모두의연구소

지난 시간 리뷰

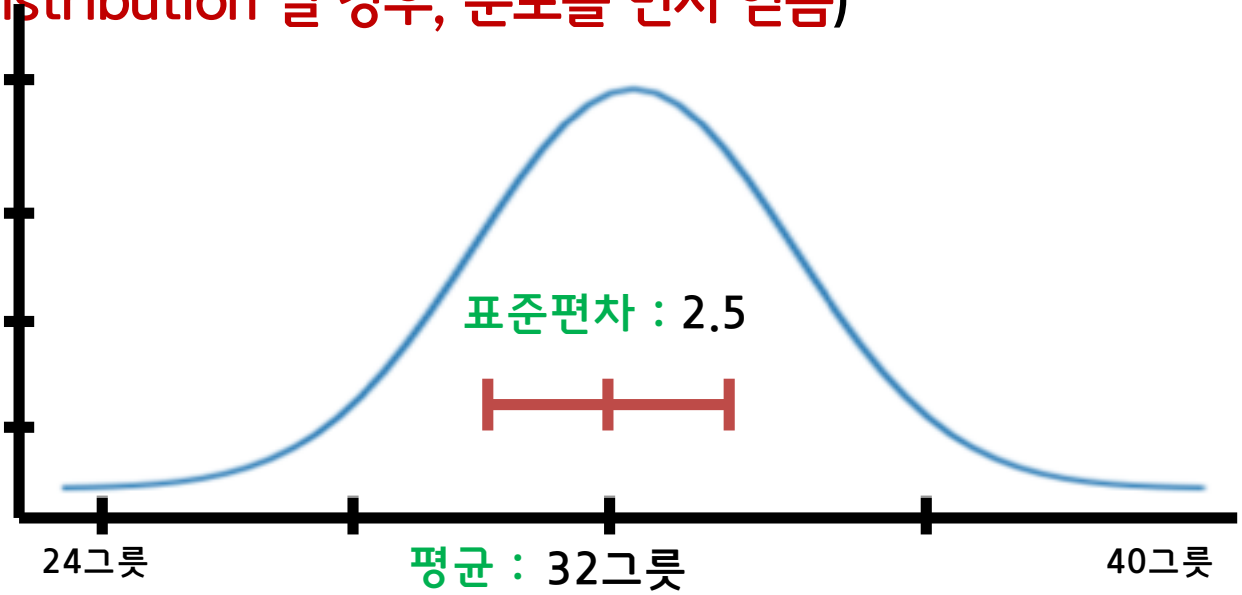
모두의연구소
박은수 책임연구원

Probability vs Likelihood

모두의연구소

Probability

- ‘딥러닝 면옥’ 체인점들의 하루 비빔냉면 판매량 (normal distribution 일 경우, 분포를 먼저 얻음)



모두의연구소

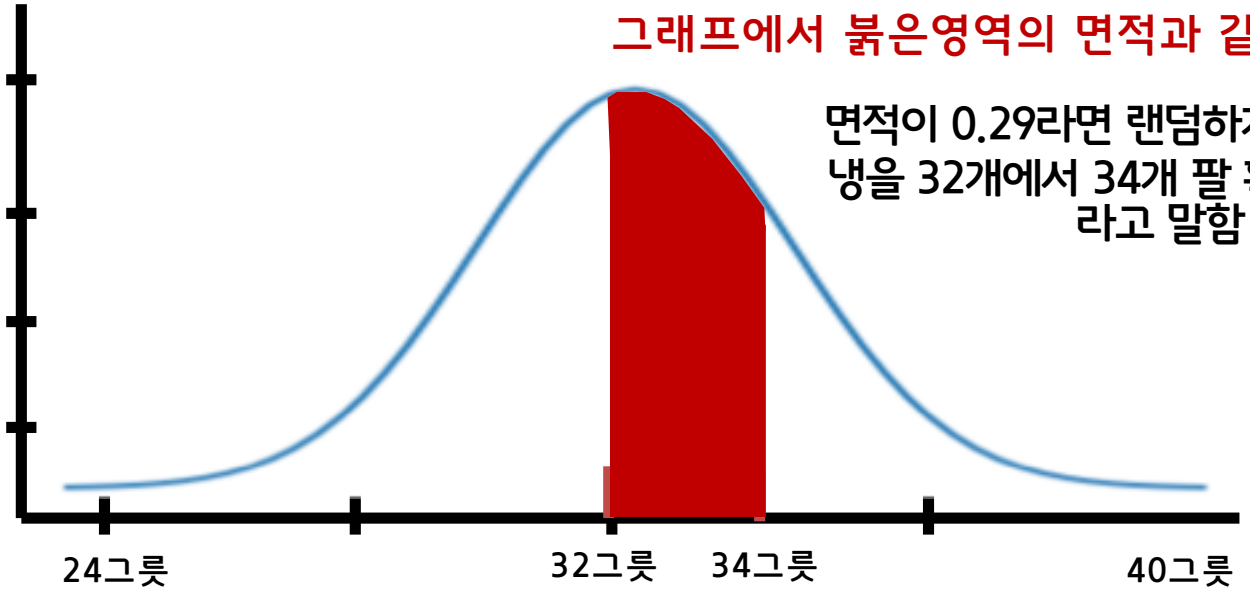
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

모두의연구소

Probability

랜덤하게 선택한 체인점이 하루에 비빔면을 32개에서 34개 판매할 확률 :



From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

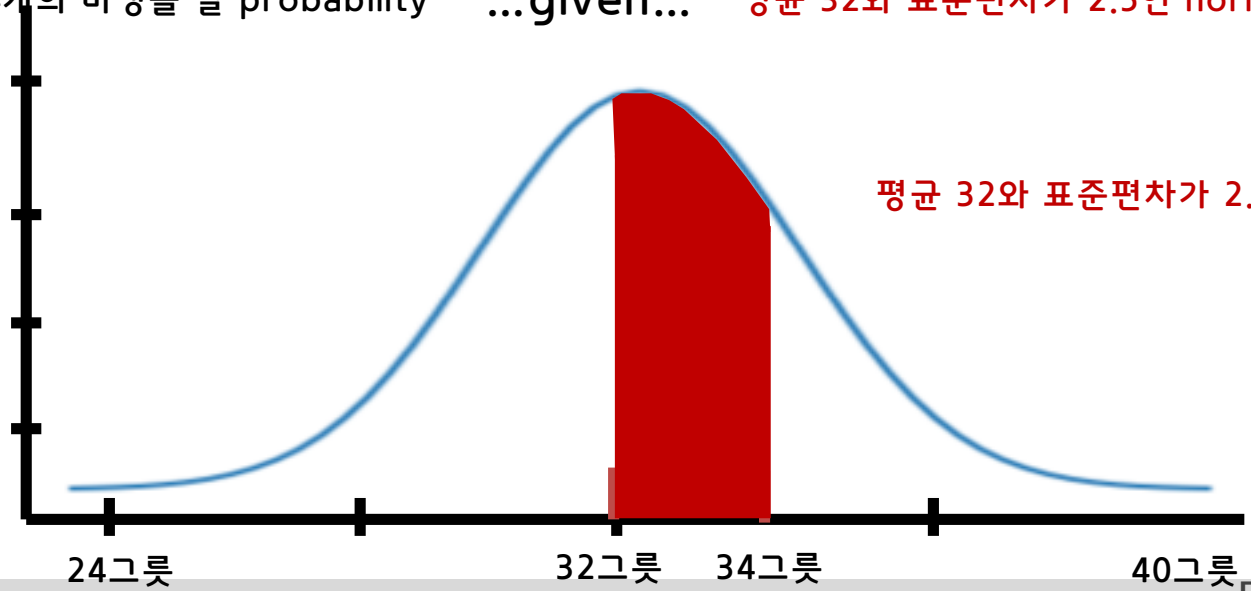
Probability vs Likelihood

Probability

수학적으로 표현해 보면

$$P(32\text{개에서 }34\text{개의 비냉을 팜} \mid \boldsymbol{mean = 32\ and\ deviation = 2.5})$$

하루 32개에서 34개의 비냉을 팔 probability ...given... 평균 32와 표준편차가 2.5인 normal 분포



평균 32와 표준편차가 2.5인 normal 분포

From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

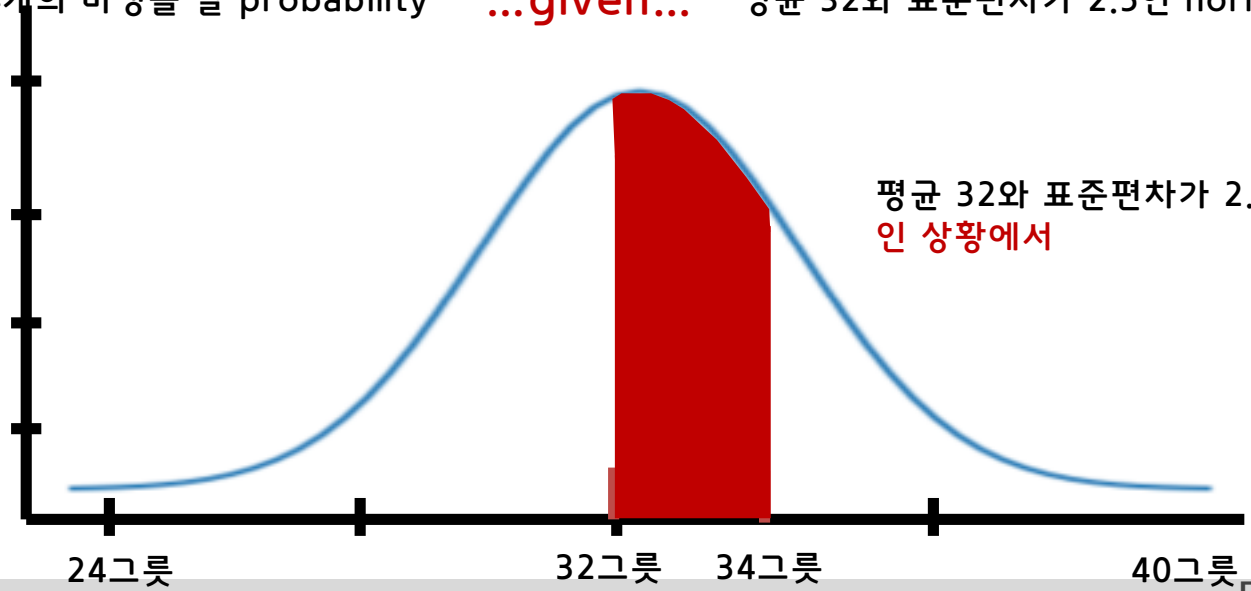
Probability vs Likelihood

Probability

수학적으로 표현해 보면

$$P(32\text{개에서 } 34\text{개의 비냉을 팜} \mid \boldsymbol{mean = 32 \; and \; deviation = 2.5})$$

하루 32개에서 34개의 비냉을 팔 probability ...given... 평균 32와 표준편차가 2.5인 normal 분포



평균 32와 표준편차가 2.5인 normal 분포
인 상황에서

From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

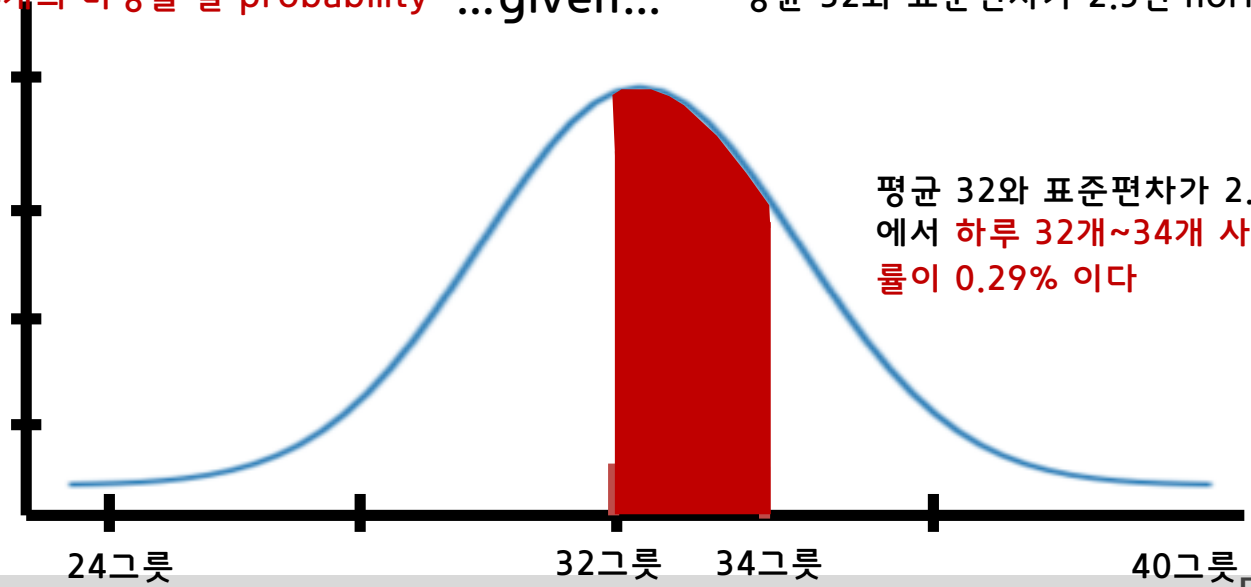
Probability vs Likelihood

Probability

수학적으로 표현해 보면

$$P(\text{32개에서 34개의 비냉을 팜} \mid mean = 32 \; and \; deviation = 2.5)$$

하루 32개에서 34개의 비냉을 팔 probability ...given... 평균 32와 표준편차가 2.5인 normal 분포



평균 32와 표준편차가 2.5인 normal 분포에서 하루 32개~34개 사이의 비냉을 팔 확률이 0.29% 이다

From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

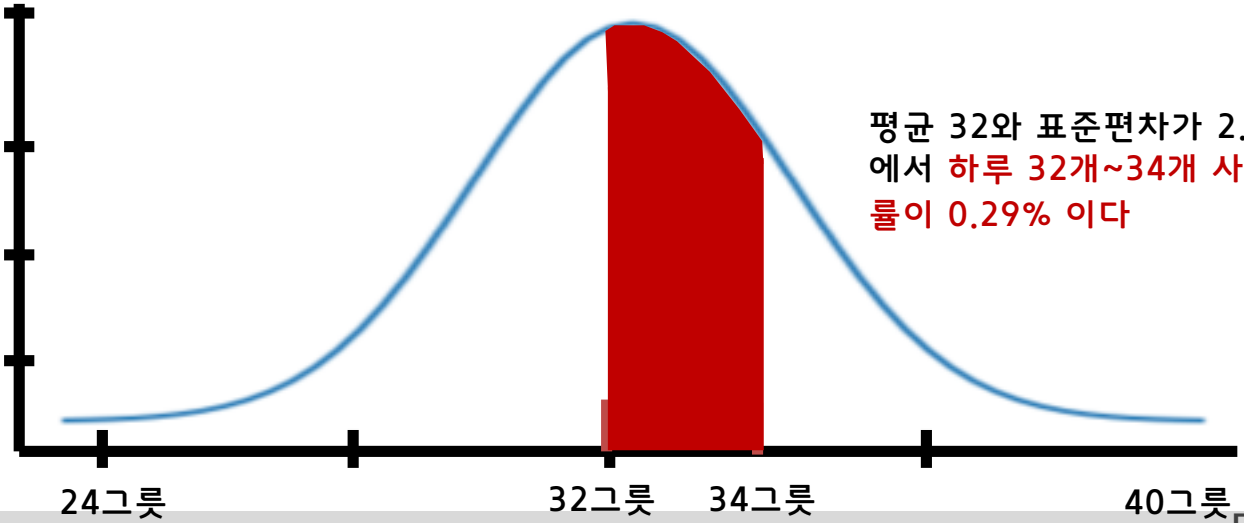
Probability vs Likelihood

Probability

수학적으로 표현해 보면

$$P(32\text{개에서 }34\text{개의 비냉을 팜} \mid \boldsymbol{mean = 32\ and\ deviation = 2.5})$$

우리가 다른 비냉판매 수에 관심이 있다면 변경 가능한 부분



평균 32와 표준편차가 2.5인 normal 분포에서 하루 32개~34개 사이의 비냉을 팔 확률이 0.29% 이다

From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

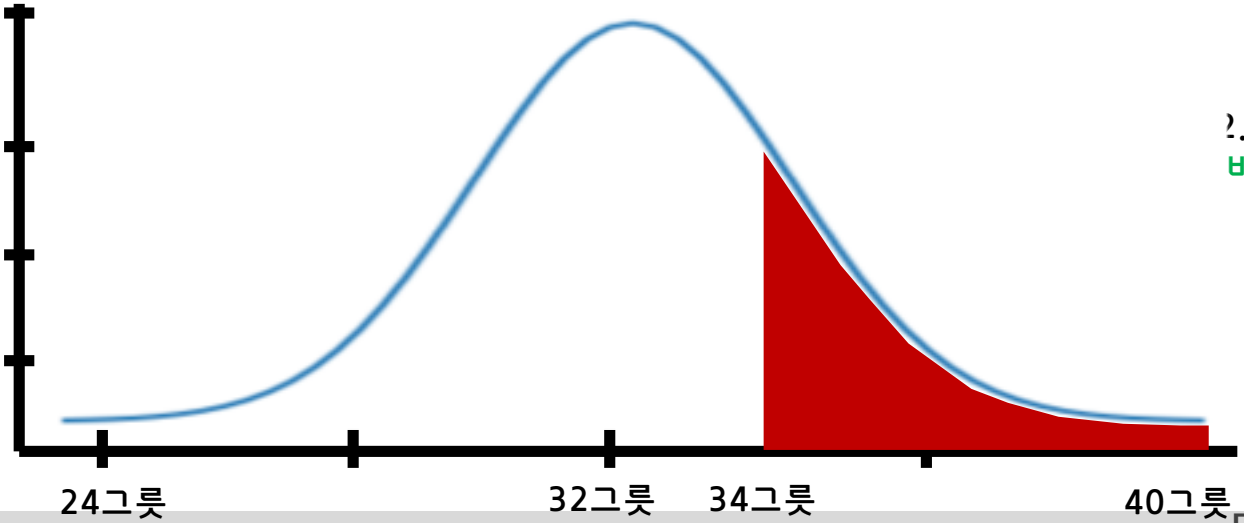
모두의연구소

Probability

수학적으로 표현해 보면

$$P(\text{비냉판매 수} > 34 \mid \underset{\text{고정}}{mean = 32 \; and \; deviation = 2.5})$$

34개 이상 비냉을 판매할 수 체인점의 probability



.5인 normal 분포
비냉을 팔

From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

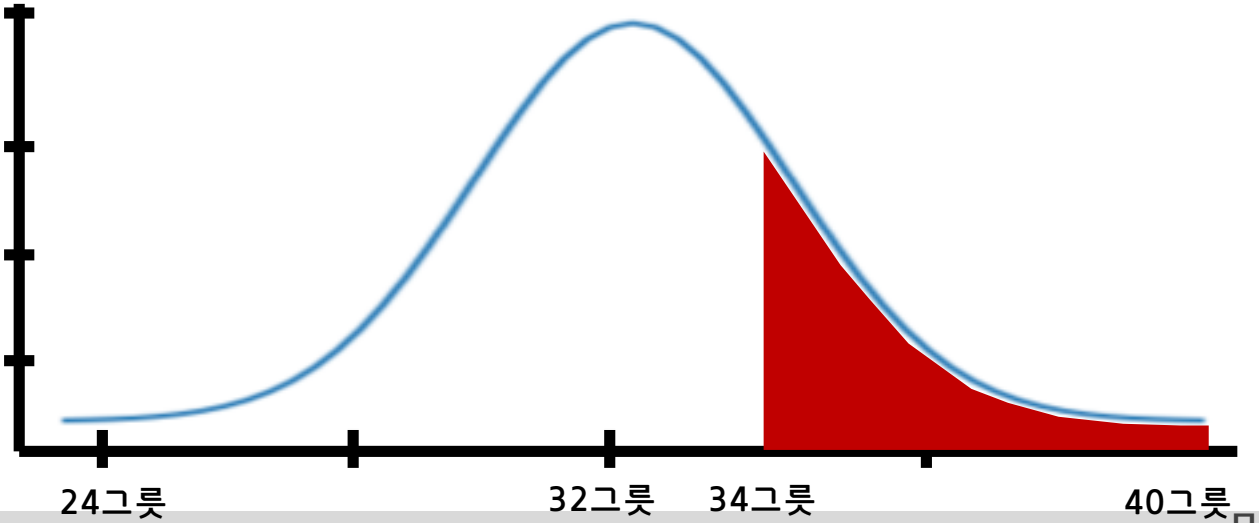
모두의연구소

Probability

수학적으로 표현해 보면

$$P(\text{비냉판매 수} > 34 \mid \underset{\text{고정}}{mean = 32\ and\ deviation = 2.5})$$

분포를 오른쪽에 고정 시켜 두고 왼쪽을 변경 시켜 새로운 probability를 얻을 수 있다



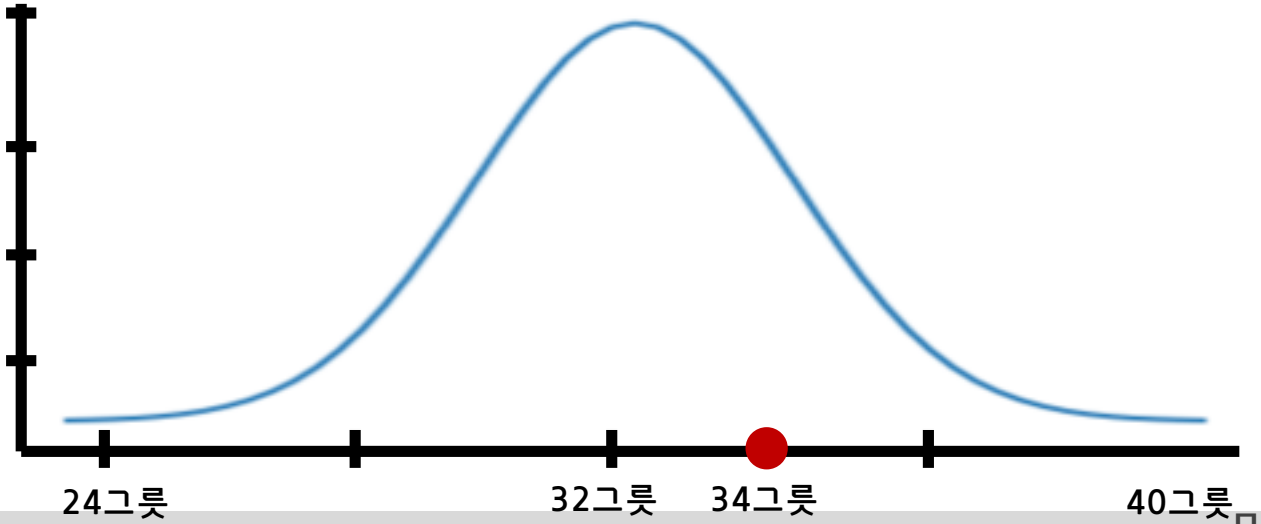
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

모두의연구소

Likelihood

먼저 체인점의 하루 평균 비냉판매량을 측정함 : 34그릇이 나왔다고 가정



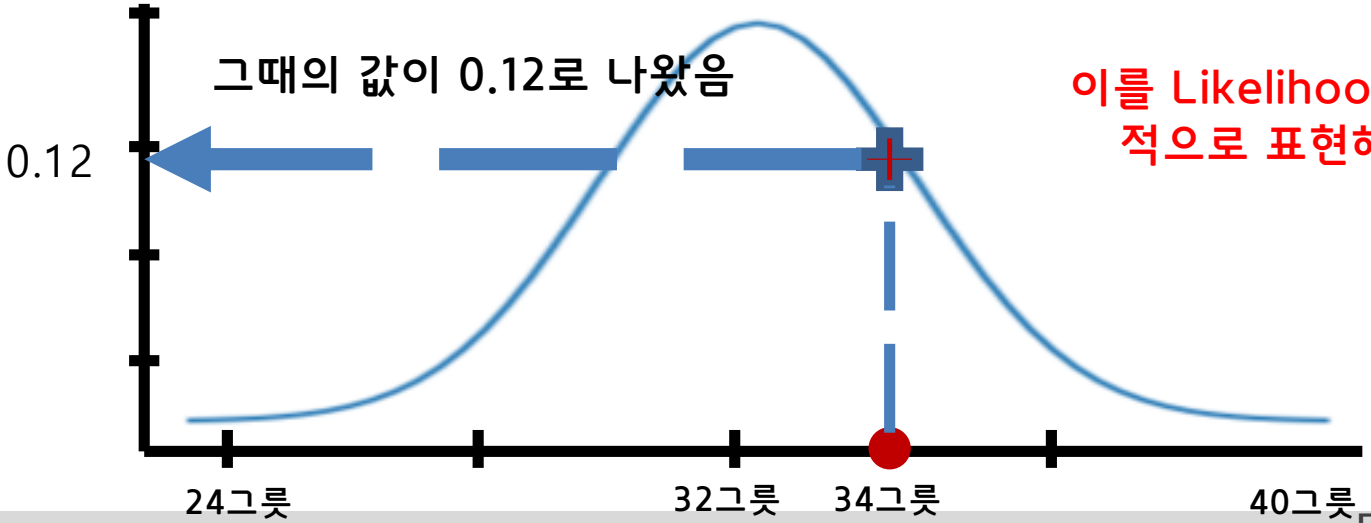
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

모두의연구소

Likelihood

먼저 체인점의 하루 평균 비냉판매량을 측정함 : 34그릇이 나왔다고 가정



이를 Likelihood를 수학적으로 표현해 보자

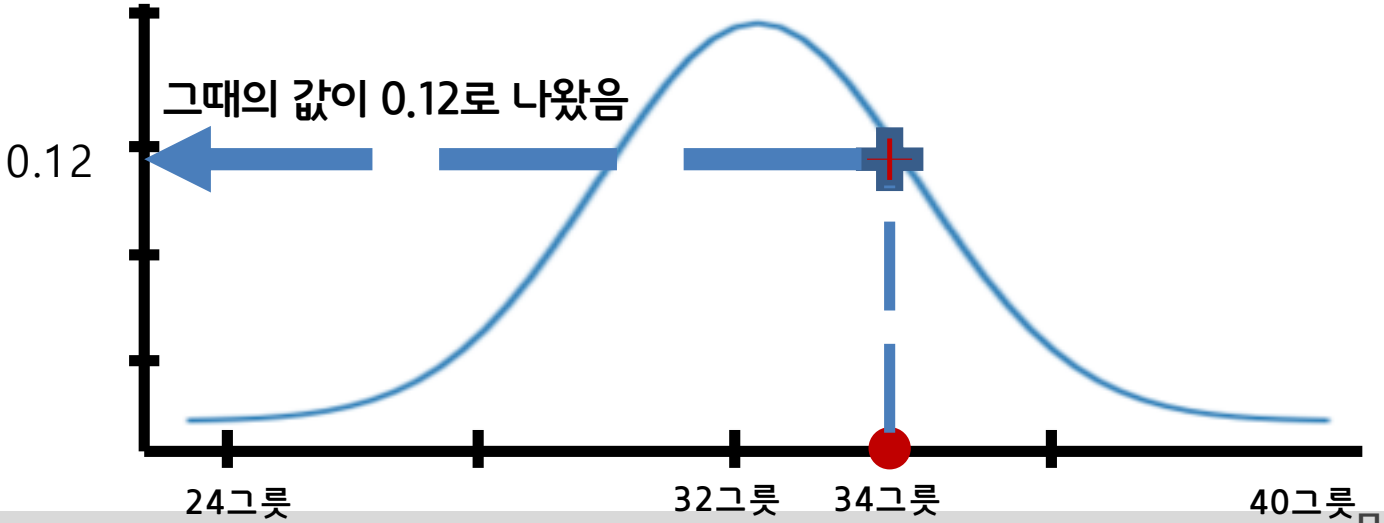
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

Likelihood

$$L(\boldsymbol{mean = 32\ and\ deviation = 2.5} \mid \text{비냉판매 수 } 34 \text{ 그릇})$$

비냉이 34그릇 팔렸을



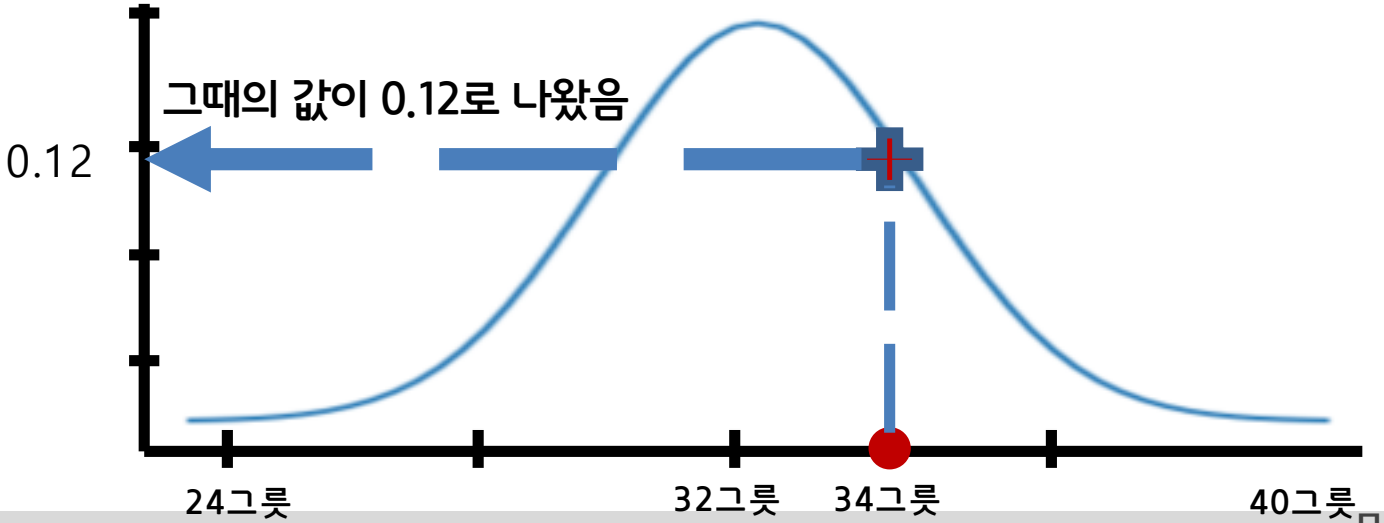
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

Likelihood

$L(\boldsymbol{mean = 32\ and\ deviation = 2.5}\ |$ 비냉판매 수 34 그릇$)$

비냉이 34그릇 팔렸을 때



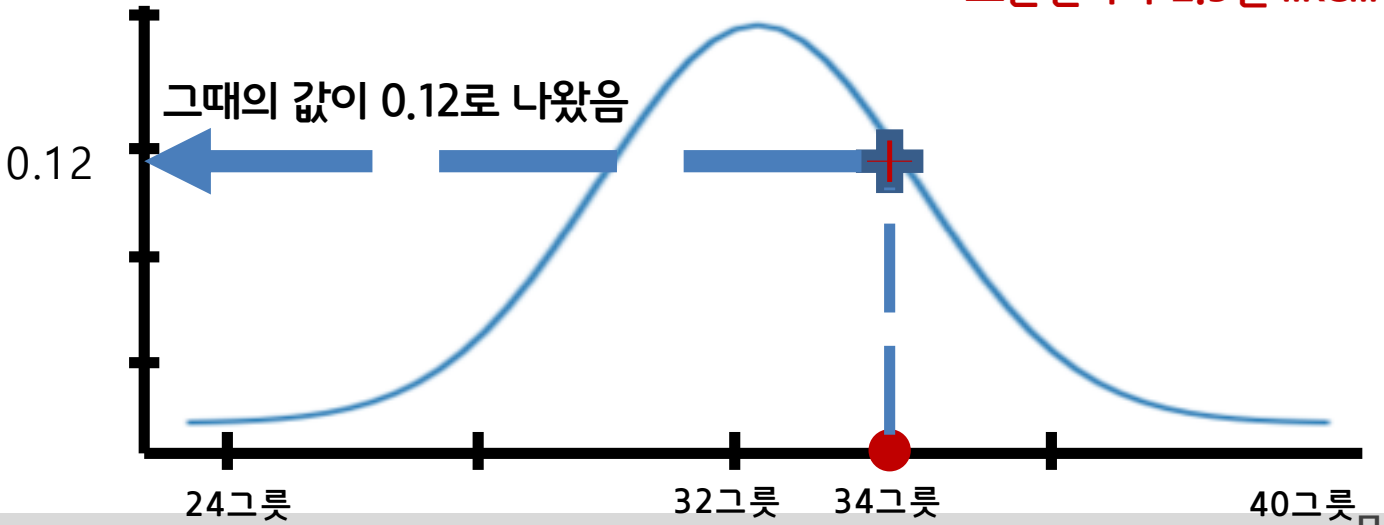
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

Likelihood

$L(mean = 32\ and\ deviation = 2.5 \mid$ 비냉판매 수 34 그릇$)$

비냉이 34그릇 팔렸을 때, 평균이 32
표준편차가 2.5일 likelihood는 0.12임.



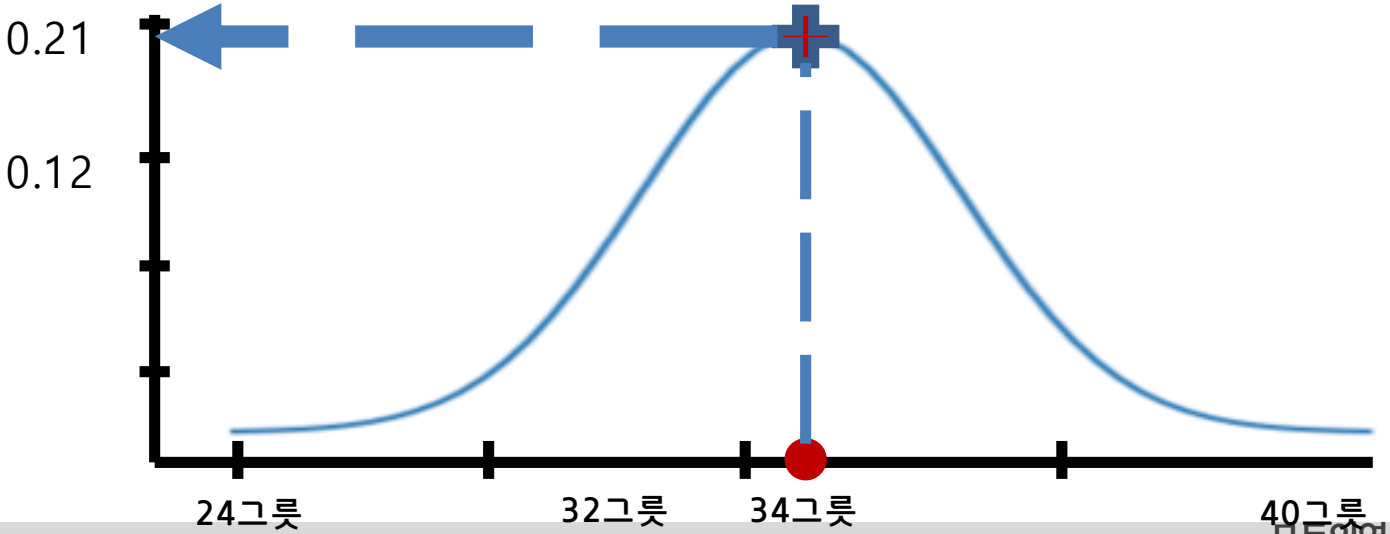
From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Probability vs Likelihood

Likelihood

$L(mean = 34 \ and \ deviation = 2.5 \mid$ 비냉판매 수 34 그릇$)$

평균이 34로 변경 될 경우의 likelihood는 0.21로 증가 함



From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

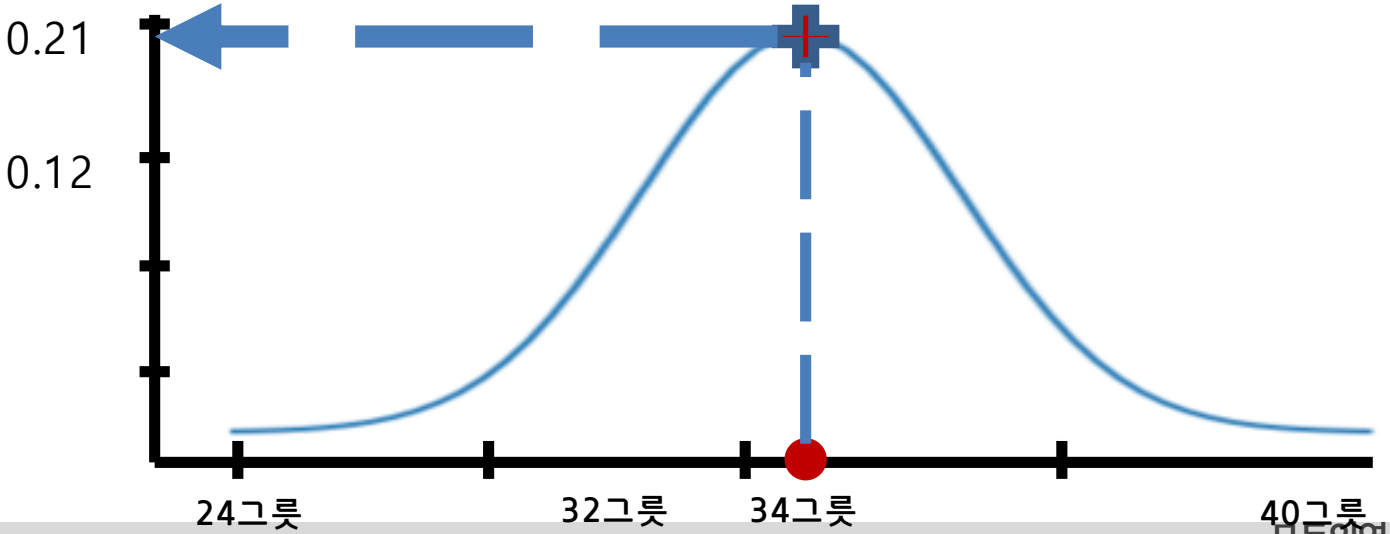
Probability vs Likelihood

Likelihood

Measurement(측정)는 고정

$L(mean = 34\ and\ deviation = 2.5 \mid$ 비냉판매 수 34 그릇$)$

Distribution을 바꾸면 likelihood가 변경 됨



From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

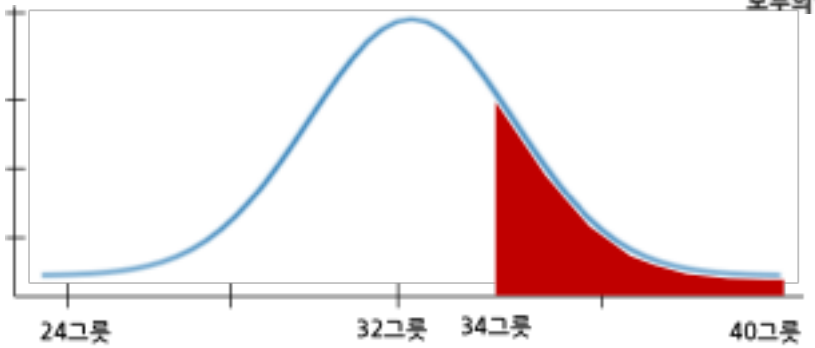
Probability vs Likelihood

모두의연구소

Probability : 고정된 분포에서의 우리가 원하는 영역의 면적

P(data | distribution)

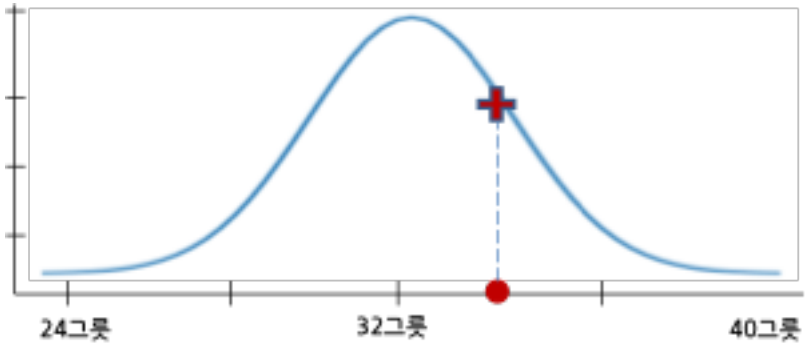
Probability of data given distribution



Likelihood : 고정된 데이터 포인트에서 변형 가능한 분포를 이용해 측정한 Y축 값

L(distribution | data)

Likelihood of distribution given data

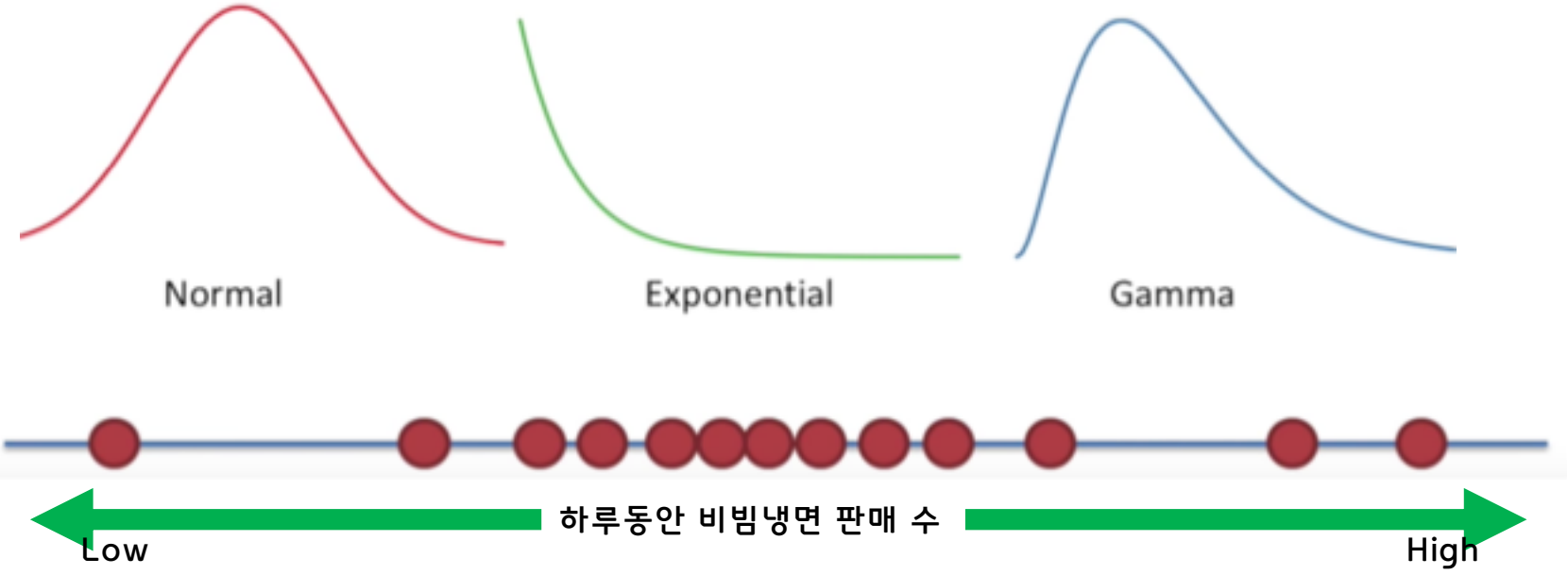


From: StatQuest : Probability vs Lokelihood (https://www.youtube.com/watch?v=pYxNSUDSFH4)

Maximum Likelihood

모두의연구소

Maximum likelihood의 목표는 데이터를 가장 잘 표현하는 분포를 찾는 것이다

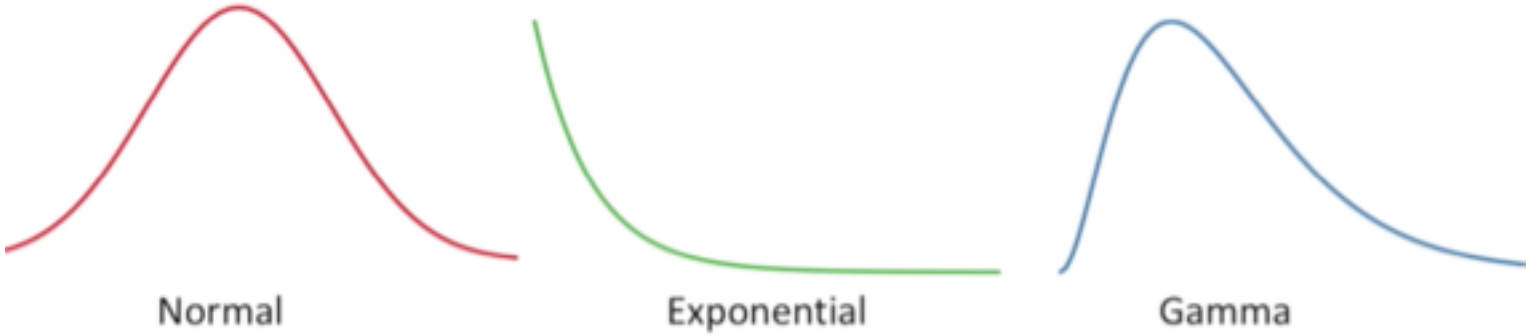


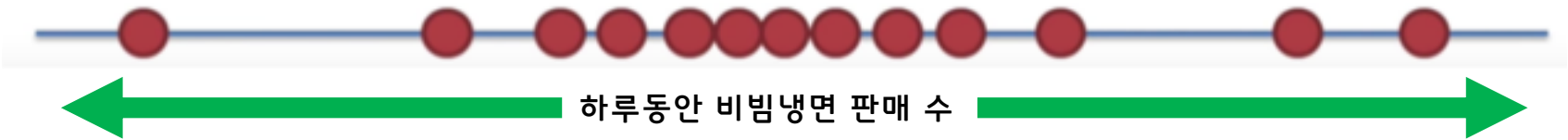
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

분포를 찾는 이유 : 다루기 쉬워지고 일반화 될 수 있다



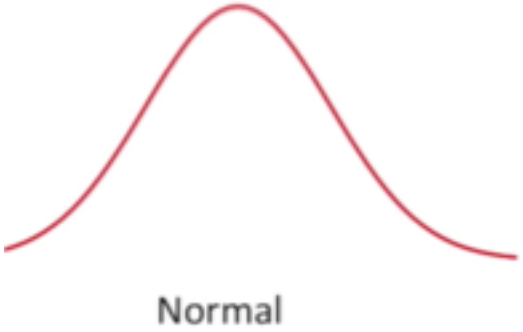


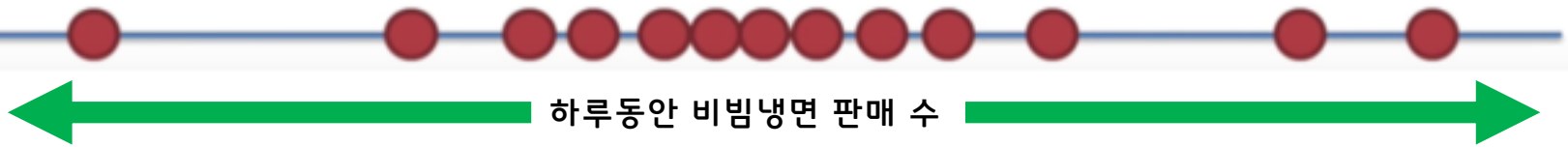
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

하루 동안의 비빔냉면 판매 수가 normal distribution이라 가정하면





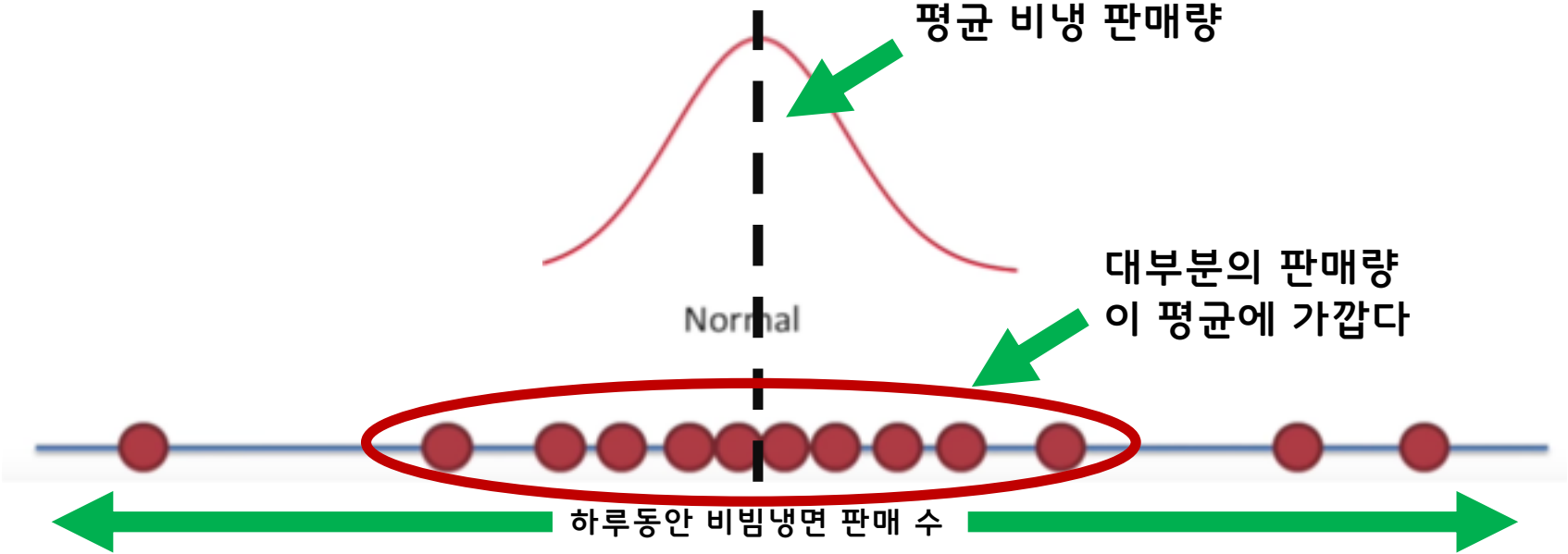
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

Normally distributed의 의미 :

1) 대부분의 측정이 평균(mean)에 가깝다



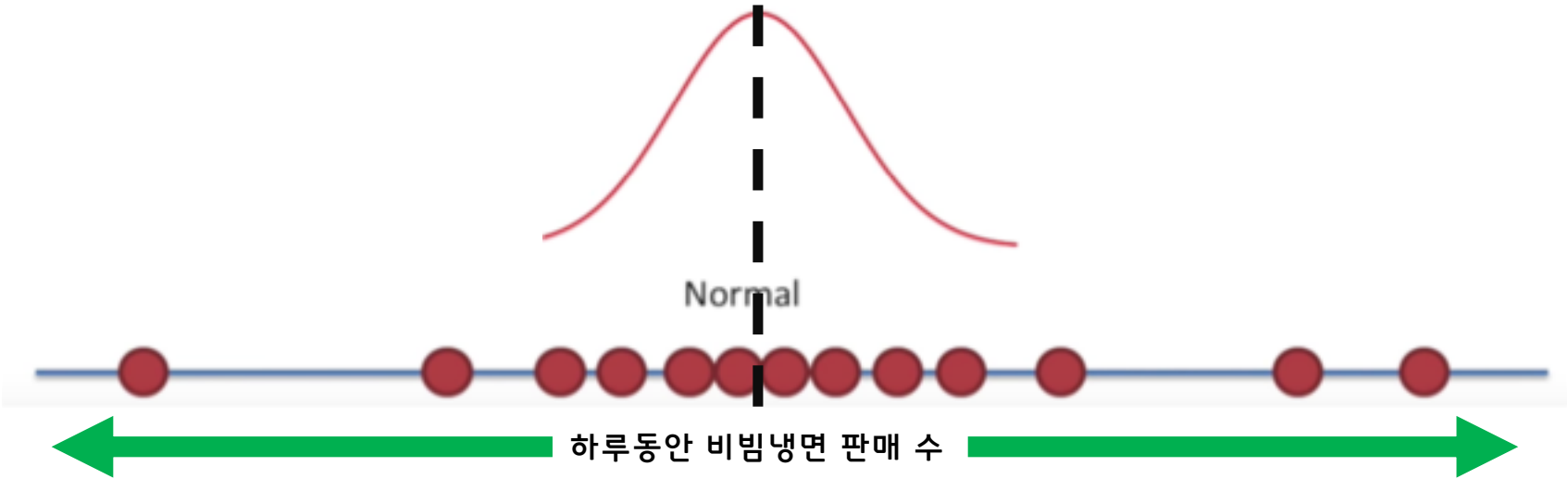
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

Normally distributed의 의미 :

1) 대부분의 측정이 평균(mean)에 가깝다
2) 측정결과가 평균을 중심으로 대략 대칭형이다

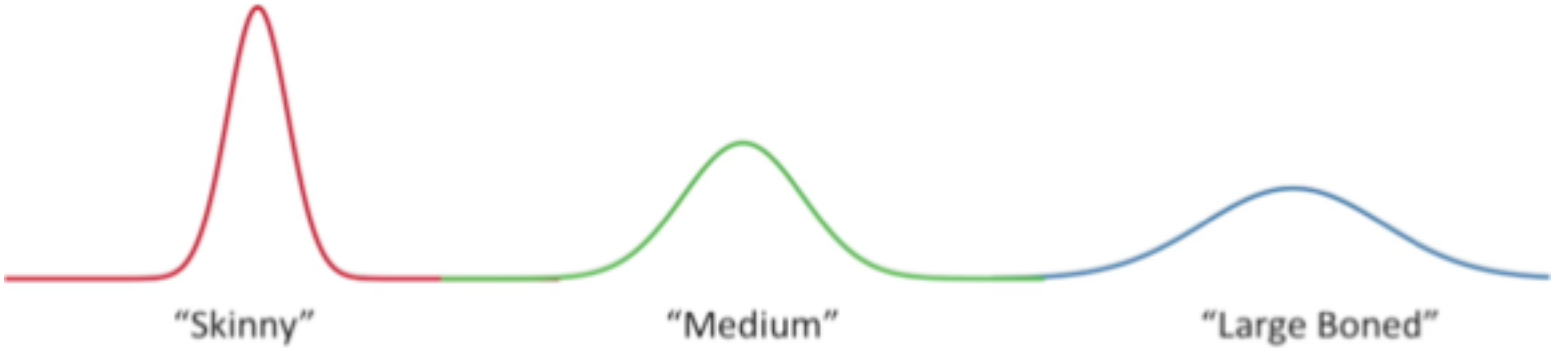


From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

Normally distribution은 여러가지 형태를 가질 수 있습니다

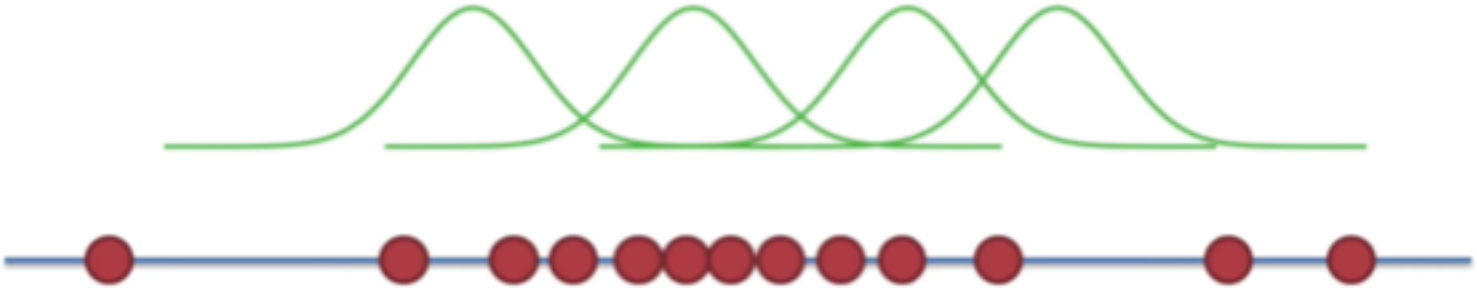


From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

형태를 찾았으면 이 분포를 어디에다 위치시킬지 결정해야 합니다

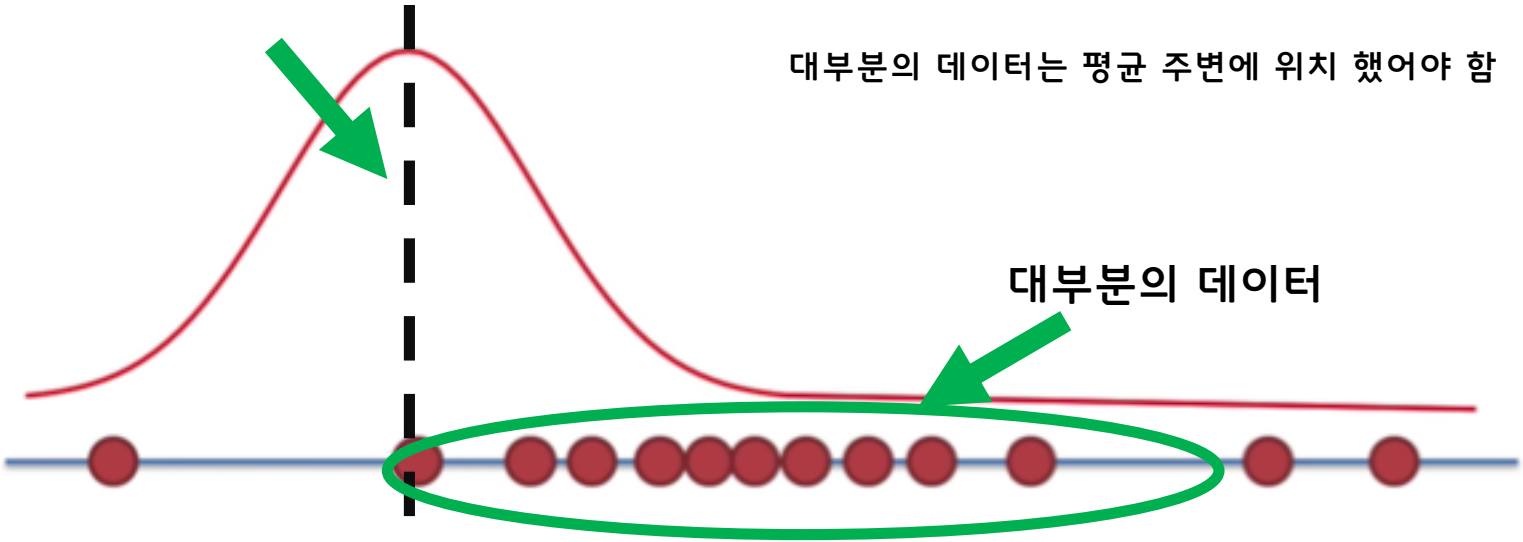


From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

분포의 평균 (mean value of the distribution)

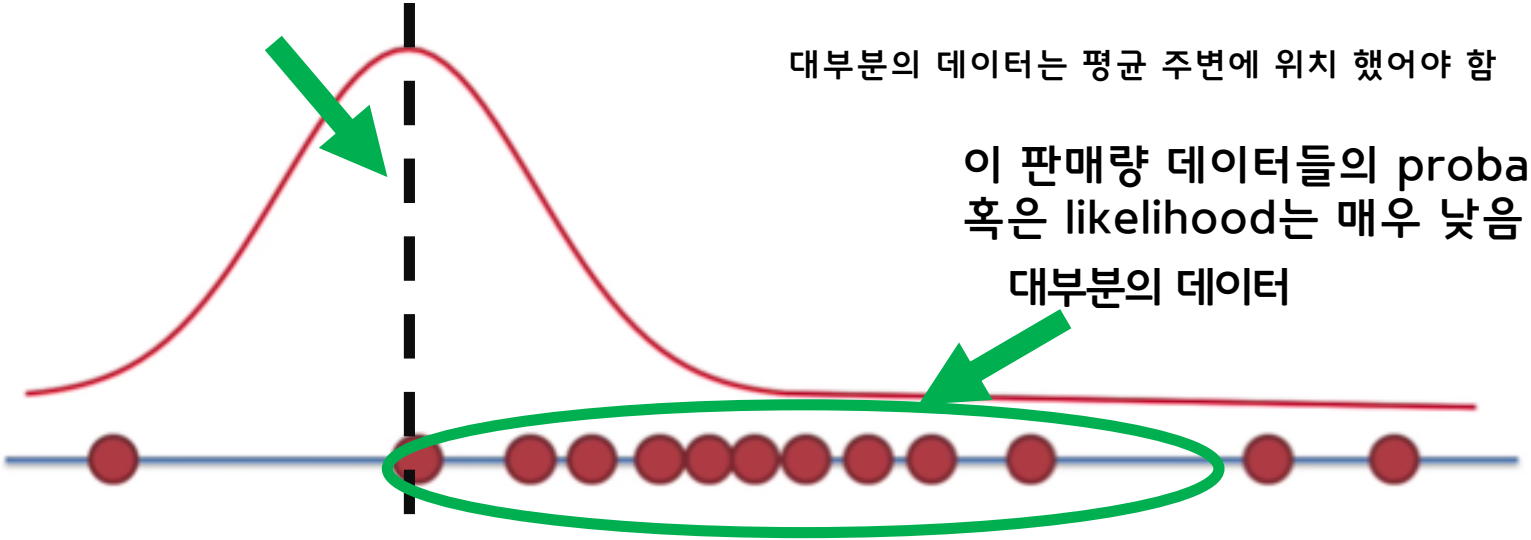


From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

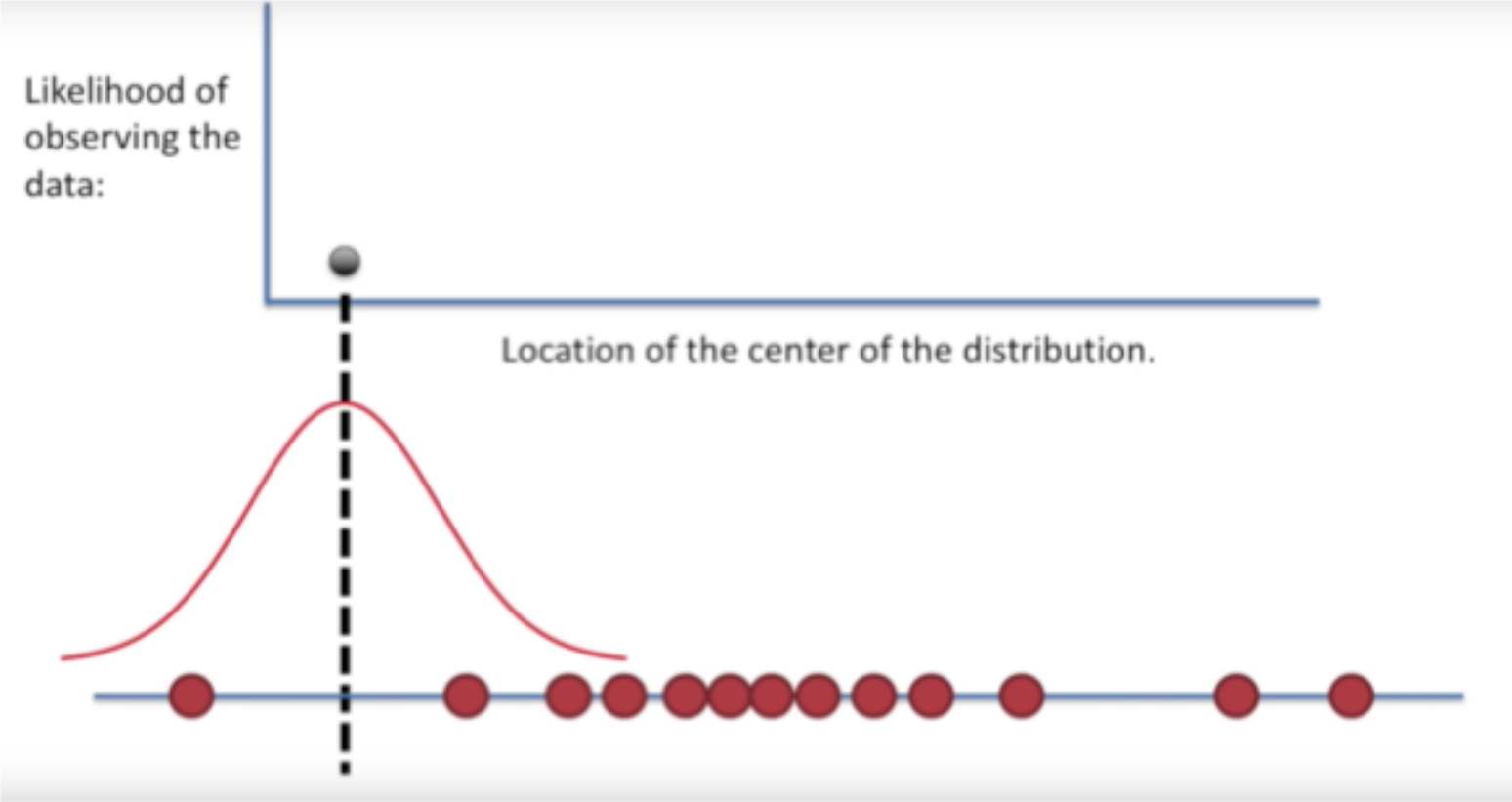
모두의연구소

분포의 평균 (mean value of the distribution)



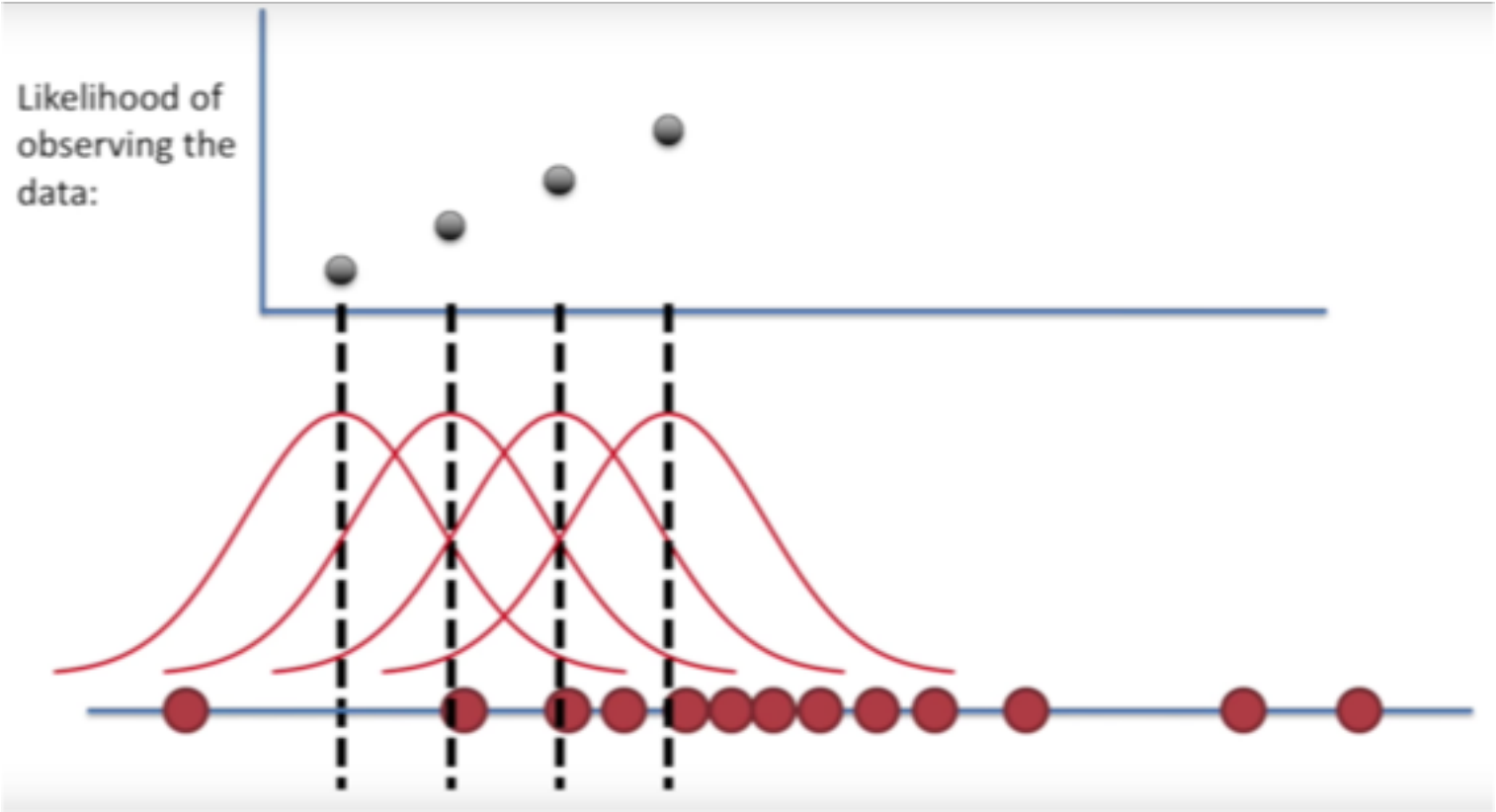
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



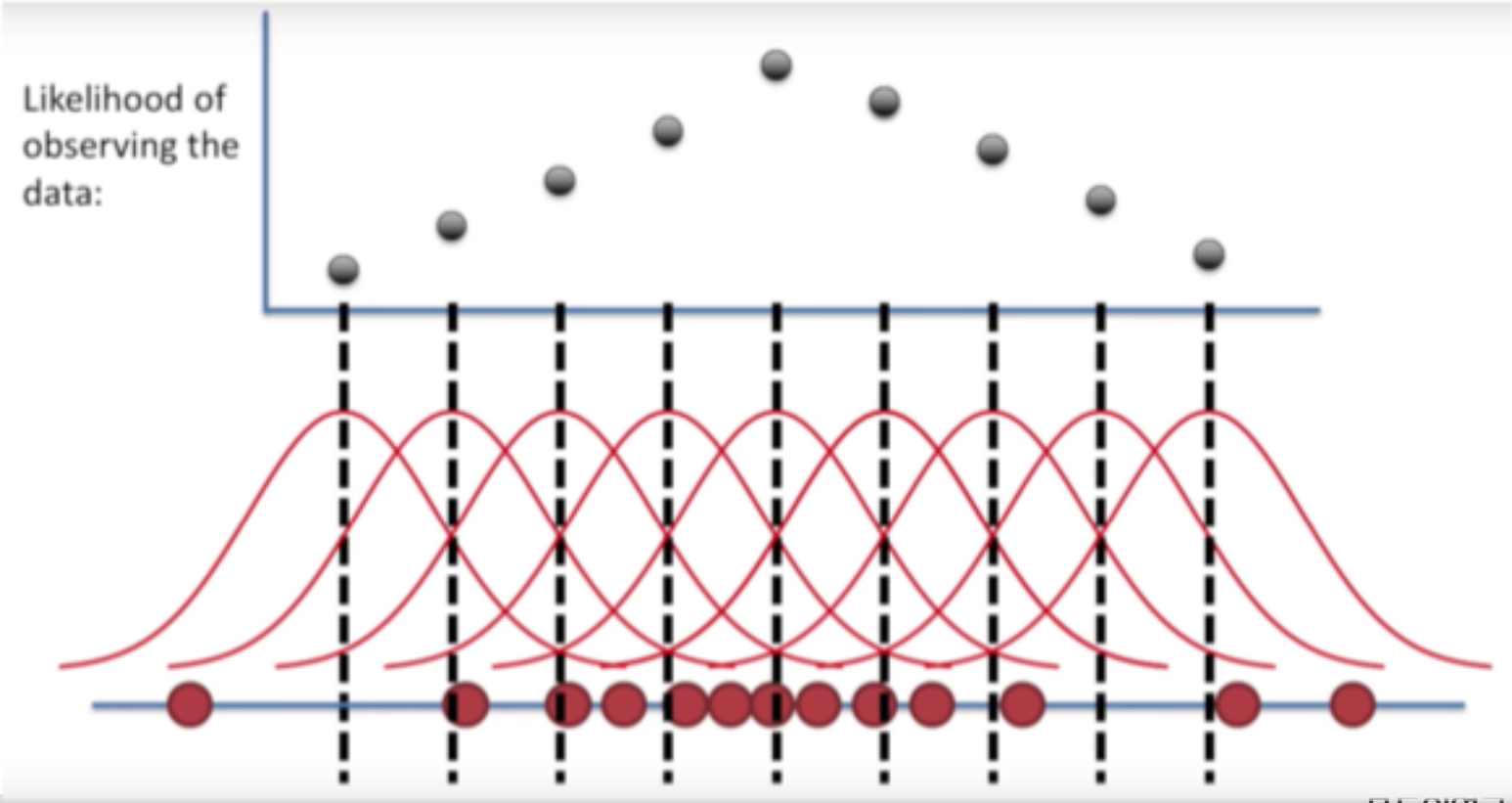
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



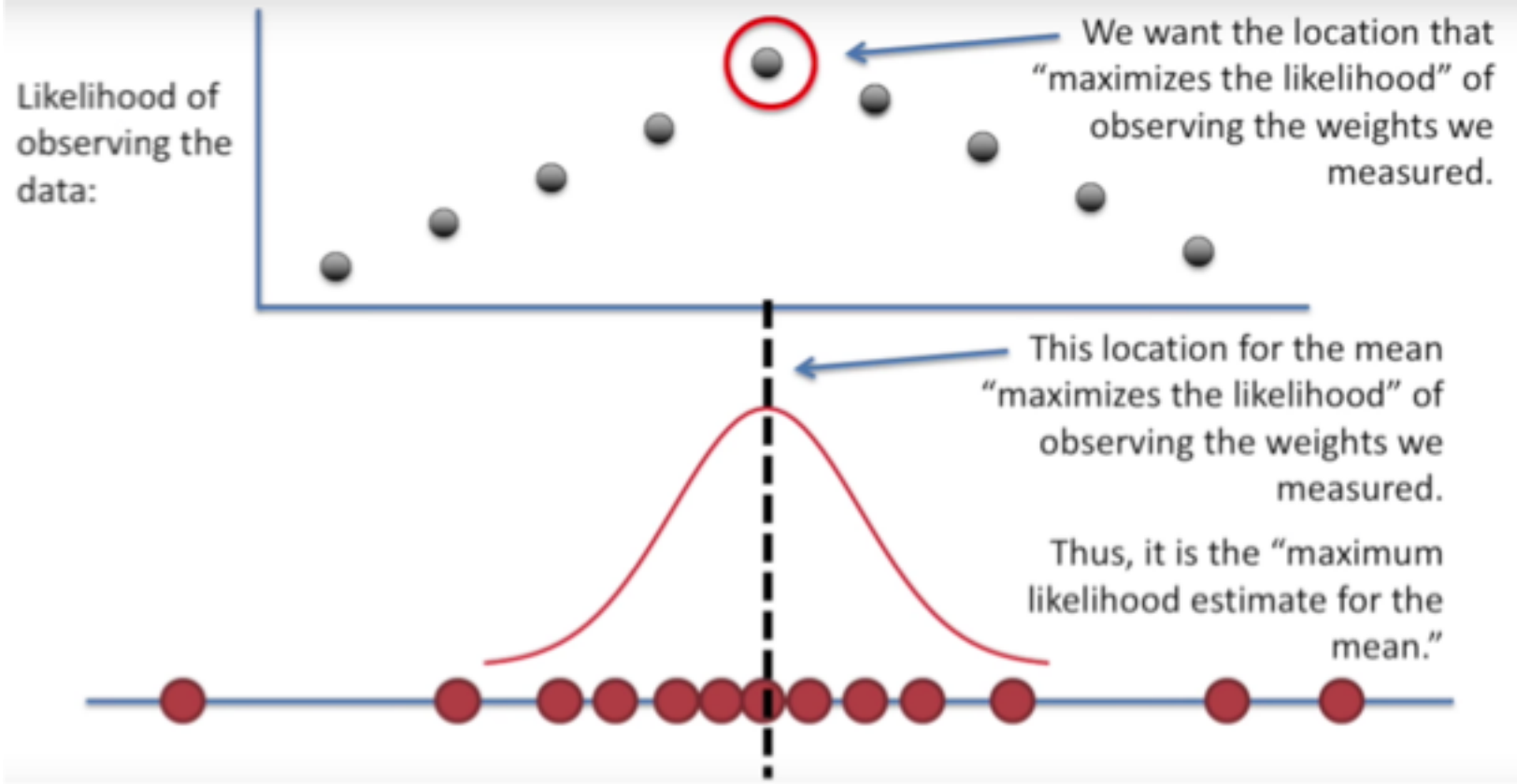
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



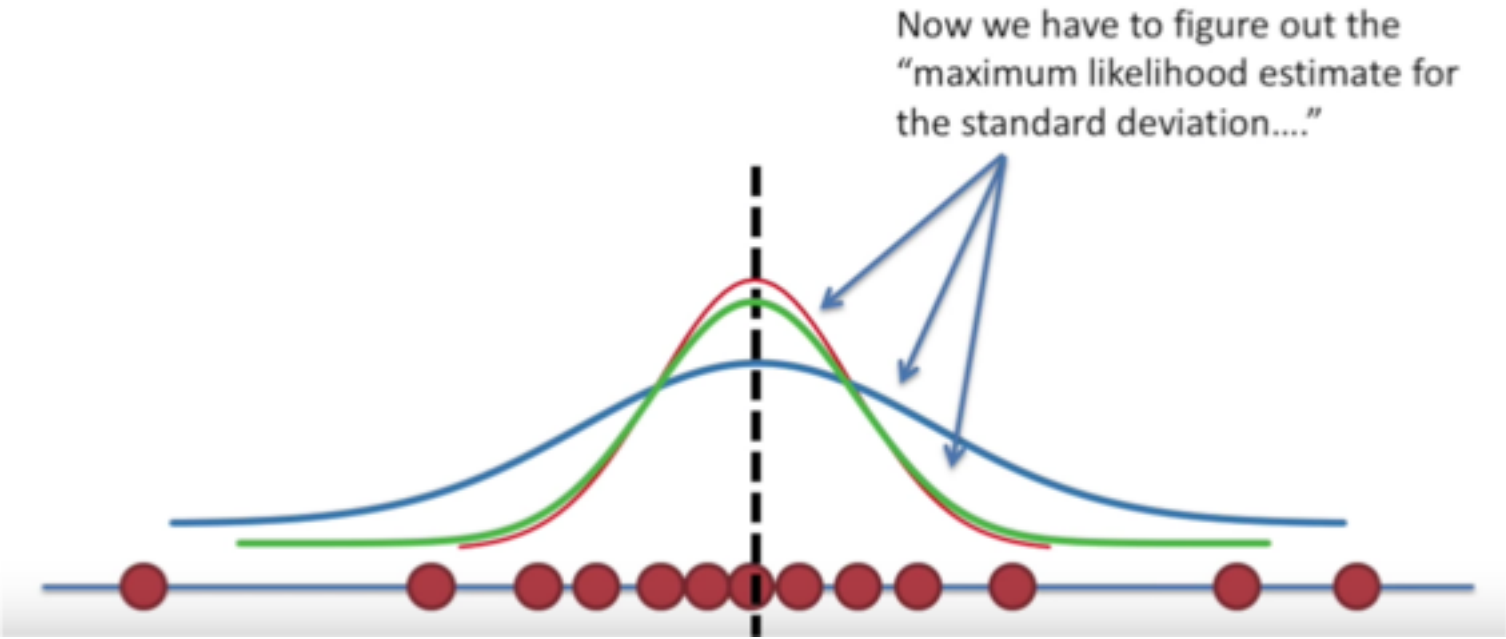
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



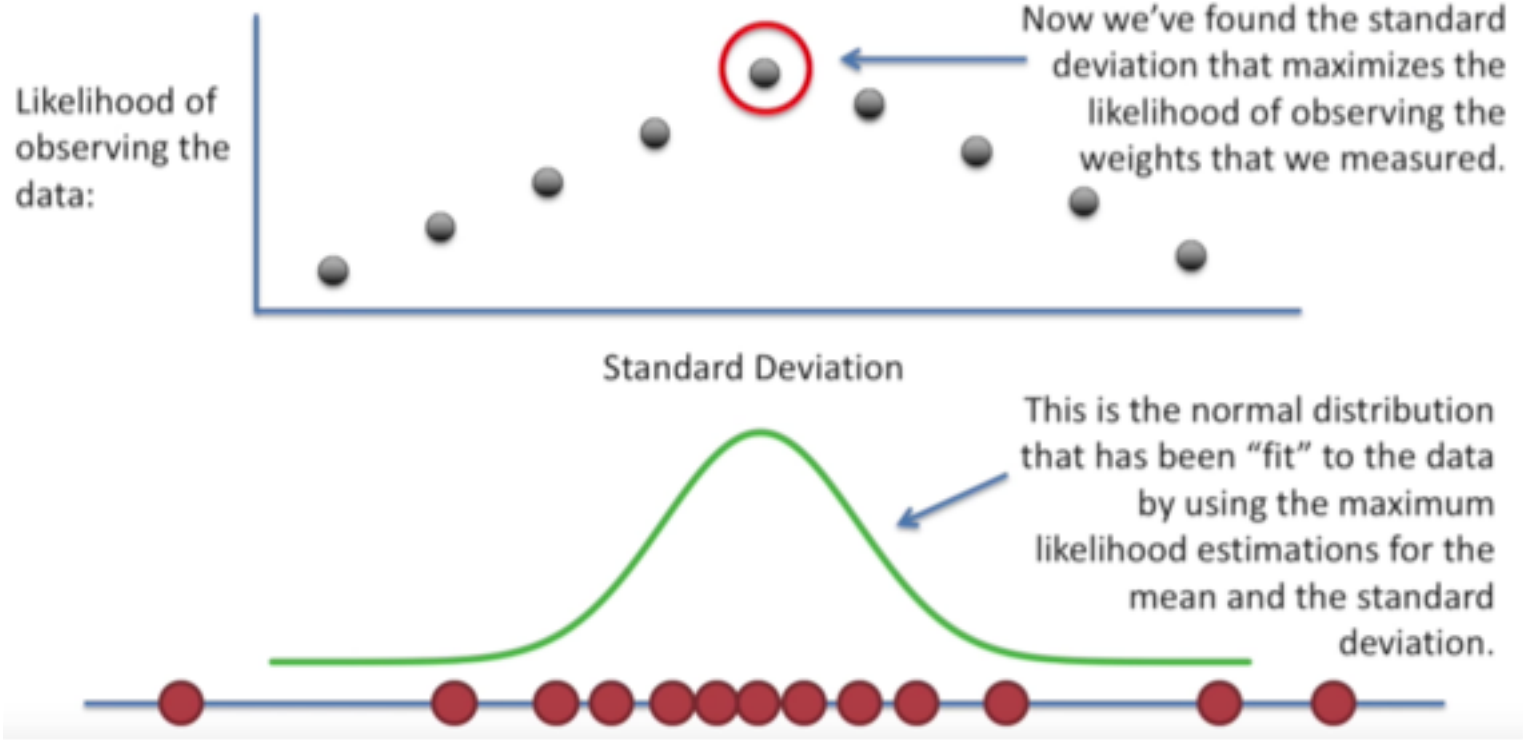
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



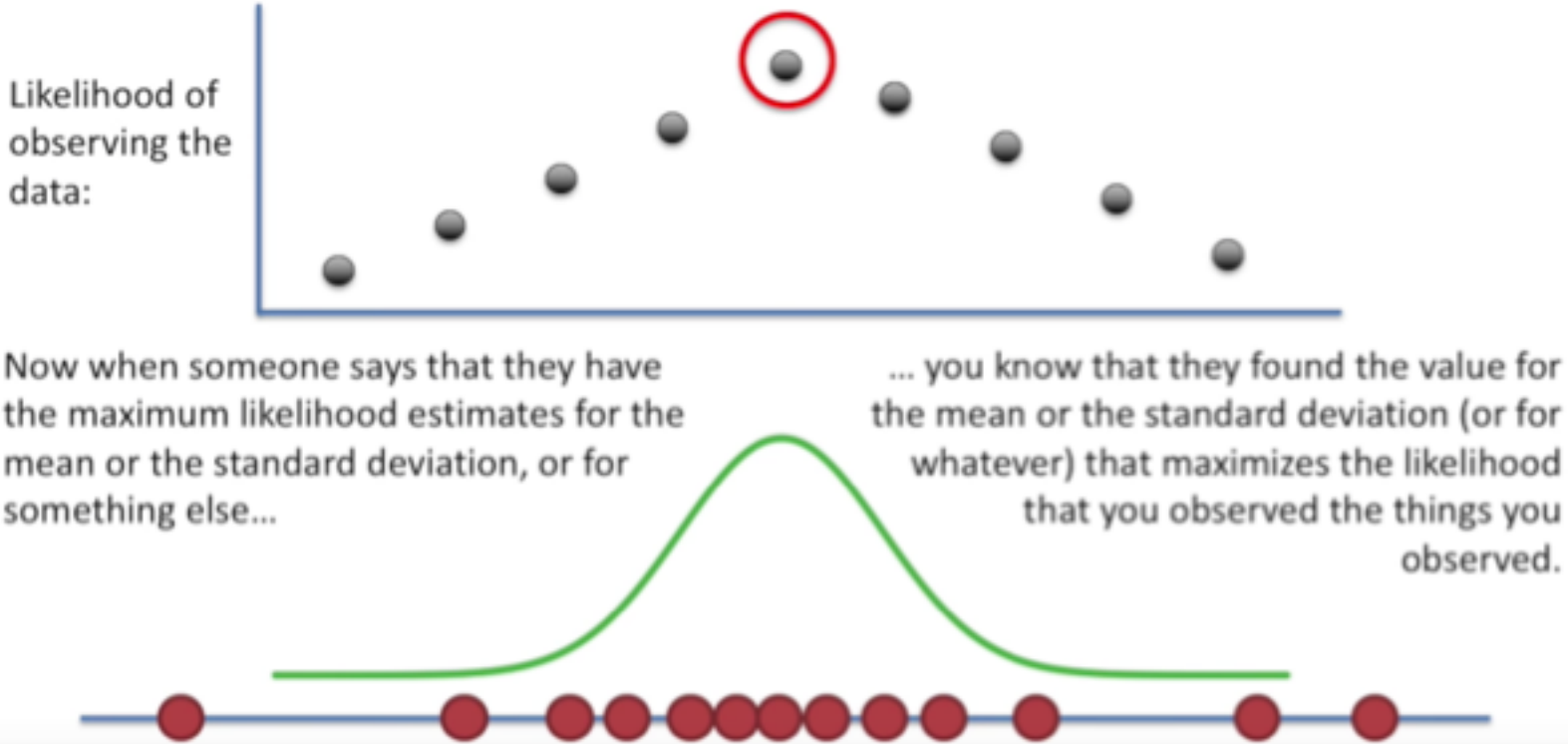
From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood



From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

Maximum Likelihood

모두의연구소

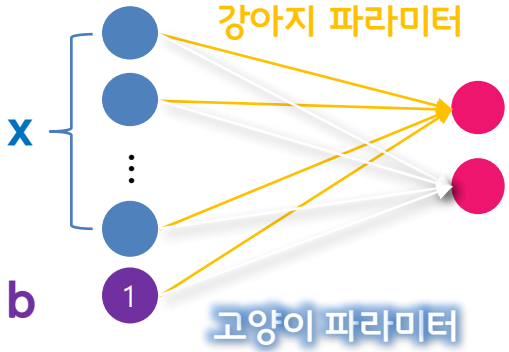
Terminology alert!

In everyday conversation, “probability” and “likelihood” mean the same thing. However, in Stats-Land, “likelihood” specifically refers to this situation we’ve covered here; where you are trying to find the optimal value for the mean or standard deviation for a distribution given a bunch of observed measurements.

From: StatQuest: Maximum Likelihood, clearly explained!!! (https://youtu.be/XepXtl9YKwc)

돌아보기 ...

모두의연구소



강아지 점수

고양이 점수

$$\underset{2\times1}{f(x, W)} = \underset{2\times3,072}{Wx} + \underset{2\times1}{b} \quad (3,072\times1)$$

32x32x3 영상 = 3,072 픽셀

강아지 파라미터	0.2	0.1	...	0.3	0.7	x	155
고양이 파라미터	0.7	0.8	...	0.9	0.4		200
	W						...
							110
							78
							x

$$+ \begin{matrix} 0.1 \\ 0.2 \end{matrix}_{b} = \begin{matrix} 0.3 \\ 0.7 \end{matrix}$$

0.3 강아지 점수

0.7 고양이 점수

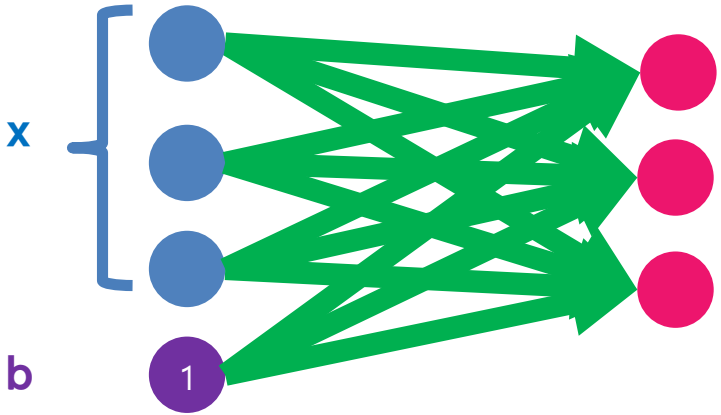
선형분류기

돌아보기 ..

모두의연구소

(3x4) (4x1)

$W_1 X^T$



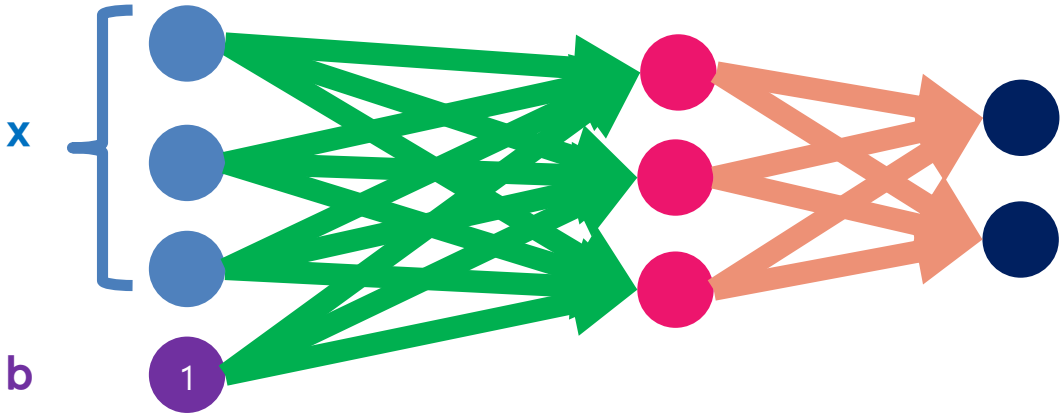
돌아보기 ..

모두의연구소

(2x3) (3x4) (4x1)

W_2 W_1X^T

레이어추가



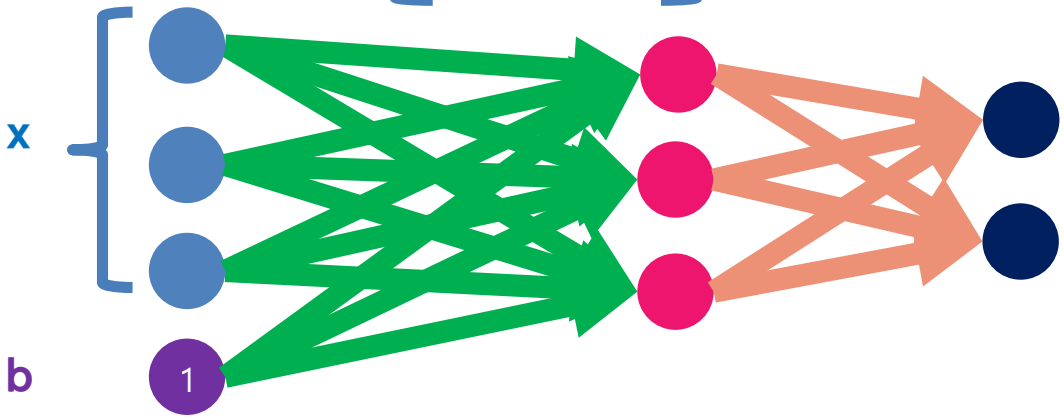
돌아보기 ..

모두의연구소

핫 ~

$$(2\times3)\ (3\times4) \quad (2\times4)$$
$$W_2\ W_1 = W$$
$$\left[\begin{matrix}(2\times3)\ (3\times4)\\ W_2\ W_1\end{matrix}\right]\begin{matrix}(4\times1)\\ X^T\end{matrix}$$

- 두개의 레이어가 하나로 표현
- 레이어를 쌓는 효과가 없어짐

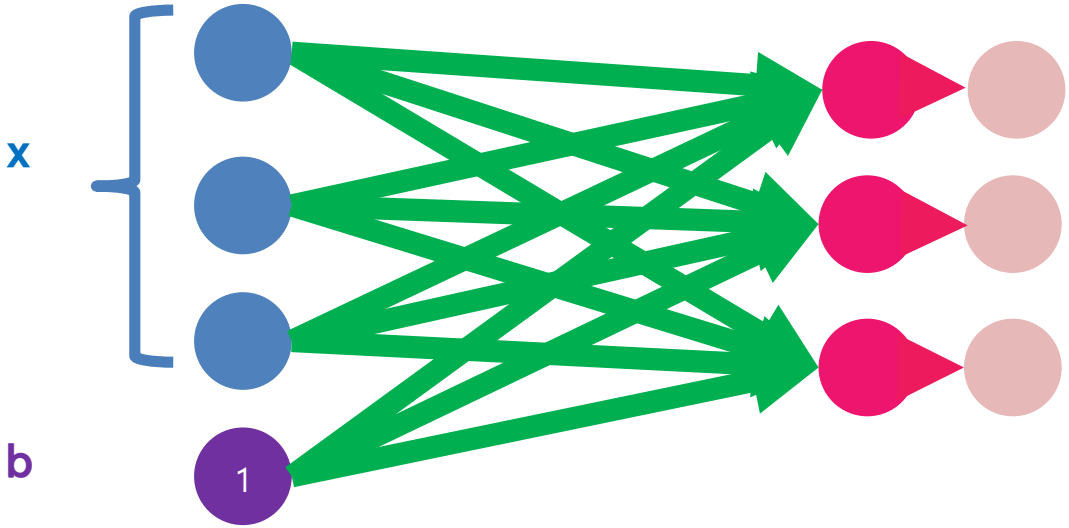


비선형 연산이 필요

돌아보기 ..

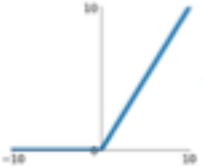
모두의연구소

$$\max(0, W_1 X^T)$$



ReLU

$$\max(0, x)$$

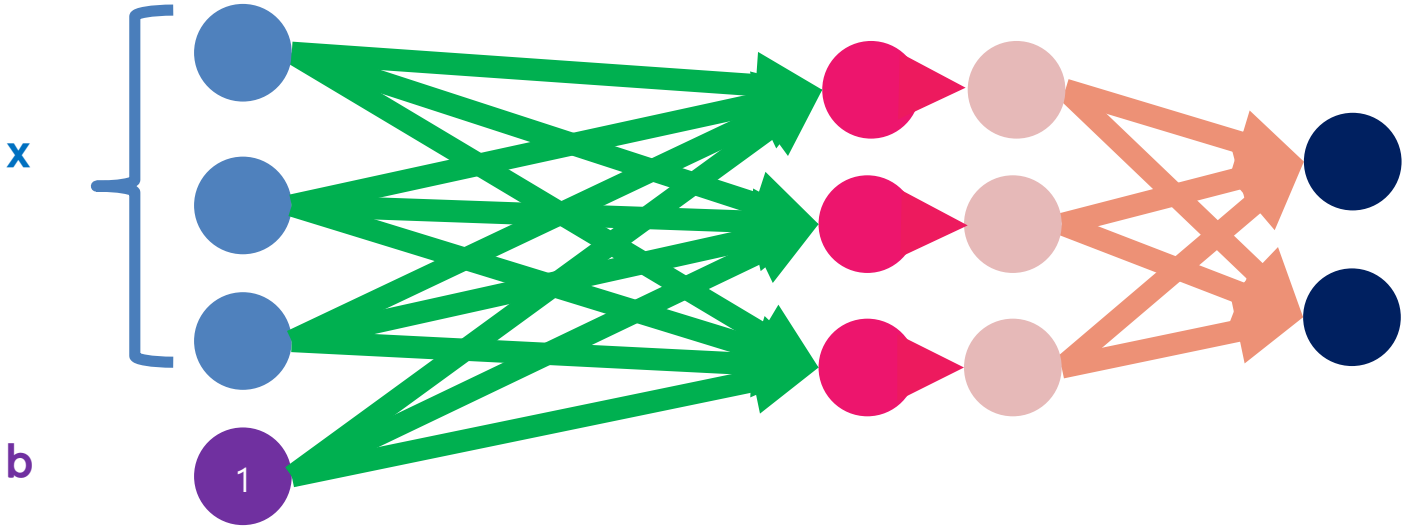


돌아보기 ..

모두의연구소

레이어 추가

$$W_2 \times \max(0, W_1 X^T)$$

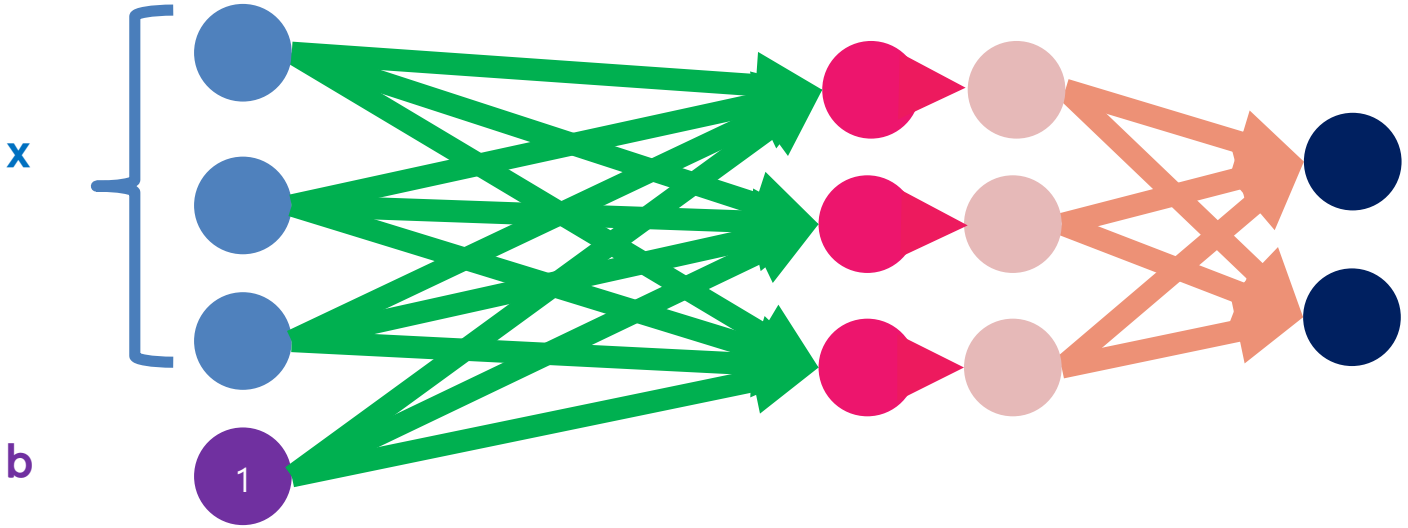


돌아보기 ..

1. Score function

레이어 추가

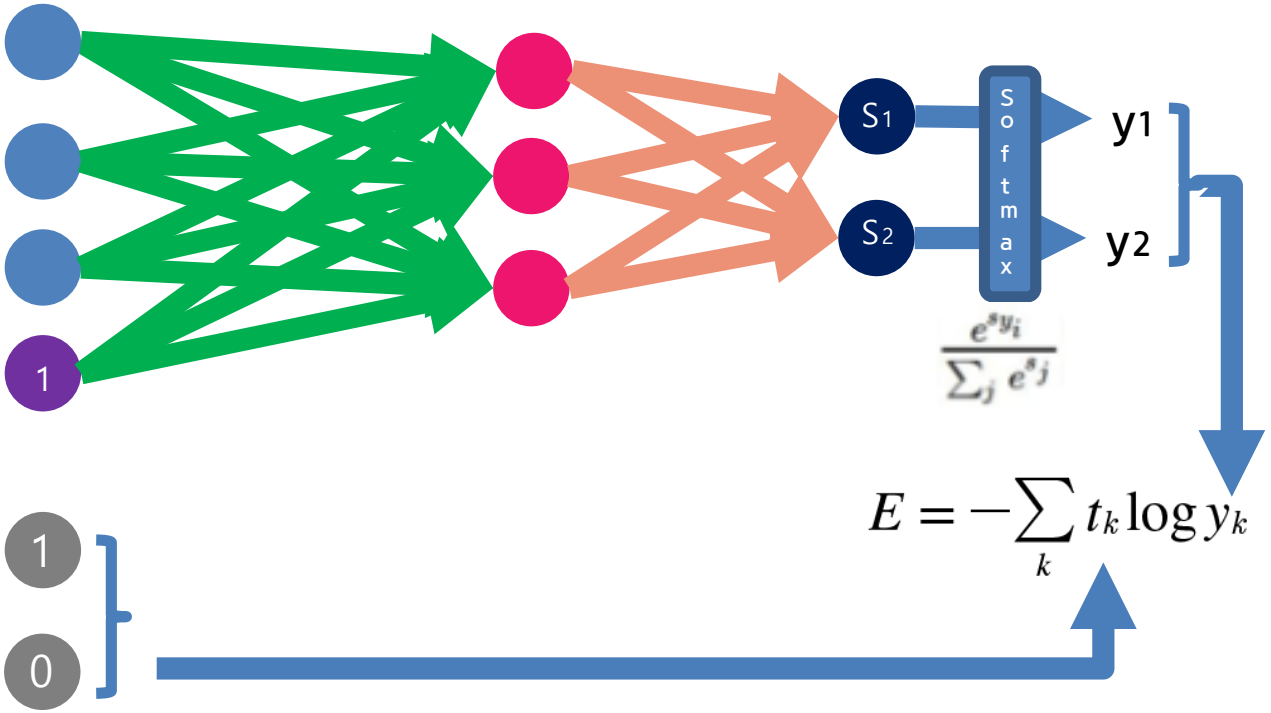
$$W_2 \times \max(0, W_1 X^T)$$



1. Score function
2. Loss function

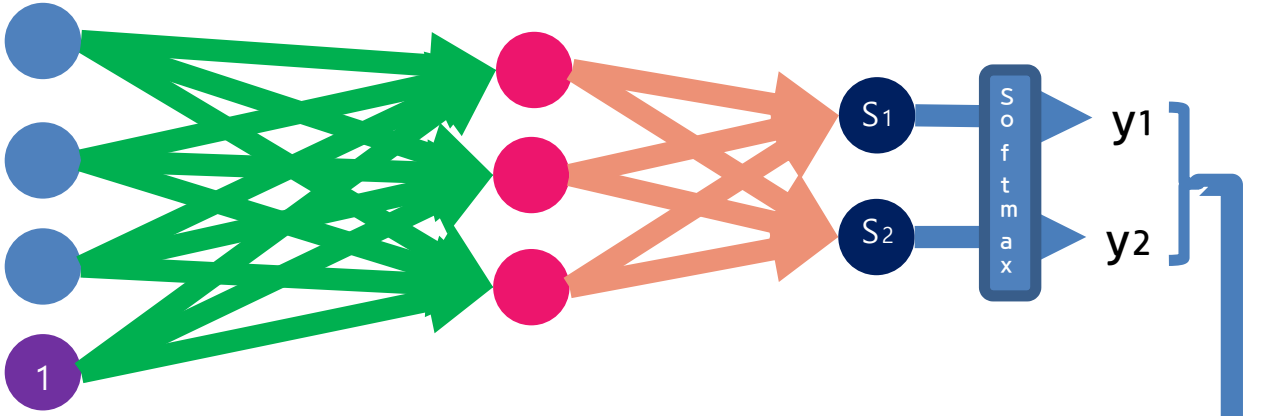
돌아보기 ..

Loss function



돌아보기 ..

1. Score function
2. Loss function
3. Optimization



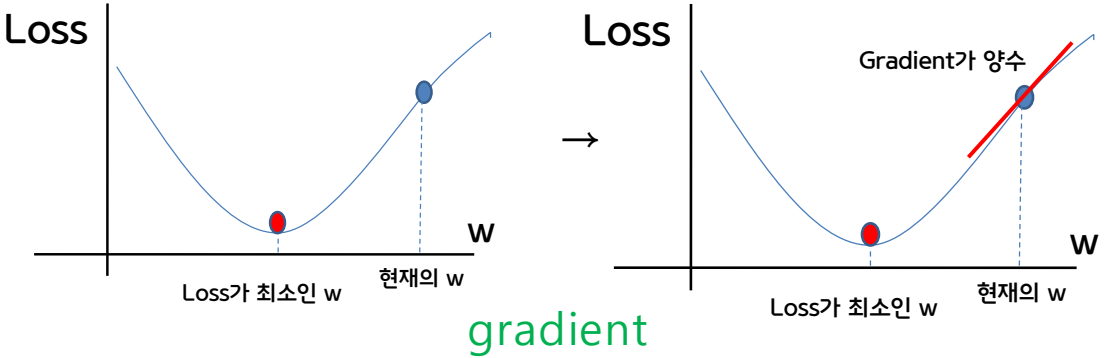
Loss를 줄이는 방법

Gradient Descent

$$Loss \leftarrow E = -\sum_k t_k \log y_k$$

Cross entropy
(Softmax loss)

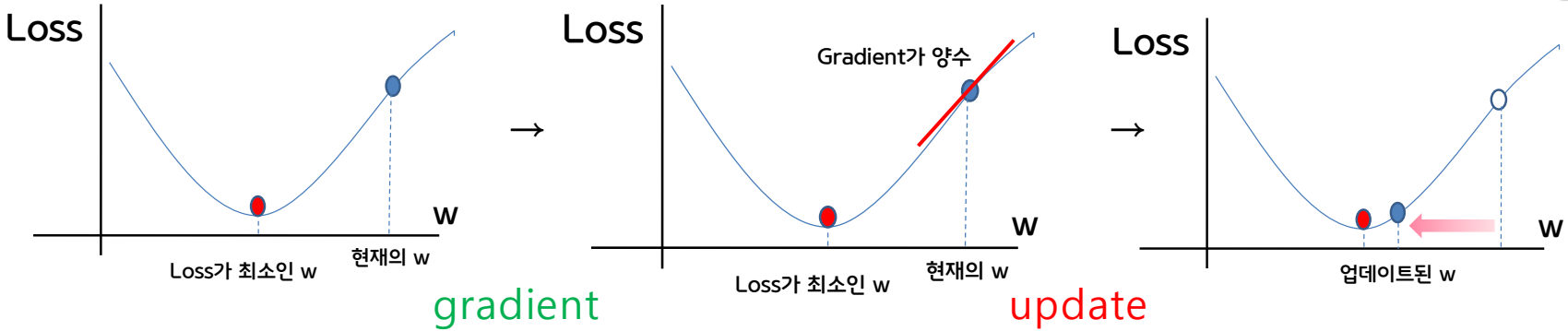
Gradient Descent



gradient

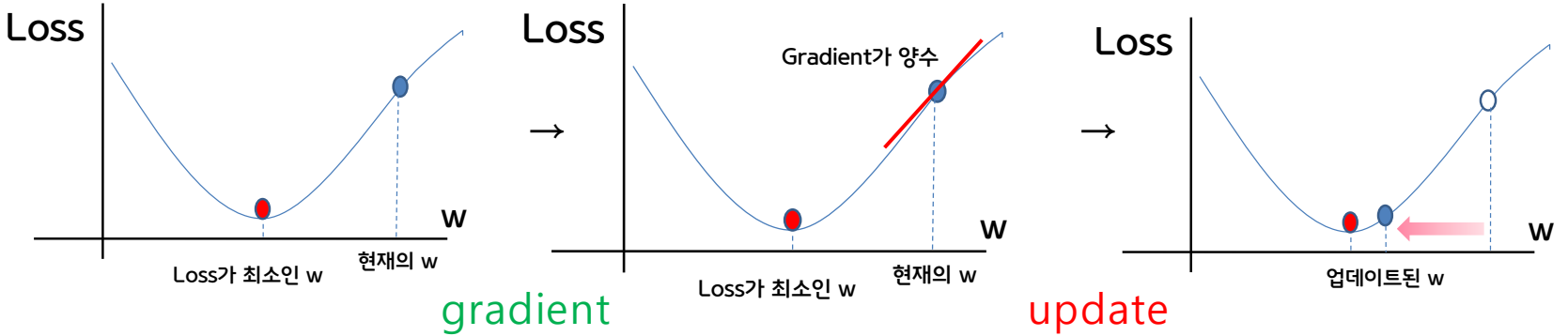
$$w = w - \eta \frac{\partial L}{\partial w}$$

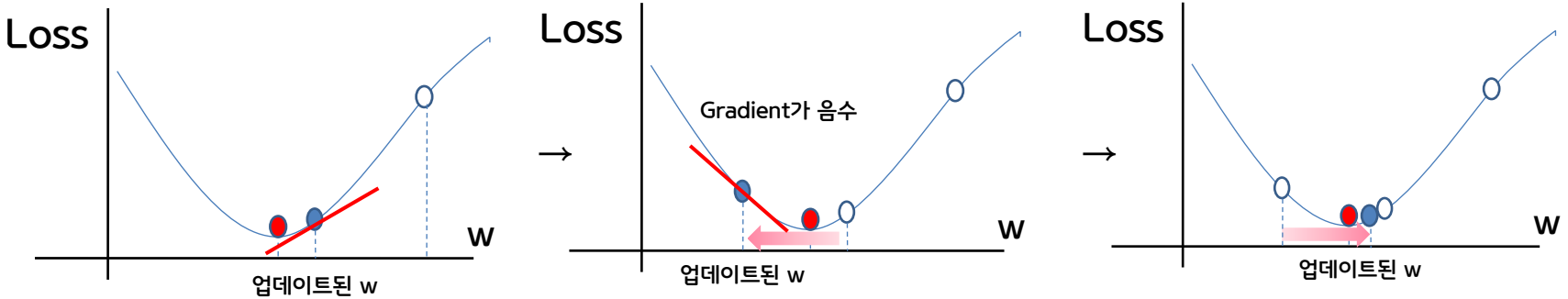
Gradient Descent



$$w = w - \eta \frac{\partial L}{\partial w}$$

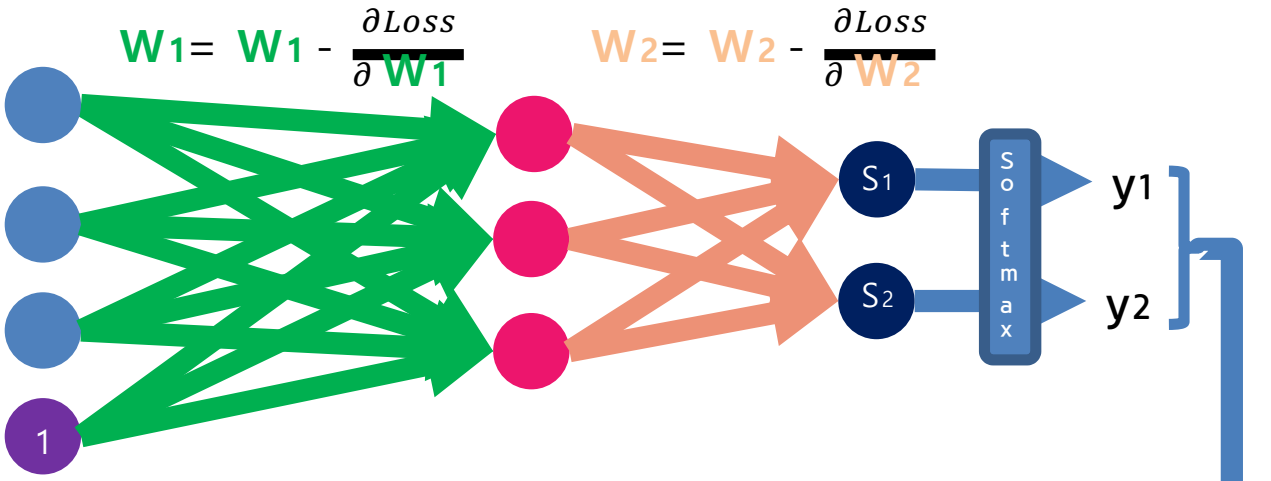
Gradient Descent : $w = w - \eta \frac{\partial L}{\partial w}$





1. Score function
2. Loss function
3. Optimization

돌아보기 ..



Loss를 줄이는 방법

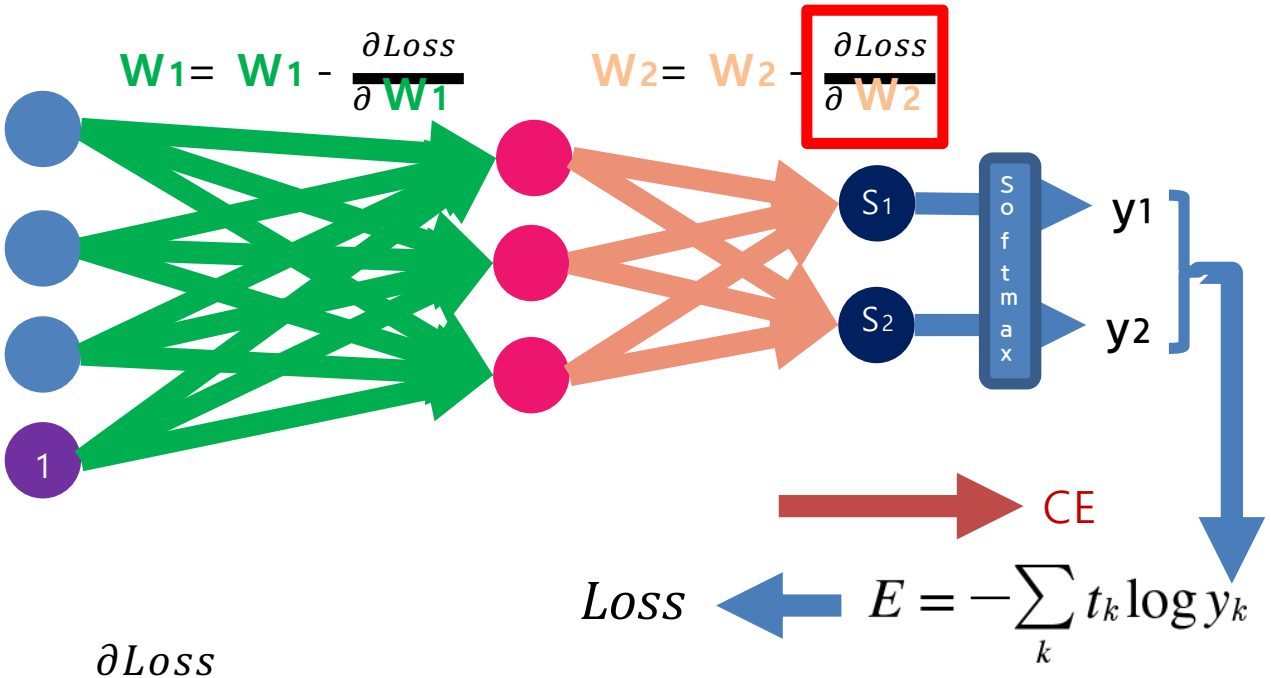
$Loss \leftarrow E = -\sum_k t_k \log y_k$

Cross entropy
(Softmax loss)

Gradient 계산방법 : Backpropagation

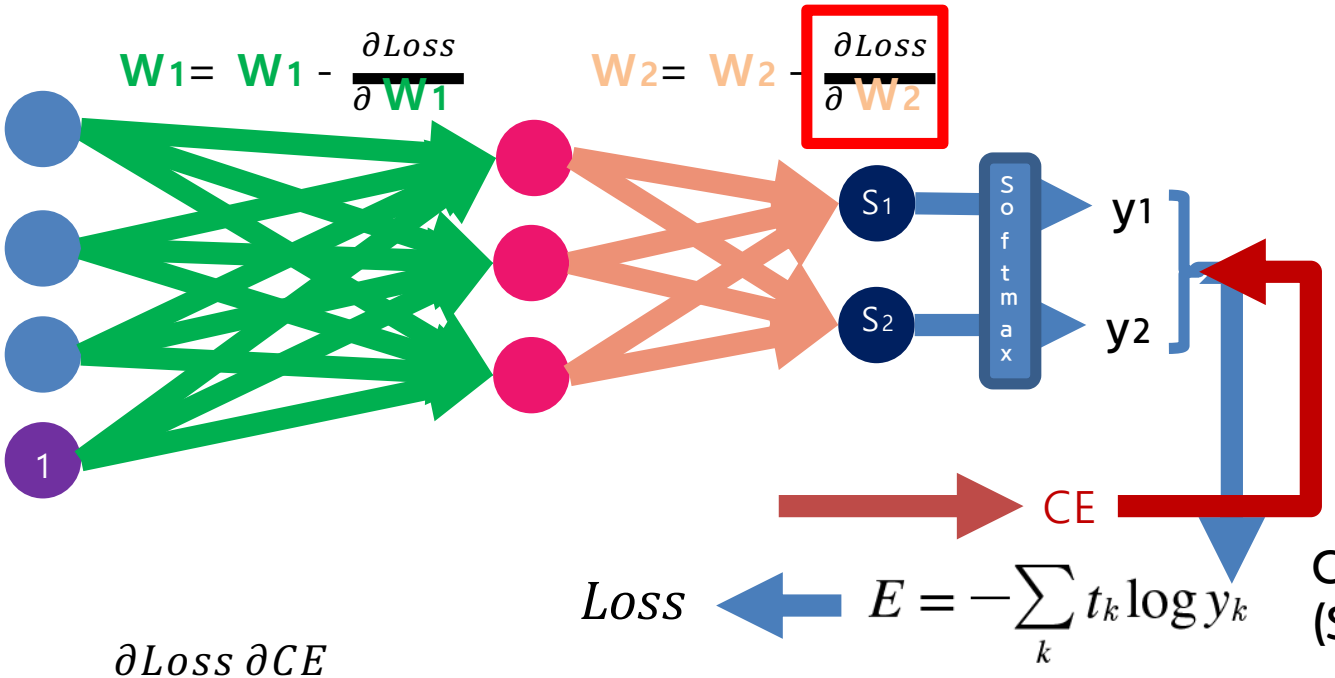
1. Score function
2. Loss function
3. Optimization

돌아보기 ..



1. Score function
2. Loss function
3. Optimization

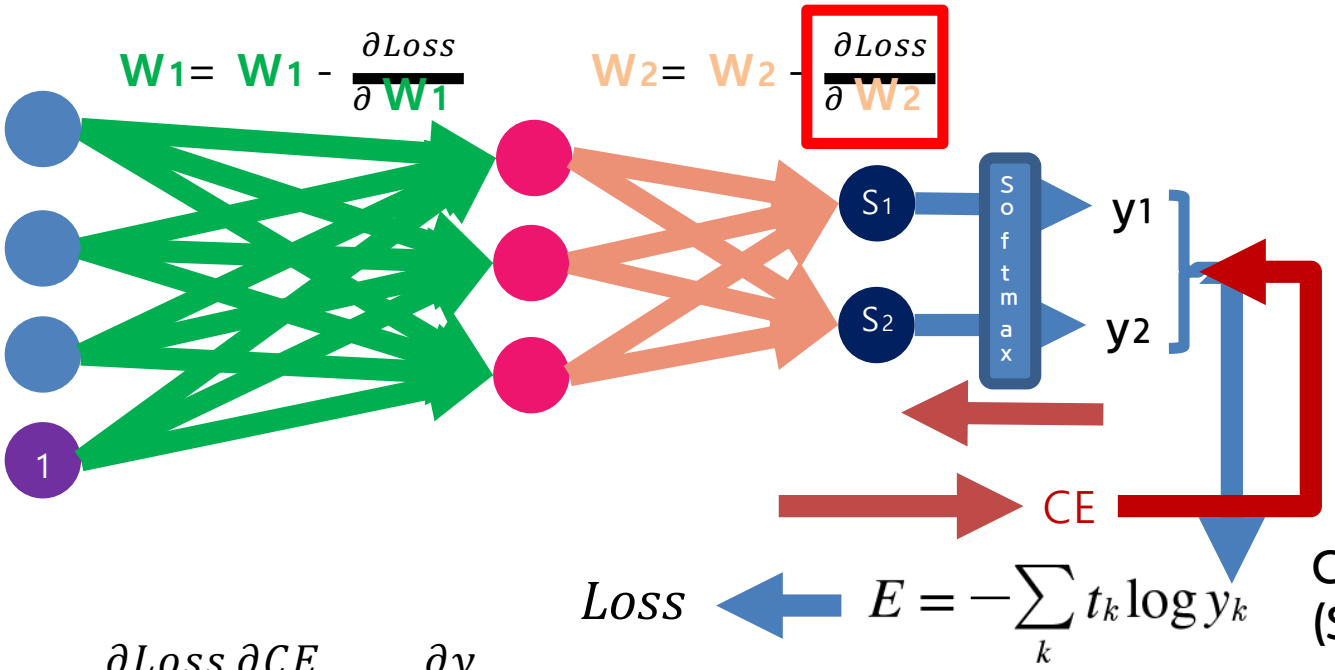
돌아보기 ..



$$\frac{\partial Loss}{\partial W_2} \rightarrow \frac{\partial Loss}{\partial CE}\frac{\partial CE}{\partial y}$$

1. Score function
2. Loss function
3. Optimization

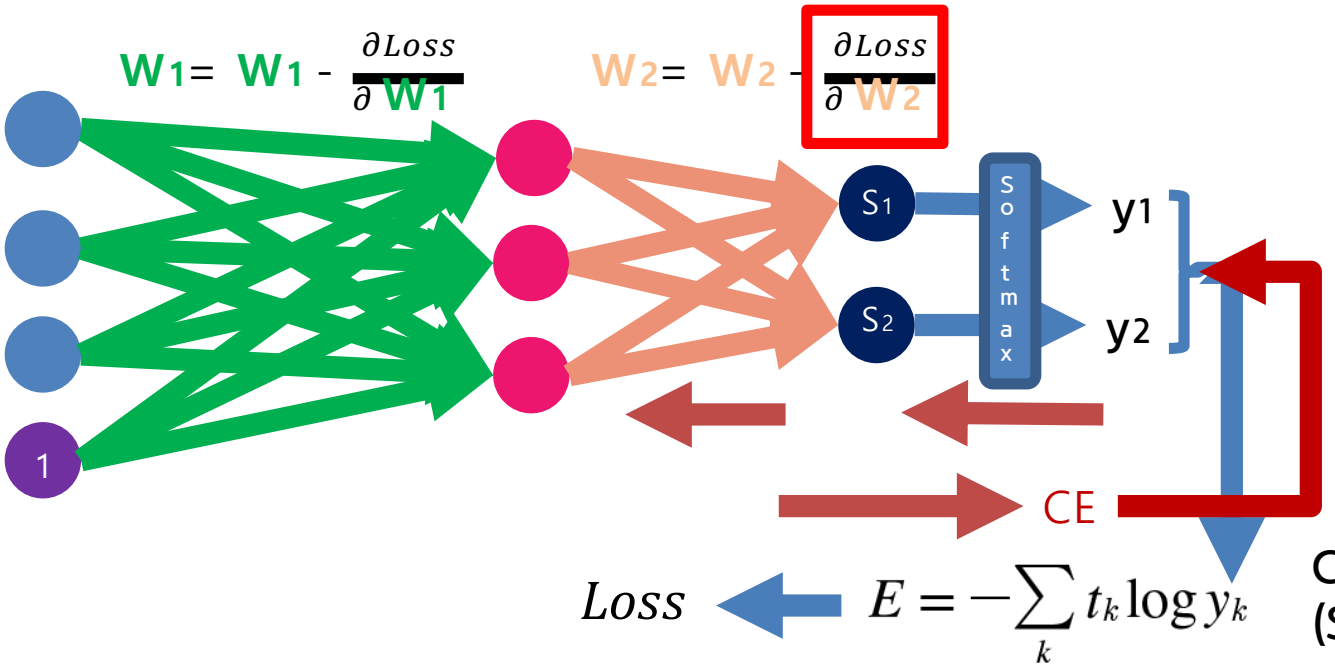
돌아보기 ..



$$\frac{\partial Loss}{\partial W_2} \rightarrow \frac{\partial Loss}{\partial CE}\frac{\partial CE}{\partial y}\frac{\partial y}{\partial \text{softmax}}$$

1. Score function
2. Loss function
3. Optimization

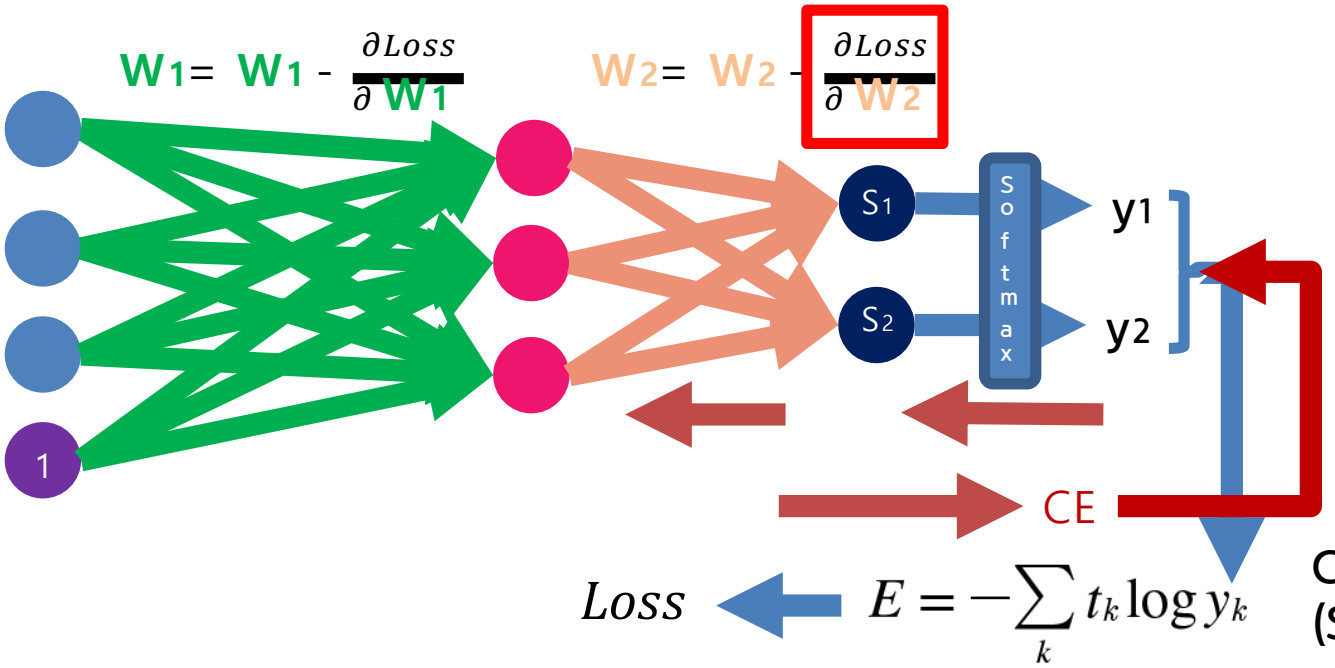
돌아보기 ..



$$\frac{\partial Loss}{\partial W_2} \rightarrow \frac{\partial Loss}{\partial CE}\frac{\partial CE}{\partial y}\frac{\partial y}{\partial softmax}\frac{\partial softmax}{\partial S}$$

1. Score function
2. Loss function
3. Optimization

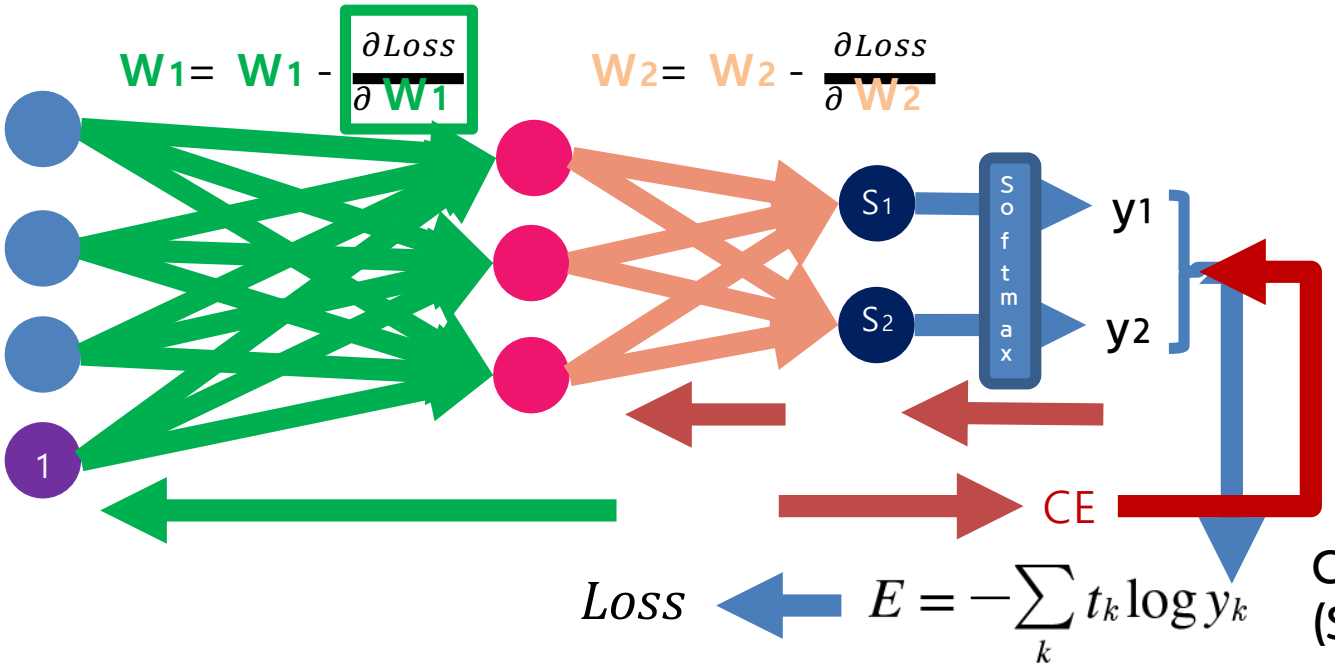
돌아보기 ..



$$\frac{\partial Loss}{\partial W_2} \rightarrow \frac{\partial Loss}{\partial CE}\frac{\partial CE}{\partial y}\frac{\partial y}{\partial softmax}\frac{\partial softmax}{\partial s}\frac{\partial s}{\partial W_2}$$

1. Score function
2. Loss function
3. Optimization

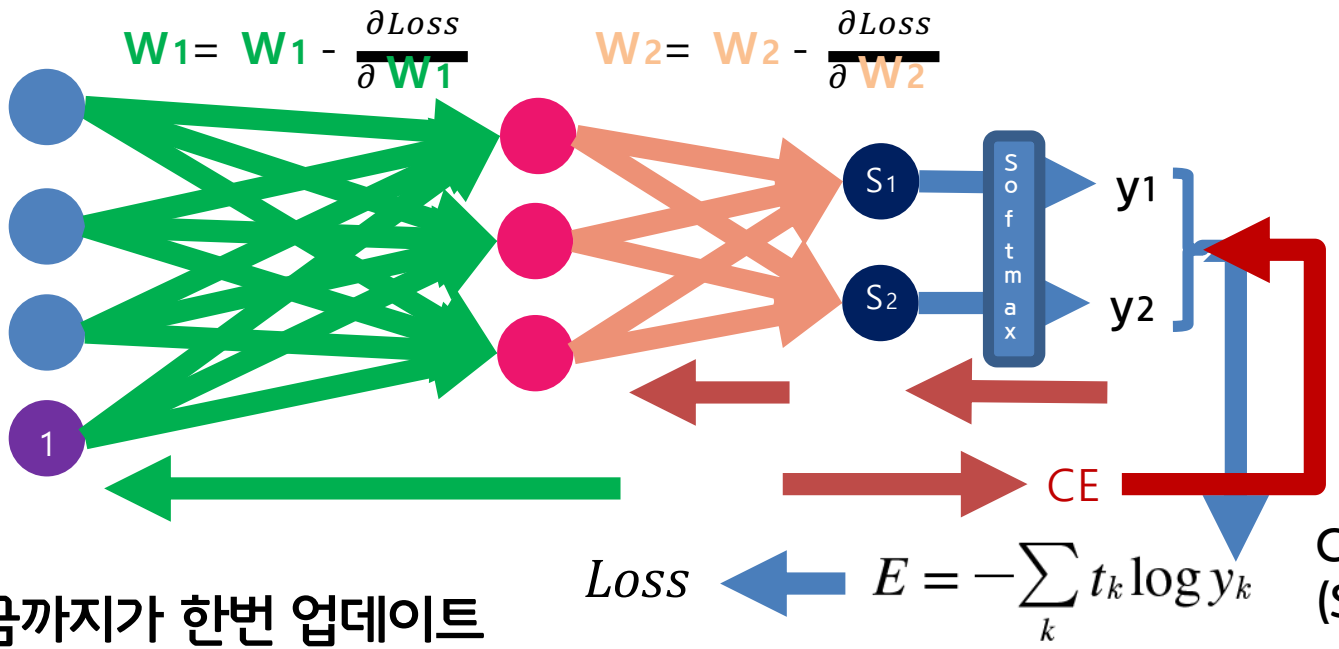
돌아보기 ..



$$\frac{\partial Loss}{\partial W_1} \rightarrow \frac{\partial Loss}{\partial CE}\frac{\partial CE}{\partial y}\frac{\partial y}{\partial softmax}\frac{\partial softmax}{\partial s}\frac{\partial s}{\partial relu}\frac{\partial relu}{\partial k}\frac{\partial k}{\partial W_1}$$

1. Score function
2. Loss function
3. Optimization

돌아보기 ..



- 지금까지가 한번 업데이트
- 모든 이미지에 대해서 반복

Multi Layer Perceptron 을 Keras로 구현해 봅시다

? 표 채우기

- 3day/prac3_mlp_mnist.py

필요한 것 Import 하기

- 모양변환 및 타입 변환
- Scale 0~1

- Categorical 변환

- ? 를 채우시오

```
8  from keras.layers import Dense, Dropout
9  from keras.optimizers import RMSprop
10
11 batch_size = 128
12 num_classes = 10
13 epochs = 20
14
15 # the data, shuffled and split between train and test sets
16 (x_train, y_train), (x_test, y_test) = mnist.load_data()
17
18 x_train = x_train.reshape(60000, 784)
19 x_test = x_test.reshape(10000, 784)
20 x_train = x_train.astype('float32')
21 x_test = x_test.astype('float32')
22 x_train /= 255
23 x_test /= 255
24 print(x_train.shape[0], 'train samples')
25 print(x_test.shape[0], 'test samples')
26
27 # convert class vectors to binary class matrices
28 y_train = keras.utils.to_categorical(y_train, num_classes)
29 y_test = keras.utils.to_categorical(y_test, num_classes)
30
31 model = Sequential()
32
33 ############################################
34 ################## Code ####################
35 ############################################
36 #### 주석에 맞게 ? 부분을 채우세요.
37 # 512개의 뉴런, activation : 'relu'의 Dense Layer
38 model.add(Dense(?, activation='?', input_shape=(784,)))
39 # 512개의 뉴런, activation : 'relu'
40 model.add(Dense(?, activation='?'))
41 # Class의 수 만큼의 뉴런, activation : 'softmax'
42 model.add(Dense(?, activation='?'))
43
44 model.summary()
45
46 # loss 는 크로스 엔트로피, Optimizer는 RMSprop(), metric : 학습이 아닌 평가할 때 사용하는 것.
47 model.compile(loss='categorical_crossentropy',
48               optimizer=RMSprop(),
49               metrics=['accuracy'])
```

모두의연구소

Convolutional Neural Networks

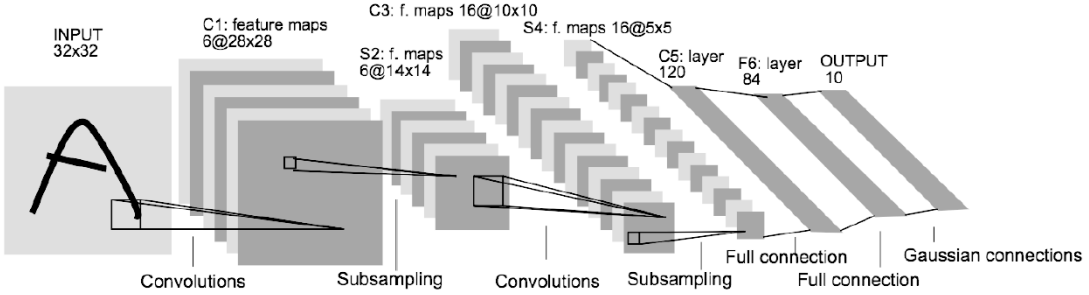
모두의연구소
박은수 책임연구원

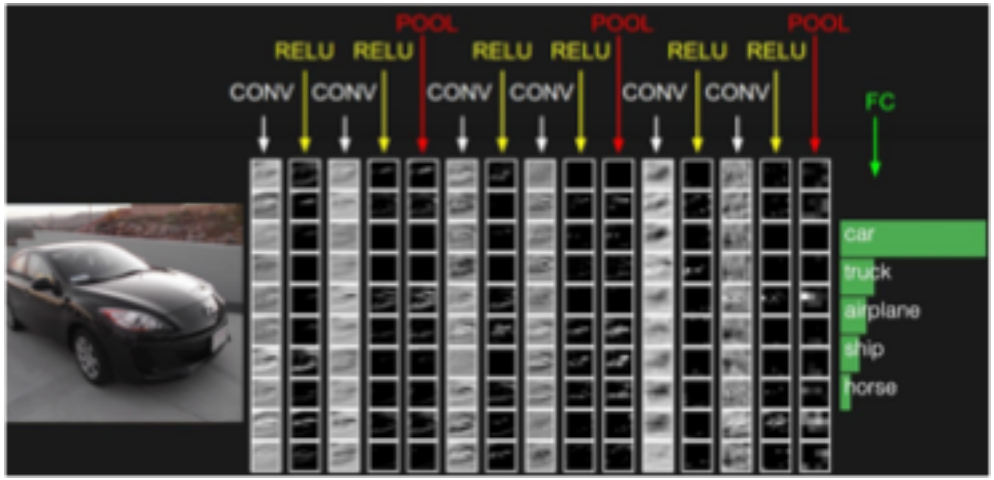
모두의연구소

큰 그림

큰 그림 : 구조

- LeNet (1998년)





전형적인 구조

CONV POOL FC

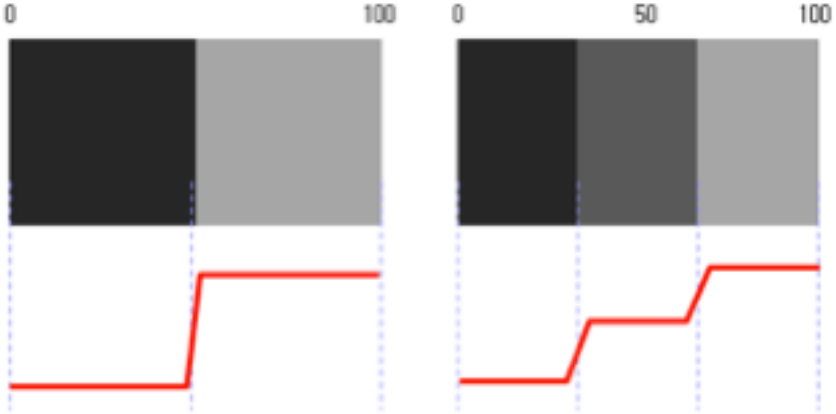
큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다

-1	0	1
-1	0	1
-1	0	1

수직 에지를 찾는
컨볼루션 필터

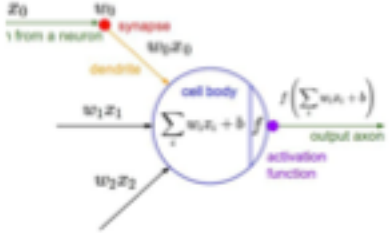


From :http://iskim3068.tistory.com/49

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

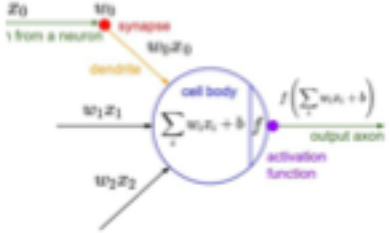
-1	0	1
-1	0	1
-1	0	1

0			

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

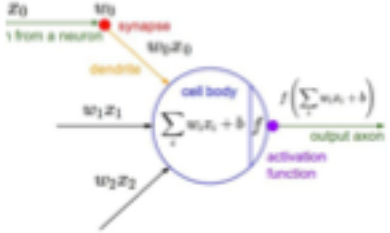
	-1	0	1		
	-1	0	1		
	-1	0	1		

0	300		

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

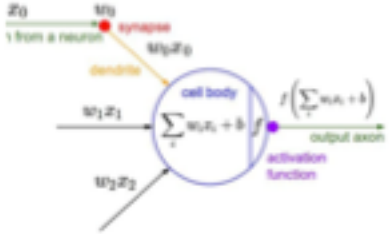
			-1	0	1
			-1	0	1
			-1	0	1

0	300	300	0

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

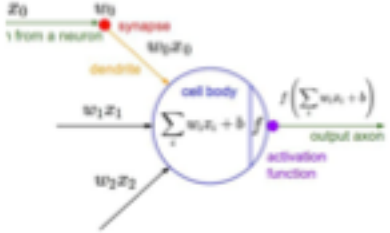
-1	0	1			
-1	0	1			
-1	0	1			

0	300	300	0
0			

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

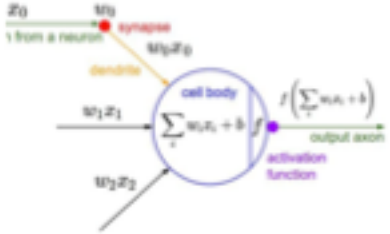
	-1	0	1		
	-1	0	1		
	-1	0	1		

0	300	300	0
0	300		

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

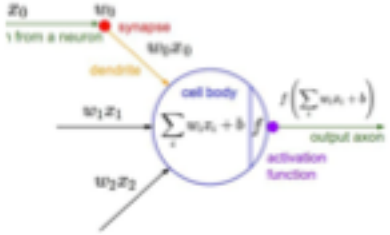
		-1	0	1	
		-1	0	1	
		-1	0	1	

0	300	300	0
0	300	300	

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다



수직 에지를 찾는
컨볼루션 필터

0 100

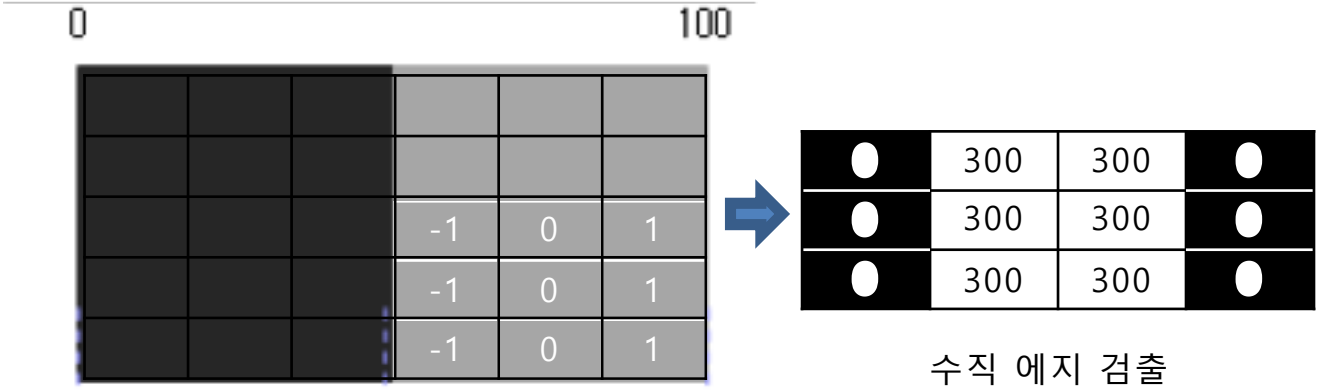
			-1	0	1
			-1	0	1
			-1	0	1

0	300	300	0
0	300	300	0
0	300	300	0

큰 그림 : 컨볼루션은 …

모두의연구소

컨볼루션 필터 : 이미지의 특징을 추출할 수 있다

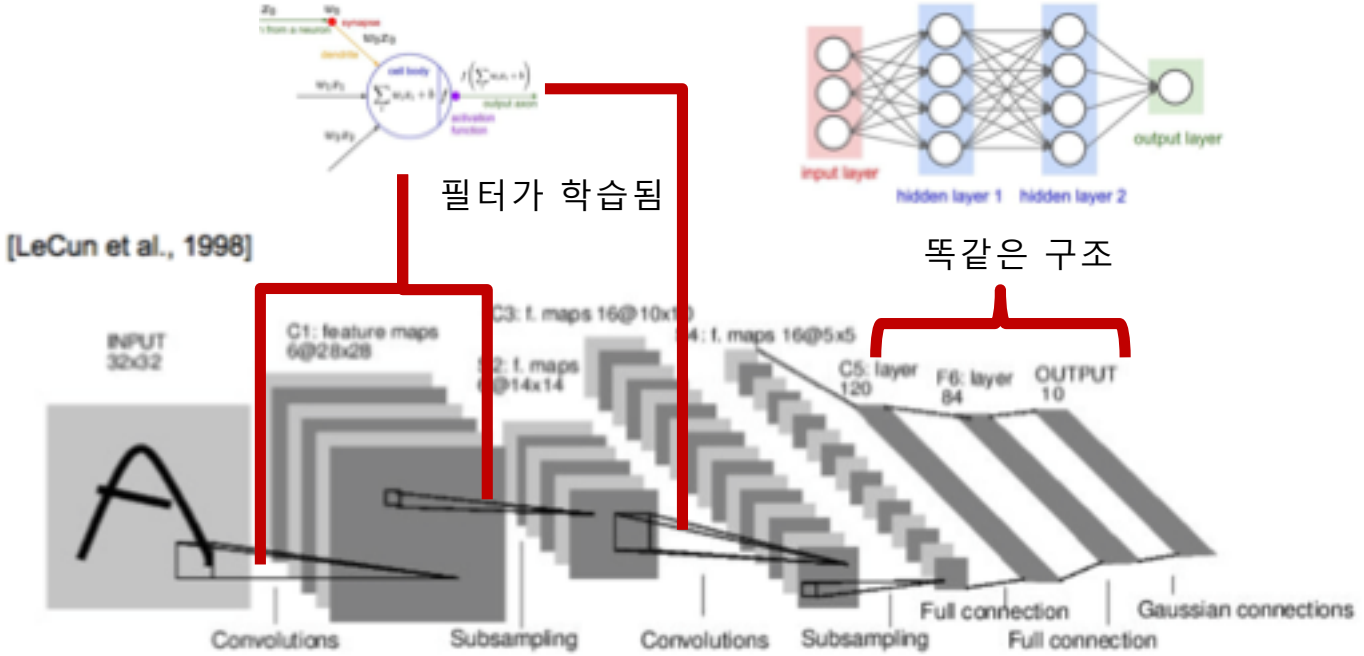


수직 에지 검출

다양한 필터로 다양한
특징을 추출할 수 있다

컨볼루셔널 뉴럴넷 (Convolutional Neural Network)

- 필터를 학습하는 구조다

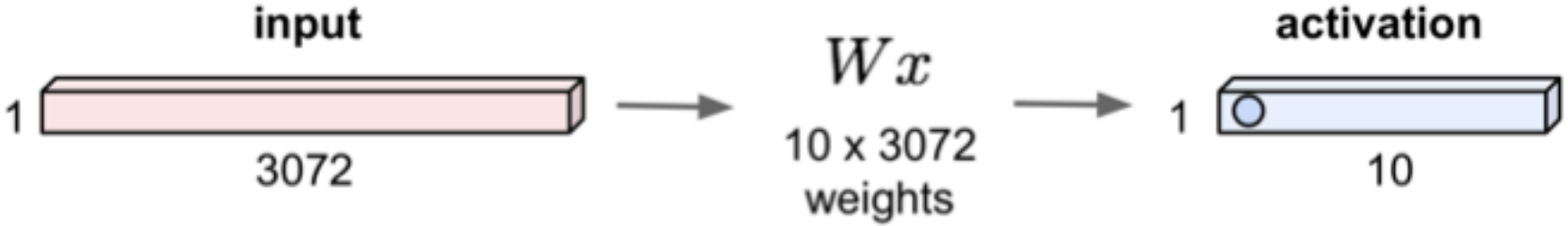


모두의연구소

이제부터
자세히 살펴봅시다

Fully Connected Layer

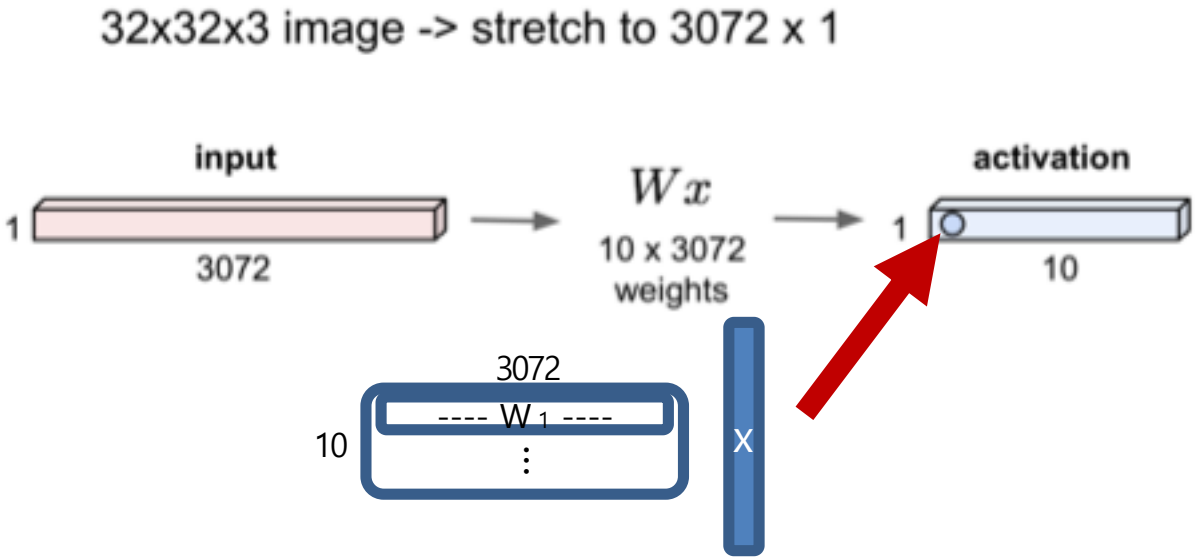
32x32x3 image -> stretch to 3072 x 1



지금까지 살펴봤펴봤던 것 입니다

Reference : Stanford University cs231n Lecture note 6

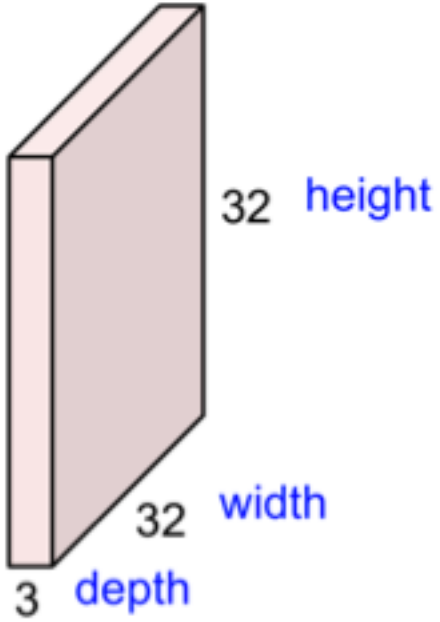
Fully Connected Layer



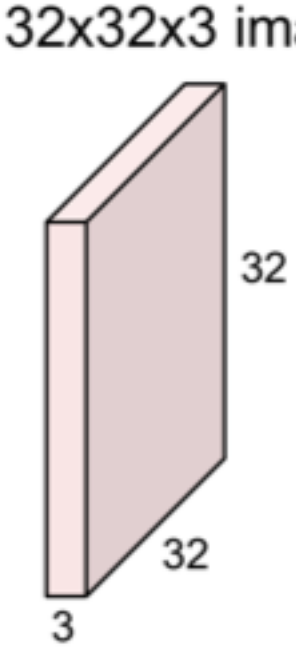
1) 전체 영상 각 픽셀에 W_1의 weighted sum으로 계산한 score
2) Activation 적용

Convolution Layer

32x32x3 image -> preserve spatial structure



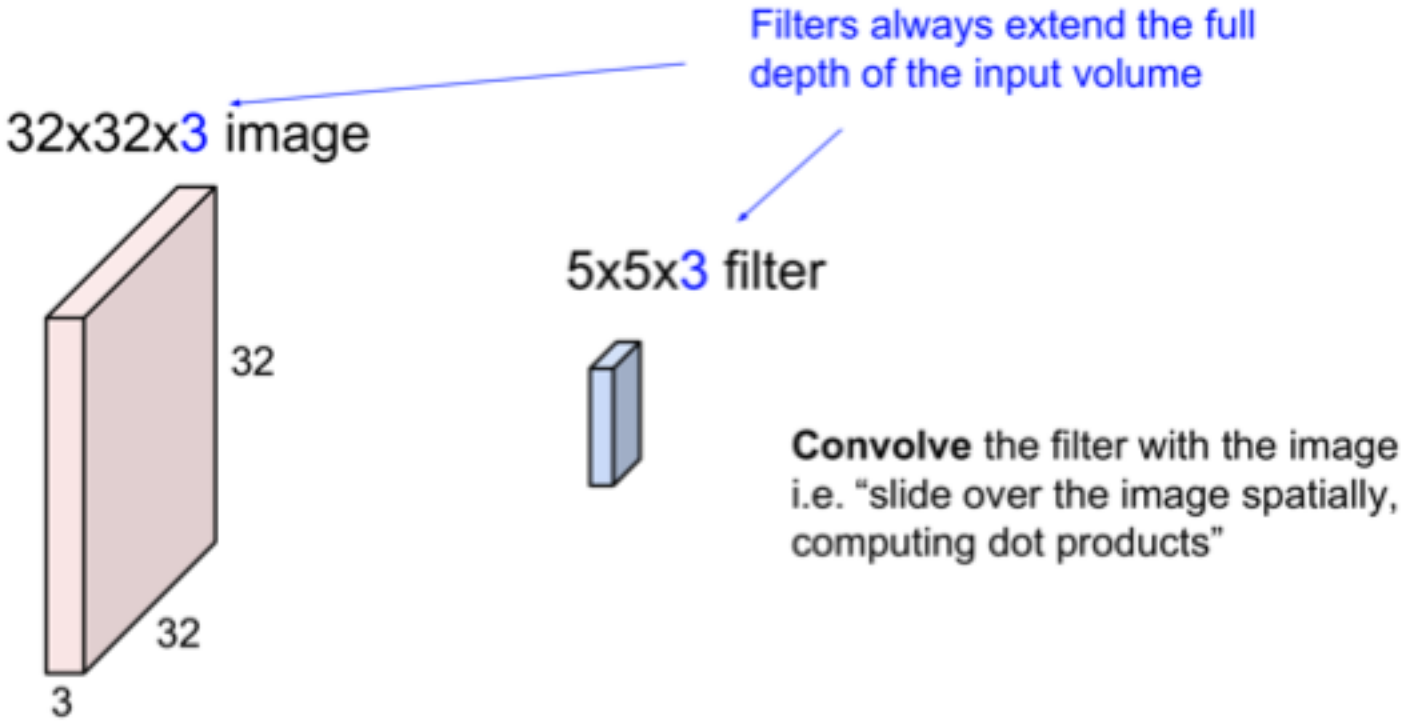
Convolution Layer



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

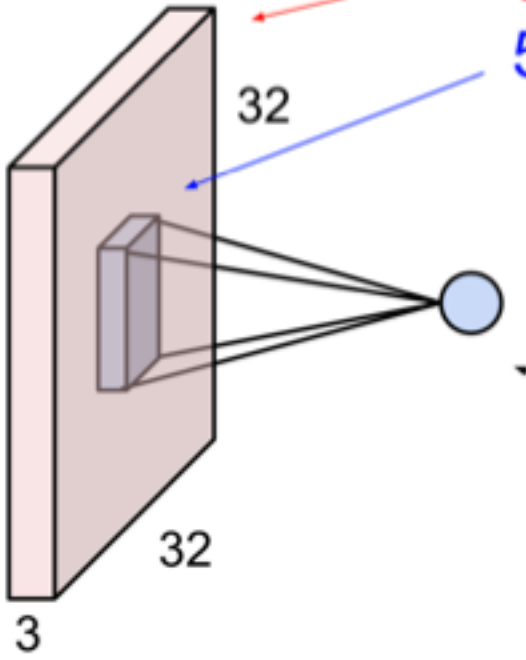
Reference : Stanford University cs231n Lecture note 6

Convolution Layer



Reference : Stanford University cs231n Lecture note 6

Convolution Layer



32x32x3 image

5x5x3 filter w

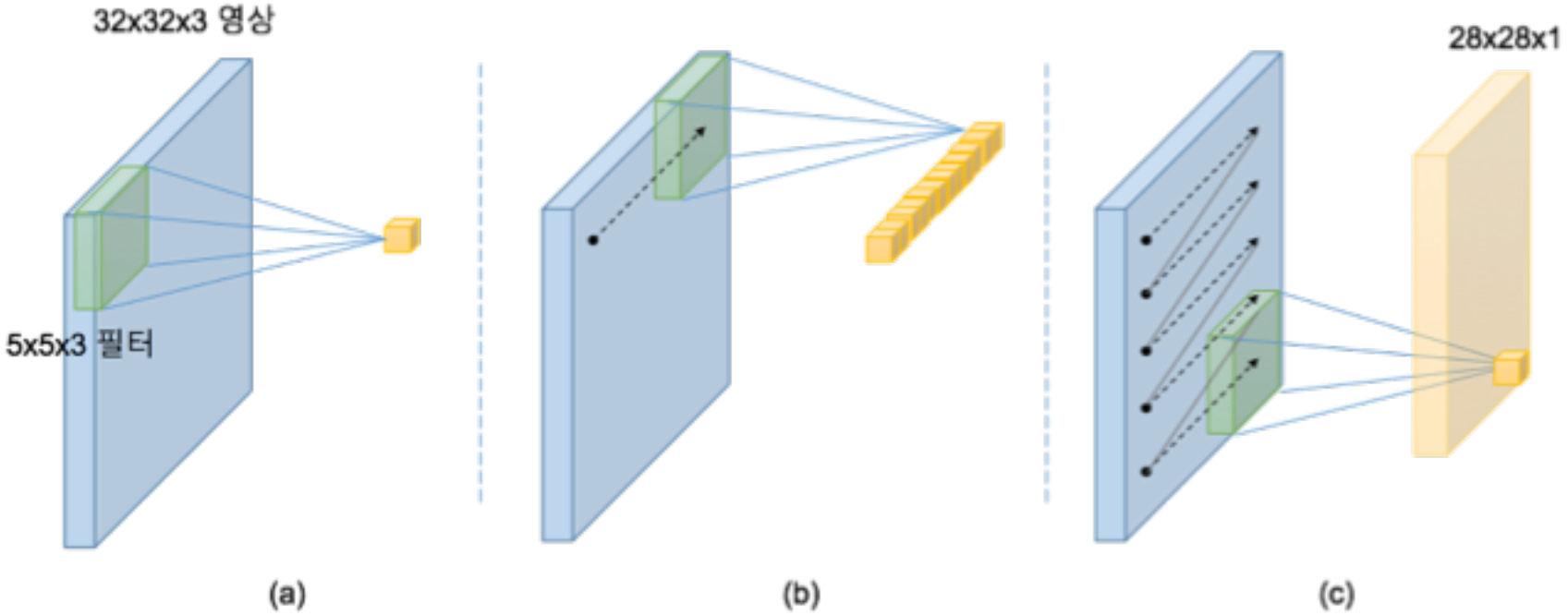
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Reference : Stanford University cs231n Lecture note 6

Convolution Layer

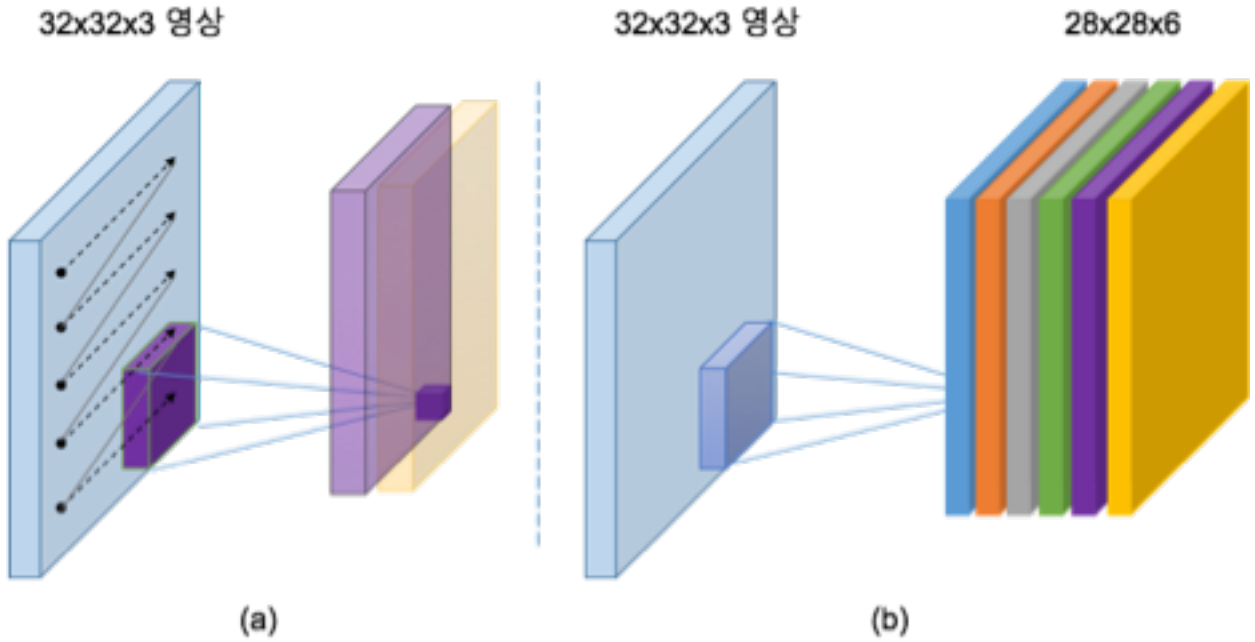
모두의연구소



Convolution Layer

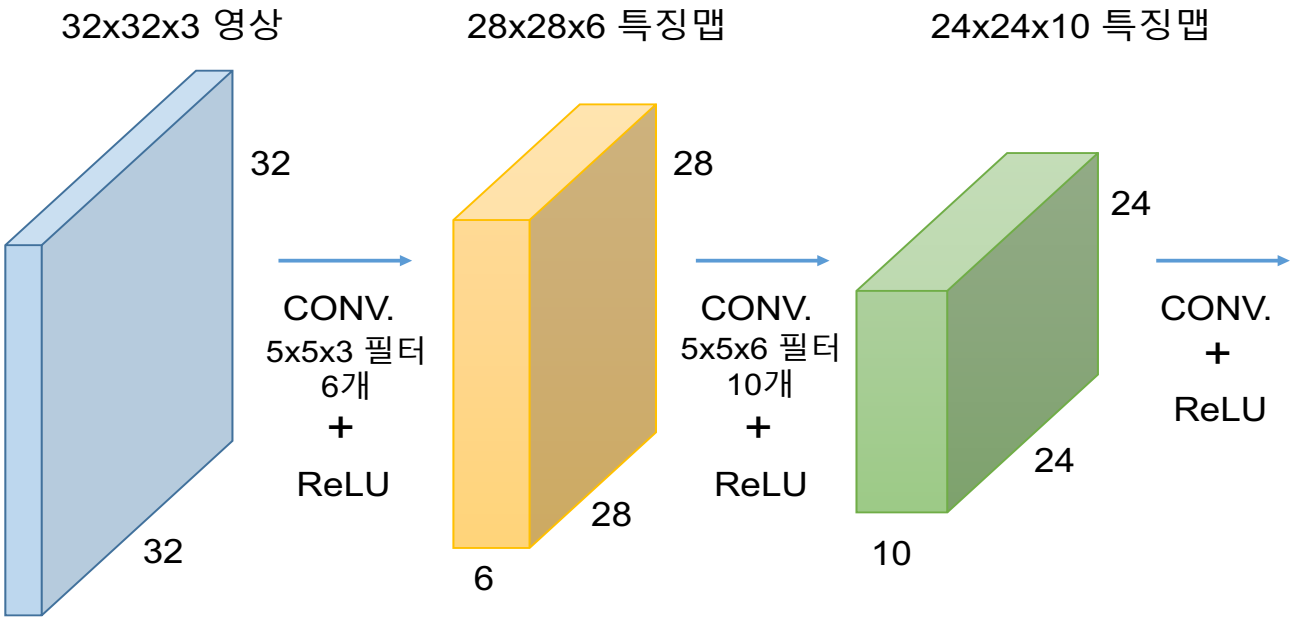
모두의연구소

똑같은 크기의 필터 6개를 더 만들어 봅시다



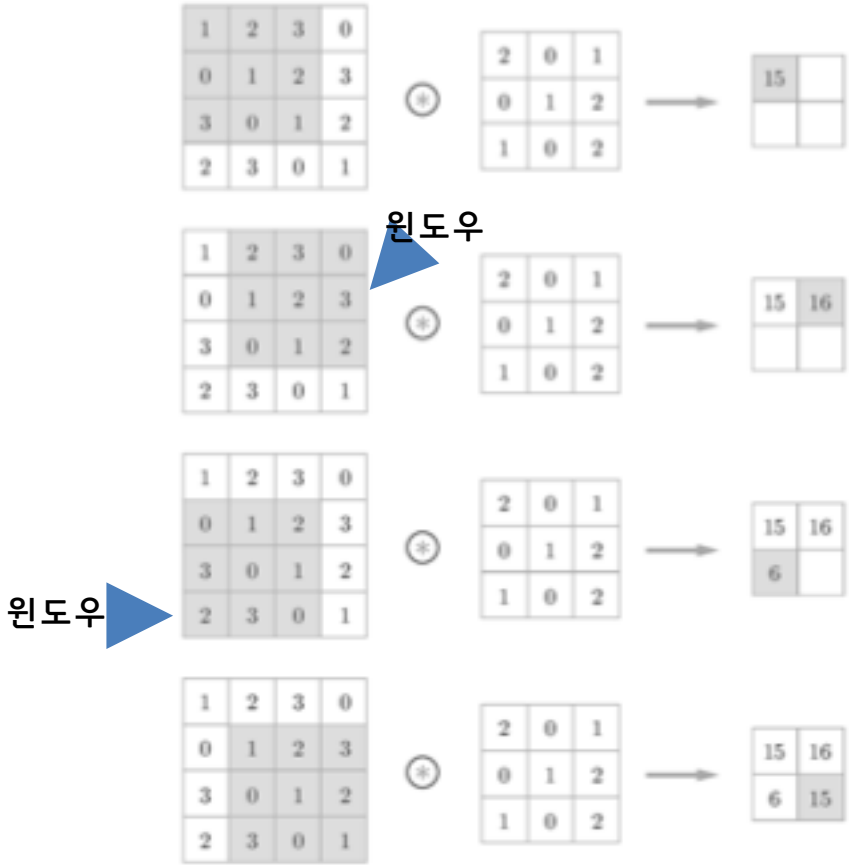
Convolution Layer

컨볼루션 네트워크는 활성화 함수를 포함한 컨볼루션 레이어의 연결 입니다



Convolution Layer

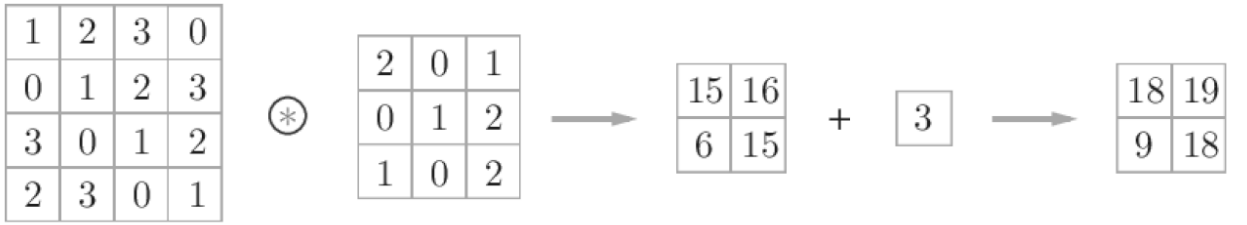
- 컨볼루션 연산
  - 1) 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용
  - 2) 입력과 필터에 대응하는 원소끼리 곱한 후 그 총합을 함 (단일 곱셈-누산 (fused multiply-add, FMA)
  - Bias 파라미터 존재함



Convolution Layer

모두의연구소

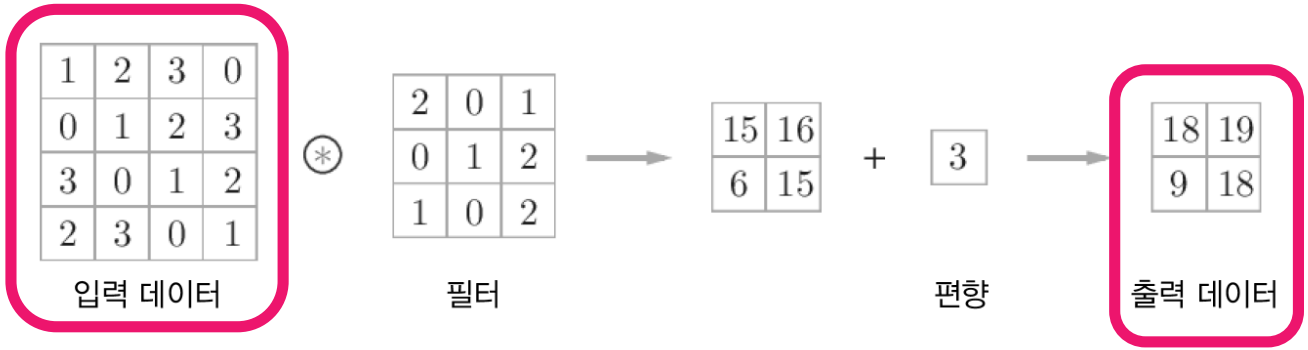
- **컨볼루션 연산**



Convolution Layer

모두의연구소

- **컨볼루션 연산**

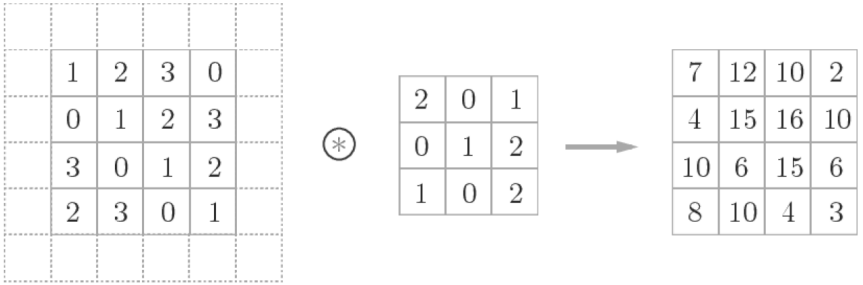


크기가 줄어드는 군요

Convolution Layer

모두의연구소

- 패딩 (padding)
  - 컨볼루션 연산의 패딩 처리 : 입력 데이터 주위에 0을 채운다 (패딩은 점선으로 표시했으며 그 안의 값 '0'은 생략함)



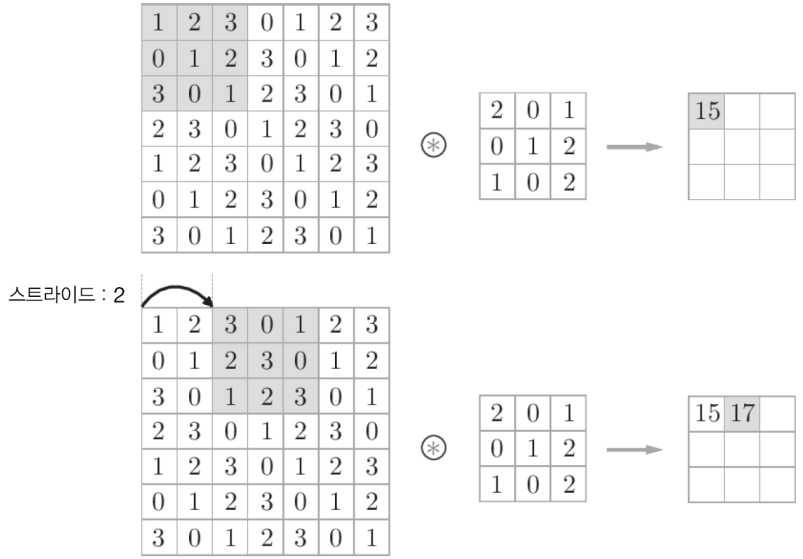
  - 패딩은 출력(특징맵(feature map))의 크기를 유지 시키고자 할때 주로 사용함

Convolution Layer

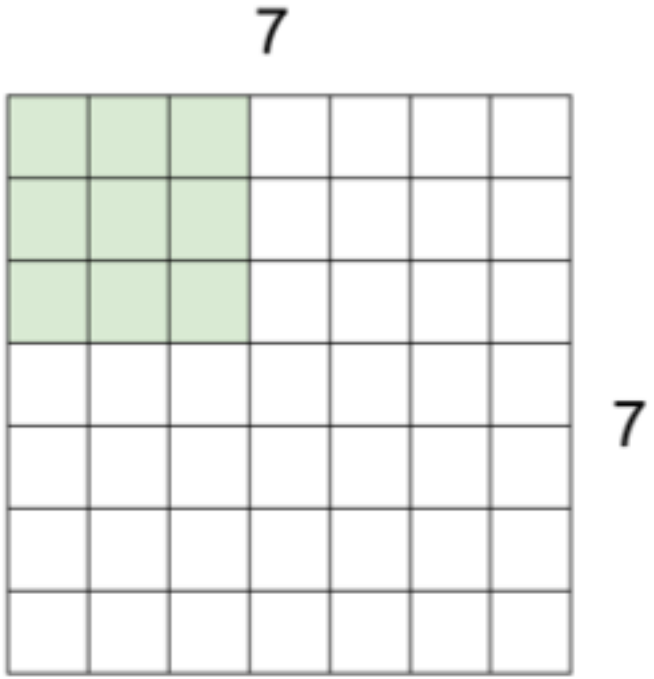
모두의연구소

- 스트라이드 (stride)
  - 필터 적용하는 위치의 간격
    - 스트라이드를 2로 하면 필터를 적용하는 윈도우가 두 칸씩 이동함

- 스트라이드를 2로 하니 출력이 3x3이 됨



특징맵의 크기변화

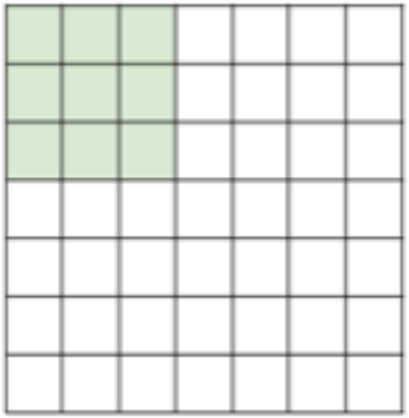


7x7 input (spatially)
assume 3x3 filter

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

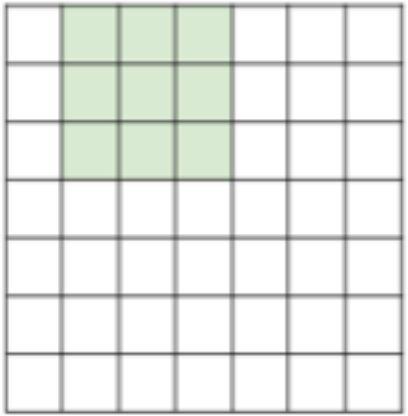


7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

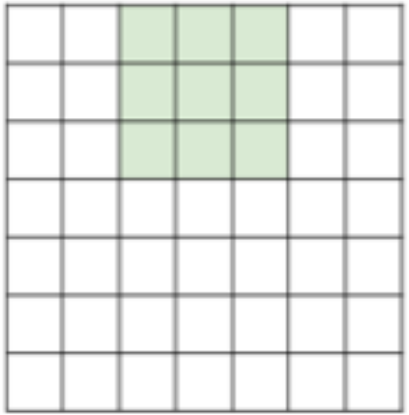


7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

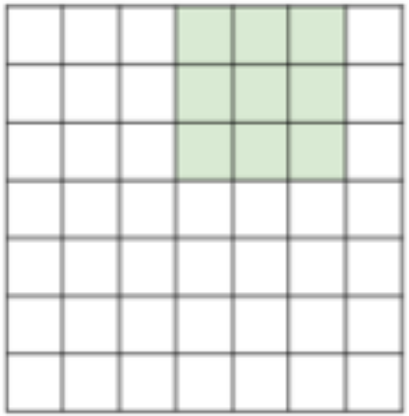


7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

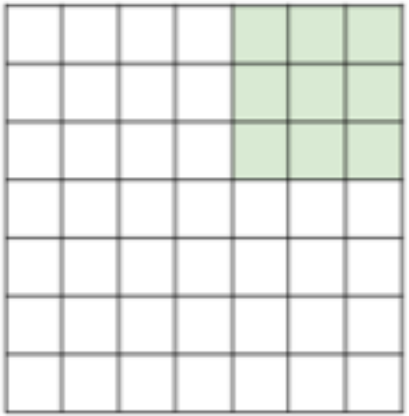


7x7 input
assume 3x3 connectivity, stride 1

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**

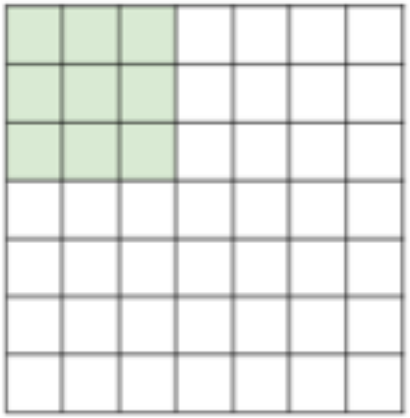


7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

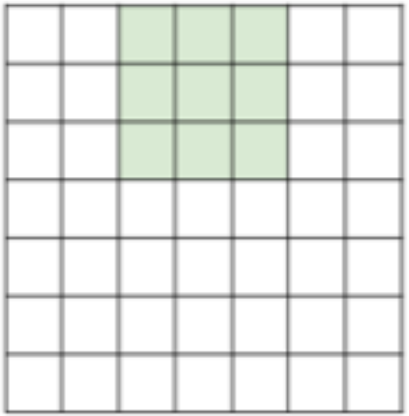
what about stride 2?

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



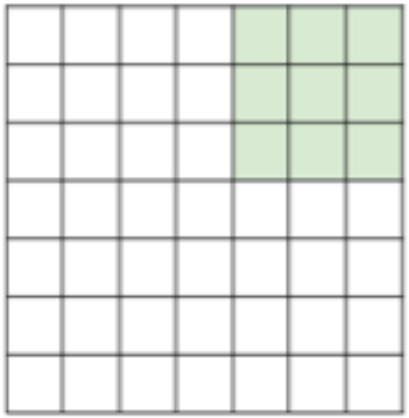
7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

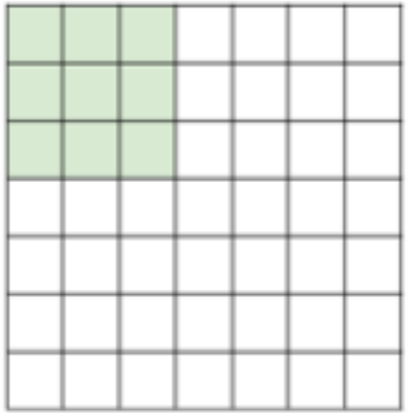
what about stride 2?
=> **3x3 output**

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

Replicate this column of hidden neurons across space, with some **stride.**



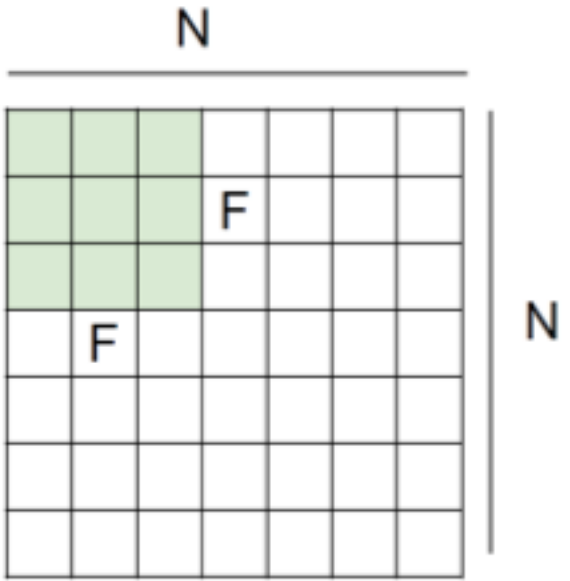
7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

what about stride 2?
=> **3x3 output**

what about stride 3? Cannot.

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화



Output size:
(N - F) / stride + 1

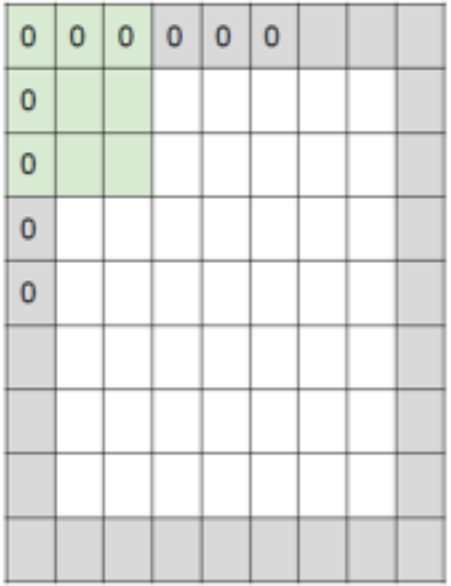
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = … :\

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

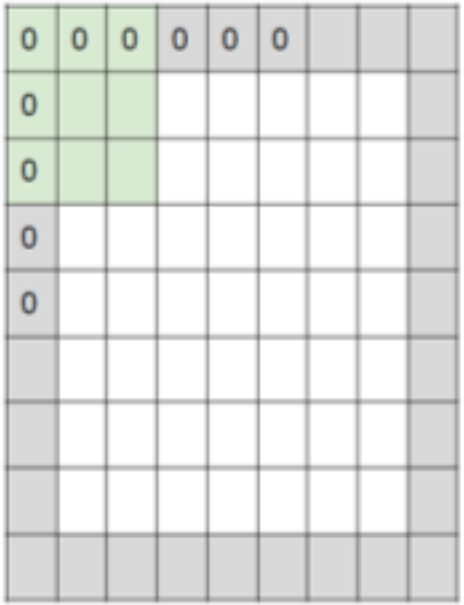
(recall:)
(N - F) / stride + 1

Reference : Stanford University cs231n Lecture note 6

특징맵의 크기변화

모두의연구소

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

특징맵의 크기변화

모두의연구소

- 출력크기 계산해 보기

- 입력크기 : (*H*, *W*)
- 필터크기 : (*FH*, *FW*)
- 출력크기 : (*OH*, *OW*)
- 패딩 : *P*
- 스트라이드 : *S*

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - FW}{S} + 1$$

특징맵의 크기변화

모두의연구소

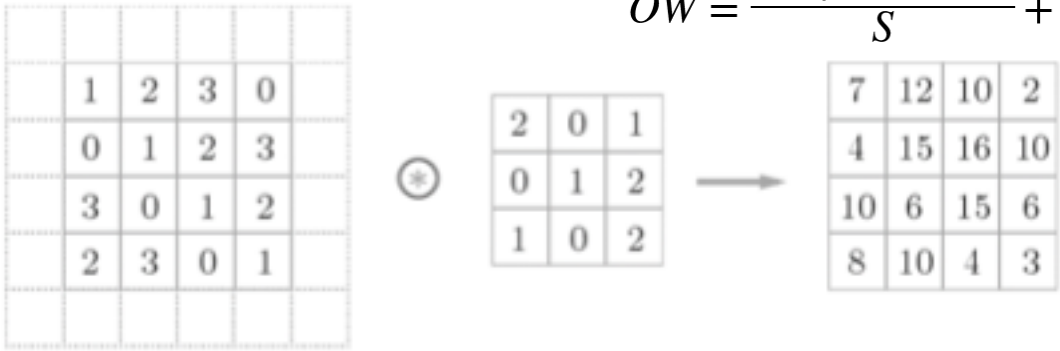
- **출력크기 계산해 보기**

입력 : (4,4), 패딩 : 1 , 스트라이드 : 1, 필터: (3,3)

$$OH = \frac{4 + 2 \cdot 1 - 3}{1} + 1 = 4$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$



(4, 4) 입력 데이터(패딩 : 1) | (3, 3) 필터 | (4, 4) 출력 데이터

특징맵의 크기변화

모두의연구소

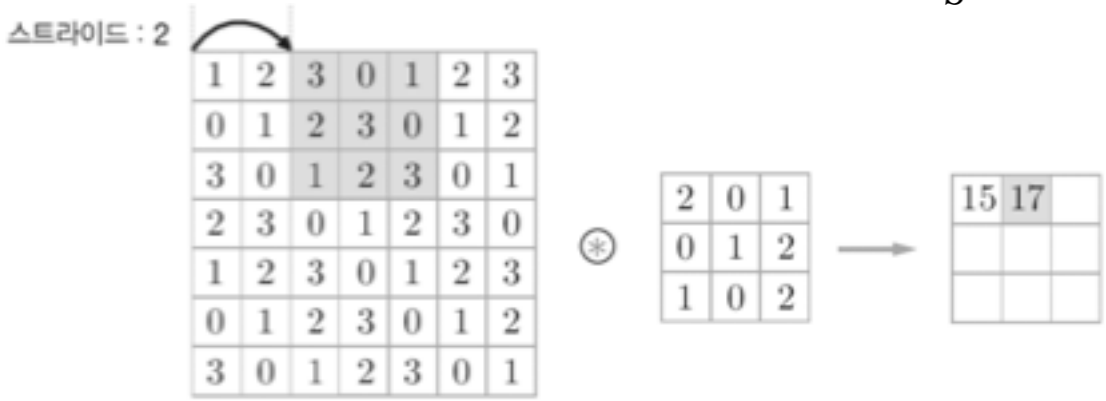
- **출력크기 계산해 보기**

입력 : (7,7), 패딩 : 0 , 스트라이드 : 2, 필터: (3,3)

$$OH = \frac{7 + 2 \cdot 0 - 3}{2} + 1 = 3$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$



특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

$$OH = ?$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = ?$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$

특징맵의 크기변화

모두의연구소

- **출력크기 계산해 보기**

입력 : (28, 31), 패딩 : 2 , 스트라이드 : 3, 필터: (5, 5)

$$OH = \frac{28 + 2 \cdot 2 - 5}{3} + 1 = 10$$

$$OW = \frac{31 + 2 \cdot 2 - 5}{3} + 1 = 11$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{\mathrm{W} + 2P - F\mathrm{W}}{S} + 1$$

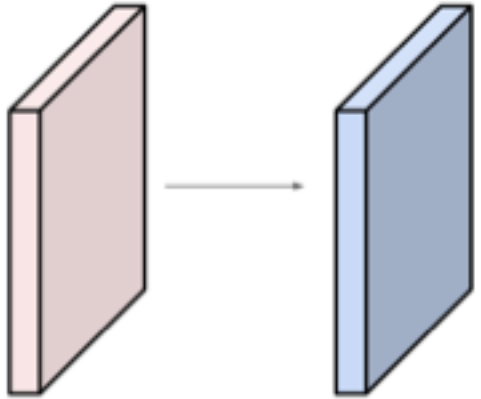
- ***OH, OW*가 정수가 아니면?**
  - 출력이 안나오는 것임. 오류를 내는 등의 대응 필요
  - 딥러닝 프레임웍은 가까운 정수로 내림 하는 등, 특별히 에러를 내지않고 진행되도록 구현되는 경우가 많음

특징맵의 크기변화

모두의연구소

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

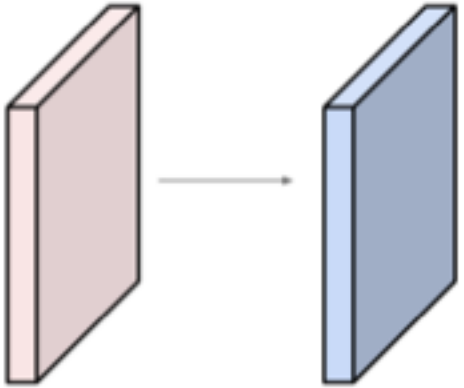


Number of parameters in this layer?

특징맵의 크기변화

모두의연구소

Examples time:



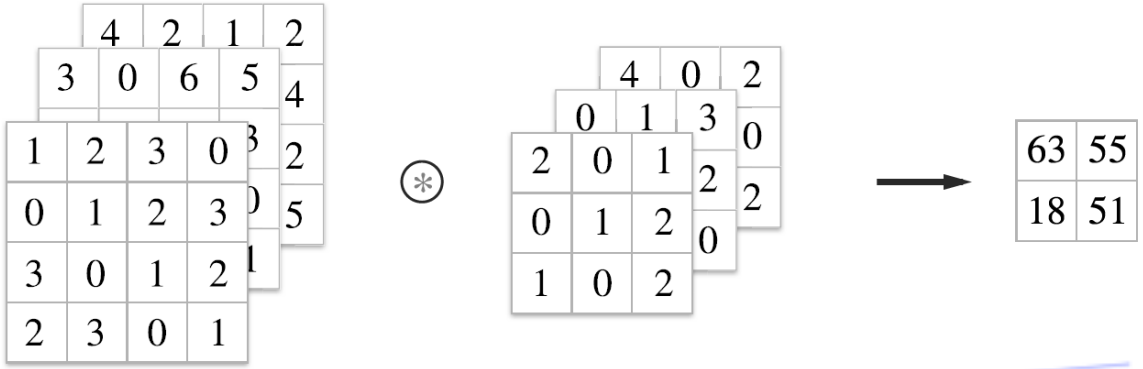
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

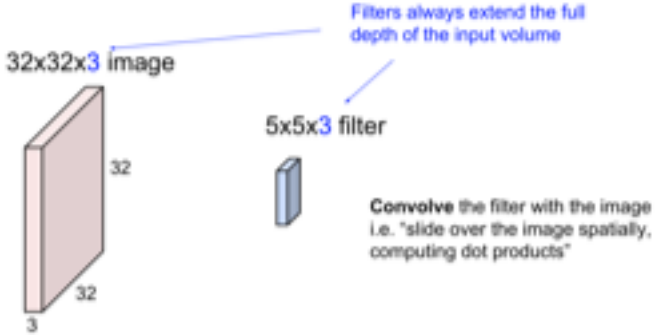
Convolution Layer

모두의연구소

- 필터의 채널 수는 입력되는 특징맵의 채널 수와 일치해야 함을 잊지 마세요



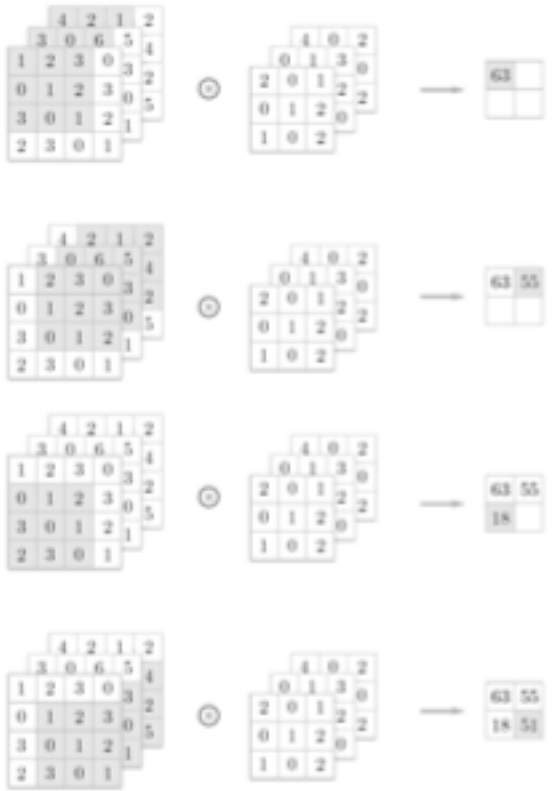
입력 데이터 필터



Convolution Layer

모두의연구소

- 3차원 데이터의 컨볼루션
  - 주의할 점은 **입력데이터의 채널 수와 필터의 채널 수가 같아야** 한다는 점
  - 각 필터의 채널크기는 같아야 함

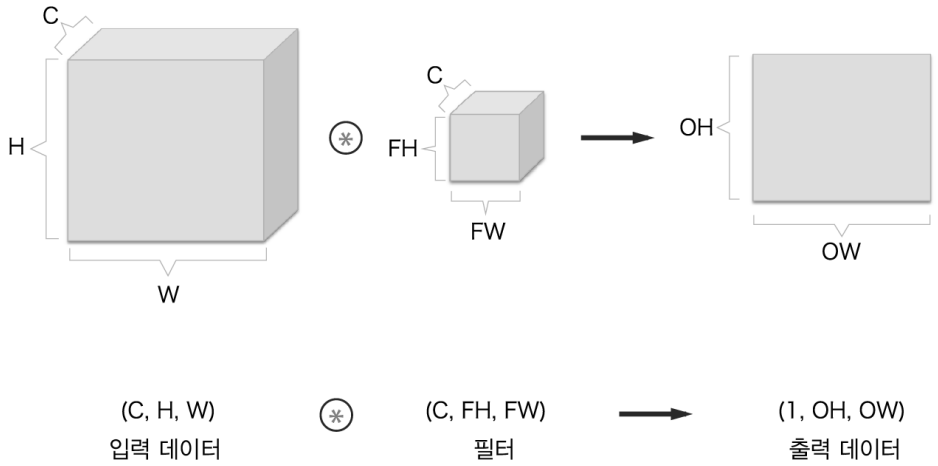


Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution Layer

모두의연구소

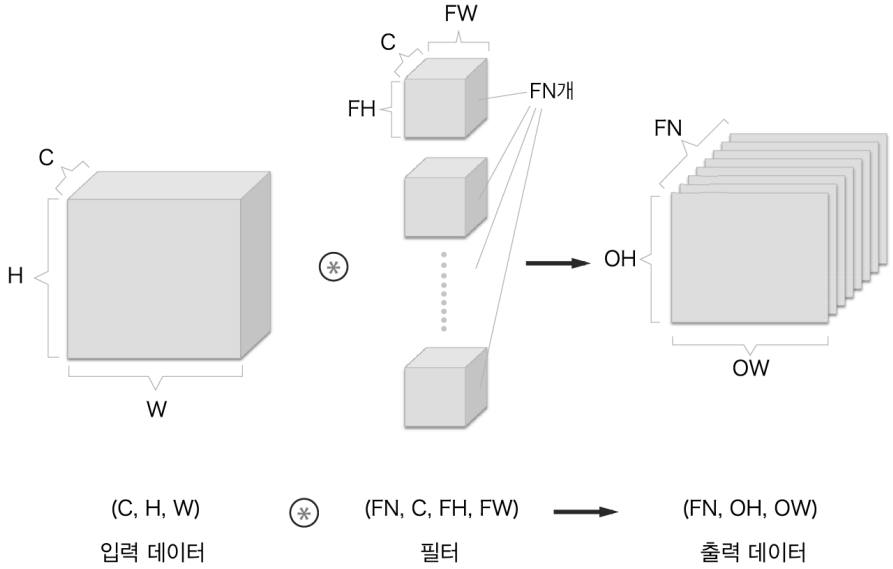
- 블록으로 생각하기



- 출력 데이터는 한장의 특징맵
- 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?

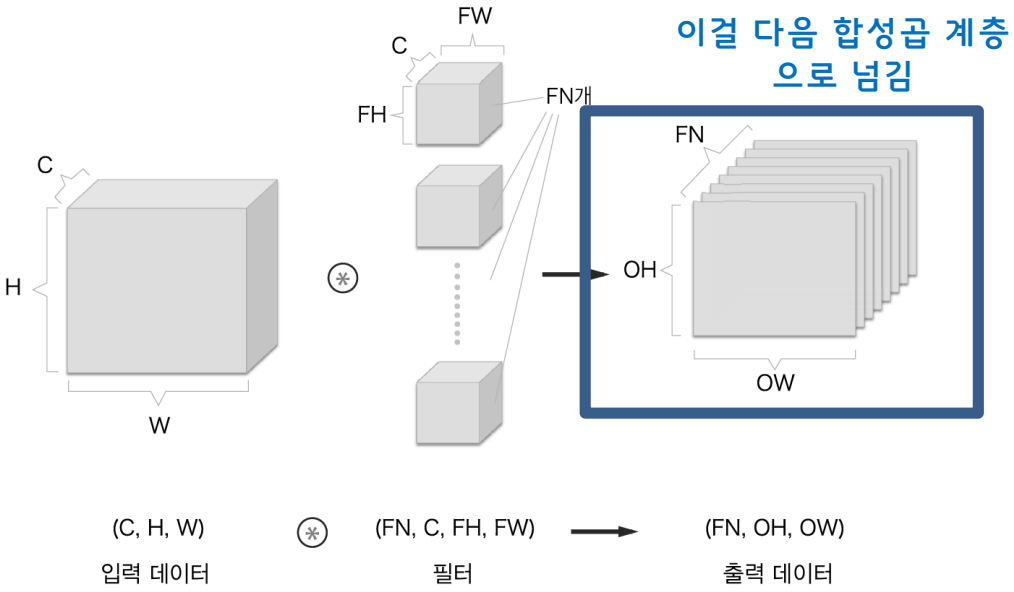
Convolution Layer

- 블록으로 생각하기
  - 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?
    - 필터를 여러개 사용하면 됨



Convolution Layer

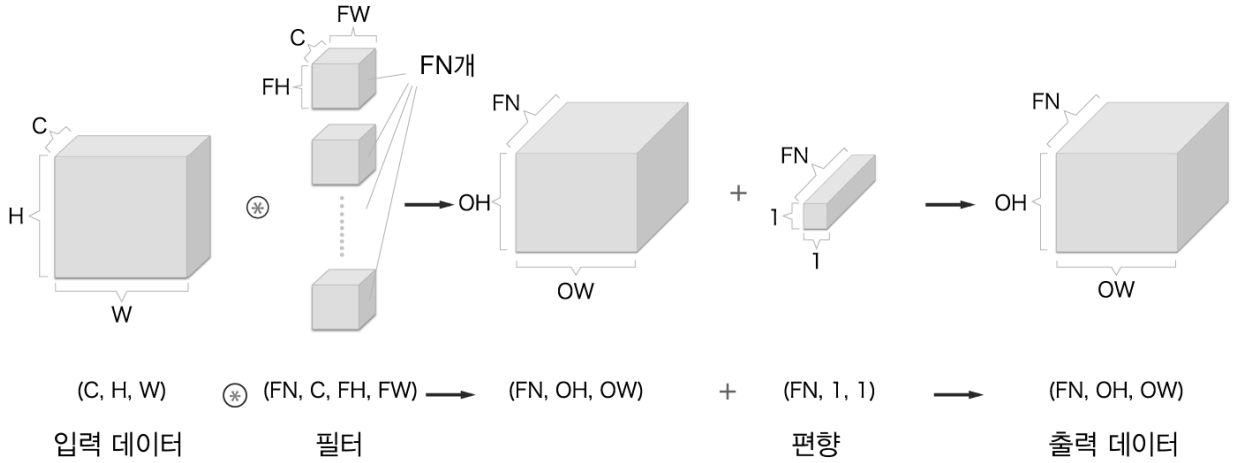
- 블록으로 생각하기
  - 한장이 아니라 여러장, 즉 다수의 채널을 내보내려면?
    - **필터를 여러개 사용하면 됨**



Convolution Layer

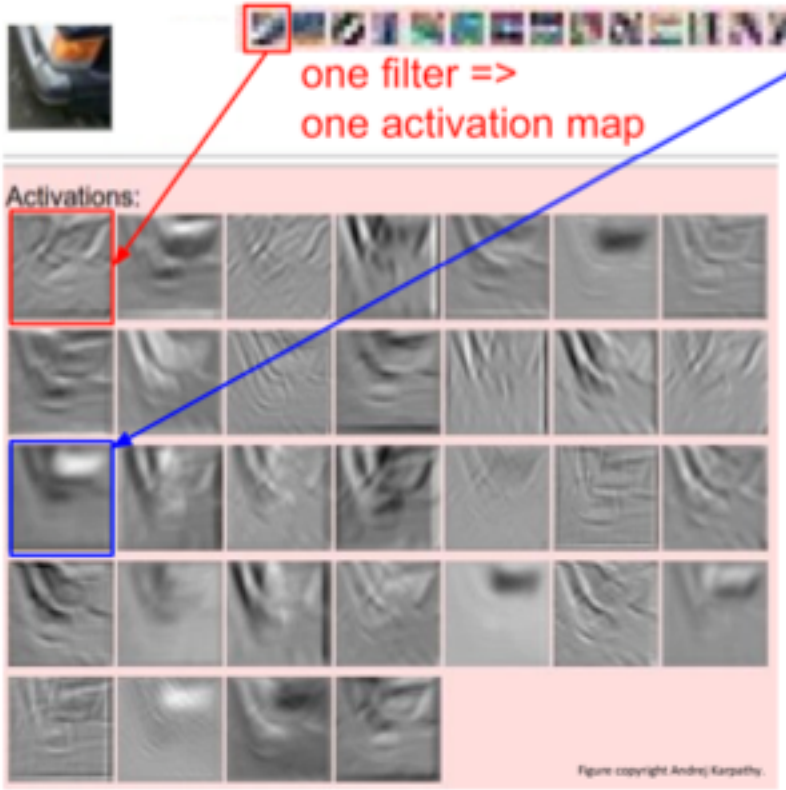
모두의연구소

- 편향 (bias)



- 필터마다 편향이 하나씩 존재

Convolution Layer



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

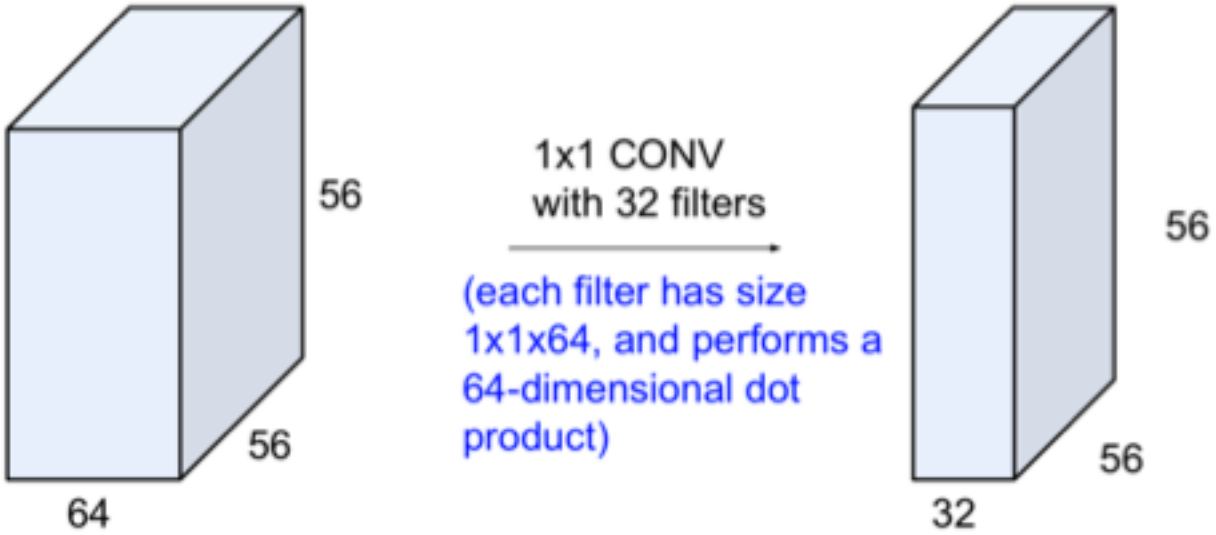
elementwise multiplication and sum of a filter and the signal (image)

Reference : Stanford University cs231n Lecture note 6

Convolution Layer

모두의연구소

(btw, 1x1 convolution layers make perfect sense)



Convolution Layer

모두의연구소

Example: CONV layer in Torch

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

SpatialConvolution

```
module = nn.SpatialConvolution(nInputPlane, nOutputPlane, kW, kH, [dW], [dH], [padW], [padH])
```

Applies a 2D convolution over an input image composed of several input planes. The `input` tensor in `forward(input)` is expected to be a 3D tensor (`nInputPlane x height x width`).

The parameters are the following:

- `nInputPlane` : The number of expected input planes in the image given into `forward()`.
- `nOutputPlane` : The number of output planes the convolution layer will produce.
- `kW` : The kernel width of the convolution
- `kH` : The kernel height of the convolution
- `dW` : The step of the convolution in the width dimension. Default is `1`.
- `dH` : The step of the convolution in the height dimension. Default is `1`.
- `padW` : The additional zeros added per width to the input planes. Default is `0`, a good number is `(kW-1)/2`.
- `padH` : The additional zeros added per height to the input planes. Default is `padW`, a good number is `(kH-1)/2`.

Note that depending of the size of your kernel, several (of the last) columns or rows of the input image might be lost. It is up to the user to add proper padding in images.

If the input image is a 3D tensor `nInputPlane x height x width`, the output image size will be `nOutputPlane x oheight x owidth` where

```
owidth  = floor((width  + 2*padW - kW) / dW + 1)
oheight = floor((height + 2*padH - kH) / dH + 1)
```


Reference : Stanford University cs231n Lecture note 6

Convolution Layer

모두의연구소

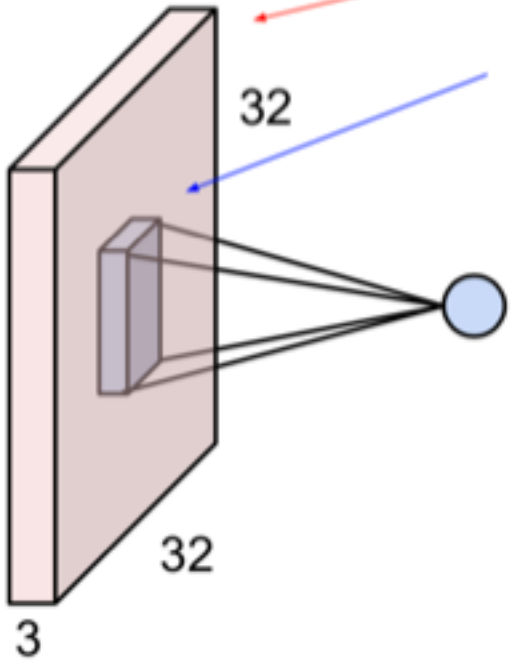
Example: CONV layer in Caffe

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  # learning rate and decay multipliers for the filters
  param { lr_mult: 1 decay_mult: 1 }
  # learning rate and decay multipliers for the biases
  param { lr_mult: 2 decay_mult: 0 }
  convolution_param {
    num_output: 96     # learn 96 filters
    kernel_size: 11    # each filter is 11x11
    stride: 4          # step 4 pixels between each filter application
    weight_filler {
      type: "gaussian" # initialize the filters from a Gaussian
      std: 0.01        # distribution with stdev 0.01 (default mean: 0)
    }
    bias_filler {
      type: "constant" # initialize the biases to zero (0)
      value: 0
    }
  }
}
```

Summary. To summarize, the Conv Layer:

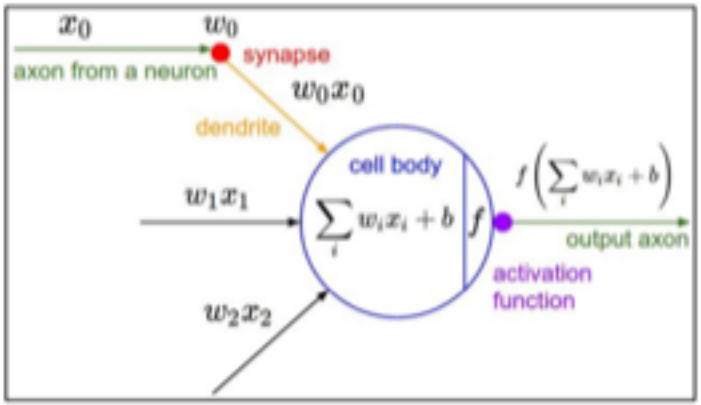
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.

The brain/neuron view of CONV Layer



32x32x3 image

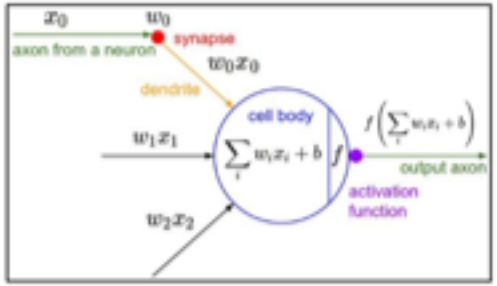
5x5x3 filter

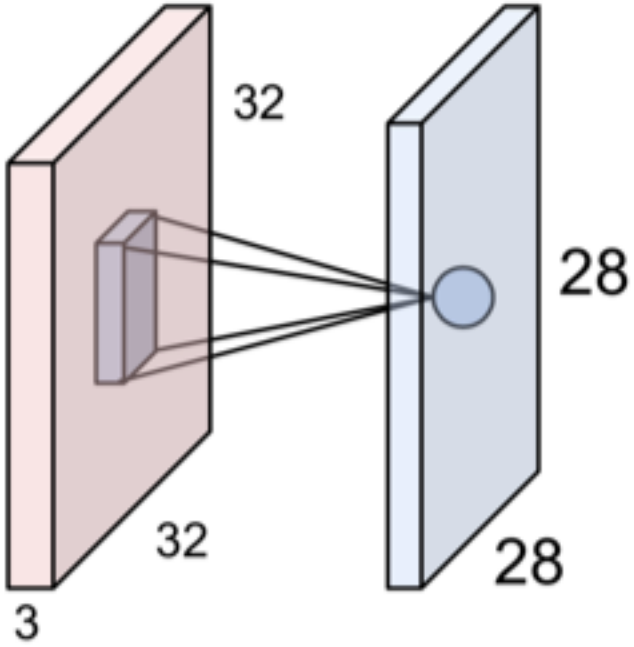


It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

The brain/neuron view of CONV Layer



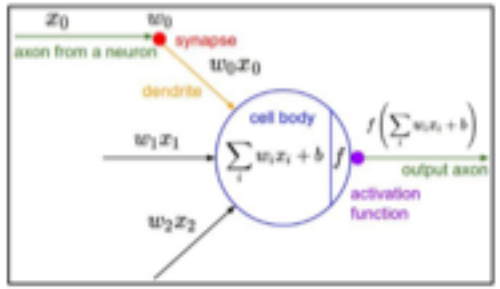


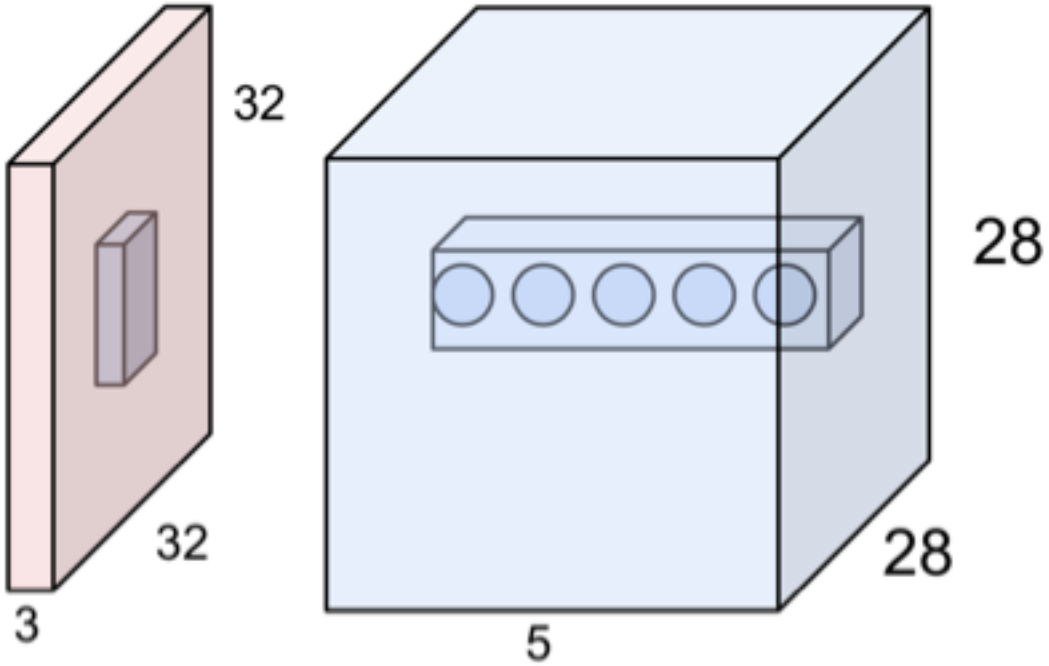
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

The brain/neuron view of CONV Layer





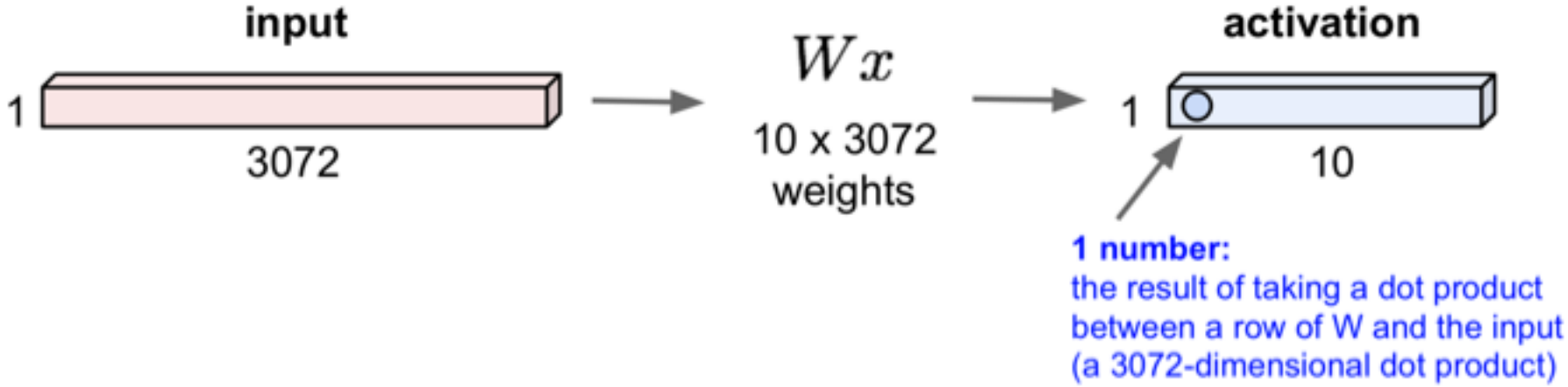
E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

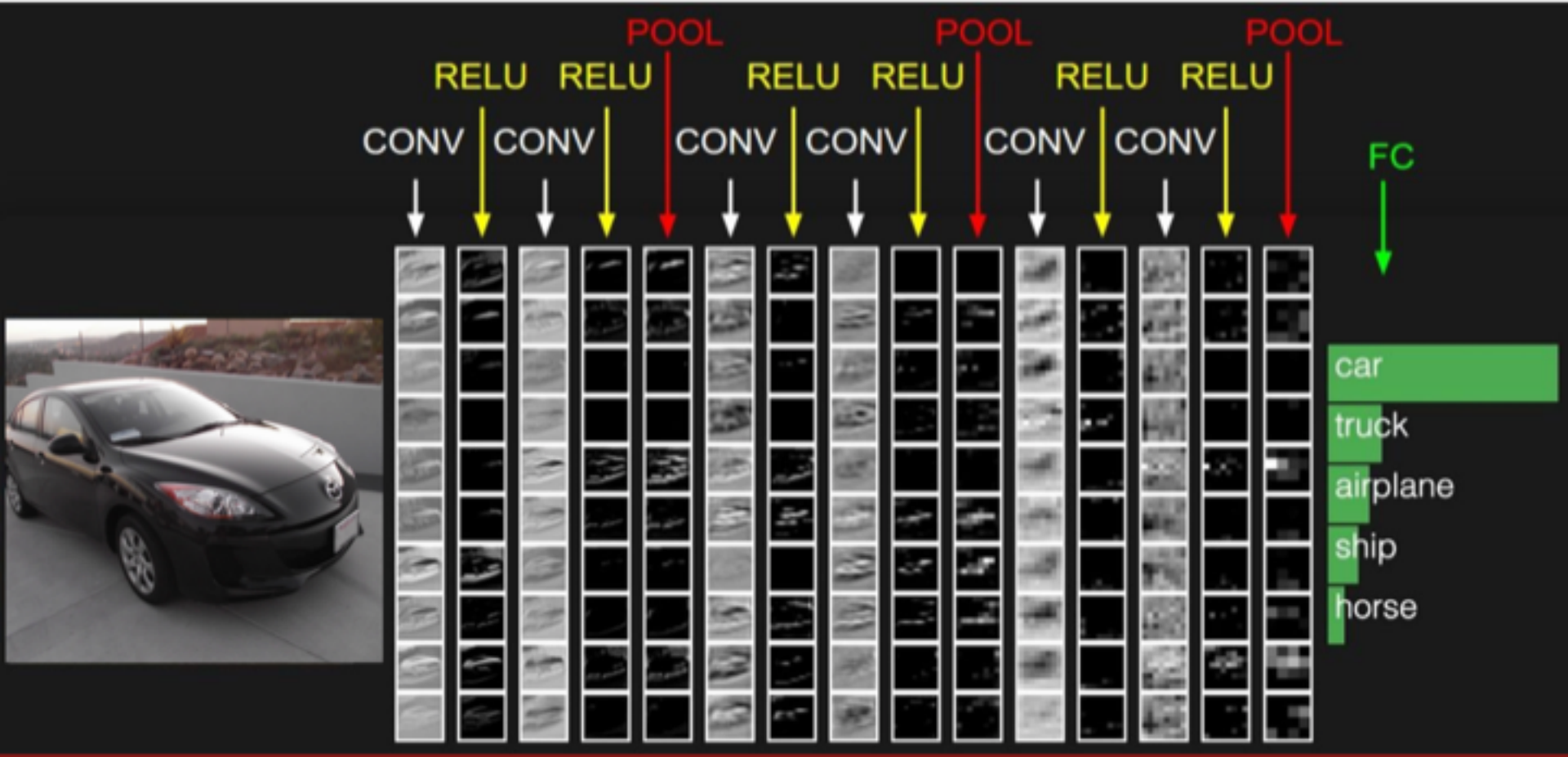
Reminder: Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume

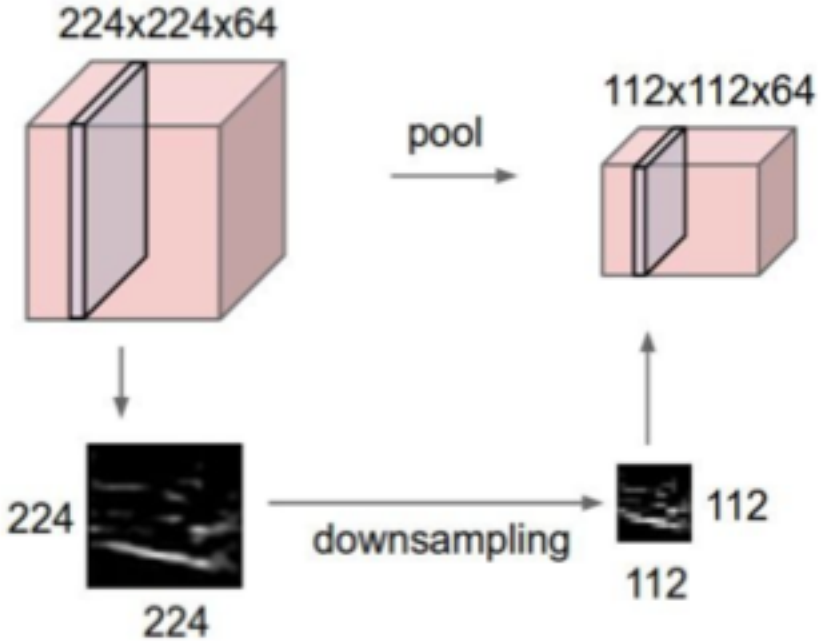


two more layers to go: POOL/FC



Pooling Layer

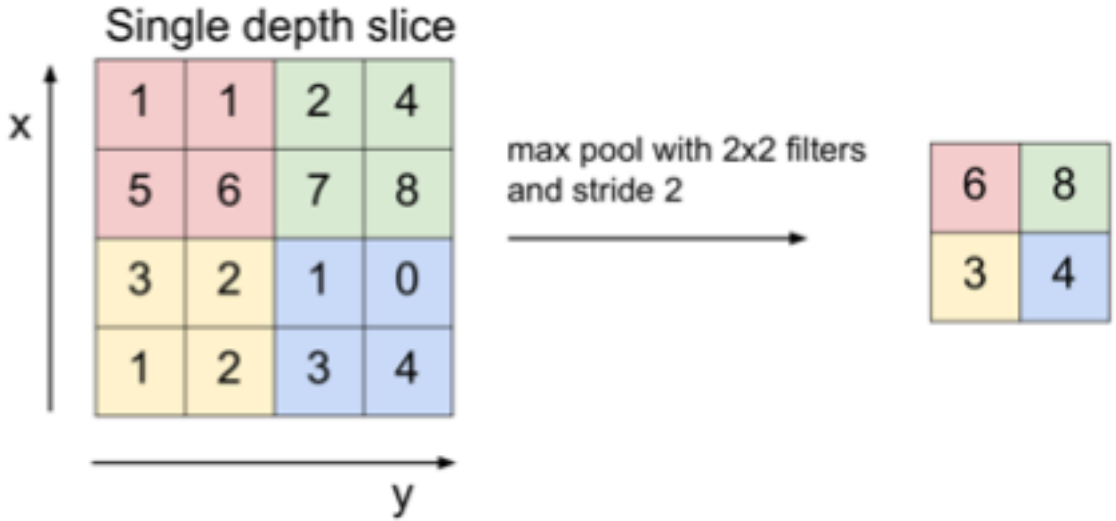
- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer

모두의연구소

- 세로·가로 방향의 공간을 줄이는 연산
  - 2x2 최대 풀링(max pooling)을 스트라이드 2로

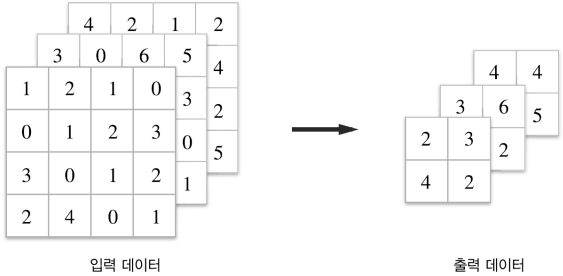


  - 평균 풀링(Average pooling)도 있습니다

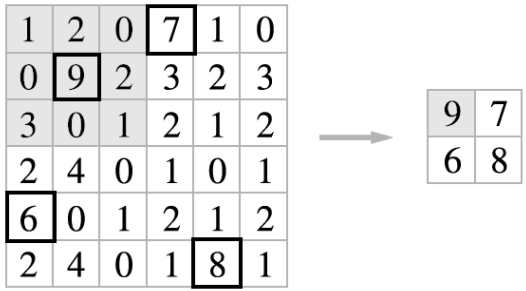
Pooling Layer

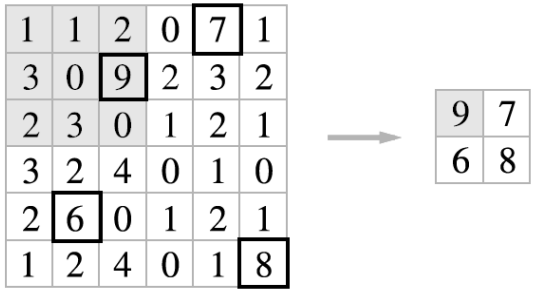
모두의연구소

- 풀링 계층의 특징
  - 학습해야 할 매개변수가 없음
  - 채널 수가 변하지 않음
  - 입력의 변화에 영향을 적게 받음 (강건하다)



데이터가 오른쪽으로 1칸씩 이동한 경우





Pooling 후 특징맵 크기 변화

모두의연구소

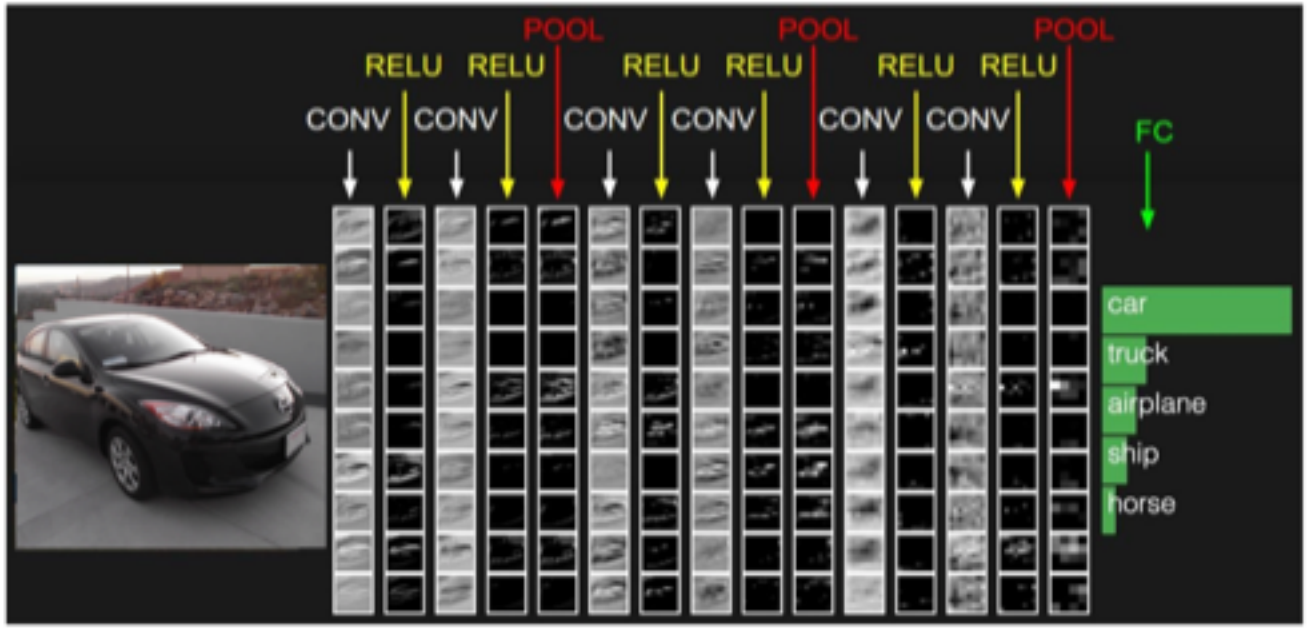
Common settings:

F = 2, S = 2
F = 3, S = 2

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

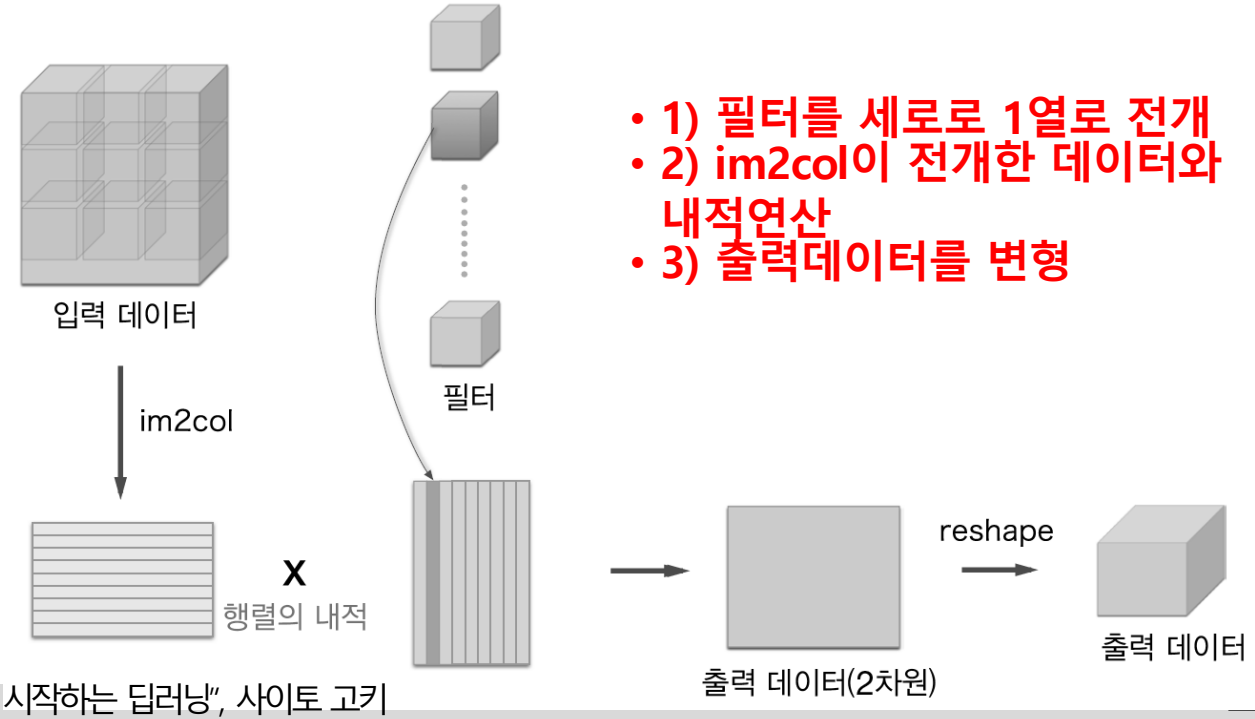
Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Convolutional Networks 구현

- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



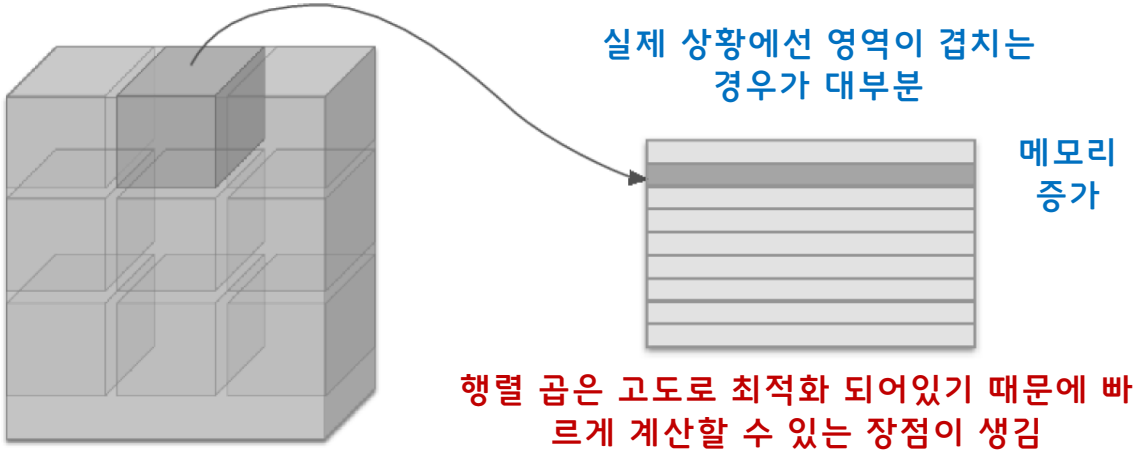
- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolutional Networks 구현

모두의연구소

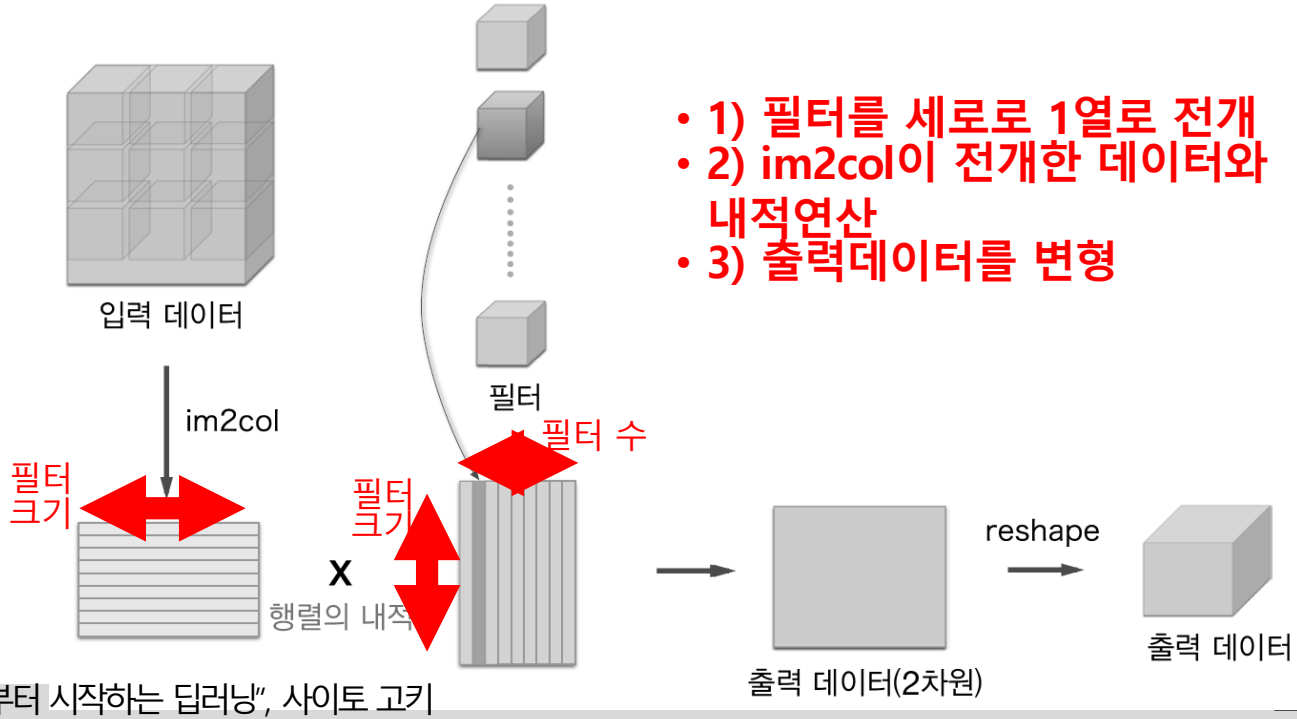
- im2col로 데이터 전개하기
  - 입력데이터에서 **필터를 적용하는 영역(3차원 블록)을 한줄로 늘어 놓습니다**
  - 이 전개를 필터를 적용하는 모든 영역에서 수행하는게 im2col입니다



필터 적용 영역을 앞에서부터 순서대로 1줄로 펼친다

Convolutional Networks 구현

- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



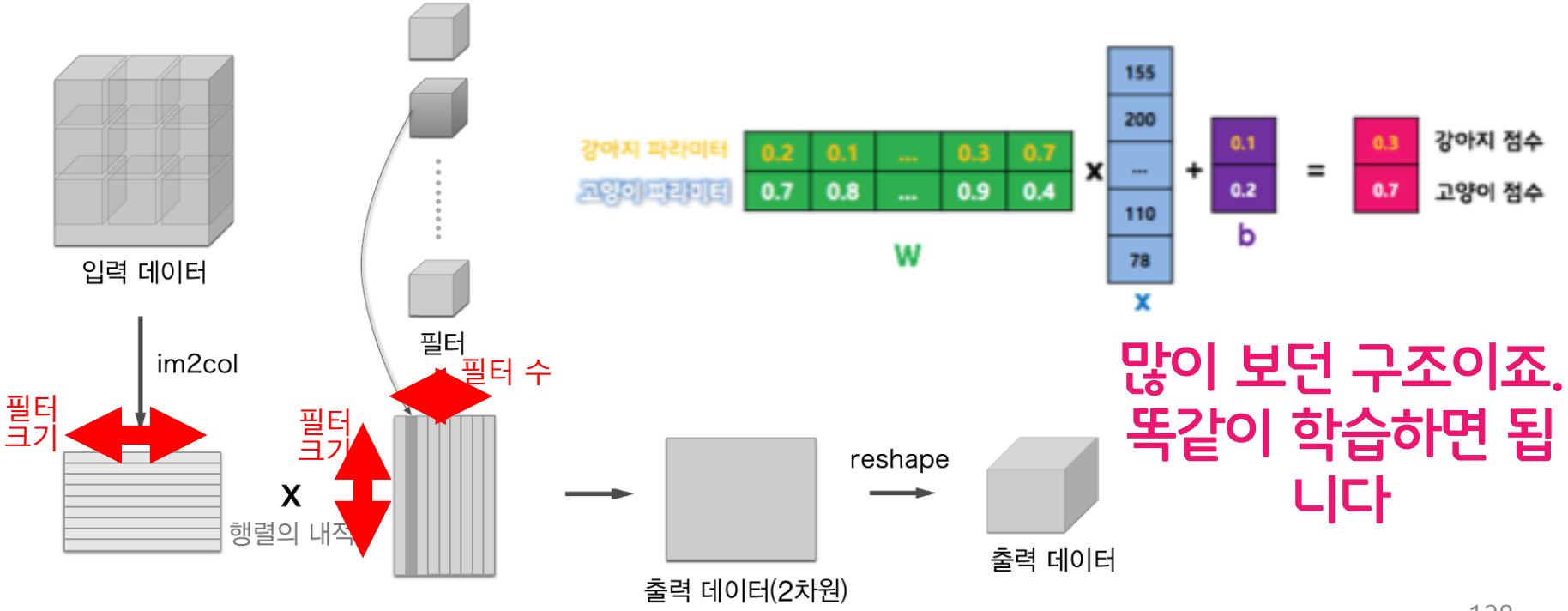
- 1) 필터를 세로로 1열로 전개
- 2) im2col이 전개한 데이터와 내적연산
- 3) 출력데이터를 변형

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

Convolution 레이어 구현하기

모두의연구소

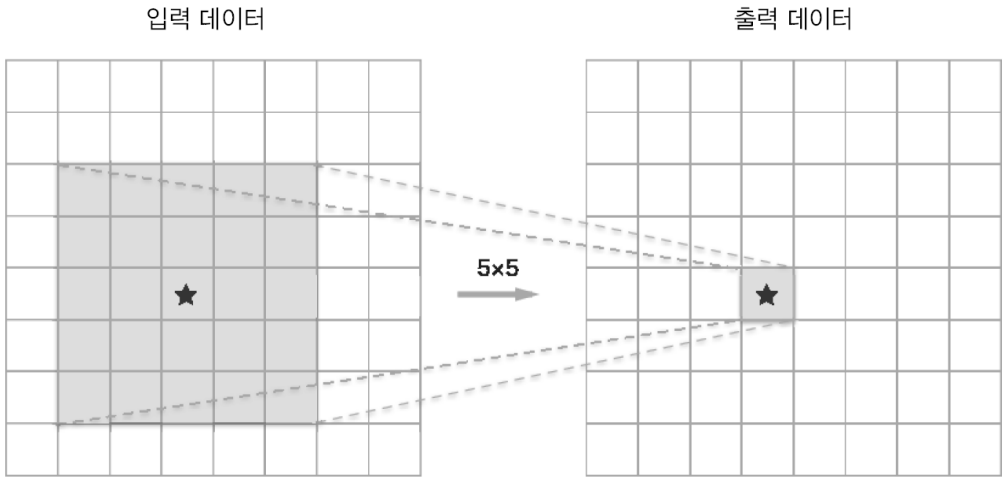
- im2col로 데이터 전개하기
  - 합성곱을 행렬 곱 연산으로 (Affine 계층연산으로)



많이 보던 구조이죠. 똑같이 학습하면 됩니다

더 작은 필터

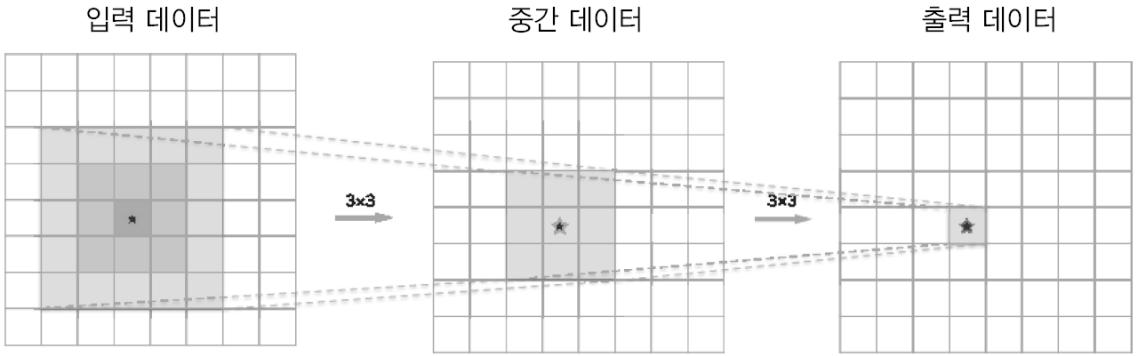
- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교



더 작은 필터

모두의연구소

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교

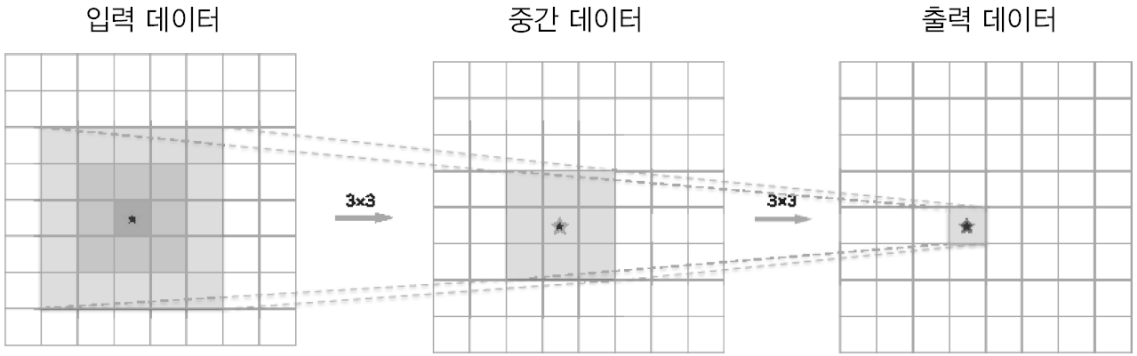


- 5x5와 같은 크기의 영역을 처리 (receptive field)
- 층이 깊어지기에 ReLU와 같은 비선형성 추가로 표현력이 개선. 비선형 함수가 겹쳐지면 더 복잡한것도 표현 가능

더 작은 필터

모두의연구소

- 작은 필터로 더 깊게
  - 5x5 한번과 3x3 두번의 비교

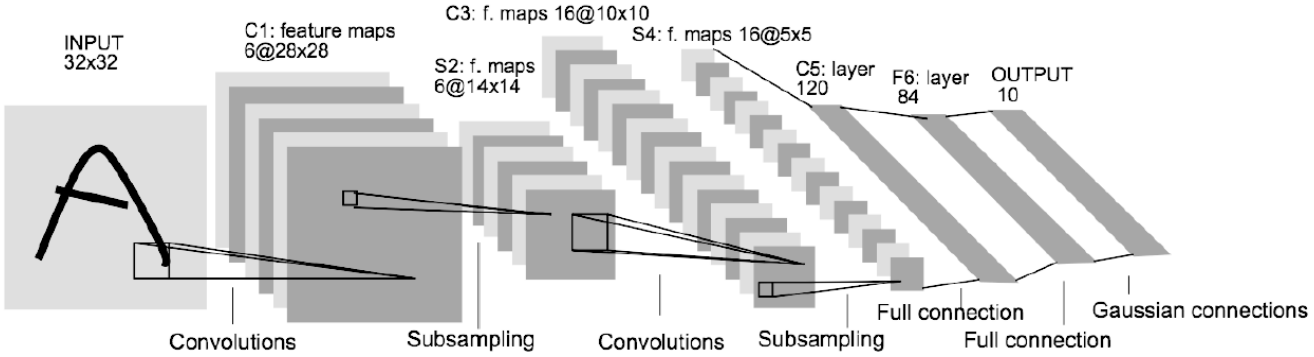


  - 매개변수 수 비교
    - 5x5 필터 1개 : 25개
    - 3x3 필터 2개 : (3x3)x2 = 18개
    - 7x7 필터 1개 : 49개
    - 3x3 필터 3개 : (3x3)x3 = 27개

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

LeNet

- LeNet (1998년)



- 현재의 CNN과의 가장 큰 차이 : 시그모이드
  - 현재는 주로 ReLU

Reference : "밑바닥부터 시작하는 딥러닝", 사이토 고키

요약

모두의연구소

- ConvNet은 CONV, POOL, FC 레이어로 구성
- 작은 필터로 더 깊은 네트워크를 구성하는게 트렌드
- FC레이어를 제거하는게 트렌드
  - 추후 영상처리 부분에서 다루게 됨
- 전형적 구조
  - [(CONV-RELU)*N-POOL]*M-(FC-RELU)*K, SOFTMAX
    - N : 보통 ~5, M : 크게, $0 \leq K \leq 2$
- 최근엔 전혀적인 구조 보다 다양한 CNN 모델 변형들이 많아짐

박 은 수 Research Director

E-mail : es.park@modulabs.co.kr