Evolving Keras Architectures for Sensor Data Analysis

Petra Vidnerová Roman Neruda

Institute of Computer Science
The Czech Academy of Sciences

FedCSIS 2017

Outline

- Introduction
  - Deep Neural Networks
  - KERAS library
  - Related Work
- Our Approach
  - Genetic Algorithms
  - Individuals
  - Genetic Operators
- Experiments
  - Sensor Data Set
  - Results
- Summary and future work

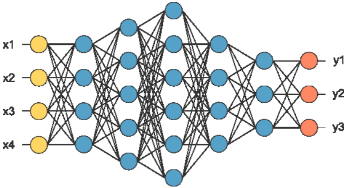
Introduction

Deep Neural Networks

- neural networks with more hidden layers
- convolutional networks - convolutional layers
- our work: feed-forward neural networks, fully connected

Network Architecture

- typically designed by humans
- trial and error method
- our goal: automatic design



KERAS Library

- widely used tool for practical applications of DNNs

```
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))


model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])
```

Related Work

- quite many attemps on architecture optimisation via evolutionary process (NEAT, HyperNEAT, COSyNE)
- architecture optimisation for DNN is very time consuming
- works focus on parts of network design
  - I. Loshchilov and F. Hutter, *CMA-ES for hyperparameter optimization of deep neural networks,* 2016
  - J. Koutník, J. Schmidhuber, and F. Gomez, *Evolving deep unsupervised convolutional networks for vision-based reinforcement learning,* GECCO '14.
- optimising deep learning architectures through evolution
  - R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Fran- con, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, *Evolving deep neural networks,* 2017

Our Approach

Keep the search space as simple as possible.

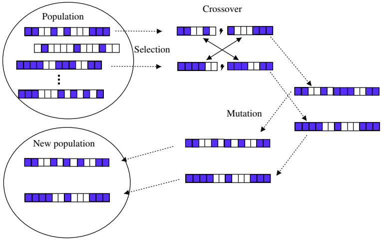
- only architecture is optimized, weights are learned by gradient based technique
- the approach is inpired by and designed for KERAS library
- architecture defined as list of layers, each layer fully connected with next layer (dense layers)
- layer defined by number of neurons, activation function, type of regularization

- future work: add convolutional and max-pooling layers
- metaparameters of learning algorithm (type of algorithm, learning rate, etc.)

Genetic Algorithm for KERAS Architectures

Genetic Algorithms

- robust optimisation technique
- work with population of *individuals* representing feasible solutions
- each individual has assigned a *fitness* value
- population evolves by means of *selection, crossover, and mutation*



Individuals

- individual - deep neural network architecture
- block - network layer

$$I = ([size_1, drop_i, act_1]_1, \ldots, [size_H, drop_H, act_H]_H)$$

H ... number of hidden layers

$size_i$ ... size of layer

$drop_i$ ... dropout rate

act_i ... activation function

- output layer is softmax or linear (classification or regression task)

Crossover

- one-point crossover working on the whole blocks (layers)

Parents:

$$I_{p1} = (B_1^{p1}, B_2^{p1}, \dots, B_k^{p1})$$

$$I_{p2} = (B_1^{p2}, B_2^{p2}, \dots, B_l^{p2}),$$

Offspring:

$$I_{o1} = (B_1^{p1}, \dots, B_{cp1}^{p1}, B_{cp2+1}^{p2}, \dots, B_l^{p2})$$

$$I_{o1} = (B_1^{p2}, \dots, B_{cp2}^{p2}, B_{cp1+1}^{p1}, \dots, B_k^{p1}).$$

Mutation

- random changes to the individual

Roulette wheel selection of:

- mutateLayer - modifies one randomly selected layer
- addLayer - adds one random layer
- delLayer - deletes one random layer

mutateLayer

- change layer size
- change dropout
- change activation
- change all - completely new layer is generated

Fitness and Selection

Fitness Evaluation

- create network defined by individual
- evaluate crossvalidation error on trainset
- KFold crossvalidation
- for each fold train network using gradient based technique

Tournament selection

- k individuals selected at random, the best one selected for repreduction

Sensor Data

Target application - Air Pollution Prediction

- a real-world data set from the application area of sensor networks for air pollution monitoring
- concentration of several gas pollutants
- 8 input values - 5 sensors, temperature, absolute and relative humidity
- 1 predicted value - concentration of CO, NO2, NOx, C6H6, and NMHC

Data Set

- First task - whole time period divided into five intervals, one for training, the rest for testing
- Second task - data for training and testing selected at random

	First experiment		**Second experiment**	
Task	train set	test set	train set	test set
CO	1469	5875	4896	2448
NO2	1479	5914	4929	2464
NOx	1480	5916	4931	2465
C6H6	1799	7192	5994	2997
NMHC	178	709	592	295

Parameter setup

Main GA	N	population size	30
	ng	number of generations	100
	p_{cx}	crossover probability	0.6
	p_{mut}	mutation probability	0.2
Individual	*nlayers*	max number of layers	5
	max_lsize	max layer size	100
	min_lsize	minimum layer size	5
Fitness	k	k-fold crossover	5
Selection	k	tournament of k individuals	3

Activation functions: relu, tanh, sigmoid, hard sigmoid, linear
Learning algorithm: RMSprop

Experiments Results: GAKeras vs. SVR

Testing errors

Task	GAKeras				SVR			
	avg	std	min	max	linear	RBF	Poly.	Sigmoid
CO_part1	**0.209**	0.014	0.188	0.236	0.340	0.280	0.285	1.533
CO_part2	0.801	0.135	0.600	1.048	0.614	**0.412**	0.621	1.753
CO_part3	**0.266**	0.029	0.222	0.309	0.314	0.408	0.377	1.427
CO_part4	**0.404**	0.226	0.186	0.865	1.127	0.692	0.535	1.375
CO_part5	0.246	0.024	0.207	0.286	0.348	0.207	**0.198**	1.568
NOx_part1	2.201	0.131	1.994	2.506	**1.062**	1.447	1.202	2.537
NOx_part2	1.705	0.284	1.239	2.282	2.162	1.838	**1.387**	2.428
NOx_part3	1.238	0.163	0.982	1.533	**0.594**	0.674	0.665	2.705
NOx_part4	1.490	0.173	1.174	1.835	0.864	0.903	**0.778**	2.462
NOx_part5	**0.551**	0.052	0.456	0.642	1.632	0.730	1.446	2.761
NO2_part1	**1.697**	0.266	1.202	2.210	2.464	2.404	2.401	2.636
NO2_part2	**2.009**	0.415	1.326	2.944	2.118	2.250	2.409	2.648
NO2_part3	**0.593**	0.082	0.532	0.815	1.308	1.195	1.213	1.984
NO2_part4	**0.737**	0.023	0.706	0.776	1.978	2.565	1.912	2.531
NO2_part5	1.265	0.158	1.054	1.580	1.0773	1.047	**0.967**	2.129
C6H6_part1	**0.013**	0.005	0.006	0.024	0.300	0.511	0.219	1.398
C6H6_part2	**0.039**	0.015	0.025	0.079	0.378	0.489	0.369	1.478
C6H6_part3	**0.019**	0.011	0.009	0.041	0.520	0.663	0.538	1.317
C6H6_part4	**0.030**	0.015	0.014	0.061	0.217	0.459	0.123	1.279
C6H6_part5	**0.017**	0.015	0.004	0.051	0.215	0.297	0.188	1.526
NMHC_part1	1.719	0.168	1.412	2.000	1.718	1.666	**1.621**	3.861
NMHC_part2	**0.623**	0.164	0.446	1.047	0.934	0.978	0.839	3.651
NMHC_part3	**1.144**	0.181	0.912	1.472	1.580	1.280	1.438	2.830
NMHC_part4	**1.220**	0.206	0.994	1.563	1.720	1.565	1.917	2.715
NMHC_part5	1.222	0.126	1.055	1.447	1.238	**0.944**	1.407	2.960
	16				**2**	**2**	**5**	**0**

Experimental Results: Evolved Architectures

- evolved networks are quite small
- typical network:
  - one hidden layer of about 70 neurons
  - dropout rate 0.3
  - ReLU activation function.

Experiments Results: GAKeras vs. fixed architecture

Testing errors

Task	GAKeras		50-1		30-10-1		30-10-30-1	
	avg	std	avg	std	avg	std	avg	std
CO_part1	**0.209**	0.014	0.230	0.032	0.250	0.023	0.377	0.103
CO_part2	0.801	0.135	0.861	0.136	**0.744**	0.142	0.858	0.173
CO_part3	0.266	0.029	**0.261**	0.040	0.305	0.043	0.302	0.046
CO_part4	**0.404**	0.226	0.621	0.279	0.638	0.213	0.454	0.158
CO_part5	**0.246**	0.024	0.283	0.072	0.270	0.032	0.309	0.032
NOx_part1	2.201	0.131	2.158	0.203	**2.095**	0.131	2.307	0.196
NOx_part2	**1.705**	0.284	1.799	0.313	1.891	0.199	2.083	0.172
NOx_part3	1.238	0.163	1.077	0.125	1.092	0.178	**0.806**	0.185
NOx_part4	1.490	0.173	**1.303**	0.208	1.797	0.461	1.600	0.643
NOx_part5	**0.551**	0.052	0.644	0.075	0.677	0.055	0.778	0.054
NO2_part1	1.697	0.266	1.659	0.250	**1.368**	0.135	1.677	0.233
NO2_part2	2.009	0.415	1.762	0.237	**1.687**	0.202	1.827	0.264
NO2_part3	0.593	0.082	0.682	0.148	**0.576**	0.044	0.603	0.069
NO2_part4	**0.737**	0.023	1.109	0.923	0.757	0.059	0.802	0.076
NO2_part5	1.265	0.158	**0.646**	0.064	0.734	0.107	0.748	0.123
C6H6_part1	0.013	0.005	**0.012**	0.006	0.081	0.030	0.190	0.060
C6H6_part2	**0.039**	0.015	0.039	0.012	0.101	0.015	0.211	0.071
C6H6_part3	**0.019**	0.011	0.024	0.007	0.091	0.047	0.115	0.031
C6H6_part4	0.030	0.015	**0.026**	0.010	0.051	0.026	0.096	0.020
C6H6_part5	**0.017**	0.015	0.025	0.008	0.113	0.025	0.176	0.058
NMHC_part1	**1.719**	0.168	1.738	0.144	1.889	0.119	2.378	0.208
NMHC_part2	0.623	0.164	**0.553**	0.045	0.650	0.078	0.799	0.096
NMHC_part3	1.144	0.181	1.128	0.089	0.901	0.124	**0.789**	0.184
NMHC_part4	1.220	0.206	1.116	0.119	0.918	0.119	**0.751**	0.096
NMHC_part5	1.222	0.126	0.970	0.094	0.889	0.085	**0.856**	0.074
	10		**6**		**5**		**4**	

Experimental Results: Second Task

Testing errors

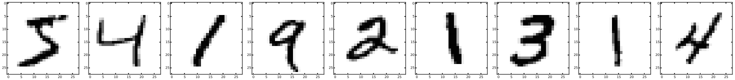
Task	GAKeras		SVR			
	avg	std	linear	RBF	Poly.	Sigmoid
CO	**0.120**	0.004	0.200	0.152	0.157	1.511
NOx	0.295	0.021	0.328	**0.211**	0.255	1.989
NO2	**0.267**	0.009	0.494	0.368	0.406	2.046
C6H6	**0.002**	0.001	0.218	0.110	0.194	1.325
NMHC	**0.266**	0.080	0.688	0.383	0.513	3.215

- evolved networks several layers, dominating activation function ReLU

MNIST classification task

Data Set

- well known data set, classification of hand written digits
- 28×28 pixels
- 60000 for training, 10000 for testing



Results

model	avg	std	min	max
baseline	98.34	0.13	98.18	98.55
GAKeras	98.64	0.05	98.55	98.73

Conlusion and Future Work

- proposed GA for DNN architecture design
- demonstrated the algorithm on experiments
- works for feed-forward DNN with dense layers

Future Work

- our goal - evolve also convolutional networks
- evolve also other parameters of learning
- evolution strategies
- speed up the evolution - asynchronous evolution, surrogate modeling

Thank you!   Questions?