Neural Feature Search: A Neural Architecture for Automated Feature Engineering

Xiangning Chen[1,*], Qingwei Lin[2,*,**], Chuan Luo[2,*], Xudong Li[3], Hongyu Zhang[4],
Yong Xu[2], Yingnong Dang[5], Kaixin Sui[2], Xu Zhang[6], Bo Qiao[2],
Weiyi Zhang[2], Wei Wu[7], Murali Chintalapati[5] and Dongmei Zhang[2]
[1]*Tsinghua University, China*
[2]*Microsoft Research, China*
[3]*University of California, Los Angeles, United States*
[4]*The University of Newcastle, Australia*
[5]*Microsoft Azure, United States*
[6]*Nanjing University, China*
[7]*University of Technology Sydney, Australia*

ICDM 2019

Total cites: 7

2021.06.30

최영제

Index

1 Introduction

Introduction & related works

- Traditional manual feature engineering relies heavily on domain knowledge
- Expansion-reduction approach
  - It first expands the feature space and then performs feature selection to reduce redundancy
  - There are several algorithms (i.e. FEADIS, Deep Feature Synthesis, One Button Machine, ExploreKit) aka heuristic approach

Perform all transformation functions on the whole dataset

- Genetic algorithm based approach: Cognito
- Reinforcement learning based approach: Transformation Graph
- Above methods are suffer from computational cost, feature space explosion problem

1 Introduction

Introduction & related works

- The contributions of neural feature search (NFS) as follow:
  - Proposed a novel neural architecture for automated feature engineering (AutoFE)
  - Neural feature search (NFS) approach does not suffer from the feature explosion problem
  - Significantly outperforms the existing state-of-the-art methods for AutoFE
- Neural architecture search (NAS)

the controller – to generate string which denoted hyperparameter of architectures

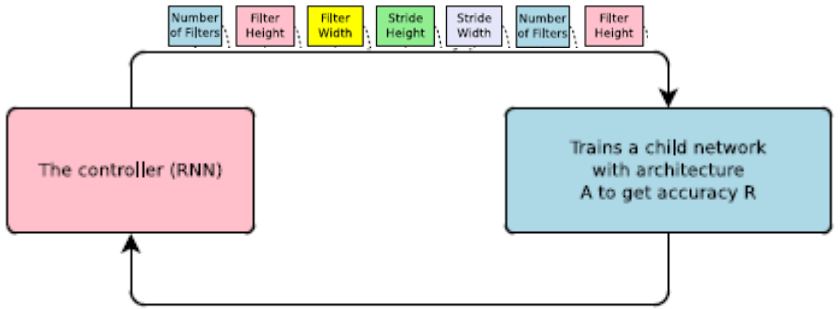


Figure 1: An overview of Neural Architecture Search.

the child network – training on the real data will result in an accuracy on a validation set

2 Methods

Overview of Neural Feature Search

- NFS는 controller로 복수의 recurrent neural networks (RNN)을 사용함(make transformation rules)
- Rules를 따라 생성된 dataset을 바탕으로 머신러닝 모델 적합 후 개선 정도를 reward로 할당
- 최적의 features를 탐색하는 architecture 구축

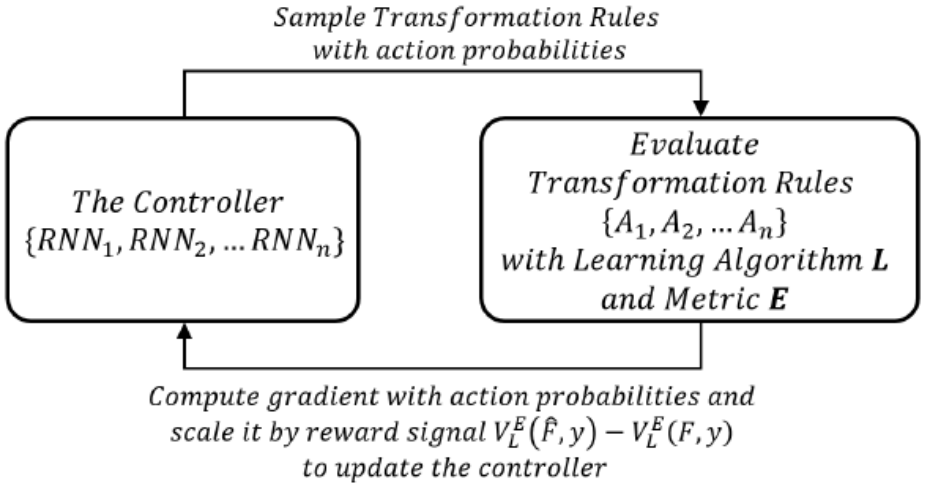


Figure 1. An Overview of Neural Feature Search

2 Methods

Problem Formulation

- Dataset: $D = < F, y >$ / Raw features: $F = \{f_1, f_2, \dots, f_n\}$ / Target vector: y
- Performance of machine learning: $V_L^E(F, y)$ / Algorithms: L / Evaluation metric: E
- Transformed feature set: $\hat{F}$
- Set of transformation functions: $\{A_1, A_2, \dots, A_n\}$
  - There are two categories of transformation functions: unary functions (e.g. square, logarithm), binary functions (e.g. plus, minus)
  - High-order features can be constructed by applying to recursive transformation (e.g. $BMI = \frac{weight}{height^2}$)
- Two activities of feature transformation
  - Generating new feature
  - Deleting exiting features

Generation of Transformation Rules by Controller

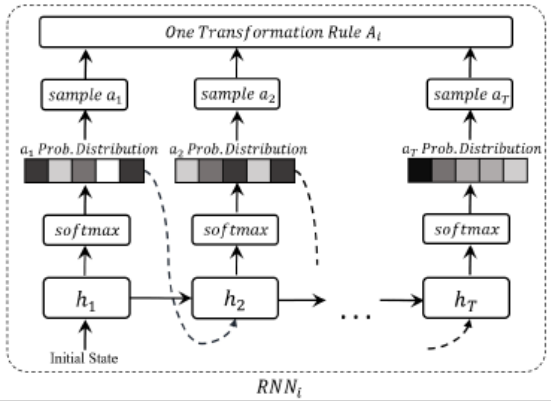
- RNN은 각 feature 마다 하나씩 존재하며 time step 마다 transformation function이 출력됨

F

f_1	f_2
2.4	-12
1.8	8
4.5	4
6.8	15
0.1	23

...

$A_1 = \{a_1, a_2, \ldots, a_T\}$



Note: Each RNN predicts transformation function a_t at time step t. A series of transformations $[a_1, a_2, .., a_T]$ form a transformation rule. Every prediction is carried out by a softmax classifier and then fed into the next time step.

Figure 2. The Generation of Transformation Rule by a RNN

$\hat{F}$

f_1	$\hat{f}_3$	$\hat{f}_4$
2.4	-12	-0.8
1.8	8	1.4
4.5	4	5.6
6.8	15	-2.1
0.1	23	0.01

$\hat{f}_i = a_t(a_{t-1}(\ldots a_2(a_1(f_i)))$

Machine learning

$V_L^E(\hat{F}, y) - V_L^E(F, y)$

$\{RNN_1, RNN_2, \ldots, RNN_n\} = \{A_1, A_2, \ldots, A_n\}$

Methods

Feature Reconstruction through High-order Transformation

- Two types of transformations: unary (takes one feature as input) and binary (takes two features as input)
- Unify a binary transformation by converting it into a set of unary transformations for each individual features f_i
  - Convert *Plus* transformation into a set of unary transformation *addition*(f_i)

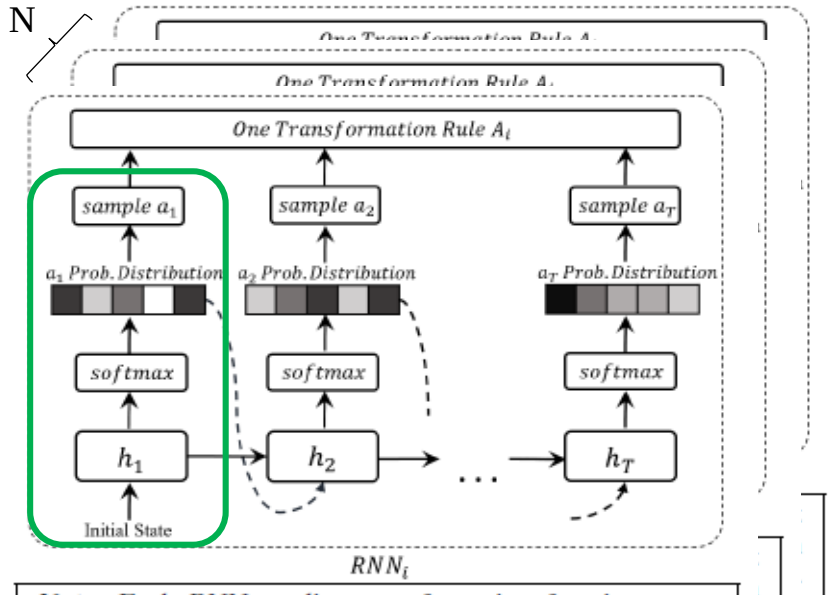
Unified transformations

Mathematical functions	Special functions
Logarithm, square, square-root, min-max-normalization, reciprocal, addition, subtraction, multiplication, division	*Delete, terminate*

- Two special functions enable NFS to eliminate features and stop at the most suitable order of transformation

Training with Policy Gradient

- 학습에 사용되는 policy gradient는 다음과 같음



Note: Each RNN predicts transformation function a_t at time step t. A series of transformations $[a_1, a_2, .., a_T]$ form a transformation rule. Every prediction is carried out by a softmax classifier and then fed into the next time step.

Figure 2. The Generation of Transformation Rule by a RNN

$$R_t = V_t - V_{t-1}$$

$V_t = V_L^E$, cross-validation score

$$G_t^{(k)} = R_t + \gamma R_t + \ldots + \gamma^k R_{t+k}$$

$$G_t^{\lambda} = (1-\lambda) \sum_{k=1}^{n \times T} \lambda^{k-1} G_t^{(k)}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{k=1}^{m} \sum_{t=1}^{n \times T} \nabla_\theta log P(a_t | a_{1:t-1}; \theta) G_{t|k|}^{\lambda}$$

Experiments and Results

Research questions & experimental settings

- RQ1: How effective is the proposed NFS approach?
- RQ2: How effective are the high-order transformations?
- RQ3: Is NFS robust to hyperparameters?

- Set the parameter beta to 50 (beta represents the top N variables by random forest variable importance)
- Use one-layer LSTM
- Evaluation metrics: F1-score (classification), 1-relative absolute error (regression)

$$1 - rae = 1 - \frac{\sum |\hat{y} - y|}{\sum |\bar{y} - y|}$$

3 SIL Experiments and Results

RQ1: How effective is the proposed NFS approach?

Random: random하게 feature generation
Random NFS: NFS 구조를 사용하되 학습 x

Table I
COMPARISON BETWEEN NFS AND THE EXISTING AUTOMATED FEATURE ENGINEERING METHODS

Dataset	Source	C\R	Instances\ Features	Random	Random$_{NFS}$	Exp-Red	Trans-Graph	NFS
Higgs Boson	UCIrvine	C	50000\28	0.699	0.723	0.682	0.729	**0.731**
Amazon Employee	Kaggle	C	32769\9	0.740	**0.945**	0.744	0.806	**0.945**
PimaIndian	UCIrvine	C	768\8	0.709	0.772	0.712	0.756	**0.805**
SpectF	UCIrvine	C	267\44	0.748	0.801	0.790	0.788	**0.876**
SVMGuide3	LibSVM	C	1243\21	0.753	0.834	0.711	0.776	**0.865**
German Credit	UCIrvine	C	1001\24	0.655	0.745	0.680	0.724	**0.805**
Bikeshare DC	Kaggle	R	10886\11	0.381	0.980	0.693	0.798	**0.990**
Housing Boston	UCIrvine	R	506\13	0.637	0.658	0.621	0.680	**0.686**
Airfoil	UCIrvine	R	1503\5	0.753	0.771	0.771	**0.801**	0.796
AP-omentum-ovary	OpenML	C	275\10936	0.710	0.831	0.725	0.820	**0.864**
Lymphography	UCIrvine	C	148\18	0.680	0.887	0.727	0.895	**0.929**
Ionoshpere	UCIrvine	C	351\34	0.934	0.923	0.939	0.941	**0.969**
Openml_618	OpenML	R	1000\50	0.428	0.403	0.411	0.587	**0.640**
Openml_589	OpenML	R	1000\25	0.571	0.511	0.650	0.689	**0.754**
Openml_616	OpenML	R	500\50	0.343	0.284	0.450	0.559	**0.673**
Openml_607	OpenML	R	1000\50	0.411	0.366	0.590	0.647	**0.688**
Openml_620	OpenML	R	1000\25	0.524	0.471	0.533	0.683	**0.732**
Openml_637	OpenML	R	500\50	0.313	0.248	0.581	0.585	**0.634**
Openml_586	OpenML	R	1000\25	0.549	0.495	0.598	0.704	**0.780**
Credit Default	UCIrvine	C	30000\25	0.766	0.798	0.802	**0.831**	0.799
Messidor_features	UCIrvine	C	1150\19	0.655	0.743	0.703	0.752	**0.762**
Wine Quality Red	UCIrvine	C	999\12	0.380	0.692	0.344	0.387	**0.708**
Wine Quality White	UCIrvine	C	4900\12	0.678	0.689	0.654	**0.722**	0.707
SpamBase	UCIrvine	C	4601\57	0.937	0.954	0.951	**0.961**	0.955

3 SIL Experiments and Results

RQ1: How effective is the proposed NFS approach?

Table II
COMPARISON BETWEEN NFS, LFE, AND OTHER RELATED METHODS

Dataset	Source	Instances\Features	Random	Exp-Red	LFE	Random$_{NFS}$	NFS
AP-omentum-lung	OpenML	203\10936	0.903	0.925	0.929	0.947	**0.981**
AP-omentum-ovary	OpenML	275\10936	0.714	0.801	0.811	0.833	**0.873**
credit-a	UCIrvine	690\6	0.657	0.521	0.771	0.770	**0.803**
diabetes	UCIrvine	768\8	0.729	0.737	0.762	0.760	**0.786**
fertility	UCIrvine	100\9	0.846	0.861	0.873	0.903	**0.913**
gisette	UCIrvine	2100\5000	0.871	0.741	0.942	0.948	**0.959**
hepatitis	UCIrvine	155\6	0.751	0.753	0.831	0.821	**0.905**
higgs-boson-subset	UCIrvine	50000\28	0.660	0.661	0.68	0.824	**0.827**
ionosphere	UCIrvine	351\34	0.899	0.912	0.932	0.941	**0.972**
labor	UCIrvine	57\8	0.877	0.855	0.896	0.853	**0.960**
lymph	OpenML	138\10936	0.594	0.534	0.757	0.943	**0.987**
madelon	UCIrvine	780\500	0.602	0.585	0.617	0.697	**0.836**
megawatt1	UCIrvine	253\37	0.865	0.882	0.894	0.897	**0.933**
pima-indians-subset	UCIrvine	768\8	0.729	0.751	0.745	0.760	**0.790**
secom	UCIrvine	470\590	0.911	0.913	0.918	0.931	**0.934**
sonar	UCIrvine	208\60	0.723	0.468	0.801	0.709	**0.839**
spambase	UCIrvine	4601\57	0.911	0.39	0.947	0.938	**0.948**

3 Experiments and Results

RQ2: How effective are the high-order transformations?

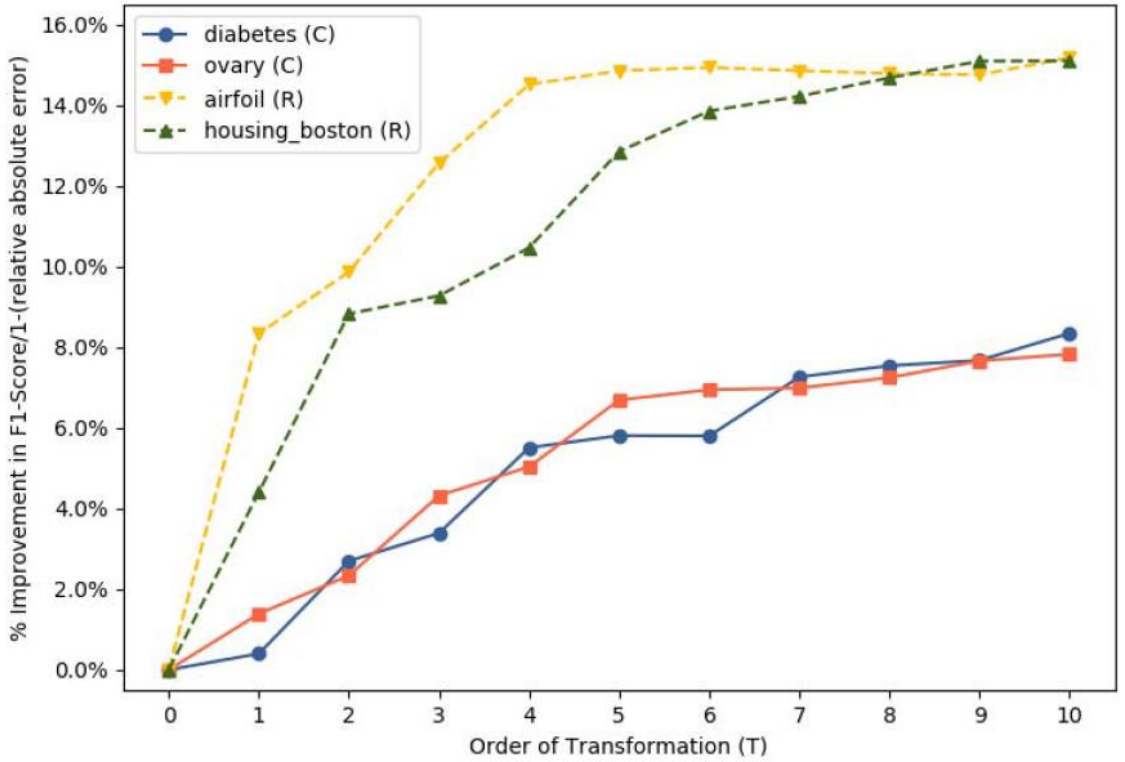


Figure 4. Effect of High-order Transformations

3 SIL Experiments and Results

RQ3: Is NFS robust to hyperparameters?

Table III
COMPARATIVE RESULTS OF NFS WITH DIFFERENT PARAMETER SETTINGS OF λ

Dataset	$\lambda = 0.2$	$\lambda = 0.25$	$\lambda = 0.3$	$\lambda = 0.35$	$\lambda = 0.4$	$\lambda = 0.45$	$\lambda = 0.5$	$\lambda = 0.55$	$\lambda = 0.6$
Airfoil	0.792	0.798	0.796	0.801	0.796	0.797	0.800	0.795	0.799
Housing Boston	0.690	0.690	0.694	0.691	0.686	0.686	0.685	0.686	0.687
AP-omentum-ovary	0.873	0.880	0.873	0.873	0.873	0.874	0.873	0.873	0.873
diabetes	0.785	0.784	0.785	0.786	0.786	0.785	0.784	0.789	0.786

Table IV
COMPARATIVE RESULTS OF NFS WITH DIFFERENT PARAMETER SETTINGS OF m

Dataset	$m = 1$	$m = 2$	$m = 4$	$m = 8$	$m = 16$	$m = 32$	$m = 64$
Airfoil	0.792	0.796	0.792	0.794	0.797	0.796	0.801
Housing Boston	0.678	0.688	0.688	0.687	0.692	0.686	0.693
AP-omentum-ovary	0.866	0.862	0.863	0.877	0.866	0.873	0.873
diabetes	0.776	0.775	0.785	0.777	0.784	0.786	0.784

3 SIL Experiments and Results

Statistics of Effective Transformations

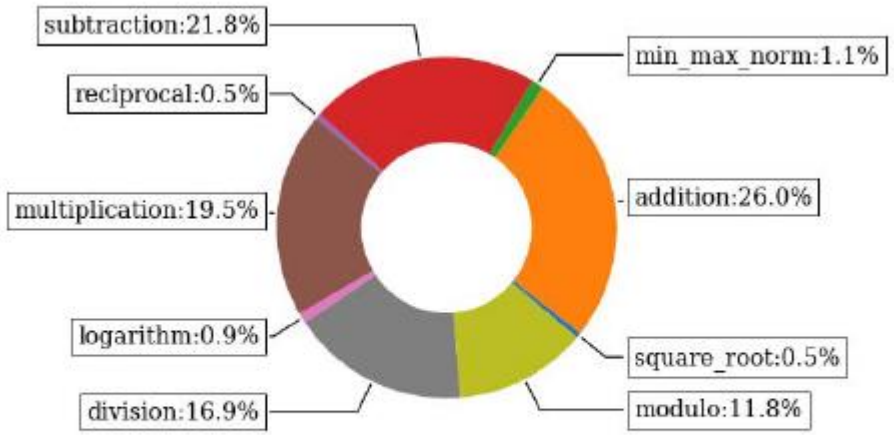


Figure 5. Statistics of Effective Transformations

Q&A