Batch-Normalization

0. abstract

Internal Covariate Shift: 매번 훈련마다 각 layer의 입력값의 distribution이 변하는 현상

-> requiring lower learning rate and careful parameter initialization

Batch Normalization을 이용

-> allow higher learning rate and less careful initialization
-> act as a regularizer (eliminate the need for Dropout)

1. introduction

SGD: loss값을 최소화하기 위해 파라미터들을 최적화하는 알고리즘

SGD + mini-batch 이점
-> mini-batch의 gradient는 전체 training set gradient의 근사값이다. (size가 커지면 성능 증가)
-> 병렬처리를 사용해 효율적인 계산이 가능하다.

SGD 단점
-> learning rate와 hyper-parameter tuning, model parameter initialization에 주의해야 한다.

1. introduction

각 layer의 입력은 이전 층의 모든 파라미터들에게 영향을 받는다.
-> 훈련 복잡해진다. (model이 deep해질수록 더 심해진다.)

하지만 매 훈련마다 layer들의 입력의 distribution이 변한다.
-> layer가 새로운 distribution에 적응해야 한다.

‘experience covariate shift’

1. introduction

l: loss / F1, F2: arbitrary transformation / theta1, 2: parameters

$$\ell = F_2(F_1(u, \Theta_1), \Theta_2)$$

$$x = F_1(u, \Theta_1)$$

$$\ell = F_2(x, \Theta_2)$$

한 번의 gradient descent 단계가 입력이 x인 F2 network 한 개가 있는 것과 동일하다.

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2}$$

즉, training 데이터와과 test 데이터의 distribution이 비슷하게 주어지면 효과적인 훈련을 수행하는 입력 distribution의 특징이 sub-network에도 적용된다. 따라서 x의 distribution을 고정하는 것이 좋다.

1. introduction

Sub-network에 대한 고정된 입력 distribution은 sub network 밖의 층에도 긍정적인 결과를 가져다 준다.

Gradient vanishing (saturating nonlinear function)
-> gradient를 사라지게 하고 모델의 학습 속도를 늦춘다. (model이 deep할수록 심해진다.)
-> 해결법: ReLU, careful initialization, small learning rate

이때 saturating nonlinear function들의 입력값의 distribution을 고정시켜주면 위 현상을 예방할 수 있다.

1. introduction

Batch Normalization의 이점
- internal covariate shift를 예방하여 빠른 훈련이 가능하다.
- Saturating nonlinear function의 문제점을 예방할 수 있다.
- regularizer의 역할을 수행한다.

2. Towards Reducing Internal Covariate Shift

whitening 기법: 평균 0, 분산 1로 표준화

단순히 whitening만 수행시키면 whitening 과정과 parameter 최적화 과정이 무관하게 진행되어 특정 파라미터가 계속 커지는 상태로 whitening이 진행된다.

$$\widehat{x} = x - \mathbb{E}[x], \text{ where } x = u + b, \mathcal{X} = \{x_1, \cdots, x_N\}$$

$$b \leftarrow b + \Delta b, \text{ where } \Delta b \propto -\frac{\partial l}{\partial \widehat{x}}$$

$$u + (b + \Delta b) - \mathbb{E}[u + (b + \Delta b)] = u + b - \mathbb{E}[u + b]$$

즉, 업데이트 전후의 Normalization 결과가 같아서 loss값이 유지된다.

2. Towards Reducing Internal Covariate Shift

$$\widehat{x} = \mathrm{Norm}(x, \mathcal{X})$$

$$\frac{\partial \mathrm{Norm}(x, \mathcal{X})}{\partial x}, \text{ and } \frac{\partial \mathrm{Norm}(x, \mathcal{X})}{\partial \mathcal{X}}$$

E(x)와 b의 Dependency를 무시한다는 것은 위 식에서 뒷 부분을 무시한다는 것이다.
또한 이런 방식의 whitening은 계산하려면 너무 많은 연산량이 필요하다.

$$\mathrm{Cov}[x] = \mathbb{E}_{x \in \mathcal{X}}[xx^T] - \mathbb{E}[x]\mathbb{E}[x]^T$$

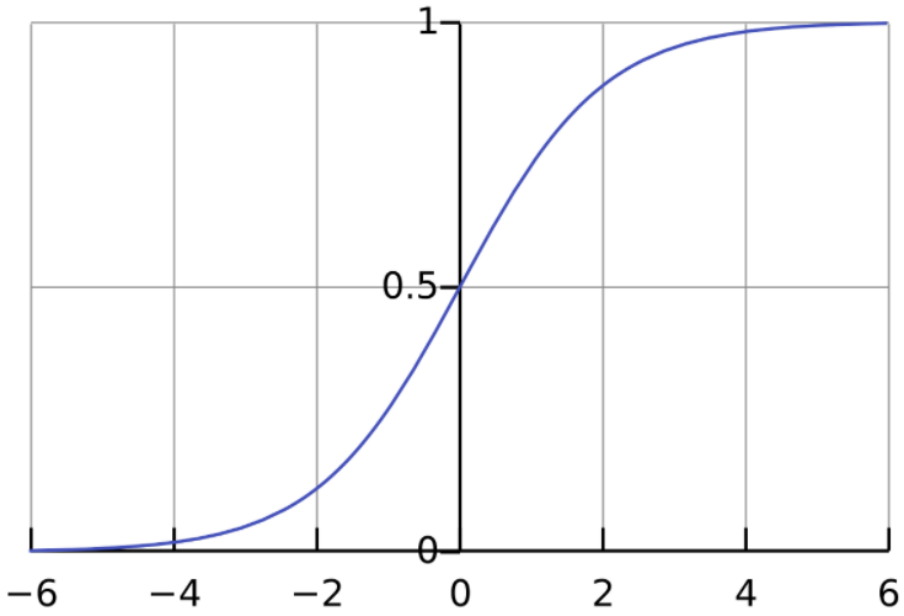
$$\mathrm{Cov}[x]^{-1/2}(x - \mathbb{E}[x])$$

Covariance Matrix를 구하려는 연산은 너무 복잡하다!

3. Normalization via Mini-Batch Statistics

Batch Normalization은 각각의 feature에 대해 정규화한다.
즉, 각 Feature들의 Mean, Variance가 0, 1이 되도록 정규화한다.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$



논문 표현: nonlinearity의 linear regime에 가둔다.
즉, 비선형성을 잃게 된다.

3. Normalization via Mini-Batch Statistics

normalize된 값을 scale, shift 해주는 학습을 통해 해결!

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

3. Normalization via Mini-Batch Statistics

Batch Normalization은 미니배치 단위로 정규화를 수행한다.

즉, mini-batch data로 layer의 입력을 normalize 해준다.
normalization에 사용된 통계값(statistics)이 모두 backpropagation에 참여할 수 있다.

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

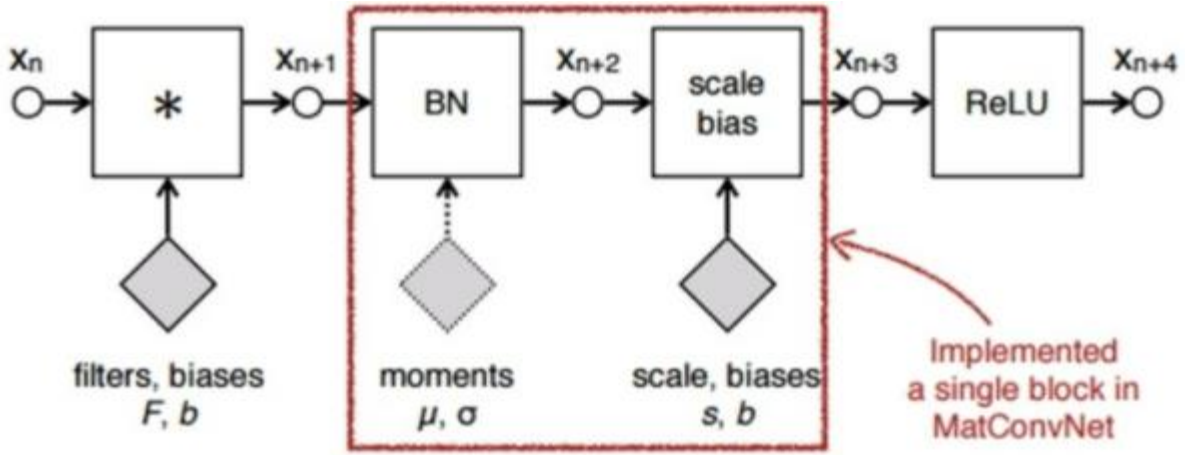
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

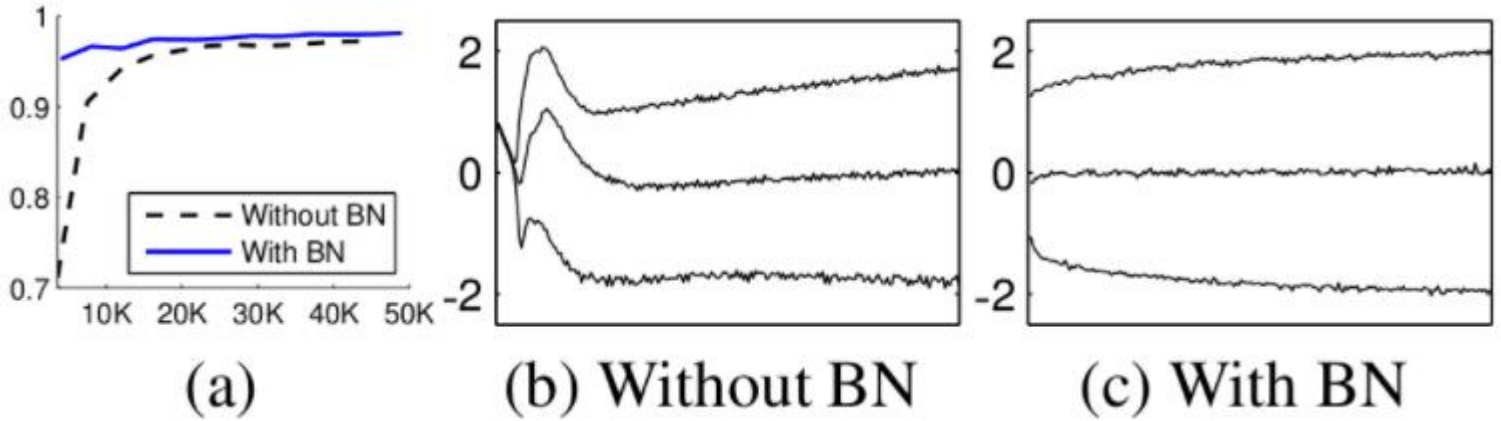
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



3. Normalization via Mini-Batch Statistics

1. 높은 Learning rate를 설정할 수 있다.
2. Regularization 효과가 있다.
3. Learning Rate Decay를 더 느리게 설정할 수 있다.

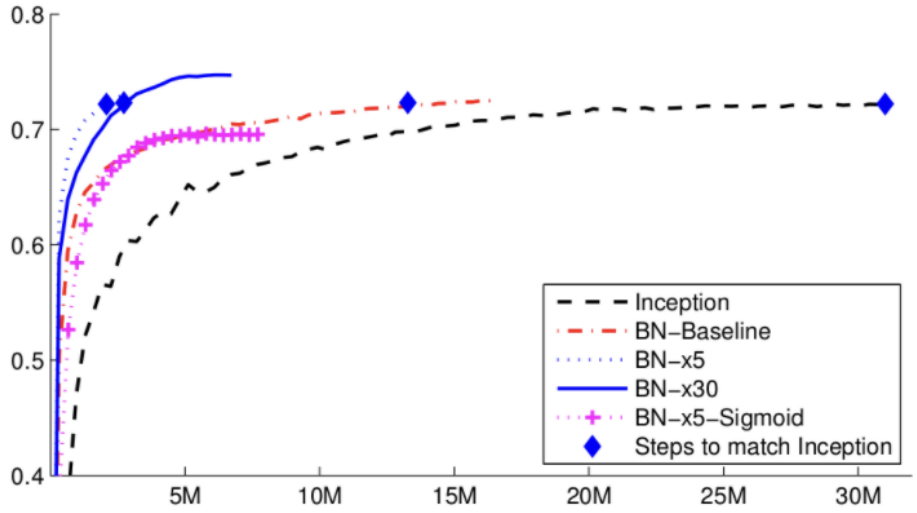
4. Experiment: MNIST



(a) test accuracy에 BN을 사용한 model이 더 빨리 수렴한다.
⇒ 학습이 빠르고 정확도가 더 높다

(b), (c) sigmoid에 입력된 분포가 BN을 사용했을 때 더 안정적이다.

4. Experiment: ImageNet



Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

(1) Inception: BN을 사용하지 않고 Learning Rate가 0.0015인 모델
(2) BN-Baseline: Inception 모델에서 BN을 적용한 모델
(3) BN-x5: BN-Baseline에서 Learning Rate가 5배인 모델
(4) BN_x30: Learning Rate가 30배인 모델
(5) BN_x5-Sigmoid: BN-x5와 동일하고 ReLU대신 Sigmoid를 사용한 모델

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***