Play Selection in American Football

Daniel Bestard, Michael Cameron, Hans-Peter Höllwirth, Akhil Lohia

April 4, 2017

1   Motivation

In the following sections we assume some basic understanding of the American football rules. We use a dynamic programming algorithm to develop an optimal policy for play choice in an American football game. In this context, the policies refer to the different plays an American football team can make at each game situation. At every field postition, the aim of the team is to choose among different possible options, such as run, pass, kick and punt, which will be explained in more detail in the next section, the one which increases the expected reward the most. The reward is the difference in number of points recieved at the end of our possesion and the anticipated points of the opposition from our final field position. Each option gives the team different advantages at different risks. Dynamic programming works backwards to allow us, given any field position, to make the optimal policy choice.

In this problem, there is a large number of possible states, 15250, which leads to computational difficulties. To solve this, we use simulations to replace the exact reward-to-go function with an approximation that makes the optimisation problem feasible to solve. Approximate Policy Iteration (API) and Optimistic Policy Iteration (OPI) to calculate this approximation.

2   The Football Model

In our American football model we consider the score of one offense drive and add to it the expected points gained by the opposing team from our final field position. More

concretely, we want to maximize the expected difference between our drive-score and the opposing team's responding drive-score (which is simply a function of our final field position). We now introduce the elements used by the dynamic programming algorithms to solve the optimisation problem.

The state of the system $i \in S$ is described by a vector of 3 quantities: $i = [x, y, d]$:

- x = the number of yards to the opposition goal line (discrete value between 1 and 100)

- y = the number of yards to go until the next first down (discrete value between 1 and 100)

- d = down number ($\in \{1, 2, 3, 4\}$)

At each state, the team can choose from one of 4 play options (actions) $u \in U$ with $U = \{R, P, U, K\}$. Each play is described using probablility distributions, to model the yards gained. For this section we use the models set out in Bertsekas and Patek's article:

- (R)un: moves $D_p - 2$ yards forward with $D_p \sim$ Poisson(6), with probability 0.05 of a fumble.

- (P)ass: moves $D_p - 2$ yards forward with $D_p \sim$ Poisson(12), with probability 0.05 of an interception, 0.45 of incompletion, 0.05 of quarterback sack and 0.45 of completion.

- P(u)nt: moves $6D_p + 6$ yards forward with $D_p \sim$ Poisson(10)

- (K)ick: successful with probability max$(0, .95 - .95x/60)$

The set of state-action pairs determine the stationary policy μ. We look to choose the policy at any given state to maximise the expected reward. The reward of the drive is determined by the final state transition.:

- Touchdown: 6.8 points (from run or pass)

- Opponent's touchdown: -6.8 points (from run or pass)

- Safety: -2.0 points (from pass or run)

- Field goal: 3.0 points (from kick)

- No score due to fumble (from run), interception (from pass), missed 4th down (from pass or run), missed field-goal (from kick) or punt

Once the inputs of the dynamic programming algorithm have been defined, the next step is construct the algorithm itself.

3 Dynamic Programming Formulation

In this model of the football problem, the solution can be computed directly. However, the problem can quickly become computationally infeasible when introducing other elements of the game, such as time or half yard intervals.

3.1 Optimal Solution

Our original problem has 15250 states. The rules of our model are such that at first down there is a unique value of y associated with each value of x. Note, that the number of possible combinations is much larger than the 15250 that we claim. There is the additional constraint that it is impossible to have the number of yards to next down greater than yards to the goal line, that is, we cannot have $y > x$.

This is computationally tractable, so we can apply the dynamic programming algorithm.

$$\mu^k(i) = arg\max_{u \in U}\left[\sum_{j \in S} p_{ij}(u)(g(i,u,j) + J^{\mu^{k-1}}(j))\right] \quad \forall i \in S$$

This algorithm chooses the policy u which maximises the expected reward. In the formula above, i represents the state we are currently in and j is the state we move to. $p_{ij}(u)$ are the transition probabilities, giving the probability from moving from state i to state j under action u. The reward function is $g(i,u,j)$, which is 0 in every stage except for the terminal state. This is because we only gain rewards after scoring or losing the ball. While $J^{\mu^{k-1}}(j)$ is the reward-to-go function from the state j.

Using this exact method we obtain an optimal policy for any state of the 4 downs. These optimal policies can been in Figure 1,1 in Bertsekas and Patek. As a benchmark, using this policy from the state $(x,y,d) = (80,10,1)$ and running simulations with the optimal policy, the expected net reward is -0.9449 points. This means that if we start each drive in this state, we are expected to lose the game!