딥러닝 모델 학습을 위한 경사 하강 기반 최적화 알고리즘

장 지아웨이 JIAWEI@IFMLAB.ORG

설립자 겸 이사

정보 융합 및 마이닝 연구실

(최초 버전: 2019년 2월, 개정: 2019년 3월).

초록

이 백서에서는 심층 신경망 모델을 학습하기 위한 경사 하강 기반 연산 알고리즘에 대해 소개합니다. 여러 비선형 투영 레이어를 포함하는 딥러닝 모델은 훈련하기가 매우 어렵습니다. 오늘날 대부분의 딥러닝 모델 훈련은 여전히 역전파 알고리즘에 의존하고 있습니다. 역전파에서는 경사 하강 기반 최적화 알고리즘으로 수렴할 때까지 모델 변수가 반복적으로 업데이트됩니다. 기존의 바닐라 경사 하강 알고리즘 외에도 최근에는 학습 성능을 향상시키기 위해 모멘텀, 아다그라드, 아담, 가담 등 다양한 경사 하강 알고리즘이 제안되고 있으며, 이 논문에서는 각각 소개합니다.

키워드: 그라데이션 하강; 최적화 알고리즘; 딥러닝

콘텐츠

1. 소개

딥러닝에 대한 실제 연구 및 응용 작업에서 딥 모델을 효과적으로 훈련하는 것은 연구자와 실무자 모두에게 여전히 가장 어려운 작업 중 하나입니다. 이러한 맥락에서 지금까지 대부분의 딥 모델 훈련은 출력 레이어의 오류를 역전파하고 경사 하강 기반 최적화 알고리즘으로 변수를 레이어별로 업데이트하는 역전파 알고리즘을 기반으로 하고 있습니다. 경사 하강은 딥러닝 모델 훈련에 중요한 역할을 하며, 최근 몇 년 동안 그 성능을 더욱 향상시키기 위해 새로운 변형 알고리즘이 많이 제안되고 있습니다. 뉴턴의 방법 등과 같은 고차 미분 기반 알고리즘에 비해 경사 하강과 함께 1차 미분 기반의 다양한 변형 알고리즘은
딥 모델에 훨씬 효율적이고 실용적입니다. 이 백서에서는 딥러닝 모델 학습 알고리즘에 대한 포괄적인 소개를 제공합니다.

공식적으로, n개의 쌍이 포함된 훈련 데이터 $T = \{(\mathbf{x}1, \mathbf{y}1), (\mathbf{x}2, \mathbf{y}2), , (\mathbf{x}n, \mathbf{y}n)\}$가 주어 집니다.

의 특징-라벨 벡터로 표현할 수 있으며, 여기서 특징 벡터 $\mathbf{x}_i \in R^d$ x와 라벨 벡터 $\mathbf{y}_i \in R^d$ y는 각각 차원 d_x 및 d_y 입니다. 데이터를 분류하는 데 사용되는 딥 러닝 모델을 매핑으로 표현할 수 있습니다: $F(-; \boldsymbol{\theta}) : X \rightarrow Y$, 여기서 X와 Y는 특징과 레이블 공간을 나타냅니다.$\boldsymbol{\theta}$는 매핑에 관련된 가변 벡터를 나타냅니다. 공식적으로, 벡터 $\mathbf{x}_i \in X$ 로 특징지어지는 입력 데이터 인스턴스가 주어지면 딥 러닝 모델의 출력은 $\hat{y}_i = F(\mathbf{x}_i ; \boldsymbol{\theta})$로 나타낼 수 있습니다. 입력의 실제 레이블 벡터 $\mathbf{y}_i$ 와 비교하면 다음과 같습니다.

예를 들어, 모델에 의해 도입된 오류를 손실 항 $L(\hat{y}_i, \mathbf{y}_i)$로 나타낼 수 있습니다. 이 백서의 나머지 부분에서는 오류와 손실을 구분하지 않고 같은 의미로 사용할 수 있습니다.

도입된 모델 오류를 측정하기 위해 다양한 손실 함수, 즉 $l(-, -)$가 제안되었습니다. 대표적인 예는 다음과 같습니다.

- 평균 제곱 오차(MSE):

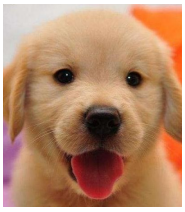
$$\ell_{MSE}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{d_y} \sum_{j=1}^{d_y} (\mathbf{y}_i(j) - \hat{\mathbf{y}}_i(j))^2 .$$

(1)

- 평균 절대 오류(MAE):

$$\ell_{MAE}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{d_y} \sum_{j=1}^{d_y} |\mathbf{y}_i(j) - \hat{\mathbf{y}}_i(j)| .$$

(2)



클래스	인스턴스 1	인스턴스 2	인스턴스 3
강아지	0.49	0.45	0.21
고양이	0.43	0.53	0.15
새	0.08	0.02	0.64

그림 1: 힌지 손실 및 교차 엔트로피 손실 함수를 설명하는 예시.

- 힌지 손실:

$$\ell_{Hinge}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \sum_{j \neq l_{\mathbf{y}_i}} \max(0, \hat{\mathbf{y}}_i(j) - \hat{\mathbf{y}}_i(l_{\mathbf{y}_i}) + 1), \tag{3}$$

여기서 $l_{\mathbf{y}}$ i 는 벡터 $\mathbf{y}_i$ 에 따른 인스턴스의 실제 클래스 레이블 인덱스를 나타냅니다.

- 교차 엔트로피 손실:

$$\ell_{CE}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = -\sum_{j=1}^{d_y} \mathbf{y}_i(j) \log \hat{\mathbf{y}}_i(j). \tag{4}$$

관련된 클래스 수와 모델에 사용되는 활성화 함수에 따라 교차 엔트로피 함수는 시그모이드 교차 엔트로피(출력 계층의 활성화 함수로 시그미오 함수를 사용하는 이진 분류 작업의 경우)

와 소프트맥스 교차 엔트로피(출력 계층의 활성화 함수로 소프트맥스 함수를 사용하는 다중 클래스 분류 작업의 경우)로 추가로 지정할 수 있습니다.

회귀 작업에는 보통 평균 제곱 오차와 평균 절대 오차가 사용되고, 분류 작업에는 보통 힌지 손실과 교차 엔트로피가 대신 사용됩니다. 위에서 소개한 손실 함수 외에도 허버 손실, 코사인 거리 등 다양한 손실 함수가 특정 학습 문제에 사용될 수 있으며, 자세한 내용은 관련 문서를 참고하시기 바랍니다. 힌지 손실과 교차 엔트로피 손실 함수에 대한 자세한 정보를 제공하기 위해 실제 문제에서 이 함수를 사용하는 예를 들어 설명하겠습니다.

예제 1 그림 1에서 볼 수 있듯이 세 개의 입력 이미지(즉, 데이터 인스턴스)를 기반으로 아 래 표와 같이 딥러닝 모델을 통해 예측 레이블을 얻을 수 있습니다. 개 입력 이미지(즉, 인스턴스1)의 경우, 실제 레이블은 $y1 = [1, 0, 0]^T$ 이어야 한다는 것을 알고 있습니다. 예측 레이블은 $y\hat{}1 = [0.49, 0.43, 0.08]^T$ 이며, 벡터의 이 세 항목은 다음과 같습니다.
는 각각 개, 고양이, 새 클래스 레이블에 해당합니다. 마찬가지로 다음과 같이 표현할 수 있습니다. 고양이 이미지와 새 이미지의 실제 레이블과 예측 레이블(즉, 인스턴스 2와 인스턴스 3)을 벡터로 $y2 = [0, 1, 0]^T$, $y\hat{}2 = [0.45, 0.53, 0.02]^T$, $y3 = [0, 0, 1]^T$, and $y\hat{}3 = [0.21, 0.15, 0.64]^T$
3
실제 레이블과 예측 레이블을 기반으로 도입된 손실을 나타낼 수 있습니다.
용어는 다음과 같이 딥러닝 모델에 의해 정의됩니다.

- 인스턴스 1: 인스턴스 1의 경우, 실제 레이블은 개(즉, 레이블 벡터의 항목 1)이고 실제 레이블 인덱스는 $\text{l}y1 = 1$. 이라는 것을 알 수 있습니다. 그러므로 인스턴스 1에 대해 도입된 힌지 손실을 다음과 같이 나타낼 수 있습니다:

$$\begin{aligned}\ell_{Hinge}(\hat{\mathbf{y}}_1, \mathbf{y}_1) &= \sum_{j \neq 1} \max(0, \hat{\mathbf{y}}_1(j) - \hat{\mathbf{y}}_1(1) + 1) \\ &= \max(0, \hat{\mathbf{y}}_1(2) - \hat{\mathbf{y}}_1(1) + 1) + \max(0, \hat{\mathbf{y}}_1(3) - \hat{\mathbf{y}}_1(1) + 1) \\ &= \max(0, 0.43 - 0.49 + 1) + \max(0, 0.08 - 0.49 + 1) \\ &= 1.53.\end{aligned} \tag{5}$$

- 인스턴스 2: 인스턴스 2의 경우, 실제 레이블은 고양이(즉, 레이블 벡터의 항목 2)이고 실제 레이블 인덱스는 $\text{l}\mathbf{y}\ 2 = 2$입니다. 따라서 인스턴스 2에 도입된 힌지 손실은 다음과 같이 표현할 수 있습니다.

$$\begin{aligned}\ell_{Hinge}(\hat{\mathbf{y}}_2, \mathbf{y}_2) &= \sum_{j \neq 2} \max(0, \hat{\mathbf{y}}_2(j) - \hat{\mathbf{y}}_2(2) + 1) \\ &= \max(0, \hat{\mathbf{y}}_2(1) - \hat{\mathbf{y}}_2(2) + 1) + \max(0, \hat{\mathbf{y}}_2(3) - \hat{\mathbf{y}}_2(2) + 1) \\ &= \max(0, 0.45 - 0.53 + 1) + \max(0, 0.02 - 0.53 + 1) \\ &= 1.41.\end{aligned} \tag{6}$$

- 인스턴스 3: 마찬가지로 입력 인스턴스 3의 경우, 실제 레이블 인덱스는 $\mathbf{y}$ 3 = 3 이며 인스턴스에 도입된 힌지 손실은 다음과 같이 나타낼 수 있습니다.

$$\begin{aligned}\ell_{Hinge}(\hat{\mathbf{y}}_3, \mathbf{y}_3) &= \sum_{j \neq 3} \max(0, \hat{\mathbf{y}}_3(j) - \hat{\mathbf{y}}_3(3) + 1) \\ &= \max(0, \hat{\mathbf{y}}_3(1) - \hat{\mathbf{y}}_3(3) + 1) + \max(0, \hat{\mathbf{y}}_2(2) - \hat{\mathbf{y}}_3(3) + 1) \\ &= \max(0, 0.21 - 0.64 + 1) + \max(0, 0.15 - 0.64 + 1) \\ &= 1.08.\end{aligned} \tag{7}$$

한편, 이 세 가지 입력 데이터 인스턴스의 실제 레이블과 예측 레이블 벡터를 기반으로 도입된 교차 엔트로피 손실을 다음과 같이 나타낼 수 있습니다:

- 인스턴스1:

$$\begin{aligned}\ell_{CE}(\hat{\mathbf{y}}_1, \mathbf{y}_1) &= -\sum_{j=1}^{d_y} \mathbf{y}_1(j) \log \hat{\mathbf{y}}_1(j) \\ &= -1 \times \log 0.49 - 0 \times \log 0.43 - 0 \times \log 0.08 \\ &= 0.713.\end{aligned} \tag{8}$$

- 인스턴스2:

$$\begin{aligned}\ell_{CE}(\hat{\mathbf{y}}_2, \mathbf{y}_2) &= -\sum_{j=1}^{d_y} \mathbf{y}_2(j) \log \hat{\mathbf{y}}_2(j) \\ &= -0 \times \log 0.45 - 1 \times \log 0.53 - 0 \times \log 0.02 \\ &= 0.635.\end{aligned} \tag{9}$$

- 인스턴스3:

$$\begin{aligned}\ell_{CE}(\hat{\mathbf{y}}_3, \mathbf{y}_3) &= -\sum_{j=1}^{d_y} \mathbf{y}_3(j) \log \hat{\mathbf{y}}_3(j) \\ &= -0 \times \log 0.21 - 0 \times \log 0.15 - 1 \times \log 0.64 \\ &= 0.446.\end{aligned} \tag{10}$$

또한 학습 세트의 모든 데이터 인스턴스에 대한 도입 손실을 기반으로 딥러닝 모델에 의한 총 도입 손실 기간을 다음과 같이 나타낼 수 있습니다.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}} \ell(F(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i). \tag{11}$$

딥러닝 모델의 학습은 학습 데이터를 피팅함으로써 학습 세트에 도입된 손실을 최 소 화 하 여 최적의 변수, 즉 최적의 **θ** 를달성하는 것을 목표로 합니다. 공식적으로 딥러닝 모델 학습의 목적 함수는 다음과 같이 표현할 수 있습니다:

$$\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta}; \mathcal{T}), \tag{12}$$

여기서 **θ**는 변수 영역을 나타내며, 학습할 변수에 다른 제약이 없는 경우 $\Theta = \mathbb{R}^{d_\theta}$ 는 변수 벡터 **θ**의 차원을 나타내는것을 가질 수 있습니다. 다음 섹션에서는 손실 함수가 볼록 또는 비볼록인 경우 딥러닝 모델에 대한 전역 최적 또는 국소 최적 변수를 학습할 수 있는 경사 하강 기반 최적화 알고리즘 그룹을 소개합니다. 이 백서의 마지막 부분에서 실제 데이터 세트에 대한 몇 가지 실습 실험을 통해 성능에 대한 보다 포괄적인 실험 분석을 제공할 것입니다

2. 기존의 경사 하강 기반 학습 알고리즘

이 섹션에서는 기존의 바닐라 경사 하강, 확률적 경사 하강 및 미니 배치 경사 하강 알고리즘에 대해 소개합니다. 이들 간의 주요 차이점은 각 반복에서 모델 변수를 업데이트하는 데 사용되는 훈련 데이터의 양에 있습니다. 또한 다음 섹션에서 소개할 변형 알고리즘의 기본 알고리즘으로도 사용됩니다.

2.1 바닐라 그라데이션 하강

바닐라 경사 하강은 일괄 경사 하강으로도 잘 알려져 있습니다. 이전 섹션에서 소개한 훈련 세트 T가 주어지면 바닐라 경사 하강 최적화 알고리즘은 다음 방정식을 사용하여 모델 변수 를 반복적으로 최적화합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; \mathcal{T}), \tag{13}$$

여기서 τ ≥ 1은 업데이트 반복을 나타냅니다.

위의 업데이트 방정식에서 $\boldsymbol{\theta}^{(\tau)}, \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; \mathcal{T})$ 및 η 표기의 물리적 의미는 다음과 같이 설명할 수 있습니다:

1. 항 $\boldsymbol{\theta}^{(\tau)}$ 은 반복 τ 을 통해 업데이트된 모델 변수 벡터를 나타냅니다.τ = 0에서 벡터 $\boldsymbol{\theta}^{(0)}$ 는 초기 모델 변수 벡터를 나타내며, 일반적으로 특정 분포(예: 균등 분포 U(a,b) 또는 정규 분포)에 따라 임의로 초기화됩니다. 균일분포 N (μ, σ^2). 더 나은 성능을 얻으려면 하이퍼 파라미터 a,b의 균등 분포 또는 정규 분포의 μ, σ를 미세 조정할 수 있습니다.

2. 항 $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}; \mathcal{T})$는 완전한 훈련 집합 T와 반복 τ - 1에서 얻은 변수 벡터 값$\boldsymbol{\theta}^{(\tau-1)}$ 를 기반으로 변수 **θ에** 대한 손실 함수 L(**θ**; T)의 도함수를 나타냅니다.

3. 용어 η는 그라데이션 하강 알고리즘의 학습 속도를 나타내며, 일반적으로 작은 값의 하 이퍼 파라미터입니다(예: 10^{-3}). 바닐라 하강 알고리즘 그라데이션의 실제 적용에서 빠른 수렴을 달성 하려면 파라미터의 미세 조정이 필요합니다.

이러한 반복적인 업데이트 과정은 수렴까지 계속되며, 수렴 시 얻어진 변수 벡터 θ는 딥러닝 모델에 대한 (전역 또는 로컬) 최적 변수로 출력됩니다. 바닐라 경사 하강 알고리즘의 의사코드는 알고리즘 1에서 확인할 수 있습니다. 알고리즘 1에서 모델 변수는 정규 분포로 초기화됩니다. 정규 분포 표준편차 σ는 입력 파라미터이며, 평균 μ는 0으로 설정됩니다. 수렴 조건은 (1) 두 번의 순차적 반복에서 손실 항의 변화가 미리 지정된 임계값보다 작거나 (2) 모델 변수 θ 변화가 미리 지정된 범위 내에 있거나 (3) 미리 지정된 반복 라운드 수에 도달한 경우일 수 있습니다.

알고리즘 1 바닐라 그라데이션 하강

Require: Training Set: T ; Learning Rate η; Normal Distribution Std: σ.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize convergence tag = F alse

3: **while** tag == False **do**

4: Compute gradient ∇**θ**L(**θ**; T) on the training set T

5: Update variable **θ** = **θ** − η · ∇**θ**L(**θ**; T)

6: **if** convergence condition holds **then**

7: tag = T rue

8: **end if**

9: **end while**

10: **Return** model variable **θ**

설명에 따르면 바닐라 경사 하강 알고리즘에서 모델 변수를 업데이트하려면 각 반복 마다 해당 변수에 대한 손실 함수의 경사, 즉 $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(\tau-1)}; T)$를 계산해야 하는데, 이는 일반적으로 다음과 같은 경우 매우 많은 시간이 소요됩니다.
대규모 훈련 세트가 필요합니다. 학습 효율을 높이기 위해 확률론적 학습을 도입합니다.
경사 하강 및 미니 배치 경사 하강 학습 알고리즘을 다음 두 하위 섹션에서 설명합니다.

2.2 확률적 그라데이션 하강

바닐라 경사 하강은 완전한 훈련 집합으로 손실 함수 경사를 계산하므로 훈련 집합에 많은 데이터 인스턴스가 포함된 경우 매우 비효율적입니다. 확률적 경사 하강(SGD) 알고리즘은 전체 훈련 집합을 사용하는 대신 손실 함수 경사 인스턴스를 인스턴스별로 계산하여 모델 변수를 업데이트합니다. 따라서 확률적 경사 하강은 바닐라 경사 하강보다 훨씬 빠를 수 있지만 업데이트 과정에서 손실 함수의 변동이 심할 수 있습니다.

공식적으로, 훈련 세트 T와 초기화된 모델 변수 벡터 $\boldsymbol{\theta}^{(0)}$ 가 주어지면 각 인스턴스$(\mathbf{x}_i, \mathbf{y}_i) \in T$에 대해 확률적 경사 하강 알고리즘은 다음 방정식을 사용하여 모델 변수를 반복적으로 업데이트합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(\tau-1)}; (\mathbf{x}_i, \mathbf{y}_i)), \tag{14}$$

여기서 손실 항 $L(\boldsymbol{\theta}; (\mathbf{x}_i, \mathbf{y}_i)) = l(F(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i)$는 데이터 인스턴스$(\mathbf{x}_i, \mathbf{y}_i)$에 대한 모델에 의해 도입된 손실을 나타냅니다. 확률적 경사 하강 학습 알고리즘의 의사 코드는 알고리 즘 2에서 확인할 수 있습니다.

알고리즘 2 확률적 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std: σ.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize convergence tag = False

3: **while** tag == False **do**

4: Shuffle the training set T

5: **for** each data instance (xi, yi) ∈ T **do**

6: Compute gradient ∇θL(θ; (xi, yi)) on the training instance (xi, yi)

7: Update variable **θ** = **θ** − η · ∇**θ**L(**θ**; (xi, yi))

8: **end for**

9: **if** convergence condition holds **then**

10: tag = True

11: **end if**

12: **end while**

13: **Return** model variable **θ**

알고리즘 2에 따르면 전체 훈련 세트는 업데이트 프로세스 전에 셔플됩니다. 확률적 경사 하강의 학습 과정에서 많은 변동이 있을 수 있지만, 실제로 확률적 경사 하강은 국부 최적에서 벗어날 수 있는 기능을 제공합니다. 한편, 학습 과정에서 작은 학습률 η를 선택하고 이를 낮추면 확률적 경사 하강은 거의 확실하게 로컬 또는 글로벌 최적에 수렴할 수 있습니다.

2.3 미니 배치 그라데이션 하강

바닐라 경사 하강과 확률적 경사 하강 학습 알고리즘 사이의 균형을 맞추기 위해 미니 배치 경사 하강은 대신 미니 배치의 훈련 인스턴스로 모델 변수를 업데이트할 것을 제안합니다. 공식적으로 B ⊂ T는 훈련의 미니 배치를 나타냅니다.
인스턴스를 훈련 세트 T 에서 샘플링합니다. 변수 업데이트 방정식은 다음과 같이 나타낼 수 있습니다. 다음과 같이 미니 배치로 딥러닝 모델에 적용합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta - \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(\tau-1)}; B), \quad (15)$$

여기서 L(**θ**; B)는 미니 배치 B에서 모델이 도입한 손실 항을 나타냅니다. 미니 배치 경사 하 강 알고리즘의 의사 코드는 알고리즘 3에 설명되어 있습니다.

알고리즘 3 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)
2: Initialize convergence tag = False
3: **while** tag == False **do**
4: Shuffle the training set T
5: **for** each data instance (xi , yi) ∈ T **do**
6: Compute gradient ∇**θ**L(**θ**; B) on the mini-batch **B**
7: Update variable **θ** = **θ** − η · ∇**θ**L(**θ**; (xi , yi))
8: end for
9: if convergence condition holds then
10: tag = T rue
11: end if
12: end while
13: Return model variable θ

미니 배치 경사 하강 알고리즘에서는 배치 크기 b가 매개변수로 제공되며 64, 128 또는 256과 같은 값을 사용할 수 있습니다. 대부분의 경우 b는 매우 크지 않아야 하며 특정 값은 훈 련 세트의 크기에 따라 크게 달라집니다. 미니 배치는 일반적으로 훈련 세트 T 에서 순차적으로 샘플링됩니다. 즉, 훈련 세트 T를 여러 개의 사이즈 B 배치로 나눌 수 있으며, 이러한 배치는 하나씩 선택할 수 있습니다. 배치를 사용하여 모델 변수를 순차적으로 업데이트할 수 있습니다. 미니 배치 경사 하강의 일부버전에서는 순차적 배치 선택 대신 훈련 세트에서 미니 배치를 무작위로 샘플링할 것을 제안하며, 이를 무작위 미니 배치 생성 프로세스라고도 합니다.

바닐라 경사 하강에 비해 미니 배치 경사 하강 알고리즘은 특히 매우 큰 크기의 훈련 세트에 훨씬 더 효율적입니다. 한편, 확률적 경사 하강에 비해 미니 배치 경사 하강 알고리즘은 모델 변수 업데이트 과정의 편차를 크게 줄이고 훨씬 더 안정적인 수렴을 달성할 수 있습니다.

2.4 바닐라 GD, SGD 및 미니 배치 GD의 성능 분석

여기에 소개된 세 가지 경사 하강 알고리즘은 많은 최적화 문제에 잘 작동하며, 모두 유망한 (로컬 또는 전역) 최적값으로 수렴할 수 있습니다. 그러나 이러한 알고리즘에는 다음과 같은 몇 가지 문제가 있습니다:

- 학습 속도 선택: 학습률 η는 경사 하강 알고리즘의 수렴에 많은 영향을 미칠 수 있습니다. 학습률이 크면 학습 과정이 달라질 수 있습니다, 학습 속도가 작으면 수렴이 너무 느려집니다. 따라서 경사 하강 알고리즘에서는 좋은 학습 속도를 선택하는 것이 매우 중요합니다.

- 학습 속도 조정 : 대부분의 경우 전체 업데이트 과정에서 고정된 학습 속도로는 경사 하 강 알고리즘이 제대로 작동하지 않습니다. 초기에는 단계에서는 알고리즘이 좋은(로컬 또는 글로벌) 최적 속도에 도달하기 위해 더 큰 학습 속도가 필요할 수 있습니다. 그러나 이후 단계에서는 알고리즘이 성능을 미세 조정하기 위해 학습 속도를 더 작은 값으로 조정해야 할 수도 있습니다.

- 가변 개별 학습률: 다른 변수의 경우 업데이트 과정에서 실제로 다른 학습률이 필요할 수 있습니다. 따라서 다양한 변수에 대해 개별 학습률을 사용하는 것이 필요하고 필수적입니다.

- 안장점 피하기 : 공식적으로 새들 포인트는 모든 치수에서 기울기가 0인 점을 나타 냅니다. 그러나 일부 치수의 경우 새들 포인트는 로컬 최소값으로 설정하는 반면, 다른 치수의 경우에는 로컬 최대값으로 설정합니다. 이러한 안장 지점에서 벗어나는 방법은 매우 어려운 문제입니다.

이 백서의 다음 부분에서는 위의 문제 중 하나 또는 여러 가지를 해결하기 위해 주로 제안되는 몇 가지 다른 경 사하강 기반 학습 알고리즘 변형을 소개합니다.

3. 모멘텀 기반 학습 알고리즘

이 섹션에서는 운동량 기반 경사 하강 알고리즘으로 명명된 현재 경사뿐만 아니라 그의 경사도 동시에 모델 변수를 업데이트할 것을 제안하는 경사 하강 변형 알고리즘 그룹을 소개합니다. 여기서는 미니 배치 GD를 기본 학습 알고리즘으로 사용하여 이러한 알고리즘에 대해 설명합니다.

3.1 모멘텀

경사 하강 알고리즘(예: 미니 배치 경사 하강)의 학습 과정에서 발생하는 변동을 완화하기 위해, 변수의 업데이트 수렴을 가속화하는 모멘텀[6]이 제안되었습니다. 공식적으로 모멘텀은 다음 방정식을 사용하여 변수를 업데이트합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \Delta\mathbf{v}^{(\tau)} \text{ where } \Delta\mathbf{v}^{(\tau)} = \rho \cdot \Delta\mathbf{v}^{(\tau-1)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}), \quad (16)$$

위의 방정식에서 $\mathbf{v}^{(\tau)}$ 는 반복 τ까지 과거 기울기를 기록하기 위해 도입된 모멘텀 항을 나 타내고 함수 L(**θ**) = L(**θ**; B)는 미니 배치 B의 손실 함수를 나타냅니다. 파라미터 ρ ∈ [0,1]은 모멘텀 항의 가중치를 나타냅니다. 그리고 모멘텀 기반 경사 하강 학습 알고리즘의 의사 코드는 다음에서 사용할 수 있습니다.

알고리즘 4 모멘텀 기반 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Momentum Term Weight: ρ.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)
2: Initialize Momentum term Δv = 0
3: Initialize convergence tag = False
4: while tag == False do
5: Shuffle the training set T
6: **for** each mini-batch B ⊂ T **do**
7: Compute gradient ∇**θ**L(**θ**; B) on the mini-batch B
8: Update term Δv = ρ · Δv + (1 − ρ) · ∇**θ**L(**θ**; B)
9: Update variable **θ** = **θ** − η · Δv
10: **end for**
11: **if** convergence condition holds then
12: tag = T rue
13: **end if**
14: **end while**

15: **Return** model variable **θ**

이러한 벡터 $\boldsymbol{\theta}^{(\tau)}$ 의 재귀적 업데이트 방정식을 기반으로 다음 정리에 따라 손실 함수의 기울기 항만으로 실제로 $\boldsymbol{\theta}^{(\tau)}$ 의 동등한 표현을 얻을 수 있습니다.

정리 1 벡터 $\Delta\mathbf{v}^{(\tau)}$ 는 공식적으로 다음 방정식으로 표현할 수 있습니다:

$$\Delta \mathbf{v}^{(\tau)} = \sum_{t=0}^{\tau-1} \rho^t \cdot (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1-t)}). \tag{17}$$

■

증명 τ 에 대한 귀납법으로 정리를 증명할 수 있습니다.

1. 기본 사례: τ = 1인 경우, **θ의** 재귀적 표현에 따르면 다음과 같습니다.

$$\begin{aligned} \Delta \mathbf{v}^{(1)} &= \rho \cdot \Delta \mathbf{v}^{(0)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(0)}) \\ &= (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(0)}), \end{aligned} \tag{18}$$

여기서 $\Delta\mathbf{v}^{(0)}$ 는 0 벡터로 초기화됩니다.

가정: 방정식이 τ = k에 대해 유지된다고 가정합니다. 즉,

$$\Delta \mathbf{v}^{(k)} = \sum_{t=0}^{k-1} \rho^t \cdot (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k-1-t)}). \tag{19}$$

3. 유도: τ = k +1의 경우, 방정식 (16)에 따라 벡터 $\Delta\mathbf{v}^{(k+1)}$ 를 다음과 같이 나타낼 수 있습니다.

$$
\begin{aligned}
\Delta \mathbf{v}^{(k+1)} &= \rho \cdot \Delta \mathbf{v}^{(k)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k)}) \\
&= \rho \cdot \sum_{t=0}^{k-1} \rho^t \cdot (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k-1-t)}) + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k)}) \\
&= \sum_{t=1}^{k} \rho^t \cdot (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k-t)}) + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k)}) \\
&= \sum_{t=0}^{k} \rho^t \cdot (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k-t)}),
\end{aligned}
\tag{20}
$$

를 사용하여 이 증명을 마무리할 수 있습니다.

매개변수 ρ ∈ [0, 1]을 고려하면 현재 반복에서 멀리 떨어진 반복의 그라데이션은 기하급 수적으로 감쇠합니다. 기존 그라데이션과 비교하면 엔트 하강 알고리즘을 사용하면 모멘텀이 더 빠른 수렴 속도를 달성할 수 있으며, 이는 위 방정식에서 업데이트된 기울기를 기반으로 설명할 수 있습니다. 그라데이션 하강의 경우 알고리즘에서 반복 τ에서 업데이트된 기울기는$\boldsymbol{\theta}$ ($\boldsymbol{\theta}^{(\tau-1)}$)로 표시할 수 있으며, 모멘텀 의 업데이트된 기울기는 $\Delta\mathbf{v}^{(\tau)}$ = (1 ρ)$\boldsymbol{\theta}$ ($\boldsymbol{\theta}^{(\tau-1)}$)+ρ $\Delta\mathbf{v}^{(\tau-1)}$ 로 표시할 수 있습니다. 용어 $\Delta\mathbf{v}^{(\tau-1)}$ 는 과거 기울기의 기록을 유지합니다. 경사 하강 알고리즘이 가파른 영역에서 상 대적으로 평평한 영역으로 변수를 업데이트할 때 $\Delta\mathbf{v}^{(\tau-1)}$ 에 저장된 큰 과거 경사 항은 제로 알고리즘이 (로컬 또는 글로벌) 최적값에 도달하도록 가속화하는 역할을 합니다 .

3.2 네스테로프 가속 그라데이션

지금까지 소개한 경사 하강 알고리즘의 경우 모두 학습 과정에서 도달할 미래 지점에 대한 지식 없이 과거 또는 현재 지점의 경사도를 기반으로 변수를 업데이트합니다. 따라서 학습 과정이 맹목적이고 학습 성능을 예측할 수 없습니다.

네스테로프 가속 경사(Nesterov Accelerated Gradient, 즉 NAG)[5] 방법은 대신 근사 미래시점의 경사로 변수를 업데이트하여 이 문제를 해결할 것을 제안합니다. 단계 τ에서 달성할 변수 **θ는 θ**$^{(\tau)}$ 로 표시할 수 있으며, 다음과 같이 추정할 수 있습니다.

$\theta\hat{}(\tau) = \boldsymbol{\theta}^{(\tau-1)} - \eta - \rho - \Delta\mathbf{v}^{(\tau-1)}$ 로, 여기서 $\Delta\mathbf{v}^{(\tau-1)}$ 는 반복 τ -1에서운동량 항을 나타냅니 다. 룩어헤드 경사 및 운동량의 도움으로 NAG의 변수 업데이트 방정식은 공식적으로 다 음과 같이 표현할 수 있습니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \Delta \mathbf{v}^{(\tau)}, \text{ where } \begin{cases} \hat{\boldsymbol{\theta}}^{(\tau)} & = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \rho \cdot \Delta \mathbf{v}^{(\tau-1)}, \\ \Delta \mathbf{v}^{(\tau)} & = \rho \cdot \Delta \mathbf{v}^{(\tau-1)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\hat{\boldsymbol{\theta}}^{(\tau)}). \end{cases} \quad (21)$$

~~가울거를 한 단계 앞으로 계산함으로써 NAG 알고리즘은 모델 변수를 훨씬 더 효과적으~~로 업데이트할 수 있습니다. NAG 학습 알고리즘의 의사 코드는 알고리즘 5에서 확인할 수 있습니다.

알고리즘 5 NAG 기반 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Momentum Term Weight: ρ.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize Momentum term Δv = 0

3: Initialize convergence tag = False

4: **while** tag == F alse **do**

5: Shuffle the training set T

6: **for** each mini-batch B ⊂ T **do**

7: Compute **θ**ˆ = **θ** − η · ρ · Δv

8: Compute gradient ∇θL(θˆ; B) on the mini-batch B

9: Update term Δv = ρ · Δv + (1 − ρ) · ∇**θ**L(**θ**ˆ; B)

10: Update variable **θ** = **θ** − η · Δv

11: **end for**

12: **if** convergence condition holds **then**

13: tag = T rue

14: **end if**

15: **end while**

16: **Return** model variable **θ**

4. 적응형 그라데이션 기반 학습 알고리즘

이 섹션에서는 적응형 학습 속도로 변수를 업데이트하는 학습 알고리즘 그룹을 소개합니다. 여기서 소개하는 알고리즘에는 각각 Adagrad, RMSprop 및 Adadelta가 포함됩니다.

4.1 Adagrad

앞서 소개한 학습 알고리즘의 경우 학습 속도는 벡터 **θ의** 모든 변수에 대해 대부분 고정되어 있고 동일하지만, 변수 업데이트 과정에서는 변수마다 필요한 학습 속도가 다를 수 있습니다. 최적값에 도달하는 변수의 경우 더 작은 학습 속도가 필요하고, 최적값에서 멀리 떨어진 변수의 경우 최적값에 더 빨리 도달하기 위해 상대적으로 더 큰 학습 속도를 사용해야 할 수 있습니다. 이 두 가지 문제를 해결하기 위해 이 부분에서는 적응형 그라디언트 방법 중 하나인 Adagrad [3]를 소개합니다.

공식적으로 Adagrad의 학습 방정식은 다음과 같이 표현할 수 있습니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{diag(\mathbf{G}^{(\tau)}) + \epsilon \cdot \mathbf{I}}}\mathbf{g}^{(\tau-1)} \text{ where } \begin{cases} \mathbf{g}^{(\tau-1)} & = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}), \\ \mathbf{G}^{(\tau)} & = \sum_{t=0}^{\tau-1} \mathbf{g}^{(t)}(\mathbf{g}^{(t)})^{\top}. \end{cases} \quad (22)$$

위의 업데이트 방정식에서 연산자 diag(**G)**는 **G와** 동일한 차원의 대각선 행렬을 **G의** 대각선에 있는 요소로만 정의하며, ϵ는 행렬 대각선에서 0값이 나뉘는 것을 방지하기 위한 평활화 용어입니다. 행렬 $\mathbf{G}^{(\tau)}$ 는 처음부터 현재 반복까지 계산된 과거 기울기의 기록을 유지합니다. 행렬 $\mathbf{G}^{(\tau)}$ 의 대각선 값이 달라진다는 점을 고려하면 학습률

벡터 **θ의** 변수는 다를 수 있습니다.

$\eta_i^{(\tau)} = \frac{\eta}{\sqrt{\mathbf{G}^{(\tau)}(i,i)+\epsilon}}$ **θ**(i)

예를들어 의 반복에서 변수의 경우

τ . 또한 각 반복마다 행렬 $\mathbf{G}^{(\tau)}$ 가 업데이트되기 때문에 같은 변수에 대한 학습률도 각 반복마다 달라지게 되는데, 이것이 이 알고리즘을 적응형 학습 알고리즘이라고 부르는 이유입니다. Adagrad의 의사 코드는 알고리즘 6에 나와 있습니다.

알고리즘 6 아다그라드 기반 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize Matrix G = 0

3: Initialize convergence tag = F alse

4: while tag == F alse do

5: Shuffle the training set T

6: for each mini-batch B ⊂ T do

7: Compute gradient vector g = ∇θL(θ; B) on the mini-batch B

8: Update matrix $\mathbf{G} = \mathbf{G} + \widetilde{\mathbf{g}}\mathbf{g}^{\top}$

9: Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{diag(\mathbf{G}) + \epsilon \cdot \mathbf{I}}}\mathbf{g}$

10: **end for**

11: **if** convergence condition holds then

12: tag = T rue

13: **end if**

14: **end while**

15: **Return** model variable **θ**

아다그라드의 이러한 학습 메커니즘은 수동 튜닝 없이도 학습 과정에서 적응형 학습률을 적용할 수 있을 뿐만 아니라 변수마다 다른 학습률을 적용할 수 있습니다. 또한 반복이 계속 될수록 행렬 **G**의 대각선 값은 감소하지 않으므로 학습 과정에서 변수의 학습률이 계속 감소하게 됩니다. 또한 변수가 더 이상 학습 데이터의 정보로 효과적으로 업데이트될 수 없기 때문에 이후 반복에서 학습에 문제가 발생할 수 있습니다. 또한 Adagrad는 학습 프로세스를 시작하기 위해 초기 학습 속도 매개 변수 η가 여전히 필요하며, 이는 Adagrad의 단점 중 하나로 취급 될 수 있습니다.

4.2 RMSprop

아다그라드에서 학습률이 단조롭게 감소하는 문제를 해결하려면 (즉, $\frac{\eta}{\sqrt{diag(\mathbf{G}^{(\tau)})+\epsilon\cdot\mathbf{I}}}$ 항목이 매트릭스 이 부분에서는 또 다른 학습 알고리즘을 소개합니다. RMSprop [9]. RMSprop은 과거 누적 그라데이션의 가중치를 적절히 감쇠시킵니다. RMSprop을 Adagrad에 정의된 행렬 **G로 대체할 수 있으며,** 업데이트 과정에서 학습 속도를 조정할 수 있습니다. 공식적으로RMSprop의 변수 업데이트 방정식은 다음 방정식으로 표현할 수 있습니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{diag(\mathbf{G}^{(\tau)}) + \epsilon \cdot \mathbf{I}}}\mathbf{g}^{(\tau-1)}, \tag{23}$$

$$\begin{cases} \mathbf{g}^{(\tau-1)} & = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}), \\ \mathbf{G}^{(\tau)} & = \rho \cdot \mathbf{G}^{(\tau-1)} + (1-\rho)\cdot \mathbf{g}^{(\tau-1)}(\mathbf{g}^{(\tau-1)})^{\top}. \end{cases} \tag{24}$$

위 방정식에서 분모 항은 일반적으로 RMS($\mathbf{g}^{(\tau-1)}$) = 다음과 같이 표시됩니다.

$\sqrt{diag(\mathbf{G}^{(\tau)}) + \epsilon \cdot \mathbf{I}}$ (RMS는 벡터 $\mathbf{g}^{(\tau-1)}$ 의 평균제곱근을 나타냅니다.) 따라서 업데이트 방정식도 일반적으로 다음과 같이 작성됩니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{RMS(\mathbf{g}^{(\tau-1)})}\mathbf{g}^{(\tau-1)}. \tag{25}$$

행렬 **G의** 표현에서 매개변수 ρ는 역사적으로 계산된 기울기의 가중치를 나타내며, Geoff Hinton의 Coursera 강의에 따르면 ρ는 일반적으로 0.9로 설정됩니다. **G**의 표현에 따라 으로의 반복 횟수에서 계산된 그라데이션에 대해 다음과 같이 계산합니다. 현재 반복의 매우 작은 가중치, 즉 $\rho^{(t)}$ - (1 - ρ)가 할당되며, 이는 t에 따라 기하급수적으로 감 소합니다. RMSprop의 의사 코드는 알고리즘 7에 제공됩니다,

여기서 학습률 η는 여전히 필요하며 알고리즘의 파라미터로 제공됩니다.

알고리즘 7 RMSprop 기반 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Parameter ρ.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize Matrix **G** = 0

3: Initialize convergence tag = False

4: **while** tag == False **do**

5: Shuffle the training set T

6: **for** each mini-batch B ⊂ T d**o**

7: Compute gradient vector g = ∇**θ**L(**θ**; B) on the mini-batch B

8: Update matrix $\mathbf{G} = \rho \cdot \mathbf{G} + (1-\rho) \cdot \mathbf{g}\mathbf{g}^{\top}$

9: Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{diag(\mathbf{G}) + \epsilon \cdot \mathbf{1}}}\mathbf{g}$

10: **end for**

11: **if** convergence condition holds **then**

12: tag = True

13: **end if**

14: **end while**

15: **Return** model variable **θ**

4.3 Adadelta

아다그라드에서 단조롭게 감소하는 학습률을 동일한 방법으로 해결하는 Adadelta [10]와 RMSprop은 거의 동시에 다른 사람들에 의해 개별적으로 제안되었습니다. 한편, RMSprop은 변수의 업데이트 방정식에서 학습률을 제거하는 메커니즘을 도입하여 Adagrad를 더욱 개선했습니다. 앞서 RMSprop에서 소개한 다음 업데이트 방정식을 기반으로 합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{RMS(\mathbf{g}^{(\tau-1)})}\mathbf{g}^{(\tau-1)}. \tag{26}$$

그러나 [10]에서 소개한 바와 같이 지금까지 소개된 학습 알고리즘은 업데이트 과정에서 학습 변수의 단위를 고려하지 못합니다. 여기서 단위는 변수의 물리적 의미를 효과적으로 나타내며, "km", "s" 또는 "kg" 등이 될 수 있습니다. 만약 매개변수 **θ**에 가상 단위가 있는 경우 매개변수의 업데이트, 즉 **Δθ** = ∇**θ** L(**θ**)가 됩니다 이것또한 동일한 단위를 가져야 합니다. 하지만 앞서 소개한 학습 알고리즘인 SGD, 모멘텀, NAD, 아다그라드의 경우 이러한 가정을 적용할 수 없습니다. 예를 들어 SGD의 경우 업데이트 기간의 단위는 실제로 **θ의** 단위의 역수에 비례합니다:

$$\text{units of } \Delta\boldsymbol{\theta} \propto \text{units of } \mathbf{g} \propto \text{units of } \frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} \propto \frac{1}{\text{units of } \boldsymbol{\theta}}. \tag{27}$$

이러한 문제를 해결하기 위해 Adadelta는 헤시안 근사를 사용하는 뉴턴의 방법과 같은 2차 방법을 살펴볼 것을 제안합니다. 뉴턴의 방법에서 업데이트되는 변수의 단위는 다음과 같이 나타낼 수 있습니다.

$$\text{units of } \Delta\boldsymbol{\theta} \propto \text{units of } \mathbf{H}^{-1}\mathbf{g} \propto \text{units of } \frac{\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}}}{\frac{\partial^2 \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}^2}} \propto \text{units of } \boldsymbol{\theta}, \tag{28}$$

여기서 $\mathbf{H} = \frac{\partial^2 \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}$ 는 손실의 미분으로 계산된 헤시안 행렬을 나타냅니다.

수에 변수를 추가할 수 있습니다.

단위를 일치시키기 위해 Adadelta는 뉴턴의 방법 업데이트 용어를 다음과 같이 재정렬합니다.

$$\Delta\boldsymbol{\theta} = -\frac{\frac{\partial\mathcal{L}(\cdot)}{\partial\boldsymbol{\theta}}}{\frac{\partial^2\mathcal{L}(\cdot)}{\partial\boldsymbol{\theta}^2}}, \tag{29}$$

이를 통해 다음과 같으 결론을 도출할 수 있습니다.

$$\mathbf{H}^{-1} = -\frac{1}{\frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}^2}} = -\frac{\Delta\boldsymbol{\theta}}{\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}}. \tag{30}$$

따라서 뉴턴의 방법에 따라 변수에 대한 업데이트 방정식을 다음과 같이 다시 작성할 수 있습니다:

$$\begin{aligned}\boldsymbol{\theta}^{(\tau)} &= \boldsymbol{\theta}^{(\tau-1)} - (\mathbf{H}^{(\tau)})^{-1}\mathbf{g}^{(\tau-1)} \\ &= \boldsymbol{\theta}^{(\tau-1)} - \frac{\Delta\boldsymbol{\theta}^{(\tau)}}{\frac{\partial\mathcal{L}(\boldsymbol{\theta}^{(\tau)})}{\partial\boldsymbol{\theta}}}\mathbf{g}^{(\tau-1)}.\end{aligned} \tag{31}$$

RMSprop과 마찬가지로 Adadelta는 위 방정식의 분모를 이전 그라데이션의 RMS로 근사화합니다. 한편, 현재 반복의 $\Delta\boldsymbol{\theta}^{(\tau)}$ 항은 아직 알려지지 않았습니다. Adadelta는 위 방정식의 분자를 근사화할 것을 제안하며, $\Delta\boldsymbol{\theta}$**를** 대신하여 RMS($\Delta\boldsymbol{\theta}$)**를** 사용하는 유사한 방식을 채택합니다. Adadelta의 변수 업데이트 방정식은 공식적으로 다음과 같이 작성할 수 있습니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{RMS(\Delta\boldsymbol{\theta}^{(\tau-1)})}{RMS(\mathbf{g}^{(\tau-1)})}\mathbf{g}^{(\tau-1)}, \tag{32}$$

여기서 RMS($\Delta\boldsymbol{\theta}^{(\tau-1)}$)는 이전 반복 τ - 1까지 이전 반복의 ∆θ에 대한 기록을 유지합니다. Adadelta 알고리즘의 의사 코드는 알고리즘 8에 나와 있습니다. 알고리즘 설명에 따르면 Adadelta는 업데이트에 학습 속도를 사용하지 않습니다. 방정식을 사용하여 앞서 소개한 아다그라드의 두 가지 약점을 효과적으로 해결했습니다.

알고리즘 8 아델타 기반 미니 배치 그라데이션 하강

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Parameter ρ.

Ensure: Model Parameter θ

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize Matrix **G** = **0**

3: Initialize Matrix **Θ** = **0**

4: Initialize convergence tag = False

5: **while** tag == False **do**

6: Shuffle the training set T

7: **for** each mini-batch B ⊂ T do

8: Compute gradient vector g = ∇θL(θ; B) on the mini-batch B

9: Update matrix $\mathbf{G} = \rho \cdot \mathbf{G} + (1-\rho) \cdot \mathbf{g}\mathbf{g}^{\top}$

10: Computer updating vector $\Delta\boldsymbol{\theta} = -\frac{\sqrt{diag(\boldsymbol{\Theta})+\epsilon \cdot \mathbf{I}}}{\sqrt{diag(\mathbf{G})+\epsilon \cdot \mathbf{I}}}\mathbf{g}$

11: Update matrix **Θ** = ρ · **Θ** + (1 − ρ) · ∆**θ**(∆**θ**)T

12: Update variable **θ** = **θ** + ∆**θ**

13: **end for**

14: **if** convergence condition holds **then**

15: tag = True

16: **end if**

17: **end while**

18: **Return** model variable **θ**

5. 모멘텀 및 적응형 그라데이션 기반 학습 알고리즘

이 섹션에서는 모멘텀 알고리즘과 적응형 학습 속도를 가진 알고리즘의 장점을 결합한 학습 알고리즘인 아담과 나담을 각각 소개합니다.

5.1 Adam

최근에는 메모리 요구량이 거의 없이 1차 기울기만 계산하는 새로운 학습 알고리즘인 적응형 모멘트 추정(Adam)[4]이 도입되었습니다. RMSprop 및 Adadelta와 마찬가지로 Adam은 과거 제곱된 1차 기울기의 기록을 유지하며, Adam은 과거 1차 기울기의 기록도 유지하는데, 이 두 기록은 학습 과정에서 기하급수적으로 감소합니다. 공식적으로 벡터 $\mathbf{m}^{(\tau)}$ 과 $\mathbf{v}^{(\tau)}$ 를 사용하여 각각 1차 기울기와 제곱된 1차 기울기를 저장하는 항을 나타낼 수 있으며, 그 구체적인 표현은 다음과 같습니다:

$$\begin{aligned}\mathbf{m}^{(\tau)} &= \beta_1 \cdot \mathbf{m}^{(\tau-1)} + (1-\beta_1)\cdot \mathbf{g}^{(\tau-1)},\\ \mathbf{v}^{(\tau)} &= \beta_2 \cdot \mathbf{v}^{(\tau-1)} + (1-\beta_2)\cdot \mathbf{g}^{(\tau-1)} \odot \mathbf{g}^{(\tau-1)},\end{aligned} \tag{33}$$

여기서 벡터 $\mathbf{g}^{(\tau-1)} = \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^{(\tau-1)})$와 $\mathbf{g}^{(\tau-1)} \odot \mathbf{g}^{(\tau-1)}$는 벡터의 원소별 곱을 나타냅니다 . 위의 방정식에서 벡터 $\mathbf{v}^{(\tau)}$ 는 실제로 동일한 정보를 저장합니다.

방정식 (24)에 사용된 diag($\mathbf{G}^{(\tau)}$)와 같습니다. 그러나 아담은 전체 행렬을 저장하는 대신 이러한 벡터 기록을 유지함으로써 공간을 덜 사용하는 경향이 있습니다. [4]에서 소개한 바와 같이, 벡터 $\mathbf{m}^{(\tau)}$ 과 $\mathbf{v}^{(\tau)}$ 는 특히 β_1 와 β_2 는 각각 0 벡터로 초기화되므로 1에 가깝습니다.$^{(0)}$ 와 $\mathbf{v}^{(0)}$ 는 각각 0 벡터로 초기화됩니다.

이러한 문제를 해결하기 위해 아담은 다음과 같이 용어의 규모를 재조정하는 방법을 소개합니다:

$$\begin{aligned}\hat{\mathbf{m}}^{(\tau)} &= \frac{\mathbf{m}^{(\tau)}}{1-\beta_1^{\tau}},\\ \hat{\mathbf{v}}^{(\tau)} &= \frac{\mathbf{v}^{(\tau)}}{1-\beta_2^{\tau}},\end{aligned} \tag{34}$$

여기서 β^{τ} 및 β^{τ} 항의 위 첨자 τ는 반복 대신 거듭제곱을 나타냅니다.

카운트 인덱스(τ)를 사용합니다. 재조정된 벡터 $\hat{m}(\tau)$ 과 행렬 $\hat{v}(\tau)$ 을 기반으로 Adam 은 다음 방정식을 사용하여 모델 변수를 업데이트합니다:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau)}} + \epsilon} \odot \hat{\mathbf{m}}^{(\tau)} \tag{35}$$

위의 설명에 따르면, Adam은 더 빠른 수렴과 적응 학습 속도를 동시에 가능하게 하는 모멘텀 알고리즘에 RMSprop 알고리즘을 통합한 것으로 볼 수 있습니다. 공식적으로 아담 학습 알고리즘의 의사 코드는 알고리즘 9에 β_1 와 β_2 이 알고리즘의 파라미터로 입력되어 있습니다. [4]에 따르면 아담의 파라미터는 다음과 같이 초기화할 수 있니다: $\epsilon = 10^{-8}$, $\beta_1 = 0.9$ 및 $\beta_2 = 0.999$.

알고리즘9 아담

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Decay

Parameters β1, β2.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize vector **m** = **0**

3: Initialize vector **v** = **0**

4: Initialize step τ = 0

5: Initialize convergence tag = False

6: **while** tag == False **do**

7: Shuffle the training set T

8: **for** each mini-batch B ⊂ T **do**

9: Update step τ = τ + 1

10: Compute gradient vector g = ∇**θ**L(**θ**; B) on the mini-batch B

11: Update vector **m** = β1 · **m** + (1 − β1) · g

12: Update vector **v** = β2 · **v** + (1 − β2) · g

13: Rescale vector $\hat{\mathbf{m}} = \mathbf{m}/(1-\beta_1^\tau)$

14: Rescale vector $\hat{\mathbf{v}} = \mathbf{v}/(1-\beta_2^\tau)$

15: Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}}+\epsilon} \odot \hat{\mathbf{m}}$

16: **end for**

17: **if** convergence condition holds **then**

18: tag = True

19: **end if**

20: **end while**

21: **Return** model variable **θ**

5.2 나담

앞서 소개한 아담은 모멘텀 기반 학습 알고리즘과 적응형 기울기 기반 학습 알고리즘이 통합된 것으로 볼 수 있으며, 여기서 바닐라 모멘텀이 채택되었습니다. 2]에서는 바닐라 모멘텀 대신 네스테로프 가속 기울기(Nesterov's accelerated gradient, NAG)로 대체하는 학습 알고리즘이 도입되었으며, 새로운 학습 알고리즘을 네스테로프 가속 아담(Nadam)이라고 부릅니다. Nadam의 업데이트 방정식을 설명하기 전에 NAG의 업데이트 방정식을 다음과 같이 시 설명하고자 합니다(방정식(21)):

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \Delta\mathbf{v}^{(\tau)}, \text{ where } \begin{cases} \Delta\mathbf{v}^{(\tau)} & = \rho \cdot \Delta\mathbf{v}^{(\tau-1)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\hat{\boldsymbol{\theta}}^{(\tau)}), \\ \hat{\boldsymbol{\theta}}^{(\tau)} & = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \rho \cdot \Delta\mathbf{v}^{(\tau-1)}. \end{cases} \tag{36}$$

위의 업데이트 방정식에 따르면, 운동량 항 $\Delta\mathbf{v}^{(\tau-1)}$ 은 이 과정에서 두 번 사용됩니다: (1) $\Delta\mathbf{v}^{(\tau-1)}$ 은 $\hat{\theta}^{(\tau)}$ 계산에 사용되며, (2) $\Delta\mathbf{v}^{(\tau-1)}$ 은 다음과 같은 계산에 사용됩니다. $\Delta\mathbf{v}^{(\tau)}$. Nadam은 다음과 같은 업데이트 방정식을 사용하여 위의 방법을 변경할 것을 제안합니다.

대신:

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \cdot \left(\rho \cdot \Delta\mathbf{v}^{(\tau)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}) \right), \tag{37}$$

$$\Delta\mathbf{v}^{(\tau)} = \rho \cdot \Delta\mathbf{v}^{(\tau-1)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)}). \tag{38}$$

NAG와 비교하면 위 방정식은 기울기 항을 계산할 때 앞을 보지 않는 반면, 바닐라 운동량과 비교하면 위 방정식은 현재 반복에서 운동량 항과 기울기 항을 모두 사용합니다. 항 $\rho \cdot \Delta\mathbf{v}^{(\tau)} + (1-\rho) \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(\tau-1)})$ 업데이트 방정식에 사용된 $\Delta\mathbf{v}^{(\tau+1)}$ 는 실제로는 다음 반복에서 Δv에 대한 근사치입니다. 변수를 업데이트할 때 앞을 내다보는 목표를 달성합니다.

한편, 방정식 (35)에 따르면 아담의 업데이트 방정식을 다시 쓸 수 있습니다. 다음과 같이 설정합니다.

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau)}} + \epsilon} \odot \hat{\mathbf{m}}^{(\tau)}, \tag{39}$$

$$\hat{\mathbf{m}}^{(\tau)} = \frac{\mathbf{m}^{(\tau)}}{1-\beta_1^{\tau}}, \text{ and } \mathbf{m}^{(\tau)} = \beta_1 \cdot \mathbf{m}^{(\tau-1)} + (1-\beta_1) \cdot \mathbf{g}^{(\tau)}. \tag{40}$$

mˆ(τ)를 교체하여 항을 방정식 (39)로 변환하면 다음과 같이 다시 작성할 수 있습니다.

$$\begin{aligned} \boldsymbol{\theta}^{(\tau)} &= \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau)}} + \epsilon} \odot \left(\frac{\beta_1 \cdot \mathbf{m}^{(\tau-1)}}{1-\beta_1^{\tau}} + \frac{(1-\beta_1) \cdot \mathbf{g}^{(\tau)}}{1-\beta_1^{\tau}} \right), \\ &= \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau)}} + \epsilon} \odot \left(\beta_1 \cdot \hat{\mathbf{m}}^{(\tau-1)} + \frac{(1-\beta_1) \cdot \mathbf{g}^{(\tau)}}{1-\beta_1^{\tau}} \right). \end{aligned} \tag{41}$$

방정식 (37)의 분석과 유사하게, 나담은 괄호 안에 사용된 mˆ(τ—1) 항을 mˆ(τ) 으로 대체하 여 다음과 같은 업데이트 방정식을 만들 수 있다고 제안합니다.

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{(\tau)}} + \epsilon} \odot \left(\beta_1 \cdot \hat{\mathbf{m}}^{(\tau)} + \frac{(1-\beta_1) \cdot \mathbf{g}^{(\tau)}}{1-\beta_1^{\tau}} \right). \tag{42}$$

나담 알고리즘의 의사 코드는 알고리즘 10에 제공되며, 변수 **θ**를 업데이트하는 마지막 줄을 제외하고 대부분의 코드가 알고리즘 9의 코드와 동일합니다.

알고리즘 10 나담

Require: Training Set T ; Learning Rate η; Normal Distribution Std σ; Mini-batch Size b; Decay

Parameters β1, β2.

Ensure: Model Parameter **θ**

1: Initialize parameter with Normal distribution **θ** ~ N(0, σ2)

2: Initialize vector **m** = **0**

3: Initialize vector **v** = **0**

4: Initialize step τ = 0

5: Initialize convergence tag = False

6: **while** tag == False **do**

7: Shuffle the training set T

8: **for** each mini-batch B ⊂ T **do**

9: Update step τ = τ + 1

Compute gradient vector g = $\nabla\boldsymbol{\theta}L(\boldsymbol{\theta}; B)$ on the mini-batch B

Update vector $\mathbf{m} = \beta_1 \cdot \mathbf{m} + (1 - \beta_1) \cdot \mathbf{g}$

Update vector $\mathbf{v} = \beta_2 \cdot \mathbf{v} + (1 - \beta_2) \cdot \mathbf{g} \odot \mathbf{g}$

Rescale vector $\hat{\mathbf{m}} = \mathbf{m}/(1 - \beta_1^\tau)$

Rescale vector $\hat{\mathbf{v}} = \mathbf{v}/(1 - \beta_2^\tau)$

Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}}+\epsilon} \odot \left(\beta_1 \cdot \hat{\mathbf{m}} + \frac{(1-\beta_1)\cdot\mathbf{g}}{1-\beta_1^\tau}\right)$

end for

if convergence condition holds **then**

tag = T rue

end if

end while

Return model variable **θ**

6. 하이브리드 진화적 경사 하강 학습 알고리즘

Adam은 다양한 변종과 함께 대규모 문제 그룹을 최적화하는 데 효과적인 것으로 나타났습니다. 그러나 딥러닝 모델의 비볼록 목적함수의 경우, Adam은 전역적으로 최적의 솔루션을 식별할 수 없으며, 반복적인 업데이트 프로세스가 로컬 최적화에 멈출 수밖에 없습니다. Adam의 성능은 그다지 견고하지 않기 때문에 모양이 매끄럽지 않은 목적 함수나 노이즈 데이터로 오염된 학습 시나리오에서는 성능이 크게 저하될 수 있습니다. 또한, Adam의 분산 계산 프로세스에는 많은 동기화가 필요하므로 대규모 클러스터 기반 분산 계산 플랫폼에서 채택하는 데 방해가 될 수 있습니다.

한편, 진화 알고리즘의 자연 선택 과정에서 영감을 얻은 메타 휴리스틱 알고리즘인 유전 알고리즘(GA)도 많은 최적화 문제의 해법을 학습하는 데 널리 사용되어 왔습니다. GA에서는 후보 솔루션의 모집단이 초기화되고 더 나은 솔루션을 향해 진화합니다. 경사 하강 기반 방법 대신 심층 신경망 모델을 훈련하기 위해 GA를 사용하려는 시도도 여러 차례 있었습니다[12, 7, 1]. GA는 여러 개의 로컬 최적점을 포함하는 비볼록 목적 함수와 같은 많은 학습 시나리오에서 뛰어난 성능을 입증했습니다.

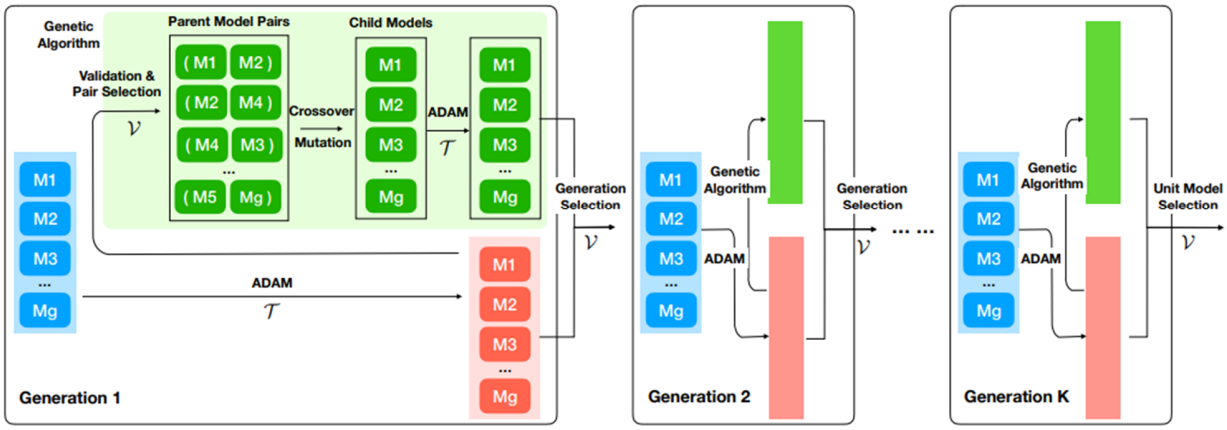


그림 2: Gadam 모델의 전체 아키텍처.

매끄럽지 않은 형태, 많은 수의 매개변수와 노이즈가 있는 환경에도 적합합니다. 또한 GA는 병렬/분산 컴퓨팅 환경에 매우 적합하며, 학습 과정을 병렬/분산 컴퓨팅 플랫폼에 쉽게 배포할 수 있습니다. 한편, GA는 Adam에 비해 최적화 목적 함수를 해결할 때 수렴하는 데 더 많은 라운드가 필요할 수 있습니다.

이 섹션에서는 아담과 GA를 통합 학습 체계에 통합한 새로운 최적화 알고리즘인 Gadam(유전적 적응 모멘텀 추정)[11]을 소개합니다. Gadam은 하나의 단일 모델 솔루션을 학습하는 대신 잠재적인 단위 모델 솔루션 그룹과 함께 작동합니다. 학습 과정에서 Gadam은 Adam을 통해 단위 모델을 학습하고 유전 알고리즘을 통해 새로운 세대로 진화시킵니다. 또한, 알고리즘 Gadam은 독립형 모드와 병렬/분산 모드 모두에서 작동할 수 있으며, 이 섹션에서는 이에 대해서도 살펴봅니다.

6.1 Gadam

그림 2에는 여러 학습 세대를 포함하는 Gadam의 모델 학습 프로세스의 전체 프레임워크가 설명되어 있습니다. 각 세대마다 데이터로부터 단위 모델 변수 그룹이 아담과 함께 학습되며, 이 변수 그룹도 유전 알고리즘을 통해 효과적으로 진화합니다. 다음 세대를 구성하기 위해 좋은 후보 변수가 선택됩니다. 이러한 반복적인 모델 학습 과정은 수렴까지 계속되며, 최종 세대에서 최적의 단위 모델 변수가 출력 모델 솔루션으로 선택됩니다. 여기서는 편의상 단위 모델과 그 변수를 구분하지 않고 혼용하여 사용하겠습니다.

6.1.1 모델 모집단 초기화

Gadam은 단위 모델 집합(즉, 기본적으로 이러한 단위 모델의 변수)을 기반으로 최적의 모델 변수를 학습하는데, 초기 단위 모델 생성은 $\mathcal{G}^{(0)} = \{M_1^{(0)}, M_2^{(0)}, \cdots, M_g^{(0)}\}$ 집합으로 나타낼 수 있습니다. (G는 모집단 크기, 위 첨자는 생성 인덱스를 나타냅니다). 초기 생성 인덱스에 따라 으로 표현할 수 있는 새로운 세대의 유닛 모델로 진화할 것입니다. $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(K)}$ 여기서 파라미터 K는 총 세대 수를 나타냅니다.

초기 세대 G의 각 유닛 모델에 대해 $\mathcal{G}^{(0)}$
(예: $M_i^{(0)}$)나, 그 변수 $\boldsymbol{\theta}_i^{(0)}$ 특정 분포에서 샘플링된 임의의 값으로 Gadam에서 초기화됩니다 (예: 표준정규 분포).
이러한 초기 생성은 시작 검색 지점 역할을 합니다. Gadam은 최적의 솔루션을 식별하기 위해

다른 지역으로 확장할 것입니다. 그 동안에, 다음 세대의 단위 모델에 대한 변수 값은 다음을 통해 생성됩니다. 각각 상위 모델의 GA입니다.

6.1.2 아담과 함께하는 모델 학습

학습 과정에서 어떤 모델 생성 $G^{(k)}$($k \in \{1, 2, - - - , K\}$)가 주어지면 Gadam은 Adam을 통 해 각 단위 모델에 대한 (로컬) 최적 변수를 학습합니다. 공식적으로, Gadam는 여러 개의 에포크로 단위 모델을 훈련합니다. 각 에포크에서 각 단위 모델 $M^{(k)} \in G^{(k)}$ 에 대해 훈련 데이터세트에서 분리된 훈련 배치가 무작위로 샘플링되며, 이는 $B = \{(\mathbf{x}1 , \mathbf{y}1), (\mathbf{x}2 , \mathbf{y}2), - -, (\mathbf{x}b , \mathbf{y}b)\} \subset T$로 나타낼 수 있습니다(여기서 b는 배치 크기를 나타내고 T는 전체 훈련 세트를 나타냄). 단위 모델에 의해 도입된 손실 함수는 다음과 같습니다. $M_i^{(k)}$ 미니 배치 B를 학습하기 위한 $\ell(\boldsymbol{\theta}_i^{(k)})$, 학습된 모델 변수 벡터는 교육 인스턴스는 다음과 같이 표현할 수 있습니다.

$$\bar{\boldsymbol{\theta}}_i^{(k)} = Adam\left(\ell(\boldsymbol{\theta}_i^{(k)})\right). \tag{43}$$

특정 단위 모델과 애플리케이션 설정에 따라 손실 함수는 평균 제곱 손실, 힌지 손실 또는 교 차 엔트로피 손실과 같이 다른 표현을 갖습니다. 이러한 학습 과정은 컨버전스까지 계속되며, 업데이트된 모델 생성 $G^{(k)}$는 (국부적으로) 최적 변수는 $\bar{\mathcal{G}}^{(k)} = \{\bar{M}_1^{(k)}, \bar{M}_2^{(k)}, \cdots, \bar{M}_g^{(k)}\}$로 나타낼 수 있으며, 그 값은 다음과 같습니다. 해당 변수 벡터는 각각 $\bar{\boldsymbol{\theta}}_1^{(k)}, \bar{\boldsymbol{\theta}}_2^{(k)}, \cdots, \bar{\boldsymbol{\theta}}_g^{(k)}\}$로 나타낼 수 있습니다.

6.1.3 유전 알고리즘을 통한 모델 진화

이 부분에서는 학습된 단위 모델, 즉 $\bar{\mathcal{G}}^{(k)}$를 기반으로 가담이 유전 알고리즘을 통해 단위 모델 에 대한 더 나은 솔루션을 효과적으로 탐색합니다.

모델 적합성 평가

성능이 좋은 단위 모델이 학습 과제에 더 적합할 수 있습니다. 가담은 무작위 모델을 진 화시키는 대신 현재 세대에서 좋은 단위 모델을 골라 진화시킬 것을 제안합니다. 단위 모델의 적합도 점수는 샘플링된 검증 배치 V ⊂ T 를 기반으로 합니다,

예를 들어, $\bar{M}_i^{(k)}$는 다음과 같이 V에 도입된 손실 항을 기반으로 효과적으로 계산할 있습니다.

$$\mathcal{L}_i^{(k)} = \mathcal{L}(\bar{M}_i^{(k)}; \mathcal{V}) = \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{V}} \ell(\mathbf{x}_j, \mathbf{y}_j; \bar{\boldsymbol{\theta}}_i^{(k)}). \tag{44}$$

계산된 손실 값에 따라 단위 모델 $\bar{M}_i^{(k)}$ 의 선택 확률은 다음과 같습니다. 다음 소프트맥스 방 정식으로 정의됩니다.

$$P(\bar{M}_i^{(k)}) = \frac{\exp^{(-\hat{\mathcal{L}}_i^{(k)})}}{\sum_{j=1}^{g} \exp^{(-\hat{\mathcal{L}}_j^{(k)})}}. \tag{45}$$

손실 값이 매우 크거나 작은 양수 또는 음수 손실 값의 경우 0 또는 ∞에 가까워질 수 있으므 로 실제 애플리케이션에서는 일반적으로 손실 값의 정규화가 필요합니다. exp($^{(-\mathcal{L}_i^{(k)})}$) 확률 방정식에서 알 수 있듯이 모든 단위 모델의 정규화된 손실 항은 공식적으로 $\mathcal{L}_i^{(k)}$. $[\hat{\mathcal{L}}_1^{(k)}, \hat{\mathcal{L}}_2^{(k)}, \cdots, \hat{\mathcal{L}}_g^{(k)}]^\top$ 로 나타낼 수 있으며, 여기서 $\hat{\mathcal{L}}_i^{(k)} \in [0,1], \forall i \in \{1, 2, \cdots, g\}$입니다. 계산된 확률에 따르면, 단위 모델 세트 $\bar{\mathcal{G}}^{(k)}$ 에서 단 위 쌍의 단위 g 단위 쌍 모델은 진화를 위한 상위 모델로 교체를 통해 선택됩니다. 로 표시 $\mathcal{P} = \{(\bar{M}_{i_1}^{(k)}, \bar{M}_{j_1}^{(k)}), (\bar{M}_{i_2}^{(k)}, \bar{M}_{j_2}^{(k)}), \cdots, (\bar{M}_{i_g}^{(k)}, \bar{M}_{j_g}^{(k)})\}$.

유닛 모델 크로스오버

단위 모델 쌍(예: $(\bar{M}_{i_p}^{(k)}, \bar{M}_{j_p}^{(k)}) \in \mathcal{P}$)이 주어지면 Gadam은 해당 변수를 다음과 같이 상속합니다. 를 크로스오버 연산을 통해 자식 모델에 적용합니다. 크로스오버에서 부모 모델, 즉 $\bar{M}_{i_p}^{(k)}$ 와 $\bar{M}_{j_p}^{(k)}$ 도 서로 경쟁하게 되며, 여기서 더 나은 퍼포머 모델을 가진 부모 모델이 더 높은 점수를 얻게 됩니다. 만스가 더 많은 이점을 갖는 경향이 있습니다. 다음에서 생성된 하위 모델을 표현할 수 있습니다.

가중치 변수 $\tilde{\theta}_p^{(k)}$ 의 각 항목에 대해 자식 모델 $\tilde{M}_p^{(k)}$ 의 Gadam은 해당 값을 다음과 같이 초기화합니다.

$$\tilde{\boldsymbol{\theta}}_p^{(k)}(m) = \mathbb{1}(\text{rand} \leq p_{i_p,j_p}^{(k)}) \cdot \bar{\boldsymbol{\theta}}_{i_p}^{(k)}(m) + \mathbb{1}(\text{rand} > p_{i_p,j_p}^{(k)}) \cdot \bar{\boldsymbol{\theta}}_{j_p}^{(k)}(m). \tag{46}$$

방정식에서 이항 함수 1(-)은 조건이 충족되면 1을 반환합니다. 용어 "rand"는 다음을 나타냅니다. 0, 1]의 난수입니다. 확률 임계값 $p_{i_p,j_p}^{(k)}$은 부모의 모델의 퍼포먼스를 기반으로 정의 됩니다.

모델의 성능:

$$p_{i_p,j_p}^{(k)} = \frac{\exp^{(-\hat{\mathcal{L}}_{i_p}^{(k)})}}{\exp^{(-\hat{\mathcal{L}}_{i_p}^{(k)})} + \exp^{(-\hat{\mathcal{L}}_{j_p}^{(k)})}}. \tag{47}$$

만약 $\hat{\mathcal{L}}_{i_p}^{(k)} > \hat{\mathcal{L}}_{j_p}^{(k)}$,즉 모델 $\bar{M}_{i_p}^{(k)}$ 가 $\bar{M}_{j_p}^{(k)}$보다 더 큰 손실을 가져오는 경우, 0 < $p_{i_p,j_p}^{(k)}$ <1/2 자식이러한 프로세스를 통해 세트 P의 전체 부모 모델 쌍을 기반으로 Gadam은 다음과 같습니다. 모델세트를 $\tilde{\mathcal{G}}^{(k)} = \{\tilde{M}_1^{(k)}, \tilde{M}_2^{(k)}, \cdots, \tilde{M}_g^{(k)}\}$ 집합으로 생성할 수 있습니다.

단위 모델 돌연변이

단위 모델이 로컬 최적점에 갇히는 것을 방지하기 위해 Gadam은 집합에서 생성된 자식 모델의 변수 값을 조정하기 위해 돌연변이라는 연산 방식을 채택합니다.

$\{\tilde{M}_1^{(k)}, \tilde{M}_2^{(k)}, \cdots, \tilde{M}_g^{(k)}\}$공식적으로, 벡터로 매개변수화된 각 자식 모델 $\tilde{M}_q^{(k)}$에 대해 $\tilde{\boldsymbol{\theta}}_q^{(k)}$, Gadam은 다음 방정식에 따라 변수 벡터를 돌연변이합니다. mth 항목은 다음과 같이 업데이트할 수 있습니다.

$$\tilde{\boldsymbol{\theta}}_q^{(k)}(m) = \mathbb{1}(\text{rand} \leq p_q) \cdot \text{rand}(0,1) + \mathbb{1}(\text{rand} > p) \cdot \tilde{\boldsymbol{\theta}}_q^{(k)}(m), \tag{48}$$

이 방정식에서 용어 pq 는 변이율을 나타내며, 이는 상위 모델의 성능과 밀접한 상관관계가 있습니다:

$$p_q = p \cdot \left(1 - P(\bar{M}_{i_q}^{(k)}) - P(\bar{M}_{j_q}^{(k)})\right). \tag{49}$$

좋은 부모 모델을 가진 자식 모델의 경우 돌연변이율이 낮습니다. 용어 p은 일반적으로 작은 값(예: 0.01)인 기본 돌연변이율과 확률을 나타냅니다. $P(\bar{M}_{i_q}^{(k)})$와 $P(\bar{M}_{j_q}^{(k)})$은 방정식에 정의 되어 질 수 있습니다 (45).

이 유닛 모델은 더욱 발전할 것입니다. 융합될 때까지 아담과 함께 훈련하여 kth번째 자식모델 세대로 이어질 것 $\tilde{\mathcal{G}}^{(k)} = \{\tilde{M}_1^{(k)}, \tilde{M}_2^{(k)}, \cdots, \tilde{M}_g^{(k)}\}$입니다.

6.1.4 새로운 세대 선택 및 진화 중지 기준

학습된 상위 모델 세트의 학습된 단위 모델 중 $\bar{\mathcal{G}}^{(k)} = \{\bar{M}_1^{(k)}, \bar{M}_2^{(k)}, \cdots, \bar{M}_g^{(k)}\}$ 입니다.

로 설정하고 자식 모델 세트 $\tilde{\mathcal{G}}^{(k)} = \{\tilde{M}_1^{(k)}, \tilde{M}_2^{(k)}, \cdots, \tilde{M}_g^{(k)}\}$ 를 설정하면 Gadam은 적합성을 다시 평가합니다.

점수는 공유된 새 검증 배치를 기반으로 합니다. $\bar{\mathcal{G}}^{(k)} \cup \tilde{\mathcal{G}}^{(k)}$ 의 모든 단위 모델 중에서 상위 g 단위 모델이 선택되어 (k + 1)th 세대를 형성하며, 식적으로 다음과 같이 표현할 수 있습니다.

집합 $\mathcal{G}^{(k+1)} = \{M_1^{(k+1)}, M_2^{(k+1)}, \cdots, M_g^{(k+1)}\}$ 로 표현됩니다. 이러한 진화 학습은 프로세스는 최대 생성 수에 도달했거나 유의미한 생성 수가 없는 우 중지됩니다.

결과 세대 간 개선(예: $G^{(k)}$와 $G^{(k+1)}$)을 비교합니다:

$$\left| \sum_{M_i^{(k)} \in \mathcal{G}^{(k)}} \mathcal{L}_i^{(k)} - \sum_{M_i^{(k+1)} \in \mathcal{G}^{(k+1)}} \mathcal{L}_i^{(k+1)} \right| \leq \lambda, \tag{50}$$

위의 공식은 Gadam의 중지 기준을 정의하며, 여기서 λ는 진화 중지 임계값입니다. 적화 알고리즘인 Gadam은 실제로 아담과 유전 알고리즘의 장점을 모두 통합한 것입니다. Adam을 사용하면 단위 모델이 몇 번의 훈련만으로 매우 빠르게 (로컬/글로벌) 최적 솔루션을 효과적으로 얻을 수 있습니다. 한편, 여러 단위 모델을 기반으로 한 유전 알고리즘을 통해 여러 출발점에서 솔루션을 검색하고 로컬 최적에서 벗어날 수 있는 기회도 제공합니다. 에 따르면 [4]에 따르면, 부드러운 함수를 위해 아담은 함수 기울기가 사라짐에 따라 수렴합니다. 반면에 유전 알고리즘은 [8]에 따라 수렴할 수도 있습니다. 이러한 사전 지식을 바탕으로 [11]에서 소개한 것처럼 가담의 수렴을 증명할 수 있습니다. Gadam을 이용한 단위 모델의 학습은 병렬/분산 컴퓨팅 플랫폼에 효과적으로 배포할 수 있으며, Gadam에 포함된 각 단위 모델은 별도의 프로세스/서버를 통해 학습할 수 있습니다. 프로세스/서버 중 통신 비용은 크로스오버 단계에만 존재할 정도로 미미합니다. 말 그대로, 각 세대의 모든 단위 모델 중에서 이들 사이의 통신 비용은 O(k - g - dθ)이며, 여기서 dθ 는 벡터의 차원을 나타냅니다. θ와 k 아담이 수렴을 달성하는 데 필요한 훈련 기간을 나타냅니다.

7. 요약

이 백서에서는 경사 하강 알고리즘과 함께 딥러닝 모델 학습을 위한 다양한 최신 변형 알고리즘을 소개했습니다. 이 방향에 대한 최근의 발전을 바탕으로 이 백서는 가까운 시 일 내에 그에 따라 업데이트될 예정입니다.

참조

[1] 일라이 데이비드와 이도 그린. 심층 신경망의 진화를 위한 유전 알고리즘. CoRR, abs/1711.07655, 2017.

[2] 티모시 도자트. 네스테로프 모멘텀을 아담에 통합하기.

[3] 존 두치, 엘라드 하잔, 요람 싱어. 온라인 학습 및 확률론적 최적화를 위한 적응형 하위 그라데이션 방법. J. Mach. Learn. Res., 12:2121-2159, July 2011.

[4] 디데릭 P. 킹마, 지미 바. Adam: 확률적 최적화를 위한 방법. CoRR, abs/1412.6980, 2014.

[5] 유리 네스테로프. 수렴율 O(1/sqr(k))로 볼록 프로그래밍 문제를 푸는 방법. 소비에트 수학 도클라디, 27:372-376, 1983.

[6] 닝 첸. 경사 하강 학습 알고리즘의 운동량 항에 대하여. 신경망, 12(1):145-151, January 1999.

[7] 펠리페 페트로스키 Such, 바쉬트 마드하반, 에도아르도 콘티, 조엘 리먼, 케네스 오 스탠리, 제프 클룬. 심층 신경 진화: 유전 알고리즘은 강화 학습을 위한 심층 신경망을 훈련하기 위한 경쟁적인 대안입니다. CoRR, abs/1712.06567, 2017.

[8] 더크 티렌스와 데이비드 E. 골드버그. 유전 알고리즘 선택 체계의 융합 모델. 진화 계산에 관한 국제 컨퍼런스 논문집. 자연에서 병렬 문제 해결에 관한 세 번째 컨퍼런스: 자연으로부터의 대립 유전자 문제 해결, PPSN III, 119-129 페이지, 영국 런던, 영국, 1994. 스프링거-베를라그.

[9] T. 티엘레만과 G. 힌튼. 강의 6.5-RmsProp: 기울기를 최근 크기의 평균으로 나눕니다. 코스: 기계 학습을 위한 신경망, 2012.

[10] 매튜 D. 자일러. ADADELTA: 적응형 학습 속도 방법. CoRR, abs/1212.5701, 2012.

[11] 지아웨이 장과 피셔 B. 구자. GADAM: 심층 신경망 최적화를 위한 유전적 진화 ADAM. CoRR, abs/1805.07500, 2018.

[12] 지아웨이 장과 피셔 B. 구자. SEGEN: 샘플 앙상블 유전 진화 네트워크 모델. CoRR, abs/1803.08631, 2018.