2021 BoostCamp Paper Review

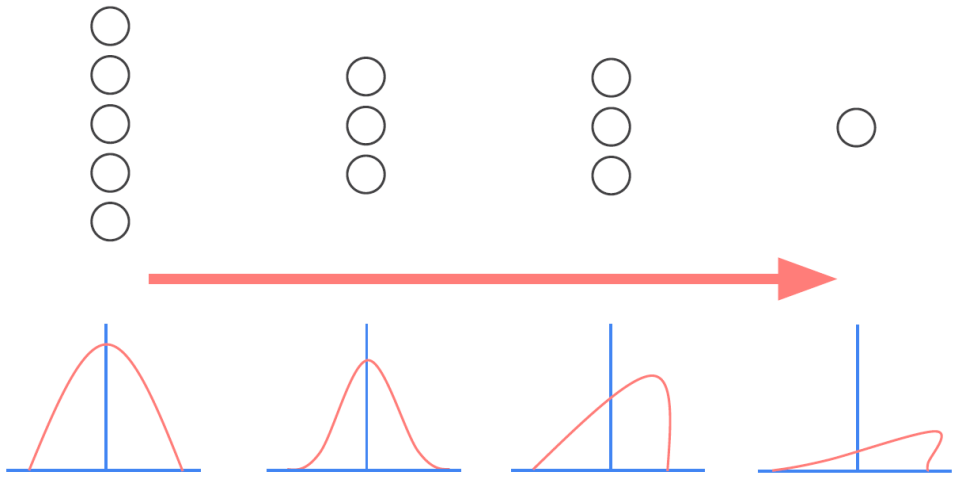
Batch Normalization (2015)

Presenter : Haram Lee

1. 배경

확률적 경사하강법(SGD)

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



- 각 레이어에 대한 입력이 이전의 모든 레이어에 대한 매개변수에 영향을 받기 때문에, **네트워크가 깊어질수록** 네트워크 매개변수에 대한 작은 변화도 증폭된다.

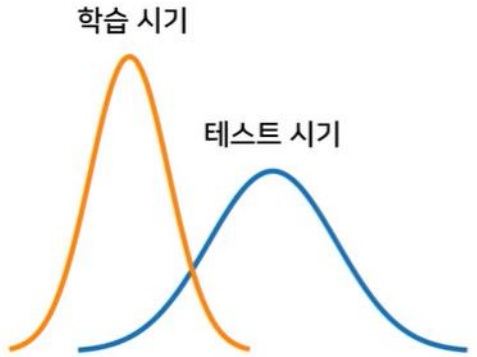
➔ **낮은 학습률(learning rate)을 사용**해야 한다.

➔ **가중치 초기화(weight initialization)에 신경**써야 한다.

1. 배경

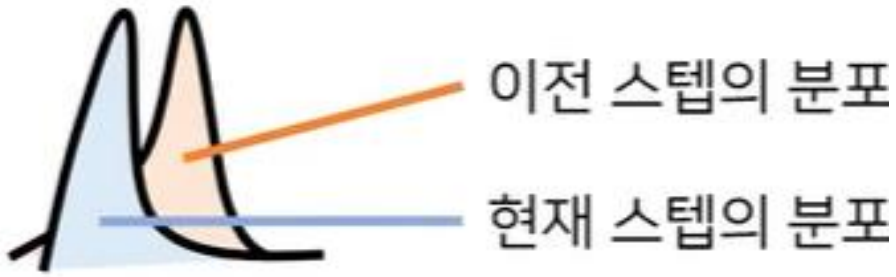
공변량 변화(Covariate Shift)

- 테스트 도메인의 데이터 분포가 학습 데이터 분포와 다른 것
- 일반적으로 domain adaptation (Jiang, 2008)을 통해 처리된다.



내부 공변량 변화(Internal Covariate Shift)

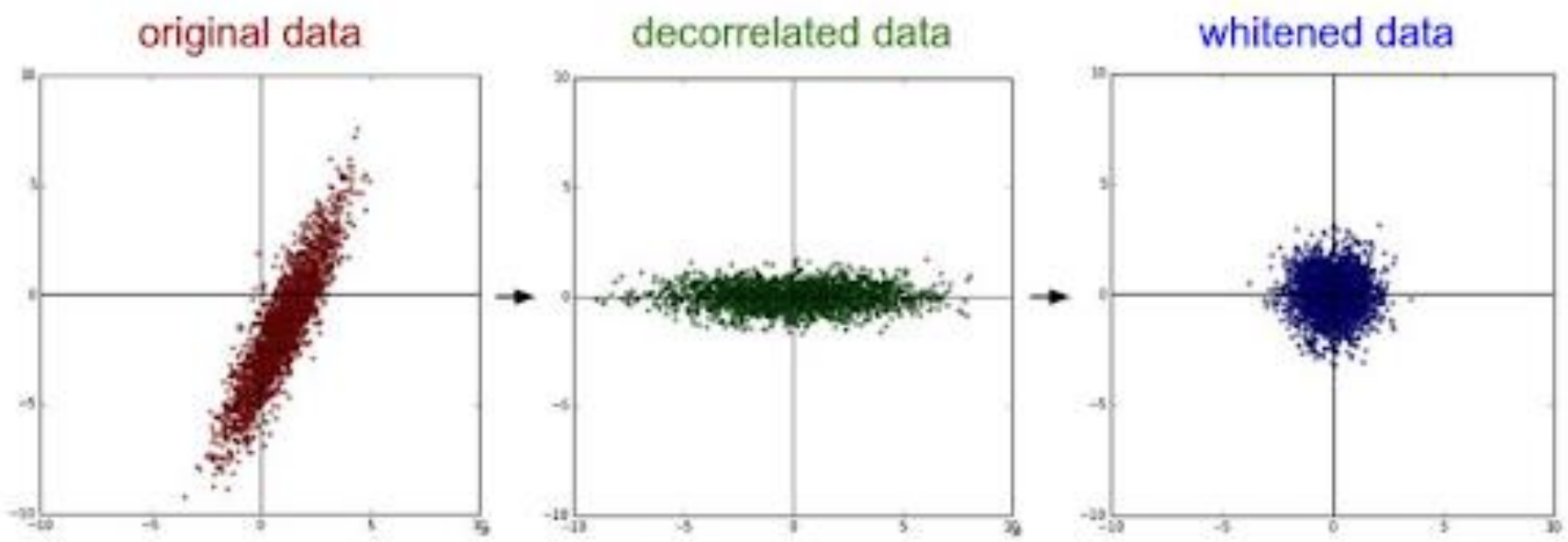
- 이전 레이어의 파라미터들이 변함으로써, 뒤의 레이어에 들어오는 입력의 분포가 학습 중에 계속 변하는 현상



2. 개념

화이트닝(Whitening)

- 평균이 0, 공분산이 단위행렬인 정규분포 형태의 데이터로 변환하는 것
- Covariance Matrix 구하는 계산량이 크고, 한번 업데이트마다 전체 데이터셋을 봐야한다.



(각 feature에 대해서 연관성을 없애는 작업 포함)

2. 개념

입력 표준화(Standardization)

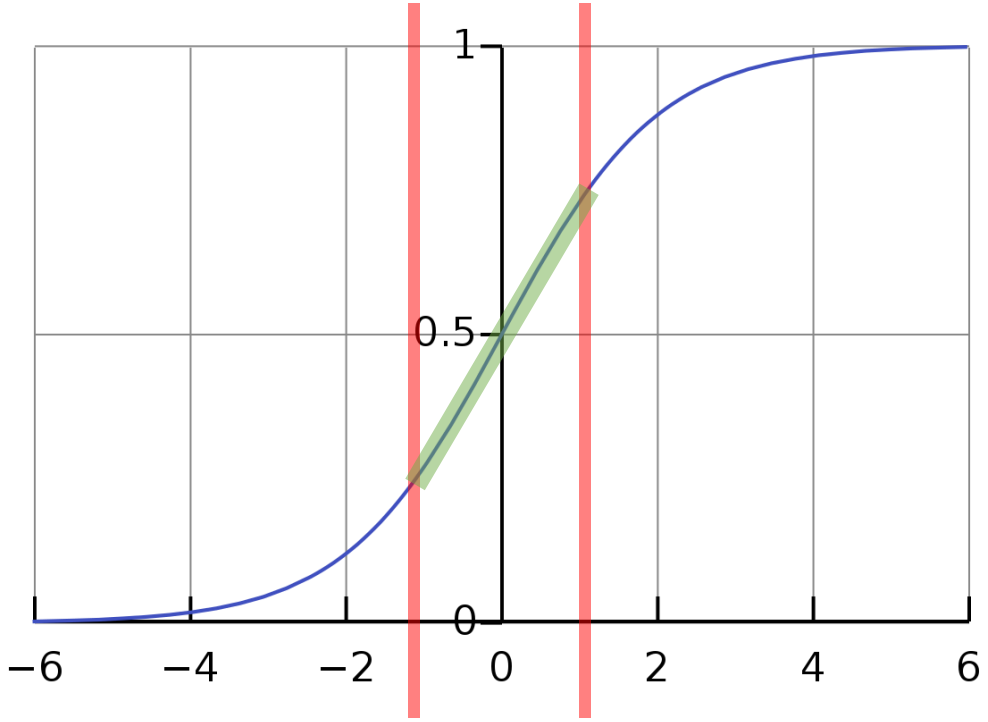
- 평균 0이고, 분산이 1인 정규분포를 따르도록 변환하는 것

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

2. 개념

비선형성의 선형체계(The linear regime of the nonlinearity)

- 예를 들어, 시그모이드(Sigmoid) 함수의 입력이 N(0,1)로 정규화될 때, 대부분의 입력에 대하여 매우 선형적으로 동작한다.



2. 개념

입력 표준화(Standardization)

- 평균 0이고, 분산이 1인 정규분포를 따르도록 변환하는 것

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

➔ **γ, β를 통해 비선형성(nonlinearity)을 유지할 수 있게 하고, 이는 학습을 통해 구한다.**

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

2. 개념

배치 정규화(Batch Normalization)

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

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

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

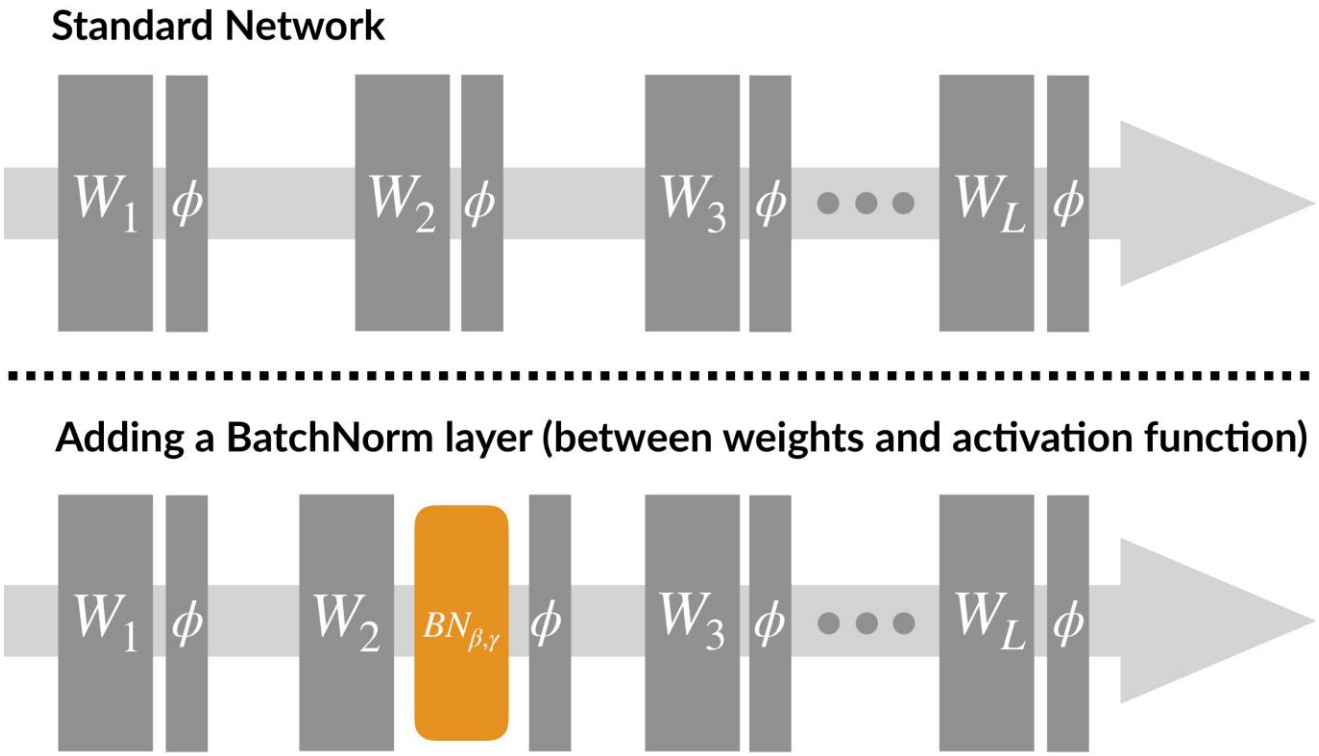
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

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

→ 새로운 파라미터 γ, β

3. 적용

배치 정규화(Batch Normalization)의 위치



$$z = g(Wu + b)$$

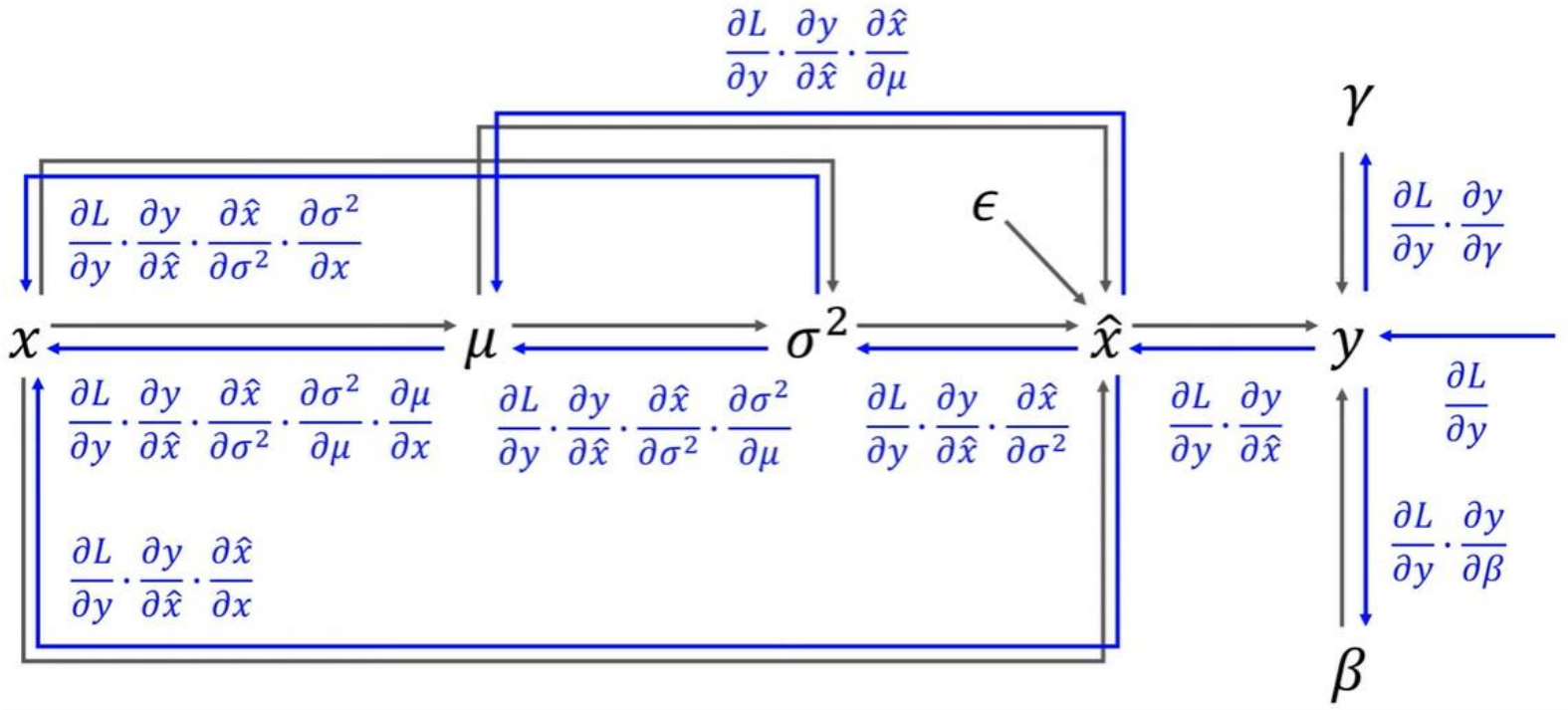
$$z = g(\mathrm{BN}(Wu))$$

출처 : https://blog.naver.com/tinz6461/221599876015

3. 적용

배치 정규화(Batch Normalization)의 역전파(Backpropagation)

- Chain rule을 통해 쉽게 구할 수 있다.



출처 : 배치 정규화(Batch Normalization) [꼼꼼한 딥러닝 논문 리뷰와 코드 실습]

3. 적용

추론(Inference)

- 배치 정규화의 파라미터(γ, β)들은 고정시킨다.
- (전체 학습 데이터셋) 여러 미니 배치의 평균의 기대값과 분산의 기대값을 이용한다.

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma^2_{\mathcal{B}}]$$

➔ 따라서, 추론 시 배치 정규화에서의 계산은 단순히 선형 변환(linear transform)이 된다.

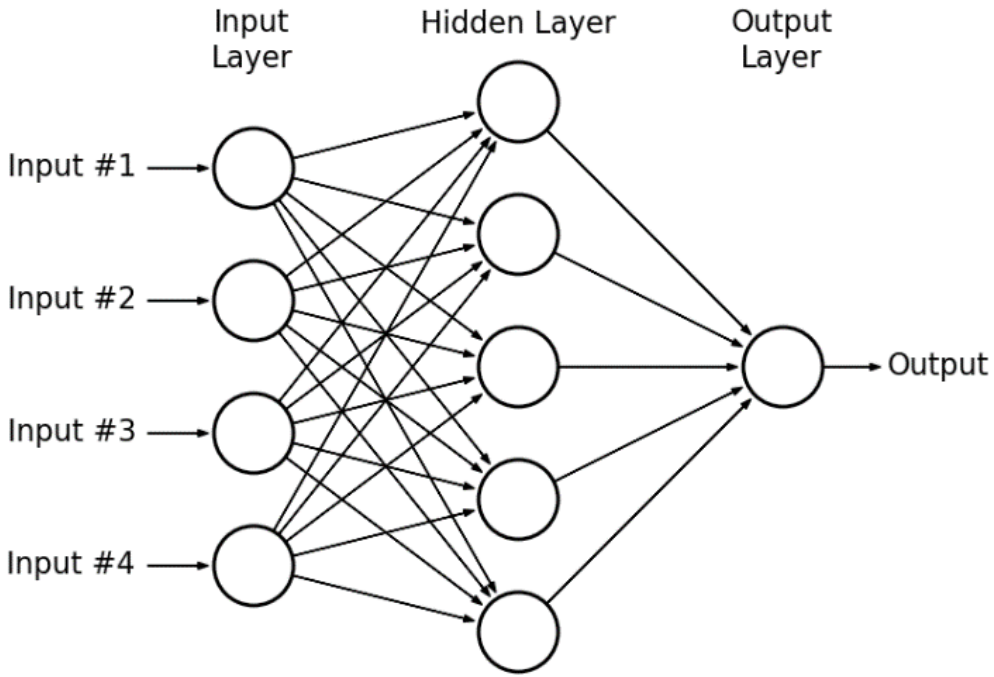
$$y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$$

3. 적용

배치 정규화(Batch Normalization) in MLP

- 각각의 유닛(hidden unit)에서 정규화가 일어난다.

- fully-connected networks
  - x : N x D
  - $\mu, \sigma, \gamma, \beta$: 1 x D

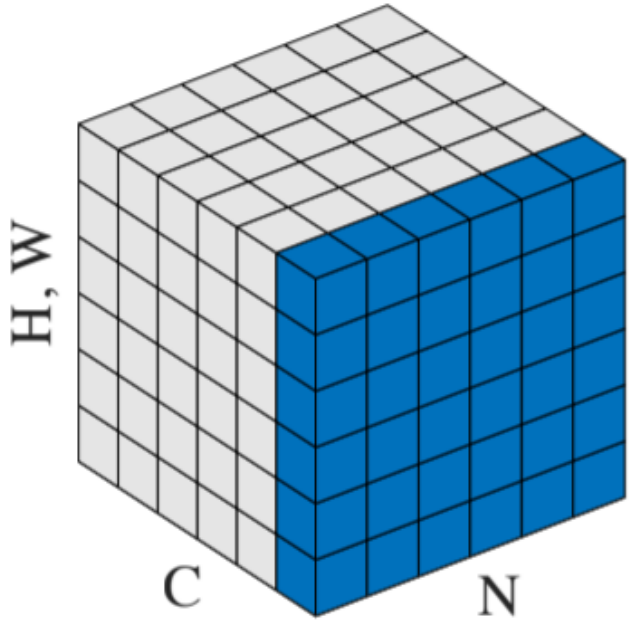


3. 적용

배치 정규화(Batch Normalization) in CNN

- 각각의 채널(channel)에서 정규화가 일어난다.

- convolutional networks
  - x : N x C x H x W
  - $\mu, \sigma, \gamma, \beta$: 1 x C x 1 x 1



3. 적용

배치 정규화(Batch Normalization) in PyTorch

- https://pytorch.org/docs/stable/nn.html#normalization-layers

Normalization Layers

`nn.BatchNorm1d`	Applies Batch Normalization over a 2D or 3D input (a mini-batch of 1D inputs with optional additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .
`nn.BatchNorm2d`	Applies Batch Normalization over a 4D input (a mini-batch of 2D inputs with additional channel dimension) as described in the paper Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift .

4. 분석

배치 정규화(Batch Normalization)의 효과

- 높은 learning rate를 사용할 수 있다. (= parameter scale에 대한 민감도 감소)

$$\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$$

$$\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial \mathrm{u}} = \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial \mathrm{u}}$$

$$\frac{\partial \mathrm{BN}((aW)\mathrm{u})}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial \mathrm{BN}(W\mathrm{u})}{\partial W}$$

- 하나의 학습 데이터에 대한 결정적인 값을 생성하지 않으므로, 네트워크의 일반화에 유리하다.
- 배치 정규화 된 네트워크에서는 Dropout을 제거하거나 강도를 줄여서 사용된다.

5. 실험 결과

Inception-v1을 변형한 모델로 실험

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	double #3×3 reduce	double #3×3	Pool +proj
convolution*	7×7/2	112×112×64	1						
max pool	3×3/2	56×56×64	0						
convolution	3×3/1	56×56×192	1		64	192			
max pool	3×3/2	28×28×192	0						
inception (3a)		28×28×256	3	64	64	64	64	96	avg + 32
inception (3b)		28×28×320	3	64	64	96	64	96	avg + 64
inception (3c)	stride 2	28×28×576	3	0	128	160	64	96	max + pass through
inception (4a)		14×14×576	3	224	64	96	96	128	avg + 128
inception (4b)		14×14×576	3	192	96	128	96	128	avg + 128
inception (4c)		14×14×576	3	160	128	160	128	160	avg + 128
inception (4d)		14×14×576	3	96	128	192	160	192	avg + 128
inception (4e)	stride 2	14×14×1024	3	0	128	192	192	256	max + pass through
inception (5a)		7×7×1024	3	352	192	320	160	224	avg + 128
inception (5b)		7×7×1024	3	352	192	320	192	224	max + 128
avg pool	7×7/1	1×1×1024	0						

ImageNet 성능 결과

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%***

6. 후속 연구

배치 정규화가 정말 잘 되는 이유

- How Does Batch Normalization Help Optimization?
- 배치 정규화는 의심할 여지없이 유용하지만, 왜 잘 되는지에 대한 이해가 부족하다.
- ICS를 줄이는 것은 상관이 없으며, 배치 정규화의 효과는 Smoothing이다.

다른 정규화 기법

- Layer Normalization
- Group Normalization
- Instance Normalization

Q & A