Deep Learning Seminar

2. Linear Classification

딥러닝은 Linear 모델이 여러개 쌓여진 것

HA SEUNG HYUN

Contents

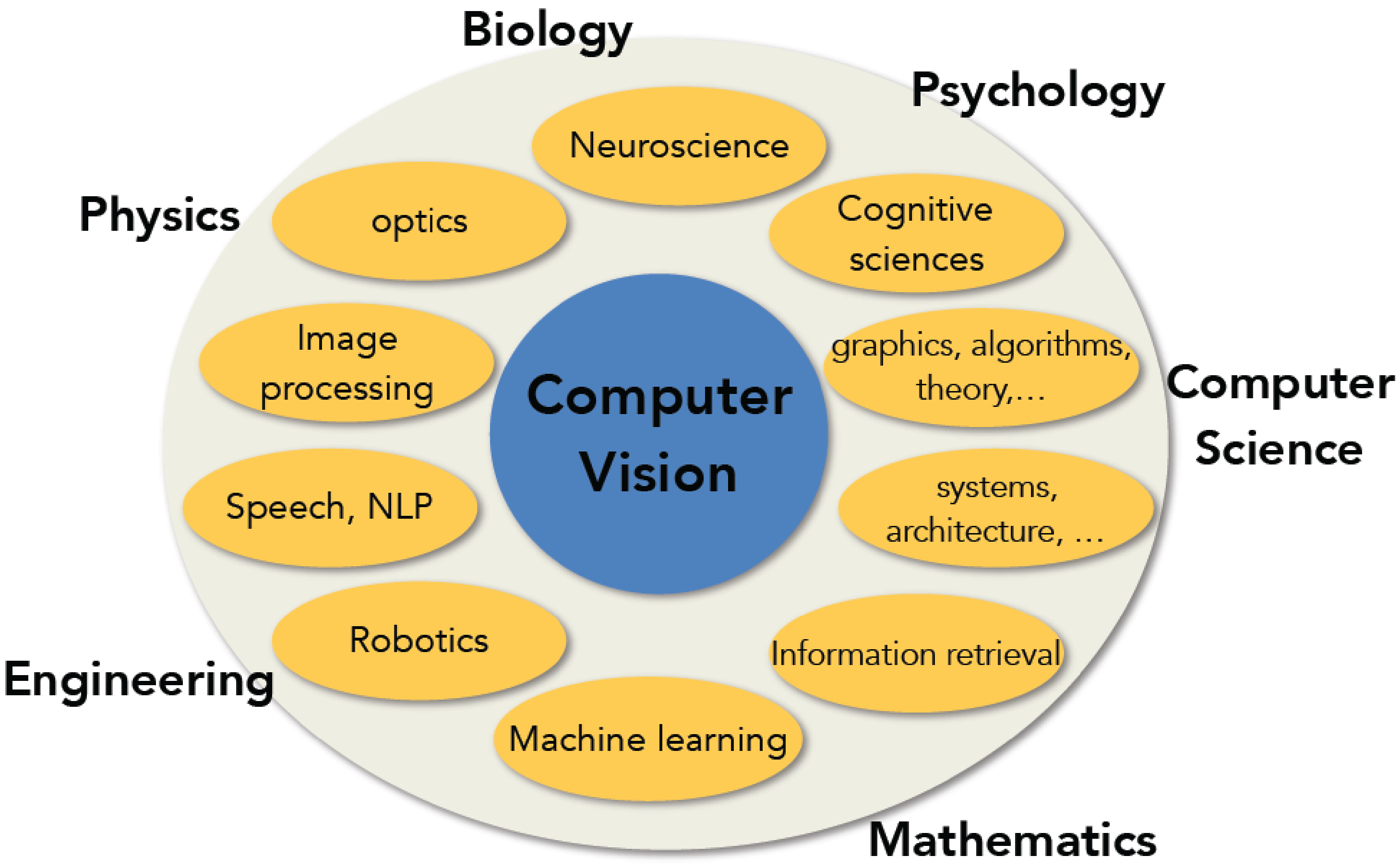
linear regression - > NN

Reference: lecture note (Fei-Fei Li)
lecture note (Andrew Ng)
모두를 위한 머신러닝 (Sung kim)

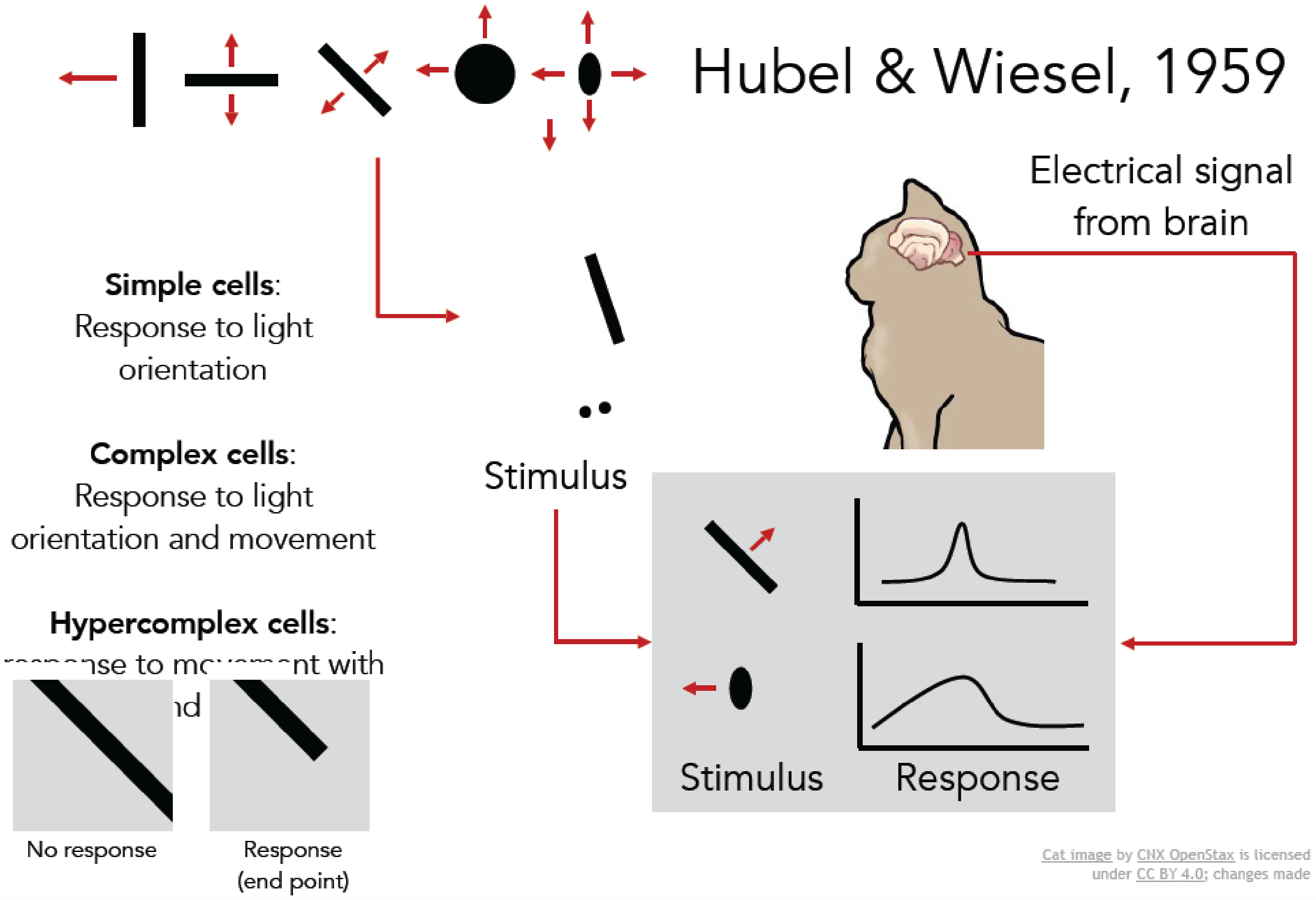
1. Computer Vision

Computer Vision

기계의 시각에 해당하는 부분을 연구하는 컴퓨터 과학
인산이 시각적으로 할수 있는 일을 기계가 그대로 하는것이 목표



Hubel & Wiesel, 1959

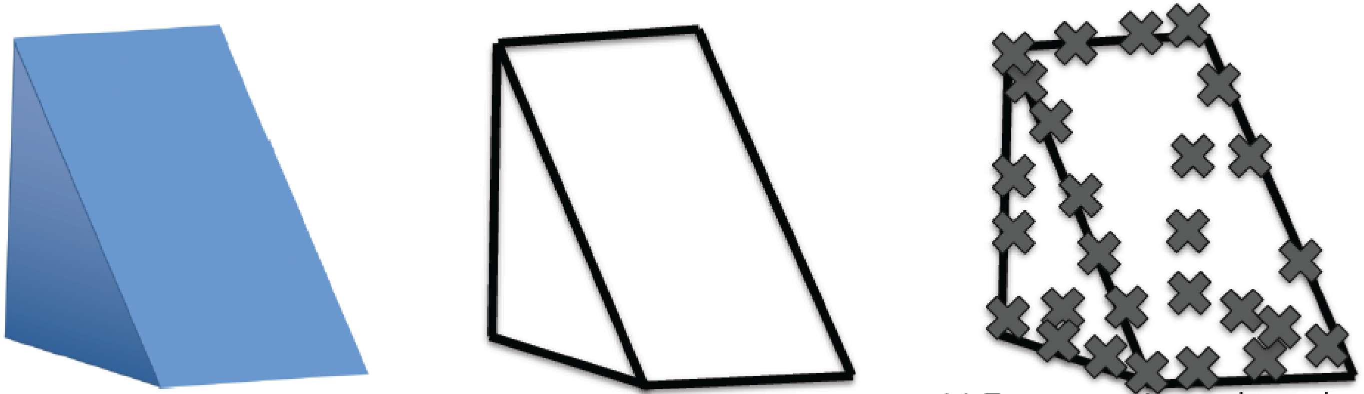


뇌가 가장 큰 반응을 보
인곳?
모서리

Edge

모서리가 중요★

경계선
- 밝기가 낮 →높
높→낮
(명암→경계→방향성 모양)



(a) Original picture (b) Differentiated picture (c) Feature points selected

이러한 명암의 변화
→ 기울기
→ 미분

영상에서 밝기의 기울기
→ gradient

Edge Detection
→ gradient algorithm

Edge

방향에 따라 바라보는 시각이 다름
→ 한계점 명확
→ 상황에 따른 상당히 다른 정보
→ CNN으로 한계 극복

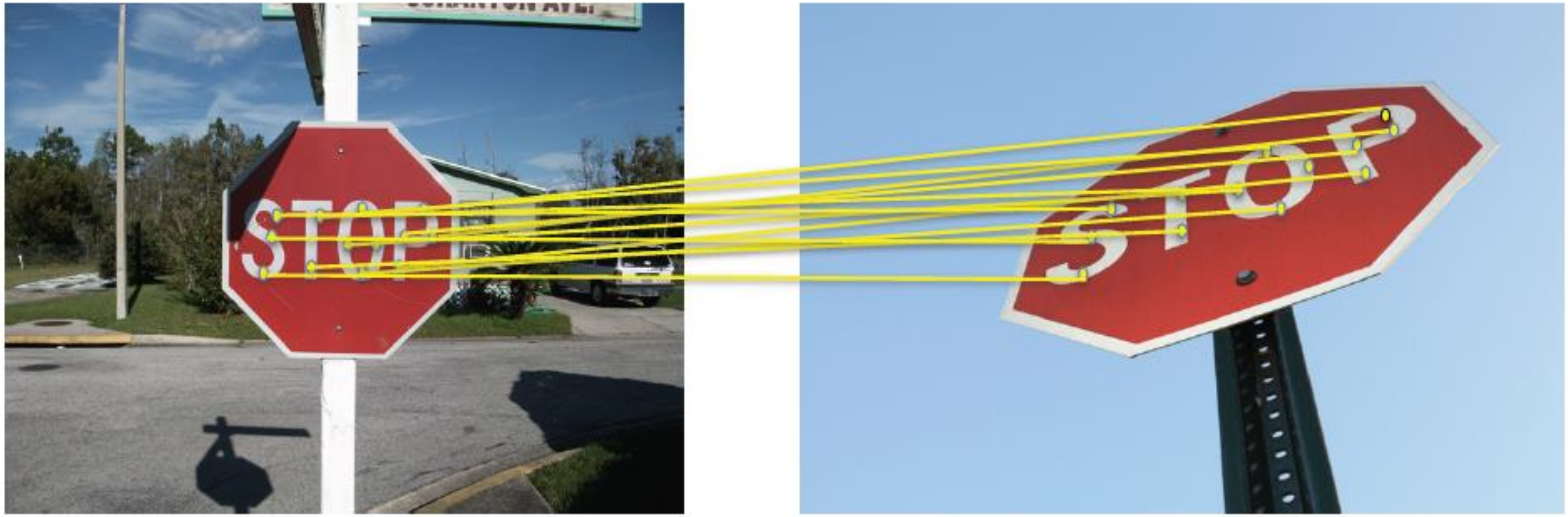


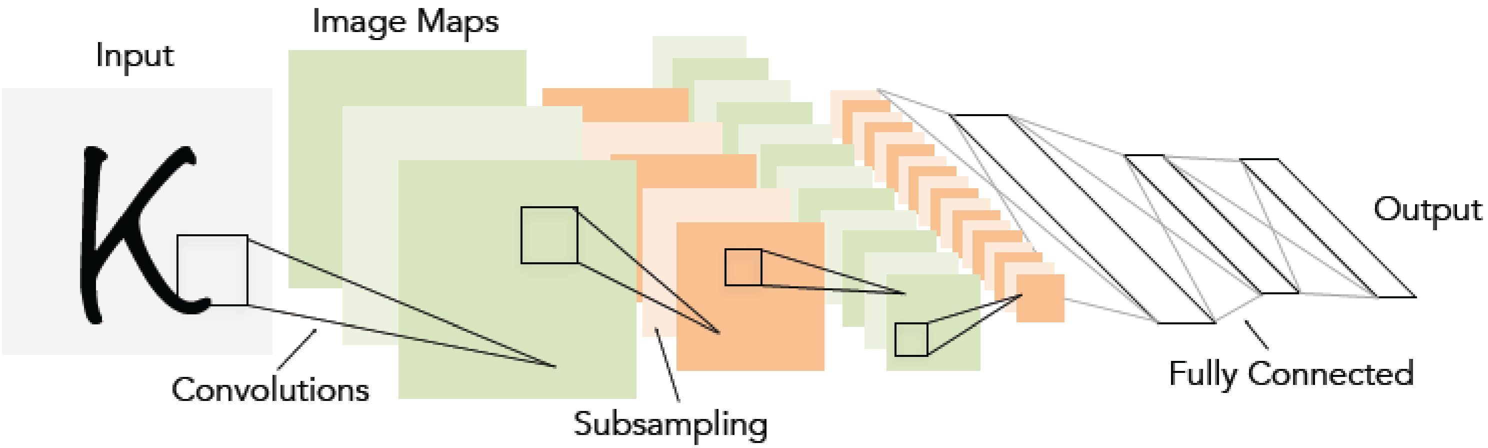
Image is public domain

Image is CC BY-SA 2.0

Convolutional Neural Network

얀르쿤
제프리 힌튼
죠슈아 벤지오

1998
LeCun et al.



# of transistors

10^6

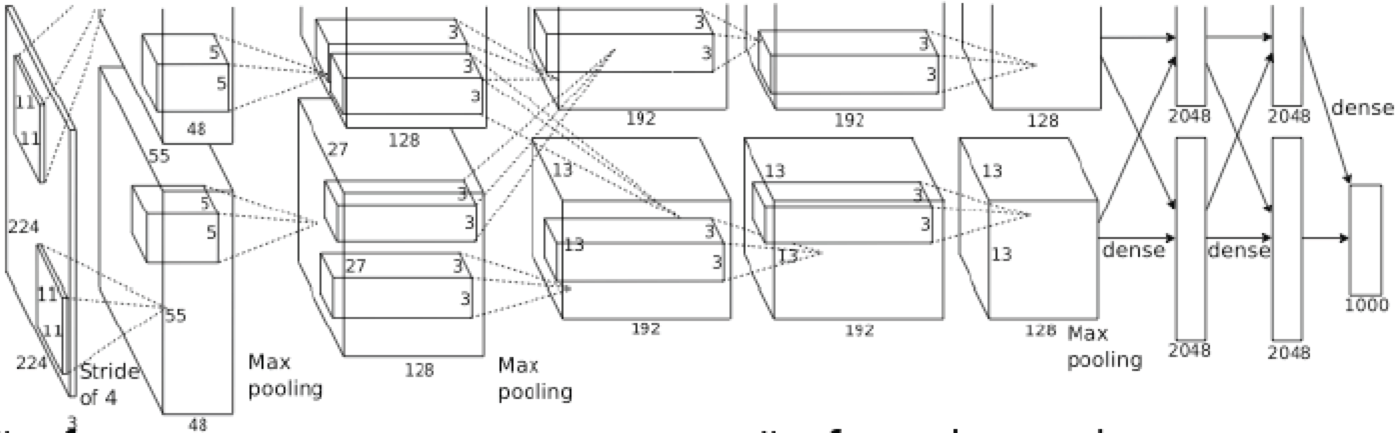
# of pixels used in training

10^7 NIST

Convolutional Neural Network

2012

Krizhevsky et al.



# of transistors

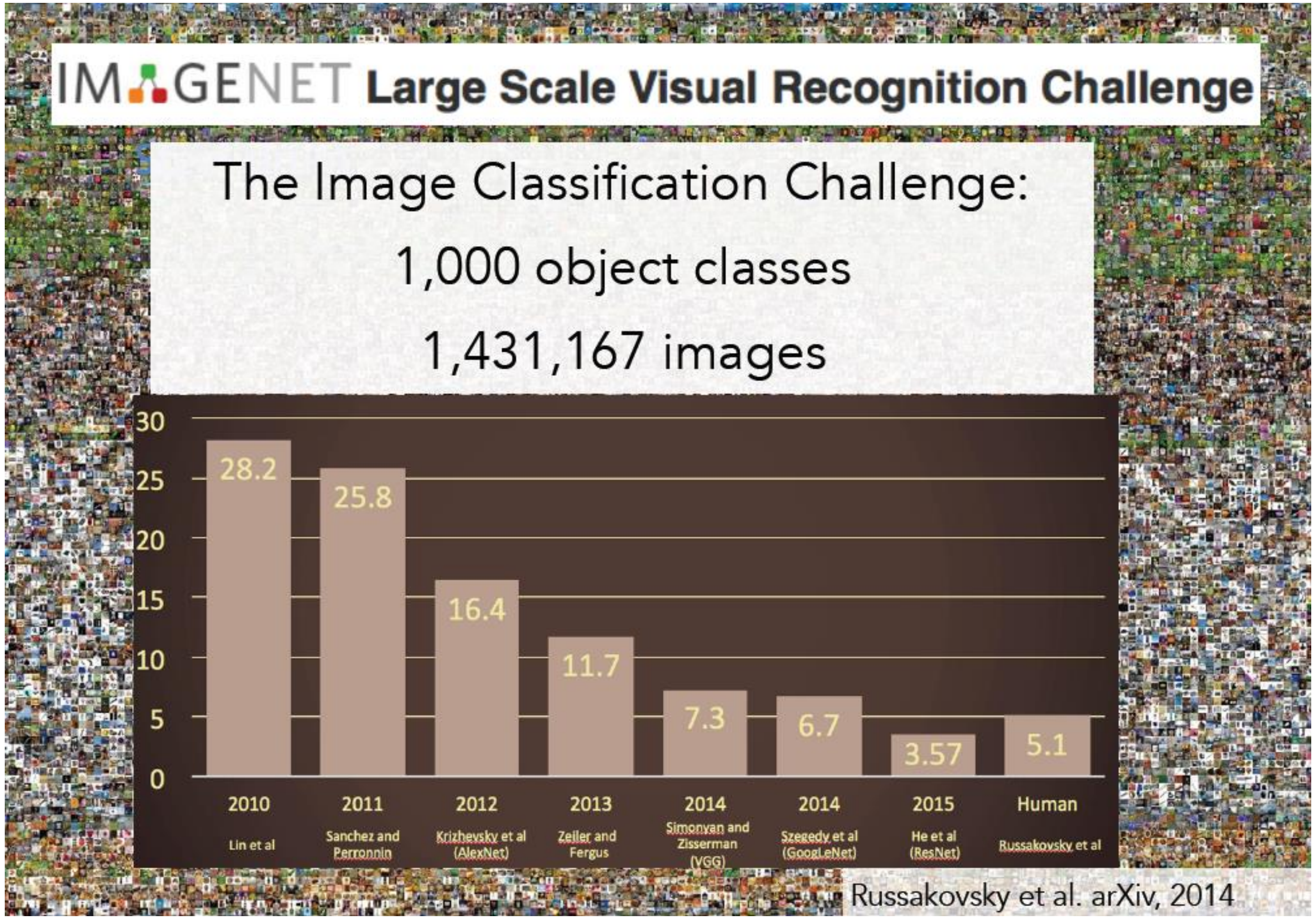
10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

IMAGENET


IMAGENET Large Scale Visual Recognition Challenge
The Image Classification Challenge:
1,000 object classes
1,431,167 images
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.57
5.1
2010
Lin et al
2011
Sanchez and Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler and Fergus
2014
Simonyan and Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al
Russakovsky et al. arXiv, 2014

2. Classification

2-1) Classification

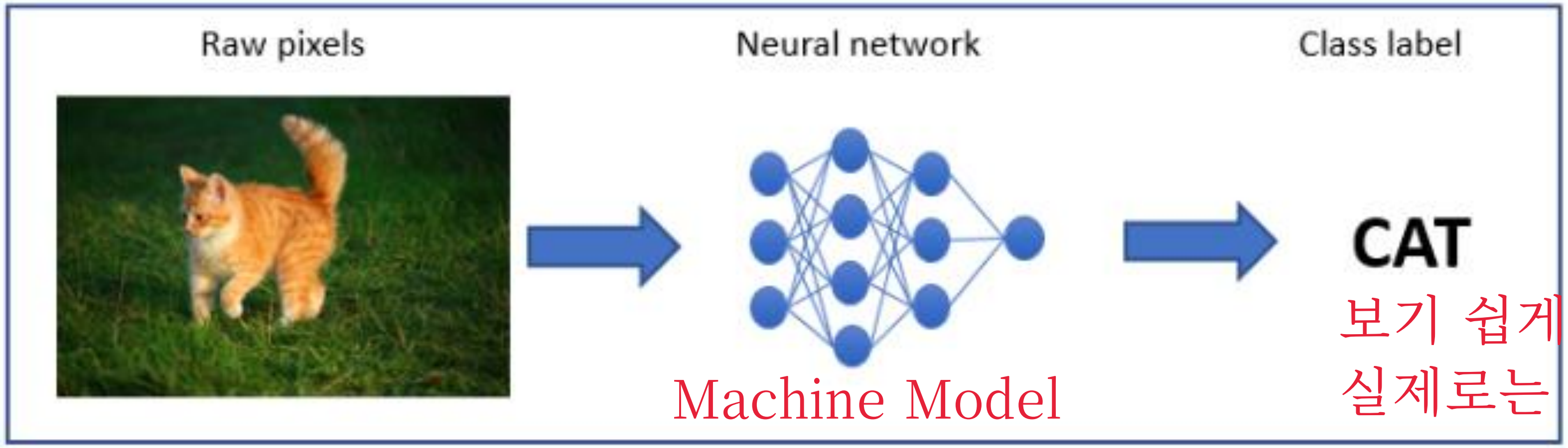
2-2) Nearest Neighbor

2-3) Cross-Validation

Classification

- Example

Linear

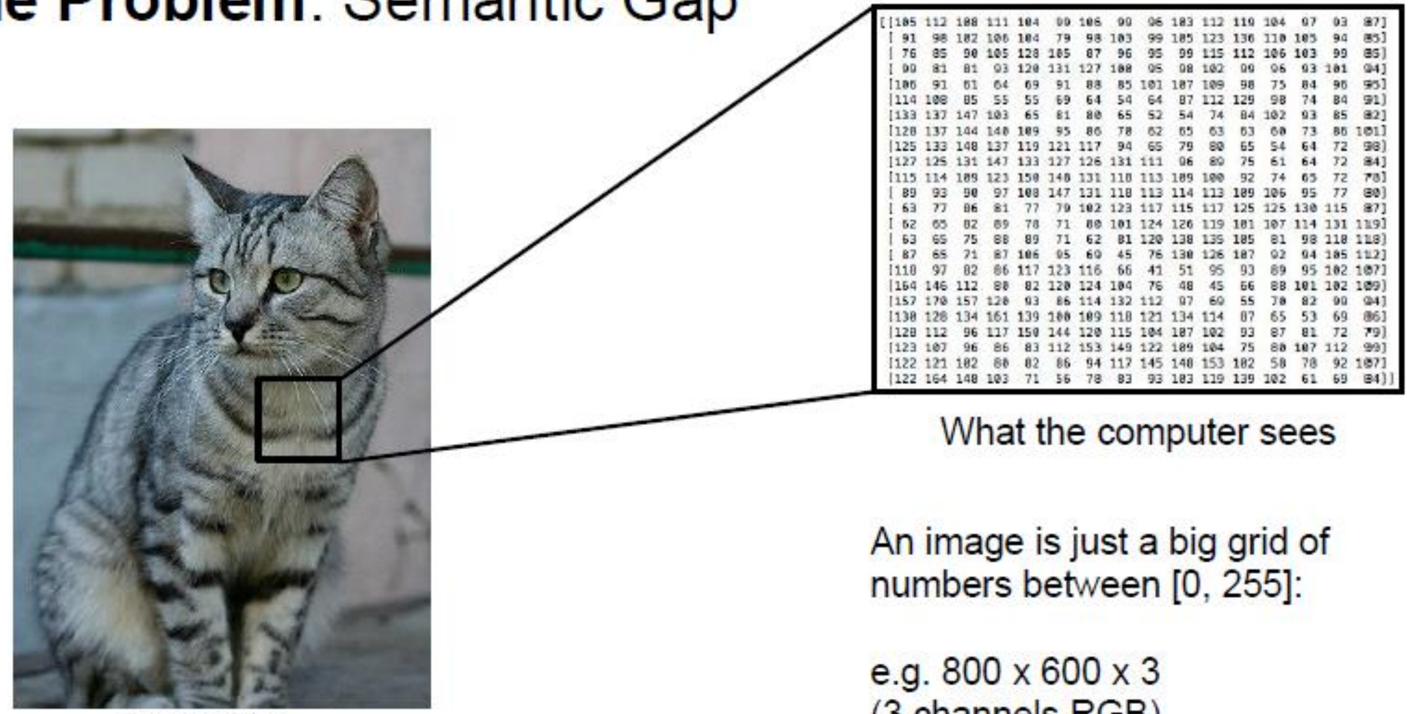


Machine Model

보기 쉽게 라벨이 딱 나온것일 뿐
실제로는 0,1,2...와 같은 분류

Classification

The Problem: Semantic Gap



What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Classification

Challenges: Illumination


This image is CC0 1.0 public domain

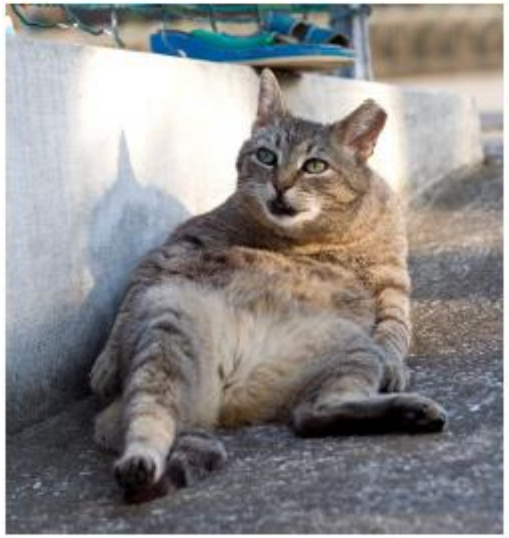

This image is CC0 1.0 public domain


This image is CC0 1.0 public domain


This image is CC0 1.0 public domain

Classification

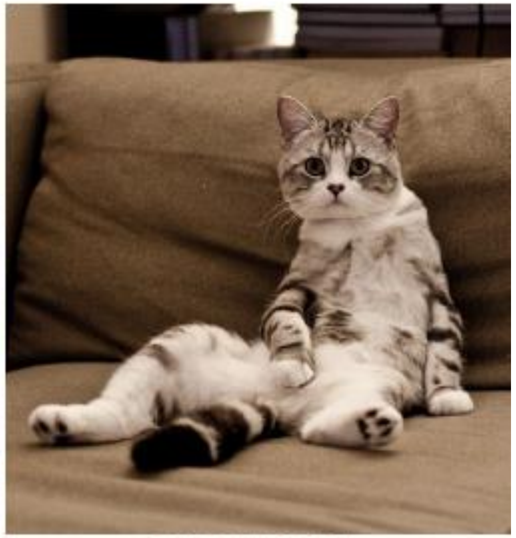
Challenges: Deformation 변형



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Classification

Challenges: Occlusion

Classification

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Classification

Challenges: Intraclass variation

Classification

분류학습 모델의 정의

An image classifier

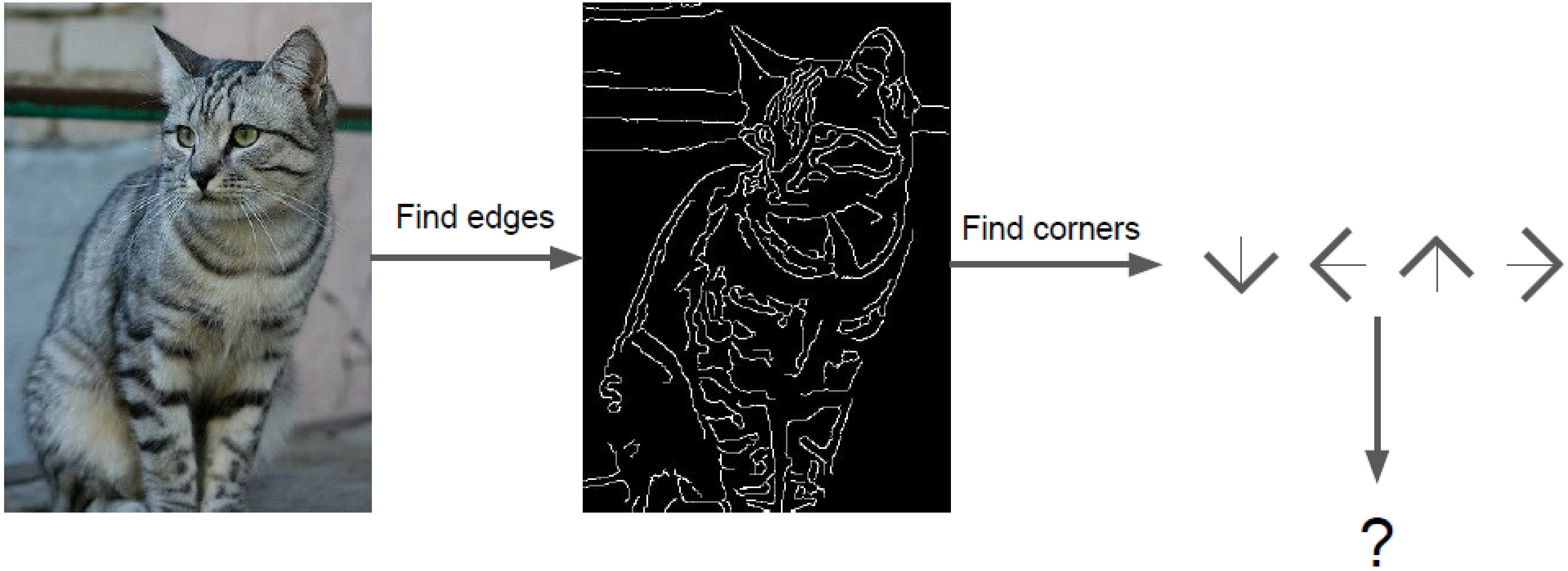
이미지(데이터)
→ 픽셀값

```
def classify_image(image):
  # Some magic here?
  return class_label
```

정체
target
class

Classification

Attempts have been made



Classification

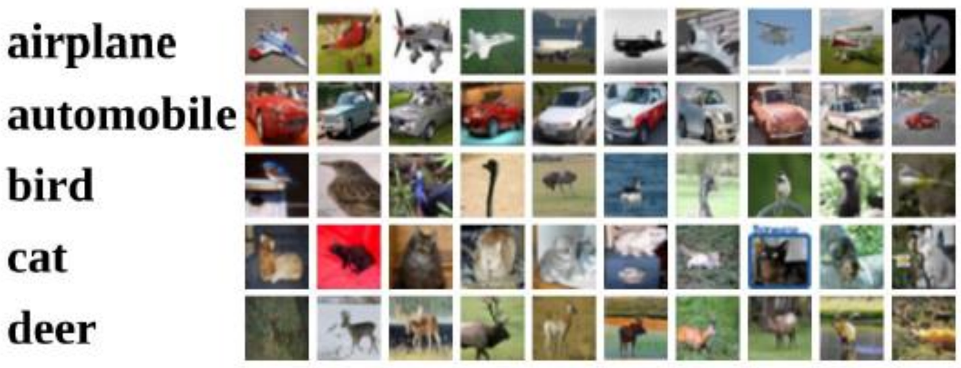
Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set



Nearest Neighbor

```
def train(images, labels):
  # Machine learning!
  return model
```

모든 특징 암기

Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Predict the label of the $\boxed{\text{most similar}}$ training image

Nearest Neighbor

- How to quantify similarity?

1) L1 Loss

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

$$\left| \overset{\text{test image}}{\begin{bmatrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{bmatrix}} - \overset{\text{training image}}{\begin{bmatrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{bmatrix}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{bmatrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{bmatrix}} \xrightarrow{\text{add}} 456$$

2) L2 Loss

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

MSE 와 비슷

Nearest Neighbor

유사도가 완벽하지 않게 나온다

k-Nearest Neighbor on images **never used.**

- Very slow at test time
- Distance metrics on pixels are not informative

Original Boxed Shifted Tinted

Cross-Validation

- Hyperparameters

Choices about the algorithm that we set rather than learn

(≈ Heuristic Values)

ex) k-fold cross validation, learning rate, epoch, and number of layers

Cross-Validation

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Cross-Validation

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Cross-Validation 참고 블로그

https://m.blog.naver.com/ckdgus1433/221599517834

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Cross-Validation

Setting Hyperparameters

Your Dataset

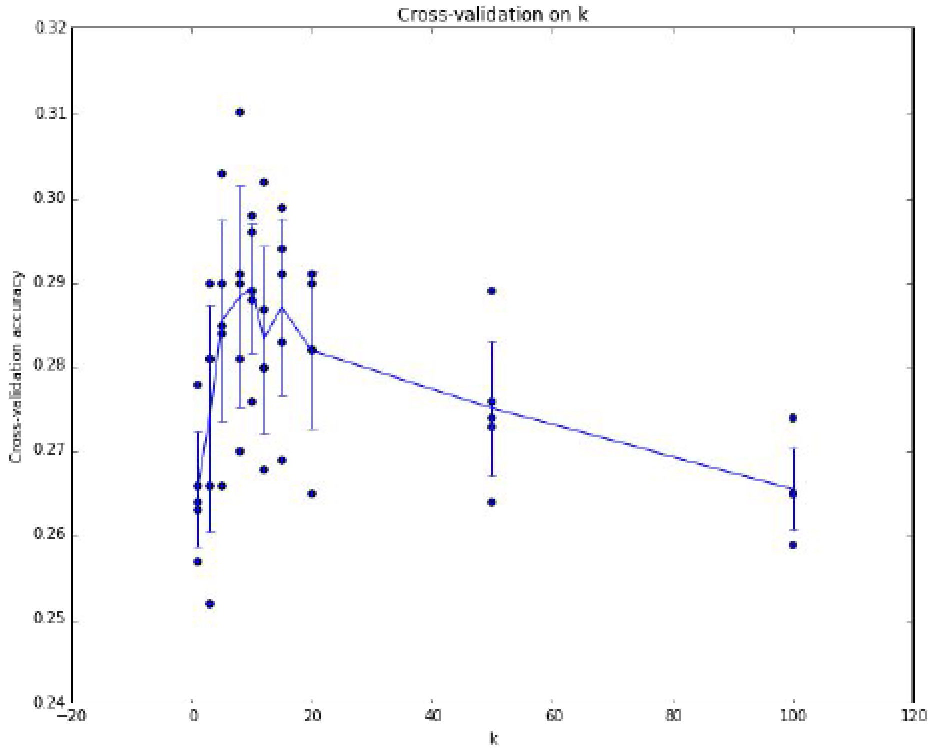
Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Cross-Validation

Setting Hyperparameters



Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

Classification

- Deep Learning Pipeline

1. Training Data Loading

2. Training Data Augmentation

linear 연산으로 이뤄짐

3. Deep Neural Network Training with Training Data

Validation

4. Deep Neural Network Testing with Testing Data

5. Inference with verified Deep Neural Network

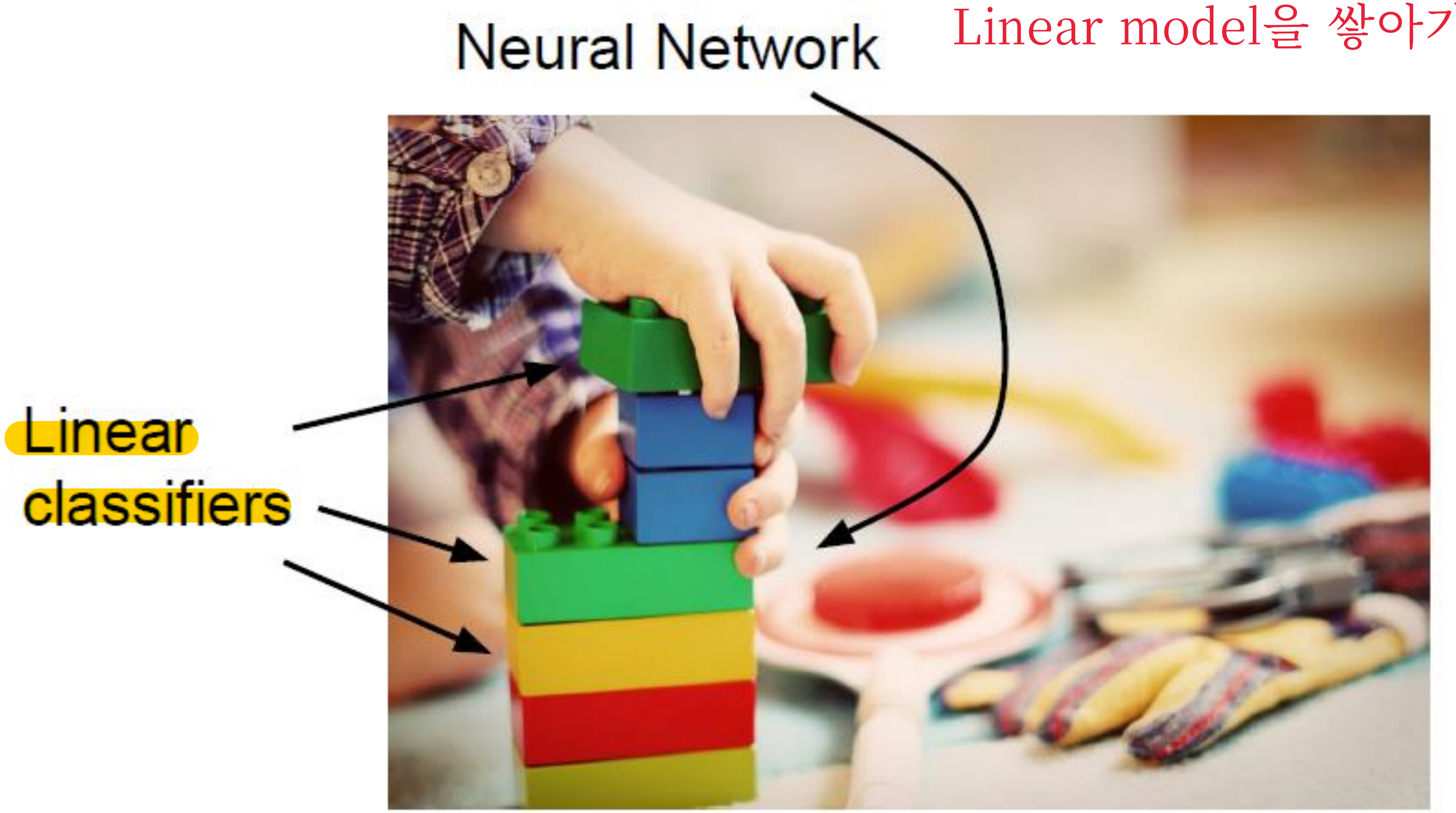
출시 전 최종 테스트 데이터

3. Linear Classification

Linear Classification

ANN - linear layer 1개
DNN - linear layer 여러 개
CNN - DNN + convolution 연산

Linear model을 쌓아가는 과정!!

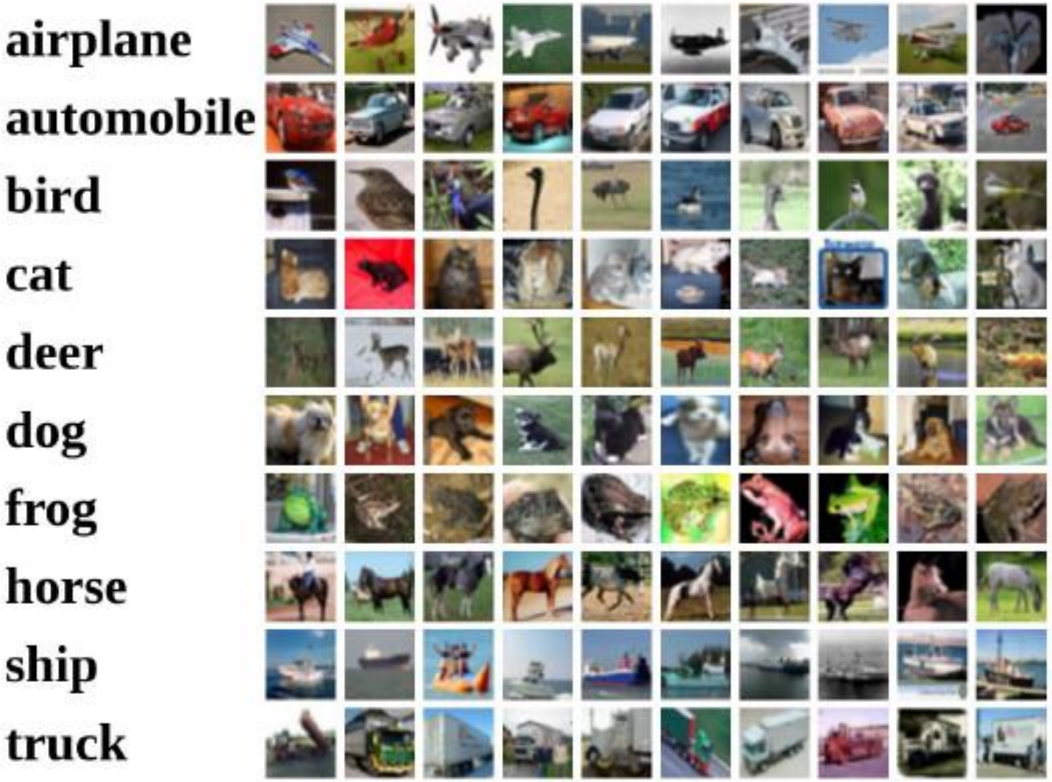


This image is CC0 1.0 public domain

Linear Classification

1. MNIST (0~9)
   흑백 28*28
2, CIFAR10 (10개 카테고리)
   칼라 32*32 ... 칼라 학습 더 어려움
3. CIFAR100

Recall CIFAR10



50,000 training images
each image is **32x32x3**

10,000 test images.

위의 데이터 acc 업데이트 트랜드 확인
→ 기술 트랜드 확인
https://paperswithcode.com/

Linear Classification

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x}, \mathbf{W})$

10 numbers giving
class scores

W

parameters
or weights

Linear Classification

$$f(x,W) = Wx$$

Image

Array of **32x32x3** numbers
(3072 numbers total)

$f(\mathbf{x},\mathbf{W})$

W

parameters
or weights

10 numbers giving
class scores

Linear Classification

입력이 픽셀값으로 들어온다는 것은
32*32의 값이 1차원으로 들어온 다는 것
그래서 아래 공식의 x가 3072*1 와 같은 1차원으로 입력 됨

3072x1

$$\boxed{f(x,W)} = \boxed{W}\boxed{x}$$

10x1 10x3072

Image

Array of **32x32x3** numbers
(3072 numbers total)

$$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$$

10 numbers giving
class scores

$$\uparrow$$

W

parameters
or weights

Linear Classification

$$\underbrace{\boxed{f(x,W)}}_{10\times 1} = \underbrace{\boxed{W}}_{10\times 3072}\overbrace{\boxed{x}}^{3072\times 1} + \boxed{b}\ \ 10\times 1$$

행렬 덧셈은 결과 값과 같은 행렬이여야 하므로 10*1

Image

Array of **32x32x3** numbers (3072 numbers total)

$\rightarrow$ f(**x**,**W**) $\rightarrow$ **10** numbers giving class scores

$\uparrow$

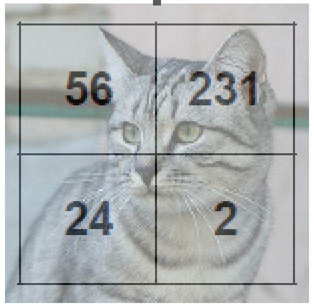
W

parameters or weights

Linear Classification

강아지로 판단(loss function)
그러므로 loss가 높게 나옴.(forward)
loss가 높으므로 미분을 통해 loss가 높게 나온 책임을 이미지 픽셀 값에 중요도를 부여한 W에 책임을 묻는다. 책임을 묻는다는 것은 W의 값들의 수정이 대대적으로 일어남 이때 W값이 컸던 것은 크게 변동 작았던 것도 크게 변동

Stretch pixels into column



Input image

4*1

f(x,W)

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$

3*1

가중치=중요도

- 각 픽셀에 대한 중요도를 머신이 알아서 정함.

W값을 지속적으로 업데이트하면서 loss를 줄임(학습의 주체)

모델이 얼마나 예측을 잘했는지를
정량화.
예측을 잘 할수록 0에 가까워진다.

4. Classification Loss Function

4-1) Hinge Loss

4-2) Softmax Loss

4-3) Regularization

Nomalization
0~1사이의 값이 아닌 8000, 80, 8, 800와 같은 값으로 나오는것을 0~1사이로 맞추는 것

Loss Function

Hinge loss의 문제점
B 모델처럼 학습이 잘하든
못하든 애매하게 학습이 됨.
(exponential한 값을 못 나타냄.)
- softmax fuction으로 보완

	A	B
cat	9	3.4
car	- 9	3.2
frog	- 9	3.2

A 학습이 잘됨.
exponential
잘한 부분 - > 과학 칭찬
못한 부분 - > 과한 야단

cat	$\boxed{\textbf{3.2}}$	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

33%

Loss Function

- **Hinge Function**

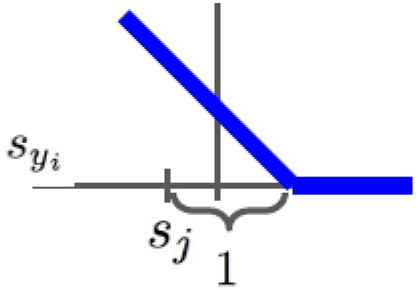
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

*** Hinge Loss (Multiclass SVM Loss)**

$$L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$$

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



Loss Function

- **Hinge Function**

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Loss Function

- **Hinge Function**

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

$$\boxed{L_i = \sum_{j\neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Loss Function

- **Hinge Function**

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Loss Function

- **Hinge Function**

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Loss Function

- **Hinge Function**

			
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Problems

Q: What happens to loss if car scores change a bit?

Q2: what is the min/max possible loss?

Q3: At initialization W is small so all s ≈ 0. What is the loss?

Loss Function

1. Normalization
2. Exponential

둘 다 충족

Hinge Function 보완

- Softmax Function

scores = unnormalized log probabilities of the classes.

$$P(Y=k|X=x_i)=\boxed{\frac{e^{s_k}}{\sum_j e^{s_j}}} \quad \text{where} \quad s=f(x_i;W)$$

Softmax function

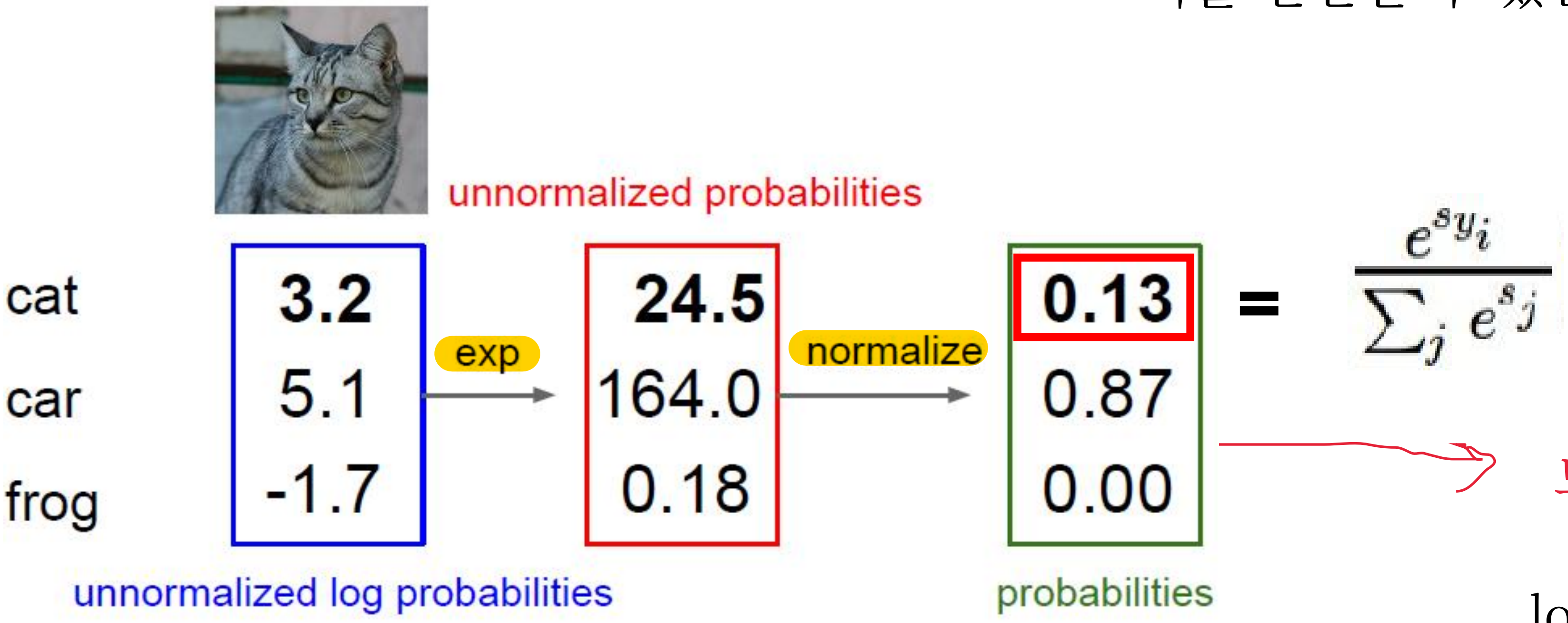
cat	**3.2**
car	5.1
frog	-1.7

exponential을 통해 값을 극대화시키고 normalization으로 확률값으로 바꿔줌.

Loss Function

- Softmax Function

이 값은 예측값이지 얼마나 예측을 잘했는지를 판단할 수 있는 값은 아니다.



모델이 예측한 예측값

loss는 시작도 안했고
loss값도 아님!!

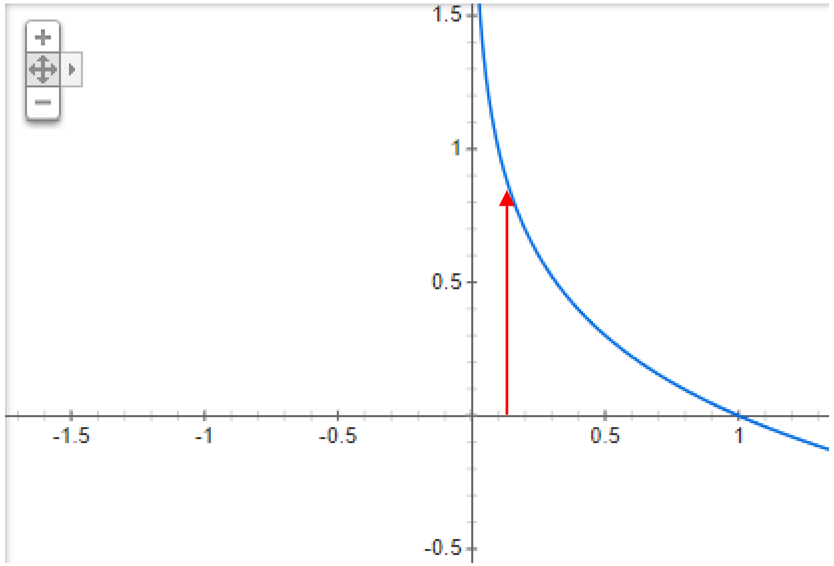
Loss Function

- **Softmax Function**

로그 함수에 위에서의 결과값들을 넣는다.

로그함수에서의 결과값이 0이랑 가까울 수록 작업을 잘함.

-log(x) 그래프



해당 값들을 로그에
넣으면 loss값이 나옴

$\boxed{0.13}$
0.87
0.00

probabilities

$$= \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Loss Function

Full loss Function

1. softmax Function (exponential + normalization)
2. - log(예측값)
3. Regularization

return 예측값

- **Regularization**

$$\underbrace{L(W)}_{\text{모델에 대한 Loss}} = \frac{1}{N}\sum_{i=1}^{N} L_i(f(\underbrace{x_i, W}_{\text{예측값}}), \underbrace{y_i}_{\text{라벨}})$$

Loss, MSE랑 비슷

loss(이미지 각각에 대한) 실제 값과의 비교 (예측값 - 정답)
만약 값이 30이 나오고 이를 정량화 한 것이 loss 값. 정답과 비교해서 loss가 나옴. 데이터가 5만개이면 loss도 5만개나옴.

x1 - > 1.jpg - > 0.2
x2 - > 2.jpg - > 0.1
. - > 0.7
. .
. .
x50000 - > 0.4 - > 이렇게 나온 모든 Loss값을 총 5만개 더해서 평균낸 값

L(Model(W1,W2,W3...W50000, b1,b2,b3....b5000)) - > L(W) : W,b 가 model에 책임이 있음.

이 모든 과정이 Forward Propagation

Loss Function

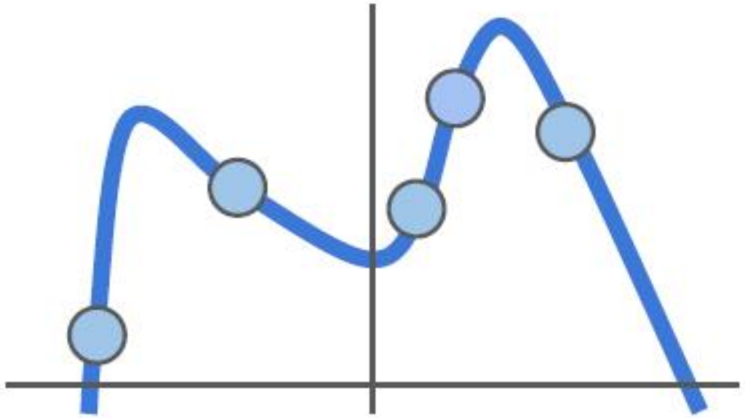
model을 쉽게 만드는 방법은 cost(loss)값이 0에 가깝도록 학습을 진행!
그러나...랜덤하게 주어지는 특정 W(가중치)가 크게 잡히는 경향이 있다.
- > overfitting ↑ (일반화 ↓)
특정 픽셀값에 대한 가중치가 너무 과도하게 잡히지 않도록 하여 overfitting 이 발생하지 않게 하는 방법이 Regularization

- Regularization

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

ex)이미지를 5개 넣었다. 학습 및 예측
- > 예측한 결과로 이미지 각각에 대한 loss추출
- > 모든 loss의 평균
- > 모델에 대한 Loss : L(W)

Data loss: Model predictions should match training data

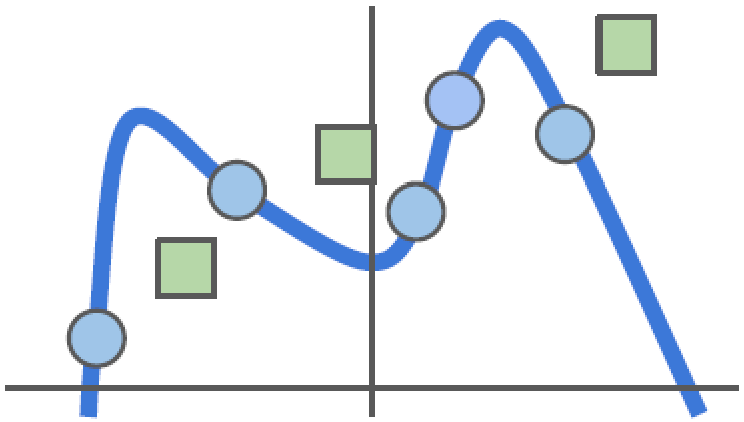


Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

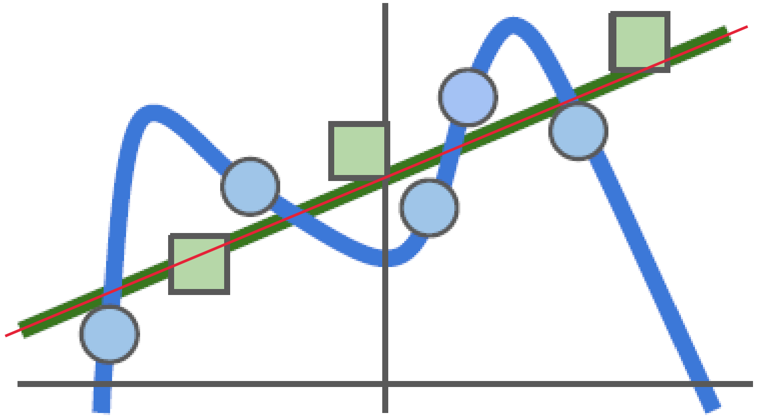


OverFitting
 - > Weight 값이 Variance한것을 막아야 한다.
고르고 크기는 작게

Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\textbf{Data loss}\text{: Model predictions should match training data}}$$

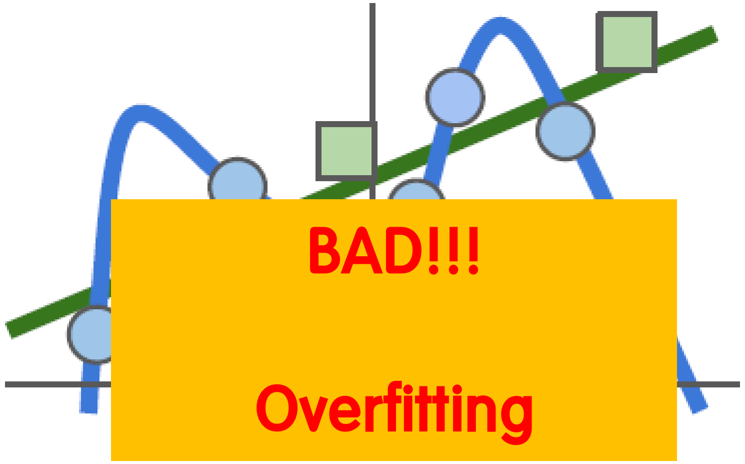


Regularization한 결과

Loss Function

- **Regularization**

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{\textbf{Data loss}: Model predictions should match training data}}$$



Loss Function

- **Regularization**

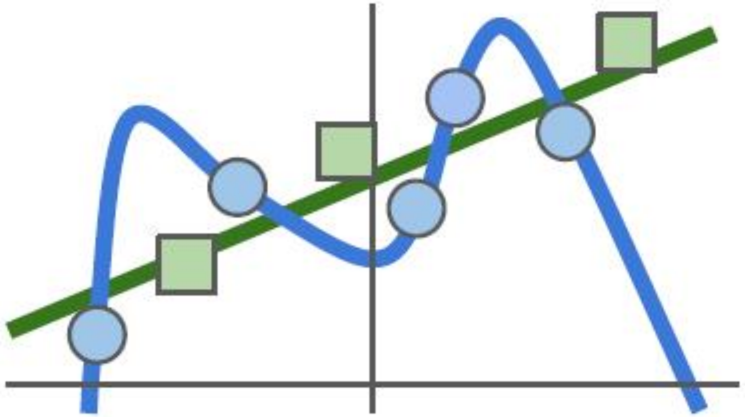
L(w): 이것으로 끝내면 overfitting 발생

$$L(W) = \underbrace{\boxed{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}}_{\textbf{Data loss}\text{: Model predictions should match training data}} \boxed{+ \lambda R(W)}$$

$$\underbrace{\boxed{+ \lambda R(W)}}_{\textbf{Regularization}\text{: Model should be "simple", so it works on test data}}$$

Data loss: Model predictions should match training data

Regularization: Model should be “simple”, so it works on test data



Loss Function

- **Regularization**

Hyperparameter (Constant)

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda \boxed{R(W)}$$

란다(상수값) 무시해도 됨

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

해당 작업으로 weight valiance
의 숨을 죽인다.

Loss Function

A, B 모델이 있다고 하자. 각각 w1,w2의 가중치를 곱한다.

A = 1*1 + 1*0 + 1*0 + 1*0 = 1
B = 1*0.25+1*0.25+1*0.25+1*0.25 = 1

- Regularization

두 모델의 예측값은 똑같다. 그런데 w1은 regularization을 안해서 w의 valiance가 크고 w2는 regularization을 하여 valiance가 적다. 결론은 softmax를 통해 나온 값은 동일하다. 그런데 regularization을 구해보면 R(w1) = 1, R(w2) = 1/4가 나온다. 즉, 이 결과를 통해 w2는 valiance는 낮춘 결과가 나타나고 w1은 낮춰지지 않는다.
weight 값을 고르게 가져간다.(정규화, 일반화)

$$x = [1, 1, 1, 1]$$

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

Rgularization X $w_1 = [1, 0, 0, 0]$

Rgularization O $w_2 = [0.25, 0.25, 0.25, 0.25]$

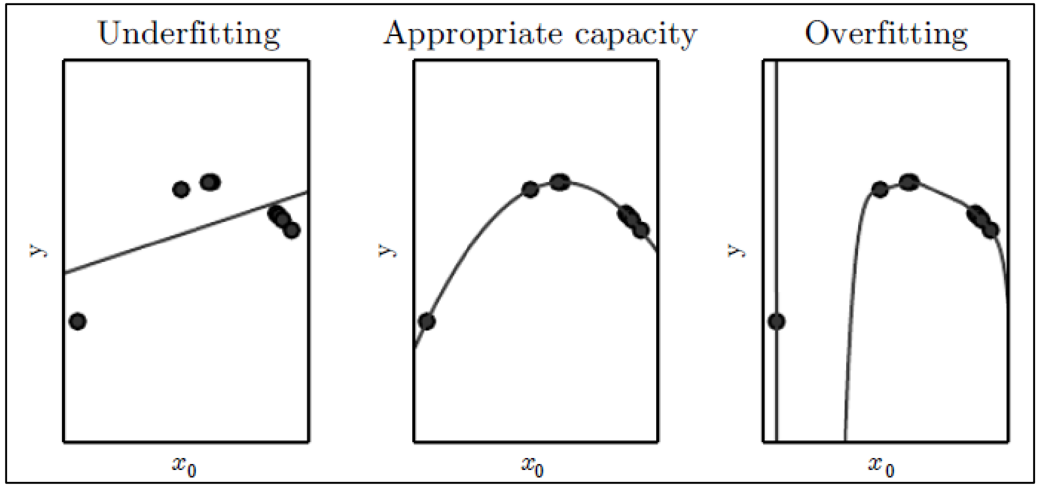
$R(w_1) = 1$ = 1^2 + 0^2 + 0^2 + 0^2

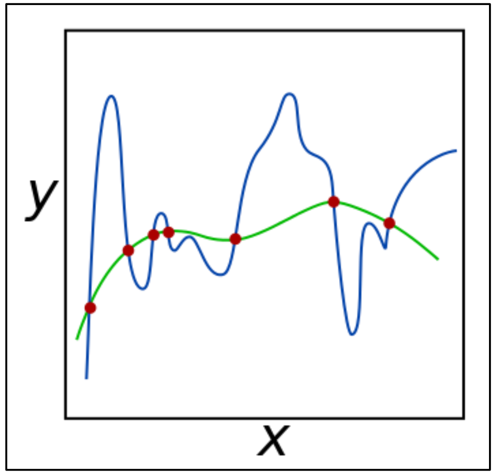
$R(w_2) = 1/4$ = (1/4)^2 + (1/4)^2 + (1/4)^2 + (1/4)^2

$$w_1^T x = w_2^T x = 1$$

Loss Function

- **Regularization**





Loss Function

- Final Loss Function

Softmax

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

L(w)

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ Full loss