Università degli Studi di Padova

Department of Mathematics ‘Tullio Levi-Civita’

Master of Science in Data Science

Dynamic hierarchical ranking in directed networks: from static SpringRank to dynamic and bayesian models using variational techniques

Supervisor: Paolo Dai Pra

Department of Mathematics

Co-Supervisor: Caterina De Bacco

Max Planck Institute for Intelligent Systems

Student: Andrea Della Vecchia

N. 1154796

Academic Year 2018/2019

Contents

1. Introduction

Every time we deal with multiples entities interacting among themselves we can ask ourself what the outcome of these interactions will be. Animals fighting for leadership, teams and players facing each other in competitions, individuals endorsing other individuals are only a small example of contexts where we can assume that interactions and their outcomes are not simply random sequence of events but that there is some kind of hierarchy or structure in the system ruling the observations. This hidden ranking, characterizing all the entities involved in this kind of systems, is obviously not explicit, but its effects are shown in the asymmetric patterns we can observe in the data. Since dominant animals tend to assert themselves over less powerful subordinates, or stronger teams tend to beat weaker ones, there must be a score measuring some kind of skill for each subject that can be the fundamental explanatory variable needed to predict future outcomes and forecast what is about to happen. Besides the natural noise affecting real data and the partial randomness governing all the observations, what makes this problem even more challenging is that, in most of the more interesting applications, the ranking changes through time. Life cycle makes human beings and all animals in general experience youth, adulthood and old age. The same in sports, where injuries, changes in the team's components, physical form and other factors can upset the current balance of power. This variability in the ranking requires a different approach even while collecting the data. Imagine for example an individual occupying at the beginning the bottom of the ranking, then reaching the top and finally going back to the bottom again (like in the life cycle example youth-adulthood-old age introduced above). The only information about the number of successes or failures of this subject is not enough to infer the evolution of his rank properly. In fact, to be able to guess this parabolic behaviour we obviously need the additional piece of information involving the time stamps of the interactions. Without any time knowledge we would probably end up assigning the individual a medium position inside the ranking, losing completely the intuition of the different phases he has gone through, with evident impact in the forecasting performances.

In this thesis, extending the work done in [1], we introduce a physically-inspired model addressing the problem of dynamic hierarchy inference and edge predictions in directed networks. The key point in dynamic ranking is to find the right equilibrium between past history and new data available: on one hand we are tempted to use all the possible information we have in the past to extract robust scores with smooth trajectories through time, on the other we want to assign more value to present observations, in order to capture promptly variations in the hierarchy and adapt the score estimates. We deal in particular with data taken from sports, where the focus is mainly on the outcomes' prediction more than on the edge existence, since the fixed schedule of the matches is given from the beginning.
We show that our algorithm performs better than a variety of existing methods and that is way faster than many of them, specially compared with Bayesian methods such as TrueSkill and WHR.

In the second part we develop a Bayesian model, exploiting variational inference to approximate the true uncomputable posterior. Various modifications of this model, specific to the application, are presented, highlighting for all of them their particular strengths and weaknesses. After a theoretical discussion of some of the modern techniques adopted in this area, such as variational EM, mean field approximation, Black-Box variational inference and others, all this knowledge is applied to our problem and tested against real data.

2. The SpringRank model

SpringRank is a physically inspired model for hierarchy inference, edge prediction, and significance testing in directed networks. The model maps each directed edge to a directed spring between the nodes that it connects, and finds real-valued positions of the nodes that minimizes the total energy of these springs. Because this optimization problem requires only linear algebra, it can be solved for networks of millions of nodes and edges in seconds.

Interactions between N entities are represented as a weighted directed network, where A_{ij} is the number of interactions $i \to j$ suggesting that i is ranked above j. This allows both ordinal and cardinal input, including multiple interactions among pairs. For instance, A_{ij} could be the number of fights between i and j that i has won, or the number of times that j has endorsed i. Given the adjacency matrix A, our goal is to find a ranking of the nodes. To do so, the SpringRank model computes the optimal location of nodes in a hierarchy by imagining the network as a physical system. Specifically, each node i is embedded at a real-valued position or rank s_i, and each directed edge $i \to j$ becomes an oriented spring with a nonzero resting length and displacement $s_i - s_j$. Since we are free to rescale the latent space and the energy scale, we set the spring constant and the resting length to 1. Thus, the spring corresponding to an edge $i \to j$ has energy:

$$H_{ij} = \frac{1}{2}(s_i - s_j - 1)^2 \tag{2.1}$$

which is minimized when $s_i - s_j = 1$.

According to this model, the optimal rankings of the nodes are the ranks $\boldsymbol{s}^* = (s_1^*, ..., s_N^*)$ which minimize the total energy of the system given by the Hamiltonian

$$H(\boldsymbol{s}) = \sum_{i,j=1}^{N} A_{ij} H_{ij} = \frac{1}{2} \sum_{i,j=1}^{N} A_{ij}(s_i - s_j - 1)^2. \tag{2.2}$$

Since this Hamiltonian is convex in $\boldsymbol{s}$, we can find $\boldsymbol{s}^*$ by setting $\nabla H(\boldsymbol{s}) = 0$. Looking at the i-th component of the gradient we obtain:

$$\frac{\partial H}{\partial s_i} = \sum_j \left[A_{ij}\left(s_i - s_j - 1\right) - A_{ji}\left(s_j - s_i - 1\right)\right] = 0\,. \tag{2.3}$$

Let the weighted out-degree and in-degree be $d_i^{\text{out}} = \sum_j A_{ij}$ and $d_i^{\text{in}} = \sum_j A_{ji}$, respectively. Then Eq. can be written as

$$\left(d_i^{\text{out}} + d_i^{\text{in}}\right) s_i - \left(d_i^{\text{out}} - d_i^{\text{in}}\right) - \sum_j \left[A_{ij} + A_{ji}\right] s_j = 0\,. \tag{2.4}$$

We now write the system of N equations together by introducing the following matrix notation. Let $D^{\text{out}} = \text{diag}(d_1^{\text{out}}, \dots, d_N^{\text{out}})$ and $D^{\text{in}} = \text{diag}(d_1^{\text{in}}, \dots, d_N^{\text{in}})$ be diagonal matrices, let $\mathbf{1}$ be the N-dimensional vector of all ones. Then Eq becomes

$$\left[D^{\text{out}} + D^{\text{in}} - \left(A + A^T\right)\right] \boldsymbol{s} = \left[D^{\text{out}} - D^{\text{in}}\right] \mathbf{1}\,. \tag{2.5}$$

The matrix on the left side of (2.5) is not invertible. This is because H is translation-invariant: it depends only on the relative ranks $s_i - s_j$, so that if $s^* = \{s_i\}$ minimizes $H(s)$ then so does $\{s_i + a\}$ for any constant a. A way to break translation invariance is to introduce an "external field" $H_0(s_i) = \frac{1}{2}\alpha s_i^2$ affecting each node, so that the combined Hamiltonian is

$$H_\alpha(s) = H(s) + \frac{\alpha}{2}\sum_{i=1}^N s_i^2\,. \tag{2.6}$$

The field H_0 corresponds to a spring that attracts every node to the origin. We can think of this as imposing a Gaussian prior on the ranks, or as a regularization term that quadratically penalizes ranks with large absolute values. This version of the model has a single tunable parameter, namely the spring constant α. Since $H(\boldsymbol{s})$ scales with the total edge weight $M = \sum_{i,j} A_{ij}$ while $H_0(\boldsymbol{s})$ scales with N, for a fixed value of α this regularization becomes less relevant as networks become more dense and the average (weighted) degree M/N increases.

For $\alpha > 0$ there is a unique s^* that minimizes H_α, given by

$$\left[D^{\text{out}} + D^{\text{in}} - \left(A + A^T\right) + \alpha\mathbb{I}\right] s^* = \left[D^{\text{out}} - D^{\text{in}}\right] \mathbf{1}\,, \tag{2.7}$$

where $\mathbb{I}$ is the identity matrix.

3. Dynamic extensions to SpringRank

3.1 Related work

The estimation of rankings in a system from pairwise directed interactions is a fundamental problem in various contexts. Various models have been proposed to study static rankings: spectral methods including Eigenvector Centrality [2], PageRank [3] and Rank Centrality [4]; approaches aimed at ordinal rankings such as Minimum Violation Rank [5, 6, 7], SerialRank [8] and SyncRank [9]; Random Utility Models [10] such as Bradley-Terry-Luce (BTL) model [11, 12]; fully generative models including Probabilistic Niche Model of ecology [13, 14, 15] or models of friendship based on social status [16], and more generally latent space models [17] which assign probabilities to the existence and direction of edges based on real-valued positions in social space. However, these approaches can be limited or insufficient for modeling dynamic environments where ranks vary in time and interactions have a chronological order. Elo score [18], commonly used for Chess rating system, has been one of the first win-loss method updating the ranks after each match rather than taking all previous interactions into account. The key idea behind the Elo System is to model the probability of the possible game outcomes as a function of the two players' skill scores s_1 and s_2. Elo Rank was later improved by the Glicko system [19] who incorporates a measure of reliability of one's rating which measures rating's uncertainty, for instance due to a period of inactivity or lack of data. We have also Park & Newman's win-lose score algorithm [20] and its dynamic extension [21]. Another Bayesian skill rating system, which can be seen as a generalization of the Elo system, is the so called TrueSkill [22] algorithm. This model uses factor graphs and approximate message passing to infer the marginal belief distribution over the skill of each player. Its straightforward extension is the so called TrueSkill Through Time (TTT) [23] that infers entire time series of skills of players by smoothing through time instead of filtering. More recently, Whole-History Rating (WHR) [24] has been developed to estimate the time-varying strengths of the players involved in paired comparisons. Like

many variations of the Elo rating system, the approach of WHR is based on the dynamic Bradley-Terry model. However, instead of using incremental approximations, WHR directly computes the exact maximum a posteriori over the whole rating history of all players.
Since in this context we do not know ground-truth rankings, the accuracy of the different methods in estimating a meaningful scores can be estimated by assessing the quality of the prediction. That is, given an estimated set of scores, predict the outcomes of unobserved interactions.

3.2 Dynamic models

As explained in the introduction the problem with static rating systems is that they do not consider the variation in time of the strengths of players. Therefore, they are appropriate for rating humans only over a short period of time.
We propose here a model that incorporates time information into the standard (static) SpringRank algorithm. This model assumes a smooth dynamics for the scores, with the Markovian assumption that the scores at time t only depend on the scores at the previous time step $t-1$.

3.2.1 Moving window SpringRank

The simplest way to extend the static SprinkRank in a dynamic context is using a sliding window model. We divide the entire dataset into consecutive windows of fixed time length D. We assume that the scores are static inside each window. In this way, we can apply standard SpringRank within each interval. We do not make any assumption about how scores are related at different time steps and thus treat each interval as independent. It is worth to notice that using this model we may have to deal with two opposite main problems. On one hand, for highly varying rankings, a short window is needed to capture the dynamics and reduce the averaging effect of the static assumption within intervals. This comes at the cost of having less data to perform inference, thus making parameters' estimate less accurate. On the other hand, a large window contains more data to infer scores more robustly but at the cost of missing most of the dynamical changes within the interval. This trade-off suggests the existence of an optimal window's length D^{opt}. We infer it using a cross-validation procedure adapted to time dependent and chronologically ordered data. Moreover, decayed-history rating systems that progressively forget old interactions have some typical structural flaws. Using the language of sport, the main problem is that the decay of games weights generates a very fast increase in the uncertainty of player ratings. With the decayed-history approach, players who stop playing for a while may experience huge jumps in their ratings when they start playing again, and players who play very frequently may have the feeling that their rating is stuck. If players do not all play at the same frequency, there is no good way to tune the speed of the decay [24].

3.2.2 MC SpringRank

Unlike in animal behaviour or in systems of endorsement, in a sport context, where usually the schedule of the matches is randomly chosen, we can not interpret the information deriving from the only *existence* of an interaction as a signal of closeness in ranking. Moreover, to be meaningful, the ranks shouldn't be too much sensitive to small perturbations but they must evolve in time as smoothly as possible.

We modify SpringRank algorithm assuming a dynamic rest length l_{ij}^t of the spring responsible for the interaction at time t between i and j. Furthermore, we assume l_{ij}^t to be a function of the score difference $s_i^{t-1} - s_j^{t-1}$ between i and j at time $t-1$:

$$\begin{aligned} H_{ij}^t\left(s_i^t, s_j^t\right) &= \frac{A_{ij}^t}{2}\left(s_i^t - s_j^t - l_{ij}^t\right)^2 && (3.1) \\ l_{ij}^t &= s_i^{t-1} - s_j^{t-1} + \ell_0 && (3.2) \end{aligned}$$

where constant $\ell_0 \neq 0$, similarly to SpringRank model. In fact the problem with the definition $l_{ij}^t = s_i^{t-1} - s_j^{t-1}$ is that at time t the model would try to minimize $H_{ij}^t\left(s_i^t, s_j^t\right) = \frac{A_{ij}^t}{2}\left(s_i^t - s_j^t - (s_i^{t-1} - s_j^{t-1})\right)^2$ obviously imposing $s_i^t = s_i^{t-1}$, $s_j^t = s_j^{t-1}$ -so no changes in the ranking- to obtain an objective always equal to 0. Thus, to avoid this useless solution, we need definition (3.2) with $\ell_0 \neq 0$.
Intuitively from this formulation we have that, if i beats j at time t, so $A_{ij}^t > 0$, the system will try to minimize the total energy increasing the previous distance $s_i^{t-1} - s_j^{t-1}$ in the ranks between i and j by an additional ℓ_0 amount (of course it doesn't happen exactly most of the time because of cycles, i.e $i \to j \to ... \to z \to i$, remember we want to minimize the total H^t and not only H_{ij}^t separately).
We can write now the total Hamiltonian:

$$H^t(s^t, s^{t-1}) = \sum_{i,j} H_{ij}^t\left(s_i^t, s_j^t\right) \tag{3.3}$$

If we define a new variable $\boldsymbol{z}^t = \boldsymbol{s}^t - \boldsymbol{s}^{t-1}$ we obtain:

$$H^t(z^t) = \sum_{i,j} H_{ij}^t \left(z_i^t, z_j^t\right) = \sum_{i,j} \frac{A_{ij}^t}{2} \left(z_i^t - z_j^t - \ell_0\right)^2$$

which is the exact same Hamiltonian used in SpringRank and then z_t^* will be the solution of the exact same equation found in [1]:

$$\left[D^{\text{out}} + D^{\text{in}} - \left(A + A^T\right)\right] \boldsymbol{z}_t^* = \left[D^{\text{out}} - D^{\text{in}}\right] \ell_0 \mathbf{1}.$$

The main problem with this formulation is that the model is not self-normalizing and the scores tend to diverge as time goes by. This means that the specific values of the ranks are not meaningful and, mostly, the model is slow to adapt to changes in the balance of power among the teams. That is why we need to introduce *'by hand'* a discount factor to keep the scores bounded into a fixed interval $\boldsymbol{s}_{t_i} = \boldsymbol{s}_{t_{i-1}} e^{-\gamma(t_i - t_{i-1})}$, where γ must be tuned depending on the specific application. Clearly this solution is not ideal and we present below a different self-normalizing model.

3.2.3 Self-Spring SpringRank

In this model again we assume an interaction between individual scores at two successive time steps, i.e. the scores at t depend on their counterpart at $t-1$ (self-interaction). We further assume this dependence is smooth, to guarantee that single scores do not change too abruptly between successive time window. We do this by extending the standard SpringRank's Hamiltonian with an extra term that models the self-interaction between past and current scores. This can be seen as an additional spring that connects the score of an individual with its previous value. Overall, each individual feels interactions with others at the same time step, and with himself at the previous time.

Interaction term:

$$H_{ij}^{int}(s_i^t, s_j^t) = \frac{A_{ij}^t}{2}(s_i^t - s_j^t - \ell_0)^2 \tag{3.4}$$

Self-interaction term:

$$H_0(s_i^t, s_i^{t-1}) = \frac{1}{2}(s_i^t - s_i^{t-1})^2 \tag{3.5}$$

Total Hamiltonian at time t:

$$H^t(s^t, s^{t-1}) = \frac{1}{N^2}\sum_{i,j}\left[H_{ij}^{int}(s_i^t, s_j^t) + H_{ji}^{int}(s_j^t, s_i^t)\right] + \frac{k_0}{N}\sum_i H_0(s_i^t, s_i^{t-1}) \tag{3.6}$$

where we normalized the first and second term by respectively $\frac{1}{N^2}$ and $\frac{1}{N}$ to reduce scale problems, with k_0 regulating the importance of the first term of interactions among nodes at time t with respect to the second one representing the self-interaction of each node with its past.

The rest length of self-interaction terms $H_0(s_i^t, s_i^{t-1})$ is zero because we want this to be minimal when $s_i^t = s_i^{t-1}$. This will also minimize the total Hamiltonian in case an individual i does not interact with anyone in a given time window, i.e. $A_{ij}^{t+1} = 0$, $A_{ji}^{t+1} = 0$ for all $j \neq i$, therefore its score should not change.

To find the minimum we start calculating the i-th component of the gradient:

$$\begin{aligned}\frac{\partial H^t}{\partial s_i^t} &= \frac{1}{N^2}\sum_j [A_{ij}^t(s_i - s_j - \ell_0) - A_{ji}^t(s_j - s_i - \ell_0)] + \frac{k_0}{N}(s_i^t - s_i^{t-1}) = \\ &= \frac{1}{N^2}\sum_j (A_{ij}^t + A_{ji}^t)s_i^t - \frac{1}{N^2}\sum_j (A_{ij}^t + A_{ji}^t)s_j^t + \\ &\quad - \frac{1}{N^2}\sum_j (A_{ij}^t - A_{ji}^t)\ell_0 + \frac{1}{N}k_0(s_i^t - s_i^{t-1}) = \\ &= \frac{1}{N^2}(d_i^{out} + d_i^{in} + Nk_0)s_i^t - \frac{1}{N^2}\sum_j (A_{ij}^t + A_{ji}^t)s_j^t + \\ &\quad - \frac{1}{N^2}(d_i^{out} - d_i^{in})\ell_0 - \frac{1}{N}k_0 s_i^{t-1}\end{aligned}$$

Imposing $\nabla H = 0$ we obtain:

$$\begin{aligned}&(d_i^{out} + d_i^{in} + Nk_0)s_i^t - \sum_j (A_{ij}^t + A_{ji}^t)s_j^t = \\ &= (d_i^{out} - d_i^{in})\ell_0 + Nk_0 s_i^{t-1}\end{aligned}$$

We would like to find a rest length $\hat{l}_i^t$ such that this model is equivalent to SpringRank, we impose then:

$$(d_i^{out} - d_i^{in})\ell_0 + Nk_0 s_i^{t-1} = (d_i^{out} - d_i^{in})\hat{l}_i^t \tag{3.7}$$

and, when $d_i^{out} - d_i^{in} \neq 0$, this leads to:

$$\hat{l}_i^t = \ell_0 + \frac{Nk_0 s_i^{t-1}}{d_i^{out} - d_i^{in}} \tag{3.8}$$

so that this model coincides with SpringRank with parameters $\hat{l}_i^t = \ell_0 + \frac{Nk_0 s_i^{t-1}}{d_i^{out} - d_i^{in}}$ and $\alpha = Nk_0$.
Since it shows the best performances, in the following we will keep using this model.

3.2.4 Model testing

Testing a ranking model in real datasets is not straightforward since we do not know the true value of the rank for each team. Nevertheless we can assume that a given ranking is accurate as long as it is helpful in predicting the outcome of new observations.

The simplest measure we can use to understand the quality of our predictions is clearly the accuracy which counts banally the number of times an observed directed link goes from the highest-in-rank node toward to the lowest one –i.e. the number of time that a *stronger* (according to our ranking) individual *beats* a weaker one. Similarly, another measure involving the weighted sum of upsets is the so called *agony* function [7]. The basic idea is that, given our predicted ranking at time t, every time we observe an outcome reverting our hierarchy we get a penalization (i.e. a positive contribution, we want the *agony* as small as possible) weighted for the difference in ordinal ranks[1]: in particular, an upset between two nodes ranked nearby counts much less than an upset between two nodes that are far away in the ranking:

$$agony = \frac{1}{M} \sum_{i,j} A_{ij} \, max(0, r_i - r_j)^d \tag{3.9}$$

where $r_i \in [0, .., n-1]$ is the ordinal rank of node i. When $d = 0$ we recover the standard number of unweighted upsets. We expect that the better the ranking is predicted, the lower the *agony* is, which means that our hierarchies are violated as least as possible. Up to now we considered only the position of each node inside the entire ordinal ranking. Nevertheless, for models such as SpringRank a notion of *score* -a real number representing the effective *strength* of the node- is naturally associated to each individual together with a probability of victory or loss depending on this score-difference. To evaluate also these probabilistic predictions we will consider two other metrics: σ_a is the average probability assigned to the correct direction of an edge, and σ_L is the log-likelihood of generating the directed edges given their existence. In the multigraph case, we ask how well P_{ij}, the probability that i beats j, approximates the fraction of interactions between i and

[1]We use *positional* ranks instead of the real valued scores to avoid scale problems comparing different algorithms

j that point from $i \to j$:

$$\sigma_a = \frac{1}{2M} \sum_{ij} |A_{ij} - \overline{A_{ij}} P_{ij}| \tag{3.10}$$

As regards σ_L we measure accuracy via the conditional log-likelihood, asking with what probability we would obtain the directed network A from the undirected one $\bar{A}$, with P_{ij} the probability that an edge between i and j points from $i \to j$ and $P_{ji} = 1 - P_{ij}$ if points from $j \to i$:

$$\begin{aligned} \sigma_L &= \log Pr\left[A|\bar{A}\right] = \\ &= \sum_{ij} \binom{A_{ij} + A_{ji}}{A_{ij}} + log\left[P_{ij}(\beta)^{A_{ij}}(1 - P_{ij}(\beta))^{A_{ji}})\right], \end{aligned} \tag{3.11}$$

where we explicitly highlight the dependence of P_{ij} on the (inverse) *temperature* parameter β which control the level of hierarchy in the predictions (for $\beta \to \inf$ the network is fully hierarchical which means that an edge between i and j, with $score(i) > score(j)$, goes from $i \to j$ with $P_{ij} = 1$; on the contrary, for $\beta = 0$ the outcome predictions are completely random with $P_{ij} = P_{ji} = 0.5$ no matter the rank of i or j). It is worth to notice also that in general maximizing either σ_a or σ_L can not be same and thus we will obtain two distinct values for β that we will denote with $\hat{\beta}_a$ and $\hat{\beta}_L$. Intuitively, the reason is that a single severe mistake where $A_{ij} = 1$ but $P_{ij} \approx 0$ reduces the likelihood by a large amount, while only reducing the accuracy by one edge. As a result, predictions using $\hat{\beta}_a$ produce fewer incorrectly oriented edges, achieving a higher σ_a on the test set, while predictions using $\hat{\beta}_L$ will produce fewer dramatically incorrect predictions where P_{ij} is very low, and thus achieve higher σ_L on the test set [1]. In practise using the language of sport, a prediction model whose goal is to maximize σ_L will tend to be more cautious in assigning high probabilities of *success* even in very unbalanced matches in order to avoid potential painful mistake; on the contrary, a model optimizing σ_a can be less conservative, ignoring isolated (even dramatic) mistakes and favouring a good frequency of predictions as close as possible to the real probability.

3.2.5 Generative model and synthetic dataset

In analogy with SpringRank generative model presented in [1], in this section we propose a *dynamic* probabilistic generative model that takes in input the scores s_i^{t-1} at $t-1$ and generates a weighted directed network with adjacency matrix A^t at time t. Also in this case, the model has a noise parameter β and a density parameter

c. Edges between each pair of nodes i, j are generated independently conditioned on the scores $\boldsymbol{s}^t$. The expected number of edges between i, j is proportional to the probability P_{ij}^t of i beating j at time t. This probability should depend on the difference $s_i^t - s_j^t$ and on the noise parameter β regulating the extent to which the edges respect the scores; also, $P_{ij}^t + P_{ji}^t = 1$. A natural choice is the softmax as in [1]:

$$P_{ij}^t = \frac{1}{1 + e^{-2\beta(s_i^t - s_j^t)}}. \tag{3.12}$$

We thus obtain:

$$\mathbb{E}[A_{ij}^t] = c\,P_{ij}^t = \frac{c}{1 + e^{-2\beta(s_i^t - s_j^t)}}. \tag{3.13}$$

with the weight A_{ij}^t drawn from a Poisson distribution with mean (3.13):

$$A_{ij}^t \sim Pois\left(\lambda = \frac{c}{1 + e^{-2\beta(s_i^t - s_j^t)}}\right) \tag{3.14}$$

Also in this case, c controls the overall density of the network given an expected number of edges at time t:

$$\mathbb{E}[M^t] = \sum_{i,j} \mathbb{E}[A_{ij}^t] = c \sum_{i,j} P_{ij}^t = c \sum_{i,j} \frac{1}{1 + e^{-2\beta(s_i^t - s_j^t)}}\,. \tag{3.15}$$

The difference with the static generative model of [1] is that here scores evolve in time. Hence, if we want to draw a chain of networks in time, we need to update the scores accordingly. They depend not only on the previous-time scores, but also on the network A^t just drawn:

$$\boldsymbol{s}^t = f(\boldsymbol{s}^{t-1}, A^t)\,, \tag{3.16}$$

where f denotes a generic function that needs to be specified. Clearly f must balance between the dependence on the past history $\boldsymbol{s}_{t-1}$ -to be useful we want the scores to evolve *smoothly* in time, big jumps should be avoided- and the importance of the information deriving from the new data A_t just observed that can reveal a change in the actual ranking.
Actually, to force changes and swaps in the ranking, at the moment of creating synthetic datasets we will choose the exact evolution though time of the score $\boldsymbol{s}_t$ for each individual, taking on purpose trajectories that repeatedly fall down and climb back up, simulating in this way the typical dynamic behaviour we expect to

find in sport contests due to injuries, changes in the team's players, form of the team etc. and in all the other fields we are interested in:

$$\boldsymbol{s}_t = g(t). \tag{3.17}$$

At each time t we use (3.14) to generate data. We can assign arbitrarily the initial values of the ranks $s_1^0, ..., s_N^0$, here we will draw them from a standard normal distribution.

3.3 Static or dynamic?

In this section we return on the problem of choosing, given a dataset, between a static or a dynamic approach. As explained in the introduction, static algorithms cannot perceive the different phases the entities we want to rank are going through. Because of that, these static models end up assigning the competitors a unique average value for the entire period of study. This clearly prevents us from effectively forecasting the outcomes of future matches and that is why, in a dynamic time-varying context, we need dynamic rating systems able to promptly capture variations in the ranking. We know that if the extracted ranking is good, and when the problem under investigation is hierarchical, the outcomes of the matches are expression of these ranks. The form of the time series governing the evolution of the score $\boldsymbol{s}$ can be whatever but smoothness is a fundamental requirement. In fact, the only way to do predictions is supposing that the ranking does not change too much between consecutive instants of time, and then that the ranks extracted up to time t are still meaningful, in good approximation, to predict outcomes at $t + 1$. In real situations this is usually confirmed due to the typical cyclical and periodic behaviour we can observe for example in sports and other contexts.

Because of this intuition, in every model we have built, we always assumed a strong correlation between the score at time t with its previous value at time $t-1$, supposing it cannot jump too much within consecutive time instants.
We try now to randomly permute the order of the observed matches. This new dataset contains the same set of matches and outcomes as the original dataset but time-stamps information is not truthful anymore. Of course, if the ranks are static and fixed in time, the order of the matches should not be important while estimating $\boldsymbol{s}_t$ and dynamic algorithms should perform the same no matter the particular permutation of the matches chosen (and perform the same of static algorithms). On the contrary, if the ranks change through time –and given the direct connection between $\boldsymbol{s}_t$ and the observed outcomes– we expect that the prediction power

of dynamic algorithms run on the chronologically ordered dataset will be statistically higher compared with the same models run on a random permutation of the dataset. In particular, this is because reversing randomly the order of the matches we are breaking the *natural* smooth and periodic behaviour we were mentioning above, which governs the evolution of the ranks. We are in practice removing the correlation between $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t-1}$, making forecasting impossible. Then we expect that if the dataset is inherently hierarchical and dynamic, the difference between the performances using the chronologically ordered or the permuted datasets is significantly relevant.

Using the metrics mentioned in 3.2.4, we test the statistical difference in the results for both synthetic and real data. We use the dynamic generative model presented above to create the synthetic datasets varying also the β parameter. Recall that when β is small -i.e. high *temperature*- the hierarchy is weak since there is no dependence between ranks and outcomes anymore (for β exactly zero the outcome of an interaction between two nodes is completely random). Clearly in this case we expect no difference in the accuracy of the predictions taking the matches in the right or randomly permuted order. On the contrary, for a highly hierarchical dataset where the dynamic scores change smoothly in time, we expect a relevant improvement using the time-stamps of the interactions and analysing the data in their chronological order, see Figure 1.
We consider also static situations in which the above generative model is used but the scores are fixed in time, i.e. $\boldsymbol{s}_t = \boldsymbol{s}_0$ for every t. Obviously once again we expect that the various dynamic rating systems do not perform better than static SpringRank. Moreover, we expect our significance test on the importance of analysing the matches in their chronological order will be negative since the ranking is stable and fixed in time, making the collection of all the time-stamps completely useless, see Figure 2.

As regard real data, Figure 3.1 shows that NBA championship is clearly an inherently hierarchical and dynamic context, where the collection of the precise time stamp for each observation is crucial. The hypothesis about the strong correlation between ranks estimated at consecutive instants is confirmed looking at the well-marked difference between the results obtained using the real chronological order of the matches against the various permutations of it. This makes NBA dataset particularly useful for our discussion and we will use it in all the following of this thesis.

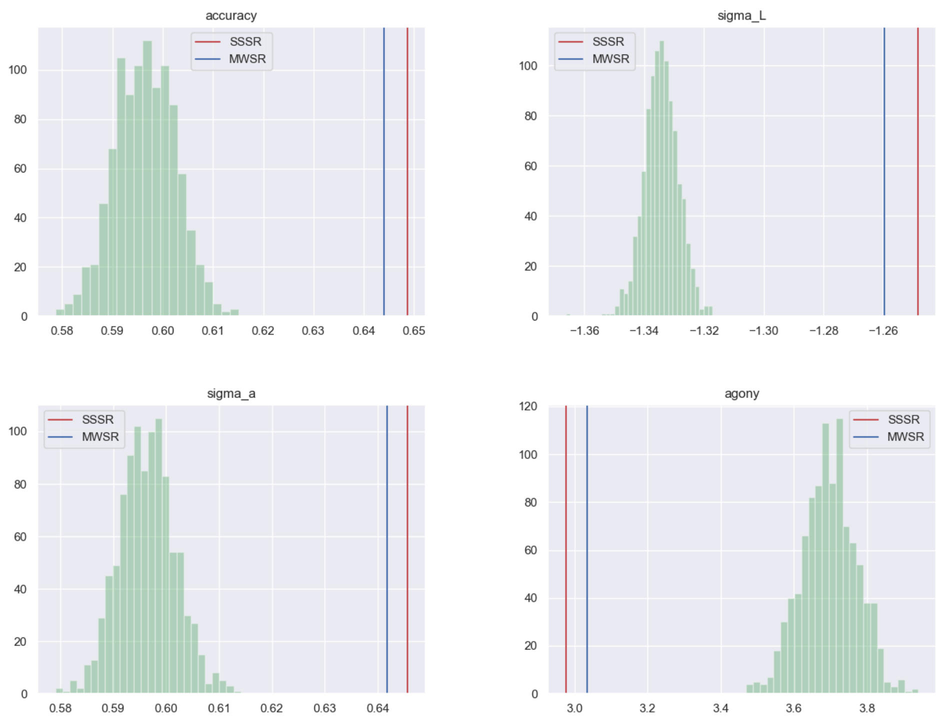


Figure 3.1: Results obtained for NBA dataset (seasons 2011/2017): analysis of the performance for different metrics of NBA real data. The red and blue lines are respectively the value obtained with Self Spring SpringRank (SSSR) algorithm and with moving window version of SpringRank (MWSR) considering the true time stamps (the real chronological order) of the matches, each entry of the histogram is instead a different realization of SSSR for a random permutation of the order of the matches (random time-stamps assigned to the matches).

3.4 Results on real and synthetic data

We want now to compare our dynamic ranking algorithm with some of the most popular existing ones such as Elo-Rank [18], TrueSkill [23] and WHR [24]. Also, SpringRank [1] and its moving window version presented above will be used as baselines. First of all we divide the dataset in two parts, one will be used for tuning the hyperparameters of each algorithms and the other for testing. The two parts will be kept separate during the entire experiment. Clearly, the parameters will be selected in order to maximize the performance of prediction on new unobserved data. The dynamic setting requires preserving the sequential and chronological order of the observations. This prevents us using a standard cross-validation approach where one splits randomly observations into a train and test set. That is because outcomes' predictions must be done only on unseen subsequent observations, i.e. observations chronologically subsequent to the ones used for training the model and estimating the ranking. Since we have time correlated interactions only the information until time t can obviously be used to infer $\boldsymbol{s}_{t+1}$. Once we have $\boldsymbol{s}_{t+1}$ we finally predict the outcome of the interactions between nodes at time $t+1$. As explained before, the goodness of the ranking estimate $\boldsymbol{s}_t$ extracted by each algorithm is clearly measured on the predictive power that this estimate assures while forecasting future outcomes, in particular rating models with better predictive performance will be considered models able to infer better estimations of the hidden ranks of the nodes. We will use common grid search to tune hyperparameters such as window size (WHR, TrueSkill and the moving average SR), k_0 (Self Spring SpringRank) etc. Once we have found the best hyperparameters for each model optimizing the performance on the train set, we infer the sequence of scores $\boldsymbol{s}_t$ in the test set and use it again to predict the results of future events (and the probabilities of these outcomes).

3.4.1 Performance on synthetic networks.

We generate synthetic datasets as explained above. For simplicity we use cosine functions to model the dynamic scores in equation (3.17):

$$s_{ti} = A_{ti}\cos(\omega_i t + \phi_i) \tag{3.18}$$

where A_{ti}, ω_i and ϕ_i are randomly chosen for each team to ensure changes and swaps in the ranking. Different values of the parameter β in (3.14) have been tried. Looking at table 3.1, it's quite clear that results are highly dependent on

the value of the *disorder* parameter β in the generative model. For β big, i.e. high hierarchy when generating the number of the wins for each team, the difference between using static SpringRank or a dynamic rating system is well-marked. In this situation our Self-Spring algorithm beats the others in almost all the different metrics. As long as we reduce β the difference between using static or dynamic algorithms decreases due to the strong noise in the observations. For the same reason, also collecting time-stamps for each match and analysing the outcomes in chronological order is useless, see Figure 1 in appendix 5.
We analyse also the behaviour of the various algorithms in a static situation where the dataset is created with the same procedure but, differently from (3.18), using fixed in time ranks $s_{ti} = s_i$. As expected, static SpringRank performs as well as dynamic algorithms (see table 3.2). Obviously, also here collecting time-stamps and analysing the observations in their chronological order is useless, as shown in Figure 2.

		SR	Elo	TS	WHR	MWSR	SSSR
$\beta = 2.0$	acc	0.702	0.879	0.878	0.878	0.869	**0.882**
	agony	0.738	0.222	0.226	0.227	0.250	**0.221**
	σ_a	0.715	0.856	0.851	0.847	0.872	**0.883**
	σ_L	-1.422	-0.742	-0.562	**-0.557**	-0.614	-0.589
$\beta = 1.0$	acc	0.668	0.822	0.818	0.824	0.820	**0.826**
	agony	0.940	0.398	0.410	0.391	0.399	**0.389**
	σ_a	0.636	0.787	0.787	0.789	0.825	**0.830**
	σ_L	-1.231	-0.986	-0.809	**-0.790**	-0.856	-0.791
$\beta = 0.5$	acc	0.685	0.699	0.699	0.698	0.696	**0.702**
	agony	**0.876**	0.897	0.896	0.903	0.914	0.901
	σ_a	0.669	0.683	0.684	0.679	0.708	**0.710**
	σ_L	-1.185	-1.358	**-1.168**	-1.177	-1.172	-1.169
$\beta = 0.1$	acc	0.528	0.519	0.528	0.515	**0.579**	0.513
	agony	**1.629**	1.710	1.701	1.741	1.685	1.773
	σ_a	**0.610**	0.603	0.603	0.601	0.579	0.575
	σ_L	**-1.379**	-1.449	-1.396	-1.403	-1.398	-1.405

Table 3.1: Results on synthetic networks: the table shows the results according to various metrics (rows) for the principal state-of-art ranking algorithms such as SpringRank (SR), EloRank, TrueSkill (TS), Whole History Rating (WHR) compared with our moving window version of SpringRank (MWSR) and Self Spring SpringRank (SSSR). Different values of the β parameter have been tried in the generative model.

	SR	Elo	TS	WHR	MWSR	SSSR
accuracy	**0.718**	**0.718**	0.715	0.699	0.712	0.702
agony	**0.495**	0.528	0.529	0.548	0.528	0.551
σ_a	0.663	0.656	0.665	0.651	**0.717**	0.705
σ_L	-1.147	-1.292	**-1.137**	-1.156	-1.742	-1.276

Table 3.2: Results obtained for synthetic data in a static framework: comparison of the performances of the various static and dynamic rating systems on a synthetic dataset where in the generation process the ranks are kept fixed along time (static framework).

3.4.2 Performance for real network

We consider real datasets of NBA regular seasons matches from 2011 to 2017. Overall, we have a total of $M = 9594$ matches distributed in 8 seasons. The number of teams is $N = 30$ and tie is not an outcome allowed (matches that are even at the end of the regular time go to overtimes until we have a winner). Table 3.3 shows the results, with our Self Spring SpringRank (SSSR) which is the best according to all the various metrics. Figure 3.2 is a pairwise comparison between SSSR and the other principal rating systems, with the focus on the percentage of times our algorithm beats the others on different specific test windows.

	SR	Elo	W-L	TS	WHR	MWSR	SSSR
accuracy	0.399	0.646	0.565	0.647	0.648	0.644	**0.649**
agony	3.650	2.979	4.068	2.995	2.994	3.034	**2.976**
σ_a	0.590	0.581	-	0.586	0.580	0.642	**0.646**
σ_L	-1.327	-1.437	-	-1.272	-1.257	-1.259	**-1.248**

Table 3.3: Results for NBA dataset seasons 11-17: the table presents the results for NBA regular seasons from 2011 to 2017. Each row is a different measure of the performance and each column one of the various state-of-art ranking algorithms and our moving window SpringRank and Self Spring SpringRank.

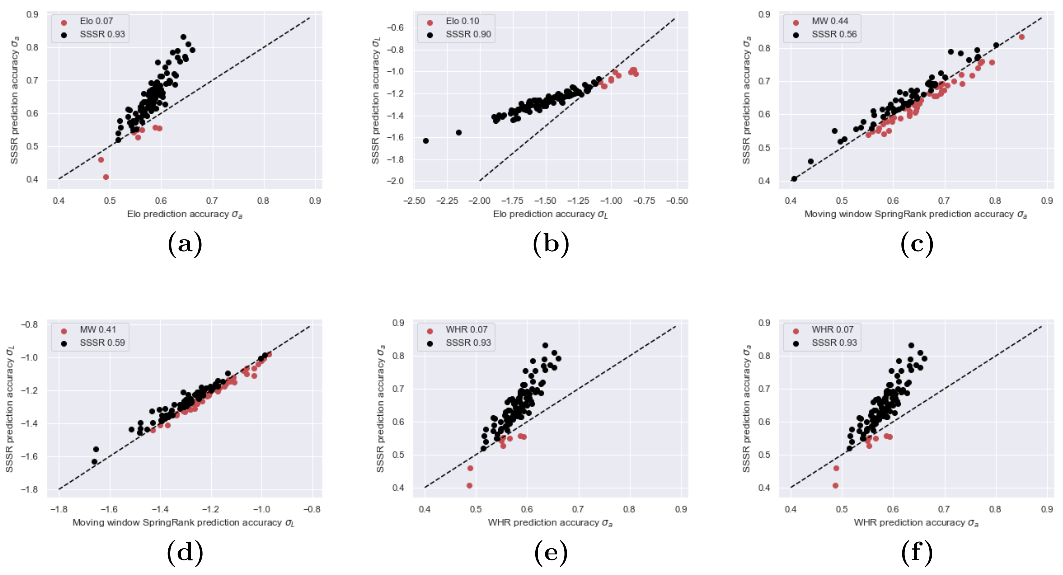


Figure 3.2: Probabilistic edge prediction accuracy σ_a and σ_L. We compare the predictive performances of SSSR against respectively EloRank (**a** and **b**), the moving average SpringRank (**c** and **d**), WHR (**e** and **f**). The two probabilistic metrics σ_a and σ_L have been chosen. Points above the diagonal, shown in black, are trials where SSSR outperforms one of the other algorithms. The fraction for which each method is superior are shown in plot legends.

4. Variational SpringRank

Probabilistic models with latent variables have become a mainstay in modern machine learning applications. With latent variables models, we posit a rich latent structure that governs our observations, infer that structure from large datasets, and use our inferences to summarize observations, draw conclusions about current data, and make predictions about new data. Central to working with latent variable models is the problem of computing the posterior distribution of the latent structure. For many interesting models, computing the posterior exactly is intractable and practitioners must then resort to approximate methods.

4.1 Approximate Inference

One of the core problems of modern statistics is to approximate difficult-to-compute probability densities. This problem is especially important in Bayesian statistics, which frames all inference about unknown quantities as a calculation about the posterior. Modern Bayesian statistics relies on models for which the posterior is not easy to compute and corresponding algorithms for approximating them.

The general problem is framed in the following way: we consider a joint density of latent variables $\boldsymbol{z} = z_{1:m}$ and observations $\boldsymbol{x} = x_{1:n}$,

$$p(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}). \tag{4.1}$$

All unknown quantities of interest are represented as latent random variables. This includes parameters that might govern all the data, as found in Bayesian models, and latent variables that are "local" to individual data points. In Bayesian models, the latent variables help govern the distribution of the data. A Bayesian model draws the latent variables from a prior density $p(\boldsymbol{z})$ and then relates them to the observations through the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$.

The inference problem is to compute the conditional density of the latent variables given the observations, i.e. computing the posterior $p(\boldsymbol{z}|\boldsymbol{x})$. This conditional can be used to produce point or interval estimates of the latent variables, from predictive densities of new data, and more. The conditional density can be written using *Bayes Theorem* as

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{x})} \tag{4.2}$$

The denominator contains the marginal density of the observations, also called the evidence. We calculate it by marginalizing out the latent variables from the joint density,

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z} \tag{4.3}$$

but for many models, this evidence integral is unavailable in closed form or requires exponential time to compute. The evidence is what we need to compute the conditional from the joint; this is why inference in such models is hard.

Exact Inference using conjugate priors We briefly explain here a typical situation in which we do not need to approximate the posterior but we are able to infer the exact one. In Bayesian probability theory, if the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ are in the same probability distribution family as the prior $p(\boldsymbol{z})$, the prior and the posterior are then called *conjugate distributions*, and the prior is called a *conjugate prior* for the likelihood function. The simplest example involves the Gaussian family which is conjugate with itself, or *self-conjugate*, with respect to a Gaussian likelihood function. This can be easily derived from (4.2):

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})} \propto p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) \tag{4.4}$$

remembering that the product of two Gaussians (both the prior and the likelihood are Gaussians) is still a Gaussian. Knowing the form of the posterior, we are not forced anymore to calculate the evidence $p(\boldsymbol{x})$ to get the exact posterior (see [25] for the explicit calculation using for example *square completing* technique). More conjugacy relationships can be found in [25], we mention here some of the more useful ones, such as a Bernoulli likelihood combined with a beta conjugate prior or uniform likelihood with pareto conjugate prior.

4.1.1 Markov Chain Monte Carlo

Most of the time, due to the impossibility of calculating the evidence in closed form, solving the problem of finding the posterior distribution requires approximate inference. For decades, the dominant paradigm for approximate inference has been MCMC [26, 27]. In MCMC, the basic idea is to build an ergodic Markov chain on $\boldsymbol{z}$ whose stationary distribution is the posterior $p(\boldsymbol{z}|\boldsymbol{x})$. Then, we sample from the chain to collect samples from the stationary distribution. Finally, we approximate the posterior with an empirical estimate constructed from (a subset of) the collected samples. Landmark developments include the Metropolis-Hastings algorithm [28, 26], the Gibbs sampler [29] and its application to Bayesian statistics [27]. Despite its key role in modern Bayesian statistics, MCMC sampling is a broad field and it is out of the scope of this thesis. For a more complete overview of this topic the reader had better refer to the above citations.

4.1.2 Variational Inference

Rather than using sampling, the main idea behind variational inference is to use optimization. In order to measure how much one probability distribution is different from a second, reference one, we start recalling the definition of the Kullback-Leibler divergence between $q(\boldsymbol{z})$ and $p(\boldsymbol{z}|\boldsymbol{x})$:

$$\mathtt{KL}(q||p) = \mathbb{E}_q\left[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}\right]. \tag{4.5}$$

Let's notice that KL divergence is a distribution-wise asymmetric measure and thus does not qualify as a statistical metric of spread (it also does not satisfy the triangle inequality). We start defining a family of approximate densities $\mathcal{D}$ over the latent variables. Then, we try to find the member of the family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior,

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z})\in\mathcal{D}}{\operatorname{argmin}}\, \mathtt{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) \tag{4.6}$$

so we approximate the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ with $q^*(\boldsymbol{z})$.

Variational inference thus turns the inference problem into an optimization problem. The quality of the approximation and the complexity of the optimization are clearly related with our choice of the set $\mathcal{D}$. One of the key ideas behind

variational inference is to choose $\mathcal{D}$ to be flexible enough to capture a density close to $p(\boldsymbol{z}|\boldsymbol{x})$, but simple for efficient optimization.
It is important to notice that MCMC and variational inference are different approaches to solving the same problem. MCMC algorithms sample a Markov chain while variational algorithms solve an optimization problem with the common aim of approximating the posterior.

The Evidence Lower Bound The main problem with (4.6) is that the objective is not computable since we do not have the evidence $\log p(\boldsymbol{x})$ related with (4.5) through (4.2) (this is the reason why we are applying variational inference from the beginning). In fact using the definition in (4.5) we obtain.

$$\begin{aligned}\texttt{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) &= \mathbb{E}_q\left[\log q(\boldsymbol{z})\right] - \mathbb{E}_q\left[\log p(\boldsymbol{z}|\boldsymbol{x})\right] \\ &= \mathbb{E}_q\left[\log q(\boldsymbol{z})\right] - \mathbb{E}_q\left[\log p(\boldsymbol{z},\boldsymbol{x})\right] + \mathbb{E}_q\left[\log p(\boldsymbol{x})\right] \qquad (4.7)\end{aligned}$$

where the dependence on $\log p(\boldsymbol{x})$ is revealed. Because we cannot compute the KL, we optimize an alternative objective function, equivalent to the KL up to $p(\boldsymbol{x})$ –which is a constant with respect to q– that we call ELBO:

$$\texttt{ELBO}(q) = \mathbb{E}_q\left[\log p(\boldsymbol{x},\boldsymbol{z})\right] - \mathbb{E}_q\left[\log q(\boldsymbol{z})\right] = \mathbb{E}_q\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}\right], \qquad (4.8)$$

here ELBO means *Evidence LOwer Bound* since, remembering that KL is not negative, we have from (4.7) and (4.8):

$$\log p(\boldsymbol{x}) = \texttt{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) + \texttt{ELBO}(q) \geq \texttt{ELBO}(q). \qquad (4.9)$$

We want to emphasize again that the ELBO is nothing else than the negative KL divergence in (4.5) plus $\log p(\boldsymbol{x})$, which is a constant with respect to q. Then maximizing the ELBO with respect to q is equivalent to minimizing the KL divergence:

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z})\in\mathcal{D}}{\operatorname{argmax}}\Big(\texttt{ELBO}(\boldsymbol{x},p,q)\Big) = \underset{q(\boldsymbol{z})\in\mathcal{D}}{\operatorname{argmin}}\,\texttt{KL}\Big(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})\Big) \qquad (4.10)$$

Rewriting the ELBO (4.8) as a sum of the expected log likelihood of the data and the KL divergence between the prior $p(\boldsymbol{z})$ and $q(\boldsymbol{z})$,

$$\begin{aligned}\texttt{ELBO}(q) &= \mathbb{E}_q\left[\log p(\boldsymbol{z})\right] + \mathbb{E}_q\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_q\left[\log q(\boldsymbol{z})\right] \qquad (4.11) \\ &= \mathbb{E}_q\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \texttt{KL}(q(\boldsymbol{z})||p(\boldsymbol{z})),\end{aligned}$$

gives intuitions about the optimal variational density. The first term is an expected likelihood; it encourages densities that place their mass on configurations of the latent variables that explain the observed data. The second term is the negative divergence between the variational density and the prior; it encourages densities close to the prior.
Usually the variational approximation q is parametrized by some parameters $\boldsymbol{\lambda} \in \mathbb{R}^d$ and the above optimization problem (4.10) can be rewritten as

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \Big(\texttt{ELBO}(\boldsymbol{x}, p, q_{\boldsymbol{\lambda}}) \Big) \tag{4.12}$$

with $q^*_{\boldsymbol{\lambda}}(\boldsymbol{z}) = q_{\boldsymbol{\lambda}^*}(\boldsymbol{z})$.

However, in its general form the lower bound $\texttt{ELBO}(\boldsymbol{x}, p, q)$ is a functional in terms of q, in other words, a mapping that takes as input a function $q(\boldsymbol{z})$, and returns as output the value of the functional. This leads naturally to the concept of the functional derivative, which in analogy to the function derivative, gives the functional changes for infinitesimal changes to the input function. This area of mathematics is called calculus of variations [30] and has been applied to many areas of mathematics, physical sciences and engineering, for example fluid mechanics, heat transfer, and control theory. Although there are no approximations in the variational theory, variational methods can be used to find approximate solutions in Bayesian inference problems. This is done by assuming that the functions over which optimization is performed have specific forms. For example, we can assume only quadratic functions or functions that are linear combinations of fixed basis functions. For Bayesian inference, a particular form that has been used with great success is the factorized one. The idea for this factorized approximation stems from theoretical physics where it is called mean field theory [31].

MCMC or Variational Inference?

When should a statistician use MCMC and when should he use variational inference? Generally MCMC methods tend to be more computationally intensive than variational inference but they also provide guarantees of producing (asymptotically) exact samples from the target density [32]. Variational inference does not enjoy such guarantees –it can only find a density close to the target– but tends to be faster than MCMC. Because it rests on optimization, variational inference easily takes advantage of methods like stochastic optimization [33] and distributed optimization.
Thus, variational inference is suited to large data sets and scenarios where we want

to quickly explore many models; MCMC is suited to smaller data sets and scenarios where we happily pay a heavier computational cost for more precise samples. For example, we might use MCMC in a setting where we spent 20 years collecting a small but expensive data set, where we are confident that our model is appropriate, and where we require precise inferences. We might use variational inference when fitting a probabilistic model of text to one billion text documents and where the inferences will be used to serve search results to a large population of users. In this scenario, we can use distributed computation and stochastic optimization to scale and speed up inference, and we can easily explore many different models of the data.

4.1.3 Mean-Field variational family

We have seen above that one of the key aspect of variational inference is the choice of the variational family $\mathcal{D}$ from which the variational approximation q is taken. Clearly, the complexity of the family determines the complexity of the optimization; it is more difficult to optimize over a complex family than a simple family. On the other hand, a broader and more complex family will result into a better approximation q of the posterior.
We focus here on the *mean field variational family*, where the latent variables are mutually independent and each governed by a distinct factor in the variational density. A generic member of the mean field variational family is

$$q_{\boldsymbol{\lambda}}(\boldsymbol{z}) = \prod_{j=1}^{n} q_{\boldsymbol{\lambda}_j}(z_j) \tag{4.13}$$

Each latent variable z_j is governed by its own variational factor, the density $q_{\boldsymbol{\lambda}_j}(z_j)$. We emphasize that the variational family is not a model of the observed data – indeed, the data $\boldsymbol{x}$ does not appear in (4.13). Instead, it is the ELBO, and the corresponding KL minimization problem, that connects the fitted variational density to the data and model.
Moreover, choosing a factorized distribution means neglecting, at least in first approximation, all the correlations between the variables in the posterior distribution, but, as just said, correlation still enters into play when we minimize the KL, where correlations between variables are instead present.

4.2 Expectation Maximization

We present here another technique which we will use in the following: the Expectation Maximization algorithm (EM). EM is nothing else than an iterative algorithm for Maximum Likelihood estimation that has a number of advantages and has become a standard methodology for solving statistical signal processing problems. However, the EM algorithm has certain requirements that seriously limit its applicability to complex problems. More recently, to relax some of these limiting requirements, variational EM –i.e EM exploiting variational inference– has been developed and it is gaining rapidly popularity.

Suppose $\boldsymbol{x}$ are the observations and $\boldsymbol{\theta}$ the unknown parameters of a model that generates $\boldsymbol{x}$. For the sake of clarity, in the following we will write $p(\boldsymbol{x}|\boldsymbol{\theta})$ when considering $\boldsymbol{\theta}$ as random variables and $p(\boldsymbol{x};\boldsymbol{\theta}) = p_{\boldsymbol{\theta}}(\boldsymbol{x})$ –called likelihood as a function of $\boldsymbol{\theta}$– when we want to imply that $\boldsymbol{\theta}$ are parameters. Since in many cases of interest direct assessment of the likelihood function $p(\boldsymbol{x};\boldsymbol{\theta})$ is complex and is either difficult or impossible to compute it directly or optimize it, we introduce the hidden variables $\boldsymbol{z}$, for which $p(\boldsymbol{x}|\boldsymbol{z})$ is easy to compute. In this framework we call $p(\boldsymbol{x};\boldsymbol{\theta})$ the *marginal likelihood* and we have

$$p(\boldsymbol{x};\boldsymbol{\theta}) = \int p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})d\boldsymbol{z} = \int p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta})p(\boldsymbol{z};\boldsymbol{\theta})d\boldsymbol{z} \tag{4.14}$$

and as seen before this integral -note the parallel with the *evidence* (4.3) in the variational terminology- is most of the time impossible to compute.
Similarly to (4.9) we can write then

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = F(q,\boldsymbol{\theta}) + \texttt{KL}(q||p) \tag{4.15}$$

with $F(q,\boldsymbol{\theta})$ -i.e the `ELBO` in the variational context- defined as

$$F(q,\boldsymbol{\theta}) = \mathbb{E}_q\left[\log\frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\right] \tag{4.16}$$

and with the `KL` divergence defined as before

$$\texttt{KL}(q||p) = \mathbb{E}_q\left[\log\frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}\right] \tag{4.17}$$

where $q(\boldsymbol{z})$ is any probability density function.

As seen before, since $\texttt{KL}(q||p) \geq 0$,

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = F(q,\boldsymbol{\theta}) + \texttt{KL}(q||p) \geq F(q,\boldsymbol{\theta}), \tag{4.18}$$

in other words, $F(q,\boldsymbol{\theta})$ is a lower bound of the (marginal) log-likelihood. Equality holds only when $\texttt{KL}(q||p) = 0$, which implies $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}) = q(\boldsymbol{z})$, see (4.17). The EM algorithm can be seen in the light of decomposition in (4.15) as the maximization of the lower bound $F(q,\boldsymbol{\theta})$ with respect to the density q and the parameters $\boldsymbol{\theta}$. In particular, the EM is a two step iterative algorithm that maximizes the lower bound $F(q,\boldsymbol{\theta})$ and hence the (marginal) log-likelihood. Assume that the current value of the parameters is $\boldsymbol{\theta}^{OLD}$. In the E-step the lower bound $F(q,\boldsymbol{\theta}^{OLD})$ is maximized with respect to $q(\boldsymbol{z})$. Of course, looking at (4.15), this is equivalent to minimize $\texttt{KL}(q||p)$ that, as we just seen, happens when $\texttt{KL}(q||p) = 0$, in other words, when $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD})$. In this case the lower bound is equal to the log-likelihood. In the subsequent M-step, $q(\boldsymbol{z})$ is held fixed and the lower bound $F(q,\boldsymbol{\theta})$ is maximized with respect to $\boldsymbol{\theta}$ to give some new value $\boldsymbol{\theta}_{NEW}$. This will cause the lower bound to increase and as a result, the corresponding log-likelihood will also increase.
Since $q(\boldsymbol{z})$ was determined using $\boldsymbol{\theta}^{OLD}$ and is held fixed in the M-step, it will not be equal to the new posterior $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}_{NEW})$ and hence the $\texttt{KL}$ distance will not be zero. Thus, the increase in the log-likelihood is greater than the increase in the lower bound.
If we substitute $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD})$ into the lower bound and expand (8) we get

$$\begin{aligned} F(q,\boldsymbol{\theta}) &= \int p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD}) \log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) d\boldsymbol{z} - \int p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD}) \log p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD}) d\boldsymbol{z} \\ &= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{OLD}) + const \end{aligned} \tag{4.19}$$

where *const* is simply the second term -i.e the entropy of $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD})$- which does not depend on $\boldsymbol{\theta}$. The function

$$Q(\boldsymbol{\theta},\boldsymbol{\theta}^{OLD}) = \int p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD}) \log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta}) d\boldsymbol{z} \tag{4.20}$$

is the expectation of the log-likelihood of the complete data (observations + hidden variables) which is maximized in the M-step.
In summary, the EM algorithm is an iterative algorithm involving the following two steps:

Algorithm 1 EM

```
1: repeat
2:     E-step: Compute $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD})$
3:     M-step: Evaluate $\boldsymbol{\theta}^{NEW} = \text{argmax}_{\boldsymbol{\theta}}\, Q(\boldsymbol{\theta},\boldsymbol{\theta}^{OLD})$
4:     $\boldsymbol{\theta}^{OLD} = \boldsymbol{\theta}^{NEW}$
5: until convergence
```

Furthermore, we would like to point out that the EM algorithm requires that $P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ is explicitly known, or at least we should be able to compute the conditional expectation (4.20) $\mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{OLD})}[\log p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})]$. In other words, we have to know the conditional pdf of the hidden variables given the observations in order to use the EM algorithm. Even if $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ is in general much easier to infer than $p(\boldsymbol{x};\boldsymbol{\theta})$, in many interesting problems this is not possible and thus the EM algorithm is not applicable.

Variational Expectation Maximization

The main problem in the EM algorithm just described above is clearly the computation of $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ in the E-step. One can bypass the requirement of exactly knowing $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$ by assuming an appropriate $q(\boldsymbol{z})$ in the decomposition of (4.15). Clearly this brings us back to variational inference. In the E-step $q(\boldsymbol{z})$ is found such that it maximizes $F(q,\boldsymbol{\theta})$ keeping $\boldsymbol{\theta}$ fixed (4.18). This is the exact same problem we were trying to solve before. We already know that to perform this maximization, a particular form of $q(\boldsymbol{z})$ must be assumed. In certain cases it is possible to assume knowledge of the form of $q(\boldsymbol{z};\boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ is a set of parameters. Thus, the lower bound $F(\boldsymbol{\lambda},\boldsymbol{\theta})$ becomes a function of these parameters and is maximized with respect to $\boldsymbol{\lambda}$ in the E-step and with respect to $\boldsymbol{\theta}$ in the M-step.

4.3 Variational Spring Rank Model

Taking inspiration from [34] we want to develop a dynamic Bayesian model for ranking which captures the evolution of ranks given a sequentially ordered sequence of observations of pairwise directed interactions among the various individuals. In the above notation the observable data $\boldsymbol{x}$ are contained in A, while the hidden variable $\boldsymbol{z}$ is the score $\boldsymbol{s}$. Of course we are interested in the posterior distribution $p(\boldsymbol{s}|A)$, which is in general (and in our case unfortunately!) uncomputable. We will use then variational inference to approximate $p(\boldsymbol{s}|A)$ as well as possible.

4.3.1 From static to dynamic Variational SpringRank

We start recalling the main assumptions made while building the probabilistic generative model of static SpringRank in [1]. The paper assumes the following prior for the ranks:

$$p(\boldsymbol{s}) \propto \prod_{i \in V} \exp(-\frac{\beta}{2}(s_i - 1)^2) \sim N(1, \frac{1}{\sqrt{\beta}}\mathbb{1}) \tag{4.21}$$

with β a simple parameter, the *inverse temperature*. The natural choice for the likelihood is instead

$$p(A|\boldsymbol{s}) = \prod_{i,j} Pois(\lambda_{ij}) = \prod_{i,j} Pois(ce^{-\frac{\beta}{2}H_{ij}}) \tag{4.22}$$

with $H_{ij} = (s_i - s_j - 1)^2$ and where c is only the *density* parameter.

We want now to extend this static model toward a dynamic framework, where the ranks $\boldsymbol{s}_t$ evolve through time. From now on so $\boldsymbol{s}_t \in \mathbb{R}^N$ will indicate the vector containing all the scores at time t. Therefore at time t the node $i \in \{1, \dots, N\}$ has rank s_{ti}. Similarly, the tensor $A \in \mathbb{N}^{T \times N \times N}$ contains the data, with A_{tij} the (directed) observations at time t from node i toward j. The main idea is to introduce a time dynamic using an autoregressive process involving some latent variables. We define the model as follow:

- **prior**

$$\begin{aligned} p(\boldsymbol{s}, \boldsymbol{\mu}) &= p(\boldsymbol{s}_0, \boldsymbol{\mu}_0) \prod_{t=1}^{T} p(\boldsymbol{s}_t, \boldsymbol{\mu}_t | \boldsymbol{s}_{t-1}, \boldsymbol{\mu}_{t-1}) \\ &= p(\boldsymbol{s}_0, \boldsymbol{\mu}_0) \prod_{t=1}^{T} p(\boldsymbol{s}_t|\boldsymbol{\mu}_t) p(\boldsymbol{\mu}_t|\boldsymbol{\mu}_{t-1}) \end{aligned} \tag{4.23}$$

  where the conditional distributions are:

$$\boldsymbol{s}_t|\boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \rho^2\mathbb{1}) \tag{4.24}$$

  and

$$\boldsymbol{\mu}_{t+1}|\boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \sigma^2\mathbb{1}) \tag{4.25}$$

  i.e. the underlying variables $\boldsymbol{\mu}_t$ assume an autoregressive process form, and the $\boldsymbol{s}_t$ is generated like in the static model, but with mean $\boldsymbol{\mu}_t$

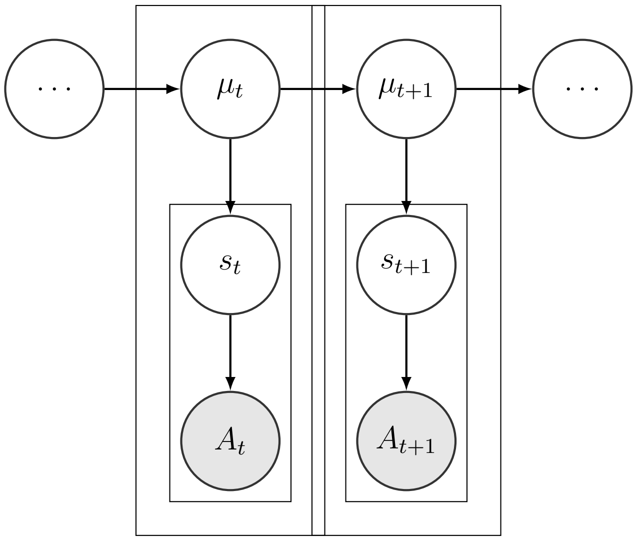


Figure 4.1: Graphical representation of the dynamic model

- **likelihood**

$$p(A|\boldsymbol{s},\boldsymbol{\mu}) = \prod_{t,i,j} Pois(\lambda_{tij} = ce^{-\frac{\beta}{2}H_{tij}})(A_{tij}) \tag{4.26}$$

  where $H_{tij} \coloneqq (s_{ti} - s_{tj} - 1)^2$

The generative model is represented in figure 4.1.

As pointed out above, the autoregressive process is not running on the ranks themselves, but rather in their means.

Variational Approximation

Clearly, we are interested in the posterior distribution of the ranks given the data observed $p(\boldsymbol{s},\boldsymbol{\mu}|A)$. The posterior of the model above anyway has not a simple form and it is not analytically computable. Therefore, one possible approach to solve this problem consists in using a variational approximation q. Similarly to [34], we use the following variational approximation:

$$q(\boldsymbol{s},\boldsymbol{\mu}) = \prod_{t,i} q(s_{ti}|\theta_{ti},\eta_{ti}^2) \cdot \prod_{t,i} q(\mu_{ti}|\hat{\mu}_{:i}) \tag{4.27}$$

where we emphasise the dependence on all the times t for $\hat{\mu}_{:i} = (\hat{\mu}_{1i}\dots\hat{\mu}_{Ti}) \in \mathbb{R}^T$. The specific shape of the q in (4.27) is

$$s_{ti}|\theta_{ti},\eta_{ti} \sim \mathcal{N}(\theta_{ti},\eta_{ti}^2) \tag{4.28}$$

$$\hat{\mu}_{ti}|\mu_{ti} \sim \mathcal{N}(\mu_{ti},\hat{\nu}_{ti}) \tag{4.29}$$

where, similarly to [34], for (4.29) we consider a Gaussian with mean and variance given by running a Kalman filter and smoother [35] on $\mu, \hat{\mu}$, with $\hat{\nu}_{ti}$ extra variational parameters regulating the variance in the Kalman structure.
As regard the filtering we have:

$$\begin{aligned}
\boldsymbol{m}_t \equiv \mathbb{E}(\boldsymbol{\mu}_t|\hat{\boldsymbol{\mu}}_{1:t}) &= \left(\frac{\hat{\boldsymbol{\nu}}_t^2}{V_{t-1}+\sigma^2+\hat{\boldsymbol{\nu}}_t^2}\right)\boldsymbol{m}_{t-1} + \left(1-\frac{\hat{\boldsymbol{\nu}}_t^2}{V_{t-1}+\sigma^2+\hat{\boldsymbol{\nu}}_t^2}\right)\hat{\boldsymbol{\mu}}_t \\
V_t \equiv \mathbb{E}((\boldsymbol{\mu}_t-\boldsymbol{m}_t)^2|\hat{\boldsymbol{\mu}}_{1:t}) &= \left(\frac{\hat{\boldsymbol{\nu}}_t^2}{V_{t-1}+\sigma^2+\hat{\boldsymbol{\nu}}_t^2}\right)(V_{t-1}+\sigma^2)
\end{aligned} \tag{4.30}$$

and for the smoothing:

$$\begin{aligned}
\tilde{\boldsymbol{m}}_{t-1} \equiv \mathbb{E}(\boldsymbol{\mu}_{t-1}|\hat{\boldsymbol{\mu}}_{1:T}) &= \left(\frac{\sigma^2}{V_{t-1}+\sigma^2}\right)\boldsymbol{m}_{t-1} + \left(1-\frac{\sigma^2}{V_{t-1}+\sigma^2}\right)\tilde{\boldsymbol{m}}_t \\
\tilde{V}_{t-1} \equiv \mathbb{E}((\boldsymbol{\mu}_{t-1}-\tilde{\boldsymbol{m}}_{t-1})^2|\hat{\boldsymbol{\mu}}_{1:T}) &= V_{t-1} + \left(\frac{V_{t-1}}{V_{t-1}+\sigma^2}\right)^2(\tilde{V}_t-(V_{t-1}+\sigma^2)).
\end{aligned} \tag{4.31}$$

Then we obtain:

$$\boldsymbol{\mu}_t|\hat{\boldsymbol{\mu}}_{1:T} \sim \mathcal{N}(\tilde{\boldsymbol{m}}_t, \tilde{V}_t) \tag{4.32}$$

To recap we have $\boldsymbol{\theta}_t$, $\boldsymbol{\eta}_t$, $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{\nu}}_t$ as variational parameters and ρ, σ, β, c as model hyperparameters.

Reduction of the variables

Since the structure of our problem is simpler than the one in [34], we can also try to get rid of some of the variables introduced above. Given that we are interested in the ranks $\boldsymbol{s}_t$ and in order to reduce the number of parameters, instead of making $\boldsymbol{\mu}_t$ evolve through time we can directly run the Kalman filter on $\boldsymbol{s}_t$, removing $\boldsymbol{\mu}_t$ from the model (see Figure 4.2). The structure of the model is still the same

$$q(\boldsymbol{s}) = \prod_{t,i} q(s_{ti}|\hat{s}_{:i}) \tag{4.33}$$

with $\hat{s}_{ti}$ corresponding to the previous $\hat{\mu}_{ti}$. The variational parameters are then only $\hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{\nu}}$ ($\hat{\boldsymbol{\nu}}$ has same role as before), while the model parameters are σ, β and c. In the following we will derive the equations for the more general extended model, but all the expressions and techniques presented below have been tried also for the reduced model, which has revealed most of the time to be not only faster but also more accurate than the former.

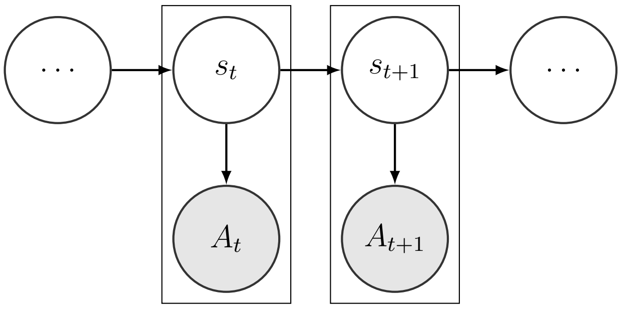


Figure 4.2: Graphical representation of the reduced form dynamic model

Computation of the ELBO

We can explicitly calculate the ELBO for this model and apply direct optimization to find the best approximation q (of the form described above) of the unknown posterior $p(\boldsymbol{s}, \boldsymbol{\mu}|A)$. If a closed form for the ELBO is available then, it is advisable to directly optimize it without passing from stochastic approximations of the expected values. We start from equations 4.23, 4.26 and 4.27 to compute

$$\mathbf{ELBO}(q) = \mathbb{E}_q[\log p(\boldsymbol{s}, \boldsymbol{\mu})] + \mathbb{E}_q[\log p(A|\boldsymbol{s}, \boldsymbol{\mu})] - \mathbb{E}_q[\log q(\boldsymbol{s}, \boldsymbol{\mu})] \tag{4.34}$$

First addend: using (4.24), (4.25), (4.27) and (1):

$$\begin{aligned}
\mathbb{E}_q[\log p(\boldsymbol{s}, \boldsymbol{\mu})] &= \int \sum_{t,i} \log[p(s_{ti}|\mu_{ti})p(\mu_{ti}|\mu_{t-1i})]q(\boldsymbol{s}, \boldsymbol{\mu})d\boldsymbol{s}d\boldsymbol{\mu} \\
&= \int \sum_{t,i} \log[\frac{1}{\sqrt{2\pi\rho^2}} e^{\frac{-(s_{ti}-\mu_{ti})^2}{2\rho^2}}]q(\boldsymbol{s}, \boldsymbol{\mu})d\boldsymbol{s}d\boldsymbol{\mu} \\
&+ \int \sum_{t,i} \log[\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\mu_{ti}-\mu_{t-1,i})^2}{2\sigma^2}}]q(\boldsymbol{\mu})d\boldsymbol{\mu} \\
&= -\frac{1}{2\rho^2} \int \sum_{t,i} [(s_{ti} - \mu_{ti})^2] q(\boldsymbol{s}, \boldsymbol{\mu})d\boldsymbol{s}d\boldsymbol{\mu} \\
&- \frac{1}{2\sigma^2} \int \sum_{t,i} [(\mu_{ti} - \mu_{t-1,i})^2] q(\boldsymbol{\mu})d\boldsymbol{\mu} - \frac{1}{2} TN \log(4\pi^2\sigma^2\rho^2) \\
&= -\frac{1}{2\rho^2} \sum_{t,i} (\eta_{ti}^2 + \tilde{V}_{ti} + (\theta_{ti} - \tilde{m}_{ti})^2) \\
&- \frac{1}{2\sigma^2} \sum_{t,i} (\tilde{V}_{ti} + \tilde{V}_{t-1i} + (\tilde{m}_{ti} - \tilde{m}_{t-1,i})^2) - \frac{1}{2} TN \log(4\pi^2\sigma^2\rho^2)
\end{aligned} \tag{4.35}$$

Second addend: using equations 1, 2:

$$\begin{aligned}
\mathbb{E}_q[\log p(A|\boldsymbol{s},\boldsymbol{\mu})] &= \mathbb{E}_q \left[\sum_{t,i,j} \log(\frac{\lambda_{tij}^{A_{tij}} e^{-\lambda_{tij}}}{A_{tij}!}) \right] = \sum_{t,i,j} \mathbb{E}_q \left[A_{tij}(\log c - \frac{\beta}{2} s_{tij}^2) - c e^{-\frac{\beta}{2} s_{tij}^2} \right] + C \\
&= \sum_{t,i,j} \mathbb{E}_q \left[-\frac{\beta A_{tij}}{2} s_{tij}^2 - c e^{-\frac{\beta}{2} s_{tij}^2} \right] + \sum_{t,i,j} A_{tij} \log c + C \\
&= -\frac{\beta}{2} \sum_{t,i,j} A_{tij} \left(\eta_{ti}^2 + \eta_{tj}^2 + (\theta_{ti} - \theta_{tj} - 1)^2 \right) + \\
&- c \sum_{t,i,j} \frac{1}{\sqrt{1 + \beta(\eta_{ti}^2 + \eta_{tj}^2)}} \exp\left(-\frac{\beta}{2} \cdot \frac{(\theta_{ti} - \theta_{tj} - 1)^2}{1 + \beta(\eta_{ti}^2 + \eta_{tj}^2)} \right) + \log c \sum_{t,i,j} A_{tij} + C
\end{aligned} \tag{4.36}$$

Third addend: the third addend is just the entropy of a factorized gaussian. In fact in the variational distribution, given the "variational observations" $\hat{\mu}_{ti}$, the latent variables μ_{ti} are independent gaussian with mean, variance $\tilde{\mu}_{ti}, \tilde{V}_{ti}$. Therefore we obtain:

$$-\mathbb{E}_q[\log q(\boldsymbol{s}, \boldsymbol{\mu})] = H(q) = \frac{1}{2} \sum_{t,i} \log(2\pi e \eta_{ti}^2) + \log(2\pi e \tilde{V}_{ti}) \tag{4.37}$$

The final results is

$$\begin{aligned}
\mathbf{ELBO}(q) = &-\frac{1}{2\rho^2} \sum_{t,i} (\eta_{ti}^2 + \tilde{V}_{ti} + (\theta_{ti} - \tilde{m}_{ti})^2) - \frac{1}{2\sigma^2} \sum_{t,i} (\tilde{V}_{ti} + \tilde{V}_{t-1i} + (\tilde{m}_{ti} - \tilde{m}_{t-1,i})^2) + \\
&+ \sum_{t,i,j} -\frac{\beta}{2} A_{tij} \left(\eta_{ti}^2 + \eta_{tj}^2 + (\theta_{ti} - \theta_{tj} - 1)^2\right) + \\
&- c \frac{1}{\sqrt{1 + \beta(\eta_{ti}^2 + \eta_{tj}^2)}} \exp\left(-\frac{\beta}{2} \cdot \frac{(\theta_{ti} - \theta_{tj} - 1)^2}{1 + \beta(\eta_{ti}^2 + \eta_{tj}^2)}\right) + \\
&+ \frac{1}{2} \sum_{t,i} \log(2\pi e \eta_{ti}^2) + \log(2\pi e \tilde{V}_{ti}) + \\
&- \frac{1}{2} TN \log(4\pi^2 \sigma^2 \rho^2) + \log c \sum_{t,i,j} A_{tij} + const
\end{aligned} \tag{4.38}$$

4.3.2 Modifying the likelihood

The assumptions underlying the SpringRank model are not satisfied in some real life experimental settings. While studying animal behaviour, for example, we assume that individuals only interact with others close to them in ranking. Also considering social relations we can reasonably consider the bare existence of an interaction as an useful information of the ranking or social status. All this a priori knowledge on the problem is exploited in the construction of the likelihood of the SpringRank generative model. Anyway, when these assumptions are not fulfilled, we can easily change the shape of the likelihood and adapt it to the specific application. In particular, in many datasets the bare fact of seeing an interaction between two nodes does not mean that the two nodes are close in rank. Taking for example sport contexts, every team plays against another one following a random schedule; the same for social networks, where a "follow" relationship can be shared between nodes quite far in the ranking. In particular, the exponent in the likelihood (4.26) $-\frac{\beta}{2}(s_{ti} - s_{tj} - 1)^2$ must be corrected. The square, with the minus in front of it, in fact penalises all the directed interactions $i \to j$, i beats j, for every couple i and j far in ranking. Instead, we look for the opposite behaviour in sports, allowing strong teams to collect several victories against the weakest ones. We propose then a different likelihood, given as follows:

$$p(A|\boldsymbol{s},\boldsymbol{\mu}) = \prod_{t,i,j} Pois\left(\lambda_{tij} = c\exp\left(\frac{\beta}{2}(s_{ti} - s_{tj})\right)\right)$$

In this way if $s_i \gg s_j$ then λ is big and we expect a high number of victories of i against j. Looking at the form of the `ELBO`, we see that the likelihood is just involved in the second term. For this reason we can rewrite it very easily. With simple calculations, similar to the ones above, the second addend of (4.34) becomes:

$$\mathbb{E}_q[\log p(A|\boldsymbol{s},\boldsymbol{\mu})] = \log c \sum_{t,i,j} A_{tij} + \frac{\beta}{2}\sum_{t,i,j} A_{tij}(\theta_{ti} - \theta_{tj}) - c\sum_{t,i,j}\exp\left(\frac{\beta}{2}(\theta_{ti} - \theta_{tj}) + \frac{\beta^2}{8}(\eta_{ti}^2 + \eta_{tj}^2)\right)$$

and the final `ELBO` becomes

$$\begin{aligned}
\mathbf{ELBO}(q) = &-\frac{1}{2\rho^2}\sum_{t,i}(\eta_{ti}^2 + \tilde{V}_{ti} + (\theta_{ti} - \tilde{m}_{ti})^2) - \frac{1}{2\sigma^2}\sum_{t,i}(\tilde{V}_{ti} + \tilde{V}_{t-1i} + (\tilde{m}_{ti} - \tilde{m}_{t-1,i})^2) \\
&+ \frac{\beta}{2}\sum_{t,i,j} A_{tij}(\theta_{ti} - \theta_{tj}) - c\sum_{t,i,j}\exp\left(\frac{\beta}{2}(\theta_{ti} - \theta_{tj}) + \frac{\beta^2}{8}(\eta_{ti}^2 + \eta_{tj}^2)\right) \\
&+ \frac{1}{2}\sum_{t,i}\log(2\pi e\eta_{ti}^2) + \log(2\pi e\tilde{V}_{ti}) \\
&- \frac{1}{2}TN\log(4\pi^2\sigma^2\rho^2) + \log c\sum_{t,i,j} A_{tij} + const
\end{aligned} \tag{4.39}$$

4.4 Binomial Likelihood

Specially when considering sport contexts, the focus is not on predicting the existence of the matches, but on forecasting their outcomes. That is because we know the exact random schedule of the championship from the beginning, while of course we do not know the future results. From this perspective, it seems quite reasonable to modify our model acting once again on the likelihood. In particular, if we exactly know the total number of matches between i and j at time t and we only need to partition this number into the amount of victories of i against j and viceversa. The natural distribution for this likelihood is then a binomial:

$$p(A|\boldsymbol{s},\boldsymbol{\mu}) = \prod_{tij} Bin(p_{tij}, n_{tij}) \tag{4.40}$$

where $p_{tij} = \frac{1}{1+e^{-2\beta(s_i-s_j)}}$ and $n_{tij} = A_{tij} + A_{tji}$. We rewrite then the second term of the likelihood:

$$\begin{aligned}
&\mathbb{E}_q[\log p(A|\boldsymbol{s},\boldsymbol{\mu})] = \\
&\quad = \mathbb{E}_q\left[\sum_{tij} \log\left[\binom{A_{tij}+A_{tji}}{A_{tij}}\left(\frac{1}{1+e^{-2\beta(s_i-s_j)}}\right)^{A_{tij}}\left(\frac{1}{1+e^{-2\beta(s_j-s_i)}}\right)^{A_{tji}}\right]\right] = \\
&\quad = -2\sum_{tij} A_{tij}\mathbb{E}_q\left[\log\left(1+e^{-2\beta(s_i-s_j)}\right)\right]
\end{aligned} \tag{4.41}$$

Unfortunately this integral is not computable in closed form and we cannot simply apply gradient ascent as before to maximize the ELBO.

4.4.1 Black-Box Variational inference

Since integral (4.41) is not computable in closed form we need to find another way to maximize the `ELBO`, different from simple gradient ascent. The idea here is to use instead stochastic optimization, which maximize a function using noisy estimates of its gradient. To do that we need to rewrite the derivative of the objective as an expectation with respect to the variational approximation and then sample from the variational approximation to get noisy but unbiased gradients, which we use to update our parameters.

Stochastic gradient derivation

To optimize the `ELBO` with stochastic optimization, we need to develop an unbiased estimator of its gradient which can be computed from samples from the variational distribution. The key idea is that the gradient of the ELBO can be written as an expectation with respect to the variational distribution. We start by differentiating the ELBO definition (4.8) with respect to the variational parameters,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \texttt{ELBO} &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}} \left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] \\
&= \nabla_{\boldsymbol{\lambda}} \int \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} \\
&= \int \nabla_{\boldsymbol{\lambda}} \left[\left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] d\boldsymbol{z} \\
&= \int \nabla_{\boldsymbol{\lambda}} \left[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} \\
&\quad + \int \nabla_{\boldsymbol{\lambda}} q_{\boldsymbol{\lambda}}(\boldsymbol{z}) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) d\boldsymbol{z} \\
&= -\mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] \\
&\quad + \int \nabla_{\boldsymbol{\lambda}} q_{\boldsymbol{\lambda}}(\boldsymbol{z}) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) d\boldsymbol{z} \qquad (4.42)
\end{aligned}$$

where we have exchanged derivatives with integrals via the *dominated convergence theorem* and used $\nabla_{\boldsymbol{\lambda}} \left[\log p(\boldsymbol{x}, \boldsymbol{z})\right] = 0$. The first term in (4.42) is zero in fact

$$\begin{aligned}
\mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] = \mathbb{E}_q \left[\frac{\nabla_{\boldsymbol{\lambda}} q_{\boldsymbol{\lambda}}(\boldsymbol{z})}{q_{\boldsymbol{\lambda}}(\boldsymbol{z})}\right] &= \int \nabla_{\boldsymbol{\lambda}} q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} \\
&= \nabla_{\boldsymbol{\lambda}} \int q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} = \nabla_{\boldsymbol{\lambda}} 1 = 0. \qquad (4.43)
\end{aligned}$$

To simplify the second term, first observe that $\nabla_{\boldsymbol{\lambda}} \left[q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] = \nabla_{\boldsymbol{\lambda}} \left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] q_{\boldsymbol{\lambda}}(\boldsymbol{z})$. This fact gives us the gradient as an expectation,

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \texttt{ELBO} &= \int \nabla_{\boldsymbol{\lambda}} q_{\boldsymbol{\lambda}}(\boldsymbol{z}) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) d\boldsymbol{z} \\
&= \int \nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right) q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} \\
&= \mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right)\right]. \qquad (4.44)
\end{aligned}$$

Now we can compute noisy unbiased gradients of the ELBO with Monte Carlo samples from the variational distribution,

$$\nabla_{\boldsymbol{\lambda}} \texttt{ELBO} \approx \frac{1}{S} \sum_{i=1}^{S} \nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}_s) \Big(\log p_i(\boldsymbol{x}, \boldsymbol{z}_s) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{z}_s) \Big), \qquad (4.45)$$

$$\text{where } \boldsymbol{z}_s \sim q_{\boldsymbol{\lambda}}(\boldsymbol{z}).$$

Note also that the score function $\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})$ and the sampling algorithms depend only on the variational distribution, not the underlying model. Thus one can easily build up a collection of these functions for various variational approximations and reuse them in a package for a variety of models. Further this method doesn't require any assumptions about the form of the model, only that the practitioner can compute the log of the joint $p(\boldsymbol{x}, \boldsymbol{z}_s)$. This algorithm significantly reduces the effort needed to implement variational inference in a wide variety of models.

Application to our binomial model

As explained above, in the variational model with binomial likelihood we cannot compute the `ELBO` in closed form so we will use the black-box argument just described. More precisely, the problem is only in the second term of the `ELBO` in the form (4.11) -i.e. the likelihood term indeed- so we can adapt (4.44):

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \texttt{ELBO} &= \nabla_{\boldsymbol{\lambda}} \left[\mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{z})\right] + \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right]\right] \\
&= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{z})\right] + \nabla_{\boldsymbol{\lambda}} \int \log(p(\boldsymbol{x}|\boldsymbol{z})) q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} - \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q(\boldsymbol{z})\right] \\
&= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{z})\right] + \int \nabla_{\lambda}\left[\log(p(\boldsymbol{x}|\boldsymbol{z})) q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] d\boldsymbol{z} - \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] \\
&= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{z})\right] + \int \log p(\boldsymbol{x}|\boldsymbol{z}) \nabla_{\boldsymbol{\lambda}}\left[\log(q_{\boldsymbol{\lambda}}(\boldsymbol{z}))\right] q_{\boldsymbol{\lambda}}(\boldsymbol{z}) d\boldsymbol{z} - \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] \\
&= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{z})\right] + \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{x}|\boldsymbol{z}) \nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right] - \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{z})\right]
\end{aligned} \tag{4.46}$$

where we haven't touched the first and third terms since we know the exact expression of the two expectations, see (4.35) and (4.37). We also emphasized the dependence of q on the variational parameters $\boldsymbol{\lambda}$, with $\boldsymbol{\lambda}$ equal to $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\nu}}$ in our case. Applying (4.46) to our problem we obtain:

$$\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \texttt{ELBO} &= \nabla_{\boldsymbol{\lambda}} \left[\mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{s}, \boldsymbol{\mu})\right] + \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(A|\boldsymbol{s}, \boldsymbol{\mu})\right] - \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q(\boldsymbol{s}, \boldsymbol{\mu})\right]\right] \\
&= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(\boldsymbol{s}, \boldsymbol{\mu})\right] + \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log p(A|\boldsymbol{s}, \boldsymbol{\mu}) \nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{s}, \boldsymbol{\mu})\right] - \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}}\left[\log q_{\boldsymbol{\lambda}}(\boldsymbol{s}, \boldsymbol{\mu})\right]
\end{aligned} \tag{4.47}$$

We can finally use this stochastic gradient in a stochastic optimization algorithm to optimize the variational parameters.

4.4.2 Reducing the variance

The main problem with the above technique is that, since we are using stochastic estimates of the gradient, the variance can be high. As shown in [36], reducing the variance of the gradient estimator is essential to the fast convergence of our algorithm. We present here two strategies for controlling the variance that we used to improve our model. The first is based on Rao-Blackwellization [37], which exploits the factorization of the variational distribution. The second is based on control variates [38, 39], using the log probability of the variational distribution.

Rao-Blackwellization Rao-Blackwellization [37] reduces the variance of a random variable by replacing it with its conditional expectation with respect to a subset of the variables. This generally requires analytically computing problem-specific integrals. Here we show how to Rao-Blackwellize the estimator for each component of the gradient without needing to compute model-specific integrals. In the simplest setting, Rao-Blackwellization replaces a function of two variables with its conditional expectation. Consider two random variables, X and Y , and a function $J(X, Y)$. Our goal is to compute its expectation $\mathbb{E}\left[J(X, Y)\right]$ with respect to the joint distribution of X and Y. Defining $\hat{J}(X) = \mathbb{E}\left[J(X, Y)|X)\right]$ we can see that $\mathbb{E}\left[\hat{J}(X)\right] = \mathbb{E}\left[J(X, Y)\right]$, which means that $\hat{J}(X)$ can be used in place of $J(X, Y)$ in a Monte Carlo approximation of $\mathbb{E}\left[J(X, Y)\right]$. What makes $\hat{J}(X)$ more suitable than $J(X, Y)$ anyway is its lower variance, in fact

$$\begin{aligned}
\mathrm{Var}\left[\hat{J}(X)\right] &= \mathbb{E}\left[\hat{J}(X)^2\right] - \mathbb{E}\left[\hat{J}(X)\right]^2 \\
&= \mathbb{E}\left[\hat{J}(X)^2\right] - \mathbb{E}\left[\mathbb{E}\left[J(X, Y)|X\right]\right]^2 \\
&= \mathbb{E}\left[\hat{J}(X)^2\right] + \mathrm{Var}\left[J(X, Y)\right] - \mathbb{E}\left[J(X, Y)^2\right] \\
&= \mathrm{Var}\left[J(X, Y)\right] - \mathbb{E}\left[J(X, Y)^2 - \hat{J}(X)^2\right] \\
&= \mathrm{Var}\left[J(X, Y)\right] - \mathbb{E}\left[(J(X, Y) - \hat{J}(X))^2\right] \qquad (4.48)
\end{aligned}$$

where for the last equality we used

$$\begin{aligned}
-2\mathbb{E}\left[(J(X, Y)\hat{J}(X)\right] &= -2\mathbb{E}\big[(J(X, Y)\mathbb{E}\left[J(X, Y)|X)\right]\big] \\
&= -2\mathbb{E}\big[\mathbb{E}\left[J(X, Y)\mathbb{E}\left[J(X, Y)|X\right]|X\right]\big] \\
&= -2\mathbb{E}\big[\mathbb{E}\left[J(X, Y)|X\right]\mathbb{E}\left[J(X, Y)|X\right]\big] \\
&= -2\mathbb{E}\left[\hat{J}(X)^2\right] \qquad (4.49)
\end{aligned}$$

In our problem we want to estimate the gradient of the `ELBO` function. In the case we are using the mean-field variational family, we have that each random variable z_i is independent and governed by its own variational distribution, $q_{\boldsymbol{\lambda}}(\boldsymbol{z}) = \prod_{i=1}^{n} q(z_i; \lambda_i)$. Consider the i-th component of the gradient. Let $q_{(i)}$ be the distribution of variables in the model that depend on the i-th variable, i.e., the Markov blanket of z_i; and let $p_i(x, z_{(i)})$ be the terms in the joint that depend on those variables. To compute the Rao-Blackwellized estimators, we need to compute conditional expectations. Due to the mean field-assumption, the conditional expectation simplifies due to the factorization

$$\begin{aligned}\mathbb{E}\left[J(X,Y)|X\right] &= \frac{\int J(x,y)p(x,y)dy}{\int p(x,y)dy} = \frac{\int J(x,y)p(x)p(y)dy}{\int p(x)p(y)dy} \\ &= \int J(x,y)p(y)dy = \mathbb{E}_y\left[J(X,Y)\right] \end{aligned} \tag{4.50}$$

Therefore, to construct a lower variance estimator when the joint distribution factorizes, all we need to do is integrate out some variables. We start writing the i-th component of the gradient

$$\begin{aligned}\nabla_{\lambda_i}\texttt{ELBO} &= \mathbb{E}_{q_1}...\mathbb{E}_{q_n}\left[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Big(\log p(\boldsymbol{x},\boldsymbol{z}) - \sum_{j=1}^{n}\log q_j(z_j;\lambda_j)\Big)\right] \\ &= \mathbb{E}_{q_1}...\mathbb{E}_{q_n}\left[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Big(\log p_i(\boldsymbol{x},z_{(i)}) + \log p_{-i}(\boldsymbol{x},\boldsymbol{z}) - \sum_{j=1}^{n}\log q_j(z_j;\lambda_j)\Big)\right] \\ &= \mathbb{E}_{q_i}\Bigg[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Bigg(\mathbb{E}_{q_{-i}}\Big[\log p_i(\boldsymbol{x},z_{(i)})\Big] - \log q_i(z_i;\lambda_i) \\ &\quad + \mathbb{E}_{q_{-i}}\Big[\log p_{-i}(\boldsymbol{x},\boldsymbol{z}) - \sum_{j=1,\, i\neq j}^{n}\log q_j(z_j;\lambda_j)\Big]\Bigg)\Bigg] \\ &= \mathbb{E}_{q_i}\Bigg[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Bigg(\mathbb{E}_{q_{-i}}\Big[\log p_i(\boldsymbol{x},z_{(i)})\Big] - \log q_i(z_i;\lambda_i) + C_i\Bigg)\Bigg] \\ &= \mathbb{E}_{q_i}\Bigg[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Bigg(\mathbb{E}_{q_{-i}}\Big[\log p_i(\boldsymbol{x},z_{(i)})\Big] - \log q_i(z_i;\lambda_i)\Bigg)\Bigg] \\ &= \mathbb{E}_{q_{(i)}}\left[\nabla_{\lambda_i}\log q_i(z_i;\lambda_i)\Big(\log p_i(\boldsymbol{x},z_{(i)}) - \log q_i(z_i;\lambda_i)\Big)\right] \end{aligned} \tag{4.51}$$

where we have leveraged the mean field assumption and made use of the identity for the expected score (4.43). This means we can Rao-Blackwellize the gradient of the variational parameter λ_i with respect to the the latent variables outside of the Markov blanket of z_i without needing model specific computations.
Finally, we construct a Monte Carlo estimator for the gradient of λ_i using samples

from the variational distribution,

$$\frac{1}{S}\sum_{i=1}^{S} \nabla_{\lambda_i} \log q_i(\boldsymbol{z}_s;\lambda_i)\Big(\log p_i(\boldsymbol{x},\boldsymbol{z}_s) - \log q_i(\boldsymbol{z}_s;\lambda_i)\Big), \quad \text{where } \boldsymbol{z}_s \sim q_{(i)}(\boldsymbol{z};\boldsymbol{\lambda}) \tag{4.52}$$

This Rao-Blackwellized estimator for each component of the gradient has lower variance.

Control Variates As we saw above, variance reduction methods work by replacing the function whose expectation is being approximated by Monte Carlo with another function that has the same expectation but smaller variance. That is, to estimate $\mathbb{E}_q[f]$ via Monte Carlo we compute the empirical average of $\hat{f}$ where $\hat{f}$ is chosen so $\mathbb{E}_q[f] = \mathbb{E}_q[\hat{f}]$ and $\mathrm{Var}_q[f] > \mathrm{Var}_q[\hat{f}]$. We start defining the term control variate [38]. A control variate is simply a family of functions with equivalent expectation. Consider for example a function h, which has a finite first moment, and a scalar a. Define $\hat{f}$ to be

$$\hat{f}(z) := f(z) - a\big(h(z) - \mathbb{E}\left[h(z)\right]\big). \tag{4.53}$$

This is a family of functions, indexed by a, and note that, by construction, $\mathbb{E}_q[\hat{f}(z)] = \mathbb{E}_q[f]$ as required. Given a particular function h, we can choose a to minimize the variance of $\hat{f}$. First, using basic properties of the variance, we can write

$$\mathrm{Var}(\hat{f}) = \mathrm{Var}(f) + a^2\mathrm{Var}(h) - 2a\mathrm{Cov}(f,h). \tag{4.54}$$

This equation implies that good control variates have high covariance with the function whose expectation is being computed.
Taking the derivative of $\mathrm{Var}(\hat{f})$ with respect to a and setting it equal to zero gives us the value of a that minimizes the variance,

$$a^* = \frac{\mathrm{Cov}(f/h)}{\mathrm{Var}(h)}. \tag{4.55}$$

With Monte Carlo estimates from the distribution, we can estimate a^* with the ratio of the empirical covariance and variance. We now apply this method to Black Box Variational Inference. To maintain the generic nature of the algorithm, we want to choose a control variate that only depends on the variational distribution and for which we can easily compute its expectation. Meeting these criteria, we choose h to be the score function of the variational approximation, $\nabla\lambda \log q(z)$, which always has expectation zero. With this control variate, we have a new Monte Carlo method to compute the Rao-Blackwellized noisy gradients of the `ELBO`. For the i-th component of the gradient, the function whose expectation is being estimated is the one found in 4.44

$$f_i(\boldsymbol{z}) = \nabla_{\lambda_i} \log q(\boldsymbol{z}; \lambda_i) \left(\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \lambda_i)\right) \tag{4.56}$$

and we defined its control variates

$$h_i(\boldsymbol{z}) = \nabla_{\lambda_i} \log q(\boldsymbol{z}; \lambda_i). \tag{4.57}$$

The estimate for the optimal choice for the scaling is given by summing over the covariance and variance for each of the n_i dimensions of λ_i. Letting the d-th dimension of f_i and h_i be f_i^d and h_i^d respectively. The optimal scaling for the gradient of the `ELBO` is given by

$$\hat{a}_i^* = \frac{\sum_{d=1}^{n_i} \hat{\text{Cov}}(f_i^d, h_i^d)}{\sum_{d=1}^{n_i} \hat{\text{Var}}(h_i^d)} \tag{4.58}$$

This gives us the following Monte Carlo method to compute noisy gradients using S samples

$$\hat{\nabla}_{\lambda_i} \texttt{ELBO} = \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q_i(\boldsymbol{z}_s; \lambda_i) \Big(\log p_i(\boldsymbol{x}, \boldsymbol{z}_s) - \log q_i(\boldsymbol{z}_s; \lambda_i) - \hat{a}_i^* \Big), \quad \text{where } \boldsymbol{z}_s \sim q_{(i)}(\boldsymbol{z}; \boldsymbol{\lambda}) \tag{4.59}$$

4.4.3 Reparameterization trick

We complete this part mentioning also another technique we used in this work to solve the problem of calculating $\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}} \texttt{ELBO}$. This method exploits a simple reparameterization trick to pass the gradient inside the expectation, returning a much more tractable expression.
Suppose we want to compute

$$\nabla_\phi f(\omega, \phi) = \nabla_\phi \mathbb{E}_{q(t|x,\phi)} \log(x|t, \omega) \tag{4.60}$$

where $t \sim q(t|x, \phi) = \mathcal{N}(m, s^2)$. Now we write $t = s\,\epsilon + m = g(\epsilon, x, \phi)$, with $\epsilon \sim p(\epsilon) = \mathcal{N}(0, 1)$. Since we can simply pass from samples extracted from $q(t)$ to samples extracted from $p(\epsilon)$ through the deterministic function g, using this change of variables we have

$$\begin{aligned}\nabla_\phi f(\omega,\phi) &= \nabla_\phi \mathbb{E}_{q(t|x,\phi)} \log(x|t,\omega)\\ &= \nabla_\phi \mathbb{E}_{p(\epsilon)} \log(x|g(\epsilon,x,\phi),\omega)\\ &= \int p(\epsilon)\nabla_\phi \log(x|g(\epsilon,x,\phi),\omega)\\ &= \mathbb{E}_{p(\epsilon)}\left[\nabla_\phi \log(x|g(\epsilon,x,\phi),\omega)\right] \end{aligned} \tag{4.61}$$

where we use the fact that $p(\epsilon)$ does not depend anymore on ϕ. So now, sampling from a standard normal $p(\epsilon)$ we can obtain the stochastic approximation

$$\tilde{\nabla} f(\omega,\phi) = \frac{1}{S}\sum_{s=1}^{S} \nabla_\phi \log(x|g(\epsilon_s,x,\phi),\omega) \qquad \text{where } \epsilon_s \sim \mathcal{N}(0,1). \tag{4.62}$$

Going back to our problem $\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{q_{\boldsymbol{\lambda}}}\texttt{ELBO}$: consider $\lambda = (\mu,\sigma)$ and $z \sim q_{(\mu,\sigma)}(z) = \mathcal{N}(z|\mu,\sigma)$, then we can write $z = \sigma\epsilon + \mu$, with $\epsilon \sim p(\epsilon) = \mathcal{N}(0,1)$

$$\begin{aligned}\nabla_{(\mu,\sigma)}\texttt{ELBO} &= \nabla_{(\mu,\sigma)}\mathbb{E}_{q_{(\mu,\sigma)}}\left[\log p(x,z) - \log q_{(\mu,\sigma)}(z)\right]\\ &= \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}\Big[\nabla_{(\mu,\sigma)}\left[\log p(x,\sigma\epsilon+\mu) - \log q_{(\mu,\sigma)}(\sigma\epsilon+\mu)\right]\Big]. \end{aligned} \tag{4.63}$$

Once again then we can obtain a stochastic approximation of the desired gradient sampling from a standard normal

$$\tilde{\nabla}_{(\mu,\sigma)}\texttt{ELBO} = \frac{1}{S}\sum_{s=1}^{S}\nabla_{(\mu,\sigma)}\left[\log p(x,\sigma\epsilon+\mu) - \log q_{(\mu,\sigma)}(\sigma\epsilon+\mu)\right] \tag{4.64}$$

$$\text{where } \epsilon_s \sim \mathcal{N}(0,1).$$

This method, when applicable, is considered the best in terms of keeping a low variance.

4.4.4 Mean-field and correction of the instability

Applying this framework to our model it is clear that we still have some problems of high variance and instability while computing the stochastic gradient. This affects inevitably the quality of the parameters' optimization and consequently the accuracy of the predictions. The flaw in the procedure is probably that, looking at our variational approximation q more carefully, we are not really in the mean-field family. Because of the Kalman smoother running through all the time t, expression

(4.27), reported here:

$$q(\boldsymbol{s}, \boldsymbol{\mu}) = \prod_{t,i} q(s_{ti}; \theta_{ti}, \eta_{ti}^2) \cdot \prod_{t,i} q(\mu_{ti}; \hat{\mu}_{:i}), \tag{4.65}$$

is not completely factorized in the term $q(\mu_{ti}; \hat{\mu}_{:i})$. This can be a problem while applying Rao-Blackwellization and the control variates technique explained above.

Mean-field approximation Even if the structure presented above, given the generative model underlying SpringRank, is the natural formulation of the problem, we have issues due to high variance and instability. We can avoid this, in the case of the binomial likelihood where the `ELBO` function is not directly computable, applying some further modifications. First of all, to speed up the computations and simplify the model, we remove completely here the Kalman filter and smoother. Now we can define the variational approximation q using pure mean-field assuring complete independence between variables in q

$$q(\boldsymbol{s}) = \prod_{ti} q(s_{ti}; m_{ti}, V_{ti}) \tag{4.66}$$

where as always the correlation of the scores with their past is still present when optimizing the whole `ELBO` function. All the previous regularizing techniques are now perfectly adapt and efficient in this new formulation.

4.4.5 Optimize the model parameters

We just saw how to optimize the variational parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\nu}}$ by maximizing the `ELBO`, nevertheless we still have some model parameters[1] ρ, σ, β, c in the prior (4.24)-(4.25) and in the likelihood (4.26) that we need to tune. For doing that we use the variational EM procedure introduced in section 4.2. In fact, as explained before, we can run a variational EM optimization loop, by alternating inference on q and optimization of the hyperparameters. Actually, from exploratory results, the best choice is to optimize only the likelihood related parameters β and c, keeping the prior's ones fixed.

[1] c parameter is present only in the Poisson-likelihood model (4.38)

Algorithm 2 EM variational algorithm

```
1: Initialize randomly $\boldsymbol{\theta}, \boldsymbol{\eta}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\nu}}, \rho, \sigma, \beta, c$
2: repeat
3:     E-step:
           $$\boldsymbol{\theta}^*, \boldsymbol{\eta}^*, \hat{\boldsymbol{\mu}}^*, \hat{\boldsymbol{\nu}}^* = \underset{\boldsymbol{\theta}, \boldsymbol{\eta}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\nu}}}{\operatorname{argmax}} \texttt{ELBO}(\boldsymbol{\theta}, \boldsymbol{\eta}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\nu}}, \rho, \sigma, \beta, c) \quad (4.67)$$
4:     M-step:
           $$\rho^*, \sigma^*, \beta^*, c^* = \underset{\rho, \sigma, \beta, c}{\operatorname{argmax}} \texttt{ELBO}(\boldsymbol{\theta}^*, \boldsymbol{\eta}^*, \hat{\boldsymbol{\mu}}^*, \hat{\boldsymbol{\nu}}^*, \rho, \sigma, \beta, c) \quad (4.68)$$
5:     $\rho, \sigma, \beta, c = \rho^*, \sigma^*, \beta^*, c^*$
6: until convergence
```

4.5 Results

We present here some results for the models introduced above, using the same NBA dataset described in the first part (see Table 4.1). As before, these results are only partial and the models must be tested also against data taken from other areas. Among the various models and small modifications mentioned above, we report only the ones with the most interesting results:

model	**likel**	**var par**	**mod par**	**acc**	**agony**	σ_a	σ_L
poiss compl	poisson	$\theta, \eta, \hat{\mu}, \hat{\nu}$	ρ, σ, β, c	0.640	3.154	0.631	-1.328
poiss reduced	poisson	$\hat{s}, \hat{\nu}$	σ, β, c	0.632	3.118	0.632	-1.319
binom compl	binom	$\theta, \eta, \hat{\mu}, \hat{\nu}$	ρ, σ, β	0.495	5.243	0.494	-1.694
binom MF	binom	m, V	σ, β	**0.649**	**2.998**	**0.647**	**-1.255**

Table 4.1: Results for some of the above presented variational models: we present in this table the results for some of the principal models introduced in Section 4.3; the columns contains the type of model, the form of the likelihood, the variational parameters and model parameters chosen; the last for columns are the metrics used for the comparison.

As expected, the model with binomial likelihood has very low performance due to the problems of high variance and instability mentioned above. The correction to the pure mean-field approximation works instead efficiently, proving that techniques mentioned in Section 4.4.2 are useful when dealing with mean-field family. This model is the best among the ones tried at the moment and quite competitive also compared to Table 3.3.

5. Conclusions and Future Research

In this thesis we presented some extensions of static SpringRank model toward a dynamic framework, where ranks vary through time. In the first part we have adapted the static Hamiltonian of [1] to a time-varying setting, introducing an external field acting on ranks consequent in time. In the second part we discussed approximate inference, necessary when calculating the exact posterior is unfeasible. We developed a Bayesian model using some modern techniques for variational inference and we finally tested the models against real NBA dataset. The performances of our models are comparable with the principal state-of-art rating systems according to various metric.

Much more work can be done testing and adapting these algorithms to other contexts, from biology to social sciences. As shown before, while keeping fixed the basic structure, better results are reached adjusting the form of the likelihood to the specific application. We intend also to apply some of the newest techniques in optimization to solve the problems of instability encountered while using stochastic gradient descent in a non mean-field framework. Significant improvements to the accuracy of the predictions can be reached introducing covariates inside the model. Let's think to data in sport again, the performance of a team when playing home or away, for example, can be quite different and this factor must be considered in order to predict the outcomes of the matches.
Moreover, it can be worth to explore some of the new frontiers of variational inference, different from the standard mean-field approximation. As mentioned above, the choice of the variational family is one of the key aspect in this field and methods taken from statistical physics are actually being investigated. Recent works show also the benefits of combining `MCMC` and variational inference, exploiting the asymptotically exactness of the former and the rapidity of the latter [40].

6. Bibliography

[1] C. De Bacco, D. B. Larremore, and C. Moore, "A physical model for efficient ranking in networks," *Science advances*, vol. 4, no. 7, p. eaar8260, 2018.

[2] P. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[3] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[4] S. Negahban, S. Oh, and D. Shah, "Rank centrality: Ranking from pairwise comparisons," *Operations Research*, vol. 65, no. 1, pp. 266–287, 2016.

[5] I. Ali, W. D. Cook, and M. Kress, "On the minimum violations ranking of a tournament," *Management Science*, vol. 32, no. 6, pp. 660–672, 1986.

[6] P. Slater, "Inconsistencies in a schedule of paired comparisons," *Biometrika*, vol. 48, no. 3/4, pp. 303–312, 1961.

[7] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Iftode, "Finding hierarchy in directed online social networks," in *Proc. 20th Intl. Conf. on the World Wide Web*, pp. 557–566, ACM, 2011.

[8] F. Fogel, A. d'Aspremont, and M. Vojnovic, "Serialrank: Spectral ranking using seriation," in *Advances in Neural Information Processing Systems*, pp. 900–908, 2014.

[9] M. Cucuringu, "Sync-rank: Robust ranking, constrained ranking and rank aggregation via eigenvector and sdp synchronization," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 1, pp. 58–79, 2016.

[10] K. E. Train, *Discrete Choice Methods with Simulation*. Cambridge University Press, 2009.

[11] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[12] R. D. Luce, "On the possible psychophysical laws.," *Psychological Review*, vol. 66, no. 2, p. 81, 1959.

[13] R. J. Williams, A. Anandanadesan, and D. Purves, "The probabilistic niche model reveals the niche structure and role of body size in a complex food web," *PLoS ONE*, vol. 5, no. 8, p. e12092, 2010.

[14] R. J. Williams and D. W. Purves, "The probabilistic niche model reveals substantial variation in the niche structure of empirical food webs," *Ecology*, vol. 92, no. 9, pp. 1849–1857, 2011.

[15] A. Z. Jacobs, J. A. Dunne, C. Moore, and A. Clauset, "Untangling the roles of parasites in food webs with generative network models," *arXiv preprint arXiv:1505.04741*, 2015.

[16] B. Ball and M. E. Newman, "Friendship networks and social status," *Network Science*, vol. 1, no. 01, pp. 16–30, 2013.

[17] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *Journal of the American Statistical Association*, vol. 97, pp. 1090–1098, 2001.

[18] A. E. Elo, *The Rating of Chessplayers, Past and Present.* Arco Pub., 1978.

[19] M. E. Glickman, "The glicko system," *Boston University*, vol. 16, 1995.

[20] J. Park and M. E. Newman, "A network-based ranking system for us college football," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 10, p. P10014, 2005.

[21] S. Motegi and N. Masuda, "A network-based dynamical ranking system for competitive sports," *Scientific reports*, vol. 2, p. 904, 2012.

[22] R. Herbrich, T. Minka, and T. Graepel, "Trueskill: a Bayesian skill rating system," in *Advances in Neural Information Processing Systems*, pp. 569–576, 2007.

[23] P. Dangauthier, R. Herbrich, T. Minka, and T. Graepel, "Trueskill through time: Revisiting the history of chess," pp. 337–344, 2008.

[24] R. Coulom, "Whole-history rating: A Bayesian rating system for players of time-varying strength," in *International Conference on Computers and Games*, pp. 113–124, Springer, 2008.

[25] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," *def*, vol. 1, no. $2\sigma 2$, p. 16, 2007.

[26] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[27] A. E. Gelfand and A. F. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American statistical association*, vol. 85, no. 410, pp. 398–409, 1990.

[28] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[29] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[30] R. Weinstock, *Calculus of variations: with applications to physics and engineering.* Courier Corporation, 1974.

[31] G. Parisi, *Statistical field theory.* Addison-Wesley, 1988.

[32] C. Robert and G. Casella, *Monte Carlo statistical methods.* Springer Science & Business Media, 2013.

[33] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[34] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd international conference on Machine learning*, pp. 113–120, ACM, 2006.

[35] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[36] R. Ranganath, S. Gerrish, and D. M. Blei, "Black box variational inference," *arXiv preprint arXiv:1401.0118*, 2013.

[37] G. Casella and C. P. Robert, "Rao-blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, 1996.

[38] S. M. Ross, *Simulation.* Elsevier, 2002.

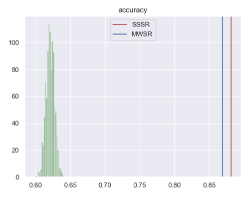
[39] J. Paisley, D. Blei, and M. Jordan, "Variational bayesian inference with stochastic search," *arXiv preprint arXiv:1206.6430*, 2012.

[40] F. J. Ruiz and M. K. Titsias, "A contrastive divergence for combining variational inference and mcmc," *arXiv preprint arXiv:1905.04062*, 2019.

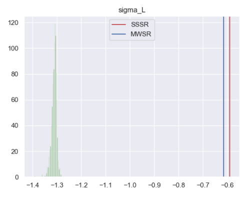
Appendices

A. Synthetic datasets analysis

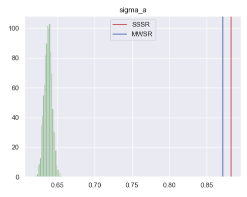
$\beta = 2.0$



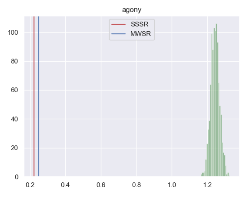
(a)



(b)

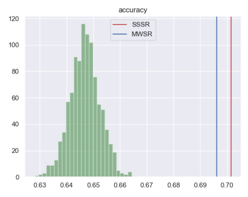


(c)

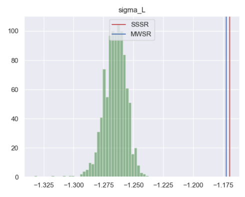


(d)

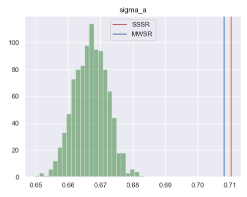
$\beta = 0.5$



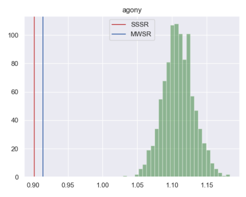
(e)



(f)

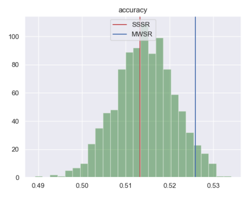


(g)

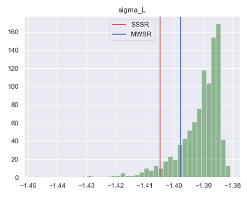


(h)

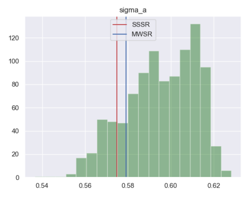
$\beta = 0.1$



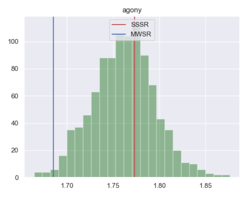
(i)



(j)



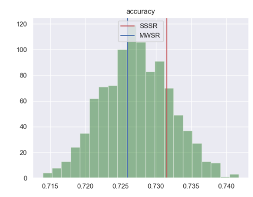
(k)



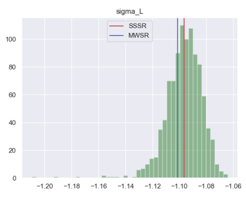
(l)

Figure 1: Results obtained for synthetic data considering all the presented metrics: the red and blue lines are respectively the value obtained with Self Spring SpringRank (SSSR) algorithm and with moving window version of SpringRank (MWSR) considering the time stamps (the chronological order) of the matches, each entry of the histogram is instead a different realization of SSSR for a random permutation in the order of the matches (no true time stamps). Each row correspond to a different level of hierarchy, controlled by the parameter β, used to create the synthetic dataset.

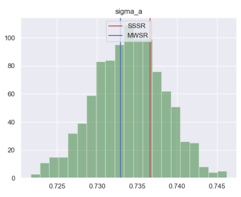
Synthetic static dataset, $\beta = 2.0$



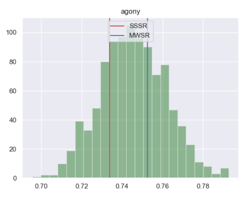
(a)



(b)



(c)



(d)

Figure 2: Results obtained for synthetic data in static framework: the red and blue lines are respectively the value obtained with Self Spring SpringRank (SSSR) algorithm and with moving window version of SpringRank (MWSR) considering the time stamps (the chronological order) of the matches, each entry of the histogram is instead a different realization of SSSR for a random permutation in the order of the matches (random time-stamps assigned to the matches).

B. ELBO: hints for calculations

We report also an useful observation that will be needed in the following calculations: consider the random variable $s_{tij} := s_{ti} - s_{tj} - 1$. Since for the variational distribution q the variables s_{ti} are uncorrelated normals, the variable s_{tij} is so with

$$s_{tij} \sim N(\theta_{ti} - \theta_{tj} - 1, \eta_{ti}^2 + \eta_{tj}^2) =: N(\theta_{tij}, \eta_{tij}^2)$$

Moreover, q is factorized over the s_{ti} so that

$$\begin{aligned}\int (s_{ti} - s_{tj} - 1)^2 q(\boldsymbol{s}) d\boldsymbol{s} &= \int (s_{ti} - s_{tj} - 1)^2 q(s_{ti}, s_{tj}) ds_{ti} ds_{tj} \\ &= \int s_{tij}^2 q(s_{tij}) ds_{tij} \\ &= Var_q(s_{tij}) + \mathbb{E}_q[s_{tij}]^2 \\ &= \eta_{ti}^2 + \eta_{tj}^2 + (\theta_{ti} - \theta_{tj} - 1)^2. \end{aligned} \tag{1}$$

Similar results hold for the random variables $s_{ti} - \mu_{ti}$ and $\mu_{ti} - \mu_{t-1,i}$, since all variables are independent for the variational distribution q.
We finally recall the expression of the moment generating function of a squared Gaussian $X \sim N(\mu, \sigma^2)$, that is

$$\mathbb{E}[e^{tX^2}] = \frac{1}{\sqrt{1 - 2t\sigma^2}} \exp\left(\frac{\mu^2 t}{1 - 2t\sigma^2}\right)$$

and in our specific case we obtain

$$\mathbb{E}_q[e^{-\frac{\beta}{2}s_{tij}^2}] = \frac{1}{\sqrt{1 + \beta\eta_{tij}^2}} \exp\left(-\frac{\beta}{2} \cdot \frac{\theta_{tij}^2}{1 + \beta\eta_{tij}^2}\right). \tag{2}$$

We use this results to compute the analytical expression of (4.34).