# Predicting Matchups and Preferences in Context

Shuo Chen
Department of Computer Science
Cornell University
Ithaca, NY, USA
shuochen@cs.cornell.edu

Thorsten Joachims
Department of Computer Science
Cornell University
Ithaca, NY, USA
tj@cs.cornell.edu

## ABSTRACT

We present a general probabilistic framework for predicting the outcome of pairwise matchups (e.g. two-player sport matches) and pairwise preferences (e.g. product preferences), both of which have widespread applications ranging from matchmaking in computer games to recommendation in e-commerce. Unlike existing models for these tasks, our model not only learns representations of the items in a more expressive latent vector space, but also models how context modifies matchup and preference outcomes. For example, the context "weather" may alter the winning probability in a tennis match, or the fact that the user is on a mobile device may alter his preferences among restaurants. More generally, the model is capable of handling any symmetric game/comparison problem that can be described by vectorized player/item and game/context features. We provide a comprehensive evaluation of its predictive performance with real datasets from both domains to show its ability to predict preference and game outcomes more accurately than existing models. Furthermore, we demonstrate on synthetic datasets the expressiveness of the model when compared against theoretical limits.

## 1. INTRODUCTION

A wide range of real-world prediction problems require modeling a pairwise relation between a potentially large set of objects. For example, when modeling competitive matchups in sports, the goal is to predict the outcome and the associated winning probability of a game between two players. By learning from historical game records among the players, traditional statistical models and their extensions [42, 7, 27] have been applied to both real-world sports prediction [29] and the matchmaking systems of online competitive video games [18]. Similarly, pairwise-relation models have been used for learning to predict human preferences and decision making. This ranges from purchasing choices people make between pairs of product, to aggregating pairwise preferences for learning the ranking of many items [12, 11]. In the context of search engines especially, it has been shown that treating clicks as revealing pairwise preferences is more reliable than treating clicks as absolute judgments [23].



*KDD '16, August 13 - 17, 2016, San Francisco, CA, USA*



ISBN 978-1-4503-4232-2/16/08...$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939764

What motivates the work in this paper is that matchups and comparisons typically take place in varying contexts that can alter the outcome, and that both contexts and objects can be described by generalizing features. For example, in modeling sports matchups, characteristics of the game (i.e. the context) include weather, the time the game is played at, the importance of the game, the referee, antd the prize money. All these factors can affect the outcome of the game, and they are different from features describing the players (i.e. objects), like age, world ranking, recent game record, whether just returning from injury or not, and playing at home or away. Similarly in the preference domain, imagine the choices between restaurants. Each restaurant could have object features like food it serves, distance from current location, the environment, whether AC or Wi-Fi is installed. However, a customer's particular choice is also affected by context like whether he is hungry or not, lunch or dinner, weekday or weekend [3].

In this paper, we propose a general probabilistic framework for modeling pairwise relations that applies to any problem that can be modeled through object and context features. In particular, we show how the framework applies to accurately modeling and predicting the outcome of any type of game between two players, as well as to modeling human pairwise choices that are affected by context. This problem of learning from contextual pairwise comparisons has not been studied before, and we discuss how it is different from existing works in Section 4. In particular, unlike previous works on learning from pairwise preferences, our model naturally incorporates both features of objects (e.g., players, choices) and features of the context (e.g, game, framing). Furthermore, by building upon a recently introduced choice model that can represent intransitivity, our approach can model inherently intransitive relations that exist in sports matchups, and it can represent apparent intransitivities due to changing contexts in preference relations. The new model is evaluated in both the matchup and the preference domains, showing that it can produce results that surpass the fidelity of conventional models.

## 2. PRELIMINARIES

In this section, we first formally define the learning task. Then we discuss several conventional preference models, showing how they do not adequately address the task. They serve as baselines for empirical investigation later.

From now on, we are going to use the following concepts interchangeably for competitive matchup and pairwise preference modeling: matchup/pairwise preference, player/item, game/context, win/prefer. In this work we focus on modeling matchups between two players, and we assume the result for each individual match cannot be a draw. In the most general setting we are concerned about, a player $a$ encounters player $b$ in a game $g$. At the end of the game,

there can be only one winner. We can assume it is $a$ without loss of generality. In addition, we also have feature vectors that describe the players and the games: $\mathbf{x}_a, \mathbf{x}_b \in \mathbf{R}^{d_p}$ and $\mathbf{z}_g \in \mathbf{R}^{d_g}$. These feature vectors take very general form and can be used to encode any information we have regarding the players and game. Think about a professional tennis game for example. The feature vector of a player could contain: identity, age, nationality, world ranking, whether on a winning/losing streak, etc. These features are different for the two players. On the other hand, the feature vector of a game could contain: weather, surface of the ground, indoor or outdoor match, how deep into the tournament bracket, etc. They are the same for both players, but could affect them differently.

By learning from a training set $D$ that contains multiple matches (triples of $(a, b, g)$ and associated feature vectors), we want to predict the outcome of any future matchup as accurately as possible in terms of the probability $\Pr(a \text{ beats } b|g)$. In the following subsections, we discuss a few conventional methods from the existing literature that can be used for this learning task.

## 2.1 Rote learning

The most straightforward method for addressing the task is estimating each matchup probability $\Pr(a \text{ beats } b)$ individually while ignoring the context. The maximum likelihood estimator gives an intuitive formula

$$\Pr(a \text{ beats } b) = \frac{n_a}{n_a + n_b}, \tag{1}$$

where $n_a$ and $n_b$ are the numbers of times $a$ and $b$ wins respectively among the $n_a + n_b$ matches. The model contains $O(n^2)$ parameters, with $n$ being the total number of players. One can imagine that, given enough data and assuming that context has no influence, it can model any matchup probability arbitrarily accurately. However, in reality not every possible matchup may be played enough times to get a good sample, or even played at all. For example, a comparatively lesser player who usually gets eliminated by a much better player in the first round of a tournament due to seeding, rarely gets a chance to play someone of similar competence. On the other hand, the duel between two big names takes place much more frequently in the later stages of the tournament. Some $n_a$ and $n_b$ could be zero, or even both of them, making it hard to model that matchup accurately. This also gives rise to a negative infinite log-likelihood, which is one of our evaluation measures. To avoid it, we do a simple add-one smoothing [24].

$$\Pr(a \text{ beats } b) = \frac{n_a + 1}{n_a + n_b + 2}, \tag{2}$$

We call this the rote learning method. Note that it cannot make use of any features of players or games except for the player's identity.

## 2.2 Bradley-Terry model

The seminal work of Bradley-Terry model [7, 27] is the basis of many research works in pairwise comparison, which naturally extends to matchup prediction. In the Bradley-Terry model, each player's strength is represented by a single real number $\gamma$, and there are $O(n)$ parameters. The probability of player $a$ beating player $b$ is modeled as

$$\begin{aligned}\Pr(a \text{ beats } b) &= \frac{\exp(\gamma_a)}{\exp(\gamma_a) + \exp(\gamma_b)} \\ &= \frac{1}{1 + \exp(-(\gamma_a - \gamma_b))} \\ &= S(M(a, b)).\end{aligned} \tag{3}$$

Here $S(x) = 1/(1 + \exp(-x))$ is the sigmoid or logistic function. $M(a, b)$ is what we call the matchup function of player $a$ and player $b$, and it simply measures the edge given to player $a$ when matched up against player $b$. In the Bradley-Terry model, it is modeled as $M(a, b) = \gamma_a - \gamma_b$, the difference of strengths between two players. Some properties of the Bradley-Terry model are:

1. The range of $M(a, b)$ is $\mathbb{R}$, with positive/negative meaning player $a$/$b$ has more than $50\%$ chance of winning, and $0$ meaning it is an even matchup.
2. When $M(a, b) \to +\infty$, $\Pr(a \text{ beats } b) \to 1$. Similarly when $M(a, b) \to -\infty$, $\Pr(a \text{ beats } b) \to 0$.
3. $M(a, b) = -M(b, a)$. This makes sure that we always have $\Pr(a \text{ beats } b) = 1 - \Pr(b \text{ beats } a)$ satisfied.

Note that these three properties follow the properties of the sigmoid function. In fact, any real-valued function $M(a, b)$ that takes two players as arguments and satisfies property 3 can be plugged in and gives us a Bradley-Terry-like model.

The learning problem is to figure out the best strength parameter for each player from the training dataset, and is usually done via maximum likelihood estimation. Although the issue with $O(n^2)$ parameters is avoided, the Bradley-Terry model still only makes use of the identity of players, not additional features for players and games.

## 2.3 Pairwise logistic regression model

There is one way to extend the Bradley-Terry model to incorporate additional player information. One can model each player's strength as a weighted sum

$$\gamma_a = \mathbf{w}^T \mathbf{x}_a. \tag{4}$$

Then the matchup function becomes $M(a, b) = \mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b)$. The model can be interpreted as a logistic regression [6]: the input is the difference of the two player's feature vectors, and the output is 1/0 for the first player to win/lose. In fact, the Bradley-Terry model can be considered as a special case of this model, where the only feature used is the player's identity. Naturally, this model could and should be trained with regularization. We use an $L2$-norm regularizer in our experiments.

The inability to handle game features remains unfixed by the logistic model. One may think game features could be added to this model in two straightforward ways. For one, one could add an additional weight vector for the game feature $M(a, b) = \mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) + \mathbf{w}'^T\mathbf{z}_g$. However, this is not a valid matchup function, because it does not satisfy property 3 introduced above. For the other, one could stack player and game feature vectors to form a new player feature vector, with the matchup function being

$$M(a, b) = \mathbf{w}^T \left( \begin{bmatrix} \mathbf{x}_a \\ \mathbf{z}_g \end{bmatrix} - \begin{bmatrix} \mathbf{x}_b \\ \mathbf{z}_g \end{bmatrix} \right). \tag{5}$$

However, the parts that are corresponding to the game features are the same for two players, and thus cancel out and have no effect on the output.

# 3. OUR FRAMEWORK

In this section, we detail our framework for general matchup modeling, which avoids the $O(n^2)$ paprameters, and can naturally incoporate all the player and game features. We first briefly review the *blade-chest* model from [10], which serves as the top layer of our framework. Then we introduce the bottom layers and explain how different features are added in.

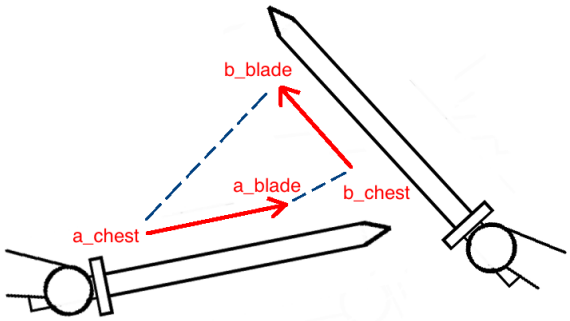


**Figure 1: A metaphorical illustration of the *blade-chest* model for intransitivity. Player $a$ and player $b$ are in a sword duel. Player $a$'s blade is closer to player $b$'s chest than vice versa, as shown by the two blue dashed lines. This illustrates how player $a$ has a better chance of winning than player $b$.**

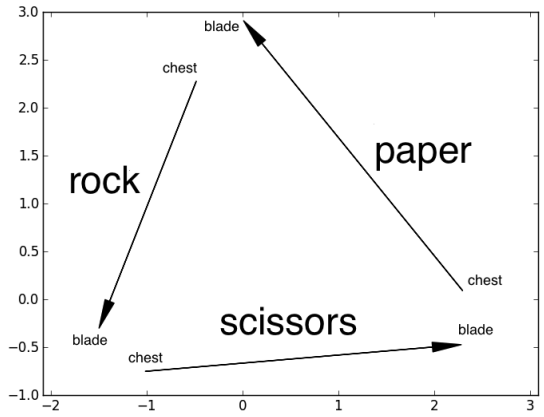


**Figure 2: The visualization of the *blade-chest-dist* model trained on a synthetic rock-paper-scissors dataset with $1,000$ games for each matchup. $d$ is set to $2$. Each player is represented by an arrow, with the head being the blade vector and the tail being the chest vector.**

## 3.1 The blade-chest model

Most of the previous works have the following in common: they use one single scalar to model the absolute strength of a player. Many of these works root from the Bradley-Terry model, and are surveyed in [8]. In some cases, a single scalar is an oversimplification, specifically since such models are not able to capture any intransitive rock-paper-scissors relation among three players if it exists in the data.

Our recent work of [10] models the intransitivity explicitly by using two $d$-dimensional vectors to represent a player. One is called the blade vector, and the other the chest vector. The winning and losing probabilities are decided based on the distance between one player's blade to his opponent's chest and vice versa. As depicted in Figure 1, player $a$ has an edge over player $b$ in this matchup.

Mathematically, one can write down the corresponding matchup function as

$$M(a,b) = ||\mathbf{b}_{\text{blade}} - \mathbf{a}_{\text{chest}}||_2^2 - ||\mathbf{a}_{\text{blade}} - \mathbf{b}_{\text{chest}}||_2^2, \quad (6)$$

which we call the *blade-chest-dist* model. The blade and chest vectors are the parameters of the model, and are learned from training data. Training this model on a synthetic rock-paper-scissors dataset gives us an interlocking visualization that correctly represents all three matchups as shown in Figure 2.

One variation of the *blade-chest-dist* model is to replace the Euclidean distance in the matchup function with the inner product, which gives

$$M(a,b) = \mathbf{a}_{\text{blade}} \cdot \mathbf{b}_{\text{chest}} - \mathbf{b}_{\text{blade}} \cdot \mathbf{a}_{\text{chest}}. \quad (7)$$

We call it the *blade-chest-inner* model. According to our empirical results in [10], modeling intransitivity this way is generally advantageous over the single-scalar methods. Between the two, the *blade-chest-inner* model usually performs better in terms of testing log-likelihood and accuracy, and is also more stable. Still these models cannot incorporate features other than the player's identity. In the later experimental section, we compare to this original *blade-chest-inner* model as a baseline, and call it the featureless model. In the rest of this section, we build upon the *blade-chest-inner* model for the general matchup modeling framework.

## 3.2 The blade-chest model as the top layer

Our framework has a two-layer structure. At the top layer, we use the *blade-chest-inner* model to output the winning probability,

$$\begin{aligned}&\Pr(a \text{ beats } b|g) = S(M(a,b|g)) \\ &= S(\mathbf{a}_{\text{blade}}(\mathbf{x}_a, \mathbf{z}_g) \cdot \mathbf{b}_{\text{chest}}(\mathbf{x}_b, \mathbf{z}_g) - \mathbf{b}_{\text{blade}}(\mathbf{x}_b, \mathbf{z}_g) \cdot \mathbf{a}_{\text{chest}}(\mathbf{x}_a, \mathbf{z}_g)).\end{aligned} \quad (8)$$

This top layer guarantees the needed symmetry, as $M(a,b|g) = -M(b,a|g)$ still holds. Instead of being $d$-dimensional vectors of free parameters for training, now the blade and chest vectors are functions of the player and game feature vectors. Our bottom layer serves as a feature mapper that bridges the blade/chest vectors and feature vectors, as detailed in the following subsections.

## 3.3 Bottom layer for player features only

For simplicity, we first discuss how the bottom layer works when we only use the player feature vectors. A natural way to link the space of blade/chest vectors and the space of feature vectors is by using a linear transformation. That is

$$\begin{aligned}\mathbf{a}_{\text{blade}}(\mathbf{x}_a) &= B\mathbf{x}_a \\ \mathbf{b}_{\text{blade}}(\mathbf{x}_b) &= B\mathbf{x}_b \\ \mathbf{a}_{\text{chest}}(\mathbf{x}_a) &= C\mathbf{x}_a \\ \mathbf{b}_{\text{chest}}(\mathbf{x}_b) &= C\mathbf{x}_b,\end{aligned} \quad (9)$$

where $B$ and $C$ are $d \times d_p$ parameter matrices that transform player feature vectors into blade or chest vectors respectively[1], and $d$ is a tunable parameter. Alternatively, we can link the two spaces by using a fully-connected feedforward neural network layer:

$$\begin{aligned}\mathbf{a}_{\text{blade}}(\mathbf{x}_a) &= f(B\mathbf{x}_a) \\ \mathbf{b}_{\text{blade}}(\mathbf{x}_b) &= f(B\mathbf{x}_b) \\ \mathbf{a}_{\text{chest}}(\mathbf{x}_a) &= f(C\mathbf{x}_a) \\ \mathbf{b}_{\text{chest}}(\mathbf{x}_b) &= f(C\mathbf{x}_b),\end{aligned} \quad (10)$$

[1]Note that when $\mathbf{x}_a$ and $\mathbf{x}_b$ are 0-1 vectors that only encode the players' identities, this recovers the original *blade-chest-inner* model as a special case.

where $f$ is the element-wise activation function. We choose hyperbolic tangent function $\tanh(\cdot)$ as $f$ (over other popular choices including the sigmoid function and the rectified linear unit), as its range includes both positive and negative values, similar to the original blade/chest vectors with free parameters.

The linear transformation Eq. (9) can also be viewed as the special case of Eq. (10), with the identity function as activation function. We will refer to the two different options as NOACT and TANH in later discussions.

## 3.4 Adding game feature vectors

Next we add game feature vectors so that they affect the two players differently, and the effects do not cancel out. Unlike for the logistic model, the concatenation of player vector and game vector is now sensible

$$\begin{aligned}
\mathbf{a}_{\text{blade}}(\mathbf{x}_a, \mathbf{z}_g) &= f_{\text{blade}}\left(B\begin{bmatrix}\mathbf{x}_a\\ \mathbf{z}_g\end{bmatrix}\right)\\
\mathbf{b}_{\text{blade}}(\mathbf{x}_b, \mathbf{z}_g) &= f_{\text{blade}}\left(B\begin{bmatrix}\mathbf{x}_b\\ \mathbf{z}_g\end{bmatrix}\right)\\
\mathbf{a}_{\text{chest}}(\mathbf{x}_a, \mathbf{z}_g) &= f_{\text{chest}}\left(C\begin{bmatrix}\mathbf{x}_a\\ \mathbf{z}_g\end{bmatrix}\right)\\
\mathbf{b}_{\text{chest}}(\mathbf{x}_b, \mathbf{z}_g) &= f_{\text{chest}}\left(C\begin{bmatrix}\mathbf{x}_b\\ \mathbf{z}_g\end{bmatrix}\right).
\end{aligned} \tag{11}$$

Passing through a NOACT layer, there will be nonzero cross terms $\mathbf{x}_a^T\mathbf{z}_g$ and $\mathbf{x}_b^T\mathbf{z}_g$ left to represent different influence of the game on each player. It applies to TANH similarly. We denote this choice of model as CONCAT. It is depicted in Figure 3.

An alternative model is to let the game feature vectors warp the blade and chest vectors directly. To do so, we first separately map the game feature vector into the same $d$-dimensional space as blade/chest vectors by applying a NOACT/TANH layer. Then we apply a Hadamard/entry-wise product for the warping,

$$\begin{aligned}
\mathbf{a}_{\text{blade}}(\mathbf{x}_a, \mathbf{z}_g) &= f(B'\mathbf{z}_g) \circ f(B\mathbf{x}_a)\\
\mathbf{b}_{\text{blade}}(\mathbf{x}_b, \mathbf{z}_g) &= f(B'\mathbf{z}_g) \circ f(B\mathbf{x}_b)\\
\mathbf{a}_{\text{chest}}(\mathbf{x}_a, \mathbf{z}_g) &= f(C'\mathbf{z}_g) \circ f(C\mathbf{x}_a)\\
\mathbf{b}_{\text{chest}}(\mathbf{x}_b, \mathbf{z}_g) &= f(C'\mathbf{z}_g) \circ f(C\mathbf{x}_b).
\end{aligned} \tag{12}$$

We call this model choice SPLIT, as the mapping of the player and game features happens separately. The entire pipeline is shown in Figure 4.

## 3.5 Training

We train our model to maximize the log-likelihood on the training dataset, that is (assuming $a$ is the winner)

$$\underset{\Theta}{\text{argmax}} \sum_{(a,b,g)\in D} \log \Pr(a \text{ beats } b|\Theta, g). \tag{13}$$

$\Theta$ denotes all the parameters, and it typically consists of $B$ and $C$, with additional $B'$ and $C'$ for the SPLIT model. We use the Frobenius norms of these parameter matrices as regularization terms.

The training of our model is done via online backpropagation [38]. Until convergence, we iterate through the entire training dataset repeatedly, feed one game to the model at a time, update the parameters immediately based on the result of backpropagation. Our implementation in C that contains all the different training options and is available at http://www.cs.cornell.edu/~shuochen/.

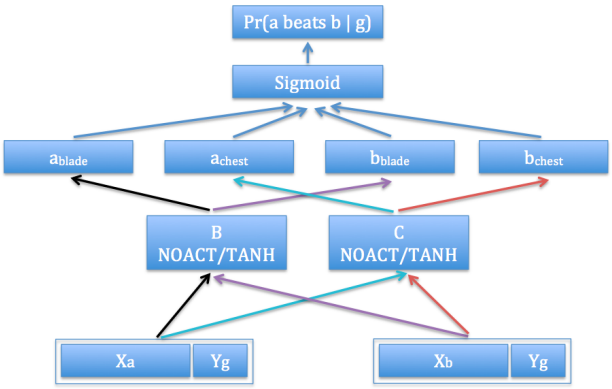


**Figure 3: Pipeline of CONCAT model.**

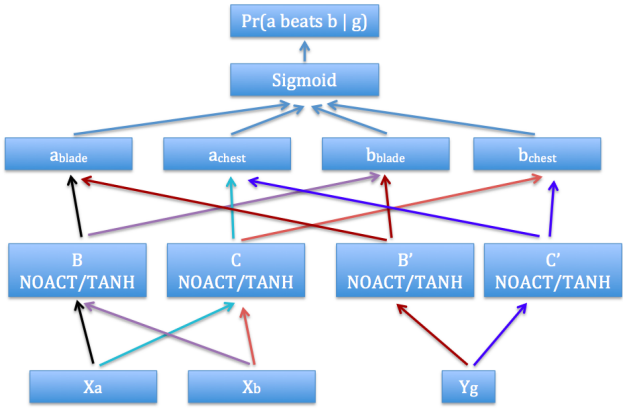


**Figure 4: Pipeline of SPLIT model.**

# 4. RELATED WORK

Pairwise comparison has been studied since the seminal work of [42], which later led to the well-known Bradley-Terry model [7, 27]. They paved the way for most of the research in the area, as surveyed in [8].

Learning to rank a player's strength in order to predict results or do matchmaking in sports or online video games has been a successful application of pairwise comparison modeling. The famous Elo rating system [17], a variation of Bradley-Terry model, started in rating chess players, and is now widely used in many sports prediction applications and matchmaking systems of online video games (a.k.a esports) [1]. Another example is the Trueskill™ ranking system [18] developed by Microsoft. It uses a univariate Gaussian distribution to model each player's skill and uncertainty, and its Bayesian inference is done via approximate message passing on a factor graph. Its follow-up works include [13, 32]. [45] proposes a factorization method to model players' ratings in different contexts[2]. There are also works that aim at inferring each player's strength through learning from group competition [19, 30]. The goal of these works and many others along the line is to learn a scalar parameter for each of the players from historical pairwise comparison data. These parameters usually represent the absolute strengths of the individuals, with larger values favored for the win over smaller values in future games.

[2]A context in their paper is represented by a set of discrete variables, e.g. a map one round of online video game is played on.

However, for many real-world settings, these types of models can be an oversimplification. For example, they are unable to capture effects like intransitivity (rock-paper-scissors relation) that exist in real data [10]. To remedy this, several works used more expressive ways of modeling players, and report improved performance. [9, 10] use one or more vectors to represent a player, and are able to explicitly model the intransitive behaviors. [21, 43] generalize the Bradley-Terry model with vectorized representations of a player's ranking. The previously discussed [19] could also fall into this category, as the player is represented by a vector of its ratings for different discrete contexts. [2] designs a matrix factorization algorithm to predict scores of professional basketball games.

It is worth noting that the intransitivity caused by multiple attributes has been studied in economics and decision theory [28, 16, 26]. Instead of taking a learning perspective, these works assume the features/attributes of items are given, and study what type of underlying thought process could cause intransitivity. Thus, it is an inverse process of what our work and the above works are about.

The BalanceNet model proposed in [14] is closely related to our work. It uses a multi-layer neural network to help matchmaking in *Ghost Recon Online*, a first-person shooter (FPS) online video game developed by Ubisoft. It can also account for player features such as number of matches played, average kill/death ratio, etc. The main differences that make it incomparable with our work are twofold. First, the game they addressed is asymmetric. The opposing players in one match are playing different roles, which means if you switch the input of them, you do not get one minus your previous probability. This comes with the nature of *Ghost Recon Online*, and makes it not applicable to general symmetric matchups unlike our framework[3]. Second, it is not designed to make use of any game features.

Most of the aforementioned works on player modeling and game prediction only take into account players' identities, with exceptions of [19] considering discrete context variables and [14] using additional player features. Different from them, our work in this paper is a general probabilistic matchup modeling framework for any two-player game with more expressive power than player-ranking. It can also utilize any vectorized player features and game features, no matter whether the features are discrete or continuous.

Another application of pairwise comparison is preference learning [15, 11, 44]. Instead of one player beating the other, here one observation in the dataset is one item being preferred (by a human) over the other. In a typical preference learning setting, the training data consists of pairwise preferences of instances, represented by a feature vector. The goal is to learn an underlying ranking function (with scalar output) that satisfies the training data as much as possible. A similar task arises in the context of search engines. It is studied as the "learning from implicit feedback problem" [12, 22, 34, 23], where user expresses his/her preference by clicking on retrieved items. The major difference between these work and our work is threefold. First, the items' identities are usually not part of the feature vectors in their empirical tests (although theoretically can be). Other auxiliary features are used instead. Second, similar to aforementioned player-ranking models, they try to learn a scalar ranking function for the instance, which has limited expressiveness in representing the relations between items. Lastly, there is no notion or no natural way of accounting for context information in which the choice is made.

[3]We would argue that a symmetric matchup is a more general form, as if the game is asymmetric, it could still be modeled as a symmetric one. One possible approach could be adding to the player feature vector to specify what asymmetric role he is playing in this game.

Learning with context has been studied in the recommendation community, and one line of research that is related to our work is context-aware recommender systems. While traditional recommender systems [37] consider only the ratings of users given to items[4], context-aware recommendation systems recognize that the context in which users rate items could be a crucial factor that impacts the evaluation [3]. There is also evidence from the research in decision-making domain that different contexts can change human decisions significantly [40, 20]. As a result, making use of the context information could significantly boost the performance of the recommender system, as reported in representative works like [5, 25, 39, 36]. Here, the task is learning from user ratings of items under different contexts to predicting any missing values, the unknown ratings for any triple of user, item and context. It is usually measured by mean average error (MAE) of the predicted ratings, or normalized discounted cumulative gain (NDCG) for retrieval tasks. This differs from our work, which concerns learning from user's pairwise preferences instead of cardinal ratings to predict the choice and its probability, not ratings for items, as accurately as possible.

# 5. EXPERIMENTS

We conducted an extensive empirical evaluation of the models on both competitive matchup data and pairwise preference data. In each domain, we report results on both real-world data to demonstrate applicability and practicality, and on synthesized data to demonstrate the expressiveness of our models.

## 5.1 Experiment setup

We first introduce the general setup of all the experiments. If not specified otherwise, the experiments are run as follows. We train various models (both ours and baselines) with different $d$'s (where applicable) and regularization intensities[5] on the training dataset. The model that reports the best performance on the validation dataset in terms of log-likelihood is then tested on the test dataset.

For evaluation, we use two metrics: average test log-likelihood and test accuracy. The average test log-likelihood is defined similarly to the training log-likelihood. For the test partition $D'$, assuming $a$ is the winner,

$$L(D'|\Theta) = \frac{1}{|D'|} \sum_{(a,b,g)\in D'} \log(\Pr(a \text{ beats } b|\Theta, g)), \qquad (14)$$

where $|D'|$ is the total number of games in the testing set. Log-likelihood is always a negative value. The higher the value is, the better the model performs. The test accuracy is defined as

$$A(D'|\Theta) = \frac{1}{|D'|} \sum_{(a,b,g)\in D'} \mathbb{1}^{\{\Pr(a \text{ beats } b|\Theta,g)\geq 0.5\}}. \qquad (15)$$

$\mathbb{1}^{\{\cdot\}}$ is the indicator function. This metric is a real number in $[0, 1]$, representing the percentage of matches whose binary outcome can be correctly predicted by the model. The higher the value is, the better the model performs.

For the baselines, we compare our methods with Rote Learning, Bradley-Terry, the Pairwise Logistic Model and the Trueskill™

[4]Works that make use of the inherent features of users and items also exist [4, 41].

[5]The selection of these hyperparameters is through grid search, with $d$ from 2 to 100, and the shared regularization hyperparameter $\lambda$ across all parameter matrices from 1E-7 to 1E3.

ranking system[6]. For our methods, we also vary the information we input into the models, from featureless (Section 3.1) , player/item features only to both player/item features and game/context features. Note that methods from context-aware recommender system are not applicable to pairwise preference prediction as explained in the last paragraph of Section 4.

## 5.2 Experiments with synthetic datasets

We first explore the expressiveness of the models on three synthetic datasets that correspond to plausible real-world scenarios, but allow us to control the effect we aim to study.

### 5.2.1 Synthetic Datasets

For all datasets, we vary the amount of data, but always split to have roughly a $5:2:3$ training, validation and testing ratio.

*Dataset "syn-rand".* With this dataset, we explore how changes in context make the outcome of a game more random (e.g., bad weather making a tennis match more random). Furthermore, we simulate how the home-field advantage gives the home player an edge. Data is generated as follows. We create $50$ players in total. In any game, there are also $2$ game features: weather being normal or bad. Each happens $50\%$ of the times. If the weather is bad, the outcome of the matchup is $50/50$ before considering the home-field's effect discussed below. Otherwise, we assign a $75/25$ chance for each player to win prior to considering home-field advantage. The favored player is selected via a coin toss. In addition to the game features, there are $3$ players features (in addition to player's identity features): playing at home, playing away or playing in a neutral environment. In any game, one of the following two is true: one player is at home and the other is away, or both players are playing in neutral environments. The two scenarios are equally likely. If one player has the home-field advantage, the player's chances of winning increase by $10$ percent (e.g. a $75/25$ matchup when the favored player is away becomes a $65/35$ matchup). Otherwise, the probability is not altered. To generate a game, we randomly sample two different players, the home-field and the weather features. We then sample the outcome according to the resulting matchup probability.

*Dataset "syn-determ".* This dataset is analogous to "*syn-rand*", but now context (i.e., weather) makes a game more deterministic. In particular, there are also $2$ game features: weather being normal or good. Good weather alters the winning probability to $100/0$ while disregarding the home-field feature. Normal weather does not alter the winning probability.

*Dataset "syn-attr".* With this dataset we study a scenario where context can switch the polarity of a preference. For example, imagine a case when the user choose between two restaurant. If the user is already in town and logs on via a mobile device, the distance to the restaurants may be the most important factor in the decision. On the other hand, if we are still at home with a desktop, the quality of the restaurant may overweigh the distance, as we need to drive anyway. Using this scenario, the data contains $50$ restaurants in total. Each restaurant is associated with two binary $0/1$ numbers that represent distance and quality. The chance for each number to be $0$ or $1$ is $50\%$. For each pairwise comparison, we uniformly select two restaurants and the mobile/desktop context, which is also equally likely. We compare the two $0/1$ numbers of the given context. If they are the same, we select the winner via a coin toss. Otherwise, the one with larger number ($1$ in this case) wins the comparison.

### 5.2.2 Empirical results

We test our models with different input information, and tune them across all training option combinations (NOACT, TANH, CONCAT, SPLIT) on the validation set. This ends up with "featureless" (no additional feature used other than identities), "player only" (best result by using only player features) and "player+game" (best result by using both player and game features)[7]. We compare the results against the aforementioned baselines. Note that "player only" and "logistic" do not apply to the preference data, as there are no player/item features other than identity. They are essentially the same as "featureless" and "Bradley-Terry".

We plot the performance metrics against the size of the dataset in Figure 5 and 6. In addition, we introduce two limits, denoted as "theoretical best" and "theoretical best rote"[8]. For the first, it is the limit we could get by being fully aware of how the data is generated and having an infinite amount of training and testing data. For the second, it is similar to "theoretical best", but the infinite amount of data only contains players' identities, not additional player and game features. This corresponds to the input information that featureless, Bradley-Terry, and Trueskill™ can make use of.

**How do our models compare against the baselines?** We observe that our models generally outperform all the baselines. With full information, our models are favored when compared with the rest in terms of log-likelihood, and are almost always better in terms of accuracy. The only exception is test accuracy for *syn-determ* data in the middle panel of Figure 6, where all methods perform well since there are so many 100/0 matchups generated.

**How do different levels of input information affect our models?** In general, we see that extra information helps improve the performance when comparing the curves of "player+game", "player only" and "featureless" in all plots, especially as the numbers of games grow. It is interesting to point out that including player feature does not help much when comparing against the featureless baselines on *syn-determ*, as the "player only" curve is barely above "featureless" in the middle panels of Figure 5 and Figure 6. This is due to the way in which the dataset is generated (when the weather is good, home-field advantage is ignored and the better player wins).

**How do our models compare against the theoretical limits?** On the *syn-rand* data where bad weather makes a game much more unpredictable, it seems that more games are needed for "featureless" to approach "theoretical best rote" (the two curves are still noticeably apart at the right end in the left panels of Figure 5 and Figure 6). Other than that we can see our context-sensitive methods "player+game" tend to converge to the corresponding theoretical limits as the numbers of games grow in all six plots. This provides evidence that our methods can indeed capture the underlying generation process, including the three distinct effects that context has in the three datasets.

## 5.3 Experiments with real-world datasets

We now turn to the real-world applications to verify the applicability and practicality of our methods. The datasets span the competitive matchup and preference domains: professional tennis and *Starcraft II* data for competitive matchups, and three datasets converted from context-aware recommender system research for pairwise preferences.

[6]The original Trueskill™ranking system runs in an online fashion: it sees a game, updates the parameters, and never sees that game again. However, to make it a fair comparison with other methods that run in batch mode, we also pass the training set through the Trueskill™ model multiple times until we have the best validation results.

[7]We only report the best results for each level of input information to keep the plots uncluttered.

[8]The derivations of the two limits are omitted due to limited space.

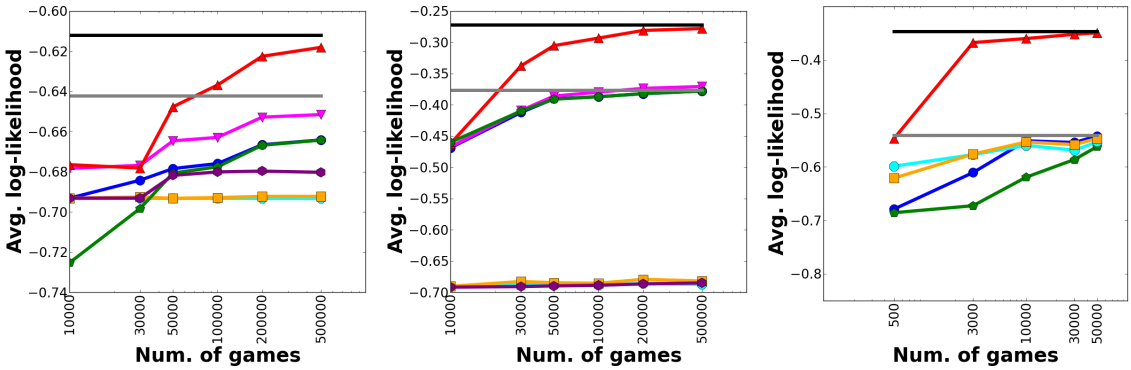


**Figure 5: Average log-likelihood on *syn-rand* (left panel), *syn-determ* (middle panel) and *syn-attr* (right panel) datasets.**

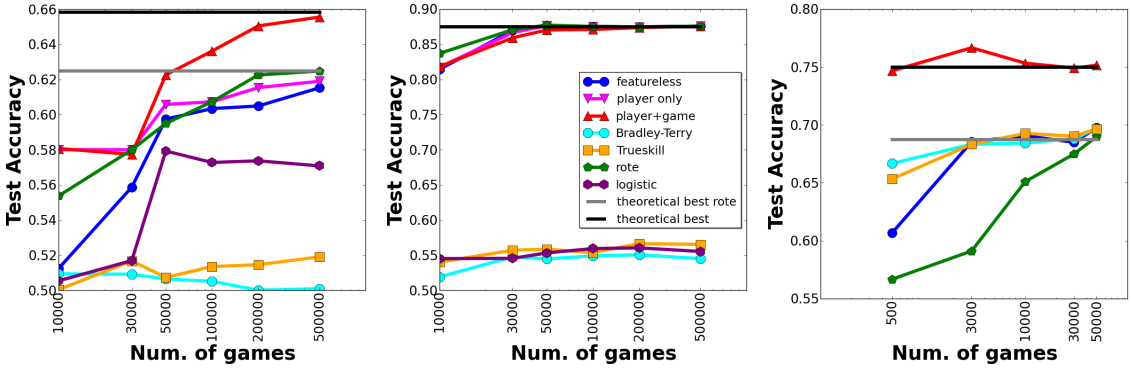


**Figure 6: Average test accuracy on *syn-rand* (left panel), *syn-determ* (middle panel) and *syn-attr* (right panel) datasets. Note that the two limits overlap in the middle panel. Moreover, due to the fact that all the generated datasets are finite, it is possible for any method that learns from them to go above the limits that is assuming infinite data due to sampling error. It is the case here in the right panel.**

### *5.3.1 Real-World Datasets*

*Tennis Datasets.* We first look at the data of matches between top male tennis players in the world from year 2005-2015 and female players from year 2007-2015. These games are from all levels of tournaments organized by Association of Tennis Professionals (ATP) and Women's Tennis Association (WTA). The full collection of data can be found at http://tennis-data.co.uk/alldata.php. It contains match history of in total 914 players and 28,054 games for ATP, and 775 players and 21,488 games for WTA. For the experiments, we use the games before (including) the year 2012 for training and validation, and the later games for testing. The training/validation ratio is roughly 4 : 1. We end up with 17,622 training games, 4,401 validation games and 6,031 testing games for ATP, and 11,725 training games, 2,928 validation games and 6,835 testing games for WTA.

The data source above also contains more detailed information about the players and games, which serves as feature vectors in our model. For players features we have player's identity, world ranking when the game was played, points for world ranking when the game was played, whether on a winning/losing streak and odds from several betting companies/websites (Bet365, Expekt, Ladbrokes, Pinnacles Sports and Stan James). The game features contain the location at which the game took place, the series the game belongs to (e.g. International Gold, Masters), indoor or outdoor court, surface of the court (e.g. Grass, Clay, Hard, Carpet), the stage the game was at (e.g. Group stage, Round of 16, Semifinal, Final) and the format of the game (usually best of 3 or best of 5). Overall, we extracted 927 player features for ATP and 788 for WTA, and also 119 games features. As one can see from above that most of the features are discrete, these feature vectors are very sparse.

*Starcraft II Dataset. Starcraft II* is an online military science fiction real-time strategy game developed and published by Blizzard Entertainment[9]. It is one of the most popular games in the esports scene. In the most common competitive setting, two players face off against each other. Each player as a war commander picks one from the three distinct in-game races: Terran, Protoss or Zerg as his army, each of which has different strengths, powers and abilities. The battle takes place on a battleground which is called a map. Both players collect resources on the map and build various of combat units to fight the opponents, until one player's force is completely wiped out and the other claims victory. The combat units all have different attributes such as building cost, building time, movement speed, attack range, toughness, etc. A set of *Starcraft II* strategies include managing different resources, building an army with right components at the right time, scouting and reacting to opponents' moves properly, etc.

We crawled *Starcraft II* competition data from a third-party website aligulac.com from the beginning of their database to April 8th, 2015. It contains 129,005 matches among 4,761 professional *Starcraft II* players from various online and offline tournaments. They are randomly divided into training, validation, testing sets according to a roughly 5 : 2 : 3 ratio, giving us 64,505 for training, 25,800 for validation, and 38,700 for testing. The reason we did not divide them according to some chronological order like the tennis

[9]http://us.battle.net/sc2/en/

data is because the game is evolving. There are at least two major phases of the game: the original release of the game with the subtitle Wings of Liberty (WoL), and an expansion three years later called Heart of the Swarm (HotS). Within each phase, there are many patches that modify the rules, the units or the maps to make the game more balanced and enjoyable. Some of these patches are more important than the others. By mixing up all games, we eliminate the possibility that the training and testing parts being based on very different phases of the game.

The website aligulac.com also contains rich information about these games, from which we select a few that we believe to be the most informative as player features. There are $4,851$ in total, including player's identity, in-game race, nationality[10], overall rating and standard deviation from aligulac.com and rating and standard deviation versus opponent's race from aligulac.com. We also use $40$ game features, including the phase of the game (WoL or HotS), game being played online or offline, the quarter of the year in which the game was played and various keyword features contained in the event names that appear most often (e.g. Group, Qualifier, Invitational, Europe, ProLeague, Playoff, Dreamhack, etc.).

*Context-Sensitive Preference Datasets.* For our experiments, the ideal setting for collecting data would be presenting two choices to a human judge in various context and ask them to pick one over the other. However, there is no such a dataset that is publicly available as far as we know. Instead, we take the datasets from context-aware recommender systems and process them into the format we need. In these datasets, each entry is a user's rating for a given item under certain context. We group all the entries by user and context. Within each group, for any two items with different ratings, we can generate a pairwise comparison along with the context. The datasets we use are:

1. The Food dataset from [33]. It contains $4,036$ comparisons among 20 food menus from 212 users[11]. The user are in three different levels of hunger, each of which could either be real or supposed situation.
2. The Tijuana restaurant dataset from [35]. It is a dataset of $50$ people taking questionnaire about their preference on nearby $40$ restaurants. There are two different contexts: time (weekday or weekend) and location (at school, work or home) when the options are provided. We generate $4,041$ comparisons in total.
3. The DePaul movie dataset from [46]. $97$ students of DePaul University participated in an online survey regarding $79$ movies for different occasions. The context contains time (weekday, weekend or N/A), location (home, cinema or N/A) and companion (alone, partner, family or N/A). We create $26,264$ comparisons out of the original data.

Each dataset is randomly divided into training, validation and testing in a roughly $5:2:3$ ratio.

### 5.3.2 Empirical results

The results are in Table 1, 2, 3 and 4. In each table, we separate the baselines and our methods with a horizontal double line. In the lower section for our models, different training options are blocked and ordered according to increasing information for training: featureless for using only player/item's identity, NOACT and TANH for using all the player/item features, and the rest for using both player/item and game/context features. There are fewer rows in Table 3 and 4 than in Table 1 and 2. This is because we do not have additional item features there and therefore some methods are essentially the same.

**How do our models compare against the baselines across different applications?** Overall, our methods are favored against all the baselines. In all of the four tables, the best results appear in the sections of our models that use all information available. We also do paired t-tests between our best results and the best baselines[12], and all of them pass $0.05$ significance level except for the accuracy on Food and WTA, for which we have a $p$-value of $0.76$ and $0.18$ against rote. When comparing our best results among "TANH" and "NOACT" with pairwise logistic regression, all of which only use player features, ours are also always better except for the accuracy on WTA ($p$-value of $0.07$, while others are less than $0.05$). On the other hand, our featureless model do not always beat its featureless counterparts in the baselines. This is similar to what we found in [10], where we need to include a Bradley-Terry-like bias term in our model in order to surpass the baselines on certain datasets. In terms of the extent of improvement over the best baselines, the results on *Starcraft II*, Tijuana and DePaul stand out.

**How do additional player features affect the models?** Comparing "featureless" with NOACT/TANH in Table 1 and 2, we can see the positive effect of adding additional player features, and all of the comparisons pass a $p < 0.05$ paired t-test. Moreover, the player features we use for tennis seem to be more influential than those used for *Starcraft II* (more than $5\%$ boost versus about $1\%$ in accuracy). Our conjecture is that the world ranking and points features, as well as the betting companies' odds features are the ones responsible. By using them, we are bootstrapping from other unknown prediction models. To test it, we run experiments while withholding some of these features. The results on ATP and WTA are quite similar: when the betting odds features are withheld, the performance dropped to around halfway in between TANH and "featureless". On the other hand, it is only slightly pushed down when the world ranking and points features are withheld. This suggests that the betting odds features are crucial in the improvement of performance on tennis data.

**Does the contextual information have different effects in the two domains?** Yes. Looking at the improvements between the best results with and without the game/context features, it seems the preference modeling, especially on Tijuana and DePaul, benefits more. In terms of accuracy, they both gain about $8\%$, while on matchup data it is around $1\%$ (all pass $p < 0.05$ t-test except for WTA with $p$-value of $0.18$). This suggests that in competition, the outcome depends more on the players' intrinsic properties than the environmental factors, while for human preference, the context in which the decision is made plays a much more important role.

**Which training options are preferred?** TANH appears to be better than NOACT for both log likelihood and accuracy and on all the datasets. When it comes to the choice between CONCAT and SPLIT, the choice is less clear. There are few occasions where SPLIT beats CONCAT as the bold numbers suggest, but usually not by a lot. On the other hand, SPLIT can be outperformed by the methods without using contextual features, on the tennis datasets for example. The significance tests suggest that TANH beats NOACT and CONCAT beats SPLIT in all scenarios ($p < 0.05$) except accuracy of TANH against NOACT on WTA and log-likelihood of CONCAT against SPLIT on Tijuana (both with $p$-values of $0.27$).

[10]We think it is an important feature as the *Starcraft II* competitive scene is mostly dominated by Korean players.

[11]We directly use the data processed by [31], which can be found at https://github.com/trungngv/gpfm

[12]The t-tests here and hereafter are done on $10$ random splits of the datasets, with hyperparameters fixed to the optimal values obtained from the main experiments. Bonferroni correction is used for multple comparisons.

| Model | ATP | WTA | *Starcraft II* |
|---|---|---|---|
| rote | $-0.6764$ | $-0.6873$ | $-0.5831$ |
| Bradley-Terry | $-0.6092$ | $-0.6721$ | $-0.5778$ |
| Trueskill™ | $-0.6209$ | $-0.6643$ | $-0.6001$ |
| logistic | $-0.5777$ | $-0.6229$ | $-0.5808$ |
| featureless | $-0.6590$ | $-0.6722$ | $-0.5886$ |
| NOACT | $-0.5974$ | $-0.6174$ | $-0.5299$ |
| TANH | $-0.5633$ | $-0.5874$ | $-0.5232$ |
| NOACT CONCAT | $-0.5970$ | $-0.6166$ | $-0.5229$ |
| TANH CONCAT | $\mathbf{-0.5616}$ | $\mathbf{-0.5865}$ | $\mathbf{-0.5177}$ |
| NOACT SPLIT | $-0.6051$ | $-0.6283$ | $-0.5249$ |
| TANH SPLIT | $-0.5981$ | $-0.6228$ | $-0.5178$ |

**Table 1: Test log-likelihood on competitive matchup datasets.**

| Model | ATP | WTA | *Starcraft II* |
|---|---|---|---|
| rote | 55.63% | 53.52% | 68.77% |
| Bradley-Terry | 66.66% | 61.26% | 69.21% |
| Trueskill™ | 66.97% | 61.96% | 69.82% |
| logistic | 69.89% | 68.02% | 71.68% |
| featureless | 63.70% | 58.45% | 73.08% |
| NOACT | 69.72% | 68.09% | 73.75% |
| TANH | 70.35% | 68.46% | 74.07% |
| NOACT CONCAT | 69.86% | 68.20% | 74.22% |
| TANH CONCAT | **70.40%** | **68.62%** | 74.66% |
| NOACT SPLIT | 69.41% | 67.18% | 73.87% |
| TANH SPLIT | 69.87% | 68.11% | **75.10%** |

**Table 2: Test accuracy on competitive matchup datasets.**

| Model | Food | Tijuana | DePaul |
|---|---|---|---|
| rote | $-0.6943$ | $-0.7371$ | $-0.6255$ |
| Bradley-Terry | $-0.6927$ | $-0.6929$ | $-0.6082$ |
| Trueskill™ | $-0.6720$ | $-0.7014$ | $-0.5916$ |
| featureless | $-0.6750$ | $-0.6864$ | $-0.6009$ |
| NOACT CONCAT | $-0.6709$ | $-0.4321$ | $-0.6033$ |
| TANH CONCAT | $-0.6709$ | $\mathbf{-0.4108}$ | $\mathbf{-0.5038}$ |
| NOACT SPLIT | $-0.6741$ | $-0.5597$ | $-0.5927$ |
| TANH SPLIT | $\mathbf{-0.6701}$ | $-0.4207$ | $-0.5531$ |

**Table 3: Test log-likelihood on pairwise preference datasets with context.**

| Model | Food | Tijuana | DePaul |
|---|---|---|---|
| rote | 59.08% | 50.33% | 65.60% |
| Bradley-Terry | 58.33% | 54.79% | 66.69% |
| Trueskill™ | 57.59% | 75.33% | 67.67% |
| featureless | 59.08% | 58.00% | 68.21% |
| NOACT CONCAT | 57.34% | 81.85% | 68.29% |
| TANH CONCAT | 57.26% | **82.10%** | **75.56%** |
| NOACT SPLIT | 58.75% | 80.53% | 71.13% |
| TANH SPLIT | **60.81%** | 80.36% | 73.13% |

**Table 4: Test accuracy on pairwise preference datasets with context.**

As a result, we would suggest TANH CONCAT to be the go-to option of our model.

# 6. CONCLUSIONS

We presented in this paper a general probabilistic framework for modeling competitive matchup and pairwise preference that can utilize any vectorized player/item and game/context features. We conducted experiments on synthetic datasets that simulate plausible real-world scenarios in both domains. The results demonstrate that our models, while outperforming the baselines, can also approach the theoretical best limit as the number of games/comparisons grows. Experimental results on the real-world datasets also clearly demonstrate the advantage of our models and the benefit of utilizing the additional features. It is interesting to see that the contextual features are more influential in the preference domain than in competitive matchups.

# Acknowledgments

We thank Arzoo Katiyar and Yexiang Xue for their valuable suggestions and advice. This research was funded in part through NSF Awards IIS-1247637, IIS-1217686, and IIS-1513692.

# 7. REFERENCES

[1] Wikipedia page on elo rating system. https://en.wikipedia.org/wiki/Elo_rating_system.

[2] R. P. Adams, G. E. Dahl, and I. Murray. Incorporating side information in probabilistic matrix factorization with gaussian processes. *arXiv preprint arXiv:1003.4944*, 2010.

[3] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[4] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

[5] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 245–248. ACM, 2009.

[6] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[7] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, pages 324–345, 1952.

[8] M. Cattelan et al. Models for paired comparison data: A review with emphasis on dependent data. *Statistical Science*, 27(3):412–433, 2012.

[9] D. Causeur and F. Husson. A 2-dimensional extension of the bradley–terry model for paired comparisons. *Journal of statistical planning and inference*, 135(2):245–259, 2005.

[10] S. Chen and T. Joachims. Modeling intransitivity in matchup and comparison data. In *Proceedings of The 9th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2016.

[11] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144. ACM, 2005.

[12] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, volume 10, page 451. MIT Press, 1998.

[13] P. Dangauthier, R. Herbrich, T. Minka, T. Graepel, et al. Trueskill through time: Revisiting the history of chess. In *NIPS*, 2007.

[14] O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. C. Ferrari, Y. Bengio, and F. Zhang. Beyond skill rating: Advanced matchmaking in ghost recon online. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(3):167–177, 2012.

[15] J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.

[16] W. V. Gehrlein. The probability of intransitivity of pairwise comparisons in individual preference. *Mathematical social sciences*, 17(1):67–75, 1989.

[17] M. E. Glickman. A comprehensive guide to chess ratings. *American Chess Journal*, 3:59–102, 1995.

[18] R. Herbrich, T. Minka, and T. Graepel. Trueskill$^{TM}$: A bayesian skill rating system. In *Advances in Neural Information Processing Systems*, pages 569–576, 2006.

[19] T.-K. Huang, C.-J. Lin, and R. C. Weng. Ranking individuals by group comparisons. In *Proceedings of the 23rd international conference on Machine learning*, pages 425–432. ACM, 2006.

[20] J. Huber, J. W. Payne, and C. Puto. Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research*, pages 90–98, 1982.

[21] D. R. Hunter. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406, 2004.

[22] T. Joachims et al. Evaluating retrieval performance using clickthrough data., 2003.

[23] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM, 2005.

[24] D. Jurafsky and J. H. Martin. *Speech & language processing*. Pearson Education India, 2000.

[25] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[26] P. Linares. Are inconsistent decisions better? an experiment with pairwise comparisons. *European Journal of Operational Research*, 193(2):492–498, 2009.

[27] R. D. Luce. *Individual Choice Behavior a Theoretical Analysis*. john Wiley and Sons, 1959.

[28] K. O. May. Intransitivity, utility, and the aggregation of preference patterns. *Econometrica: Journal of the Econometric Society*, pages 1–13, 1954.

[29] I. McHale and A. Morton. A bradley-terry type model for forecasting tennis match results. *International Journal of Forecasting*, 27(2):619–630, 2011.

[30] J. E. Menke and T. R. Martinez. A bradley–terry artificial neural network model for individual ratings in group competitions. *Neural computing and Applications*, 17(2):175–186, 2008.

[31] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 63–72. ACM, 2014.

[32] S. Nikolenko and A. Sirotkin. A new bayesian rating system for team competitions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 601–608, 2011.

[33] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. *UMAP*, 9:102–113, 2009.

[34] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM, 2005.

[35] X. Ramirez-Garcia and M. García-Valdez. Post-filtering for a restaurant context-aware recommender system. In O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk, editors, *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, volume 547 of *Studies in Computational Intelligence*, pages 695–707. Springer International Publishing, 2014.

[36] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[37] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

[39] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 155–164. ACM, 2012.

[40] I. Simonson and A. Tversky. Choice in context: Tradeoff contrast and extremeness aversion. *JMR, Journal of Marketing Research*, 29(3):281, 1992.

[41] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, pages 111–120. ACM, 2009.

[42] L. L. Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

[43] S. Usami. Individual differences multidimensional bradley-terry model using reversible jump markov chain monte carlo algorithm. *Behaviormetrika*, 37(2):135–155, 2010.

[44] G. N. Yannakakis, M. Maragoudakis, and J. Hallam. Preference learning for cognitive modeling: a case study on entertainment preferences. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(6):1165–1175, 2009.

[45] L. Zhang, J. Wu, Z.-C. Wang, and C.-J. Wang. A factor-based model for context-sensitive skill rating systems. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 2, pages 249–255. IEEE, 2010.

[46] Y. Zheng, B. Mobasher, and R. Burke. Carskit: A java-based context-aware recommendation engine. In *Proceedings of the 15th IEEE International Conference on Data Mining Workshops*. IEEE, 2015.