Time Out Before Time is Up: A Classification Model to Determine Timeouts at Crucial Moments In a Game

Sriharsha Guduguntla
University of California, Berkeley
sguduguntla@berkeley.edu

Jai Sankar
University of California, Berkeley
jai.sankar00@berkeley.edu

Chinmay Gharpure
University of California, Berkeley
cgharpure@berkeley.edu

1. TOPIC INTRODUCTION

The modern NBA is demanding. Players are expected to have more diverse skill sets. Organizations are expected to build super teams. League offices are expected to handle complicated logistics. Meanwhile, the pressure is mounting on coaches to formulate new strategies and inspire players. Among their many tough decisions, coaches often face immense criticism for choosing unnecessary or ineffective timeouts. This is seen in a recent incident from game five of the NBA finals. Raptors head coach Nick Nurse was heavily criticized for disrupting the Raptor's momentum in the fourth quarter by taking an unnecessary timeout. Immediately, the Warriors struck back with three straight three-point shots that sealed the outcome of the game. Similarly, in game four of the 2018 Western Conference Finals, Warriors head coach Steve Kerr chose not to take a timeout in the last ten seconds of the game, causing dysfunction in the team's offensive strategy. Several decisions like these have cost teams critical games in the past. To help decrease the frequency of these miscounts, we present a novel tool to predict whether a coach should take a timeout at a certain moment in the game. The model utilizes supervised machine learning techniques and deep learning to accurately formulate predictions.

2. HYPOTHESIS

Given eleven unique metrics quantifying team and coach history, our model will predict whether a timeout should be taken at any given moment in a game. These metrics or *features* are the number of successful timeouts for the coach's career (see Proposed Methodology), the coach's career winning percentage, the number of star players on the team, the point differential between timeouts, the FG percentage of the team, the number of turnovers, the lead differential, the number of timeouts left, the seconds passed between timeouts, whether the team is currently down, and the remaining time left in the game. While some features describe what has transpired up to the moment of interest in the game, others quantify the team's ability and the coach's history of success. After normalizing the features, we will train a model to produce a binary output of 1 and 0, 1 being that a timeout *should be taken* and 0 being that a timeout *should not be taken*.

3. POTENTIAL APPLICATIONS

Primarily, our model will transform the way coaches approach timeouts in a game. Modern coaches rely on video footage, advice from other coaching staff, player input, and past games and strategies to improve their decision-making. Film sessions in particular provide coaches with a strong understanding of the flow of the game. Meanwhile, player and staff input give them more interesting, but often subjective ideas. However, our model considers not only the flow of the game and thousands of successful timeout examples, but also considers the team's past performance and the coach's success rate. Ideally, the model will reduce bias as much as possible to present coaches with an objective result, enabling them to leverage the power of ten years of statistical data at the click of a button. With every new NBA game, the tool grows smarter, yielding more accurate predictions. Over time, coaches may notice patterns in the predictions and start to adopt some of the suggestions in future games. In the long term, we would like the tool to output specific reasons for taking a timeout. However, given that the inner workings of deep neural networks is still largely a mystery, pinpointing the exact reason for taking a timeout is difficult.

Another relevant application of the model is to the proposed call-challenge system starting in the 2019-2020 NBA season. According to the new system, a challenge is a timeout immediately after a play dealing with out-of-bounds calls, goaltending or basket interference, or a personal foul (NBA Rulebook). An example use of our model in conjunction with the call-challenge system is presented in a game between Team A and Team B. After a player on Team A commits a foul, Team B gets possession of the ball in the last five minutes of the game. Guided by the model, Team A knows that a timeout should be taken if they are on defense in the last five minutes of the game. Also, the model tells Team A to not take a timeout on defense in the last two minutes of the game. As a result, Team A is incentivized to challenge the play before the final two minutes in order to gain offensive possession.

4. PROPOSED METHODOLOGY

To create our model, we split our methodology up into four steps. Play-by-play data will be used to analyze how the team performs between the current timeout and the next

timeout. If the team's point differential increases and is positive, then we will label the timeout as *successful*. Otherwise, *unsuccessful*.

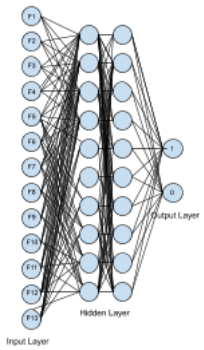
4.1 Compiling Features for Timeout Data

Using the play-by-play data from the last ten years, we will compile all the timeouts that have occurred in each game. From there, we will compute the eleven features, outlined in the hypothesis, for each timeout.

1. **Coach's Career Winning Percentage**
2. **Is Down** - If the current team is down at the moment of the timeout.
3. **# of Successful Career Timeouts for Coach** - Every successful timeout in every game by that coach
4. **# of Star Players** - The # of players on the team with twenty PPG in the previous season
5. **Point Differential Between Timeouts**
6. **FG % between Timeouts**
7. **# of Turnovers between Timeouts**
8. **Lead Differential** - Largest lead - Current lead
9. **# of Timeouts Left**
10. **Time Between Timeouts**- The overall time elapsed between the previous timeout and the current timeout.
11. **Remaining Time in the Game**

4.2 Creating an ML Model

We propose two different algorithms to solve this classification problem: leveraging deep learning by implementing a *multilayer perceptron* (MLP) or constructing a *support vector machine* (SVM) model with an input of the eleven features described in Step 4.1. The MLP will consist of four layers: the input layer, two hidden layers, and an output layer. The input layer will house eleven neurons representing each feature. The output layer will consist of one neurons representing whether or not a timeout should be taken. The model will consist of two hidden layers with ten neurons each. Since the timeout data is not linearly separable, the hidden layers will help improve the accuracy of our predictions. Since the features are normalized, the value of each neuron in the input layer is between 0 and 1. As this network feeds forward, each neuron will be a weighted sum of the previous layers neurons applied to an activation function. In this case, we will use a sigmoid function as the activation function. The value of the output neurons are determined by the following equation: $\mathrm{b}_m = \sigma(\sum_{i=1}^{n} w_i a_i - \beta)$



By doing so, the value of the one final neuron is used to classify whether a timeout should be taken or not. In one forward pass through the MLP, there are 108 different weights and 13 different biases. Therefore, we have to choose the correct weights and bias values in order to classify the final two neurons as accurately as possible. In order to do so, we will train our model using the labeled timeout data.

Initially, we will set our weights and biases to random values. Then we will use the eleven features of our first labeled data point and run through the MLP to calculate the value of the output layer neurons. We can then subtract the values from the output layer neurons and the labeled data to calculate the error. We can then create a cost function where the parameters are the 121 weights and bias values and the output is the error. Now, we perform gradient descent by taking the negative gradient of the cost function with respect to each of the bias and weight variables. Then, we can modify the weights and biases by this value. If we repeat this process for all the training data points, the weights and biases will be able to more accurately predict the values of the outer layer neurons.

For our model, an SVM will classify whether a timeout should or should not be taken at a given moment. The SVM takes in the data of the eleven features to classify each timeout as well as the result of the timeout. Using this data, the SVM generates multiple hyperplanes . These hyperplanes divide data classified as timeouts that should and should not be taken. A *hyperplane* is a subspace with a dimension one less than that of its surrounding dimension. We can set the equation for the hyperplane to be: $w^T x + b$. The vector x represents the different axes, w represents the coefficients, and v is the translation vector. To find the hyperplane that separates both sets of data, find the vector w that maximizes the distance between the two hyperplanes $w^T x + b = 1$ and $w^T x + b = -1$, setting the final hyperplane equation to be $w^T x + b = 0$.

5. TYPES OF DATA USED

To determine the individual features, we will use play-by-play data from the NBA as well as Basketball Reference. In order to obtain the defensive rating per team, we will use published team data from the NBA website. Finally, in order to determine league rule changes for timeouts and the challenge flag, we will use articles and datasets from the NBA website.