공학석사 학위논문

딥러닝을 활용한 저압막여과 공정의 적정 플럭스 도출

2021년 2월

부경대학교 대학원

토 목 공 학 과

김 누 리

공학석사 학위논문

딥러닝을 활용한 저압막여과 공정의 적정 플럭스 도출

지도교수 김 수 한

이 논문을 공학석사 학위논문으로 제출함.

2021년 2월

부경대학교 대학원

토 목 공 학 과

김 누 리

김누리의 공학석사 학위논문을
인준함.

2021년 2월 19일

주 심 공학박사 이 상 호 (인)

위 원 공학박사 서 용 철 (인)

위 원 공학박사 김 수 한 (인)

목 차

표 목차 ····· iii

그림 목차 ····· iv

Abstract ····· vi

제1장 서론 ····· 1

1.1 연구배경 ····· 1

1.2 연구의 기본 가설 설정 ····· 2

1.3 연구내용 ····· 3

제2장 문헌연구 ····· 4

2.1 막여과 ····· 4

2.1.1 개요 ····· 4

2.1.2 파울링 ····· 11

2.2 기계 학습 ····· 16

2.2.1 기계학습 알고리즘 ····· 16

2.2.2 딥 러닝 ····· 18

2.2.3 Python ····· 24

제3장 연구방법 ····· 29

3.1 연구 개요 ····· 29

3.2 연구 방법 ····· 29
3.2.1 저압 막여과 공정 시뮬레이터 ····· 29
3.2.2 Step feedback ····· 40
3.2.3 AI feedback ····· 49

제4장 연구 결과 ····· 61
4.1 저압 막여과 공정 시뮬레이터 ····· 61
4.2 Step feedback ····· 65
4.3 AI feedback ····· 71

제5장 결론 ····· 81
5.1 연구 결과 요약 ····· 81
5.2 향후 연구 내용 ····· 82

참고문헌 ····· 84

표 목차

표 2.1 막 종류에 따른 분리경과 제거물질·····5
표 2.2 제거물질별 전처리방안·····14

그림 목차

그림 2.1 막여과 종류 및 제거적용 범위 ····· 5
그림 2.2 저압 막여과 물질 제거 원리 ····· 8
그림 2.3 고압 막여과 물질 제거 원리 ····· 9
그림 2.4 막여과 방식에 따른 차압계산 방법 ····· 11
그림 2.5 막 오염 모식도 ····· 12
그림 2.6 퍼셉트론의 개념도 ····· 19
그림 2.7 시그모이드 함수 ····· 20
그림 2.8 tanh 함수 ····· 21
그림 2.9 ReLU 함수 ····· 22
그림 2.10 인공신경망(ANN) ····· 23
그림 2.11 Simple Neural Network와 DNN ····· 24
그림 2.12 구글 Colabotory 홈페이지 ····· 25
그림 2.13 Tensorflow 홈페이지 ····· 27
그림 3.1 저압 막여과 공정 시뮬레이터 ····· 31
그림 3.2 시간에 따른 최저압력과 최고압력의 차이 ····· 37
그림 3.3 한 사이클에서 최저압력과 최고압력 ····· 38
그림 3.4 Step feedback의 개념 ····· 40
그림 3.5 플럭스를 내리는 step feedback ····· 46
그림 3.6 플럭스를 올리는 step feedback ····· 46
그림 3.7 AI feedback의 모식도 ····· 49
그림 3.8 Step feedback 데이터 확보 ····· 50
그림 3.9 은닉층 깊이에 따른 NRMSE 변화 ····· 53
그림 3.10 노드 수에 따른 NRMSE 변화 ····· 54
그림 3.11 활성화 함수에 따른 NRMSE 변화 ····· 55

그림 3.12 학습 횟수에 따른 NRMSE 변화 ····· 55
그림 3.13 학습 데이터 수에 따른 NRMSE 변화 ····· 56
그림 3.14 강화학습의 적용 모식도 ····· 58
그림 3.15 AI feedback + Step feedback ····· 59
그림 4.1 여과-역세별 차압 변동 ····· 61
그림 4.2 파울링에 의한 차압상승 효과 ····· 61
그림 4.3 플럭스에 대한 시뮬레이션 결과 변화 ····· 62
그림 4.4 여과시간에 대한 시뮬레이션 결과 변화 ····· 63
그림 4.5 원수 TOC에 대한 시뮬레이션 결과 변화 ····· 63
그림 4.6 원수 탁도에 대한 시뮬레이션 결과 변화 ····· 64
그림 4.7 Flux의 부적절한 입력에 따른 step feedback 결과 ····· 65
그림 4.8 Step feedback의 FRI graph ····· 65
그림 4.9 수질이 변경되는 경우 차압 상승의 변화 ····· 66
그림 4.10 수질 변동, Step feedback 적용 ····· 67
그림 4.11 Step feedback 적용 시 FRI 변화 ····· 68
그림 4.12 Step feedback 적용 시 문제점 ····· 69
그림 4.13 Step feedback의 score function ····· 70
그림 4.14 AI feedback과 step feedback의 score funtion ····· 71
그림 4.15 AI feedback과 step feedback의 비교 ····· 72
그림 4.16 1년 동안 step feedback과 AI feedback의 비교 ····· 72
그림 4.17 학습 데이터 별 NRMSE 변화 ····· 73
그림 4.18 학습 데이터별 강화학습 적용 시 최종 NRMSE 비교 ····· 74
그림 4.19 학습 데이터 10개와 50개의 score function 비교 ····· 75
그림 4.20 학습 데이터가 10일 때 모델별 NRMSE 변화 ····· 76
그림 4.21 첫번째 모델과 두번째 모델의 성능 비교 ····· 77
그림 4.22 첫번째 모델 ~ 네번째 모델의 비교 ····· 77
그림 4.23 모든 모델의 성능 비교 ····· 78
그림 4.24 모델 별 score 점수 ····· 79

Finding appropriate flux in low-pressure membrane process using deep learning

Noori Kim

Department of Civil Engineering, The Graduate School,
Pukyong National University

Abstract

The low-pressure membrane filtration process includes microfiltration and ultrafiltration, which are mainly operated at pressures within 200 KPa. Membrane fouling reduces the operating efficiency of the low pressure membrane filtration process, so it is important to determine an proper flux to minimize that. The best way to determine the proper flux is to go through a long-term pilot test with one or two membrane modules applied in the field, which can take at least 6 months to over a year and pilot. There is a disadvantage that it is difficult to apply if the quality of the raw water in the water changes.

In this study, by using the characteristic that fouling increases or decreases when the flux is increased or decreased, an algorithm is developed to lower the flux when the fouling index exceeds the limit value through artificial intelligence automatic control, and to increase the flux when the value is not reached. It was attempted to derive an proper flux.

The reference value of the fouling index was calculated through the low-pressure membrane filtration model, and the flux fluctuation was made to find an appropriate value by using deep learning. The membrane filtration process operation result when the flux fluctuates was scored through a function of the fouling index, and the subsequent operation result when the artificial intelligence fluctuated the flux was feedback, and through this feedback result, the appropriate flux fluctuation value was found by itself.

Finally, an proper flux suitable for operating conditions such as raw water quality was derived.

제1장 서론

1.1 연구배경

막여과 공정이란 선택적 투과성을 갖는 분리막을 이용하여 액상-액상, 고상-액상 및 고상-기상의 혼합물을 분리, 농축, 정제 또는 제거하는 공정을 말한다.

현재 국내에서는 지례정수장(300 m^3/일, 2004년) 시험연구를 시작으로 시흥정수장(3,600 m^3/일, 2004년), 이동정수장(1,500 m^3/일, 2005년), 양동정수장(1,000 m^3/일, 2005년)등과 같은 소규모 시설에 적용되었으며, 최근에는 공주정수장(30,000 m^3/일, 2009년), 영등포정수장(50,000 m^3/일, 2011년)등 중, 대규모 정수장을 건설하고 있다(한국수자원공사, 2011).

이러한 막여과 정수장에서 사용되는 막모듈은 주로 정밀여과(micro filtraion; MF) 막 또는 한외여과(ultra filtration; UF) 막을 사용하고 있다. 막여과 정수처리시설의 경제성을 결정짓는 인자는 막의 비가역적 파울링(fouling, 막 오염 현상)을 판단하는 기준인 막간 차압(원수 측과 여과수 측의 압력차)이다. 정유량(유량을 일정하게 유지) 제어를 도입한 막여과 공정에서 파울링이 증가할수록 동일한 유량을 유지하기 위해서 차압을 증가시켜야만 한다.

파울링이 증가하여 차압이 일정 기준 이상 높아지면 화학세정을 실시한다. 이 때 기준이 되는 차압을 “한계차압”이라 하며 실제 현장에서는 대부분 이 개념을 도입하여 화학세척시기를 판단한다. 한계차압으로 화학세척시기를 결정짓는 이유는 막여과 공정에서 버텨줄 수 있는 기계적인 압력을 초과하지 않도록 주의하기 위해서이다. 중요한 점은 한계차압으로 화학세척시기를 판단한다고 하더라도 막여과 공정의 유지관리적 측면에서

현재 막의 오염이 어느 정도 진행이 되고 있는지는 쉽고, 정확하며 빠르게 판단이 가능하여야 한다.

그러나 현재 플랜트 운영 및 유지 관리 기술은 몇몇 경험 있는 기술자들의 노하우에 의해 관리되고 있다(두산중공업, 2016). 기술자가 막여과 플랜트에서 출력되는 여러 데이터(막간 차압, 원수 수질 등)를 읽고 현재 플랜트가 정상적으로 운전되고 있는지, 아니면 막이 막히거나 여과 공극이 찢어지는 등 어떠한 문제가 생겼는지를 판단하여 후속 조치를 취하게 된다. 이와 같은 방법은 언제 발생할 지 모르는 문제를 해결하기 위한 기술자가 항시 대기하고 있어야 하며, 문제 발생을 파악하기 위해 지속적으로 데이터를 읽고 판단해야 한다. 기술자가 부재중이거나 불가피하게 손을 쓸 수 없는 상황에 문제 상황이 발생하거나, 또는 주기적으로 데이터를 읽는다면 데이터를 읽는 사이에 문제가 발생할 경우 상당기간동안 플랜트를 방치하게 된다. 이러한 이유 때문에 사람이 데이터를 읽은 후 문제 발생 확인, 조치를 취하는 것 보다 플랜트에서 직접 출력된 데이터를 분석하여 PC를 통한 자동제어가 가능하다면 기존의 기술자가 데이터를 읽고 판단하는 것보다 훨씬 안정적이고 빠르게 문제 상황을 해결하는 막여과 공정을 확보할 수 있을 것이다.

1.2 연구의 기본 가설 설정

막여과 장치를 운전하기 위해 입력한 데이터(플럭스, 여과시간, 역세시간 등)와 막여과 장치를 운전하며 출력되는 데이터(막간차압 등)를 통해 막여과 장치를 운전할 때, 파울링이 쌓여 막간차압이 비정상적으로 높아지기 전에 미리 파악하고 장치가 플럭스를 자동으로 조절해 파울링이 쌓이는 속도를 억제하면 효율적인 막여과 공정을 유지할 수 있을 것이다.

1.3 연구내용

본 연구의 목적은 '저압 막여과 공정의 자동 제어'로 세부 연구내용은 다음과 같다.

(1) 저압 막여과 파일럿 플랜트를 대체할 수 있는 시뮬레이터 제작

(2) 플럭스 자동 제어를 통한 파울링 억제 실현

(3) 기계학습을 도입하여 보다 안정적이고 빠른 자동 제어 구현

제2장 문헌연구

2.1 막여과

2.1.1 개요

막여과 정수처리공정 도입이 오래된 미국의 지표수관리법에서는 막여과를 아래와 같이 정의하고 있다(한국수자원공사, 2011).

- 막여과는 가압 또는 부압으로 운전되고 1 μm이상의 입자성 물질은 체거름 효과에 의해 제거되어야 한다.
- 대상 특정물질의 제거효율은 직접적인 안전성 시험을 통해 입증할 수 있어야 한다.

국내 상수도시설기준에 따르면, 막여과란 막을 여재로 하여 물을 통과시켜서 원수 중의 불순물을 분리, 제거하여 깨끗한 여과수를 얻는 방법을 말한다(한국상하수도협회, 2010).

일반적으로 정수처리에 사용되는 막은 공칭 공경에 따라 정밀여과막, 한외여과막, 나노여과(nano filtraton; NF) 막, 역삼투(reverse osmosis; RO) 막으로 크게 네 개로 분류한다. 그림 2.1과 표 2.1은 각 막의 공칭공경 범위와 제거대상물질을 정리한 것이다. 여기서, 표 2.1 의 제거가능 물질은 MF에서 제거 가능한 물질은 UF에서 제거 가능하며, 추가로 제거되는 물질만 표기해두었다. 마찬가지로 UF에서 제거 가능한 물질은 NF에서, NF에서 제거 가능한 물질들은 RO에서 제거 가능하다.

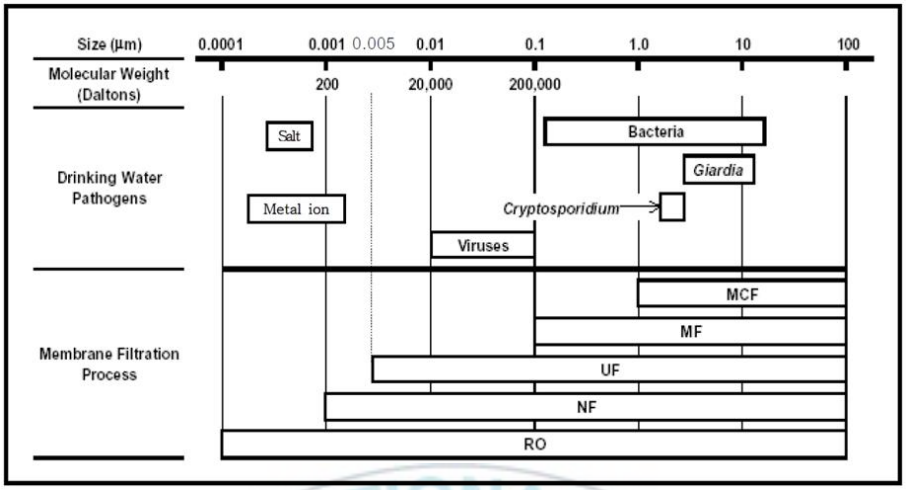

그림 2.1 막여과 종류 및 제거적용 범위

한국 수자원공사 (2011)

표 2.1 막 종류에 따른 분리경과 제거물질

사용막	여과법	분리경	제거가능 물질
정밀여과막 (MF)	정밀여과법	공칭공경 0.01μm 이상	부유물질, 콜로이드, 세균, 조류, 바이러스, 크립토스포리디움 난포낭, 지아디아 난포낭 등
한외여과막 (UF)	한외여과법	분획 분자량 100,000 Dalton이하	부유물질, 콜로이드, 세균, 조류, 바이러스, 크립토스포리디움 난포낭, 지아디아 난포낭, 부식산 등
나노여과막 (NF)	나노여과법	염화나트륨 제거율 5~93% 미만	유기물, 농약, 맛·냄새물질, 합성세제, 칼슘이온, 마그네슘이온, 황산이온, 질산성질소 등
역삼투막 (RO)	역삼투법	염화나트륨 제거율 93% 이상	금속이온, 염소이온 등
해수담수화 역삼투막 (해수담수화RO)	역삼투법	염화나트륨 제거율 99% 이상	해수 중의 염분

한국상하수도협회 (2010)

(1) 정밀여과막

정밀여과막은 체거름 원리에 의해 부유물질이나 원충, 세균, 바이러스 등을 제거한다. 다른 여과막에 비해 공경이 크기에 높은 막여과 유속을 가지고 있다. 하지만 막의 내부폐색이 일어날 우려가 크다. 이러한 문제는 적절한 공경, 공정 설계 및 운전을 통해 해결할 수 있다. 정밀여과막은 높은 투과막 여과유속, 세척의 용이함, 적용의 유연성 및 경제성 등 다양한 장점을 가지고 있어 적용 분야가 빠르게 확대되고 있다. 정밀여과막은 정수처리 뿐만 아니라 식품산업, 화학공업, 금속공업, 생화학공업, 제지펄프 공업, 제약산업 등 다양한 분야에 적용되고 있다.

(2) 한외여과 막

한외여과막은 정밀여과막과 마찬가지로 체거름 원리에 의해 부유물질이나 원충, 세균, 바이러스, 고분자량 물질 등을 분자의 크기로 분리한다. 한외여과막은 저분자와 고분자 물질을 분리하기에 수중에 용해된 고분자의 농축이나 정제에 이용되고 있다. 정밀여과막 보다 더 세밀한 분리능을 가지고 있기에 공경을 μm로 표시하지 않고 분획분자량(molecular weight of cut-off; MWCO)으로 나타낸다. 이처럼 정밀여과 보다 미세공을 가지고 있어 저분자 물질 및 단백질, 효소 등 고분자 물질이나 콜로이드성 물질을 분리할 수 있다. 단, 염(이온)은 통과한다.

정밀여과막에 비해 공경이 미세하기에 운전에 소요되는 에너지는 증가하고 막여과 유속은 감소하며 제거된 용질의 크기 또한 감소한다. 또한 막면에 생성된 농도분극층 또는 Gel층의 저항이 막여과 유속에 영향을 미치기에 난류를 발생시키거나 인위적으로 오염층을 제거해 주어야 한다 (Flemming and Schaule, 1988).

수처리에서는 초순수와 무균수의 제조, 폐액·폐수처리, 배출수의 재이용 등에 사용되고 있다.

(3) 나노여과막

나노여과막은 1986년에 상용화된 막으로 주로 계면활성법에 의한 복합막으로 폴리아미드, 술폰화폴리술폰 등을 소재로 한 역삼투막과 한외여과막의 중간적인 특성을 보인다.

물질제거원리는 역삼투막과 같이 물과 이온의 속도분리차로 이루어지며, 역삼투막에 비해 분획분자량이 크기에 낮은 압력에서도 운전이 가능하다. 나노여과막은 일부 염을 저지하며, 유기물 또한 제거가 가능하다.

나노여과막은 주로 중·저분자 물질분리, 탈염 및 경수의 연화 등에 사용된다. 나노여과막은 NaCl에 대하여 60 %(5 bar, 2,000 ppm 용질), 중탄산칼슘은 80 %, 황산마그네슘, glucose, sucrose에 대해서는 98 %의 제거율을 가진다(Crozes et al., 1993).

(4) 역삼투막

역삼투막은 용액과 용매가 격리되어 용매가 용액 쪽으로 정삼투압 현상을 일으킬 때 용액 쪽으로 정삼투압보다 더 큰 압력을 작용시키는 삼투압 방향의 역방향인 용액 쪽으로 용매가 반투막을 통과하여 용액은 농축되고 용매가 분리되는 원리이다.

역삼투막은 초기에는 전기투석법과 더불어 해수담수화(탈염)를 목적으로 사용되었지만 최근에는 담수의 TDS 및 1가 이온의 농도를 낮추기 위해서도 사용된다(이성우 등, 2003). 또한 초순수 제조나 액체식품의 탈염 등에도 사용되고 있다.

다음은 막여과의 기초에 대해 문헌조사를 실시하였으며, 주로 저압 막여과와 고압 막여과로 구분이 되어 있기에 그 각각의 원리 및 유속을 다루고자 한다.

(1) 막여과 원리

막여과의 원리는 정밀여과, 한외여과와 같은 저압 막여과와 나노여과, 역삼투여과와 같은 고압 막여과의 물질 제거 원리는 차이가 있다.

주로 저압 막여과에서는 막공극 크기에 의한 체거름 기작을 통해 현탁물, 불용해성 물질들을 제거하고 공극보다 작은 물질은 통과하는 원리이다.

그림 2.2 저압 막여과 물질 제거 원리

이와 다르게 고압 막여과에서는 물과 이온의 속도분리 차를 이용하여 물을 이온보다 많이 투과시키는 막여과 방법이다.

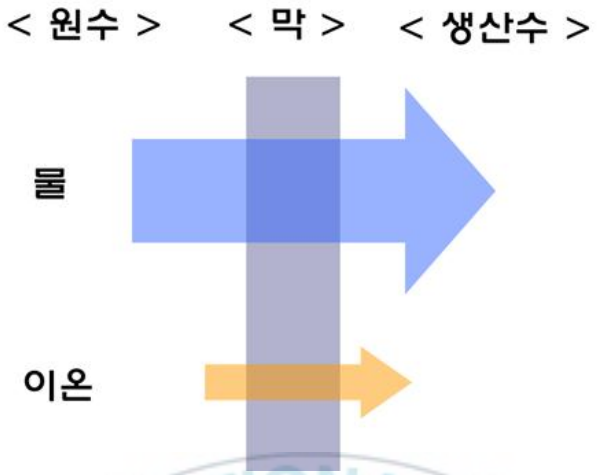

그림 2.3 고압 막여과 물질 제거 원리

(2) 막여과 유속(flux)

막여과 유속은 단위시간에 단위 막면적을 통과하는 수량으로 일반적으로 $m^3/m^2 \cdot day$, m/d, LMH의 세 방법으로 나타낸다. 일반적으로 저압 막여과의 경우 0.5 ~ 1.0 m/d 정도를 목표로 한다. 막여과 유속은 막의 재질, 종류, 여과방식 수온 등에 따라 동일한 막간 차압(transe membrane pressure; TMP)이라도 막여과 유속은 달라진다. 저압용 막여과와 고압용 막여과에서 사용되는 막여과 유속 식은 물질의 제거 원리에 차이가 있기에 계산식에 차이점이 있다. 우선 저압용 막여과에서는 식 (2.1)을 사용한다.

$$J = \frac{\triangle P}{\mu(R_m + R_f)} \tag{2.1}$$

여기서, J는 막여과 유속(m/d), $\triangle P$는 막간 차압(Pa), μ는 물의 동점성계수(Pa・s), R_m은 막 고유의 저항(m^{-1}), R_f는 막 오염에 의한 저항

(m^{-1}) 이다.

고압용 막여과에서 사용되는 막여과 유속 식은 저압용 막여과 식에서 삼투압이 추가된 다음 식 (2.2)를 사용한다.

$$J = \frac{\triangle P - \triangle \pi}{\mu (R_m + R_f)} \tag{2.2}$$

여기서, $\triangle \pi$는 삼투압(Pa) 이다.

(3) 막간 차압

막간 차압은 막을 기준으로 유입수와 생산수의 압력차를 나타내는 것으로 막 오염을 판단하고 막여과 시설을 운영하는데 중요한 지표가 된다. 막간 차압은 여과방식(가압/침지, 전량/순환)에 따라 계산 방식에는 차이가 있으며, 그림 2.4는 막여과 방식에 따른 차압 계산방법을 나타낸다.

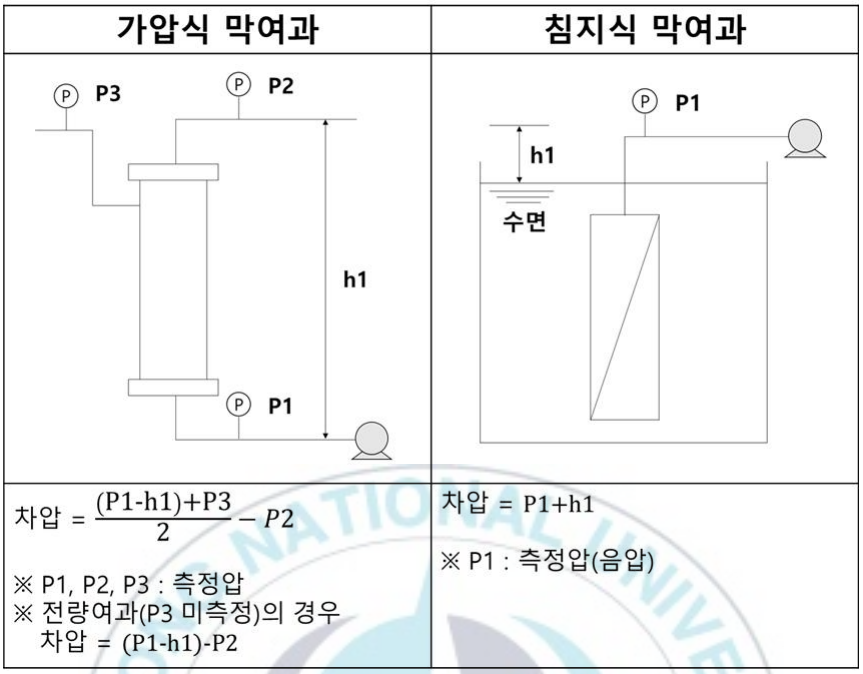

그림 2.4 막여과 방식에 따른 차압계산 방법

한국수자원공사 (2011)

2.1.2 파울링

모든 막여과 공정의 경제성을 좌우하는 가장 큰 요인은 막 오염에 의한 막여과 유속 감소이다. 막여과 공정의 경제적인 측면에서의 장점을 가지기 위해서는 높은 막여과 유속은 꼭 필요하다. 막 오염을 정의하자면, 원수 내 이물질이 막의 표면이나 내부공극에 부착되어 막이 본래의 기능을 다하지 못하도록 억제하는 현상이라 정의할 수 있다(Belfort et al., 1994).

막 오염은 그림 2.5 처럼 입자와 막의 공극 크기에 따라 크게 세 가지로 구분할 수 있다(Cha et al., 2012).

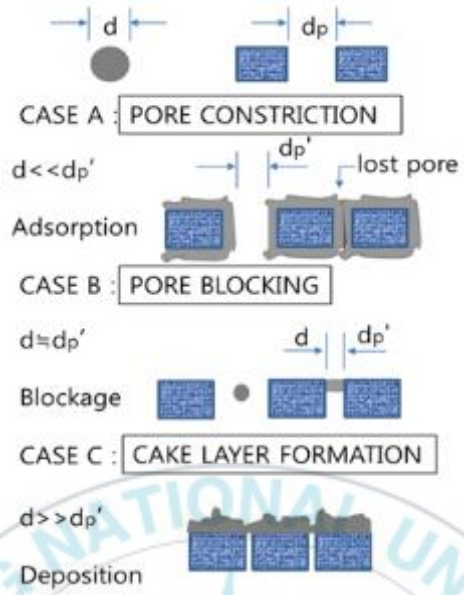

그림 2.5 막 오염 모식도

Cha et al. (2012)

(1) 공극 축소(pore constriction): 막 공극의 지름보다 작은 공경의 입자가 공극 내부에 흡착하여 공극의 크기가 축소되는 현상이다. 이 흡착 정도에 따라 공극이 완전히 막아버리는 경우도 발생하게 된다.

(2) 공극 막힘(pore blocking): 막 공극의 지름과 거의 같은 공경의 입자가 공극을 완전히 막아버리는 현상을 말한다.

(3) 케이크층 형성(cake layer formation): 막 공극의 지름보다 큰 입자들이 막 표면에 쌓이면서 막힌 공극 위에 여러 입자들이 계속 층을 형성하는 것으로, 이를 케이크층이라 한다.

총 세 가지 파울링 현상 중에서 공극 축소나 공극 막힘 현상과 같이 입자가 공극 내부로 침투한 경우에는 막의 회복이 매우 어려울 가능성이 있다. 회복이 불가능한 파울링을 비가역적 파울링이라 하며, 이와 반대로 세척으로 회복이 가능한 파울링을 가역적 파울링이라 한다.

일반적으로 케이크층 형성은 원수를 급수하는 과정 또는 막 세척에 의해 파울링이 제거되어 막 성능 회복이 가능하므로 가역적 파울링에 속한다. 따라서 막의 공극 크기를 선정할 때 원수 내에 존재하는 입자의 크기보다 작은 공극 크기를 결정하는 것이 처리 효율과 더불어 비가역적인 파울링을 막기 위해서도 매우 중요하다고 할 수 있다. 이러한 이론을 확인하기 위해 실제로 정수 공정에 정밀여과와 한외여과를 적용시켜 본 연구결과가 있다(Hiroyuki et al., 1998). 운전 초기에는 정밀여과 막이 막여과 유속이 높아 생산수를 많이 확보할 수 있으나, 막의 공극 내에 흡착한 입자들에 의해 급격히 막여과 유속이 저하되는 결과를 보였다. 정밀여과 막보다 공극이 작은 한외여과 막에서는 초기 막여과 유속은 낮았지만 막 오염의 진행속도가 느렸다. 따라서 장기적으로 보자면 막여과 유속 측면에서는 한외여과가 더 나은 결과를 보인다. 그러므로 원수를 구성하는 물질이 막의 공극보다 작은 경우 막 오염의 진행속도도 늦출 수 있으며, 비가역적인 파울링이 발생하는 것도 억제가 가능하다.

막 오염을 대처하기 위한 방안으로는 두 가지를 생각할 수 있다. 하나는 막에 공급되는 원수에서 막 오염을 일으킬 물질을 미리 제거하는 전처리를 실시하는 것이고, 다른 하나는 막 오염이 발생했을 때 이를 물리적 또는 화학적으로 세정하여 주는 것이다.

(1) 전처리

모든 막여과 공정에서 유지관리, 경제적인 측면을 생각하면 전처리는 꼭 필요하다. 유기막의 경우 원수 중의 협잡물과 토사 등에 의해 막의 표면에 손상이 발생할 수 있고, 협잡물에 의해 유로의 폐색 등에 의한 문제를 방지할 수 있다. 이와 같이 원수 중에 협잡물 등의 오염물질이 유입되지 않도록 스크린이나 프리필터 등이 전처리에 사용된다(표 2.3 참조).

정수처리용으로 사용되는 정밀여과막, 한외여과막의 여과수가 먹는물 수질기준을 만족하지 못하는 경우 전처리가 도입되거나 다른 처리법과의 조합 공정이 요구된다. 특히 소량의 망간이 여과수에 함량 된 경우 염소소독(후처리)에 의해 질산화를 일으키게 되고, 이를 물리세정수로 사용하는 경우 막 차압의 상승을 초래할 수 있다. 따라서 막 여과법에서는 망간에 대한 제어가 필요한 경우 전처리로 제거한다.

또한 유기물질이 막에 흡착되어 물리세정으로도 회복이 불가능한 비가역적 오염이 발생한 경우, 원수 내 유기물 농도가 높은 경우 막여과 공정의 부하를 저감하기 위해서도 전처리를 사용한다. 주로 탁질부하가 높은 경우에는 응집제를 사용하고, 유기물의 농도가 높은 경우에는 분말활성탄 및 오존 등의 산화제를 주입하여 전처리를 실시한다(WHO, 1993).

표 2.2 제거물질별 전처리방안

제거물질	전처리방안
용해성 무기물	이온교환(연수), 불용화(약품첨가, pH조정), 흡착제(Mn 등) 이온봉쇄제, 킬레이트제, pH조정 등에 의한 가용화
용해성 저분자 유기물	흡착제(이온교환수지, 활성탄 등), 산화제(오존, 염소 등), 생물처리, pH 조정(가용화)
용해성 고분자 유기물	염소/불용화(약품, pH, 가열), 막분리, 산화제, 염소/생물처리 가용화 또는 불용화(약품, pH, 가열)
에멀젼/유지	유상분리, 원심분리, oil separator, 계면활성제, 상평형의 변화
현탁물질/침전물	스크린, prefilter, 원심분리, 응집침전, 사여과, 규조토여과 응집, body feeder
미생물	필터, 산화제(염소 등), 살균(약품, 가열, 자외선), 응집여과 응집, body feeder

WHO (1993)

(2) 막의 세정

막의 세정방법은 물리세정과 화학세정 두 가지가 있다. 물리세정은 가압식 MF/UF 막여과 공정에서 이루어진다. 물리세정은 원수가 막을 거쳐 생산수가 되는 여과 과정을 역으로 하여 생산수가 막을 거쳐 막의 표면 또는 내부에 흡착된 물질을 빼내어주는 역할을 한다. 이러한 물리세정은 일정 여과 시간 이후 바로 진행되어 막 오염의 진행속도를 더디게 하는 효과를 볼 수 있다. 화학세정은 이러한 물리세척으로도 회복이 불가능한 오염 즉, 비가역적 막 오염이 발생한 경우 실시하게 된다. 특히 철이나 망간과 같은 용존성 물질이 막 공극 내부에서 산화, 석출되어 흡착된 경우라면 물리세척으로 제거가 힘들기에 화학세척을 통해 제거한다.

2.2 기계 학습

2.2.1 기계학습 알고리즘

기계 학습 또는 머신 러닝(machine learning)은 경험을 통해 자동으로 성능이 향상되는 컴퓨터 알고리즘을 만들고 연구하는 것이며 인공지능의 한 부분으로 간주된다. 기계 학습 알고리즘은 명시적으로 프로그래밍하지 않고 “학습 데이터”라고 불리는 샘플 데이터를 기반으로 모델을 구축한다. 예를 들어 명시적인 프로그래밍이란 [1, 2, 3, 4, 5] 라는 데이터를 순서대로 넣어서 [2, 3, 4, 5, 6] 이라는 데이터를 만들고 싶을 때 프로그래머가 직접 y = x+1 이라는 수식을 입력하여 6을 넣으면 7이 계산되도록 하는 것이고, 기계 학습은 학습 데이터 [1, 2, 3, 4, 5]와 예측 데이터(기계학습을 통해 얻고자 하는 데이터) [2, 3, 4, 5, 6]을 여러 기계학습 알고리즘 중 가장 적합하다고 생각되는 하나를 선택하여 학습시켜 6을 넣으면 7이 예측되어 나오도록 하는 것이다. 기계 학습을 사용하는 이유는 간단하다. 명시적인 프로그래밍을 통해서 수식을 만드는 작업을 하기 까다로운 데이터인 경우, 또는 수식을 만들 수 없는 경우, 학습 데이터와 예측 데이터간의 관계를 알 수 없는 경우와 같은 사람이 빠르게 해결하기 어렵거나 해결할 수 없는 데이터를 다룰 때 사용된다.

대다수의 기계 학습 알고리즘은 학습 데이터와 예측 데이터를 사용하여 통계 분석을 하는 데 사용되고 있으나, 모든 기계 학습 알고리즘이 통계 분석을 수행하는 것은 아니다. 통계 분석 외에도 이메일 필터링 및 컴퓨터 비전(Computer vision, 디지털 이미지를 수집, 처리, 분석 및 이해하는 방법)과 같은 다양한 작업에서 사용된다.

기계 학습 방식에는 학습 시스템에서 사용하는 “signal”또는 “feedback”에 따라 지도학습, 비지도학습, 강화학습 세 가지 범주로 나눌

수 있다.

(1) 지도학습(supervised learning) : 모델에 “teacher”가 제공하는 예시 입력과 출력물이 제시되며, 지도학습의 목표는 입력물을 출력물과 연관시키는 일반적인 규칙을 배우는 것

(2) 비지도학습(unsupervised learning) : 학습 알고리즘에 어떤 라벨도 주어지지 않으며 스스로 구조(데이터간의 관계)를 찾는다. 비지도 학습은 그 자체로 목표(데이터간의 숨겨진 패턴을 발견하는 것)가 될 수도 있고, 목적을 향한 수단(특징 학습)이 될 수도 있다.

(3) 강화학습(reinforcement learning) : 컴퓨터 프로그래밍은 특정한 목표를 수행해야 하는 동적 환경과 상호작용한다. 문제 공간을 탐색하면서 프로그램은 보상과 유사한 피드백을 제공받는데, 강화학습은 이 피드백을 극대화 시키는 방향으로 계속해서 학습한다.

예를 들어 핫도그를 분류하는 기계학습을 위해 지도학습을 시행한다고 하면, 핫도그를 정의하는 라벨을 프로그래머가 직접 설정해야 한다. ‘핫도그는 길쭉하다’, ‘핫도그는 빵 사이에 소시지가 들어있다’, ‘핫도그는 케첩이 뿌려져있다’ 와 같은 라벨을 프로그래머가 직접 지정하고, 핫도그의 사진을 여러 장 넣으면 기계학습 모델은 주어진 라벨을 통해 ‘이 사진은 XX % 의 확률로 핫도그이다’와 같이 예측을 하게 되는 것이다.

그에 반해 비지도학습을 사용하면 매우 많은 핫도그의 사진을 학습 데이터로 사용하여 기계학습 모델을 구성한다. 프로그래머가 어떤 라벨을 붙이거나 하는 작업은 일체 하지 않고 기계학습 알고리즘이 스스로 방대한 양의 핫도그 사진을 학습하여 핫도그가 어떤 것인지를 알아서 구분할 수 있는 알고리즘이 생성되고, 핫도그가 들어있는 사진을 모델에 입력하면 그 사진을 알아서 읽고 판단하여 핫도그를 구분할 수 있게 되는 것이

다.

강화학습은 지도학습, 비지도학습처럼 핫도그를 예시로 들어서는 설명할 수 없다. 강화학습을 설명하기 위해서는 이세돌과의 5판 대국으로 화제를 끌었던 구글의 바둑 AI 인 'Alpha-Go'를 예로 들 수 있다. 앞서 예시로 들었던 핫도그를 구분하는 것은 그 알고리즘을 학습하는 데 어떠한 변수도 존재하지 않았다. 지도학습이면 핫도그에 대한 라벨을 달아주면 핫도그일 확률을 알아서 잘 맞추게 되었고, 비지도학습이면 핫도그의 사진을 엄청나게 학습시키면 핫도그를 맞추게 되었다. 공통적으로 (1) 핫도그의 사진을 학습시키면 (2) 핫도그를 맞추는 모델이 생성되었다. 그러나 바둑을 두는 것은 '나'와 '상대'가 존재하며, '나'의 행동에 따라 '상대'의 행동이 바뀌는 상호작용을 한다. 따라서 바둑에는 (1) -> (2) -> (3) -> (4) 와 같이 무조건 순서대로 시행되는 작업이 없고, 상대방이 어떤 자리에 돌을 두었을 때 내가 어떤 자리에 돌을 두면 상대방이 어떤 반응을 보일 것인지, 또 경기를 이길 확률이 얼마나 높은지를 수읽기를 하며 계산한다. 이와 같이 상호작용을 하는 작업은 지도학습과 비지도학습으로 만든 모델으로는 예측 성능을 전혀 기대할 수 없다. 따라서 강화학습은 매 순간마다 상대적인 반응을 보이는 작업을 학습하고자 할 때 사용된다. 기본적인 학습 데이터가 매우 많이 필요하고, 또한 한번 구성된 모델을 계속해서 사용하는 것이 아니라 모델을 사용해서 데이터를 수집하고, 그 데이터를 다시 학습 데이터에 추가한 후 재학습하여 예측 모델의 성능을 향상시킨다.

2.2.2 딥 러닝(deep learning)

딥 러닝은 인공 신경망(artificial neural network, ANN)에서 시작되었

다. 인공 신경망은 정보처리 및 생물학적 시스템의 분산 통신 노드에서 영감을 받았다. ANN을 구성하는 '퍼셉트론(perceptron)'은 인체 두뇌의 뉴런의 기작을 보고 영감을 받아 만들었다. 하지만 ANN은 두뇌와는 많은 부분에서 차이를 보인다. 특히 신경망은 프로그래머가 코딩한대로만 움직이기 때문에 모든 인체활동을 주관하는 두뇌와는 큰 차이가 있다. (Available from <https://en.wikipedia.org/wiki/Deep_learning> [Accessed 20 December 2020])

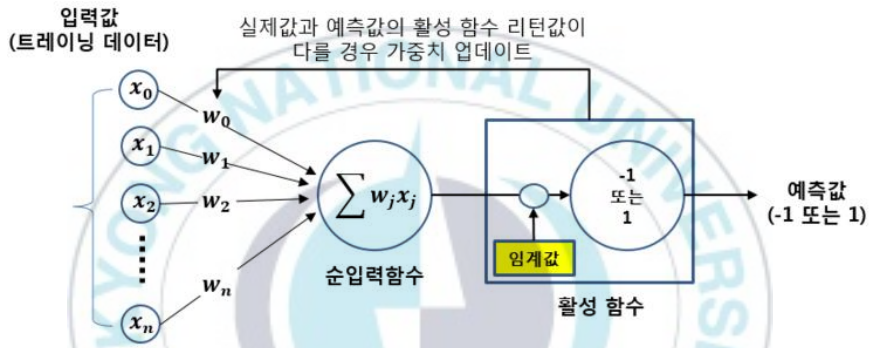

그림 2.6 퍼셉트론의 개념도

퍼셉트론에 입력 데이터가 들어오면 입력 데이터들을 범주별로 구별(x_0, x_1, x_2, ⋯ , x_n)한다. 구별된 데이터별로 가중치(w_0, w_1, w_2, ⋯ , w_n)를 부여한다. 이 때 가중치는 '손실함수'를 줄이는 방향으로 부여된다. 손실함수는 mse(평균 제곱근 오차, mean square error), mae(평균 절댓값 오차, mean absolute error), Binary Crossentropy 등 여러가지가 있는데, 결론적으로 예측 데이터와 실제 데이터의 차이를 계산하는 함수이다. 따라서 손실함수를 줄인다는 것은 예측 데이터가 실제 데이터와 비슷해지도록 한다는 의미이므로 예측 모델의 성능이 향상된다. 그림 2.6에서 '순입력함수'라고 나타난 부분이 바로 가중치와 구별된 데이터를 조합하는 부

분이며, 이를 통해 계산된 값을 '임계값'이라고 한다.

퍼셉트론에서 계산된 임계값을 사용할 수 있는가를 판별하기 위해 '활성화 함수(activation function)'을 사용한다. 활성화 함수는 '비선형'함수이며 임계값을 변환하고 해석하여 임계값을 계산하는데 사용된 가중치 조합을 사용할 수 있는지를 알려준다. 그림 2.6 에 나타난 것 처럼 -1 또는 1을 나타내는 활성화 함수의 경우 임계값을 넣어 계산했을 때 -1 이면 사용할 수 없는 가중치 조합, 1 이면 사용할 수 있는 가중치 조합이라는 결론이 날 것이다. 고전적인 활성화 함수로 시그모이드(sigmoid) 함수를 들 수 있는데, 어떤 값을 입력해도 0 ~ 1의 값을 반환하는 함수이다.

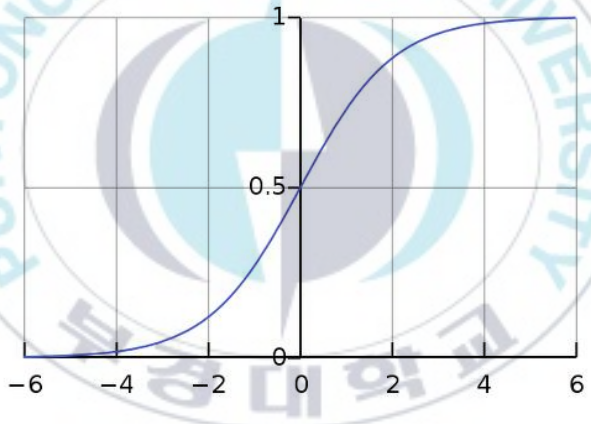

그림 2.7 시그모이드 함수

시그모이드 함수와 비슷하지만 다른 활성화 함수는 tanh 함수가 있다.

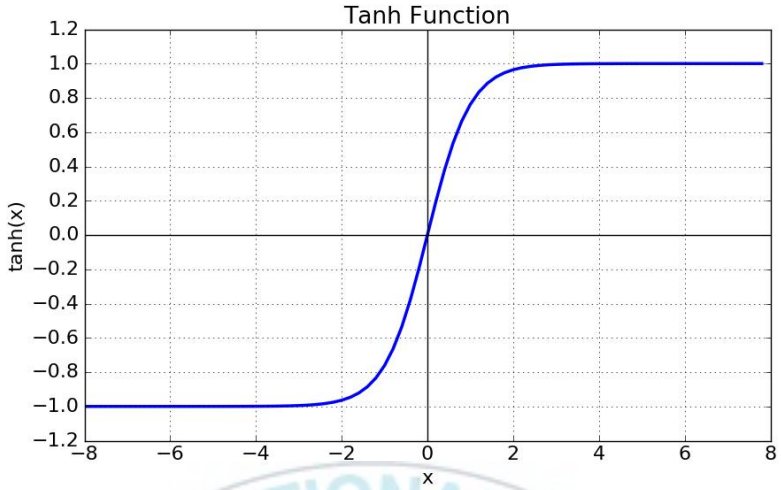

그림 2.8 tanh 함수

tanh 함수는 -1 ~ 1 의 값으로 나타나서 시그모이드 함수보다 조금 더 넓은 범위로 나타나는 것이 차이점이다. 그러나 시그모이드 함수와 tanh 함수는 오차 역전파(backpropagation), 가중치 소실(vanishing gradient)과 같은 문제점이 있기 때문에 현재는 많이 사용돼지 않는다. 최근 가장 많이 사용돼는 활성화 함수로는 rectified linear unit(ReLU) 함수가 있다. ReLU 함수는 0보다 작은값이 입력되면 0을, 0보다 큰 값이 입력되면 그 값을 반환하는 함수이다.

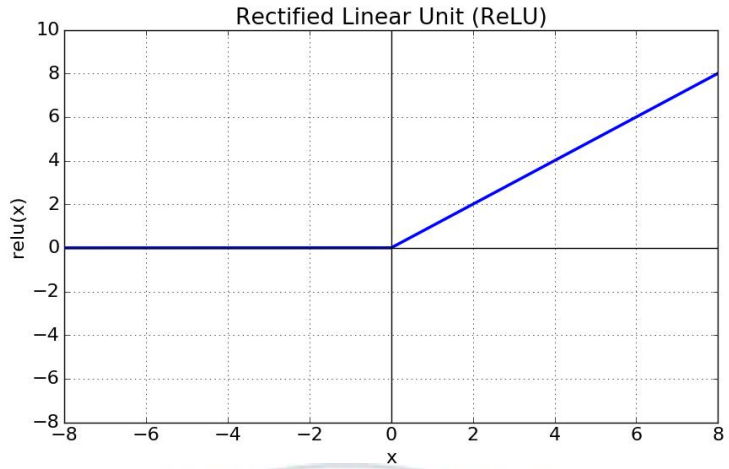

그림 2.9 ReLU 함수

ReLU 함수는 연산 비용이 크지않고 구현이 매우 간단하며 시그모이드, tanh 함수보다 학습이 빨라지는 장점이 있다. 그러나 여전히 0보다 작은값들에 대해서는 기울기가 0이기 때문에 뉴런이 죽을 수 있는 단점이 있다.

퍼셉트론을 여러개 모으면 인공신경망(artificial neural network; ANN)을 형성할 수 있다. 입력층(input layer), 은닉층(hidden layer), 출력층(output layer)로 구성되어 있으며, 퍼셉트론에서 순입력함수와 활성화함수가 적용되는 부분이 은닉층이며 계산된 가중치 조합을 다음(예측값으로서 출력하거나, 은닉층이 여러겹인 경우 다음 은닉층으로 보내는 것)으로 보내는 것이 출력층이다.

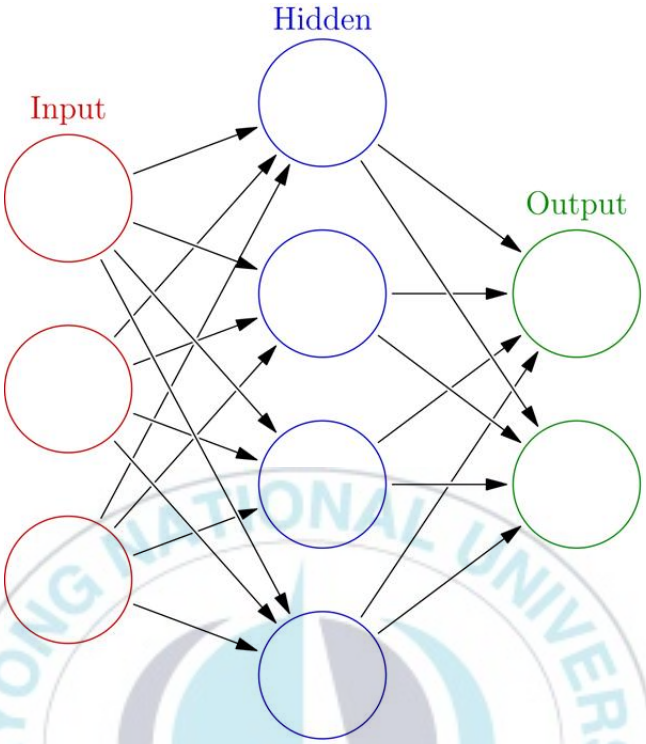

그림 2.10 인공신경망(ANN)

딥 러닝의 형용사 'deep'이라는 단어는 데이터가 변환되는 레이어의 수를 나타낸다. 즉, 은닉층이 무수하게 많은 인공 신경망을 deep learning neural network(DNN)이라 하는 것이다. 딥 러닝은 수많은 기계학습 알고리즘 가운데 한 종류이다. 컴퓨터 비전, 머신 비전, 음성 인식, 자연어 처리, 오디오 인식, 소셜 네트워크 필터링, 기계 번역, 생물 정보학 등의 분야에서 deep neural network, deep belief networks, recurrent neural networks, convolutional neural network와 같은 심층 학습 아키텍처가 적용되었다. 이미 딥 러닝을 적용한 약물 디자인, 의료 이미지 분석, 재료 검사 및 보드게임 프로그램은 전문가가 직접 작업한 성과에 필적하는 결과를 보였으며 어떤 경우에는 더 뛰어난 성과를 보이기도 했다.

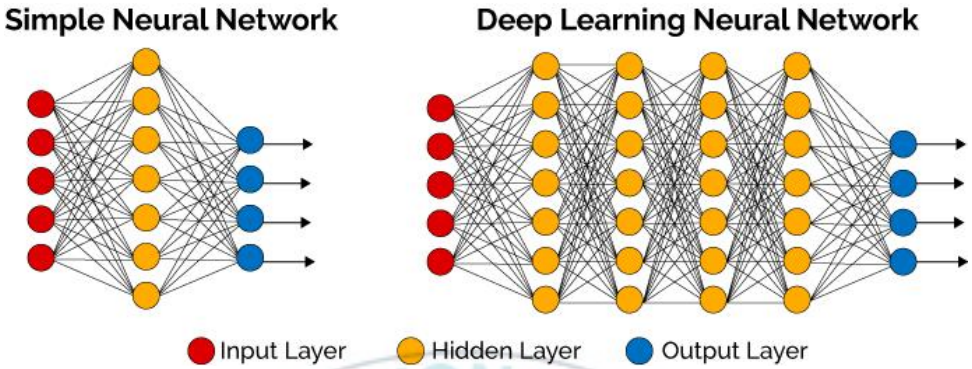

그림 2.11 Simple Neural Network와 DNN

딥 러닝은 비지도학습에 적용할 수 있다. 라벨을 지정할 수 있는 데이터(패턴이 파악 가능한 데이터)보다 라벨을 지정할 수 없는 데이터(패턴을 파악할 수 없거나 파악하기 어려운 데이터)가 더 많기 때문에 이것은 매우 중요한 장점이다.

딥 러닝을 구현하기 위해 Python을 사용했고, 기계학습 툴로 Keras API와 구글에서 제공하는 Tensorflow를 사용했다.

2.2.3 Python

Python은 1991년 프로그래머인 Guido van Rossum이 발표한 고급 프로그래밍 언어로, 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다. 인터프리터식 프로그래밍 언어란 소스 코드를 컴파일러 없이 바로 실행할 수 있다는 의미이며, 객체 지향 프로그래밍은 프로그램을 유연하고 변경이 용이하여 대규모 소프트웨어 개발에 많이 사용된다. 또한 프로그래밍 입문이 쉬워지며, 소프트웨어의 개발과 보수가 간편해지며 보다 직관적인 코드 분석을 가능하게 한다. 동적 타이핑은 자

료형 검사의 대부분이 컴파일 타임이 아닌 실행 시간에 수행되는 경우를 뜻한다. 동적 타이핑에서 값은 자료형을 가지고 있지만 변수는 그렇지 않다. 따라서 사용자가 원하는 대로 변수를 사용할 수 있다. 또한 Python은 자체로 제공하는 기능들뿐만 아니라 다른 사용자가 만들어서 인터넷에 올려둔 기능들을 다운로드 받아서 사용할 수 있다. 이런 기능들을 모아둔 것을 라이브러리라고 한다.

Python 공식 홈페이지에서 다운로드 받아서 사용할 수 있으며, Python 프로그램 그 자체로도 코딩 및 실행이 가능하지만 코드 관리 및 실행 등을 더 편하게 하는 툴이 많이 존재한다. 그 중 구글에서 제공하는 Colab은 구글 서버에서 CPU와 RAM을 제공하는 것을 사용자가 빌려서 쓸 수 있는데 구글 계정만 있으면 별도의 Python 프로그램을 설치할 필요도 없고, 구글 서버를 사용하는 것이기 때문에 인터넷만 연결되어 있다면 장치의 성능과 무관하게 코딩을 진행할 수 있다.(휴대폰과 같은 모바일 기기로도 코딩 작업이 가능하다.)

그림 2.12 구글 Colabotory 홈페이지

Available from <https://colab.research.google.com/notebooks/intro.ipynb>

[Accessed 20 December 2020]

구글 Colab을 사용하면 또 다른 장점이 존재한다. 바로 기계학습을 위한 Tensorflow를 별도 설치하지 않아도 된다는 것이다. 기존 Python 프로그램과, 다른 툴에서는 Tensorflow를 사용하기 위해 라이브러리를 내려받아야하지만 구글 Colab은 원래 기계학습을 시작하는 사람들이 Tensorflow를 쉽게 접근하게 하기 위해 만들어진 툴이므로 Tensorflow 라이브러리가 기본으로 포함되어 있다. 뿐만 아니라 기계학습에 유용하게 사용되는 여러 라이브러리가 기본적으로 내장되어 있기 때문에 기계학습을 시작하는 사람에겐 구글 Colab이 가장 편리하다고 할 수 있다. 이외에 수학적 계산이 가장 빠르다고 알려진 Anaconda라는 툴이 Tensorflow를 가동하기 적합하다는 평을 받고 있다.

Tensorflow는 Colab과 마찬가지로 구글에서 개발, 제공하는 기계학습을 위한 오픈소스 플랫폼이다. 도구, 라이브러리, 커뮤니티 리소스로 구성된 포괄적이고 유연한 생태계를 통해 연구자들은 머신러닝에서 첨단 기술을 구현할 수 있고 개발자들은 머신러닝이 접목된 애플리케이션을 손쉽게 빌드 및 배포할 수 있다. Tensorflow는 Window뿐만 아니라 리눅스, MacOS등의 다른 PC 운영체제 및 안드로이드, iOS와 같은 모바일 환경에서도 구동될 수 있다.

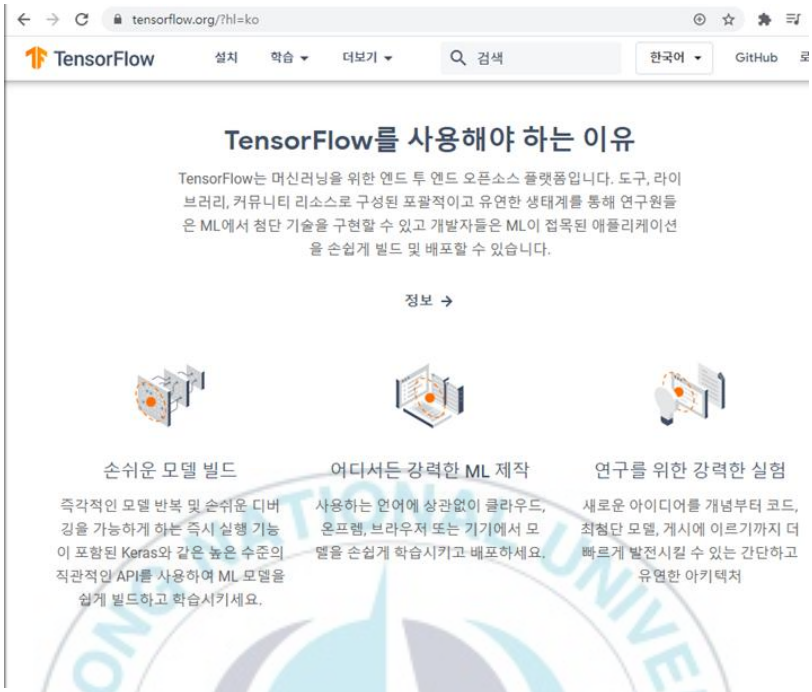

그림 2.13 Tensorflow 홈페이지

Available from <https://www.tensorflow.org/> [Accessed 20 December 2020]

Tensorflow 홈페이지에 들어가면 여러 예제들을 통해 기계학습을 접할 수 있도록 구성되어 있으며, 또한 인터넷에 여러 예제들이 구글 Colab을 이용하여 구동할 수 있도록 업로드 되어 있으므로 쉽게 기계학습과 Tensorflow 공부를 시작할 수 있다.

Keras는 Python으로 작성된 오픈 소스 신경망 라이브러리이다. MXNet, Deeplearning4j, Tensorflow, Microsoft Cognitive Toolkit, Theano 에서 실행될 수 있다. 딥 신경망과의 빠른 실험을 가능케 하도록 설계되었으며 최소한의 모듈 방식의 확장 가능성에 초점을 두고 있다. 2017년에 구글 Tensorflow 팀이 Tensorflow의 코어 라이브러리에 Keras를 지원하기로 결정하면서부터 Tensorflow를 구현하기 가장 적합한

API(Application Programming Interface)로 Keras를 사용하고 있다. 따라서 원래는 Tensorflow와 Keras가 서로 다른 라이브러리였으나, 이제는 Python에 한정해서는 크게 다른점이 없다고 봐도 무방할 정도로 깊게 연계되어있다.

제3장 연구방법

3.1 연구 개요

현재 저압 막여과 공정은 숙련된 기술자가 막여과 장치에서 출력되는 데이터를 읽고 분석하여 정상적으로 가동되고 있는지 아닌지를 판단하고, 그 결과가 비정상적이라면 어떤 조치를 취해야 할 것인지 판단 후 행동하는 방식으로 운전되고 있다. 이와 같은 방식으로는 문제 상황에 대한 즉각적인 대처가 힘들뿐만 아니라 기술자가 장치를 제어할 수 없을 때에 문제가 발생한다면 기술자가 돌아올 때 까지 비정상적인 상태로 방치된다. 따라서 안정적인 저압 막여과 공정을 위해서는 장치에서 출력되는 데이터를 바로 PC로 입력, 사전에 구성해 둔 알고리즘을 따르는 자동 제어를 구현하는 것이 합리적이다. 또한 자동 제어 전, 후의 데이터를 사용하여 기계학습을 시행하여 더 빠르고 효율적인 자동 제어 알고리즘을 구성한다면 안정적인 저압 막여과 공정 운전이 가능할 것이다.

3.2 연구방법

3.2.1 저압 막여과 공정 시뮬레이터

저압 막여과 공정에는 정밀여과와 한외여과가 있고, 주로 200 KPa 이내의 압력에서 운영된다. 이 때 압력은 막간 차압(trans membrane pressure; TMP, 원수압력과 생산수압력의 차이)을 의미한다. 막간 차압은 시간이 지날수록 점점 상승하는데, 원수에 포함되어있던 막 오염물질(용해되어 있지 않은 물질들)로 인하여 파울링(막 오염)이 생기기 때문이다. 일반적으로 플럭스(막 면적에 대한 유량)가 높을수록, 원수 수질이 나쁠수

록 파울링이 많이 발생한다. 따라서 원수 수질의 변동과 플럭스의 조정에 따라서 막간 차압의 상승 속도가 변하게 된다.

파울링이 발생하여 막간 차압이 상승하는 속도가 빨라지면 한계 압력(200 KPa)에 도달하는 시점이 빨라지고, 저압 막여과 공정을 설계할 때 예정했던 화학세정 주기가 아직 오지 않았더라도 한계 압력에 도달했기 때문에 화학세정을 실시하게 된다. 예정하지 않았던 화학세정이 실시되었으므로 물 생산량이 부족해지고, 이를 보충하기 위해 예비 막여과 라인이 가동되거나, 여건상 예비 라인이 없는 경우 기존 설계 플럭스보다 높게 운전을 할 수 밖에 없다. 또한 예비 라인을 가동하게 되는 경우 추가 전력소비가 생기게 되고, 기존 플럭스보다 높여 운전을 하는 경우에는 화학세정 시기가 앞당겨지는 등 원하지 않았던 상황이 발생하게 된다.

이와 같이 파울링은 저압 막여과 공정의 효율을 하락시키므로 이를 최소화하기 위해서 적정 플럭스 값을 결정하는 것이 중요하다. 적정 플럭스란 저압 막여과 공정을 막간 차압에서 문제가 발생하지 않으면서 최대 유량을 유지할 수 있는 플럭스를 의미한다. 본 연구에서는 적정 플럭스를 구하기 위해 적정플럭스를 '저압 막여과 공정 운전 시작부터 화학세정을 실시하기 직전에 한계 압력인 200 KPa에 도달하는 플럭스'로 정의했으며, 수자원공사의 '막여과 공정 설계 운영관리 매뉴얼'에 따라 화학세정 주기를 1년으로 정했다.

적정 플럭스 값을 결정하기 위한 최선의 방법은 저압 막여과 공정 현장에 적용될 막모듈 1, 2개를 이용한 장기 파일럿 테스트를 실시하는 것이나, 이는 적어도 6개월에서 1년 이상의 기간이 소요되며 파일럿 테스트 상황과 실제 상황에서의 원수 수질이 다르다면 적용하기 어렵다는 단점이 있다. 본 연구에서 파일럿 테스트는 시간이 너무 오래 소요된다는 점과 적정 플럭스는 저압 막여과 공정 설계마다 다를수 있다는 점을 고려해

Python 프로그램을 사용해 저압 막여과 공정 시뮬레이터를 구성한 후 적정 플럭스를 도출해서 연구를 진행했다.

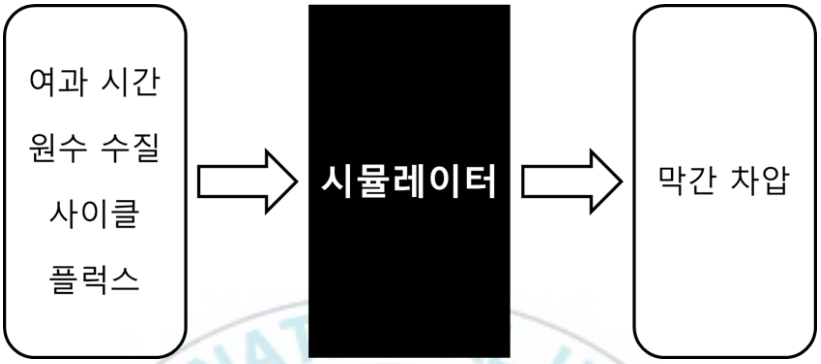

그림 3.1 저압 막여과 공정 시뮬레이터

저압 막여과 공정 시뮬레이터를 만들기 위해 다음과 같은 저압 막여과 식을 사용했다.

$$J = \frac{\triangle P}{\mu R} \tag{3.1}$$

여기서 J는 막여과 유속(m/day), △P는 막간 차압(Pa), μ는 물의 동점성계수(Pa·s), R은 전체 막 저항(m^{-1})이다.

$$\frac{d(\triangle P_i)}{dt} = k(\triangle P_i)^n \tag{3.2}$$

$\triangle p_i$는 i번째 사이클에서의 막간 차압(Pa), t는 시간(sec), k는 equation parameter(Pa^{1-n}/sec), n은 파울링 지수(fouling characteristic parameter,

n = 0일 때 케이크 층 형성(cake filtration), n = 1.5 일 때 공극 수축(pore constriction), n = 2.0 일 때 공극 막힘(complete blackage process))이다. 이 때 k는 k ∝ J를 따른다.

$$R_{i+1,0} = R_{i,0} + (R_{i,f} - R_{i,0}) \times RIF \tag{3.3}$$

$R_{i,0}$은 i번째 사이클이 시작할 때의 전체 막 저항, $R_{i,f}$는 i번째 사이클이 끝날 때의 전체 막 저항, RIF는 비가역적 파울링의 비율이다. 이 때 RIF는 $RIF \propto (R_{i,f} - R_{i,0})$을 따른다.

식 (3.2)의 미분 방적식을 풀면 다음과 같다.(Park et al., 2012)

$$\triangle P_{i,1} = \left((1-n) \times k \times \triangle t + \triangle P_{i,0}{}^{(1-n)}\right)^{\left(\frac{1}{1-n}\right)} \tag{3.4}$$

여기서 $\triangle P_{i,0}$는 한 사이클을 시작할 때의 압력(KPa)이고 $\triangle P_{i,1}$는 여과가 끝난 직후의 압력(KPa)이다. 역세 후 압력을 구하기 위해서는 물리 역세로 제거되지 않는 비가역적 파울링을 고려해야 한다. 비가역적 파울링을 계산하기 위해 $\triangle P_{i,1}$일 때의 저항을 식 (3.1)을 통해 구하면 다음과 같다.

$$R_{i,1} = \frac{\triangle P_{i,2}}{\mu J} \tag{3.5}$$

식 (3.3)과 $R_{i,1}$를 사용하여 역세 후의 막 저항을 계산할 수 있다.

$$R_{i,1} = R_0 + (R_{i,0} - R_0) \times RIF\left(\frac{R_{i,0} - R_0}{R_0}\right) \quad (3.6)$$

R_0는 막 초기 저항을 의미한다.

저압 막여과 공정 시뮬레이터를 만들기 위한 핵심 Python 코드는 다음의 코드이다.

```
# 저압 막여과 공정 시뮬레이터 코드
for i in range(0, cy, 1):
    if i > 0 :
      if (i+1) % 1000 == 0:
        print("현재 %d 사이클 진행중" %(i+1))

      try:
        waterq = pd.read_csv("./waterq.csv")
        cycleq = waterq['Cycle']
        if (i+1) % 96 == 0:
          waterqq = waterq[waterq['Cycle'].isin([qindex])].reset_index(
drop=True)
          turb = waterqq.loc[0][1]/100 # 정규화
          TOC = waterqq.loc[0][2]/10 # 정규화
          k = 3.31E-06 * (turb*100) ** 0.2 + 3.7E-05 * (TOC*10) ** 0.2
          qindex += 96
        read = 0
      except:
        if (i+1) % 96 == 0:
          turb = random.randint(1,100)/100
          TOC = random.randint(1,10)/10
          k = 3.31E-06 * (turb*100) ** 0.2 + 3.7E-05 * (TOC*10) ** 0.2
        read = 1

    # 여과 주기
    if deltaKP > ChemClean: # 화학 세정
      J0 = Jfirst
```

```
    MJ = J0 / 24 / 60 / 60
    deltaP = dP * J0
    deltaKP = deltaP / 1000
    Rt1 = deltaP / MJ / bs/10**10
    Rt = Rt1
    d0 = deltaP
    GP0 = deltaP
    GP2 = deltaP
    Findex = 0
    updown = 0
    step = 0
    FS = 0

  for jenga in range(0, mt):
    # 여과 시간을 1분 단위로 나누어 계산을 진행한다.
    Mtime = Mtime + 1
    Stime = Mtime * 60

    # 분당 차압, 막 저항 계산
    deltaP = (((1 - n) * J0 * k * (1 * 60) + deltaP ** (1 - n)) ** (1
/ (1 - n)))
    deltaKP = deltaP / 1000
    MJ = J0 / 24 / 60 / 60
    Rt2 = deltaP / MJ / bs/10**10 # 여과 후 막 저항 계산
    Dtime = Mtime / 60 / 24

  # 여과 후 압력, 한 사이클 내 최고 압력이며 논문에서 deltaP_(i,1)로 표현
  GP1 = deltaP

  # 역세 진행
  for karla in range(0, ct):
      Mtime = Mtime + 1
      Stime = Mtime * 60
      MJ = J0 / 24 / 60 / 60

      # 역세 후 막 저항 계산.
      Rt3 = Rt1 + (Rt2 - Rt1) * (RIF * (Rt2 - Rt1) / Rt)
```

```
# 역세 후 막 저항을 이용하여 역세 후 차압을 계산한다.
deltaP = bs * Rt3 * MJ*10**10
deltaKP = deltaP / 1000
Dtime = Mtime / 60 / 24

# 이번 사이클의 역세 후 저항이 다음 사이클의 첫 저항이 된다.
Rt1 = Rt3
```

이 코드는 ① 사이클 수 만큼 계산을 반복하고 ② 사이클 마다 여과-역세를 반복하는 코드이다. 이 코드는 2일마다 랜덤하게 수질을 변경하는데, 미리 수질을 저장해둔 csv 파일이 있다면 그 파일을 따라 수질이 변동되도록 수행하고, 저장된 수질이 없다면 탁도는 1 ~ 100 NTU, TOC는 1 ~ 10 ppm 까지의 범위 안에서 랜덤하게 변하며, 각각 100, 10을 나누어 정규화 하는 코드이다. # 분당 차압, 막 저항 계산이라고 주석이 달려있는 부분은 자세히 보면 식 (3.4)와 유사하지만 한 부분이 다른 것을 확인할 수 있는데, 이것은 k가 플럭스에 비례한다는 것을 적용하여 k를 그대로 사용하지 않고 플럭스를 곱하여 J × k 를 적용했기 때문이다.

저압 막여과 공정 시뮬레이터를 사용하여 적정플럭스를 도출한다. MF, UF등의 저압 막여과 공정은 주로 200 KPa 안에서 운전되며, 이 압력을 한계압력이라고 하면 한계압력이 넘어서는 차압이 측정될 때 화학세정을 실시하여 물리 역세로는 제거되지 않는 파울링인 비가역적 파울링을 제거한다. 비가역적 파울링을 제거하면 차압이 거의 막을 개봉했을 때와 비슷하게 떨어진다. 그러나 화학세정은 결국 산, 염기 등의 화학약품을 사용하는 것이므로 막에 악영향을 끼칠 수 있어서 가능한 한 설계한 화학세정 주기대로만 실시하는 것이 가장 좋다.

시뮬레이터의 입력조건인 원수 탁도, 원수 TOC, 플럭스, 여과시간, 역세시간, 사이클수는 각각 10 NTU, 2 ppm, 1 m/day, 29 min, 1 min,

17520 사이클로 입력했다. 각 항목에 대하여 조건을 선택한 이유는 다음과 같다.

1. 원수 탁도 : 저압 막여과에 사용되는 MF, UF막에서 일반적으로 99%이상의 탁도 제거 성능을 보인다. 먹는물 수질 기준에 따르면 생산수에서 탁도는 0.5 NTU 이하로 측정되어야 한다.
2. 원수 TOC : 수질환경기준에 따르면, 하천수와 호소수의 경우 TOC에 대하여 '매우좋음(Ia)' 등급을 받기 위해서는 TOC 2 ppm 이하여야 하므로 2 ppm을 기준으로 설정했다.
3. 여과시간, 역세시간 : 여과시간 및 역세시간의 경우 MF 막여과를 실제 운영되고 있는 정수장들인 공주 정수장의 여과 30분, 역세 30초와 금산 정수장의 여과 29분, 역세 1분을 참고하여 여과시간 29분, 역세시간 1분으로 정했다.
4. 사이클 수 : 3에서 정해진 여과시간과 역세시간을 '여과주기'라고 하며 한 여과주기가 한 사이클이다. 화학세정 기간을 1년으로 잡았으므로 30분 주기로 1년이 되는 사이클인 17520 사이클을 입력했다.
5. 플럭스 : 1 ~ 4 까지의 조건을 바탕으로 1년에 200 KPa 까지 차압이 상승하는 플럭스를 구했을 때 1 m/day 를 구할 수 있었다.

따라서 저압 막여과 공정 시뮬레이션을 통해 구한 적정 플럭스는 1 m/day이며 이 때 입력조건을 '기준'으로 정했다. 적정 플럭스를 이용하여 문제 상황 발생을 판단하기 위한 기준을 정해야한다. 이를 위해 파울링 지수(fouling rate index, FRI)를 도입했다.

파울링 지수는 한 사이클 내의 최저압력과 최고압력을 통해 계산할 수 있다. 파울링이 쌓이면 한 사이클의 시간은 같더라도 최고압력과 최저압

력의 차이가 커진다.

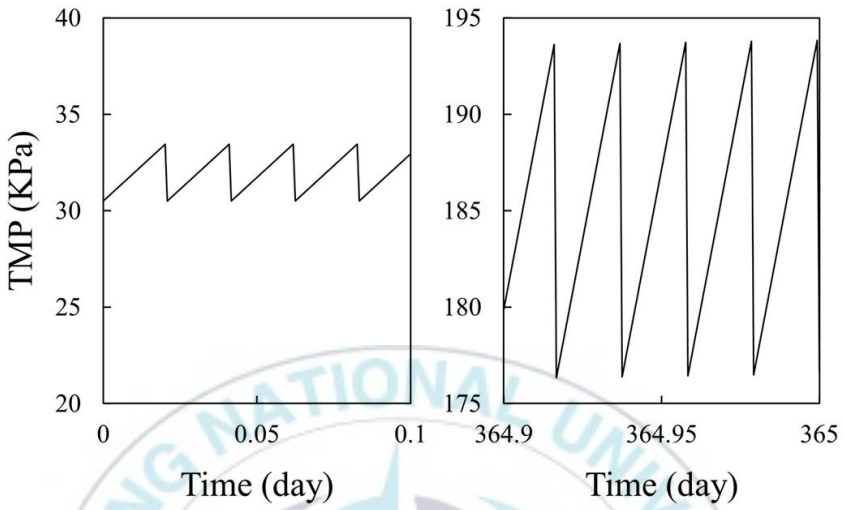

그림 3.2 시간에 따른 최저압력과 최고압력의 차이

실제로는 최저압력부터 최저압력까지 압력의 변화는 미세한 곡선이지만 직선으로 생각해도 무방하다. 직선으로 생각하면 기울기를 구할 수 있는데, 이 기울기가 차압 상승 속도이며 파울링이 쌓일수록 빨라진다. 차압 상승 속도를 계산하는 식은 다음과 같다.

$$v_{\triangle p} = \frac{\triangle P_1 - \triangle P_0}{t_1 - t_0} \tag{3.7}$$

여기서 P_0가 한 사이클에서 최저 차압, P_1이 한 사이클에서 최고 차압을 의미하며 t_0와 t_1은 그 압력이 발생한 시간을 의미한다.

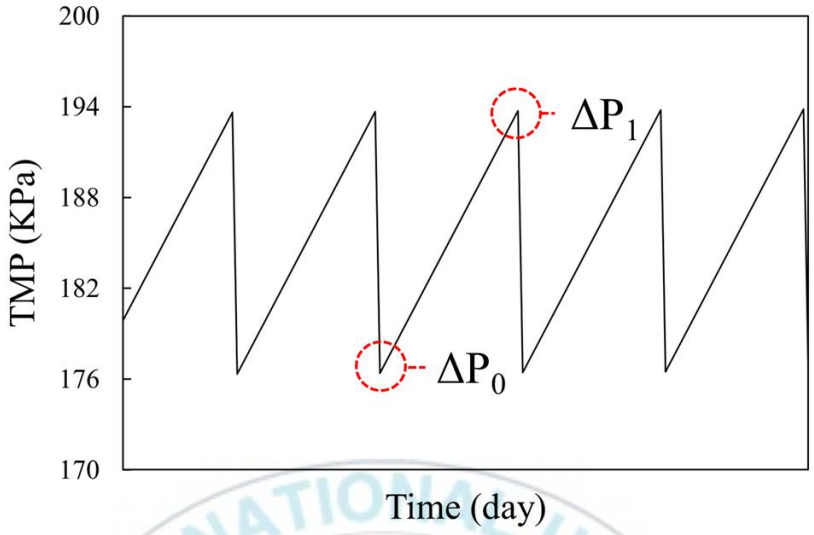

그림 3.3 한 사이클에서 최저압력과 최고압력

차압 상승 속도를 바로 문제 발생 판단의 기준으로 정할 수는 없다. 첫 번째로는 차압 상승 속도는 시간에 따른 변화가 매우 크다는 점이 있다. 두 번째로는 그림 3.4에서와 같이 처음 입력된 플럭스가 적정 플럭스와 달라서 초기 압력이 다른 것을 고려하지 못한다는 점이 있다. 이러한 이유 때문에 파울링 지수를 구하는 식은 다음과 같다.

$$FRI = \frac{\left(\frac{d\triangle P}{dt}\right)}{\triangle P_i} \tag{3.8}$$

이 때 P_i는 i 번째 사이클에서 P_0와 P_1의 평균값이다. 차압이 상승한 정도를 시간이 변한 정도로 나누면(즉, 기울기를 구하면) 파울링이 얼마나 쌓였는지를 확인할 수 있다.

적정 플럭스일 때 FRI 값은 0.003275(1년에 차압이 200 KPa 까지 상

승하는 파울링 지수) 이며 첫 사이클부터 마지막 사이클까지 FRI의 평균값을 계산하여 사용하였고, 평균값과 첫 사이클, 마지막 사이클의 FRI 값의 차이가 평균값의 약 1 % 정도에 해당하여 기준으로 정하기에 합당하다는 판단을 했다. 이 때, 적정 플럭스의 FRI 값을 한계 FRI(FRI_{cri})라고 하여 막여과 운전 중 매 사이클마다 FRI를 계산하여 그 값이 FRI_{cri} 값을 넘어선다면 파울링이 쌓이는 속도가 비정상적이므로 플럭스를 조정해야 한다는 판단의 근거가 된다. FRI_{cri} 값을 정확히 지키도록 한다면 적정 플럭스에서도 평균값과 최저값, 최고값이 약 1 %의 차이가 있기 때문에 비정상적으로 파울링이 발생하는 문제 상황이라고 판단할 것이기 때문에, 계산된 FRI가 한계 FRI보다 5 % 크거나 작다면 문제가 발생했다고 판단하도록 했다.

문제 상황이 발생한 것이 확인되면 플럭스를 조절하여 문제 상황을 벗어나도록 유도한다. 파울링 지수가 높으면 플럭스를 내려서 파울링 발생을 억제하고, 파울링 지수가 낮으면 플럭스를 높여서 물 생산량을 높이는 반응을 한다. 그 과정을 정리하자면 다음과 같다.

$$\begin{aligned} FRI > FRI_{cri} \times 1.05 &\rightarrow Flux\,Down \\ FRI < FRI_{cri} \times 0.95 &\rightarrow Flux\,Up \end{aligned} \tag{3.9}$$

3.2.2 Step feedback

문제 상황이 발생했다는 것을 인지하고 플럭스를 조절하기까지를 step feedback 이라고 했다. 플럭스를 조절하는 방식이 단계별로 진행되어 그래프로 그렸을때 보이는 결과가 계단 모양이기 때문에 step feedback이라고 했는데, 기본적인 개념은 다음과 같다.

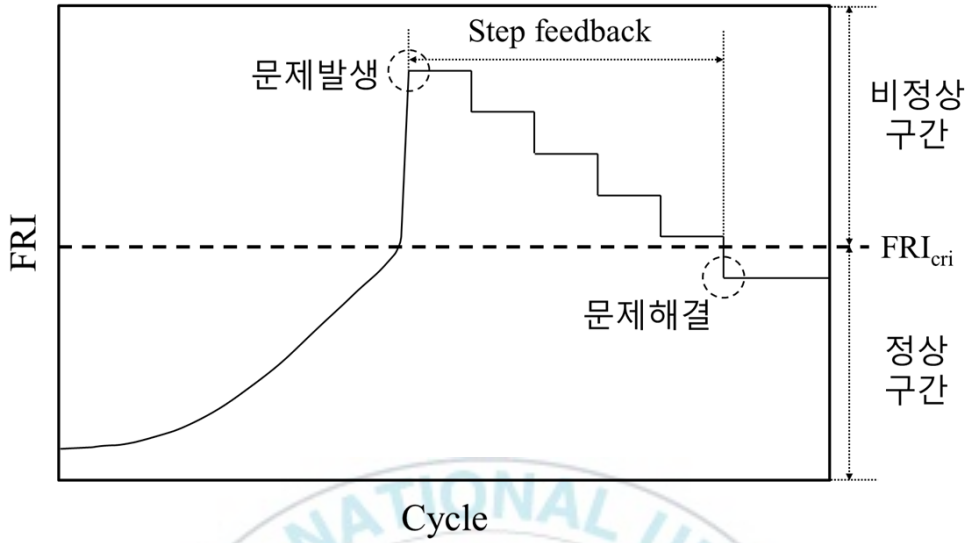

그림 3.4 Step feedback의 개념

막여과 공정 운전중에 사이클마다 계산되는 FRI가 FRI$_{cri}$의 5 % 범위 안에서 계산되다가 그 범위를 벗어날 만큼 높아졌을 때 플럭스를 조절한다. 플럭스를 조절은 다음과 같이 시행된다.

$$Flux_1 = Flux_0 \times (1 + 0.05FS) \tag{3.10}$$

$$FRI > FRI_{cri} \rightarrow FS < 0$$

$$FRI < FRI_{cri} \rightarrow FS > 0$$

여기서 FS는 피드백 강도(feedback strength)를 의미한다. 플럭스가 높을때와 낮을때 피드백 식이 다른 이유는 플럭스가 높을 때 피드백이 발생할 때는 파울링 지수가 높은 것이 원인이기 때문에 파울링 지수를 감소시키기 위하여 빠르게 플럭스를 조절해야 하고, 플럭스가 낮을 때 피드백이 발생할 때는 생산수량을 더 만들 수 있는 가능성이 있는 것이 원인이므로

파울링 지수에서 문제가 생기지 않도록 천천히 플럭스를 올리기 위해서이다. 플럭스를 내리는 경우에 1회 이상의 피드백이 발생하는 경우, FS 는 1씩 증가하며 플럭스는 초기 플럭스가 아닌 피드백으로 인해 조정되어 감소된 플럭스에 다시 피드백이 적용되어 피드백이 진행될수록 플럭스가 크게 감소하게 된다. 반대로 플럭스를 올리는 경우에는 로그함수를 사용해 피드백이 진행될수록 플럭스 증가량이 줄어들게 하였다.

Step feedback 의 시행 코드는 다음과 같다.

```
    # Step Feedback 진행. Findex 변수를 사용해 Feedback이 시작하는지(0) 아니면 진행중인지(1)를 판단한다.
    if Findex == 0 : #1
      if FRI > cri * (1+crirange) : #2 플럭스를 내리는 피드백
          list31.append(Dtime)
          list32.append(J0)
          list33.append(turb)
          list34.append(TOC)
          list35.append(FRI)
          list36.append(FS)
          list37.append("")
          FS -= 1  # 플럭스를 내리기 때문에 FS가 - 로 적용된다
          J0 = list32[0] * (1 + (alpha * FS))  # 플럭스를 내리기 위한 식은 같으며, 피드백 시작시 플럭스에 대하여 적용된다
          updown = 1  # 플럭스를 내리는 피드백이라는 표시
          Findex = 1  # 피드백 과정 중이라는 표시; 1로 표시하였다
          step += 1  # 피드백이 진행된 기간을 늘린다

      elif FRI < cri * (1-crirange) : #2 플럭스를 올리는 피드백
          list31.append(Dtime)
          list32.append(J0)
          list33.append(turb)
          list34.append(TOC)
          list35.append(FRI)
          list36.append(FS)
```

```
            list37.append("")
            FS += 1  # 플럭스를 올리기 때문에 양수로 적용
            J0 = list32[0] * (1 + (alpha * FS))
            updown = 2  # 플럭스를 올리는 피드백; 2로 나타냄
            Findex = 1
            step += 1

    else : #1  Findex = 1; 피드백 과정중에 있음
      if turb != list33[0] or TOC != list34[0]: # 피드백 중 수질이 변하는
 경우 피드백 조건이 변하므로 기존 데이터를 버린다
        FS = 0
        step = 0
        updown = 0
        Findex = 0
        list61=[]
        list62=[]
        list63=[]
        list64=[]
        list65=[]
        list66=[]
        list67=[]

      else: #2  피드백 중 수질이 변하지 않은 경우(일반적인 경우)
        if updown == 1 : #3  플럭스를 내리는 피드백
          if FRI > cri : #4  FRI 가 cri 보다 높다면 무조건 피드백을 계속한
다
              list31.append(Dtime)
              list32.append(J0)
              list33.append(turb)
              list34.append(TOC)
              list35.append(FRI)
              list36.append(FS)
              FS -= 1
              J0 = list32[0] * (1 + (alpha * FS))
               score = 0  # FRI 가 cri 보다 높기 때문에 파울링이 많이 발생
하고 있으므로 0점을 준다
              list37.append(score)
```

```
            step += 1

        else : #4  FRi 가 cri와 같거나 cri 보다 낮으므로 데이터를 기록하고 피드백을 종료
            list31.append(Dtime)
            list32.append(J0)
            list33.append(turb)
            list34.append(TOC)
            list35.append(FRI)
            list36.append(FS)
            score = SCORE(cri,FRI)
            if score < 0:  # score function으로 기록된 점수가 음수인 경우, 데이터를 0점 처리한다; 현재 score function 에서는 일어날 수 없는 일이지만, 에러가 생길 경우를 대비하여 삭제하지 않음
              score = 0
            list37.append(score)

            # 기계 학습을 위한 데이터 생성; step feedback만 있는 현재 코드에서는 사용되지 않는다
            if len(list31) > 1:
              for down in range(0,len(list31)-1):  # 양질의 학습 데이터를 얻기위한 반복 코드
                list41.append([list31[down],list32[down],list33[down],list34[down],list35[down],(list36[-1]-list36[down]),list37[-1]])

            # score 평가용 데이터 생성; score에 피드백 기간을 나눠서 점수를 재평가하는 과정을 거친다
            if len(list32) > 1:
              list42.append([list31[0],list32[0],list33[0],list34[0],list35[0],list36[-1],abs(list37[-1]/step)])

            step = 0
            FS = 0
            updown = 0
            Findex = 0
            score = 0
            list31=[]
```

```
            list32=[]
            list33=[]
            list34=[]
            list35=[]
            list36=[]
            list37=[]

        elif updown == 2: #3  플럭스를 올리는 피드백
          if FRI < cri : #4  피드백을 계속하는 구간
            list31.append(Dtime)
            list32.append(J0)
            list33.append(turb)
            list34.append(TOC)
            list35.append(FRI)
            list36.append(FS)
            FS += 1
            J0 = list32[0] * (1 + (alpha * FS))
            score = SCORE(cri,FRI)
            if score < 0:
              score = 0
            list37.append(score)
            step += 1

          else : #4  피드백 종료; FRI 가 cri 보다 높아지면 종료 후, 데이터
를 기록하지 않고 피드백이 일어나기 이전의 데이터를 사용한다.
            for jat in range(0,len(list31)-1):  # 마지막 피드백을 제외한
모든 점수를 0점 처리한다
              list37[jat] = 0

            # 학습용
            if len(list31) > 1:
              for up in range(0,len(list31)-1):
                list41.append([list31[up],list32[up],list33[up],list34[
up],list35[up],(list36[-1]-list36[up]),list37[-1]])

            # 평가용
            if len(list32) > 1:
```

```
            list42.append([list31[0],list32[0],list33[0],list34[0],li
st35[0],list36[-1],abs(list37[-1]/step)])

            step = 0
            FS = 0
            updown = 0
            Findex = 0
            score = 0
            list31=[]
            list32=[]
            list33=[]
            list34=[]
            list35=[]
            list36=[]
            list37=[]
```

위 코드에서 alpha는 플럭스 조절 양, FS는 피드백 강도(feedback strength)를 뜻한다. 피드백 시간이 늘어날수록 증가하는 FS에 따라 얼마나 플럭스를 내리거나 높일지를 정하는 플럭스 조절 양을 FS에 곱하여 플럭스 조절 수치를 구하고, 그것을 피드백 발생시의 플럭스에 곱해 플럭스를 조절한다. 이때 문제 상황이 발생하였는가를 판단하는 것은 여과 후 차압이 상승한 결과를 사용해 FRI를 계산하여 한계 FRI의 5 % 보다 높은지, 낮은지를 판단한 후 플럭스 조절 피드백을 실행한다. 문제 상황이 해결되기 전 까지는 피드백 코드를 반복 실행하면서 바뀌는 데이터(플럭스, 차압 등)를 기록하고, 피드백이 종료되면 기록된 데이터를 표(2차원 배열)의 형태로 저장한다.

위 코드를 사용하여 step feedback이 적용되는 과정을 그림으로 나타내면 다음과 같은 두 그림으로 표현할 수 있다.

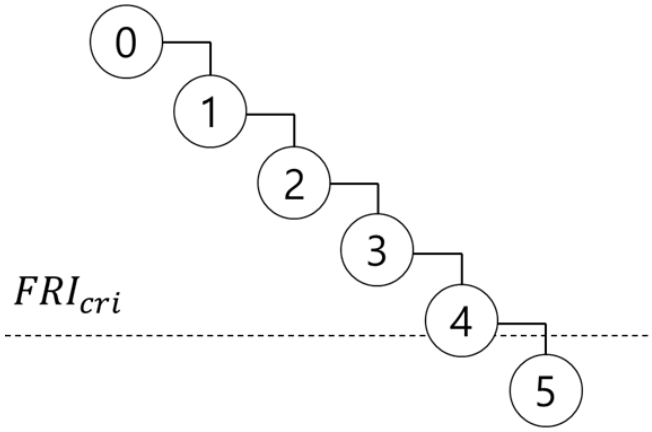

그림 3.5 플럭스를 내리는 step feedback

플럭스를 내리는 피드백(FRI > cri)인 경우 그림 3.5와 같이 작동한다. 피드백이 시작된 플럭스에 플럭스 하강 비율(alpha = 0.05)과 FS(첫 FS는 -1, 피드백이 진행될 때 마다 -1씩 증가)를 곱해서 플럭스를 내린다. 플럭스는 계산된 FRI가 FRI_{cri}보다 낮아질 때 까지 반복되며, 그림 3.5 에서 피드백은 5단계 진행되었다.

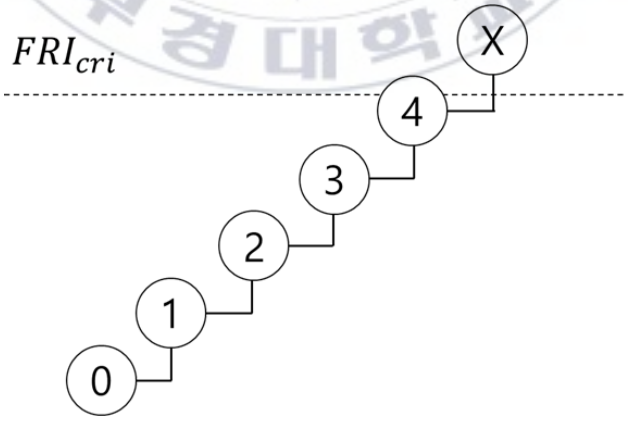

그림 3.6 플럭스를 올리는 step feedback

플럭스를 올리는 피드백은 조금 다르게 적용된다. 한계 FRI보다 5% 이상 높거나 낮으면 피드백이 시행되는 것은 동일하지만, 플럭스를 올리는 피드백은 계산된 FRI가 FRI_{cri}보다 높아지면 그 이전 피드백의 데이터(그림에서 4로 나타난 부분)를 가져와서 사용한다. 또한 피드백은 4단계 진행된 것으로 적용한다. 이와 같은 방법으로 피드백을 적용할 수 있는 이유는 한계 FRI보다 계산된 FRI가 높을 때는 파울링이 쌓이는 속도가 기준 플럭스보다 높을 때이기 때문이다.

이와 같은 방법들로 적용된 step feedback의 성능을 수치적으로 확인하기 위해 다음과 같은 계산식을 도입한다.

$$Score(FRI) = \frac{20FRI}{FRI_{cri}} - 15 \tag{3.11}$$

식 (3.11)은 score function 이라고 한다. Score function은 매 사이클마다 계산되는 FRI가 FRI_{cri}와 가까울수록 높은 점수(점수는 0 ~ 5 점으로 구분한다.)를 매긴다. 점수가 높을수록 적정 플럭스에 가까운, 파울링 지수가 높지 않으면서 동시에 가능한 한 가장 많은 유량을 생산하는 플럭스임을 의미한다. 그림 3.5 에서 5번, 그림 3.6에서 4번이 '높은 점수'를 나타낼 것이다. 그러나 피드백 성능을 식 (3.11)로만 계산한다면 step feedback은 반드시 높은 점수를 받게된다. 매우 좁은 간격(피드백 발생 시 플럭스의 5 %)으로 플럭스를 조금씩 조절하기 때문에 피드백이 종료되었을 때는 반드시 한계 FRI와 계산된 FRI가 비슷할 수 밖에 없기 때문이다. 그러나 제대로 된 피드백 성능을 고려하기 위해서는 피드백 기간을 고려해야만 한다. 본 연구는 실험 장치로써 실제 데이터를 기반으로 한 시뮬레이터를 사용하고 있기 때문에 피드백을 적용하면 그 데이터를 즉각

적으로 확보할 수 있다. 그러나 실제 막여과 공정에서는 피드백을 적용한 후 데이터를 확보하기 까지 상당한 시간(막여과 공정마다 설계내용이 다르기 때문에 일반적으로 정해진 시간은 없으나, 본 연구에 적용된 코드를 그대로 적용할 경우 여과주기(여과+역세)당 한 번 데이터를 수집한다.)이 소요된다. 피드백을 적용하고 종료되기까지 시간이 길고, 문제상황에서 방치되는 시간이 길기 때문에 막여과 공정에 악영향을 미치게 된다. 따라서 step feedback이 시작부터 종료되기까지의 기간(코드에서는 step 이라고 표현했다.)을 피드백 종료시 score(그림 3.5에서 5, 그림 3.6에서 4에서의 점수)에 나눠주어야 제대로 된 피드백의 성능을 평가할 수 있을 것이다.

위와 같은 방법으로 점수를 평가했을 때, step feedback은 아쉬운 성능(추후 4장에서 그림으로 확인할 수 있다.)을 보인다. 보다 빠르게 더 좋은 성능의 피드백을 도입할 방법이 있다면 어떤 것일까 고민했을 때 막여과 공정 시뮬레이터에서 다량의 데이터가 생성되고, 피드백이 시작될 때의 데이터를 넣으면 바로 피드백 강도를 도출할 수 있는 무언가가 필요하다는 점에서 기계학습을 도입하여 예측 모델을 만들어 FS를 구하는데 사용하면 피드백의 성능을 강화시킬 수 있을 것이라 판단했다.

3.2.3 AI Feedback

기계 학습을 진행하기 위해 구글에서 제공하는 Python 라이브러리인 tensorflow와, 마찬가지로 기계학습을 위해 사용되는 Python API 인 Keras를 사용했다.

무수히 많은 기계 학습 알고리즘이 있으나 막여과 공정에서는 매우 많은 데이터가 생성된다는 점에서 기계학습을 도입하자는 생각을 하게 되었으며, 또한 그 부분이 장점으로 사용될 수 있을 것이라고 판단했기 때문에 많은 데이터를 입력한 후 그것을 바탕으로 기계학습 모델을 형성하는

알고리즘인 딥 러닝을 선택했다.

딥 러닝 모델을 저압 막여과 공정 시뮬레이터에 적용하기 위한 방법은 두가지가 있는데 첫 번째는 미리 장기간의 파일럿 테스트 또는 실제 플랜트 운전을 통해 다량의 step feedback 데이터를 수집한 후 이것을 학습하는 방법이다. 그러나 이 방법은 앞서 적정 플럭스를 장기간 파일럿 테스트를 통해 구하는 것과 같은 맥락으로 고려하지 않았다.

두 번째 방법은 장치를 운전하며, 초기에는 step feedback 으로 운전을 하다가 수집한 step feedback 데이터가 일정량 이상 축적되면 그것을 학습시켜 딥 러닝 모델을 구성하여 AI feedback으로 전환하는 방법이다. 이 방법은 막여과 공정을 운전하기 전에 먼저 실시해야하는 실험이 없다. 그러나 AI feedback이 적용되지 않은 공정 가동 초기에는 step feedback 으로 공정이 가동되기 때문에 낮은 성능의 피드백을 시행하게 된다는 단점이 있다.

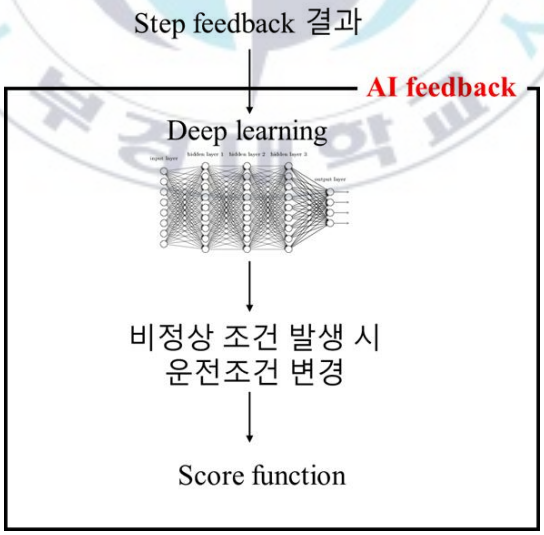

그림 3.7 AI feedback의 모식도

딥 러닝 학습을 위해 step feedback에서 피드백시작-피드백종료를 한 세트로 하여 피드백 시작시 플럭스, 원수 수질과 피드백 종료시 FS, score(피드백 기간으로 나누지 않은)를 수집한다. 그런데 이렇게 피드백 한번 당 데이터 세트 하나를 수집하면 충분한 데이터 량을 확보하는데 걸리는 시간이 얼마나 될지 알 수 없다. 따라서 기계학습을 진행하기 위해 데이터를 확보하는 방법을 두 가지 적용했다.

첫번째 데이터 확보 방법은 step feedback에서 피드백 종료시의 데이터는 고정하고, 피드백 과정에서 생기는 중간 데이터를 모두 피드백 시작 데이터로 하는 데이터 세트를 추가하는 것이다. 그림으로 표현하면 다음과 같이 표현할 수 있다.

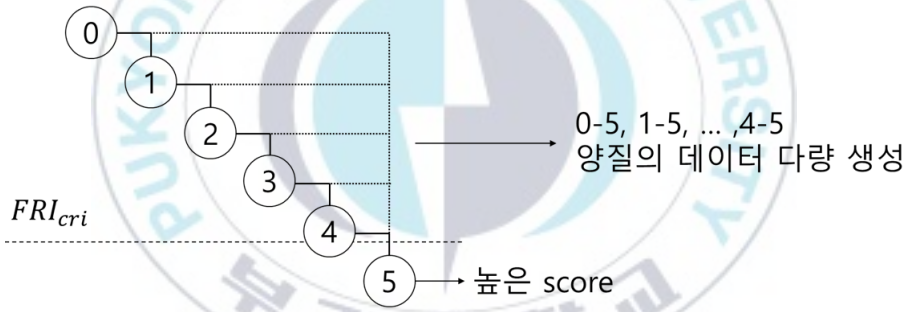

그림 3.8 Step feedback 데이터 확보

한 feedback 이 시행(FRI 0)되고 종료(FRI 5)되었을 때의 데이터 FRI 0-5 를 한 세트로하여 피드백 데이터를 모으던 기존 매커니즘과 달리, 피드백 중간 데이터인 1 ~ 4 를 사용하여 1-5, 2-5, ... , 4-5 데이터를 만들어 데이터 세트 4개를 추가할 수 있다. Step feedback은 일반적으로 한 번에 피드백이 종료되는 상황이 거의 없기 때문에 이와 같은 방법으로 데이터를 확보하면 높은 score(feedback 기간이 적용되지 않았을 때)의 학습 데이터를 상당수 확보할 수 있다.

두번째의 데이터 확보 방법은 각 데이터 세트를 score에 따라 양을 늘리는 것이다. 그림 3.8 에서 5 지점의 점수가 5점이었다면, 0-5, 1-5, ... , 4-5 데이터세트를 1개에서 5개로 늘려서 학습 데이터를 확보하는 것이다. 이 방법을 적용하면 피드백 점수가 높을수록 그 데이터가 기계학습 모델을 구성하는 기여도를 높일 수 있다. 이와 같은 방법으로 데이터를 확보하여 기계학습 모델을 구성하고, AI feedback을 적용할 수 있다.

AI feedback에 적용된 주요 코드는 두 가지가 있다. 첫번째는 기계학습을 위하여 기계학습 모델을 만드는 코드이다.

```
def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(train_d
ataset.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.001)
  # 학습을 어떤것을 기준으로 할 지 정한다. loss는 mse, metric은
 mae, mse 둘 다 확인한다.
  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```

이 코드는 build_model 이라는 모델을 만드는 function을 만드는 코드이다. function을 만드는 이유는 어떠한 작업을 위해 복잡하게 코딩이 진행되었는데, 이 작업을 여러번 수행해야하는 경우 반복해서 복잡한 코딩을 타이핑하지 않기 위해서이다. function을 통해 그 복잡한 코딩을 일종

의 함수로써 정의하고 쉽게 사용할 수 있게 한다. build_model 또한 그와 같은 목적으로 정의되었다. build_model을 자세히 보면 keras API의 'Sequential' 기능을 사용하여 DNN을 만들고 있다. 본 연구에서는 은닉층 2층으로 뉴런 64개, 출력층 1개로 구성했으며, 사용하는 활성화함수는 'ReLU'를 사용했다. 입력층의 경우 은닉층 첫번째 코드에 input_shape 로 나타난 부분이 입력층이다. train_dataset을 그대로 입력층으로 사용한다는 뜻으로 이해하면 충분하다. 학습 데이터는 플럭스, 원수 탁도, 원수 TOC, FRI 를 score만큼 가중치를 주어 입력하였고 생성된 모델으로 피드백 강도를 예측했다.

이 때 DNN 구성을 은닉층 2층, 뉴런 64개로 정한 이유는 기계학습 모델의 최적화 작업을 거쳐 가장 적합한 구성을 선택했기 때문이다. 모델을 구성하기 위한 세부사항 중 사용자가 조절할 수 있는 부분(은닉층의 깊이, 은닉층을 구성하는 노드 수, 활성화 함수의 종류, 학습 횟수, 학습 데이터 수 등)이 많아서 모델의 구성을 어떻게 하느냐에 따라 기계학습 모델의 예측 성능이 달라진다. 또한 학습 데이터가 달라진다면 한 데이터에서 예측이 잘 되던 구성이더라도 다른 데이터에서는 예측이 잘 되지 않을 수 있다. 따라서 모델의 구성을 변경하며 각각의 모델 성능을 확인하고, 가장 적합한 구성을 선택해 모델을 구성할 필요가 있다.

최적화를 하기 위해 기계학습 모델의 성능을 판단하는 지표로 score function 외에 NRMSE(normalized root mean squere error, mse의 제곱근을 취하여 학습값의 평균으로 나눈 값)를 계산했다. NRMSE는 학습 데이터의 FS와 학습 데이터를 기계학습 모델에 넣어 예측한 FS 값을 비교하여 계산한다.

1) 은닉층 깊이에 따른 기계학습 모델 NRMSE 변화

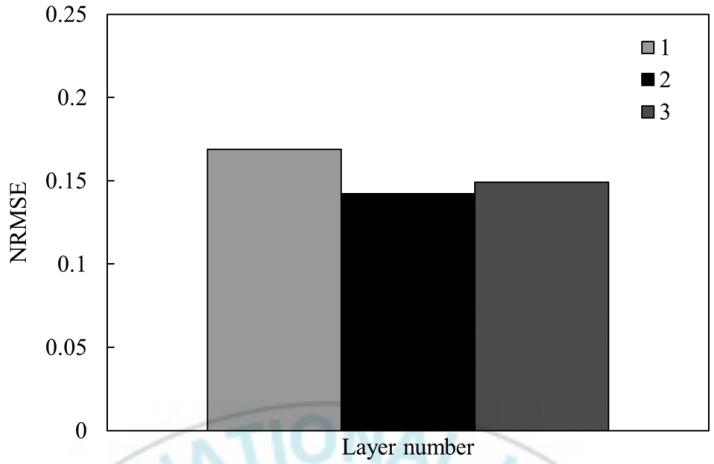

그림 3.9 은닉층 깊이에 따른 NRMSE 변화

은닉층 깊이에 따른 NRMSE 변화 결과이다. 은닉층 1층일 때 가장 NRMSE가 높으며, 은닉층 2층일 때 가장 낮다. 3층일 때 오히려 다시 조금 높아진 것은 학습된 데이터에 의해 모델이 과적합 되어 제대로 예측을 하지 못했다고 판단할 수 있다. 과적합이란 어떤 데이터를 학습했을때, 그 데이터의 경향은 매우 잘 예측하지만 그 외 다른 데이터(학습된 데이터와 비슷한 규칙을 가지고 있는 다른 데이터)를 입력했을 때는 전혀 다르게 예측하는 것을 말한다. 따라서 모델이 과적합 되면 제대로 된 예측을 기대할 수 없기 때문에 은닉층 2층이 가장 적합하다고 할 수 있다.

2) 노드 수에 따른 NRMSE 변화

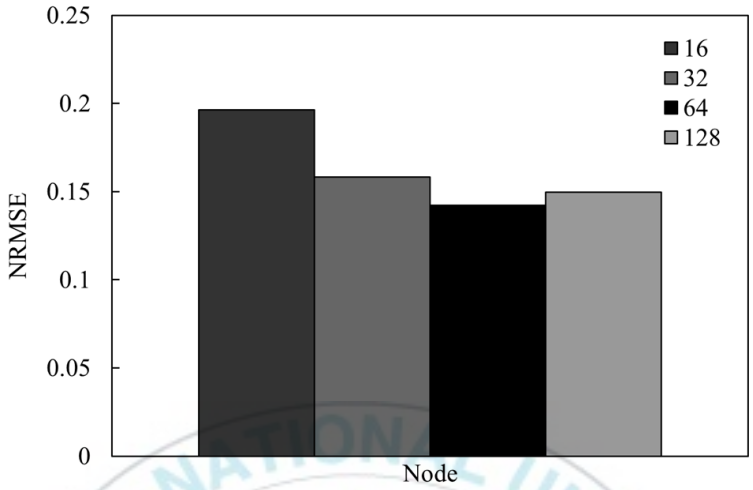

그림 3.10 노드 수에 따른 NRMSE 변화

노드 수에 따른 NRMSE 변화를 보면, 앞서 은닉층의 깊이별 비교와 마찬가지로 노드수가 많아질수록 NRMSE가 작아지다가 노드 수 128 개에서 다시 커지는 것을 확인할 수 있다. 은닉층 깊이별 비교와 마찬가지로 과적합이 발생한 것으로, 비교한 노드 수 16개, 32개, 64개, 128개 중 64개가 가장 적합하다고 판단할 수 있다. 노드 수가 2의 배수인 이유는 tensorflow의 계산에 지원되는 GPU의 RAM의 용량이 2의 배수로 되어있어 노드 수가 2의 배수일 때가 가장 계산이 용이하다고 알려져 있기 때문이다.

3. 활성화 함수에 따른 NRMSE 변화

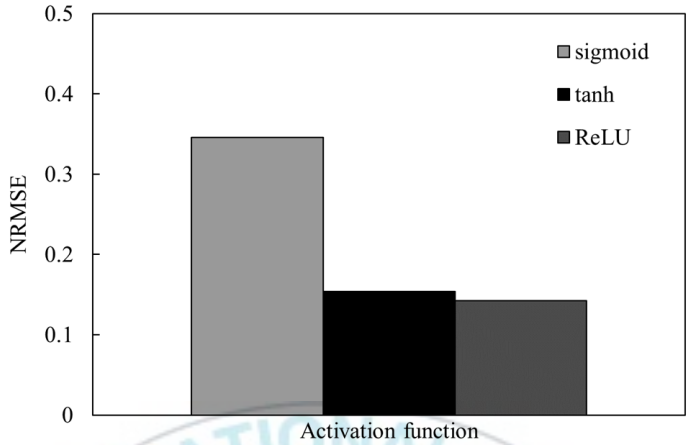

그림 3.11 활성화 함수에 따른 NRMSE 변화

활성화 함수는 sigmoid일 때 가장 성능이 좋지 않고, tanh일 때와 ReLU.일 때가 비슷한 성능을 보였으나 활성화 함수가 바뀐다고 과적합이 발생하거나 모델 예측에 영향을 미치는 어떠한 현상도 발생하지 않기 때문에 조금이라도 성능이 더 좋은 ReLU 함수를 선택할 수 있다.

4) 학습 횟수에 따른 NRMSE 변화

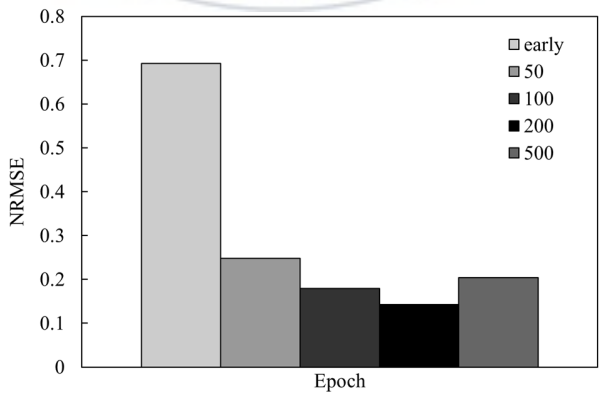

그림 3.12 학습 횟수에 따른 NRMSE 변화

학습 횟수(epoch)에 따른 NRMSE 변화를 나타낸 그래프이다. 여기서 early 라고 나타난 데이터는 매 학습시 일정한 구간(본 연구에서는 10 epoch)동안 모델의 학습 성능이 발전하지 않았다고 판단될 시 학습을 모든 epoch를 진행하지 않고(200 epoch라면 200 번 학습하지 않고) 학습을 종료하는 방법을 의미한다. 본 연구에서는 학습 데이터의 종류가 많지 않기(플럭스, 탁도, TOC의 세 종류) 때문에 early_stop 방법이 잘 적용되지 않았다. early_stop을 제외한 나머지 데이터를 확인하면 epoch 200 에서 가장 낮은 NRMSE를 나타냈다. 데이터에 따라 epoch가 길어져도 과적합이 발생할 수 있기 때문에 epoch 200 이 가장 적합하다고 할 수 있다.

5) 학습 데이터 수에 따른 NRMSE 변화

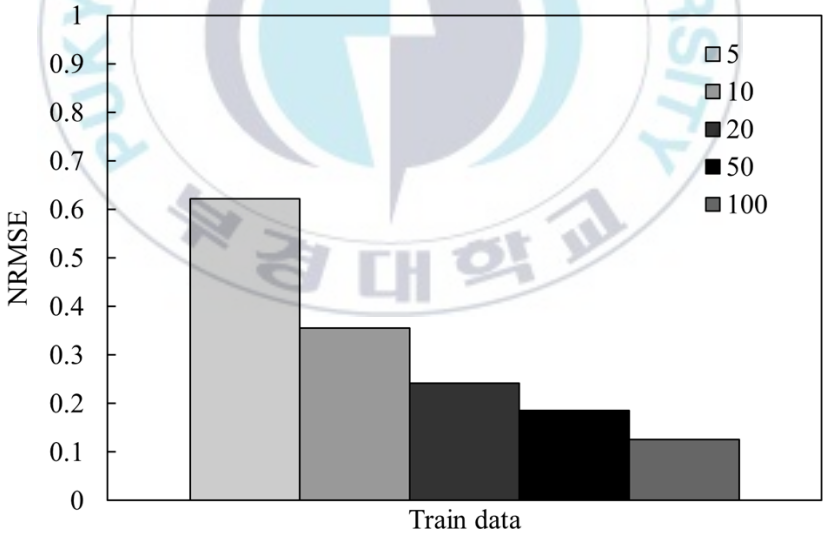

그림 3.13 학습데이터 수에 따른 NRMSE 변화

학습데이터 수에 따른 NRMSE 변화이다. 학습 데이터가 많을수록 예

측 성능이 더 좋아진다는 것을 확인할 수 있다. 따라서 최대한 많은 데이터를 확보한 뒤 기계학습을 수행하면 되겠으나, 100 개의 데이터를 모으기 까지는 시뮬레이터에서도 200일 이상의 긴 기간이 걸리므로 50개의 데이터가 가장 적합하다고 판단했다.

따라서 본 연구에서 기계학습 모델의 최적화 조건은 은닉층 2층, 노드 수 64개, 활성화 함수 ReLU, 학습 횟수 200회, 학습 데이터 50개로 기계학습 모델을 구성하는 것이다.

그러나 학습 데이터 50 개로 기계학습 모델을 구성하는 것에는 매우 큰 문제점이 있다. 본 연구에서는 실험장치로 저압 막여과 공정 시뮬레이터를 사용하여 feedback을 적용한 이후의 결과를 즉각적으로 확보할 수 있었으나, 실제 막여과 공정에서는 feedback을 적용한 후의 데이터를 얻기까지 시간이 꽤 걸린다는 점이다. 따라서 학습 데이터를 많이 확보하려고 할수록 step feedback을 적용하는 기간이 길어지고, step feedback은 feedback 시작부터 종료까지의 시간이 길기 때문에 FRI가 한계 FRI보다 높은 기간이 오랫동안 유지된다. AI feedback을 도입했던 목적인 '빠른' 문제 해결과 매우 큰 차이가 있다.

'빠른' 문제 해결을 구현하기 위해서 학습 데이터를 최소화 할 필요가 있다. 그러나 그림 3.13에서 나타난 학습 데이터의 양에 따른 NRMSE 그래프를 보면 학습 데이터의 양이 적을수록 모델의 성능이 좋지 않다. 이 문제를 해결하기 위해 '강화학습'을 도입하기로 했다.

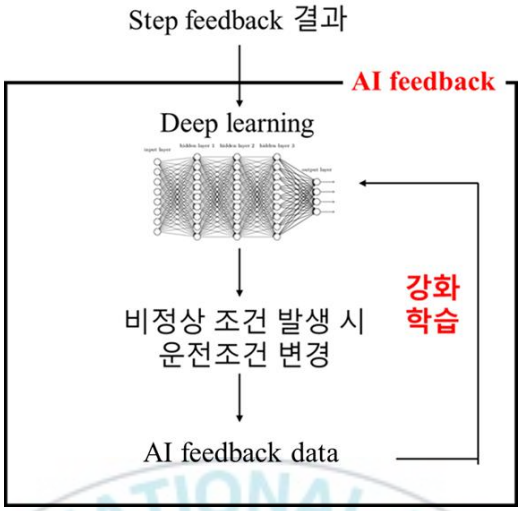

그림 3.14 강화학습의 적용 모식도

강화학습이란 AI feedback을 적용하여 구한 피드백 데이터를 다시 기계학습 모델의 학습 데이터로 사용하는 것이다. 이 방법을 통해 첫 학습의 데이터를 적게 해서 성능이 낮은 AI feedback을 시행하더라도, 그 결과를 통해 강화학습을 시행하여 AI feedback의 성능을 높여갈 수 있을 것이다.

성능이 낮은 AI feedback의 경우 기계학습 모델으로 예측된 FS를 사용해 플럭스를 조절하더라도 파울링 문제 상황이 해결되지 않을 수 있다. 이 때 다시 성능이 낮은 예측 모델으로 FS를 구해도 파울링 문제 상황이 해결되리라는 확신이 없기 때문에 지속적인 AI feedback을 시행하는 것보다 step feedback을 부분적으로 적용하여 문제 상황을 해결하도록 하는 것이 더 안정적인 막여과 공정 가동에 도움이 될 것이다.

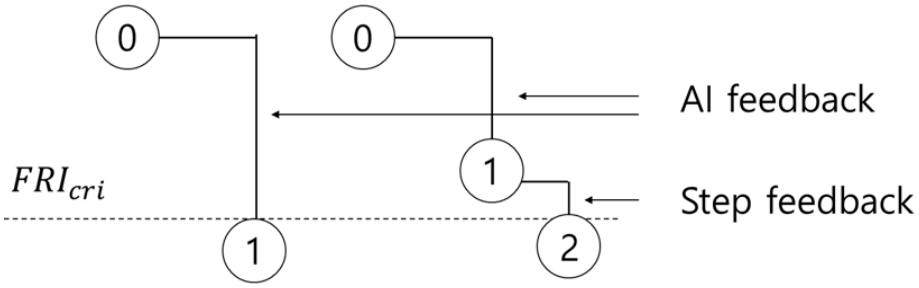

그림 3.15 AI feedback + Step feedback

그림 3.16에서 왼쪽은 성능이 좋은(학습 데이터가 많은) AI feedback을 적용하여 한번에 정확한 예측을 통해 문제 상황을 해결할 때의 FRI 변화 그림이다. 오른쪽은 성능이 낮은(학습 데이터가 적은) AI feedback을 적용하여 첫 예측값이 한계 FRI에 근접은 했으나 문제 상황을 해결하지 못한 경우 step feedback을 적용하여 문제를 해결하는 모습을 모식화한 것이다.

제4장 연구 결과

본 연구는 현장 저압 막여과 공정에서 기술자가 데이터를 읽고 문제 상황을 판단한 후 조치를 취하는 과정에서 걸리는 시간을 줄여 안정적인 막여과 공정 운영을 위하여 문제 상황을 인지하면 자동으로 플럭스를 조절하여 파울링이 비정상적으로 많이 발생하는 문제를 해결하는 알고리즘을 만드는 연구를 진행하였다.

4.1 저압 막여과 공정 시뮬레이터

식 (3.4) 를 사용하여 매 사이클마다 여과압력-역세압력을 계산하여 저압 막여과 공정 시뮬레이터를 만들었다. 저압 막여과 공정 시뮬레이터는 원수 탁도, 원수 TOC, 플럭스, 여과시간, 사이클 수를 입력하면 매 사이클마다의 여과-역세 차압을 출력하는 동시에 파울링의 영향으로 인해 시간이 지남에 따라(사이클이 진행됨에 따라) 차압이 상승하는 경향도 나타내고 있다.

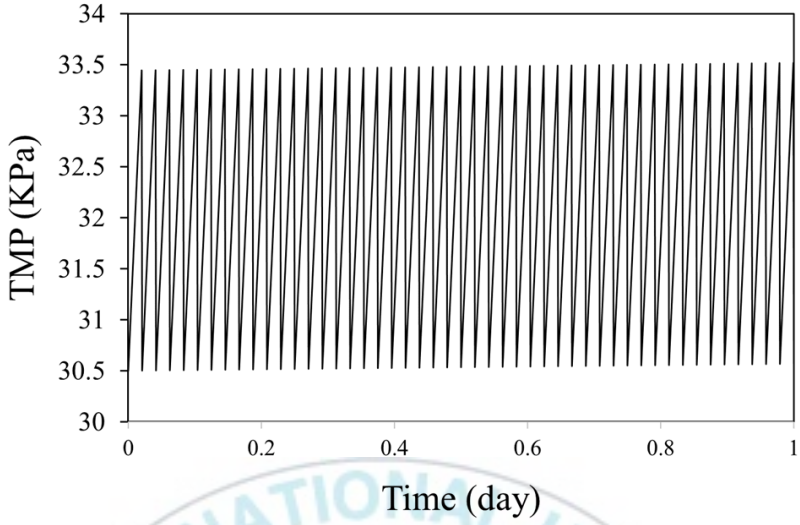

그림 4.1 여과-역세별 차압 변동

그림 4.1은 저압 막여과 공정 시뮬레이터가 작동하며 여과-역세 시 차압의 변화를 그래프로 나타낸 것이다.

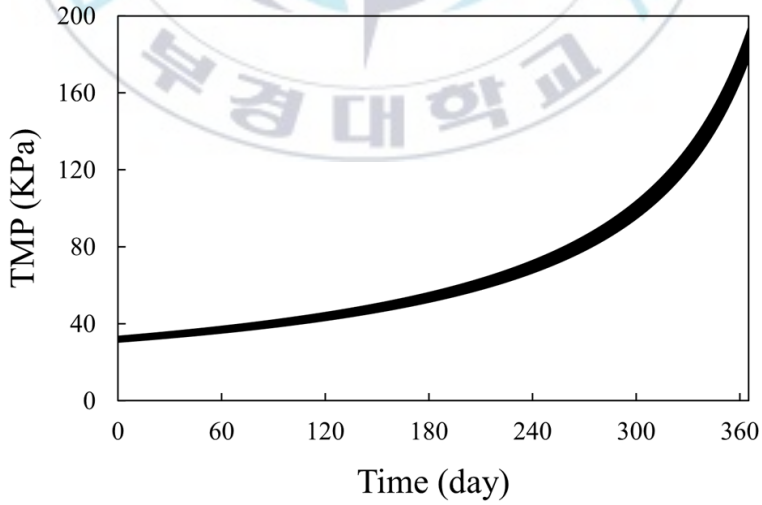

그림 4.2 파울링에 의한 차압상승 효과

그림 4.2는 저압 막여과 공정 시뮬레이터가 작동하며 시간이 지남에 따라 비가역적 파울링의 영향으로 인해 차압이 상승하는 모습을 나타낸 것이다.

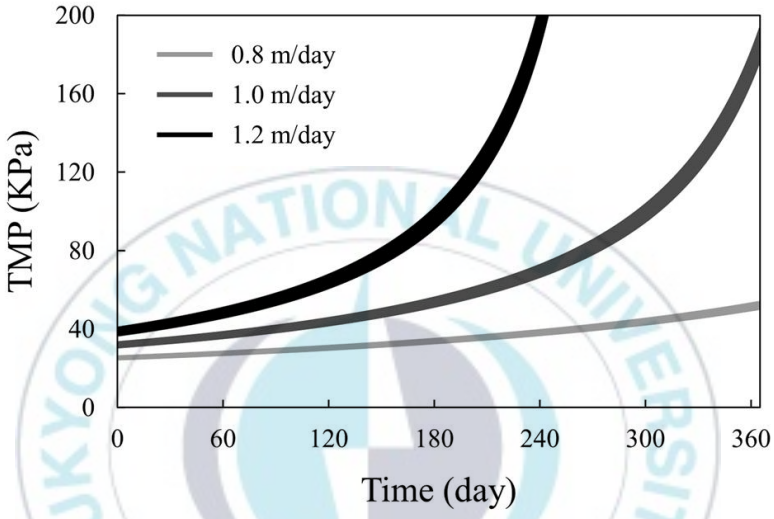

그림 4.3 플럭스에 대한 시뮬레이션 결과 변화

그림 4.3을 보면 입력된 플럭스에 따라 파울링이 발생하는 정도가 달라 차압이 상승하는 경향도 달라지는 것을 확인할 수 있다. 플럭스가 낮을 때보다 플럭스가 높을 때 시간당 유량이 많기 때문에 파울링이 많이 발생한다.

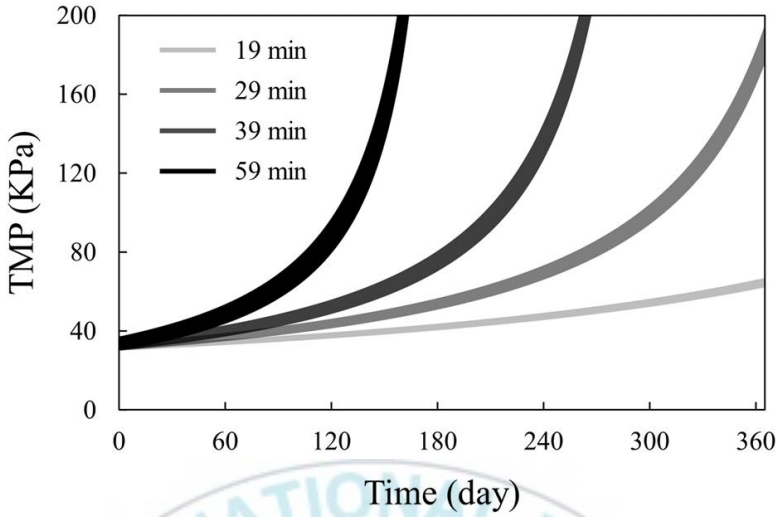

그림 4.4 여과시간에 대한 시뮬레이션 결과 변화

그림 4.4를 통해 역세시간이 동일할 때, 여과시간이 길어질수록 파울링이 많이 발생하는 것을 확인할 수 있다.

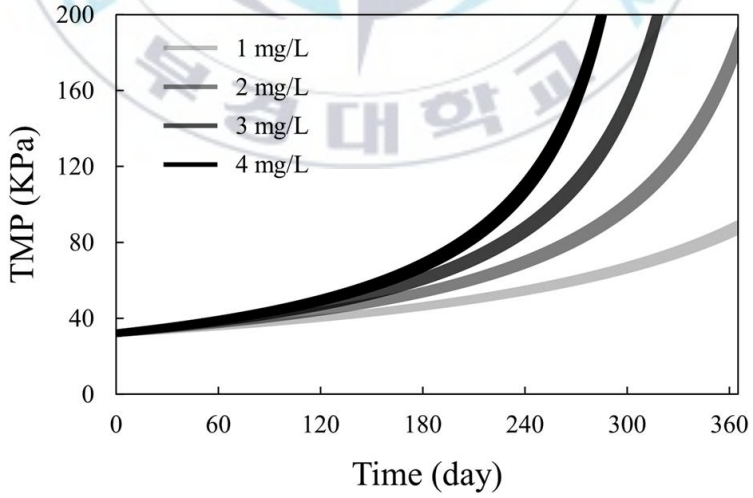

그림 4.5 원수 TOC에 대한 시뮬레이션 결과 변화

그림 4.5를 통해 원수 TOC가 높을수록 파울링이 많이 발생한다는 것을 확인할 수 있다.

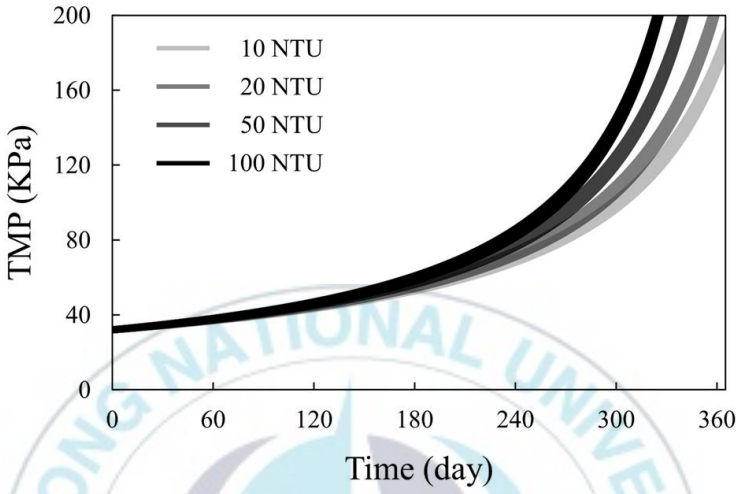

그림 4.6 원수 탁도에 대한 시뮬레이션 결과 변화

그림 4.6 에서는 원수 탁도가 높을수록 파울링이 많이 발생한다는 것을 알 수 있다. 이상 그림 4.1 ~ 4.6 을 통해 저압 막여과 공정 시뮬레이터가 정상 작동됨을 확인했다.

4.2 Step feedback

Step feedback이 적용되는 대표적인 두 가지 사례를 생각해 볼 수 있다. 첫 번째로 막여과 공정 운전을 시작할 때 초기 플럭스를 너무 높거나 너무 낮게 잘못 입력한 경우이다. 초기 플럭스를 잘못 입력한 경우 장기간 운전 양상이 어떻게 되는지는 이미 그림 3.4 에서 다른 입력조건들은 고정하고 플럭스만 적정 플럭스인 1 m/day와 0.8 m/day, 1.2 m/day를 입

력했을 때의 차이를 비교했었다. 다음은 step feedback을 적용한 경우에 운전 초반 플럭스의 변화를 나타낸 그래프이다.

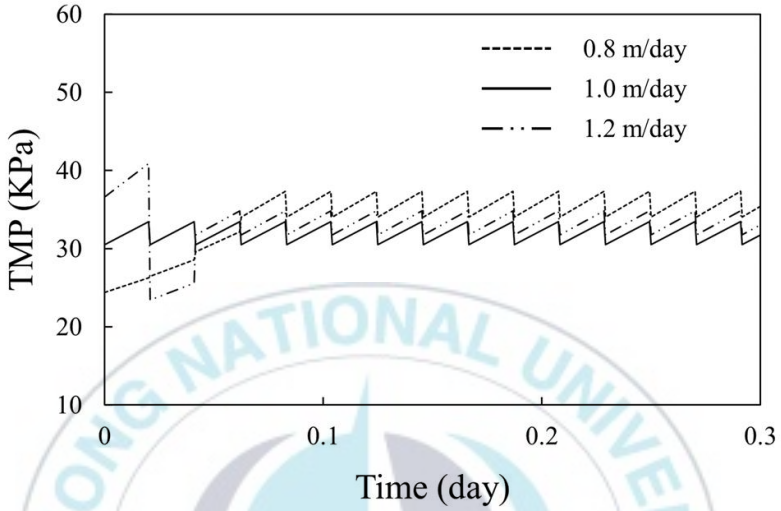

그림 4.7 Flux의 부적절한 입력에 따른 step feedback 결과

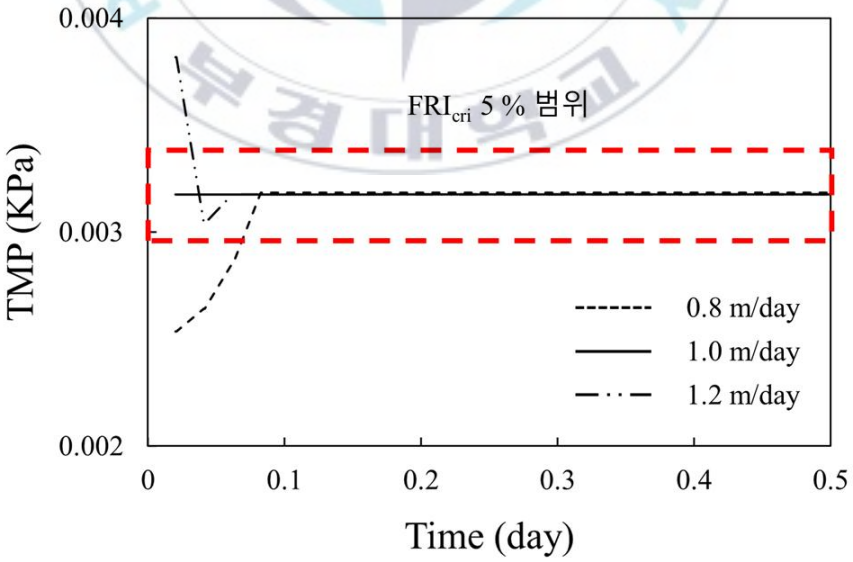

그림 4.8 Step feedback의 FRI graph

그림 4.7, 4.8 과 같이 막여과 공정 운전을 시작할 때 플럭스를 부적절하게 입력한 경우 FRI가 FRI_{cri}의 5 % 범위 안으로 들어오도록 막여과 운전 초기에 플럭스가 조절되어 한계 FRI와 거의 같아지는 것을 확인할 수 있다.

두 번째로는 입력조건은 동일하나, 막여과 공정을 운전하는 중 원수 조건이 변하는 경우이다. 앞서 '기준' 원수 수질조건은 원수 탁도 10 NTU, 원수 TOC 2 ppm 이었다. 원수 수질을 무작위로 변화시키면서 적정 플럭스로 피드백 없이 운전하는 경우 다음과 같은 그래프가 나타난다.

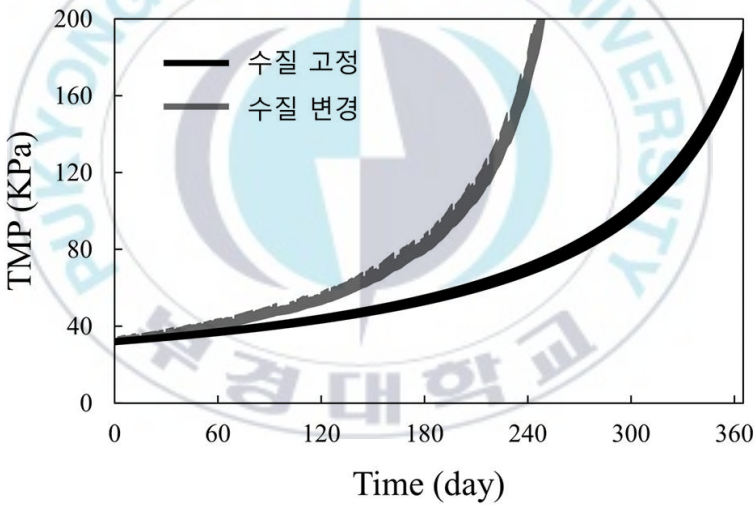

그림 4.9 수질이 변경되는 경우 차압 상승의 변화

수질은 이틀(96 사이클)에 한번씩 불규칙하게 변경되도록 했다. 그 결과 기존 1년에 200 KPa 상승하던 것이 수질이 악화되는 경우 약 240 일 정도에 200 KPa까지 도달하게 되었다. 실제로는 수질이 이틀에 한번 바뀌는 정도가 아니라 실시간으로 변하게 되기 때문에 이보다 더 가파른 차

압 상승을 보일 수 있다. 이와 같은 조건에 step feedback을 적용하면 다음과 같은 결과를 확인할 수 있다.

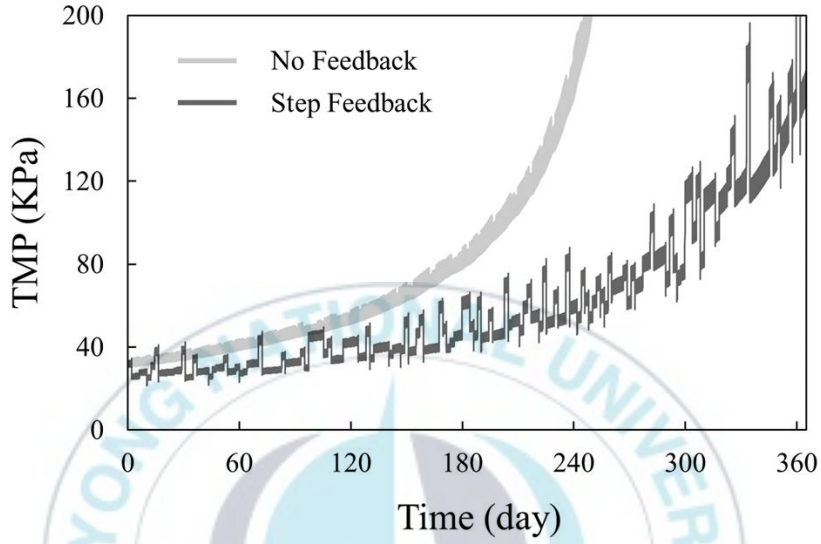

그림 4.10 수질 변동, step feedback 적용

그림 4.10 과 같이 기존 240일 정도에 차압이 200 KPa 에 도달하던 것이 step feedback을 적용하니 1년동안 200 KPa에 도달하지 않게 되었다. 다음은 step feedback이 적용될 때 FRI 의 변화를 나타낸 그래프이다.

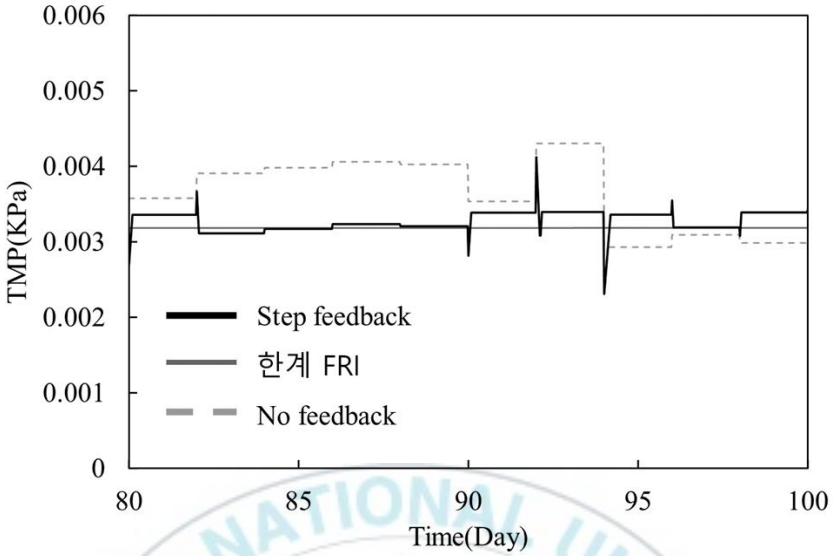

그림 4.11 Step feedback 적용 시 FRI 변화

Step feedback이 적용되면 기존 한계 FRI와 큰 차이가 있던 no feedback의 FRI가 한계 FRI 근처로 조정되는 것을 확인할 수 있다.

이와 같이 step feedback이 적용되면 원수 조건에 따른 적정 플럭스를 파울링 지수를 통해 계산하여 일정한 차압 상승 속도를 유지하는 동시에 최대 생산수량을 유지할 수 있도록 한다.

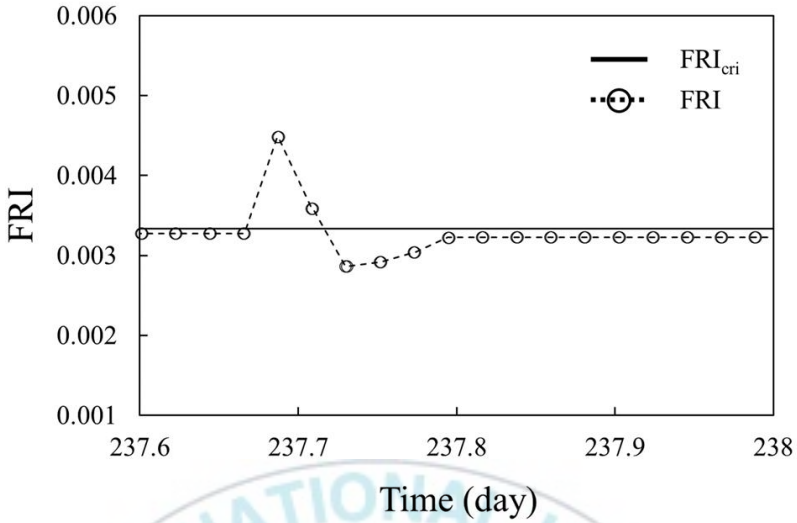

그림 4.12 Step feedback 적용 시 문제점

그림 4.12 에서 나타난 것과 같이 step feedback을 적용하면 FRI_{cri} 의 5 % 안에 들어올 때 까지 계속해서 피드백이 진행된다. 그림 4.12 에서 문제상황 발생부터 해결까지 총 6 사이클이 걸렸다. 플럭스를 내리는 과정에서 한번 내린 플럭스가 FRI_{cri}의 5 % 범위 보다 커서 에 들어오지 못해서 한번 더 플럭스를 내렸고, 그 결과 이번에는 5 % 범위보다 작아지게 되어 플럭스를 올리는 피드백을 수행하게 되었다. Step feedback에서는 이와 같은 무의미한 작업이 추가로 수행되고 있다. 이것은 안정적이고 빠른 해결을 위한 본 연구의 목적과 맞지 않는 모습을 보여주고 있다.

그림 4.10 에서 step feedback에 score function을 적용하면 다음과 같은 결과를 얻을 수 있다.

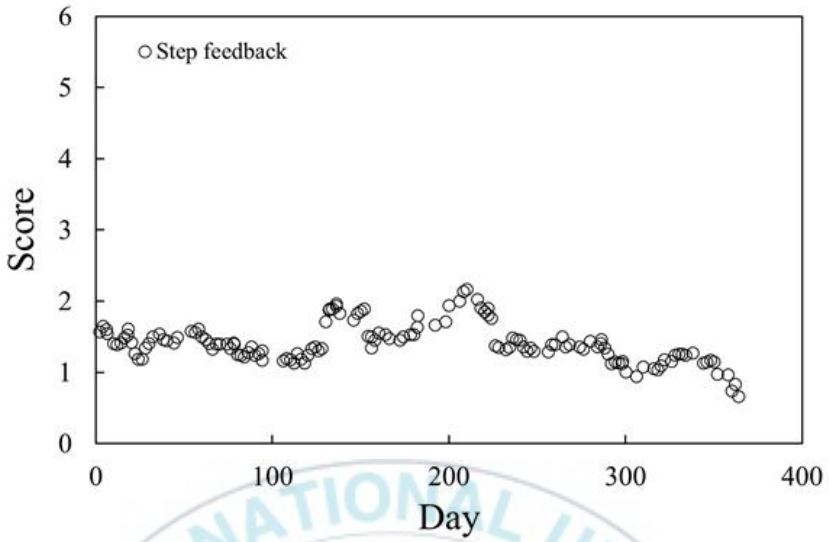

그림 4.13 Step feedback의 score function

Step feedback이 시작될 때의 사이클과 종료될 때의 FRI 를 데이터화 하여 score function을 적용해 점수를 구한다. 이후 점수가 어떻게 변하는지 경향을 찾기 위하여 20개씩 이동 평균을 구해 그래프를 그렸다. 그 결과 그림 4.10 에 나타난 step feedback의 성능은 잠깐 3점대를 보이긴 하지만 대체적으로 2점과 1점의 성능을 보이고 있음을 확인할 수 있다.

4.3 AI feedback

AI feedback의 score와 step feedback의 score를 비교하면 다음과 같이 나타난다.

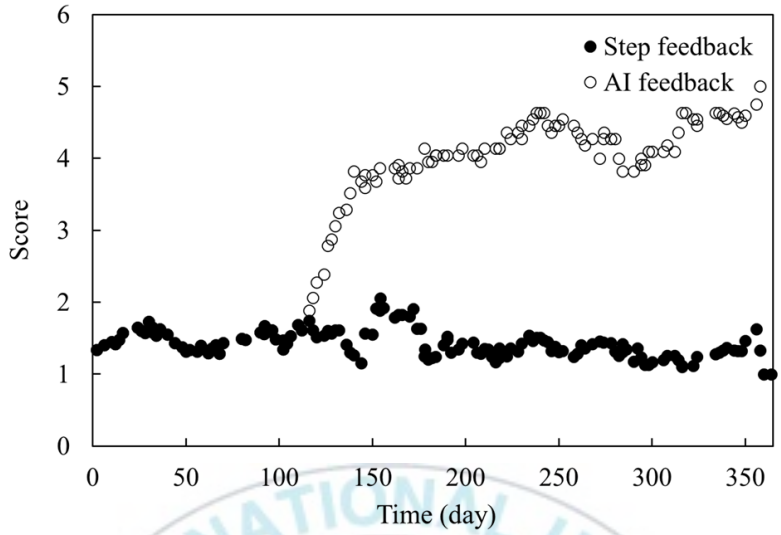

그림 4.14 AI feedback과 step feedback의 score funtion

막여과 공정 운전 초기에는 step feedback으로 진행되기 때문에 피드백 점수가 동일하다. 이후 AI feedback이 실행되면서 피드백 점수 차이가 벌어지기 시작한다. Step feedback의 점수는 1 ~ 2 점대가 유지되고 있는 반면, AI feedback의 점수는 빠르게 점수가 상승하여 4 ~ 5점대 까지 점차 점수가 상승할 것이다. 이처럼 피드백 점수에서는 AI feedback이 step feedback 보다 좋은 점수가 나오는 것을 확인했다.

다음은 그림 4.12 처럼 피드백이 발생하는 구간을 비교한다.

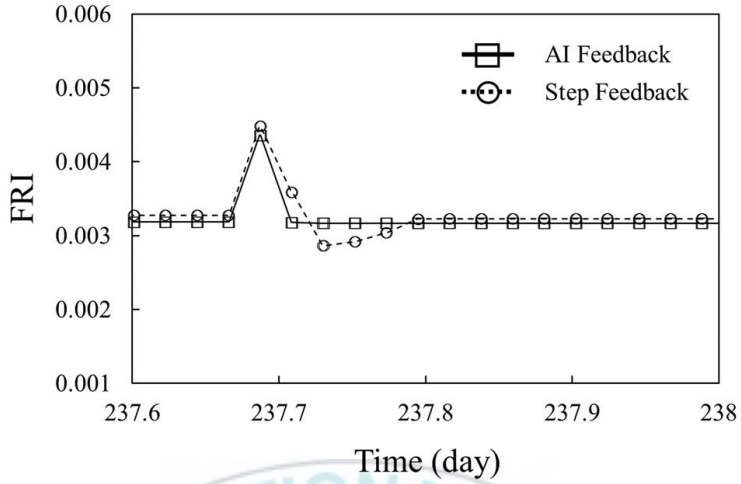

그림 4.15 AI feedback 과 step feedback의 비교

기존 step feedback 에서는 6 사이클이 걸려야 해결되던 문제가 AI feedback 에서는 1 사이클 만에 해결되는 것을 확인할 수 있었다.

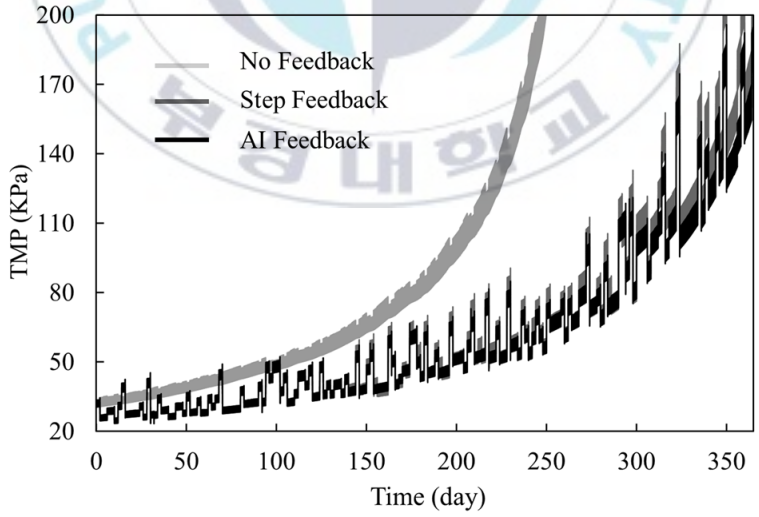

그림 4.16 1년 동안 step feedback과 AI feedback의 비교

1년 동안 운전했을 때 차압 그래프를 그려보면 365일이 되기 조금 전에 200 KPa 에 도달하는 step feedback에 비해 365이 될 때 200 KPa 에 도달하는 AI feedback을 확인할 수 있다. AI feedback이 step feedback에 비해 파울링 억제를 통한 차압 조절을 더 효과적으로 잘 하고있기 때문이다.

빠르게 step feedback에서 AI feedback으로 전환하기 위해 학습 데이터의 수를 50개에서 10개로 줄였다. 이 때 학습데이터가 적을수록 NRMSE가 낮으므로, 성능을 개선하기 위해 강화학습을 도입하여 지속적으로 기계학습 모델을 업데이트 한다. 학습 데이터 50개와 학습 데이터 10개일 때 강화학습을 도입한 모델의 NRMSE 변화가 다음과 같다.

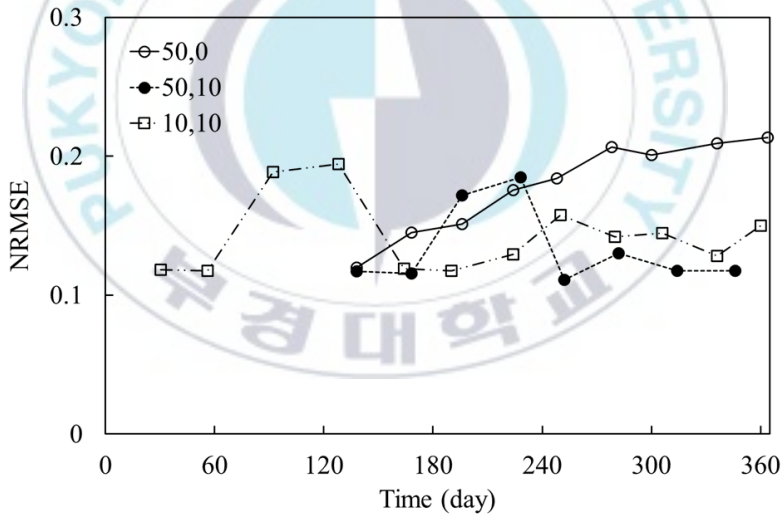

그림 4.17 학습 데이터 별 NRMSE 변화

그림 4.17에서 [50,0]은 기존 강화학습을 도입하지 않은 학습 데이터 50개 일때의 기계학습 모델이며, [50,10]은 학습데이터 50개일 때 다시 피드백 데이터 10개가 쌓일 때 마다 강화학습을 시행한 데이터이고 [10,10]

은 학습데이터 10개일 때 다시 피드백 데이터 10개가 쌓이면 강화학습을 시행한 데이터이다. 그림 4.17에서 알 수 있는 것은 기존 방식대로 강화학습을 적용하지 않으면 시간이 지날수록 NRMSE가 상승하지만, 강화학습이 시행되면 학습 데이터가 적어서 강화학습 초반에 NRMSE가 높더라도 계속해서 학습이 시행되면 NRMSE가 낮아진다는 것을 확인할 수 있다. [50,10]과 [10,10]의 NRMSE를 비교하면 다음과 같다.

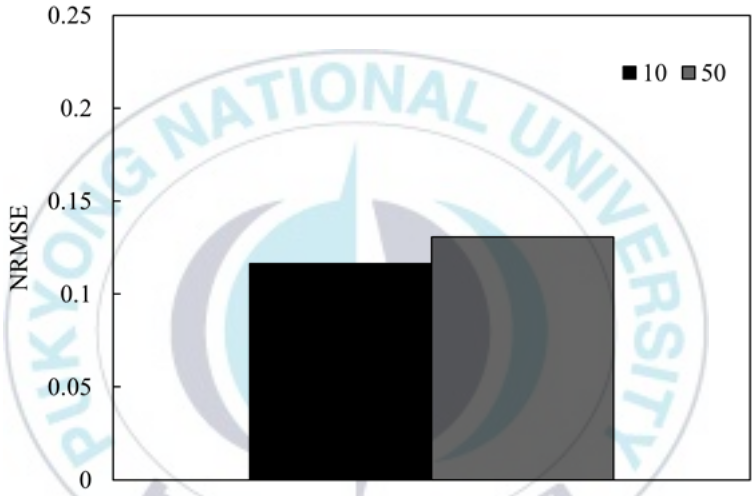

그림 4.18 학습 데이터별 강화학습 적용 시 최종 NRMSE 비교

학습 데이터가 10일 때 강화학습을 시행한 데이터가 더 낮은 NRMSE를 보이고 있다. 그러나 기계학습 모델은 학습 데이터에 따라 성능이 미세하게 변할 수 있으므로 이 정도면 NRMSE 차이가 거의 없다고 할 수 있다. Score function 그래프로 확인하면 다음과 같다.

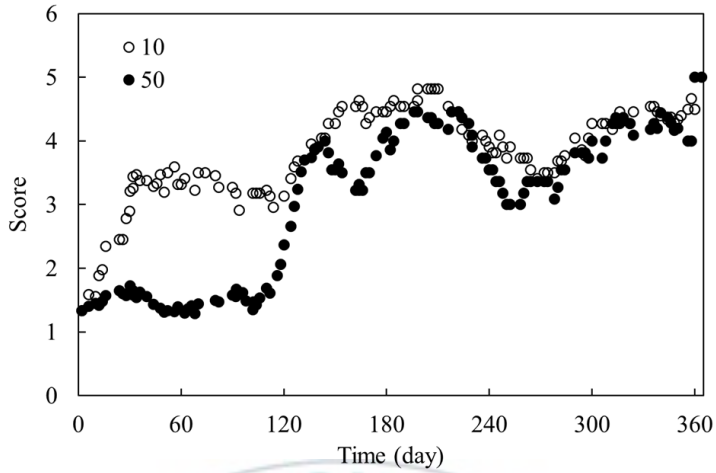

그림 4.19 학습 데이터 10개와 50개의 score function 비교

학습 데이터 50개일 때에 비해 학습 데이터 10개일 때는 매우 빠르게 학습이 시작되어 3점대로 상승, 이후 계속된 강화학습을 통해 score가 지속적으로 높아지고 있다. 학습 데이터가 50개 모여서 기계학습 모델을 구성했을 때에는 이미 5번의 강화 학습이 발생하여 성능이 비슷해진 것을 확인할 수 있다. 그림 4.20를 통해 학습 데이터가 적을 때에는 강화학습을 시행하면 성능을 지속적으로 상승시킬 수 있다는 것을 알 수 있는 동시에, 그림 4.17과 비교하여 학습 데이터가 많을 때에는 강화 학습을 시행하면 오히려 성능이 다시 나빠질 수 있다는 것을 알 수 있다. 따라서 학습 데이터 10개로 학습을 시작하여 강화학습을 시행하는 것이 가장 적합하다.

강화학습이 시행될 때 마다 모델이 업데이트되며 성능이 변한다. 강화학습 모델별 성능을 확인하기 위해 모델별 NRMSE를 계산했다.

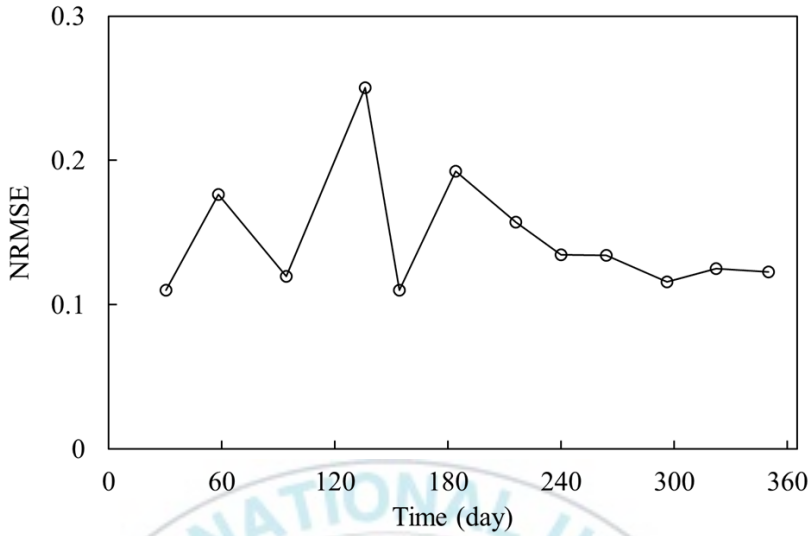

그림 4.20 학습데이터 10일 때 모델별 NRMSE 변화

이 데이터는 각 모델마다의 학습데이터를 학습시킨 모델의 입력값으로 넣어 FS를 예측하여 NRMSE를 계산한 결과이다. 이 데이터에 따르면 첫 모델보다 두 번째 모델의 NRMSE가 더 높은것을 알 수 있다. NRMSE는 낮을수록 성능이 더 좋다는 뜻인데, 첫 모델이 두 번째 모델보다 성능이 좋은 것인지 확인하기 위해 두 번째 모델의 학습 데이터를 첫 번째 모델에 입력하여 FS를 예측한 후 NRMSE를 계산하여 첫 번째 모델과 두 번째 모델의 성능을 비교했다.

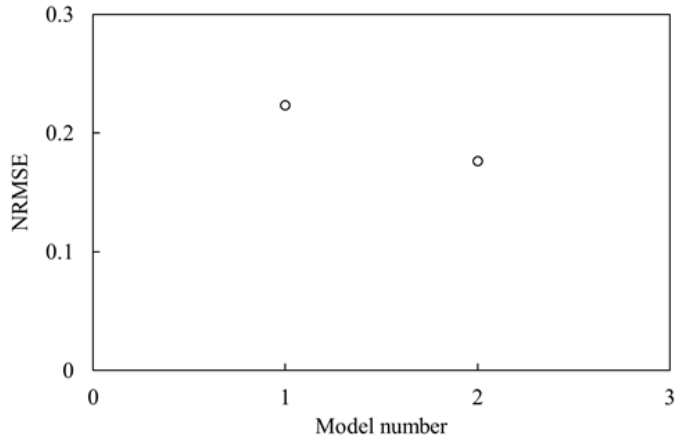

그림 4.21 첫번째 모델과 두번째 모델의 성능비교

첫 번째 모델의 NRMSE가 더 높기 때문에 첫 번째 모델의 성능이 더 좋지 않음을 확인할 수 있다. 그림 4.20에서 첫 번째, 두 번째 모델과 같은 경향을 보인 세 번째 모델과 네 번째 모델을 같은 방법으로 비교하면 다음과 같다.

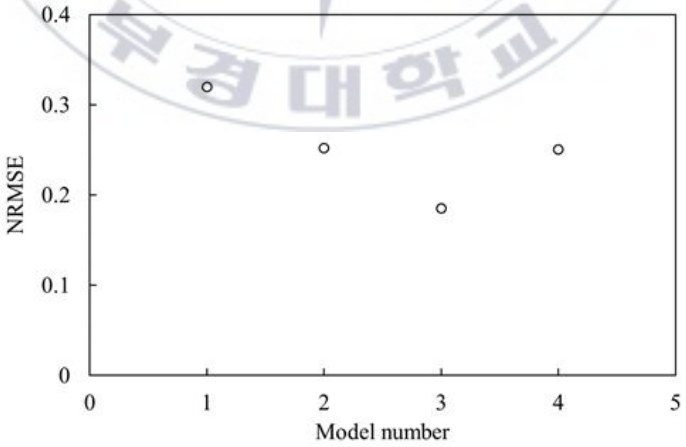

그림 4.22 첫번째 모델 ~ 네번째 모델의 비교

이번에는 세 번째 모델의 성능이 네 번째 모델의 성능보다 더 좋았음을 확인할 수 있다. 이번에는 마지막 모델을 구성한 데이터를 사용해 모든 모델의 예측 성능을 확인했다.

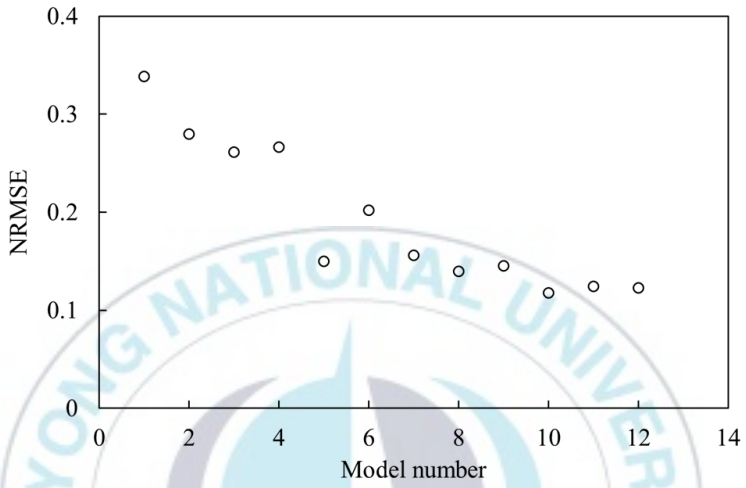

그림 4.23 모든 모델의 성능 비교

NRMSE가 계속해서 낮아지다 최종적으로 0.1 정도에서 수렴되는 경향을 나타내고 있다. NRMSE가 더 낮아지지 않는 것은 이전 모델의 FS 를 통해 구성된 새 모델의 FS가 같을 수 없기 때문에 생기는 절대적인 차이로 판단된다. 모델 별 score 점수를 나타내면 다음과 같다.

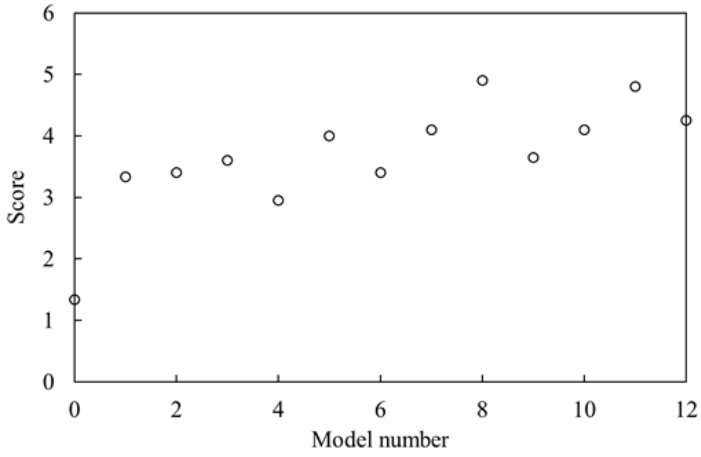

그림 4.24 모델 별 score 점수

NRMSE와 마찬가지로 학습이 반복될수록 score 성능이 높아지는 것을 확인할 수 있다.

제5장 결론

5.1 연구 결과 요약

(1) 연구를 위해 6개월 ~ 1년의 많은 시간이 소요되는 장기 파일럿 테스트를 통하여 적정 플럭스를 찾는 것이 현실적이지 못해서 Python 언어를 사용한 저압 막여과 차압상승 시뮬레이터를 만들어 적정 플럭스를 찾는 과정을 진행했다.

(2) 적정 플럭스의 기준을 명백히 한 뒤 저압 막여과 차압상승 시뮬레이터를 통해 적정 플럭스를 구했다. 적정 플럭스의 기준은 원수 탁도 10 NTU, 원수 TOC 2 ppm, 여과시간 29분, 역세시간 1분 조건에서 일 년 동안 저압 막여과 공정을 운전했을 때 막간 차압이 200 KPa 가 되는 플럭스로 정했으며, 그 플럭스는 1 m/day로 구해졌다.

(3) 파울링이 비정상적으로 발생하는 문제 상황을 정의하기 위하여 파울링 지수(FRI)를 도입했다. 적정 플럭스일 때의 FRI를 한계 FRI로 하여, 기준과 다른 조건에서 저압 막여과 차압상승 시뮬레이션을 실시했을 때 매 사이클마다 계산되는 FRI가 한계 FRI의 5 % 범위를 벗어났을 때 플럭스를 내리거나 올린다. 이 때 플럭스 조절이 일정한 간격으로 실시되기 때문에 step feedback 이라 한다.

(4) 피드백 성능을 확인하기 위해 score function을 도입했다. Score function은 step feedback을 통해 변한 플럭스로 계산된 FRI가 얼마나 한계 FRI와 근접해 있는가를 계산하여 점수를 0점부터 5점까지 매기는 역

할을 한다. Step feedback의 피드백 점수는 1 ~ 2 점대로 나타났으며, 그 성능이 탁월하지 않다는 것을 확인할 수 있었다.

(5) 피드백 성능 향상을 위해 딥 러닝을 도입하여 초기에는 step feedback으로 운전하다 일정량 데이터가 축적되면 기계 학습을 실시하여 예측 모델을 구성해 AI feedback으로 전환한다. 여러가지 조건을 비교하는 과정을 통해 구한 기계학습 모델 최적화 조건은 은닉층 깊이 2층, 노드 수 64개, 활성화 함수 ReLU, 학습횟수 200회, 학습 데이터 50개로 구성된 모델이었다.

(6) 학습 데이터 50개를 모으는 시간이 너무 길다고 판단되어 학습 데이터를 10개로 줄인 후 기계학습 모델을 구성하여 AI feedback을 진행하다 데이터가 일정량 다시 모이면 모델에 사용되었던 학습 데이터와 새로 수집한 데이터를 모두 학습시킨 모델을 새로 구성한다. 이 과정을 강화학습이라고 하며, 강화학습으로 구성된 기계학습 모델은 강화학습이 거듭될수록 첫 데이터의 수를 많게하여 구성한 기계학습 모델보다 성능이 좋아지는 것을 확인할 수 있었다.

본 연구에서는 실제 저압 막여과 공정에서 안정적인 운전을 위해서는 파울링 속도를 제어하기 위한 적절한 피드백이 필요하며, 파울링 지수를 계산하여 플럭스를 조절하는 step feedback을 통해 안정적인 운전이 가능하나 딥 러닝을 사용해 구성한 기계학습 모델을 통해 AI feedback을 적용하면 step feedback보다 더 좋은 성능의 피드백을 적용할 수 있다. 또한 학습 데이터를 적게하여 AI feedback을 적용한 후 구해지는 피드백 데이터를 사용하여 강화학습 모델을 적용하면 훨씬 더 안정적인 운전이

가능하다는 것을 확인할 수 있었다.

기존 저압 막여과 공정은 특히 규모가 작을수록 공정의 자동제어가 필요했다. 그러나 현재까지 자동제어는 기술자가 반드시 필요한 반쪽짜리 자동제어였다. 본 연구를 통해 기존 기술자가 필수적이던 저압 막여과 공정에 기계학습을 도입하여 무인 자동화 공정이 구현 가능함을 확인할 수 있었다.

5.2 향후 연구 내용

시뮬레이터에서는 기계학습을 통한 AI feedback이 잘 적용되어 저압 막여과 공정의 자동제어가 구현 가능함을 확인했다. 하지만 변수 통제가 가능한 시뮬레이터에서 실시된 연구로는 실제 상황에 적용이 가능할 것이라는 확신이 불가능하며, 연구의 최종 목적인 실제 저압 막여과 공정의 자동제어를 구현하기 위해서는 추후 저압 막여과 파일럿 플랜트를 설치하고 PC를 통한 자동, 원격제어가 가능하도록 하여 변수 통제가 힘든 실제 상황에서도 AI feedback이 제대로 작동하도록 하는 연구가 필요할 것이다.

참고문헌

두산중공업 (2016) 해수담수화 플랜트 고효율운영 및 유지관리 최적화 기술 개발 보고서.

김수한 (2013) 정유량 막여과 파울링 모델을 이용한 막여과 정수 플랜트 공정 진단 기법, 대한상하수도학회지, Vol.27, No.1, 139-146.

한국수자원공사 (2011) 막여과공정 설계·운영관리 매뉴얼, 한국수자원공사.

한국상하수도협회 (2010) 환경부 제정 상수도시설기준, 건설도서.

김종두 (2008) 정밀여과막 시스템을 이용한 정수처리의 최적운전조건 도출, 서울시립대학교 환경공학과 석사논문.

유명진, 방기웅, 구자용, 명규남, 이준호, 정신호, 구윤희, 박귀수, 장미정 (2006) 상수도 공학 -계획, 설계 및 운전-, 동화기술.

김수한 (2003) Polydispersity 조건에서 Crossflow 방식 정밀여과의 케이크 형성 특성 및 제어방안에 관한 연구, 한국과학기술원 건설 및 환경공학과 박사논문.

이성우, 이현동, 한명호, 곽동희, 김충환 (2003) 고도상수처리 -원리 및 응용-, 동화기술.

정철우, 한승우, 강임석 (2002) 막의 재질에 따른 유기물질 성상별 흡착 특성, 대한환경공학회지 논문, Vol.24, No.8, 1339-1348.

배현웅 (2001) (엑셀을 이용한)통계학의 기초와 활용기법, 교우사.

Hiroyuki et al. (1998) 막여과를 이용한 정수처리방법의 연구・개발상황과 과제, 첨단환경기술, 제4권, 제5호, 11-19.

Cha, D., Park, H., Kim, S., Lim, J.L., Kang, S., and Kim, C.H. (2012) A statistical approach to analyze factors affecting silt density index, *Desal. Water Treat.*, 45, 276 - 283.

Park et al., (2012) The effect of fluctuation in flow rate on the performance of conventional membrane water treatment for a smart water grid, Desalination and Water Treatment 47, 17-23

Jin, X., Jawor, A., Kim, S., and Hoek, Eric M.V.. (2009) Effects of feed water temperature on separation performance and organic fouling of brackish water RO membranes, *Desalination*, 239, 346-359.

Katsoufidoua K., Yiantsiosa S.G, and Karabelas A.J. (2008) An experimental study of UF membrane fouling by humic acid and sodium alginate solutions: the effect of backwashing on flux recovery, *Desalination*, 220, 214-227.

Mueller, U., Witte, M. (2008) Ceramic membrane application for spent filter backwash water treatment, Techneau.

Sharma, R.R. (2005) Temperatue effects on transport of water, charged, and uncharged solutes across polymeric thin film composite nanofiltration membranes: an investigation into pore-transport mechanism s and electrokinetic properties, A Dissertation Presented to the Faculty of The Department of Civil and Environmental Engineering University of Houston.

Lim A.L., Bai Renbi (2003) Membrane fouling and cleaning in microfiltration of activated sludge wastewater, *Journal of membrane*

science, 216, 279-290.

Fan, L., Harris, J.L., Roddick, F.A., and Booker, N.A. (2001) Influence of the characteristics of natural organic matter on the fouling of microfiltration membranes, *Wat. Res.*, Vol. 35, No. 18, 4455-4463.

Schafer, A.I., Fane, A.G., and Waite, T.D. (2000) Fouling effects on rejection in the membrane filtration of natural waters, *Desalination*, 131, 215-224.

Hong, S., Elimelech, M. (1997) Chemical and physical aspects of natural organic matter (NOM) fouling of nanofiltration membranes, *Journal of membrane science*, 132, 159-181.

Belfort R., Robert H.D., and Andrew L.Z. (1994) Review: The behavior of suspensions and macromolecular solution in crossflow microfiltration, *Journal of Membrane Science*, Vol. 96: 1-58.

Glater, J., Hong, S.K., and Elimelech, M. (1994) The search for a chlorine-resistant reverse osmosis membrane, *Desalination*, 95, 325-345.

Juncker C., Clark M.M. (1994) Adsorption of aquatic humic substances on hydrophobic ultrafiltration membranes, *Journal of membrane science*, 97, 37-52.

Crozes, G., Anselme, C., and Mallevialle, J. (1993) Effect of adsorption of organic matter on fouling of ultrafiltration membrane, *Journal of membrane science*, Vol. 84, Issues 1-2, 61-77.

WHO (1993) Guidelines for drinking water quality.

Flemming, H.C., Schaule, G. (1988) Biofouling on membranes-A

microbiological approach, *Desalination*, Vol. 70, Issues 1-3, 95-119.

Thurman E.M. (1985) Organic Geochemistry of natural water. Boston, MA: Martinus Nijhoff, Dr W. Junk Publishers.

Potts, D.E., Ahlert R.C., and Wang S.S. (1981) A critical review of fouling of reverse osmosis membranes, *Desalination*, 36, 235-264.

Carroll, D. (1970) Clay minerals: a guide to their X-ray identification, The Geological Society of America, INC, Colorado.

Available from <https://en.wikipedia.org/wiki/Deep_learning> [Accessed 20 December 2020]

Available from <https://www.tensorflow.org/> [Accessed 20 December 2020]

Available from <https://colab.research.google.com/notebooks/intro.ipynb> [Accessed 20 December 2020]

Available from <http://www.daejungchem.co.kr/main/main.asp> [Accessed 20 December 2020]

Available from <https://www.shimadzu.com/> [Accessed 20 December 2020]

Available from <https://www.ysi.com/> [Accessed 20 December 2020]