NICTA

Modelling Go Positions with Planar CRFs

Authors

Dmitry Kamenetsky | Nic Schraudolph | Simon Günter | S.V.N. Vishwanathan

Statistical Machine Learning Program

Abstract

We apply Conditional Random Fields (CRFs) to territory prediction in the game of Go. We propose a two-stage graph reduction of the Go position, with the first stage used for CRF parameter estimation and the second for inference. The interaction potentials in our model are calculated from generic shape features; the associated parameters are shared between semantically equivalent feature types. Our experiments indicate that this architecture is very efficient at propagating relevant information across the graph.

Go

- Two players (black and white) alternate in placing stones on the intersections of a 9×9 (or 19×19) grid.
- Neighbouring stones of the same colour form a contiguous *block*.
- A block can be *captured* if all its empty neighbours (*liberties*) are occupied by opponent stones.
- Players aim to maximize the area (*territory*) of blocks that cannot be captured.
- *Territory prediction*: given a board position, determine which player controls each intersection (see Figure 1 for an example).

Graph Abstraction

Common fate graph

- *Common fate property*: blocks always live or a die as a unit.
- *Common fate graph* G_f (Graepel et al., 2001) merges all black stones in a block into a single node ● and white stones into ○ (Figure 2).
- Block's size and shape can be represented in the corresponding node's features.

Block graph

- Empty regions are usually divided between players and so cannot be fully collapsed. However if we do not collapse them at all, we end up with large, unwieldy graphs.
- Our *block graph* G_b is a compromise between these two extremes (Figure 3).
- We collapse empty intersections of G_f into *black surround* ■, *white surround* □ and *neutral* ◇ depending on the Manhattan distance to the nearest black and white stones.
- G_b is more concise than G_f, but preserves all the information required for predicting territory.
- Since G_b is planar, we employ exact polynomial-time parameter estimation based on the work of Globerson and Jaakkola (2007).

Group graph

- *Group*: set of blocks of the same colour that share at least one surround.
- We construct the *group graph* G_g by collapsing groups of G_b (Figure 3).
- Blocks belonging to the same group are likely to have the same fate, therefore we use G_g for inference.

Example

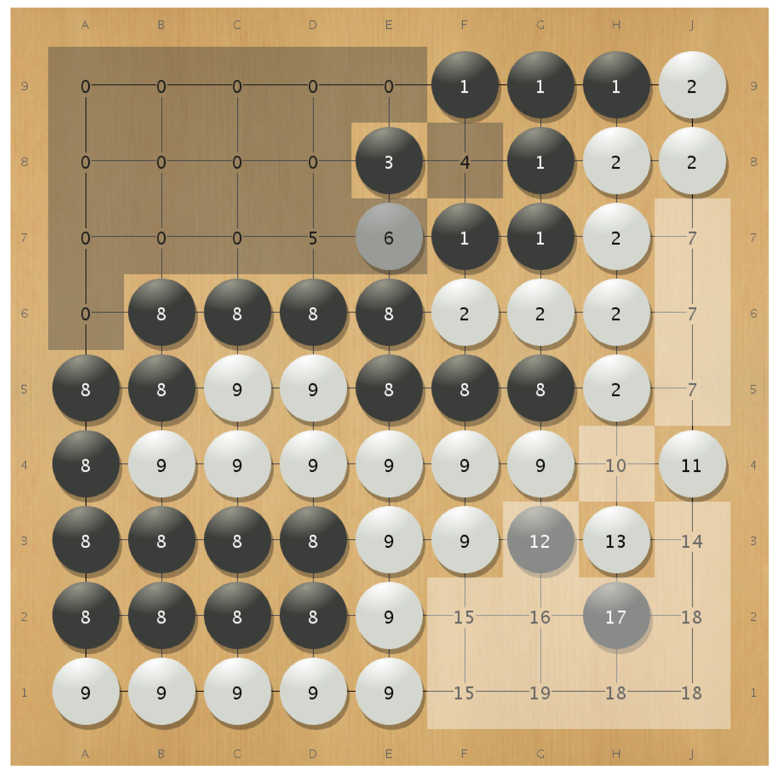


Figure 1. A typical 9×9 board position represented as a grid graph G. Black's territory is shaded in dark-gray, while White's territory is shaded in light-gray. The stones inside the opponent's territory are dead; the rest are alive.

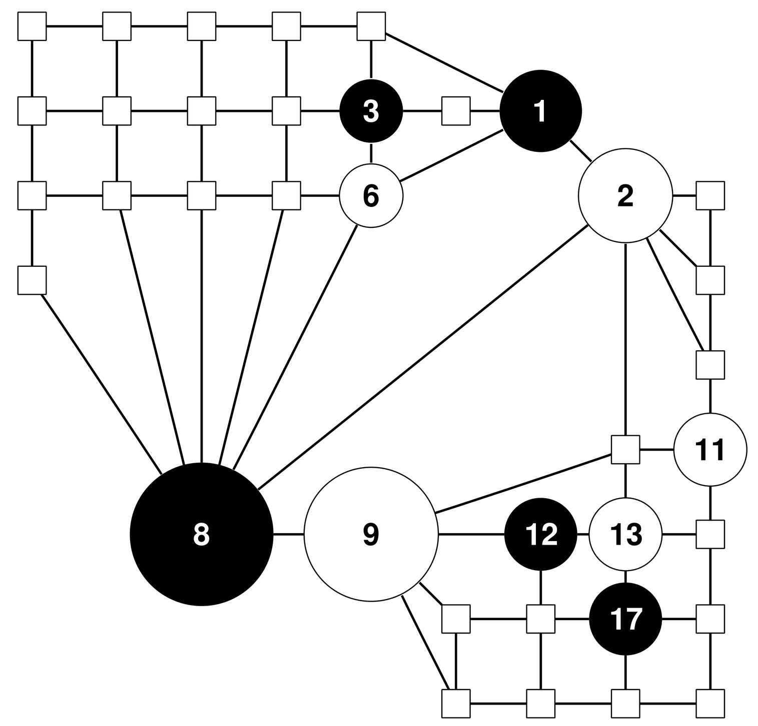


Figure 2. Corresponding common fate graph G_f.

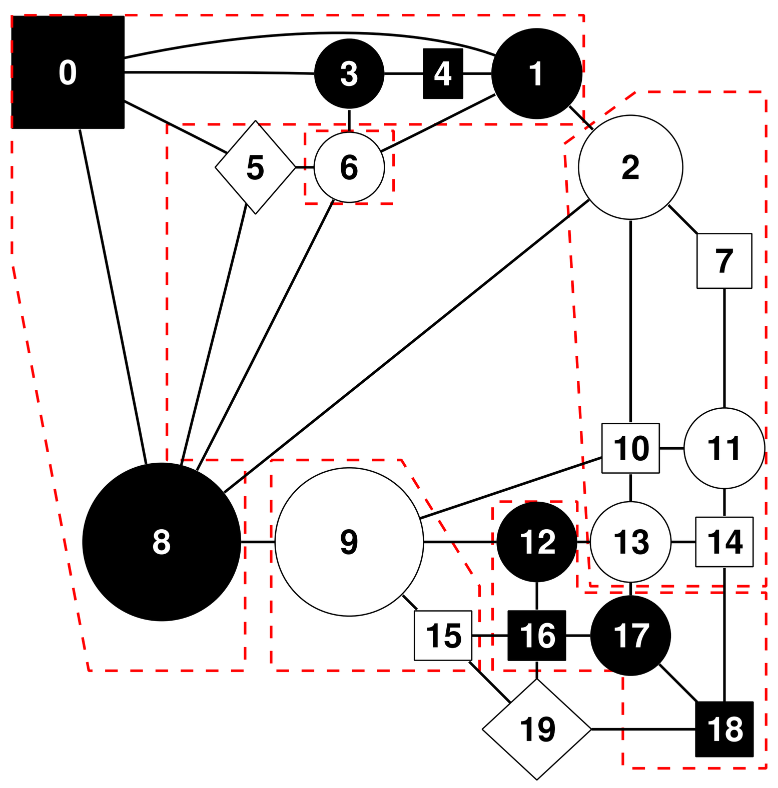


Figure 3. Corresponding block graph G_b. ○ represent *stones*, □ represent *surrounds*, and ◇ are *neutral*. Dashed lines indicate nodes of the group graph G_g.

Features and Parameters

Nodes

- Given a node $v \in G_b$, for each point $i \in v$ compute the number of adjacent points A_i that are also in v (see Figure 4 top).
- Node's feature is a vector F, where $F_k = |\{i : A_i = k\}|$.
- F provides a powerful summary of the region's shape. This is important for classifying the region's eye-forming potential and thus its ability to live/die.

Edges

- Given two adjacent nodes $v_1, v_2 \in G_b$, for each point $i \in v_1$ compute the number of adjacent points A_i^1 that are in v_2 and *vice versa* for A_i^2.
- Edge's features are two vectors F^1 and F^2 that use A^1 and A^2 respectively.
- Our edge features provide additional information that is not conveyed by the features of the nodes they connect (see Figure 4 bottom).

Parameter sharing

- We encode colour symmetry by appropriate parameter sharing.
- 3 node parameters: *stone* ($\theta_{\bigcirc}$), *surround* ($\theta_{\square}$), and *neutral* ($\theta_{\diamond}$).
- 8 edge parameters: one for each possible type of node pairing.
- For each edge we also include features of its neighbouring nodes and edges. These features are associated with the parameter vector θ^n.

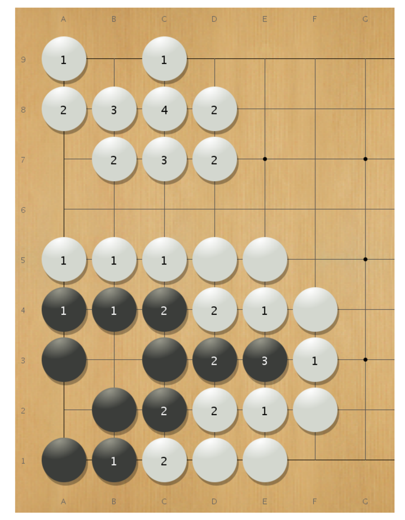
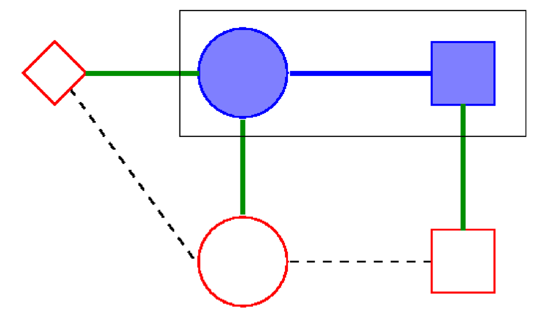


Figure 4. Top: Node features, resulting in feature vector [2, 4, 2, 1]. Bottom: Edge features, resulting in feature vectors [6, 3, 0] and [3, 3, 1] for edges ● → ○ and ○ → ●, respectively.

Figure 5. Parameters and features used to compute the potential of one particular edge (blue) in a small block graph (below).



	Current Edge		Neighbour Edges	
	Param.	Feat.	Param.	Feat.
Nodes	$\theta_{\bigcirc}$	●	$\theta^n_{\diamond}$	◇
	$\theta_{\square}$	■	$\theta^n_{\bigcirc}$	○
			$\theta^n_{\square}$	□
Edges	$\theta_{\bigcirc\square}$	● → ■	$\theta^n_{\diamond\bigcirc}$	◇ → ●
	$\theta_{\square\bigcirc}$	■ → ●	$\theta^n_{\bigcirc\diamond}$	● → ◇
			$\theta^n_{\bigcirc\bigcirc}$	○ → ● ● → ○
			$\theta^n_{\square\square}$	□ → ■ ■ → □

Experiments

- We use 9×9 endgame positions of van der Werf et al. (2005).
- Each intersection is labeled as either BLACK or WHITE.
- 4 measures of error
  - **Vertex**: percentage of misclassified *non-neutral* nodes in G.
  - **Block**: percentage of misclassified *stone* nodes in G_b.
  - **Winner**: percentage of games whose winner is predicted incorrectly.
  - **Game**: percentage of games that have at least one vertex error.
- Naive error assumes all stones are alive, *i.e.*, ● is BLACK and ○ is WHITE.

	Error (%)			
Algorithm	Vertex	Block	Winner	Game
Naive	6.79	17.57	30.79	75.70
Stern et al. (2004)	4.77	7.36	13.80	38.30
Block graph	2.36	3.56	4.53	13.02
Block graph + neighbour features	1.87	2.76	3.42	9.60
Block graph + other enhancements	1.54	2.20	2.09	7.90
* GnuGo	-	-	-	1.32
* van der Werf et al. (2005)	0.19	≤ 1.00	0.50	1.10

Figure 6. Prediction error (percentage) for various algorithms.
*: employs Go- specific features and was used to label training data.

Conclusion and Future Work

- Our system is clearly outperforming previous generic models, however it is worse than approaches that incorporate Go-specific features.
- Our high-order graph representation enables the system to capture the subtleties of life and death in Go, despite our simple generic features.
- Future work will focus on better ways of classifying empty regions and incorporation of more domain-specific knowledge.

References

T. Graepel, M. Goutrié, M. Krüger, and R. Herbrich. Learning on graphs in the game of Go. In *ICANN*, 2001.

A. Globerson and T. Jaakkola. Approximate inference using planar graph decomposition. In *NIPS*, 2007.

E. C. D. van der Werf, H. J. van den Herik, and J. W. H. M. Uiterwijk. Learning to score final positions in the game of Go. *Theoretical Computer Science*, 349(2), 2005.

D. H. Stern, T. Graepel, and D. J. C. MacKay. Modelling uncertainty in the game of Go. In *NIPS*, 2004.