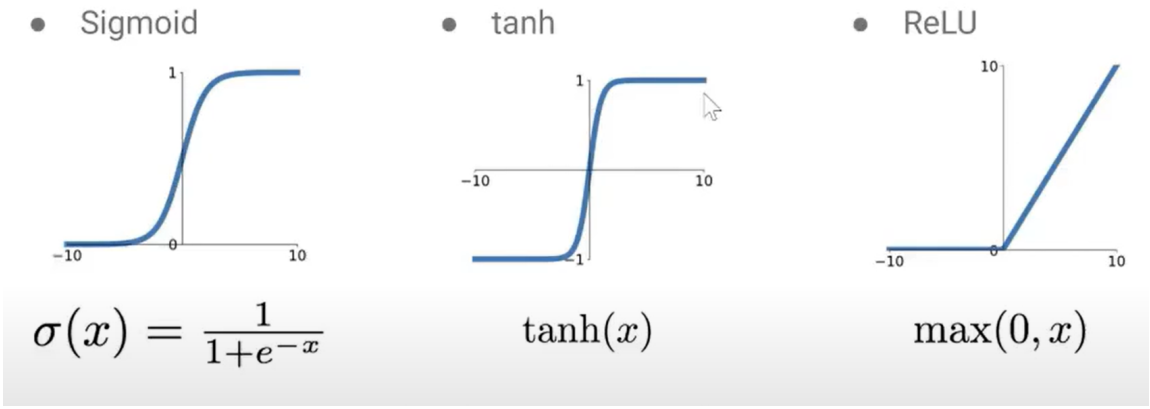
Lecture 6

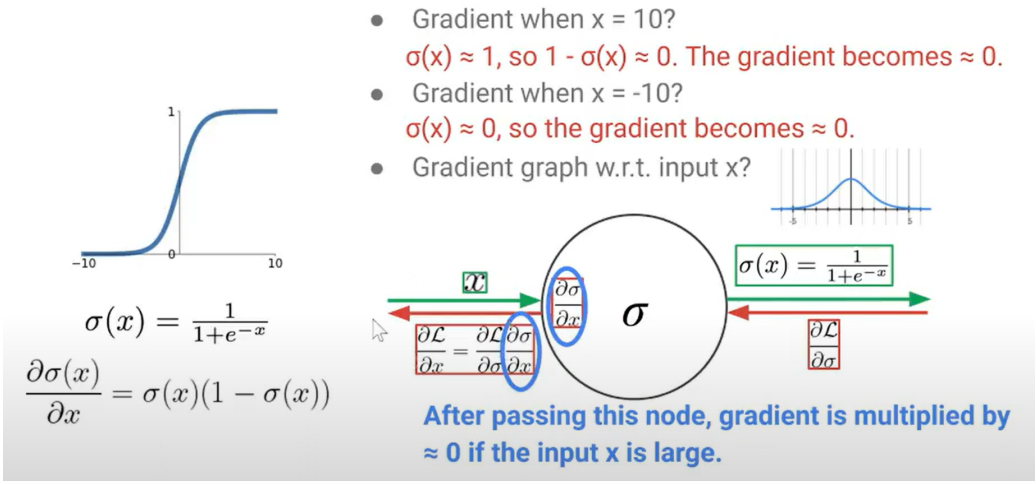
Activation functions

- non-linearity를 허용해주는 activation function



- 종류별 설명
  - sigmoid
    - gradient를 죽임

      upstream gradient와 상관없이 local에서 자꾸 0에 가까워짐 → 더 이상 update 안됨

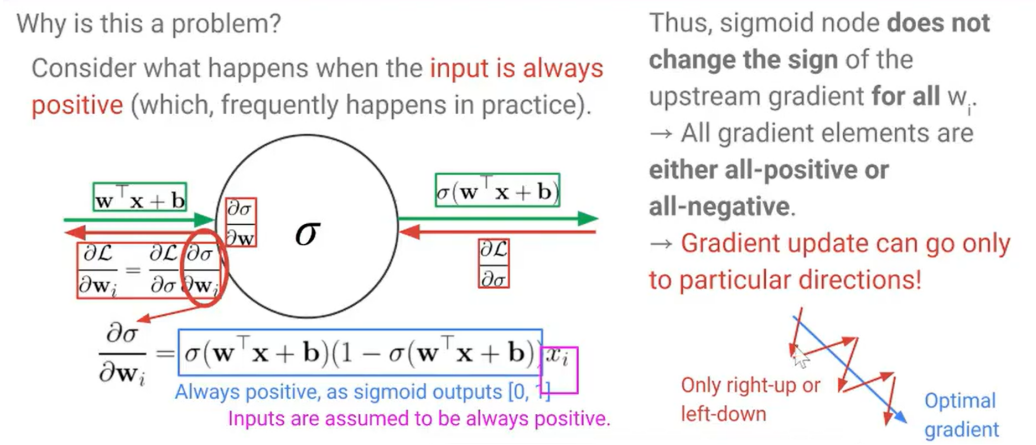


- 기댓값이 [0,1] 사이라 zero-centered하지 않음

  gradient가 모두 양수면 양수, 음수면 음수 → gradient가 갈 수 있는 방향에 제약이 생김

  update를 이상적인 방향으로 진행하지 못하고 시간과 계산량이 더 많아짐

  killing gradient보다는 덜 심각한 문제 (미니 배치로 할 때 단점들이 약간 보완됨)



- 컴퓨터 연산이 많이 걸림

◦ Tanh function

- zero-centered 문제 해결
- 하지만 killing gradient 문제 해결 X

  tanh function도 sigmoid function의 일부

◦ ReLU (Rectified Linear Unit)

- 양수인 구간과 음수인 구간으로 나뉨 → plus인 구간에서 saturation 안 일어남

- 계산량이 적고 training이 빠름
- zero-centered ouput이 안되어 항상 양수만 나옴
- x=0일 때 미분 불가능
- 처음 output이 음수면 gradient가 0이라서 update가 안됨

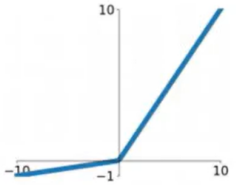
◦ Leaky ReLU

Reason to use

- Does not saturate (**in all regions**).
- Computationally very efficient.
- Converges much faster than sigmoid/tanh in practice.
- No Dead ReLU problem

Reason to NOT use

- An additional hyperparameter (slope where x < 0)



$$\max(0.01x, x)$$

◦ ELU

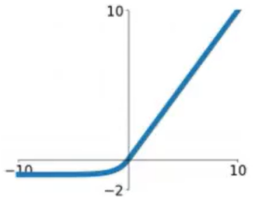
- zero-centered output을 디자인하고 싶어 만듦

Reason to use

- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared to (Leaky) ReLU adds some robustness to noise.

Reason to NOT use

- exp() is **computationally expensive**.

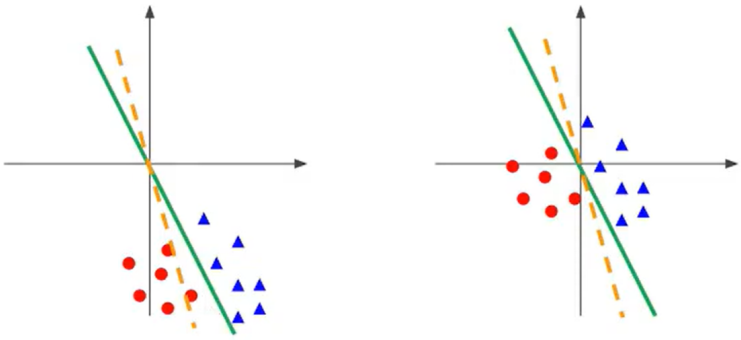


$$\begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

Data Preprocessing

- Zero-centering

- Classification becomes less sensitive to small changes in weights.
  - See the slope change in the figure (green to orange).
- Easier to optimize.



  - 방법 : PCA & Whitening
    - PCA를 이미지 데이터에서 픽셀 단위로는 사용하지 않음
    - whitening
- Normalization
- 실제로 데이터 전처리 예시

- Subtract the mean image (e.g., AlexNet)
  - mean image = [32,32,3] array
- Subtract per-channel mean (e.g., VGGNet)
  - mean along each channel = 3 numbers
- Subtract per-channel mean and Divide by per-channel std (e.g., ResNet)
  - mean along each channel = 3 numbers
- PCA and whitening (e.g., YouTube 8M)
  - Applied for video classification
  - PCA is less common on pixel space, but widely used in a feature space.

We will learn more about these models later in this course.

Data Augmentation

- Data Augmentation
  - 모델이 다각도의 데이터를 학습할 수 있도록 하기 위해
  - 종류

- horizontal flip
- random crop
- scaling : 이미지 크기를 확대 혹은 축소시킴
- color jitter : 이미지의 색깔을 조금 다르게 radomized

Weight Initialization

- Small Gaussian Random
  - 작은 값이 곱해진 normal distribution 이용
  - activation function이 갈수록 0에 가까워짐

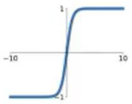
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for d_in, d_out in zip(dims[:-1], dims[1:]):
  W = 0.01 * np.random.randn(d_in, d_out)
  x = np.tanh(x.dot(W))
  hs.append(x)
```

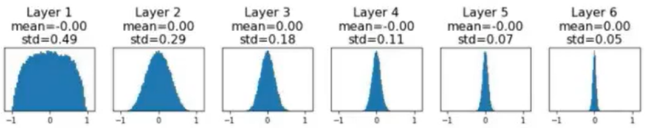
Standard normal distribution, multiplied by a small constant (0.01).

Widely used in traditional shallow network, and worked okay.

For a deeper network?

Activations shrink like:
(With tanh, activation ≈ 0 for input ≈ 0.)





Gradient ∂ℒ/∂W?

∂tanh(Wx+b)/∂W = tanh(Wx+b)′ × x

As x ≈ 0, all gradients will be ≈ 0.
No learning! 😨

- Large Gaussian Random
  - 큰 값을 곱함
  - activation 값이 거의 1에 가까워짐 → 미분한 값이 0이 되어 학습 X

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for d_in, d_out in zip(dims[:-1], dims[1:]):
  W = 0.5 * np.random.randn(d_in, d_out)
  x = np.tanh(x.dot(W))
  hs.append(x)
```

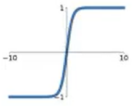
Okay, what if we increase the scale?

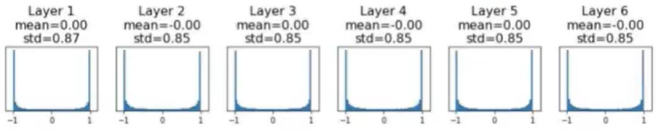
Since too small weights made all activations to zero, this might solve the issue!

For a deeper network?

Activations saturate like:

(With tanh, activation ≈ 1 for input > 1.)





Gradient $\partial \mathcal{L}/\partial W$?

$\partial \tanh(Wx+b)/\partial W = \tanh(Wx+b)' \times x$

$= (1 - \tanh^2(Wx+b)) \times x$

As $\tanh^2(Wx+b) \approx 1$, all gradients will be ≈ 0. No learning again! 😨

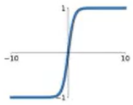
- Xavier Initialization
  - input 크기에 sqrt를 씌워 넣은 것
  - 적당히 분포하는 activation 값들

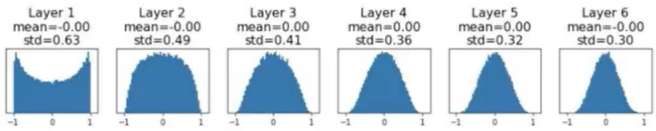
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for d_in, d_out in zip(dims[:-1], dims[1:]):
  W = np.random.randn(d_in, d_out)/ np.sqrt(d_in)
  x = np.tanh(x.dot(W))
  hs.append(x)
```

Just right scale, where activations are nicely scale for all layers.

For a deeper network?

Activations look like:





For conv layers?

$d\_in = F^2C$

F: Filter size

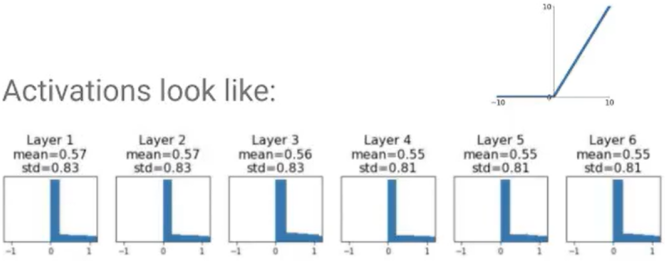
C: Number of channels

- Kaiming/MSRA Initalization for ReLU

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for d_in, d_out in zip(dims[:-1], dims[1:]):
  W = np.random.randn(d_in, d_out)/ np.sqrt(2/d_in)
  x = np.maximum(0, x.dot(W))
  hs.append(x)
```

Just right scale, where activations are nicely scale for all layers.

Activations look like:



- 초기화와 관련된 논문들

Proper initialization is an active area of research...

- Understanding the difficulty of training deep feedforward neural networks (AISTATS 2010)
- Exact solutions to the nonlinear dynamics of learning in deep linear neural networks (ICLR 2014)
- Random walk initialization for training very deep feedforward networks (ICLR 2015)
- Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification (ICCV 2015)
- Data-dependent Initializations of Convolutional Neural Networks (ICLR 2016)
- All you need is a good init (CVPR 2017)
- Fixup Initialization: Residual Learning Without Normalization (ICLR 2019)
- The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (ICLR 2019)