

강의계획표

주	해당 장	주제
1	1장 2장, 3장	머신러닝이란
2		머신러닝을 위한 기초지식, 구현을 위한 도구
3	4장	선형 회귀로 이해하는 지도학습
4	5장	분류와 군집화로 이해하는 지도 학습과 비지도 학습
5	6장	다양한 머신러닝 기법들 - 다항 회귀, Logistic Regression - 정보이론, 결정트리 - SVM, Ensemble
6		
7	7장	인공 신경망 기초 - 문제와 돌파구
8		중간고사 (04-20)
9	8장	고급 인공 신경망 구현
10	9장	신경망 부흥의 시작, 합성곱 신경망
11	10장	순환 신경망
12	11장	차원축소와 매니폴드 학습
13	12장	오토인코더와 잠재표현 학습
14	13장	AI의현재와미래(GAN, 챗GPT)
15		보강주
16		기말고사

7장 인공 신경망 기초

- 문제와 돌파구

7 주차

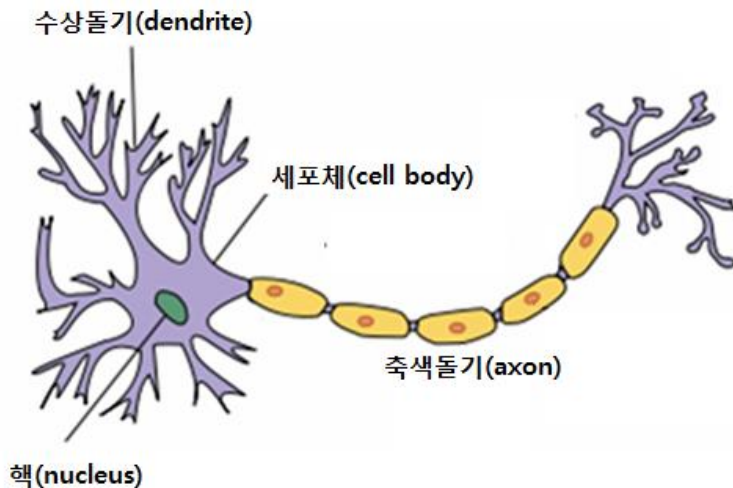
이장에서 배울 것들

- 연결주의가 무엇이고 어떤 목표를 갖고 있는지 살펴 보자.
- 각광받던 신경망 모델들이 관심 밖으로 밀려난 이유는 무엇일까.
- 신경망을 복잡하게 구성하는 일은 다시 어떤 벽을 만나게 되었을까.
- 오차를 이용하여 신경망의 가중치를 개선하기 위한 획기적인 방법은 무엇일까.

Neural Network (1)

❖ 신경망(neural network, artificial neural network)

- 인간 두뇌에 대한 계산적 모델을 통해 인공지능을 구현하려는 분야
- **연결주의(connectionism)** : 신경세포의 동작을 흉내 내는 장치나 소프트웨어를 만들어 뇌가 수행하는 인지나 사고 능력을 갖춘 인공지능 연구 방식
- **신경세포 (neuron)**



신경세포 8.6×10^{10} 개

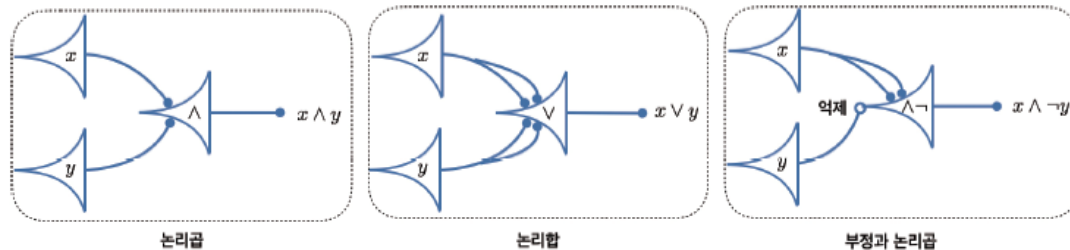
신경연접 1.5×10^{14} 개

- 수상돌기(樹狀突起, dendrite) : 다른 신경세포의 축색돌기와 연결되어 **전기화학적 신호**를 받아들이는 부위
- 축색돌기(軸索突起, axon) : 수신한 전기화학적 신호의 합성결과 값이 특정 임계값이 이상이면 신호를 내보내는 부위.
- 신경연접(神經連接, synapse) : 수상돌기와 축색돌기 연결 부위
 - 전달되는 신호의 증폭 또는 감쇄

Neural Network (2)

❖ 간략한 역사

- 1943, McCulloch, Pitts 최초 신경망 제안(신경세포의 신호 전달 방식을 모방)

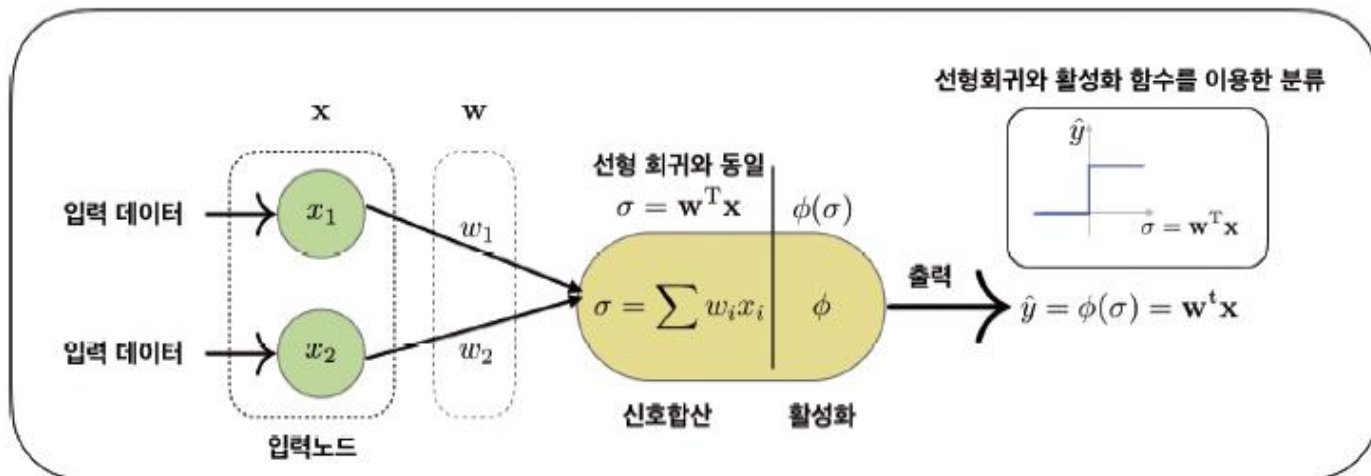


- 1949, Hebb의 학습 알고리즘
- 1958, Rosenblatt 퍼셉트론(perceptron)
- 1960, Widrow와 Hoff, Adaline과 Madaline, LMS(최소평균제곱)
- 1969, Minsky와 Papert, Perceptrons라는 저서에서 퍼셉트론 한계 지적
 - 퍼셉트론은 선형 분류기로 XOR도 해결 못함, 이후 신경망 연구 퇴조
- 1986, Rumelhart, Hinton, Williams, 다층 퍼셉트론과 오류 역전파 학습 알고리즘
- 2013, Geoffrey Hinton, Deep Learning

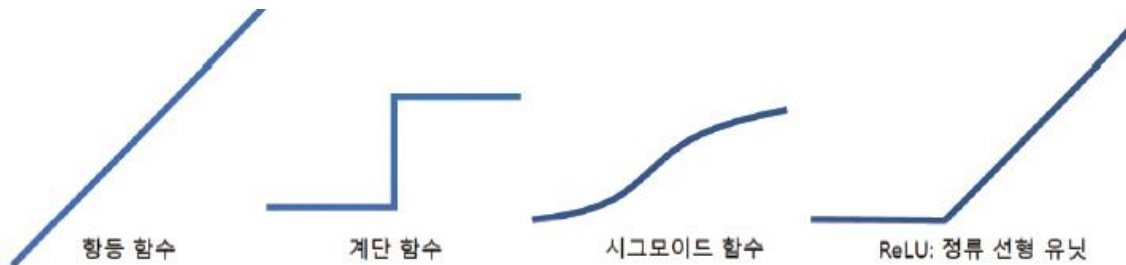
Perceptron (1)

❖ 퍼셉트론(perceptron)

- 1957년에 로젠블라트(Frank Rosenblatt)
- 연결강도^{weight} w_i 를 학습



▪ 활성화 함수



Perceptron (2)

❖ 학습

- 경험을 통해 기능이나, 성능이 바뀌는 것
- **연결강도** weight w_i 학습: 활성화 함수의 출력과 정답을 비교하여 오차를 줄이도록 **연결강도**를 바꾸는 과정
- **파라미터** parameter: **연결강도** weight w_i
- **최적화** optimization: 오차를 줄이는 방향으로 연결강도를 조정하는 일 (경사 하강법, 미분 가능 함수)
- **하이퍼파라미터** hyperparameter: 학습과정을 조절하는 인자(연결강도 조정 정도/학습률, 최적화 방법, 학습 반복 횟수 등)

Perceptron (3)

❖ LAB⁷⁻¹ AND / OR 연산을 수행하는 퍼셉트론

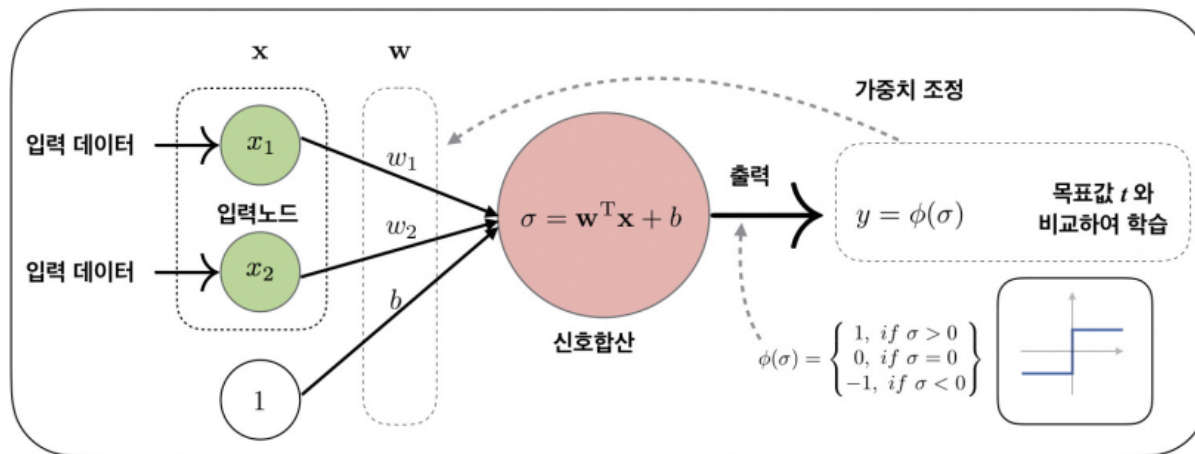
실습 목표

퍼셉트론 모델을 이용하여 참(1)과 거짓(-1)의 값을 갖는 두 입력에 대해 논리곱 AND 연산과, 논리합 OR 연산을 수행할 수 있는 퍼셉트론을 만들어 보자.



힌트

퍼셉트론 모델은 아래와 같이 두 개의 입력에 대해 \mathbf{w} 벡터가 곱해지고 편향 b 가 더한 결과를 활성화 함수를 통과하게 만들면 된다. 거짓과 참을 표현할 때 0,1을 사용하는 방법도 있고, -1, 1을 사용하는 방법도 있는데, 신호를 크게 분리하기 위해 우선은 -1과 1을 사용하자. 활성화 함수 $\phi(\cdot)$ 는 입력이 0보다 크면 1, 그렇지 않으면 -1을 반환한다.

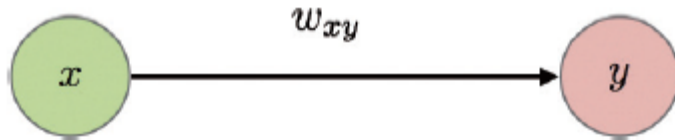


https://colab.research.google.com/drive/1USlwBX_eo9h-3aMyqFe8pDxFbSbkMQZN

학습의 원리 - 연결강도 변경(1)

❖ Hebb의 학습 법칙

- "함께 활성화되는 세포는 함께 연결된다"
- 두뇌 신경망의 기능이 고정되어 있지 않고 바뀔 수 있는 가소성(plasticity)을 가지고 있음을 설명



▪ 활성화 정도

- 두 신경세포가 내는 신호의 곱이고, 함께 연결된다는 것은 연결의 강도 w_{xy} 가 커짐 $w_{xy} = xy$
- m번 관측한 경우 매번 관측치의 평균 $w_{xy} = \frac{1}{m} \sum_{k=1}^m x^{(k)} y^{(k)}$

▪ Hebb의 학습법칙

- k+1 번째의 연결강도 변경
- 신경세포에 가해지는 입력과 출력에 따라 새로운 가중치를 바꾸어 감

$$w_{xy}^{(k+1)} = w_{xy}^{(k)} + x^{(k+1)} y^{(k+1)}$$

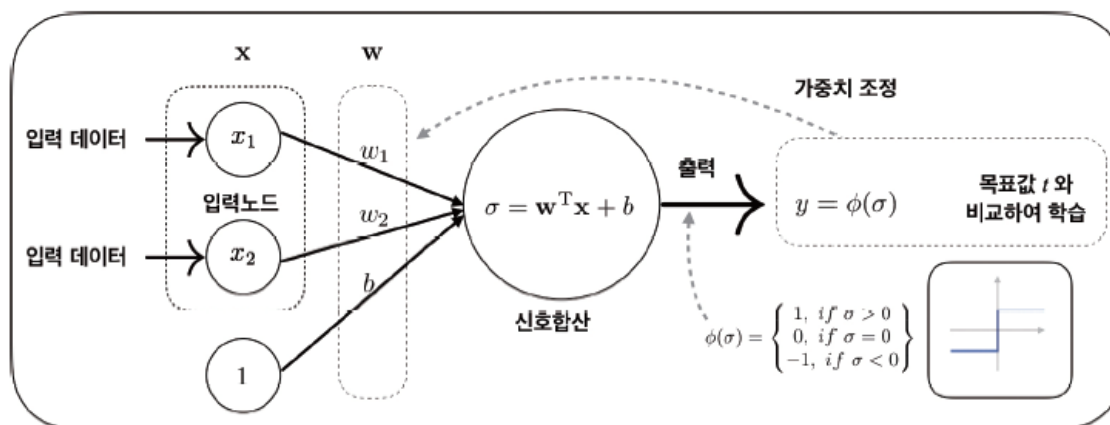
학습의 원리 - 연결강도 변경(2)

❖ 퍼셉트론의 학습

- 목표치를 제시하고 가중치가 목표치를 발생시키도록 만들어 가는 것
- 헵의 학습규칙에서 출력에 해당하는 부분을 목표값 t 로 변경
- 가중치의 갱신도 **학습률** learning rate η 를 통해 조금씩 이루어지게 만들

$$w_{xy}^{(k+1)} = w_{xy}^{(k)} + \eta x^{(k+1)} t^{(k+1)}$$

❖ 이항 불리언 연산 binary boolearn operation



- x_1, x_2 : true 혹은 false
- b (bias, 편향): 신호값을 양의 방향이나 음의 방향으로 이동시키는 역할

학습의 원리 - 연결강도 변경(3)

❖ LAB⁷⁻² 논리합을 수행하는 퍼셉트론 만들기

실습 목표

퍼셉트론 모델을 이용하여 참(1)과 거짓(-1)의 값을 갖는 두 입력에 대해 논리합을 수행할 수 있도록 모델을 학습시켜 보라.



힌트

퍼셉트론 모델은 입력 노드 둘을 담은 벡터 X 와 이 입력에 곱해질 가중치 벡터 W 를 준비하고, 편향 b 를 더해주면 된다. 출력은 $W \cdot X + b$ 의 값을 활성화 함수에 넣어 구한다.

❖ LAB⁷⁻³ 다양한 논리 연산이 가능하게 퍼셉트론 훈련하기

실습 목표

LAB⁷⁻²를 이용하여 다양한 논리 연산이 가능하도록 퍼셉트론을 학습시켜 보자.



힌트

학습을 실시하는 `train` 함수를 이용하여 필요한 논리 연산의 입력과 출력을 제공하고 훈련을 실시하면 다양한 논리 연산을 구현할 수 있다.

https://colab.research.google.com/drive/1USlwBX_eo9h-3aMyqFe8pDxFbSbkMQZN

학습의 원리 - 연결강도 변경(4)



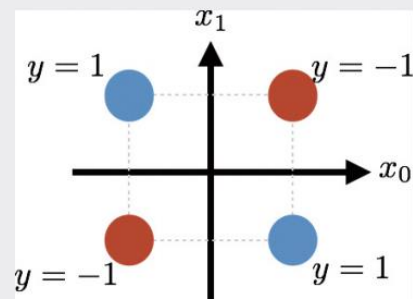
도전문제 7.1 XOR의 진리표를 이용하여 퍼셉트론을 학습시켜 보자

XOR는 아래와 같은 기호로 표시된다. 진리표가 그 아래 나타나 있다. 이런 논리 연산을 수행하는 퍼셉트론을 만들 수 있을까? 만들어지지 않는다면 그 이유를 생각해 보라.



$$y = (x_0 \wedge \neg x_1) \vee (\neg x_0 \wedge x_1)$$

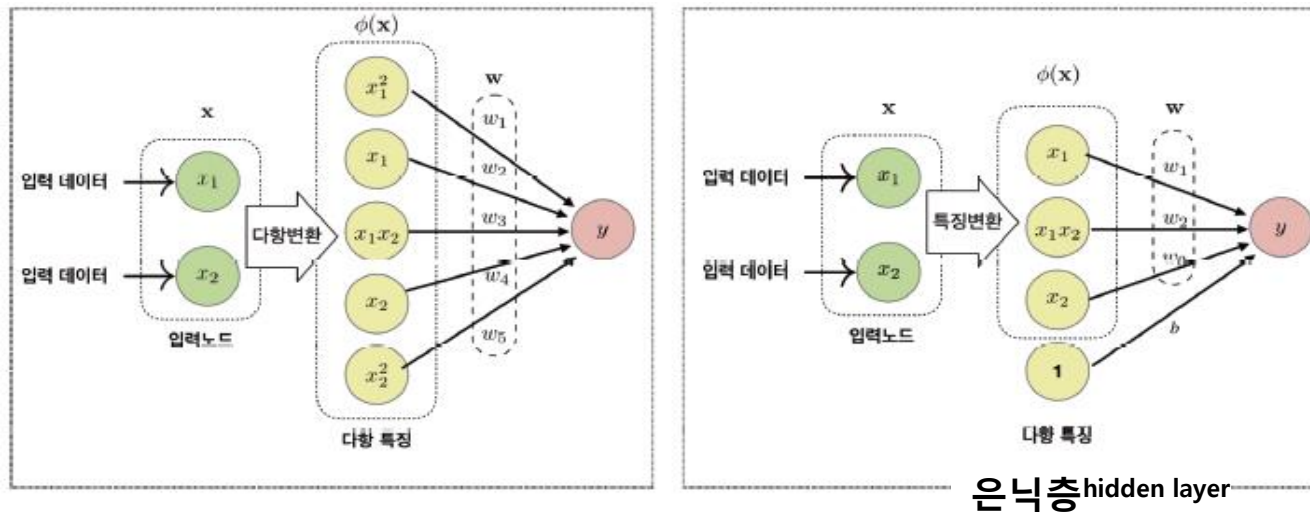
x_0	x_1	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



XOR 문제를 풀 수 있는 퍼셉트론 (1)

❖ 특징 벡터의 다항화

- 선형함수의 모델을 비선형 함수로 변형
- $(x_1, x_2) \Rightarrow (x_1^2, x_1, x_1x_2, x_2, x_2^2)$



❖ LAB⁷⁻⁴ 입력 다항화로 XOR를 해결해 보기

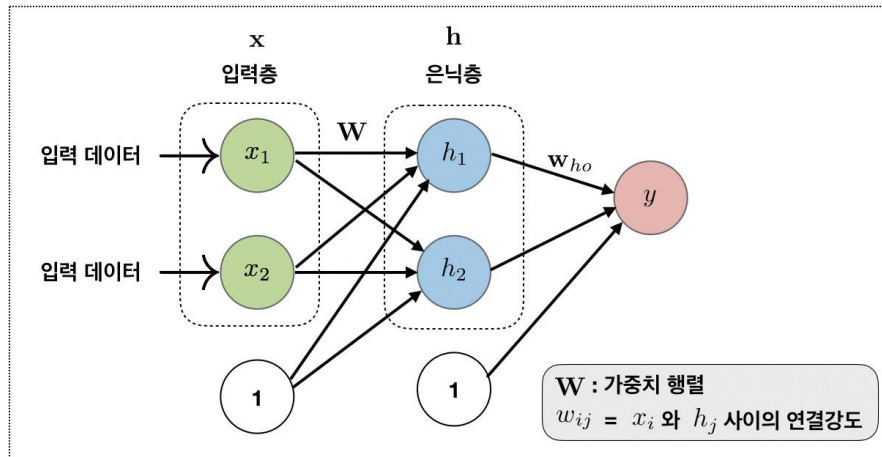
실습 목표

참 또는 거짓의 값을 갖는 두 개의 입력에 대해 XOR 연산을 수행할 수 있도록, 입력에 대한 다항 변환을 실시하고, 이를 통해 얻은 다항 특징에 대해 퍼셉트론 학습을 통해 XOR 연산을 수행하는 모델을 구현하자.

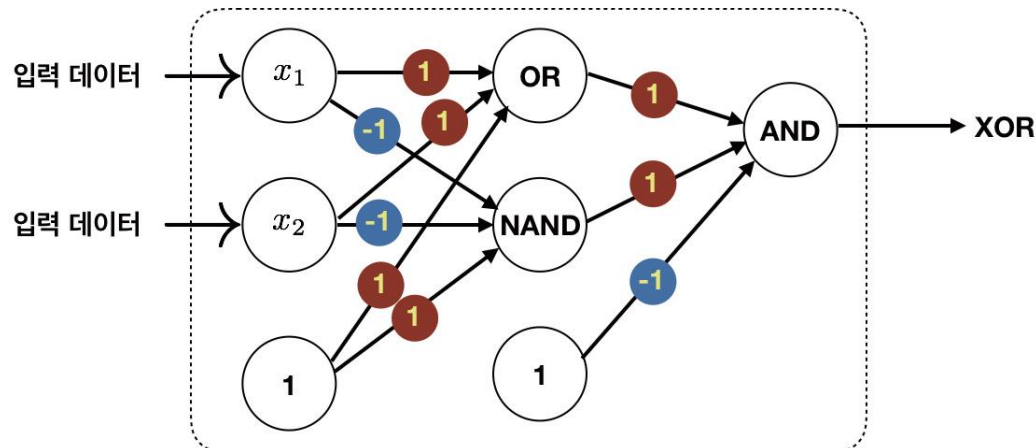
https://colab.research.google.com/drive/1USlwBX_eo9h-3aMyqFe8pDxFbSbkMQZN

XOR 문제를 풀 수 있는 퍼셉트론 (2)

❖ 다층 퍼셉트론(MLP) multi-layer perceptron




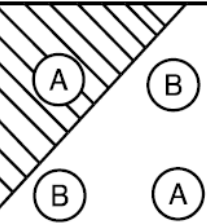
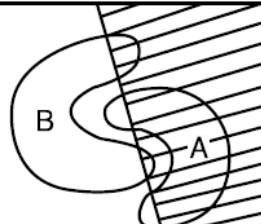
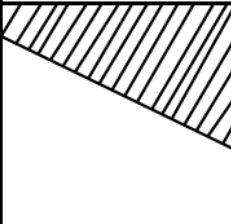
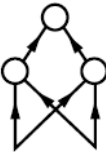
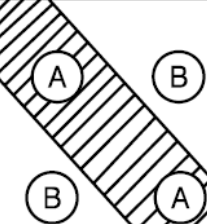
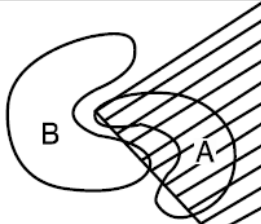
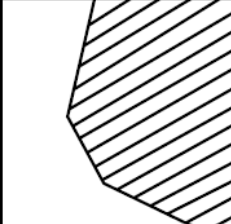

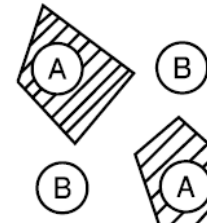
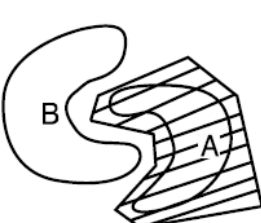
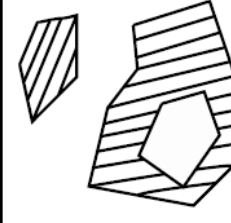
❖ XOR MLP



Multi Layer Perceptron

❖ 다층 퍼셉트론

- 대부분의 입력 패턴은 선형으로 분리 불가능한 문제
- 여러 개의 퍼셉트론(비선형 활성화 함수)을 여러 층으로 연결하여 복잡한 영역을 곡면으로 둘러싸는 결정 영역을 구하는 것

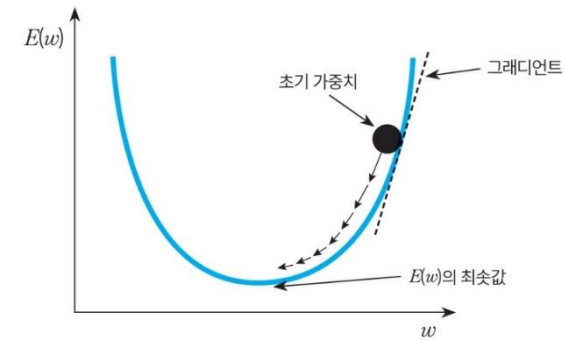
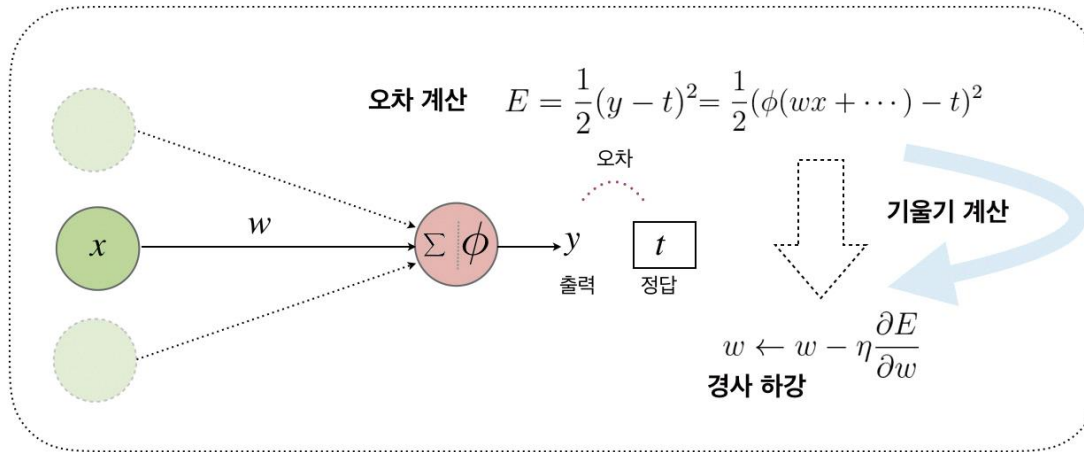
구조	결정 영역의 유형	XOR 문제	맞물린 영역 모양	전형적인 영역 모양
1층 구조 	초평면에 의한 평면 경계			
2층 구조 	열린 블록 혹은 닫힌 블록 경계			
3층 구조 	임의의 결정 경계 (노드의 수에 따라 모양이 결정)			

[그림 13-13]
신경망 층에 따른
결정 영역

MLP 학습 (1)

❖ 경사 하강법 gradient descent

- 오차 곡면의 기울기를 연결강도에 대해 구한 후 기울기 반대방향진행



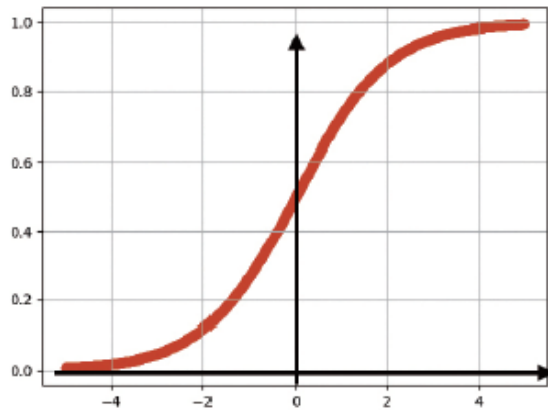
- 오차를 연결강도 w 에 대해 편미분

$$\begin{aligned}
 \frac{\partial E}{\partial w} &= \frac{1}{2} \frac{\partial}{\partial w} (\phi(wx + \dots) - t)^2 \\
 &= (\phi(wx + \dots) - t) \cdot \frac{\partial}{\partial w} \phi(wx + \dots) \\
 &= (\phi(wx + \dots) - t) \cdot \phi'(wx + \dots) \cdot \frac{\partial}{\partial w} (wx + \dots) \\
 &= (y - t) \cdot \phi'(wx + \dots) \cdot x
 \end{aligned}$$

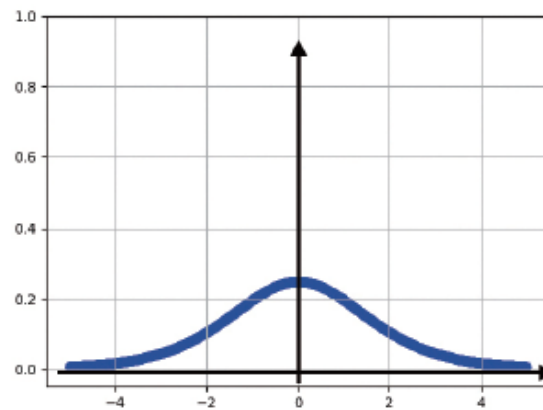
MLP 학습 (2)

❖ 활성화 함수: 시그모이드 sigmoid, 로지스틱 logistic 함수

$$\phi(x) = \frac{1}{1+e^{-x}}$$



$$\phi'(x) = \frac{1}{1+e^{-x}} \left(1 - \frac{1}{1+e^{-x}}\right) = \phi(x)(1-\phi(x))$$



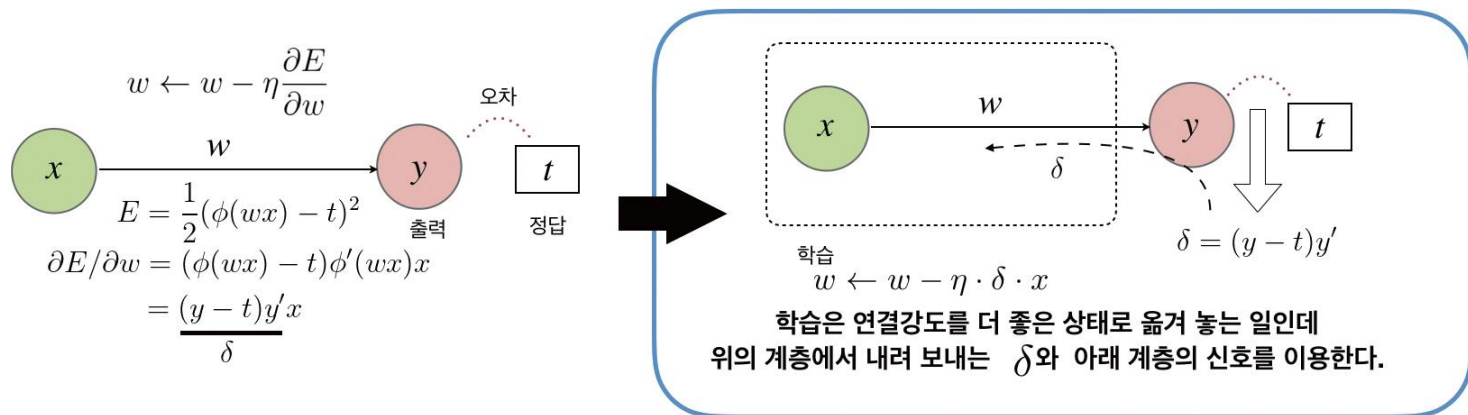
- $\frac{\partial E}{\partial w} = (y-t) \cdot \phi'(wx + \dots) \cdot x$
 $= (y-t) \cdot \phi(wx + \dots) \cdot (1 - \phi(wx + \dots)) \cdot x$
 $= (y-t) \cdot y \cdot (1-y) \cdot x$
- $\delta = (y-t) \phi'(wx + \dots) = (y-t) \cdot y \cdot (1-y)$
- 가중치 학습

$$w \leftarrow w - \eta \cdot \frac{\partial E}{\partial w} \quad w \leftarrow w - \eta \cdot \delta \cdot x$$

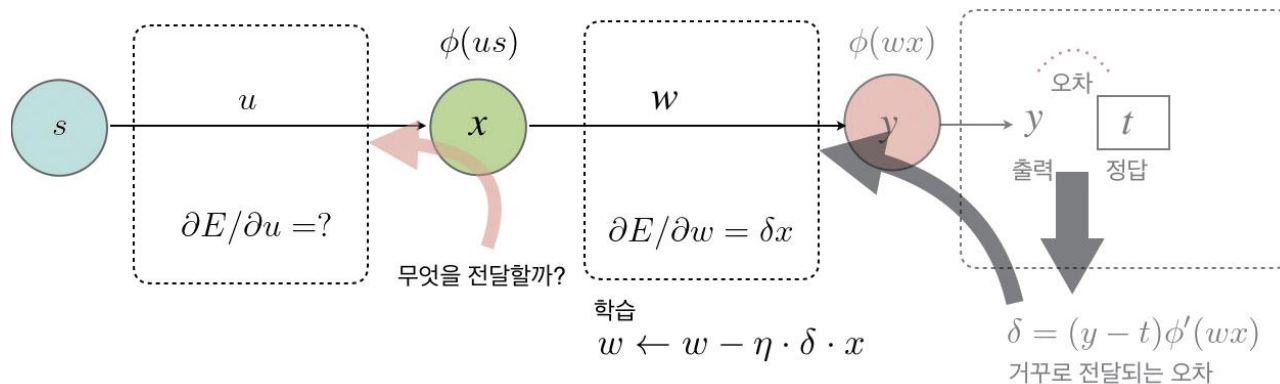
MLP 학습 (3)

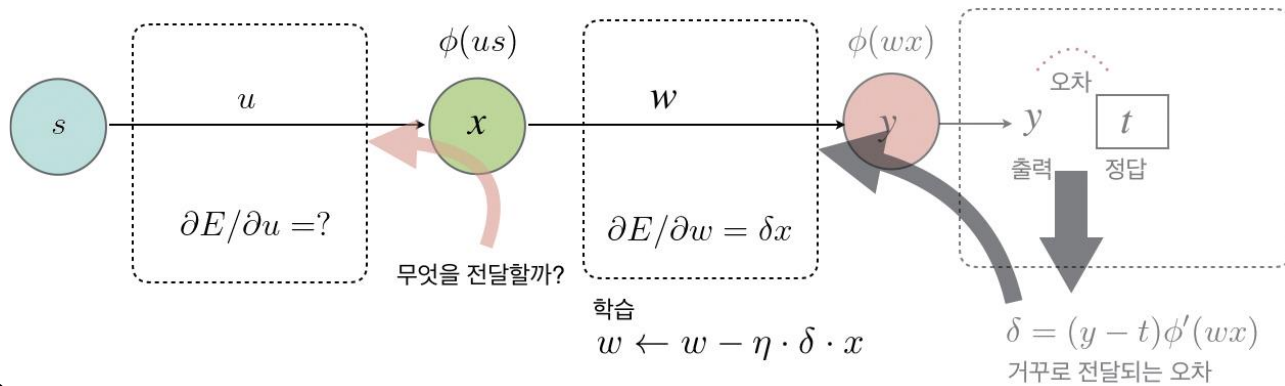
❖ 가중치 학습

- 목표 t 와 출력 y 의 차이, 그리고 출력의 미분 y' 가 연결망을 거꾸로 전달되어 가중치를 조정

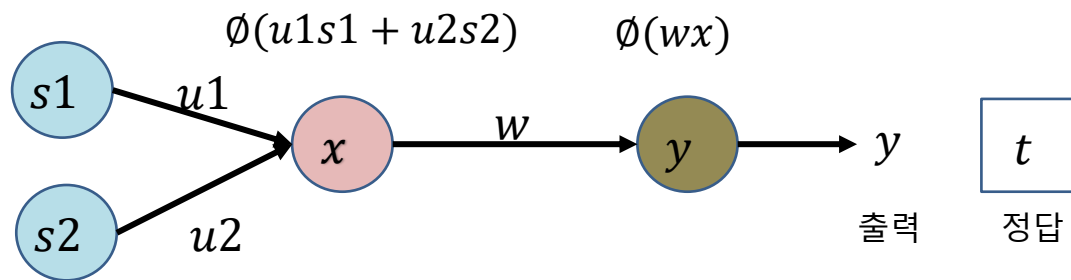


❖ 오차 역전파 (error backpropagation)





❖ $\phi(us)$



❖ $u1 \leftarrow u1 - \eta \cdot \delta \cdot w \cdot \phi'(u1s1) \cdot s1$

❖ $u2 \leftarrow u2 - \eta \cdot \delta \cdot w \cdot \phi'(u2s2) \cdot s2$

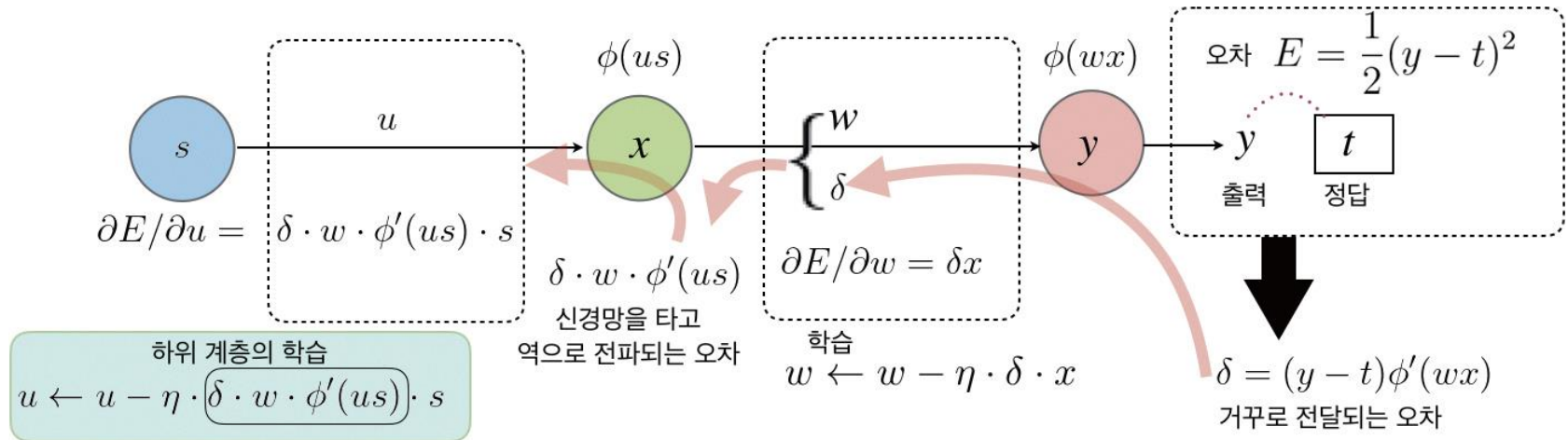
MLP 학습 (4)

❖ 미분의 연쇄법칙 chain rule

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial x} \cdot \frac{\partial x}{\partial u}$$

$$\partial E / \partial x = (y - t) y' w = \delta w$$

$$\frac{\partial E}{\partial u} = \frac{\partial E}{\partial x} \frac{\partial x}{\partial u} = \delta \cdot w \cdot \frac{\partial \phi(us)}{\partial u} = \delta \cdot w \cdot \phi'(us) \cdot s$$



❖ 역전파 알고리즘 정리

순전파 단계의 신호 증폭/감쇠 후 합산과 활성화 함수 적용

역전파 단계의 오차 증폭/감쇠 후 합산과 활성화 함수 미분 적용

The diagram illustrates the backpropagation process in a neural network. It shows the forward pass (activation and differentiation) and the backward pass (error propagation and weight/offset updates).

Forward Pass (Left to Right):

- Input Layer:** Inputs S_1 and S_2 are multiplied by weights u_1 and u_2 to produce x and x' .
- Hidden Layer:** x and x' are passed through an activation function $\phi(\Sigma x)$ and its derivative $\phi'(\Sigma x)$.
- Output Layer:** The hidden layer outputs are multiplied by weights w_1 and w_2 to produce the final outputs y_1 and y_2 .

Backward Pass (Right to Left):

- Output Layer:** Target values t_1 and t_2 are compared with predicted values y_1 and y_2 to calculate errors e_1 and e_2 .
- Hidden Layer:** Errors e_1 and e_2 are propagated back through the weights w_1 and w_2 to calculate the error e_x at the hidden layer.
- Input Layer:** The error e_x is propagated back through the weights u_1 and u_2 to calculate the error δ_x at the input layer.

Weight and Offset Updates:

- Input weights: $\Delta u_i = -\eta \delta_x s_i$
- Hidden weights: $\Delta w_i = -\eta \delta_i x$

Node Labels:

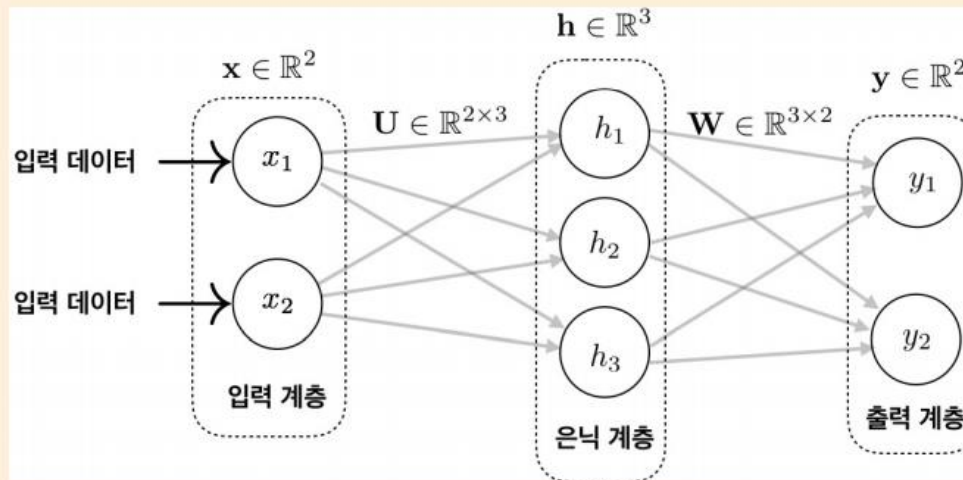
- \mathbf{X} : Hidden layer nodes
- \mathbf{Y}_1 : Output node 1
- \mathbf{Y}_2 : Output node 2

MLP 학습 (6)

❖ LAB⁷⁻⁵ XOR 연산이 가능한 다층 퍼셉트론 만들기

실습 목표

역전파 모델로 아래와 같은 다층 퍼셉트론을 훈련시켜 XOR 연산을 수행하게 하라. 이 퍼셉트론의 출력은 y_1 이 켜지면 참, y_2 가 켜지면 거짓을 나타낸다.



힌트

앞에서 살펴본 바와 같이 각 계층으로 유입되는 오차를 미분치에 곱해서 델타 벡터를 만들고, 이를 이용하여 연결강도를 수정한다. 그리고 이 델타 벡터는 가중치를 타고 내려가게 하면 된다.

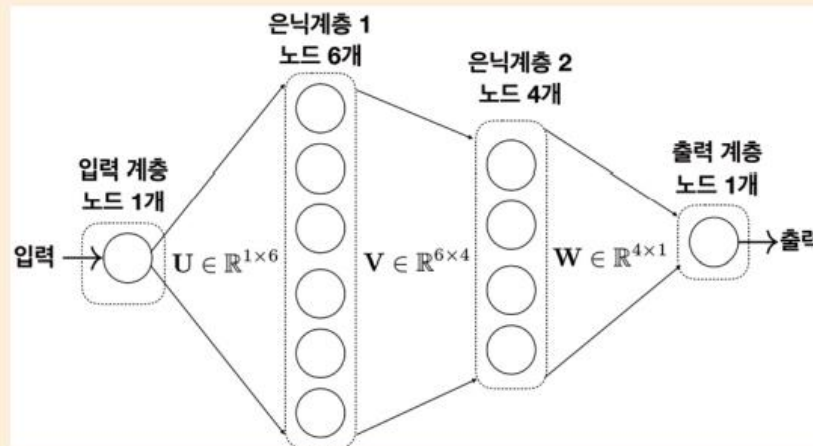
MLP 학습 (7)

❖ LAB⁷⁻⁶ 다층 퍼셉트론으로 비선형 회귀 구현하기

실습 목표

퍼셉트론의 층을 증가시켜 아래와 같은 모델로 비선형 회귀를 구현해 보자. 적용할 데이터는 LAB⁶⁻¹에서 사용한 것과 같이 다음 URL에 있는 데이터이다.

<https://github.com/dknife/ML/raw/main/data/nonlinear.csv>



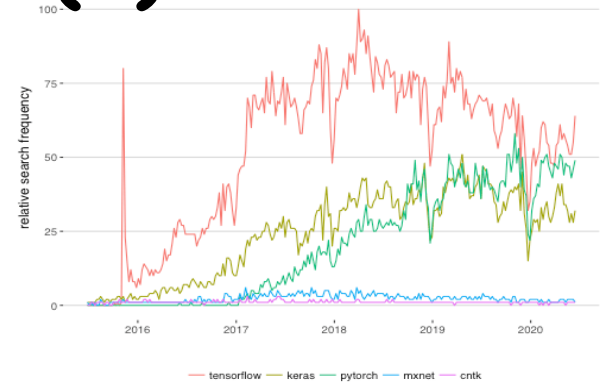
힌트

층을 늘리는 것은 순전파와 역전파 단계가 하나 늘어나는 것뿐이고, 계산은 동일한 방식으로 이루어진다.

https://colab.research.google.com/drive/1USlwBX_eo9h-3aMyqFe8pDxFbSbkMQZN

TensorFlow & Keras (1)

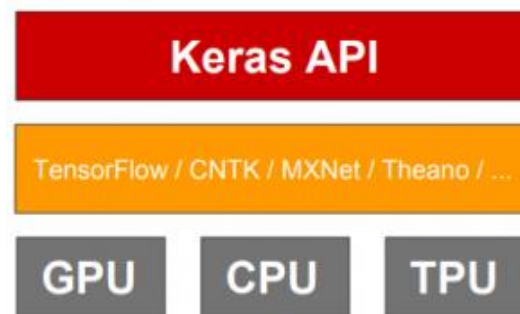
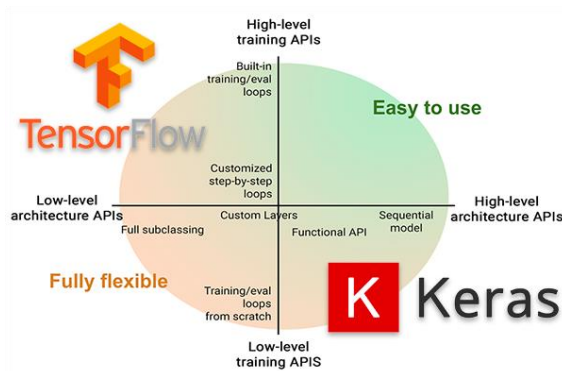
- ❖ Deep Learning programming architecture
 - Tensorflow: low-level architecture
 - Keras, Pytorch, ... : high-level architecture



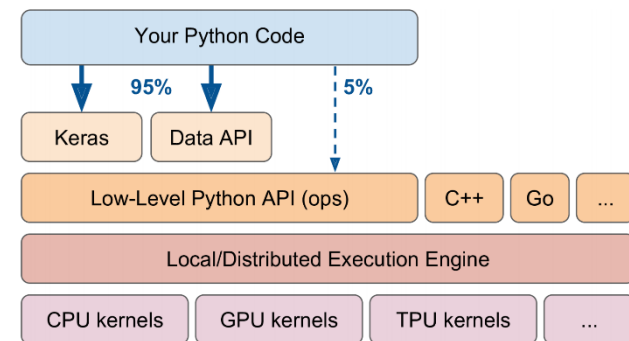
<https://opendatascience.com/deep-learning-with-tensorflow-2-pytorch/>

❖ Keras

- <https://keras.io>
- <https://www.youtube.com/watch?v=UYRBHFAvLSs>
- <https://www.youtube.com/watch?v=uhzGTijaw8A>



<https://punchplatform.com/2019/07/08/tensorflow-keras-pml-pipeline/>



TensorFlow & Keras (2)

❖ Keras API reference <https://keras.io/api>

Python For Data Science Cheat Sheet

Keras

Learn Python for data science [interactively](https://www.datacamp.com) at www.datacamp.com



Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random(1000, 100)
>>> labels = np.random.randint(10, size=(1000, 1))
>>> model = Sequential()
>>> model.add(Dense(10,
>>>                 activation='relu',
>>>                 input_shape=(100,)))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='adam',
>>>               loss='binary_crossentropy',
>>>               metrics=['accuracy'])
>>> model.fit(data, labels, epochs=10, batch_size=10)
>>> predictions = model.predict(data)
```

Data

[Also see NumPy, Pandas & SciKit Learn](#)

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import mnist, fashion_mnist, cifar10, cifar100, imnet
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>> (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
>>> (x_train, y_train), (x_test, y_test) = cifar10.load_data()
>>> (x_train, y_train), (x_test, y_test) = cifar100.load_data()
>>> (x_train, y_train), (x_test, y_test) = imnet.load_data()
>>> num_classes = 10
```

Other

```
>>> from sklearn.metrics import accuracy
>>> data = np.loadtxt('mnist_train.csv', delimiter=',')
>>> (x_train, y_train), (x_test, y_test) = sklearn.cross_validation.train_test_split(
>>>     data, data[:, 10], test_size=0.1, random_state=0)
>>> x = data[:, 0:9]
>>> y = data[:, 10]
```

Preprocessing

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train = sequence.pad_sequences(x_train, maxlen=80)
>>> x_test = sequence.pad_sequences(x_test, maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> y_train = to_categorical(y_train, num_classes)
>>> y_test = to_categorical(y_test, num_classes)
>>> x_train = to_categorical(x_train, num_classes)
>>> x_test = to_categorical(x_test, num_classes)
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model.add(Dense(10))
>>> model.add(Dense(1))
```

Multilayer Perceptron (MLP)

Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(10,
>>>                 input_shape=(10,),
>>>                 kernel_initializer='uniform',
>>>                 activation='relu'))
>>> model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
>>> model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Multi-Class Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(10, activation='relu', input_shape=(10,)))
>>> model.add(Dense(10))
>>> model.add(Dense(10, activation='relu'))
>>> model.add(Dense(10))
>>> model.add(Dense(10, activation='softmax'))
```

Regression

```
>>> model.add(Dense(10, activation='relu', input_shape=train_data.shape[1:]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation, Conv2D, MaxPooling2D, Flatten
>>> model.add(Conv2D(32, (3, 3), padding='same', input_shape=train_data.shape[1:]))
>>> model.add(Activation('relu'))
>>> model.add(Conv2D(32, (3, 3)))
>>> model.add(Activation('relu'))
>>> model.add(MaxPooling2D(pool_size=(2, 2)))
>>> model.add(Dropout(0.25))
>>> model.add(Conv2D(64, (3, 3), padding='same'))
>>> model.add(Activation('relu'))
>>> model.add(Conv2D(64, (3, 3)))
>>> model.add(Activation('relu'))
>>> model.add(MaxPooling2D(pool_size=(2, 2)))
>>> model.add(Dropout(0.25))
>>> model.add(Flatten())
>>> model.add(Dense(512))
>>> model.add(Activation('relu'))
>>> model.add(Dense(512))
>>> model.add(Activation('relu'))
>>> model.add(Dropout(0.5))
>>> model.add(Dense(num_classes))
>>> model.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.layers import Embedding, LSTM
>>> model.add(Embedding(10000, 128))
>>> model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
>>> model.add(Dense(1, activation='sigmoid'))
```

Inspect Model

```
>>> model.output_shape
>>> model.summary()
>>> model.get_weights()
>>> model.get_weights()
```

Model output shape
Model summary representation
Model configuration
List all weight tensors in the model

Compile Model

MLP: Binary Classification

```
>>> model.compile(optimizer='adam',
>>>               loss='binary_crossentropy',
>>>               metrics=['accuracy'])
```

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='adam',
>>>               loss='categorical_crossentropy',
>>>               metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='adam',
>>>               loss='mse',
>>>               metrics=['mse'])
```

Recurrent Neural Network

```
>>> model.compile(optimizer='adam',
>>>               loss='binary_crossentropy',
>>>               metrics=['accuracy'])
```

Model Training

```
>>> model.fit(x_train,
>>>           y_train,
>>>           batch_size=32,
>>>           epochs=10,
>>>           validation_data=(x_test, y_test))
```

Evaluate Your Model's Performance

```
>>> score = model.evaluate(x_test,
>>>                        y_test,
>>>                        batch_size=32)
```

Prediction

```
>>> model.predict(x_test, batch_size=32)
>>> model.predict_classes(x_test, batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model.save('model.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

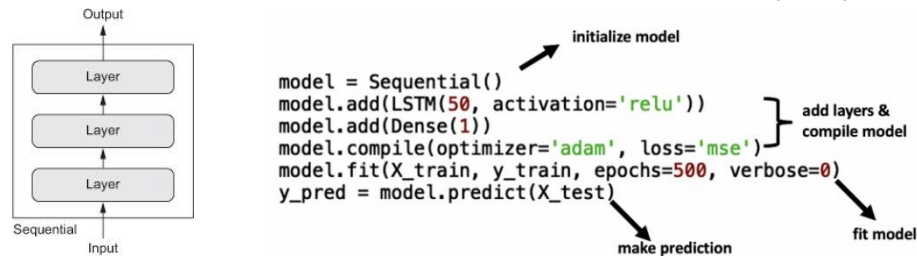
```
>>> from keras.optimizers import Adam
>>> opt = Adam(lr=0.001, decay=1e-6)
>>> model.compile(optimizer=opt, loss='binary_crossentropy',
>>>               metrics=['accuracy'])
```

Early Stopping

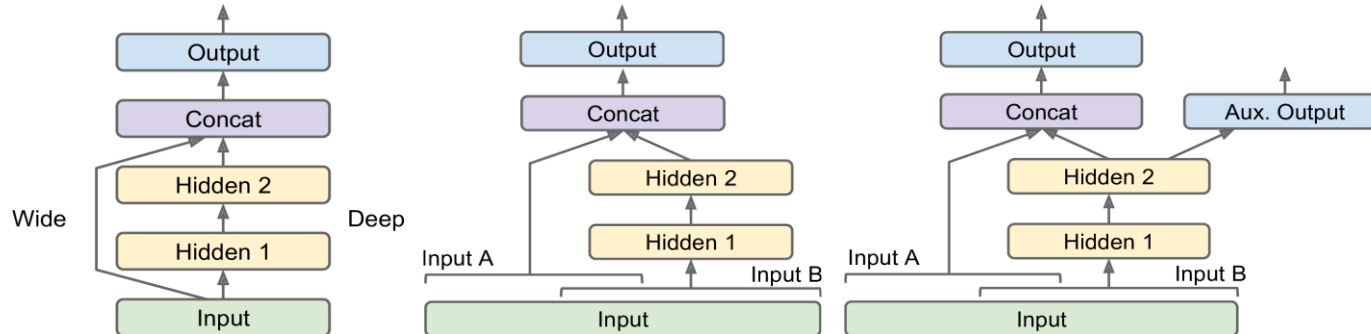
```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=10)
>>> model.fit(x_train,
>>>           y_train,
>>>           batch_size=32,
>>>           epochs=10,
>>>           validation_data=(x_test, y_test),
>>>           callbacks=[early_stopping_monitor])
```

TensorFlow & Keras (3)

❖ Sequential API:



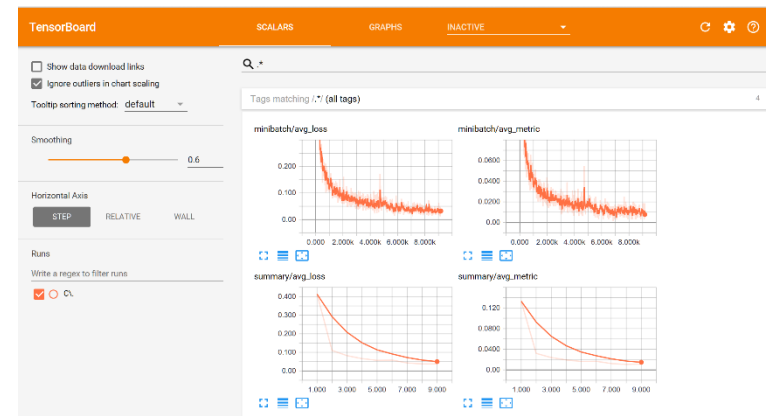
❖ Functional API: Building Complex Models



❖ Subclassing API: Building Dynamic Models

❖ Callbacks

❖ Visualization Using TensorBoard



Google PlayGround 실습 (1)

❖ <https://developers.google.com/machine-learning/crash-course?hl=ko>

The screenshot shows the Google Machine Learning Crash Course landing page in Korean. The top navigation bar includes links for '머신러닝' (Machine Learning), '과정' (Course), '실습' (Lab), '가이드' (Guide), and '용어집' (Glossary). Below this is a secondary navigation bar with '단기집중과정' (Intensive Course), '문제 프레임' (Problem Framework), '데이터 준비' (Data Preparation), '클러스터링' (Clustering), and '추천 시스템' (Recommendation System). The main content area features a large image of a woman writing on a whiteboard. The whiteboard has several handwritten notes: 'Non-Linear Classification' with a definition of 'Feature Cross', '머신러닝 단기집중과정' (Machine Learning Intensive Course), '텐서플로우 API 사용' (TensorFlow API Usage), 'Google의 실용적인 머신러닝 속성 입문 과정' (Google's Practical Machine Learning Property Introduction Course), the equation $Y_3 = X_1 X_2$, the linear equation $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x$, 'Neural Network' with a definition, and the 'Sigmoid Activation Function' formula. Below the whiteboard image, the text reads '머신러닝을 배우려는 실무자를 위한 자기 주도형 학습 가이드' (Self-paced learning guide for practitioners who want to learn machine learning). Below this is a paragraph: '머신러닝 단기집중과정은 동영상 강의와 실제 우수사례, 실습이 포함된 일련의 강의로 구성되어 있습니다.' (The Machine Learning Intensive Course is a series of lectures including videos, real-world best practices, and hands-on exercises). At the bottom, there are six icons with corresponding text: '실습 40개 이상' (More than 40 hands-on exercises), '강의 25개' (25 lectures), '15시간' (15 hours), 'Google 연구원의 강의' (Lectures by Google researchers), '실무 활용 우수사례' (Real-world best practices for application), and '실제로 작동하는 알고리즘의 명확한 시각화' (Clear visualization of algorithms that actually work).

머신러닝 과정 실습 가이드 용어집

단기집중과정 문제 프레임 데이터 준비 클러스터링 추천 시스템

Non-Linear Classification Feature Cross: A synthetic feature formed by crossing (multiplying or taking a cartesian product of) individual features. $[A \times B]$ or $[A \times B \times C \times D \times E]$

머신러닝 단기집중과정
텐서플로우 API 사용

Google의 실용적인 머신러닝 속성 입문 과정

입문 교육 과정 시작 사전 준비사항 보기

$Y_3 = X_1 X_2$

$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x$

Neural Network
A model that is composed of layers (min 1)

Sigmoid Activation Function

머신러닝을 배우려는 실무자를 위한
자기 주도형 학습 가이드

머신러닝 단기집중과정은 동영상 강의와 실제 우수사례, 실습이 포함된 일련의 강의로 구성되어 있습니다.

실습 40개 이상

강의 25개

15시간

Google 연구원의 강의

실무 활용 우수사례

실제로 작동하는 알고리즘의 명확한 시각화

Google PlayGround 실습 (2)

❖ <https://playground.tensorflow.org>

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

