MATHÉMATIQUES
VISION
APPRENTISSAGE

université
PARIS-SACLAY

Automated Deep Learning

LIU Zhengying
Laboratoire en Recherche Informatique (LRI)
U. Paris-Sud / Inria / U. Paris Saclay

9 Mar 2020

Contents

- AutoML: an intro
- AutoML methods
  with application to Deep Learning
- AutoDL challenges

AutoML: an intro

Past — Hard-coding

Present — Machine Learning

Future — AutoML

Hard-coding

Hard-coded algorithm:
"if..else" rules, HOG, SIFT, etc

$$X \xrightarrow{\alpha_\theta} Y$$

 $\mapsto$ "dog"

 $\mapsto$ "cat"

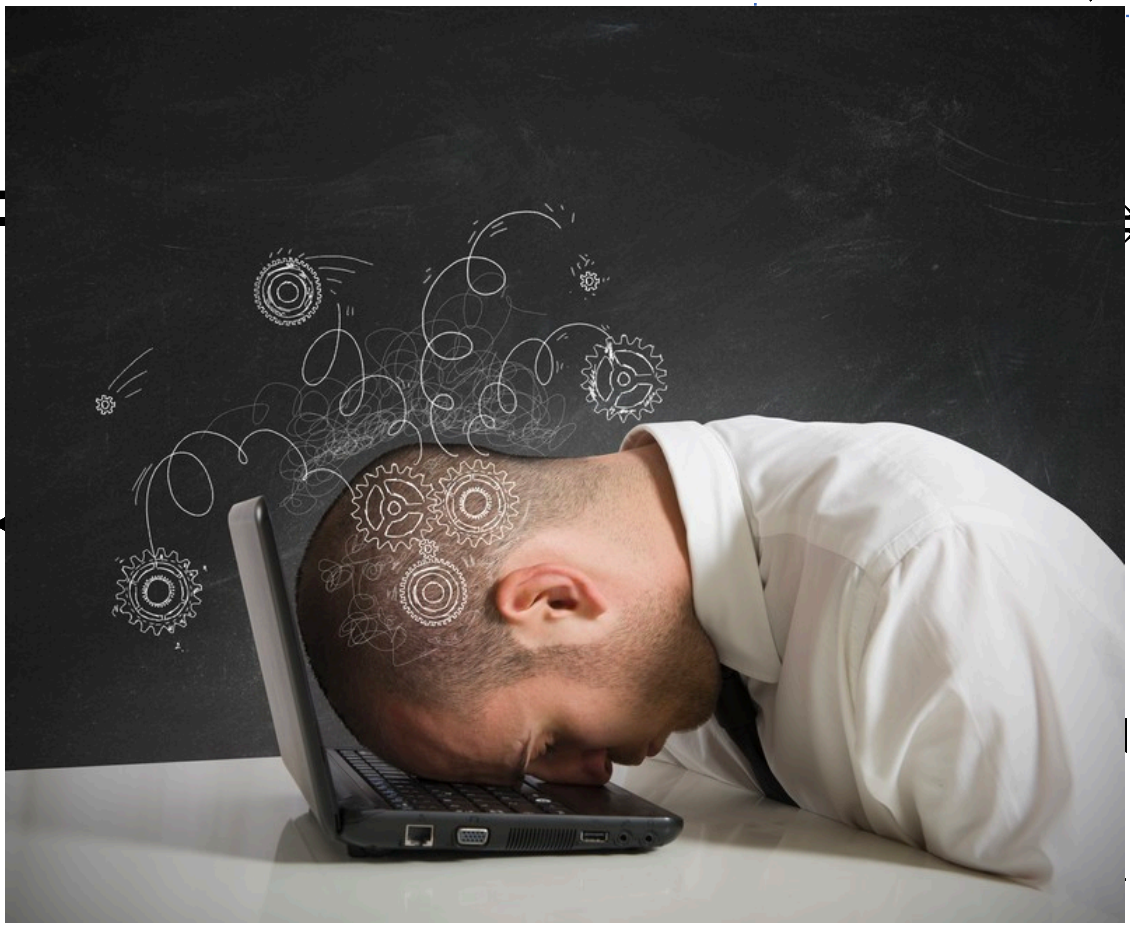
encoded by: parameters $\theta \in \Theta$

hand-crafted by <u>engineers</u>

Machine Learning

Machine Learning algorithm:
Decision Tree, CNN, SVM, etc

$D =$



$P(\alpha_\theta)$

performance
(e.g. accuracy)

Dogs vs Cats
dataset

=

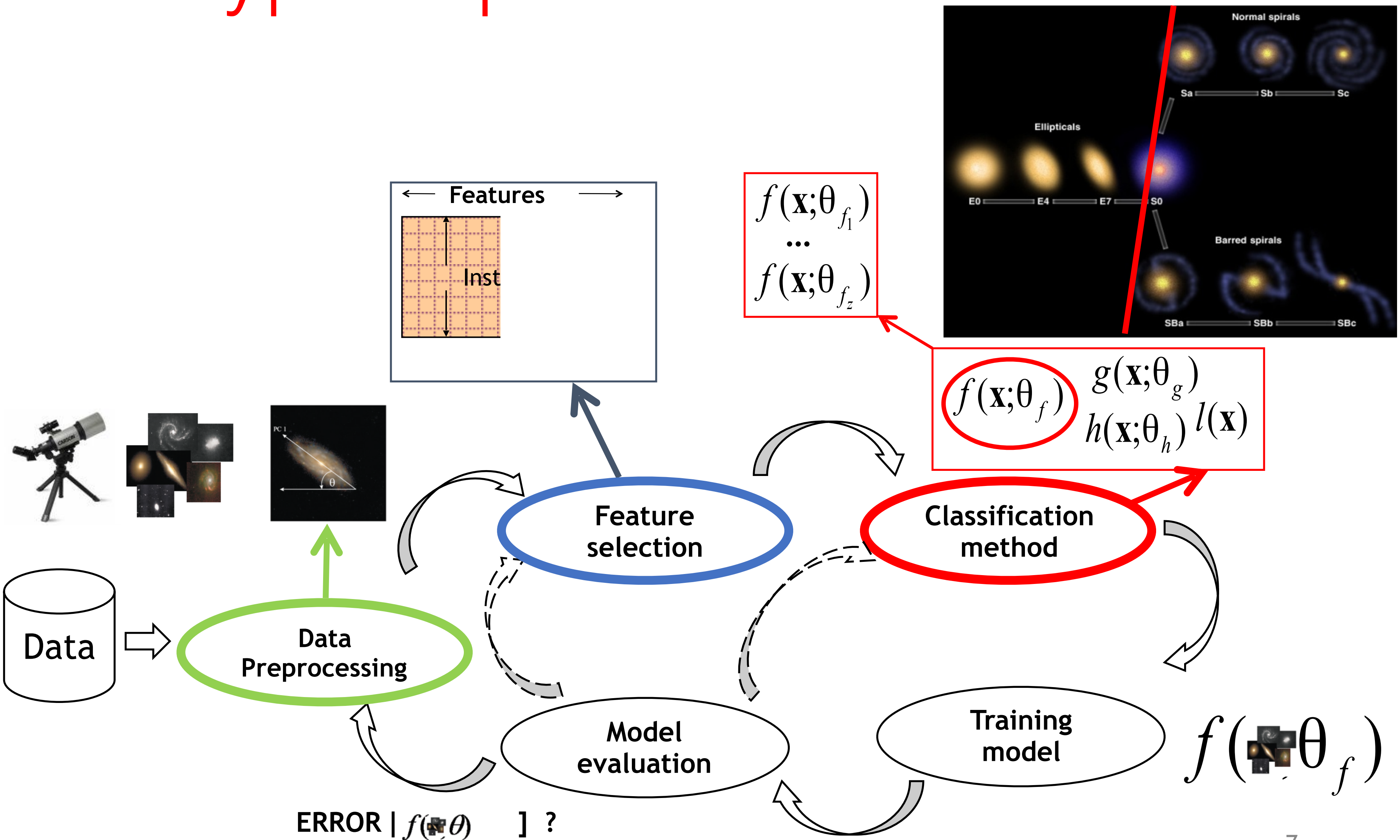
N (for another A)

nother A)

encoded by: hyperparameters $\lambda \in \Lambda$

hand-crafted by ML experts

The typical process…



Machine Learning

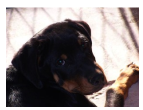
Machine Learning algorithm:
Decision Tree, CNN, SVM, etc

$$D = \{X_i, Y_i\} \xrightarrow{\beta_\lambda} \alpha_\theta \overset{D_{\text{va}}}{\Longrightarrow} P(\alpha_\theta)$$

(or $p_\theta(y \mid x)$)

performance
(e.g. accuracy)

Dogs vs Cats dataset = {

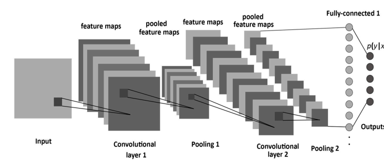


"dog"

"cat"

...

} $\mapsto$



trained CNN

CIFAR-10 dataset $\mapsto$ another trained CNN (for another A)

Iris dataset $\mapsto$ trained SVM (for another A)

encoded by: hyperparameters $\lambda \in \Lambda$

hand-crafted by <u>ML experts</u>

AutoML
(Hyperparameter Optimization)

HPO algorithm:
SMAC (Auto-sklearn), Neural Architecture Search, MCTS, etc

$$D = \{X_i, Y_i\} \xrightarrow{\quad \pi \quad} \beta_\lambda \overset{D_{\text{tr}}}{\Longrightarrow} \alpha_\theta \overset{D_{\text{va}}}{\Longrightarrow} P$$

$(D = D_{\text{tr}} \amalg D_{\text{va}})$

{

"dog"

"cat"

...

} $\mapsto$ untrained CNN $\mapsto$ trained CNN

zero hyperparameters

no hand-crafting at all!

actually defines a beta: $\beta_\pi(D) := \pi(D)(D)$

can be learned too $\Longrightarrow$ meta-learning

AutoML
(Meta-learning)

Meta-dataset

Meta-learning algorithm:
Auto-sklearn, SVM applied to meta-features, etc

$$\mathfrak{D} = \{D_j, \beta_j, P_j\} \xmapsto{\gamma} \pi \quad \text{or} \quad \beta_\pi$$

can be considered as reward/fitness

$$\left\{ \begin{array}{c} \text{Dogs vs Cats}, CNN_{\lambda_1}, 0.67 \\ \text{Iris}, DT_{\lambda_2}, 0.78 \\ \text{CIFAR-10}, SVM_{\lambda_3}, 0.55 \\ \text{MNIST}, SVM_{\lambda_4}, 0.88 \\ \text{CIFAR-10}, CNN_{\lambda_5}, 0.80 \\ \ldots \end{array} \right\} \mapsto \text{Learned HPO algorithm}$$

can be combined with Hyperparameter Optimization

we can also have $\mathfrak{D} = \{D_j, \beta_j, \alpha_j, P_j\}$

with trained $\alpha_j \implies$ transfer learning

Overview

	Schema	Problem
Hard-coding	$X \xmapsto{\alpha} Y$	$\max_{\alpha} P(\alpha; D_{te})$
Machine Learning	$D_{tr} \xmapsto{\beta} \alpha$	$\max_{\beta} P(\hat{\alpha}; D_{te})$ where $\hat{\alpha} = \beta(D_{tr})$
AutoML	$\mathfrak{D}_{tr} \xmapsto{\gamma} \beta$	$\max_{\gamma} \sum_{D_{tr}, D_{te} \in \mathfrak{D}_{te}} P(\hat{\hat{\alpha}}; D_{te})$ where $\hat{\hat{\alpha}} = \hat{\beta}(D_{tr})$ and $\hat{\beta} = \gamma(\mathfrak{D}_{tr})$

Test score: $P(\alpha; D_{te}) = \frac{1}{|D_{te}|} \sum_{X, Y \in D_{te}} S(\alpha(X), Y))$

unknown true test labels at training time

use estimation: cross-validation, hold-out validation, etc

The AutoML problem: definition

$$\max_{\gamma} \sum_{\substack{D_{tr}, D_{te} \\ \in \mathfrak{D}_{te}}} P(\hat{\hat{\alpha}}; D_{te})$$ where $\hat{\hat{\alpha}} = \hat{\beta}(D_{tr})$ and $\hat{\beta} = \gamma(\mathfrak{D}_{tr})$

supervised learning

reinforcement learning

learning to learn ⇐ two layers of learning

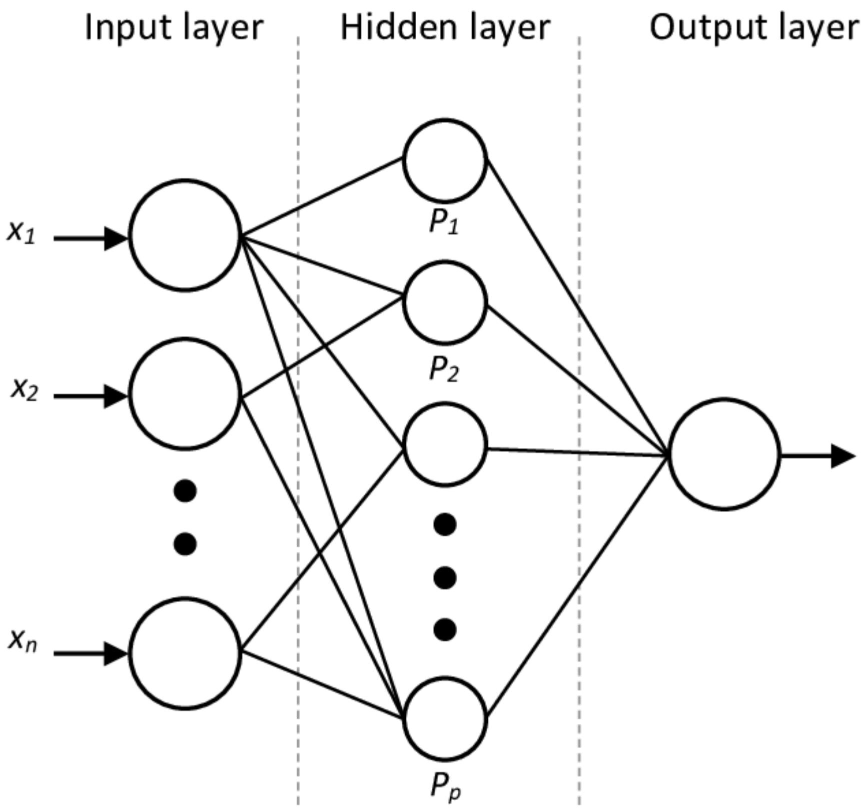
$P(\hat{\hat{\alpha}}; D_{te})$ may involve **time** ⇒ computational efficiency: should be not only correct but also fast

initially we may have $\mathfrak{D}_{tr} = \varnothing$ ⇒ no prior experience BUT can be generated

$(D_{tr}, \beta_1, \alpha_1, P_1)\ , (D_{tr}, \beta_2, \alpha_2, P_2)\ , (D_{tr}, \beta_3, \alpha_3, P_3)\ , \ldots$

Exercise: a toy example

X



$Y = \alpha(X) \in \{0,1\}$

neural network with one hidden layer

θ? λ? hard-coding approach? ML? AutoML?

AutoML: what's exciting?

- 100% autonomous
- Beat "no free lunch"
- Any time
- Any resource

→ AI for everyone

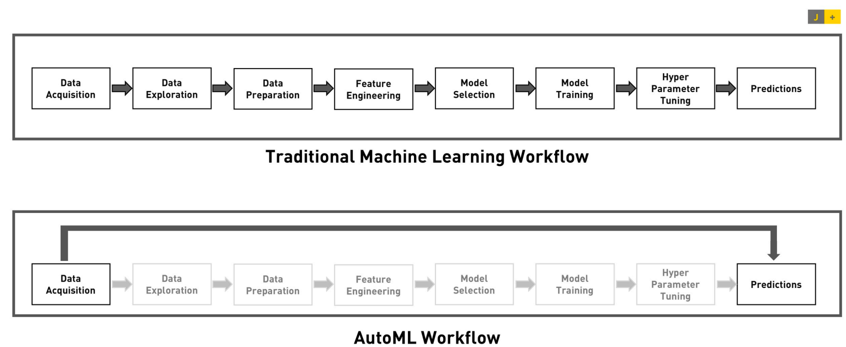
AutoML: already a hot topic

Google's AutoML

AutoDL

AutoML.org

AutoML



AUTO KERAS

Auto-Sklearn

AutoML methods

with application to Deep Learning

We'll focus on the simplest case

$\mathfrak{D}_{tr} = \varnothing$ (initially) and $\mathfrak{D}_{te} = \{(D_{tr}, D_{te})\}$ (single dataset)

$\Rightarrow$ Hyperparameter Optimization

$\Rightarrow$ single fixed training dataset: D_{tr}

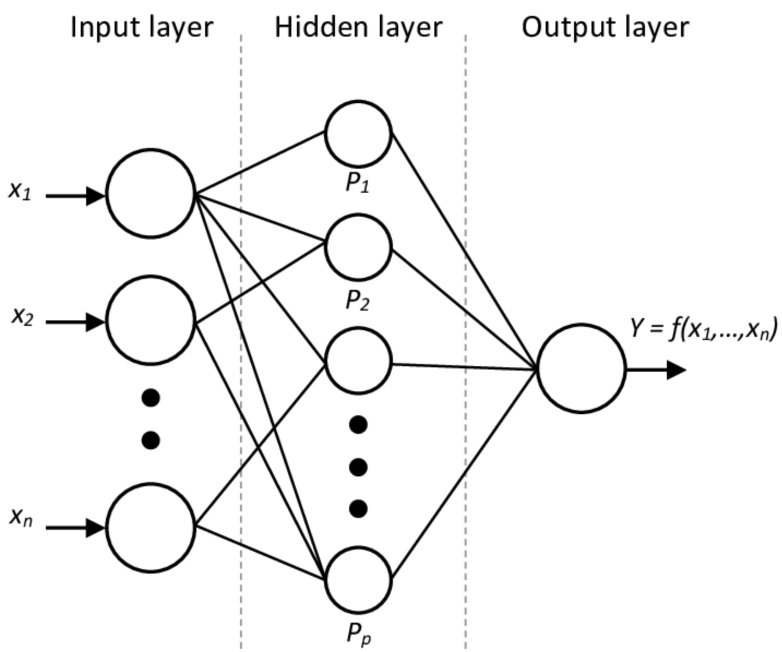
$\Rightarrow$ we only need to focus on $\beta_\lambda, \lambda \in \Lambda$

Reminder:

$$\boxed{\max_{\gamma} \sum_{\substack{D_{tr}, D_{te} \\ \in \mathfrak{D}_{te}}} P(\hat{\hat{\alpha}}; D_{te}) \qquad \text{where } \hat{\hat{\alpha}} = \hat{\beta}(D_{tr}) \text{ and } \hat{\beta} = \gamma(\mathfrak{D}_{tr})}$$

Search Space

How do we describe (encode) a learning algorithm?



in natural language:
"a feed-forward neural network with one hidden layer of p=10 neurons, using ReLU as activation and Adam as optimizer, with learning rate lr=0.001, ..."

formally:

$$\beta_\lambda, \lambda \in \Lambda \ ??$$

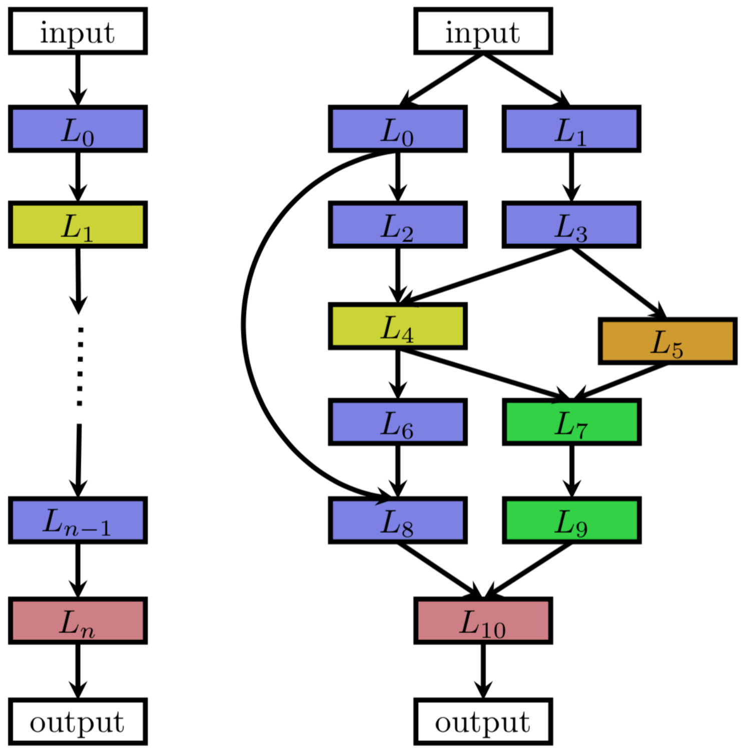
Search Space (for DL)

$\beta_\lambda, \lambda \in \Lambda$: architecture, optimizer, regularization, etc

chain-structured (feed-forward)

$A = L_n \circ L_{n-1} \circ \ldots \circ L_0$

$L_i^{in} = L_{i-1}^{out}$



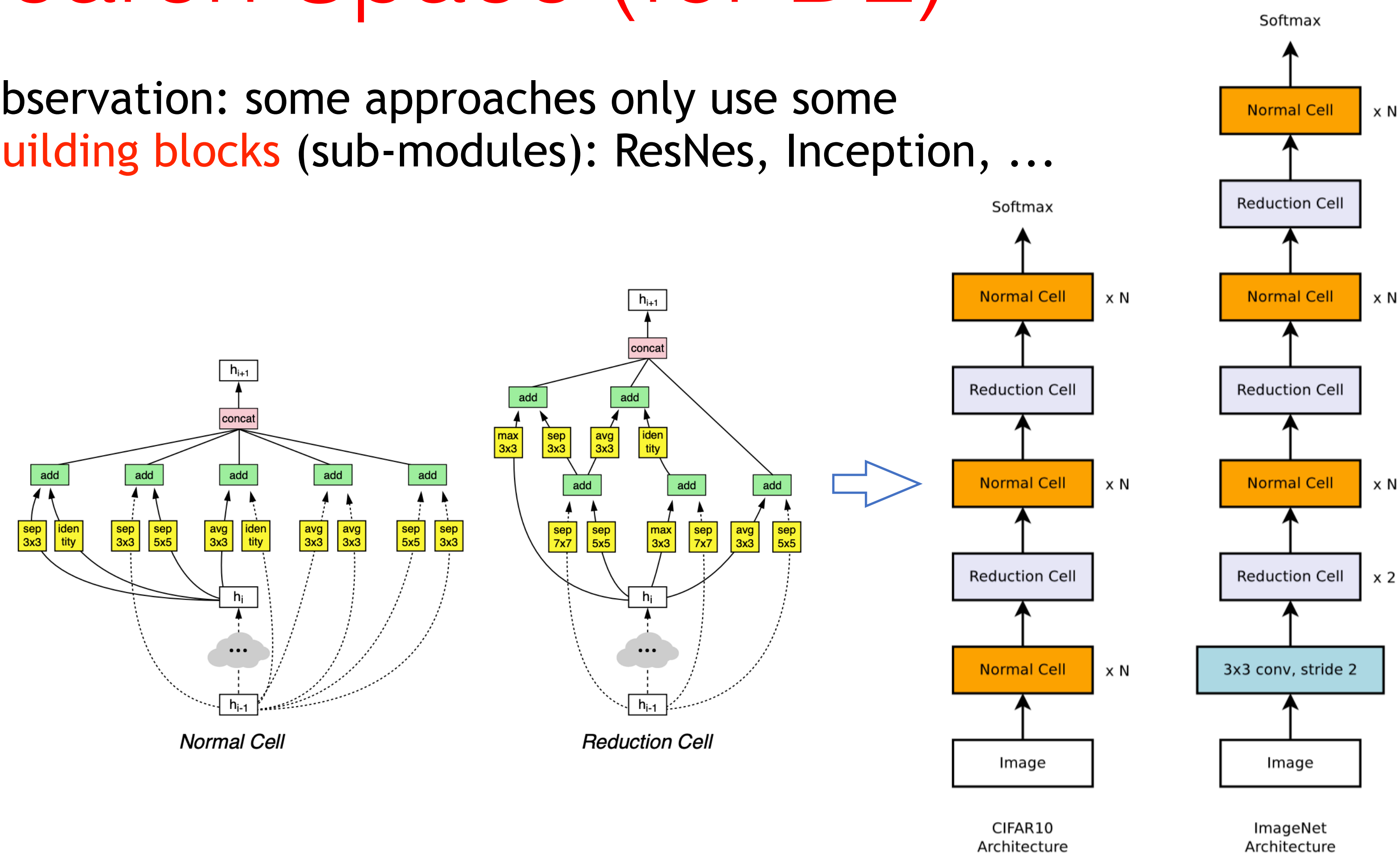
multi-branch

$L_i^{in} = g_i(L_{i-1}^{out}, \ldots, L_0^{out})$

Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Search Space (for DL)

observation: some approaches only use some building blocks (sub-modules): ResNes, Inception, ...



"NASNet search space" only uses two building blocks

Zoph B, Vasudevan V, Shlens J, Le QV. Learning Transferable Architectures for Scalable Image Recognition. *CVPR2018*

Hyperparameter Optimization: a reformulation

an HPO algorithm aims to solve: $\max_{\lambda \in \Lambda} P(\hat{\alpha}; D_{te})$ where $\hat{\alpha} = \beta_\lambda(D_{tr})$

unknown test score: $P(\hat{\alpha}; D_{te})$ $\Rightarrow$ use an estimation (e.g. CV): $\hat{P}(\lambda)$

so usually the problem becomes

$$\boxed{\max_{\lambda \in \Lambda} \hat{P}(\lambda)}$$

black-box optimization

expensive to compute

$\Rightarrow$ surrogate model (not discussed)

where

$$\hat{P} : \Lambda \to \mathbb{R}$$
$$\lambda \mapsto s = \hat{P}(\lambda) \approx P(\beta_\lambda(D_{tr}), D_{te})$$

is an estimation of the test score

Remark: some approaches optimize λ and θ at the same time $\Rightarrow$ bi-level optimization (to be discussed later with DARTS)

Search Strategy

- Heuristic search
  - Grid Search
  - Random Search
  - Evolutionary Algorithms
- Bayesian Optimization
- Reinforcement Learning methods
- Differentiable methods

Grid Search (exhaustive search)

$$\Lambda = \Lambda_1 \times \Lambda_2 \text{ with } \Lambda_1 = \{1,2,3,4\} \text{ and } \Lambda_2 = \{0.001, 0.001, 0.1, 1\}$$

neurons in hidden layer

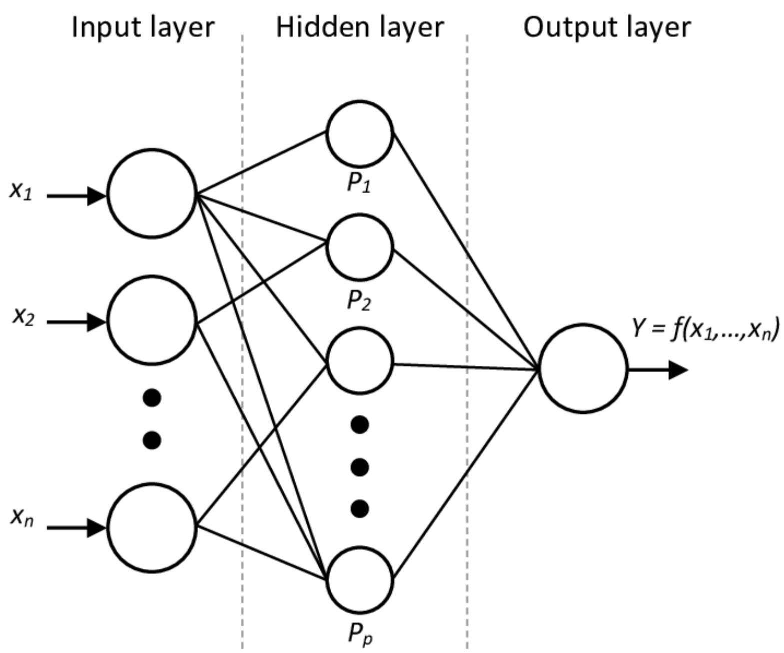
learning rate

try every possible combination in

$$\Lambda = \Lambda_1 \times \Lambda_2$$

evaluate it and return argmax in the end

curse of dimensionality!



Random Search

$\Lambda = \Lambda_1 \times \Lambda_2$ with $\Lambda_1 = \{1,2,3,4\}$ and $\Lambda_2 = \{0.001, 0.001, 0.1, 1\}$

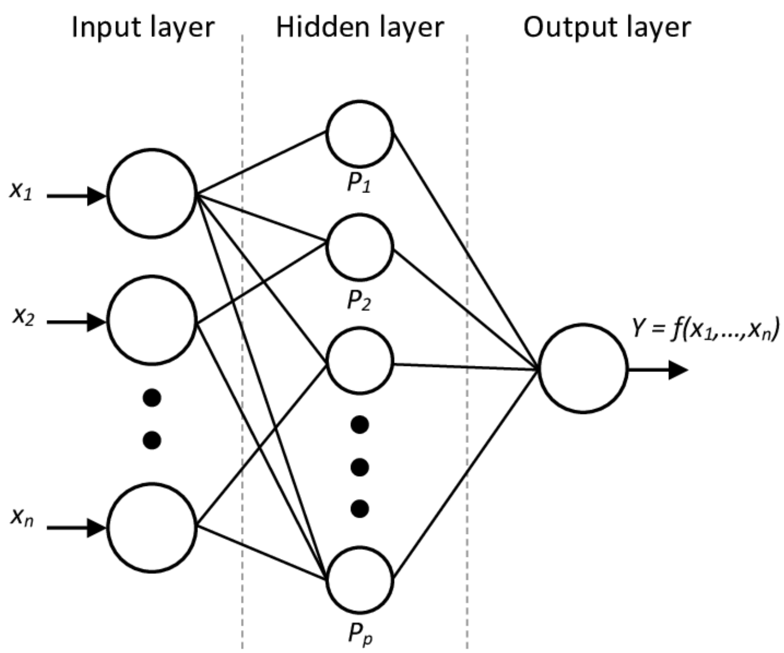
neurons in hidden layer

learning rate

Randomly sample certain number of combinations in

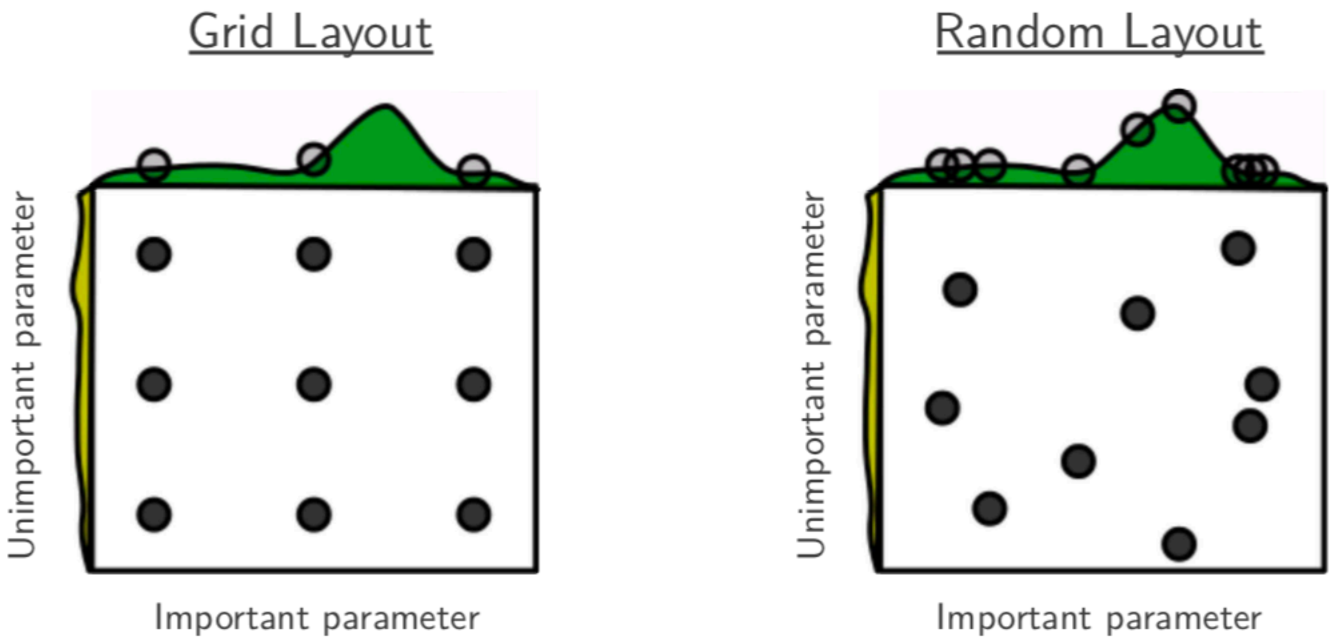
$$\Lambda = \Lambda_1 \times \Lambda_2$$

evaluate it and return argmax in the end



Grid Search and Random Search

two model-free black-box optimization methods



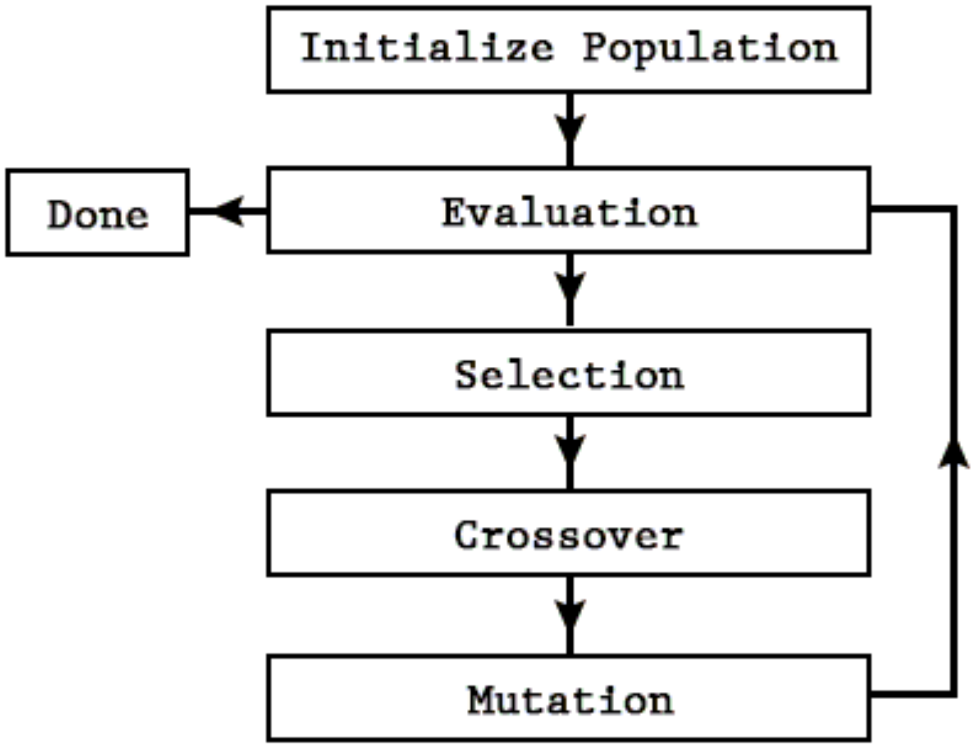
RS tends to perform better than GS when some HP are more important than others

Random Search provides already a strong HPO baseline (surprisingly...?)

Bergstra J, Bengio Y. **Random Search for Hyper-Parameter Optimization**. *JMLR*. 2012

Evolutionary Algorithms

Population-based derivative-free optimization methods



Optimize w.r.t a population (a set of points) or a distribution instead of one single point

Often encode an individual by "chromosome"

Explore new points by mutation or crossover

Select individuals by fitness

Just some vocabulary...but the idea is simple

Easy to parallelize

similar to: genetic algorithms, evolutionary strategies, particle swarm optimization

Evolutionary Algorithm: an example

Real E, Moore S, Selle A, et al. **Large-Scale Evolution of Image Classifiers**. *ICML2017*

1000 individuals

fitness: accuracy on validation dataset

pair-wise competition
(select two individuals and kill the weaker one)

the winner gets to reproduce and mutate

massively-parallel
(due to huge computation cost)

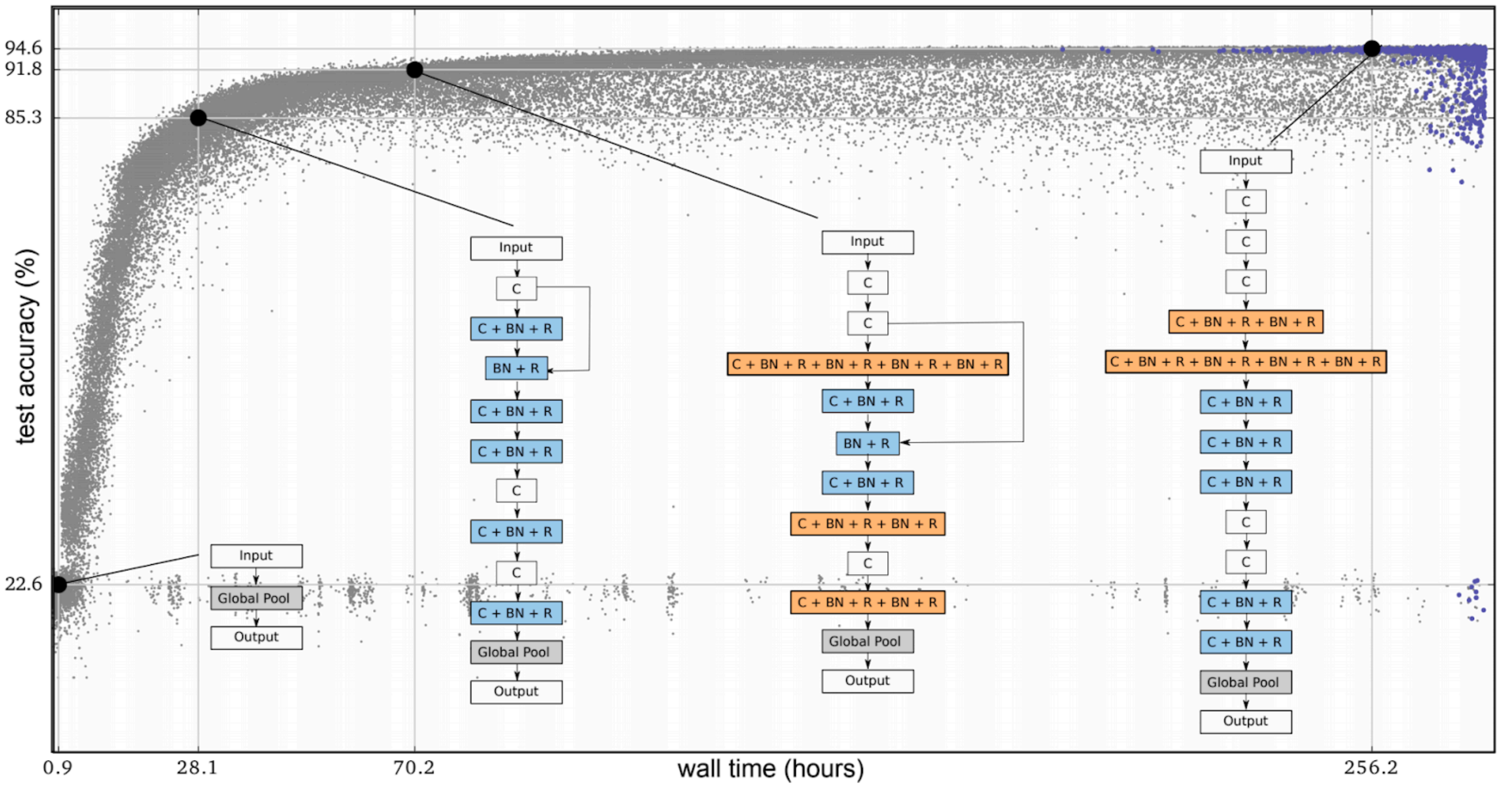
chromosome (DNA): tensor graph

begins from single layer individuals

possible mutations:

- ALTER-LEARNING-RATE
- IDENTITY
- RESET-WEIGHTS
- INSERT-CONVOLUTION
- REMOVE-CONVOLUTION.
- ALTER-STRIDE
- ALTER-NUMBER-OF-CHANNELS
- FILTER-SIZE
- INSERT-ONE-TO-ONE
- ADD-SKIP
- REMOVE-SKIP

Evolutionary Algorithm: an example



Real E, Moore S, Selle A, et al. Large-Scale Evolution of Image Classifiers. *ICML2017*

Bayesian Optimization

$$\max_{\lambda \in \Lambda} \hat{P}(\lambda) \qquad \text{with } \hat{P} : \Lambda \to \mathbb{R}, \quad \lambda \mapsto s$$

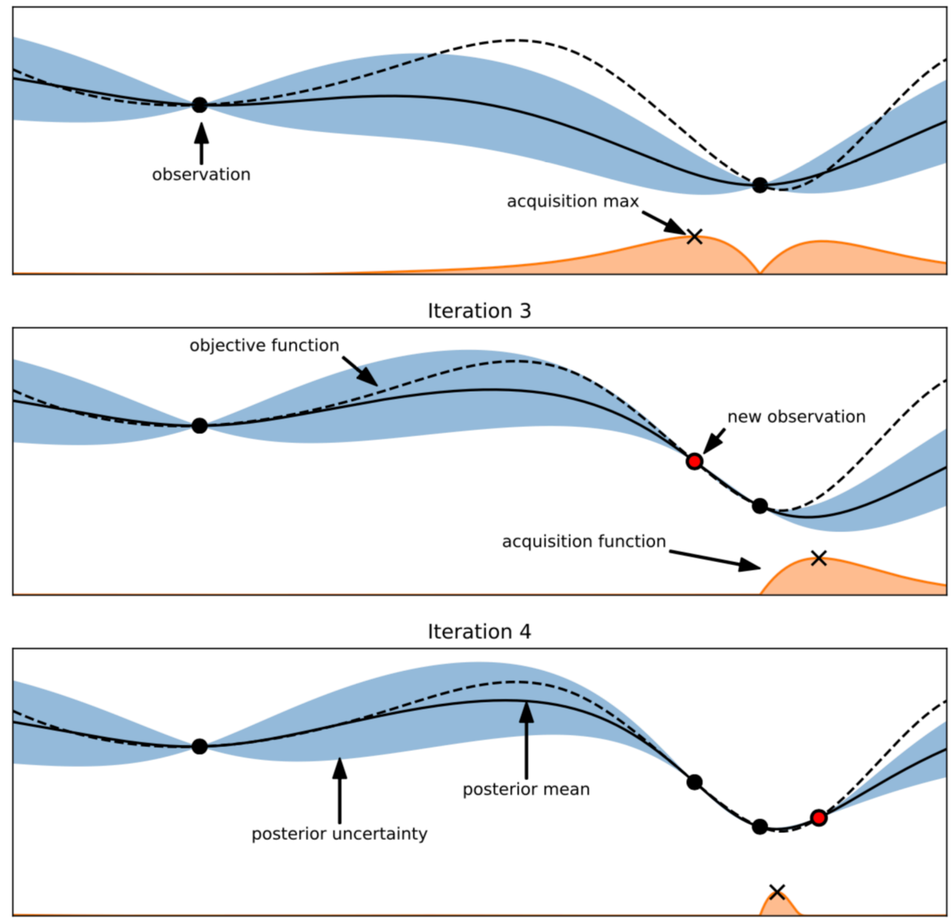
Original idea:

λ and $s = \hat{P}(\lambda)$ follow prior distributions $p(\lambda), p(s\,|\,\lambda)$

we choose next point to evaluate by maximizing an **acquisition function** (active learning-like)

we gain more information and update $p(\lambda)$ and $p(s\,|\,\lambda)$ (or $p(s, \lambda)$)

repeat until convergence



Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Bayesian Optimization (cont'd)

$$\max_{\lambda \in \Lambda} \hat{P}(\lambda) \qquad \text{with } \hat{P} : \Lambda \to \mathbb{R}, \quad \lambda \mapsto s$$

usual acquisition function:
Expected Improvement (EI)

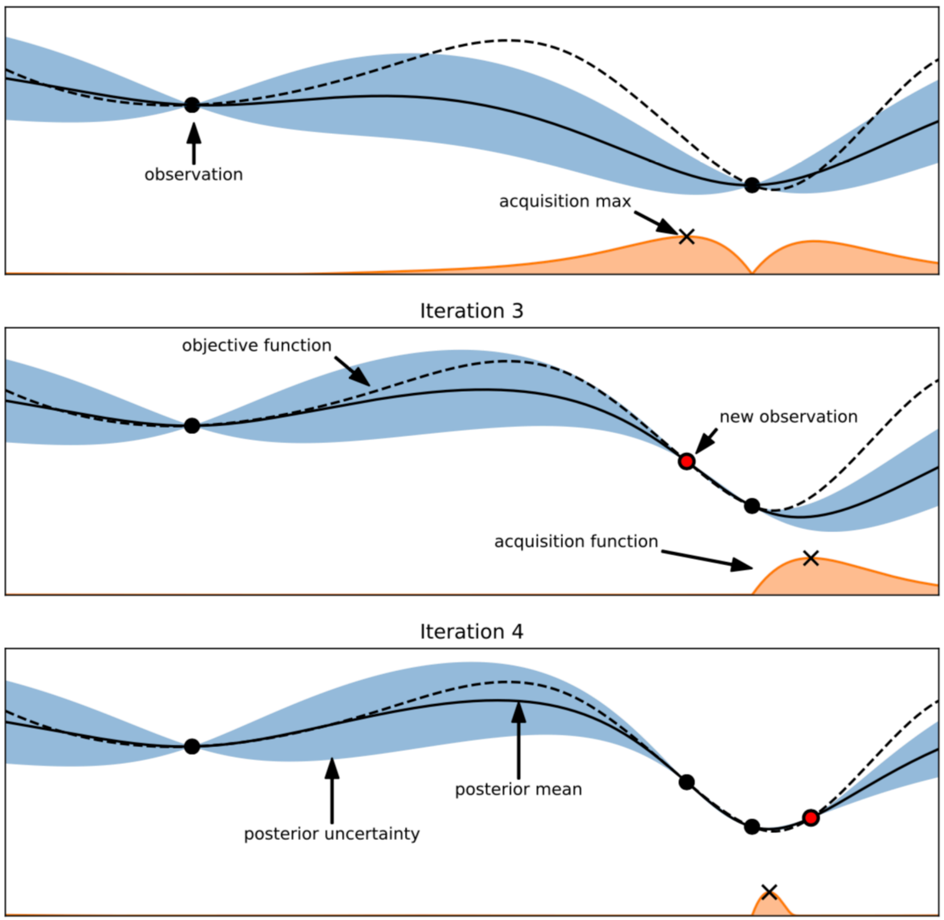
$$a_{EI}(\lambda \mid D_n) = \mathbb{E}[\max(\hat{P}(\lambda) - s_{\max}, 0)]$$

usual prior model:
Gaussian Process (GP)

but state-of-the-art tends to use tree-based classifier such as **Random Forest** to model

$\hat{P}(\lambda)$ (or $p(s \mid \lambda)$)

(thus not so Bayesian anymore...), see Auto-sklearn



Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Bayesian Optimization: an example

Bergstra JS, Bardenet R, Bengio Y, Kégl B. **Algorithms for Hyper-Parameter Optimization**. *NIPS2011*

Tree Parzen Estimator (TPE) -> Hyperopt

model $p(\lambda \,|\, s < \alpha)$ and $p(\lambda \,|\, s > \alpha)$ instead of $p(s \,|\, \lambda)$

use notation $f : x \mapsto y$ instead of $\hat{P} : \lambda \mapsto s$

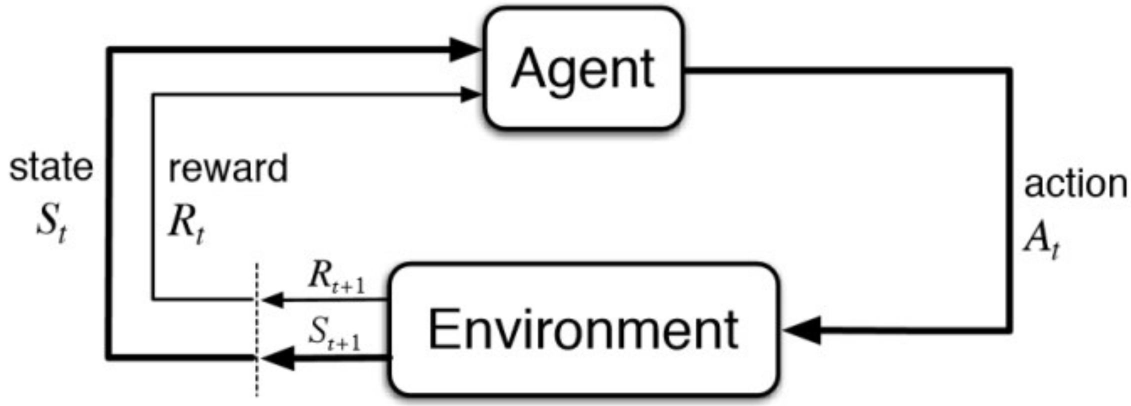
$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^*, \end{cases}$$

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy$$

$$EI_{y^*}(x) = \frac{\gamma y^* \ell(x) - \ell(x) \int_{-\infty}^{y^*} p(y) dy}{\gamma \ell(x) + (1-\gamma) g(x)} \propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma)\right)^{-1}$$

Reinforcement Learning

A reminder:



State space: S

Action space: A

Transition model: $\mathscr{P}^a_{ss'} = p(s'|s,a) : S \times A \times S \rightarrow [0,1]$

Reward: $\mathscr{R}^a_{ss'} : S \times A \times S \rightarrow \mathbb{R}$

Goal: Learn a policy: $\pi(s,a) = p(a|s) : S \times A \rightarrow [0,1]$

that maximizes the (discounted) expected return

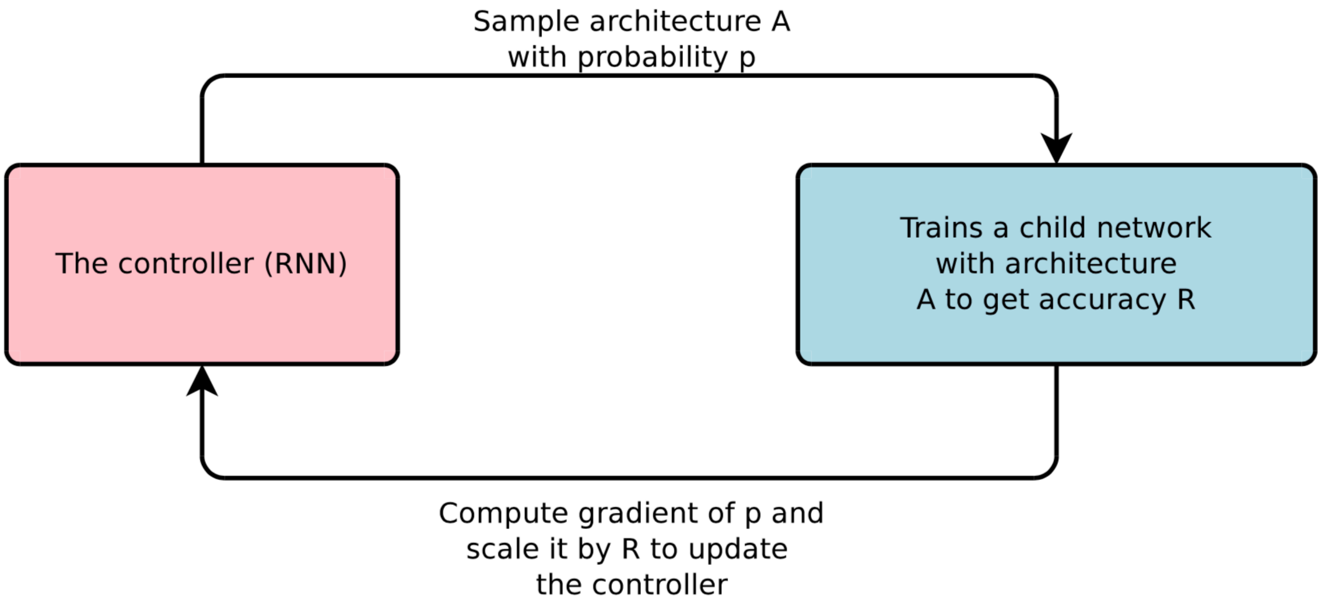
$$\mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \right]$$

with $T \in [0, +\infty], \gamma \in [0,1]$ and $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$ the agent's trajectory

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Reinforcement Learning: an example

Zoph B, Le QV. **Neural Architecture Search with Reinforcement Learning**. ICLR 2017



Objective: $J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$

REINCFORCE rule: $\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\big[\nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R\big]$

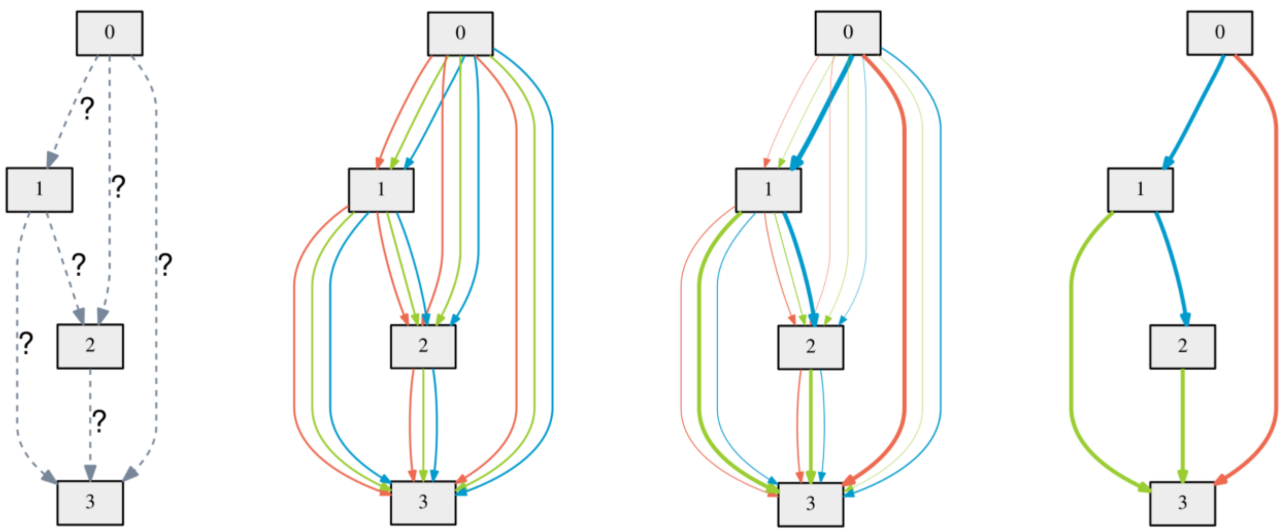
an estimation: $\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R_k$

Differentiable Methods

Liu H, Simonyan K, Yang Y. **DARTS: Differentiable Architecture Search.** ICLR2019

Idea: relaxation of "hard" choice of operations (convolution, max-pooling, zero) to "soft" choice (linear combination of these operations)

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$



(a) (b) (c) (d)

Differentiable Methods

Liu H, Simonyan K, Yang Y. **DARTS: Differentiable Architecture Search.** ICLR2019

algorithm parametrized by
architecture: α
weights: w

→

Bi-level optimization:
1. update W on training set
2. update α on validation set

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha)$$

Inner step is very expensive => approximate full training by on step:

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

Results:
Close to state-of-the-art (SotA) on image (CIFAR10, ImageNet) but much **faster**
SotA on language modeling (Penn Treebank, WikiText-2)

Can be considered as having one single big architecture but with different training method => **Lottery Ticket Hypothesis?**

Frankle J, Carbin M. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.* ICLR2019

Summary

Method	Type	How to take next action	Update/Learn
Grid Search	model-free	loop over all choices (Cartesian product)	take max
Random Search	model-free	totally random	take max
Bayesian Optimization	sequential-based	maximizes acquisition function	update surrogate model
Evolutionary Algorithms	population-based	each individual randomly mutates	eliminate the weakest (with least fitness)
Reinforcement Learning	mixed/can be very general	according to learned policy	policy gradient method
Differentiable Methods	gradient-based	follow (negative) gradient	gradient descent

There is learning in EVERY method

Is there exploration-exploitation trade-off in each method?

How do we do benchmarking and fairly evaluate these methods?

⇨ AutoDL challenge!!!

Some other AutoML methods

Transfer Learning

Meta-learning

Ensemble methods
(competition winners)

embedded methods*: bi-level optimization methods
(related to transfer learning)

filter methods*: narrowing down the model space, without training the learning machine
(related to meta-learning)

* Guyon I, Bennett K, Cawley G, et al. Design of the 2015 ChaLearn AutoML challenge. *IJCNN 2015*

AutoDL challenges

Competition track
@ NeurIPS 2019

AutoDL challenges

Zhengying Liu

Inria / LRI, France - `zhengying.liu@inria.fr`

14 Dec 2019 - Vancouver, Canada

Thanks

Olivier Bousquet (Google, Switzerland)

André Elisseeff (Google, Switzerland)

Isabelle Guyon (U. Paris-Saclay; UPSud/INRIA, France; ChaLearn, USA)

Hugo Jair Escalante (IANOE, Mexico; ChaLearn, USA)

Sergio Escalera (University of Barcelona; ChaLearn, USA)

Wei-Wei Tu (4paradigm, China)

And many more...

Stephane Ayache (AMU, France), Hubert Jacob Banville (INRIA, France), Mahsa Behzadi (Google, Switzerland), Kristin Bennett (RPI, New York, USA), Sergio Escalera (U. Barcelona, Spain and ChaLearn, USA), Gavin Cawley (U. East Anglia, UK), Baiyu Chen (UC Berkeley, USA), Albert Clapes i Sintes (U. Barcelona, Spain), Alexandre Gramfort (U. Paris-Saclay; INRIA, France), Yi-Qi Hu (4paradigm, China), Julio Jacques Jr. (U. Barcelona, Spain), Meysam Madani (U. Barcelona, Spain), Tatiana Merkulova (Google, Switzerland), Adrien Pavao (U. Paris-Saclay; INRIA, France and ChaLearn, USA), Shangeth Rajaa (BITS Pilani, India), Herilalaina Rakotoarison (U. Paris-Saclay, INRIA, France), Mehreen Saeed (FAST Nat. U. Lahore, Pakistan), Marc Schoenauer (U. Paris-Saclay, INRIA, France), Michele Sebag (U. Paris-Saclay; CNRS, France), Danny Silver (Acadia University, Canada), Lisheng Sun (U. Paris-Saclay; UPSud, France), Sebastien Treger (La Pallaisse, France), Fengfu Li (4paradigm, China), Lichuan Xiang (4paradigm, China), Jun Wan (Chinese Academy of Sciences, China), Mengshuo Wang (4paradigm, China), Jingsong Wang (4paradigm, China), Ju Xu (4paradigm, China), Zhen Xu (Ecole Polytechnique and U. Paris-Saclay; INRIA, France), Eric Carmichael (CKCollab, USA), Tyler Thomas (CKCollab, USA)

CHA LEARN

Google

4Paradigm 第四范式

Microsoft

INAOE

CodaLab

NVIDIA

Inria

MACHINE LEARNING POWER TO THE DATA

LAMDA Learning And Mining from DatA

quinten

UNIVERSITÉ PARIS SUD

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

ACADIA UNIVERSITY

université PARIS-SACLAY

Aix*Marseille université

NIPS

ICML

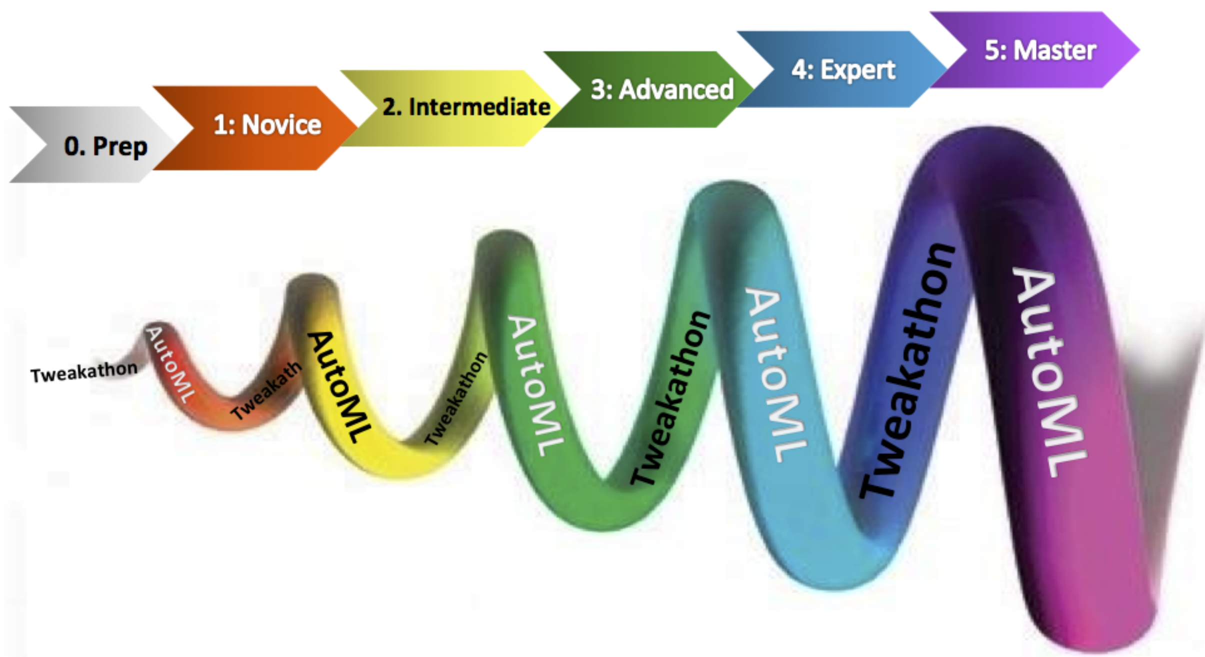
ECMLPKDD Würzburg 2019

IJCNN

PRICAI

AutoML challenges

the origin - 2015-2018



Auto-Sklearn

AutoDL challenges

Towards fully automated multi-label classification for

image, video, text, speech, tabular

AutoML challenges

Auto Computer Vision

Auto Computer Vision 2

Auto NLP

AUTO SPEECH

AutoWSL

Technical Support

CodaLab

docker

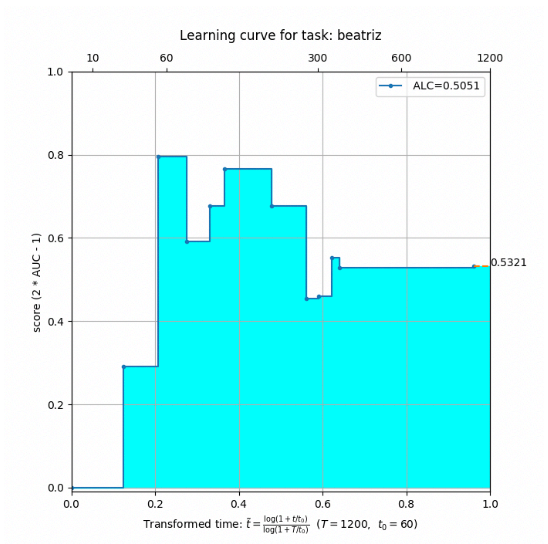
Google Cloud

NVIDIA®

Nvidia **P100** GPU

New Features

compared to AutoML challenges



Any-time learning

Raw data

Large scale

What is the goal of AutoDL?

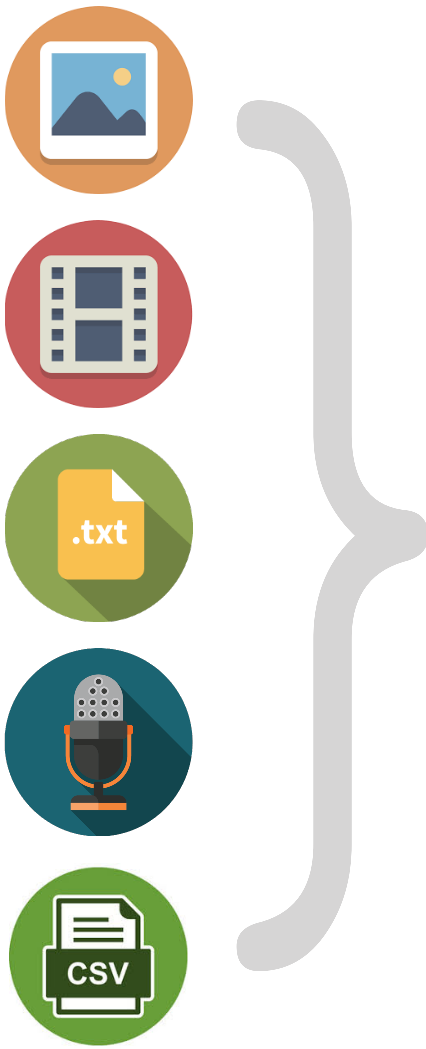
Three-level formulation of AutoML

algorithm in each level is characterized uniquely by their **input** and **output**

	Input	Output	Comp. Ex.
Alpha level: predict() in sklearn, a **classifier**	x example/sample (e.g. an image)	y labels	**Code Jam LeetCode**
Beta level: fit() in sklearn, a **learning algo.**	D ML task (dataset)	α alpha-level algo	**Kaggle**
Gamma level: meta_fit()? on a meta-dataset	$\mathfrak{D}$ Meta-dataset	β beta-level algo	**AutoDL** (powered by CodaLab)

Z. Liu et al, "Overview and unifying conceptualization of Automated Machine Learning"

Data



4-D Tensor of shape

(time, row, column, channel)

we also provide data
converter for PyTorch

Data → Meta-learning

15 image + 10 video + 15 speech + 15 text + 50 tabular

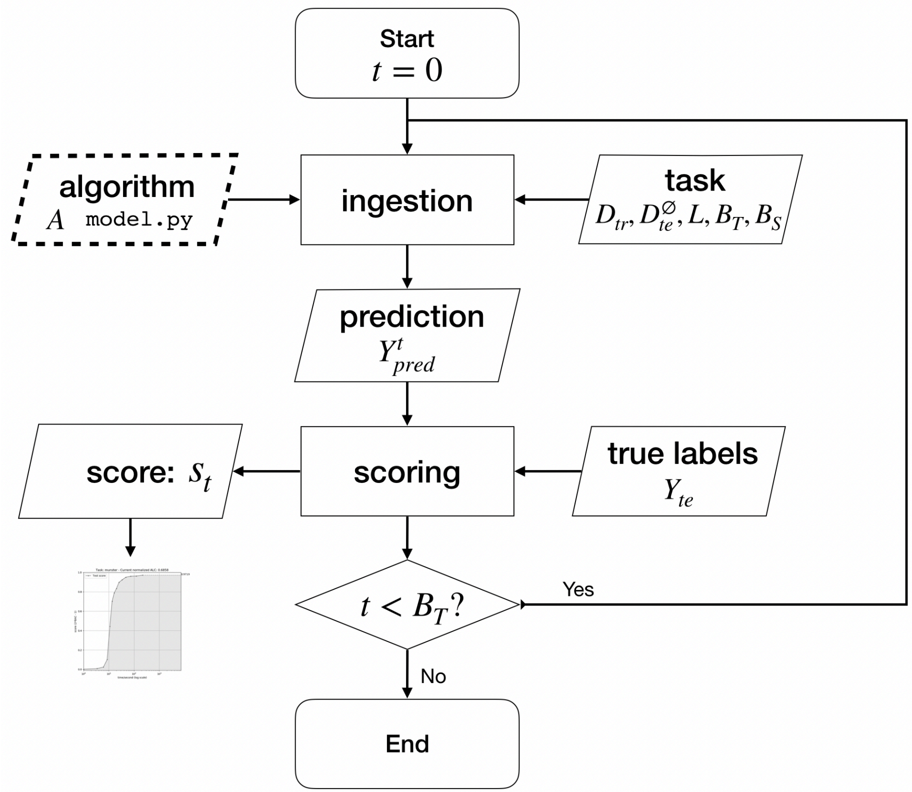
#	Dataset	Challenge	Phase	Domain	Class number	Sample number		Tensor dimension			
						train	test	time	row	col	channel
1	Munster	AutoCV	public	hand-writing	10	60000	10000	1	28	28	1
2	Chucky	AutoCV	public	objects	100	48061	11939	1	32	32	3
3	Pedro	AutoCV	public	people	26	80095	19905	1	var	var	3
4	Decal	AutoCV	public	aerial	11	634	166	1	var	var	3
5	Hammer	AutoCV	public	medical	7	8050	1965	1	600	450	3
6	Ukulele	AutoCV	feedback	hand-writing	3	6979	1719	1	var	var	3
7	Caucase	AutoCV	feedback	objects	257	24518	6089	1	var	var	3
8	Beatriz	AutoCV	feedback	people	15	4406	1094	1	350	350	3
9	Saturn	AutoCV	feedback	aerial	3	324000	81000	1	28	28	4
10	Hippocrate	AutoCV	feedback	medical	2	175917	44108	1	96	96	3
11	Loukoum	AutoCV	final	hand-writing	3	27938	6939	1	var	var	3
12	Tim	AutoCV	final	objects	200	80000	20000	1	32	32	3
13	Apollon	AutoCV	final	people	100	6077	1514	1	var	var	3
14	Ideal	AutoCV	final	aerial	45	25231	6269	1	256	256	3
15	Ray	AutoCV	final	medical	7	4492	1114	1	976	976	3
16	Kraut	AutoCV2	public	action	4	1528	863	var	120	160	1
17	Katze	AutoCV2	public	action	6	1528	863	var	120	160	1
18	Kreatur	AutoCV2	public	action	4	1528	863	var	60	80	1
19	Ideal	AutoCV2	feedback	aerial	45	25231	6269	1	256	256	3
20	Freddy	AutoCV2	feedback	hand-writing	2	546055	136371	var	var	var	3
21	Homer	AutoCV2	feedback	action	12	1354	353	var	var	var	3
22	Isaac2	AutoCV2	feedback	action	249	38372	9561	var	102	78	1
23	Formula	AutoCV2	feedback	miscellaneous	4	32994	8203	var	80	80	3
24	Apollon	AutoCV2	final	people	100	6077	1514	1	var	var	3
25	Loukoum	AutoCV2	final	hand-writing	3	27938	6939	1	var	var	3
26	Fiona	AutoCV2	final	action	6	8038	1962	var	var	var	3
27	Monica1	AutoCV2	final	action	20	10380	2565	var	168	168	3
28	Kitsune	AutoCV2	final	action	25	18602	4963	var	46	82	3

dataset formatting toolkit available at:
https://github.com/zhengying-liu/autodl-contrib

Liu Z, Xu Z, Escalera S, Guyon I, Treguer S, Tu W-W. Towards Automated Computer Vision: Analysis of the AutoCV Challenges 2019. :7.

Evaluation

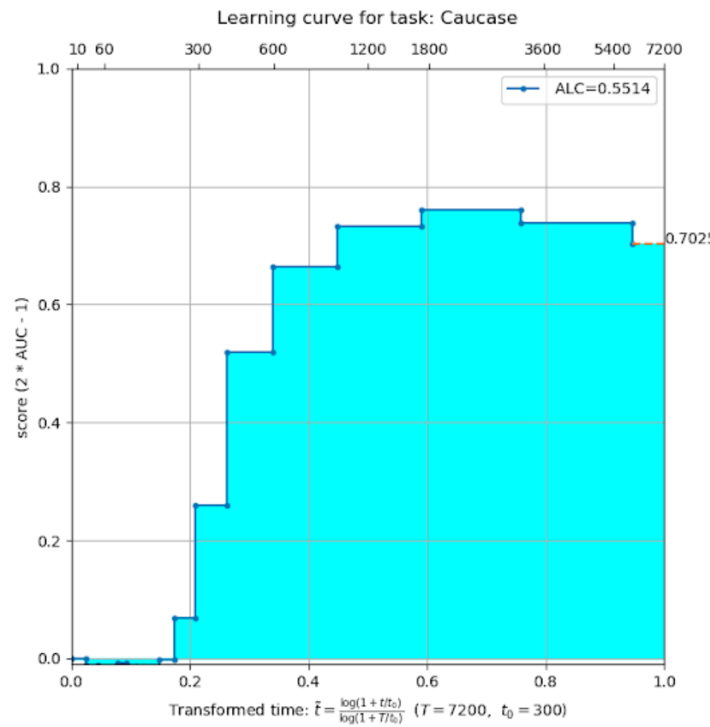
- **Multiple predictions** to make
- **ROC AUC**
- Area under Learning Curve (**ALC**)
- Average rank



Evaluation

Time rescaling: $$\tilde{t}(t) = \frac{\log(1 + t/t_0)}{\log(1 + T/t_0)}$$

$$\begin{aligned} ALC &= \int_0^1 s(t) d\tilde{t}(t) \\ &= \int_0^T s(t) \tilde{t}'(t) dt \\ &= \frac{1}{\log(1 + T/t_0)} \int_0^T \frac{s(t)}{t + t_0} dt \end{aligned}$$



Participation

challenge name	Collocated with	#participants	#submissions	begin date (2019)	end date (2019)
AutoCV	IJCNN	**102**	**938**	May 1	Jun 29
AutoCV2	ECML PKDD	**34**	**336**	July 2	Aug 20
AutoNLP	WAIC	**66**	**420**	Aug 2	Aug 31
AutoSpeech	ACML	**33**	**234**	Sep 16	Oct 16
AutoWSL	ACML	**26**	**439**	Sep 24	Oct 29

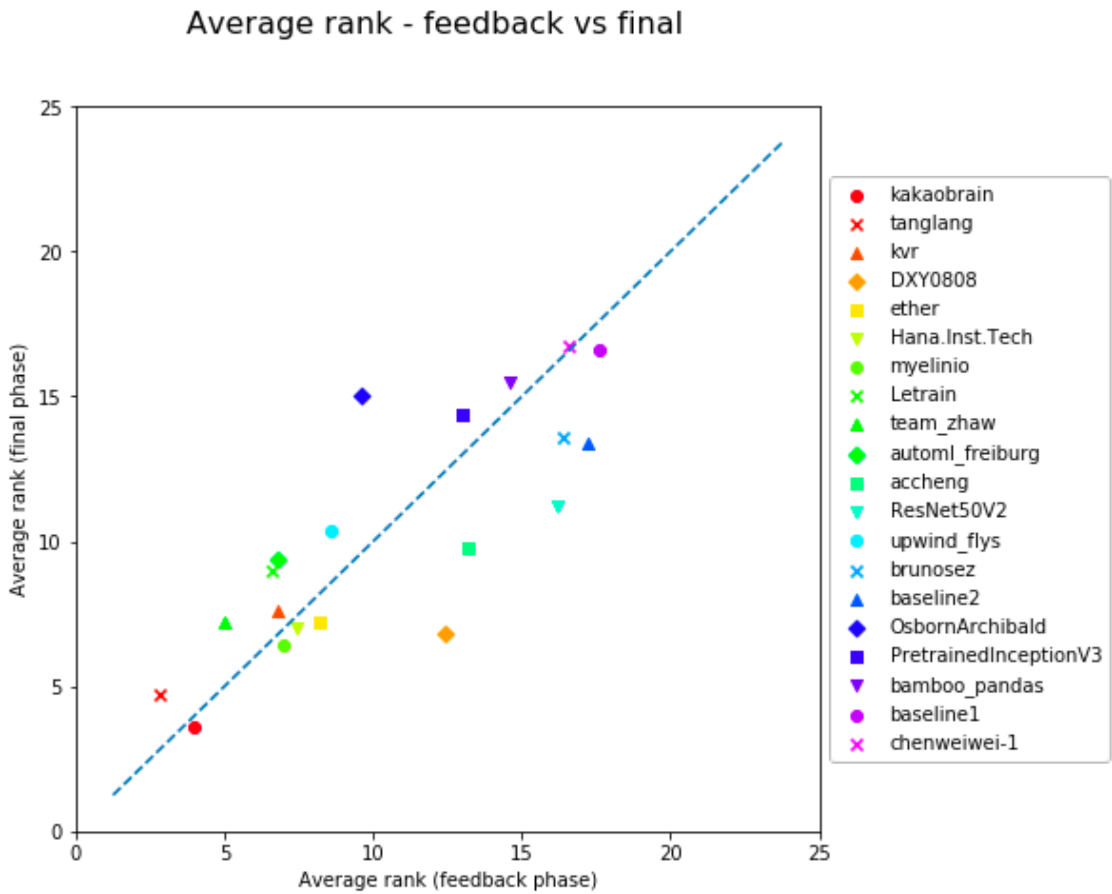
Winners

Challenge	1st place ($2000)	2rd place ($1500)	3nd place ($500)
AutoCV	**kakaobrain** (Kakao Brain)	**DKKimHCLee** (Hana. Tech. Inst.)	**base_1** (Hanyang University)
AutoCV2	**kakaobrain** (Kakao Brain)	**tanglang** (Xiamen University)	**kvr** (-)
AutoNLP	**DeepBlueAI** (DeepBlue Technology)	**upwind_flys** (Lenovo)	**txta** (gsdata.cn)
AutoSpeech	**PASA_NJU** (Nanjing University)	**DeepWisdom** (fuzhi.ai)	**Kon** (NS Solutions Corporation)
AutoWSL	**DeepWisdom** (fuzhi.ai)	**Meta_Learners** (Tsinghua University)	**lhg1992** (inspur.com)

All winners' code is now **open-sourced** on GitHub

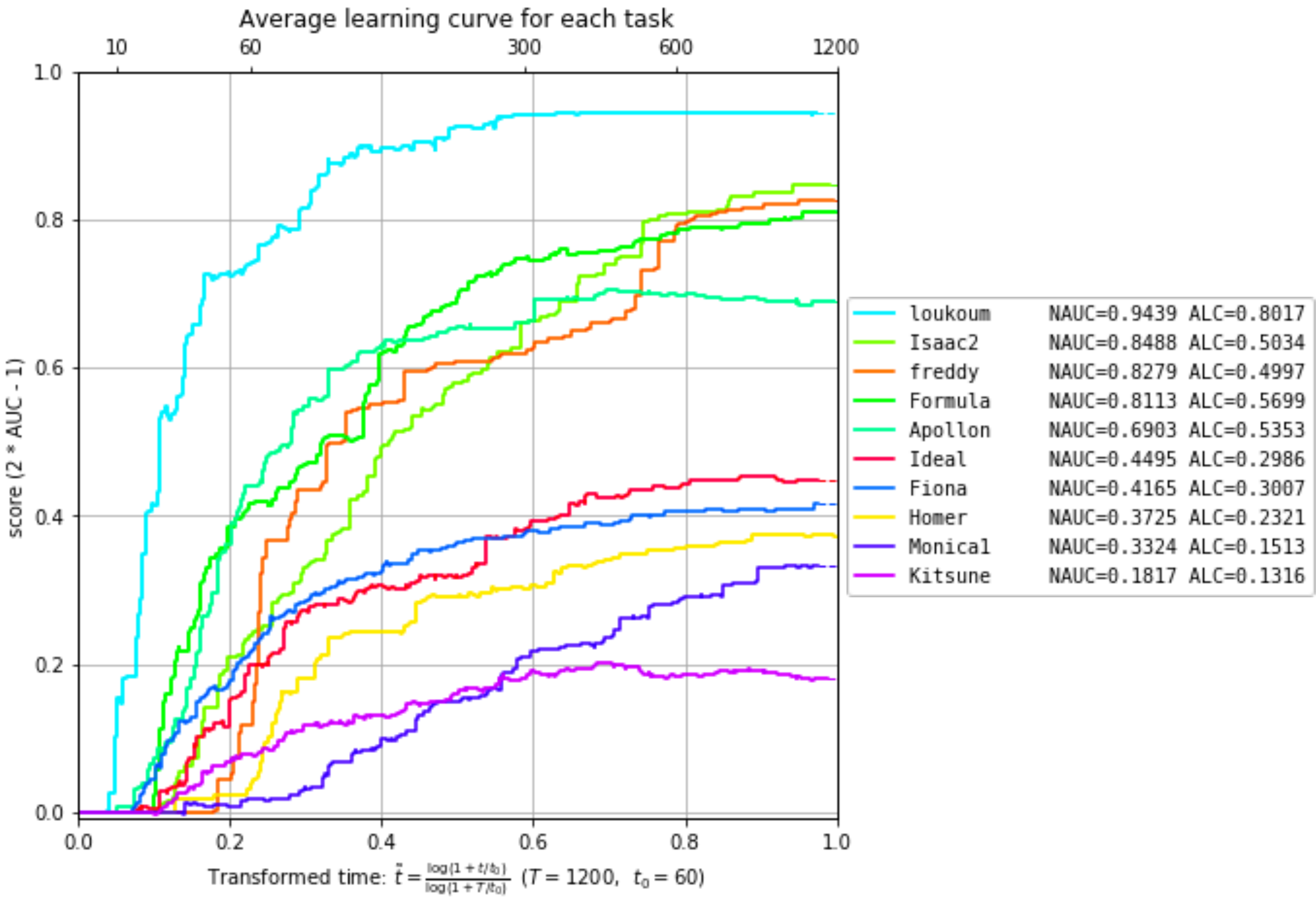
URLs can be found on: **autodl.chalearn.org**

Leaderboard overfitting

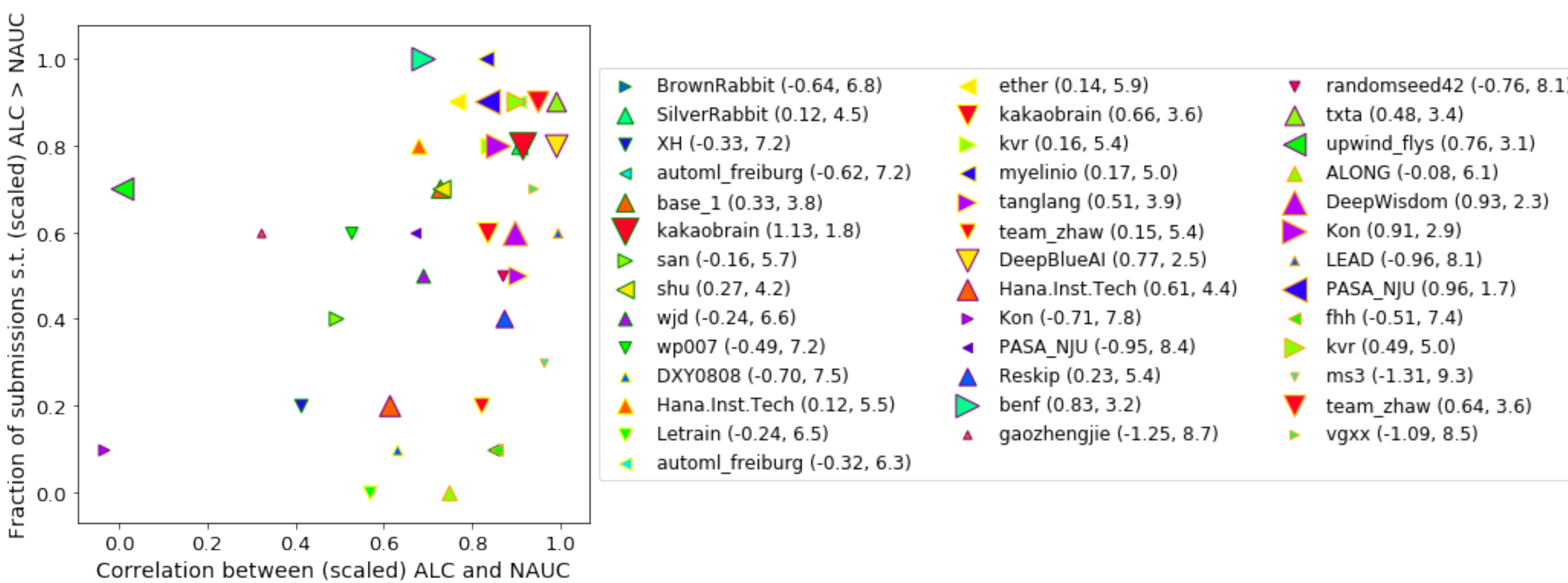


No leaderboard overfitting => universal AutoML solutions

Dataset difficulty



Any-time learning problem



Some teams have good final performance but bad any-time performace => any-time learning aspect to be studied further

Conclusion

Take-home messages

Exploration-exploitation trade-off: keep it in mind!

Domain specific AutoML solution generalizes

Hand-crafted gamma-level learning
=> Cross-domain meta-learning yet to be studied

Any-time learning aspect to be studied further

AutoDL challenge still on-going!

Open problems

Theoretical possibility of AutoML

Can we beat "No Free Lunch"? Why? How?

Computational considerations

Statistical vs Computational trade-off: what's the limit?

Theoretical guarantee of ensemble methods

Rigorous mathematical proof of the effectiveness of ensemble methods

MATHÉMATIQUES VISION APPRENTISSAGE

université PARIS-SACLAY

Automated Deep Learning

LIU Zhengying

U. Paris-Sud / Inria / U. Paris Saclay

Thank you! Questions?

Internship opportunities:

1. **AutoDL Benchmark** with extensive GPU usage
2. **Meta-learning challenge** design and implementation

Contact: zhengying.liu@inria.fr