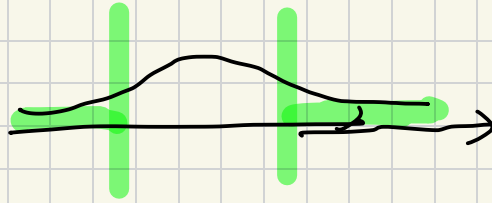


Activation function

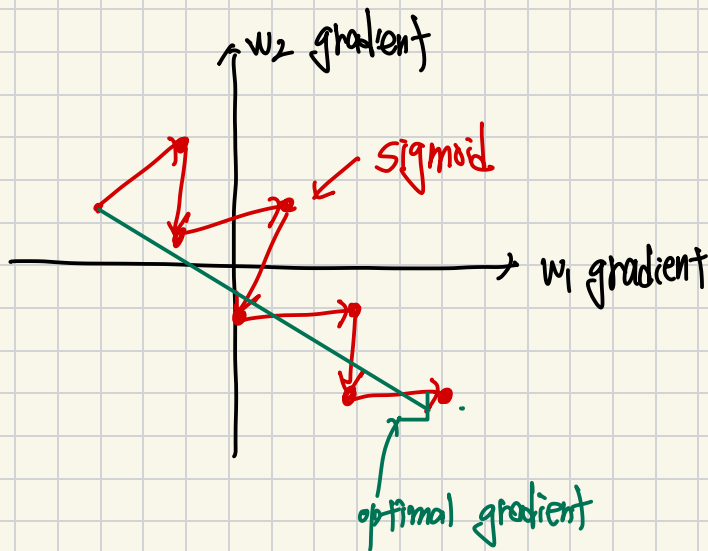
(1) sigmoid kill gradient $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x))$
non zero centered.

$\frac{\partial \sigma(x)}{\partial x} \Rightarrow$  거의 almost loss가 0으로 수렴하게 된다.

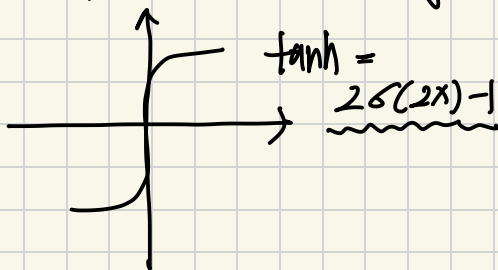
\Rightarrow 1보다 작은 gradient가 음(-)이기에 gradient가 0으로 수렴한다.

(f) Input이 모든 data가 양수인 경우.

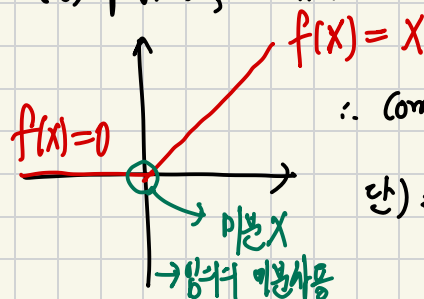
$$\frac{\partial \sigma}{\partial w} = \underbrace{\sigma(w^T w + b)(1 - \sigma(w^T w + b))}_{>0} \underbrace{x_i}_{>0} \Rightarrow \text{모두 양수. 결과가 양수만 나온다.}$$



(2) tanh (\Rightarrow also kill gradient)



(3) ReLU, $\max(0, x)$



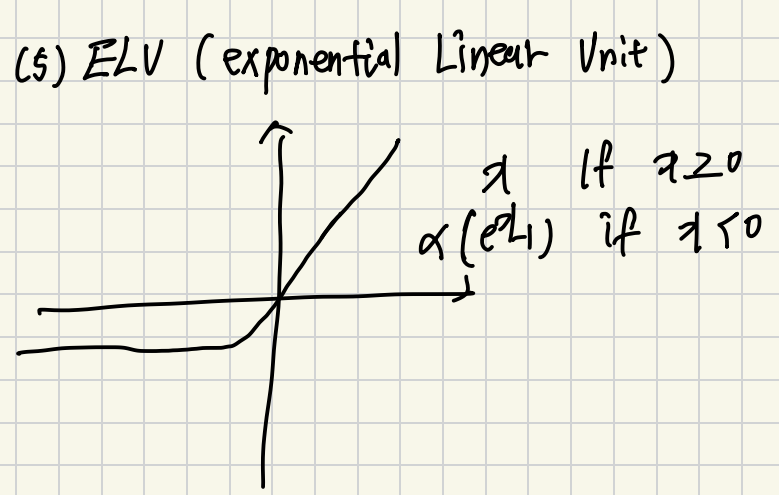
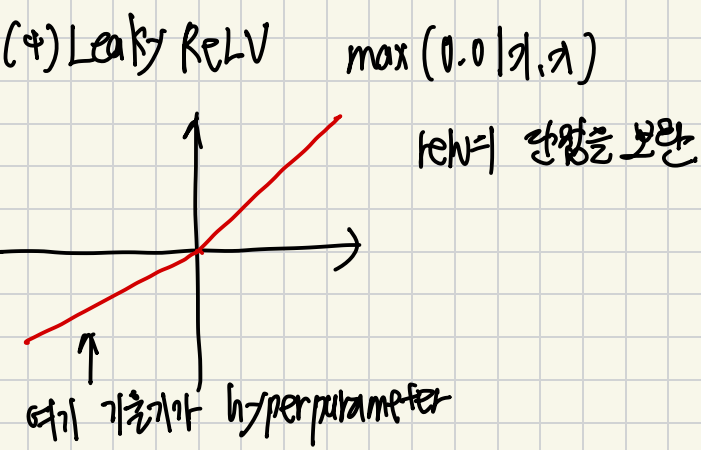
\therefore computation이 매우 효율적이다.

단) zero centered X

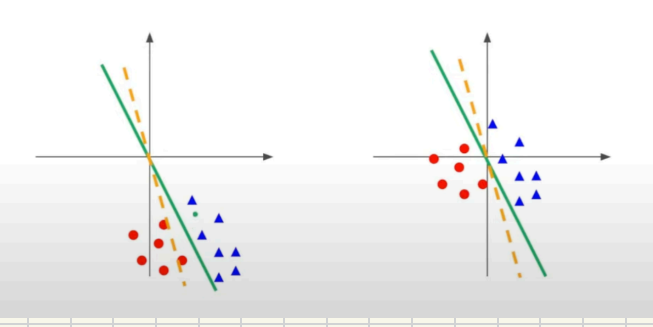
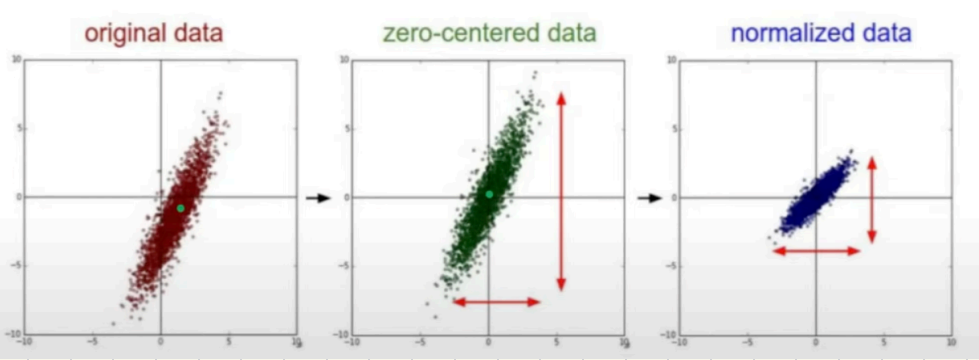
1) $w < 0$ 이 한번이라도 나오면

해당 w 는 계속 0이 나온다.

→ 가중치 초기화 need.

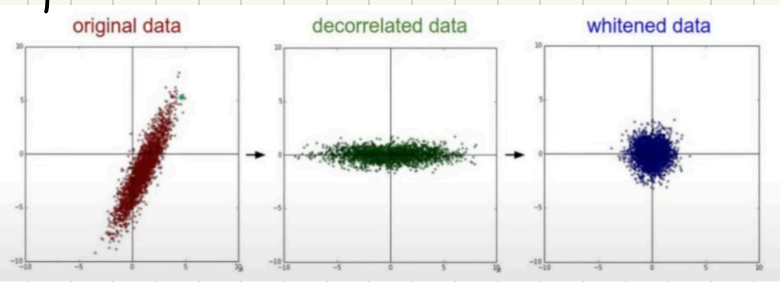


Data preprocessing & Augmentation



수행적으로 정규화가
 매우 중요하게 임하기때
 진행하면 된다.

<PCA>

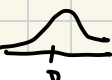


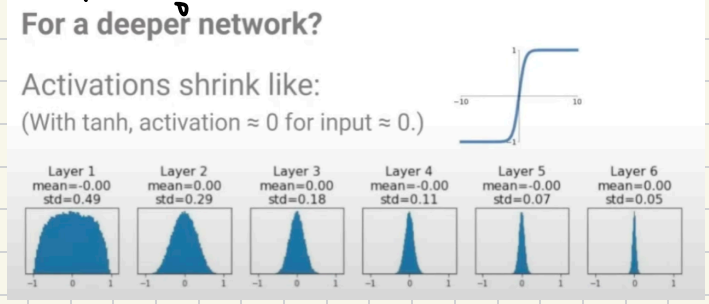
PCA는 몇개 PC 축으로 최적으로
 variance도 나눠주기 됩니다.

Data augmentation \Rightarrow data 증강시키기. (의미 자체가 바뀌지 않으면서 pixel은 변형)

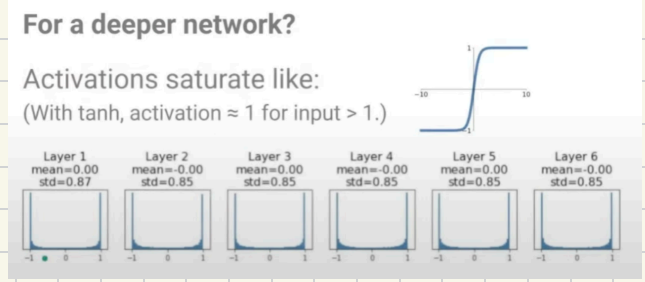
- 1) Flip
- 2) RandomCrops
- 3) color jitter (색조도 change) RGB \Rightarrow HSV

Weight Initialize.

초기화 =  랜덤 value x 0.01 정도 사용.



0.01이 아니라 어쨌든 0.9 정도 Constant로

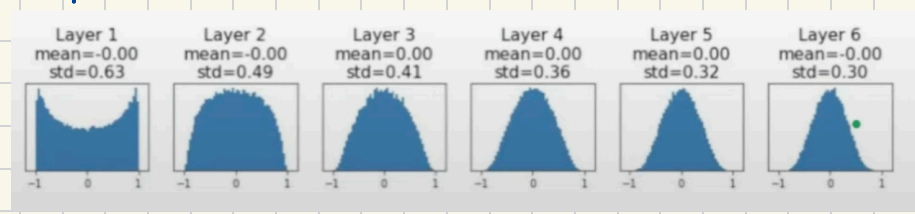


점진적 gradient이 x를 향해서 쏠리면서 output 값이 0으로 수렴. || 반대로 값이 극단으로 가서 gradient = 0이 나온다.

다른 권장한 constant가 무엇이냐고

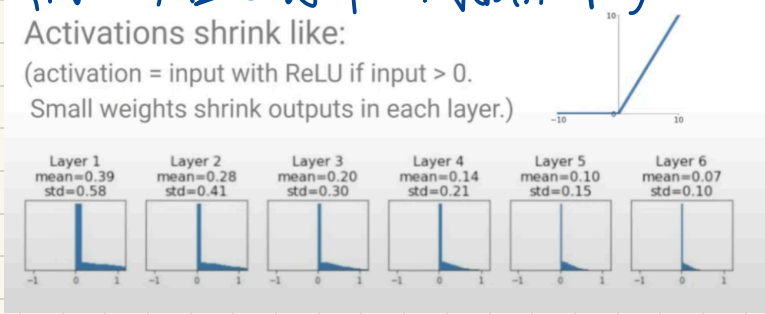
Xavier initialization : `np.random.rand()` / `np.sqrt(d_in)`

input channel의 크기 제곱근으로 나눠주자!

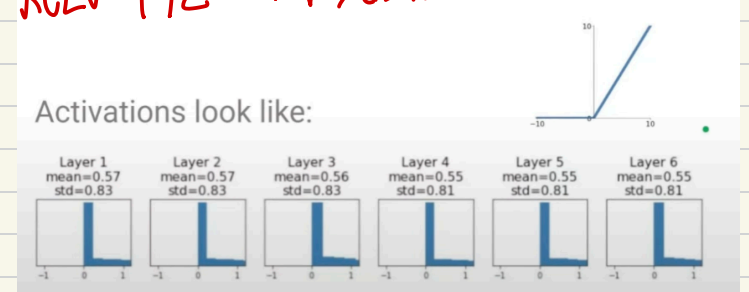


Input의 variance와 output의 variance를 어느정도 유지해준다.

<ReLU에서도 동등하게 $1/\sqrt{d_{in}}$ 해주세요

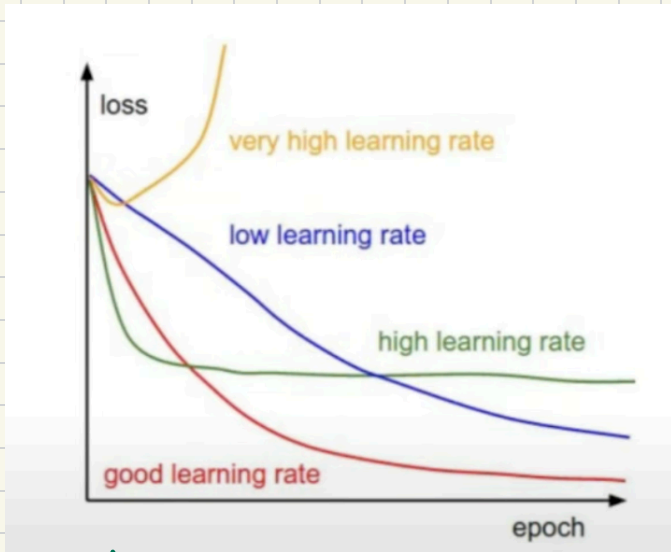


ReLU에서는 $1/\sqrt{2d_{in}}$ 을 사용



Learning Rate Scheduling

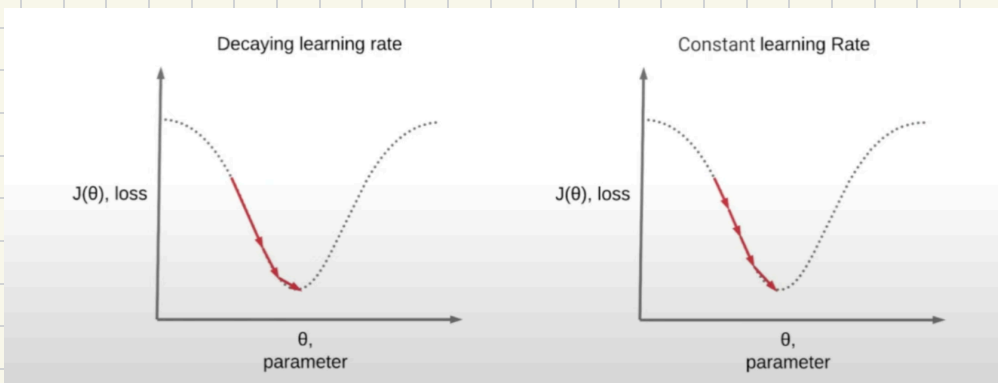
실험적으로 lr을 찾아야한다.



⇒ high learning rate의 optimum 이라고 판단할 수 있겠다.

lr을 정해 줄 수 없다. ⇒ 실험을 통해서.

Learning rate decay



코기이 lr을 크게 유지하여 점점 감소시키는 방법 또한 사용이 가능하다.