Skill Rating for Multiplayer Games Introducing Hypernode Graphs and their Spectral Theory

Thomas Ricatte, Rémi Gilleron, Marc Tommasi

Journal of Machine Learning Research 21 (2020) 1-18 Submitted 12/13; Revised 3/20; Published 4/20

Skill Rating for Multiplayer Games
Introducing Hypernode Graphs and their Spectral Theory

Thomas Ricatte THOMAS@RICATTE.FR
Rémi Gilleron REMI.GILLERON@UNIV-LILLE.FR
Marc Tommasi MARC.TOMMASI@UNIV-LILLE.FR
Université de Lille and Inria Lille Nord Europe
40, Av. Halley, 59650 Villeneuve d'Ascq, France

Editor: David Blei

Abstract

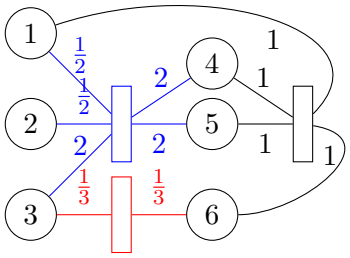
We consider the skill rating problem for multiplayer games, that is how to infer player skills from game outcomes in multiplayer games. We formulate the problem as a minimization problem $\arg\min_s s^T \Delta s$ where Δ is a positive semidefinite matrix and s a real-valued function, of which some entries are the skill values to be inferred and other entries are constrained by the game outcomes. We leverage graph-based semi-supervised learning (SSL) algorithms for this problem. We apply our algorithms on several data sets of multiplayer games and obtain very promising results compared to Elo Duelling (see Elo, 1978) and TrueSkill (see Herbrich et al., 2006). As we leverage graph-based SSL algorithms and because games can be seen as relations between sets of players, we then generalize the approach. For this aim, we introduce a new finite model, called *hypernode graph*, defined to be a set of weighted binary relations between sets of nodes. We define Laplacians of hypernode graphs. Then, we show that the skill rating problem for multiplayer games can be formulated as $\arg\min_s s^T \Delta s$ where Δ is the Laplacian of a hypernode graph constructed from a set of games. From a fundamental perspective, we show that hypernode graph Laplacians are symmetric positive semidefinite matrices with constant functions in their null space. We show that problems on hypernode graphs can not be solved with graph constructions and graph kernels. We relate hypernode graphs to signed graphs showing that positive relations between groups can lead to negative relations between individuals.

Keywords: Hypergraphs, Graph Laplacians, Graph Kernels, Spectral Learning, Semi-supervised Learning, Multiplayer Games, Skill Rating Algorithms.

1. Introduction

We consider the skill rating problem for multiplayer games such as team sports and online games. We consider games between two teams where each team is composed of an arbitrary number of players. We assume, as in Herbrich et al. (2006), that the skill of a team is a weighted sum of the player skills. The problem is to infer individual player skills from a limited number of game outcomes. The skills can then be used for match-making, skill ranking or outcome prediction for new games.

Given a set of players and a set of games between teams of players, we model the player skills and the game outcomes by real-valued functions. We show that the skill rating problem is equivalent to solving a minimization problem $\arg\min_s s^T \Delta s$ where Δ is a positive



$$G = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 \\ 1 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & \frac{\sqrt{3}}{3} & 0 & 0 & -\frac{\sqrt{3}}{3} \end{pmatrix}$$

Figure 1: Left: a hypernode graph $\mathbf{h}$ with three hyperedges $h_{blue} = \{\{1,2,3\},\{4,5\}\}$, $h_{black} = \{\{4,5\},\{1,6\}\}$ and $h_{red} = \{\{3\},\{6\}\}$ where a hyperedge is represented by a rectangle and nodes in a hypernode are connected to the same side of the rectangle. Right : a gradient G of $\mathbf{h}$ which values are square root of the weights and are positive for one hypernode and negative for the other. The matrix G can be seen as an alternative definition of the hypernode graph $\mathbf{h}$.

semidefinite matrix and some entries of s are constrained by the game outcomes. In order to solve this problem, we propose two algorithms. The first one is the semi-supervised learning (SSL) algorithm presented in Zhu et al. (2003) originally designed for graph-based SSL. The second one, also inspired from graph-based SSL, is to use a regression support vector machine using the Moore-Penrose pseudoinverse $\Delta^\dagger$ of the matrix Δ as a kernel. In the experiments, we consider real data sets of multiplayer games (double tennis and online games) and apply our method in a batch setting. We get very competitive results with specialized algorithms such as Elo duelling (see Elo, 1978) and TrueSkill (see Herbrich et al., 2006).

The above minimization problem can be solved by graph-based SSL algorithms because its formulation is similar to the case of graphs where Δ is a graph Laplacian. Here, the positive semidefinite matrix Δ expresses conditions on the players skills based on the game outcomes, where each game can be considered as a binary relation between two sets of players. Driven by these remarks, we introduce hypernode graphs as a finite model for relations between sets and we show that our matrix Δ is the Laplacian of such an hypernode graph.

We define a *hypernode graph* to be a set of hyperedges. A hyperedge is a pair of disjoint hypernodes, where a hypernode is a set of nodes. Every node of a hypernode is given a non negative real-valued weight. Roughly speaking, a hypernode models a group, a hyperedge models a relation between two groups, and individual weights correspond to the contribution of each individual to the relation between the two groups. An example of hypernode graph is shown in Figure 1. Hypernode graphs generalize over graphs because there is a one to one correspondence between undirected graphs and hypernode graphs in which all hypernodes are singleton sets.

We consider real-valued node functions over hypernode graphs and we assume an additive model where the hypernode valuation is the linear combination of its node valuations. We assume that connected hypernodes tend to have similar values. For real-valued node functions over discrete structures, the gradient is a discrete derivative that allows to mea-

sure such a similarity. Therefore, we define a gradient for hypernode graphs and we denote by G the gradient matrix of a hypernode graph $\mathbf{h}$. An example is shown in Figure 1. We define the Laplacian Δ of $\mathbf{h}$ to be $\Delta = G^T G$. Then, the quantity $f^T \Delta f$ measures the smoothness of a real-valued node function f on the hypernode graph $\mathbf{h}$. In other words, $f^T \Delta f$ is small when connected hypernodes have close values. Then, we show that a set of multiplayer games can be modeled by a hypernode graph and that our algorithm for the skill rating problem is finding a real-valued node function s minimizing $\Omega(s) = s^T \Delta s$, where Δ is the Laplacian of the constructed hypernode graph, under constraints on the s-values for game outcomes.

One question raised by the introduction of our new framework is whether problems on hypernode graphs can be solved using graphs. Indeed, for hypergraphs (a generalization of graphs where a hyperedge is a set of nodes), it has been proved in Agarwal et al. (2006) that the hypergraph Laplacians introduced at this time (other hypergraph Laplacians have been defined since (see Chan et al., 2018, and references within)), do not extend graph Laplacians because learning based on a hypergraph Laplacian can be proved equivalent to learning on a graph Laplacian using an adequate graph construction (such as clique expansion or star expansion). This is not the case for hypernode graphs. Indeed, we show that no graph construction allows to define a set of smooth functions over the graph equal to the set of smooth functions over a given hypernode graph. Thus, we state that hypernode graph Laplacians strictly generalize over graph Laplacians.

It can be noted that the class of hypernode graph Laplacians strictly contains the class of graph Laplacians and also contains the class of graph kernels (the Moore-Penrose pseudoinverse of a graph Laplacian). A convex linear combination of graph kernels (used in multiple graph kernel learning as in Argyriou et al. (2005)) is not, in general, a graph kernel but we prove that it is a hypernode graph kernel (the Moore-Penrose pseudoinverse of a hypernode graph Laplacian). We conjecture that the class of hypernode graph Laplacians is the smallest class closed by convex linear combinations which contains all graph kernels.

We also show that, for every hypernode graph, we can construct a signed graph with adjacency matrix W and degree matrix D such that $D-W$ is the Laplacian of the hypernode graph. This shows that positive relations between groups can lead to negative relations between individuals. Let us recall that, for arbitrary signed graphs, the matrix $D-W$ can be indefinite. From a spectral theory perspective, hypernode graphs correspond to the class of signed graphs such that $D-W$ is positive semidefinite introduced by Koren et al. (2002) for the problem of drawing graphs. Last, we study how to define a distance between nodes in hypernode graphs and we leave open questions on connected components and random walks in hypernode graphs.

2. Skill Rating for Multiplayer Games

We consider competitive games between two teams where each team is composed of an arbitrary number of players. A first objective is to compute the skills of individual players from a batch of games with their outcomes. A second objective is to predict a game outcome for a new game. In this section, we show how to model a batch of games and to infer player skills that allow to predict game outcomes of new games.

2.1 Skill Rating as an Optimization Problem

Let us consider a set of players $P = \{1, \ldots, n\}$ and a set of games $\Gamma = \{1, \ldots, m\}$ together with their respective game outcome. Each game is between two teams of an arbitrary number of players. Let us also consider that a player i contributes to a game γ with a positive real value $c_\gamma(i)$. Let us assume that each player has a skill $s(i)$, we suppose an additive model as in Herbrich et al. (2006) stating that the skill of a team is the weighted sum of the skills of the players in the team. More formally, given two teams of players $A = \{a_1, a_2, \ldots, a_\ell\}$ and $B = \{b_1, b_2, \ldots, b_k\}$ playing game γ, then A is predicted to be the winner if

$$\sum_{i=1}^{\ell} c_\gamma(a_i)s(a_i) > \sum_{i=1}^{k} c_\gamma(b_i)s(b_i) \ . \tag{1}$$

Equivalently, one can rewrite this inequality by introducing a positive real number o_γ on the right hand side such that

$$\sum_{i=1}^{\ell} c_\gamma(a_i)s(a_i) = o_\gamma + \sum_{i=1}^{k} c_\gamma(b_i)s(b_i) \ , \tag{2}$$

where the real number o_γ quantifies the game outcome. In the case of a draw, there is an equality between the team skills, thus the game outcome o_γ is set to 0. Given a set of games with their respective outcome, the goal is to infer a skill rating function $s \in \mathbb{R}^n$ that respects as much as possible Equation (2) for every game. We define the cost of a game γ with outcome o_γ for a skill function s by

$$C_\gamma(s) = \| \sum_{i=1}^{\ell} c_\gamma(a_i)s(a_i) - \big(o_\gamma + \sum_{i=1}^{k} c_\gamma(b_i)s(b_i)\big) \|^2 \ .$$

Consequently, given a set of games Γ and the corresponding game outcomes, the goal is to find a skill rating function s^* that minimizes the sum of the different costs:

$$s^* = \arg\min_s \sum_{\gamma \in \Gamma} C_\gamma(s) \ . \tag{3}$$

We now reformulate this problem as a SSL problem over a positive semidefinite matrix. For this, let us define the matrix $G \in \mathbb{R}^{m \times (n+m)}$, where n is the number of players and m the number of games, as follows. Let $\gamma \in \Gamma$ be a game between teams A and B where A is supposed to be the winner, the γ-th row of G models game γ and is defined by: for $1 \leq i \leq n$, $G_{\gamma,i} = c_\gamma(a_i)$ if $a_i \in A$, $G_{\gamma,i} = -c_\gamma(a_i)$ if $a_i \in B$, and 0 otherwise (the player i is not involved in the game γ); and, for $1 \leq \gamma' \leq m$, $G_{i,n+\gamma'} = -1$ for $\gamma = \gamma'$ and 0 otherwise.

Then, it is easy to show that the skill rating problem (3) is equivalent to find a real-valued node function $s \in \mathbb{R}^{(n+m)}$ solution of

$$\begin{array}{ll} \underset{s}{\text{minimize}} & s^T \Delta s \\ \text{subject to} & \text{for every game } \gamma \text{ with outcome } o_\gamma, \ \ s(n+\gamma) = o_\gamma \end{array} \tag{4}$$

where $\Delta = G^T G$. We can note that the matrix Δ is positive semidefinite.

2.2 Regularizing the Optimization Problem

When the number of games is small, player skills can be defined independently while satisfying the constraints and it will be irrelevant to compare them. In order to solve this issue, we introduce in Equation 4 a regularization term based on the standard deviation $\sigma(s_p)$ of players skills where $s_p = (s(1), \ldots, s(n))$. This allows us to reduce the spread of the player skills and leads us to find a real valued node function $s \in \mathbb{R}^{(n+m)}$ solution of

$$\begin{array}{ll} \underset{s}{\text{minimize}} & s^T \Delta s + \mu \sigma(s_p)^2 \\ \text{subject to} & \text{for every game } \gamma \text{ with outcome } o_\gamma, \;\; s(n+\gamma) = o_\gamma \end{array} \tag{5}$$

where μ is a regularization parameter. That is, we control the spread of s_p avoiding extreme values for players involved in a small number of games. In order to apply SSL algorithms over a positive semidefinite matrix, we propose to rewrite Problem 5.

First, let us recall that if $\overline{s}$ is the mean of the player skills vector s_p, then, for all $q \in \mathbb{R}$, we have $\sigma(s_p)^2 = \frac{1}{n}\sum_{i=1}^{n}(s(i)-\overline{s})^2 \leq \frac{1}{n}\sum_{i=1}^{n}(s(i)-q)^2$. Thus, Problem 5 is equivalent to

$$\begin{array}{ll} \underset{s\in\mathbb{R}^{(n+m)}, q\in\mathbb{R}}{\text{minimize}} & s^T \Delta s + \dfrac{\mu}{n}\displaystyle\sum_{i=1}^{n}(s(i)-q)^2 \\ \text{subject to} & \text{for every game } \gamma \text{ with outcome } o_\gamma, \;\; s(n+\gamma) = o_\gamma \end{array} \tag{6}$$

We now define the matrix $G_\mu \in \mathbb{R}^{(m+n)\times(n+m+1)}$. The first m rows of G_μ are the rows of G with an additional null column. The last n rows of G_μ are defined by, for every $1 \leq i \leq n$, $G_{\mu,(m+i,i)} = -\sqrt{\frac{\mu}{n}}$, $G_{\mu,(m+i,n+m+1)} = \sqrt{\frac{\mu}{n}}$, and 0 otherwise. Note that the last n rows of G_μ allow to compute $\frac{\mu}{n}\sum_{i=1}^{n}(s(i)-q)^2$ where $s(n+m+1) = q$ when computing $s^T G_\mu^T G_\mu s$. Thus Problem 6 is equivalent to finding a real-valued node function $s \in \mathbb{R}^{(n+m+1)}$ solution of

$$\begin{array}{ll} \underset{s}{\text{minimize}} & s^T \Delta_\mu s \\ \text{subject to} & \text{for every game } \gamma \text{ with outcome } o_\gamma, \;\; s(n+\gamma) = o_\gamma \end{array} \tag{7}$$

where $\Delta_\mu = G_\mu^T G_\mu$ is positive semidefinite.

Note that, assuming unitary contributions and a Gaussian distribution for skill ratings, we can prove that the value of μ/n should have the same order of magnitude than the average number of games played by a player. Indeed, this will allow us to obtain close expected values for the two terms $s^T \Delta s$ and $\mu\sigma(s_p)^2$. We follow this guideline in the experiments presented below.

2.3 Inferring Skill Ratings and Predicting Game Outcomes

Problem 7 can be viewed as a SSL problem because the question is to predict player skills (values $s(i)$ for $1 \leq i \leq n$) from known outcomes (values $s(n+\gamma)$ for $1 \leq \gamma \leq m$). We propose two algorithms:

H-ZGL: as the matrix Δ_μ is positive semidefinite, we can use the SSL algorithm presented in Zhu et al. (2003). This algorithm was originally designed for graphs and solves exactly Problem 7 by putting hard constraints on the outcome values. It should be noted that the algorithm is parameter free.

H-SVR: another approach to solve the SSL problem is to use a regression algorithm. For this, we consider the Moore-Penrose pseudoinverse $\Delta_\mu^\dagger$ of the matrix Δ_μ, we train a regression support vector machine and predict player skills. The main parameter is the soft margin parameter.

Given the player skills, we can deduce team skills, and finally predict game outcomes for new games. For this, we suppose that we are given a training set Γ_l of games with known outcomes together with a test set Γ_u of games for which game outcomes are hidden. The goal is to predict game outcomes for the test set Γ_u. We follow the following protocol.

Algorithm 1 Predicting game outcomes

Input: a training set Γ_l of games and a test set Γ_u of games

1: Build, as described in Sections 2.1 and 2.2, the matrix Δ_μ using game set Γ_l
2: Compute an optimal skill rating s^* using H-ZGL or H-SVR for Problem 7
3: Compute the mean skill $\bar{s}$ among players in Γ_l
4: **for** each game in Γ_u **do**
5: Assign skills given by s^* to players involved in Γ_l, and $\bar{s}$ otherwise
6: Compare the team skills and predict the winner according to Equation 1
7: **end for**

2.4 Experiments

In this section, we report experimental results for the inference of player skills and the prediction of game outcomes for different data sets in the batch setting described above. Note that other works have considered the online setting as in Herbrich et al. (2006) or Elo (1978).

Tennis Doubles

We consider a data set of tennis doubles collected between January 2009 and September 2011 from ATP tournaments (World Tour, Challengers and Futures). Tennis doubles are played by two teams of two players. Each game has a winner (no draw is allowed). A game is played in two or three winning sets. The final score corresponds to the number of sets won by each team during the game. The data set Γ consists in 10028 games with $n = 1834$ players.

Along the experimental process, given a proportion ρ, we draw randomly a training set Γ_l of size $\rho\%$ of the number of games in Γ. The remaining games define the test set Γ_u. We present in Figure 2 several statistics related to this process on the Tennis data set. First, it can be noticed that many players have played a small number of games. Second, when the number of games in the training set is small, half of the players are involved in games in the test set. Therefore, the skill rating problem and the game outcome prediction problem become far more difficult to solve when few games are used for learning.

The experimental setting is defined as follows. Let ρ be a proportion varying from 10% to 90% by 10%. For every value of ρ, we repeat ten times: draw a random training set Γ_l of size $\rho\%$ of the number of games in Γ; let the remaining games define the test set Γ_u; infer player skills and predict game outcomes following Algorithm 1. We set all player

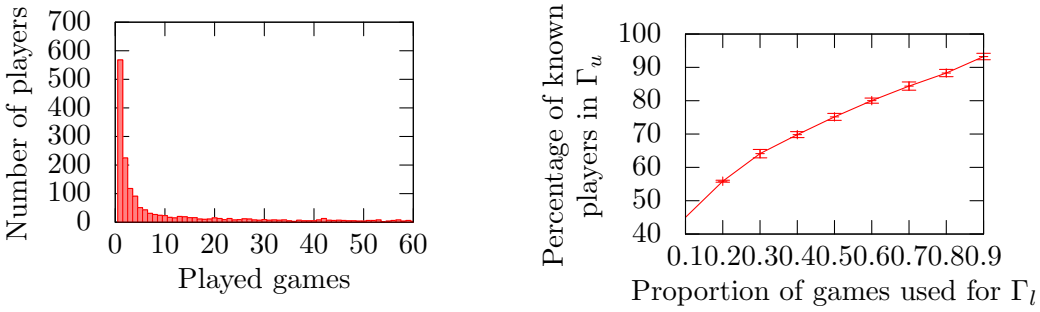


Figure 2: [*left*] Distribution of the number of players against the number of played games; [*right*] Average percentage of players in Γ_u which are involved in some game in Γ_l

contributions to 1. The game outcomes are set to be the difference between the number of sets won by the two teams, thus the possible outcome values are 1, 2 or 3. Note that, as there are only three outcome values, we can reduce the dimension of the matrix G from $n \times (n+m)$ where m is the number of games to $n \times (n+3)$. Thus the matrix Δ_μ is a square matrix of dimension $n + 3 + 1 = 1838$. As announced above, we choose the value of μ so that μ/n is close to the average number of games played by a player in the training set. The experiments presented in this section use $\mu/n = 16$.

We report in Figure 3 prediction results using: Algorithm 1 with H-ZGL, Algorithm 1 with H-SVR, Elo Duelling, and TrueSkill.[1] It can be noted that Elo Duelling performs poorly but Elo was designed for one against one games. It should be noted that we have done similar experiments for tennis singles. The results are not reported here but they show that Elo Duelling and TrueSkill obtain similar results but are outperformed by H-ZGL and H-SVR. For the data set of tennis doubles, it must be noted that H-ZGL and H-SVR outperform TrueSkill for a small number of training games. Also, H-ZGL outperforms TrueSkill whatever is the number of training games. On this set of experiments H-ZGL outperforms H-SVR but it should be noted that H-ZGL is parameter free and that we use H-SVR with the default soft margin parameter value.

Xbox Title Halo2

The Halo2 data set was generated by Bungie Studio during the beta testing of the XBox title Halo2. It has been notably used in Herbrich et al. (2006) to evaluate the performance of the TrueSkill algorithm. We consider the *Small Teams* data set with $n = 4992$ players and 27536 games opposing up to 12 players in two teams which can have a different size. Each game can result in a draw or a win of one of the two teams. The proportion of draws is 22.8%. As reported in Herbrich et al. (2006), the prediction of draws is challenging and

1. TrueSkill and Elo implementations are from Hamilton (2012). Results were double-checked using Lee (2013b) and Lee (2013a). Parameters of Elo and TrueSkill are the default parameters of Hamilton (2012) ($K = 32$ for Elo, $\mu_0 = 25, \beta = 12.5, \sigma = 8.33$ and $\tau = 0.25$ for TrueSkill).

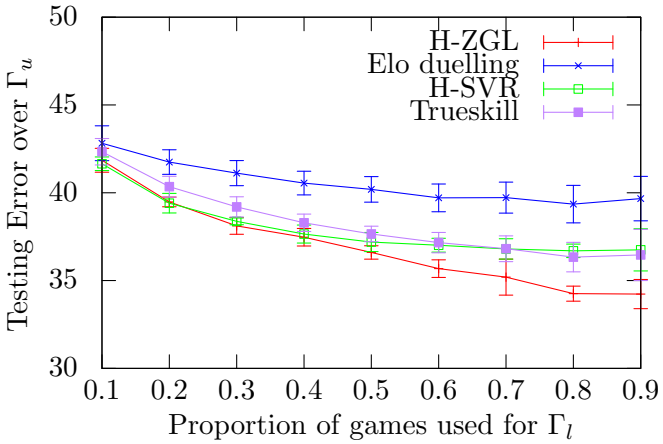


Figure 3: Predictive error for Double Tennis data set

it should be noted that TrueSkill and our algorithm fail to outperform a random guess for the prediction of draw.

We consider the same experimental process. For the construction of the hypernode graph in Algorithm 1, we fix all players contributions in games to 1 and we again set the parameter value μ/n to 16. In the optimization problem (7), the game outcomes are set to 1 when the game has a winner and 0 otherwise because game scores vary depending on the type of game. The matrix Δ_μ has dimension $n + 2 + 1 = 4995$. We again compare the skill rating algorithms H-ZGL, H-SVR, Elo Duelling and TrueSkill. The number of prediction errors over game outcomes is computed assuming that a draw can be regarded as half a win, half a loss as in Lasek et al. (2013). We present the experimental results in Figure 4. For a proportion of 10% of games in the training set, H-ZGL, H-SVR and TrueSkill give similar results while with larger training sets, our hypernode graph learning algorithms outperform TrueSkill. Contrary to the previous experiment, H-SVR (with default parameter value) outperforms H-ZGL.

3. Hypernode Graphs

In the previous section, we consider the skill rating problem as a SSL problem and we express it as a minimization problem using a positive semidefinite matrix Δ which expresses conditions on the players skills based on the game outcomes. This allows us to use graph-based SSL algorithms. As games can be considered as relations between sets of players, there is a discrete structure underlying the skill-rating problem. The existing discrete structure and the similarity with graph-based SSL motivate the work presented in this section. Indeed, we now introduce hypernode graphs as a finite model for relations between sets, we show that our matrix Δ is the Laplacian of such an hypernode graph, and we show that Problem 7 expresses the smoothness of a real-valued node function over the hypernode graph with some constraints on node values.

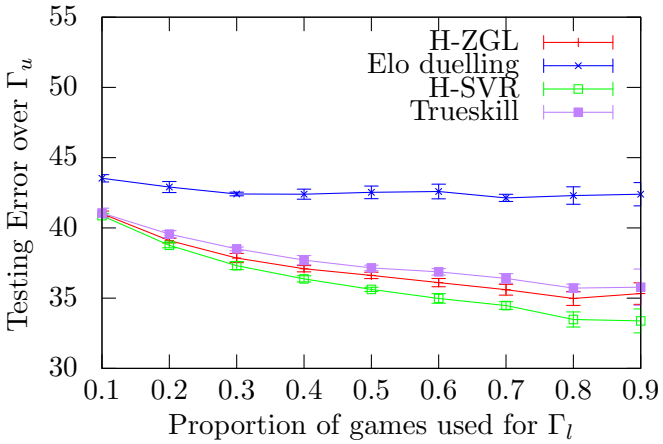


Figure 4: Predictive error for Halo2 data set

3.1 Introducing Hypernode Graphs

The following definition is our contribution to the modeling of binary relationships between sets of entities.

Definition 1 *A* hypernode graph $\mathbf{h} = (V, H)$ *is a set of nodes* V *and a set of hyperedges* H*. Each hyperedge* $h \in H$ *is an unordered pair* $\{s_h, t_h\}$ *of two non empty and disjoint hypernodes (a hypernode is a subset of* V*). Each hyperedge* $h \in H$ *has a weight function* w_h *mapping every node* i *in* $s_h \cup t_h$ *to a positive real number* $w_h(i)$ *(for* $i \notin s_h \cup t_h$*, we define* $w_h(i) = 0$*). Each weight function* w_h *must satisfy the* Equilibrium Condition *defined by*

$$\sum_{i \in t_h} \sqrt{w_h(i)} = \sum_{i \in s_h} \sqrt{w_h(i)} \ . \tag{8}$$

We denote by n the number of nodes. We say that a node i *belongs to* a hyperedge h, that we denote by $i \in h$, if $w_h(i) \neq 0$. We define the degree of a node i by $\deg(i) = \sum_h w_h(i)$. The degree of a node is positive when it belongs to at least one hyperedge. We define the diagonal degree matrix by $D = \mathrm{diag}(\deg(1), \ldots, \deg(n))$.

An example of hypernode graph is shown in Figure 1. The hyperedge h_{blue} links the sets $\{1, 2, 3\}$ and $\{4, 5\}$. The weights of h_{blue} satisfy the Equilibrium condition: $\sqrt{1/2} + \sqrt{1/2} + \sqrt{2} = \sqrt{2} + \sqrt{2}$. The hyperedge h_{black} links the sets $\{4, 5\}$ and $\{1, 6\}$ and satisfies the Equilibrium condition. The hyperedge h_{red} links the two singletons $\{3\}$ and $\{6\}$ with equal weights, thus the hyperedge h_{red} can be viewed as an edge with edge weight $1/3$.

As noted in the previous example, when a hyperedge h is an unordered pair $\{\{i\}, \{j\}\}$, the Equilibrium Condition states that the weights $w_h(i)$ and $w_h(j)$ are equal. Therefore, every hypernode graph such that all hyperedges are unordered pairs of singleton nodes can be viewed as a graph with adjacency matrix W defined by $W_{i,j} = W_{j,i} = w_h(i) = w_h(j)$ for every hyperedge $\{\{i\}, \{j\}\}$, and 0 otherwise. Conversely, every graph can be viewed as a hypernode graph where every hypernode is a singleton set.

3.2 Laplacians and Kernels of Hypernode Graphs

We are interested in evaluating real-valued functions on nodes and on hypernodes. We assume a weighted linear model such that any real-valued node function f can be extended to a hypernode u of a hyperedge h by $f(u) = \sum_{i \in u} f(i)\sqrt{w_h(i)}$. Moreover, we will consider an homophilic assumption stating that connected hypernodes tend to have similar values. In order to implement this assumption, we introduce a gradient for hypernode graphs by

Definition 2 *Let $\mathbf{h} = (V, H)$ be a hypernode graph and f be a real-valued node function, the* (hypernode graph) unnormalized gradient *of $\mathbf{h}$ is a linear application, denoted by* grad*, that maps every real-valued node function f into a real-valued hyperedge function* $\mathrm{grad}(f)$ *defined, for every $h = \{s_h, t_h\}$ in H, by*

$$\mathrm{grad}(f)(h) = f(t_h) - f(s_h) = \sum_{i \in t_h} f(i)\sqrt{w_h(i)} - \sum_{i \in s_h} f(i)\sqrt{w_h(i)} \ , \tag{9}$$

where an arbitrary orientation of the hyperedges has been chosen. We denote by $G \in \mathbb{R}^{p \times n}$ the matrix of grad*. The square $n \times n$ real-valued matrix $\Delta = G^T G$ is defined to be the* unnormalized Laplacian *of the hypernode graph $\mathbf{h}$.*

As suggested by an anonymous reviewer, a hypernode graph could be defined by a gradient matrix as shown in Figure 1.

It should be noted that the Laplacian Δ does not depend on the arbitrary orientation of the hyperedges chosen for defining the gradient. Also, by definition, the Laplacian Δ is positive semidefinite. Moreover, because of the definition of the additive model and because of the Equilibrium Condition 8, the gradient of every constant node function is the zero-valued hyperedge function. Consequently, we have $\mathbf{1} \in \mathrm{Null}(\Delta)$ where $\mathrm{Null}(\Delta)$ is the null space of Δ. Last, it could be noted that, when the hypernode graph is a graph (all hypernodes are singleton sets), then the hypernode graph Laplacian is equal to the unnormalized graph Laplacian.

The Laplacian allows to define the *smoothness* of a real-valued node function f over a hypernode graph $\mathbf{h}$ to be $\Omega(f) = f^T \Delta f$.

Last, we define the *hypernode graph kernel* of a hypernode graph $\mathbf{h}$ to be the Moore-Penrose pseudoinverse $\Delta^\dagger$ of the hypernode graph Laplacian Δ.

Because a hypernode graph Laplacian is positive semidefinite, we can leverage the spectral learning algorithms defined in Von Luxburg (2007), Zhou et al. (2005), and Zhu et al. (2003) from graphs to hypernode graphs.

3.3 Modeling the Skill Rating Problem with Hypernode Graphs

In this section, we show how to model a batch of games by a hypernode graph and how the problems (4) and (7) translate on the constructed hypernode graphs.

We consider a set of m games as in Section 2.1. The construction mimics the construction of the previous section with the additional trick to add a new node in order to satisfy the Equilibrium condition 8 for hyperedges. Indeed, we consider $n + m + 1$ nodes, n nodes for the players, m nodes for the outcomes and one lazy node. We can define, for every game γ a hyperedge h as follows.

1. The players of A define one of the two hypernodes of h, the weight of a player node i is defined to be $c(i)^2$;
2. do the same construction for the second team B;
3. add the *outcome node* associated with γ to the set of nodes of the losing team, its weight is set to 1;
4. add the *lazy node* to the set of nodes corresponding to the winning team, its weight is chosen in order to ensure the Equilibrium condition for the hyperedge h.

This defines the hypernode graph $\mathbf{h} = (V, H)$. Then, it is easy to show that Problem 4 is equivalent to minimizing $s^T \Delta s$ where Δ is the Laplacian of $\mathbf{h}$ with constraints for game outcomes and adding the additional constraint $s(n + m + 1) = 0$ for the lazy node.

For the regularization term, we define the hypernode graph $\mathbf{h}_\mu$ obtained from the hypernode graph $\mathbf{h}$ by adding a regularizer node and adding n hyperedges between every player node and the regularizer node R with (node) weights set to μ/n. Then, it is easy to show that Problem 7 is equivalent to minimizing $s^T \Delta_\mu s$ where Δ_μ is the Laplacian of $\mathbf{h}_\mu$ with constraints for game outcomes and adding the additional constraint $s(n + m + 1) = 0$ for the lazy node.

Then as said before, we can leverage graph-based SSL algorithms and consider, for instance, the algorithms H-ZGL and H-SVR of Section 2.3.

3.4 The Class of Hypernode Graphs Laplacians

Proposition 3 *The class of hypernode graph Laplacians is the class of symmetric positive semidefinite real-valued matrices M such that $\mathbf{1} \in \mathrm{Null}(M)$ (the null space of M).*

Proof As a consequence of Definition 2, every hypernode graph Laplacian M is a symmetric positive semidefinite real-valued matrix such that $\mathbf{1} \in \mathrm{Null}(M)$. Conversely, let us consider the following algorithm

Algorithm 2 Construction of a hypernode graph

Input: $M \in \mathbb{R}^{n \times n}$ symmetric positive semidefinite such that $\mathbf{1} \in \mathrm{Null}(M)$
Set V to $\{1, \ldots, n\}$; set H to $\emptyset$
Compute a square root decomposition $M = G^T G$
for each row $\mathbf{r}$ of G **do**
  Create a new hyperedge $h = \{s_h, t_h\}$ with $s_h = t_h = \emptyset$
  for each node $i \in V$ **do**
    Define $w_h(i) = \mathbf{r}(i)^2$
    if $\mathbf{r}(i) < 0$ **then** Add i to s_h **else if** $\mathbf{r}(i) > 0$ Add i to t_h **end if**
  end for
  Add h to H
end for
return The hypernode graph $\mathbf{h} = (V, H)$

From a square root decomposition $G^T G$ of M, for each line of G, we define a hyperedge $h = \{s_h, t_h\}$ where nodes in s_h (respectively in t_h) have positive (respectively negative)

values in the line, and node weights are chosen to be squares of values in the line. Then, the Equilibrium condition 8 is satisfied because $\mathbf{1} \in \text{Null}(M)$. And it is easy to check that G is a gradient matrix of $\mathbf{h}$ and, consequently, that $M = G^T G$ is the Laplacian of $\mathbf{h}$. ■

As a consequence of Proposition 3, the class of hypernode graph Laplacians

- is closed under convex linear combination,
- is closed under pseudoinverse,
- and strictly contains the class of graph Laplacians and the class of graph kernels (a graph kernel is the Moore-Penrose pseudoinverse of a graph Laplacian).

Linear combinations of graph kernels have been used for SSL (see for instance Argyriou et al., 2005). Another consequence of Proposition 3 is that any convex linear combination of graph kernels is a hypernode graph kernel (but not necessarily a graph kernel). To conclude this section, we propose

Conjecture: The class of hypernode graph Laplacians is the smallest class closed by convex linear combination which contains all graph kernels.

The main point is to prove that every hypernode graph Laplacian is a convex linear combination of graph kernels which is difficult because a graph kernel has no simple analytic form.

3.5 Hypernode Graph Laplacians Strictly Generalize Graph Laplacians

As said above, the class of hypernode graph Laplacians strictly contains the class of graph Laplacians. But this does not allow us to claim that hypernode graph Laplacians provide a gain of expressiveness over graph Laplacians. Indeed, it could be the case that through a well chosen graph construction, the set of smooth functions f defined on hypernode graphs by $f^T \Delta f = 0$ could be made to coincide exactly with the set of smooth functions defined on graphs. The class of hypergraph Laplacians has been studied in that perspective of expressiveness. Indeed, it has been shown in Agarwal et al. (2006) that hypergraph Laplacians such as the Δ_{B} from Bolla (1993), Δ_{R} from Rodríguez (2003) and Δ_{ZHS} from Zhou et al. (2007) can be defined as (restrictions of) graph Laplacians using a graph construction such as the clique expansion (where each hyperedge is replaced by a clique graph with uniform weights) or the star expansion (where, for every hyperedge, a new node is added and is linked with all the nodes in the hyperedge).

While one can think of similar constructions for the case of hypernode graphs, we prove that there does not exist a (finite) graph expansion of a hypernode graph which defines the same set of smooth functions over the original set of nodes. The proof is based on the very simple hypernode graph $\mathbf{h}$ shown in Figure 5. Note that a function is smooth over $\mathbf{h}$ if and only if it satisfies

$$(f(1) + f(2) - f(3) - f(4))^2 = 0 \ . \qquad (\mathcal{C})$$

We show that

Proposition 4 *There do not exist a finite graph* $\mathbf{g}$ *whose node set contains* $\{1, 2, 3, 4\}$ *and that satisfies the conditions:*

1. *All the smooth functions on* $\mathbf{g}$ *satisfy* $(\mathcal{C})$.
2. *Any function that satisfy* $(\mathcal{C})$ *can be extended to a smooth function on* $\mathbf{g}$.

Proof Let us define $S = \{1, 3\}$ and denote by Δ the Laplacian matrix of $\mathbf{h}$. The indicator vector $\mathbf{1}_S$ is in Null(Δ) and, thus, define a smooth function on $\mathbf{h}$. Let us consider a graph $\mathbf{g}_e = (N_e, E_e)$ whose nodeset N_e contains the nodeset $N = \{1, 2, 3, 4\}$ and that satisfies the two conditions presented in Proposition 4. We denote by Δ_e the Laplacian matrix of $\mathbf{g}_e$. Because of the second condition, the function $\mathbf{1}_S$ can be extended to a smooth function f_e on $\mathbf{g}_e$.

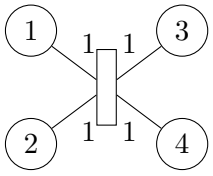
Since $\mathbf{g}_e$ is a graph, we know that Null(Δ_e) is spanned by the indicator vectors of the connected components of $\mathbf{g}_e$ (see for instance Von Luxburg, 2007). Since $f_e(1) = \mathbf{1}_S(1) \neq f_e(2) = \mathbf{1}_S(2)$, 1 and 2 must be in different components in $\mathbf{g}_e$. Note that the same holds with $f_e(1) \neq f_e(4)$, $f_e(3) \neq f_e(2)$ and $f_e(3) \neq f_e(4)$. By following a similar reasoning with $S = \{1, 4\}$, we eventually deduce that the four original nodes 1, 2, 3 and 4 must be in distinct components in $\mathbf{g}_e$.

Let us now consider the components C of $\mathbf{g}_e$ that contains the node 1. As stated above, we have necessarily $f'_e(2) = f'_e(3) = f'_e(4) = 0$. The indicator function f'_e of C is smooth on $\mathbf{g}_e$ but we have

$$(f'_e(1) + f'_e(2) - f'_e(3) - f'_e(4))^2 = (1 + 0 - 0 - 0)^2 \neq 0 \ .$$

Consequently, f'_e does not satisfy $(\mathcal{C})$, which violates the first condition and concludes the proof. ■

The first condition in Proposition 4 ensures that smooth functions on $\mathbf{g}$ are related to smooth functions over the original hypernode graph $\mathbf{h}$. The second condition in Proposition 4 ensures that smooth functions on the original hypernode graph can be defined from smooth functions over expanded finite graphs. The combination of these two conditions ensure that the problem of finding smooth functions on $\mathbf{h}$ can be rephrased as a graph problem (finding the smooth functions on $\mathbf{g}$ and compute their restrictions to $\{1, 2, 3, 4\}$). The fact that no graph $\mathbf{g}$ can satisfy both conditions at the same time for a simple hypernode graph $\mathbf{h}$ shows that smoothness properties over hypernode graphs can not be defined over undirected expanded graphs.

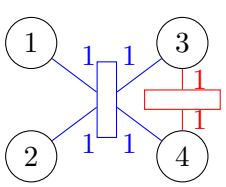


$$\Delta = \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$$

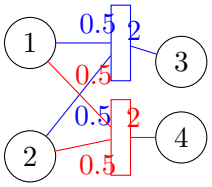
Figure 5: A hypernode graph $\mathbf{h}$ and its Laplacian Δ

3.6 L-equivalent Hypernode Graphs and Signed Graphs

Let us consider a symmetric positive semidefinite real-valued matrix M such that $\mathbf{1} \in$ Null(M), M is the Laplacian of a hypernode graph by Proposition 3. But, as the square



$$G = \begin{pmatrix} 1 & 1 & -1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$



$$G = \begin{pmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \sqrt{2} & 0 \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & \sqrt{2} \end{pmatrix}$$

Figure 6: two L-equivalent hypernode graphs with their respective gradient which are square root of the Laplacian Δ shown in Figure 7.

$$\Delta = \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{pmatrix} ; W = \begin{pmatrix} 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} ; D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

Figure 7: Laplacian matrix, pairwise weight matrix, and degree matrix for the two L-equivalent hypernode graphs shown in Figure 6.

root decomposition is not unique, there are several hypernode graphs with the same Laplacian that we called *L-equivalent*. Examples of L-equivalent hypernode graphs are given in Figure 6. In order to study the L-equivalent relation and to decide whether a hypernode graph is L-equivalent to a graph, we first show that hypernode graphs are related to signed graphs.

It is known that the Laplacian matrix Δ of a graph can be written $D - W$ where W is the adjacency matrix of the graph and D is the corresponding degree matrix. Let us consider a hypernode graph $\mathbf{h}$ with Laplacian Δ, we define the *pairwise weight matrix* W of $\mathbf{h}$ by $W_{i,j} = -\Delta_{i,j}$ if $i \neq j$, and 0 otherwise. Note that the pairwise weight matrix coincides with the classic adjacency matrix when the hypernode graph is a graph. Let us define the degree of a node i to be $\deg(i) = \sum_{j \in V} W_{i,j}$ and let us denote by D the diagonal matrix of all $\deg(i)$. Then, because of the property $\mathbf{1} \in \text{Null}(M)$ in Proposition 3, it is immediate that, for every i, $\Delta_{i,i} = \sum_{j \in V} W_{i,j}$. As a consequence, we have

Proposition 5 *Let $\mathbf{h} = (V, H)$ be a hypernode graph, let W be the pairwise weight matrix of $\mathbf{h}$, and let D be the diagonal degree matrix of $\mathbf{h}$. Then, the Laplacian of $\mathbf{h}$ is $\Delta = D - W$.*

An example is shown in Figure 7. As a consequence of the above proposition, we can leverage the pairwise weight matrix to characterize L-equivalent hypernode graphs by

Proposition 6 *Two hypernode graphs are L-equivalent if and only if they have the same pairwise weight matrix. Two L-equivalent hypernode graphs have the same degree matrix.*

The pairwise weight matrix W contains in general negative weights and can be thus interpreted as the adjacency matrix of a signed graph. Therefore, we define the *reduced*

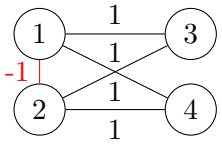


Figure 8: The reduced signed graph with adjacency matrix W of Figure 7 of the two L-equivalent hypernode graphs shown in Figure 6.

signed graph of a hypernode graph $\mathbf{h}$ to be the signed graph $\tilde{\mathbf{g}}$ with adjacency matrix W. An example is shown in Figure 8. Then, we can show that

- the reduced signed graph of a graph $\mathbf{g}$ (viewed as a hypernode graph) is the graph $\mathbf{g}$,
- a hypernode graph $\mathbf{h}$ is L-equivalent to a graph if and only if its reduced signed graph is a graph,
- a signed graph with adjacency matrix W is the reduced signed graph of a hypernode graph if and only if the matrix $D - W$ is positive semidefinite.

The definition of the pairwise weight matrix may seem ad hoc to mimic the definition of the Laplacian in the graph case. But, we can show that the pairwise weight matrix can be defined directly from the hypernode graph using the following formula

$$\forall i \neq j, \quad W_{i,j} = \sum_{h \in H} P(h,i,j)\sqrt{w_h(i)}\sqrt{w_h(j)} \ , \tag{10}$$

where $P(h,i,j) = 1$ if i and j belongs to two different ends of h, $P(h,i,j) = -1$ if i and j belongs to the same end of h, and 0 otherwise. The pairwise weight $W_{i,j}$ can be interpreted as a flow measure between node i and j in the hypergraph. It is computed as a sum over all the hyperedges involving nodes i and j. For every term in the sum, the quantity $\sqrt{w_h(i)}$ can be viewed as the cost of entering the hyperedge h at node i, the quantity $\sqrt{w_h(j)}$ as the cost of exiting the hyperedge h at node j, and $P(h,i,j)$ as a sign depending whether i and j are in the same end or in different ends of the hyperedge.

3.7 Defining a distance in Hypernode Graphs

We now study whether a distance can be defined between nodes of a hypernode graph. Let us consider a hypernode graph $\mathbf{h} = (V, H)$ with Laplacian Δ. Let $\Delta^\dagger$ be the Moore-Penrose pseudo-inverse of Δ, let us define d by, for every i, j in V,

$$d(i,j) = \left(\sqrt{\Delta^\dagger_{i,i} + \Delta^\dagger_{j,j} - 2\Delta^\dagger_{i,j}}\right) \ . \tag{11}$$

Because $\Delta^\dagger$ is symmetric positive semidefinite, we have

Proposition 7 *d defines a pseudometric on $\mathbf{h}$: it is positive, symmetric and satisfies the triangle inequality (for all i, j, k the inequality $d(i,j) \leq d(i,k) + d(k,j)$ holds).*

For the pseudometric d to be a distance, d should satisfy also the coincidence axiom: $d(i,j) = 0 \Rightarrow i = j$. This is not true in general as $d(1,2) = 0$ for the hypernode graph in Figure 5. The intuition is that the nodes 1 and 2 can not be distinguished because the smoothness condition is on the sum $f(1) + f(2)$. Nevertheless we can show that

Proposition 8 *When* $\text{Null}(\Delta) = \text{Span}(\mathbf{1})$*, d defines a metric (or distance) on* $\mathbf{h}$*.*

Let us recall that, in the graph case, Proposition 7 holds and that Proposition 8 holds when the graph is connected. By analogy with the case of graphs, the condition $\text{Null}(\Delta) = \text{Span}(\mathbf{1})$ can be viewed as an algebraic definition of a connected hypergraph. So far, we have not found an alternative algorithmic definition of connected components in hypernode graphs.

Finally, let us note that, in the graph case, for connected graphs, both d and d^2 are metrics while, for hypernode graphs, d^2 does not satisfy the triangle inequality even if $\text{Null}(\Delta) = \text{Span}(\mathbf{1})$. We provide in Ricatte et al. (2015) a formulation of d^2 in terms of diffusion potentials but do not find an interpretation of d^2 using random walks in hypernode graphs.

4. Conclusion

We have introduced a new model and new algorithms to learn from binary relations between groups. We showed that our method obtained state-of-the-art results for skill rating in multiplayer games. We are convinced that our method could be successfully applied to other games where players have different roles (such as team sports like basketball, american football or baseball) by modeling roles with weights. From a machine learning perspective, we believe that this model will open the way to solving new learning problems in networks when relations between sets of nodes are meaningful. From a fundamental perspective, our model is a strict generalization of graphs. Many theoretical studies remain to be done on the spectral theory of hypernode graphs. Note that we leave open some questions such as finding a characterization of the class of hypernode graphs, finding an algorithmic definition of connected components, and how to interpret random walks and distances on hypernode graphs. Also the study of directed hypernode graphs remains to be done from a machine learning perspective.

Acknowledgments

The authors are very grateful to the referees for their constructive comments and to the Editor David Blei for his unwavering support.

References

Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher Order Learning with Graphs. In *Proceedings of the 23rd International conference on Machine learning (ICML-06)*, pages 17–24, 2006.

Andreas Argyriou, Mark Herbster, and Massimiliano Pontil. Combining Graph Laplacians for Semi-Supervised Learning. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS-05)*, pages 67–74, 2005.

Marianna Bolla. Spectra, euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*, 117(1):19–39, 1993.

Hubert T.-H. Chan, Louis Anand, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)*, 65(3 - 15), 2018.

Arpad Emrick Elo. *The Rating of Chess Players, Past and Present.* Arco Publishing, 1978.

Scott Hamilton. PythonSkills: Implementation of the TrueSkill, Glicko and Elo Ranking Algorithms. `https://pypi.python.org/pypi/skills`, 2012.

Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill$^{\text{TM}}$: A Bayesian Skill Rating System. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS-06)*, pages 569–576, 2006.

Yeruda Koren, Liran Carmel, and David Harel. ACE: a fast multiscale eigenvectors computation for drawing huge graphs. In *IEEE Symposium on Information Visualization (INFOVIS-02)*, pages 137–144, 2002.

Jan Lasek, Zoltán Szlávik, and Sandjai Bhulai. The predictive power of ranking systems in association football. *International Journal of Applied Pattern Recognition*, 1(1):27–46, 2013.

Heungsub Lee. Python implementation of Elo: A rating system for chess tournaments. `https://pypi.python.org/pypi/elo/0.1.dev`, 2013a.

Heungsub Lee. Python implementation of TrueSkill: The video game rating system. `http://trueskill.org/`, 2013b.

Thomas Ricatte, Rémi Gilleron, and Marc Tommasi. Hypernode Graphs for Learning from Binary Relations between Groups in Networks. Research report, INRIA Lille, January 2015. URL `https://hal.inria.fr/hal-01247103`.

JA Rodríguez. On the Laplacian spectrum and walk-regular hypergraphs. *Linear and Multilinear Algebra*, 51(3):285–297, 2003.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International conference on Machine learning (ICML-05)*, pages 1036–1043, 2005.

Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems (NIPS-06)*, pages 1601–1608, 2007.

Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.