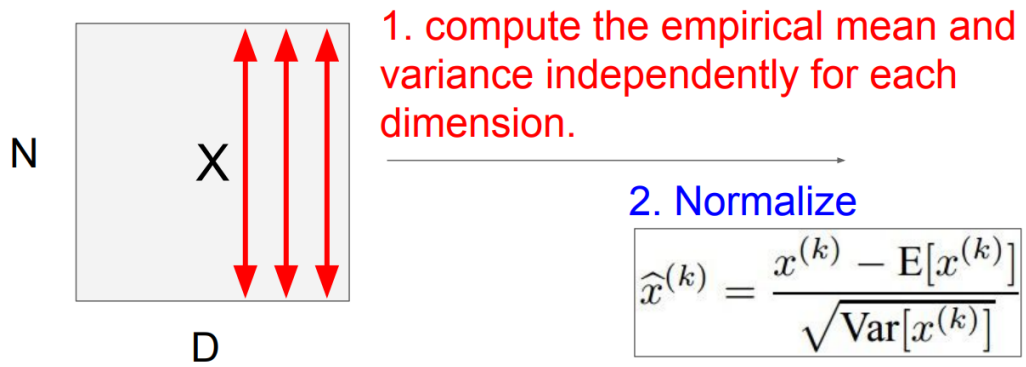
# Lecture 6-4

| | |
|---|---|
| Title | Training Neural Networks I |
| slide | http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf |

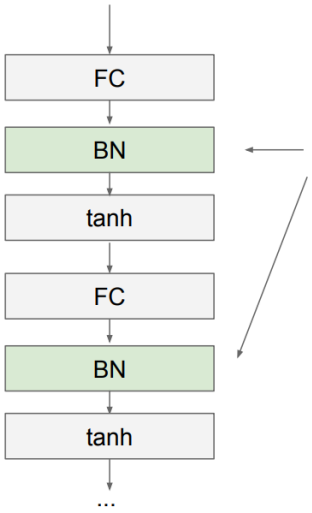
## Batch Normalization

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

gaussian(정규분포)의 범위로 activation을 유지시키는 것에 관련한 또 다른 방법은 Batch normalization이다. 어떤 레이어로부터 나온 Batch 단위 만큼의 activations이 있다고 했을 때, 우린 이 값들이 Unit gaussian이기를 원하므로, 이렇게 만들어 주자는 아이디어이다. 학습할 때마다 각 레이어에서 현재 Batch에서 계산한 mean과 variance를 이용해서 Normalization하여 모든 레이어가 Unit gaussian이 되도록 해주어, 결국 학습하는 동안 모든 레이어의 입력이 Unit gaussian이 되도록 한다. 평균과 분산을 "상수" 로 가지고 있으면 언제든지 미분이 가능하며, 따라서 Backprop이 가능하게 됩니다.




Batch당 N개의 학습 데이터가 있고, 각 데이터가 D차원이라고 해보자. 각 차원별로(feature element별로) 평균을 각각 구하고, 한 Batch내에서 Normalize 하여 FC나 Cov Layer직후에 넣어준다. 깊은 네트워크에서 각 레이어의 W가 지속적으로 곱해져서 Bad scaling effect가 발생했지만, Normalization은 그 bad effect를 상쇄시킨다.




Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

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

BN은 입력의 스케일만 살짝 조정해 주는 역할이기 때문에 FC와 Conv 어디에든 적용할 수 있다. Conv Layer에서 차이점이 있다면 Normalization을 차원마다 독립적으로 수행하는 것이 아니라, 같은 Activation Map의 같은 채널에 있는 요소들은 같이 Normalize 해준다는 것이다. 이는 Conv 특성 상 같은 방식으로 normalize 시켜야 하기 때문이다. 따라서 Conv Layer의 경우에는 Activation map(채널, Depth)마다 평균과 분산을 하나만 구하고 현재 Batch 에 있는 모든 데이터로 Normalize 해준다.

Normalize:

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

And then allow the network to squash the range if it wants to:

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

Note, the network can learn:

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$

$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

to recover the identity mapping.

BN의 한 가지 의문점은 FC layer를 거칠 때마다 매번 normalization를 해주는 것에 대해서, tanh의 입력이 정말 unit gaussian이기를 바라는 것일까?이다. normalization은 입력이 tanh의 linear한 영역에만 존재하도록 강제하는 것이다. 그렇게 되면 saturation이 전혀 일어나지 않게 되는데, 우리는 saturation 이 전혀 일어나지 않는 것 보다 "얼마나"saturation 을 조절할 수 있다면 더 좋다.

BN에서는 normalization 연산 이후에 scaling 연산을 추가하여, Unit gaussian으로 normalize 된 값들을 감마로 스케일링의 효과를, 베타로 이동의 효과를 준다. 이렇게 하면 normalized 된 값들을 다시 원상 복구 할 수 있도록 해준다. 네트워크가 값들을 원상 복구

하고 싶다면 "감마=분산", "베타=평균" 로 이동 시켜주면 된다. 이것은 네트워크가 데이터를 tanh에 얼마나 saturation 시킬지를 학습하기 때문에 우리는 유연성을 얻을 수 있다.

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

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

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

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

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

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization in a funny way, and slightly reduces the need for dropout, maybe

Batch normalization를 요약해 보면, 입력이 있고 우리는 모든 mini-batch마다 각각 평균과 분산을 계산하여 이를 통해 Normalize한 이후에 다시 추가적인 scaling, shifting factor를 사용한다. BN은 gradient의 흐름을 보다 원활하게 해주며 결국 더 학습이 더 잘되게(robust) 해준다. BN을 쓰면 learning rates를 더 키울 수도 있고 다양한 초기화 기법들도 사용해 볼 수 있다. 또한 BN은 regularization의 역할도 한다. 각 레이어의 출력은 해당 데이터 하나 뿐만 아니라 평균과 분산 때문에 batch 안에 존재하는 모든 데이터들에 영향을 받는다. 그렇기 때문에 각 레이어의 출력은 이제 오직 하나의 샘플에 대한 deterministic한 값이 아니게 되고, Batch 내의 모든 데이터가 입력으로 한 곳에 묶인다고 할 수 있다. 그러므로 더 이상 레이어의 출력은 deterministic하지 않고 조금씩 바뀌어 regularization effect를 준다.