희귀암의 약물치료 의사결정 AI 모델 개발

주제: 데이터 기반 최적의 치료를 선택해주는 의사 결정 AI를 개발하는 것을 목표로 합니다.

DIGITAL HEALTH Hackathon 대회
SAIHST(삼성융합의과학원) 디지털헬스학과

DHH 디지털헬스해커톤 2020

Work Team Name & Members

CURED & 양형조, 김리나, 송용단

Work Schedule

2020.08.11 데이터셋 공개
2020.08.15 안내사항 수정
2020.08.19 Test Data 및 Scoreboard 공개
2020.09.07 결과물 제출

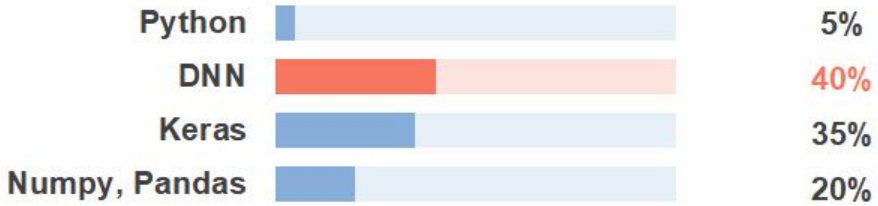
Work Rule

Data Set : Training, Test Data

Medical AI Chanllenge

- 의료 빅데이터는 이러한 치료 결과에 대한 대량의 정보를 제공할 수 있기 때문에, 이를 이용하면 환자 개인의 특성에 따른 맞춤형 치료가 가능할 것으로 기대하고 있습니다.
- 과거 데이터를 이용하여 치료가 적합했던 환자와 적합하지 않았던 환자를 구별해 내고, 앞으로 치료를 진행하면 경과가 좋아질 사람을 찾아내어 의사에게 추천해 줄 수 있는 의사 결정 보조 인공지능을 만들어 보는 것 입니다.

Skills



희귀암 약물치료 의사결정 AI 모델 개발

CURED;
양형조, 김리나, 송용단

목차

팀원 소개 및 역할

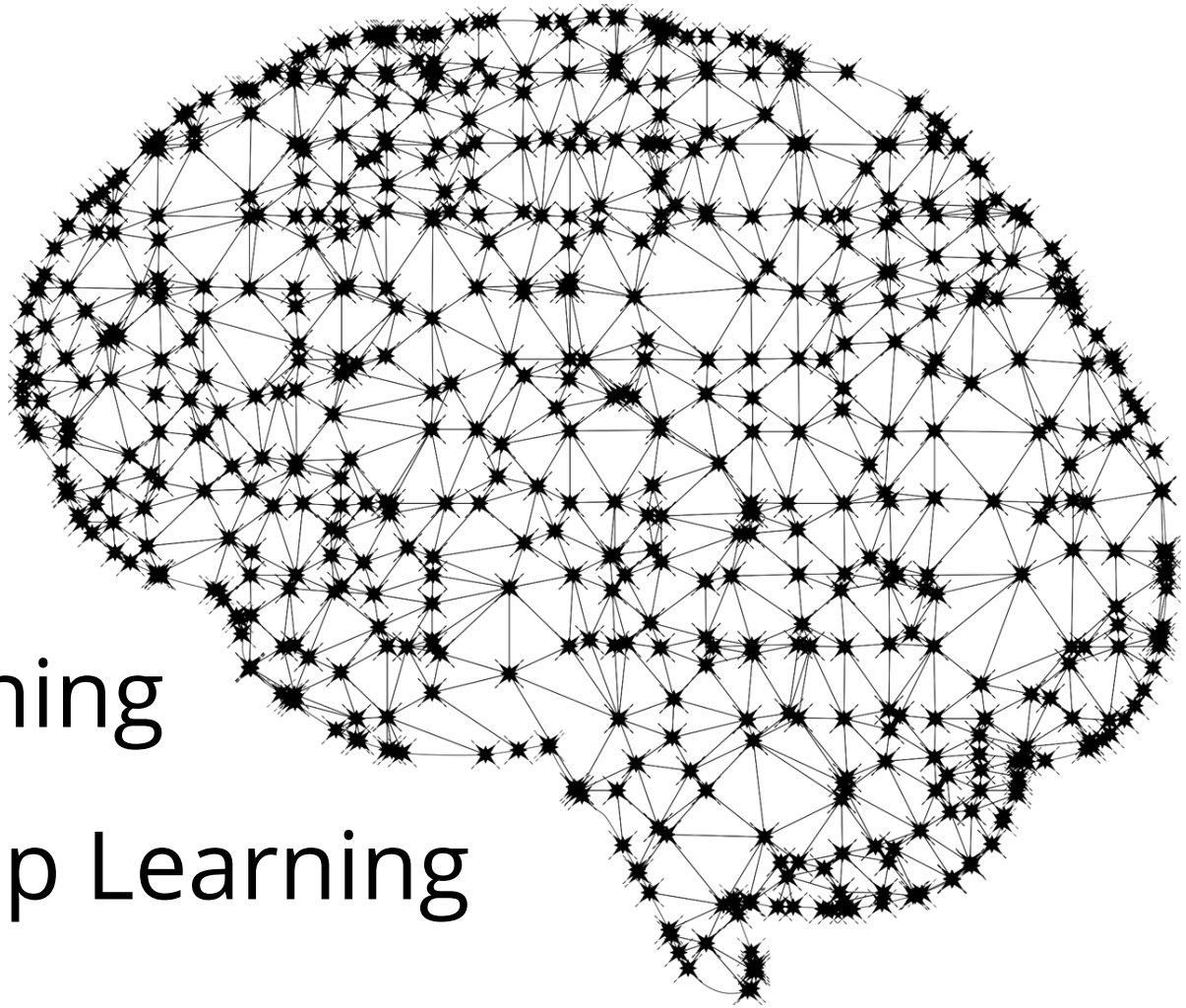
양형조	김리나	송용단
소프트웨어 아키텍처 설계	데이터 전처리	데이터 전처리
데이터 전처리	의사 결정 AI 모델 설계	의사 결정 AI 모델 설계
의사 결정 AI 모델 설계	발표 자료 준비	PPT 작성 및 발표

KOSTA EDU

AI

Machine Learning

Deep Learning



DIGITAL
HEALTH
HACKATHON
2020

삼성융합의과학원 개원 10주년 기념

Since 2016

제 5 회

DHH 2020 디지털헬스해커톤

변화된 의료의 새로운 시작을 꿈꾸다

비즈니스 트랙 안내사항

AI 트랙 팀 안내사항

AI 트랙 테스트 결과 제출

AI 트랙 스코어 보드 확인

프로젝트 소개

- 희귀암 약물 치료 의사 결정 AI 모델
  - 약물 치료의 효과는 환자 개인의 특성에 따라 다릅니다.
  - 의료 빅데이터는 약물 치료 결과에 대한 대량의 정보를 제공합니다.
  - 환자들의 과거 데이터를 통해 치료에 적합한 사람과 적합하지 않은 사람을 구분합니다.
  - CURED는 의사에게 치료가 적합한 사람과 적합하지 않은 사람을 구별할 수 있도록 돕는 의사 결정 보조 소프트웨어입니다.

대회규칙 - 훈련 데이터

- trainX.csv
  - 0 ~ 1 사이의 continuous data이며, 환자들의 상태를 표현합니다.
  - X0 ~ X16까지의 column, 17개의 변수로 구성되며 이는 환자의 개인별 상태를 반영합니다.
  - 이 중 일부는 교란 변수로 존재하여 치료에 영향을 주는 동시에 환자의 최종 생존 기간 (Y)에 영향을 줍니다.
- trainY.csv
  - 개별 환자의 최종 생존 기간을 나타내며 time, event의 2개의 column이 있습니다.
  - 최종 생존 기간은 의료진이 관심 있는 이 치료의 최종 목표이며 적절한 환자에게 치료한 경우 생존기간이 연장될 수 있지만, 어떤 환자에게 이 치료법은 생존기간을 단축시킵니다.
  - time은 음수가 아닌 continuous data로서 환자의 생존 시간입니다.
  - event는 binary data로서 1은 환자의 event발생, 0은 censored 입니다. (본 데이터에서는 분석의 편의상 censored data는 없는 것으로 하였습니다.)

대회규칙 - 훈련 데이터

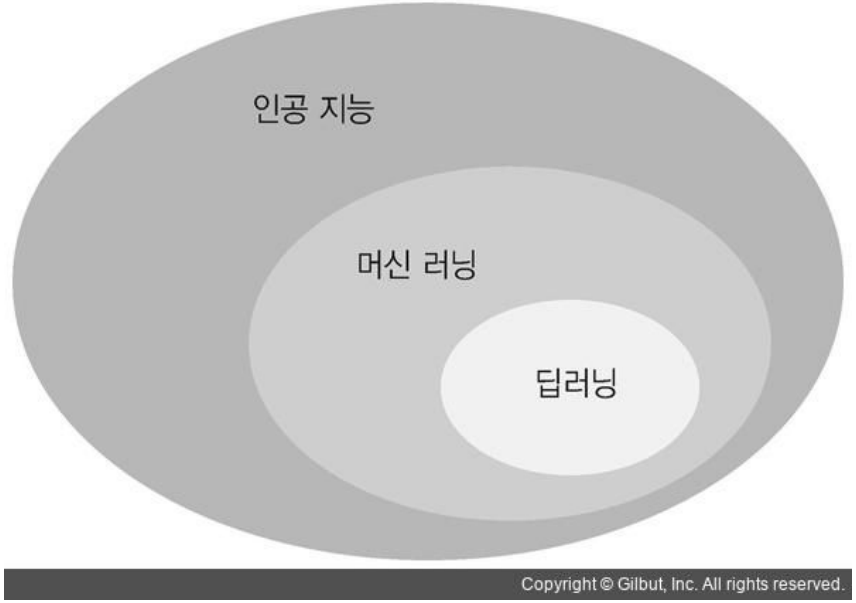
- trainA.csv
  - binary data이며, treatment (T) 즉 치료 여부를 의미합니다.
  - 과거에 어떻게 치료하였는가에 대하여 표현합니다.(이것이 정답을 뜻하진 않습니다. 잘 치료된 경우도 있고, 잘 못해서 생존이 오히려 줄어든 경우도 있을 겁니다.)
  - 1은 치료한다, 0은 치료하지 않는다를 뜻합니다.

대회규칙 - 검증 데이터

- testX.csv
  - trainX.csv와 형식은 같은 test data입니다.

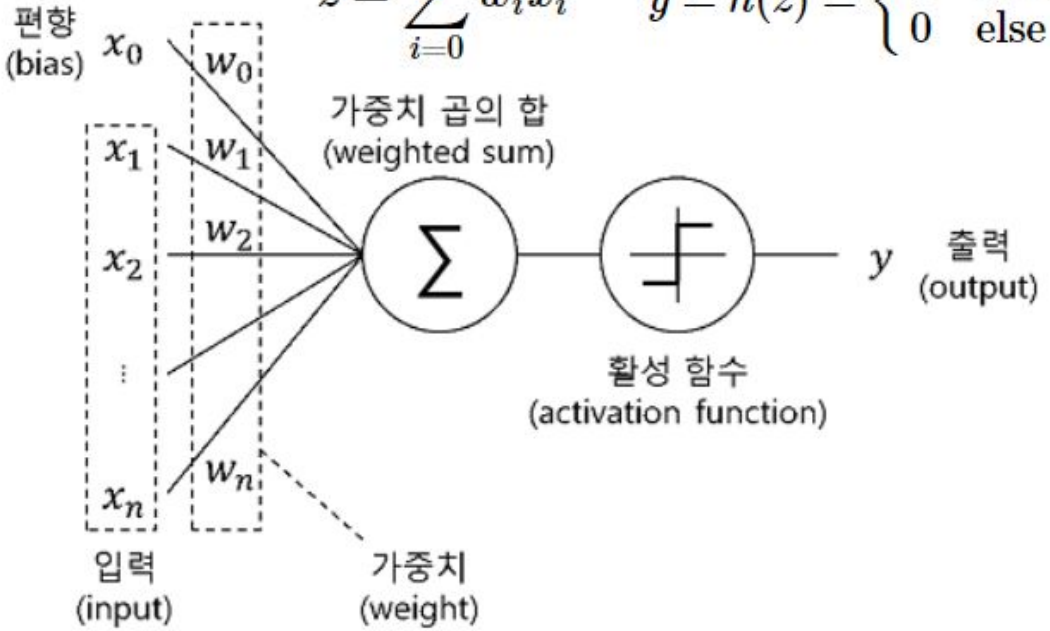
딥러닝

- 인공지능(artifical intelligence, AI) 실현 도구
- 머신러닝(machine learning, ML)의 한 종류
- 인공 신경망(artificial neural network, ANN)의 발전된 모델

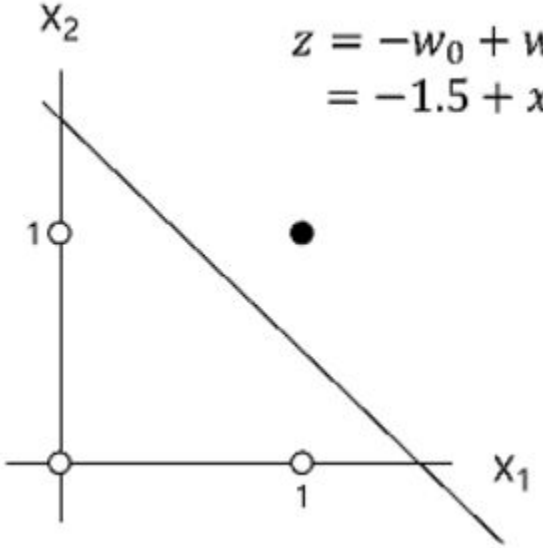


퍼셉트론(Perceptron)

$$z = \sum_{i=0}^{n} w_i x_i \qquad y = h(z) = \begin{cases} 1 & \text{if z} > \text{threshold} \\ 0 & \text{else} \end{cases}$$



퍼셉트론(Perceptron)



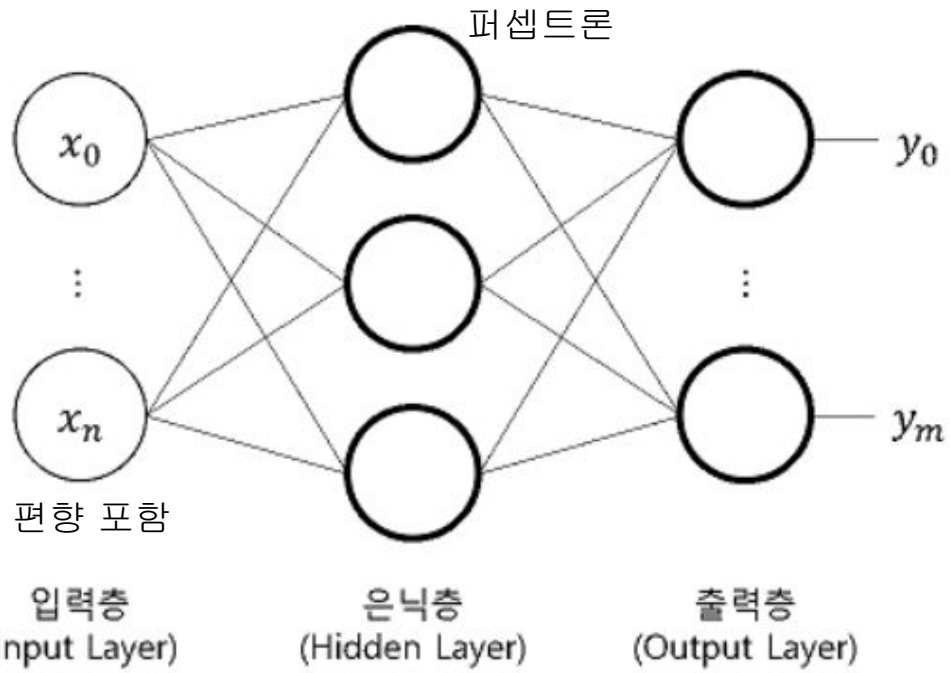
$$z = -w_0 + w_1 x_1 + w_2 x_2$$
$$= -1.5 + x_1 + x_2$$

$$h(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{그 외} \end{cases} = y$$

x_1	x_2	z	y
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

인공 신경망 Neural Network

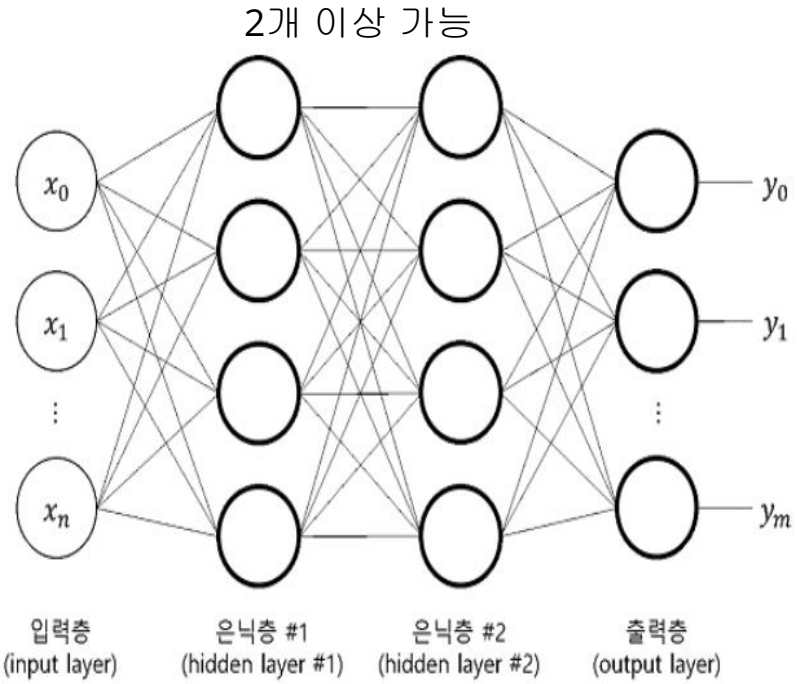
- 인공 신경망(artificial neural network)은 다층 퍼셉트론에서 조금 더 발전된 모델입니다.



인공 신경망의 구조

심층 신경망 Deep Neural Network

- 은닉층의 수를 제외하고는 인공 신경망과 같습니다.
- 은닉층 수를 많이 쌓음으로써 인공 신경망이 비선형 문제를 좀 더 잘 학습할 가능성이 높아집니다. 그러나 은닉층이 많아지면 학습에 필요한 컴퓨팅 비용이 높아지고 사람이 학습을 추적하기가 매우 어려워집니다.
- 인공 신경망에서 사용하는 활성화 함수들을 그대로 적용할 수 있습니다.
- 딥러닝(deep learning)은 이러한 심층 신경망을 이용한 머신러닝을 의미한다고 볼 수 있습니다.

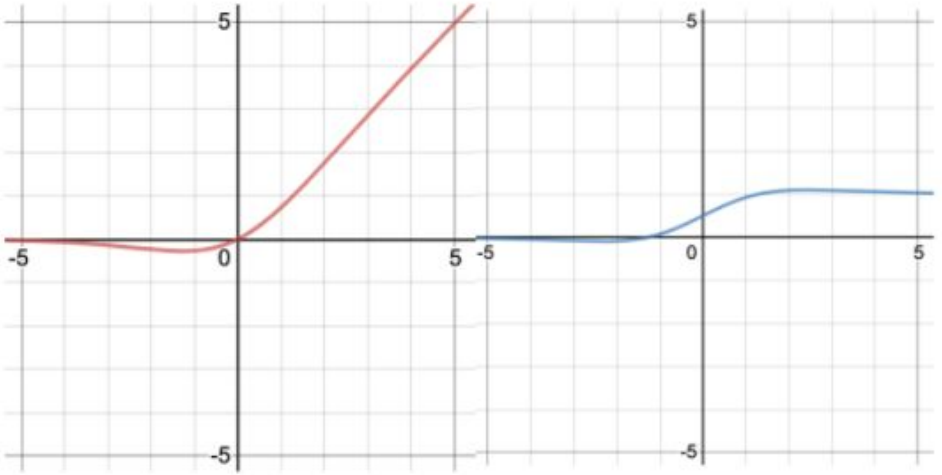


심층 신경망의 구조

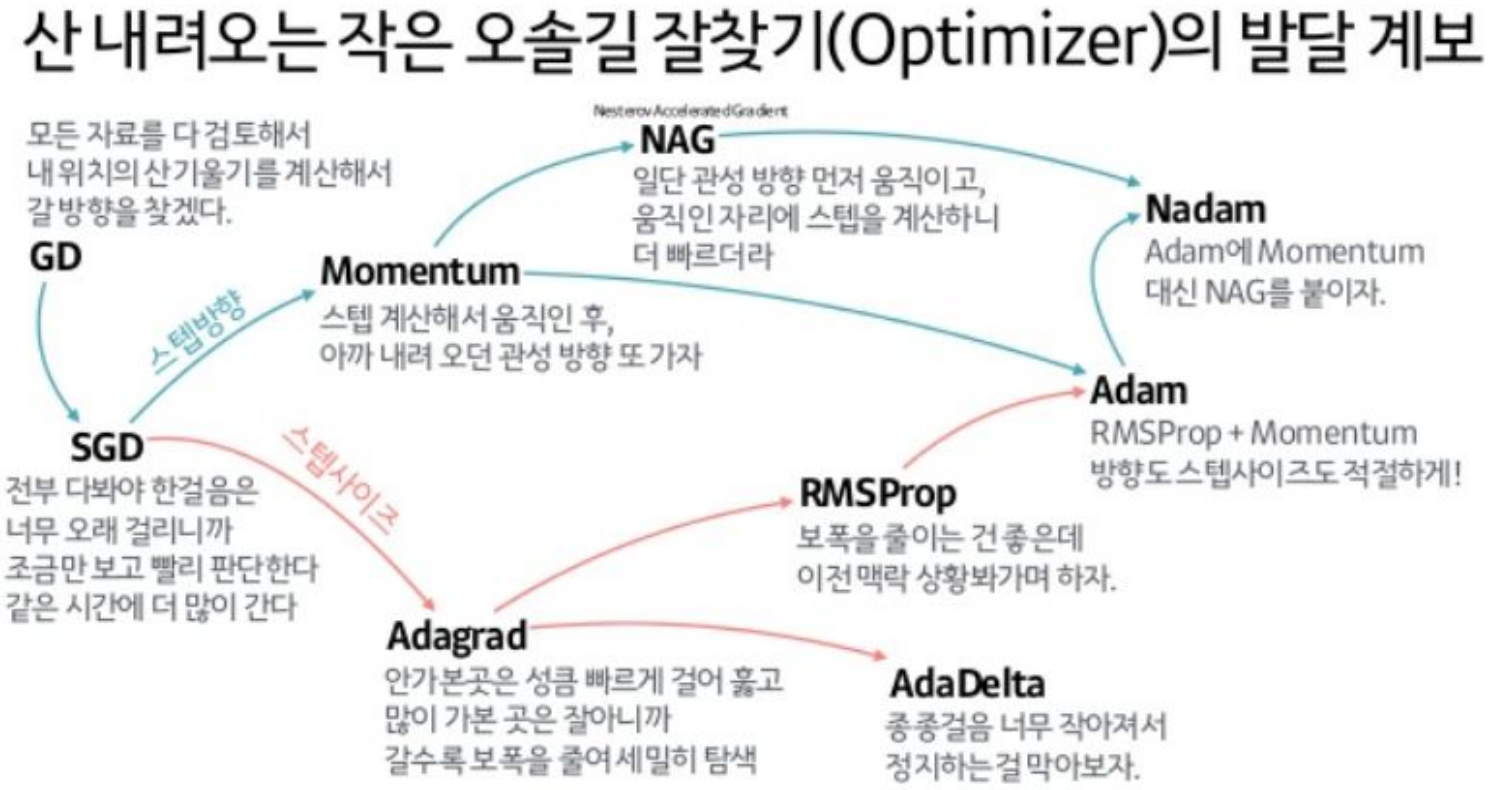
Swish 활성화 함수

- ReLU를 대체하기 위해 구글이 고안한 함수입니다.
- 시그모이드 함수에 x를 곱한 아주 간단한 형태를 보입니다.
- 깊은 레이어를 학습시킬 때 ReLU보다 더 뛰어난 성능을 보여준다고 합니다.

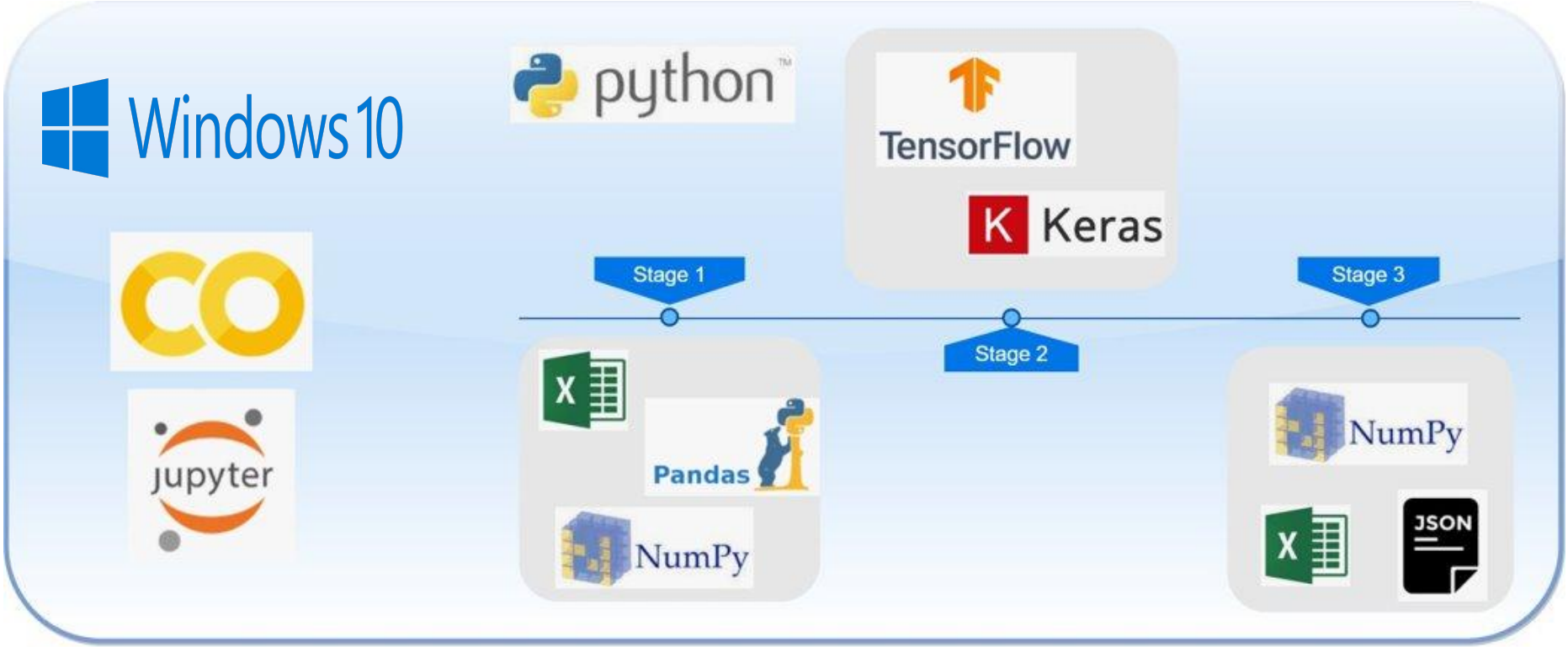
$$f(x)=\frac{1}{1+e^{-x}}x \qquad f'(x)=f(x)+sigmoid(x)(1-f(x))$$



Adam 최적화 함수


산 내려오는 작은 오솔길 잘찾기(Optimizer)의 발달 계보
모든 자료를 다 검토해서
내 위치의 산기울기를 계산해서
갈 방향을 찾겠다.
GD
스텝방향
SGD
전부 다봐야 한걸음은
너무 오래 걸리니까
조금만 보고 빨리 판단한다
같은 시간에 더 많이 간다
Momentum
스텝 계산해서 움직인 후,
아까 내려 오던 관성 방향 또 가자
Nesterov Accelerated Gradient
NAG
일단 관성 방향 먼저 움직이고,
움직인 자리에 스텝을 계산하니
더 빠르더라
Nadam
Adam에 Momentum
대신 NAG를 붙이자.
스텝사이즈
Adagrad
안가본곳은 성큼 빠르게 걸어 훑고
많이 가본 곳은 잘아니까
갈수록 보폭을 줄여 세밀히 탐색
RMSProp
보폭을 줄이는 건 좋은데
이전 맥락 상황봐가며 하자.
Adam
RMSProp + Momentum
방향도 스텝사이즈도 적절하게!
AdaDelta
종종걸음 너무 작아져서
정지하는걸 막아보자.

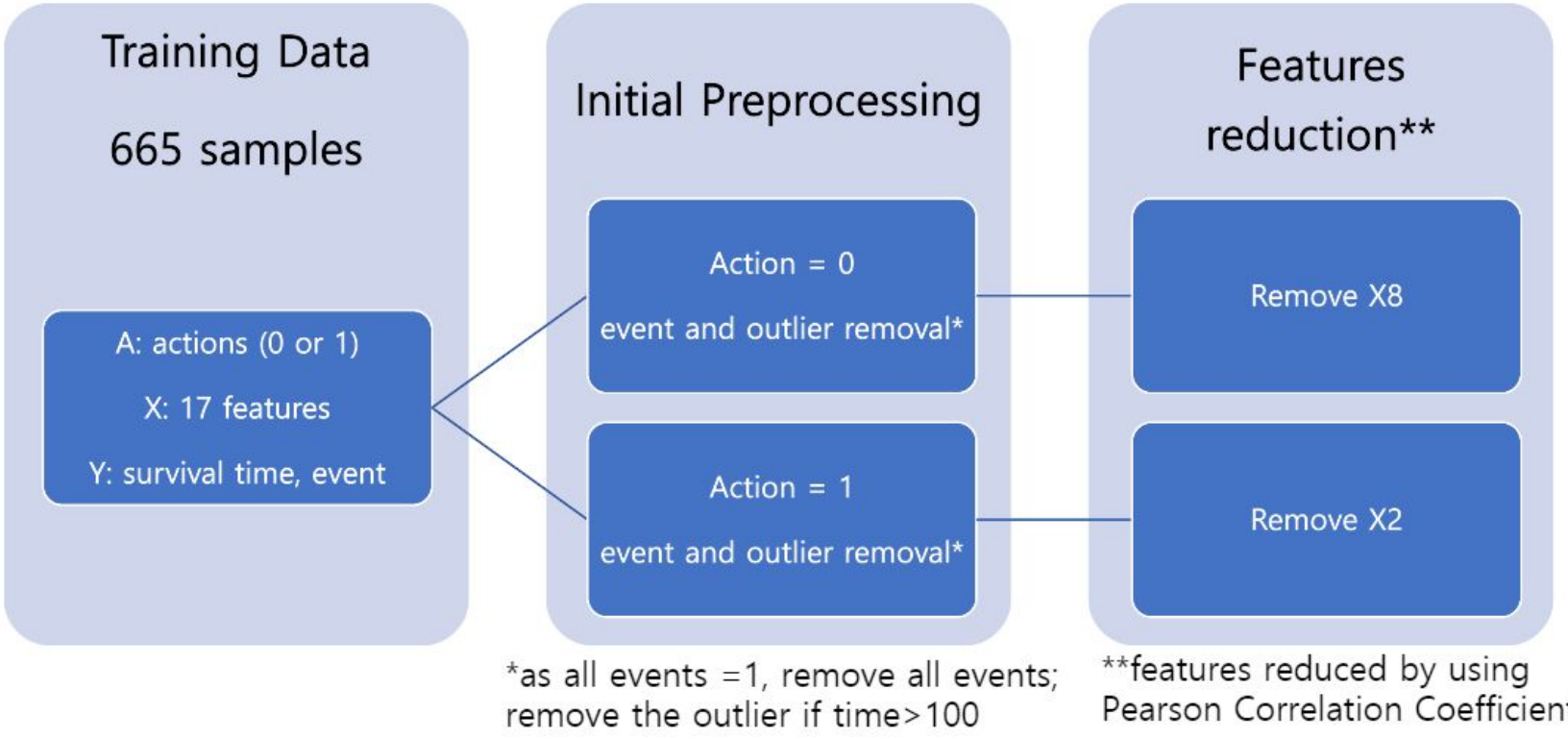
Tech Stack


Windows 10
python
TensorFlow
K Keras
CO
jupyter
Stage 1
Stage 2
Stage 3
Pandas
NumPy
NumPy
JSON

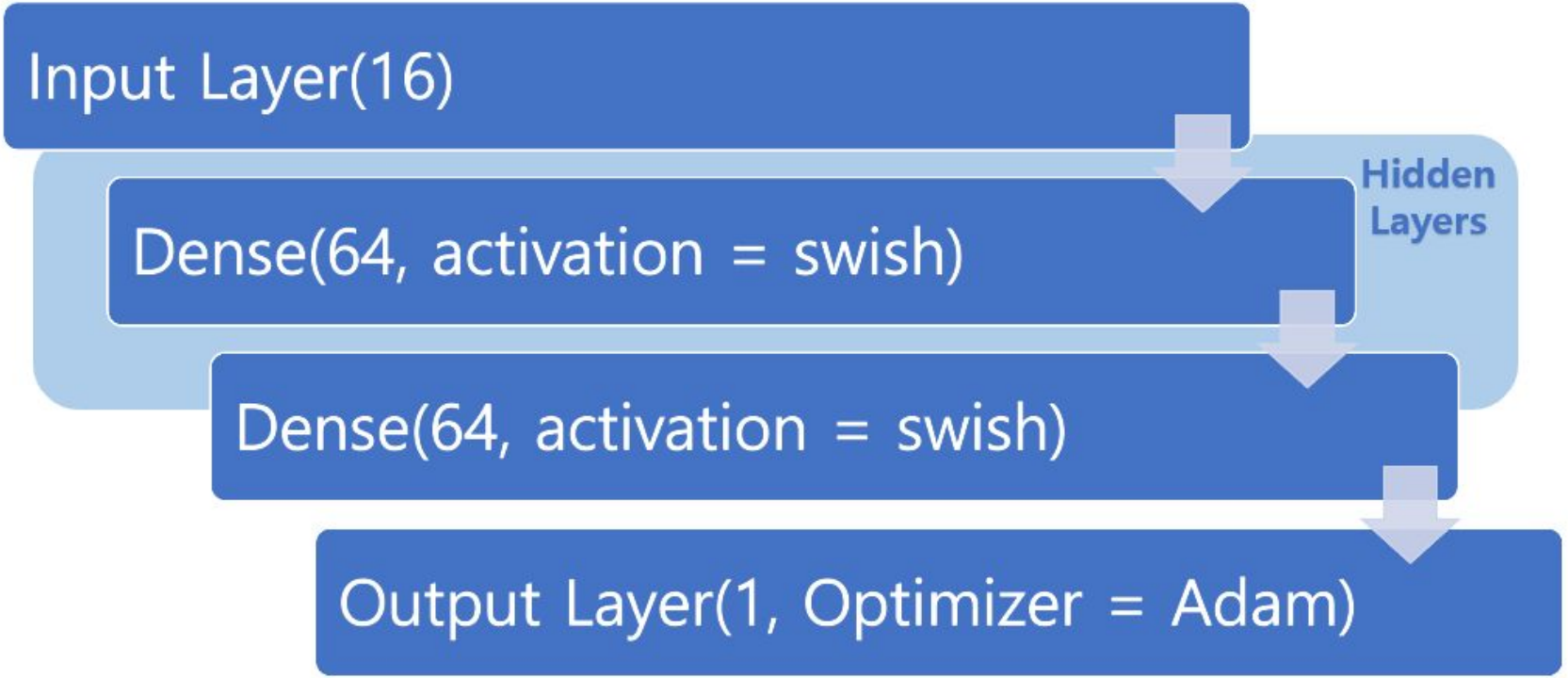
프로젝트 진행 과정

- Data preprocessing
- DNN modeling, training, prediction
- Decision making

Data Preprocessing



DNN Model

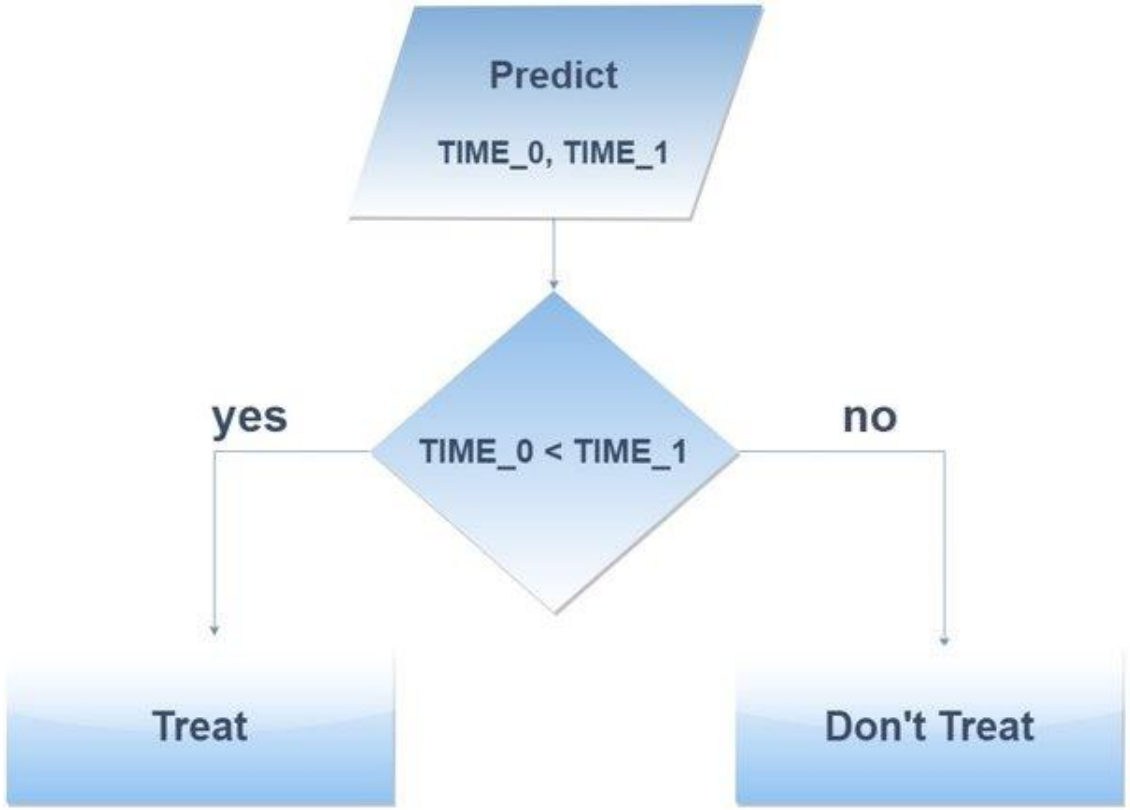

Input Layer(16)
Hidden Layers
Dense(64, activation = swish)
Dense(64, activation = swish)
Output Layer(1, Optimizer = Adam)

Training

```
 9 model = build_model_C_100_1024() # 최저 loss 값 : 0.0716
10 history = model.fit(train_data_C, train_targets_C, epochs=20000, verbose=0)
11 model.fit(train_data_C, train_targets_C, epochs=10)
```

```
Epoch 1/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1379 - accuracy: 0.0000e+00
Epoch 2/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1941 - accuracy: 0.0000e+00
Epoch 3/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1873 - accuracy: 0.0000e+00
Epoch 4/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1387 - accuracy: 0.0000e+00
Epoch 5/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1605 - accuracy: 0.0000e+00
Epoch 6/10
7/7 [==============================] - 0s 8ms/step - loss: 0.2319 - accuracy: 0.0000e+00
Epoch 7/10
7/7 [==============================] - 0s 7ms/step - loss: 0.0547 - accuracy: 0.0000e+00
Epoch 8/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1879 - accuracy: 0.0000e+00
Epoch 9/10
7/7 [==============================] - 0s 7ms/step - loss: 0.2005 - accuracy: 0.0000e+00
Epoch 10/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1759 - accuracy: 0.0000e+00
<tensorflow.python.keras.callbacks.History at 0x7f059c0f2ef0>
```

Prediction & Decision making algorithm



모델 연구과정

- 모델 수정
  - Hidden layer를 1개에서 20개까지 늘렸습니다.
  - Hidden layer의 dense 층의 유닛 개수를 4개에서 2048개까지 늘렸습니다.
  - 활성화 함수를 tanh, relu, swish를 모두 사용해본 결과 swish 함수를 사용했을 때 점수가 가장 잘 나왔습니다. 모델에 swish 함수를 적용했습니다.
  - 최적화 함수를 rmsprop, adam을 사용한 결과 Adam 함수를 사용했을 때 점수가 가장 잘 나왔습니다. 모델에 adam 함수를 적용했습니다.
  - 약물 치료 의사 결정 모델 스코어가 144에서 248로 향상되었습니다.

Demo - 모델 훈련

```
 9 model = build_model_C_100_1024() # 최저 loss 값 : 0.0716
10 history = model.fit(train_data_C, train_targets_C, epochs=20000, verbose=0)
11 model.fit(train_data_C, train_targets_C, epochs=10)
```

```
Epoch 1/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1379 - accuracy: 0.0000e+00
Epoch 2/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1941 - accuracy: 0.0000e+00
Epoch 3/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1873 - accuracy: 0.0000e+00
Epoch 4/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1387 - accuracy: 0.0000e+00
Epoch 5/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1605 - accuracy: 0.0000e+00
Epoch 6/10
7/7 [==============================] - 0s 8ms/step - loss: 0.2319 - accuracy: 0.0000e+00
Epoch 7/10
7/7 [==============================] - 0s 7ms/step - loss: 0.0547 - accuracy: 0.0000e+00
Epoch 8/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1879 - accuracy: 0.0000e+00
Epoch 9/10
7/7 [==============================] - 0s 7ms/step - loss: 0.2005 - accuracy: 0.0000e+00
Epoch 10/10
7/7 [==============================] - 0s 7ms/step - loss: 0.1759 - accuracy: 0.0000e+00
<tensorflow.python.keras.callbacks.History at 0x7f059c0f2ef0>
```

Demo - 치료 여부 예측

```
# 치료할 경우 생존 시간을 구한다.
cured = model.predict(치료_환자_데이터)
# 치료하지 않을 경우 생존 시간을 구한다.
non_cured = model_NC.predict(비치료_환자_데이터)

# 치료한 경우 생존 시간 > 치료하지 않은 경우 생존 시간 : 1,
# 치료한 경우 생존 시간 > 치료하지 않은 경우 생존 시간 : 0
result = np.where(cured > non_cured, 1, 0)
print(result)
```

```
[[1]
 [1]
 [1]
 [0]
 [1]
 [1]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
```

성과

- 2020년 9월 1일, 22개 팀들 중 9위
- 286명 환자 중 248명 환자의 치료 여부 판단을 정확하게 함

보완할 점

- 은닉층을 보완해서 치료여부 판단 점수를 248점에서 286점으로 올리겠습니다.

	현재	목표
치료 여부 판단 점수	248	286

참고문헌

- http://blog.quantylab.com/dl_overview.html
- http://blog.quantylab.com/stock_rl_mod_net.html
- https://yeomko.tistory.com/39
- https://www.slideshare.net/yongho/ss-79607172/49
- https://cdn.pixabay.com/photo/2017/09/08/19/07/a-2729794_1280.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Windows_10_Logo.svg/1920px-Windows_10_Logo.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/f/f8/Python_logo_and_wordmark.svg/1920px-Python_logo_and_wordmark.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/TensorFlowLogo.svg/1024px-TensorFlowLogo.svg.png
- https://www.google.co.kr/url?sa=i&url=https%3A%2F%2Ftensorflow.blog%2F%25EC%25BC%2580%25EB%259D%25BC%25EC%258A%25A4-%25EB%2594%25A5%25EB%259F%25AC%25EB%258B%259D%2F1-%25EB%2594%25A5%25EB%259F%25AC%25EB%258B%259D%25EC%259D%25B4%25EB%259E%2580-%25EB%25AC%25B4%25EC%2597%2587%25EC%259D%25B8%25EA%25B0%2580%2F&psig=AOvVaw3siUyZffmbJiVtgAYVZDnJ&ust=1599103947433000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCOCSxsnEyesCFQAAAAAdAAAAABAD
- https://img1.daumcdn.net/thumb/R720x0.q80/?scode=mtistory2&fname=http%3A%2F%2Fcfile7.uf.tistory.com%2Fimage%2F997E924F5CDBC1A6283C93
- http://blog.quantylab.com/dl_hist.html

경청해 주셔서 감사합니다!