Cross-data Automatic Feature Engineering via Meta-learning and Reinforcement Learning

Jianyu Zhang[1(✉)], Jianye Hao[1], and Françoise Fogelman-Soulié[2]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
{edzhang,jianye.hao}@tju.edu.cn
[2] Hub France IA, Paris, France
francoise.soulie@hub-franceia.fr

PAKDD 2020

Total cites : 0

2021.07.21

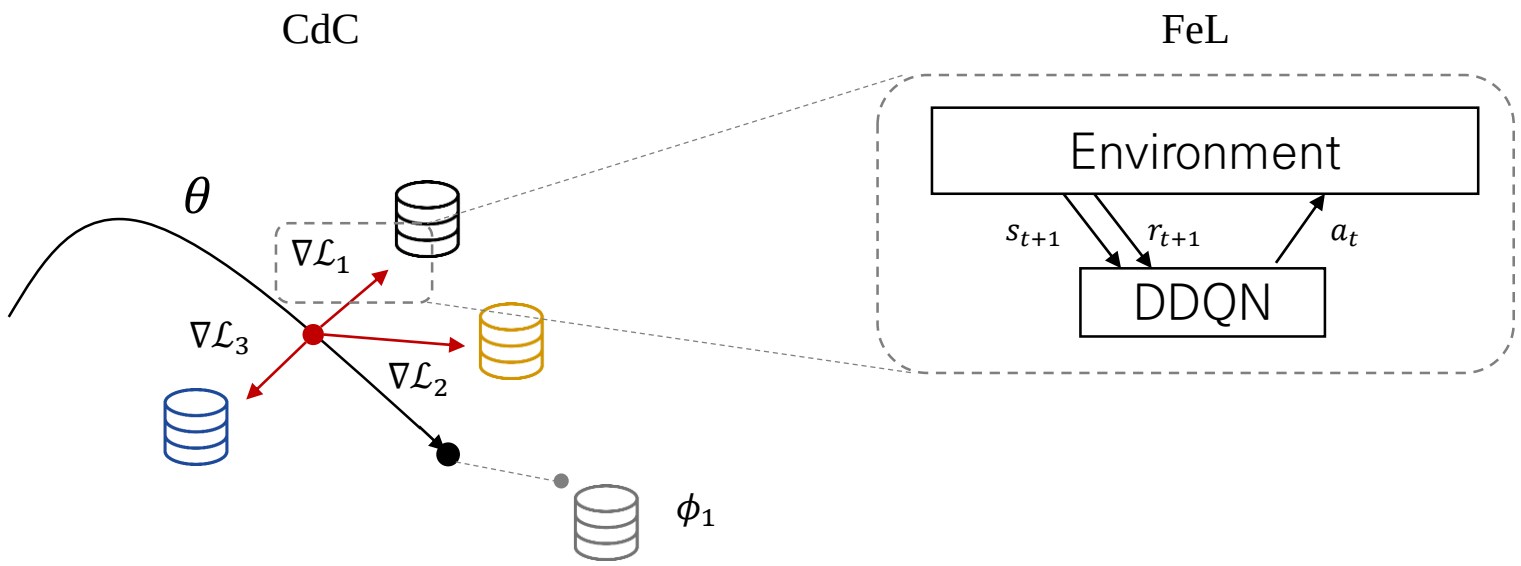
최영제

Index

1 Introduction

Overview of cross-data automatic feature engineering machine (CAFEM)

- Automated feature engineering (AutoFE) + meta-learning
- 단일 dataset에 국한되지 않는 general한 algorithm을 구축하고자 함
- 크게 두 가지 part로 구성이 되며 각각 1) feature engineering learner (FeL), 2) cross-data component (CdC)로 명명함

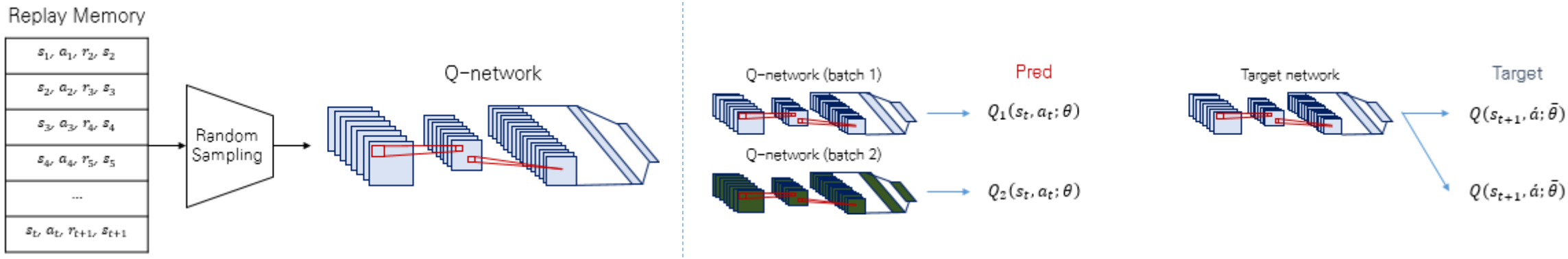


1 Introduction

Deep q-network (DQN)

- DQN 이전 neural network를 강화학습에 적용한 approach는 다음과 같은 단점이 존재하였음
  1) Small volum of training data
  2) High correlation between samples
  3) Non-stationary target problem

- DQN은 experience replay와 fixed Q-target으로 위 문제를 해결하였음

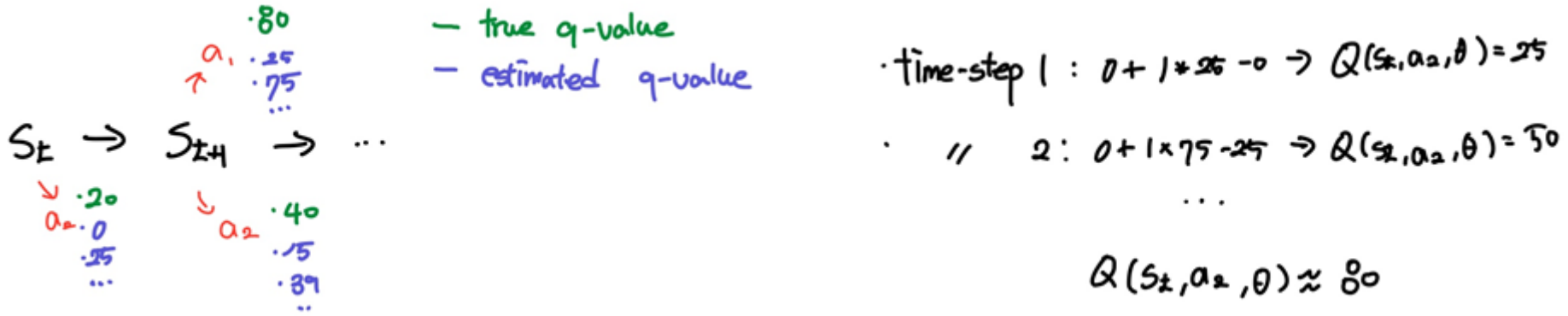


$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

Introduction

Double deep q-network (DDQN)

- Q-learning을 기반으로 하는 DQN의 경우 다음 시점의 maximum q-target을 사용하기에 overestimation 문제가 발생함(single estimator)



- DDQN은 이를 해결하기 위해 q-value를 구하는 network (target network)와 action을 select 하는 network (DQN)를 분리하였음 (double estimator)

$$a = \max_a Q_{qnet}(s_{t+1}, a) \qquad q_{\text{estimated}} = Q_{\text{tnet}}(s_{t+1}, a) \qquad Q_{qnet}(s_t, a_t) \leftarrow R_{t+1} + \gamma Q_{tnet}(s_{t+1}, a)$$

$$Y_t^{\text{Double DQN}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t^-\right)$$

The update to the target network stays unchanged from DQN, and remains a periodic copy of the online network.

Model agnostic meta learning (MAML)

- Meta learning의 목적은 기존 tasks의 학습 경험을 바탕으로 새로운 task에 대하여 빠르게 학습하는 것임
- 각각의 task T_i에 대해서 가중치 θ를 adaptation 후 이를 토대로 meta-learning을 진행

$$\theta_i^{'} = \theta - \alpha \bigtriangledown_\theta \mathcal{L}_{T_i}(g_\theta)$$

[Adaptation]

$$\theta = \theta - \beta \bigtriangledown_\theta \sum_{T_i \in \{T\}} \mathcal{L}_{T_i}(g_{\theta_i'})$$

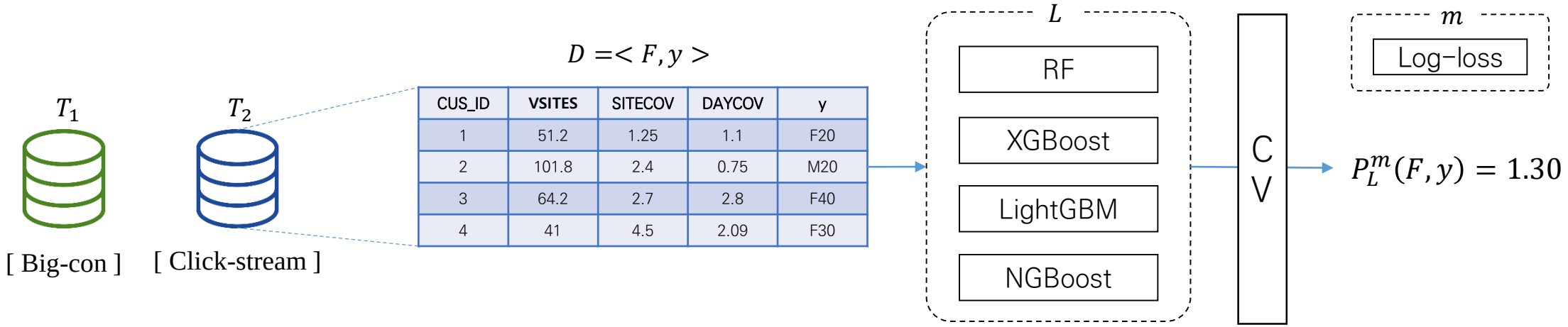
[Meta-learning]

- 본 논문에서 가중치 θ는 DDQN의 parameters이며 각각의 tasks는 서로 다른 datasets이 해당

2 SIL Methods

Problem formulation

- A collection of typical supervised learning tasks: $T = \{T_1, T_2, \dots, T_N\}$
- Each task T_i can be represented as $T_i =< D, L, m >$,where $D =< F, y >$ is a dataset with a set of features $F = \{f_1, f_2, \dots, f_n\}$
- A corresponding target variable y, L is a learning algorithm (e.g. Random Forest, Logistic Regression)
- m is a evalutation metric (e.g. relative absolute error (RAE), f1-score)



Methods

Problem formulation

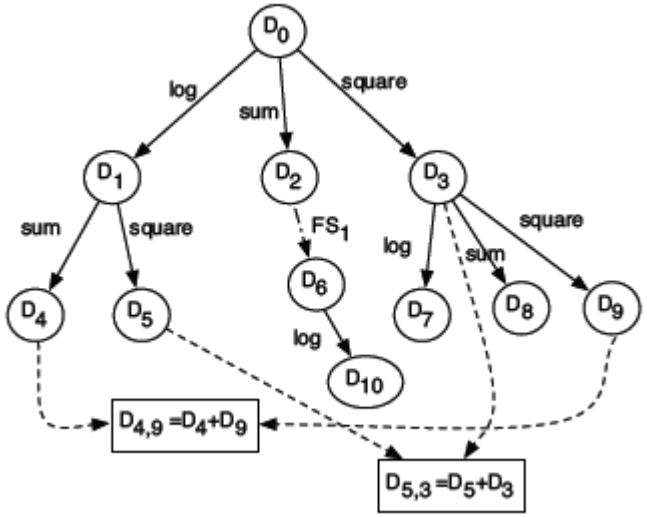
- τ is transformation operator and f_+ is the set of derived features.
  - Order-1 new features: $f_+ = \tau(\{f_i\})$
  - Order-2 new features: $f_+ = \tau(\{f_i, f_j\})$
- Feature engineering aims at constructing a subset of features $F^* = F_o \cup F_+ - F_-$
  - F_o is the set of original features in dataset
  - F_- is the set of features that decide to drop out from original features
- The goal of feature engineering is to find a good policy π^* that maximizes the model performance

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, P_L^m(\pi(D), y)$$

Methods

Related works: Transformation Graph (AAAI '18)

- TransGraph는 아래 그림과 같이 datasets에 존재하는 모든 feature에 대해서 동일한 operation을 진행 후 이를 결합 및 selection 진행
- 이에 따라 graph의 leaf node로 갈수록 feature dimension이 급격히 증가하는 특징을 보임



[TransGraph architecture]

CUS_ID	VSITES	SITECOV	y
1	51.2	1.25	F20
2	101.8	2.4	M20
3	64.2	2.7	F40

log

CUS_ID	VSITES	SITECOV	y
1	51.2	1.25	F20
2	101.8	2.4	M20
3	64.2	2.7	F40

sqaure

CUS_ID	VSITES	SITECOV	y
1	51.2	1.25	F20
2	101.8	2.4	M20
3	64.2	2.7	F40

CUS_ID	VSITES_l	SITECOV_l	VSITES_s	SITECOV_s	y
1	1.709	0.09	2621.44	1.56	F20
2	2.007	3.80	10363.2	5.76	M20
3	1.807	4.31	4121.64	7.29	F40

Methods

Feature transformation graph

- Feature transformation graph (FTG)는 개별 feature 마다 FE process를 적용함
- FTG는 G로 표기하며 directed acyclic dynamic graph임
- 각각의 nodes는 feature를 의미하고 edge는 transformation operator τ를 뜻함
- FE의 시작 지점에서 G는 총 n개(# of features)의 node를 보유하며 t 시점의 FTG를 도식화하면 아래와 같음

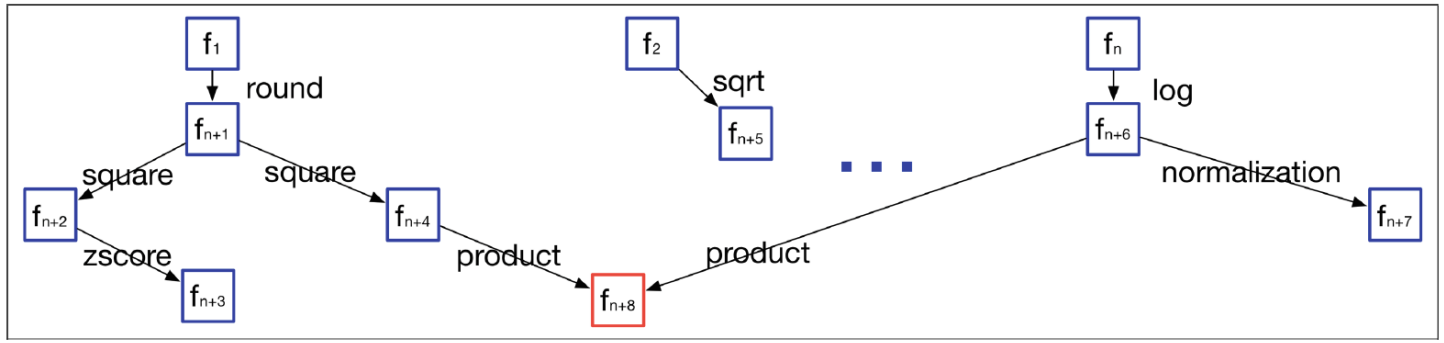


Fig. 1. *Example of FTG*

Methods

Markov decision precess formulation

- 최적의 transformation rules를 찾는 방법은 graph search도 존재(Cognito, ICDM ‘16)하나 저자들은 강화학습을 사용함
- 매 time-step t에서 state s_t는 graph에 대한 정보 G_t와 feature 집단 $\{f_i\}$로 구성됨
  - $\{f_i\}$는 하나 혹은 여러 개의 features로 구성될 수 있음
  - 예를 들어 order-1 operator (e.g. log)이면 하나의 feature를, order-2 operator (e.g. product)이면 두 개의 features를 포함
- Action a_t는 성격에 따라 1) feature generation, 2) feature selection으로 구분할 수 있음
  - Feature selection이 선택될 경우 current feature f_t를 버리고, previous feature f_{t-1}로 돌아감
  - 만약 $t = 1$ 일 때 feature selection이 선택되면 해당 feature를 버리고 FE process를 중단함
- Reward r_t는 직전 time-step과의 performance 증가량으로 구성됨

$$r_t = P_L^m(D_{t+1}) - P_L^m(D_t)$$

Methods

CAFEM Framework – feature engineering learner (FeL)

- FeL은 단일 dataset에 대해서 DDQN을 학습하는 과정임
- $\{\mathbb{A}_i\}$로 표기되는 DDQN agent는 # of features 만큼 존재하며 순차적으로 학습됨

Algorithm 1. FeL

input: An dataset $D = \langle F, y \rangle$ with n features $F = \{f_0, ...f_n\}$, n replay buffer $\{R_i, ..., R_n\}$ for each features, n off-policy DRL agents $\{\mathbb{A}_i\}$, number of epochs and episodes E, M, batches to train N

```
1: while epoch = 1, E do
2:     for f_i in F do
3:         for episode = 1, M do
4:             Get initial state s_0
5:             while not terminal do
6:                 Get an action a_t by ε-greedy and execute a_t on f: f̂ = a_t(f)
7:                 Obtain reward r_t and next state s_{t+1}
8:                 Store transition (s_t, a_t, r_t, s_{t+1}) in R_i and reset current feature: f ← f̂
9:             end while
10:        end for
11:        for t = 1, N do
12:            Sample a mini-batch from replay buffer R_i
13:            Perform one optimization step on A_i
14:        end for
15:    end for
16:    Reset dataset D = ⟨{f_i, ..., f_n}, y⟩
17: end while
```

Feature loop
Line 3-14

Line 3-10: epsilon greedy policy, stores transition in replay buffer

Line 11-14: training DDQN agent

Methods

CAFEM Framework – cross data component (CdC)

- FE process의 속도를 높이고 다양한 datasets으로 부터 오는 학습 이점을 활용하기 위해 MAML을 적용함
- 각각의 DDQN agent는 g_θ로 표기됨

Algorithm 2. CrossDataComponent

input: a set of tasks $\{T\}$, an off-policy DRL agent represented by g_θ, number of epochs and episodes E, M

Randomly initialize θ
while epoch $= 1, E$ **do**
  Sample batch of tasks $\{T_i\}$ from $\{T\}$
  for all $\{T_i\}$ **do**
    Perform M episodes on task T_i with ϵ-greedy
    Store all transitions in R_i
    Sample K transitions $\mathcal{T}$ from R_i
    Compute adapted parameters with gradient descent: $\theta_i^{'} = \theta - \alpha \bigtriangledown_\theta \mathcal{L}_{T_i}(f_\theta)$
    Sample transitions $\mathcal{T}_i^{'}$ from R_i for meta-update
  end for
  Update $\theta = \theta - \beta \bigtriangledown_\theta \sum_{T_i \sim \{T\}} \mathcal{L}_{T_i}(f_{\theta_i^{'}})$ using each $\mathcal{T}_i^{'}$ and $\mathcal{L}_{T_i}$ in Equation 3
end while

Meta-update

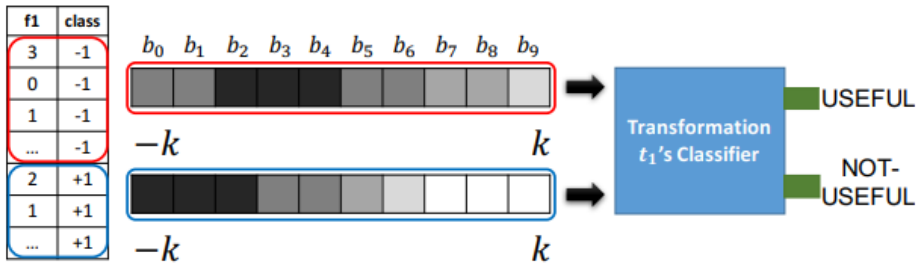
Methods

CAFEM Framework – network design

- 앞서 언급하였듯 state s_t는 graph에 대한 정보 G_t와 feature 집단 $\{f_i\}$로 구성되며 이를 자세히 서술하면 다음과 같음
  - Previous N-step FE history on FTG
  - The number of each transformation operators used in G_t
  - The number of next node visited for each action
  - The number of each operator used from $\{f_t\}$ to its root
  - Node depth of a feature in HTG (?)
  - Average performance improvement of each action
  - Extended quantile sketch array (ExQSA) representation of features

Totally, we use 293 features to represent each state. A neural network with three fully connected hidden layers (128-128-64 neurons) and ReLU activation function is used to approximate Q-values.

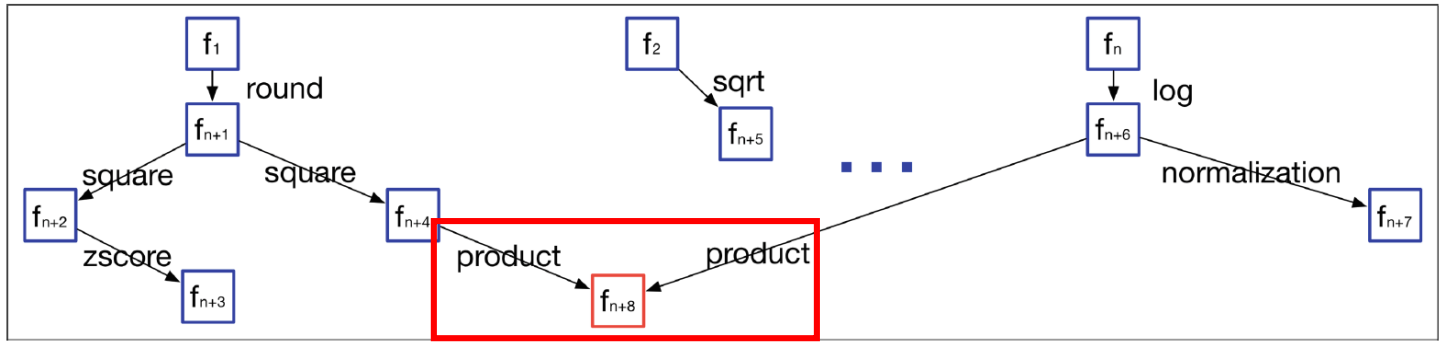
- Quantile sketch array는 feature representation 방법 중 하나이며 다음과 같이 적용 가능함



1. Original feature에 binning을 적용
2. Target class별 binning의 분포로 표현
3. ExQSA는 이를 regression target으로 확장한 방법 (y < median(y) and y < median(y))

Methods

Remaining questions



어떻게 구성하지 ?

As in [15], we introduce features along feature complexity, driving simple features first (e.g. unary features) then complex features (e.g. binary features).

3 Experiments and Results

Experimental settings

- 총 120개의 dataset을 확보 후 100개는 meta-train, 20개는 meta-test로 활용
- Transformation operators는 총 13개를 사용하였음
  - Order-1: log, round, sigmoid, tanh, square, square root, zscore, min-max
  - Order-2: sum, difference, product, division
- Learning algorith으로는 random fores와 logistic regressio을 사용하며 F1-score와 1-RAE를 performance measure로 사용
- 5-fold cross validation을 사용하였으며 order-2 operation의 경우 feature 후보군이 상당히 많기에 FeL은 time-step 별 sampling을 적용

Experiments and Results

Experimental settings

- 비교 대상은 다음과 같음
  - Baseline: applies learning algorithm on original dataset
  - Random-FeL (RS): FTG에 random graph search method를 적용하여 최적의 rules 탐색 (Cognito와 유사)
  - Brute-force (BF): 모든 original features에 적용 가능한 transformation을 적용 후 feature selection을 진행
  - LFE: IJCAI ’17에 등재된 AutoFE 방법론, classification task만 가능
  - FERL: AAAI ’18에 등재된 Transformation Graph를 변형(Order-2에서 연산량을 줄이기 위해 pruning을 진행)
- 위 방법 중 source code가 공개되지 않은 것들이 존재하여 저자들이 맞게끔 구축함

Experiments and Results

Results

Datasets	#Row	#Feature	Baseline	Order-1					Order-1 & 2				
				FeL	BF	LFE	RS	FERL	FeL	BF	LFE	RS	FERL
Balance_scale	625	5	88.2%	88.3%	86.4%	88.2%	88.2%	**88.6%**	95.0%	**97.0%**	95.1%	92.7%	-
Boston	506	21	88.2%	**90.2%**	86.7%	89.2%	89.5%	88.7%	**89.9%**	85.6%	88.2%	89.8%	-
ClimateModel	540	21	95.5%	**96.0%**	95.6%	95.5%	95.7%	95.9%	**96.1%**	95.5%	95.5%	**96.1%**	-
Cpu_small	8,192	13	86.3%	**87.1%**	84.5%	85.8%	86.6%	86.8%	**87.1%**	86.2%	86.3%	87.0%	-
Credit card	14,240	31	50.5%	**68.7%**	64.8%	50.5%	63.8%	64.0%	**71.4%**	65.1%	65.1%	64.6%	-
Disclosure_x	662	4	44.8%	**51.7%**	46.6%	46.8%	49.7%	49.8%	51.4%	46.4%	46.4%	51.4%	**51.8%**
Disclosure_z	662	4	53.8%	**57.7%**	55.6%	53.1%	55.6%	57.0%	**57.0%**	53.8%	55.0%	56.7%	56.9%
fri_c1_1000_25	1,000	26	84.9%	87.7%	85.8%	85.8%	86.7%	**88.0%**	**87.1%**	77.9%	82.1%	**87.1%**	-
Fri_c2_100_10	1,000	11	86.3%	**89.7%**	85.8%	86.8%	88.6%	89.3%	**91.0%**	87.2%	86.7%	89.3%	-
Fri_c3_100_5	1,000	6	88.2%	89.2%	88.5%	88.2%	88.4%	**89.4%**	**90.7%**	87.3%	87.1%	89.3%	-
fri_c3_1000_50	1,000	51	79.7%	83.7%	**88.5%**	80.9%	80.7%	87.8%	83.1%	**88.4%**	78.3%	80.8%	-
Gina_agnostic	3,468	971	92.3%	92.8%	78.9%	92.3%	92.8%	**93.5%**	**92.8%**	-	92.5%	92.8%	-
Hill-valley	1,212	101	57.5%	**61.7%**	59.2%	57.5%	60.8%	61.1%	**100%**	**100%**	57.5%	99.9%	-
Ilpd	583	11	41.3%	**45.7%**	38.7%	38.9%	43.6%	44.9%	**45.9%**	**45.9%**	42.4%	44.8%	-
Kc1	2,109	22	40.4%	**44.5%**	35.3%	38.9%	42.0%	42.7%	**44.4%**	39.9%	38.8%	43.4%	-
openml_589	1,000	25	66.9%	67.7%	55.0%	X	67.2%	**72.6%**	75.0%	**76.9%**	X	68.1%	-
Pc4	1,458	38	47.7%	57.0%	36.2%	45.3%	53.8%	**58.4%**	**58.1%**	50.1%	55.1%	56.5%	-
Pc3+C14	1,563	38	25.9%	**33.4%**	27.9%	23.0%	30.3%	32.0%	**33.3%**	24.6%	27.4%	31.6%	-
Spectrometer	531	103	77.3%	**83.9%**	80.0%	75.2%	80.4%	83.0%	82.7%	**90.8%**	73.2%	81.8%	-
Strikes	625	7	96.6%	**99.5%**	98.7%	97.8%	99.1%	98.9%	**99.5%**	97.8%	93.4%	99.4%	98.9%

Experiments and Results

Results

- Meta-learning의 효과는 아래와 같음

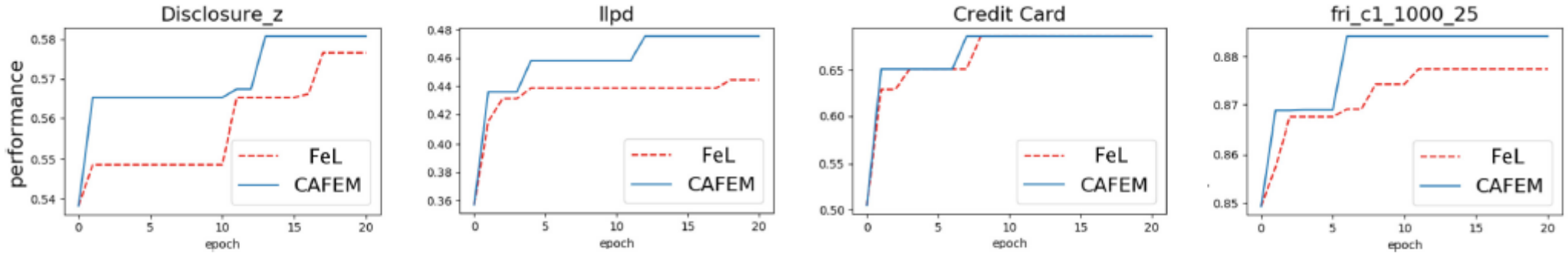


Fig. 2. *CAFEM vs FeL over 4 different datasets*

Q&A