A technique for estimating player style networks and clusters

Fabian Schaipp

1 Introduction

In sports media players, especially young and arising talents, are often compared to a famous player on their position in order to categorize their playing style. However, these relations are mostly based on very superficial associations and characteristics like body type or origin. Obviously, these narratives are not sufficient in order to assess tactical characteristics, strengths and weaknesses of these players. The main objective of this article is to propose a new approach of assessing playing styles fully based on statistics and data.

There are already many concepts of analyzing the playing style as well as the strengths of a specific player. One very common way is to use radar charts with certain pre-selected categories (e.g. from StatsBomb). However, apart from the known visualization issues of radar charts (see [Knutson, 2017]) this only gives information about one specific player which makes it hard to draw conclusions for a large number of players as well as find similar players without a priori knowledge.

Hence, one main aspect of our approach is to explore connections in playing styles over a big amount of players. This is made possible by computing nearest neighbors of players in terms of playing styles and visualizing it via a graph (in the following also called network).

2 Method

2.1 Estimating the player network

Notation: We use the notation $[N] := \{i \in \mathbb{N} : 1 \leq i \leq N\}$.

Consider that we have given a set of players $i = 1, \dots, N$ and a set of statistics $j = 1, \dots, M$ with values for each player. This data is given in the form of a matrix

$$S := \begin{bmatrix} s_{1,1} & \dots & s_{1,M} \\ \vdots & & \\ s_{N,1} & \dots & s_{N,M} \end{bmatrix} \in \mathbb{R}^{N \times M}$$

Furthermore, consider the columns of S - denoted by $s_j \in \mathbb{R}^N$ - to be standardized, i.e. zero-mean and unit-variance.

Why do we assume the columns to be standardized? Obviously, statistics like "number of interceptions" and "pass success rate" will have different scales. In order to make them comparable we have to standardize the given data as a first step.

Remark: If $\tilde{s}_j$ are non-standardized statistics, standardization is simply done by

$$s_j = \frac{1}{\sqrt{v_j}}(\tilde{s}_j - m_j)$$

where $m_j := \frac{1}{N}\sum_{i=1}^{N} s_{j,i}$ and $v_j := \frac{1}{N}\sum_{i=1}^{N}(s_{j,i} - m_j)^2$ are the empirical mean and variance.

Note that for player i we can interprete the i-th row of S - denoted by $p_i \in \mathbb{R}^M$ - as a vector encoding the player's playing style.

The main step is now to compute a distance matrix $D \in \mathbb{R}^{N\times N}$ where $D_{i,j} := d(p_i, p_j)$.
Here, $d(\cdot,\cdot)$ denotes a metric like - for example - the standard euclidean metric.

As a last step we define a graph $\mathscr{G} := ([N], E)$ with the players as nodes and edges

$$(i_1, i_2) \in E \iff i_1 \text{ is one of the k-nearest neighbors of } i_2 \text{ or vice versa} \quad (i_1 \neq i_2).$$

Here i_1 being k-nearest neighbor of i_2 means that $d(p_{i_1}, p_{i_2}) \leq d(p_{i_3}, p_{i_2})$ for all but (at most) k many players $i_3 \in [N]\backslash\{i_2\}$.

As an extension, we can assign weights to the edges via the weight function

$$\phi : E \mapsto R_0^+, \quad \phi((i_1, i_2)) = d(p_{i_1}, p_{i_2})$$

As D is symmetric this is well-defined, i.e. it does not matter for an edge weight which players was nearest neighbor of the respective other.

Alltogether, we obtain a weighted graph $\mathscr{G}$ in which two players are connected by an edge if one of them is very similar in playing style to the other - relative to the sample of players. Obviously, other edge rules are possible: one alternative would be that each edge fullfilling a distance threshold is contained in the graph, i.e.

$$(i_1, i_2) \in E \iff d(p_{i_1}, p_{i_2}) \leq \gamma$$

for a chosen paramter $\gamma \in \mathbb{R}^+$.

2.2 Choice of the metric

An essential part of the computation of the player network is the choice of the metric $d(\cdot,\cdot)$. The most obvious choice would be to use the euclidean metric $d(x,y) = \|x-y\|_2$. The problem with the euclidean metric is that it assigns big distances to a pair of players that may be very similar in style but different in quality. We will make this clearer by an example:

Imagine a pair of players $i_1, i_2 \in [N]$ with $p_{i_2} = \eta\, p_{i_1}$ with $\eta \in [0,1]$. In that case player i_2 - interpreted as a point in $\mathbb{R}^M$ - has exactly the same angle as i_1 but with smaller radius. If we interpret each dimension as a playing style characteristic, we come to the conclusion that the two players i_1 and i_2 are very similar in style, however for p_1 those characteristics are more pronounced.
If we are interested in measuring more the style and less the quality of a player the euclidean metric is not the best choice as it also encodes the quality. On the other hand, the cosine distance which is given by

$$d_{\cos}(x,y) := 1 - \frac{x \circ y}{\|x\|\,\|y\|}$$

where $\circ$ is the standard scalar product and $\|\cdot\|$ its respective norm. However, we should mention that $d_{\cos}(\cdot,\cdot)$ is not a metric in the formal sense.
For our example above the cosine distance would result in

$$d_{\cos}(p_{i_2}, p_{i_1}) = 0.$$

Hence, the cosine distance can be zero for non-identical players. Indeed it is zero if and only if the players are located on the same ray from the origin, like in our example (which follows from Cauchy-Schwarz).
We used the cosine distance for the analysis below, however other choices of metric are also reasonable: for example a weighted absolute differences metric, i.e.

$$d(x,y) = \sum_{i=1}^{M} w_i |x_i - y_i|, \quad w_i \geq 0, \sum_{i=1}^{M} w_i = 1$$

is easily implementable and opens the possibility to weight the used statistics in distinct magnitudes.

2.3 Extension: player clustering

The described method can be combined with a clustering strategy for the different player types. A clustering can be useful for exploring the different styles that are dominant in the data set as well as for visualization purposes: for a large number of players it is hard to show the whole network with node labels. However, using clusters you can easily show subgraphs that contain only one cluster.

In our case we face a task of unsupervised clustering as there are no pre-labeled player types in the data set. There are many methods applicable to such problems and many of them are easy to implement using for example Python's scikit-learn (see [Pedregosa et al., 2011]).
However, we want to use a clustering method which is consistent with the computation of the player network before. As the network mainly depends on the choice of metric $d(\cdot,\cdot)$, it seems natural to choose a clustering method which only uses the - already precomputed - distance matrix D. For this reason, we used DBSCAN [Ester et al., 1996] for clustering.

DBSCAN only needs a distance matrix as input and uses two parameters, namely *minPts* and ε: a point is then called a core point if it has at least *minPts* other points in an ε-neighborhood.
Points are called reachable from a core point, if there is a path of points all lying in ε-neighborhoods to its precursor.

Points that are not reachable from any core points are called noise. Every cluster is now defined by a core point and all points (core or not) reachable from the core point.

For the choice of paramaters several rules of thumb are available: it is proposed to choose *minPts* to be approximately $2 \cdot M$, where M was the number of statistics. The paramater ε subsequently can be found by computing for each point the distance to the $minPts - 1$-nearest neighbor, ordering from the largest to the smallest value and choosing the value around the region on the y-axis, where the plot shows a "knee" or "elbow" ([Sander et al., 1998] , [Schubert et al., 2017]). The authors of [Schubert et al., 2017] write that smaller values for ε typically work better.

References

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases

with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

[Knutson, 2017] Knutson, T. (2017). Revisiting radars. `https://statsbomb.com/2017/05/revisiting-radars/`. [Online; accessed 21-July-2019].

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Sander et al., 1998] Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194.

[Schubert et al., 2017] Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3):19:1–19:21.