1. 텐서플로 소개

TensorFlow 한글 문서

1.1 머신 러닝이란

머신 러닝(영어: machine learning, ML)

경험을 통해 자동으로 개선하는 컴퓨터 알고리즘의 연구이다. 방대한 데이터를 분석해 '미래를 예측하는 기술'이자 인공지능의 한 분야로 간주된다. 기계 학습은 복잡한 패턴에 대한 학습을 통해 상황에 대한 예측과 의사 결정을 돕는다. 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야이다.

컴퓨터를 인간처럼 학습시킴으로써 컴퓨터가 새로운 규칙을 생성할 수 있지 않을까 하는 시도에서 시작되어 컴퓨터가 스스로 학습할 수 있도록 도와주는 알고리즘이나 기술을 개발하는 분야를 말합니다.

인공지능의 한 분야로, 데이터를 이용하여 컴퓨터가 스스로 학습하고 지능적인 작업을 수행하도록 하는 기술이다.

인간이 학습하는 방식을 모방하기 위한 데이터와 알고리즘의 사용에 초점을 맞춘 인공 지능(AI)의 한 분야로서, 시간이 지남에 따라 점차 정확도를 향상시킵니다.

컴퓨터가 명시적으로 프로그램되지 않고도 학습할 수 있도록 하는 연구 분야(field of study that gives computers the ability to learn without being explicitly programmed)

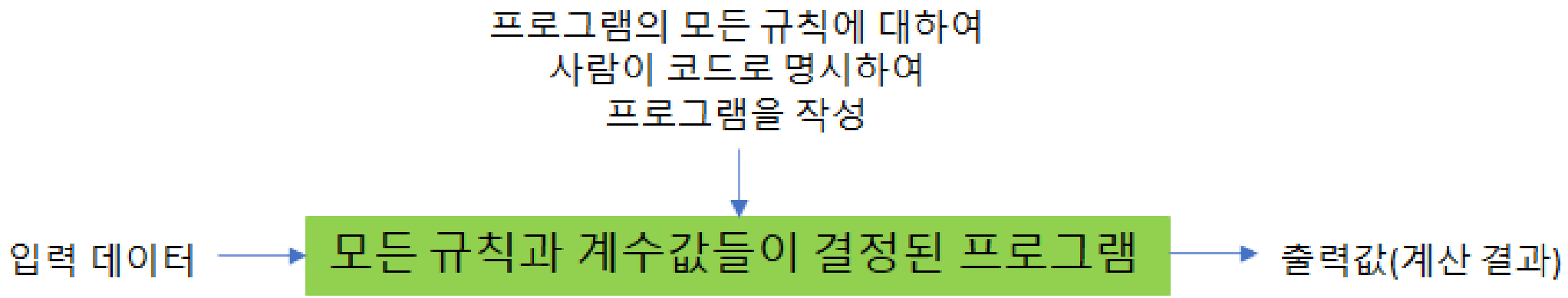
Machine Learning

1.2 전통적인 프로그래밍의 한계

전통적인 프로그래밍

전통적인 프로그래밍에서는 프로그래머가 명시적인 지침을 제공합니다. 컴퓨터가 따라야 하는 규칙. 이러한 지침은 프로그래밍 언어로 작성되어 컴퓨터가 이해할 수 있는 코드로 변환됩니다.

보통의 전통적인 프로그래밍 방식은 입력된 데이터를 처리하는 규칙을 모두 사람이 정하여 코드에 명확하게 지정하는 것입니다.



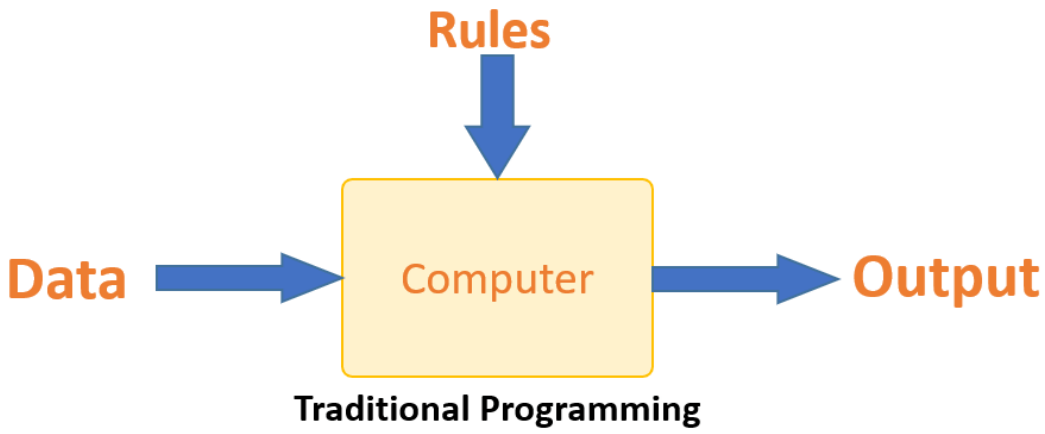
전통적인 프로그램 방식

1.3 프로그래밍에서 학습으로

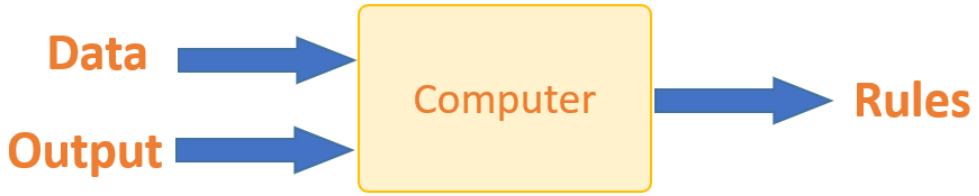
머신 러닝

머신 러닝에서는 프로그래머가 예제를 제공하고 컴퓨터가 스스로 학습하도록 합니다. 컴퓨터는 데이터의 패턴을 식별하고 이러한 패턴을 사용하여 새로운 예측을 합니다.

전통적인 보통의 프로그램이 모든 규칙과 계수의 값들을 프로그램 안에 명확하게 정해 주어야 하는데 반하여 기계학습 프로그래밍 방법은 규칙과 사용할 함수의 모양만을 명시적으로 결정한 프로그램을 만듭니다. 그리고 데이터 셋을 가지고 훈련(Training)과정을 통하여 최선의 계수 값들을 찾아냅니다. 이 과정을 통해 얻은 계수 값들을 프로그램에서 불러서 새로운 데이터를 입력 받아서 결과값을 출력하는 과정을 추론(Inferencing)이라고 합니다. 기계학습 방법으로 만든 프로그램은 이렇게 훈련(Training)과 추론(Inferencing), 2단계 과정을 통하여 프로그램이 완성되는 프로그래밍 방법입니다

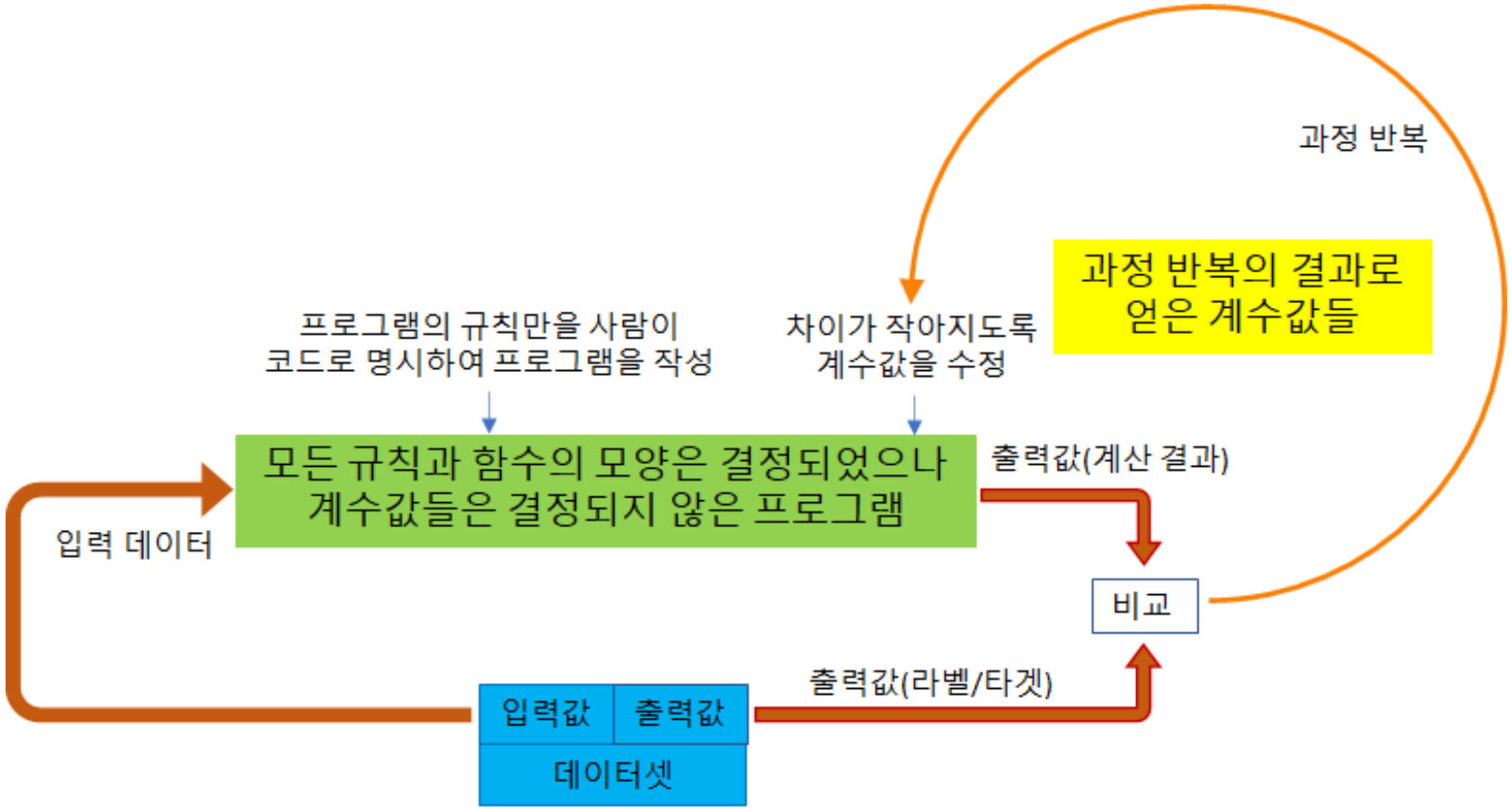


Traditional Programming



Machine Learning

1.3 프로그래밍에서 학습으로

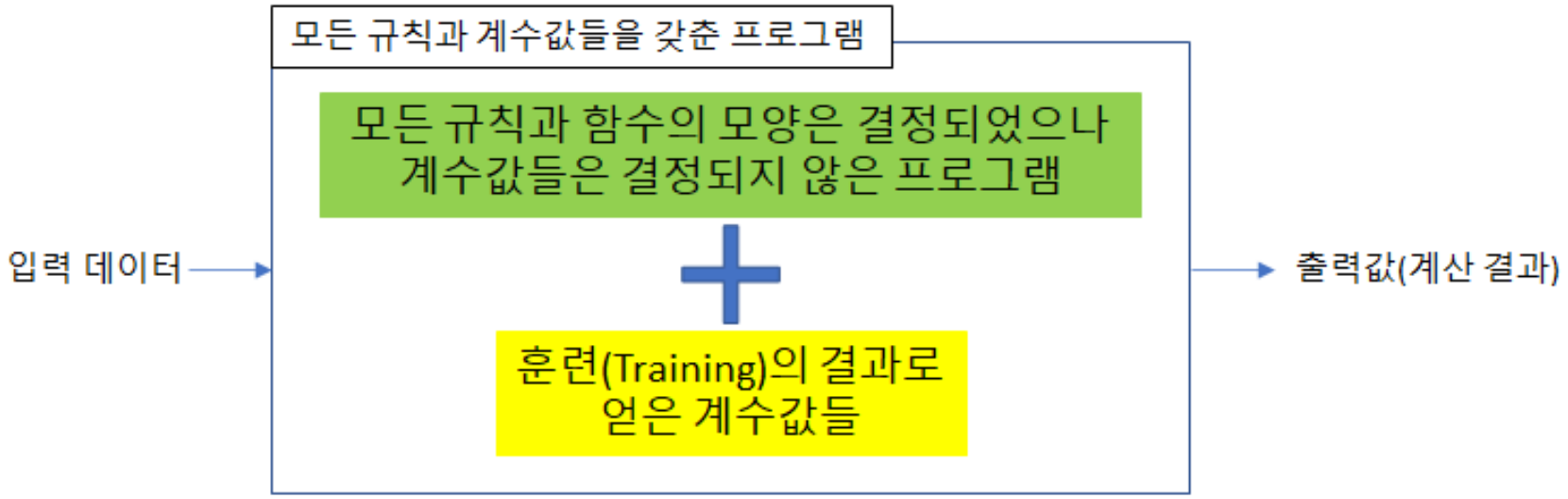


기계학습 프로그램 방식의 훈련(Training) 단계

그림에서 처음의 계수 값은 무작위로 주거나 특정한 방법에 의해 초기화합니다. 그 다음 데이터 셋에 있는 입력 데이터를 초기화된 계수 값을 갖는 프로그램으로 계산하여 출력 값을 계산하고 데이터셋에서 입력데이터와 쌍으로 있던 출력 값을 꺼내서 비교합니다.

계산해서 얻은 값과 데이터셋에서 꺼내온 값의 차이가 작아지는 방향으로 계수 값을 변경합니다. 이러한 작업을 전체 데이터 셋에 대하여 여러 번 반복하여 최적의 계수 값들을 찾아내는 과정이 기계 학습의 훈련 단계입니다. 보통은 이런 과정을 통해 최종 값으로 얻은 계수 값을 별도의 파일에 저장하여 추론 프로그램과 함께 사용하게 됩니다.

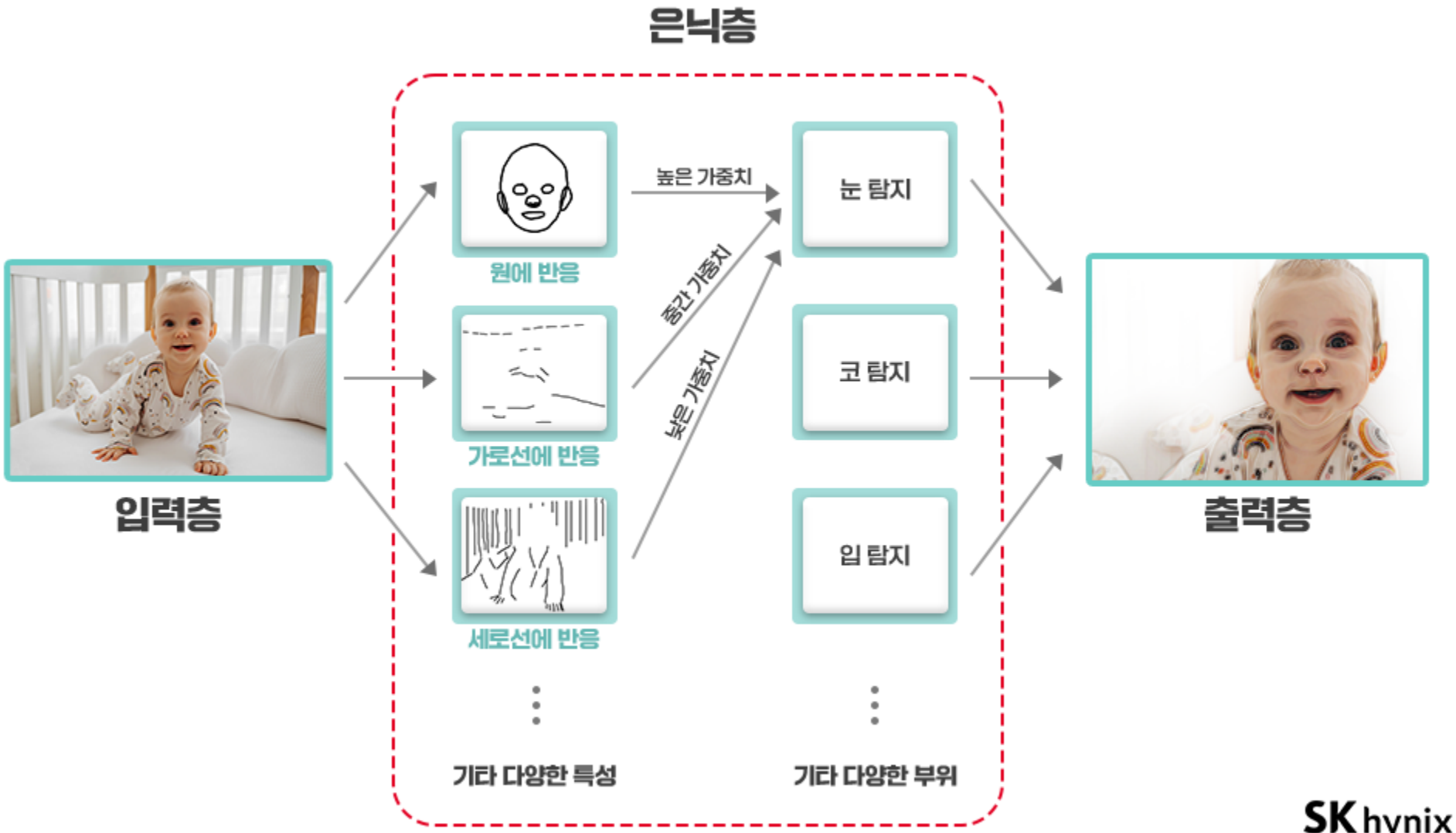
1.3 프로그래밍에서 학습으로



기계학습 프로그램 방식의 추론(Inferencing) 단계

위 그림은 추론단계를 보여줍니다. 훈련단계를 통하여 최적의 계수값들을 구하고 나면 이제 새로운 데이터를 입력으로 하여 원하는 결과값을 계산해 내는 프로그램을 준비해야 합니다. 이렇게 훈련을 통해 얻은 계수값을 이용하여 새로운 데이터에 대한 계산 결과를 얻는 과정을 추론이라고 하며, 이때 사용하는 프로그램을 추론 프로그램이라고 합니다.
기계학습은 궁극적으로 이 추론 프로그램을 사용하기 위한 프로그래밍 방법입니다.

1.3 프로그래밍에서 학습으로



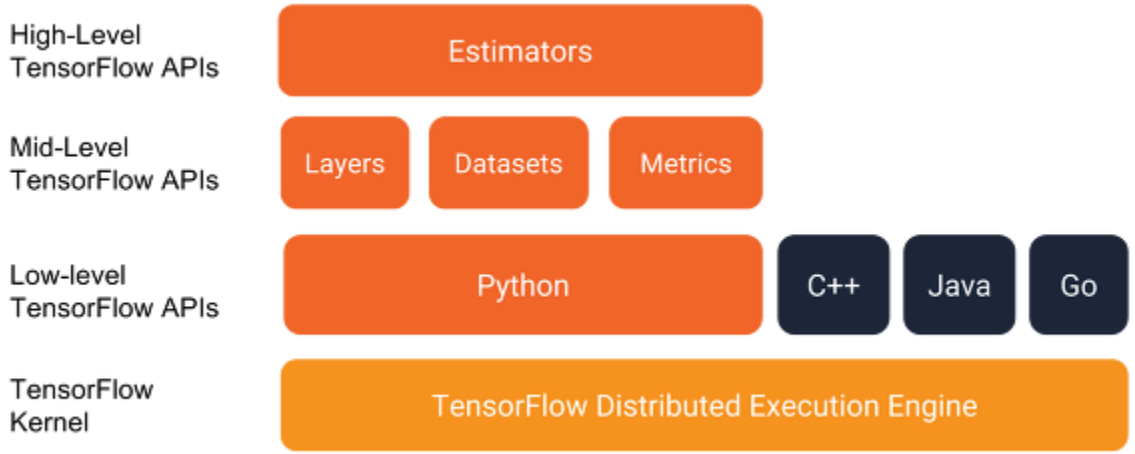
1.4 텐서플로란

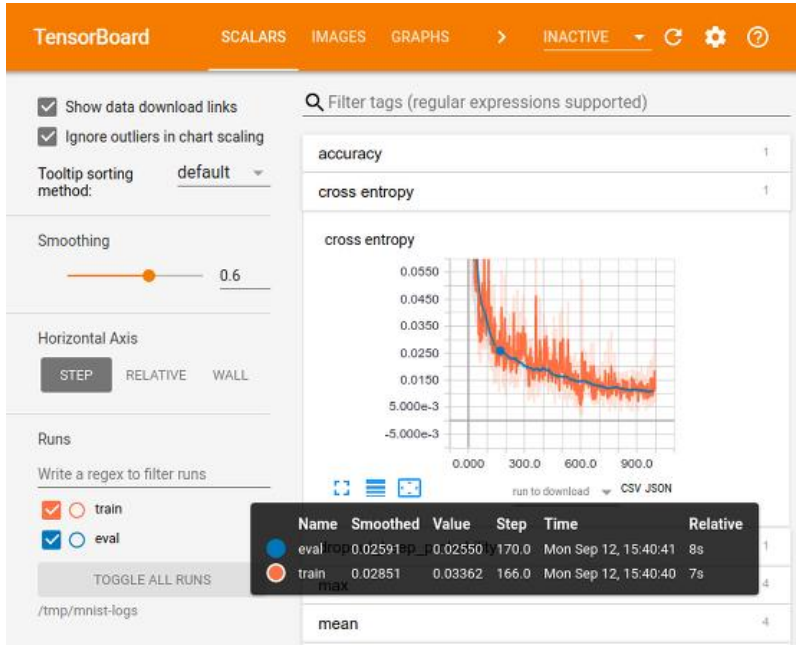
텐서플로

구글리서치 산하의 딥러닝 팀인 구글브레인 팀이 오픈 소스로 공개한 기계학습 라이브러리. 기계학습 분야를 일반인들도 사용하기 쉽도록 다양한 기능들을 제공한다.

TensorFlow를 사용하면 어떤 환경에서도 실행할 수 있는 ML 모델을 쉽게 만들 수 있습니다.

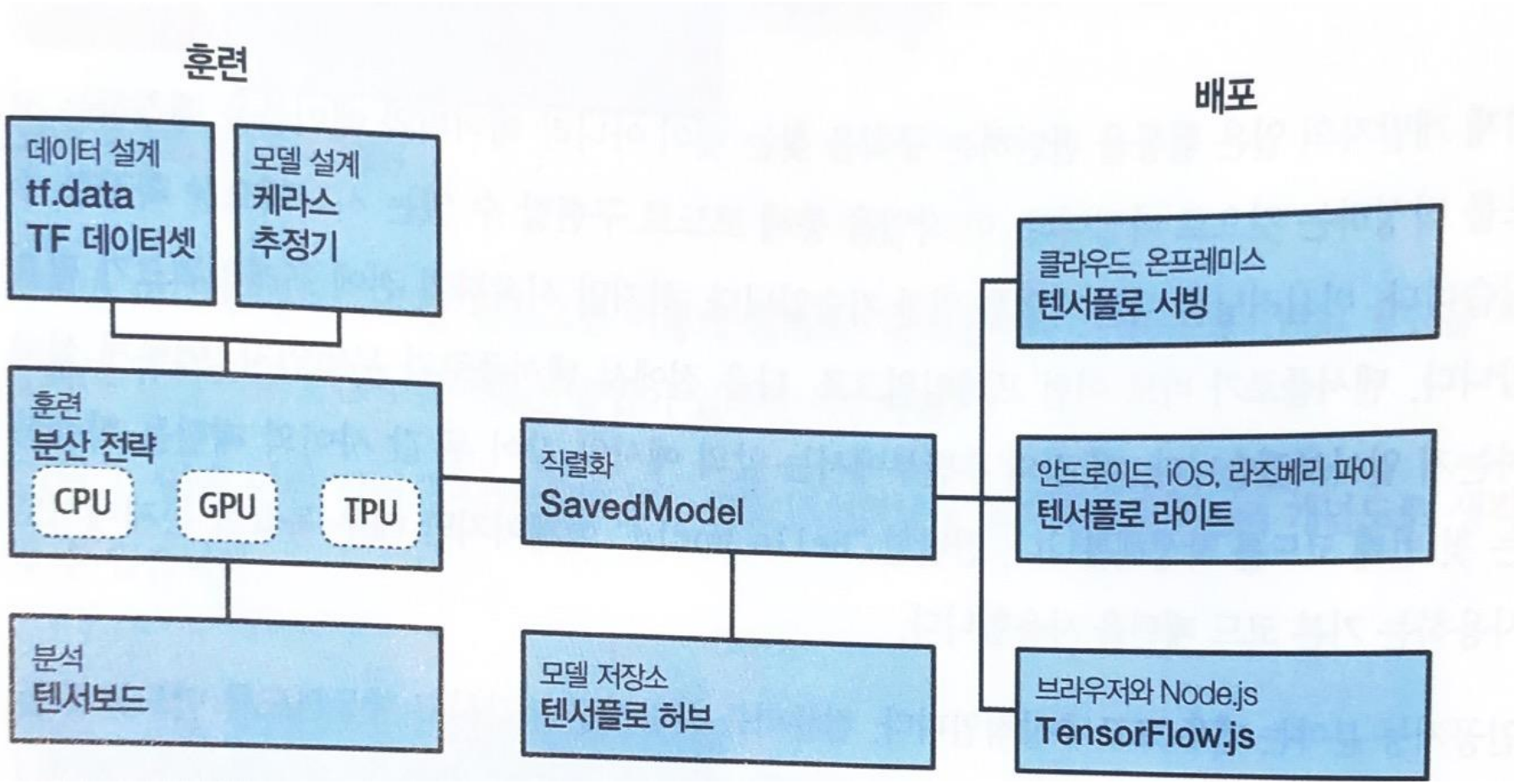
텐서플로(TensorFlow) 또는 텐서플로우는 다양한 작업에 대해 데이터 흐름 프로그래밍을 위한 오픈소스 소프트웨어 라이브러리이다. 심볼릭 수학 라이브러리이자, 인공 신경망같은 기계 학습 응용프로그램 및 딥러닝(deep Learning)에도 사용된다.





1.4 텐서플로란

텐서플로의 고수준 구조



1.5 텐서플로 사용하기

텐서플로 설치하기

1, Python 설치하기

https://www.python.org/downloads/

2. Anaconda 설치하기

https://www.anaconda.com/download

3. TensorFlow 설치하기

1) pip 업그레이드 : pip –m pip install --upgrade pip
2) Conda 환결 만들기 : conda create –n tensorflowEx1 python=3.5
3) TensorFlow 설치하기 : activate tensorflowEx1
pip install tensorflow

1.5 텐서플로 사용하기

구글 콜랩에서 텐서플로 사용하기

Colab(코랩)이란

- 구글 colaboratory 서비스의 줄임말로 브라우저에서 python을 작성하고 실행 가능한 플랫폼이다.
- 별도의 pythoon 설치가 필요 없고
- 데이터 분석 사용되는 Tensor Flow, Keras, matplotlib, scikit-learn, panda와 같은 패키지가 기본적으로 설치되어 있고
- GPU를 무료로 사용 가능하고
- Jupyter 노트북과 비슷하지만 더 좋은 기능을 제공하고
- 깃과 연동이 가능하여 사람들과 협업하여 코딩이 가능하다.

```
!pip install tensorflow==2.1.0                    #텐서플로 버전 변경
# ! 문자를 사용하면 다음에 나오는 셀 명령을 실행할 수 있음
# !pip install –U tensorflow                      #최신 버전 업데이트
# 텐서플로 업데이트후에는 [RESTART RUNTIME] 버튼을 클랙해 콜랩 런타임을 다시 시작해야 함
```

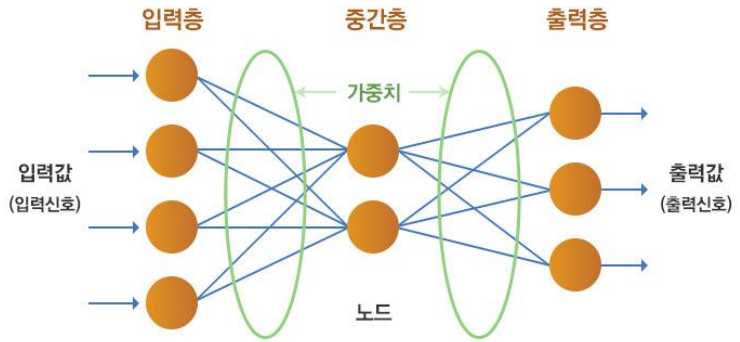
1.5 텐서플로 사용하기

머신러닝 시작하기

X = -1, 0, 1, 2, 3, 4
Y = -3, -1, 1, 3, 5, 7

1.5 텐서플로 사용하기

머신러닝 시작하기




```
import tensorflow as tf
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

dense = Dense(units=1, input_shape=[1])
model = Sequential([dense])
```

텐서플로를 사용할 때 Sequential 클래스로 층을 정의합니다. Sequential 클래스 안에서 층의 형태를 지정합니다. Sequential 클래스 안에 한 줄의 코드만 있으므로 하나의 층만 가집니다.

많은 종류의 층이 있지만 여기에서는 Dense 층을 사용합니다. 'Dense'는 뉴런이 완전히 연결되어 있다는 것을 의미합니다. Dense 층을 units=1로 설정했기 때문에 이 층은 1개의 뉴런을 가집니다.
마지막으로 신경망에 등장하는 첫 번째 층을 정의할 때 입력 데이터의 크기를 지정합니다. 이 경우 입력 데이터는 X이고, 숫자 하나이므로 [1]로 지정합니다.

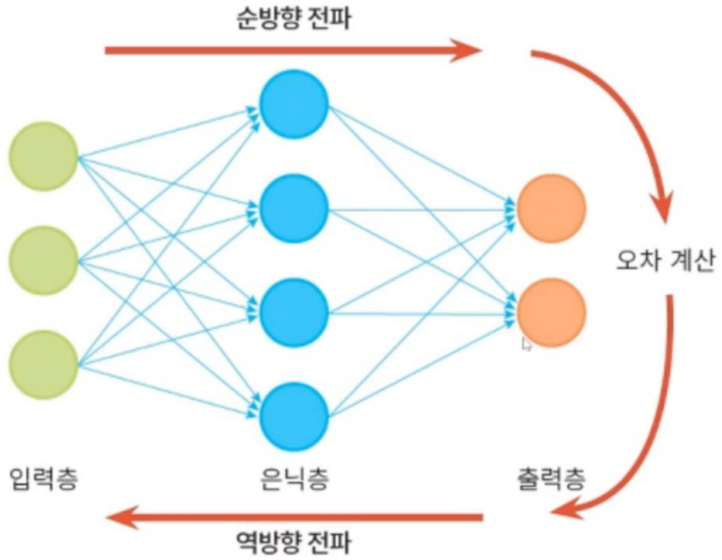
1.5 텐서플로 사용하기

머신러닝 시작하기

```
model.compile(optimizer='sgd', loss='mean_squared_error')
```

컴퓨터는 X와 Y 사이의 관계를 모르기 때문에 추측을 시작합니다. 예를 들어 Y = 10X + 10으로 가정하고 그 다음 이 추측이 얼마나 좋고 나쁜지를 측정합니다. 이를 담당하는 것이 손실 함수(loss function)입니다.
X다 -1, 0 , 1, 2, 3, 4일 때 정답을 알고 있으므로 손실함수는 추측으로 만든 답과 비교합니다. Y = 10X + 10으로 추측했다면 X가 -1일 때 Y는 0이 됩니다. 정답은 -3이므로 정답과 약간 벗어났습니다. 하지만 X가 4일 때 추측한 답은 50이고 정답은 7입니다. 이 경우에는 차이가 크네요.
이 정보를 바탕으로 컴퓨터는 다시 추측을 시작합니다. 이를 담당하는 것이 옵티마이저(optimizer)입니다. 이 경우 sgd라는 옵티마이저를 선택했습니다. 확률적 경사하강법(Stochastic Gradient Descent)의 약자입니다. 추측한 값에서 오차 과정을 반복해 손실을 최소화하며 이를 통해 추측한 식이 정답에 점점 더 가깝도록 만듭니다.

1.5 텐서플로 사용하기



머신러닝 시작하기

```
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
```

파이썬에서 텐서플로는 넘파이라는 라이브러리를 사용하므로 X와 Y 숫자를 넘파이 배열로 만듭니다.

```
model.fit(xs, ys, epochs=500)
```

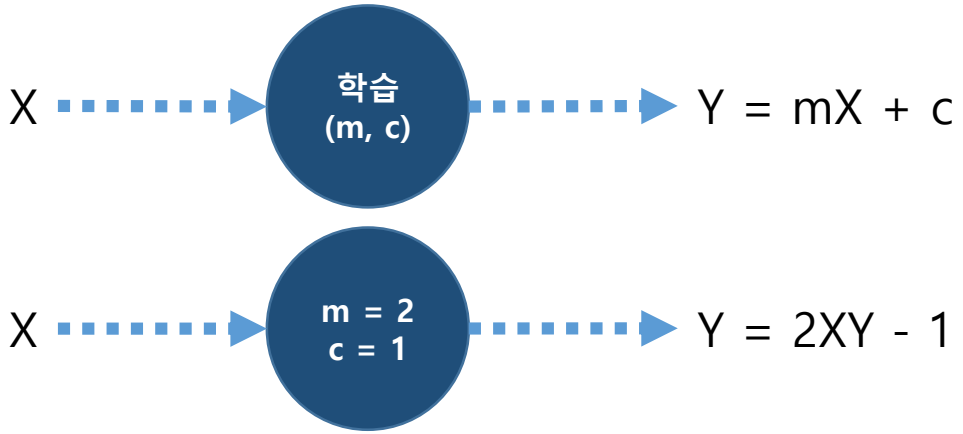
훈련 과정은 model.fit 명령으로 시작됩니다. 이 코드는 'X와 Y를 사용해 훈련하고 "500번 반복하라 " 라고 읽을 수 있습니다. 첫 번째 반복에서 컴퓨터는 관계(예를 들면 Y = 10X + 10)를 추측하고 이 추측이 얼마나 좋고 나쁜지 측정합니다. 그 다음 이 결과를 옵티마이저에 피드백해서 새로운 추측을 생성합니다. 손실(또는 오차)가 시간이 지남에 따라 줄어드는 로직을 사용해 이 과정을 반복합니다. 결과적으로 이 추측은 점점 더 좋아집니다.

1.5 텐서플로 사용하기

머신러닝 시작하기

```
print(model.predict([10.0]))
```

훈련된 모델을 사용해 예측을 실행합니다.
이 모델은 하나의 뉴런만 가지고 있습니다. 뉴런은 가중치(weight)와 절편(bias)을 학습합니다. 따라서 뉴런은 Y = WX + B와 같이 나타낼 수 있습니다. 이는 Y = 2X – 1 관계와 정확히 같으므로 W = 2, B = -1을 학습하면 됩니다. 모델이 여섯 개의 데이터로만 훈련했기 때문에 예측이 정답과 정확히 맞지는 않지만 정답에 매우 가까울 것입니다.
하지만 신경망이 처음에는 랜덤하게 초기화되기 때문에 여러분이 얻은 값은 조금 다를 수 있습니다. 즉 신경망의 초기 추측이 각자 조금씩 다를 것입니다.



1.5 텐서플로 사용하기

신경망이 학습한 것 확인하기

```
print("신경망이 학습한 것: {}".format(dense.get_weights()))
print("신경망이 학습한 것: {}".format(model.layers[0].get_weights()))
```

이 예제는 선형 관계에 있는 X와 Y를 매핑하는 매우 간단한 예제입니다. 이 뉴런은 학습한 가중치와 절편을 갖습니다. 따라서 하나의 뉴런이 Y = 2X – 1같은 관계를 학습하기에 충분합니다. 여기에서 가중치는 2이고 절편은 -1입니다. 학습이 끝난 후 이 층이 학습한 값(가중치)을 출력할 수 있습니다.
기대했던 Y = 2X – 1과 매우 비슷합니다. 다른 값이 나오더라도 이런 관계가 유지된다고 가정하면 실제와 아주 가까운 값이라고 생각할 수 있습니다.

1.5 텐서플로 사용하기

전체 코드

```
import tensorflow as tf
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

dense = Dense(units=1, input_shape=[1])
model = Sequential([dense])

model.compile(optimizer='sgd', loss='mean_squared_error')
```

```
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))

print("신경망이 학습한 것: {}".format(dense.get_weights()))
print("신경망이 학습한 것: {}".format(model.layers[0].get_weights()))
```

Reference

개발자를 위한 머신러닝&딥러닝, Laurence Moroney, 박해선, 한빛미디어

외 본문 링크