KAIT 한국정보통신진흥협회
Korea Association for ICT Promotion

이미지/비디오 BM
-컴퓨터 비전과 분류학습-

김 승 환

swkim4610@inha.ac.kr

목차 및 학습 필요사항

1. 컴퓨터 비전과 분류학습
2. 이미지 / 비디오 자료처리
3. 딥러닝
4. 이미지 응용 기술

- 강의는 아나콘다 3.7 파이썬 언어를 사용하고 있음
- 강의에서 사용한 OpenCV 버전은 4.1.0 임
- 프로그램 소스 및 관련 데이터는 아래 URL에서 받을 수 있음

  https://drive.google.com/open?id=1ZwJ0dQSZ0CEgVbSxsu7cLh8eNAUE1QUN

1. 컴퓨터 비전과 분류학습

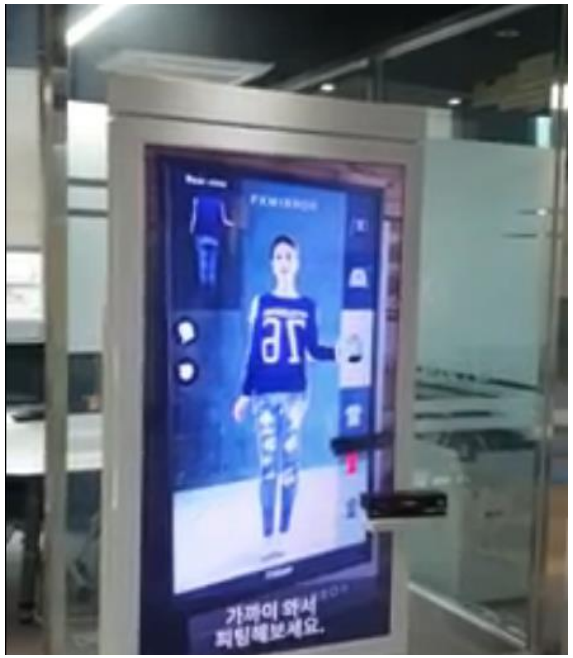
1.1 컴퓨터 비전

1.2 특징 추출

1.3 분류 학습 원리

1. 1 컴퓨터 비전

https://drive.google.com/open?id=1j6rJwmAh53kPy6lehvk6V56_8pu5648G



https://drive.google.com/open?id=1wm6gv9c4EbQJhkM0cEW9fiSpkC_qVqq7

1. 1 컴퓨터 비전

https://youtu.be/Zam-2u7zT0c

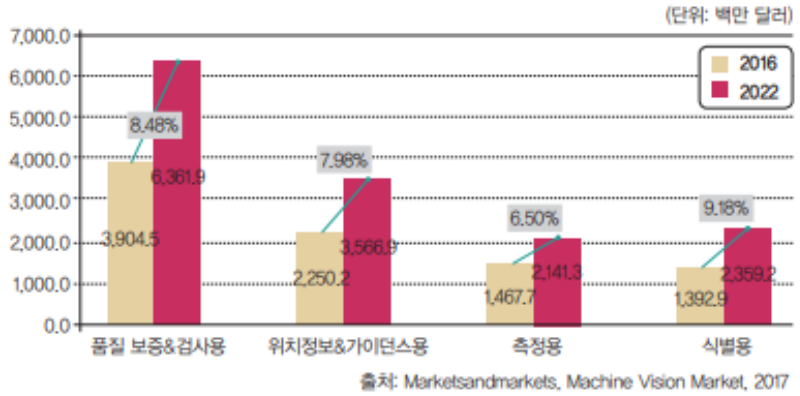
1. 1 컴퓨터 비전

- 비전관련 산업은 사람의 눈과 머리로 할 수 있는 모든 분야가 대상임
- 2012년 이후, 컴퓨터 비전을 이용한 기술의 인간의 능력을 뛰어 넘는 결과를 보여줌
- 비전 관련 BM은 폭발적으로 증가하고 있음

아시아-태평양 지역 전체 머신비전 시장 32% 점유

전 세계 머신비전 시장 2022년까지
연평균 8.15% 증가, 약 145억 달러 규모 전망

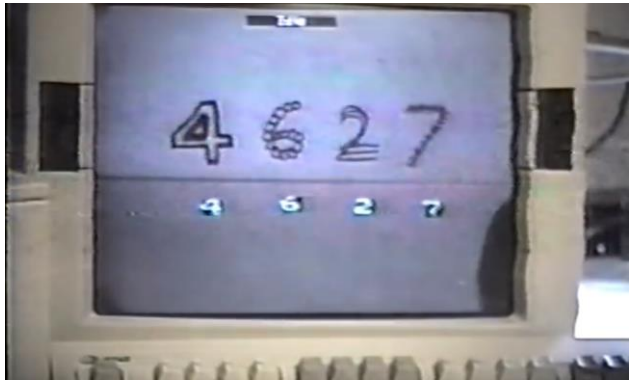
시장조사업체 마켓앤마켓(Marketsandmarkets)이 발표한 전 세계 머신비전 시장 전망에 따르면 2022년까지 연평균 성장률 8.15%로 증가해 144억 3,000만 달러에 이를 것으로 추정된다. 마켓앤마켓이 발표한 시장 전망 자료를 토대로 머신비전 시장을 분야별로 소개한다.



출처: Marketsandmarkets, Machine Vision Market, 2017

1. 1 컴퓨터 비전

- 컴퓨터 비전은 인공지능 이전부터 연구되어 온 분야임
- 과거에는 Rule based Algorithm을 탑재한 컴퓨터 비전 제품들이 많았으나, 최근 딥러닝의 획기적인 발전에 힘입어 엄청난 성장을 한 분야임
- 1989년 Yann LeCun은 우편번호를 인식하는 LeNet-5 모형을 소개하여 Rule Base가 아닌 컴퓨터 스스로 학습을 통해 구현되는 방법을 제안했으나 많은 문제점을 노출하면서 사용되지 않았음
- 그 문제가 아주 유명한 Gradient Vanishing problem 임
- 2006년 Hinton 교수에 의해 이 문제가 해결되고, 빅데이터 시대가 되면서 과거 이슈가 해결될 수 있는 상황이 만들어 지면서 현재 컴퓨터 비전은 이미 인간의 한계를 뛰어 넘은 영역임



https://youtu.be/FwFduRA_L6Q

https://drive.google.com/open?id=1DONXzi5SmnWMN5lWaIf4LjXoBAQTmCeH

t2000은 실존 인물일지도 모른다

1. 1 컴퓨터 비전

- 사람은 고양이를 인식하고 차선을 인식할 수 있음
- 서로 다른 사진에서 에펠탑도 인식함





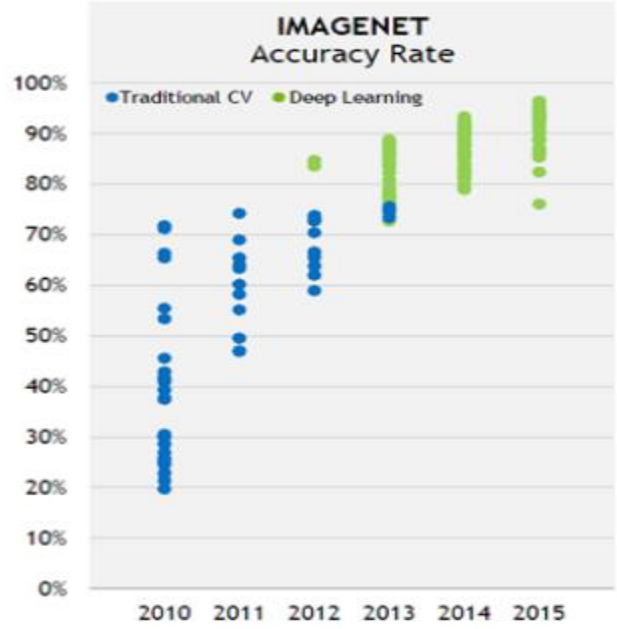
1. 1 컴퓨터 비전

- 컴퓨터가 인식을 잘 하기 위해 데이터 전처리가 필수적임
- 이러한 전처리를 위해서는 이미지 자료를 직접적으로 변환하는 과정이 필요함

▪ preprocessing

1. 1 컴퓨터 비전

- 2012년 IMAGENET 이미지 분류대회에서 힌튼 교수의 제자 알렉스가 알렉스넷(AlexNet)이라는 딥러닝 기반 알고리즘으로 84.7%의 정확도로 우승하면서 딥러닝이 세상의 주목을 받음
- 현재, 이 데이터셋의 인식률은 95%를 상회하고 있고 이는 인간보다 뛰어난 인식률임
- 갑자기 무슨 일이 일어난 것일까?



1. 2 특징 추출

- 과거, 컴퓨터 비전은 이미지의 특징점을 추출하는 방식으로 이미지의 일부 정보만 사용
- 정보량이 작아서 같은 특징점이 유사한데 완전 다른 이미지도 존재함
- 통계학의 모수절약의 법칙에 의해 Feature 최소화
- 현재는 딥러닝은 픽셀 자체를 정보로 활용→ Feature 최대화(?)
- CNN으로 특징추출 자동화를 통해 정확도를 획기적으로 개선함

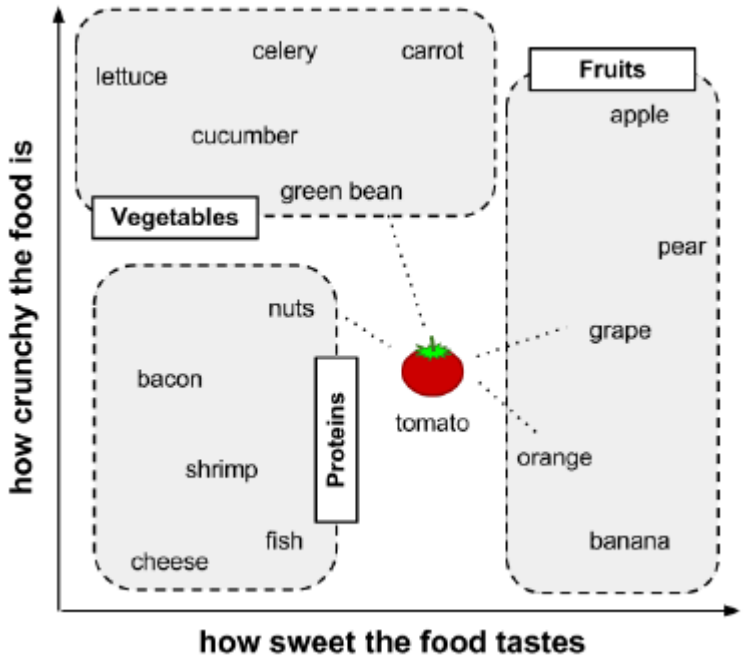
1. 2 특징 추출

- 아래는 알파벳 26글자를 학습하기 위해 만든 Feature의 예임
  과거에는 이미지를 서로 구분할 수 있는 핵심적 특징을 수치로 표현하고 이 수치를 분포 차이를 이용하여 이미지를 인식하는 방법을 사용함. 이 방법은 훌륭하지만, Feature를 만들어 내는 것이 고통스러운 작업임

1.	lettr	capital letter	(26 values from A to Z)
2.	x-box	horizontal position of box	(integer)
3.	y-box	vertical position of box	(integer)
4.	width	width of box	(integer)
5.	high	height of box	(integer)
6.	onpix	total # on pixels	(integer)
7.	x-bar	mean x of on pixels in box	(integer)
8.	y-bar	mean y of on pixels in box	(integer)
9.	x2bar	mean x variance	(integer)
10.	y2bar	mean y variance	(integer)
11.	xybar	mean x y correlation	(integer)
12.	x2ybr	mean of x * x * y	(integer)
13.	xy2br	mean of x * y * y	(integer)
14.	x-ege	mean edge count left to right	(integer)
15.	xegvy	correlation of x-ege with y	(integer)
16.	y-ege	mean edge count bottom to top	(integer)
17.	yegvx	correlation of y-ege with x	(integer)

1. 3 분류학습 이해

- 아래는 통계적 분류기 중 가장 기본적인 kNN 모형을 설명하는 그림임
- Recognition은 유사성을 측정하는 것임(유사성 = 거리 = 확률)



ingredient	sweetness	crunchiness	food type	distance to the tomato
grape	8	5	fruit	*sqrt((6 - 8)^2 + (4 - 5)^2) = 2.2*
green bean	3	7	vegetable	*sqrt((6 - 3)^2 + (4 - 7)^2) = 4.2*
nuts	3	6	protein	*sqrt((6 - 3)^2 + (4 - 6)^2) = 3.6*
orange	7	3	fruit	*sqrt((6 - 7)^2 + (4 - 3)^2) = 1.4*

1. 3 분류학습 이해

- kNN은 거리에 의해 유사도를 측정하기 때문에 모든 입력변수는 양적변수이고 Scale을 통일해야 함
- 이러한 조건이 위배되면 부정확한 결과를 줄 수 있으므로 주의해야 함

The traditional method of rescaling features for kNN is **min-max normalization**. This process transforms a feature such that all of its values fall in a range between 0 and 1. The formula for normalizing a feature is as follows. Essentially, the formula subtracts the minimum of feature X from each value and divides by the range of X:

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Normalized feature values can be interpreted as indicating how far, from 0 percent to 100 percent, the original value fell along the range between the original minimum and maximum.

Another common transformation is called **z-score standardization**. The following formula subtracts the mean value of feature X and divides by the standard deviation of X:

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$

1. 3 분류학습 이해

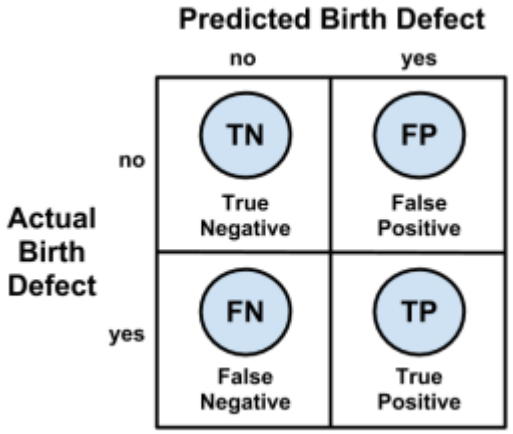
- kNN 알고리즘으로 숫자를 판별해 보자. kNN.py

```
from sklearn.datasets import load_digits
from matplotlib import pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

digits = load_digits()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

kNN = KNeighborsClassifier(n_neighbors=3)
kNN.fit(X_train, y_train)
pred = kNN.predict(X_test)

print(metrics.confusion_matrix(y_test, pred))
```

1. 3 분류학습 이해

- accuracy, error rate는 분류기 성능의 기본적인 측도임
- imbalanced Data의 경우, accuracy는 적절치 못함
- accuracy는 높은데 recall(실제 기형아일 경우, 맞추는 비율) 이 낮을 수 있음
- recall을 높이고자 Positive로 판정하는 비율을 높이면 precision이 낮아지게 됨
- 그러한 이유로 F-measure를 많이 사용함

Confusion Matrix



$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{error rate} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \text{accuracy}$$

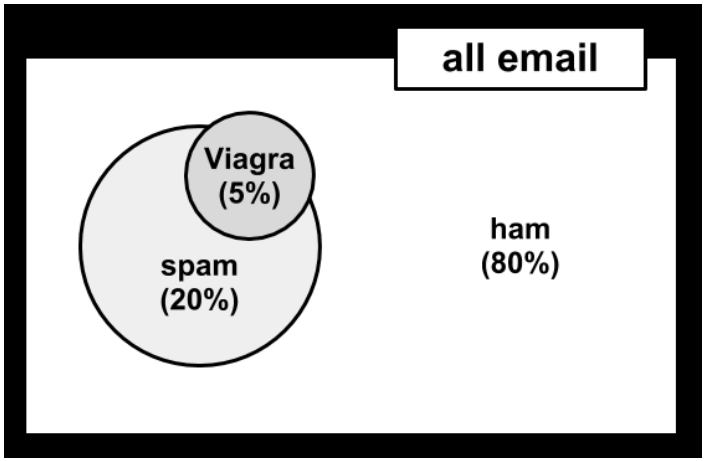
$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{recall} + \text{precision}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

1. 3 분류학습 이해

- Bayes 분류기를 설명하는 그림임



$$P(\text{spam} \mid \text{Viagra}) = \frac{P(\text{Viagra} \mid \text{spam})\,P(\text{spam})}{P(\text{Viagra})}$$

likelihood
prior probability
posterior probability
marginal likelihood

1. 3 분류학습 이해

- 이메일 수신시, “비아그라”, “수신거부” 단어가 있을 경우, 스팸일 확률은 아래와 같음

Using Bayes' theorem, we can define the problem as shown in the following formula, which captures the probability that a message is spam, given that Viagra = Yes, Money = No, Groceries = No, and Unsubscribe = Yes:

$$P(\text{Spam} \mid W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 \mid \text{spam})\,P(\text{spam})}{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)}$$

- $P(W_1 \cap !W_2 \cap !W_3 \cap W_4|Spam)$ 확률 계산에서 W_i들의 수가 많으면 스팸 중 해당 경우의 Case가 작아 확률 계산이 부정확해 짐
- 적은 데이터와 정확한 계산 vs 많은 데이터와 근사계산 중 후자를 선택함(Naive Bayes Algorithm)
- 아래의 식에서 분모는 계산하지 않아도 됨

$$= \frac{P(W_1|spam)P(!W_2|spam)P(!W_3|spam)P(W_4|spam)P(spam)}{P(W_1 \cap !W_2 \cap !W_3 \cap W_4|spam)P(spam) + P(W_1 \cap !W_2 \cap !W_3 \cap W_4|ham)P(ham)}$$

1. 3 분류학습 이해

- “비아그라”, “수신거부” 단어가 있을 경우, 나이브 베이즈 알고리즘으로 계산한 스팸일 확률은 아래와 같음

	Viagra (W_1)		Money (W_2)		Groceries (W_3)		Unsubscribe (W_4)		
Likelihood	Yes	No	Yes	No	Yes	No	Yes	No	Total
spam	4 / 20	16 / 20	10 / 20	10 / 20	0 / 20	20 / 20	12 / 20	8 / 20	20
ham	1 / 80	79 / 80	14 / 80	66 / 80	8 / 80	71 / 80	23 / 80	57 / 80	80
Total	5 / 100	95 / 100	24 / 100	76 / 100	8 / 100	91 / 100	35 / 100	65 / 100	100

The overall likelihood of spam

(4/20) * (10/20) * (20/20) * (12/20) * (20/100) = 0.012

likelihood of ham

(1/80) * (66/80) * (71/80) * (23/80) * (80/100) = 0.002

The probability of spam is equal to the likelihood that the message is spam divided by the likelihood that the message is either spam or ham: 0.012 / (0.012 + 0.002) = 0.857

1. 3 분류학습 이해

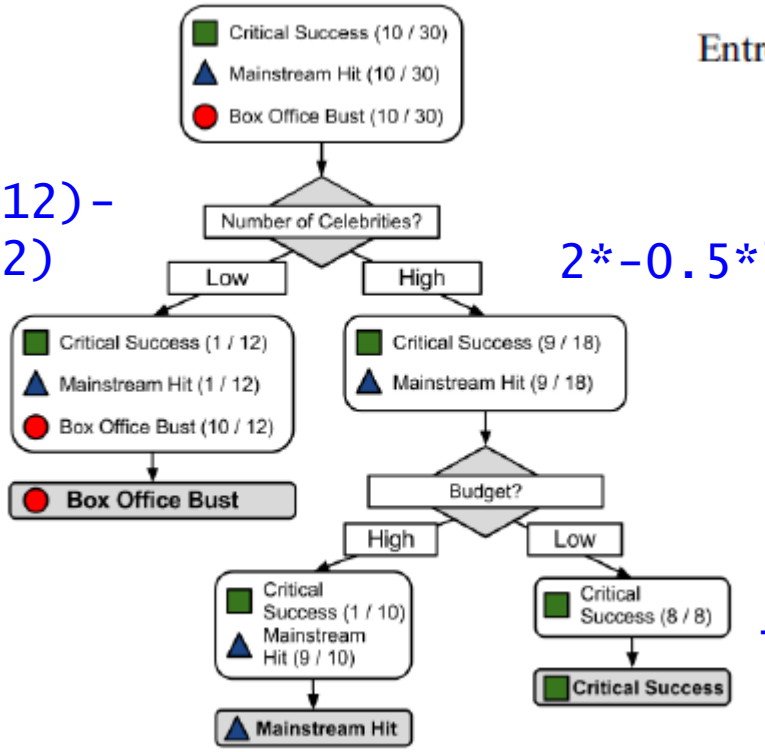
NaiveBayes.py

```
from sklearn.datasets import load_digits
from matplotlib import pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

digits = load_digits()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
prob = clf.predict_proba(X_test)

print(metrics.confusion_matrix(y_test, pred))
```

1. 3 분류학습 이해

- 아래는 Decision Tree 기법을 설명하는 그림임
- Decision Tree는 엔트로피 등을 사용하여 불확실성을 최소화되는 방향으로 나무가지를 생성함

3*-0.33*log2(0.33)=1.58



$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

2*-1/12*log2(1/12)-10/12*log2(10/12)=0.82

2*-0.5*log2(0.5)=1

-1*log2(1)=0

-0.1*log2(0.1)-0.9*log2(0.9)=0.47

1. 3 분류학습 이해

DecisionTree.py

```
from sklearn.datasets import load_digits
from matplotlib import pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

digits = load_digits()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

model = DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)
pred = model.predict(X_test)
print(metrics.confusion_matrix(y_test, pred))
matches = (pred == y_test)
print(matches.sum() / len(matches))
```

1. 3 분류학습 이해

```
import numpy as np

def boostrap(x): #Boostrap Sampling
    n = len(x)
    samp = set()
    for i in range(n):
        samp.add(np.random.choice(x))
    return samp

x = np.array(list(range(1,101)))
samp = boostrap(x)
print(len(samp))
```

특정수가 뽑힐 확률 = 1- 특정수가 안 뽑힐 확률

= 1- (1-뽑힐 확률)

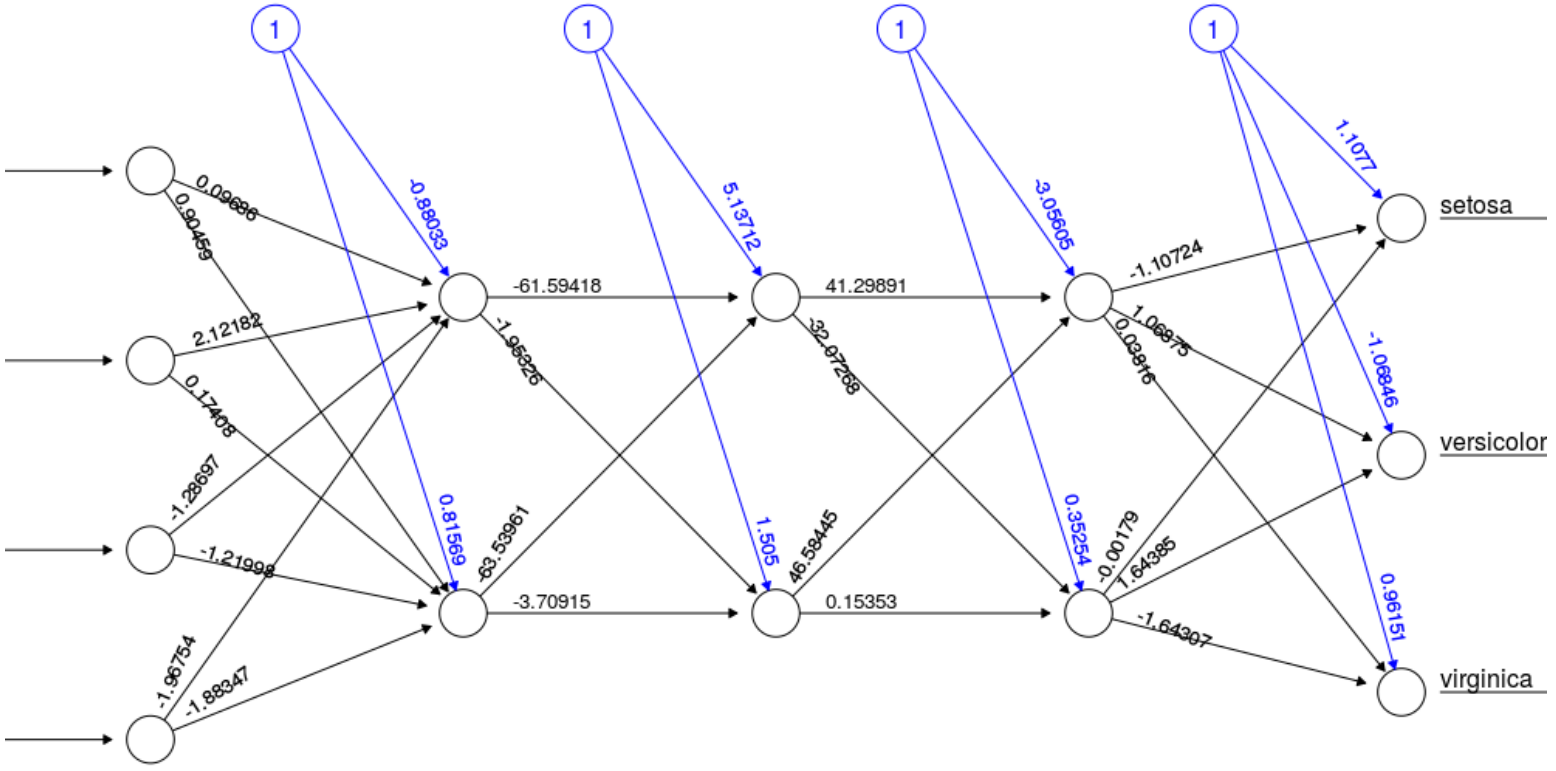
= 1- (100번 연속 안 뽑힐 확률)

$$Pr(i \in \text{Bootstrap Sample}) = 1 - 0.99^{100} = 0.634$$

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target)
model = RandomForestClassifier()
model.fit(X_train, y_train)
pred = model.predict(X_test)
matches = (pred == y_test)
print(matches.sum() / len(matches))
```

1. 3 분류학습 이해

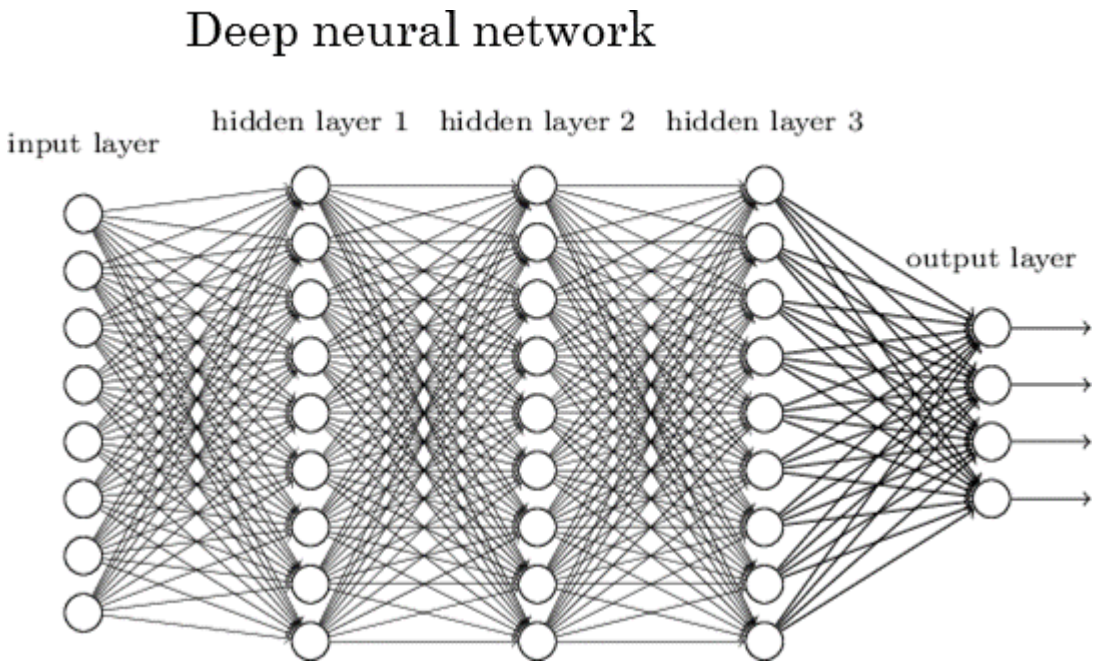
- 아래는 신경망 모형의 예임
- 아래 모형에서 선에 있는 가중값을 계산하는 것이 목표임
- 원하는 출력을 얻기 위해 “어떤 가중값을 가져야 하는가?” Backpropagation 알고리즘을 사용함



Error: 0.001274 Steps: 972

1. 3 분류학습 이해

- 신경망 모형을 deep 하게 만들어 정확도를 높이는 방향으로 진화하고 있음
- 딥러닝 모형의 경우, 데이터도 많이 필요하고, 동시에 컴퓨팅 자원도 많이 필요해짐
- 최근, 컴퓨터 비전과 결합하여 괄목할 만한 성과를 내고 있음



감사합니다