week8: 논문 스터디: Batch normalization

예습	미완료
복습	미완료
복습과제 날짜	@2022년 10월 31일
예습과제 날짜	
내용	

논문

논문 제목: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

https://arxiv.org/abs/1502.03167

요약

Internal Covariate Shift 문제를 해결하기 위해 input을 정규화하는 방법으로 batch normalization을 제시한다. mini batch 단위의 input normalization을 network 구조의 일부로 포함 할 수 있다. 이러한 batch normalization을 사용하면 다음과 같은 장점이 있다.

- high learning rate
- weight init을 신경쓰지 않아도 된자.
- Drop out 이 필요 없다.

Introduction

SGD는 loss를 최소하하는 매개변수를 찾는 알고리즘이다.

SGD를 이용하는데는 hyper parameter을 잘 설정해야하고 모델의 매개변수 초기화를 잘 해야한다. 또한 각 레이어의 input을 모두 이전 레이어의 매개변수에 영향을 받아 모델이 더욱 복잡해진다.

레이어가 가지는 분포가 매 트레이닝 마다 바뀌면 레이어는 매번 새로운 분포에 맞춰야하는 문제점이 있다. 우리는 해당 문제점을 Internal Convariate Shiftf현상이라고 표현한다. 네트워크를 이루는 sub network input distribution이 일정하게 유지되면 학습성능이 개선 될 것으로 보인다.

Towards Reducing Internal Covariate Shift

Internal Covariate Shift를 줄이기 위한 방법을 소개한다. 첫번째로 데이터들의 분포 평균을 0, 분산을 1로 만들고 decorrelated시키는 whitening기법을 적용하였다. Internal Covariate Shift는 제거할 수 있지만, back-propagation이 제대로 이뤄지지 않는다.

Normalization via Mini-batch Statistics

Whitening 과정은 비용이 높고 미분이 불가해 입력 데이터를 각 스칼라 값 차원에서 독립적으로 정규화한다.Sigmoid 함수 그래프를 보면, -1~1구간은 거의 수직에 가까운 구간이기 때문에 activation function의 비선형성 특징이 없어진다. 그렇게 되면 레이어를 많이 쌓는 의미가 없어진다. 이러한 문제를 해결하기 위해 정규화된 값을 변경하거나 정규화 전의 값을 전달해야한다.

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

- 각 dimension마다 새로운 파라미터를 도입한다. (Scalining & shifting) 정규화된 값에 scalining과 shifting을 적용하는 것이다.
- 배치를 이용한 학습에선 전체의 평균과 분산을 구할 수 없다. 따라서 배치의 평균과 분산값을 이용해 정규화를 진행해야한다.

최종적으로 다음과 같은 알고리즘을 가지게 된다.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

전체 구간에서 미분이 가능하고, 정규화과정이 진행되어 input 분포를 일정하게 유지할 수 있어 Internal Covariate Shift 문제를 해결하고 학습 속도를 향상할 수 있다.

Transfer Learning

https://tutorials.pytorch.kr/beginner/transfer_learning_tutorial.html

https://colab.research.google.com/github/PyTorchKorea/tutorials-kr/blob/master/docs/_downloads/74249e7f9f1f398f57ccd094a4f3021b/transfer_learning_tutorial.ipynb

전이학습의 주요 2가지

- 합성곱 신경망의 미세조정

- 고정된 특징 추출기로써의 합성곱 신경망

데어터 : 개미와 벌을 분류하는 모델 학습. 각각의 학습용 이미지는 대략 120장, 75장의 검증용 이미지.

모델 학습 방법

- Learning rate scheduling

  학습중에는 순전파+ 역전파+ 최적화를 통해 통계를 낸뒤, 모델을 Deep Copy한다. 이 중 가장 나은 모델 가중치를 불러온다.

```
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
# 여기서 각 출력 샘플의 크기는 2로 설정합니다.
# 또는, nn.Linear(num_ftrs, len (class_names))로 일반화할 수 있습니다.
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# 모든 매개변수들이 최적화되었는지 관찰
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

# 7 에폭마다 0.1씩 학습률 감소
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                       num_epochs=25)
```

- 최적의 모델 구하기

```
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# 새로 생성된 모듈의 매개변수는 기본값이 requires_grad=True 임
  backward()중 경사가 계산되지 않도록 한다.
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

# 이전과는 다르게 마지막 계층의 매개변수들만 최적화되는지 관찰
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

# 7 에폭마다 0.1씩 학습률 감소
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
```

Classifier

일반적으로 데이터를 다룰때 numpy배열로 불러온 뒤, torch의 tensor로 변환한다. 각 데이터별로 주요 사용 패키지가 있다.

- 이미지는 Pillow나 OpenCV 같은 패키지가 유용합니다.

- 오디오를 처리할 때는 SciPy와 LibROSA가 유용하고요.
- 텍스트의 경우에는 그냥 Python이나 Cython을 사용해도 되고, NLTK나 SpaCy도 유용합니다.
- 영상 분야 - torchvision (일반적인 데이터 로더, 변환기가 포함되어있다)

CIFAR10 데이터셋 : 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭으로 분류되는 이미지 데이터셋, 각각 3*32*32의 사이즈를 가지고 있다.

이미지 분류기 학습 순서

1. Torchvision을 이용하여 CIFAR10의 데이터셋을 불러와 정규화한다.
2. 합성곱신경망을 정의

   3채널 이미지를 처리할 수 있다록 한다.
3. 손실함수 정의

   Cross-Entropy loss와 모멘텀을 가지는 SGD 사용
4. 학습용 데이터로 신경망 학습
5. 시험용 데이터로 신경망 검사