodology proposed by Purucker [3], was examined to determine if a similar method would be appropriate here. Next, the available data was analyzed using Matlab to look for statistics or groups of statistics that may be valuable in prediction. An example of this analysis is found in Figure 1, a plot of the outcome of each of the 208 games

based on total yardage differential, time of possession differential, and turnover differential. Wins are indicated by an asterisk and losses by a circle.

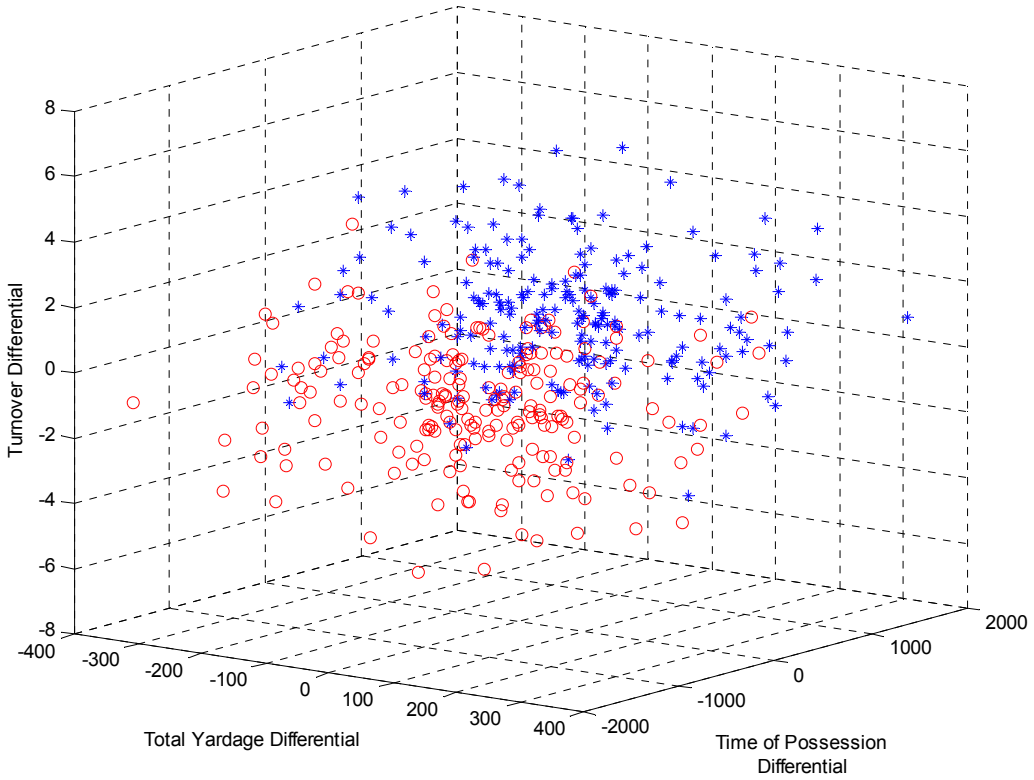

Figure 1: Outcome of 208 games (weeks 1-14) based on total yardage differential, time of possession differential, and turnover differential.

As seen in Figure 1, although no hyperplane can be drawn to definitively separate the data into wins and losses, there does appear to be some predictive value in these statistics, if the appropriate neural network is used. Before continuing, it is necessary to note that differential statistics were used as they are indicative of both offensive and defensive performance – both are key to winning a football game. A differential statistic is the difference between the offensive and defensive statistics. For example, the total yardage differential is found by subtracting the total yards allowed by the defense from total yards gained on offense. From an offensive standpoint, it is preferable to have a positive total yardage differential. As seen in the plot above, a positive yardage differential is also sometimes indicative of a win, as most scoring is done by the offense.

Using both the preliminary study results and the methodology described by Purucker, the following statistics were chosen as being most indicative of a winning football team. Any statistics relating to scoring (including point differential) were excluded as this would bias the network toward a single feature in determining the outputs.

- Total yardage differential
- Rushing yardage differential
- Time of possession differential (in seconds)
- Turnover differential
- Home or away

Originally, red zone efficiency was also included; however, this data was not recorded by NFL.com for week 12 games, making it an unusable statistic.

Training and Prediction Set Creation

With a small subset of data, the Perl script was modified to calculate the appropriate information from the available box scores. Using box score row titles, the Perl script determines where each of the desired data exists in the box score and then calculates the differential between the two teams. The data set for each game is then written to an output file, which can be seen in Figure 2. One output file is created for each week and prior to script completion, all of the output files are combined into a single training file that contains data of every game. Each row in the output files contains the data that will eventually be used as the input feature vectors to the neural network. The feature vectors contain the above statistics for both teams as well as the outcome of the game for both teams.

100	56	-30	-1	-1	-100	-56	30	1	1	-1	1
61	31	586	-1	-1	-61	-31	-586	1	1	-1	1
21	-17	-286	4	-1	-21	17	286	-4	1	1	-1
-142	-45	-692	0	-1	142	45	692	0	1	-1	1
31	7	286	0	-1	-31	-7	-286	0	1	-1	1
-49	-156	-518	-1	-1	49	156	518	1	1	-1	1
128	136	36	-1	-1	-128	-136	-36	1	1	1	-1
117	68	920	0	-1	-117	-68	-920	0	1	1	-1
-93	68	28	5	-1	93	-68	-28	-5	1	1	-1

Figure 2: Sample output file. Columns 1-5 contain data for the away team and columns 6-10 data for the home team. Columns 11 and 12 indicate the outcome of the game, column 11 is the away team's result and column 12 is the home team result. -1 indicates a loss, +1 indicates a win.

Figure 2, contains the week 9 output from the Perl script. As previously mentioned, each row in the file contains the results of each game for the given week. Columns one to five each row contains the statistics for the away team and columns six to ten contains the statistics for the home team. The last two columns provide the outcome of the game. Column eleven contains the result for the away team and column twelve, the result for the home team. A value of -1 indicates that the respective team lost the game, whereas a value of +1 designates the team as the winner. Table 1 provides definitions for the data contained in each column of the output file.

Table 1: Column definitions for data in Perl script output file.

Column(s)	*Data*
1, 6	Total yardage differential (away/home)
2, 7	Rushing yardage differential (away/home)
3, 8	Time of possession differential (away/home)
4, 9	Turnover differential (away/home)
5, 10	Away team (-1), home team (1)
11	Away team outcome
12	Home team outcome

After creation of the an output file for each week and a single file containing data on all 208 games (to be used as the training set for the neural network), a testing set that would be used to predict the outcome of future games could be created.

In actuality, two prediction sets were created – one containing the season average for each of the feature statistics and the other containing three week averages for each team. In both cases, the data was in the format as shown in Figure 2; however, the outcome of the game (columns 11 and 12) was not included. Two prediction sets were created, one based on season averages and the other on three-week averages, as the predictive capability of the two was not known. In other words, at this point it was unknown which of the two would be a better prediction of future performance – a long-term or short-term average. The three-week average set was generated using the Perl script, season averages were obtained from NFL.com. As will be explained later, both sets were applied to the neural network and only one was chosen as it provided better results.

One final set of data was created as a means to test the output of the neural network. This data set was essentially the data set for week 14, which included the outcome of each of the games. In addition, a training set containing 192 games was created that did not include the week 14 games. Testing was used to determine the effectiveness of the neural network prediction with known, instead of predicted values.

Neural Network Selection

A wide variety of neural networks were considered for this project, however, several factors lead to the choice of a back-propagation multi-layer perceptron network. First, the use of the back-propagation network allows the use of supervised learning, which is appropriate for this topic. As previously mentioned, with the large number of modes in which a team can win a football game, exposure to a large number of possibilities under supervised learning conditions would yield a network that had good predictive capability. In addition, this neural network provides a great deal of flexibility in terms of how it is set-up and the parameters that it uses (such as the learning rate). This factor allowed the creation of a neural network structure well-suited for the data set. Finally, the problem in this project essentially boiled down to a classification problem, a topic back-propagation networks are known to be able to solve.

The algorithm and code used to build the back-propagation multi-layer perceptron is based on the code supplied by Professor Hu on the course web site. Additions were made so that the network had a predictive capacity. Once the network was trained using the supplied code, the prediction set data was applied to the network and results recorded. Additional details will be provided momentarily.

Neural Network Parameter Selection

One of the benefits of using a back-propagation network is in its flexibility. A great deal of testing was performed in order to select the best network structure and parameters for the given data set. In order to perform structure and parameter selection, the test data set (explained above) was applied to networks with a variety of configurations. Each configuration was different from the previous in only one parameter value, thus allowing the selection of the best

value for each parameter. The data set was applied to each configuration a total of eight times and the classification rate for the week 14 test data (contains actual values and outcomes) was recorded and averaged. In this manner, the best value for the learning coefficient, momentum, number of hidden neurons, and network structure were determined. Tables 2 to 4 displays the results of this testing.

Table 2: Determination of learning coefficient and momentum parameters.

mu = 0.8				*mu=0*	
alpha	*0.01*	*0.05*	*0.1*	*0.01*	*0.05*
1	87.5	75	75	93.75	75
2	87.5	81.25	75	93.75	93.75
3	75	81.25	81.25	93.75	81.25
4	87.5	81.25	81.25	87.5	75
5	81.25	87.5	75	87.5	75
6	81.25	87.5	81.25	93.75	81.25
7	93.75	87.5	81.25	93.75	87.5
8	87.5	81.25	81.25	93.75	81.25
Average	85.15625	82.8125	78.90625	92.1875	81.25

Table 3: Determination of the number of hidden neurons.

	Number of Hidden Neurons					
	2	*3*	*4*	*5*	*6*	*8*
1	75	93.75	87.5	87.5	81.25	87.5
2	81.25	93.75	87.5	81.25	87.5	81.25
3	87.5	93.75	87.5	87.5	81.25	81.25
4	81.25	87.5	87.5	87.5	87.5	87.5
5	87.5	93.75	87.5	87.5	81.25	81.25
6	87.5	93.75	87.5	81.25	81.25	87.5
7	87.5	93.75	81.25	87.5	87.5	81.25
8	75	93.75	81.25	87.5	81.25	81.25
Average	82.8125	92.96875	85.9375	85.9375	83.59375	83.59375

Table 4: Determination of neural network structure.

	Network Structure	
	12-3-2	*12-3-3-2*
1	93.75	93.75
2	93.75	87.5
3	93.75	93.75
4	87.5	93.75
5	93.75	93.75
6	93.75	93.75
7	93.75	87.5
8	93.75	87.5
Average	92.96875	91.40625

The results of the parameter and structure testing led to a back-propagation neural network with a learning coefficient (α) of 0.01, momentum (μ) of 0, and a network structure of 10-3-2. This configuration was found to be the most effective in classifying data found in the testing set. It

employs a slow learning coefficient, no momentum, and a simple structure. As will be discussed later, this network yielded very high classification rates for known data.

Data Preprocessing

Prior to applying the training, testing, or prediction data to the neural network, some preprocessing was performed on the data. Processing included singular value decomposition, which gives additional weight to the most pertinent features prior to input to the neural network, thus allowing more effective training of the network. Figure 3 is the code snippet that was used to perform this processing on the data. This code was used on all three data set types. This was the last step prior to applying the data to the neural network.

```
% Load training and testing data
data = load ('training.txt');
% Remove output vectors
tr = data (: , 1:10);

% Perform analysis
[p1 s1 q1] = svd (tr);
v_p1 = p1(: , 1:4);
pm1 = v_p1 * v_p1';
newdata1 = pm1 * tr;

% Replace output vectors
newdata1(: , 11:12) = train(: , 11:12);
```

Figure 3: Code used to perform principal component analysis on training, testing, and prediction data.

Making Predictions

Once the data was ready to be applied to the neural network, three additional steps were involved in making the actual predictions. First, the training set was applied to the network as a means of training. Next, the prediction set was applied to the network; this produced two outputs for each prediction vector (game) – the predicted outcome for the away team and the prediction for the home team. The data was applied to the network three times and the results were recorded. Finally, these results were averaged to determine a final prediction. The neuron with the higher relative output was considered the winner. In other words, if output neuron one had a higher output value than output neuron two, the away team was predicted as the winner (and vice versa). This technique was applied for both the three-week average prediction set and the season average prediction set.

Results

Predictions were made using both prediction sets and were tested for weeks 14 and 15 of the 2003 NFL season. In both cases, the season average prediction set was more effective in predicting the outcome of the games. For week 14, the season average prediction set generated 75% correct outcomes, whereas the three week average set correctly predicted 62.5% of the games. The week 15 prediction rate was 75% using the season average and only 37.5% of the games using the three week average. Tables 5 and 6 display the results of weeks 14 and 15,

respectively. Games that were predicted incorrectly using the season average prediction sets are highlighted.

Table 5: Results of week 14 of the NFL 2003 season. Games incorrectly predicted using the season average prediction set are highlighted.

Green Bay def. Chicago	Baltimore def. Cincinnati
Philadelphia def. Dallas	Jacksonville def. Houston
Indianapolis def. Tennessee	Pittsburgh def. Oakland
San Diego def. Detroit	Minnesota def. Seattle
Tampa Bay def. New Orleans	New York Giants def. Washington
San Francisco def. Arizona	Denver def. Kansas City
New England def. Miami	Buffalo def. New York Jets
Atlanta def. Carolina	St. Louis def. Cleveland

Table 6: Results of week 15 of the NFL 2003 season. Games incorrectly predicted using the season average prediction set are highlighted.

Indianapolis def. Atlanta	Tennessee def. Buffalo
Kansas City def. Detroit	Tampa Bay def. Houston
New England def. Jacksonville	Minnesota def. Chicago
New York Jets def. Pittsburgh	St. Louis def. Seattle
Cincinnati def. San Francisco	Oakland def. Baltimore
Denver def. Cleveland	Carolina def. Arizona
Dallas def. Washington	Green Bay def. San Diego
New Orleans def. NY Giants	Philadelphia def. Miami

When the testing set was applied to the neural network, a classification rate of 93.75%, which is substantially higher than the output when the prediction sets are applied. This indicates that improved prediction data would lead to more accurate predictions. Still, some fallibility exists in the network, as the classification rate is not 100%.

Baseline Study

In order to validate this project, it is necessary to compare its results with those of other prognostication systems. First, Purucker's 1996 study [3] correctly predicts NFL games with an average accuracy of 60.7%, which is significantly less than the accuracy of this system. Purucker's study also tested several other neural network architectures, such as a self-organizing map (SOM); however, the back-propagation network was proven to be the most effective.

The system developed in this project was also tested against the predictions of eight sportscasters on ESPN.com. Through week 13 of the 2003 NFL season, these experts had correctly picked an average of 63% of the total games. In week 14, 57% of the games were picked correctly, on average, and 87% were correctly predicted in week 15. On average, the system designed in this project was 3% more accurate in predicting games during weeks 14 and 15. In both cases, the neural network system was more effective in correctly predicting the outcome of NFL football games.

Discussion

Of the eight games that were incorrectly predicted by the neural network over the course of weeks 14 and 15, two could be considered "upsets." These are games in which a team defeats a team with a substantially higher winning percentage. Four of the remaining size games were games that were "too close to call," or games in which the winner is very difficult to determine. The last two games, from a subjective standpoint, are considered misclassifications in which the neural network predicted the incorrect outcome. Table 7 classifies each of the incorrectly predicted games in one of these three categories.

Table 7: Games incorrectly classified by the neural network and possible associated reasoning as to misclassification.

Game	*Misclassification Reasoning*
Philadelphia def. Dallas	Misclassification
San Diego def. Detroit	Too close to call
Atlanta def. Carolina	Upset
Minnesota def. Seattle	Too close to call
New England def. Jacksonville	Misclassification
New York Jets def. Pittsburgh	Too close to call
Cincinnati def. San Francisco	Too close to call
Oakland def. Baltimore	Upset

As previously mentioned, both season average prediction data and three-week average prediction data was applied to the network. In the end, the season prediction data was more effective in predicting the outcome of games. One possible explanation for this is that NFL teams are fairly consistent over the long-term, thus meaning that long-term data is more effective in predicting future outcome. On the other hand, many believe that sport teams can get "hot," or go on a streak in which they play at an extremely high level. If the latter were true, then the three week average data would be more effective. Further study would be required to definitively answer this question.

There is the possibility that the neural network classification rate could be improved no matter which prediction data is used with additional training data; however, this additional data may be met with diminishing returns. Due to the typically volatile nature of sports, outcomes can be difficult to predict based solely on objective data. It is possible that the additional of a "human element" to the prediction scheme would further enhance its accuracy. For example, the Las Vegas betting line or the subject team rankings published by any number of authorities could be added as additional feature vectors. These elements take into consideration the aforementioned intangible aspects of sports, such as injuries.

This study was conducted towards the end of an NFL season, thus providing a large statistical basis for training. The question, therefore, is the effectiveness of this system in predicting games earlier in the season. Although not studied here, it would be reasonable to assume that statistics from past seasons could be used to train the network as the modes in which teams win games is not likely to change over time. Overall, this project indicates that an objective, statistically based system can be implemented to predict the outcome of NFL football games.

References

1. Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

2. ESPN.com, http://www.espn.com [Retrieved Dec 2003].

3. Purucker, M.C. (1996) Neural Network Quarterbacking. *Potentials, IEEE*, vol. 15:3, pp. 9-15.

4. NFL.com, http://www.nfl.com [Retrieved Dec 2003].

Appendix 1 – Sample Box Score

Chicago Bears versus Green Bay Packers

Week 14 – December 7, 2003

	CHI	GB
Points	21	34
1st Downs	13	17
Rushing	3	6
Passing	10	10
Penalty	0	1
3rd-Down Conversions	5-17	5-17
4th-Down Conversions	0-3	0-1
Punts-Average	6-42.7	6-38.0
Return Yards	281	212
Punts-Returns	1-25	3-3
Kickoffs-Returns	8-211	3-61
Int.-Returns	1-45	3-148
Penalties-Yards	5-45	8-57
Fumbles-Lost	2-2	1-0
Time Of Pos.	24:36	35:24
Total Net Yards	275	307
Total Plays	63	71
Average Gain	4.4	4.3
Net Yards Rushing	44	97
Rushes	20	38
Avg. Per Rush	2.2	2.6
Net Yards Passing	231	210
Comp.-Att.	17-40	22-33
Yards Per Pass	5.4	6.4
Sacked-Yards Lost	3-25	0-0
Had Intercepted	3	1
Touchdowns	3	3
Rushing	0	1
Passing	1	1
Other	2	1
Extra Points	3-3	3-3
Passing	0-0	1-1
Field Goals	0-0	4-4
Red Zone Efficiency	0-1-0%	1-3-33%
Goal To Go Efficiency	0-0-0%	1-1-100%
Safeties	0	0

Source: http://www.nfl.com/

Appendix 2 – Source Code

```
#!/usr/bin/perl

#########################################################################
# This script builds all necessary files to run a neural network
# back-propagation network to predict the outcome of NFL games.  The script
# requires Microsoft Excel files containing one worksheet for each game that
# has been played previously in the season.  One Excel file should be created
# for each week.  The script will parse these files to create weekly schedules
# and statistics.  These statistics are then used as inputs to the BP algorithm
#
# The script creates a training file that should be used to train the BP
# algorithm.  In addition, a predicition file is created that contains
# statistical predictions for each game in the coming week.  These predictions
# are based on the average of the previous three weeks of play for each team.
# Predicitions are to be entered into the BP, which will predict the outcome
# of the coming week's games.
#
# Josh Kahn
#
# V1.0 - December 5, 2003
#
# Documentation for Spreadsheet::ParseExcel module:
#   http://search.cpan.org/~kwitknr/Spreadsheet-ParseExcel-0.2602/ParseExcel.pm
#########################################################################

use strict;
use Spreadsheet::ParseExcel;

#################
#     MAIN      #
#################

# Define a few variables for later use
my @prevWeeks = ("Week12" , "Week13" , "Week14");
my $currSched = "Week15Sched.txt";
my $outFile   = "Week15Pred.txt";
my %prevWeeksData;

# Create a feature vector file for each week of the season
&createFeatureVectorFiles;
# Create one training file that is the combination of each of the above
&createTrainingFile;
# Determine the average for the last three weeks - this is used for prediction
#   of the next week's results
&determine3WeekAverage;


#################
#  SUBROUTINES  #
#################

#########################################################################
# Subroutine: createFeatureVectorFiles
#
# This subroutine parses Microsoft Excel files containing box scores from
# http://www.nfl.com and gathers a total of seven statistics.  These statistics
# are then written to a text file as an input vector for a BP network.
#
# Output vector format:
```

```
#  team1_total_yards  team1_rush_yards  team1_time_of_possession ...
#     team1_turnovers  team1_redzone_percentage  team2_total_yards...
#     team2_rush_yards  team2_time_of_possession  team2_turnovers...
#     team2_redzone_percentage  team1_result  team2_result
#
# Results vector: 1 for win, -1 for loss
#
# Parameters: none
# Returns:    none
#########################################################################
sub createFeatureVectorFiles {
   # Markers in spreadsheet
   my $points    = "Points";
   my $xtrapts   = "Extra Points";
   my $fblslost  = "Fumbles-Lost";
   my $timeofpos = "Time Of Pos.";
   my $totalyds  = "Total Net Yards";
   my $rushyds   = "Net Yards Rushing";
   my $intercep  = "Had Intercepted";
   my $redzneff  = "Red Zone Efficiency";

   my @statFiles = glob ("*.xls");

   my ($infile , $statfile , $schedfile);
   my (@scores , @turnovers , @timeofpos , @totalyds , @rushyds);

   foreach $infile (@statFiles) {
      # Set-up output file name
      $_ = $infile;
      ($statfile) = /^(\w+)\.xls$/;
      $schedfile  = $statfile . "Sched.txt";
      $statfile   = $statfile . ".txt";

      # Check to ensure the input spreadsheet actually exists
      unless (-e $infile) {
         print "File does not exist!!\n\n";
         exit (0);
      }
      # Open a text file for writing
      open (OUT , ">$statfile") || die "Could not write file $statfile: $!\n";
      open (SCHED , ">$schedfile") || die "Could not write file $schedfile: $!\n";

      # Constructor for perl excel parser
      my $wrkbk = new Spreadsheet::ParseExcel::Workbook->Parse($infile);

      # Determine the number of worksheets in Excel workbook
      my $wrkshtCount = $wrkbk->{SheetCount};

      for (my $shtNum = 0 ; $shtNum < $wrkshtCount ; $shtNum++) {
         my $sheet = $wrkbk->{Worksheet}[$shtNum];
         print SCHED "$sheet->{Name}\n";

         @turnovers = (0 , 0);

         foreach my $row (0..$sheet->{MaxRow}) {
            # Skip empty cells -- only looking at header column now
            next unless defined $sheet->{Cells}[$row][0];

            # Store header
            my $header = $sheet->{Cells}[$row][0]->{Val};

            # Check headers, perform operations if appropriate header found
            # Need to avoid "Extra Points" header when searching for "Points"
```

```
            if (($header =~ /$points/) && (! ($header =~ /$xtrapts/))) {
               my $team1score = $sheet->{Cells}[$row][1]->{Val};
               my $team2score = $sheet->{Cells}[$row][2]->{Val};

               if ($team1score > $team2score) {
                  @scores = (1 , -1);
               }
               else {
                  @scores = (-1 , 1);
               }
            }
            elsif ($header =~ /$fblslost/) {
               my @team1fbls = split (/-/ , $sheet->{Cells}[$row][1]->{Val});
               my @team2fbls = split (/-/ , $sheet->{Cells}[$row][2]->{Val});

               @turnovers = ($turnovers[0] + $team1fbls[1] , $turnovers[1] +
$team2fbls[1]);
            }
            elsif ($header =~ /$timeofpos/) {
               my $team1pos = &setTimeOfPossession ($sheet->{Cells}[$row][1]->{Val});
               my $team2pos = &setTimeOfPossession ($sheet->{Cells}[$row][2]->{Val});

               if (($team1pos + $team2pos) != 3600) {
                  print "Error in time of possession in $wrkbk->{File}:$sheet-
>{Name}\n";
               }

               @timeofpos = ($team1pos - $team2pos , $team2pos - $team1pos);
            }
            elsif ($header =~ /$totalyds/) {
               my $team1yds = $sheet->{Cells}[$row][1]->{Val};
               my $team2yds = $sheet->{Cells}[$row][2]->{Val};

               @totalyds = ($team1yds - $team2yds , $team2yds - $team1yds);
            }
            elsif ($header =~ /$rushyds/) {
               my $team1yds = $sheet->{Cells}[$row][1]->{Val};
               my $team2yds = $sheet->{Cells}[$row][2]->{Val};

               @rushyds = ($team1yds - $team2yds , $team2yds - $team1yds);
            }
            elsif ($header =~ /$intercep/) {
               my $team1ints = $sheet->{Cells}[$row][1]->{Val};
               my $team2ints = $sheet->{Cells}[$row][2]->{Val};

               @turnovers = ($turnovers[0] + $team1ints , $turnovers[1] + $team2ints);
            }
            else {}
         }

         # Find turnover differential
         @turnovers = ($turnovers[1] - $turnovers[0] , $turnovers[0] - $turnovers[1]);

         # Write feature vector to output file
         # Team 1 features - including away status
         print OUT "$totalyds[0]\t$rushyds[0]\t$timeofpos[0]\t$turnovers[0]\t-1\t";
         # Team 2 features - including home status
         print OUT "$totalyds[1]\t$rushyds[1]\t$timeofpos[1]\t$turnovers[1]\t1\t";
         # Output (final score) features
         print OUT "$scores[0]\t$scores[1]\n";

      }
```

```
      close (OUT);
      close (SCHED);
   }
}


#########################################################################
# Subroutine: setTimeOfPossession
#
# Calculates a team's time of possession in seconds from possession data in
# the format MIN:SEC.
#
# Parameters: time of possession data in xx:xx format
# Returns:    time of possession in seconds
#########################################################################
sub setTimeOfPossession {
   my $data = shift;

   my @digits = split (/:/ , $data);

   # Time of possession will be measured in seconds
   my $top = ($digits[0] * 60) + $digits[1];
   return $top;
}


#########################################################################
# Subroutine: createTrainingFile
#
# Builds a training file containing all of vectors created by the
# createFeatureVectorFiles subroutine.  This subroutine essentially builds one
# large training file that is the composite of all known vectors.
#
# Parameters: none
# Returns:    none
#########################################################################
sub createTrainingFile {
   # Test file
   my $trainFile = "training.txt";

   my @files = glob ("*.txt");

   open (OUT , ">$trainFile") || die "Could not open $trainFile: $!\n";

   foreach my $file (@files) {
      unless (($file =~ /Sched/) || ($file =~ /Pred/)) {
         open (IN , "$file") || "Could not open $file: $!\n";

         while (<IN>) {
            print OUT "$_";
         }

         close (IN);
      }
   }

   close (OUT);
}


#########################################################################
# Subroutine: determine3WeekAverage
#
```

```
# Determines the three week average and sets-up the input vector for BP
# prediction.  The subroutine parses the schedule file for the current week
# (defined above) and then parses the schedule files for the previous three
# weeks (also above).  It then uses this information to parse the weekly
# input vector files to determine a team's average statistics over the past
# three weeks.  The average for both teams is then written as a single input
# vector to an output file.  This output file is then input into the BP
# algorithm to predict the outcome of the coming week's NFL matchups.
#
# Parameters: none
# Returns:    none
##############################################################################
sub determine3WeekAverage {
   # Store previous week statistics to arrays
   for (my $i = 0 ; $i <= $#prevWeeks ; $i++) {
      open (IN , "<$prevWeeks[$i].txt") || die "Could not open $prevWeeks[$i].txt:
$!\n";
      my @week = <IN>;
      close (IN);

      # Create a hash of arrays
      $prevWeeksData{$i} = [@week];
   }

   # Open next week's schedule
   open (CURR , "<$currSched") || die "Could not open $currSched: $!\n";
   my @sched = <CURR>;
   close (CURR);

   # Open output file
   open (OUT , ">$outFile") || die "Could not open $outFile: $!\n";

   # Sort through next week's schedule
   foreach $_ (@sched) {
      (my $team1 , my $team2) = split (/-/ , $_);
      chomp ($team2);

      print "$team1 vs $team2\n";

      # Gather the data from the previous three weeks for each team
      my (%team1data , %team2data);
      for (my $i = 0 ; $i <= $#prevWeeks ; $i++) {
         my @teamIndices = &findTeamIndices ("$prevWeeks[$i]Sched.txt" , $team1 ,
$team2);

         $team1data{$i}  = [&gatherWeekData ($i , $teamIndices[0] , $teamIndices[1])];
         $team2data{$i}  = [&gatherWeekData ($i , $teamIndices[2] , $teamIndices[3])];
      }

      # Calculate the average value of each feature
      my (@team1ave , @team2ave) = (0 , 0 , 0 , 0 , 0);
      for (my $i = 0 ; $i <= $#prevWeeks ; $i++) {
         for (my $j = 0 ; $j < 5 ; $j++) {
            $team1ave[$j] = $team1ave[$j] + ($i + 1) * $team1data{$i}[$j];
            $team2ave[$j] = $team2ave[$j] + ($i + 1) * $team2data{$i}[$j];
         }
      }
      for (my $j = 0 ; $j < 4 ; $j++) {
         $team1ave[$j] = $team1ave[$j] / 6;
         $team2ave[$j] = $team2ave[$j] / 6;
      }

      $team1ave[4] = -1;
```

```
      $team2ave[4] = 1;

      # Write the result
      my $doOnce = 1;
      foreach $_ (@team1ave) {
         if ($doOnce) {
            printf OUT ("%.2f" , $_);
            $doOnce = 0;
         }
         else {
            printf OUT ("\t%.2f" , $_);
         }
      }
      foreach $_ (@team2ave) {
         if ($doOnce) {
            printf OUT ("%.2f" , $_);
            $doOnce = 0;
         }
         else {
            printf OUT ("\t%.2f" , $_);
         }
      }
      print OUT "\n";
   }

   close (OUT);
}


#########################################################################
# Subroutine: findTeamIndices
#
# This subroutine searches a specific schedule file to determine the position
# of the given team's game listing in the weekly input vector file.  Using
# the schedule file, it is then determined if the team's statistics listing is
# first or second in the input vector in the weekly file.  The subroutine then
# returns the index of the team's game in the weekly input vector file as well
# as whether it is the first or second listing.
#
# Parameters: 1) schedule to check for a specific teams game
#             2) team 1 symbol
#             3) team 2 symbol
# Returns:    1) the index of the team 1's game in the data file
#             2) position of team 1's stats (as team 1 or 2 in vector)
#             3) the index of the team 2's game in the data file
#             4) position of team 2's stats (as team 1 or 2 in vector)
#########################################################################
sub findTeamIndices {
   my $sched = shift;
   my $t1    = shift;
   my $t2    = shift;

   my $lineCount = 0;
   my @indices;

   # Open the schedule for past weeks - parse through and gather statistics
   open (IN , "<$sched") || die "Could not open $sched: $!\n";
   while (<IN>) {
      if (/$t1/) {
         # Store the location of this team's game in the schedule
         $indices[0] = $lineCount;

         # Determine if the team is the first or second team listed in the schedule
```

```
         my @teams = split (/-/ , $_);
         chomp ($teams[1]);

         if ($t1 =~ /$teams[0]/) {
            $indices[1] = 1;
         }
         else {
            $indices[1] = 2;
         }
      }
      if (/$t2/) {
         # Store the location of this team's game in the schedule
         $indices[2] = $lineCount;

         # Determine if the team is the first or second team listed in the schedule
         my @teams = split (/-/ , $_);
         chomp ($teams[1]);

         if ($t2 =~ /$teams[0]/) {
            $indices[3] = 1;
         }
         else {
            $indices[3] = 2;
         }
      }
      $lineCount++;
   }
   close (IN);

   return @indices;
}


##############################################################################
# Subroutine: gatherWeekData
#
# Seeks out the appropriate data for a team in a given week.
#
# Parameters: 1) index of which of the three past weeks' data to parse
#             2) index of the game in which the team is involved
#             3) determines if the team is listed first or second
# Returns:    team's data for the given week and game
##############################################################################
sub gatherWeekData {
   my $weekIndex = shift;
   my $gameIndex = shift;
   my $teamIndex = shift;

   # Get the array entry at $gameIndex from the appropriate week's data
   my $line = $prevWeeksData{$weekIndex}[$gameIndex];

   my @data = split (/\t/ , $line);

   # Grab only the data needed for the specific team
   my $retval;
   if ($teamIndex == 1) {
      @data = @data[0..5];  # Get data from columns 0 to 4
   }
   else {
      @data = @data[6..10];  # Get data from columns 5 to 9
   }
   return @data;
}
```