# Features in TrueSkill™

**Nathaniel J. McAleese**
Department of Computer Science
Cambridge University
Cambridge, CB2 1TA
nm583@cam.ac.uk

## Abstract

TrueSkill presented a Bayesian ranking framework that generalised the Elo score. This work extends that framework to allow for the incorporation of match features that impact the outcome of the game. In order to do so, TrueSkill inference is implemented in a probabilistic programming language. This implementation is shown empirically to closely replicate the parameters inferred by expectation propagation, whilst being much easier to extend. The inclusion of match features then provides improvements in predictive accuracy on synthetic and real-world datasets.

## 1 Introduction

We often wish to predict the outcome of a competitive and random game based on the past performance of the players. Whilst several approaches to this problem have been devised, TrueSkill [15] is arguably the one deployed at the largest scale. However TrueSkill is not easily adapted to include additional information. Consider chess, in which there is an established first-mover advantage. In an ideal system, we should be able infer that a player that wins from the weaker position is stronger than one that succeeds with this advantage. Equally, many games have subtle variations that can play to particular competitor's advantage; yet in naive TrueSkill these are unaccounted for. This paper lays out how a probabilistic programming language can be used to implement a direct analogue of TrueSkill that is easily amenable to extension with features that account for variation between games, such as the first-mover advantage in Chess or the court surface in tennis. In principle this should allow for more accurate prediction of future outcomes, and this is verified on synthetic data in which the features contribute a great deal to the outcome of the game. However whilst the results on real world data consistently show small improvements, the effects are not large enough to be statistically significant.

## 2 Related Work

There are broadly two categories of rating system - incremental, and offline [9]. In an incremental rating system, each match outcome changes only the skills of the players that participate. This is appealing because it usually reduces the computational load of running the ranking algorithm, and simplifies understanding for the players. However it has shortcomings. Suppose a small number of extremely strong new player appears amongst a cohort of other newcomers. In a model that optimises for predictive accuracy, we would ideally like to update the ratings of all players who defeated a newcomer later, after we have determined that new player's strength.

Most naive rating schemes, and also the famous Elo score, are incremental. So too is TrueSkill, as described in the original work and widely implemented [18]. Other rating systems, however, run in an offline fashion and seek to optimise the accuracy of the model's predicted outcomes for future matches. These include directly seeking to do inference on the Bradley-Terry model [20], TrueSkill Through Time [10], and further optimisations such as Edo and Whole History Rating [9, 13].

The work presented here is an offline rating system. After a set of matches results are collected, skills are inferred for all players simultaneously, and the addition of new data may alter the scores of any of the existing players.

# 3 Method

## 3.1 Formulation of TrueSkill

TrueSkill presents a graphical model that simulates matches in a series of competitive games. Each player $i$ is presumed to have an unknown skill $s_i \in \mathbb{R}$. A priori, the skills are assumed to be distributed according to a factorising Gaussian $p(s_i) = \mathcal{N}(s_i; \mu_0, \sigma_0^2)$. The model then supposes that outcome of game $k$ between players $i$ and $j$, denoted $y_{ij}(k)$ is determined as:

$$p_i(k) \sim \text{Normal}(s_i, \beta^2) \tag{1}$$

$$p_j(k) \sim \text{Normal}(s_j, \beta^2) \tag{2}$$

$$\Delta p_{ij}(k) = p_i(k) - p_j(k) \tag{3}$$

$$\tag{4}$$

$$y_{ij}(k) = \begin{cases} -1 & \Delta p_{ij}(k) \leq -\epsilon \quad \textit{(player j wins)} \\ 0 & -\epsilon < \Delta p_{ij}(k) < \epsilon \quad \textit{(draw)} \\ +1 & \Delta p_{ij}(k) \geq \epsilon \quad \textit{(player i wins)} \end{cases} \tag{5}$$

Where $\epsilon$ is a hyper-parameter reflecting a draw margin. The marginal distribution of the skills is then approximated by Expectation Propagation [19] in a factor graph. From here on, this solution is referred to as "EP-TrueSkill".

To implement the TrueSkill in a probabilistic programming language, the problem is instead treated as a hierarchical Bayesian model in which the difference in skills determines the value of a categorical distribution over three values (loss, win and draw). The probability of each may be determined as:

$$\Delta s_{ij}(k) = s_i - s_j \tag{6}$$

$$l_{ij} = \hat{\Phi}\left(\frac{-\epsilon - \Delta s_{ij}(k)}{2\beta}\right) \qquad r_{ij} = \hat{\Phi}\left(\frac{+\epsilon - \Delta s_{ij}(k)}{2\beta}\right) \tag{7}$$

$$P(Y_{ij} = -1) = l_{ij} \qquad P(Y_{ij} = 0) = r_{ij} - l_{ij} \qquad P(Y_{ij} = +1) = 1 - r_{ij} \tag{8}$$

Where $\hat{\Phi}$ is a closed form approximation of the CDF of the standard normal distribution [5]. In theory this approximation is unnecessary, because the derivative of $\Phi$ is well known, but in practice it simplifies the specification of the model by providing an implementation of $\Phi$ that is amenable to automatic differentiation without manually specifying a new low-level kernel for all hardware.

## 3.2 Inference, Hamiltonian Monte Carlo & Probabilistic Programming

This model can then be directly implemented using the PYMC3 [21] probabilistic programming library in Python. This uses Theano [3] to provide tensor manipulation and automatic differentiation primitives that may be combined with PYMC3 representations of probability distributions. The framework allows the user to construct hierarchical Bayesian models by chaining together operations on primitive `Distribution` types which determines the structure of the model. This construction is then used to build a computational graph of the log-likelihood of the desired distribution. Automatic differentiation can then be used to find maximum likelihood (MLE) and maximum a posteriori (MAP) estimates of the latent parameters of the model conditioned on the observed data from some node or nodes in the graph.

In our case, the observed data are the match outcomes, and the latent variables of interest are the player skills. Numerous inference procedures are available within this general framework, and so to determine what is appropriate it is useful to distinguish between the two types of uncertainty that are

implicitly present in both formulations of TrueSkill. "Aleotoric" uncertainty represents the intrinsic randomness present in the process under consideration, whereas "epistemic" uncertainty captures the uncertainty in a prediction due to the nature of the model [17]. In TrueSkill, the $\beta$ parameter reflects aleotoric uncertainty by modelling the variation in a players performance in a particular game. The variance associated with each player's skill, by contrast, is a measure of epistemic uncertainty that reflects how much information the model has about the player.

Our PPL framework offers a way to compute maximum a posteriori (MAP) estimates by exploiting the gradient information available in our model specification to use convex optimisation techniques such as BFGS [6], an approach that will henceforth be described as MAP-TrueSkill. Note that this approach requires some careful consideration. Firstly, the log density of our posterior distribution may well be multimodal (non-convex), and thus BFGS is not guaranteed to find the true MAP parameters, as it may get stuck in a local minimum. Section 4 shows empirically that this is not an issue for any of the six datasets, but it is a cause for concern.

Secondly, MAP inference gives a point estimate, namely a mode[1] of the posterior density. Unlike the results of classical TrueSkill, it does not include an estimate of epistemic uncertainty (although it does explicitly model the aleotoric uncertainty of the performance variation $\beta$ term). This can potentially be limiting, but it very much depends on the intended use case for the skill estimates produced. Note for example, that because EP-TrueSkill models the marginal skill of each player as a Gaussian with symmetric variance representing uncertainty, the variances associated with each player do not influence the prediction of match outcomes (eg, given $\mu$ and $\sigma$ for two players, only the difference in $\mu$ determines what discrete outcome we predict for the match).

It is also possible to obtain uncertainty estimates from the proposed model by Markov chain Monte Carlo (MCMC) sampling, and thus fully reproduce the information captured by EP-TrueSkill.

We can motivate the need for MCMC very generally. Bayesian methods often require the estimation of analytically intractable integrals. In particular, given a probability density function $\pi(q)$ and a parameter space $Q$, we aim to estimate the expectation of a function $f$ with respect to $Q$,

$$\mathbb{E}_\pi[f] = \int_Q \pi(q) f(q) \, \mathrm{d}q$$

Due to the "curse of dimensionality", naive quadrature scales extremely poorly with the dimensionality of Q. In particular, this is because not all points in the space contribute equally to the expectation. Contributions to the expectation scale with the product of density and volume which is only large in a small region of the parameter space known as the "typical set" [4].

The most popular scheme for surmounting this issue for high dimensional spaces is MCMC. In this paradigm we seek transitions that depend solely on one point in the parameter space (to satisfy the Markov property) that also "preserve" [12] the distribution $\pi$. We say a conditional transition distribution $\mathbb{T}$ preserves a distribution $\pi$ when

$$\pi(q) = \int_Q \mathbb{T}(q \mid q') \pi(q') \, \mathrm{d}q'$$

Given such a transition, we may then in theory randomly sample points from the parameter space along a Markov chain with transitions given by $\mathbb{T}$ to characterise the typical set and thus accurately approximate the desired expectation.

Metropolis-Hastings, Gibbs sampling and Hamiltonian Monte Carlo are means by which to determine such a transition and thus implement MCMC. In Metropolis-Hastings, we use a proposal mechanism (almost always a Gaussian perturbation) $\mathbb{Q}$ to generate random transitions from $q$ to $q'$ in the state space, and accept that transition with probability:

$$a(q \mid q') = \min\left(1, \frac{\pi(q')\mathbb{Q}(q' \mid q)}{\pi(q)\mathbb{Q}(q \mid q')}\right)$$

[1] Ideally *the* mode, but this is not guaranteed by BFGS on a non-convex function.

Intuitively, this proceeds by randomly generating random perturbations of q and preferentially jumping to more likely points in the space. However, as the dimensionality of the state space scales, the acceptance probability diminishes and the rate of exploration slows to a crawl.

Hamiltonian (or "Hybrid") Monte Carlo is a solution to this problem for certain classes of distribution. It uses first order gradient information and standard results from differential geometry to determine transitions that produce a high acceptance probability under the Metropolis-Hastings ratio [12]. The gradient of $\pi$ points towards a mode from any point in the parameter space, and intuitively the typical set can be characterised by an orbit in the parameter space that is exactly analogous to (and governed by the same Hamiltonian mechanics as) a physical orbit in a vector field.

This general mechanism can be applied to sample from our TrueSkill model's latent parameter space, and thus estimate the epistemic uncertainty present in the model. This model, providing an estimate of player skills and an associated uncertainty, is referred to as MCMC-TrueSkill.

### 3.3 Models of Match and Player Features

We might consider many ways in which games might vary. This work considers two particularlly common cases - a first mover, or "P0 Advantage", and the variation in player talents over a set of variations of the game - "Match Type Affinity". These were motivated by the white advantage in chess and the various available surfaces in tennis respectively.

Both schemes introduce the match features by modifying $\Delta s_{ij}(k)$. To introduce a first mover advantage, we posit there is some latent variable $a$ with prior distribution Normal$(0, a_0^2)$ that contributes to all matches. We then define $\Delta s_{ij}(k)_{\text{Advantage}} = s_i - s_j + a$ and may do inference exactly as before.

To allow for variation due to the match type (such as the surface in tennis), we introduce an affinity for each player for each match type. These act as skills that only contribute to the outcome for those particular match types. A feature weight, $\alpha$ may be used to determine how much these features contribute to skill difference and thus the outcome; setting $\alpha$ to one corresponds to asserting that a player's skill in each match type is totally independent, whereas setting it to zero ignores match type variation entirely. Where $\boldsymbol{a}_i$ is a vector with prior distribution Normal$(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{t}_k$ is a one-hot encoding of the match type.

$$\Delta s_{ij}(k)_{\text{Affinity}} = (1-\alpha)(s_i - s_j) + \alpha(\boldsymbol{a}_i - \boldsymbol{a}_j) \cdot \boldsymbol{t}_k \tag{9}$$

## 4 Results

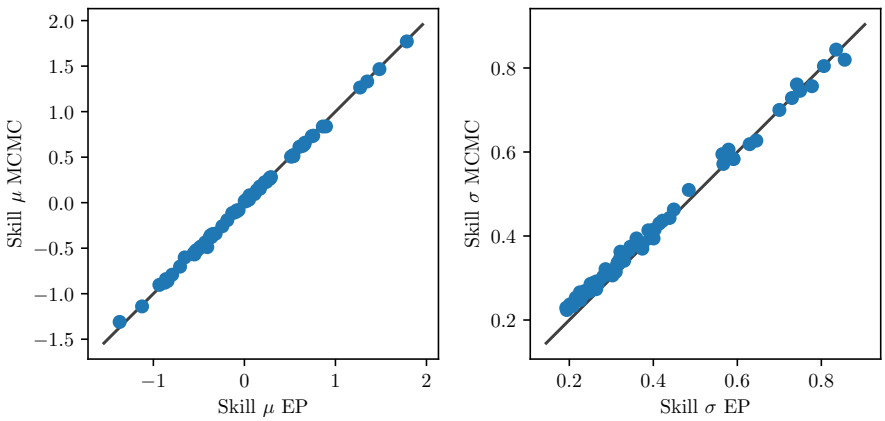


**Figure 1:** These plots show the similarity between the results given by the reference TrueSkill EP implementation [25], on the $x$ axis, and by the proposed MCMC procedure. These skills were computed from tennis matches between the top 60 players (according to the ATP) in 2014. The grey line shows $y = x$.

Synthetic datasets were generated by selecting ground truth skills from the prior distribution and selecting pairs of participants for matches uniformly at random. Exact values of the prior distributions

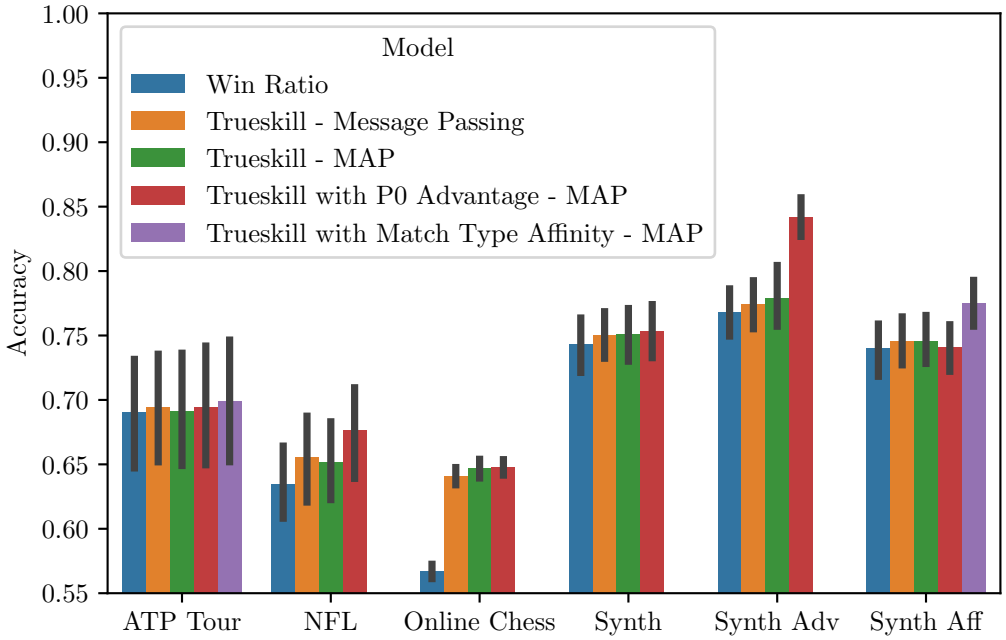


**Figure 2:** Synthetic data demonstrates that for large effect sizes, feature inference in the model can dramatically improve predictive accuracy. However whilst all of the real world datasets show small improvements when features are included, these results are not statistically significant. Error bars show 95% confidence interval from 10-fold cross validation for the real data, and from 10-fold cross validation on 5 simulated datasets for each synthetic run.

used to generate the test datasets are included in the appendix. In the synthetic "Adv" and "Aff" models, the random proposed P0 advantage and match type affinities factors are included in the ground truth generation of the dataset.

Real world data was collected from several sources. The ATP dataset consists of 2,783 professional tennis matches from the 2014 ATP series [2], between 396 players. The NFL dataset is the 2013 and 2014 season results, 267 games between 32 teams [11], and the online chess dataset 20,000 games with 15,000 distinct players [7]. In the NFL case entire teams were modeled as a single player, as a simplification and because of the low rate of change of team membership between games.

Figure 1 shows that the reference implementation of EP-TrueSkill and MCMC-TrueSkill give very similar estimates of player skill on small datasets.

The models were also used to predict the outcomes of held-out matches in a cross-validation procedure. Cross validation was on randomly ordered data, and no effort was made to ensure that players occurred in both the training and test set for each fold; thus some percentage of the predictions for each model in each were for matches that include one or more previously unseen players. It is a reasonable expectation that a good ranking system should present sensible predictions in these edge cases; and indeed it may be very important in cases with high player turnover, so it is worth explicitly noting their inclusion.

MAP-TrueSkill was used for the prediction of match outcomes because small-scale tests showed that the predictive performance of MAP-TrueSkill and MCMC-TrueSkill were similar, whilst the computational overhead of MCMC-TrueSkill is larger. The results of these experiments are shown in Figure 2. As a baseline, a "win ratio" model was also included in which the winner was predicted to be the participant that had won a larger proportion of their games, falling back to random selection on ties. These results show that MAP-TrueSkill is similar or better than EP-TrueSkill in all cases; in the synthetic tasks where the feature effects are large the models that explicitly account for them perform significantly better. In the real-world data, the models that incorporate features show small improvements on every task, none of which are statistically significant. One potentially surprising result was the performance of win ration on the ATP dataset, this is potentially due to the ATP's

seeding protocol that reduces the number of "upsets" in the early tournament rounds that make up most of the games.

The latent variables inferred by these models can also be used to answer other questions about the players and games involved, but empirical evaluation of the results is more difficult than in the case of predictive accuracy. Nonetheless, it is satisfying to note some examples where the predictions of the model correspond well with reality, suggesting that Roger Federer is worst on clay [24], that the Jacksonville Jaguars were the worst team in the NFL in 2013/14 [16], and that the probability of a home team victory between teams of equal skill in the NFL is 60% [1]. These examples serve to inspire more confidence in the model's predictions where popular opinion is less forthcoming. A more formal approach might use domain knowledge about each game to develop application specific discrepancy functions in order to produce posterior predictive checks that reflect sensible beliefs about the latent variables [14]; unfortunately the author lacks appropriate sporting expertise.

## 5 Future Work & Conclusion

Communication with the maintainers of one of the most active PPL packages has been begun in order to merge support for this style of comparative model upstream [22], and allow for automatic transformation of comparative statements into categorical RVs. This would simplify the implementation of models like TrueSkill; offering the opportunity for quick experimentation to future users.

TrueSkill inference within such a general framework also offers the opportunity for the exploration of much more complex games. For example, the Edward PPL framework [23] is used extensively by the AI community who often seek to develop models that learn from human preferences though comparative feedback [8]. In these "matches" between agents, the particular human giving feedback and the particular task that an agent is attempting could both be modeled as match features.

This project demonstrates that TrueSkill can be implemented in a probabilistic programming language, and empirically suggests that the results given by MCMC inference in TrueSkill are very similar to those given by expectation propagation on real data. It shows that this implementation can then be extended to model and infer match features that can greatly improve the predictive accuracy of the model when the feature effects are large, and that may offer smaller improvements in predictive accuracy for real-life games of skill.

## References


[1] *A Home Playoff Game Is A Big Advantage — Unless You Play Hockey | FiveThirtyEight*. `https://fivethirtyeight.com/features/a-home-playoff-game-is-a-big-advantage-unless-you-play-hockey/`. (Accessed on 02/27/2018).

[2] *Association of Tennis Professionals Matches | Kaggle*. `https://www.kaggle.com/gmadevs/atp-matches-dataset`. (Accessed on 02/21/2018).

[3] James Bergstra et al. "Theano: A CPU and GPU math compiler in Python". In: *Proc. 9th Python in Science Conf*. 2010, pp. 1–7.

[4] Michael Betancourt. "A conceptual introduction to Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1701.02434* (2017).

[5] Shannon R Bowling et al. "A logistic approximation to the cumulative normal distribution". In: *Journal of Industrial Engineering and Management* 2.1 (2009).

[6] Charles George Broyden. "The convergence of a class of double-rank minimization algorithms". In: *IMA Journal of Applied Mathematics* 6.1 (1970), pp. 76–90.

[7] *Chess Game Dataset (Lichess) | Kaggle*. `https://www.kaggle.com/datasnaek/chess/data`. (Accessed on 02/21/2018).

[8] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4302–4310.

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

[10] Pierre Dangauthier et al. "Trueskill through time: Revisiting the history of chess". In: *Advances in Neural Information Processing Systems*. 2008, pp. 337–344.

[11] *devstopfix/nfl_results: Results from NFL games 1978-2014 in CSV format*. https://github.com/devstopfix/nfl_results. (Accessed on 02/21/2018).

[12] Simon Duane et al. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[13] Rod Edwards. *Edo Historical Chess Ratings*. http://www.edochess.ca/. (Accessed on 02/19/2018).

[14] Andrew Gelman, Xiao-Li Meng, and Hal Stern. "Posterior predictive assessment of model fitness via realized discrepancies". In: *Statistica sinica* (1996), pp. 733–760.

[15] Ralf Herbrich, Tom Minka, and Thore Graepel. "TrueSkill™: a Bayesian skill rating system". In: *Advances in neural information processing systems*. 2007, pp. 569–576.

[16] *Just How Bad Are the 2013 Jacksonville Jaguars? | Bleacher Report*. http://bleacherreport.com/articles/1783577-are-the-2013-jacksonville-jaguars-a-historically-bad-football-team. (Accessed on 02/27/2018).

[17] Alex Kendall and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in Neural Information Processing Systems*. 2017, pp. 5580–5590.

[18] Heungsub Lee. *TrueSkill Python Implementation — trueskill 0.4.4 documentation*. http://trueskill.org/. (Accessed on 02/19/2018).

[19] Thomas Peter Minka. "A family of algorithms for approximate Bayesian inference". PhD thesis. Massachusetts Institute of Technology, 2001.

[20] PV Rao and Lawrence L Kupper. "Ties in paired-comparison experiments: A generalization of the Bradley-Terry model". In: *Journal of the American Statistical Association* 62.317 (1967), pp. 194–204.

[21] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3". In: *PeerJ Computer Science* 2 (2016), e55.

[22] Dustin Tran. *Operations on random variables automatically returning random variables · Issue #773 · blei-lab/edward*. https://github.com/blei-lab/edward/issues/773. (Accessed on 02/27/2018).

[23] Dustin Tran et al. "Edward: A library for probabilistic modeling, inference, and criticism". In: *arXiv preprint arXiv:1610.09787* (2016).

[24] *Why is Roger Federer weak on clay? - Quora*. https://www.quora.com/Why-is-Roger-Federer-weak-on-clay. (Accessed on 02/27/2018).

[25] Damon Wischik and Carl Rasmussen. *Probabilistic Machine Learning, Coursework 2*. https://notebooks.azure.com/djw1005/libraries/probml/html/coursework2.ipynb. (Accessed on 02/20/2018). Jan. 2017.