MIT SLOAN SPORTS ANALYTICS CONFERENCE presented by ESPN

`(batter|pitcher)2vec`: Statistic-Free Talent Modeling With Neural Player Embeddings

Michael A. Alcorn[1]
Baseball
5435

Abstract

This paper introduces `(batter|pitcher)2vec`, a neural network algorithm inspired by `word2vec` that learns distributed representations of Major League Baseball players. The representations are discovered through a supervised learning task that attempts to predict the outcome of an at-bat (e.g., strike out, home run) given the context of a specific batter and pitcher. The learned representations qualitatively appear to better reflect baseball intuition than traditional baseball statistics, for example, by grouping together pitchers who rely primarily on pitches with dramatic movement. Further, like `word2vec`, the representations possess intriguing algebraic properties, for example, capturing the fact that Bryce Harper might be considered Mike Trout's left-handed doppelgänger. Lastly, `(batter|pitcher)2vec` is significantly more accurate at modeling future at-bat outcomes for previously unseen matchups than simpler approaches.

1. Introduction

Baseball is notorious for its culture of statistical bookkeeping. The depth and breadth of baseball statistics allows fans and professionals alike to compare players and forecast outcomes with varying degrees of accuracy. Although many traditional baseball statistics can be quite informative and useful, they can also be somewhat arbitrary, and thus may not accurately reflect the true talent of any given player.

The field of Sabermetrics was developed in an effort to address some of the inherent limitations of standard baseball statistics. For example, Wins Above Replacement (WAR) “offers an estimate to answer the question, ‘If this player got injured and their team had to replace them with a freely available minor leaguer or a AAAA player from their bench, how much value would the team be losing?’“ [1]. However, the WAR formula (1) is, itself, somewhat ad hoc, reflecting the intuition of the statistic's designer(s):

$$WAR = \frac{BR + BRR + FR + PA + LA + RR}{RPW} \quad (1)$$

where *BR* is batting runs, *BRR* is base running runs, *FR* is fielding runs, *PA* is a positional adjustment, *LA* is a league adjustment, *RR* is replacement runs, and *RPW* is runs per win.

[1] Personal website: https://sites.google.com/view/michaelaalcorn
Code: https://github.com/airalcorn2/batter-pitcher-2vec

Whereas the WAR statistic uses a combination of conventional baseball statistics to quantify an individual player's impact on his team's overall record, the Player Empirical Comparison and Optimization Test Algorithm (PECOTA) forecasts player performance by identifying a neighborhood of players with historically similar statistics (both traditional and of the Sabermetrics variety) and then performing an age correction [2]. However, the neighborhoods produced by PECOTA are proprietary, which precludes deeper investigation by the community. The proprietary Deserved Run Average (DRA) incorporates contextual information to quantify player value, but is limited in its expressiveness as a strictly linear model [3][4].

When scouts assess the talent of a player, they do not simply count the number of times the player made it to first base or struck out. They consider the player's tendencies in a number of different contexts. For example, a scout might notice that a certain left-handed batter tends to ground out to third base when facing left-handed curveball pitchers. Or that a certain right-handed fastball pitcher tends to struggle against short right-handed batters. An algorithm capable of learning these subtleties would be a valuable tool for assessing player talent.

The task of extracting informative measures of talent for Major League Baseball (MLB) players has a surprising parallel in the field of natural language processing — the task of constructing useful word embeddings. Words, like MLB players, can be considered distinct elements in a set, and one common way to represent such categorical data in machine learning algorithms is as one-hot encodings. A one-hot encoding is an N-dimensional vector (where N is the number of elements in the set) consisting entirely of zeros except for a single one at the location corresponding to the element's index. For example, the one-hot encodings for a vocabulary consisting of the words “dog“, “cat”, and “constitutional“ might be [1 0 0], [0 1 0], and [0 0 1], respectively.

One drawback of one-hot encodings is that each vector in the set is orthogonal to and equidistant from every other vector in the set; i.e., every element in the set is equally similar (or dissimilar) to every other element in the set. Words, however, *do* exhibit varying degrees of semantic similarity. For example, the words “dog“ and “cat“ are clearly more semantically similar than the words “dog“ and “constitutional”. Word embedding algorithms learn to mathematically encode such similarities as geometric relationships between vectors (e.g., cosine similarity or Euclidean distance).

Perhaps the best-known word embedding algorithm is `word2vec` [5], a neural network that learns distributed representations of words in a supervised setting. The `word2vec` algorithm can take two different forms: ***(1)*** the continuous bag-of-words (CBOW) model or ***(2)*** the skip-gram model (see **Figure 1**). The CBOW model attempts to predict a word (w_t) given some surrounding words (e.g., w_{t-1} and w_{t+1}, although the context can be larger). In contrast, the skip-gram model attempts to predict the surrounding words of a given central word. Incorporating pre-trained word embeddings into models built for other natural language processing tasks often leads to improved performance (an example of “transfer learning”) [6].

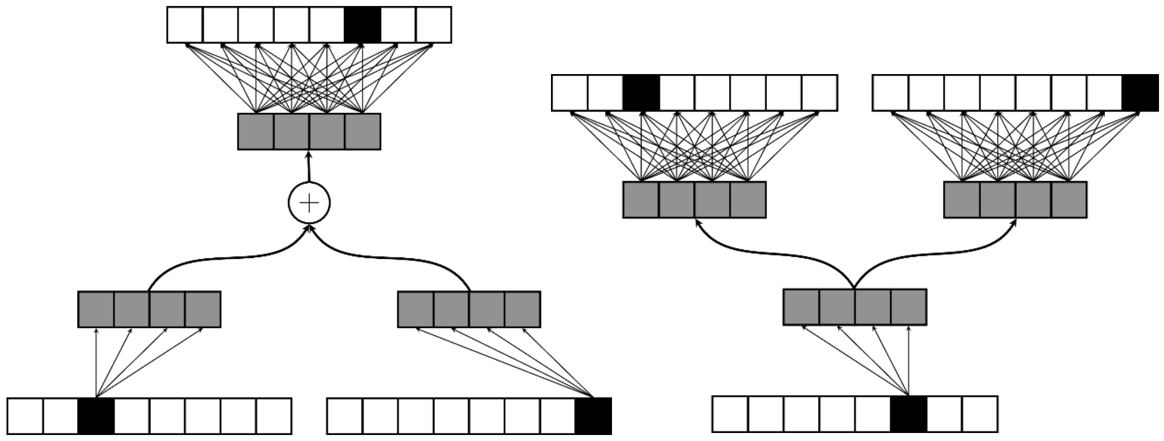

Figure 1. The CBOW (left) and skip-gram (right) model architectures for a window of three words. A one-hot vector is depicted as a black square among an array of white squares while an embedding is depicted as an array of gray squares.

This paper introduces `(batter|pitcher)2vec`, a neural network algorithm that adapts representation learning concepts (like those found in `word2vec`) to a baseball setting, modeling player talent by learning to predict the outcome of an at-bat given the context of a specific batter and pitcher. Unlike many Sabermetrics statistics, `(batter|pitcher)2vec` learns from "raw" baseball events as opposed to aggregate statistics, which allows it to incorporate additional contextual information when modeling player talent.

Previous work modeling player "style" in other sports has ignored the context in which events occur, and has relied on aggregate statistics for data. [7] applied latent Dirichlet allocation to aggregate statistics of mixed martial arts finishing moves to discover factors describing fighter style. [8] used dictionary learning to discover tennis shot prototypes, and then aggregated these prototypes to describe player styles. Finally, [9] utilized aggregate location data to characterize hockey player style. Both [10] and [11] described models that predict the probability of different events occurring during basketball games given the players on the court, but the models are strictly linear in the player parameters, which limits their expressiveness. `(batter|pitcher)2vec` marries the expressiveness of latent variable models with the predictive power of contextual models to improve upon each.

2. Methods

2.1. Data

Play-by-play data for each game from the 2013, 2014, 2015, and 2016 seasons were obtained from the Retrosheet website [12]. Each play description was converted into a tuple consisting of the batter, pitcher, and at-bat outcome; for example, (Mike Trout, Anthony Bass, HR), where HR is the symbol denoting a home run. There are 52 potential at-bat outcomes (see **Table 1**), but the model only considers 49 different outcomes because there were no observed instances of doubles first fielded by the catcher, triples first fielded by the catcher, or triples first fielded by the third baseman. The raw training data set consisted of 557,436 at-bats from the 2013, 2014, and 2015 seasons representing 1,634 different batters and 1,226 different pitchers. To ensure an adequate amount of data was available for learning player representations, only the most frequent batters and pitchers involved in 90% of at-bats were included in the analysis. The final data set consisted of 461,231 at-bats with 524 batters and 565 pitchers.

Table 1. The possible at-bat outcomes.

Symbol	Description	Symbol	Description
1-9	Out first handled by fielder #	E1-E9	Error first handled by fielder #
S1-S9	Single first handled by fielder #	K	Strike out
D1-D9	Double first handled by fielder #	W	Walk
DG	Ground rule double	IW	Intentional walk
T1-T9	Triple first handled by fielder #	BK	Balk
HR	Home run	HP	Hit by a pitch

2.2. Model

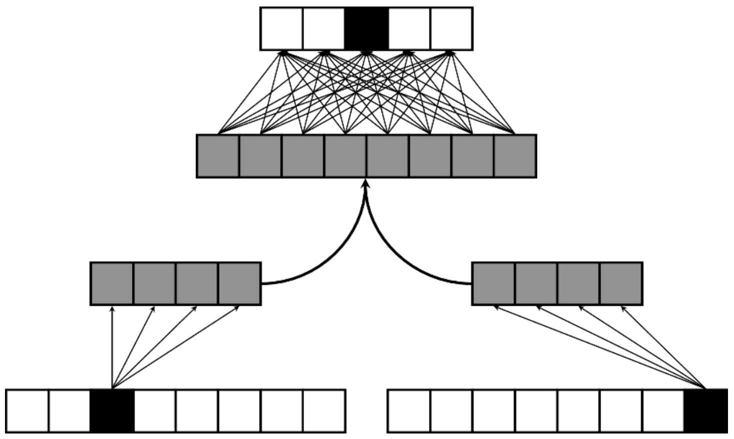

Figure 2. The `(batter|pitcher)2vec` model architecture.

The `(batter|pitcher)2vec` model architecture can be seen in **Figure 2**, and the similarities between `(batter|pitcher)2vec` and `word2vec` should be readily apparent. The model takes one-hot encodings of a batter and pitcher as input and then selects the corresponding player weights from the batter (2) and pitcher (3) weight matrices, respectively. The player weight vectors are then passed through a "sigmoid"/logistic activation function.

$$w_b = \sigma(W_b \cdot h_b) \tag{2}$$

$$w_p = \sigma(W_p \cdot h_p) \tag{3}$$

Here, h_b is the N_B-dimensional one-hot vector (where N_B is the number of batters) for the batter indexed by b, W_B is the batter embedding matrix, σ is the logistic activation function (i.e., $\frac{1}{1+e^{-x}}$) and w_b is the batter's embedding. Likewise, h_p is the N_P-dimensional one-hot vector for the pitcher indexed by p, W_P is the pitcher embedding matrix, and w_p is the pitcher's embedding.

The batter and pitcher embeddings are then concatenated together (4) and fed into a standard softmax layer (5) and (6), which outputs a probability distribution over at-bat outcomes.

$$w_{b \oplus p} = w_b \oplus w_p \tag{4}$$

$$z = W_o \cdot w_{b \oplus p} + b_o \tag{5}$$

$$p\left(o_i \middle| h_b, h_p\right) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

Here, o_i is the at-bat outcome indexed by i and K is the number of possible at-bat outcomes.

Maximizing the likelihood of this model is equivalent to minimizing the model's average cross entropy (7), which is the standard loss function used in machine learning classification tasks:

$$\mathcal{L}(D) = \frac{1}{N}\sum_{i=1}^{N} H(p_i, q_i) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} p_i(o_j)\log(q_i(o_j)) \tag{7}$$

where D is the training data, N is the number of training samples, $H(p_i,q_i)$ is the cross entropy between the probability distributions p_i and q_i, p_i is the true at-bat outcome distribution for training sample i, and q_i is the predicted outcome distribution for training sample i.

The model was implemented in Keras 2.0 [13] and trained on a laptop with 16 gigabytes of RAM and an Intel i7 CPU. The model was trained for 100 epochs using Nesterov accelerated mini-batch (100 samples/mini-batch) gradient descent with the learning rate, momentum, and decay hyperparameters set at 0.01, 0.9, and 10^{-6}, respectively. Both batters and pitchers were embedded into a nine-dimensional space. The code to generate the results described in this paper can be found at https://github.com/airalcorn2/batter-pitcher-2vec.

3. Results

3.1. Visual inspection of the player embeddings

Visually inspecting low-dimensional approximations of neural network representations can often provide some intuition for what the model learned. Several plots of the first two principal components of the batter embeddings can be seen in **Figure 3**. A number of trends are readily apparent; for example, left-handed hitters are clearly distinguishable from right-handed hitters, and batters with high single rates are antipodal to batters with low single rates (a similar pattern is visible for home run rates). At the least, `(batter|pitcher)2vec` appears to be capable of capturing the information contained in standard baseball statistics.

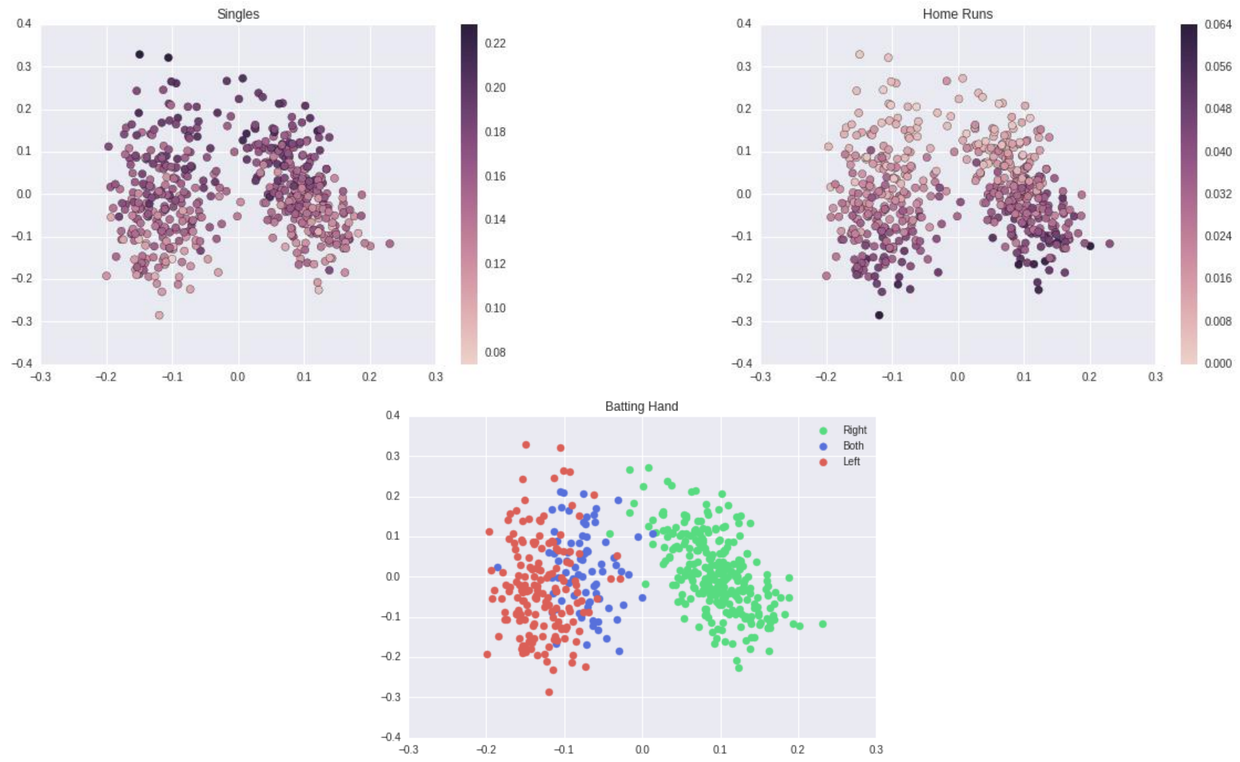

Figure 3. Plots of the first two principal components of the batter embeddings colored by various batter qualities.

3.2. Nearest neighbors

Probing the neighborhoods of individual embeddings can also yield deeper insights about the model. The t-SNE algorithm [14] was used to map the batter and pitcher embeddings into two dimensions so that they could be visualized (**Figure 4**). Intriguing player clusters are readily apparent, with close pairs including: Mike Trout/Paul Goldschmidt, Dee Gordon/Ichiro Suzuki, and Aroldis Chapman/Dellin Betances.

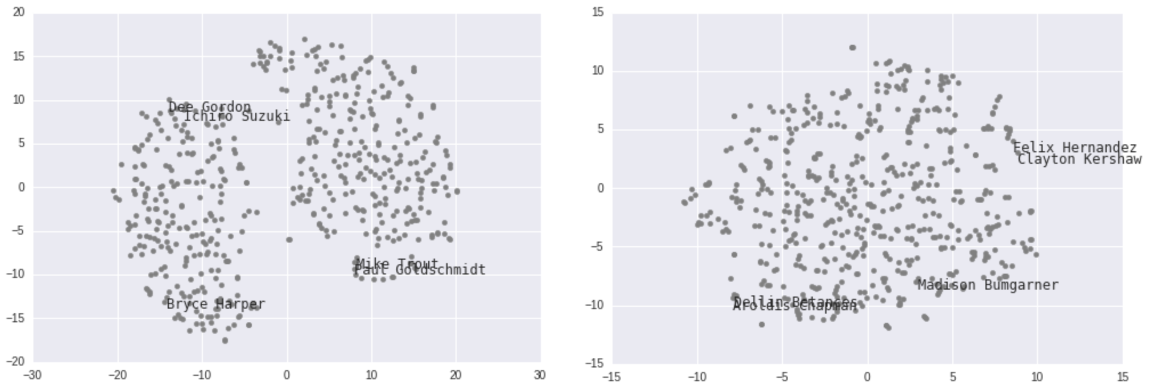

Figure 4. Two-dimensional t-SNE map of the learned batter (top) and pitcher (bottom) representations.

When calculating nearest neighbors in the learned embedding space, Paul Goldschmidt is indeed Mike Trout's nearest neighbor; an unsurprising result considering how each athlete is known for his rare blend of speed and power [15]. Similarly, Ichiro Suzuki is Dee Gordon's nearest neighbor, which is to be expected as both have a reputation for being able to get on base [16]. Notably, when clustering players on common MLB stats (e.g., HRs, RBIs), Paul Goldschmidt is not among Mike Trout's ten nearest neighbors, nor is Ichiro Suzuki among Dee Gordon's ten nearest neighbors.

For pitchers, Craig Kimbrel is Aroldis Chapman's nearest neighbor, which is unsurprising considering both are known as elite closers with overpowering fastballs [17]. Similarly, Félix Hernández, much like his two nearest neighbors, Jean Machi and Carlos Carrasco, is known for having a pitch with incredible movement in his repertoire [18][19][20]. Other nearest neighbors are not as immediately obvious, for example, Clayton Kershaw and Craig Stammen (although, Zack Greinke is Kershaw's second nearest neighbor), but a method like `(batter|pitcher)2vec` could potentially reveal surprising similarities between players who are not considered similar by human standards.

3.3. Opposite-handed doppelgängers

One of the most fascinating properties of effective word embeddings is their analogy capabilities [21]. For example, when using `word2vec` word embeddings, subtracting the vector for "France" from the vector for "Paris" and then adding the vector for "Italy" produces a vector that is very close to the vector for "Rome", which corresponds to the analogy Paris : France :: Rome : Italy [21].

To investigate the algebraic properties of the `(batter|pitcher)2vec` representations, the average batter vector was calculated for both left-handed and right-handed hitters and then subtracted from select players in an attempt to generate opposite-handed doppelgängers. For example, subtracting the average left-handed batter vector from the vector for Mike Trout (a right-handed batter) produces a vector with Chris Davis, David Ortiz, and Bryce Harper (all powerful, left-handed batters) as the three nearest neighbors, which is suggestive of a valid batter algebra (see [22] for a discussion of the similarities between Mike Trout and Bryce Harper). Similarly, subtracting the average left-handed batter vector from the vector for Dee Gordon yields a vector that is very close to the vector for Tyler Saladino, a fitting candidate for Gordon's opposite-handed doppelgänger [23].

3.4. Modeling previously unseen at-bat matchups

The representations learned by neural networks are theoretically interesting because they suggest the neural networks are discovering causal processes when the models are able to generalize (or transfer) well [24]. In the case of `(batter|pitcher)2vec`, the ability to accurately model at-bat outcome probability distributions for previously unseen batter/pitcher pairs would indicate the neural network was extracting important aspects of baseball talent during learning. To test this hypothesis, at-bat outcomes were collected from the 2016 season for previously unseen matchups that included batters and pitchers from the training set. In all, there were 21,479 previously unseen matchups corresponding to 51,580 at-bats.

Table 2. Average cross entropy for a naïve strategy, multinomial logistic regression, and `(batter|pitcher)2vec` on previously unseen batter/pitcher matchups.

Model	**Average Cross Entropy**
Naïve	2.8113
Multinomial Logistic Regression	2.8118
`(batter|pitcher)2vec`	2.7848

A naïve strategy was used as a baseline for performance comparisons. For any given batter, the probability that an at-bat would result in a specific outcome was defined as:

$$p(o_i|b_j) = \frac{c_{i,j} + r_i}{\sum_{k=1}^{K} c_{j,k} + 1} \tag{8}$$

where o_i denotes the outcome indexed by i, b_j represents the batter indexed by j, $c_{i,j}$ is the number of times the batter indexed by j had an at-bat resulting in the outcome indexed by i in the training data, r_i is the proportion of all at-bats that resulted in the outcome indexed by i in the training data, and K is the number of outcomes. Essentially, the procedure adds one at-bat to each batter's data, but distributes the probability mass of that single at-bat across all possible outcomes based on data from all batters. r_i can thus be considered a type of "prior" or smoothing factor. $p(o_i|t_j)$ was similarly defined for pitchers. The naïve expected outcome distribution for a given batter, b_j, and pitcher, t_k, matchup is thus defined as:

$$p(o_i|b_j, t_k) = \frac{p(o_i|b_j) + p(o_i|t_k)}{2} \tag{9}$$

The cross entropy was calculated for each at-bat in the test set using both the naïve approach and `(batter|pitcher)2vec`. The naïve approach produced an average cross entropy of 2.8113 on the test set while `(batter|pitcher)2vec` produced a significantly ($p < 0.001$) lower average cross entropy of 2.7848, a 0.94% improvement (**Table 2**). For comparison, a multinomial logistic regression model (**Figure 5**) trained and tested on identical data sets produced an average cross entropy of 2.8118, which is slightly worse than the naïve approach (**Table 2**). `(batter|pitcher)2vec` thus appears to be exceptional in its ability to model at-bat outcome distributions for previously unseen matchups.

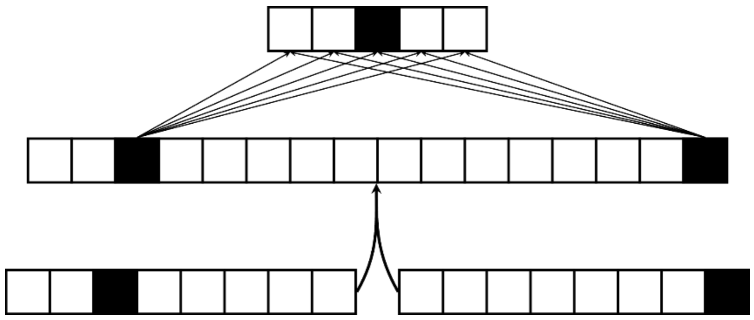
Figure 5. A graphical model depiction of multinomial logistic regression.

4. Future directions

These results prove neural embedding algorithms offer a principled means of modeling talent from "raw data", i.e., without resorting to ad hoc statistics. Just as pre-trained word embeddings can be used to improve the performance of models in various natural language processing tasks, player embeddings could be used to better inform baseball strategy. For example, by swapping the embeddings of players in a proposed trade and "back simulating" games from earlier in the season, teams would be able to assess how many more wins (or losses) they would have obtained with the candidate player(s) on the roster (effectively establishing a counterfactual). Likewise, after first applying `(batter|pitcher)2vec` to minor league baseball players, a second model could be trained that learns to map a player's minor league representation to his MLB representation. Such a model would allow teams to scout prospects by surveying their neighboring MLB players in the mapped space (this framework is conceptually similar to the multimodal model described in [25], which learns a map between audio and video representations).

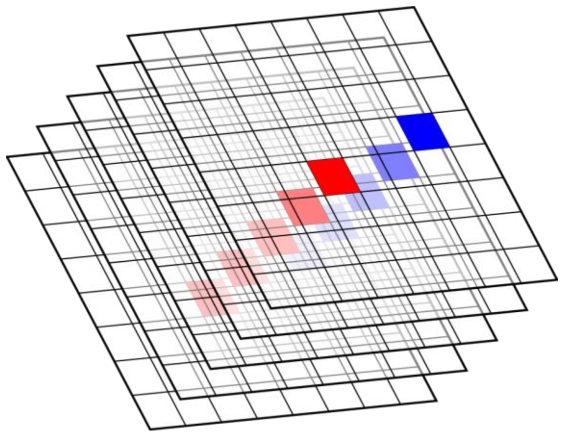

Figure 6. Image-like representations of plays could incorporate both spatial and talent information, with the (x, y) coordinates of the "image" encoding athlete location and the "channels"/embeddings encoding player talent. Here, the red and blue squares (which extend in the z direction) depict two different player embeddings.

Because the neural embedding template is so flexible, it can easily be adapted to suit a variety of inputs and outputs in different sports settings. Modifying `(batter|pitcher)2vec` to predict PITCHf/x measurements as opposed to discrete outcomes could be fruitful as PITCHf/x data would likely convey more information about a player's physical characteristics. `(batter|pitcher)2vec` could also be extended to include the pitcher's supporting defense as one of the model's inputs, thereby providing additional context when predicting at-bat outcomes (mirroring the approach taken in [26] with `doc2vec`. A similar architecture was proposed by [27] to model National Football League offenses and defenses and was used in [28] to capture soccer team "styles". Player embeddings could even make powerful additions to models with more complex spatiotemporal components. For example, [29] represented National Basketball Association plays as a sequence of images where each player was colored according to his position, and a player was assigned to a position based on his nearest neighbors in a latent feature space acquired from an autoencoder trained on aggregate statistics. Rather than using hard position assignments to color players, the "color channels" of the image could contain player embeddings (**Figure 6**), which would enable much richer modeling of play dynamics.

Acknowledgments

I would like to thank my colleague Erik Erlandson for his suggestion to investigate opposite-handed doppelgängers.

References

[1] *What is WAR*? URL: http://www.fangraphs.com/library/misc/war/ (visited on 04/08/2017).
[2] Nate Silver. *Baseball Prospectus Basics: The Science of Forecasting*. 2004. URL: https://www.baseballprospectus.com/news/article/2659/baseball-prospectus-basics-the-science-of-forecasting/ (visited on 04/10/2017).

[3] Jonathan Judge. *DRA: An In-Depth Discussion*. 2015. URL: https://www.baseballprospectus.com/news/article/26196/prospectus-feature-dra-an-in-depth-discussion/ (visited on 10/27/2017).
[4] Dan Turkenkopf, Harry Pavlidis, and Jonathan Judge. *Introducing Deserved Run Average (DRA) And All Its Friends*. 2015. URL: https://www.baseballprospectus.com/news/article/26195/prospectus-feature-introducing-deserved-run-average-draand-all-its-friends/ (visited on 10/27/2017).
[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Neural Information Processing Systems* (2013), pp. 1–9. URL: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.
[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. "A Neural Probabilistic Language Model". In: *The Journal of Machine Learning Research* 3 (2003), pp. 1137–1155. URL: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.
[7] Sean R. Hackett and John D. Storey. "Mixed Membership Martial Arts: Data-Driven Analysis of Winning Martial Arts Styles". In: *MIT Sloan Sports Analytics Conference* (2017), pp. 1–17. URL: http://www.sloansportsconference.com/wp-content/uploads/2017/02/1575.pdf.
[8] Xinyu Wei, Patrick Lucey, Stuart Morgan, Machar Reid, and Sridha Sridharan. ""The Thin Edge of the Wedge": Accurately Predicting Shot Outcomes in Tennis using Style and Context Priors". In: *MIT Sloan Sports Analytics Conference* (2016). URL: http://www.sloansportsconference.com/wp-content/uploads/2016/02/1475-Other-Sport.pdf.
[9] Oliver Schulte and Zeyu Zhao. "Apples-to-Apples: Clustering and Ranking NHL Players Using Location Information and Scoring Impact". In: *MIT Sloan Sports Analytics Conference* (2017). URL: http://www.sloansportsconference.com/wp-content/uploads/2017/02/1625.pdf.
[10] Min-Hwan Oh, Suraj Keshri, and Garud Iyengar. "Graphical Model for Basketball Match Simulation". In: *MIT Sloan Sports Analytics Conference* (2015). URL: http://www.sloansportsconference.com/wp-content/uploads/2015/02/SSAC15-RP-Finalist-Graphical-model-for-basketball-match-simulation.pdf.
[11] Joseph Kuehn. "Accounting for Complementary Skill Sets When Evaluating NBA Players' Values to a Specific Team". In: *MIT Sloan Sports Analytics Conference* (2016). URL: http://www.sloansportsconference.com/wp-content/uploads/2016/02/1425-Basketball.pdf.
[12] Retrosheet Event Files. URL: http://www.retrosheet.org/game.htm (visited on 04/08/2017).
[13] François Chollet. Keras. 2015. URL: https://github.com/fchollet/keras.
[14] L J P van der Maaten and G E Hinton. "Visualizing High-Dimensional Data Using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf.
[15] Matthew Kory. *Paul Goldschmidt Might Really Be This Good*. 2015. URL: https://sports.vice.com/en_us/article/paul-goldschmidt-might-really-be-this-good (visited on 04/19/2017).
[16] Jeff Sullivan. *Dee Gordon Has Been Going Full Ichiro*. 2015. URL: http://www.fangraphs.com/blogs/dee-gordon-has-been-going-full-ichiro/ (visited on 04/19/2017).

[17] Steve Mirsky. "Arms Race". In: *Scientific American* 314.6 (May 2016), pp. 78–78. URL: https://www.scientificamerican.com/article/the-documentary-fastball-tosses-some-physics-at-fans/.
[18] Ben Buchanan. *Do the Red Sox have anything in Jean Machi?* 2015. URL: http://www.overthemonster.com/2015/7/29/9066447/what-is-a-jean-machi-red-sox-giants (visited on 04/24/2017).
[19] Ryan Romano. *Carlos Carrasco's incredible, vexing slider*. 2015. URL: http://www.beyondtheboxscore.com/2015/7/22/9007741/carlos-carrascos-incredible-vexing-slider-indians-mlb (visited on 04/24/2017).
[20] Ted Berg. *The 7 most extreme pitches in Major League Baseball*. 2016. URL: http://ftw.usatoday.com/2016/03/mlb-extreme-pitches-gifs-chapman-felix-sale-jansen-familia (visited on 04/24/2017).
[21] Tomas Mikolov, K Chen, G Corrado, and J Dean. "Efficient Estimation of Word Representations in Vector Space". In: *ArXiv e-prints* (2013). URL: https://arxiv.org/pdf/1301.3781.pdf.
[22] Jesse Spector. *Mike Trout vs. Bryce Harper: The SN50 rivalry that defines this generation of stars*. 2016. URL: http://www.sportingnews.com/mlb/news/sn50-2016-best-baseball-players-miketrout-bryce-harper/mk3kmorbiyhr1f7onb7t5pehq (visited on 04/19/2017).
[23] Alex Chamberlain. *All Aboard the Tyler Saladino Hype Train*. 2017. URL: http://www.fangraphs.com/fantasy/all-aboard-the-tyler-saladino-hype-train/ (visited on 04/19/2017).
[24] Ian Goodfellow, Yoshua Bengio, and Aaaron Courville. "Representation Learning". In: *Deep Learning*. MIT Press, 2016. Chap. 15, pp. 526–557. URL: http://www.deeplearningbook.org/contents/representation.html.
[25] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. "Multimodal Deep Learning". In: *Proceedings of The 28th International Conference on Machine Learning (ICML)* (2011), pp. 689–696. URL: https://people.csail.mit.edu/khosla/papers/icml2011_ngiam.pdf.
[26] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *International Conference on Machine Learning - ICML* 2014 32 (May 2014), pp. 1188–1196. URL: http://arxiv.org/abs/1405.4053.
[27] Michael A. Alcorn. *Learning to Coach Football*. 2016. URL: https://sites.google.com/view/michaelaalcorn/blog/learning-to-coach-football (visited on 04/19/2017).
[28] Hoang M. Le, Peter Carr, Yisong Yue, and Patrick Lucey. "Data-Driven Ghosting Using Deep Imitation Learning". In: *MIT Sloan Sports Analytics Conference* (2017). URL: http://www.sloansportsconference.com/wp-content/uploads/2017/02/1671-2.pdf.
[29] Kuan-Chieh Wang and Richard Zemel. "Classifying NBA Offensive Plays Using Neural Networks". In: *MIT Sloan Sports Analytics Conference* (2016). URL: http://www.sloansportsconference.com/wp-content/uploads/2016/02/1536-Classifying-NBA-Offensive-Plays-Using-Neural-Networks.pdf.