

進捗報告

1 今週やったこと

- 論文読んだ
- CNN の縮小

2 論文

貢献度分配を導入した方策勾配による Neural Architecture Search の高速化 [1]

2.1 はじめに

アーキテクチャ最適化は, 学習に時間がかかる

One-shot アーキテクチャ アーキテクチャとパラメータを同時学習. アーキテクチャを連続的に表現して、微分可能にする.

2.2 Neural Architecture Search

2.2.1 有効非巡回グラフとしての DNN

ノード 特徴 x

エッジ オペレーション ϕ

a_i エッジ i のアーキテクチャ. $|a_i| = 1$ のとき one-hot ベクトル.

2.2.2 One-shot Architecture Search

$R(a; w_a)$ 評価関数. ここでは負の交差エントロピー誤差.

w_a パラメータ. \hat{w} に保存した学習済みパラメータから取得することで学習コストを削減.

2.3 関連研究

2.3.1 ENAS

REINFORCE

目的関数 $E_{a \sim p_\theta(a)}[R(a)]$

アーキテクチャ a

2.3.2 DARTS

連続値アーキテクチャ

評価関数 $E[R(\mu(\theta))]$

アーキテクチャ $\theta \in [0, 1]^{|I||J|}$

学習後確率分布 θ から one-hot 化する

$$a_{ij} = \lim_{\lambda \rightarrow 0} [\mu_{ij}(\theta/\lambda)]$$

$$j = \arg \max_j \mu_{ij}(\theta)$$

2.3.3 SNAS

連続値アーキテクチャ

評価関数 $E_G[R(m(\mu_{ij}(\theta), G))]$

アーキテクチャ DARTS 同様

G 標準 Gumbel 乱数

$m_{ij}(\theta, G)$ $\mu_{ij}(\theta)$ に乱数 G と温度パラメータ λ を導入

2.3.4 ProxylessNAS

評価関数 $E_N[R(z(\mu(\theta), N))]$

アーキテクチャ DARTS 同様

$z_i(\mu(\theta), N)$ 乱数 N から two-hot ベクトルを生成

2.4 提案手法

- DARTS, SNAS
オペレーション候補ごとに計算. $O(|J|)$.
- 提案手法
貢献度分配で効率化. $|J|$ に依存しない

2.4.1 貢献度

各エッジごとに, 全体に与える評価 $R(a)$ の善し悪しがある.

各演算子を一度に更新できる

貢献度 A : 全体評価 $R(a)$ に与える影響. 各エッジの評価

$$A(a_i, S_i) = R(a_i, S_i) - R(\vec{0}, S_i)$$

S_i : a から要素 a_i を除いた集合

2.4.2 効率的な貢献度の計算

貢献度 A を近似して R の評価回数を減らす (13) (14)

2.5 実験

2.5.1 アーキテクチャ探索

- セル
(エッジ 14, ノード 7) × 8 個
- ノード
c-1, c-2 番目のセルからの入力, c 番目のセルの出力, 中間ノード × 4
- エッジ候補
3x3 / 5x5 sep conv, 3x3 / 5x5 dilated sep conv, 3x3 max pooling, 3x3 average pooling, 恒等写像, 零写像の 8 つ
- 学習
~ 50 epoch(重み学習), 50 epoch ~ 150 epoch(θ も更新)

結果 得られた θ を one-hot 化し, 重みを再学習

- DARTS, SNAS に迫る精度
- DARTS, SNAS の 3 割程度の学習時間

3 考察

- アーキテクチャをカテゴリカルから連続的な確率分布に
- ネットワークの重みを再利用して学習を削減
- 見つけたアーキテクチャの重みを再学習する (acc 6 割 → 9 割)

重みの再利用

1. 冗長にネットワーク構造を決めておく. (あるノードはそれ以前のノード全てに接続可能とする)
2. 重みを学習して, 各エッジ, 各演算子ごとに重みを保存
3. アーキテクチャ (接続するか? + 演算子) を探索

得られたセルでは, 直前のセルからの入力を恒等写像でそのまま出力していた.

畳み込み層は全て separable convolution を利用している. パラメータ数を減らしたかった?

アーキテクチャの表現法

- RNN でパラメータを生成 (カテゴリカル)
- GA で個体表現 (カテゴリカル)
- 演算子の確率分布集合として表現

4 RNN の動作実験

目的: RNN でパラメータを生成

4.1

input: 系列データ [1, 0, 1, 0]

output: 系列データ [1, 1, 0, 0]

loss はほぼ 0 になり学習できた

4.2

直前の出力を入力することで再帰的にパラメータを出力

input: 初期乱数 z

output: 系列データ [1, 1, 0, 0]

出力が常に 0 となり, 学習できなかった.

5 今後の予定

- RNN の再帰的な入力の学習実験

6 ソースコード

[https://github.com/tatsuya-sugiyama/
WeeklyReport/blob/2020_0703/report/2020_
07_03/RNN_test.ipynb](https://github.com/tatsuya-sugiyama/WeeklyReport/blob/2020_0703/report/2020_07_03/RNN_test.ipynb)

参考文献

- [1] 佐藤 怜, 秋本 洋平, and 佐久間 淳. 貢献度分配を導入した方策勾配による neural architecture search の高速化. 人工知能学会全国大会論文集, JSAI2019:2P3J202–2P3J202, 2019.