퓨샷 러닝 및 경량 딥러닝 응용을 위한 개선된 경사 하강법

이재오, 정태순, 김재현, 이상진, 정찬호*

한밭대학교 전기공학과

20181383@edu.hanbat.ac.kr, 20172566@edu.hanbat.ac.kr, 20181422@edu.hanbat.ac.kr,

20151709@edu.hanbat.ac.kr, peterjung@hanbat.ac.kr*

An Improved Gradient Descent Algorithm for Few-Shot Learning and Lightweight Deep Learning Applications

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

Hanbat National University

요 약

본 논문에서는 퓨샷 러닝 및 경량 딥러닝 응용을 위한 개선된 경사 하강법을 제안한다. 제안하는 방법에서는 손실함수 기반 적응적 학습률 제어(ALRC, adaptive learning rate control)를 통해 기존 경사 하강법에 비해 향상된 성능을 제공하는 것을 목표로 한다. 제안하는 방법의 검증을 위해 MNIST 데이터셋을 이용하였다. 퓨샷 러닝 및 경량 딥러닝 응용을 위해 클래스당 학습데이터 수를 제한하였고, 매우 간단한 네트워크 아키텍쳐를 이용하였다. 제안하는 방법 및 기존 방법의 성능 비교를 위해 배치사이즈, 학습률, 클래스당 학습데이터 수 변화에 따른 분류 정확도 변화를 측정하였다. 또한 제안하는 방법의 generality를 평가하기 위하여 총 3개의 최적화 기법을 이용하였다. 실험 결과를 통해 1) 배치사이즈가 1, 8, 2) 학습률이 0.1, 3) 최적화 기법이 "Momentum", "Adagrad"일 때 제안하는 방법이 기존 방법보다 우수한 성능을 보인 것을 확인하였다.

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 하이퍼파라미터의 최적화 기법이 제안 및 이용되어왔다.

일반적인 경우 학습률이 고정되어 이터레이션마다 적절한 학습이 되지 않을 가능성이 있다. 따라서 본 논문에서는 ALRC 방법을 이용하여 이터레이션마다 최적의 학습률을 탐색하고자 한다. 이를 위해 매 이터레이션마다 최적화 전후의 loss 값을 비교하여 학습률을 조정해줄 값을 얻는다. 이 값을 통해 적절한 학습률로 조정하여 개선된 경사 하강법을 제안한다. 개선된 학습률을 통해 발산하거나 local minima에 수렴하는 것을 방지할 수 있다. 본 논문에서는 퓨샷 러닝[1] 및 경량 딥러닝[2] 응용의 가능성을 확인하기 위해 제한된 학습데이터 수와 매우 간단한 네트워크인 two-layer network를 사용하였다. 배치사이즈, 학습률, 클래스당 학습데이터 수 및 옵티마이저를 바꿔가며 실험을 진행하여 기존 방법과 제안하는 방법의 실험 결과를 비교 평가하였다. 실험 결과를 통해 1) 배치사이즈가 1, 8, 2) 학습률이 0.1, 3) 최적화 기법이 "Momentum", "Adagrad[3]"일 때 제안하는 방법이 기존 방법보다 우수한 성능을 보인 것을 확인하였다.

Ⅱ. 제안하는 방법

본 논문에서는 기존 방법의 파라미터 최적화를 진행하기 전과 후의 loss 값을 비교하여 학습률 조정을 위한 α 값을 구한다. α는 다음과 같이 정의된다.

$$\alpha_i = \frac{L(W_i)}{L(\widehat{W}_i)} \tag{1}$$

식 (1)에서 α_i는 i번째 이터레이션에서 얻은 학습률 조정 값이다. W_i는 i번째 이터레이션에서 갱신되기 전의 파라미터이고 $\widehat{W}_i$는 i번째 이터레이션에서 기존 방법을 이용하여 갱신된 후의 파라미터이다. $L(W_i)$는 i번째 이터레이션에서 손실 함수에 W_i를 넣어 얻은 loss 값이고 $L(\widehat{W}_i)$은 $\widehat{W}_i$를 넣어 얻은 loss 값을 의미한다. α_i를 이용해 초기 학습률 η에 곱하여 η_i를 구한다. η_i는 다음과 같이 정의된다.

$$\eta_i = \alpha_i \eta \tag{2}$$

식 (2)에서 η는 초기에 설정된 학습률이고, η_i는 α_i에 의해 개선된 i번째 학습률을 의미한다. 최종적으로 개선된 학습률 η_i를 이용해 W_i를 다시 최적화한다.

Ⅲ. 실험 결과

기존 방법과 제안하는 방법의 성능 비교를 위하여 MNIST 데이터셋을 이용하였다. 실험에서는 퓨샷 러닝 응용을 위해 클래스당 학습데이터 수를 각각 1, 2, 3개로 제한하여 총 10, 20, 30개의 학습데이터를 이용하였다. 성능 평가를 위해 10,000장의 샘플로 구성된 MNIST 테스트셋을 이용하였다. 네트워크의 경우 경량 딥러닝 응용을 위해 two-layer network를 사용하였고, hidden layer size를 20으로 고정하였다. 제안하는 방법의 generality를 확인하기 위해 옵티마이저는 "SGD", "Momentum", "Adagrad"로 설정하였다. 학습률과 배치사이즈 변화에 따른 기존 방법과 제안하는 방법의 성능 비교를 위해 학습률은 0.01,

표 1. 옵티마이저, 배치사이즈, 학습률, 클래스당 학습데이터 개수 변화에 따른 기존 방법 및 제안하는 방법 간의 성능 비교. 테스트셋에 대한 정확도는 모든 epoch에 걸쳐서 정확도를 측정하고 그 중 최댓값을 선택.

배치사이즈	학습률	클래스당 학습데이터 수	SGD		Momentum		Adagrad	
			w/o α	α	w/o α	α	w/o α	α
1	0.01	1	**51.74**	51.66	**51.76**	51.62	45.87	**46.29**
		2	56.20	**56.23**	54.31	**54.84**	**47.79**	47.64
		3	**60.15**	60.11	**59.42**	44.70	**54.47**	52.49
	0.1	1	**51.00**	45.23	20.35	**24.14**	23.31	**37.82**
		2	**55.42**	45.41	13.31	**17.57**	27.26	**32.73**
		3	**58.53**	49.45	12.56	**12.94**	**38.06**	35.81
4	0.01	1	**51.57**	51.56	51.35	**51.42**	**49.02**	48.91
		2	**56.33**	56.31	56.06	**56.08**	**53.64**	53.47
		3	**60.09**	60.09	59.75	**59.77**	58.38	**58.78**
	0.1	1	**51.63**	51.53	45.12	**49.80**	**37.19**	26.62
		2	**56.28**	56.16	52.06	**53.98**	**41.12**	35.16
		3	**60.46**	60.37	44.51	**57.85**	**45.21**	36.93
8	0.01	1	**51.21**	51.19	51.04	**51.14**	**49.81**	49.79
		2	56.26	**56.29**	56.04	**56.05**	**54.30**	54.26
		3	**59.83**	59.81	59.63	**59.64**	**60.15**	59.74
	0.1	1	51.39	**51.40**	45.03	**48.07**	40.21	**48.04**
		2	**56.23**	56.14	53.09	**54.25**	37.92	**51.15**
		3	60.14	**60.13**	56.35	**58.58**	43.38	**54.89**

0.1로, 배치사이즈는 1, 4, 8로 설정하였다. 표 1은 기존 방법과 제안하는 방법의 옵티마이저, 배치사이즈, 학습률, 클래스당 학습데이터 수 변화에 따른 정량적 비교를 나타낸다. 또한, 에폭마다 테스트셋으로 측정해 얻은 정확도 중 최대 정확도를 이용하여 각 방법의 성능으로 판단하였다. SGD의 경우 전반적으로 α값이 1에 수렴하여 기존 방법과 제안하는 방법의 성능 차이가 크지 않은 것을 확인하였다. Momentum의 경우 전반적으로 제안하는 방법이 기존 방법보다 좋은 성능을 보여주었고, 특히 학습률이 0.1일 때 더 우수한 성능을 확인할 수 있었다. Adagrad의 경우 학습률이 0.01일 때는 성능의 큰 차이가 보이지 않았다. 반면 0.1인 경우 배치사이즈가 1과 8일 때 성능 개선을 확인할 수 있었다.

Ⅳ. 결론

본 논문에서는 퓨샷 러닝 및 경량 딥러닝 응용을 위한 개선된 경사 하강법을 제안하였다. 실험결과를 통해 1) 학습률이 0.1, 2) 배치사이즈가 각각 1, 8, 3) 옵티마이저로 “Momentum”, “Adagrad”를 사용할 때 제안하는 방법이 기존 방법보다 우수한 성능을 보인 것을 확인할 수 있었다. 따라서 퓨샷 러닝 및 경량 딥러닝에서 제안하는 방법의 적용 가능성을 확인할 수 있었다.

ACKNOWLEDGMENT

본 논문은 국립대학육성사업의 지원을 받아 작성되었습니다.

참 고 문 헌

[1] RAVI, S., and LAROCHELLE, H. “Optimization as a model for few-shot learning,” 2016

[2] Y. J. Lee, Y. H. Moon, J. Y. Park, and O. G. Min, “Recent R&D Trends for Lightweight Deep Learning,” Electronics and Telecommunications Trends, vol. 34, no. 2, pp. 40-50, Apr. 2019.

[3] RUDER, S. “An overview of gradient descent optimization algorithms,” arXiv preprint arXiv:1609.04747, pp. 2-6, 2016