ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

SVM and Pattern-Enriched Common Fate Graphs for the Game of Go

Liva Ralaivola[1,2],Lin Wu[2], Pierre Baldi[2]

1- Centre de Mathématique et Informatique, Laboratoire d'Informatique Fondamentale
39, rue F. Joliot Curie, F-13453 Marseille cedex 13, France

2- School of Information and Computer Science, Institute for Genomics and Bioinformatics
University of California Irvine, Irvine, CA 92697-3425, USA

Abstract. We propose a pattern-based approach combined with the concept of *Enriched Common Fate Graph* for the problem of classifying Go positions. A kernel function for weighted graphs to compute the similarity between two board positions is proposed and used to learn a support vector machine and address the problem of position evaluation. Numerical simulations are carried out using a set of human played games and show the relevance of our approach.

1 Introduction

The game of Go, is the board game that poses the most difficult challenges for computer programmers. The difficulty to build a high-level program comes from the complex nature of the different concepts to take into account: liberties, territory, life and death, etc. In addition, the branching factor is so large, i.e. roughly between 50 and 65, in the middle of a game played on a 9×9 board, that any search strategy easily becomes prohibitively expensive. Therefore, the effort toward the construction of a strong Go program has to be devoted to an accurate evaluation function capable of discriminating winning and losing positions at any stage of a game. We propose an approach to this problem which builds on pattern extraction ideas [1] and the *common fate graph* representation [2].

This paper is organized as follows. We first (section 2) briefly recall the rules of the game of Go. Then, focusing on boards of size 9×9, section 3 describes our new *Enriched Common Fate Graph* (ECFG) representation of positions and a kernel feature map for ECFG so kernel methods such as support vector machines can be used to learn the evaluation function. In section 4, we report results on the effectiveness of our approach for the classification of positions, showing in particular how the information provided by the patterns is strongly relevant.

2 Game of Go

Go is a two-player board game played on 19×19 (official size) 9×9 (for beginners and computer programs) and 13×13 square boards. The goal of both black and white players is to finish the game with the largest *territory*. To do so, Black and White obey the simplified set of rules: starting with an empty board, Black and White move alternatively, starting with Black. Each move consists in putting a stone at an empty intersection of two lines, which may lead to the *capture* of one or several opposite

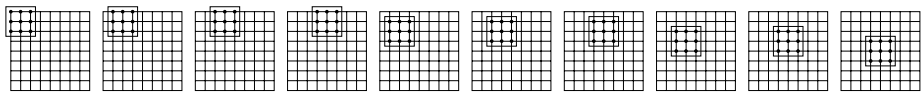


Fig. 1: Locations of the 3×3 patterns. Taking into account all the symmetries and rotations that leave the strength of a position unchanged, 10 distinct locations must be considered.

stones, when these stones happen to no longer have any *liberties*. Liberties correspond to empty intersections adjacent to *groups* or *chains*, defined as sets of stones of the same color connected through the lines of the board; all the stones of a chain share the same liberties. When the number of liberties of a chain becomes zero, all the stones of the chain are captured and removed from the board. A player may pass his turn at any time and the game is over when both of them do so. The territory owned by a player corresponds to the number of intersections occupied or surrounded by his stones.

The hardness of building an accurate and reliable evaluation function stems from the need of balancing the opposing characteristics of a position, namely the spatial local configurations and the general shape on the one hand, and, on the other hand, the tactical and the strategical aspects. The following section presents our strategy to automatically learn a powerful evaluation function from a set of human-played games.

3 Pattern Enrichment of Common Fate Graph

We propose to envision the task of building an evaluation function as a learning problem, using a set $\mathcal{S}_b = \{(\mathbf{b}_1, y_1), \ldots, (\mathbf{b}_\ell, y_\ell)\}$ of labeled positions where each $\mathbf{b}_i$ is a position (see, e.g., Fig. 2, left) and $y_i \in \{win, loss\}$ depends upon whether $\mathbf{b}_i$ is a winning or a losing position for Black. Using a target value related to the outcome of the game, instead of one that corresponds to an immediate return allows us to put more emphasis on the strategical aspect of a position. However, given the total number of configurations of a 9×9 board ($\approx 10^{37}$ if invariances are considered), it is hard to learn a classifier from a limited set of positions coded as raw boards. To overcome this issue, we propose the Enriched Common Fate Graph structure, inspired on [2].

3.1 Automatic Pattern Extraction

A common approach used in board game programming is to divide the board in small areas of interest, to assign values to the configuration of the material, i.e stones, or *patterns*, in those small areas and then to combine these values to get a final value for the whole position. We propose to automatically extract and assign a value to all 3×3 square patterns, located at different positions, that can be encountered on a 9×9 board (see Fig. 1). Using a set of games with known outcome and final territory possessions, we define for each pattern p, the values $\text{stab}(p)$ and $\text{imp}(p)$, computed as:

$$\text{stab}(p) = \frac{\text{bedge}(\text{loc}(p), p) - \text{wedge}(\text{locloss}(p), p)}{\text{occ}(p)}, \quad \text{imp}(p) = \frac{\text{win}(p) - \text{loss}(p)}{\text{occ}(p)} \quad (1)$$

where $\text{occ}(p)$ is the number of times p occurs in a position (throughout all the games), $\text{loc}(p)$ the location of p on the board, $\text{bedge}(\text{loc}(p), p)$ (resp. $\text{wedge}(\text{loc}(p), p)$) the

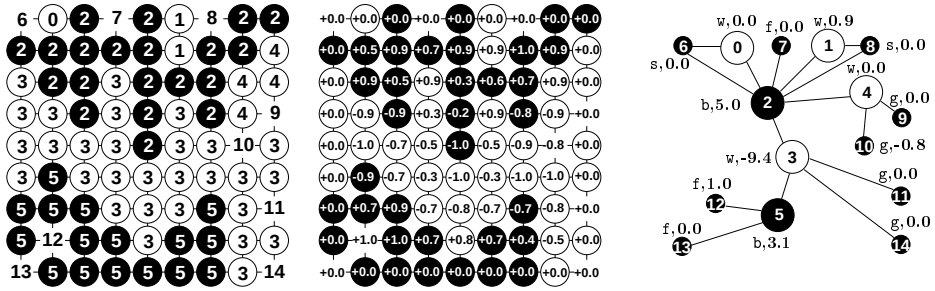


Fig. 2: Left: a position with, at each location, the number of the group an intersection belongs to. Middle: the weights associated to each location not on the edge of the board as provided by the pattern database. Right: the Enriched Common Fate Graph corresponding to the position; each node corresponds to a group/empty location and has a label from Table 1 and a weight obtained as a function (here, the sum) of the weights of the intersections in the group.

number of times there is a larger territory for black (resp. white) at $\text{loc}(p)$ at the end of the game when p occurred during the game and, finally, $\text{win}(p)$ (resp. $\text{win}(p)$) the number of times p occurs in a winning (resp. losing) position for Black. $\text{stab}(p)$ is a value that we call *stability* and, even though it is computed with respect to the final result of the game, it can be seen as a tactical feature. On the contrary, $\text{imp}(p)$ is a value that measures the *impact* of a pattern toward the outcome of the game.

These values can easily be used to give a weight to each (occupied or unoccupied) intersection of the board not on the edge as shown on Fig. 2, middle. However, even if each intersection of a board may now be assigned a value, learning a classifier still requires a way to both reduce the number of possible board positions and take into account the crucial property that all the stones of a group share a *common fate*, i.e. they either all live or die. To respond to this need, we consider the Enriched Common Fate Graph representation.

3.2 Enriched Common Fate Graph

The *Common Fate Graph* (CFG) [2] approach consists in transforming a position into a node-labeled graph, with the three labels `b` (black), `w` (white) and `e` (empty), where each `b` or `w` node corresponds to a black or white chain, respectively, while `e` nodes are empty intersections; an edge between two nodes is present when the corresponding elements on the board are adjacent. This representation is compact, independent of the orientation of the board and it uses the common fate idea. However, the information about the local patterns occurring on the board and the different types of empty squares is lost. To overcome this, we introduce the *Enriched Common Fate Graph* (ECFG) structure, which extends the CFG approach with (a) new labels for empty intersections, see Table 1, and (b) the assignment of a weight to each node of the graph (Fig. 2, right).

Dealing with the new node labels is rather straightforward given their definitions (see Table 1) and the same strategy as that used for CFG can be used. As for the weights, if we consider the stability feature, for instance, the weight $\text{wgt}(\mathbf{n})$ of a node

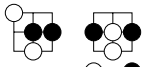
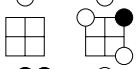
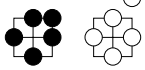
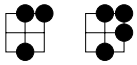
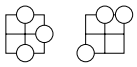
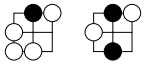
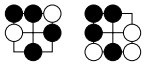
Label	Description	Example
b, w	black or white stone	
j	empty intersection surrounded by empty intersections	
f , g	empty intersection surrounded by black stones and white stones, respectively (cf. *eye* and *false-eye*)	
f 1, f 2, f 3	empty intersection needing 1,2 or 3 black stone(s) to become of type f	
g1, g2, g3	empty intersection needing 1,2 or 3 white stone(s) to become of type g	
m	empty intersection adjacent to another empty intersection, a black stone and a white stone	
s	empty intersection totally surrounded by black and white stones	

Table 1: The labels used for the nodes of Enriched Common Fate Graphs. The last column represents small patterns centered at intersections having the corresponding label.

$\mathbf{n}$ corresponding to the intersections $\{n_1, \ldots, n_q\}$ is computed as:

$$\text{wgt}(\mathbf{n}) = \sigma\left(\sum_{i \in \{n_1,\ldots,n_q\}} \text{stab}(p_i)\right) \tag{2}$$

where p_i is the pattern centered at the location i (cf. Fig. 2, right) and σ a non-decreasing function.

Like CFG, ECFG is a compact representation that is independent of the rotations and symmetries of the board and that conveys the importance of the common fate property. In addition, it precisely characterizes empty intersections and can encompass the strength of local features through the weights of the nodes.

3.3 Kernel Feature Map for ECFG

ECFG is a nice representation of board positions for the reasons mentioned previously and we use it for learning a classifier. To do so, given the set of labeled positions $\mathcal{S}_b = \{(\mathbf{b}_1, y_1), \ldots, (\mathbf{b}_\ell, y_\ell)\}$, we consider the 'equivalent' training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, where $\mathbf{x}_i$ is the ECFG of $\mathbf{b}_i$ and we learn a Support Vector Machine (SVM)[3, 4]; the use of an SVM is motivated by the fact that it is a handy and powerful classifier as soon as a suitable kernel function is available.

To build kernels for node-weighted labeled graphs, we propose the feature map $\boldsymbol{\phi}_d(\mathbf{x}) = (\phi_{path}(\mathbf{x}))_{path \in P(d)}$ for a graph $\mathbf{x}$, where, for some d, $P(d)$ is the set of all

possible strings of length up to d obtained by concatenating the labels of Table 1 and

$$\phi_{path}(\mathbf{x}) = \begin{cases} 0, \text{ if } path \text{ is not the label of any path of } \mathbf{x} \\ \dfrac{1}{\text{length}(path) \cdot |Q_{\mathbf{x}}(path)|} \displaystyle\sum_{t \in Q_{\mathbf{x}}(path)} \text{wgt}(t) \end{cases}$$

where length$(path)$ is the length of the string $path$, $Q_{\mathbf{x}}(path)$ the set of the paths in $\mathbf{x}$ having the label[1] $path$ and wgt(t) the sum of the weights of the nodes in the path t.

Using this feature map, it is easy to derive several kernels such as, for instance the kernel k_d, defined for some d as – $\mathbf{x}_i$ and $\mathbf{x}_j$ being two weighted labeled graphs:

$$k_d(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}_d(\mathbf{x}_i), \boldsymbol{\phi}_d(\mathbf{x}_j) \rangle = \sum_{path \in P(d)} \phi_{path}(\mathbf{x}_i) \phi_{path}(\mathbf{x}_j). \quad (3)$$

4 Numerical Simulations

We carry out experiments using a set of roughly 3500 games kindly pre-processed and provided by Nicol N. Schraudolph. We separate the dataset in a training set $\bar{t}$ of 3150 games and a test set t of 350 games. All the positions of the games are 'normalized' (such that two positions that are equal up to a rotation or a symmetry are mapped to the same position) and, if necessary, the stones are switched so that Black is the next player to move. The label (win or $loss$) associated to a position corresponds to the largest number of black wins/losses this position appears in.

We compute the stability values and impact values for the 3×3 patterns on $\bar{t}$ according to (1) and we use the kernel k_d of (3) for $d = 1, 2, 3$. In order to measure the improvement due to the use of ECFG, we also consider ECFG representations without weights – which corresponds to the CFG representation with a wider range of labels. To compute the weight of a group, we use (2) and $\sigma(x) = (1 + \exp(-x))^{-1}$ since our preliminary experiments showed it performs better than the simple identity function. The SVM implementation we use is SVM^{light} [4].

We focus on boards having between 20 and 50 stones – which corresponds to the most difficult part of the game – and consider the training sets $\bar{t}_{20}, \ldots, \bar{t}_{50}$ and testing sets $t_{20}, \ldots, t_{50}$ respectively comprising boards with $20, \ldots, 50$ stones. The values 500, 1000, 2000, 4000 and 8000 are tested for the parameter C of the SVM (see, e.g. [4] for the meaning of C) and we retain for each feature (i.e. impact, stability or 'count' – the latter corresponds to the kernel used in [2]) and depth d combination the one that minimizes the average testing error on $\bar{t}_{s-1}$ and $\bar{t}_{s+1}$. For each given number s of stones, we then estimate the generalization accuracy as the average accuracy measured on t_{s-1}, t_s and t_{s+1}.

The results plotted on Fig. 3, middle, correspond to the values of the classification accuracy averaged on the three different features for the different values of d. It is obvious from this graph that larger values of d lead to better accuracies. This is in agreement with the intuition that the longer the paths that are extracted from a graph the more the information on the structure of the graph is preserved. Looking at Fig. 3,

[1] The label of a path is the string obtained by concatenating the labels of the nodes of the path.

d	imp.	stab.	count
1	62.5	61.9	56.9
2	62.2	63.2	60.5
3	61.7	64.1	64.3

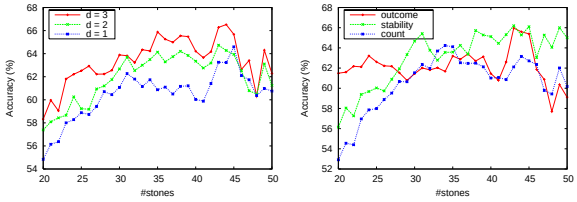


Fig. 3: Left: averaged generalization accuracies for feature/d combinations ('imp.' and 'stab.' stand for impact and stability, respectively. Middle: the test accuracy averaged across the different features for different values of d. Right: the test accuracy averaged across the values of d for the different features.

right, it clearly appears that the accuracies, when averaged on the different values of d, give the edge to the ECFG representation with the stability feature. However, it must be noticed that when the accuracies are not averaged and considered one by one (Fig. 3, left), the ECFG representation with the count feature and $d = 3$, provide an accuracy of 64.3%, slightly better than the second best accuracy, obtained by ECFG with $d = 3$ and the stability feature which lead to an accuracy of 64.1%. We suspect this is due to an overfitting problem caused by the discrepancy between the richness of ECFG and the relatively small number of training positions (≈ 3150). Notwithstanding these counterintuitive results, we see that the classification accuracies achieved on the test sets are well above the chance level for the well-balanced datasets used.

5 Conclusion

We have proposed the Pattern-Enriched Common Fate Graph representation of Go positions, which mixes a pattern based approach and a graph-based approach to model Go positions. Using an SVM we have conducted simulations on a set of human played games and obtained encouraging results. In the near future, we plan to conduct experiments on larger training sets to avoid the overfitting problem witnessed in some of the simulations presented here. A gradient descent strategy to learn the feature map and weight the different patterns is also envisioned.

References

[1] M. Buro. Experiments with Multi-ProbCut and a New High-Quality Evaluation Function for Othello. In *Games in AI Research*, 2000.

[2] T. Graepel, M. Goutrie, M. Krüger, and R. Herbrich. Learning on graphs in the game of go. In *Proc. of Int. Conf. on Artificial Neural Networks (ICANN-01)*, Vienna, Austria, 2001.

[3] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:1–25, 1995.

[4] T. Joachims. Making Large-Scale Support Vector Machine Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Adv. in Kernel Methods – Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1998.