



RNN 소개

RNN은 순환 신경망으로, 시계열 데이터 처리에 탁월하여 이전 정보를 기억하고 현재 정보를 처리하여 예측, 분류, 생성 등 다양한 작업을 수행



작성자: **sanggoo cho**

RNN은 시계열 데이터의 패턴을 학습할 수 있습니다. 따라서 주식 시장 예측, 음성 인식, 자연어 처리 등 다양한 분야에서 활용됩니다.

1 시계열 데이터

- 텍스트와 같이 입력의 크기가 변화하는 경우(Variable-Length Inputs)
- (Time-Series) 시간 순서대로 발생하는 데이터(주가, 언어, 노래 등)를 분석하는 데 적합

3 다양한 분야

- LSTM, GRU, Attention, Transformer(BERT, ChatGPT)등을 이해하는데 필수
- 주식 시장 예측, 음성 인식, 자연어 처리 등 다양한 분야에서 활용

2 패턴 학습

- 과거 데이터를 기반으로 미래를 예측(Past Recurrent Event)



RNN의 필요성

❖ 시간적 종속성 처리:

- ❖ RNN은 이전 단계의 정보를 기억하고 다음 단계에서 활용하여 시퀀스 간의 시간적 연관성을 반영할 수 있습니다. 반면 FNN은 이를 처리하지 못합니다.

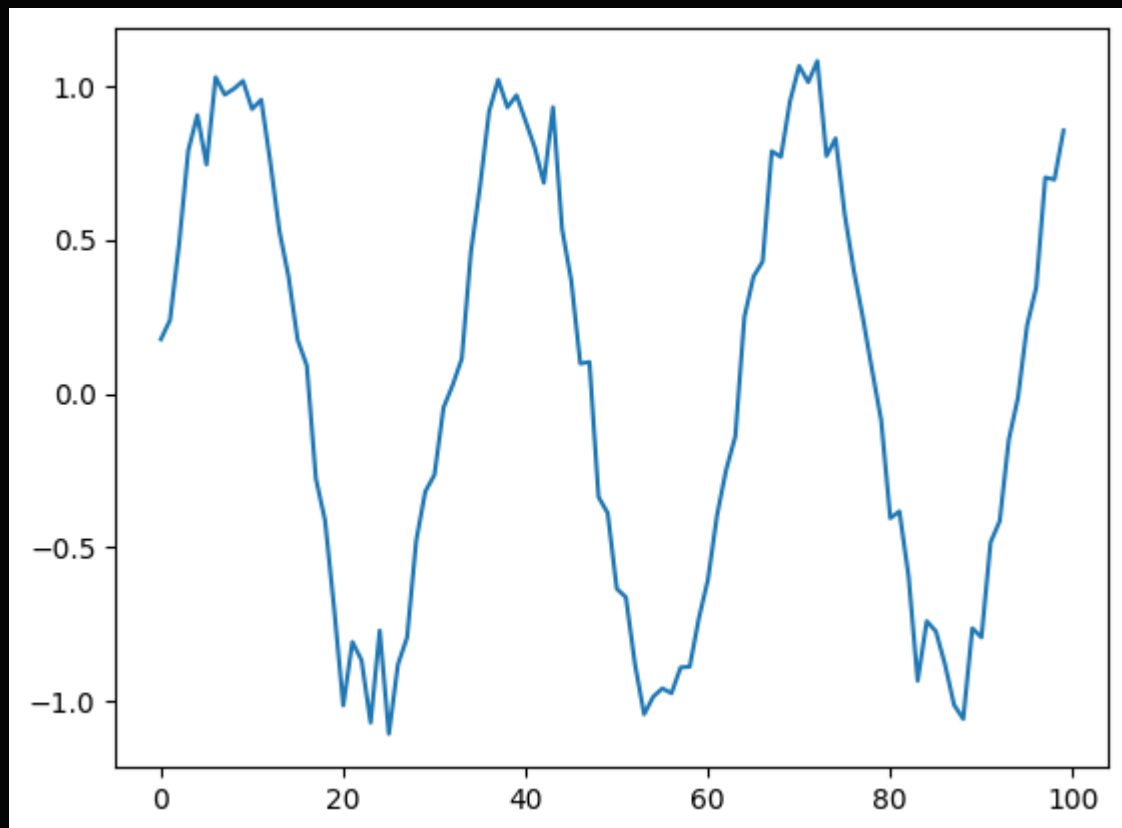
❖ 가변적 입력 길이 처리:

- ❖ RNN은 내부 상태를 이용해 가변적인 길이의 입력 시퀀스를 처리할 수 있어 다양한 길이의 데이터에 유연하게 대응합니다.

❖ 연속적 데이터 처리:

- ❖ RNN은 시계열 데이터나 이벤트 스트림처럼 순차적으로 변화하는 데이터를 다루는 데 강점을 가지며, FNN은 이러한 연속적 데이터를 효과적으로 처리하지 못합니다.

```
array([[ 0.17640523,  0.23868505,  0.48729214,  0.78873179,  0.90411189,
         0.7437432 ,  1.02704793,  0.97031401,  0.98925172,  1.01490748,
        0.92370178,  0.95392375,  0.75156695,  0.52766887,  0.37937447,
        0.17448744,  0.09103376, -0.27605693, -0.41121367, -0.69726746])
```



x

```
[[0.17640523 0.23868505 0.48729214 0.78873179 0.90411189]
 [0.23868505 0.48729214 0.78873179 0.90411189 0.7437432 ]
 [0.48729214 0.78873179 0.90411189 0.7437432  1.02704793]
 [0.78873179 0.90411189 0.7437432  1.02704793 0.97031401]
 [0.90411189 0.7437432  1.02704793 0.97031401 0.98925172]]
```

y

```
[0.7437432  1.02704793 0.97031401 0.98925172 1.01490748]
```

RNN 모델

- ❖ 입력 시퀀스의 길이(과거 5개 시계열자료)와 특징의 개수(1개, 1 tensor)를 지정
- ❖ RNN Hidden Layer 1개, 노드(unit)는 3개 뉴런으로 구성
- ❖ 출력층 1개, 출력층 뉴런 1개

```
model = Sequential()
model.add(SimpleRNN(units=3,
                    activation='tanh',
                    input_shape=(sequence_length, 1))) # 입력 형태 지정
model.add(Dense(units=1))
```

RNN 모델의 파라미터

RNN 모델의 파라미터 개수는 입력 크기, 은닉층 크기(뉴런), 출력 크기 등에 의해 결정

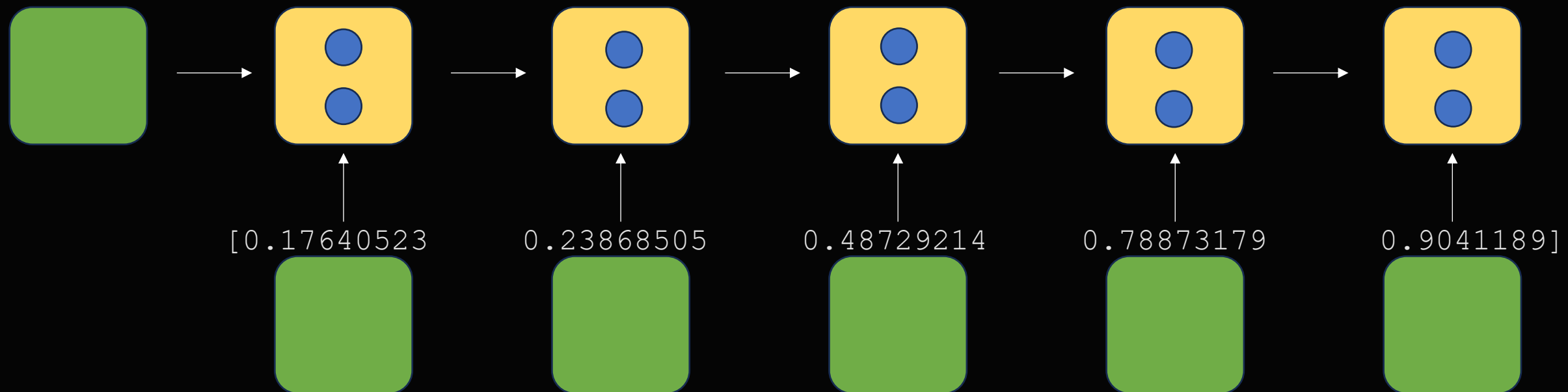
Model: "sequential"

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 3)	15
dense (Dense)	(None, 1)	4

Total params: 19 (76.00 B)

Trainable params: 19 (76.00 B)

Non-trainable params: 0 (0.00 B)



RNN 모델의 파라미터

RNN 모델의 파라미터 개수는 입력 크기, 은닉층 크기(뉴런), 출력 크기 등에 의해 결정

SimpleRNN 레이어의 파라미터:

1.입력 가중치 (W_{xh}): 크기 (1, 2)

1. 입력에서 은닉 상태로 가는 가중치 행렬
2. 총 가중치 수는 2개

2.은닉 가중치 (W_{hh}): 크기 (2, 2)

1. 이전 은닉 상태에서 현재 은닉 상태로 가는 가중치 행렬
2. 총 가중치 수는 4개

3.편향 (b_h): 크기 (2,)

1. 각 은닉 유닛마다 하나의 편향이 존재
2. 총 편향 수는 2개

SimpleRNN 레이어의 총 파라미터 수: $2 (W_{xh}) + 4 (W_{hh}) + 2 (b_h) = 8$ 개

Dense 레이어의 파라미터:

1.가중치 (W): 크기 (2, 1)

1. 은닉 유닛에서 출력으로 가는 가중치 행렬
2. 총 가중치 수는 2개

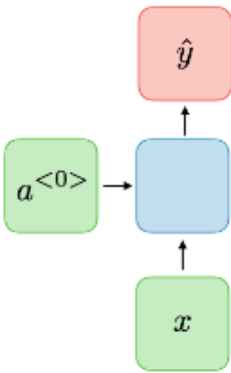
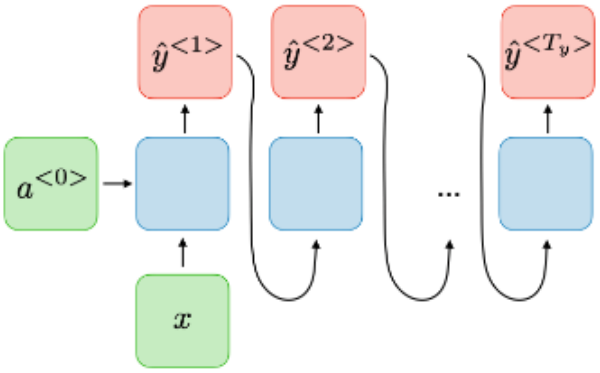
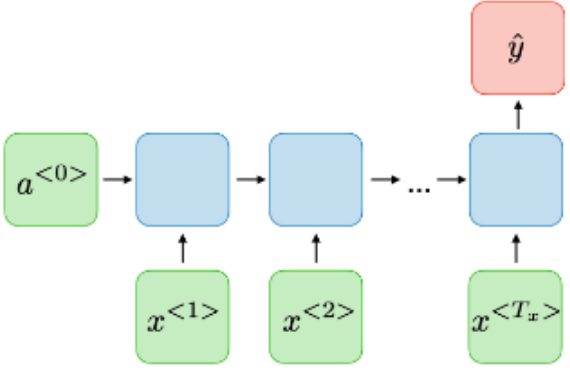
2.편향 (b): 크기 (1,)

1. 출력 유닛마다 하나의 편향이 존재
2. 총 편향 수는 1개

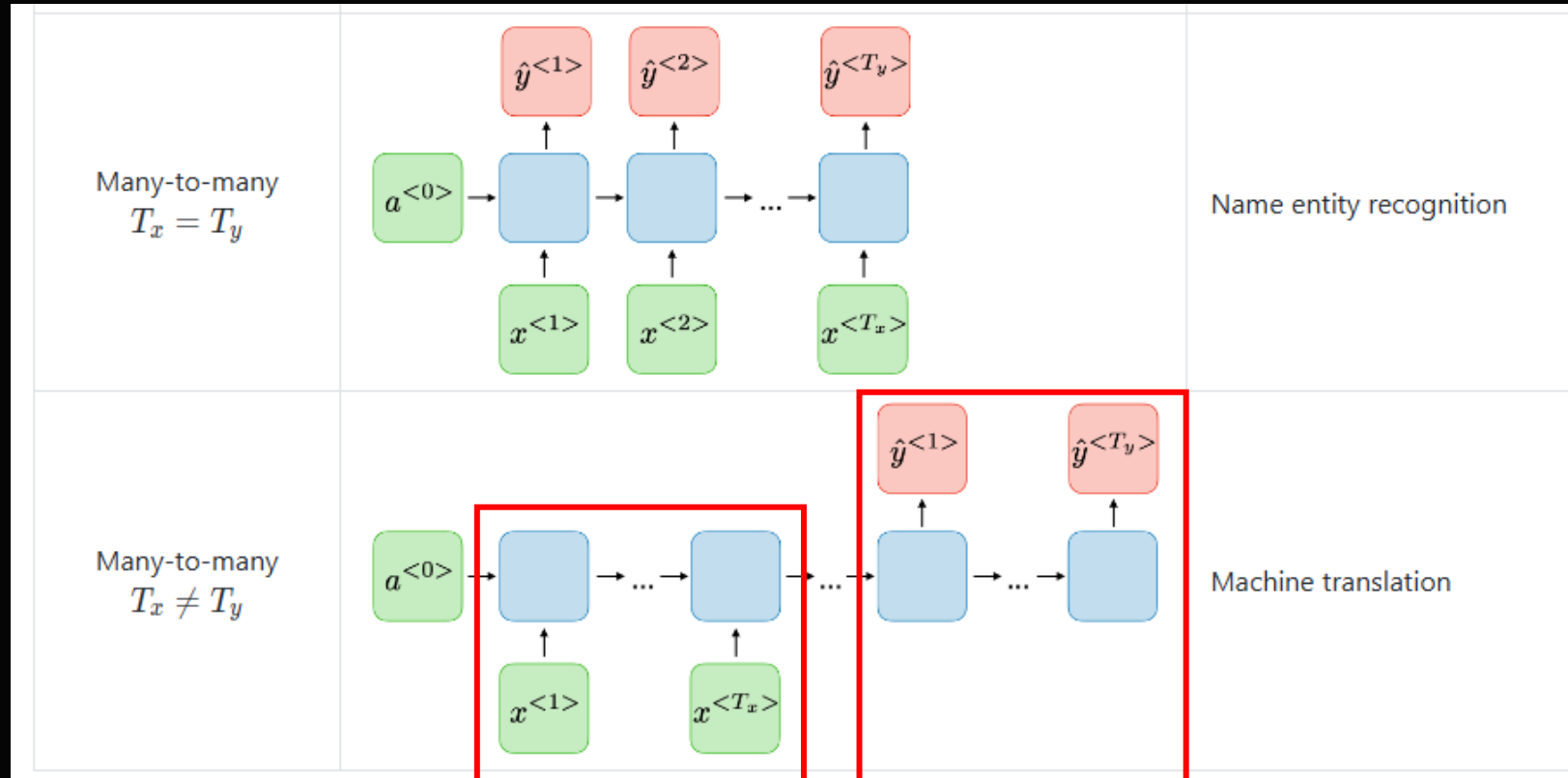
Dense 레이어의 총 파라미터 수: $2 (W) + 1 (b) = 3$ 개

RNN의 아키텍처 타입

□ **Applications of RNNs** — RNN models are mostly used in the fields of natural language processing and speech recognition. The different applications are summed up in the table below:

Type of RNN	Illustration	Example
One-to-one $T_x = T_y = 1$		Traditional neural network
One-to-many $T_x = 1, T_y > 1$		Music generation
Many-to-one $T_x > 1, T_y = 1$		Sentiment classification

RNN의 아키텍처 타입



Encoder

Decoder

영어

↔

한국어

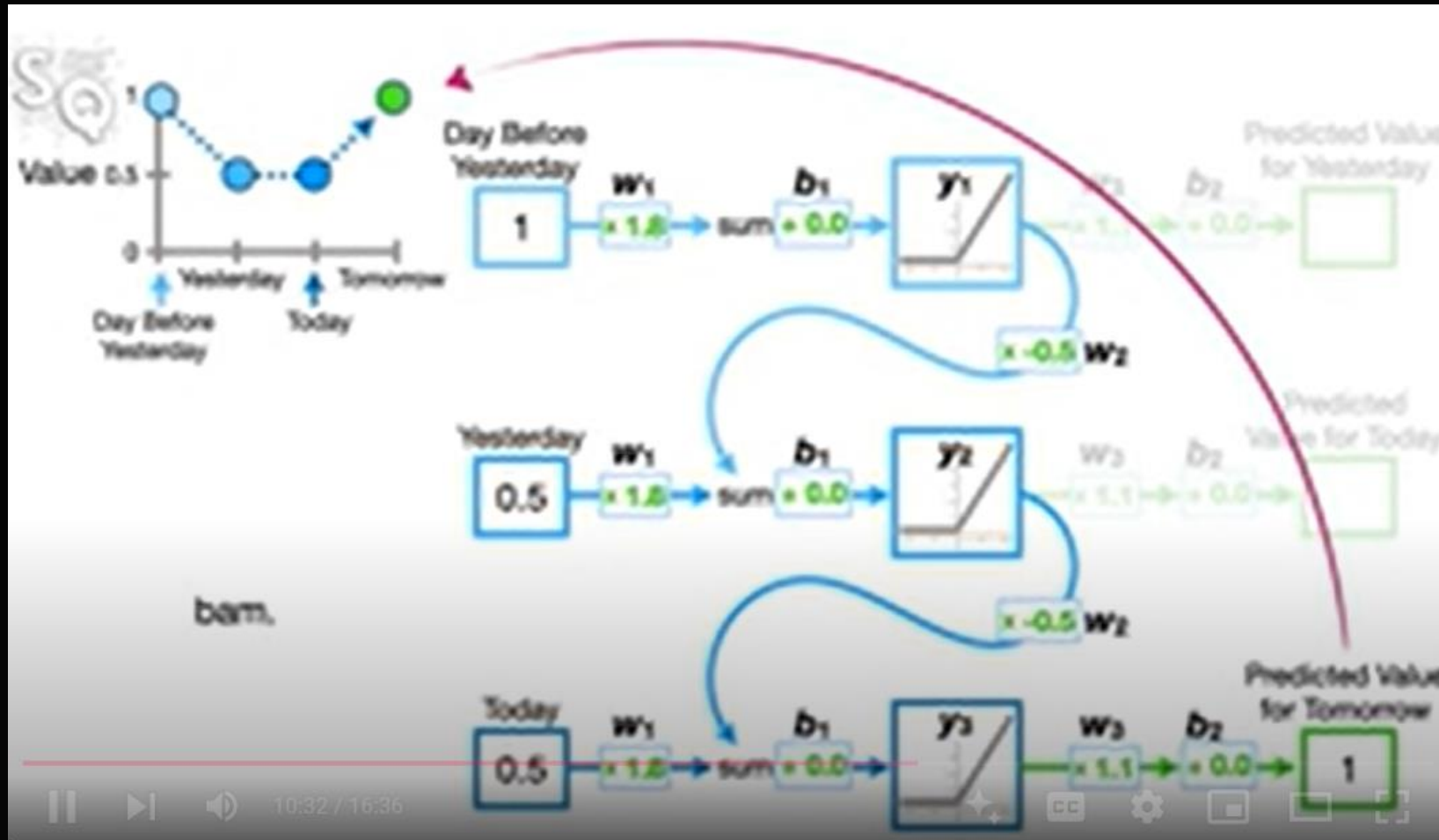
I am a boy

×

나는 소년이다
naneun sonyeon-ida

RNN 설명

- Weight와 Bias는 변하지 않으며 Time sequence에 유연하게 RNN 모델에 입력자료 투입이 가능
- Time Sequence의 Length 크기와 동일하게 Recurrence



<https://www.youtube.com/watch?v=AsNTP8Kwu80>

RNN 계열

RNN 아키텍처에는 다양한 종류가 있으며 각 아키텍처는 장단점이 다르며 특정 작업에 더 적합

Vanilla RNN

기본적인 RNN 모델로, 간단하고 빠르지만 장기 의존성을 학습하는 데 어려움때문에 거의 사용 안함
(Gradient Vanishing Problem)

LSTM

장기 의존성을 학습하는 데 뛰어나지만 계산량이 많음

GRU

LSTM보다 간단하고 효율적이며 장기 의존성을 학습하는 데 유용.

RNN 모델 훈련 과정

RNN 모델은 훈련 데이터를 입력하여 가중치(Weights)와 편향(Bias)을 학습하며 손실 함수(Loss Function)를 최소화하는 방향(Optimization)으로 파라미터를 업데이트

1

데이터 전처리

시계열 데이터를 모델에 적합하게 변환

2

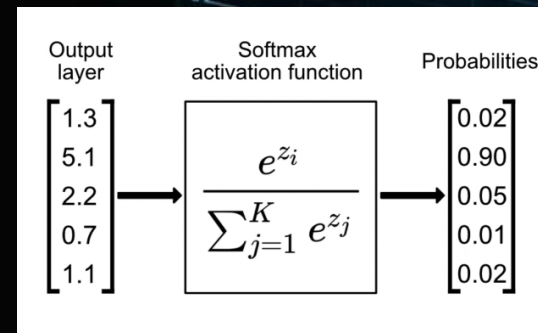
모델 훈련

훈련 데이터를 입력하여 모델의 파라미터를 학습

3

모델 평가

훈련된 모델의 성능을 평가



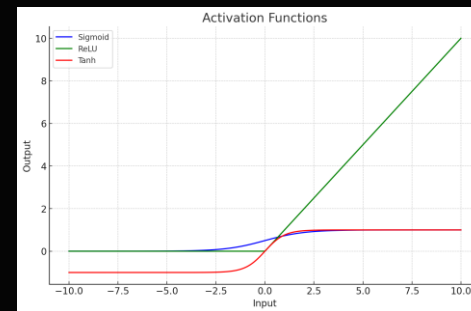
Loss Function:

$$L(y, \hat{y}) = -y \ln(\hat{y}) - (1 - y) \ln(1 - \hat{y})$$

Deep learning Terminology

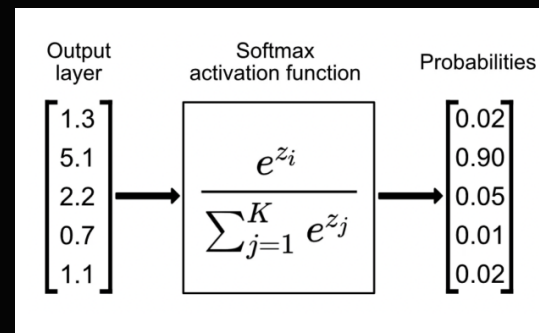
1

활성화함수(Activation Function)



2

Softmax Function

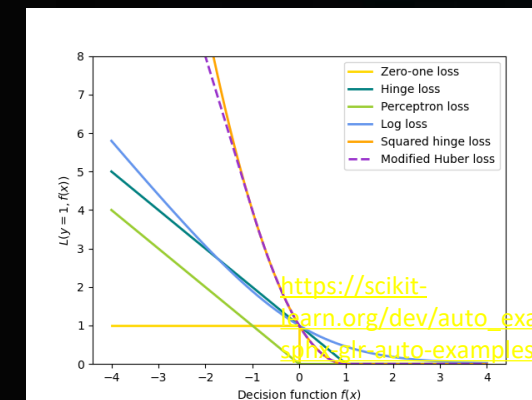


3

Loss Function : log loss

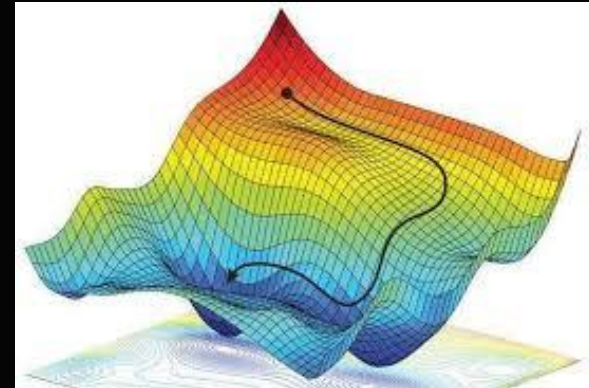
Loss Function:

$$L(y, \hat{y}) = -y \ln(\hat{y}) - (1 - y) \ln(1 - \hat{y})$$



https://scikit-learn.org/dev/auto_examples/linear_model/plot_sgd_loss_functions.html#sphxglr-auto-examples-linear-model-plot-sgd-loss-functions-py

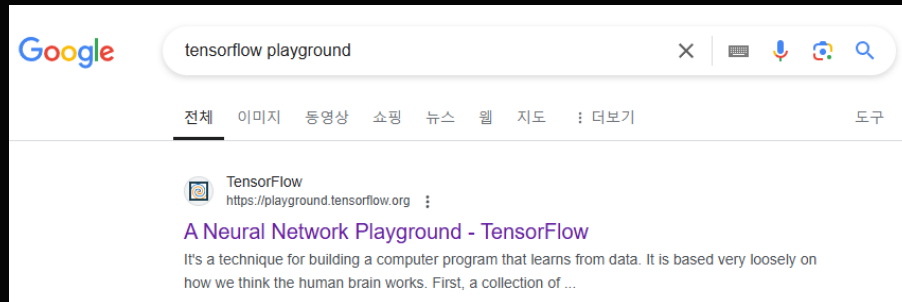
경사하강알고리즘(Stochastic Gradient Descent)



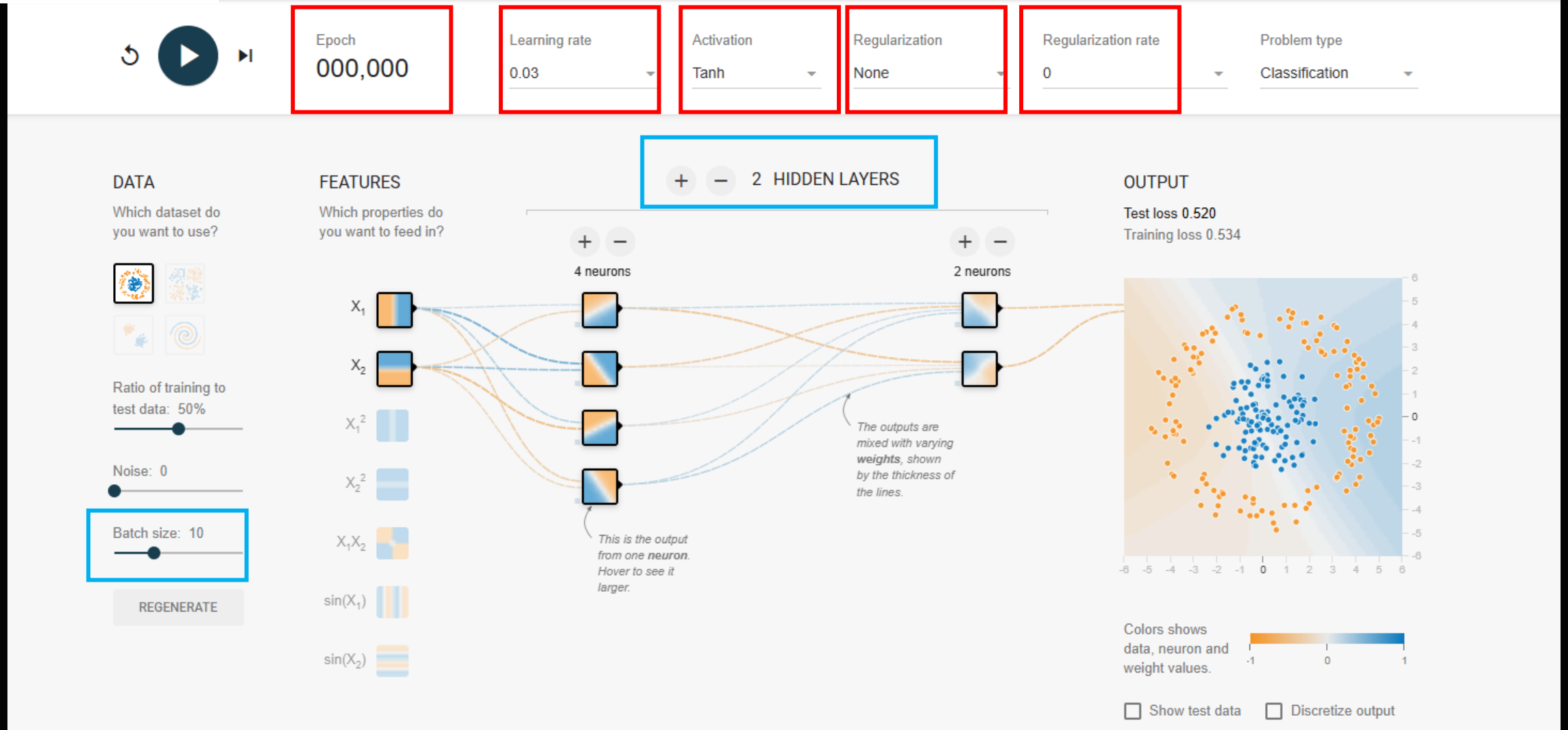
SUM : ✖ ✔ f_x =E3-\$A\$2*H3									
A	B	C	D	E	F	G	H	I	J
learning rate	Iterations	Data		model = y = a+bx			y_hat - y		
0.01		x	y	절편 (α)	기울기 (β)	y_hat	= error	square(error)	absolute error
1 epoch	0	1	1	0.0000	0.0000	0.0000	-1.0000	1.0000	1.0000
	1	2	=E3-\$A\$2*H3		.00	0.0300	-2.9700	8.8209	2.9700
	2	4	3	0.0397	0.0694	0.3173	-2.6827	7.1969	2.6827
	3	3	2	0.0665					
2 epoch	4	5	5	0.0806					
	5	1	1	0.1188					
	6	2	3	0.1235					
	7	4	3	0.1440					
	8	3	2	0.1543					

SUM : ✖ ✔ f_x =F3-\$A\$2*H3*C3									
A	B	C	D	E	F	G	H	I	J
learning rate	Iterations	Data		model = y = a+bx			y_hat - y		
0.01		x	y	절편 (α)	기울기 (β)	y_hat	= error	square(error)	absolute error
1 epoch	0	1	1	0.0000	0.0000	0.0000	-1.0000	1.0000	1.0000
	1	2	3	=F3-\$A\$2*H3*C3			-2.9700	8.8209	2.9700
	2	4	3	0.0397	0.0694	0.3173	-2.6827	7.1969	2.6827
	3	3	2	0.0665	0.1767	0.5967	-1.4033	1.9694	1.4033
2 epoch	4	5	5	0.0806	0.2188	1.1746	-3.8254	14.6337	3.8254
	5	1	1	0.1188	0.4101	0.5289	-0.4711	0.2219	0.4711
	6	2	3	0.1235	0.4148	0.9531	-2.0469	4.1898	2.0469
	7	4	3	0.1440	0.4557	1.9669	-1.0331	1.0673	1.0331
3 epoch	8	3	2	0.1543	0.4971	1.6455	-0.3545	0.1257	0.3545
	9	5	5	0.1579	0.5077	2.6963	-2.3037	5.3070	2.3037
	10	1	1	0.1809	0.6229	0.8038	-0.1962	0.0385	0.1962
	11	2	3	0.1829	0.6248	1.4325	-1.5675	2.4569	1.5675
	12	4	3	0.1985	0.6562	2.8233	-0.1767	0.0312	0.1767

Simulate and Understand Deep Learning



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



RNN RNN

RNN 모델 성능 평가

RNN 모델의 성능은 정확도, 정밀도, 재현율, F1 점수 등 다양한 지표로 평가하며 모델의 목표에 따라 적절한 지표를 선택

지표	설명
정확도	모델이 정확하게 예측한 비율
정밀도	모델이 예측한 결과 중 실제로 맞는 비율
재현율	실제로 맞는 결과 중 모델이 예측한 비율
F1 점수	정밀도와 재현율의 조화 평균



RNN의 응용 분야와 향후 전망

RNN은 자연어 처리, 음성 인식, 기계 번역, 이미지 캡셔닝 등 다양한 분야에서 활용
앞으로 더욱 발전하여 더욱 정교하고 효율적인 인공지능 시스템을 구축하는 데 기여



자연어 처리

챗봇, 감정 분석, 기계 번역 등 다양한 자연어 처리 작업에 활용



음성 인식

음성을 텍스트로 변환하는 음성 인식 시스템 개발에 활용



로봇 제어

로봇의 움직임을 제어하고 환경을 인식하는 데 활용



뇌 과학

뇌 활동 패턴을 분석하고 예측하는 데 활용