狸王
アピール文書

ザイオソフト コンピュータ将棋サークル
野田久順 岡部淳
鈴木崇啓 河野明男

目次

- 狸王
- 主な工夫点
  - NNUE評価関数
  - 強化学習における教師局面生成
    - 思考ノード数の固定
    - 棋譜生成器の探索パラメータの調整
  - 探索パラメーターの自動調整
  - クラスタリング
- 使用ライブラリ

狸王

野田「『〇〇〇〇〇〇〇〇タヌポン』っていう
　　参加名にしようと思うんですけど…」
岡部「・・・」
那須「・・・第6回電王トーナメントに使う
　　予定だった名前のほうが良いと思います」
野田「…あぁ、…そうしましょうか」

狸王(続)

- もし第6回電王トーナメントが
  開催されていたとしたら、
  この名前で出場する予定でした
- ディフェンディングチャンピョン(＝王者)
  という意味が込められています

NNUE評価関数

- 狸王は評価関数にNNUE評価関数を使用しています
  - ネットワーク構造にhalfkp_256x2-32-32を使用しています

強化学習における教師局面生成

- 狸王では評価関数を強化学習により強化しています
  - 「教師データが不足した環境での機械学習結果改善手法」と「elmo式学習法」を用いています
- 教師局面の生成に工夫を加えています
  - 探索深さと思考ノード数の制限
  - 棋譜生成器の探索パラメータの調整

思考ノード数の固定

- これまでの教師局面生成では、
  探索深さを固定して自己対局を行い、
  棋譜を生成してきました
  - 探索深さを深くした場合、対局の終盤において、
    思考ノード数が極端に増え、
    思考時間が長くなる場合がありました
- 狸王では探索深さ加え、思考ノード数も制限し
  思考時間が長くなりすぎるのを抑制しています
  - 棋譜生成時間が短くなりました

棋譜生成器の探索パラメータの調整

- 棋譜生成器の探索パラメータを、自己対局のパラメータに合わせて最適化しています
  - 後述の手法でパラメータの自動調整を行っています
- これにより、より質の高い教師局面が得られるようになったと考えています

探索パラメーターの自動調整

- WCSC26で導入した、
  探索パラメータの自動調整を行っています
  - hyperoptで探索パラメータと
    自己対局の勝率のサンプリングを行い、
    Gaussian Processを用いて
    なるべく良さそうなパラメータを選んでいます

探索パラメーターの自動調整 (続)

※WCSC27PR文書より抜粋・改変

- 探索パラメーターを入力、ベースとなる思考エンジンに対する勝率を出力とする関数を考えます
- 関数の値を最大化する探索パラメーターを探します
  - 大域的最適化問題にモデル化できます
- 自己対戦により得られる勝率には大きなノイズが含まれています

探索パラメーターの自動調整 (続)

- Tree-structured Parzen Estimator (TPE)
  - 機械学習のハイパーパラメーターの最適化に用いられるアルゴリズムの1つです
  - PythonライブラリHyperoptに実装されています
  - 1探索パラメーターセットあたり48対局、数百パラメーターセット分の自己対戦の勝率をサンプリングします
  - Hyperoptの出力をそのまま使ったらダメ！
    - Hyperoptは最高の値を出力した最初の値を出力します
    - サンプルの誤差が大きい場合、本来弱いはずの探索パラメーターが最終結果として出力されてしまう場合があります

探索パラメーターの自動調整 (続)

- Gaussian Process (GP)
  - ノンパラメトリックな回帰分析などに用いられています
  - sklearn等に実装されています
  - Hyperoptで得られたサンプルをGaussian Processでノンパラメトリック回帰分析します
  - 探索パラメーターに対する勝率の関数の確率分布が得られます
  - サンプルの中で関数の確率分布の平均値が最も高いものを最終的な解として出力します

探索パラメーターの自動調整(続)

- 前回からの変更点1
  - サンプリング初期の、ランダムサーチによるサンプリング回数を増やしています
    - これにより、局所最適解に陥りにくくなると考えています

探索パラメーターの自動調整(続)

- 前回からの変更点2
  - 自己対戦を思考ノード数を固定して行っています
    - 時間を固定した場合、各思考エンジンに割り当てられる
      CPUリソースがばらつき、
      勝率が50%に寄る問題があります
    - 思考ノード数を固定することにより、
      この問題を解決しました
    - 合わせて、同時対局数を物理CPUコア数から
      論理CPUコア数に増やし、サンプリング精度を
      上げています

クラスタリング

- 以下のクラスタ手法を組み合わせます
  - Lazy Cluster (eXtream Lazy Smp)
  - ゲーム木の分割
  - Multi Ponder
  - 詰将棋専用ノード

クラスタリング (続)

- Lazy Cluster (eXtream Lazy Smp)
  - 思考ノード間で置換表の内容を共有します
  - USIコマンドを拡張して置換表エントリを送受信します
  - PV上の局面に対応する置換表エントリのみ共有します

クラスタリング (続)

- ゲーム木の分割
  - ルート局面でmultipvで手を生成します
  - ルート局面と生成された手それぞれに思考ノードを割り当て、思考させます
    - 深い探索
      - 子局面から読み始めることにより、より深い探索ができると考えられます
    - 読み抜けの軽減
      - Stockfish型探索ルーチンは、non-PVでは強い枝刈りが行われるため、読みが雑になるとされています
      - ルート局面のnon-PVを、子局面担当ノードがPVとして読むため、読み抜けが減る効果があると考えられます

クラスタリング (続)

- ゲーム木の分割
  - 探索結果はLazy Clusterで全ノード間で共有します
    - Multi Ponderにヒットした場合、そのノードが探索していたPVの探索結果のキャッシュが共有されるため、その他のノードが子局面を探索する際に、探索効率が良くなると考えられます

クラスタリング (続)

- Multi Ponder
  - WCSC28でHefeweizenが採用した手法です
  - Ponder時、直前の局面でmultipvで手を生成します
  - 生成された手それぞれに思考ノードを割り当てます
  - Ponderヒット時は、ヒットした局面をルート局面として
    思考を続行します
    - それ以外のノードはゲーム木の分割により
      並列探索を行います
  - Ponderアンヒット時は、 ゲーム木の分割により
    並列探索を行います

クラスタリング (続)

- 詰将棋専用ノード
  - 複数の詰将棋エンジンに、現在の局面・子局面・Multi Ponderの各局面をそれぞれ割り当て思考させます
  - 現在の局面・Multi Ponderの各局面では即詰みを、子局面では頓死をそれぞれ発見できます
  - 探索結果はLazy Clusterで思考ノードに共有します

使用ライブラリ

- Apery
  - tanuki-wcsc28開発時点で評価関数が最も強かったため評価関数をランダムスタートで機械学習させるための教師局面の生成に使用しました
- やねうら王
  - 独自の工夫を加えるにあたり、改造しやすく、レーティングも高いため、使用しました
- tanuki-
  - 強化学習のもとになる評価関数として使用しました

よろしくお願いします