FLUKE

第30回世界コンピュータ将棋選手権アピール文章

作成: 井本 康宏 作成日: 2019/3/29

忙しい人のためのチートシート

これさえあれば、1分で全てわかる

- Flukeは以下の要素で成り立っています
- YaneuraOuの評価関数を改造
 - 去年はDL勢だったけど、一時的に鞍替え
 - それに伴って名前をWindfallからFlukeに変更
- NNUEのネットワーク構造をテーマに試行錯誤
 - なんちゃってHighway Networkで大会に挑む予定
 - 実装が間に合うかどうか
 - 間に合わない場合は標準NNUEに自前で学習したパラメータで参加

開発者自己紹介

- 開発者
 - 井本 康宏 (twitter: @Windfall_shogi)
- 職業
 - エンジニア
- 棋力
 - 測ったことがないのでわかりませんが、すごく弱いです
 - 学生時代に囲碁・将棋部だったなんて言えない
- 開発のきっかけ
 - 趣味、好奇心、研究スキル、プログラミングスキルの向上を目的として、当時、話題だったこともあり開発を始めた

HIGHWAY NETWORK

- *Srivastava, Rupesh Kumar; Greff, Klaus; Schmidhuber, Jürgen (2 May 2015). "Highway Networks". [arXiv:1505.00387](https://arxiv.org/abs/1505.00387)*
 - $y = T(x) \otimes H(x) + (1 - T(x)) \otimes x$ ⊗は要素ごとの積
 - $T(x)$ は適当な演算でシグモイド関数で正規化されている
 - $H(x)$ は全結合層
 - 深い構造のネットワークの学習に有効
 - Residual Networkの拡張版といえる

Highway NetworkのアイデアをNNUEに取り込む

なんちゃってHIGHWAY NETWORK

- NNUEで使えるようにする必要がある
 - 入力と出力の次元数が違う
 - 前項の式を見ればわかるが入力と出力の次元が同じ
 - シグモイド関数は使えない
 - 整数型での演算では無理
- 改造版
 - $y = U(x) \otimes H_1(x) + (1 - U(x)) \otimes H_2(x)$
 - x の代わりに $H_2(x)$ で次元数を調整
 - $U(x)$ はclipped reluで正規化
 - $U(x)$ によって次元ごとに割合が変化するので、単純な全結合層で置き換えはできないはず

なんちゃってHIGHWAY NETWORKの副作用

- 計算量が増える
 - $H_1(x)$, $H_2(x)$, $U(x)$ の3個を計算
- 解決策
 - 全結合層の行列をブロック対角にして演算回数を減らす
 - 個々の全結合層の表現力が低下しているので、Highway全体の表現力が普通の全結合層より勝っているのかは不明
- そもそも実装が間に合うかという問題もある
 - ダメだった場合は標準NNUEで行きます

おまけ

試行錯誤の残骸

試したけど、ダメだったよ

何の成果も得られませんでした！

- さすがに皆無ではない
- お蔵入りが多い
 - 標準NNUEのクロスエントロピー損失と同程度かそれより悪いことが多い
 - きちんと対局での性能を評価したわけではないが、期待できる感じではない
 - 中には全く最適化が成功しなかったものもある
 - optimizerの種類や学習係数など、またネットワークの初期値によっては改善するかも？

残骸1

- clipped reluの範囲を[0, 127]から[0, 255]に拡大
 - オーバーフローの可能性が出てくるが、非線形性が増えて逆に好都合
- 学習時間が増える割に性能に差がない
 - オーバーフローの頻度を考えると当然でしょう

残骸2

- 全結合層への入力を $[-128, 127]$ 、行列のパラメータを $[0, 127]$ に符号ありと符号なしを入れ替え
- 全く学習できませんでした
 - パラメータではなく入力値の方が変化しないといけないので、学習が非常に難しいのだと思う

残骸3

- 標準NNUEだけど普通の強化学習の手法で学習
 - 振り飛車を学習してくれたら儲けものという軽い考え
 - 学習に独自性が発揮できる
- 全く学習できませんでした
 - 全サンプルに対する評価が同じ値になってどうしようもなかった
 - 学習係数やパラメータの初期値で改善できると今は思う
 - ネットワークが小さいので局面Aの評価値はそのままで、局面Bの評価値だけを修正するというのがかなり難しいように思う
- サンプルの生成速度が遅くてつらい

残骸4

- optimizerの種類を変えてみる
 - 標準NNUEを楽に学習できるとうれしいな
 - 学習のやり直しが楽になるので、評価関数を量産できる
- SGDが一番良さそう
 - 勾配が適応的なものはembedding層の学習ができていない
 - NNUEは勾配が0になりやすいうえに入力が疎
 - 勾配のモーメントの計算に工夫が必要
 - 全結合層とembedding層で別のoptimizerを使ってみたが、それぞれの学習係数の調整が難しい
 - 一般の機械学習でもSGDが最も性能が高くなるといわれているので、NNUEでも同様かと

残骸5

- 大量の教師データで学習
 - 大量の局面の評価値を覚えられたなら、きっとそれは強い
 - 強化学習に懲りた反動
- 数億程度が覚えられる限界かな？
 - 数億を超えても目的関数の値を僅かながら改善し続けられる
 - しかし、評価を大幅に間違えている局面が一定数は存在する
 - VC次元を考えれば当然？
 - 少数の訓練データを完全に覚えさせたほうがいいのかも
 - 負けた気がするので、それはやっていない

残骸6

- 評価値を勝率にして ± 0.1 の誤差を許容して学習
 - 誤差を許容することで、覚えられる局面が増えるかも
- 別に変わらないかむしろ悪化する
 - 誤差 ± 0.1 の範囲で勾配を0にしても誤差のヒストグラムは0を中心に単峰になる
 - ただし、学習を続けると誤差のヒストグラムは $+0.1$ と -0.1 の2カ所に山ができる

残骸7

- なんちゃってAttention mechanism
 - softmaxを省略して、単純に掛け合わせる
- なぜかたまに勾配の値が爆発する
 - $1e+30$ とかすごい値
 - よくわからない
 - 先後同型の場合は同じパラメータについて何度も計算されるので、そのせいと予想

ちょっと使えそうなもの

- 順方向と逆方向で活性化関数が同じである必要はない
 - 順方向でclipped relu
 - 逆方向でsigmoid
 - 効果のほどは不明
 - 勾配が消える心配はしなくてもいい
 - そもそも初期値ですべてが決まる気もする
- Yogi optimizer
 - Adamの改良版
 - SGDの場合との違いは確認できなかった
 - NNUEは小さいので、学習が進めばすぐに性能が頭打ちになるせいだと思う