Predicting Game Play Direction in Football Videos

Amit Bawaskar
School of EECS
Oregon State University
bawaskar@onid.orst.edu

Michael Lam
School of EECS
Oregon State University
lamm@onid.orst.edu

Abstract

The aim of this project is to make a machine learning system which will detect the direction in which the offensive team is playing in football videos. This information is necessary when we have to make automated computer vision systems which will provide us with analysis of the footabll games for coaching purposes. To develop such a system, we have used the KLT Tracking system to track seemingly interesting points in the videos, and then use machine learning algorithms of Decision Tree and Decision Stump with Boosting to identify which player tracks are important in making the decision for the direction of the offensive play.

1 Introduction

To come up with good and effective solutions for football coaching software, we require to track players, look for prominently used strategies by different teams, etc. In general we need to find all those properties of a football video that the coach can use to better train his own team, given a set of videos of the opposition team. To do so, in the absence of such a coaching software, the coach or his team of assistants have to look at numerous plays of the team and decipher the playing strategy of the teams. Even after the coach is done doing this, he now has to send out a great deal of documentation to the players so that they know their roles in the game. This leads to a lot of time spent in auxiliary work which could have been automated.

The aim of this project is to help develop a part of such a coaching software. This project will aim at recognizing the direction of the offensive play given a video. This analysis can then be used to make comments about the team that is on offense, the orientation of the players on the offensive team and the player movements in both the teams. It will serve as a major step in developing the coaching software as many other elements of the coaching software are dependent on it.

2 Methodology

In order to recognize the direction of offense we need to know some domain knowledge about football. We will provide with a brief overview of the domain knowledge that we require for this project. Usually when the two teams line up for a play in the game of football, we have two players known as "Wide Receivers" standing at the ends of the field along the line of scrimmage. The line of scrimmage is the line that separates the two teams at the moment of snap. The moment of snap is the time at which the players start to play the game for each play.

These two wide receivers are instrumental in the detection of the direction of the play as they run fast down the field in the direction of offense from the moment of snap and can be used to determine the direction of the play. If we are able to get good, long tracks on these players, and if we are able to tell that these tracks belong to the wide receivers, we can find the direction of those tracks and ultimately determine the offensive direction of play. Figure 1 shows an example of video footage

Figure 1: An example sequence of a particular football play video footage. Note the wide receiver (bottommost player in white) dashes off to the left. The goal is to classify this video as "left" by tracking the wide receiver.

Figure 2: Another example sequence of a particular football play shown with KLT tracks extracted. Green tracks denote left direction and red tracks denote right direction. Faster moving objects have longer tracks. Using domain knowledge, we can make a good guess for which tracks in a video belong to the wide receiver. Note the wide receiver (bottommost player in black) dashes off to the left. The green KLT tracks on this player get longer and are near to the edge of the field.

where the wide receiver runs fast down the field in the direction of offense, which is to the left of the video frame.

We decided to use KLT tracks [1,2,3] to do the video tracking on different points in the video. The KLT tracking method selects points that have a high gradient with respect to its neighbors. This makes it select "interesting" points such as corners in the video frame. We want to use these interesting points and specify which of these points correspond to the wide receiver on the field. Using KLT tracks, we can get the length of the track easily by simple geometric computation on the track features. Also, we can get the distance of the track from the bottom edge of the frame. Figure 2 visualizes these KLT tracks.

We use KLT tracks to generate a dataset for the videos with the following features:

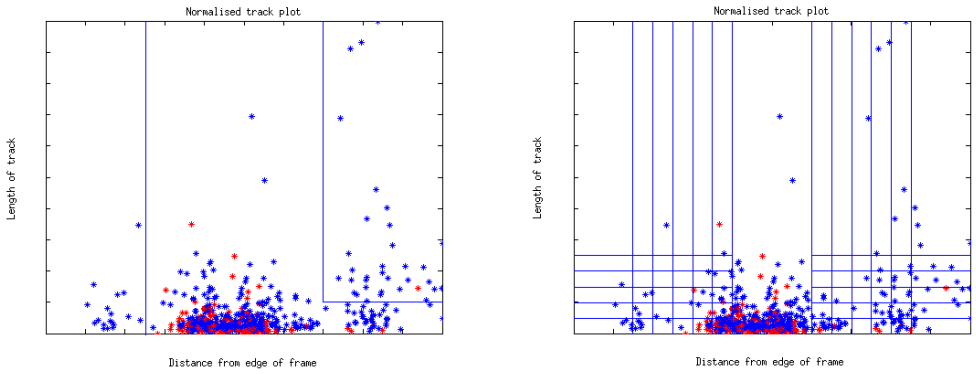
1. Track Number
2. Starting Point Co-ordinates of the track
3. Ending Point Co-ordinates of the track
4. Normalized length of the track.
5. Normalized distance of the track from the bottom edge of the figure.
6. Direction of the track.

After getting such a dataset from the KLT tracks, we have to find out a way to determine which of these tracks belong to the wide receiver. We can now use the domain knowledge to determine where to look for the receivers. They will be at the near end and the far end of the field. Also, since they run fast, they will have longer tracks as compared to anyone else on the field. Hence, we have to look for long tracks on both the ends of the field.

We can plot a sample dataset as shown in figure 3a.

Here the y-direction represents the length of the track while the x direction represents the closeness to the bottom edge of the field. If $x = 1$, then the track is nearest to the bottom. If $x = 0$ then it is farthest from the bottom.

Now, we have to find a region in this plot where the receiver tracks will be. Using the domain knowledge and the inspection of various videos, we come to know that the receiver tracks will more often than not be in the rectangles in figure 3a. The rectangle on the left part of the image search for



(a) Sample dataset

(b) 60 rectangles

Figure 3: (a) Plot of a sample video dataset where the x represents the distance to the bottom edge of the field and the y-direction represents the length of the track. Red data points are left tracks and blue data points are right tracks. The blue rectangles denote where the wide receiver tracks are located. The left rectangle is the wide receiver on the far part of the image and the right rectangle is the near side receiver. (b) Construction of the 60 rectangles. They are biased toward the left and right sides of the plot to capture the far end and near end wide receivers. For each rectangle, its final feature value for the machine learning algorithm is computed by counting the left and right points that fall within that rectangle.

the wide receiver on the far part of the image while that on the right part of the plot search for the near side receiver. However, these regions will change from video to video as the orientation of the videos, the resolution quality and even plays will be different in different situations. Here is where we use the machine learning algorithms to determine the regions to be considered for determining the direction of play.

Note that normally, we do not get good tracks on the far receiver because of occlusion of the player and distortion of the image since the object is far away. So in most situations searching for the far side receiver is futile. But, in situations where there is no near side receiver the system would fail and give arbitrary results. To avoid this we decided to try capturing both the near side and far side receiver.

The goal then is to identify the best rectangles which will denote the direction of play. We predict the direction of play by considering a majority vote of all the tracks that fall within the rectangles. If the majority are moving in the left direction then we say that the direction of the play is left.

3 Features

The features that we have considered here are as follows:

Length of the track.
We know from the domain knowledge that the wide player will be the one who will make a big rush down field so that we will get good, continuous and long tracks on him. Also, the other players, like free safety, quarter back or the guard players will not move much in the first one and a half second and will have relatively smaller tracks. So, we wish to select those tracks that have a high length as feature.

Distance from the edge of the field.
As mentioned before we know that the wide players will be on the near side of the field. The orientation of the play recording is always such that the wide player that we want to track will always be on the bottom most part of the frame. So, we have selected the distance from the edge of the field as our next feature.

Track direction with respect to the field.
This information is vital, since one player will have multiple tracks in one and a half second

which will all be close by and going along the same direction. So, if we include the information of the direction of the track (left or right) we will be able to make the differentiation between the noisy tracks and actual wide player tracks better. Thus, the reason for selecting this as a feature.

In short, we want to track those points that have a high length and are closer to the edge of the field, moving in one direction so that we can say that the tracked point is a wide receiver. Again, figure 2 summarizes these features' relevance.

4 Learning

Recall that the goal is to learn the best rectangles in the plot where the y-direction represents the length of the track while the x direction represents the closeness to the bottom edge of the field. A majority vote of track directions in this rectangle will denote the direction of play.

Our unique strategy is to use many different rectangles as features for our learning algorithm and learn a linear combination of those features. First we construct many different rectangles with various sizes and positions in the plot. Figure 3b shows this construction. These rectangles are fixed across every video. For each video, we count the number of left and right tracks falling in each rectangle and produce a measure of "leftness" and "rightness" for that rectangle. Specifically, for a particular rectangle, we add up all left track counts falling in that rectangle, subtract all right track counts and normalize by the number of tracks; this produces a "net normalized direction count."

In the end, we use 60 different features (rectangles) for each video to submit to our learning algorithms. These 60 features are "predictions" of the 60 different rectangles. These 60 features will be our machine learning features to input into a learning algorithm. Since each video is also manually annotated as left or right offensive direction, we can use a supervised machine learning algorithm to learn a hypothesis that predicts the offensive direction given a video.

Our strategy naturally lends to using boosting since we want to learn a linear combination of weak classifiers. We decided to use AdaBoost [4] as our meta-learning algorithm with decision stumps as the base learner. Our weak classifiers are the decision stumps operating on one feature, which means a weak classifier is some rectangle in the plot with a threshold of the "net normalized direction count." Each feature would be given a weight for the final prediction, which comes from the AdaBoost algorithm. Therefore we can interpret the task as learning the "weights" of each rectangle for making a final prediction as a weighted linear combination of the 60 rectangles. What follows is that the "best rectangle" for prediction we are seeking is essentially some linear combination of the 60 rectangles.

We also decided to use decision trees [5] to compare our results with AdaBoost since it is tree-based. We used the C4.5 algorithm for generating the decision tree based on information gain. Since our features are continuous, we also threshold each feature and select the threshold that maximizes the information gain, before selecting the maximum information gain from all the features. Note that the decision stump is simply a one-level version of the decision tree used in the AdaBoost meta-learning algorithm.

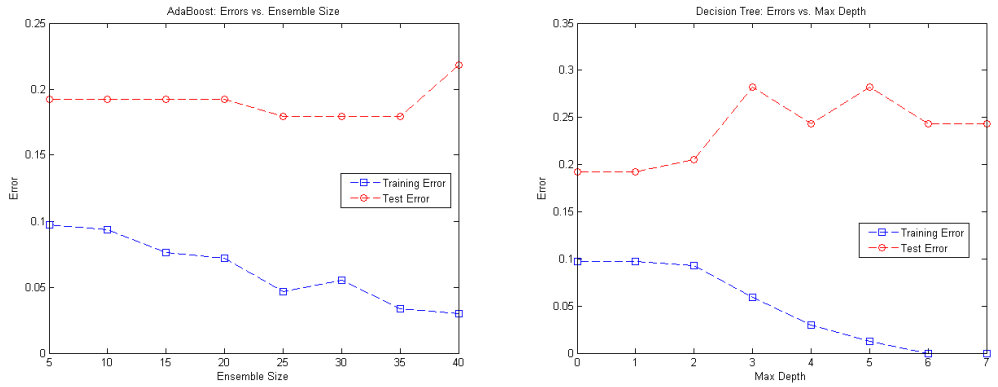
While the hypothesis generated from the decision tree makes sense for capturing decisions on a subset of the 60 rectangles, it is not immediately clear when to stop growing the tree. We investigate this as a parameter in our experiments.

5 Experiments

We describe the experimental setup and results.

5.1 Setup

Our football dataset consists of three games, which we will call game1, game2 and game3. Game1 consists of 95 videos, game2 consists of 95 videos, and game3 consists of 124 videos. Each video represents a play that contains the moment of snap.



(a) AdaBoost

(b) Decision Tree

Figure 4: Training and test errors for AdaBoost and decision tree. (a) AdaBoost training and test errors are a function of ensemble size. (b) Decision tree training and test errors are a function of maximum depth size.

To form our training and test sets for evaluation, we concatenated all three game datasets into one dataset of 314 instances. Each instance corresponds to a video with the 60 features described earlier and its true label (i.e. left or right direction of offensive gameplay). We randomly selected 75% of the instances for our training set and the remaining 25% as our test set.

For empirical evaluation, we compared the performance of using AdaBoost and decision tree on varying parameters. For AdaBoost, we vary the ensemble size as the parameter. For the decision tree, we vary the maximum depth size as the parameter; the tree is only allowed to grow up to the maximum depth.

5.2 Results

Figure 4a presents the plot of training and test errors as a function of ensemble size. We see that the training error decreases reasonably. We also observe that the test error decreases until ensemble size 35. The test error at that point is a little under 20%, which is reasonable.

Figure 4b presents the plot of training and test errors as a function of the maximum decision tree depth. We also see that the training error decreases reasonably. In fact it goes to zero at around a maximum depth of 6. At that point the test error is not good compared to smaller maximum depth sizes, which indicates that the decision tree is over-fitting the data. We also observe that the test error begins to increase early on. The best maximum depth is probably around 2. The test error is a little under 20%, which is reasonable and comparable to the error from AdaBoost.

The overall classifier that we have predicts the correct output most of the times. Our best result is the AdaBoost on ensemble size 35. There we predicted 64 out of 78 test videos correctly, an 82% generalization accuracy, which means our system is usable. The reason for failure in the other cases can be when the assumptions that we have made in developing the system are incorrect.

For example, we consider one and a half second of video footage from the moment of snap. During this short interval, it is expected that the camera covers all of the “interesting” area of the field like the quarterback, free safety, wide runners, along with the line of scrimmage. If some of these features, especially the wide receivers are missing, then the system would not work as efficiently as it should. We have come across videos where there are no wide receivers and since the system heavily relies on these players, it would be very hard to get the correct results.

Also, if there is camera motion during this time (though highly unlikely), then the tracks would give us inaccurate lengths and directions of the play. Negation of camera motion is a hard problem which the computer vision community face even today, and in such circumstances the results would be bad.

6 Conclusion

In this paper, we have motivated the problem of predicting the direction of the offensive play in a video. We have presented a framework for extracting useful features from a video and using them with machine learning to make predictions. Our results using boosting and decision trees yield promising accurate predictions.

Acknowledgments

We would like to thank Dr. Alan Fern and Dr. Sinisa Todorovic for motivating and formulating the problem of game play direction prediction by tracking wide receivers.

References

[1] Bruce D. Lucas and Takeo Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision.* International Joint Conference on Artificial Intelligence, pages 674-679, 1981.

[2] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features.* Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[3] Jianbo Shi and Carlo Tomasi. *Good Features to Track.* IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, 1994.

[4] Yoav Freund and Robert E Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.* Proceedings of the Second European Conference on Computational Learning Theory, pages 23-37, 1995.

[5] J. Ross Quinlan. *Improved Use of Continuous Attributes in C4.5.* Journal of Artificial Intelligence Research, 4:77-90, 1996.