PART 1

단층 퍼셉트론(SLP)

1장 회귀분석

2장 이진판단

3장 선택분류

딥러닝 & 강화학습 담당
이재화 강사

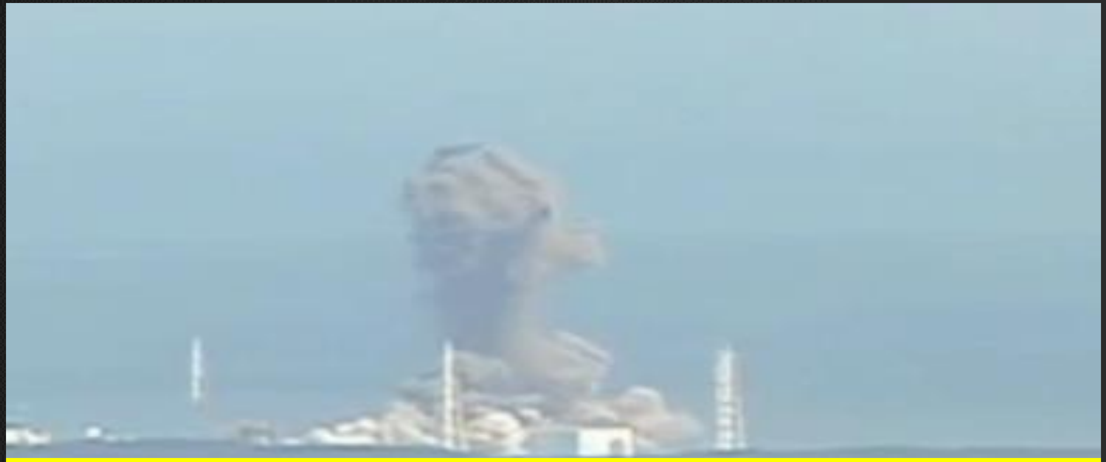
이 장에서 다룰 내용

1. 불량 철판 판별 문제
2. 선택분류 문제의 신경망 처리 방법
3. 소프트맥스 함수와 소프트맥스 교차 엔트로피
4. 시그모이드 함수와 소프트맥스 함수의 관계
5. 불량 철판 판별 신경망 구현과 실행

실습에 사용할 데이터셋 : faults.csv

3.1 불량 철판 판별 문제

2011.03 후쿠시마 원자력 발전소 사고

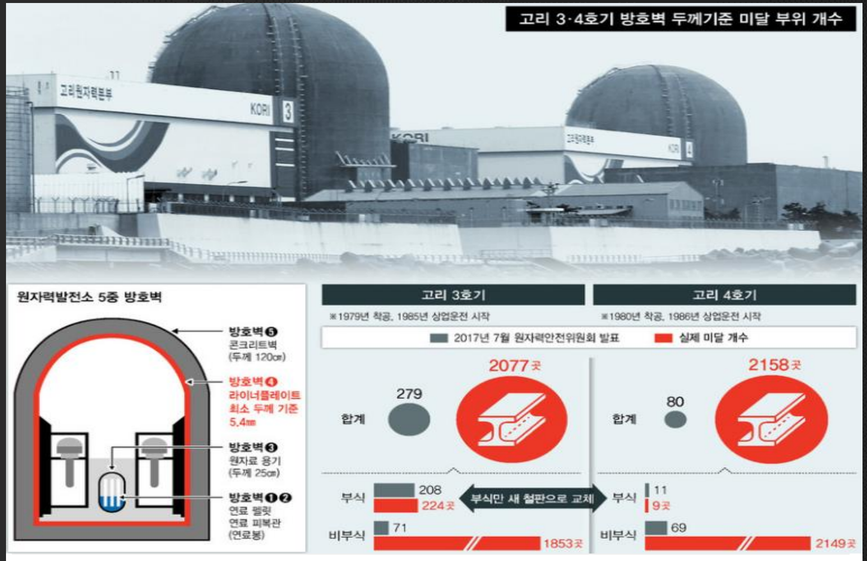


국제 원자력 사고 4~7등급 목록

[펼치기 · 접기]

4등급	셀라필드 원자력 단지 (1955~1979)	생로랑 원자력 발전소 사고 (1980)
	도카이 촌 방사능 누출사고 (1999)	
5등급	퍼스트 초크 강 사고 (1952)	윈드스케일 화재 (1957)
	스리마일 섬 원자력 발전소 사고 (1979)	고이아니아 방사능 유출사고 (1987)
6등급	키시팀 사고 (1957)	
7등급	체르노빌 원자력 발전소 폭발 사고 (1986)	후쿠시마 원자력 발전소 사고 (2011)

2018.05 고리원자력 발전소 불량 철판 문제



경제 산업·재계

[단독] 고리 3·4호기, 철판 4235곳이 '두께 불량'…30년간 몰랐다

고리원전 '불량 철판' 철저한 조사 촉구

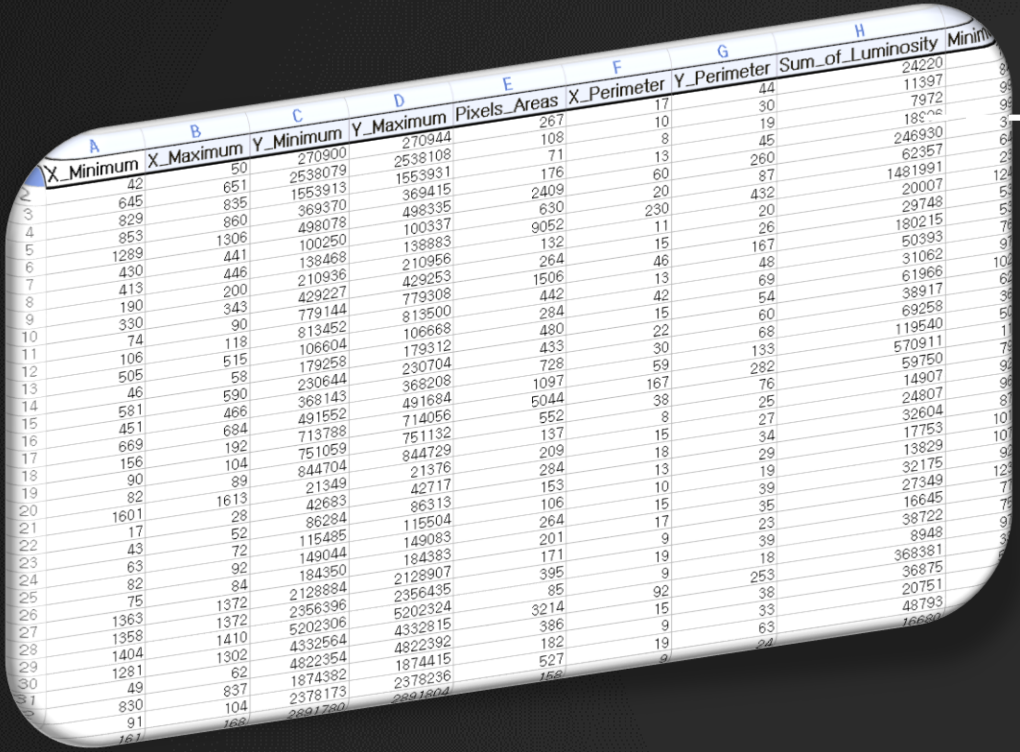
탈핵부산연대, 규탄 기자회견

시민단체, 원자력발전소 철판 '두께 불량' 대책 마련 촉구

2016년부터 부식 점검 시작했으나 올해까지 9998곳 발견, 안전성 조사 필요

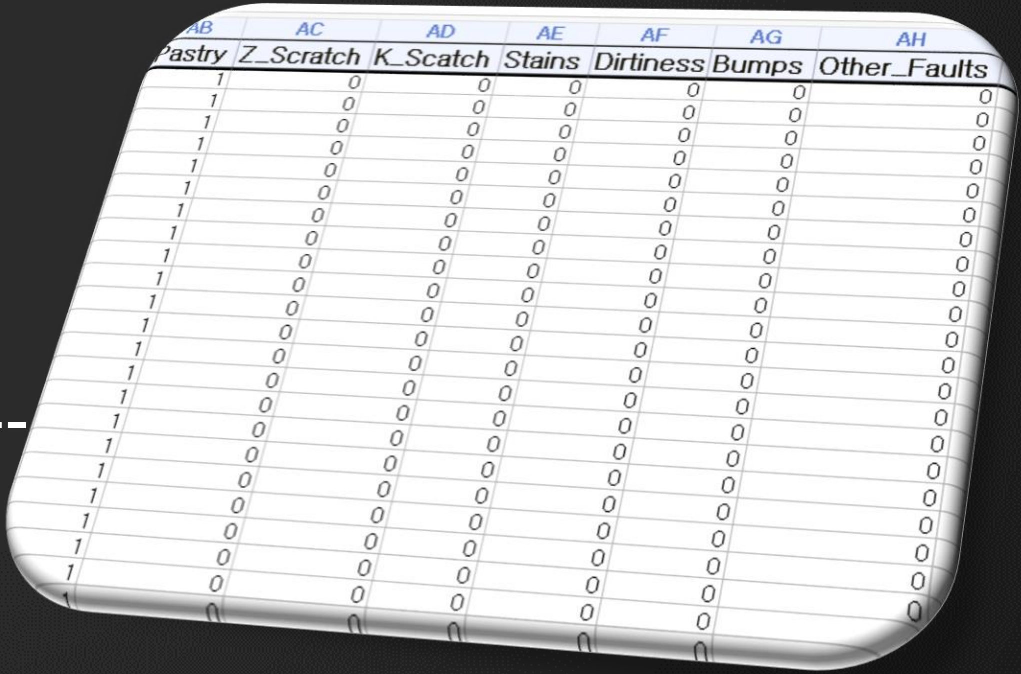
박호경 기자(=부산) | 2019-10-08 15:53:29 | 2019-10-08 15:53:38

3.1 불량 철판 판별 문제



steel_dataset

※ 27가지 특징값



one_hot vector

※ 7가지 철판 불량 상태

3.2 선택분류 문제의 신경망 처리

선택 분류 문제

: 몇 가지 정해진 후보 가운데 하나를 골라 답하는 문제.

: 이진판단과 성격이 비슷.

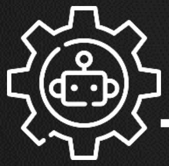
 # 각 후보 항목

A →

B →

C →

⋮



"선택분류에서는 'N개의 퍼셉트론'이 필요해!"

1 각 후보 항목에 대한 **『로그 척도』**의 **『상대적 추천 강도』** (로짓값 추정*)

2 퍼셉트론 하나가 후보 하나에 대한 **로짓값 출력***

로짓값

: 상대적인 가능성을 **『로그』**로 이용해 나타낸 값

: 로짓값의 표현 대상은 각 후보 항목을 답으로 추정할 확률

A → 3

B → 1

$$e^{3-1} = e^2 = 7.39$$

```python
print(np.exp(3-1))
```

```
7.38905609893065
```

→ A를 답으로 추정할 확률이 B보다 7.39배 크다.

A → 36.5

B → 34.5

$$e^{3-1} = e^2 = 7.39$$

→ A를 답으로 추정할 확률이 B보다 7.39배 크다.

★ 로짓값 사이의『차이』가 중요

※ 이진판단과 달리 거짓에 대한 로짓값을 0으로 간주하던 암묵적 요소 X

3.2 선택분류 문제의 신경망 처리

선택 분류 문제

: 몇 가지 정해진 후보 가운데 하나를 골라 답하는 문제.

: 이진판단과 성격이 비슷.

#오늘도 공부하는 라이언

선택분류만 생각한다면
로짓값들을 구하는 것으로 충분하지 않을까?!

#오늘도 공부하는 라이언

로짓값이 크면 자연스럽게 확률도 커지니까!
우린 그냥 가장 큰 로짓값만 선택하면 되는거 아냐?

#인공지능 비서 에이림

"아쉽게도 로짓값만으로는 마땅한 학습방법을
찾기가 힘들어… ㅠㅠ"

딥러닝 연구가 한동한 답보 상태에 빠졌습니다.

신경망이 추정한 확률분포

▼

교차 엔트로피

▲

정답이 나타내는 확률분포

#손실함수 삼아 학습을 수행하면
신경망의 선택을 점점 더 정답에 근접 시킬 수 있음

복수의 후보 항목들에 대한 로짓값 벡터를
확률분포 벡터로 변환하는 함수가 필요.

#softmax

이러한 확률분포와 정답에 나타난 확률분포 사이의
교차 엔트로피를 계산해주는 함수의 필요성

Part 1. 단층 퍼셉트론

3.3 소프트맥스 함수

#softmax

: 『로짓값 벡터』를 『확률분포 벡터』로 변환해주는 비선형 함수

1.1 로짓값들은 후보항들에 대한 선택확률을 『로그 척도』로 표현한 것 → #로짓값 벡터 1

1.2 로짓값 벡터에서 『최대항』이 어느 것인지만 확인하면, 신경망이 『어느 후보를 골랐는지 바로 확인 가능』

1.1 선택분류를 위한 손실함숫값도 이『로짓값 벡터』를 이용하면 바로 계산 가능

1.2 굳이 확률값으로 변환하지 않고 『로짓값으로도 순전파의 모든 처리 가능』

2.1 로짓값 벡터를 확률값 벡터 즉, softmax()를 거쳐야 하는 이유 :

역전파에서 『손실함수에 대한 편미분』을 구할 때 이 확률 분포가 필요

#확률값 벡터

2 #손실함수 편미분 계산

P_0.0	P_0.1	P_0.2	P_0.3	P_0.4	P_0.5	P_0.6
P_1.0	P_1.1	P_1.2	P_1.3	P_1.4	P_1.5	P_1.6
...	...	...	...	...	...	...

→ 확률분포, 즉 각 후보항목이 정답일 확률값들로 구성된 벡터

SOFTMAX()

1 #손실 함수 계산

#로짓값 벡터

출력 크기 [N,7]

L_0.0	L_0.1	L_0.2	L_0.3	L_0.4	L_0.5	L_0.6
L_1.0	L_1.1	L_1.2	L_1.3	L_1.4	L_1.5	L_1.6
...	...	...	...	...	...	...

가중치 : [27,7]

편향 : [7]

#퍼셉트론 P0 P1 P2 P3 P4 P5 P6

#데이터 벡터 [N,27]

D_0.0	D_0.1	D_0.2	...	...	...	D_0.24	D_0.25	D_0.26
D_1.0	D_1.1	D_1.2	...	...	...	D_1.24	D_1.25	D_1.26
...	...	...				...	...	...

미니배치 크기 : N

데이터 특성 : 27

3.3.0 소프트맥스 함수 일반식 도출과정

#softmax

소프트맥스 함수의 성질들을 이용한
「소프트맥스 함수의 일반식」

$$y_i = \frac{e^{x_i}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}$$

(일반식 : 계산과정에서 오류 발생)

$$y_i = \frac{e^{x_i - x_k}}{e^{x_1 - x_k} + e^{x_2 - x_k} + \cdots + e^{x_n - x_k}}$$

(변형식)

이번 월드컵 4강팀의 우승 가능성을
어떻게 예상하시나요?

네, 우선 로짓값으로
대한민국 2.0
미국 1.0
호주 1.2
독일 0.7점을 주겠습니다.

그 말인 즉,
대한민국은 47.9%, 미국은 17.6%
호주는 21.5%, 독일은 13.0% 으로
우승확률을 예상하는 거군요?!

 # 연산 과정

$$e^{2.0} : e^{1.0} : e^{1.2} : e^{0.7}$$

= 7.39 : 2.72 : 3.32 : 2.01

비율 유지 및
전체 합이 1이 되는 **비율**

= 0.479 : 0.176 : 0.215 : 0.130

$$y_i = \frac{e^{x_i}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}$$

소프트맥스

'소프트맥스'라는 단어는
max 함수와 비슷한 의미
(max 함수의 약화된 버전)

max함수와 비슷하게
큰 값을 강조, 작은 값은 약화
효과를 갖는다.

e를 밑으로 하는 지수함수를
취하기 때문.

3.3.1 소프트맥스 함수 변형식이 필요한 이유 그리고 도출과정

소프트맥스 함수의 성질들을 이용한
『소프트맥스 함수의 일반식』

$$y_i = \frac{e^{x_i}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}$$

(일반식 : 계산과정에서 오류 발생)

$$y_i = \frac{e^{x_i - x_k}}{e^{x_1 - x_k} + e^{x_2 - x_k} + \cdots + e^{x_n - x_k}}$$

(변형식)

문제발생 문제발생!

에러 1. overflow 발생 및 inf

```
np.exp(10000)
```

```
C:\Users\leeyua\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarning: overflow encountered in exp
  """Entry point for launching an IPython kernel.
inf
```

에러 2. 모든 값이 아주 큰 음수로 쏠리게 되면
분자, 분모 값 0 수렴 및 0 나눗셈 오류 발생

```
print(np.exp(-100))
print(np.exp(-1000))
```

```
3.720075976020836e-44
0.0
```

3.3.1 소프트맥스 함수 변형식이 필요한 이유 그리고 도출과정

그 말인 즉,
대한민국은 47.9%, 미국은 17.6%
호주는 21.5%, 독일은 13.0% 으로
우승확률을 예상하는 거군요?!

네, 우선 로짓값으로
대한민국 2.0
미국 1.0
호주 1.2
독일 0.7점을 주겠습니다.

소프트맥스 함수의 성질들을 이용한
『소프트맥스 함수의 일반식』

$$y_i = \frac{e^{x_i}}{e^{x_1} + e^{x_2} + \cdots + e^{x_n}}$$

(일반식 : 계산과정에서 오류 발생)

$$y_i = \frac{e^{x_i - x_k}}{e^{x_1 - x_k} + e^{x_2 - x_k} + \cdots + e^{x_n - x_k}}$$

(변형식)

 # 최댓값을 이용한 overflow 문제 해결

: 최댓값을 찾아 정의식 분모와 분자를 동시에 나눠준다.
: 분자와 분모에 존재하는 항들이 모두 0~1 사이의 값. (overflow 해결)

$e^{2.0} : e^{1.0} : e^{1.2} : e^{0.7}$

최댓값에 해당하는 $e^{2.0}$ 찾아 빼준다.

$e^{0.0} : e^{-1.0} : e^{-0.8} : e^{-1.3}$
= 1.00 : 0.37 : 0.45 : 0.27
= 0.479 : 0.176 : 0.215 : 0.130

$e^{2.0} : e^{1.0} : e^{1.2} : e^{0.7}$
= 7.39 : 2.72 : 3.32 : 2.01
= 0.479 : 0.176 : 0.215 : 0.130

변형식 도출 결과 == 일반 정의식 도출 결과

 # 0 나눗셈 문제 해결

$e^{x_k - x_k} = e^0 = 1$ → 분모에 존재하게 되기에,
분모는 언제나 1 보다 커짐으로써 0 나눗셈 문제 또한 해결

3.4 소프트맥스 함수의 편미분

확률 **벡터**를 출력

SOFTMAX()

다대다 관계의 편미분

로짓값 **벡터**를 입력

※ 실제 딥러닝 학습에 이용되지 않습니다!

편미분 대상은 손실함수를 구해주는
소프트 맥스 교차 엔트로피

#확률값 벡터

2 *#손실함수 편미분 계산*

P_0.0	P_0.1	P_0.2	P_0.3	P_0.4	P_0.5	P_0.6
P_1.0	P_1.1	P_1.2	P_1.3	P_1.4	P_1.5	P_1.6
...	...	...	...	...	...	...

→ 확률분포, 즉 각 후보항목이 정답일 확률값들로 구성된 벡터

SOFTMAX()

#로짓값 벡터 1

1 *#손실 함수 계산*

출력 크기 [N,7]

L_0.0	L_0.1	L_0.2	L_0.3	L_0.4	L_0.5	L_0.6
L_1.0	L_1.1	L_1.2	L_1.3	L_1.4	L_1.5	L_1.6
...	...	...	...	...	...	...

3.5 소프트맥스 교차 엔트로피

신경망의 로짓값 벡터 ⟷ 두 확률분포 사이의 교차 엔트로피 ⟷ 정답 확률분포 벡터

$a_1, \dots, a_n$

↓

softmax()

↓

확률 분포 Q

$$H(P, Q) = -\sum p_i \log q_i$$

$y_1, \dots, y_n$

확률 분포 P

문제발생 문제발생!

$$H(P, Q) = -\sum p_i \log q_i \approx H(P, Q) = -\sum p_i \log(q_i + \varepsilon)$$

아주 작은 값을 더해줌으로써 문제 해결
※ 미미한 존재로 별다른 영향을 주지 않는다.
※ 0에 매우 가까운 하한선 역할을 수행

확률의 성질상 모든 원숫값이 항상 0 ~ 1 값을 갖는다.

→ 교차 엔트로피 연산 중 0, 1 의 값이 들어오게 되면 -inf , 0의 값을 출력

```
print(np.log(0))
print(np.log(1))
```

```
-inf
0.0
```

```
ε = 1.0e-10
print(np.log(0 + ε))
print(np.log(1 + ε))
```

```
-23.025850929940457
1.000000082690371e-10
```

3.6 소프트맥스 교차 엔트로피 편미분

신경망의 로짓값 벡터

$x_1, \dots, x_n$

↓

softmax()

↓

확률 분포 Q

$q_1, \dots, q_n$

정답 확률분포 벡터

$p_1, \dots, p_n$

확률 분포 P

$$\frac{\partial H}{\partial x_i} = q_i - p_i$$

#확률값 벡터

2 #손실함수 편미분 계산

P_0.0	P_0.1	P_0.2	P_0.3	P_0.4	P_0.5	P_0.6
P_1.0	P_1.1	P_1.2	P_1.3	P_1.4	P_1.5	P_1.6
...	...	...	...	...	...	...

→ 확률분포, 즉 각 후보항목이 정답일 확률값들로 구성된 벡터

SOFTMAX()

#로짓값 벡터 1

출력 크기 [N,7]

1 #손실 함수 계산

L_0.0	L_0.1	L_0.2	L_0.3	L_0.4	L_0.5	L_0.6
L_1.0	L_1.1	L_1.2	L_1.3	L_1.4	L_1.5	L_1.6
...	...	...	...	...	...	...

"지금까지의 과정은 다소 복잡했지,
하지만, 이 하나를 통해 모든것을 관통하게 된다네."

교차 엔트로피 정의식

$$H = -\sum_{i=1}^{n} p_i \log q_i$$

손실함수의 편미분

$$\frac{\partial H}{\partial x_i} = -\frac{\partial}{\partial x_i}\sum_{k=1}^{n} p_k \log q_k$$

편미분의 분배법칙 성립

$$\frac{\partial H}{\partial x_i} = -\sum_{k=1}^{n} \frac{\partial}{\partial x_i} p_k \log q_k$$

정답벡터 상수 (밖으로 빼냄)

$$\frac{\partial H}{\partial x_i} = -\sum_{k=1}^{n} p_k \frac{\partial}{\partial x_i} \log q_k$$

log함수 미분 및 체인룰 적용

$$\frac{\partial H}{\partial x_i} = -\sum_{k=1}^{n} p_k \frac{1}{q_k} \frac{\partial q_k}{\partial x_i}$$

소프트맥스 함수의 편미분

소프트맥스 함수의 편미분

$i \neq j$

$$\frac{\partial y_j}{\partial x_i} = -y_i y_j$$

$i = j$

$$\frac{\partial y_j}{\partial x_i} = y_i - y_i2$$

두 경우에 대한 처리식

$$\frac{\partial H}{\partial x_i} = -\sum_{k=1}^{n} \frac{p_k}{q_k} \frac{\partial q_k}{\partial x_i} = -\sum_{k=i} \frac{p_k}{q_k} \frac{\partial q_k}{\partial x_i} - \sum_{k \neq i}^{n} \frac{p_k}{q_k} \frac{\partial q_k}{\partial x_i} = -\frac{p_i}{q_i} \frac{\partial q_i}{\partial x_i} - \sum_{k \neq i}^{n} \frac{p_k}{q_k} \frac{\partial q_k}{\partial x_i}$$

대입결과

$$\frac{\partial H}{\partial x_i} = -\frac{p_i}{q_i} \frac{\partial q_i}{\partial x_i} - \sum_{k \neq i}^{n} \frac{p_k}{q_k} \frac{\partial q_k}{\partial x_i} = -\frac{p_i(q_i - {q_i}^2)}{q_i} - \sum_{k \neq i} \frac{p_k(-q_i q_k)}{q_k}$$

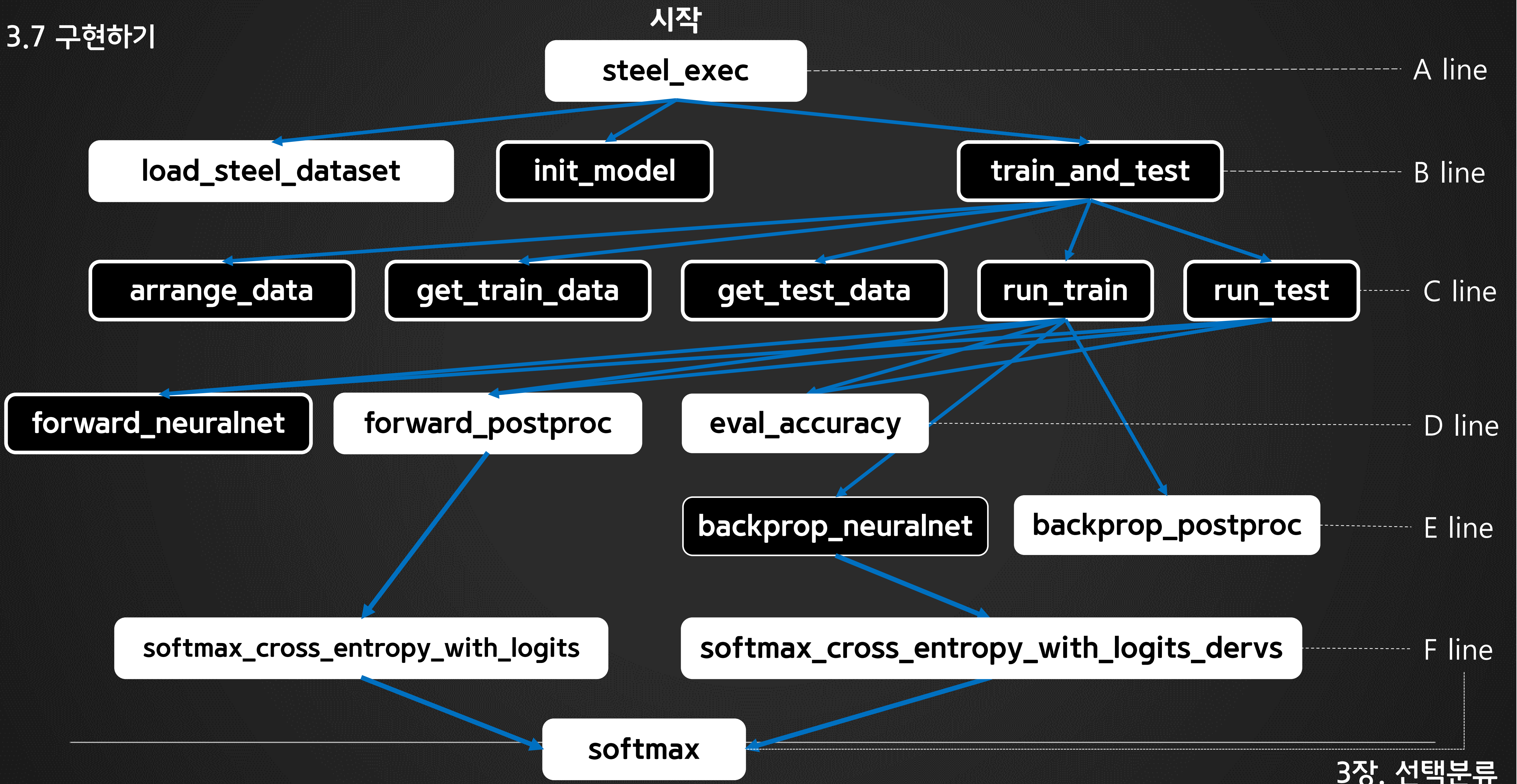
약분 과정

$$\frac{\partial H}{\partial x_i} = p_i q_i - p_i + \sum_{k \neq i} p_k q_i = \sum_{k=1}^{n} p_k q_i - p_i = q_i \sum_{k=1}^{n} p_k - p_i$$

$$\sum_{k=1}^{n} p_k = 1$$

$$\boldsymbol{\frac{\partial H}{\partial x_i} = q_i - p_i}$$

3.7 구현하기


시작
steel_exec
A line
load_steel_dataset
init_model
train_and_test
B line
arrange_data
get_train_data
get_test_data
run_train
run_test
C line
forward_neuralnet
forward_postproc
eval_accuracy
D line
backprop_neuralnet
backprop_postproc
E line
softmax_cross_entropy_with_logits
softmax_cross_entropy_with_logits_dervs
F line
softmax