Lecture 4
Machine Learning Basic

Jaeyun Kang

Aritificial Intelligence

Aritificial Intelligence

A.I. ?

"인간같이 생각하고 판단할 수 있는 컴퓨터"

그러나 컴퓨터는
기본적으로..

Aritificial Intelligence

A.I. ?

"인간같이 생각하고 판단할 수 있는 컴퓨터"

그러나 컴퓨터는
기본적으로..

인간에 비해 높은 연산능력
복잡한 계산을 빠르게 처리

Aritificial Intelligence

A.I. ?

"인간같이 생각하고 판단할 수 있는 컴퓨터"

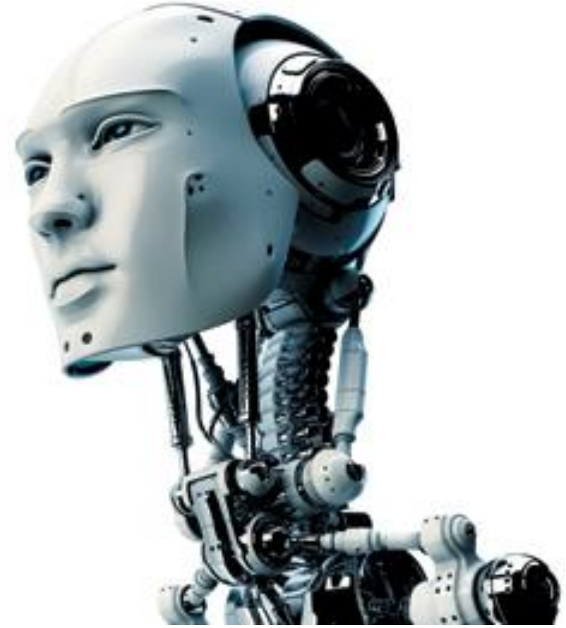
그러나 컴퓨터는
기본적으로..

인간에 비해 높은 연산능력
복잡한 계산을 빠르게 처리

인간에 비해 낮은 인지능력
개와 고양이 구분도 못한다!

Aritificial Intelligence

아직 갈 길이 먼 A.I.



이상

현실

과거의 AI: Rule-Based AI

개	고양이
귀가 쳐져있다	귀가 뾰족하다
눈매가 착하다	눈매가 매섭다
덩치가 크다	덩치가 작다

컴퓨터가 개와 고양이 사진을 구분하기 위해 필요한
'규칙' 혹은 '패턴'을 직접 제시!

과거의 AI: Rule-Based AI

과거의 AI: Rule-Based AI

눈매가 착하다? 매섭다?
덩치가 작다? 크다?
귀가 뾰족하다? 쳐져있다?

과거의 AI: Rule-Based AI

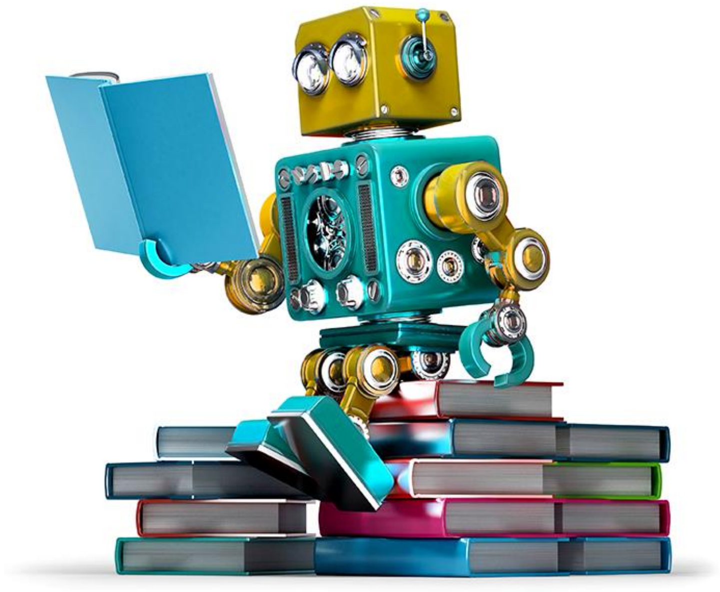
Rule-Based AI의 한계: 모든 규칙을 우리가 직접 정해줄 수 없다!

과거의 AI: Rule-Based AI

컴퓨터에게 수 많은 개와 고양이 사진을 보여주고
컴퓨터 스스로 그 규칙을 찾아나가게 하자!

Data-Based AI

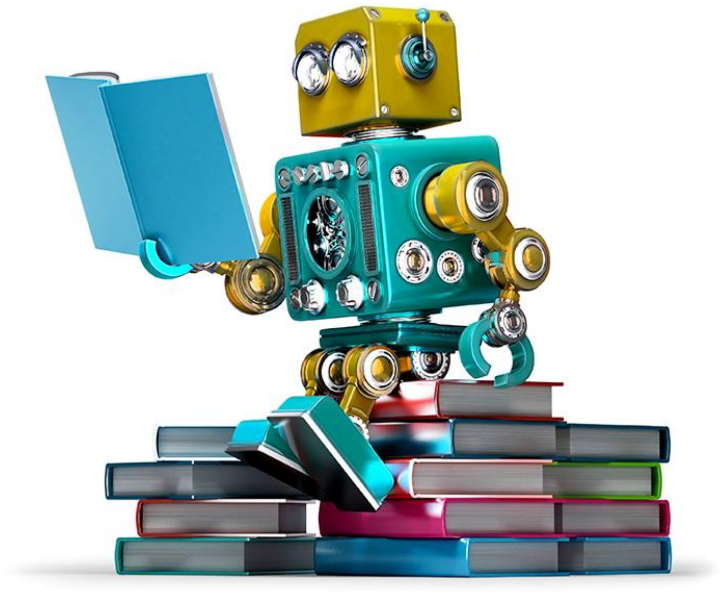
현재의 AI: Data-Based AI



'데이터'를 많이 제공하여 컴퓨터가 '스스로' 규칙을 배우도록 하자

Data-Based AI

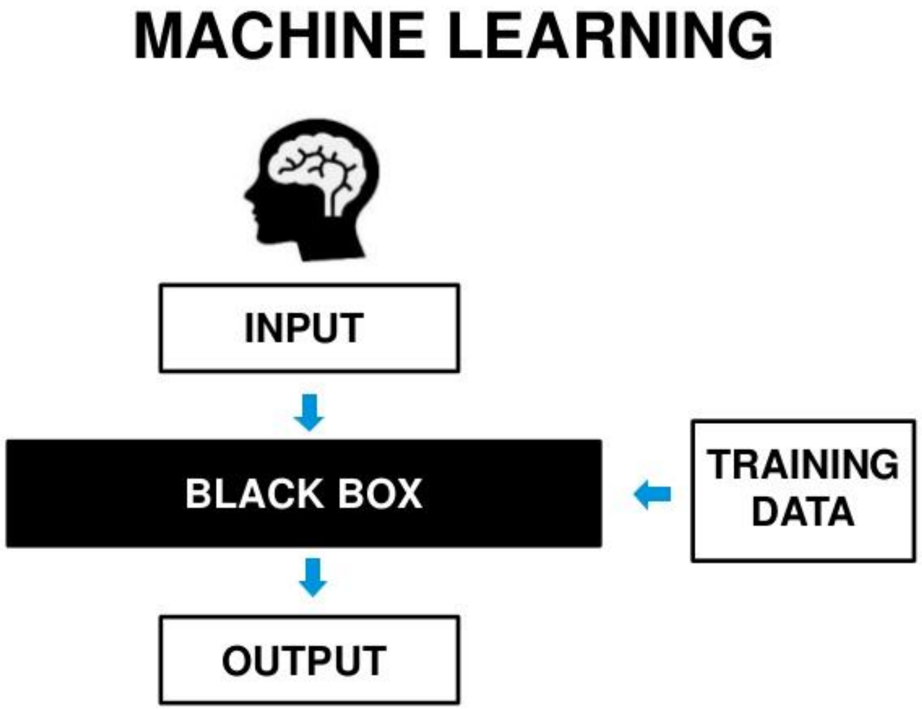
현재의 AI: Data-Based AI



'데이터'를 많이 제공하여 컴퓨터가 '스스로' 규칙을 배우도록 하자

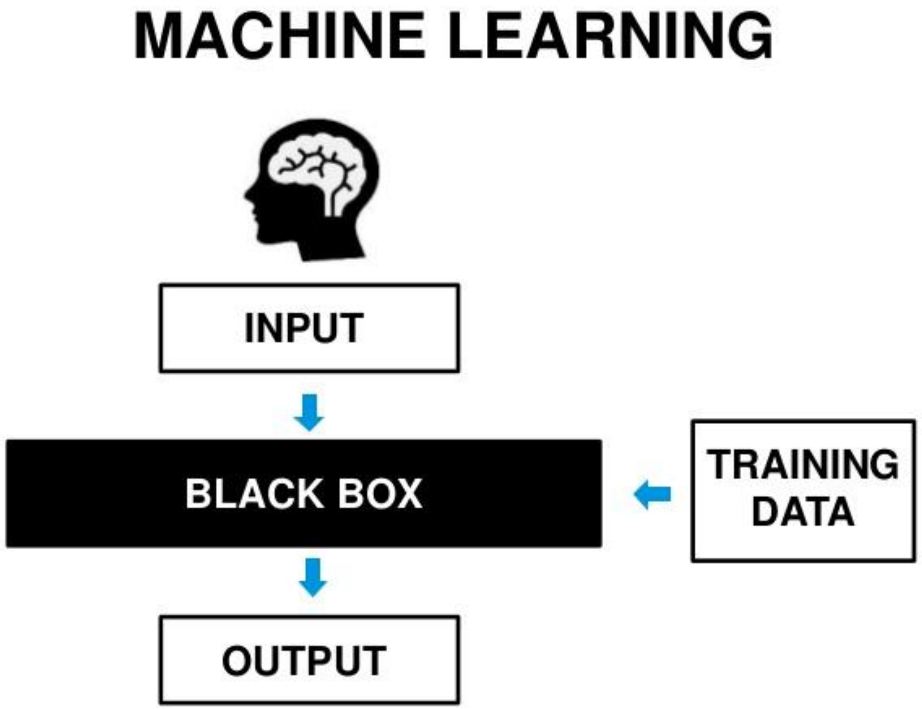
Data-Based AI = Machine Learning

Machine Learning 의 원리



http://www.slideshare.net/AlexPoon1

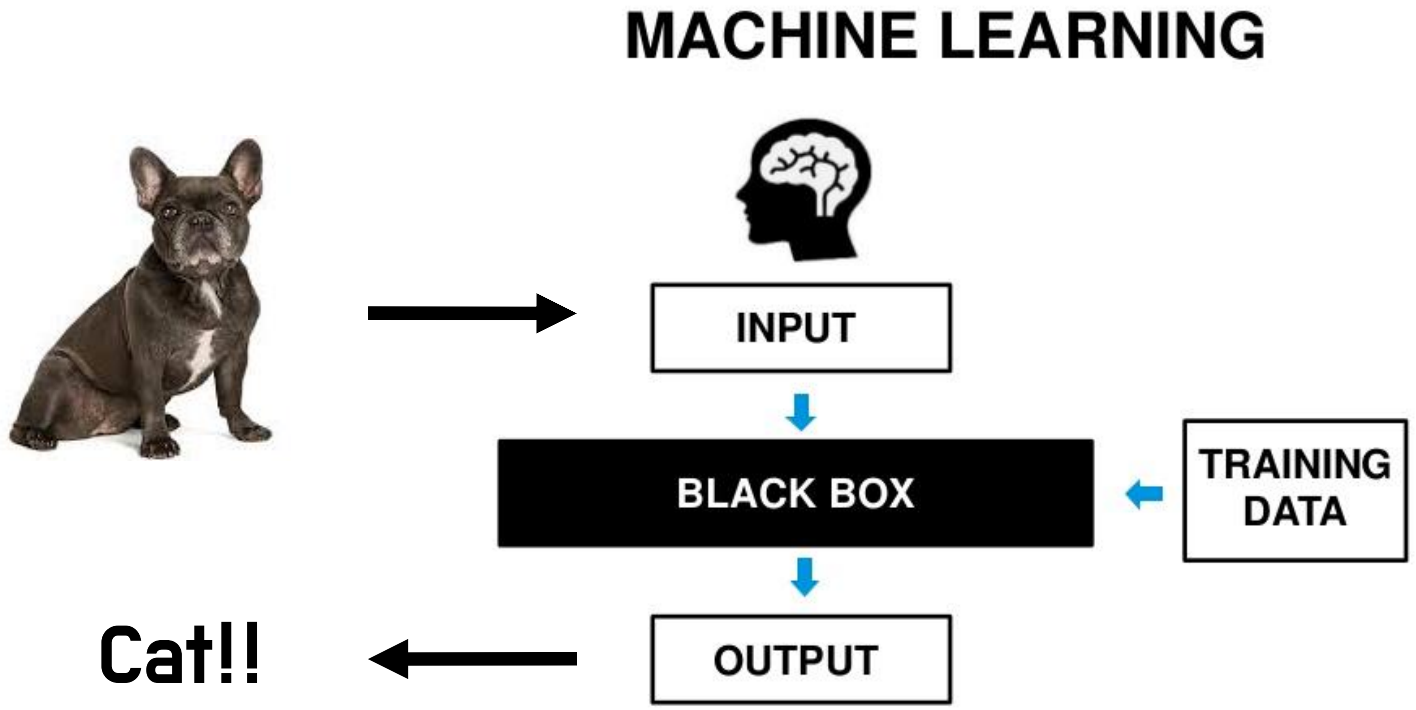
Machine Learning 의 원리



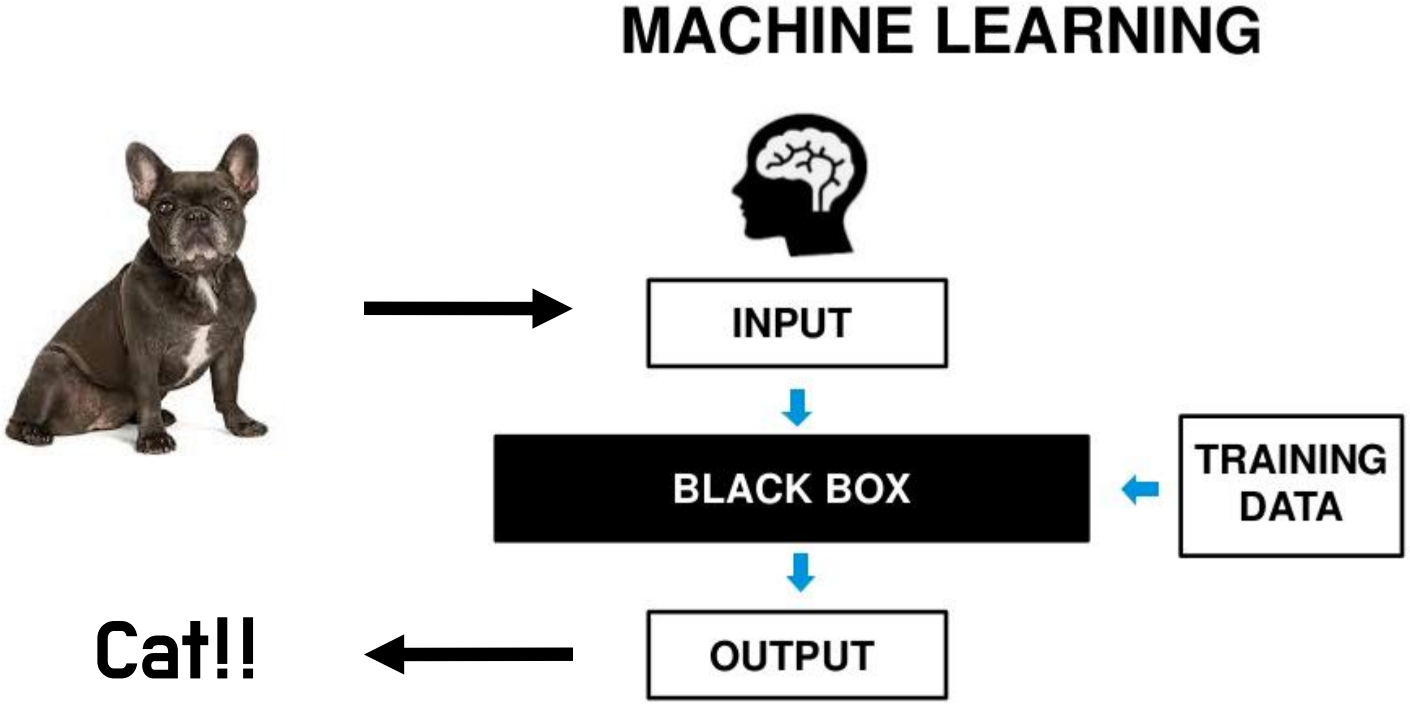
http://www.slideshare.net/AlexPoon1

목표: 개와 고양이를 구분하는 A.I. Black Box!

Machine Learning 의 원리

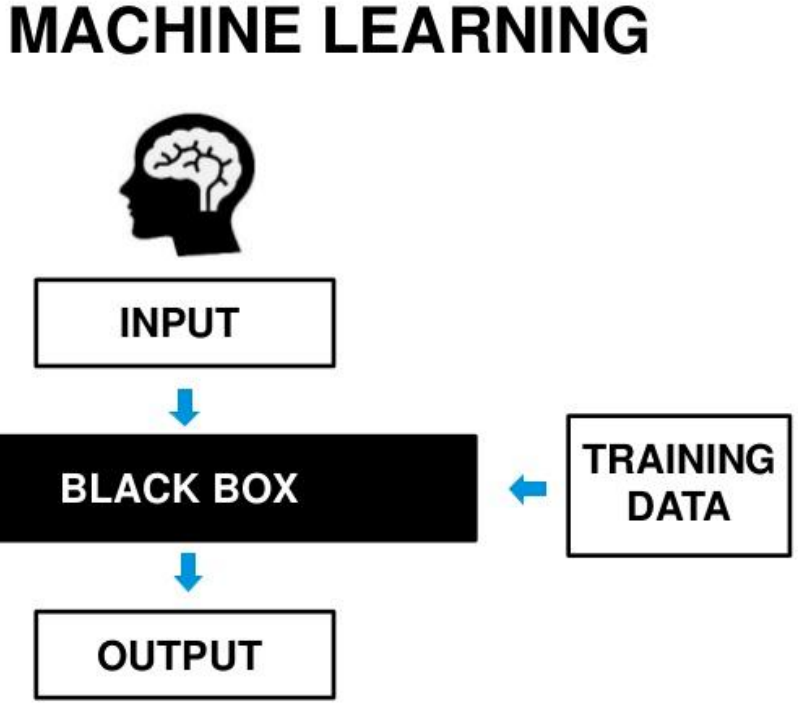


Machine Learning 의 원리



문제: Black Box가 개와 고양이를 구분하는 '규칙'을 알지 못한다!

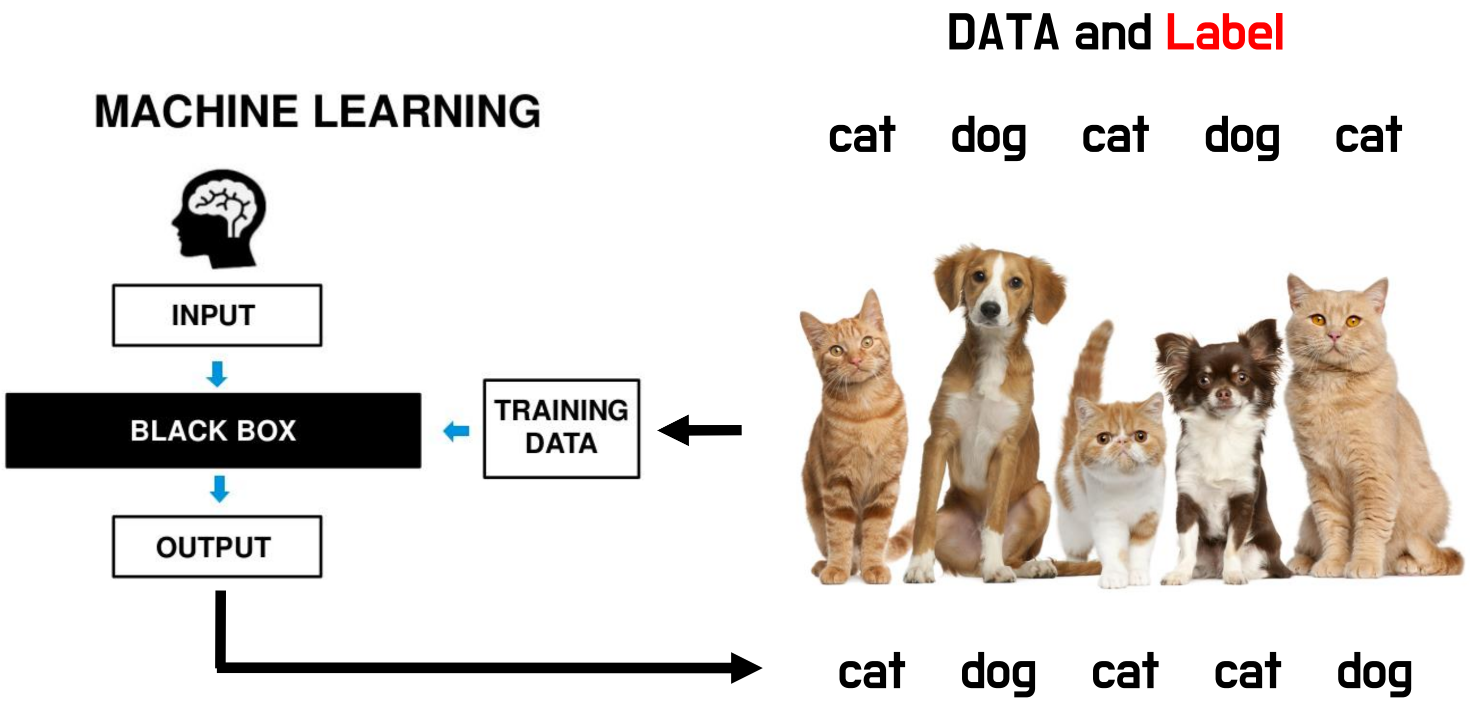
Machine Learning 의 원리



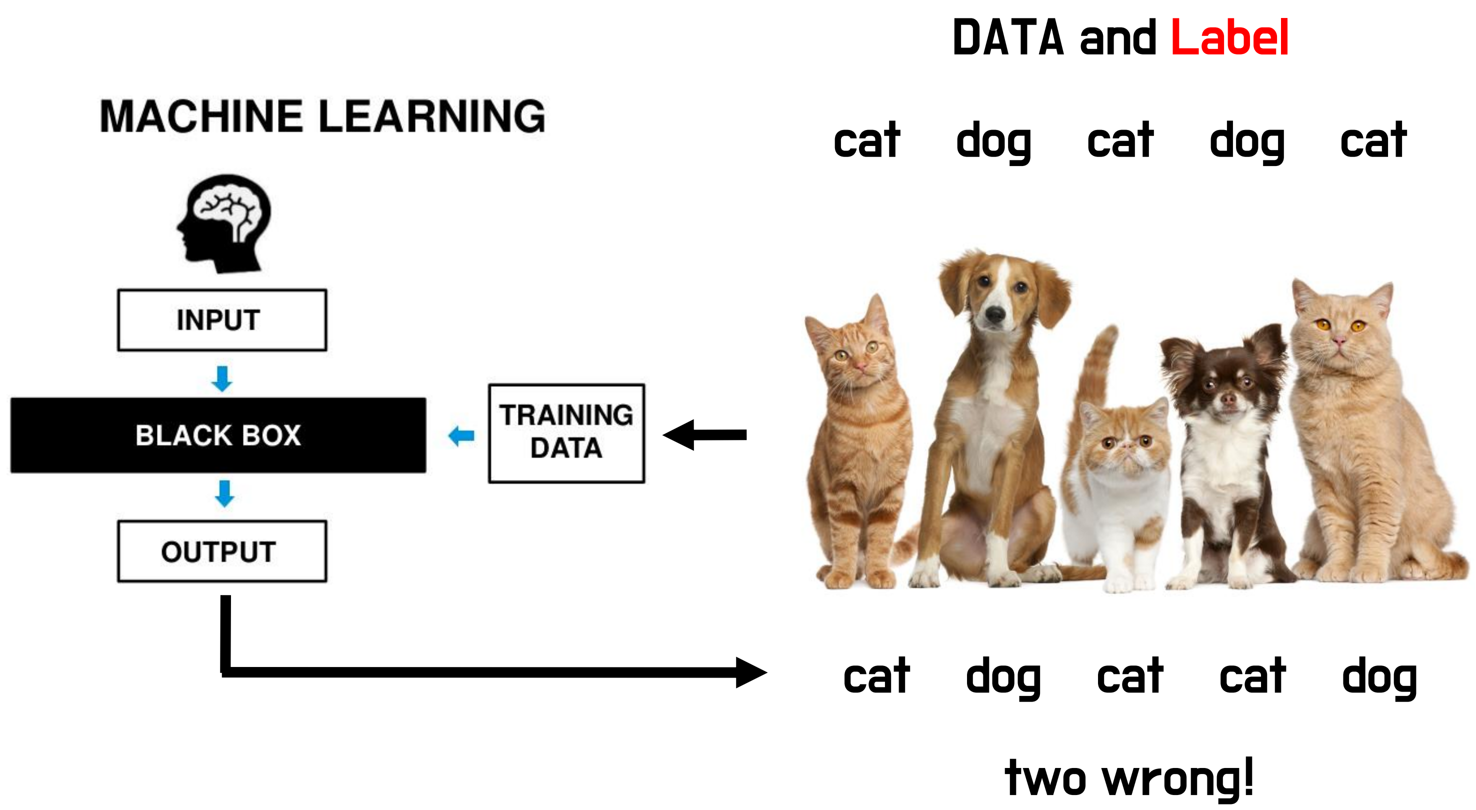
Cat!!

해결: Data를 통해 Black Box를 학습시켜 올바른 '규칙'을 찾도록 만들자!

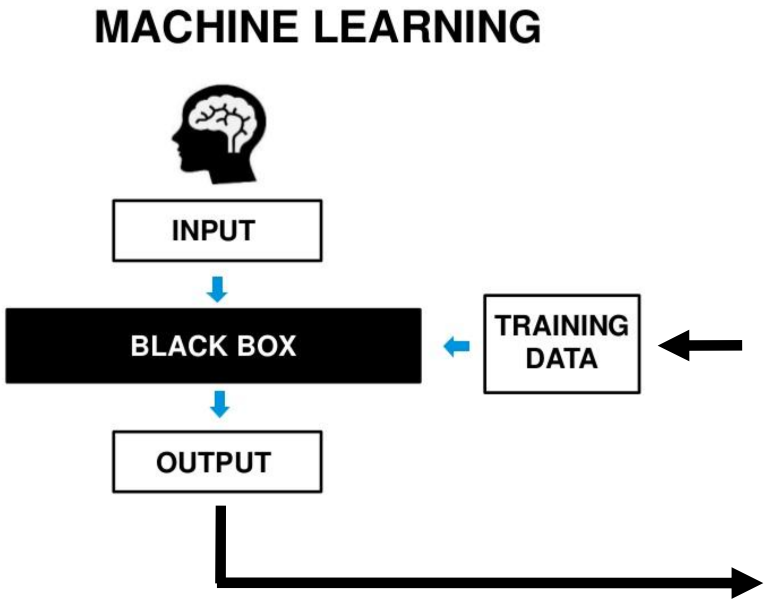
Machine Learning 의 원리



Machine Learning 의 원리



Machine Learning 의 원리



DATA and Label

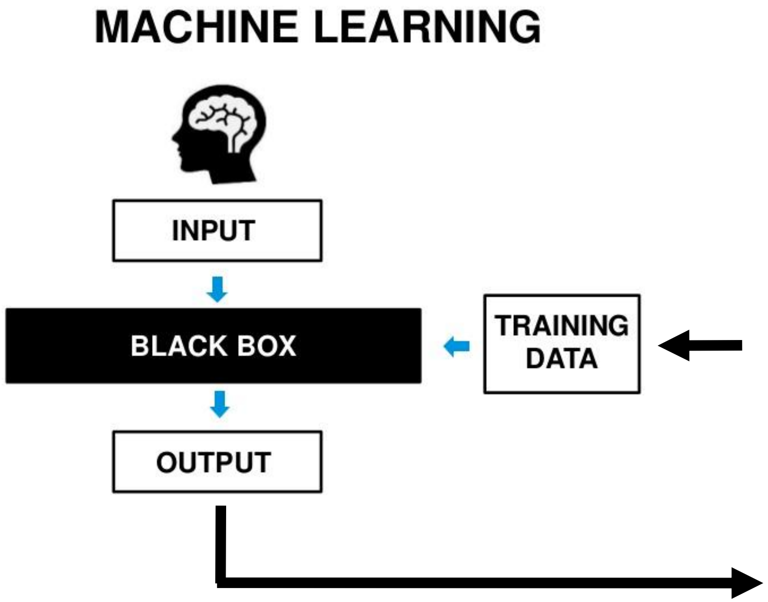
cat dog cat dog cat

cat dog cat cat dog

one wrong!

Black Box 내부에서 개와 고양이를 판단하는 규칙을 수정한다

Machine Learning 의 원리



DATA and Label

cat dog cat dog cat

cat dog cat dog dog

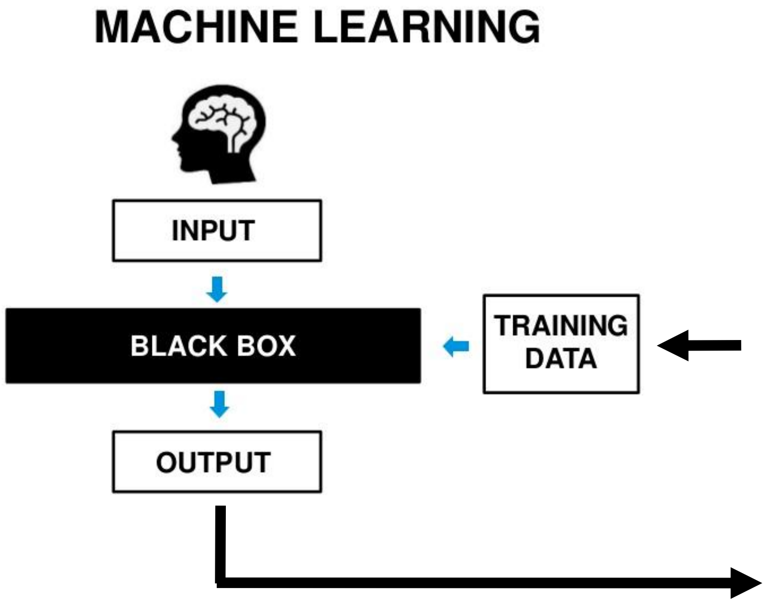
one wrong!

Black Box 내부에서 개와 고양이를 판단하는 규칙을 '계속' 수정한다

Machine Learning 의 원리

DATA and Label

cat dog cat dog cat



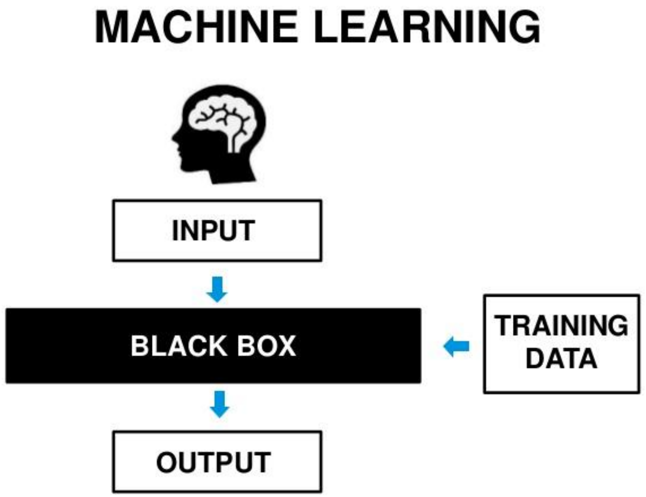
cat dog cat dog dog

two wrong!

Black Box 내부에서 개와 고양이를 판단하는 규칙을 수정한다

Machine Learning 의 원리

DATA and Label



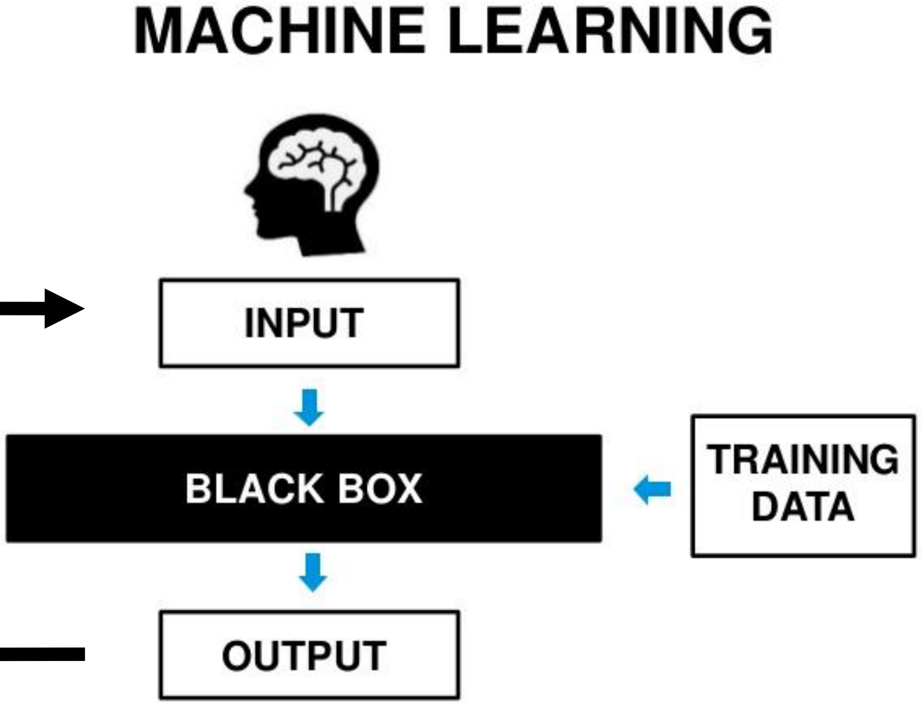
cat dog cat dog cat

cat dog cat dog cat

right answer!

Black Box 는 학습을 마쳤다!

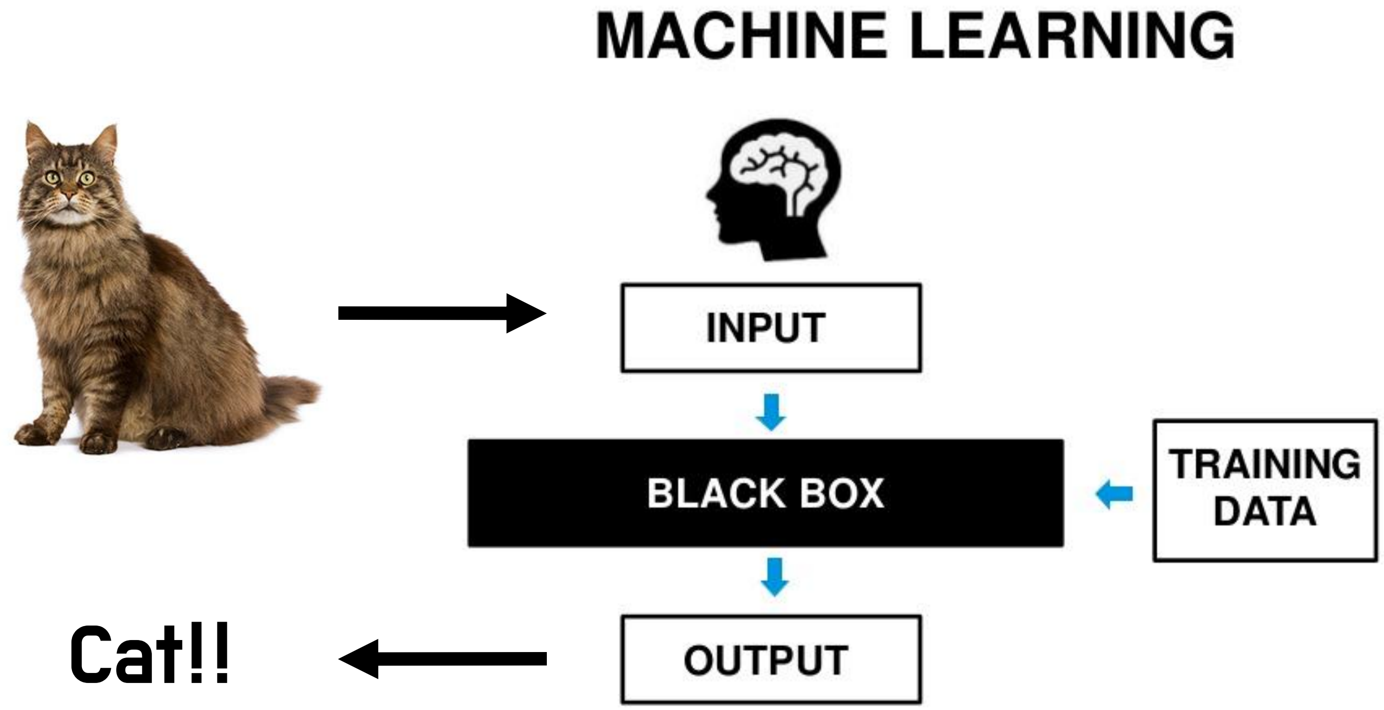
Machine Learning 의 원리



Dog!!

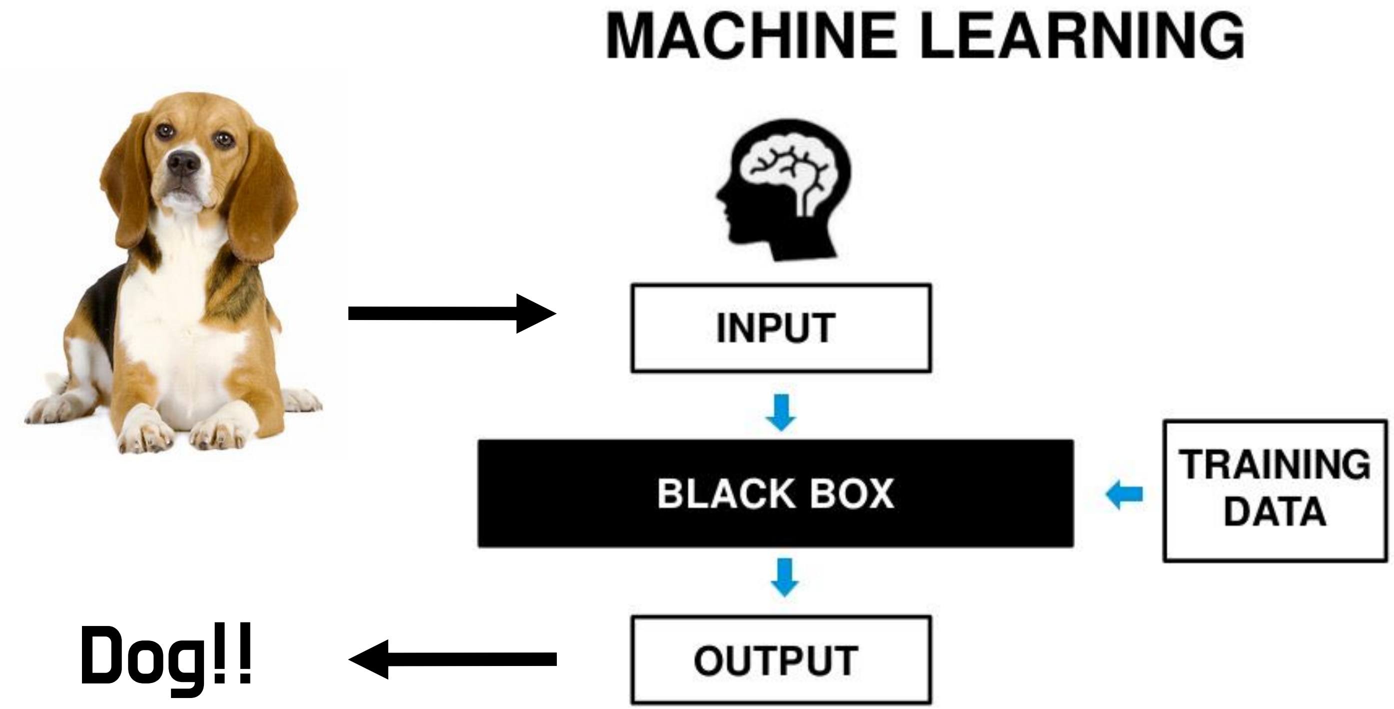
학습한 내용을 바탕으로 올바른 판단을 내릴 수 있게 되었다!

Machine Learning 의 원리



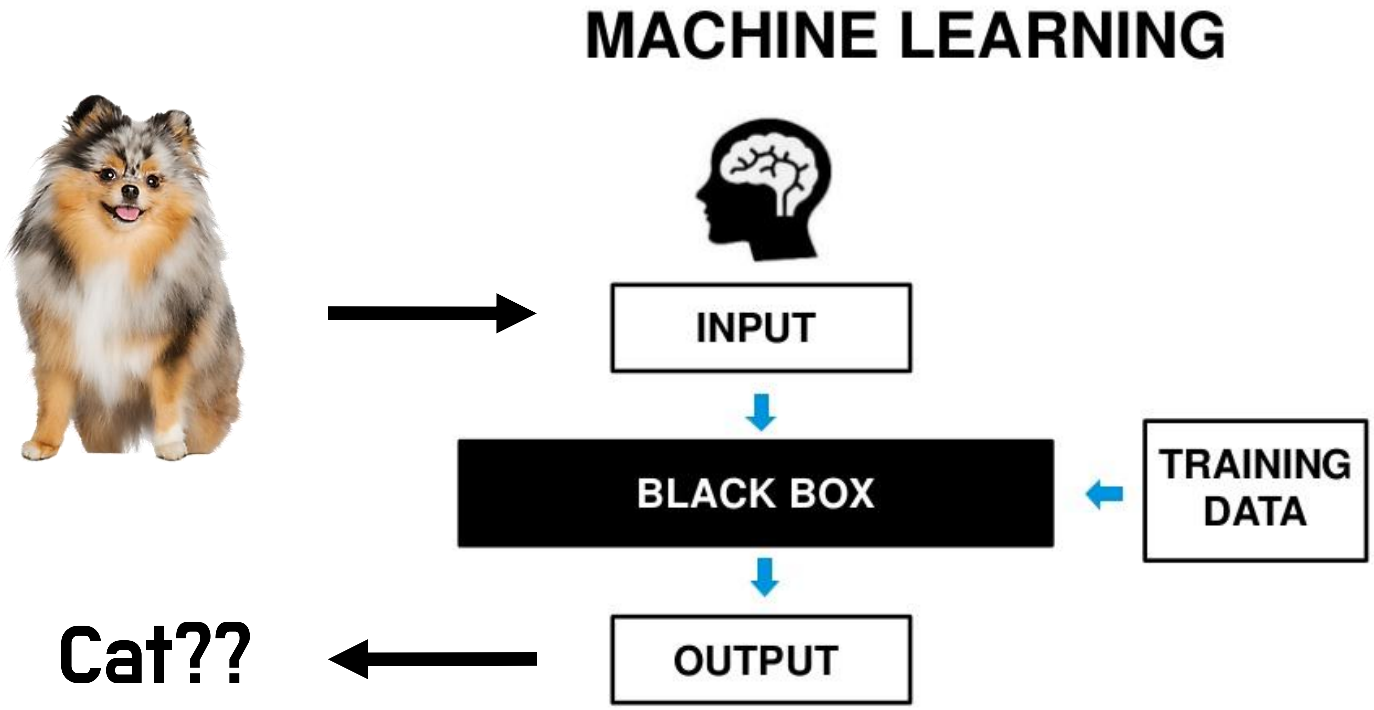
학습한 내용을 바탕으로 올바른 판단을 내릴 수 있게 되었다!

Machine Learning 의 원리



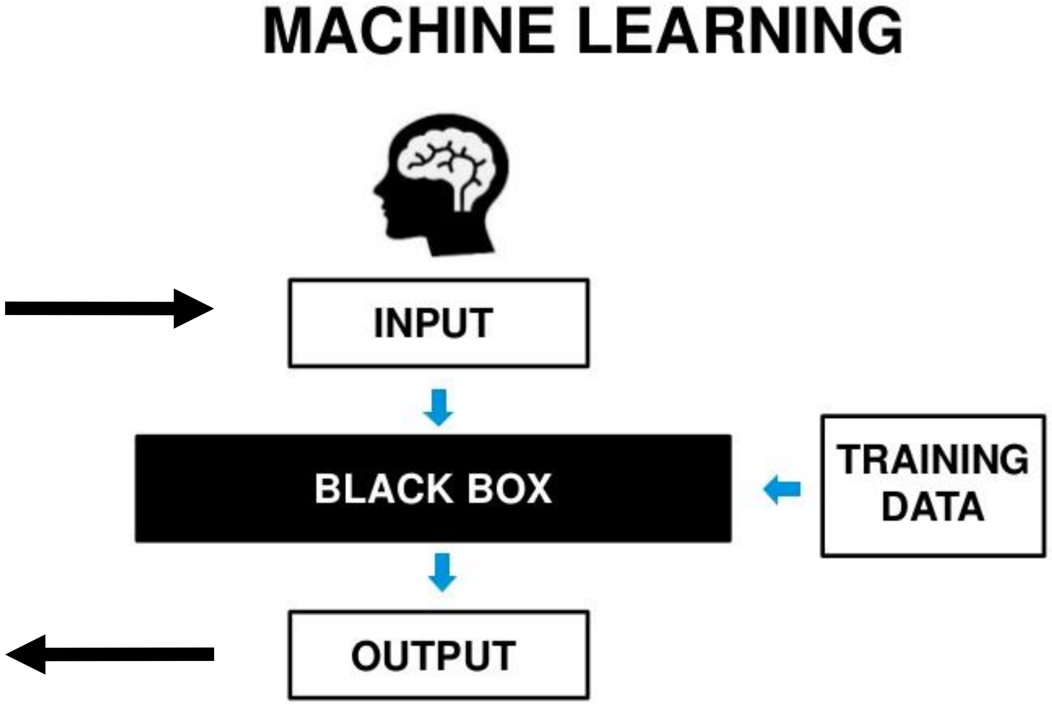
학습한 내용을 바탕으로 올바른 판단을 내릴 수 있게 되었다!

Machine Learning 의 원리



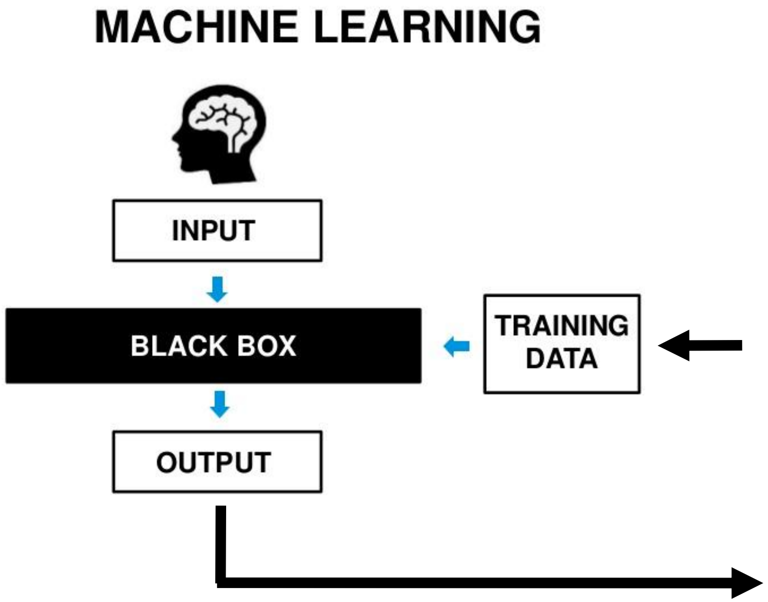
어떠한 사진은 제대로 판단하지 못할 수 있다!
완전히 강아지와 고양이를 구분하는 규칙을 파악하지 못한 것

Machine Learning 의 원리



모든 강아지와 고양이를 완전하게 구분하는 규칙을 찾기 위해서는?

Machine Learning 의 원리



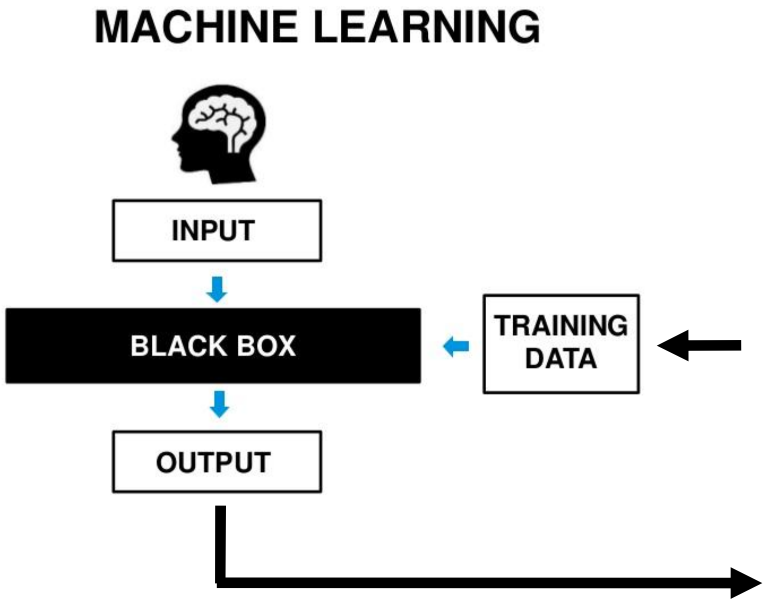
DATA and Label

cat dog cat dog cat

cat dog cat cat dog

5마리의 개와 고양이 사진으로 모든 규칙을 학습 할 수 있나요?

Machine Learning 의 원리



DATA and Label

cat dog cat dog cat

cat dog cat cat dog

5마리의 개와 고양이 사진으로 모든 규칙을 학습 할 수 있나요?

NO

Machine Learning 과 Data

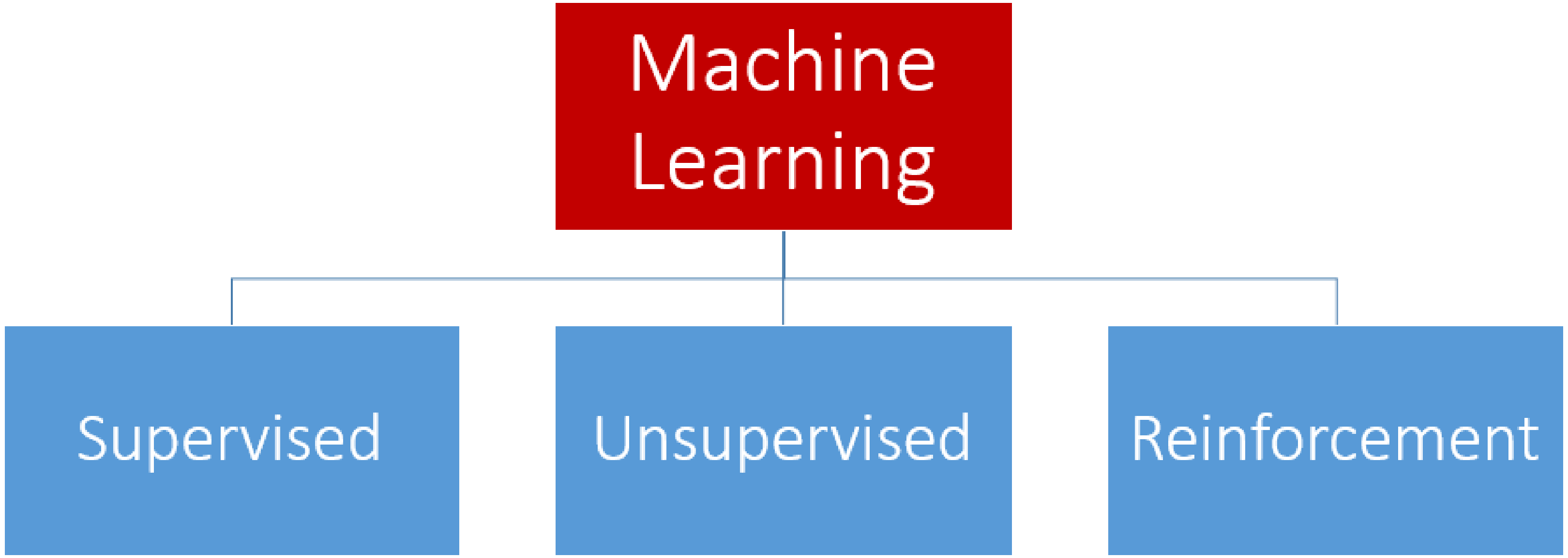


모든 규칙을 완전히 학습하기 위해서는 엄청나게 많은 훈련 Data가 필요

Machine Learning 과 Data

학습 데이터의 양이 많으면 많을 수록 정확도가 상승!

Three Types of Machine Learning

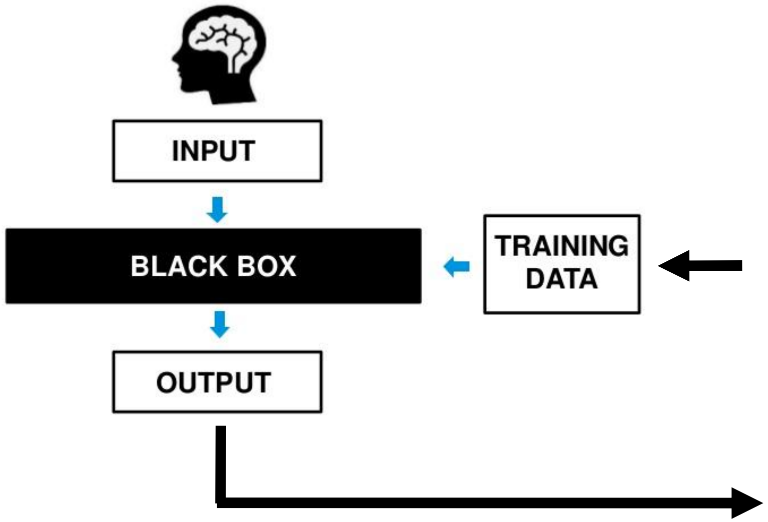


Supervised Learning

DATA and Label

cat dog cat dog cat

MACHINE LEARNING



cat dog cat cat dog

Supervised Learning: Learning from labeled data

Supervised Learning

Supervised Learning: Learning from labeled data

Image Classification

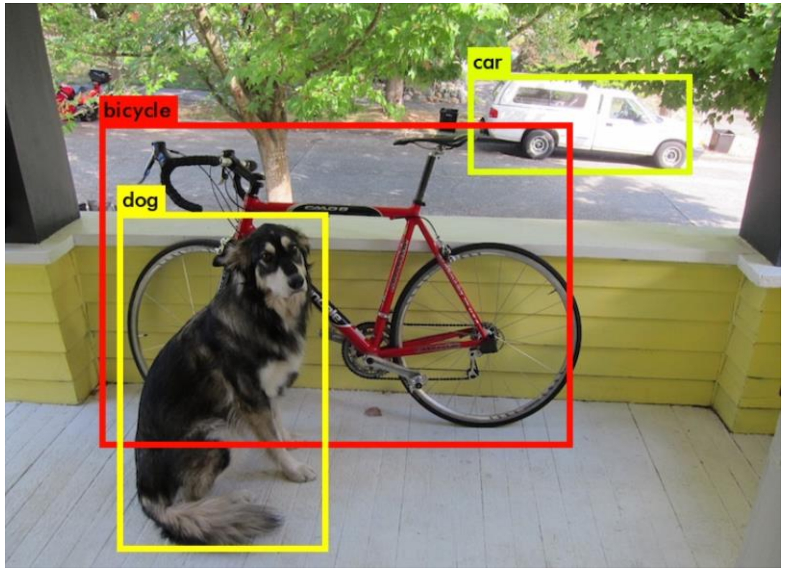
cat dog cat dog cat

Data: Pictures Label: Classes

Supervised Learning

Supervised Learning: Learning from labeled data

Object Detection

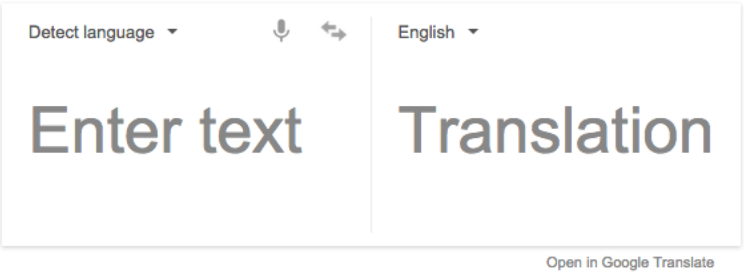


Data: Pictures **Label**: Boxes & Classes

Supervised Learning

Supervised Learning: Learning from labeled data

Machine Translation

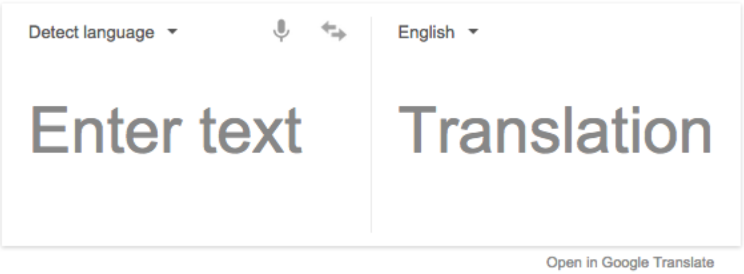


Data: 번역 이전의 문장 Label: 번역된 문장

Supervised Learning

Supervised Learning: Learning from labeled data

Machine Translation



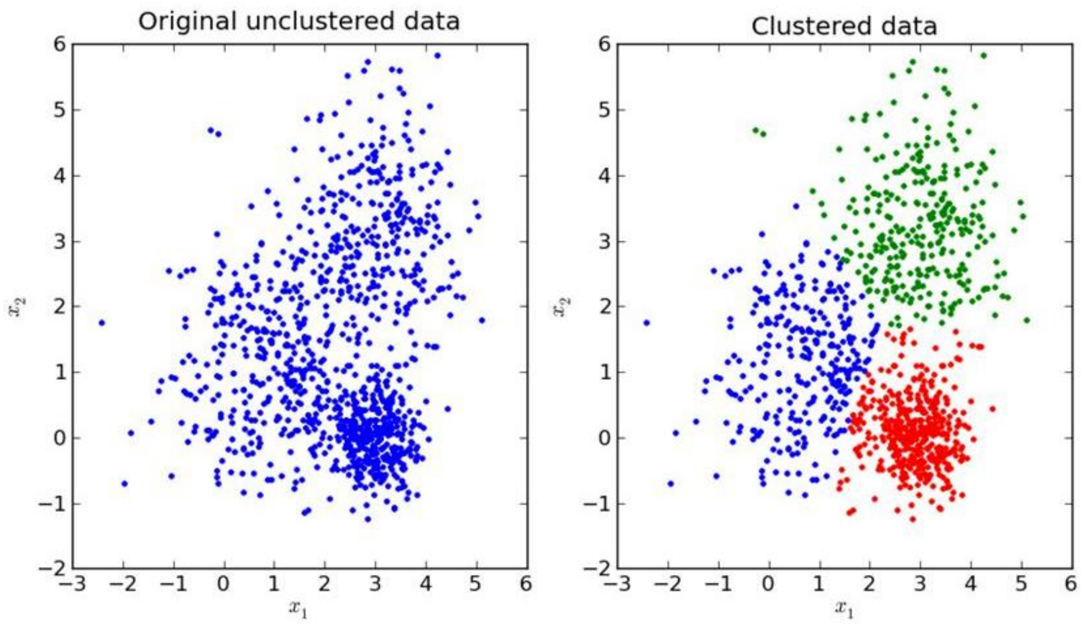
Data: 번역 이전의 문장   Label: 번역된 문장

대부분의 Machine Learning 적용 사례는 Supervised Learning!

Supervised Learning

Unsupervised Learning: Learning from unlabeled data

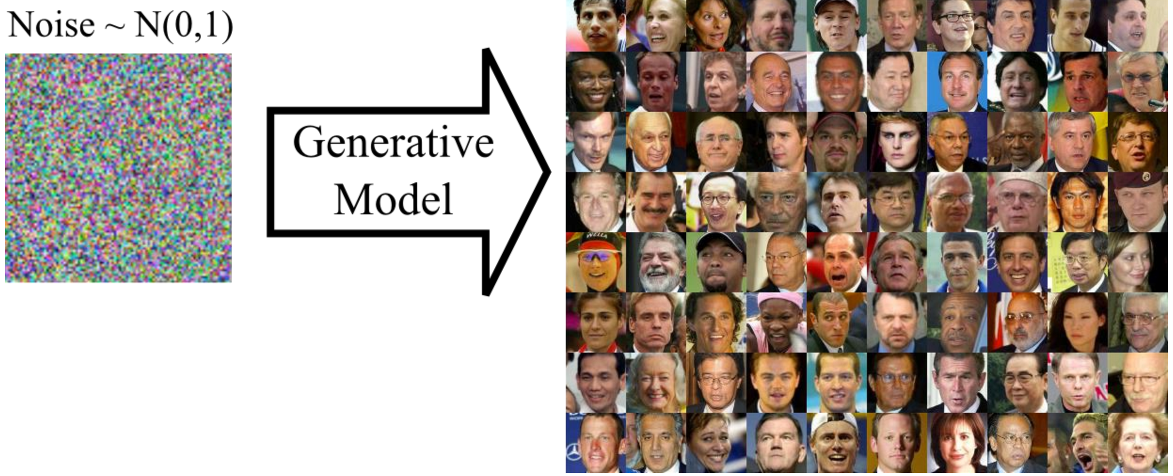
Clustering



Supervised Learning

Unsupervised Learning: Learning from unlabeled data

Image Generation

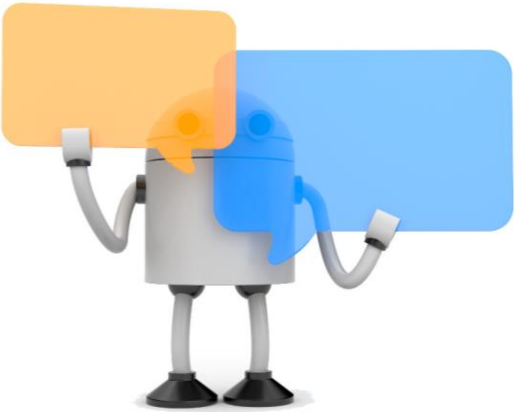


Reinforcement Learning

Reinforcement Learning: Evaluate after action

Game Play

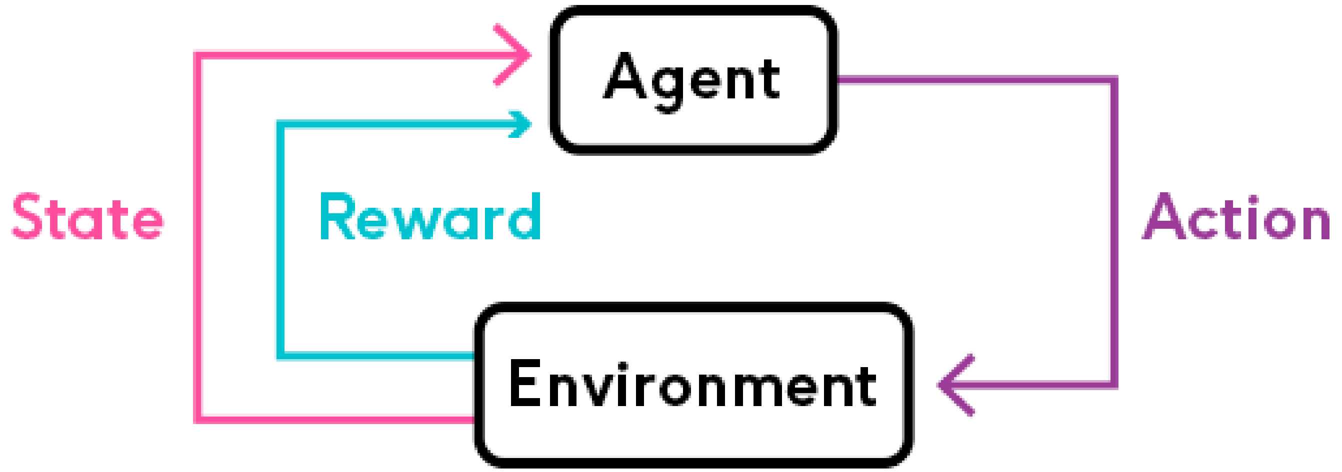
Chatbot



Reinforcement Learning

Reinforcement Learning: Evaluate after action

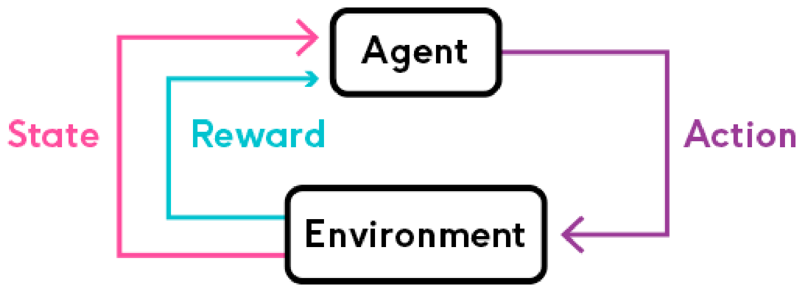
일단 Action을 취한 뒤 Reward를 받고 새로운 State를 가진다.
각 State에서 높은 Reward를 받을 수 있는 Action의 확률을 높인다!

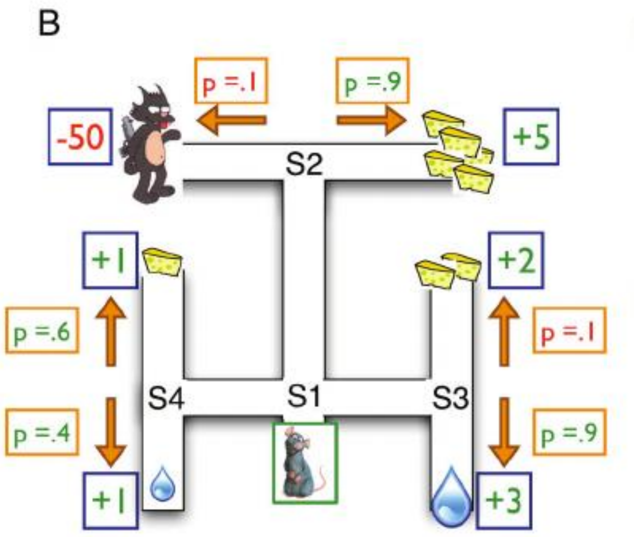


Reinforcement Learning

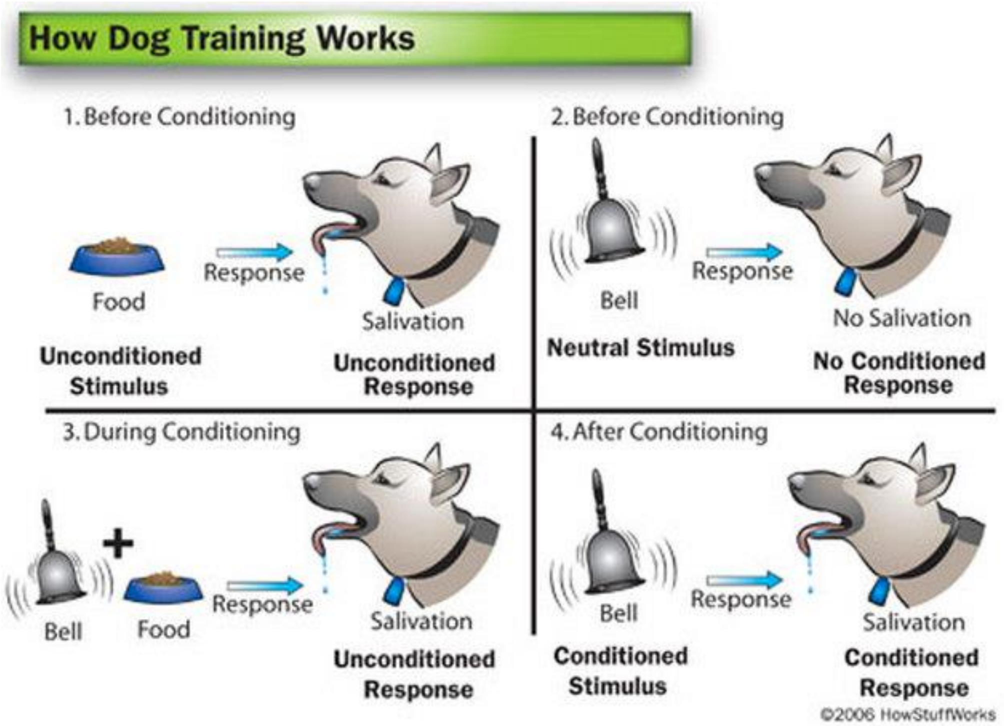
Reinforcement Learning: Evaluate after action

일단 Action을 취한 뒤 Reward를 받고 새로운 State를 가진다.
각 State에서 높은 Reward를 받을 수 있는 Action의 확률을 높인다!





Reinforcement Learning



인간/동물의 학습방식과 가장 유사한 학습방식

Three Types of ML 비교

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
개념	Learning from labeled data	Learning from unlabeled data	Evaluate after Action
적용	Classification Object Detection Machine Translation ...	Clustering Image Generation ...	Game AI Chatbot System ...
기타	대부분의 Machine Learning 사례	-	인간의 학습방식과 가장 유사
국문	지도학습	비지도학습	강화학습

Focus on Supervised Learning

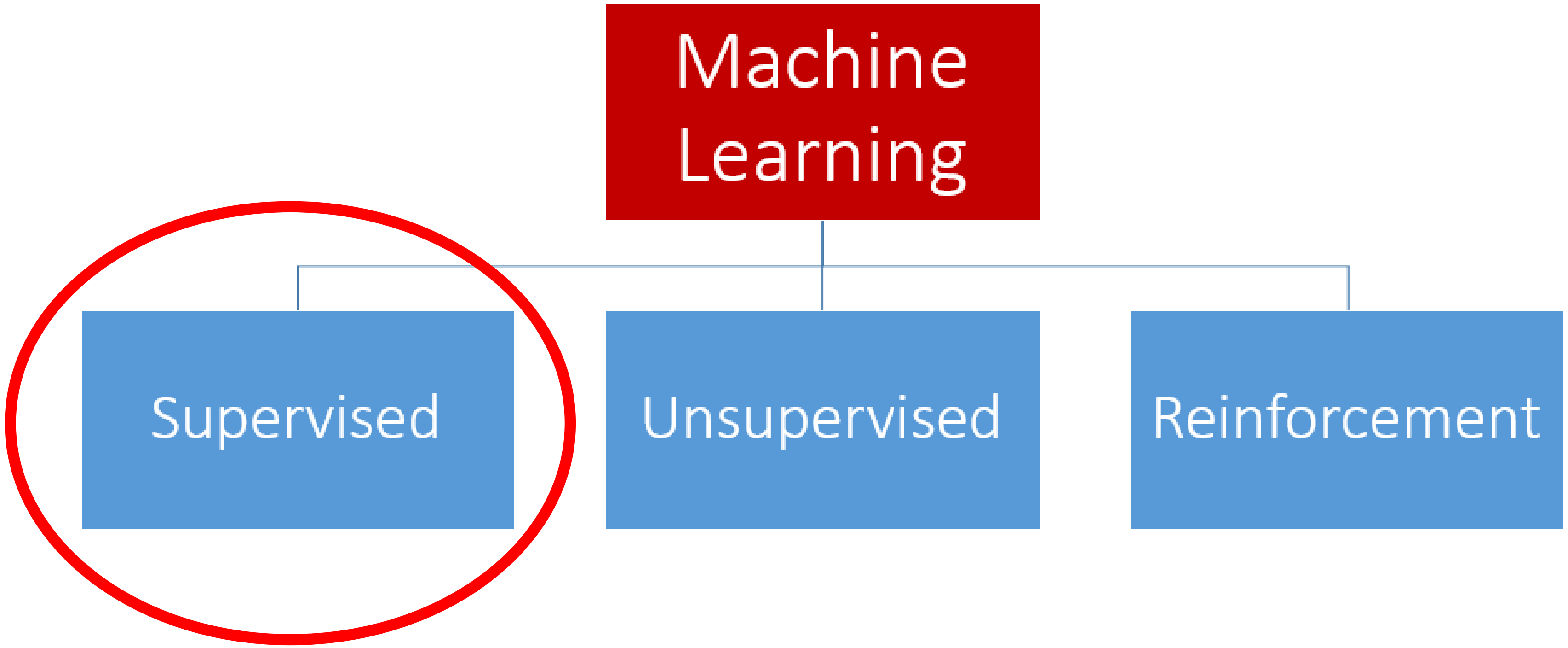


Image Classification

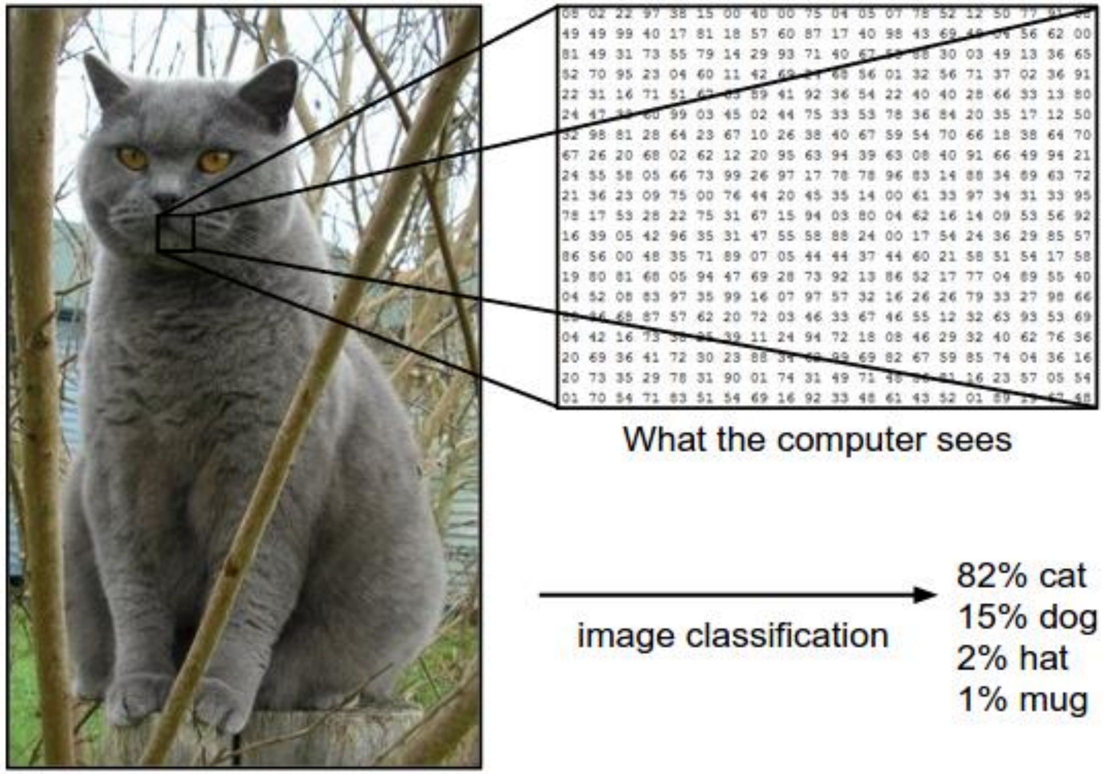
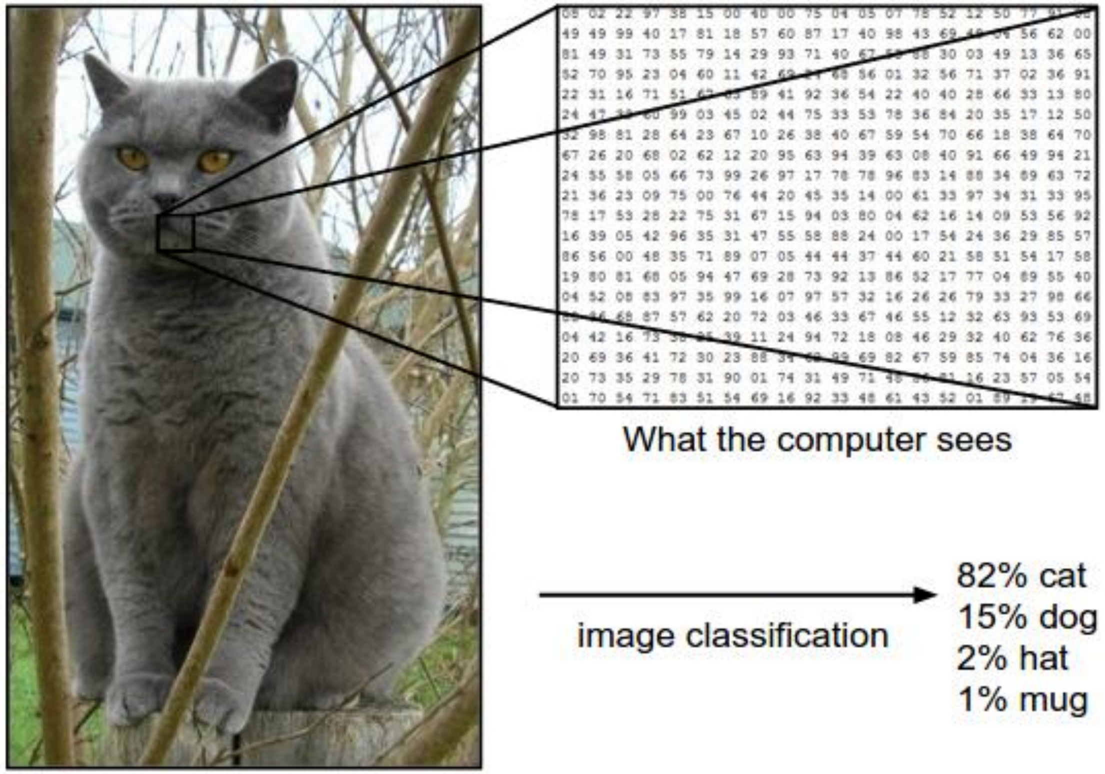


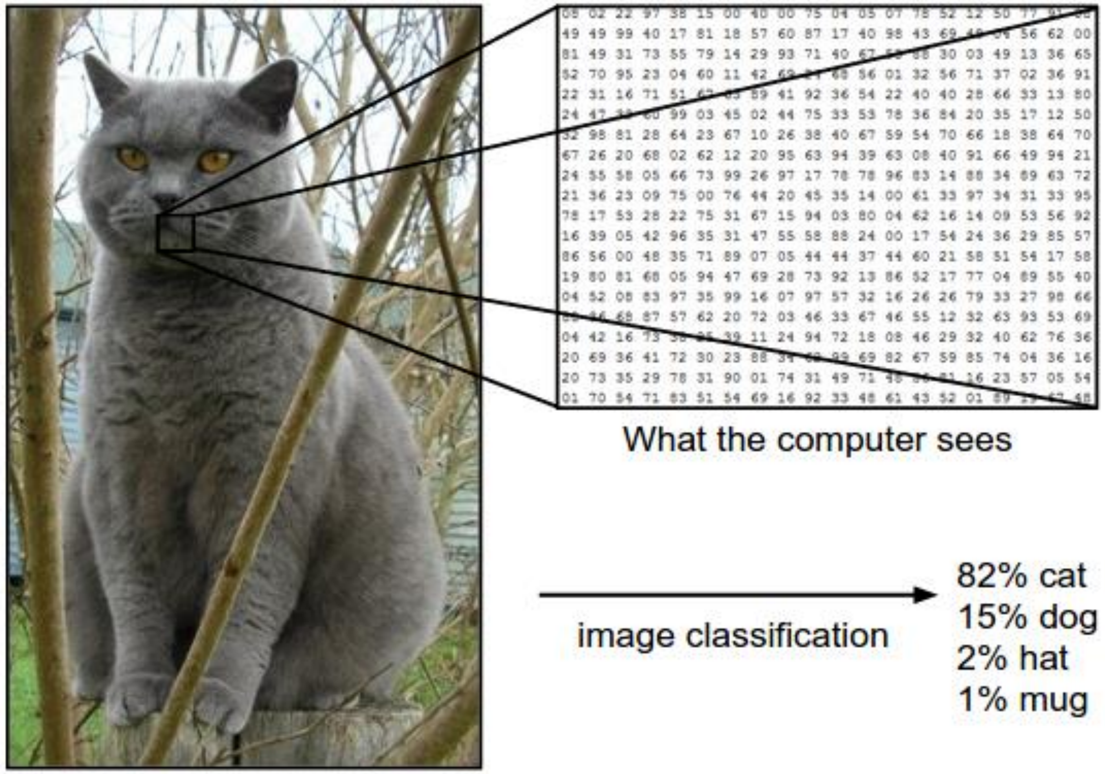
Image Classification



DATA and Label

cat dog cat dog cat

Image Classification



$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

2 rows, 3 columns
2x3 matrix

For Image,
32x32 image is
32x32 matrix

Linear Algebra

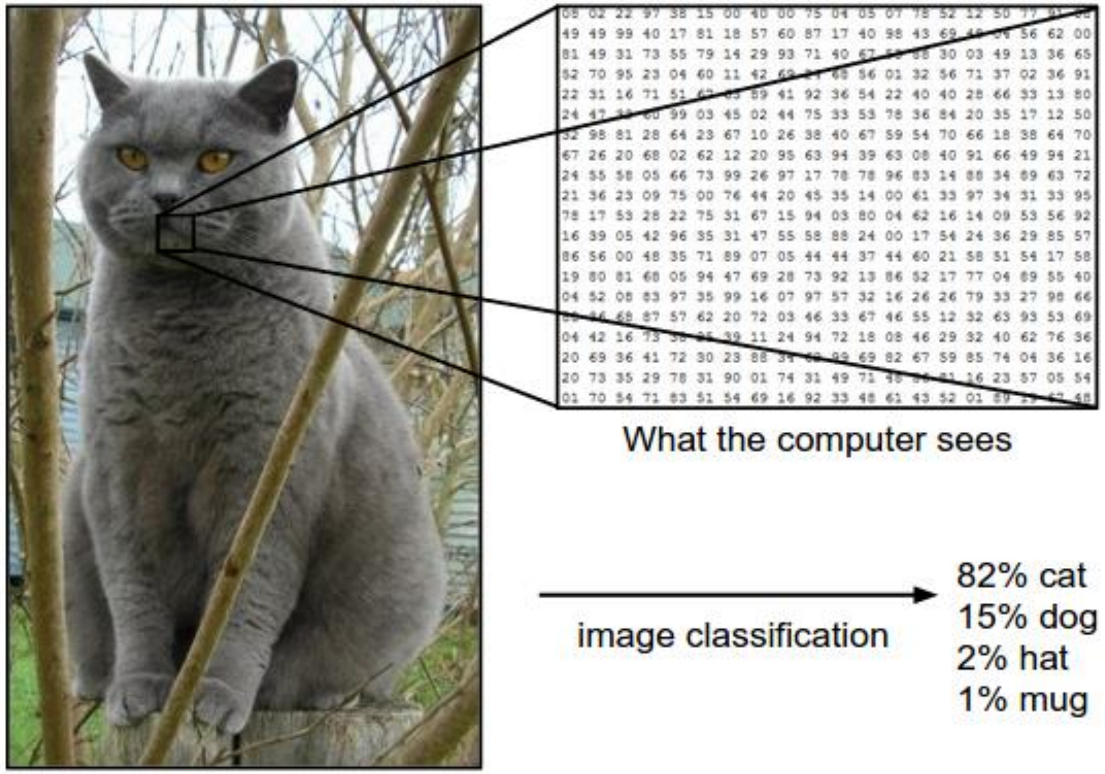
$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

2 rows, 3 columns
2x3 matrix

$$\begin{bmatrix} 1 \\ -9 \\ 8 \end{bmatrix}$$

We call nx1 or 1xn matrix
"vector"

Image Classification



$$\begin{bmatrix} 1 \\ -9 \\ 8 \end{bmatrix}$$

vector

For Image,
32x32 image is
1024-size vector

Linear Algebra

$-(2)=-2$

$$-\begin{bmatrix} 2 & -4 \\ 7 & 10 \end{bmatrix} = \begin{bmatrix} -2 & 4 \\ -7 & -10 \end{bmatrix}$$

negative

$3-4=-1$

$$\begin{bmatrix} 3 & 8 \\ 4 & 6 \end{bmatrix} - \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} -1 & 8 \\ 3 & 15 \end{bmatrix}$$

subtracting

The two matrices must
be the same size.

Linear Algebra

$2 \times 4 = 8$

$$2 \times \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 2 & -18 \end{bmatrix}$$

multiply by constant

We call the constant a **scalar**,
so officially this is called "scalar multiplication".

Linear Algebra

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

transposing

To "transpose" a matrix, swap the rows and columns.

We put a "T" in the top right-hand corner to mean transpose:

Linear Algebra

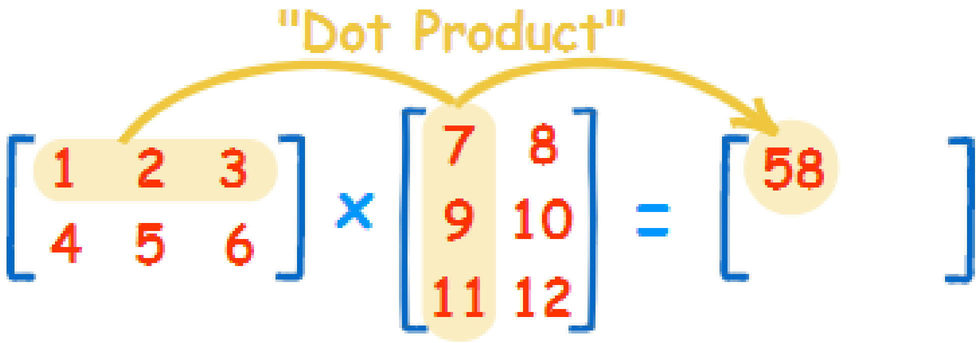
$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$$

notation

Rows go left-right

Columns go up-down

Linear Algebra



Matrix Multiplication

The "**Dot Product**" is where we **multiply matching members**, **then sum up:**

$(1, 2, 3) \cdot (7, 9, 11) = 1\times7 + 2\times9 + 3\times11 = 58$

AxB = C, $c_{i,j}$ is dot product of i_{th} row of A and j_{th} row of B

Linear Algebra

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ & \end{bmatrix}$$

Matrix Multiplication

$$(1, 2, 3) \cdot (8, 10, 12) = 1\times8 + 2\times10 + 3\times12 = 64$$

Linear Algebra

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

Matrix Multiplication

We can do the same thing for the 2nd row and 1st column:
$(4, 5, 6) \cdot (7, 9, 11) = 4×7 + 5×9 + 6×11 = 139$

And for the 2nd row and 2nd column:
$(4, 5, 6) \cdot (8, 10, 12) = 4×8 + 5×10 + 6×12 = 154$

Linear Algebra

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

Matrix Multiplication

A: 2x3 matrix, B: 3x2 matrix
C=AxB: 2x2 matrix

A: nxm, B: mxk
C=AxB: nxk matrix

Image Classification

test.py

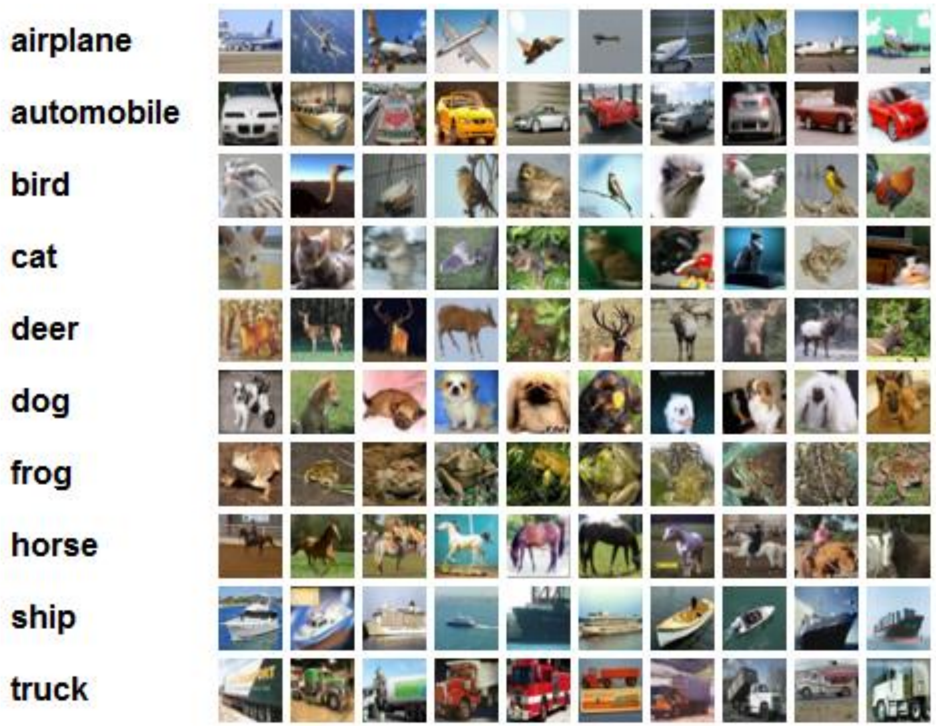
```python
class ImageClassifier:

    def train(self, images, labels):
        model = --
        # Machine Learning Models!
        return model

    def predict(self, model, test_images):
        test_labels = --
        # Use model to predict labels
        return test_labels

```

train the model

predict the label from trained model!

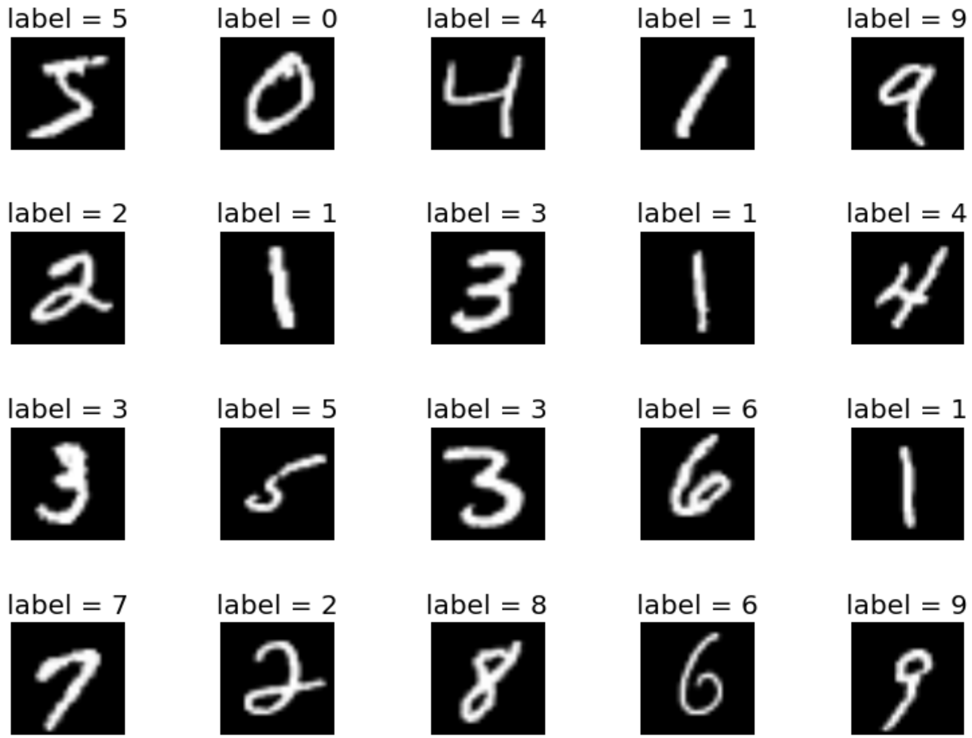
or infer (inference)

Popular & Simple Datasets



CIFAR10

32x32x3



MNIST

28x28x1

Image Classification with ML

test.py

```python
class ImageClassifier:

    def train(self, images, labels):
        model = --
        # Machine Learning Models!
        return model

    def predict(self, model, test_images):
        test_labels = --
        # Use model to predict labels
        return test_labels

```

train the model

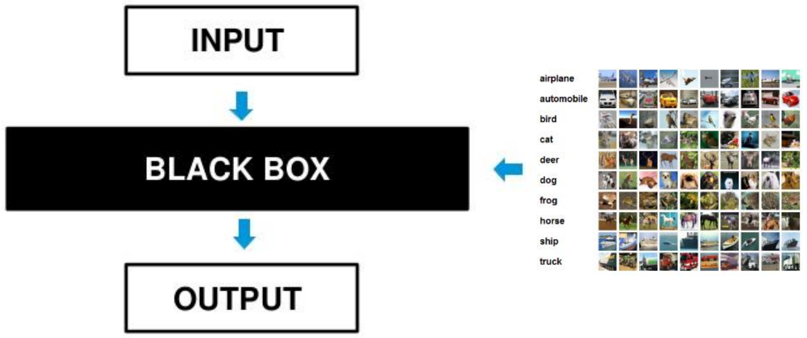
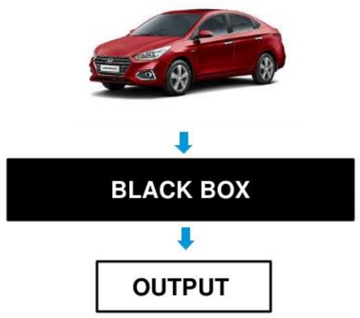


Image Classification with ML

test.py

```python
class ImageClassifier:

    def train(self, images, labels):
        model = --
        # Machine Learning Models!
        return model

    def predict(self, model, test_images):
        test_labels = --
        # Use model to predict labels
        return test_labels

```



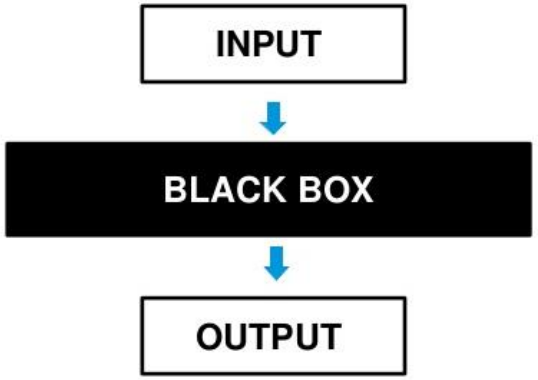
predict the label from trained model!

or infer (inference)

Image Classification with ML

test.py

```
class ImageClassifier:

    def train(self, images, labels):
        model = --
        # Machine Learning Models!
        return model

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



Then,
What is model? (black box)

Linear Classifier

model is mathematical function!

Image

Array of 32x32x3 numbers
(3072 numbers total)

model

f(x, W)

10 numbers giving
class scores

W

parameters or weights

Linear Classifier

Linear Classifer

$f(x, W) = Wx$

Image

Array of 32x32x3 numbers
(3072 numbers total)

model

$f(x, W)$

W

parameters or weights

10 numbers giving
class scores

Linear Classifier

Linear Classifer

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

3072x1
vector

Image

Array of 32x32x3 numbers
(3072 numbers total)

model

$f(x, W)$

W

parameters or weights

10 numbers giving
class scores

Linear Classifier

Image

Array of 32x32x3 numbers
(3072 numbers total)

Linear Classifer

10x1 vector $\boxed{f(x, W)} = W\boxed{x}$ 3072x1 vector

model

$f(x, W)$

W

parameters or weights

10 numbers giving class scores

Linear Classifier

Linear Classifer

10x1 vector

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

3072x1 vector

10x3072 matrix !

Image

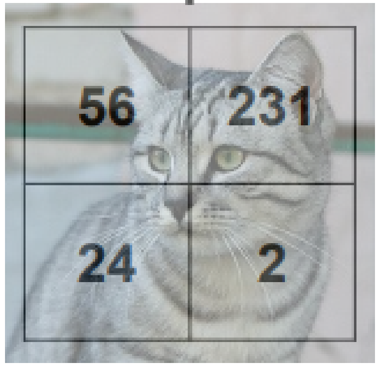
Array of 32x32x3 numbers
(3072 numbers total)

f(x, W)

W
parameters or weights

10 numbers giving
class scores

Linear Classifier (Simpler)



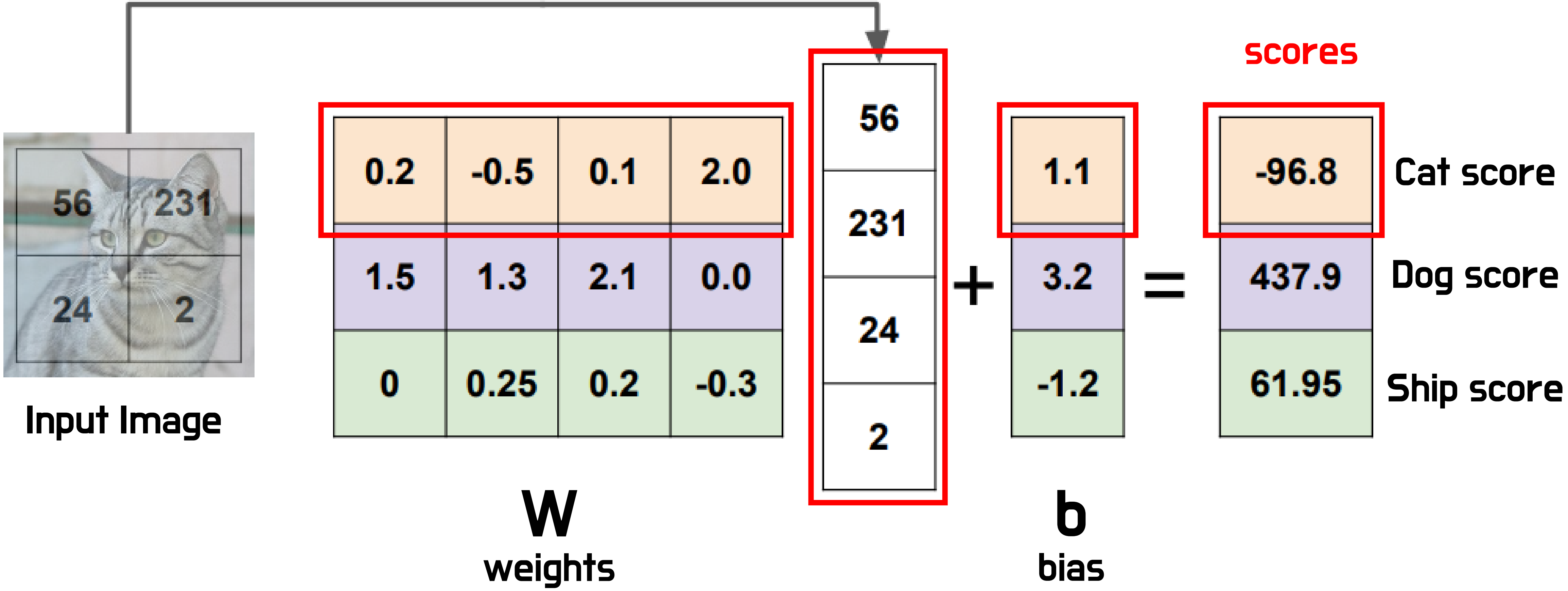
Input Image

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

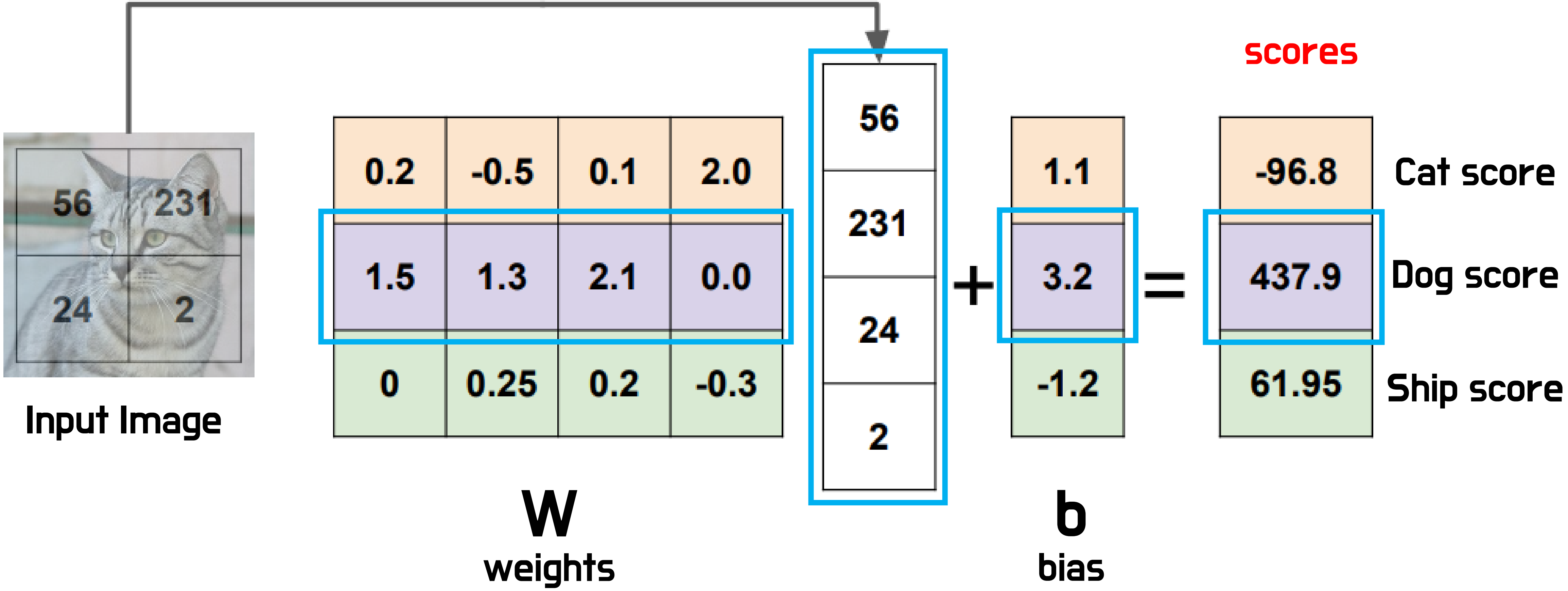
weights, this should be changed during training

bias, also

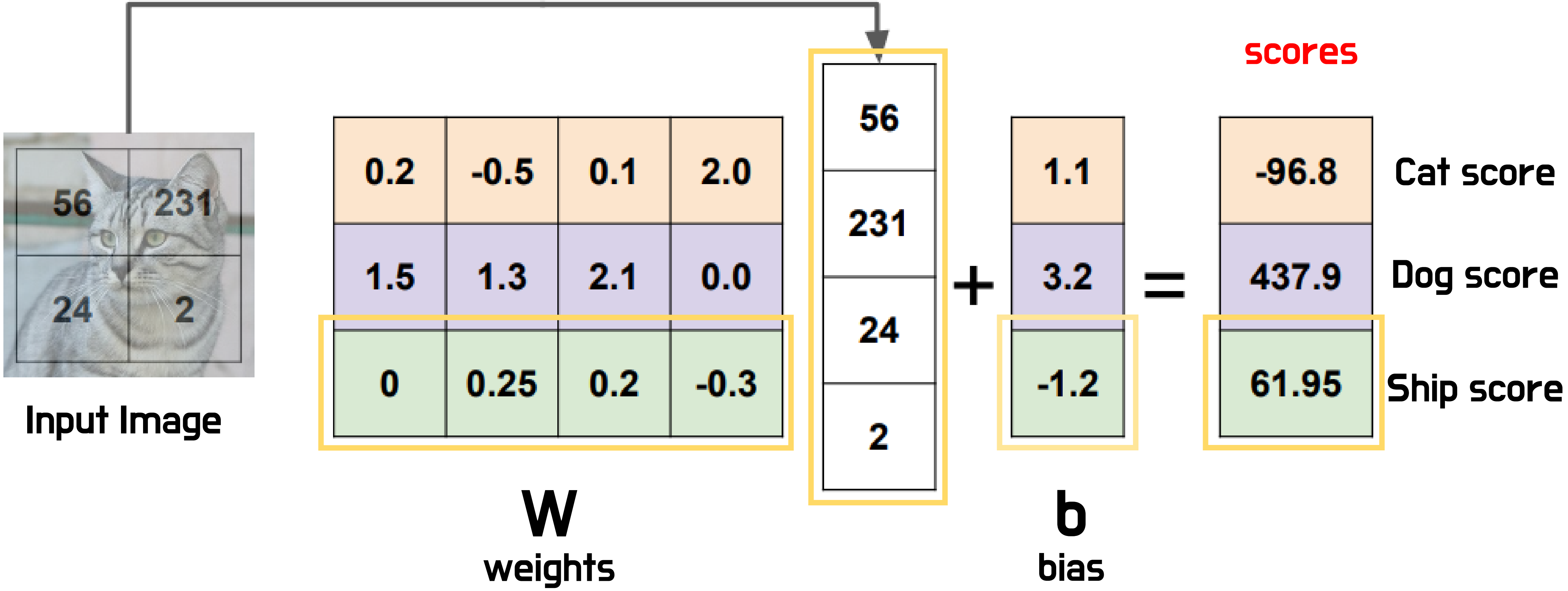
Linear Classifier (Simpler)



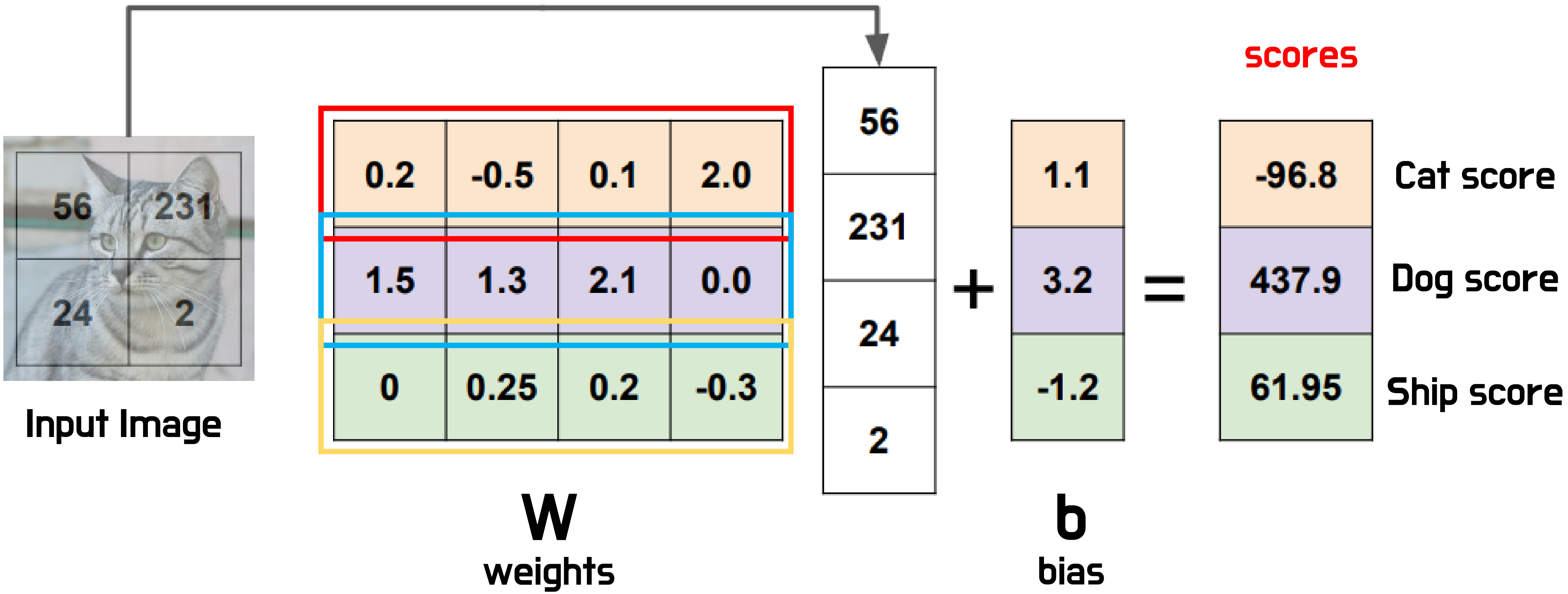
Linear Classifier (Simpler)



Linear Classifier (Simpler)



Linear Classifier (Simpler)



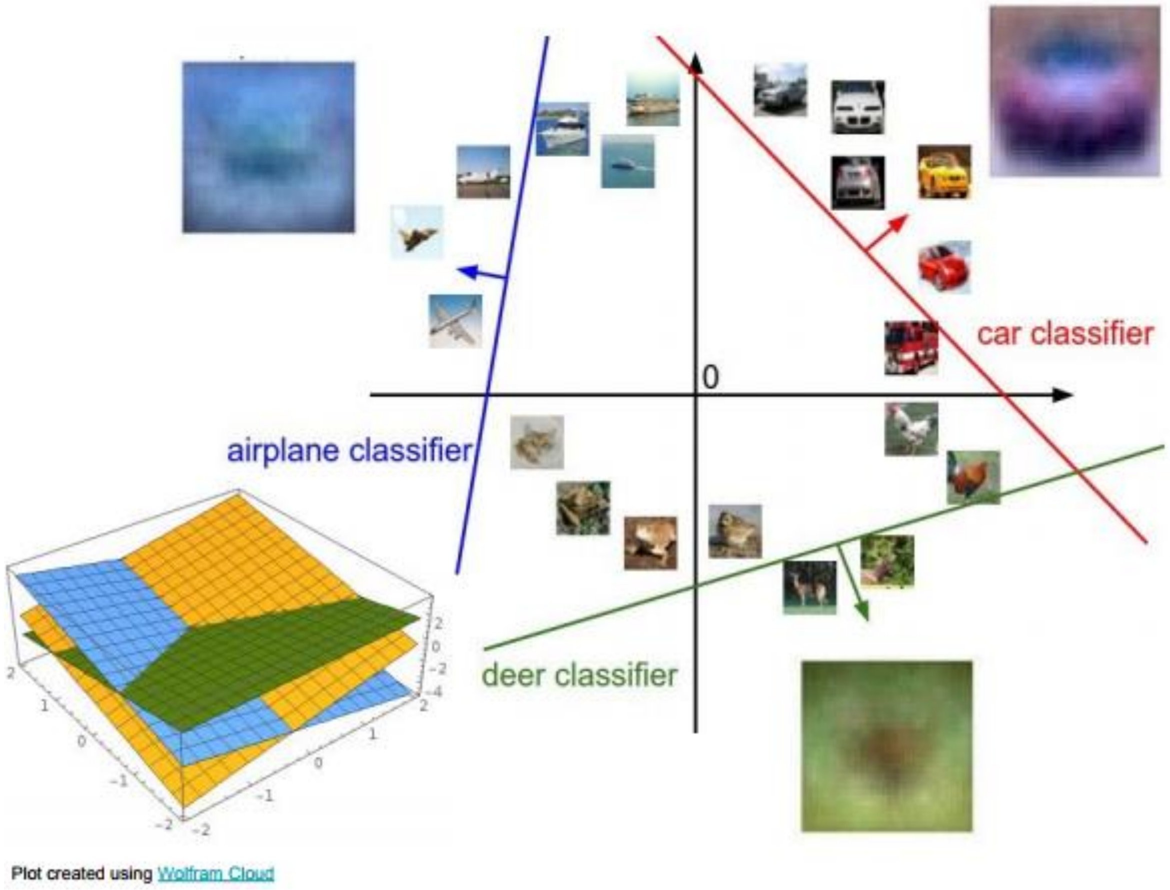
if it trained well, what each row vector means?

Interpreting a Linear Classifier

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

Example trained weights of a linear classifier trained on CIFAR-10:

Interpreting a Linear Classifier

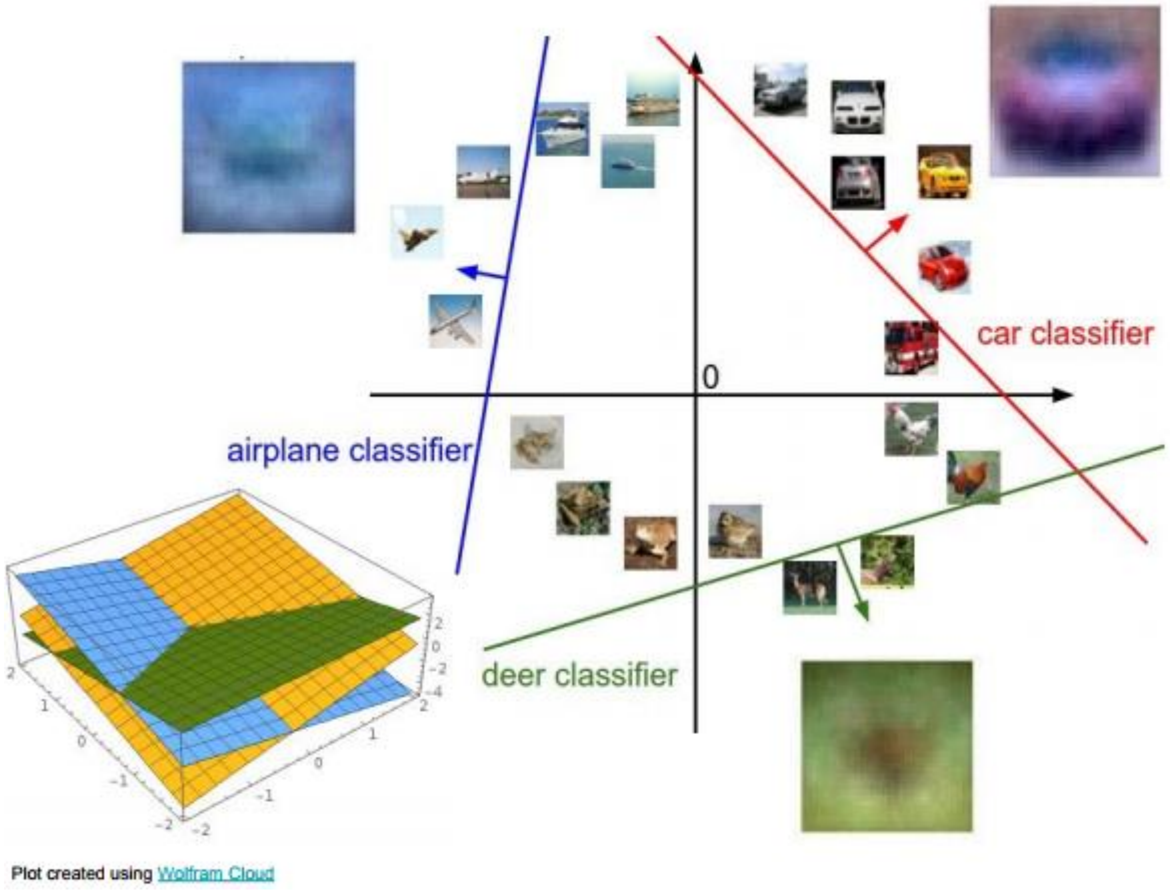


For simple explanation,

assume x vector's size is 2

all images can be located at 2-dim plane

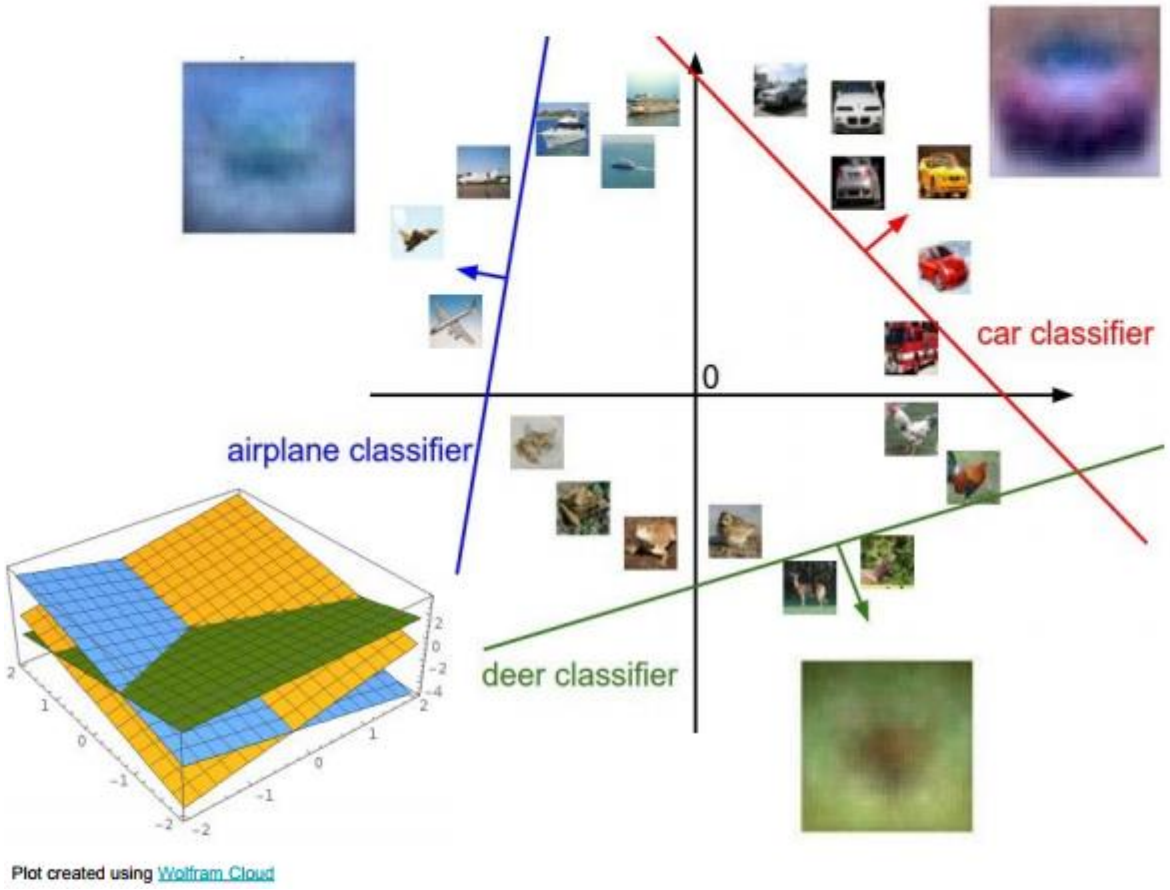
Interpreting a Linear Classifier



Each line is
$W_c x = 0$

where W_c is
each class'
row vector of W

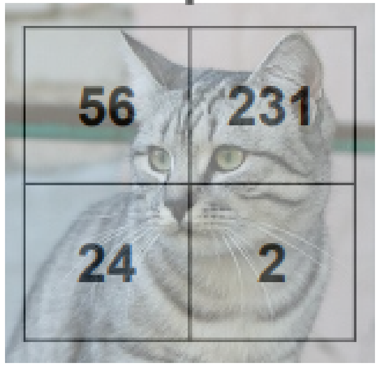
Interpreting a Linear Classifier



Linear Classifier means

the classifier which can classify classes by linear lines

Linear Classifier (Simpler)



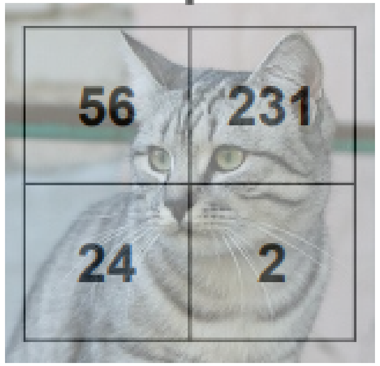
Input Image

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

weights, this should be changed during training

Is it good score vector?

Linear Classifier (Simpler)



Input Image

scores

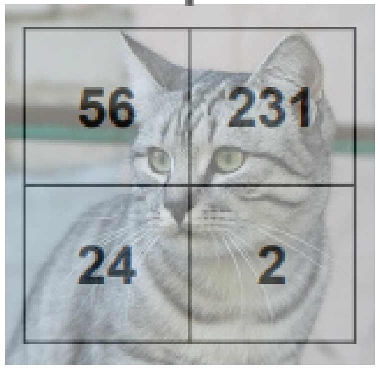
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{array}{cl} \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} & \begin{array}{l}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

weights, this should be changed during training

Is it good score vector?

how to we evaluate score?

Linear Classifier (Simpler)



Input Image

scores

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

weights, this should be changed during training

Is it good score vector?

how to we evaluate score?
loss

Loss Function

loss function:
주어진 scores 와 answer(label)에 대하여
score가 얼마나 정답에 가까운지를 평가하는 함수

loss가 클 수록 잘못된 scores를 의미함!

대표적인 loss function 종류: Softmax, SVM

Loss Function

Softmax Loss

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

?????????

Loss Function

cat	**3.2**
car	5.1
frog	-1.7

scores

Loss Function

$$s_j \rightarrow e^{s_j}\ ,\quad j \text{ is each class}$$

	scores	exponentiation
cat	**3.2**	**24.5**
car	5.1	164.0
frog	-1.7	0.18

Loss Function

$$s_j \rightarrow \frac{e^{s_j}}{\sum_j e^{s_j}} \quad , j \text{ is each class}$$

	scores	exponentiation	normalization
cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

Loss Function

$$s_j \to \frac{e^{s_j}}{\sum_j e^{s_j}} \quad , j \text{ is each class}$$

	scores	exponentiation	normalization	
cat	3.2	24.5	0.13	
car	5.1	164.0	0.87	= probability
frog	-1.7	0.18	0.00	

Loss Function

$$s_j \rightarrow \boxed{\frac{e^{s_j}}{\sum_j e^{s_j}}}$$

softmax function

	scores	exponentiation	normalization
cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

= probability

Loss Function

$$s_j \rightarrow \boxed{\frac{e^{s_j}}{\sum_j e^{s_j}}}$$

softmax function

#one hot encoding

	scores	exponentiation	normalization	label
cat	**3.2**	**24.5**	**0.13**	**1**
car	5.1	164.0	0.87	0
frog	-1.7	0.18	0.00	0

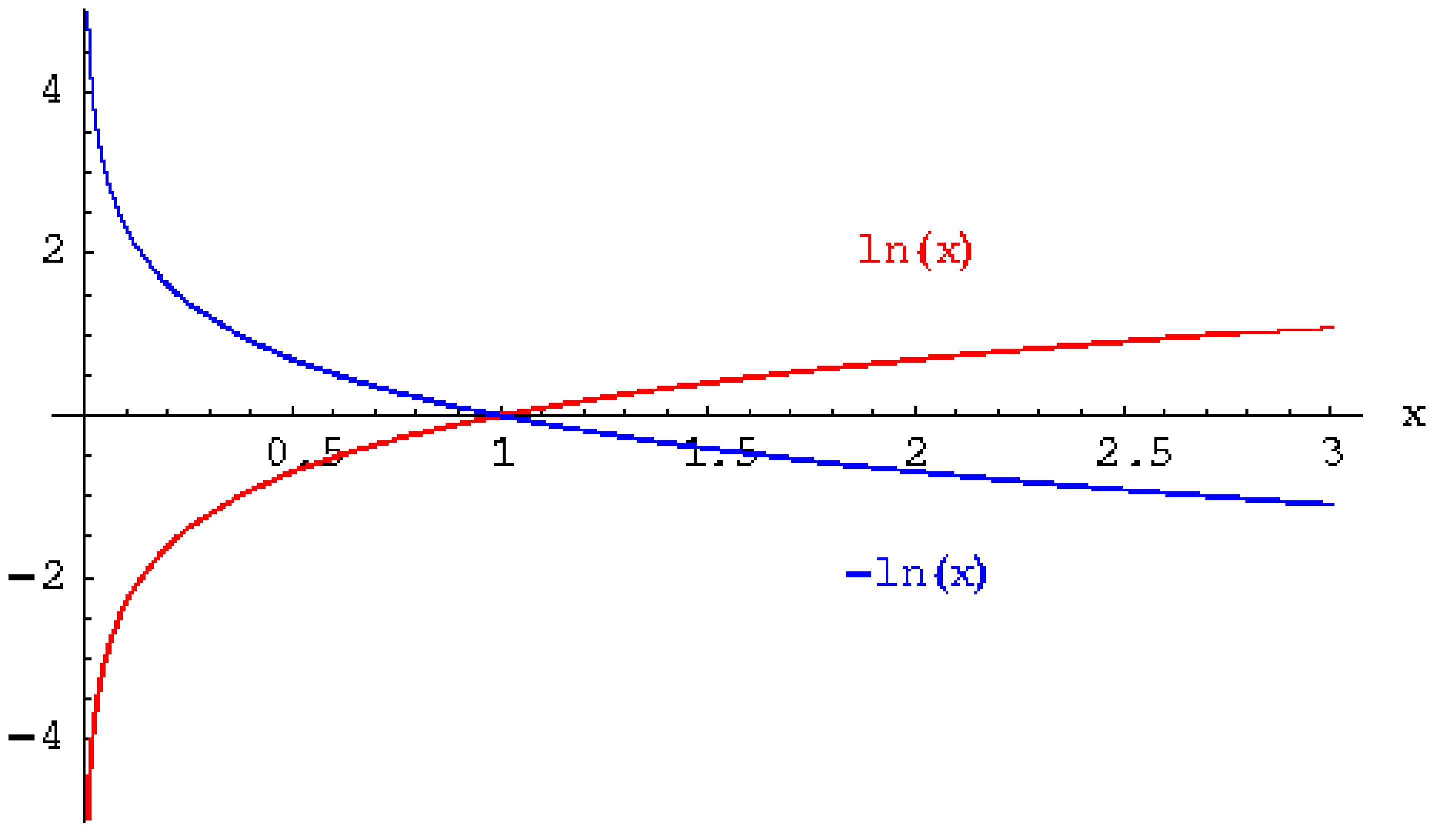
Loss Function

$$s_j \rightarrow \frac{e^{s_j}}{\sum_j e^{s_j}}$$

if cat score goes to 1
loss goes to 0 (low)
if cat score goes to 0
loss goes to inf (high)

	scores	exponentiation	normalization	label
cat	**3.2**	**24.5**	**0.13**	**1**
car	5.1	164.0	0.87	0
frog	-1.7	0.18	0.00	0

Loss Function



Loss Function

$$s_j \rightarrow \frac{e^{s_j}}{\sum_j e^{s_j}}$$

loss
= −log(class score)
= −log(0.13)

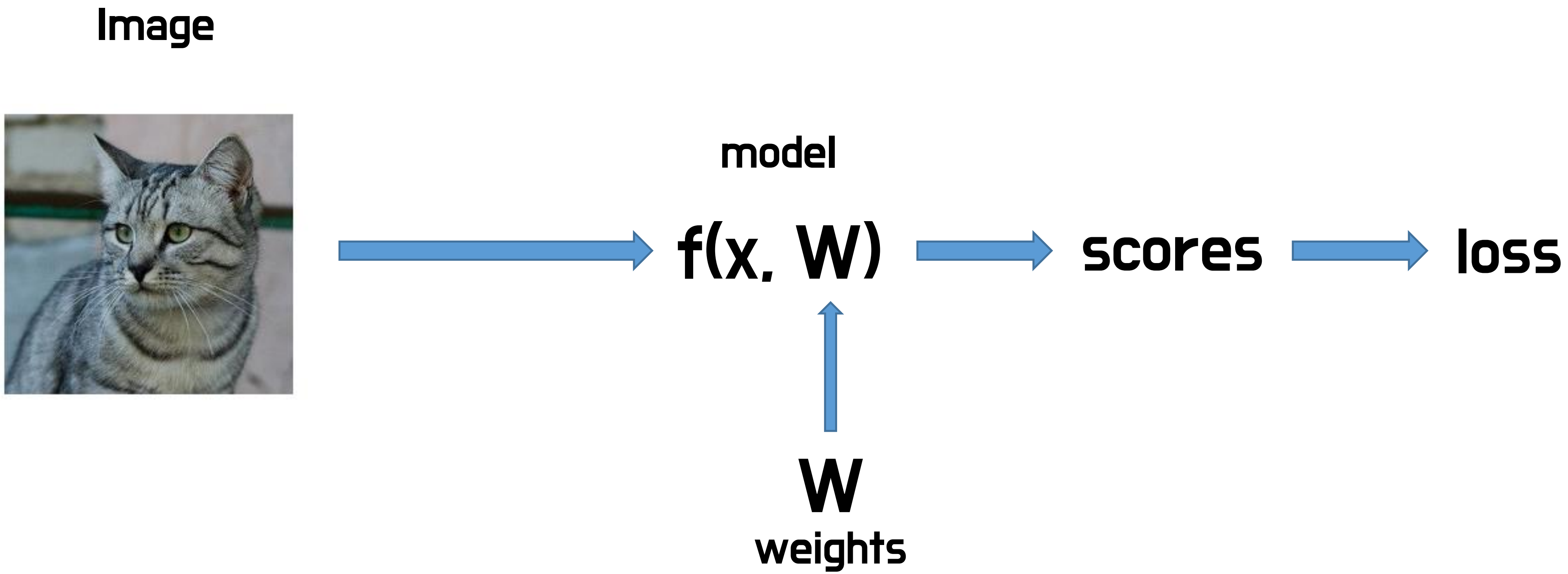
	scores	exponentiation	normalization	label
cat	**3.2**	**24.5**	**0.13**	**1**
car	5.1	164.0	0.87	0
frog	-1.7	0.18	0.00	0

Loss Function

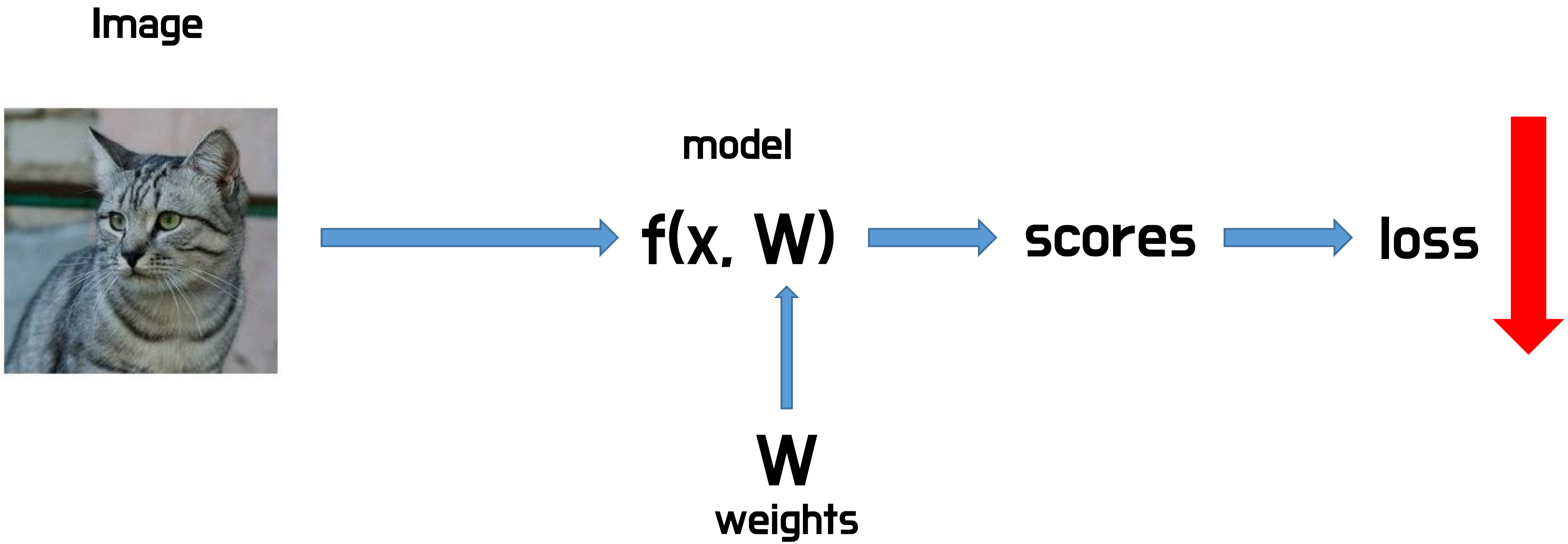
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

	scores	exponentiation	normalization	label
cat	**3.2**	**24.5**	**0.13**	**1**
car	5.1	164.0	0.87	0
frog	-1.7	0.18	0.00	0

Loss Function



How to minimize the loss?



How to minimize the loss?

Image

model

f(x, W) → scores → loss

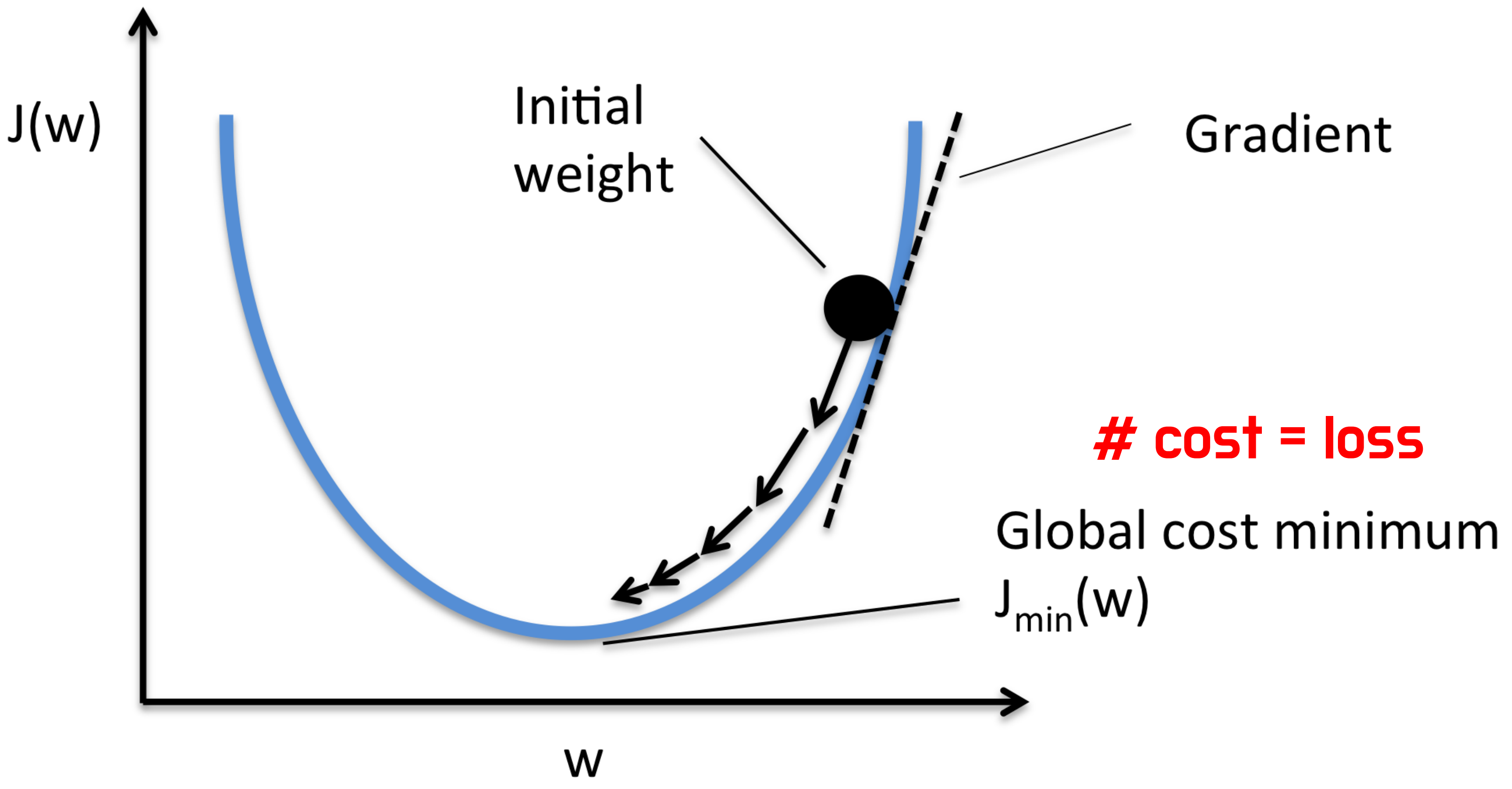
W

weights

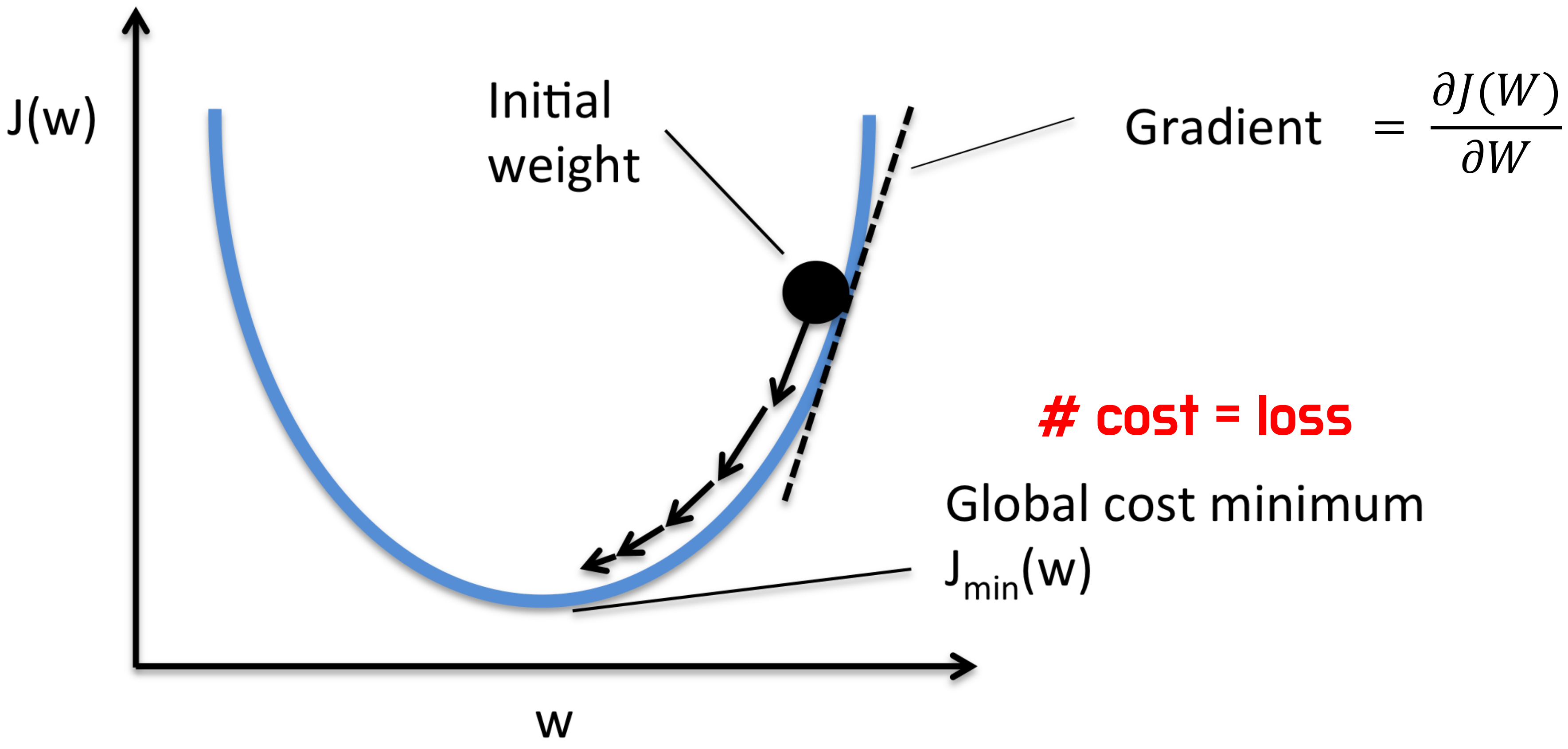
how to find W

minimize the loss?

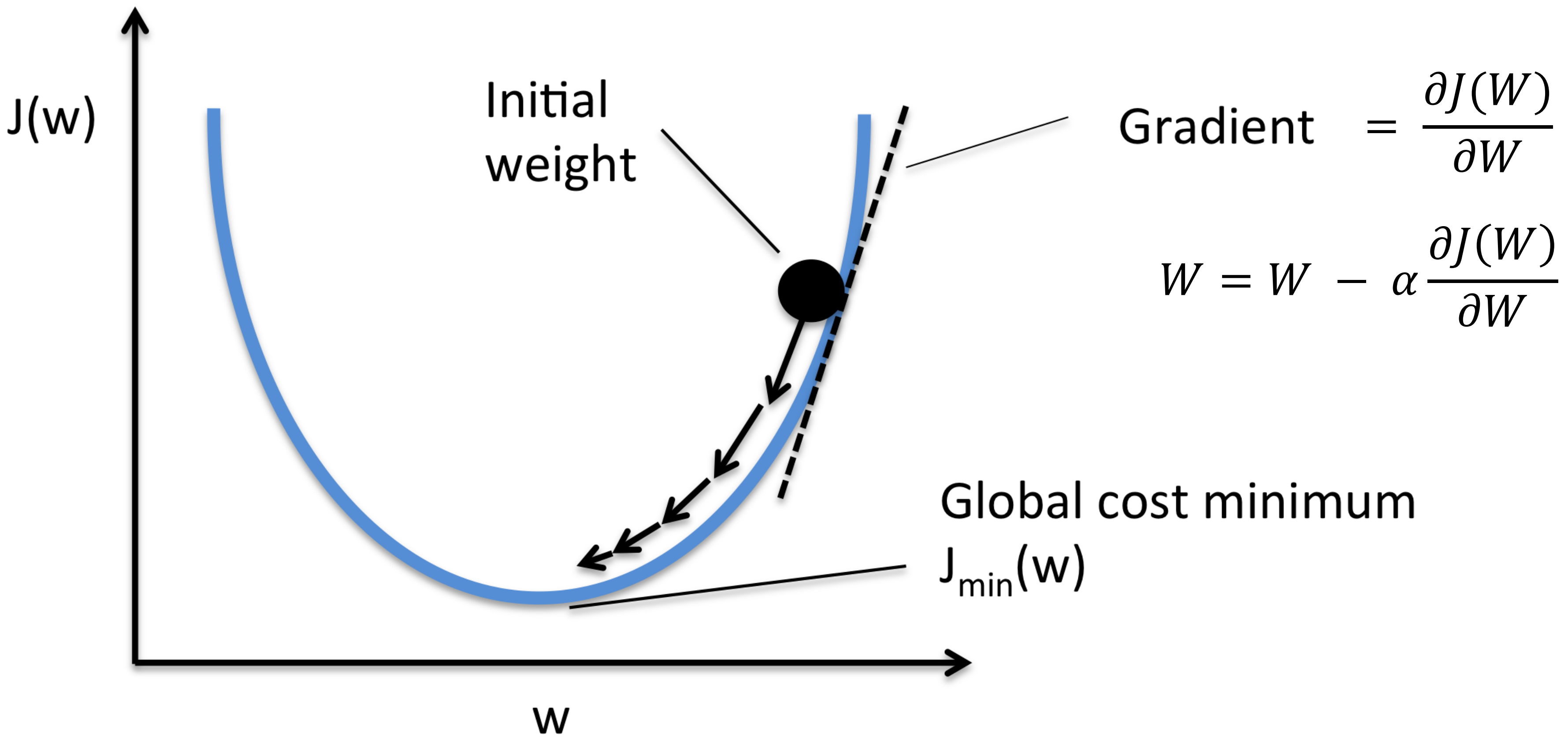
Gradient Descent Algorithm



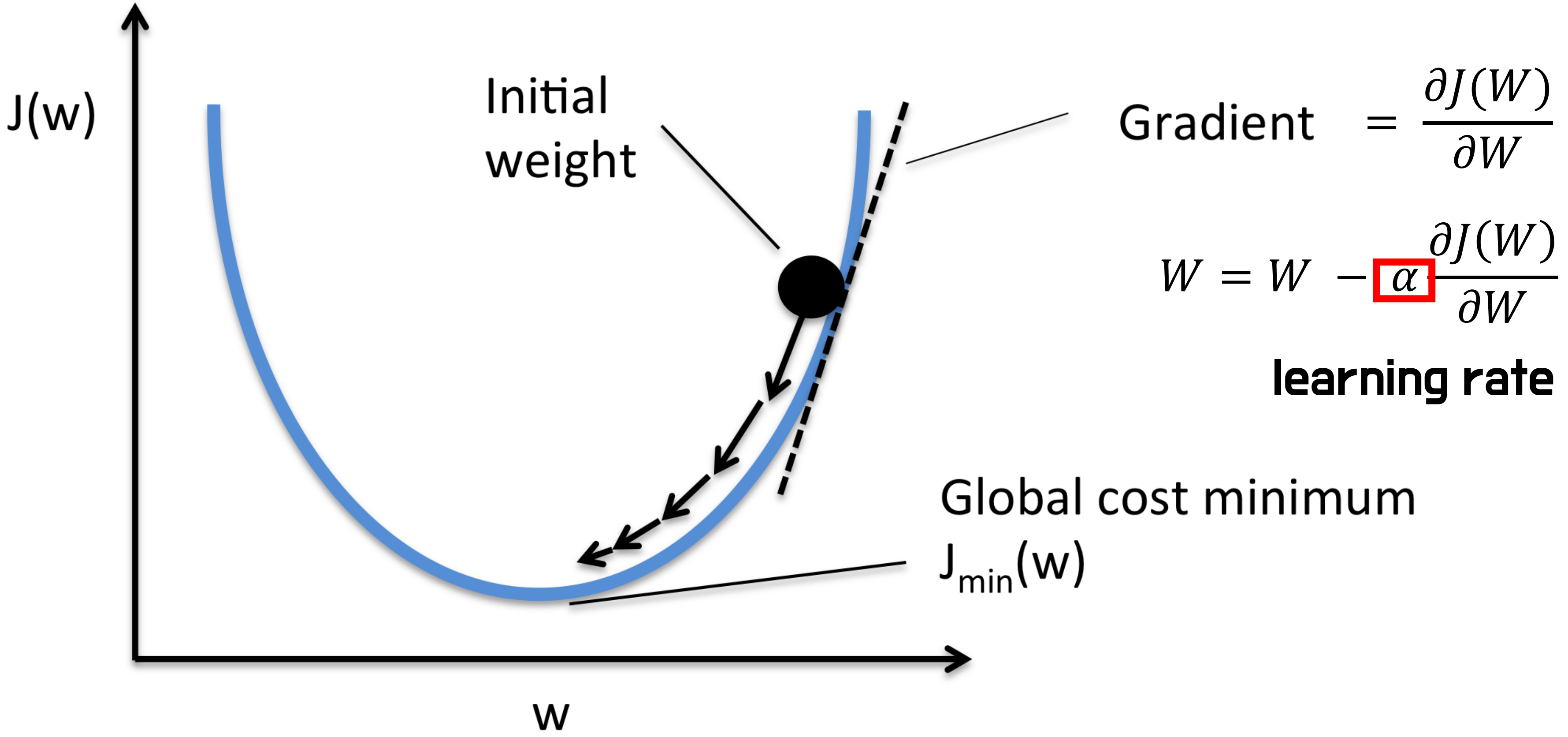
Gradient Descent Algorithm



Gradient Descent Algorithm

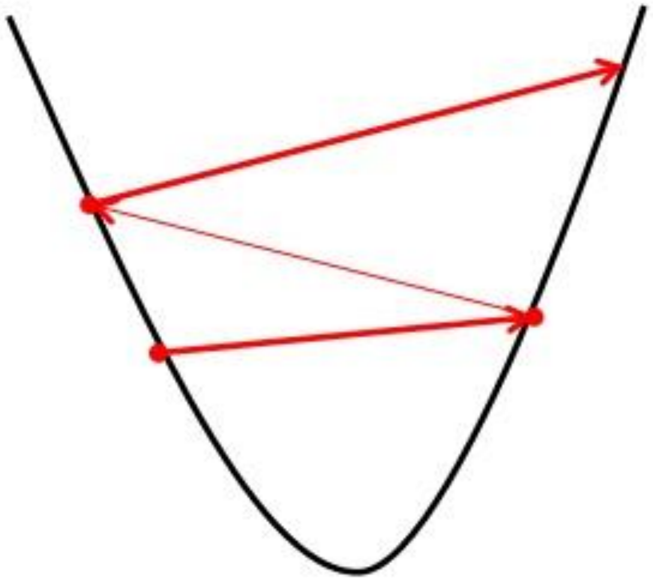


Gradient Descent Algorithm



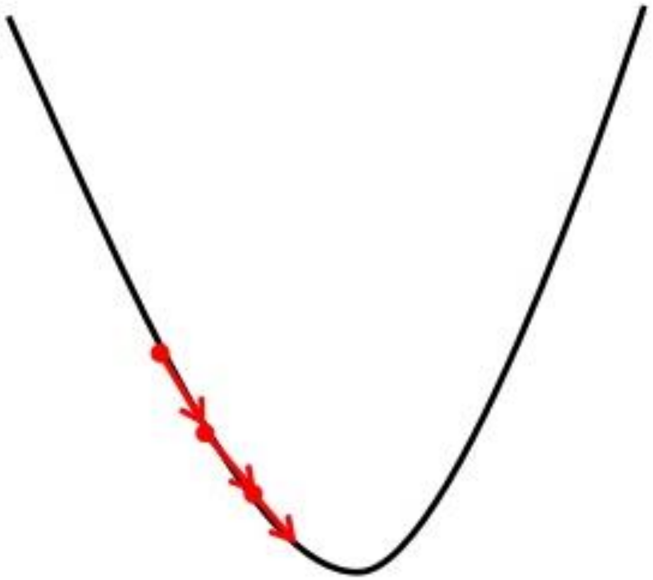
Gradient Descent Algorithm

big learning rate



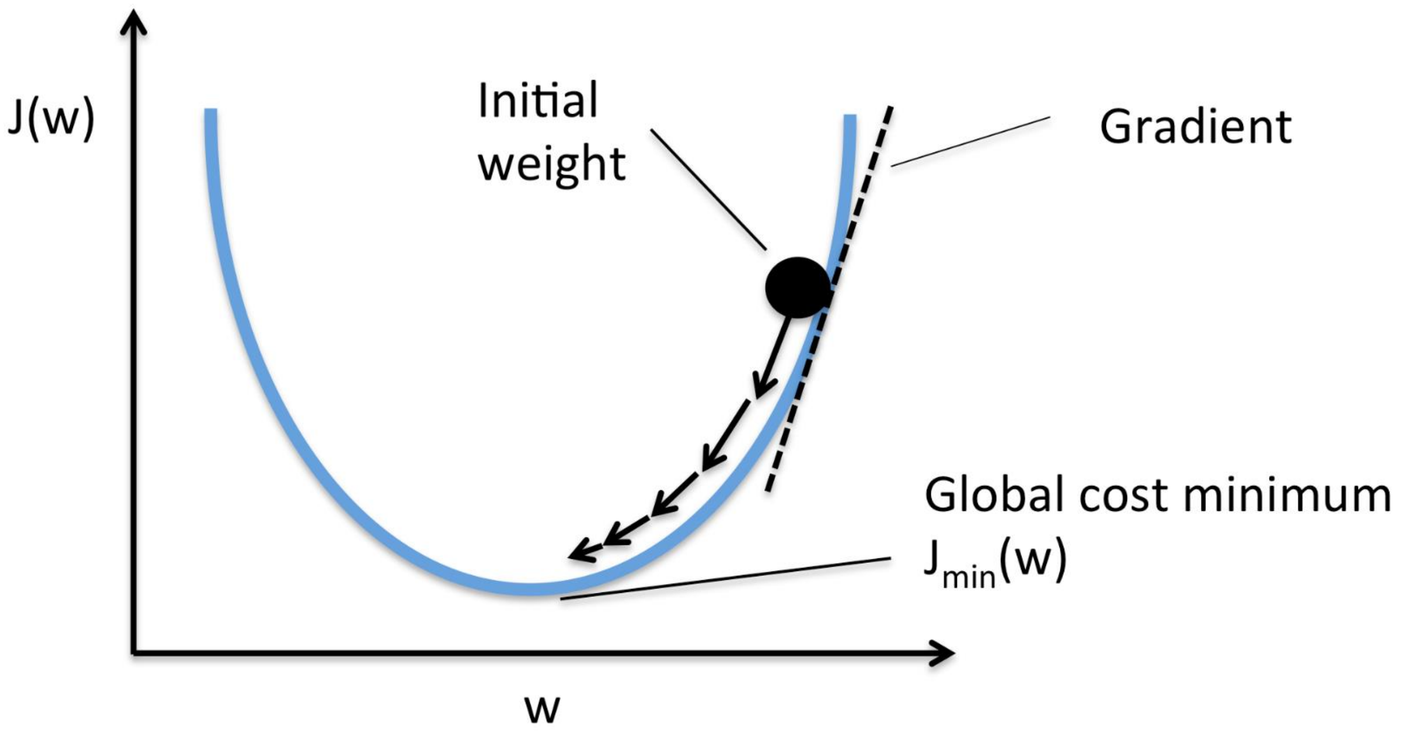
overshooting

small learning rate



too slow learning

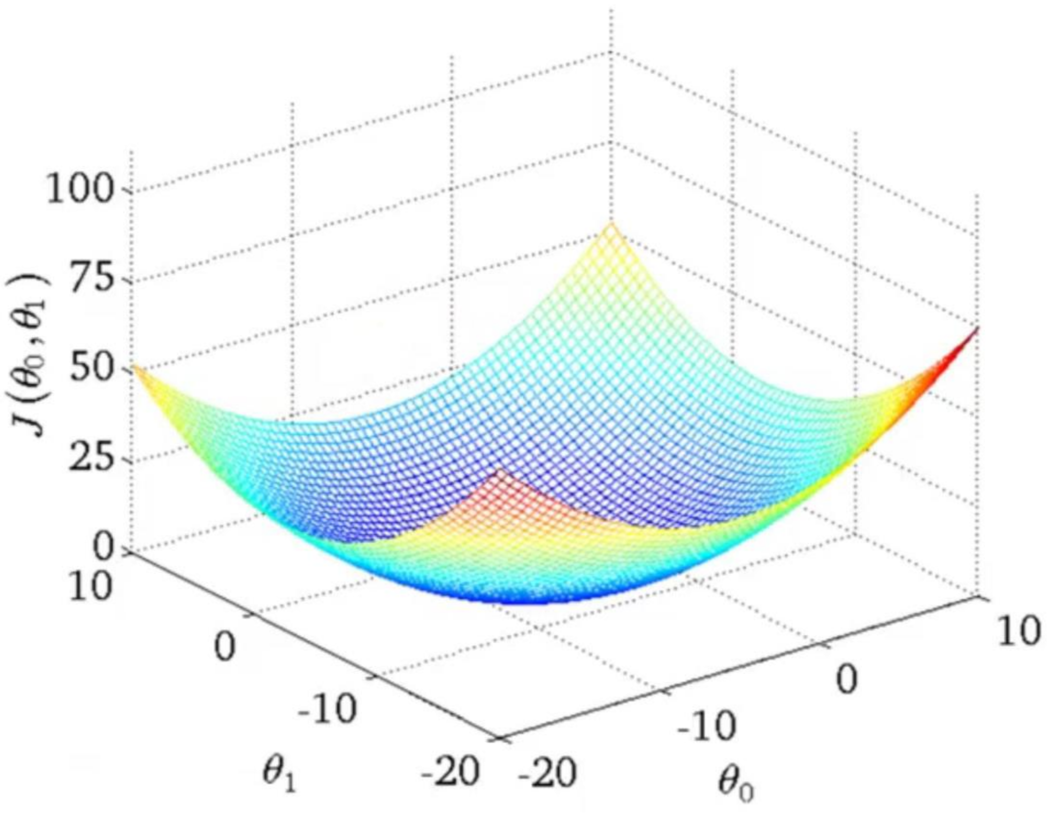
Gradient Descent Algorithm



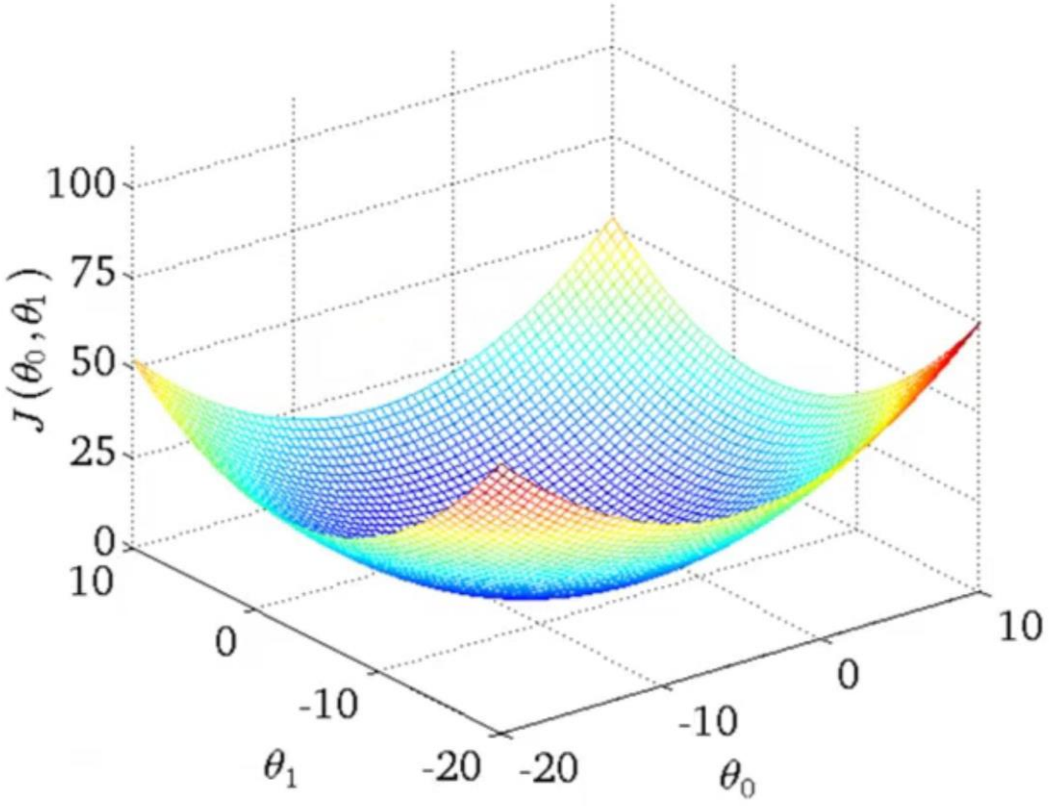
w is not scalar!

if w is 1x2 matrix
we have to change
2 weights

Gradient Descent Algorithm



Gradient Descent Algorithm

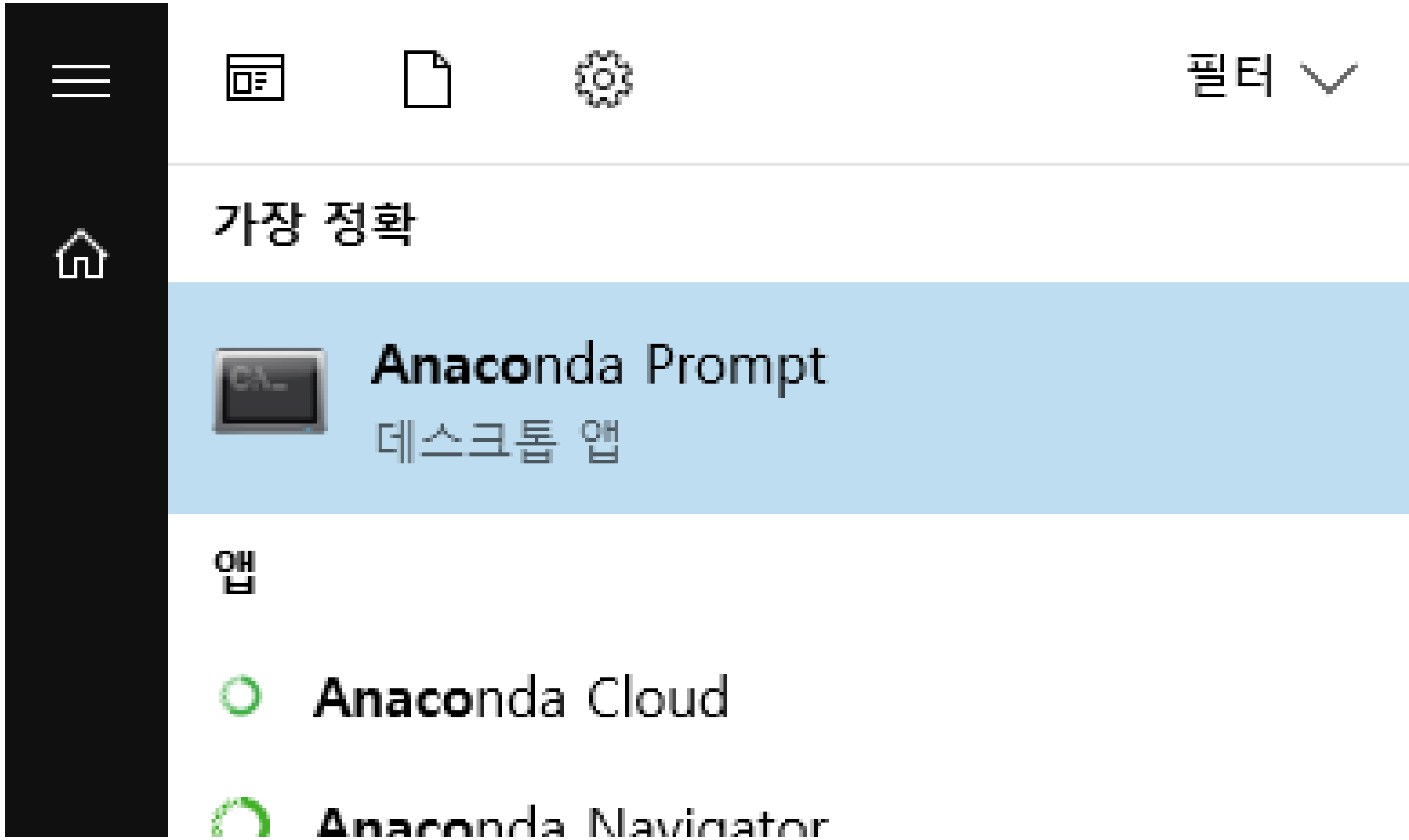


what if w is 6x7 matrix?

we cannot draw ☹

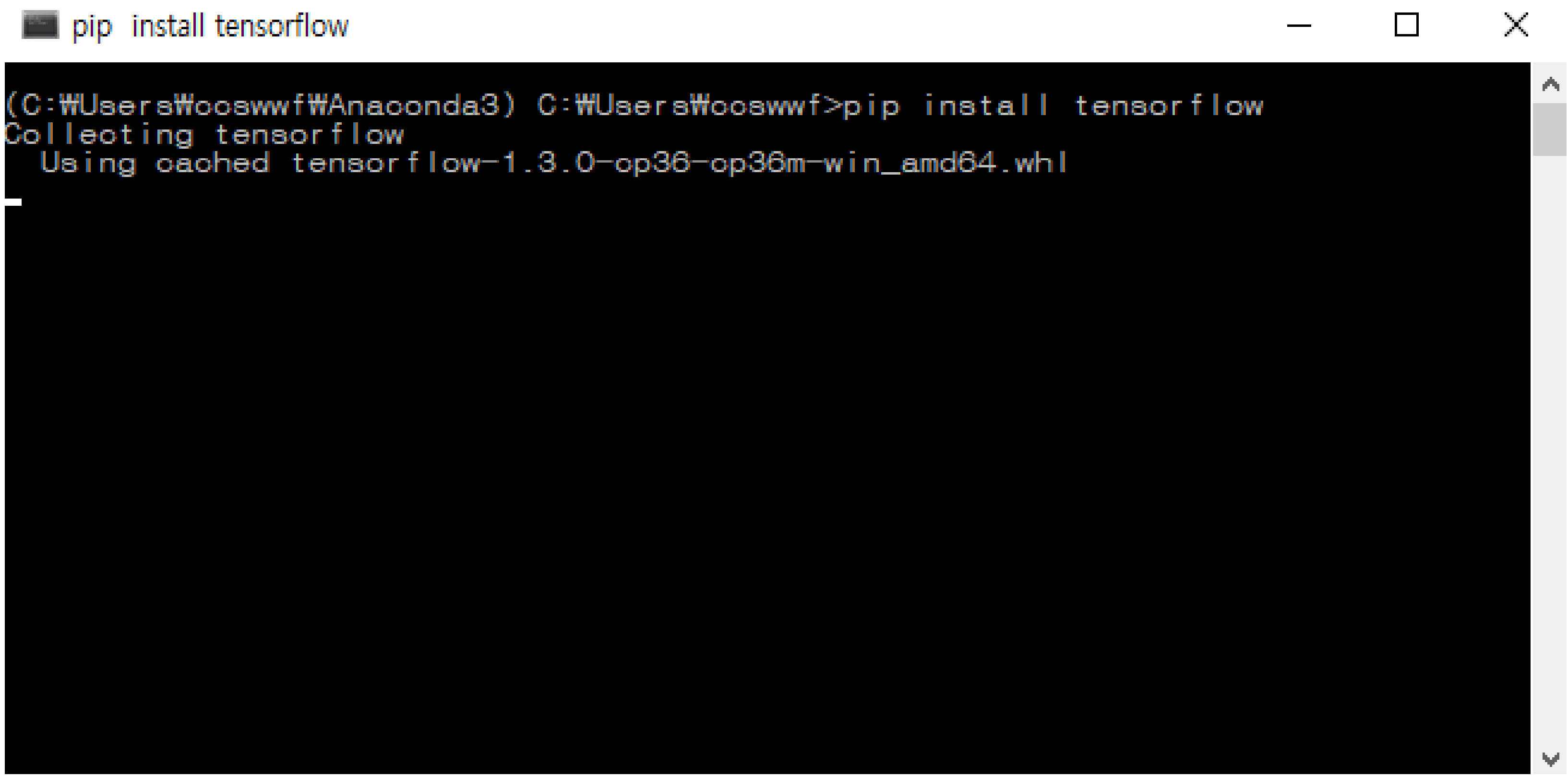
just for visualization,
we assume w has 2 or 1 dim

Tensorflow Installation



Tensorflow Installation

pip install tensorflow



Tensorflow Installation

pip install tensorflow

```
(C:₩Users₩ooswwf₩Anaconda3) C:₩Users₩ooswwf>pip install tensorflow
Collecting tensorflow
  Using cached tensorflow-1.3.0-cp36-cp36m-win_amd64.whl
Requirement already satisfied: numpy>=1.11.0 in c:₩users₩ooswwf₩anaconda3₩lib₩si
te-packages (from tensorflow)
Collecting protobuf>=3.3.0 (from tensorflow)
  Using cached protobuf-3.4.0-py2.py3-none-any.whl
Requirement already satisfied: wheel>=0.26 in c:₩users₩ooswwf₩anaconda3₩lib₩site
-packages (from tensorflow)
Requirement already satisfied: six>=1.10.0 in c:₩users₩ooswwf₩anaconda3₩lib₩site
-packages (from tensorflow)
Collecting tensorflow-tensorboard<0.2.0,>=0.1.0 (from tensorflow)
  Downloading tensorflow_tensorboard-0.1.8-py3-none-any.whl (1.6MB)
    45% |██████████████                  | 737kB 7.2kB/s eta 0:02:05
```

Tensorflow Basic

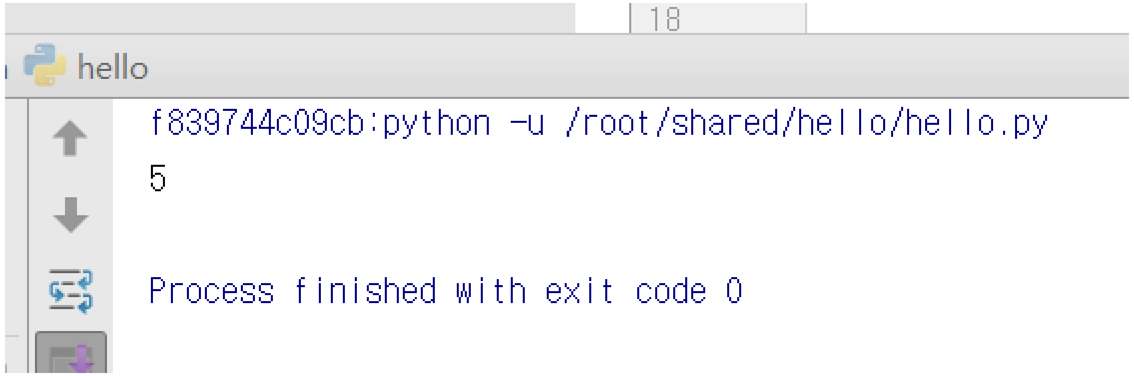
```
import tensorflow as tf

hello = tf.constant('Hello, Tensorflow!')

sess = tf.Session()

print (hello)

print (sess.run(hello))
```

```
Tensor("Const:0", shape=(), dtype=string)
b'Hello, Tensorflow!'

Process finished with exit code 0
```

Tensorflow Basic

v2.7

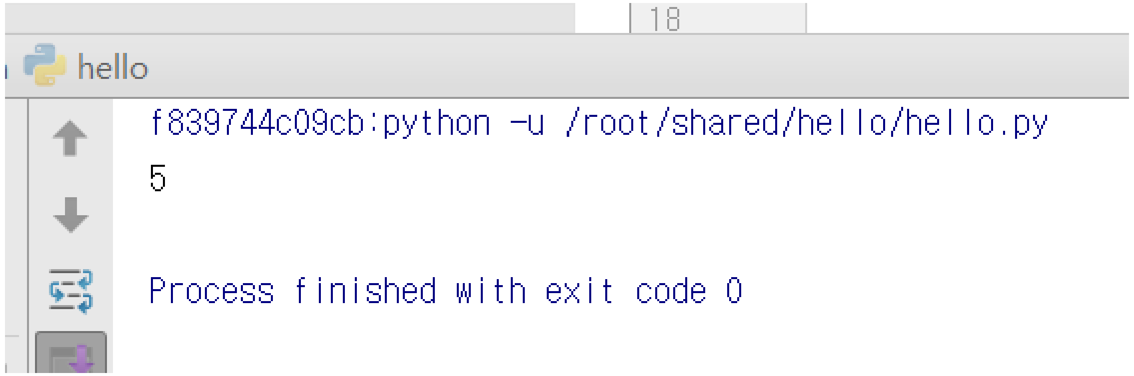
```python
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)

c = a + b

sess = tf.Session()

print sess.run(c)
```



Tensorflow Basic

v2.7

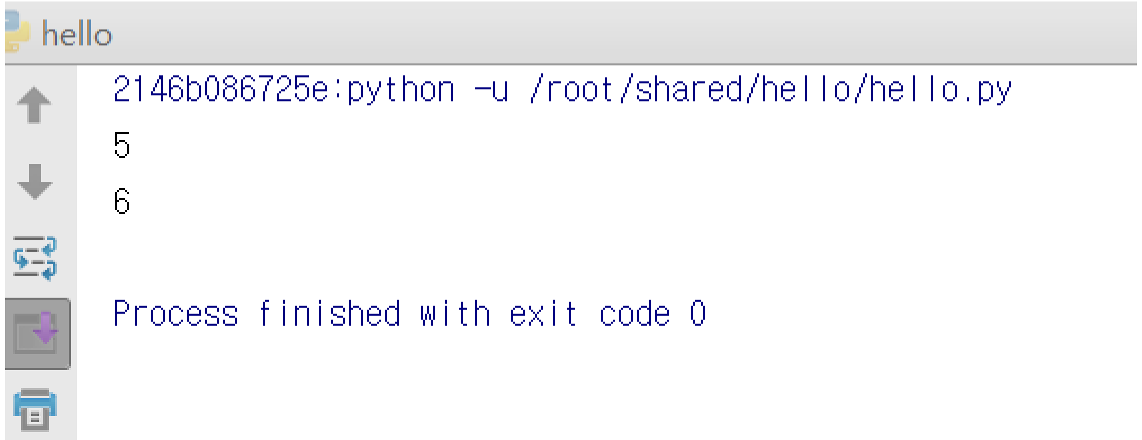
```python
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)

c = tf.add(a, b)

sess = tf.Session()

print sess.run(c)
```



Tensorflow Basic

v2.7

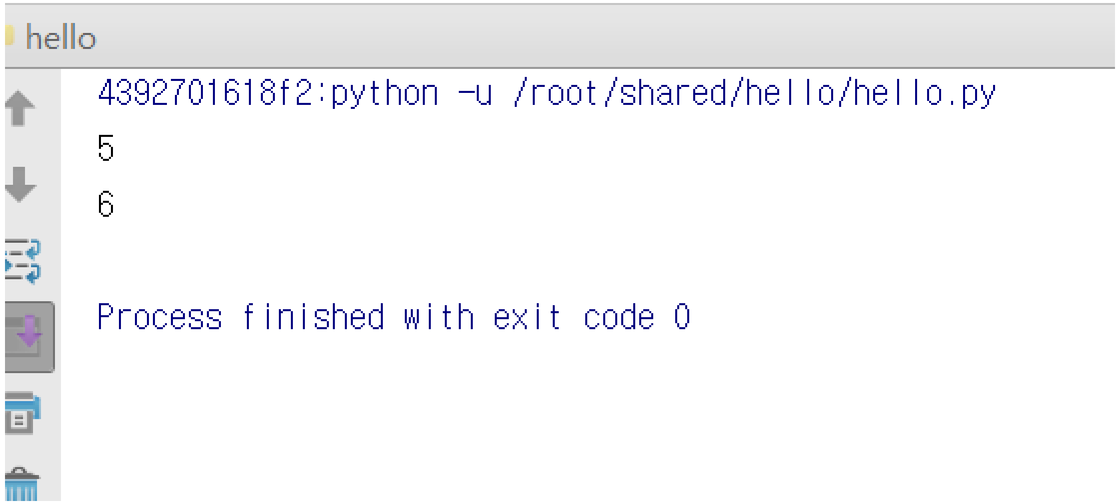
```
import tensorflow as tf

a = tf.constant(2)
b = tf.constant(3)

sess = tf.Session()

print sess.run(a+b)
print sess.run(a*b)
```



Placeholder

v2.7

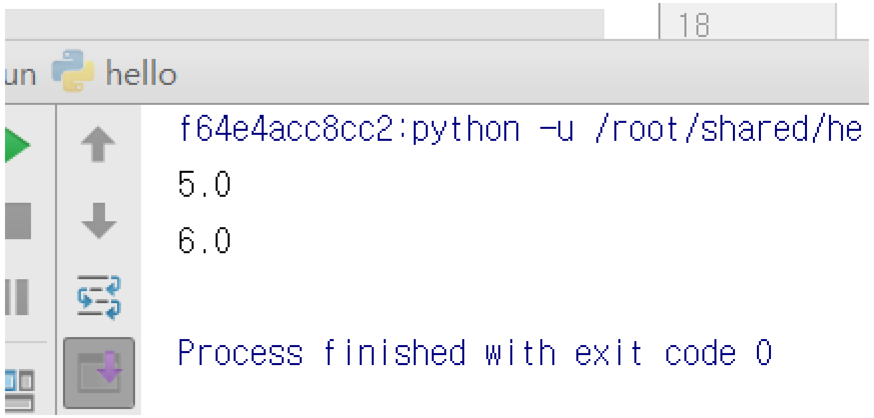
```python
import tensorflow as tf

a = tf.placeholder(tf.int16)
b = tf.placeholder(tf.int16)

add = a + b
mul = a * b

sess = tf.Session()

print sess.run(add, feed_dict={a: 2, b: 3})
print sess.run(mul, feed_dict={a: 2, b: 3})
```



Placeholder

v2.7

```
import tensorflow as tf

a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)

add = a + b
mul = a * b

sess = tf.Session()

print sess.run(add, feed_dict={a: 2.0, b: 3.0})
print sess.run(mul, feed_dict={a: 2.0, b: 3.0})
```

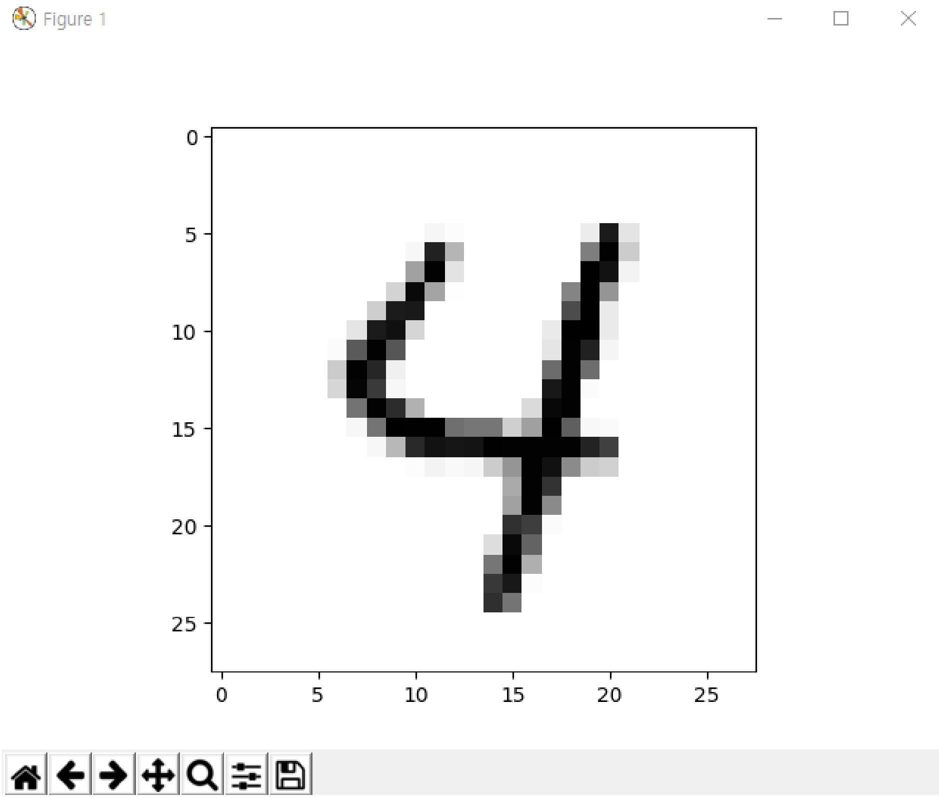


Tensorflow Practice

linear_classifier_mnist.py

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

from random import randint
import matplotlib.pyplot as plt

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

r = randint(0, mnist.test.num_examples - 1)
plt.imshow(mnist.test.images[r:r+1].reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
print(mnist.test.labels[r:r+1])
```

Tensorflow Practice




```
C:₩Users₩ccswwf₩Anaconda3₩envs₩tensorflow₩python.e
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
[[ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]]
```

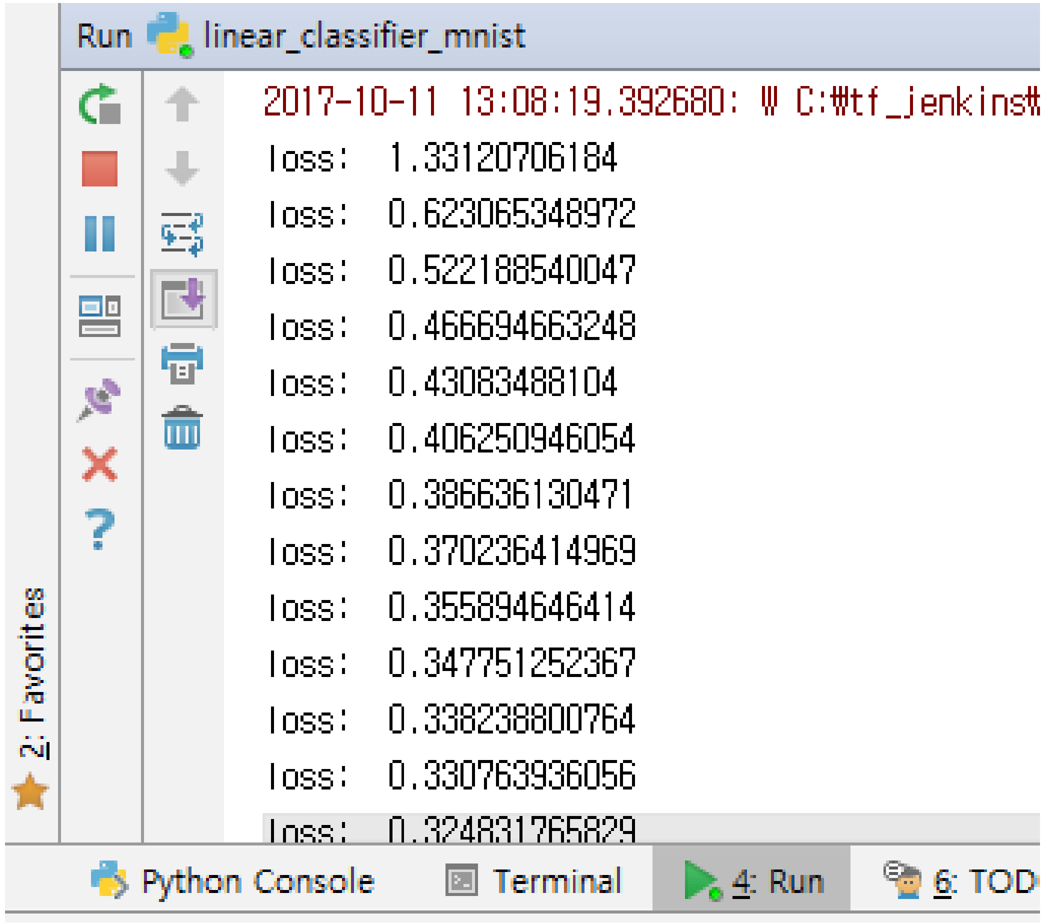
Tensorflow Practice

```
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

W = tf.Variable(tf.random_normal(shape=(784, 10), mean=0.0, stddev=1.0, dtype=tf.float32))
b = tf.Variable(tf.random_normal(shape=(10,)))

scores = tf.matmul(x, W) + b
prob = tf.nn.softmax(scores)

cross_entrophy_loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels = y, logits = scores))

train = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entrophy_loss)
```

Tensorflow Practice

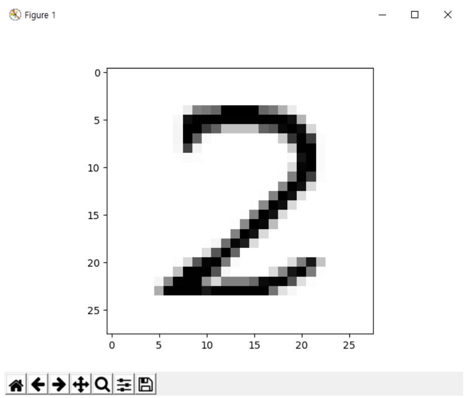
```python
init = tf.global_variables_initializer()

with tf.Session() as sess:

    sess.run(init)

    for epoch in range(30):
        avg_loss = 0.
        for step in range(mnist.train.num_examples // 100):
            batch_x, batch_y = mnist.train.next_batch(100)
            loss, _ = sess.run([cross_entrophy_loss, train], feed_dict={x:batch_x, y:batch_y})
            avg_loss += loss / (mnist.train.num_examples // 100)
        print("loss: ", avg_loss)
```

Tensorflow Practice



Tensorflow Practice

```python
    print("loss: ", avg_loss)


r = randint(0, mnist.test.num_examples - 1)
plt.imshow(mnist.test.images[r:r+1].reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
print("Prediction: ", sess.run(tf.argmax(scores, 1), feed_dict={x: mnist.test.images[r:r+1]}))
```




```
loss:  0.27857595399
loss:  0.276507998928
loss:  0.275508060469
Prediction:  [2]

Process finished with exit code 0
```

Tensorflow Practice

training time vs prediction time ?

How about rule-based (algorithmic) classifier?

At service field, prediction(test) time is much more important!

Deep Learning Library

We can implement this with pure python code.

But.. did we compute softmax function ?
gradient ? optimization ?
all of these processes were done by tensorflow!

And.. There are other similar libraries!

Deep Learning Library

TensorFlow

PYTORCH

Caffe2

Deep Learning Library

Google

facebook

NVIDIA

TensorFlow

PYTORCH

Caffe2

Deep Learning Library

Google

TensorFlow

facebook

NVIDIA

PYTORCH

Caffe2

Question