6장. 복합 출력

부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공
201724579 정현모

1. 개요

딥러닝 모델이 빠르고 정확하게 학습하기 위해 필요한 것이 Optimizer이다. Optimizer는 각각의 특성을 이해하고 모델에 적절한 Optimizer를 사용하는 것이 중요하다. 우리는 기존의 Stochastic Gradient Descent와 새로운 Optimizer인 Adam Optimizer에 대해 알아본다.

전이 학습이란 한 도메인에서 학습시킨 결과를 다른 도메인에 활용하는 기법이다. 이를 통해 고비용 작업인 레이블링을 일부 도메인에서 줄일 수 있다. 데이터가 적을 때 효과적이며 학습 속도도 빠르다. 또한 random한 값에서 학습하는 것 보다 훨씬 높은 정확도를 제공하기도 한다. 이번 과제에서 실제로 전이학습이 얼마나 어떻게 영향을 미치는 지 실험을 통해 알아본다.

2. 학습 결과

2.1. Office31 학습 비교

2.1.1. hidden layer, epoch count, learning rate 에 따른 학습 비교

파란색 하이라이트는 변경된 부분을 의미한다.

office_model_1

구분	내용	값
조건	Epoch	**50**
	hidden layer	**[64, 32, 10]**
	Learning rate	**0.001**
	Adam Optimizer	False
최종 결과	Accuarcy	66.1% + 4.8%
	Train time	**44 secs**
결과 화면	Epoch 10: cost=4.296, accuracy=0.685+0.035/0.790+0.040 (22/22 secs) Epoch 20: cost=4.268, accuracy=0.685+0.037/0.780+0.020 (22/44 secs) Model office31_model_1 train ended in 44 secs: Model office31_model_1 test report: accuracy = 0.661+0.048, (0 secs)	

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]

6장. 복합 출력

office_model_2

구분	내용	값
조건	Epoch	**35**
	hidden layer	**[32, 10]**
	Learning rate	**0.001**
	Adam Optimizer	False
최종 결과	Accuarcy	66.1% + 3.2%
	Train time	138 secs
결과 화면	Model office31_model_2 train started: Epoch 10: cost=4.296, accuracy=0.685+0.039/0.770+0.030 (46/46 secs) Epoch 20: cost=4.268, accuracy=0.685+0.036/0.830+0.030 (45/91 secs) Epoch 30: cost=4.263, accuracy=0.685+0.036/0.790+0.040 (47/138 secs) Model office31_model_2 train ended in 160 secs: Model office31_model_2 test report: accuracy = 0.661+0.032, (0 secs)	

office_model_3

구분	내용	값
조건	Epoch	**50**
	hidden layer	**[64, 32, 10]**
	Learning rate	**0.001**
	Adam Optimizer	False
최종 결과	Accuarcy	82.4% + 24.3%
	Train time	402 secs
결과 화면	Model office31_model_3 train started: Epoch 10: cost=3.142, accuracy=0.842+0.217/0.880+0.150 (79/79 secs) Epoch 20: cost=2.828, accuracy=0.870+0.272/0.830+0.170 (78/157 secs) Epoch 30: cost=2.617, accuracy=0.884+0.315/0.840+0.180 (77/234 secs) Epoch 40: cost=2.460, accuracy=0.901+0.346/0.810+0.200 (80/314 secs) Epoch 50: cost=2.296, accuracy=0.900+0.384/0.910+0.260 (88/402 secs) Model office31_model_3 train ended in 402 secs: Model office31_model_3 test report: accuracy = 0.824+0.243, (0 secs)	

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]

6장. 복합 출력

office_model_4

구분	내용	값
조건	Epoch	**50**
	hidden layer	**[64, 32, 10]**
	Learning rate	**0.0001**
	Adam Optimizer	False
최종 결과	Accuarcy	**86.7% + 19.7%**
	Train time	411 secs
결과 화면	Model office31_model_4 train started: Epoch 10: cost=3.745, accuracy=0.805+0.098/0.860+0.080 (83/83 secs) Epoch 20: cost=3.471, accuracy=0.843+0.146/0.830+0.090 (82/165 secs) Epoch 30: cost=3.263, accuracy=0.866+0.198/0.840+0.100 (82/247 secs) Epoch 40: cost=3.106, accuracy=0.880+0.223/0.880+0.140 (82/329 secs) Epoch 50: cost=2.964, accuracy=0.889+0.252/0.880+0.160 (82/411 secs) Model office31_model_4 train ended in 411 secs: Model office31_model_4 test report: accuracy = 0.867+0.197, (0 secs)	

2.1.2. Adam optimizer 에 따른 학습 비교

위의 학습에서 가장 성능이 좋았던 4 번 모델의 Epoch 와 hidden layer, learning rate 를 가지고 Adam Optimizer 의 유무에 따른 결과를 비교해 보겠다.

office_model_5

구분	내용	값
조건	Epoch	50
	hidden layer	[64, 32, 10]
	Learning rate	0.0001
	Adam Optimizer	**True**
최종 결과	Accuarcy	**86.7% + 24.3%**
	Train time	1757 secs
결과 화면	Model office31_model_5 train started: Epoch 10: cost=3.418, accuracy=0.829+0.125/0.890+0.090 (339/339 secs) Epoch 20: cost=3.099, accuracy=0.862+0.191/0.780+0.180 (343/682 secs) Epoch 30: cost=2.853, accuracy=0.889+0.234/0.860+0.150 (360/1042 secs) Epoch 40: cost=2.689, accuracy=0.887+0.274/0.880+0.160 (367/1409 secs) Epoch 50: cost=2.550, accuracy=0.891+0.309/0.880+0.220 (348/1757 secs) Model office31_model_5 train ended in 1757 secs: Model office31_model_5 test report: accuracy = 0.867+0.243, (0 secs)	

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]

6장. 복합 출력

3. 결론

3.1. 전이 학습

전이 학습은 아주 큰 데이터셋에서 훈련된 모델의 가중치를 가지고 와서 우리가 해결하고 하는 과제에 맞게 재보정해서 사용하는 것을 의미한다. 이번 과제에선 도메인을 학습한 모델로 상품을 분류하는 모델에 함께 사용하고 있다. 하나의 모델에서 두 가지 출력에 대해 함께 사용하고 있어서 모든 모델에서 정확도가 낮음을 확인할 수 있다. 하지만 지금처럼 각 클래스에 대한 데이터가 부족한 경우, 혹은 높은 학습 비용을 감당하기 힘든 경우, 전이학습이 하나의 좋은 방법이 될 수 있다.

3.2. Optimizer

3.2.1. Stochastic Optimizer

지금까지 과제에서 사용한 최적화 방법이다. 엄밀하게 말하면 Mini-batch gradient descent(MSGD)로 단순히 Stochastic Optimizer라고 부르기도 한다. Gradient Descent는 학습데이터가 너무 많을 경우 학습 시간이 너무 오래 걸린다는 단점이 있다. 하지만 전체 데이터 대신 MSGD는 일부 데이터(Mini-Batch)를 사용하여 계산한다면 훨씬 빠른 속도로 학습을 진행할 수 있다. 계산 속도가 매우 빠르기 때문에 같은 시간에 더 많은 step을 갈 수 있으며, Gradient Descent에서 빠질 수 있는 Local Minima에 빠지지 않고 더 좋은 방향으로 수렴할 가능성도 있다는 장점이 있다.

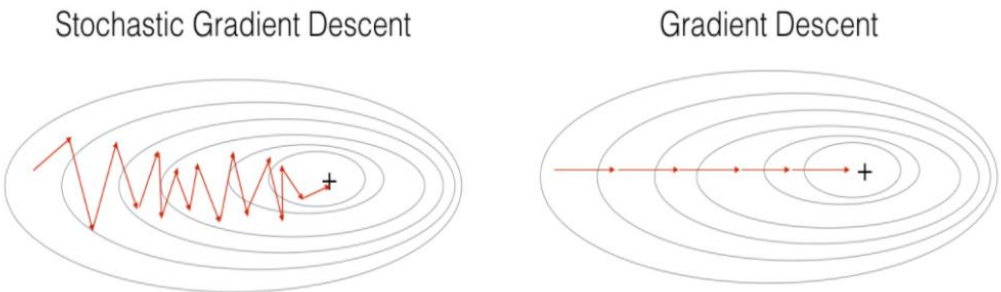

그림 1. (좌)MSGD (우)BGD

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]

3.2.2. Adam Optimizer

기존의 Gradient Descent에서 여러 아이디어를 합쳐서 만들어진 최적화 기법이다. 첫번째 아이디어는 각 parameter 별로 상대적인 변화를 주어 높은 값을 가지는 parameter는 상대적으로 적은 변화를 주고 반대로 적게 이동한 parameter는 큰 변화를 주는 방법이다. 이를 RMSProp Optimizer이라고 부른다. RMSProp Optimizer는 최근 변화량의 변수간 상대적인 크기 차이는 유지할 수 있다는 장점이 있다. 이로 인해 학습 속도가 0에 수렴하는 것은 방지할 수 있다. 하지만 속도가 느리다는 단점이 있다.

$$G_t = \gamma G_{t-1} + (1-\gamma)(\nabla_\omega J(\omega_t))^2$$

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla_\omega J(\omega_t) \quad (1)$$

그림 2. RMSProp Optimizer

두번째 아이디어는 관성이다. Gradient Descent를 통해 파라미터 값이 변할 때, 그 변화량만큼 기울기를 박차고 올라가는 기법이다. 이를 이용하면 local minima에 빠지는 경우를 관성력을 이용해 빠져나올 수 있다. 이 두 아이디어를 합쳐서 Adam Optimizer라고 부른다. 아래 식에서 엡실론은 아주 작은 값 (1^{-8}), 베타1은 0.9, 베타2는 0.999로 보통 대입한다.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\omega J(\omega_t)$$

$$v_t = \beta_2 m_{t-1} + (1-\beta_2)(\nabla_\omega J(\omega_t))^2$$

$$\omega_{t+1} = \omega_t - m_t \frac{\eta}{\sqrt{v_t + \epsilon}}$$

그림 3. Adam Optimizer

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]

6장. 복합 출력

3.3. 실험 결과

지금까지 진행해 온 과제들을 통해 epoch는 높을수록, hidden layer는 많을수록 대체로 좋은 성능을 보여주는 것을 확인하였다. 이번 실험에서 역시 두 하이퍼 파라미터가 클수록 좋은 결과를 보여주었다. 이번 과제에선 전이 학습을 통해 기존에 학습한 내용을 다른 학습내용에 사용하는 것을 보여주었는데, 두 학습 데이터에 교집합이 있어서 성능이 많이 낮게 나온 것이라고 생각한다. 같은 가방 사진에서 도메인 학습결과는 다르게 나와야 하고, 상품 결과는 같게 나와야 한다는 점이 문제였다고 생각한다. 전이학습을 할 때 두 데이터의 특성이 비슷하다면 더 좋은 결과가 나올 것이라고 생각한다.

Adam Optimizer를 사용했을 때, 성능이 약간 더 좋아졌지만 학습 시간이 너무 오래 걸리는 단점이 있다. 최적화 함수가 복잡해져서 각 연산에 걸리는 시간이 오래 걸렸다. 이번 실험의 경우 더 짧은 시간이 걸리는 MSGD 방식을 사용하는 것도 괜찮다고 생각한다.

[1] 출처: https://onevision.tistory.com/entry/Optimizer-의-종류와-특성-Momentum-RMSProp-Adam [312 개인 메모장]