Learning the Temperature of a Game

Cavendish Laboratory
Philipp Hennig
David Stern (MSR) & Thore Graepel (MSR)
UNIVERSITY OF CAMBRIDGE

Abstract: We attempt to combine the pure mathematical theory of combinatorial games with applied machine learning techniques to develop a parallelizable architecture for approximate game tree search.

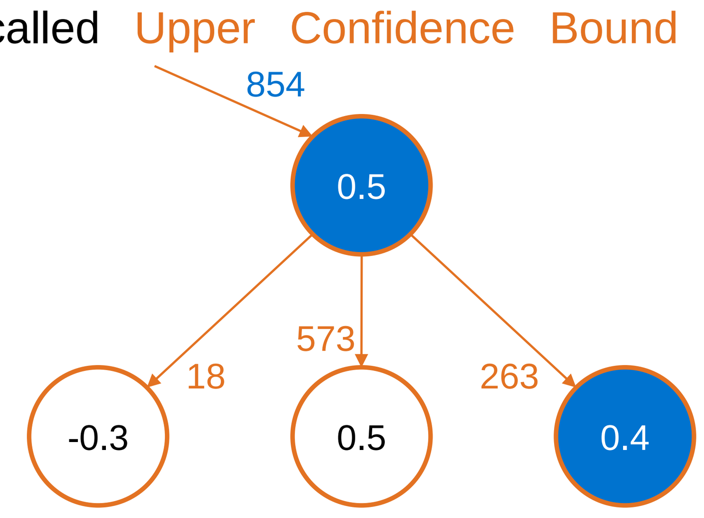
Background

Games

Games are interesting for machine learning research because they pose a hard problem in a very concise package of rules. So far, machines have mostly won against humans by exploiting their superior computing power: Deep Blue beat Kasparov by the brute force of a huge database and very efficient analytical search [1].
A tougher challenge is Go, an ancient Board game with a huge following in Asia. Go's game tree has 10^{400} nodes, making exhaustive search physically impossible. So machines will have to focus on "interesting" parts of the tree. The knowledge needed for this can be provided by abstract reasoning tools from pure mathematics or be "discovered" by the machines themselves, using Machine Learning techniques. This work is an attempt to combine both.

UCT: Greedy Reinforcement Learning on trees

The machine *MoGo* beat a professional human player on a small, 9x9 Go board earlier this year [2]. It uses a method called Upper Confidence Bound for Trees (UCT).



Per board situation, the machine performs a large number (~100k) of "descents" into the game, starting at the root (the current board situation) and choosing moves step by step, all the way to a terminal position. In situations seen previously, it explores the move j that maximises

$$\bar{X}_j + \sqrt{\frac{2\log n}{T_j(n)}}$$

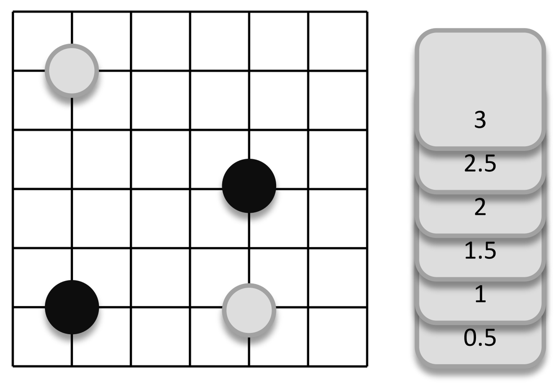
where $\bar{X}_j$ is the average of the results achieved playing this move $T_j(n)$ times in those previous n descents passing this situation. UCT strikes a balance between exploration and exploitation, exploring moves that have given good results so far and have not been played very often. If a new, unknown node is reached, it performs a roll-out, choosing random moves until a terminal node is reached, which provides an update to the X_j of all moves played in this descent. With each descent, UCT selectively expands the search tree.
Professional Go is played on a much larger, 19x19 board. To scale up to this complexity, UCT has to make use of massively parallel computer architectures. Since each machine in the cluster has to have access to a full search tree at any point in time, parallelisation is tricky.

Combinatorial Game Theory

Go games develop along several parallel "battles" – sub-games whose values are approximately indepen-dent of each other. However, the perfect line of play (the "Principal Variation") shifts back and forth between battles in a complex pattern. Combinatorial Game Theory [4] deals with optimal play in such sums of independent games. It uses concepts called the Temperature *T(G)* and Mean $\mu(G)$ of a game *G*. *T(G)* represents the price (in game result points) an optimal player would be willing to pay to be allowed to make the first move in *G*. It quantifies the urgency of making a move in a particular game. The Mean represents the average outcome of many copies of *G* if both players get to start half the time. It measures the value a game. Using these concepts, Combinatorial Game Theory has developed several strategies for nearly optimal play.

Enriched Environments

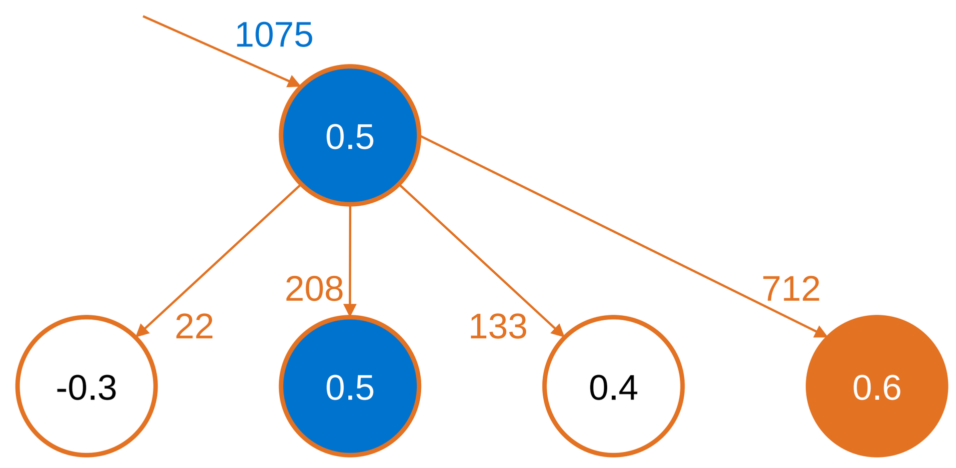
The Temperature of a game *G* can be measured by adding a coupon stack *C* of coupons of value V_m, V_m-δ,V_m-2δ,... giving players the chance to score points by taking a coupon and passing in *G*.



The coupon value *V* at which an optimal player stops taking coupons for the first time and starts to play in *G* is an upper bound on the temperature of *G*. Searching for this value analytically [5] is even more expensive than searching for the optimal solution of *G* alone. An approximate method is needed.

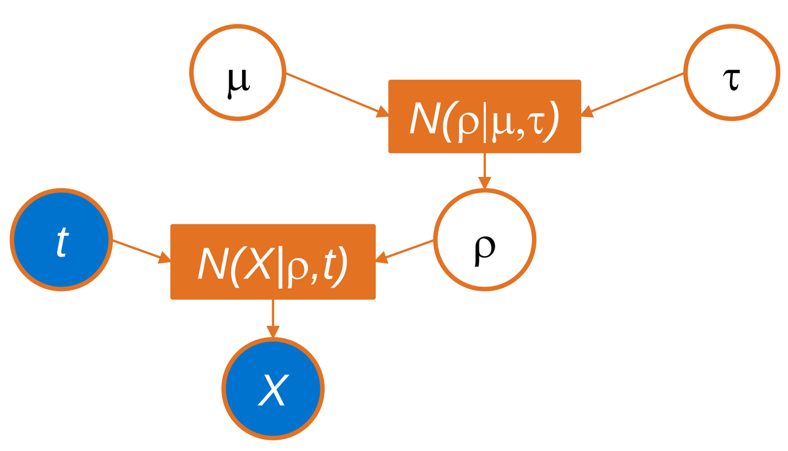
Our Work

We are trying to use a greedy reinforcement learning method to measure the temperatures of small games, using an enriched environment. The search tree is the Game tree of *G* with "taking the next coupon off *C*" added as a legal move at every node.



The Thomson Heuristic – a Bayesian alternative to UCT

UCT relies on point estimates to make its decisions: At any given point in time, UCT believes in just one approximation to the true solution. But Enriched Env-ironments often allow for multiple, equally optimal lines of play, with only one (with the lowest *V*) conveying the true temperature. We thus need a Bayesian search algorithm, which keeps track of a whole distribution of possible paths through the search tree.



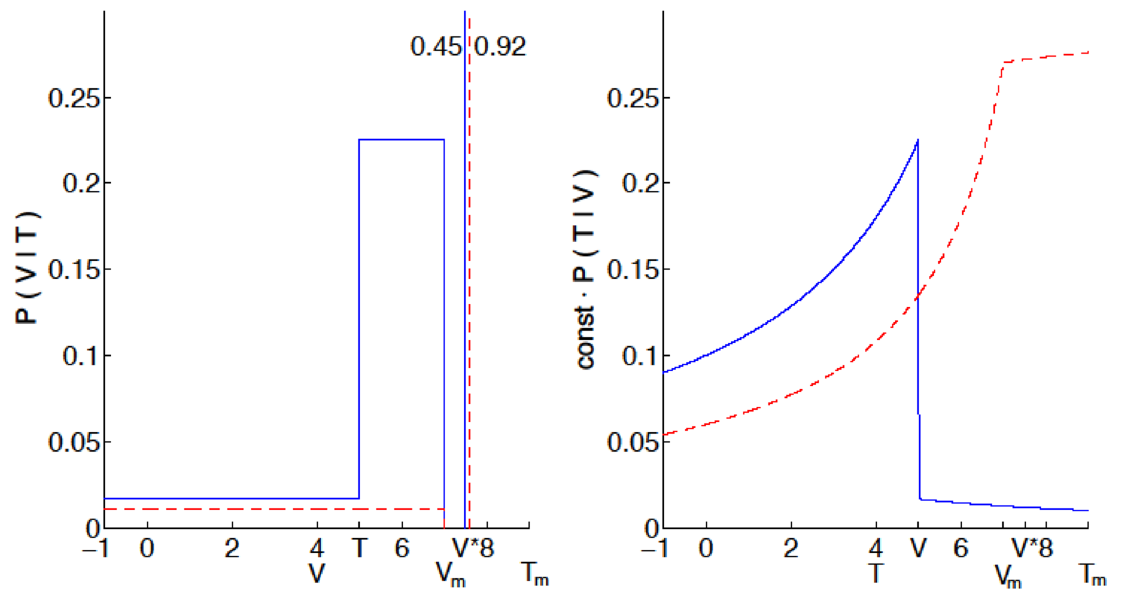
Our model assumes playing a particular move will lead to a final reward ρ, which is normally distributed with mean μ and precision τ, which we would like to know. The actual reward *X* seen after a roll-out is influenced by the dynamics of the underlying children (which change their behaviour during learning), which we model very simplistically with Gaussian noise of mean ρ and precision t. After each roll-out, the estimates for μ and τ are updated (using Bayes' rule), and the machine slowly becomes more confident about what the distribution over lines of play should be.

Iterative Parameter-Updates

Both UCT and our model are greedy methods: The decision taken at a given board position uses only locally available information. Performance drops if optimal play necessitates a long sequence of co-ordinated moves. Unfortunately, an Enriched Environ-ment is exactly such a situation, so it is important to keep the sequence leading to *V* as short as possible. We thus update V_m and δ iteratively during the search.

A Likelihood-Model

As noted above, the values of *V* produced by an optimal player are only upper bounds on the true temperature *T*. We can, however, use them as data that gives us a clue about *T*.

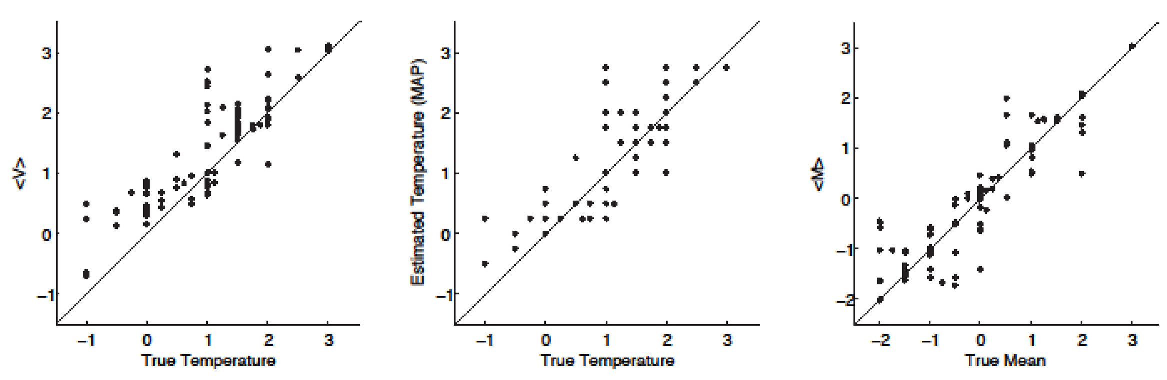


The left plot shows *P(V|T)* as a function of V: For a given T, all values *V>T* are equally likely. Values *V<T* are less likely, but possible, and also uniformly distributed over this range. The plot on the right shows *P(V|T)* as a function of *T*: Observing a value for *V* makes all *T>V* much less likely. The dashed lines represent the special degenerate case of the player starting play directly in *G*, which conveys less information than other values of *V*.

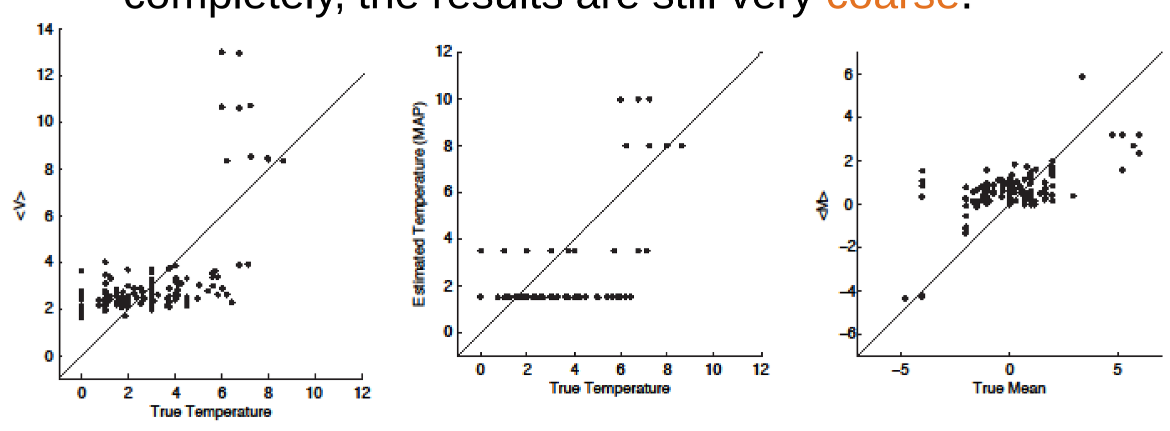
Preliminary Results

We are using the game Amazons as a test case, because it has been studied extensively in the CGT community, and analytical solutions are available, at least for small games.

1. The Bayesian searcher performs well in games against UCT, winning 73% of the time when given the same computing resources
2. On small boards the algorithm produces encouraging results



3. On bigger boards, where exhaustive methods fail completely, the results are still very coarse.



Conclusions

What we have achieved:
1. Coarse estimates of the temperatures of games that are too complex to be searched analytically
2. A Bayesian search algorithm that beats UCT on our test set.

What remains to be done:
1. Better estimates of Temperatures for large games
2. A sub-game identifier for Go, either based on roll-out statistics or a stand-alone, unsupervised method

References

[1] Campbell, Hoane and Hsu; "Deep Blue"; *Artificial Intelligence* **134**, pp. 57-83, 2002
[2] Gelly and Silver; "Achieving master level play in 9x9 computer Go"; *Adv. in Artificial Intelligence*, 2008
[3] Auer, Cesa-Bianchi and Fischer; "Finite-time analysis of the multi-armed bandit problem"; *Machine Learning* **47**, pp. 235-256, 2002
[4] Berlekamp, Conway and Guy; "Winning ways for your mathematical plays"; Peters 2004
[5] Muller, Enzensberger, Schaeffer. "Temperature Discovery Search"; *Adv. in Artificial Intelligence*, 2004