퓨샷 러닝 및 경량 딥러닝 응용을 위한 최적의 학습률 탐색 알고리즘

김재현, 이재오, 정태순, 이상진, 정찬호*

한밭대학교 전기공학과

20181422@edu.hanbat.ac.kr, 20181383@edu.hanbat.ac.kr, 20172566@edu.hanbat.ac.kr,
20151709@edu.hanbat.ac.kr, peterjung@hanbat.ac.kr*

An Algorithm to Search Optimal Learning Rate for Few-Shot Learning and Lightweight Deep Learning Applications

Jaehyeon Kim, Jaeoh Lee, Taesun Chung, Sangjin Lee, Chanho Jung*

Hanbat National University

요 약

본 논문에서는 퓨샷 러닝 및 경량 딥러닝 응용을 위한 최적의 학습률 탐색 알고리즘을 제안한다. 제안하는 방법에서는 매 반복(iteration)마다 손실을 최소로 하는 학습률을 선택한다. 제안하는 방법과 기존 방법의 성능을 정성적 및 정량적으로 비교 평가하기 위해 MNIST 데이터셋을 이용하였다. 본 논문에서는 퓨샷 러닝을 위해 클래스당 학습데이터 개수를 제한하였다. 또한 경량 딥러닝 응용을 위해 간단한 구조를 가지는 네트워크 아키텍처를 이용하였다. 실험결과는 제안하는 방법이 기존 방법과 유사한 성능을 제공함을 보여준다. 뿐만 아니라 유의미한 학습률 탐색 결과를 볼 수 있었다.

Ⅰ. 서 론

딥러닝 모델을 학습시키는 과정에서 학습률과 같은 하이퍼파라미터를 적절하게 설정하는 것은 딥러닝 모델의 예측 성능과 밀접한 관련이 있는 것으로 알려져 왔다. 이러한 이유로 다양한 학습률 스케줄링 기법이 제안 및 이용되어왔다.

본 논문에서는 퓨샷 러닝 및 경량 딥러닝 응용을 위한 최적의 학습률 탐색 알고리즘을 제안한다. 구체적으로 매 반복(iteration)마다 손실을 최소로 하는 학습률을 선택한다. 이는 다음과 같은 가정을 기반으로 한다: “각 반복에서 갱신된 손실함수의 값이 global minimum에 가까운 값이 아닌 local minima에 가까운 값일 가능성이 있다.”. 본 논문에서는 퓨샷 러닝을 위해 클래스당 학습데이터 개수를 제한하였다. 또한 경량 딥러닝 응용을 위해 간단한 구조를 가지는 네트워크 아키텍처(two-layer network)를 이용하였고, 히든 레이어의 사이즈가 입력 레이어의 사이즈 대비 매우 작도록 설정하였다. MNIST 데이터셋을 이용하여 제안하는 방법과 베이스라인의 성능을 정성적 및 정량적으로 비교 평가하였다. 실험결과는 제안하는 방법의 정확도 및 베이스라인의 정확도가 비슷함을 보여주었다. 또한 제안하는 방법의 유의미한 학습률 탐색 결과를 볼 수 있었다.

Ⅱ. 제안하는 방법

본 논문에서는 매 반복마다 손실을 최소로 하는 학습률 선택 방법을 제안한다. 이때, 배치 사이즈가 작을 경우 전체 손실이 발산하는 문제를 실험을 통해 확인하였다. 따라서 학습률의 상한값을 설정하여 문제를 해결하고자 한다. 이는 다음과 같은 가정을 기반으로 한다: “큰 학습률로 인해 미니배치로 추출한 데이터에 과적합 학습이 되는 상황일 가능성이 있다.”. 제안하는 방법에서는 그림 1과 같은 동작 순서를 따른다.

$$\delta = \underset{x \in H}{\arg\max}\, G(x) \qquad (1)$$

Step 1: 학습률에 대한 proposal 집합 $H = \{\eta_1, \eta_2, \eta_3\}$를 정의한다.

Step 2: (a), (b), (c)중 만족하는 경우를 선택한다.

(a) $G_{\max} > \alpha$인 경우: H의 모든 원소에 $\frac{1}{10}$을 곱한다.

(b) $G_{\max} < \beta$인 경우: 식 (1)을 통해 δ을 결정한다.

(c) 그 외의 경우: η_1을 학습률로 설정하여 1회 갱신한다.

Step 3: δ이 결정될 때까지 Step 2를 반복한다.

Step 4: 학습률에 대한 새로운 proposal 집합 $K = \left\{\frac{1}{2}\delta, \delta\right\}$를 정의한다.

Step 5: 집합 K에서 매 반복마다 손실을 최소로 하는 학습률을 선택한다.

그림 1. 제안하는 알고리즘의 동작 순서

식 (1)과 그림 1에서 G는 갱신된 기울기 부호의 변화량(%)을 구하는 함수, δ는 학습률의 상한값, α와 β는 사전에 정의된 threshold를 나타낸다.

Ⅲ. 실험결과

본 논문에서는 제안하는 방법의 성능을 평가하기 위하여 two-layer network를 이용하였다. 히든 레이어 사이즈의 영향을 평가하기 위하여 히든 레이어의 사이즈를 각각 20, 50으로 설정하였다. 제안하는 방법의 구현에서 학습률에 대한 proposal 집합은 각각 $H_1 = \{0.01, 0.1, 1\}$, $H_2 = \{1, 10, 100\}$으로 설정하였다. 또한 그림 1의 α, β는 각각 8, 5로 설정하였다. MNIST 학습데이터셋으로부터 추출한 데이터를 two-layer network 학습을 위한 데이터로 이용하였다. 클래스당 학습데이터 개수에 따른 성능 변화 정도를 알아보기 위해 학습데이터를 추출하는 과정에서 클래스당 학습데이터 개수가 각각 1, 2, 3, 4, 5가 되도록 하였다. 성능 평가를 위해 10,000장의 샘플로 구성된 MNIST 테스트데이터셋을 이용하였

표 1. 배치 사이즈, 학습률, 클래스당 학습데이터 개수, 히든 레이어 내 뉴런 개수 변화에 따른 베이스라인 및 제안하는 방법 간의 성능 비교 (1행: 베이스라인(히든 레이어 내 뉴런 개수: 20), 2행: 제안하는 방법(히든 레이어 내 뉴런 개수: 20), 3행: 베이스라인(히든 레이어 내 뉴런 개수: 50), 4행: 제안하는 방법(히든 레이어 내 뉴런 개수: 50))

배치 사이즈	학습률	클래스당 학습데이터 개수				
		1	2	3	4	5
2	0.1	0.5221	**0.5505**	**0.6138**	**0.6165**	0.6494
	H_1	0.5221	0.5481	0.6049	0.6117	**0.6509**
	0.1	0.5123	0.5546	**0.6106**	**0.6430**	0.6569
	H_1	**0.5137**	**0.5561**	0.6080	0.6359	**0.6589**
4	0.1	0.5283	0.5575	**0.6139**	0.6145	**0.6459**
	H_1	**0.5304**	**0.5626**	0.6130	**0.6150**	0.6447
	0.1	0.5158	**0.5584**	0.6074	0.6387	0.6466
	H_1	**0.5162**	0.5548	**0.6088**	**0.6388**	**0.6495**
8	0.1	0.5236	0.5589	**0.6071**	0.6138	**0.6460**
	H_1	**0.5247**	**0.5593**	0.6053	**0.6153**	0.6440
	0.1	0.5124	**0.5567**	0.6081	**0.6397**	**0.6468**
	H_1	**0.5133**	0.5566	**0.6084**	0.6388	0.6467
2	0.01	0.5196	**0.5605**	0.6046	0.6133	0.6456
	H_2	**0.5207**	0.5466	**0.6051**	**0.6172**	**0.6509**
	0.01	**0.5160**	**0.5586**	0.6082	**0.6397**	0.6476
	H_2	0.5106	0.5535	**0.6097**	0.6363	**0.6538**
4	0.01	0.5224	**0.5605**	0.6039	0.6124	**0.6456**
	H_2	**0.5286**	0.5587	**0.6062**	**0.6231**	0.6397
	0.01	0.5142	**0.5599**	**0.6080**	**0.6405**	**0.6489**
	H_2	**0.5151**	0.5572	0.6067	0.6387	0.6457
8	0.01	0.5248	**0.5619**	0.5978	0.6044	**0.6450**
	H_2	**0.5257**	0.5578	**0.6066**	**0.6169**	0.6433
	0.01	0.5067	0.5545	0.6040	0.6342	**0.6494**
	H_2	**0.5096**	**0.5571**	**0.6068**	**0.6378**	0.6466

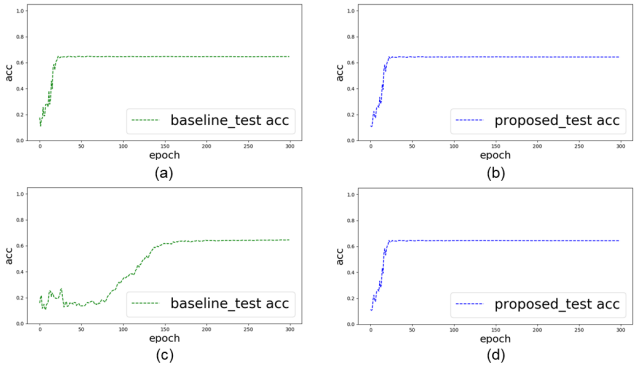

그림 2. 학습률에 따른 베이스라인과 제안하는 방법의 학습 경과
(a) 학습률: 0.1, (b) 학습률: H_1, (c) 학습률: 0.01, (d) 학습률: H_2

다. 배치 사이즈에 따른 성능 변화를 평가하기 위하여 배치 사이즈를 각각 2, 4, 8이 되도록 설정하였다. 표 1은 배치 사이즈, 학습률, 클래스당 학습 데이터 개수, 히든 레이어 내 뉴런 개수 변화에 따른 제안하는 방법과 기존 방법의 정량적 성능 결과 비교를 보여준다. 학습 종료 후 테스트데이터셋에 대한 분류 정확도를 측정하여 각 방법의 성능으로 판단하였다. 표 1에서 보는 바와 같이 제안하는 방법이 베이스라인과 유사한 성능을 제공함을 알 수 있었다. 그림 2는 학습률의 변화에 따른 제안하는 방법과 기존 방법의 정성적 비교를 보여준다. 매 epoch마다 테스트데이터셋에 대한 분류 정확도를 측정하였다. 그림 2에서 보는 바와 같이 제안하는 방법(b, d)은 기존 방법(a, c)과 달리 학습률의 변화에도 불구하고 정확도 그래프가 크게 다르지 않음을 확인하였다.

Ⅳ. 결론

본 논문에서는 퓨샷 러닝 및 경량 딥러닝 응용을 위한 최적의 학습률 탐색 알고리즘을 제안하였다. 실험결과를 통해 제안하는 방법이 베이스라인과 유사한 성능을 제공함을 알 수 있었다. 또한 제안하는 방법의 유의미한 학습률 탐색 결과를 보여준다.

ACKNOWLEDGMENT

이 논문은 국립대학육성사업의 지원을 받아 작성되었습니다.

참 고 문 헌

[1] Sung, Flood, et al. “Learning to compare: Relation network for few-shot learning,” *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1199-1208, 2018.

[2] Wang, Yaqing, et al. “Generalizing from a few examples: A survey on few-shot learning,” *ACM Computing Surveys*, pp. 1-34, 2020.

[3] Lee, Y. J, et al. “Recent R&D Trends for Lightweight deep learning,” *Electronics and Telecommunications Trends*, pp. 40-50, 2019.