Decision Making Algorithm for BaroNow Project

Final Progress Check

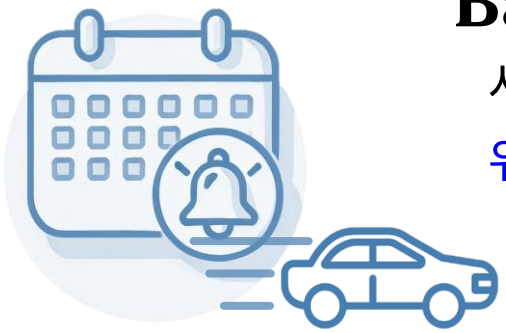
Min-hwan Oh | Seoul National University

Dae Soon Kim, Chai Won Kim, Seung Gyu Song, Sung Woo Cho

Seoul National University
Graduate School of Data Science

Contents

Introduction



BaroNow Project

사용자가 등록한 일정에 대해 최적의 출발시간과 맞춤형 정보를 제공하는 위치기반 AI 스케쥴러.

→ 기존 방식과 달리, 사용자의 실시간 Context를 활용하여, 도착 예상시간의 Uncertainty를 기반으로 시간/공간적 context 변화에 따른 교통상황 변화를 학습하여, 최적의 출발시간을 예측.

Definition of the Problem

탐색을 최소화하는 조건하에서, 정확한 출발시간을 예측하기 위한 Context기반의 의사결정 알고리즘 개발

1. 정확한 출발 시간 예측

   도착 시간으로부터 여유시간 13분 이내 만족 (95% 신뢰구간 내)

2. 경로 정보를 얻기 위한 API Call 횟수 최소화

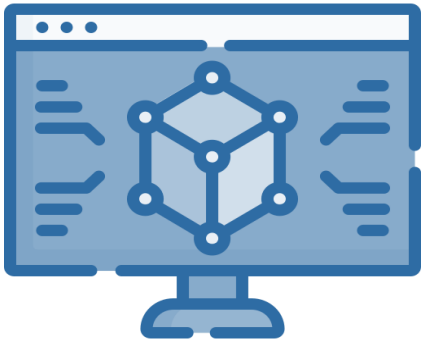
   고정위치 조건 3회 이하, 이동위치 조건 10회 이하

Dataset

AIgorithm 학습에 사용할 21개의 feature를 선정

NAME	DATA_TYPE	EXPLANATIONS	EXAMPLE_DATA
id	VARCHAR(50)	simulation ID	240815_214845_00000014
group	VARCHAR(50)	class of context	history
transportation	VARCHAR(50)	type of transportation	personal car
round	INT(16)	current round	0
start_time	TIME_STAMP	time when scheduel registered	Mon. 07:34
target_time	TIME_STAMP	time when user need to arrive a destination	Tue. 05:34
start_point	TUPLE	coordinate where schedule register	[0.07035964 0.23935531]
target_point	TUPLE	destination coordinates	[0.60523111 0.36777999]
call_time_TimeMachine	TIME_STAMP	time when TimeMachine called	Tue. 04:03
path_TimeMachine	LIST	predicted path from TimeMachine	[1, 2, 3, 6, 10, 19, 21, 23]
sub_path_time_TimeMachine	DICTIONARY	each taking time for sub path from TimeMac hine	{1: 15.2, 2:3.8, ... , 23: 8.2}
path_time_TimeMachine	FLOAT(32)	predicted taking time from current point to destination by TimeMachine	53.19339518
cur_time	TIME_STAMP	current time	Mon. 07:34
cur_point	TUPLE	current located coordinate	(0.07035964045822773, 0.23935530562232232)
call_time_LastAPI	TIME_STAMP	time when Last API called	Mon. 07:34
call_point_LastAPI	TUPLE	coordinate where Last API called	(0.07035964045822773, 0.23935530562232232)
path_LastAPI	LIST	predicted path from Last API	[1, 2, 3, 6, 10, 15, 19, 21, 23]
path_time_LastAPI	FLOAT(32)	predicted taking time from current point to destination by LastAPI	96.33106516
sub_path_time_LastAPI	DICTIONARY	each taking time for sub path from LastAPI	{1: 21.5, 2:7.8, ... , 23: 19.8}
weather	VARCHAR(20)	current weather information	
event	VARCHAR(200)	column for reference	register_schedule

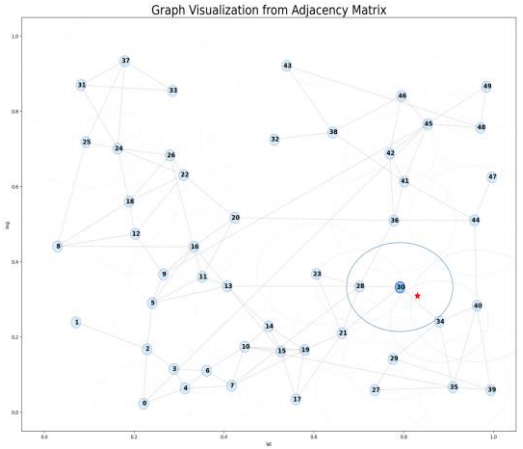
Simulation Environment

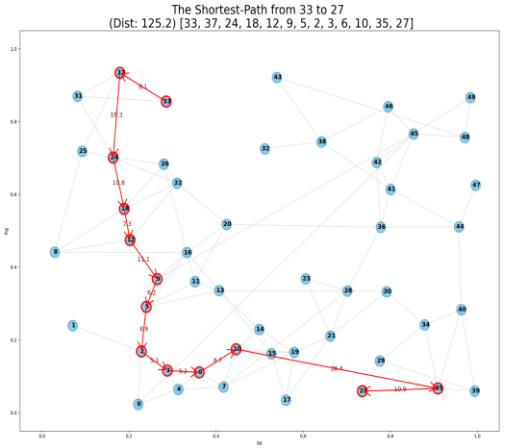


Graph Based Simulation

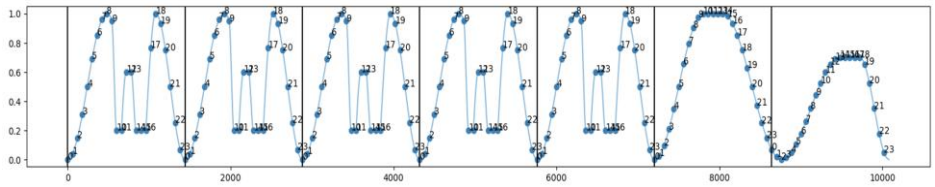
Graph 구조와 Dijkstra 알고리즘을 활용하여, 실제 경로안내 시스템을 모사하는 Simulation 환경을 개발

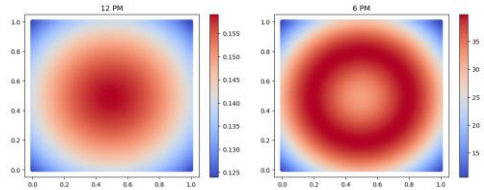
➔ **최종 모델 선정을 위한 Simulator로 활용**





BaroNow_Simulator_v4.5.zip





PersonalCar Algorithm

Introduction & Motivation

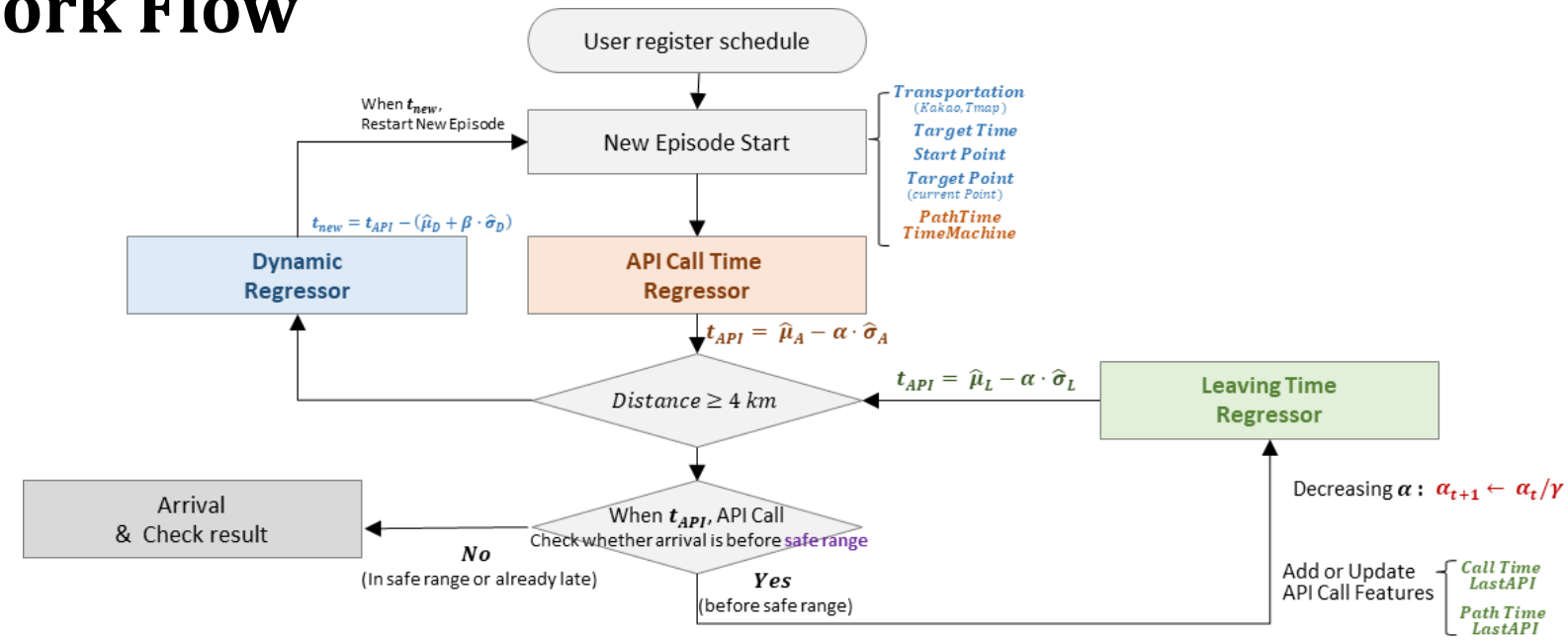
- 최적의 소요시간을 얻을 수 있는 API Call은 출발 바로 직전에 Call한 정보로부터 얻을 수 있으며, 교통정보에 대한 Noise는 Gaussian Distribution을 따른다고 가정 $\eta \sim N(0, \sigma^2)$
- 다음 정보를 얻어야 할 시점(API Call시점)에 대한 평균과, 편차를 동시에 예측함으로서, 매우 높은 확률로 최적의 출발시간보다 늦지 않도록 성능 보장 가능
- 사용자의 실시간 이동상황에서 최악의 경로이동 (Worst-Case)를 가정하고, 평균과 편차를 동시에 예측함으로서 매우 높은 확률로 정지상태(Static) 알고리즘이 유효하도록 성능 보장

Technical Key-point

- Feature Representation : 시·공간 feature의 표현력 부여
- Uncertainty Modeling : Target의 평균과 편차를 동시에 예측을 통한 확률적 성능 보장
- Recurrent Structure : 정보를 얻을 시점을 예측함으로서, 최신정보를 활용한 예측력 향상
- Worst-Case Assumption : 동적 상황에서 최악의 경로가정에 따른 성능 보장

PersonalCar Algorithm

Work Flow



(Static Case)

- 다음 정보를 얻을 시점의 평균과 편차를 예측하여 Client(BaroNow)에 전달
- 경로정보를 받으면, 해당 시간에 출발했을 때 목표시간범위(-13~0분)내 도착여부 확인

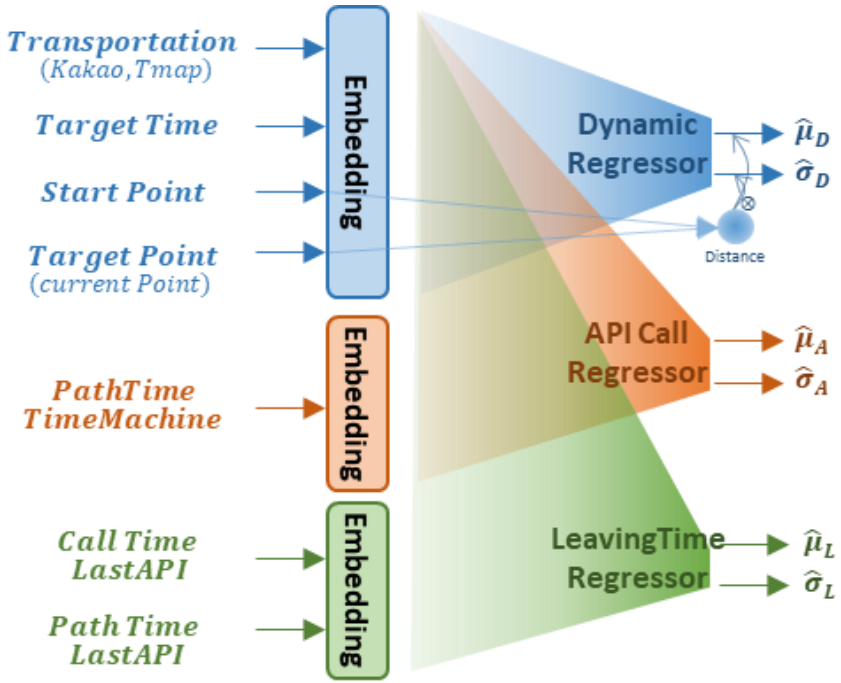
  → 도착가능 : 바로 출발, 도착불가 : 다음 정보확인 시점 예측 (discount factor γ를 통해 sigma 계수축소)

(Dynamic Case)

- 4km이상 벗어나면, 정지상태(Static Case)의 정보를 얻어야할 시점(API Call) 혹은 출발시간(Leaving-Time)을 기준으로 사용자가 이동한 거리에 대한 예상 시간만큼 차감한 시점에 Schedule 재실행

PersonalCar Algorithm

Architecture

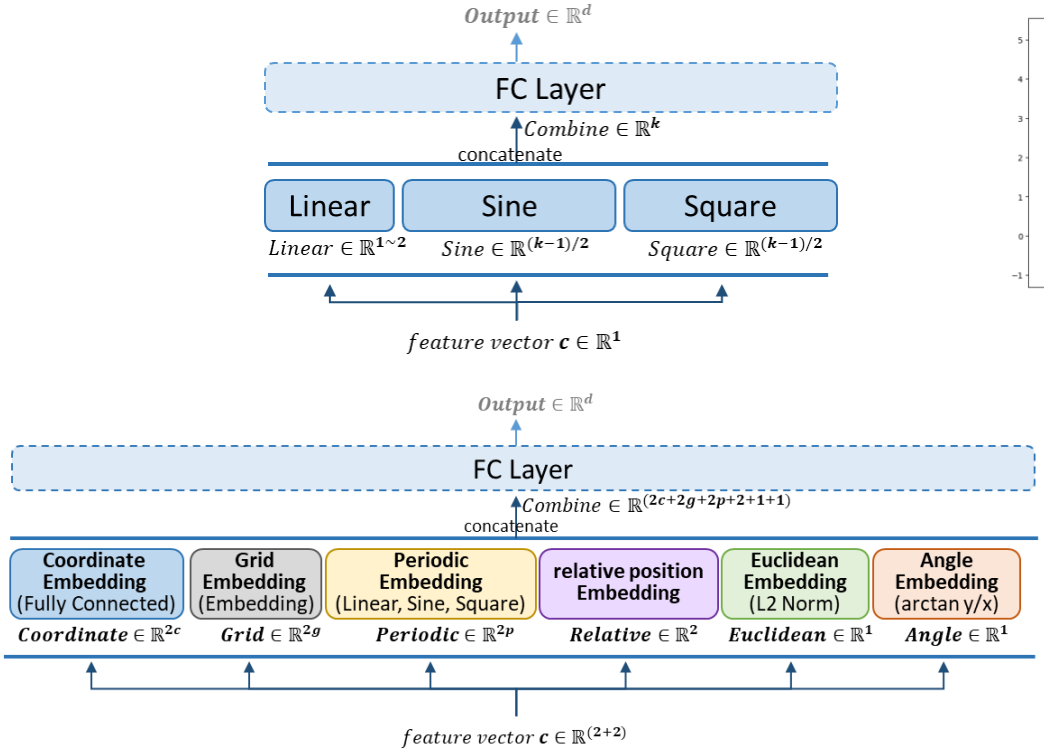


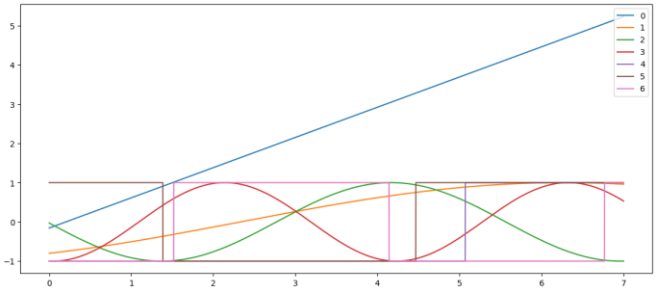
Feature	Type	Embedding
Transportation	Categorical	Categorical Embedding
Traget_time	timestamp	Temporal Embedding
Start_Point	tuple (float, float)	Spatial Embedding
Target_Point	tuple (float, float)	
PathTime_ TimeMachine	float	FullyConnected Embedding
CallTime_LastAPI	timestamp	Temporal Embedding
PathTime_LastAPI	float	FullyConnected Embedding

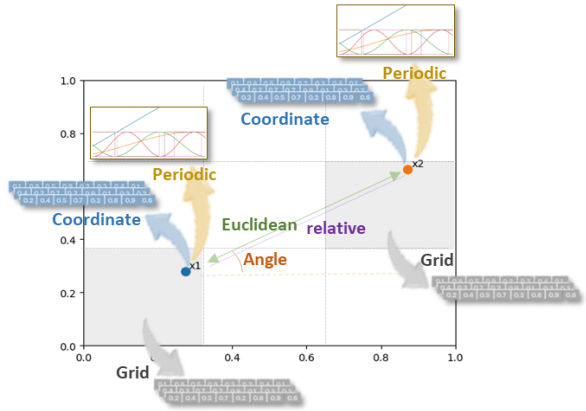
- Static/Dynamic Case 모델 통합
- Transportation feature를 토큰화(tokenize)하여 Kakao / Tmap Model 통합
  . 지도 App끼리 Weight를 공유함으로서 Data가 적은 App에 대한 예측 강건성(robust) 향상
- 모델마다 같은 형식으로 평균 및 표준편차를 예측함으로서, Recurrent 구조 도입 가능

PersonalCar Algorithm

Feature Representation



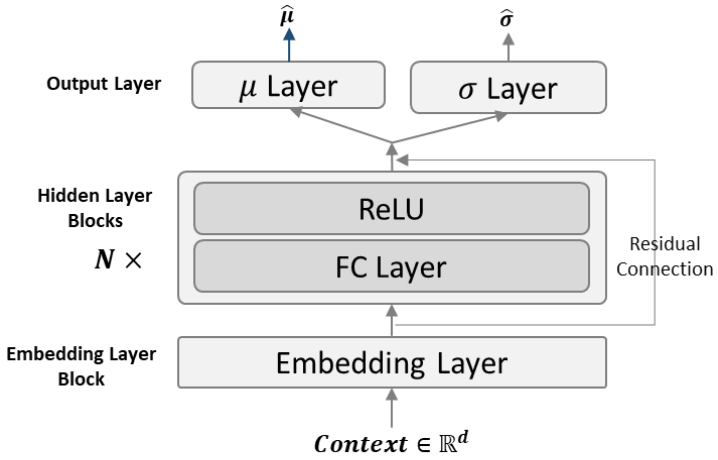




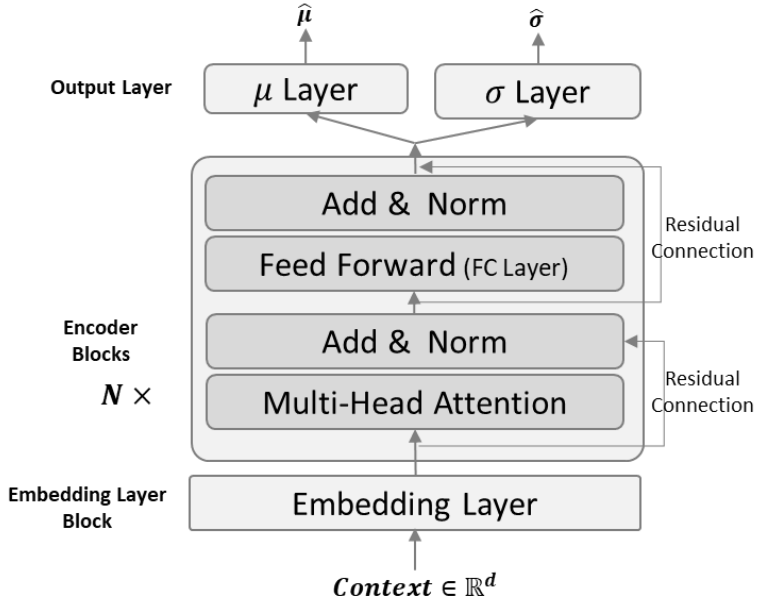
- BaroNow 알고리즘에서 사용하는 대부분의 시간, 공간 Feature에 대해, 다양한 표현력을 부여하기 위해 시·공간 맞춤형 Embedding Layer 구현 및 적용

PersonalCar Algorithm

Classifier Architecture



[ResNet Based Architecture]



[TransformerEncoder Based Architecture]

(ResNet based) # of prarameter : 252,427

- 기본적인 FC Layer + Residual Connection 구조
- 적은 Parameter 갯수로 학습에 필요한 data가 상대적으로 적게 필요하고 빠른 수렴성을 가지면서도, 높은 예측 성능 확보
- Residual Connection을 통한 계층별 Feature 사용가능

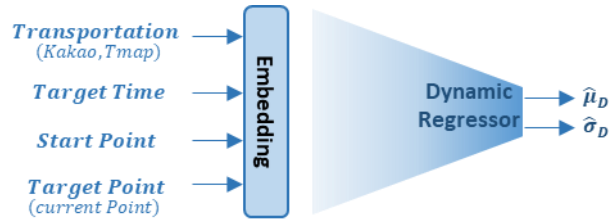
(Transformer based) # of prarameter : 1,121,467

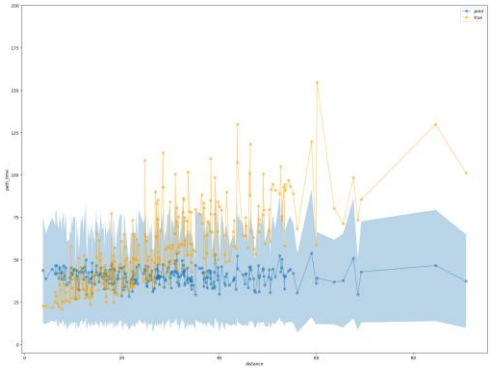
- Transformer Encoder 기반 Architure
  . Multi-Head Attention + LayerNorm + Residual Connection
- 많은 Parameter를 이용하여, 좋은 Performance를 내기 위해서는 대량의 data와 많은 학습 횟수가 필요

PersonalCar Algorithm

Distance Bias Trick

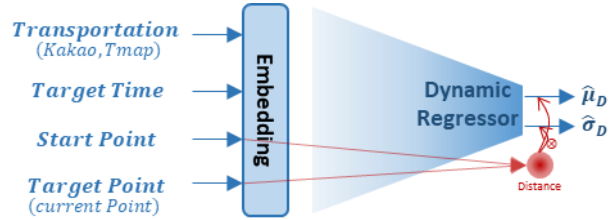
(초기 버전 구조)

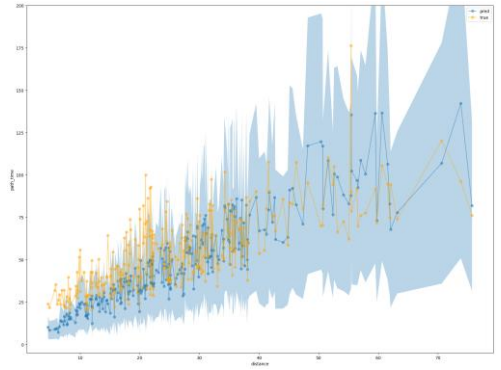




- Input Feature의 시·공간 Embedding에만 전적으로 의존
- 공간 Embedding시 Distance에 해당하는 feature가 있으나, 위치좌표 등의 다른 feature들의 역할에 비해 distance feature 비중이 적어 거리에 따른 소요시간 증가 경향을 잘 학습하지 못함

(Final 버전 구조)



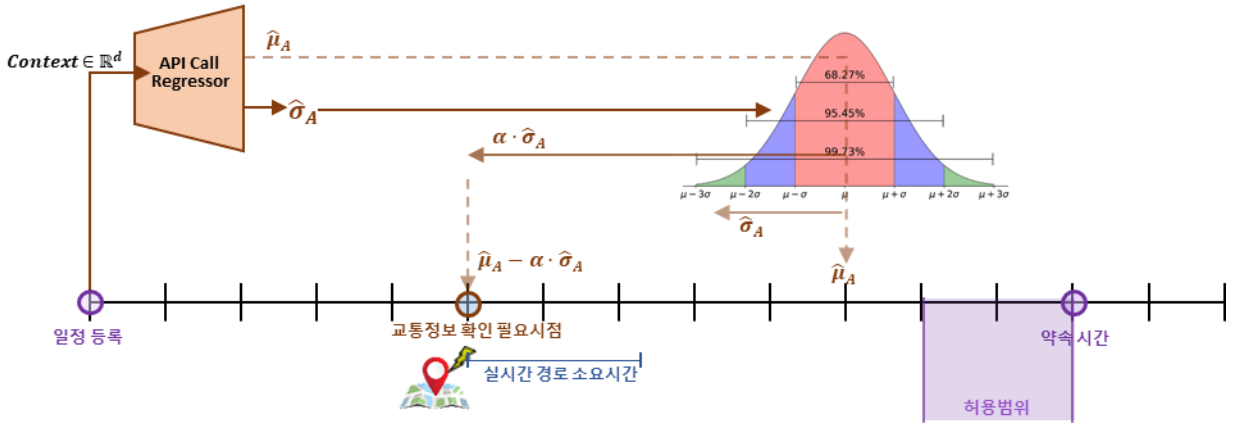


- Distance feature를 output 마지막에 직접적으로 곱해줌으로서(Inductive Bias 부여), 거리증가에 따른 소요시간 및 편차 증가 경향을 자연스럽게 부여하여 예측력 향상

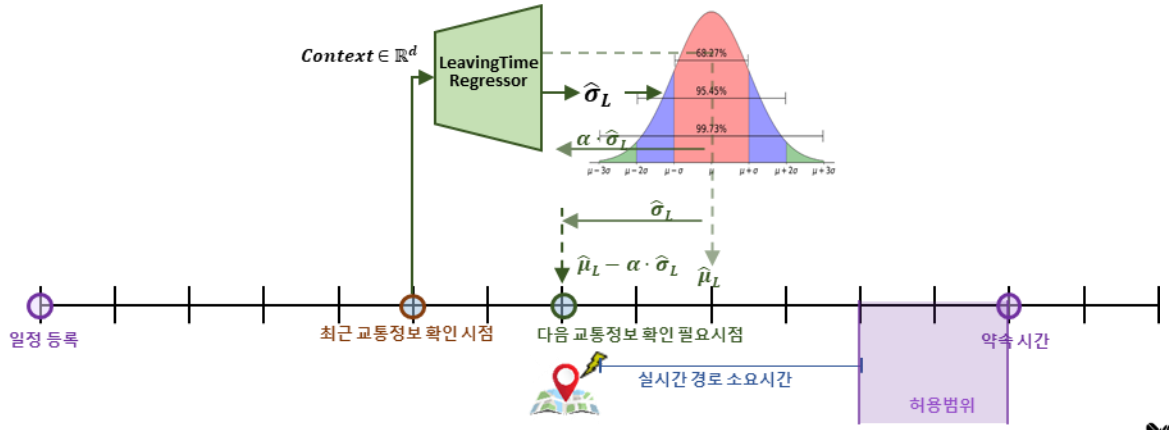
PersonalCar Algorithm

Mechanism of the algorithm (Static Case)

① **API Call Regressor :** 초기정보를 기반으로 Optimal API Call의 μ, σ 추정



②**Leaving Time Regressor** : API Call 정보를 추가로 이용하여 Optimal API Call의 μ, σ 추정

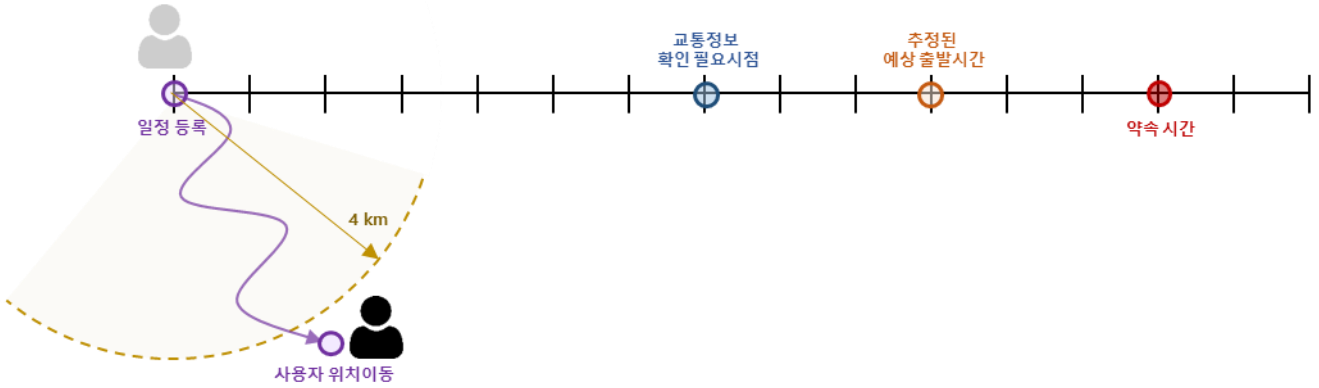


PersonalCar Algorithm

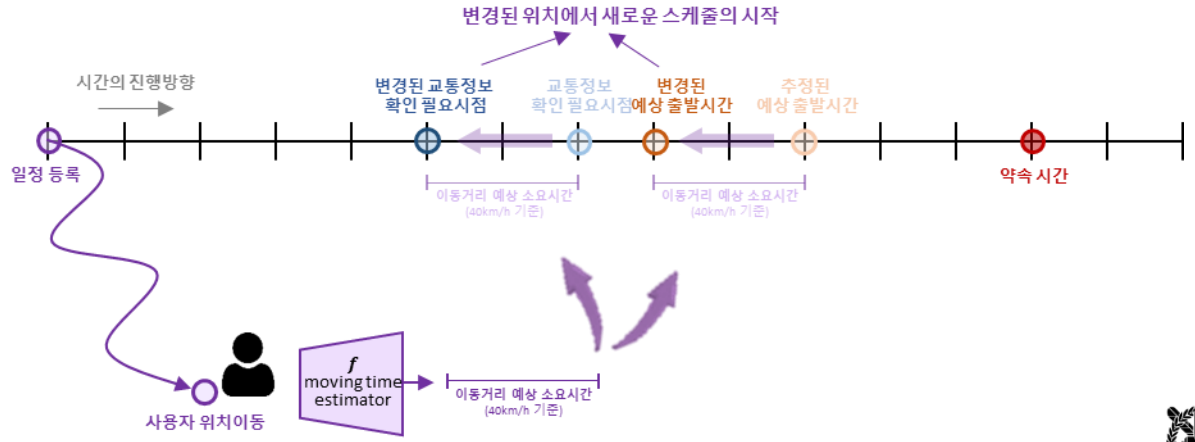
Mechanism of the algorithm (Dynamic Case)

③ 사용자의 실시간 이동에 따른 예측 성능감소를 고려하여 처음부터 다시 추정이 필요할 시점을 예측

[그림 3] ③-1 경험적 요소 : 반경 4km 이동시



[그림 4] ③-2 확률기반 최악의 상황을 고려한 교통상황 확인 or 출발시간 보정



PersonalCar Algorithm

Training Log

model_name	MidTermModel	resnet_V3	resnet_V3	transformer_V3	transformer_V3
type	-	Train_set	Test_set	Train_set	Test_set
dynamic_gaussian_loss	-	3.74	3.72	3.79	3.80
dynamic_rmse_loss	-	18.78	18.63	19.57	20.08
api_gaussian_loss	2.22~2.36	2.38	2.38	2.40	2.40
api_rmse_loss	-	7.51	7.51	7.31	7.31
leaving_gaussian_loss	-	1.69	1.63	1.67	1.70
leaving_rmse_loss	3.95~4.32	4.47	4.41	4.57	4.49

Offline Test (Static Case)

test_type	model	n_episodes	alpha	gamma	max api_call	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
Rule		**9,167**	**3**			**94.06%**	**2.48%**	**3.47%**	**5.05**	**-8.29**	**3.23**	**2.94**	**-16.01**
	24hr	10,000	3			95.30%	0.90%	3.80%	5.03	-8.58	2.63	2.47	-15.71
	commute (7-10, 14-19)	9,167	3			92.70%	4.20%	3.10%	5.08	-7.96	3.87	3.46	-16.33
Alg3		**9,195**	**3**		**1**	**87.43%**	**7.57%**	**5.05%**	**1.99**	**-6.16**	**4.34**	**3.88**	**-15.65**
(Beta Test)	**24hr**	10,000	3		1	90.30%	2.40%	7.30%	1.98	-6.97	3.15	2.32	-15.65
	commute (7-10, 14-19)	9,195	3		1	84.30%	**13.20%**	2.60%	2.00	-5.28	5.64	5.58	-15.65
resnet		**9,179**	**4**	**2**		**94.51%**	**3.03%**	**2.51%**	**3.65**	**-7.71**	**3.32**	**3.43**	**-15.74**
		9,180	**4**	**2.5**		**94.32%**	**3.21%**	**2.47%**	**3.47**	**-7.40**	**3.39**	**3.56**	**-15.78**
	24hr	10,000	4	2		95.80%	1.50%	2.80%	3.49	-8.14	2.79	2.88	-15.69
		10,000	4	2.5		95.90%	1.10%	3.00%	3.33	-7.95	2.71	2.69	-15.21
	commute (7-10, 14-19)	9,179	4	2		93.10%	4.70%	2.20%	3.81	-7.25	3.91	4.02	-15.79
		9,180	4	2.5		92.60%	5.50%	1.90%	3.61	-6.79	4.12	4.51	-16.40
transformer		**9,166**	**4**	**2**		**93.73%**	**3.44%**	**2.87%**	**3.75**	**-7.65**	**3.51**	**3.89**	**-15.70**
		9,182	**4**	**2.5**		**93.35%**	**4.18%**	**2.48%**	**3.55**	**-7.28**	**3.56**	**3.22**	**-15.37**
	24hr	10,000	4	2		95.50%	1.10%	3.40%	3.84	-8.26	2.70	2.94	-15.51
		10,000	4	2.5		95.50%	1.40%	3.10%	3.62	-7.94	2.79	2.32	-15.35
	commute (7-10, 14-19)	9,166	4	2		91.80%	6.00%	2.30%	3.65	-6.99	4.39	4.92	-15.90
		9,182	4	2.5		91.00%	7.20%	1.80%	3.46	-6.57	4.40	4.20	-15.39

➔ 기존 RuleBased 모델 **동등수준 성능** 확보 및 **API Call수 평균 약 1.5회 절약 가능**

PersonalCar Algorithm

Offline Test (Static Case)

Max API Call 1회 제한시 성능 비교

test_type	model	n_episodes	alpha	gamma	max api_call	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
24 hr	**Rule**	300	3		1	**21.00%**	**0.00%**	79.00%	1.99	-21.13	9.94		-24.32
	Alg3 (Beta test)	300	3		1	91.70%	**1.70%**	6.70%	1.98	-7.05	2.88	1.09	-16.58
	resnet	300	3		1	92.00%	**1.70%**	6.30%	2.00	-7.65	3.21	1.78	-16.16
	transformer	300	3		1	83.00%	4.30%	12.70%	1.99	-6.94	4.16	2.62	-15.31
commute	**Rule**	282	3		1	**27.70%**	2.60%	69.70%	2.00	-20.17	12.39	4.86	-25.95
	Alg3 (Beta test)	279	3		1	86.40%	12.20%	1.40%	2.00	-5.29	5.88	6.28	-14.64
	resnet	267	3		1	79.80%	9.40%	10.90%	2.00	-6.97	5.90	4.81	-16.36
	transformer	284	3		1	75.40%	15.50%	9.20%	2.00	-5.17	7.03	5.56	-19.13

➔ 약속시간 초과Case **BetaTest모델과 동등수준, 통근시간 외 API Call 1회 제한시 초과 Case 2%이내 확보**

Minimum Call interval 성능 비교

test_type	model	n_episodes	alpha	gamma	max api_call	min api_interval	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
24 hr	resnet	300	4	2		3	97.70%	0.70%	1.70%	3.51	-8.27	2.87	2.43	-25.04
	resnet	300	4	2		**6**	93.70%	3.30%	3.00%	3.43	-6.54	3.46	3.18	-14.34
	resnet	300	4	2.5		3	96.00%	1.00%	3.00%	3.28	-8.03	2.51	1.77	-15.00
	resnet	300	4	2.5		**6**	93.00%	1.30%	5.70%	3.31	-7.07	3.03	0.72	-15.90
	transformer	300	4	2		3	94.70%	0.70%	4.70%	3.83	-8.43	2.77	0.33	-18.24
	transformer	300	4	2		**6**	94.70%	3.00%	2.30%	3.52	-6.76	3.31	3.04	-13.59
	transformer	300	4	2.5		3	95.70%	2.70%	1.70%	3.67	-7.82	2.81	1.10	-14.13
	transformer	300	4	2.5		**6**	95.70%	1.30%	3.00%	3.46	-6.73	3.04	2.09	-16.47
commute	resnet	275	4	2		3	94.90%	3.60%	1.50%	3.78	-7.21	3.48	2.21	-16.68
	resnet	281	4	2		**6**	90.70%	7.50%	1.80%	3.84	-5.94	4.73	5.04	-14.69
	resnet	274	4	2.5		3	93.40%	4.70%	1.80%	3.61	-7.00	4.04	4.82	-17.65
	resnet	276	4	2.5		**6**	91.70%	6.90%	1.40%	3.66	-5.87	4.67	5.95	-15.73
	transformer	272	4	2		3	93.00%	5.10%	1.80%	3.71	-7.24	3.74	2.98	-14.00
	transformer	283	4	2		**6**	89.40%	8.80%	1.80%	3.31	-6.10	4.50	3.32	-15.42
	transformer	274	4	2.5		3	91.20%	6.20%	2.60%	3.43	-6.84	4.26	4.48	-15.99
	transformer	274	4	2.5		**6**	90.10%	9.10%	0.70%	3.25	-5.89	4.26	3.05	-19.81

➔ **Minimum Call interval 3분 → 6분 증가시 성능수준 약 3% 감소, API Call수 이득효과 미미**

Performance Measure

Online Test : Personal Car

Online environment

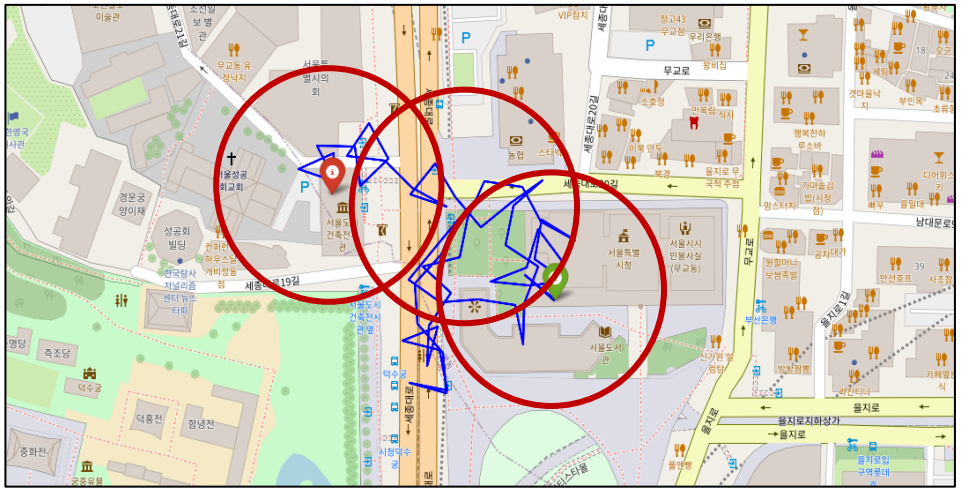
Moving agent의 속도:

1.8km/h ~ 100km/h

Moving agent의 방향:

랜덤으로 선택: $0 \sim 2\pi$

→100m 반경이 넘어갈 경우 새로운 Context를 저장한다.



Online test for Dynamic case ('12.19~23; 3.5 Days)

model	API	test_type	N episodes	alpha	gamma	max api_call	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
Resnet	**Kakao**	**24hr : commute = 1:1**	**289**	**4**	**2.5**		94.81%	**1.73%**	**3.46%**	**3.86**	**-10.04**	**3.11**	**3.88**	**-14.92**
		Commute (6-10, 14-18)	131	4	2.5		92.37%	2.29%	5.34%	4.15	-9.53	3.87	5.80	-15.21
	Tmap	**24hr : commute = 1:1**	**289**	**4**	**2.5**		**95.16%**	**1.73%**	**3.11%**	**3.98**	**-9.94**	**2.88**	**1.25**	**-15.26**
		Commute (6-10, 14-18)	131	4	2.5		93.13%	3.82%	3.05%	4.34	-9.37	3.43	1.25	-15.77
Transformer	**Kakao**	**24hr : commute = 1:1**	**235**	**4**	**2.5**		94.47%	**0.85%**	**4.68%**	**4.90**	**-10.45**	**2.81**	**5.99**	**-14.95**
		Commute (6-10, 14-18)	108	4	2.5		92.59%	1.85%	5.56%	5.31	-10.12	3.63	5.99	-15.76
	Tmap	**24hr : commute = 1:1**	**235**	**4**	**2.5**		**95.32%**	**0.43%**	**4.26%**	**5.02**	**-10.24**	**2.83**	**12.58**	**-14.35**
		Commute (6-10, 14-18)	108	4	2.5		92.59%	0.93%	6.48%	5.76	-9.97	3.50	12.58	-14.46

Accept 비율 95.0%, Excess 비율 1.7%, API Call수 **평균 3.9회** (Resnet; 24hr:commute = 1:1 기준)

Accept 비율 94.9%, Excess 비율 0.6%, API Call수 **평균 4.9회** (Transformer ; 24hr:commute = 1:1 기준)

➔ **목표 수준 달성**

Public-Transportation Algorithm

Introduction & Motivation

- 최적의 소요시간을 얻을 수 있는 API Call은 출발 바로 직전에 Call한 정보로부터 얻을 수 있으며, 교통정보에 대한 Noise는 Gaussian Distribution을 따른다고 가정 $\eta \sim N(0, \sigma^2)$

- 다음 정보를 얻어야 할 시점(API Call시점)에 대한 평균과, 편차를 동시에 예측함으로서, 매우 높은 확률로 최적의 출발시간보다 늦지 않도록 성능 보장 가능

- 사용자의 실시간 이동상황에서 최악의 경로이동 (Worst-Case)를 가정하고, 평균과 편차를 동시에 예측함으로서 매우 높은 확률로 정지상태(Static) 알고리즘이 유효하도록 성능 보장

➔ **PersonalCar와 다른점** : API치는 시간에 따라 결과 변하지 않음 (대중교통 API의 특성)
따라서 **유저가 움직이는 dynamic case만 고려하는 문제**

Technical Key-point

- **Feature Representation** : 시공간 feature의 표현력 부여
- **Uncertainty Modeling** : Target의 평균과 편차를 동시에 예측을 통한 확률적 성능 보장
- **Recurrent Structure** : 정보를 얻을 시점을 예측함으로서, 최신정보를 활용한 예측력 향상
- **Worst-Case Assumption** : 동적 상황에서 최악의 경로가정에 따른 성능 보장

Public-Transportation Algorithm

Dataset

NAME	DATA_TYPE	EXPLANATIONS	EXAMPLE_DATA
id	VARCHAR(50)	simulation ID	240815_214845_00000014
~~**group**~~	~~VARCHAR(50)~~	~~class of context~~	~~history~~
transportation	VARCHAR(50)	type of transportation	personal car
round	INT(16)	current round	0
~~**start_time**~~	~~TIME_STAMP~~	~~time when schedule registered~~	~~Mon. 07:34~~
~~**target_time**~~	~~TIME_STAMP~~	~~time when user need to arrive a destination~~	~~Tue. 05:34~~
start_point	TUPLE	coordinate where schedule register	[0.07035964 0.23935531]
target_point	TUPLE	destination coordinates	[0.60523111 0.36777999]
~~**call_time_TimeMachine**~~	~~TIME_STAMP~~	~~time when TimeMachine called~~	~~Tue. 04:03~~
~~**path_TimeMachine**~~	~~LIST~~	~~predicted path from TimeMachine~~	~~[1, 2, 3, 6, 10, 19, 21, 23]~~
~~**sub_path_time_TimeMachine**~~	~~DICTIONARY~~	~~each taking time for sub-path from TimeMachine~~	~~{1: 15.2, 2:3.8, ... , 23: 8.2}~~
path_time_TimeMachine (path_time_start)	FLOAT(32)	~~predicted~~ taking time from start point to destination by ~~Timemachine~~ API	53.19339518
cur_time	TIME_STAMP	current time	Mon. 07:34
cur_point	TUPLE	current located coordinate	(0.07035964045822773, 0.23935530562232232)
call_time_LastAPI	TIME_STAMP	time when Last API called	Mon. 07:34
~~**call_point_LastAPI**~~	~~TUPLE~~	~~coordinate where Last API called~~	~~(0.07035964045822773, 0.23935530562232232)~~
~~**path_LastAPI**~~	~~LIST~~	~~predicted path from Last API~~	~~[1, 2, 3, 6, 10, 15, 19, 21, 23]~~
path_time_LastAPI	FLOAT(32)	predicted taking time from current point to destination by LastAPI	96.33106516
~~**sub_path_time_LastAPI**~~	~~DICTIONARY~~	~~each taking time for sub-path from LastAPI~~	~~{1: 21.5, 2:7.8, ... , 23: 19.8}~~
~~**weather**~~	~~VARCHAR(20)~~	~~current weather information~~	
~~**event**~~	~~VARCHAR(200)~~	~~column for reference~~	~~register_schedule~~

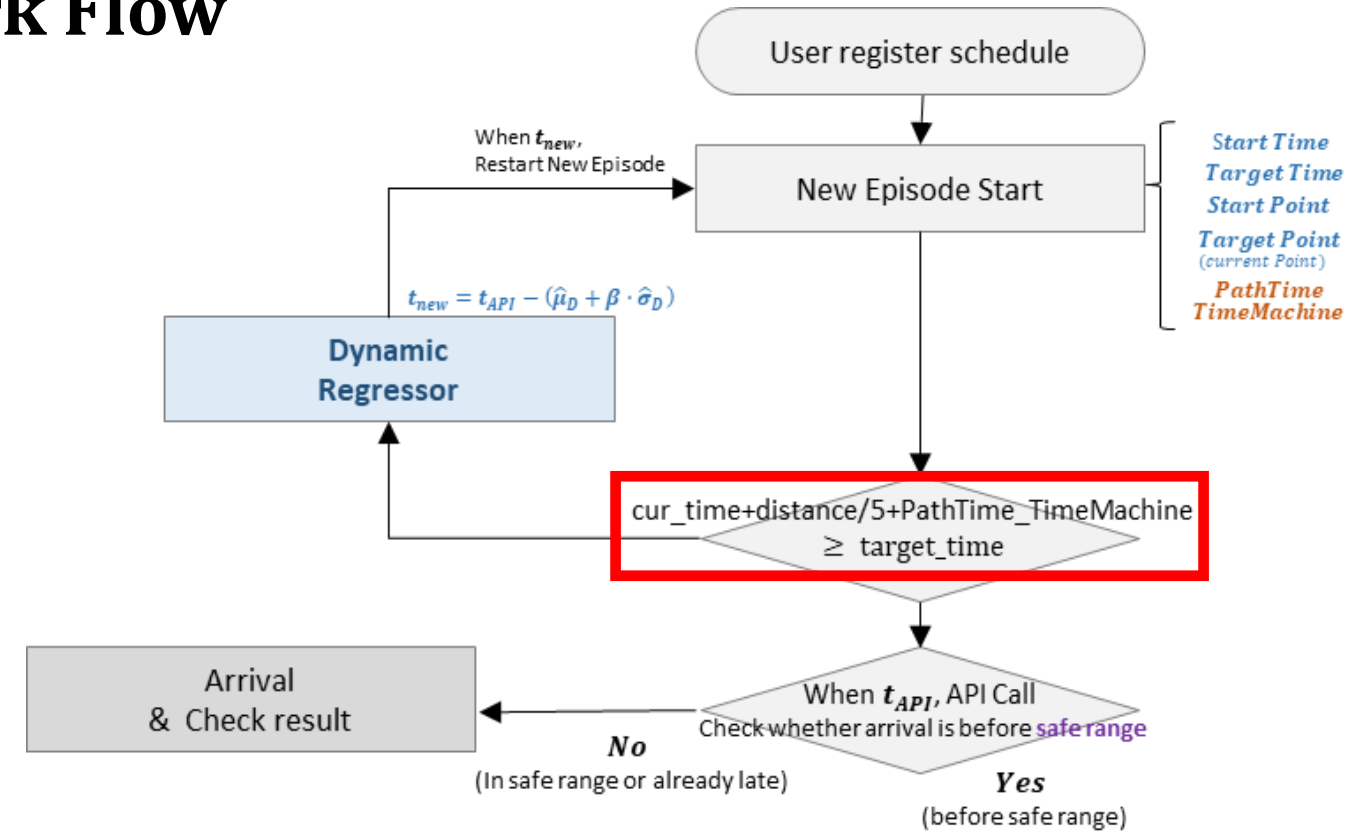
Public-Transportation Algorithm

Dataset

NAME	DATA_TYPE	EXPLANATIONS	EXAMPLE_DATA
id	VARCHAR(50)	simulation ID	240815_214845_00000014
group	VARCHAR(50)	class of context	history
transportation	VARCHAR(50)	type of transportation	personal car
round	INT(16)	current round	0
start_time	TIME_STAMP	time when scheduel registered	Mon. 07:34
target_time	TIME_STAMP	time when user need to arrive a destination	Tue. 05:34
start_point	TUPLE	coordinate where schedule register	[0.07035964 0.23935531]
target_point	TUPLE	destination coordinates	[0.60523111 0.36777999]
call_time_TimeMachine	TIME_STAMP	time when TimeMachine called	Tue. 04:03
path_TimeMachine	LIST	predicted path from TimeMachine	[1, 2, 3, 6, 10, 19, 21, 23]
sub_path_time_TimeMachine	DICTIONARY	each taking time for sub path from TimeMachine	{1: 15.2, 2:3.8, ... , 23: 8.2}
path_time_TimeMachine (path_time_start)	FLOAT(32)	~~predicted~~ taking time from start point to destination by ~~Timemachine~~ API	53.19339518
cur_time	TIME_STAMP	current time	Mon. 07:34
cur_point	TUPLE	current located coordinate	(0.07035964045822773, 0.23935530562232232)
call_time_LastAPI	TIME_STAMP	time when Last API called	Mon. 07:34
call_point_LastAPI	TUPLE	coordinate where Last API called	(0.07035964045822773, 0.23935530562232232)
path_LastAPI	LIST	predicted path from Last API	[1, 2, 3, 6, 10, 15, 19, 21, 23]
path_time_LastAPI	FLOAT(32)	predicted taking time from current point to destination by LastAPI	96.33106516
sub_path_time_LastAPI	DICTIONARY	each taking time for sub path from LastAPI	{1: 21.5, 2:7.8, ... , 23: 19.8}
weather	VARCHAR(20)	current weather information	
event	VARCHAR(200)	column for reference	register_schedule

Public-Transportation Algorithm

Work Flow

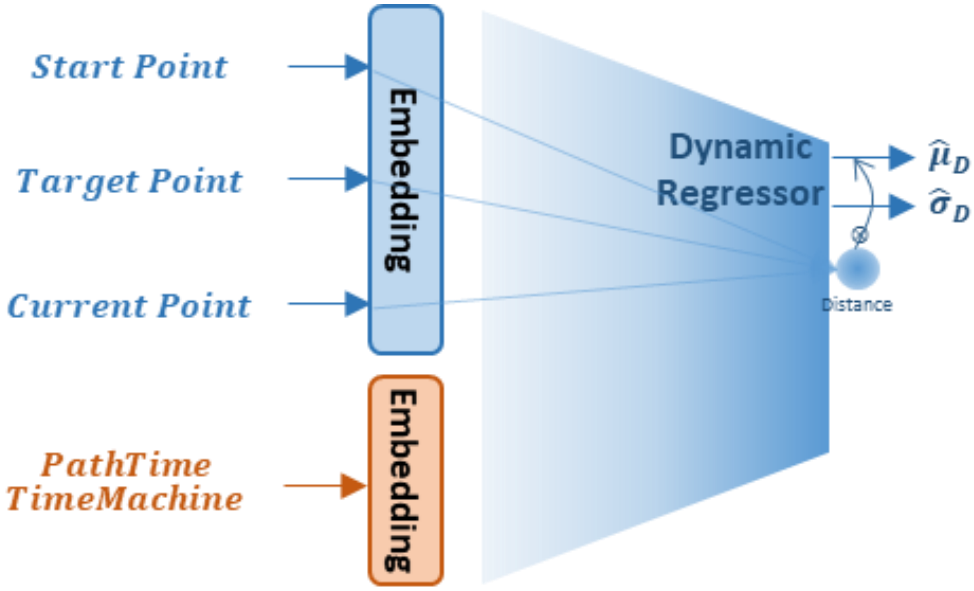


(Dynmaic Case)

- Dynamic Regressor 기준으로 새로운 정보를 얻어야 할 시점(API Call) AND 출발시간(Leaving Time)을 기준으로 사용자가 이동한 거리에 대한 예상 시간만큼 차감한 시점임을 만족할 때 Schedule 재실행

Public-Transportation Algorithm

Architecture

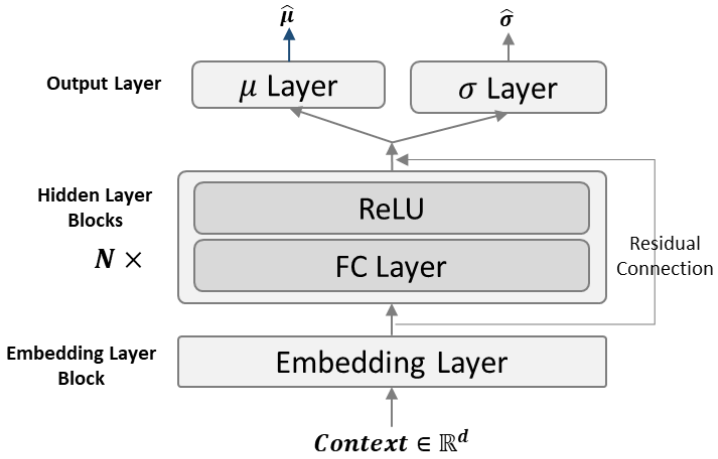


Feature	Type	Embedding
~~Transportation~~	~~Categorical~~	~~Categorical Embedding~~
~~Traget_time~~	~~timestamp~~	~~Temporal Embedding~~
Start_Point	tuple (float, float)	Spatial Embedding
Target_Point	tuple (float, float)	
Current_Point	tuple (float, float)	
PathTime_ TimeMachine	float	FullyConnected Embedding
~~CallTime_LastAPI~~	~~timestamp~~	~~Temporal Embedding~~
~~PathTime_LastAPI~~	~~float~~	~~FullyConnected Embedding~~

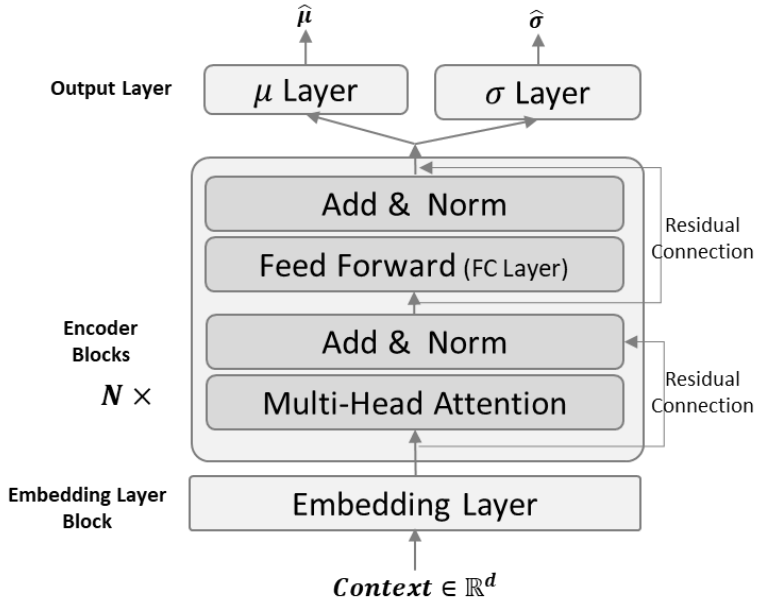
- ~~Static/~~Dynamic Case ~~모델 통합~~
- ~~Transportation feature를 토큰화(tokenize)하여 Kakao / Tmap Model 통합~~
  - ~~. 지도 App끼리 Weight를 공유함으로서 Data가 적은 App에 대한 예측 강건성(roubst) 향상~~
- ~~모델마다 같은 형식으로 평균 및 표준편차를 예측함으로서, Recurrent 구조 도입 가능~~

Public-Transportation Algorithm

Classifier Architecture



[ResNet Based Architecture]



[TransformerEncoder Based Architecture]

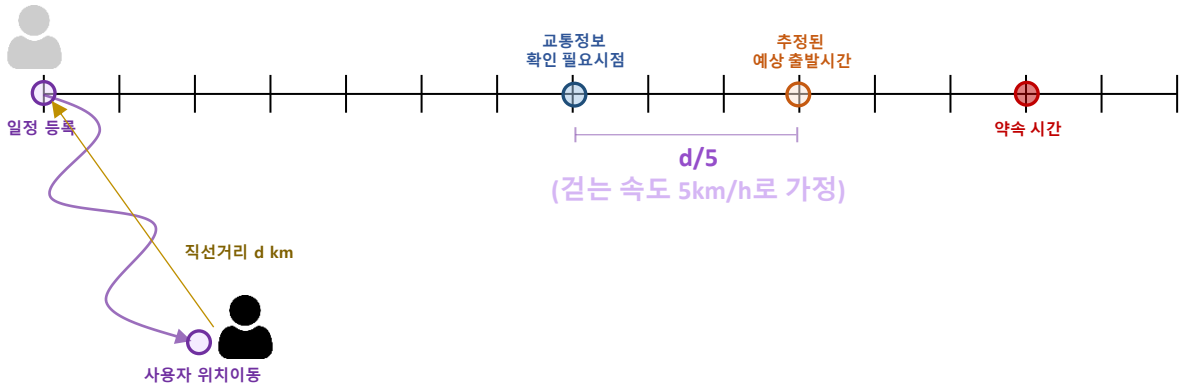
(ResNet based)

(Transformer based)

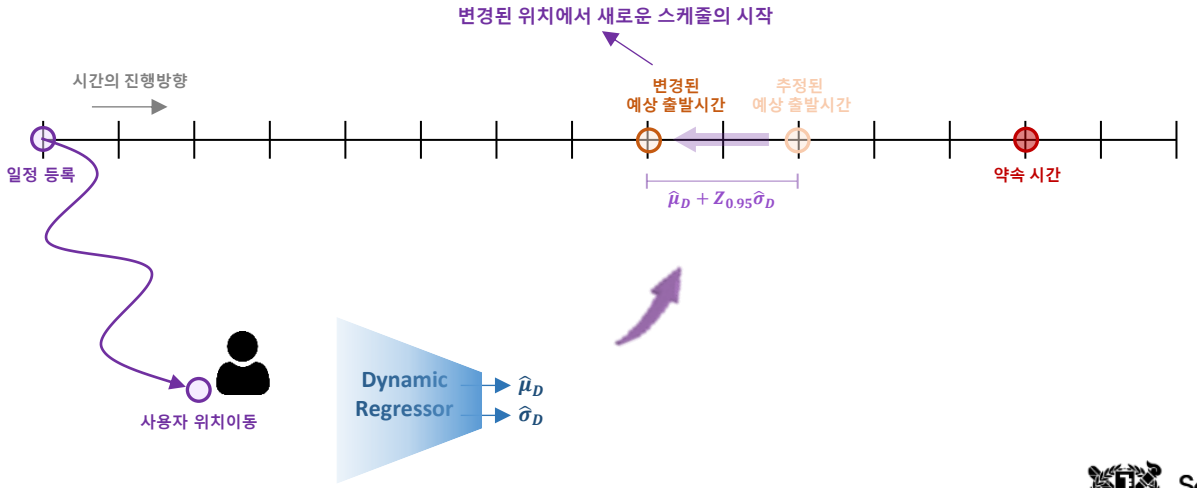
Public-Transportation Algorithm

Mechanism of the algorithm (Dynamic Case)

[그림 3] ③-1 worst case : start_point로 걸어서 돌아옴



[그림 4] ③-2 Dynamic Regressor를 이용한 출발시간 보정



Public-Transportation Algorithm

Training Result

model_name	mean_dist_mode	coverage	RMSE	mae	diff_mean
NN	0	96.37%	22.36	18.89	18.35
	1	96.08%	21.43	19.14	18.41
	2	96.16%	21.71	19.25	18.52
ResNet	0	95.94%	24.24	19.37	18.78
	1	96.07%	21.40	19.11	18.38
	2	96.07%	21.10	18.52	17.84
Transformer	0	96.19%	23.10	18.92	18.32
	1	96.07%	21.41	19.12	18.39
	2	94.59%	26.77	23.64	22.19

- 사용자의 실시간 이동상황에서 최악의 경로이동 (Worst-Case)를 가정하고, 바뀐 경로 시간에 대한 평균과 편차를 동시에 예측

Performance Measure

Online Test : Public Transportation

Online environment

Moving agent의 속도:

0km/h ~ 15km/h

Moving agent의 방향:

랜덤으로 선택: $0 \sim 2\pi$

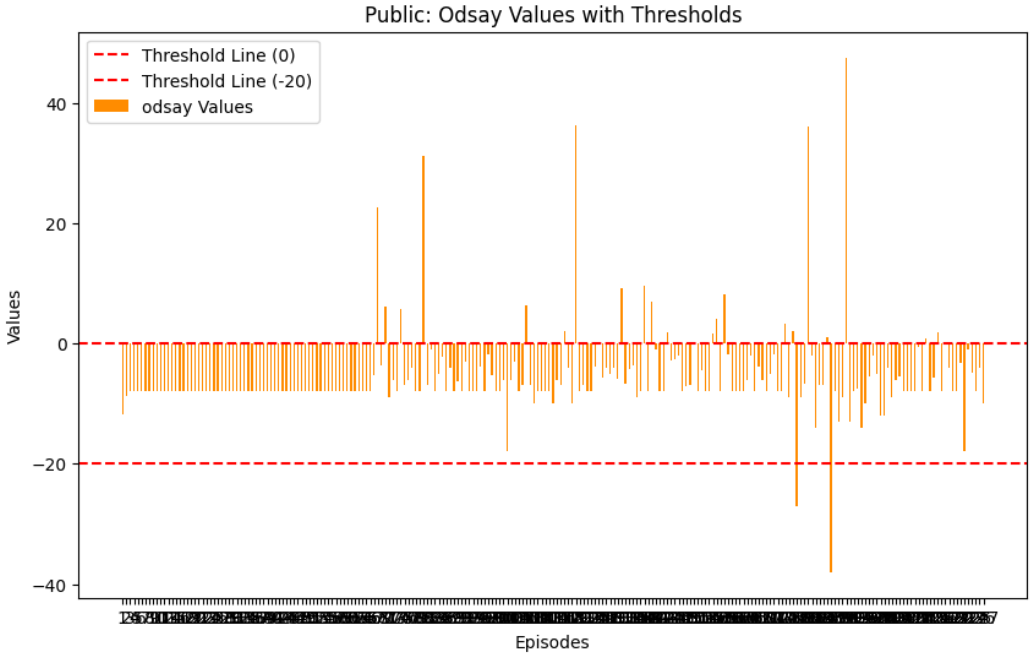
→100m 반경이 넘어갈 경우 새로운 Context를 저장한다.

Online test for Dynamic case

Total Episodes: 227

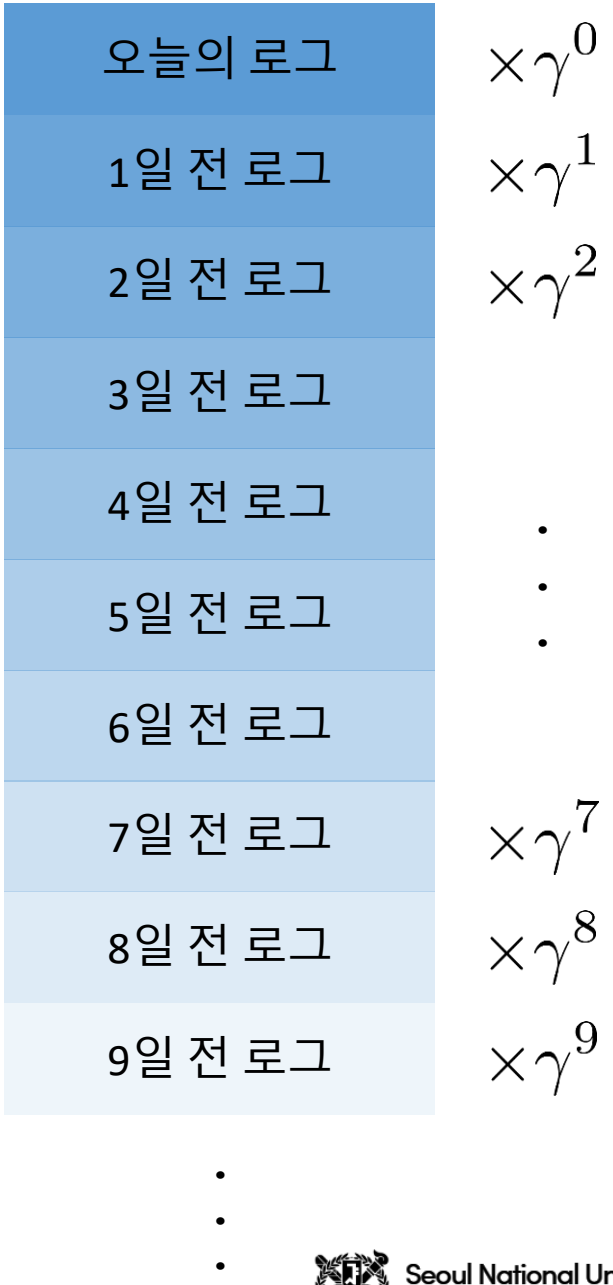
Odsay Data:

In range: 204
Early Arrival: 2
Late Arrival: 21
Avg # of API call: 6.215
Percentage in range: 89.87%



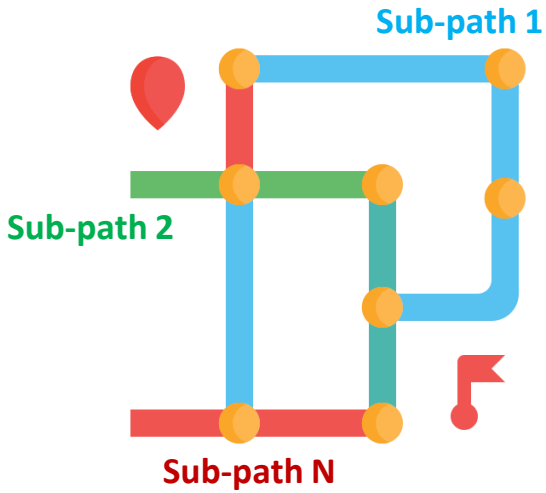
BatchUpdate

- 로그 기록에서 데이터를 샘플링하여 모델을 **일단위 업데이트**
- 할인율 $\gamma = 0.9$을 사용하여 **최근 데이터를 우선**적으로 반영하면서도, 이전의 실적 Data들도 함께 Update하여 강건성 확보



Plan for Improvement

PersonalCar : Sub-Path



Sub-path 1: $\hat{\sigma}_1$
Sub-path 2: $\hat{\sigma}_2$
⋮
Sub-path N: $\hat{\sigma}_N$

$$\sum_{i=1}^{N} \hat{\sigma}_i$$

Sub-set: $S_k = \{\hat{\sigma}_1, \hat{\sigma}_3, \dots\}$
Sub-set: $S_m = \{\hat{\sigma}_2, \hat{\sigma}_3, \dots\}$
⋮

중간경로 활용 방안

- API call을 통해 얻은 중간 경로 정보를 이용하여 구간별 교통 uncertainty 예측.
- Uncertainty가 큰 구간에서 추가 API call 의사결정 고도화.
- 경로가 비슷한 user들과 중간 경로 uncertainty 공유.

알고리즘 고도화 방안

#1 Sub-Path별 sigma의 합

- 실시간으로 수집된 Data log를 통해 sub-path 별 uncertainty 예측.
- Sub-path별 sigma 예측 값 저장.
- 저장된 Sub-path가 포함된 작업들이 감지될 경우, 이때 예측한 sub-path's sigma들을 전체 합산하여 사용.

#2 적응형 sigma

- 실시간 교통 상황, 남은 시간 및 총 API call 수에 따라 동적으로 sigma 조정
- Sub-paths uncertainty가 높을수록 sigma 호출 빈도 증가.
- Sub-paths uncertainty가 낮을수록 sigma 호출 빈도 감소.
- 출발 필요시각 임박 시 더 세밀한 보정 적용.

Plan for Improvement

PersonalCar : Reinforcement Learning

RL Algorithm preview

State:

현재 시간 t, 현재 남은 예상 시간 Δt, 현재 교통 상황에 대한 사전 추정치, 이전 API 호출 후 얻은 실시간 예측, API 호출을 안 한 기간, 교통 상황의 불확실성 등 **다양한 context 정보 고려**:

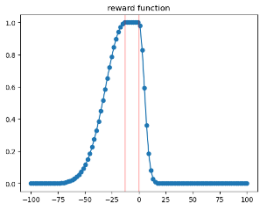
$$s_t = \{ t, \Delta t, context \}$$

Action:

$a_t \in \{0,1\}$ $0: No\ API\ call, 1: API\ call$

Multi-Objective Reward:

$$\boldsymbol{Reward = R_1 - \beta \cdot P}$$



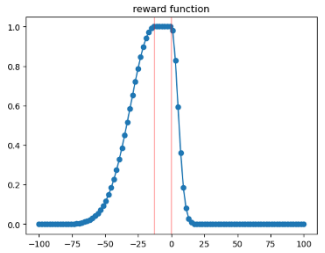
$\boldsymbol{R_1}$: 목표시간내 도착 Reward

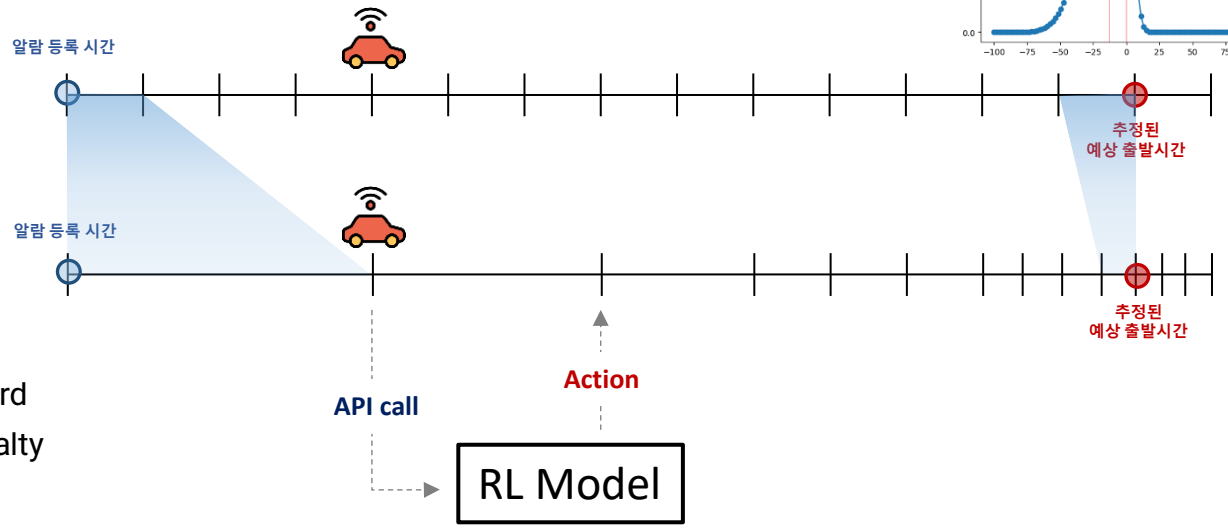
$\boldsymbol{P}$: API Call수에 따른 Penalty

- **Exploration 효율성 향상**

최적의 출발 시간과 멀리 떨어진 구간에서는 Exploration 횟수 감소

최적의 출발시간과 가까운 구간에서는 Exploration 횟수 증가



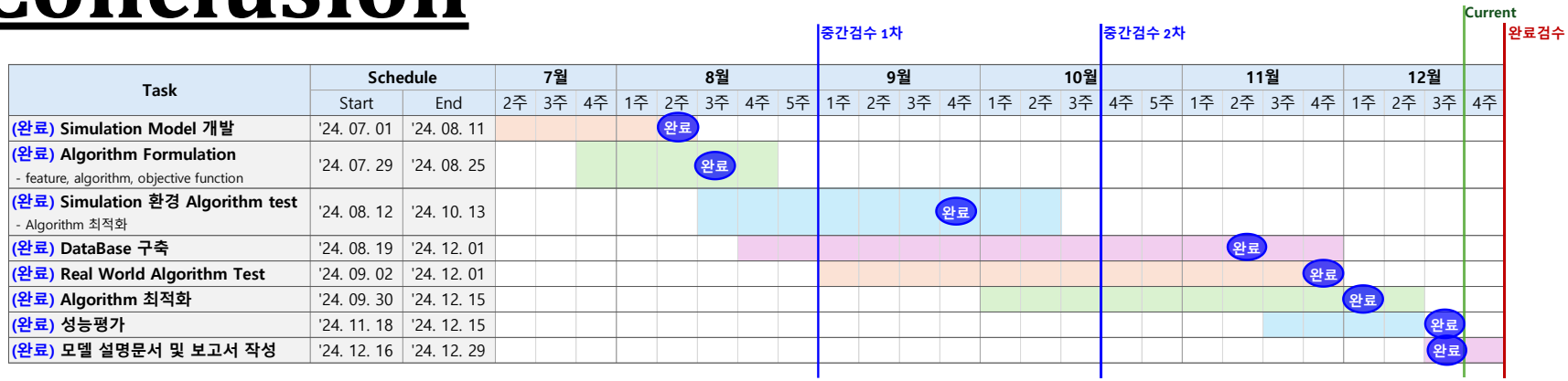


Given Data:

Offline Data log: $\mathcal{D}_{offline} = \{(s_i, a_i, r_i, s_i')\}$

Online Data log: $\mathcal{D}_{online}$

Conclusion



Task	Schedule		7월			8월					9월				10월			11월				12월			
	Start	End	2주	3주	4주	1주	2주	3주	4주	5주	1주	2주	3주	4주	1주	2주	3주	4주	5주	1주	2주	3주	4주	1주	2주	3주	4주
(완료) Simulation Model 개발	'24. 07. 01	'24. 08. 11					완료																				
(완료) Algorithm Formulation - feature, algorithm, objective function	'24. 07. 29	'24. 08. 25						완료																			
(완료) Simulation 환경 Algorithm test - Algorithm 최적화	'24. 08. 12	'24. 10. 13												완료													
(완료) DataBase 구축	'24. 08. 19	'24. 12. 01																			완료						
(완료) Real World Algorithm Test	'24. 09. 02	'24. 12. 01																					완료				
(완료) Algorithm 최적화	'24. 09. 30	'24. 12. 15																						완료			
(완료) 성능평가	'24. 11. 18	'24. 12. 15																							완료		
(완료) 모델 설명문서 및 보고서 작성	'24. 12. 16	'24. 12. 29																							완료		

PersonalCar Algorithm

- 정보획득 및 출발필요시간에 대해 Recurrently 평균 및 편차 동시 예측을 통한 확률기반 모델 성능 보장
- 사용자 이동시 최악의 경로 상황(Worst-Case)을 고려한 정지상태 알고리즘 유효성 보상
- 기존 BaroNow Rule-Based 모델대비 API Call 횟수를 줄였음에도, 동등 이상 수준의 성능을 확보
  . Offline Test (Static-Case) : Accept 비율 **94.5%**, Excess 비율 **3.0%**, API Call수 **평균 3.6회** (Rule; Accept :94.0%, Excess 2.5%, API Call수 5.1회)
  . Online Test (Dynamic-Case; 289 Case) : Accept 비율 **95.0%**, Excess 비율 **1.7%**, API Call수 **평균 3.9회** (Resnet; 24hr:commute = 1:1 기준)
  Accept 비율 **94.9%**, Excess 비율 **0.6%**, API Call수 **평균 4.9회** (Transformer ; 24hr:commute = 1:1 기준)

Public-Transportation Algorithm

- 사용자 이동 Case 모델링, 최악의 경로 상황(Worst-Case)을 가정하고 평균과 편차 동시 예측을 통한 성능 보장
- 실시간 교통상황 정보가 없음에도 우수한 성능 확보
  . Online Test (Dynamic-Case; 227 Case) : Accept 비율 **89.9%**, Excess 비율 **9.6%**, API Call수 평균 **6.1회**

Plan for Improvement

1) Sub-Path 활용 알고리즘 고도화
2) Reinforcement Learning 적용을 통한 모델 고도화

[Reference] PersonalCar Algorithm

Offline Test (Static Case) – Hyper-parameter Tunning

전체 시간대 기준 성능

test_type	model	n_episodes	alpha	gamma	max api_call	min api_interval	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
24 hr	Rule	300	3	1000	3	1	21.00%	0.00%	79.00%	1.99	-21.13	9.94		-24.32
	Alg3	300	3		3	1	91.70%	1.70%	6.70%	1.98	-7.05	2.88	1.09	-16.58
	resnet	300	3	1000	3	1	92.00%	1.70%	6.30%	2.00	-7.65	3.21	1.78	-16.16
	transformer	300	3	1000	3	1	83.00%	4.30%	12.70%	1.99	-6.94	4.16	2.62	-15.31
	Rule	300	3		3		95.00%	0.30%	4.70%	5.24	-8.49	2.81	4.08	-16.72
	Rule	10000	3		3		95.30%	0.90%	3.80%	5.03	-8.58	2.63	2.47	-15.71
	Alg3	10000	3		3		90.30%	2.40%	7.30%	1.98	-6.97	3.15	2.32	-15.65
	resnet	300	3	1.5	3		96.30%	0.30%	3.30%	3.49	-8.39	2.31	0.60	-16.01
	resnet	300	4	2	3		97.70%	0.70%	1.70%	3.51	-8.27	2.87	2.43	-25.04
	resnet	10000	4	2	3		95.80%	1.50%	2.80%	3.49	-8.14	2.79	2.88	-15.69
	resnet	300	4	2	6		93.70%	3.30%	3.00%	3.43	-6.54	3.46	3.18	-14.34
	resnet	300	4	2.5	3		96.00%	1.00%	3.00%	3.28	-8.03	2.51	1.77	-15.00
	resnet	10000	4	2.5	3		95.90%	1.10%	3.00%	3.33	-7.95	2.71	2.69	-15.21
	resnet	300	4	2.5	6		93.00%	1.30%	5.70%	3.31	-7.07	3.03	0.72	-15.90
	resnet	300	5	2.5	3		94.70%	1.30%	4.00%	3.52	-8.00	2.80	2.89	-15.92
	transformer	300	3	1.5	3		94.00%	1.70%	4.30%	4.04	-8.26	2.76	1.64	-15.18
	transformer	300	4	2	3		94.70%	0.70%	4.70%	3.83	-8.43	2.77	0.33	-18.24
	transformer	10000	4	2	3		95.50%	1.10%	3.40%	3.84	-8.26	2.70	2.94	-15.51
	transformer	300	4	2	6		94.70%	3.00%	2.30%	3.52	-6.76	3.31	3.04	-13.59
	transformer	300	4	2.5	3		95.70%	2.70%	1.70%	3.67	-7.82	2.81	1.10	-14.13
	transformer	10000	4	2.5	3		95.50%	1.40%	3.10%	3.62	-7.94	2.79	2.32	-15.35
	transformer	300	4	2.5	6		95.70%	1.30%	3.00%	3.46	-6.73	3.04	2.09	-16.47
	transformer	300	5	2.5	3		94.70%	1.00%	4.30%	3.89	-8.08	2.63	1.29	-15.76

[Reference] PersonalCar Algorithm

Offline Test (Static Case) – Hyper-parameter Tunning

출퇴근 시간대 기준 성능

test_type	model	n_episodes	alpha	gamma	max api_call	min api_interval	accept ratio	excess ratio	early ratio	mean n_api_call	total mean	total std	excess mean	early mean
commute	Rule	274	3	1000	3	1	27.70%	2.60%	69.70%	2.00	-20.17	12.39	4.86	-25.95
	Alg3	279	3		3	1	86.40%	12.20%	1.40%	2.00	-5.29	5.88	6.28	-14.64
	Alg3	9195	3		3	1	84.30%	13.20%	2.60%	2.00	-5.28	5.64	5.58	-15.65
	resnet	267	3	1000	3	1	79.80%	9.40%	10.90%	2.00	-6.97	5.90	4.81	-16.36
	transformer	284	3	1000	3	1	75.40%	15.50%	9.20%	2.00	-5.17	7.03	5.56	-19.13
	Rule	282	3		3		91.50%	4.30%	4.30%	4.88	-8.26	3.99	3.54	-16.99
	Rule	9167	3		3		92.70%	4.20%	3.10%	5.08	-7.96	3.87	3.46	-16.33
	resnet	276	3	1.5	3		93.10%	4.30%	2.50%	3.86	-7.57	4.41	6.73	-14.82
	resnet	275	4	2	3		94.90%	3.60%	1.50%	3.78	-7.21	3.48	2.21	-16.68
	resnet	9179	4	2	3		93.10%	4.70%	2.20%	3.81	-7.25	3.91	4.02	-15.79
	resnet	281	4	2	6		90.70%	7.50%	1.80%	3.84	-5.94	4.73	5.04	-14.69
	resnet	276	4	2.3	3		90.20%	8.00%	1.80%	3.72	-6.52	4.78	5.13	-16.27
	resnet	274	4	2.5	3		93.40%	4.70%	1.80%	3.61	-7.00	4.04	4.82	-17.65
	resnet	9180	4	2.5	3		92.60%	5.50%	1.90%	3.61	-6.79	4.12	4.51	-16.40
	resnet	276	4	2.5	6		91.70%	6.90%	1.40%	3.66	-5.87	4.67	5.95	-15.73
	resnet	273	5	2	3		92.70%	5.90%	1.50%	4.05	-7.17	4.19	4.42	-16.43
	resnet	256	5	2.3	3		92.20%	5.90%	2.00%	3.87	-6.90	3.31	1.69	-13.98
	resnet	269	5	2.5	3		92.90%	5.60%	1.50%	3.81	-6.92	4.04	4.71	-13.99
	resnet	271	6	3	3		92.30%	7.00%	0.70%	3.79	-6.32	4.03	3.82	-15.32
	transformer	285	3	1.5	3		93.00%	4.60%	2.50%	3.67	-7.55	3.86	3.82	-16.77
	transformer	272	4	2	3		93.00%	5.10%	1.80%	3.71	-7.24	3.74	2.98	-14.00
	transformer	9166	4	2	3		91.80%	6.00%	2.30%	3.65	-6.99	4.39	4.92	-15.90
	transformer	283	4	2	6		89.40%	8.80%	1.80%	3.31	-6.10	4.50	3.32	-15.42
	transformer	277	4	2.3	3		92.10%	6.90%	1.10%	3.55	-6.67	4.71	6.20	-14.87
	transformer	274	4	2.5	3		91.20%	6.20%	2.60%	3.43	-6.84	4.26	4.48	-15.99
	transformer	9182	4	2.5	3		91.00%	7.20%	1.80%	3.46	-6.57	4.40	4.20	-15.39
	transformer	274	4	2.5	6		90.10%	9.10%	0.70%	3.25	-5.89	4.26	3.05	-19.81
	transformer	275	5	2	3		90.20%	6.20%	3.60%	3.91	-7.19	4.24	4.26	-14.89
	transformer	283	5	2.3	3		91.90%	5.30%	2.80%	3.64	-6.89	3.95	3.83	-14.87
	transformer	271	5	2.5	3		89.30%	7.40%	3.30%	3.70	-6.33	4.92	5.14	-16.32
	transformer	281	6	3	3		89.70%	7.80%	2.50%	3.78	-6.89	5.05	4.63	-18.78