Massachusetts Institute of Technology

Meta Learning

MIT
Iddo Drori, Fall 2020

Automated Machine Learning

- Why automate machine learning?

- Which elements of machine learning can we automate?

Automated Machine Learning

- Activation functions
- Optimizers
- Data augmentation
- Algorithm selection and hyperparameter optimization
- Neural network architectures
- Feature extraction and selection
- Machine learning and data science pipelines

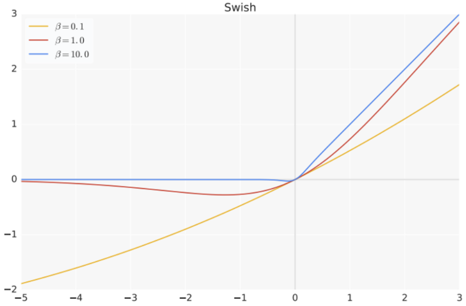
Automated Activation Functions

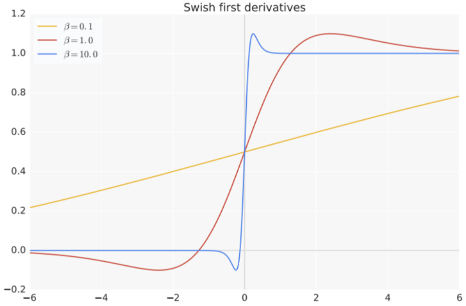
- Search space:

Unary functions: $x, -x, |x|, x^2, x^3, \sqrt{x}, \beta x, x+\beta, \log(|x|+\epsilon), \exp(x)\ \sin(x), \cos(x), \sinh(x), \cosh(x), \tanh(x), \sinh^{-1}(x), \tan^{-1}(x), \mathrm{sinc}(x), \max(x,0), \min(x,0), \sigma(x), \log(1+\exp(x)), \exp(-x^2), \mathrm{erf}(x), \beta$

Binary functions: $x_1+x_2, x_1 \cdot x_2, x_1-x_2, \frac{x_1}{x_2+\epsilon}, \max(x_1,x_2), \min(x_1,x_2), \sigma(x_1)\cdot x_2, \exp(-\beta(x_1-x_2)^2), \exp(-\beta|x_1-x_2|), \beta x_1+(1-\beta)x_2$

Function	RN	WRN	DN
ReLU [$\max(x,0)$]	93.8	95.3	94.8
$x \cdot \sigma(\beta x)$	94.5	95.5	94.9
$\max(x, \sigma(x))$	94.3	95.3	94.8
$\cos(x) - x$	94.1	94.8	94.6
$\min(x, \sin(x))$	94.0	95.1	94.4
$(\tan^{-1}(x))^2 - x$	93.9	94.7	94.9
$\max(x, \tanh(x))$	93.9	94.2	94.5
$\mathrm{sinc}(x) + x$	91.5	92.1	92.0
$x \cdot (\sinh^{-1}(x))^2$	85.1	92.1	91.1





Source: Searching for Activation Functions, Ramachandran et al 2017.

Automated Optimizer

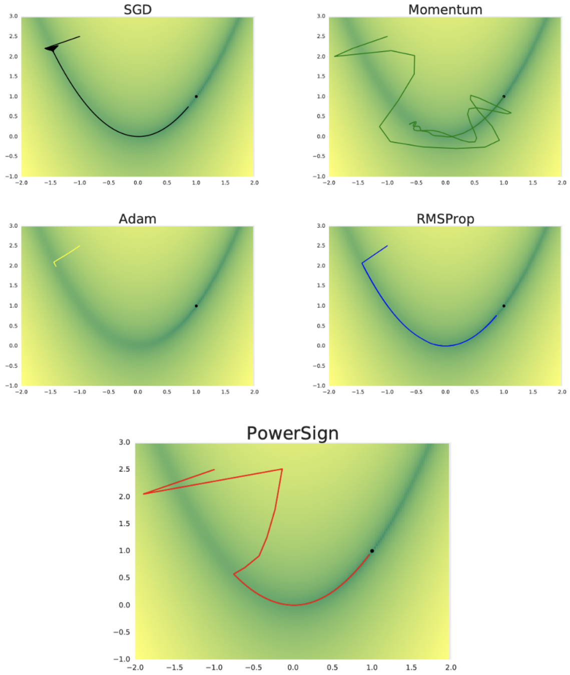
- Search space:

Operands: g, g^2, g^3, $\hat{m}$, $\hat{v}$, $\hat{\gamma}$, $\text{sign}(g)$, $\text{sign}(\hat{m})$, 1, 2, $\epsilon \sim N(0, 0.01)$, $10^{-4}w$, $10^{-3}w$, $10^{-2}w$, $10^{-1}w$, Adam and RMSProp.

Unary functions which map input x to: x, $-x$, e^x, $\log|x|$, $\sqrt{|x|}$, $clip(x, 10^{-5})$, $clip(x, 10^{-4})$, $clip(x, 10^{-3})$, $drop(x, 0.1)$, $drop(x, 0.3)$, $drop(x, 0.5)$ and $\text{sign}(x)$.

Binary functions which map (x, y) to $x + y$ (addition), $x - y$ (subtraction), $x * y$ (multiplication), $\frac{x}{y+\delta}$ (division), x^y (exponentiation) or x (keep left).

PowerSign: $\alpha^{f(t)*\text{sign}(g)*\text{sign}(m)} * g$



Optimizer	Best Test	Final Test
SGD	93.0	92.3
Momentum	93.0	92.2
Adam	92.6	92.3
RMSProp	92.3	91.6
PowerSign	93.0	92.4
PowerSign-ld	93.6	93.4
PowerSign-cd	93.7	93.1
PowerSign-rd$_{10}$	94.2	92.6
PowerSign-rd$_{20}$	94.4	92.0
AddSign	93.0	92.6
AddSign-ld	93.5	92.0
AddSign-cd	93.6	92.4
AddSign-rd$_{10}$	94.2	94.0
AddSign-rd$_{20}$	**94.4**	**94.3**

Figure source: Neural Optimizer Search with Reinforcement Learning, Bello et al 2017.

Automated Data Augmentation

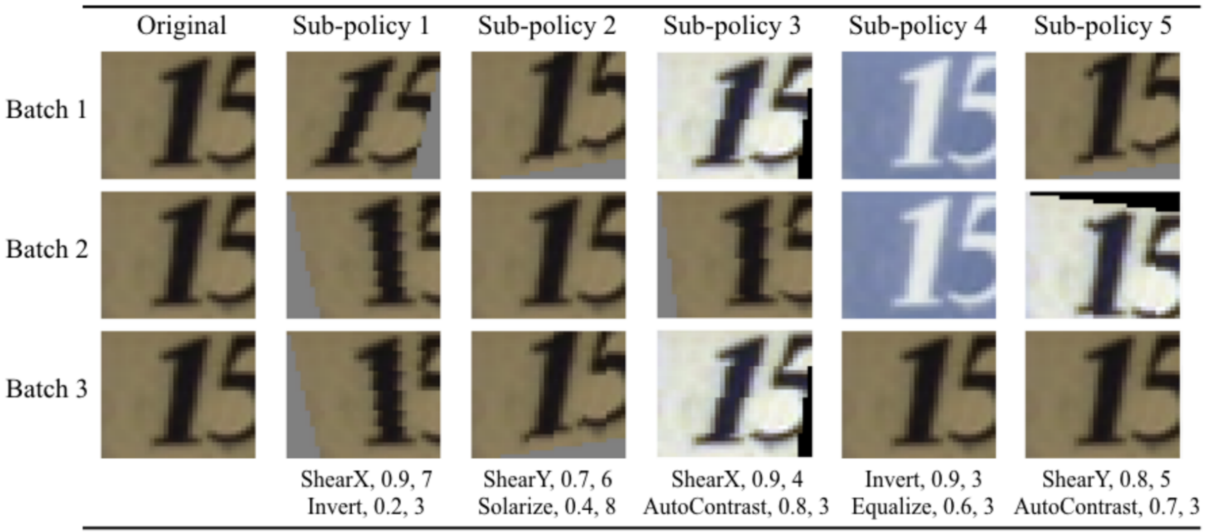
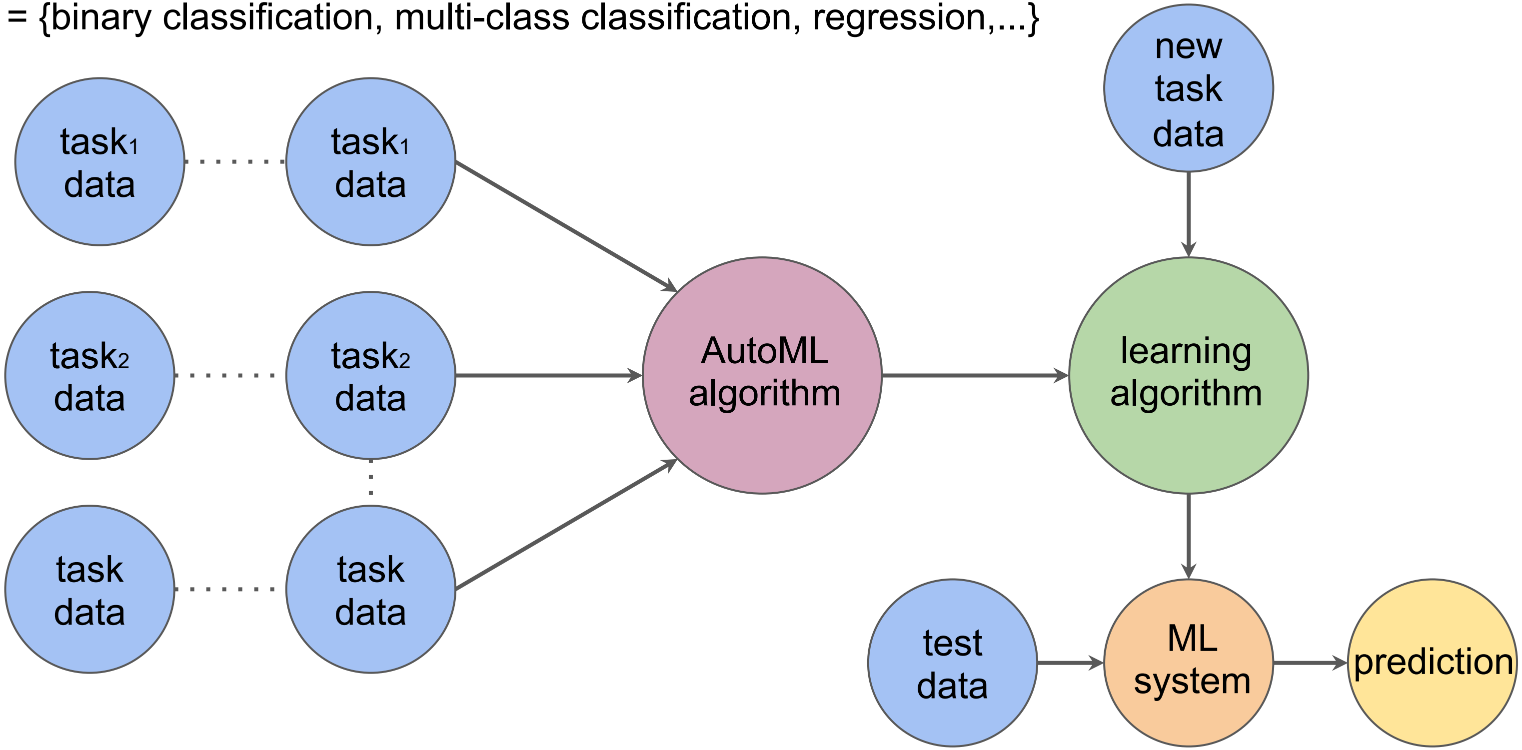


Figure source: AutoAugment: Learning Augmentation Strategies from Data, Cubuk et al, 2019.

Automated Machine Learning

tasks = {binary classification, multi-class classification, regression,...}



Hyperparameter Optimization

Hyperparameters

- Learning rate alpha
- Momentum term beta
- Minibatch size
- # of layers
- # of hidden units
- Learning rate decay

...

	Name	Range	Default	log scale	Type	Conditional
Network hyperparameters	batch size	$[32, 4096]$	32	✓	float	-
	number of updates	$[50, 2500]$	200	✓	int	-
	number of layers	$[1, 6]$	1	-	int	-
	learning rate	$[10^{-6}, 1.0]$	10^{-2}	✓	float	-
	L_2 regularization	$[10^{-7}, 10^{-2}]$	10^{-4}	✓	float	-
	dropout output layer	$[0.0, 0.99]$	0.5	✓	float	-
	solver type	{SGD, Momentum, Adam, Adadelta, Adagrad, smorm, Nesterov }	smorm3s	-	cat	-
	lr-policy	{Fixed, Inv, Exp, Step}	fixed	-	cat	-
Conditioned on solver type	β_1	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	β_2	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	ρ	$[0.05, 0.99]$	0.95	✓	float	✓
	momentum	$[0.3, 0.999]$	0.9	✓	float	✓
Conditioned on lr-policy	γ	$[10^{-3}, 10^{-1}]$	10^{-2}	✓	float	✓
	k	$[0.0, 1.0]$	0.5	-	float	✓
	s	$[2, 20]$	2	-	int	✓
Per-layer hyperparameters	activation-type	{Sigmoid, TanH, ScaledTanH, ELU, ReLU, Leaky, Linear}	ReLU	-	cat	✓
	number of units	$[64, 4096]$	128	✓	int	✓
	dropout in layer	$[0.0, 0.99]$	0.5	-	float	✓
	weight initialization	{Constant, Normal, Uniform, Glorot-Uniform, Glorot-Normal, He-Normal, He-Uniform, Orthogonal, Sparse}	He-Normal	-	cat	✓
	std. normal init.	$[10^{-7}, 0.1]$	0.0005	-	float	✓
	leakiness	$[0.01, 0.99]$	$\frac{1}{3}$	-	float	✓
	tanh scale in	$[0.5, 1.0]$	$2/3$	-	float	✓
	tanh scale out	$[1.1, 3.0]$	1.7159	✓	float	✓

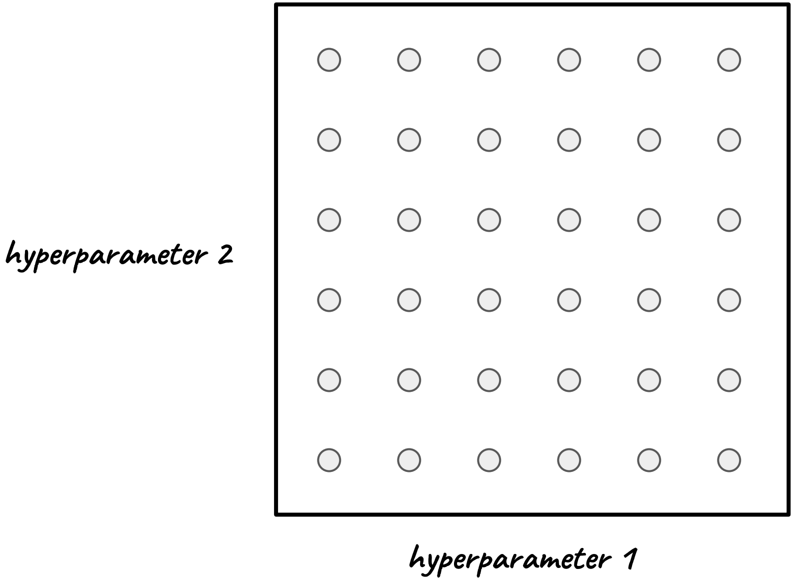
Hyperparameter Optimization

- Given dataset D find hyperparameters θ which minimize the loss of a model generated by algorithm A trained on D_{train} and evaluated on D_{valid}

$$\theta^* = arg\min_{\theta} \mathbb{E}_{(D_{train}, D_{valid}) \sim D} V(\mathcal{L}, A_\theta, D_{train}, D_{valid})$$

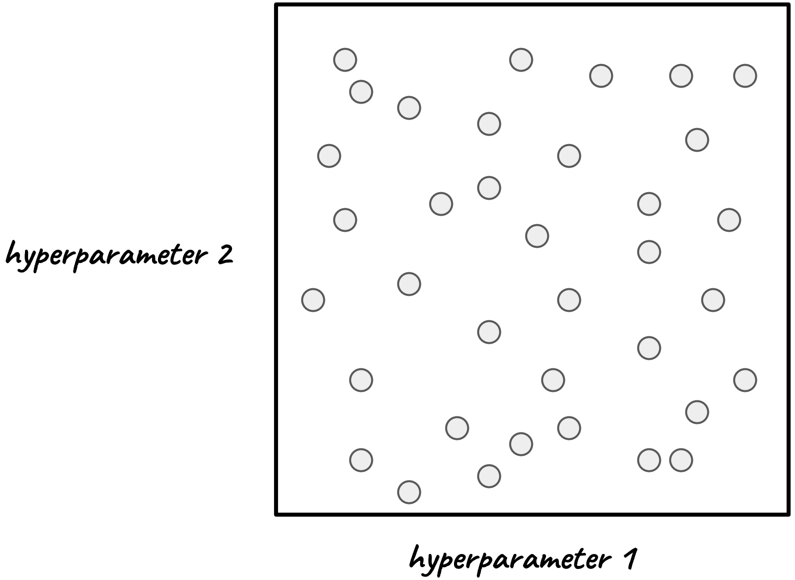
Grid Search

- Regularly sample grid
- Test grid values

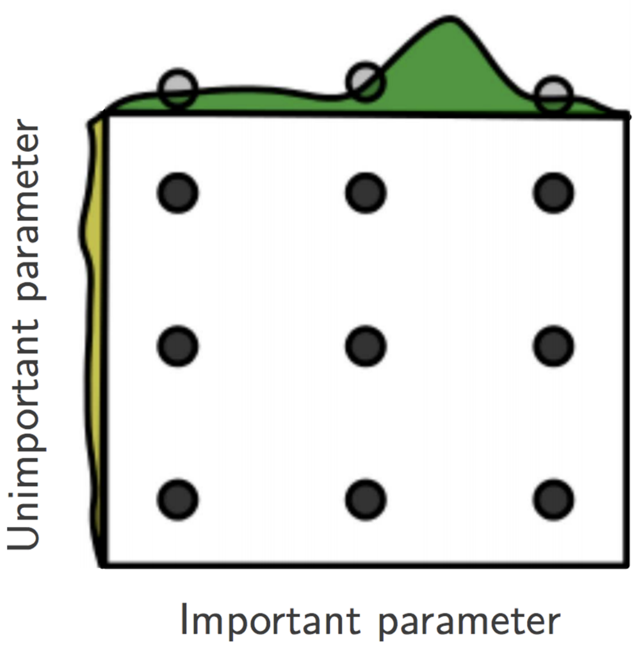


Random Search

- Randomly sample grid
- Test random values



Grid Search vs. Random Search



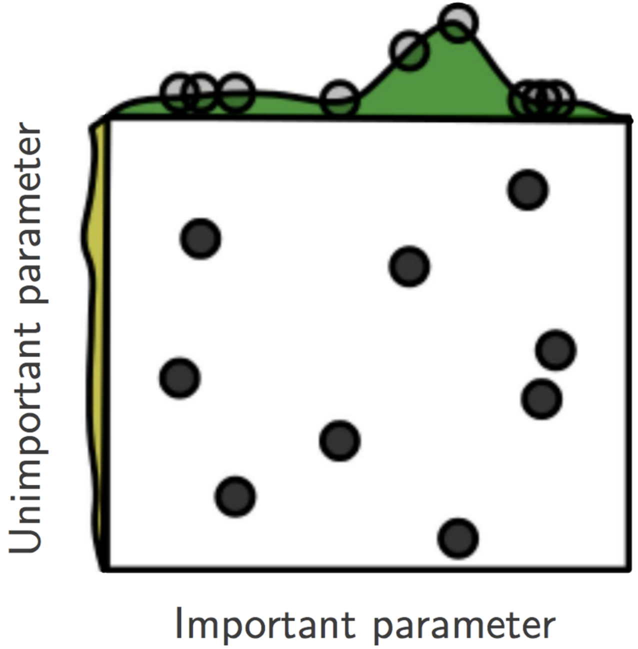


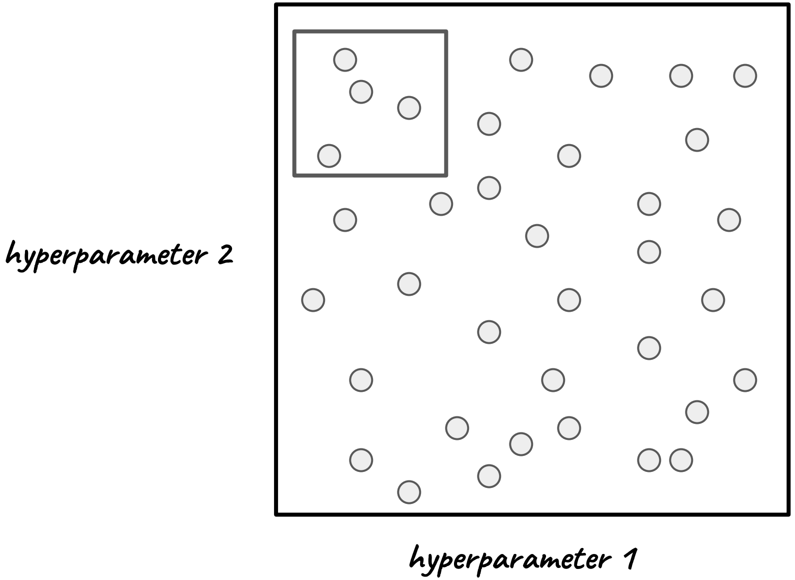
Figure source: Random search for hyper-parameter optimization, James Bergstra and Yoshua Bengio, JMLR 2012

Coarse to Fine Optimization

- Efficient optimization
- Sample examples
- Sample features

Adaptive Coarse to Fine Sampling

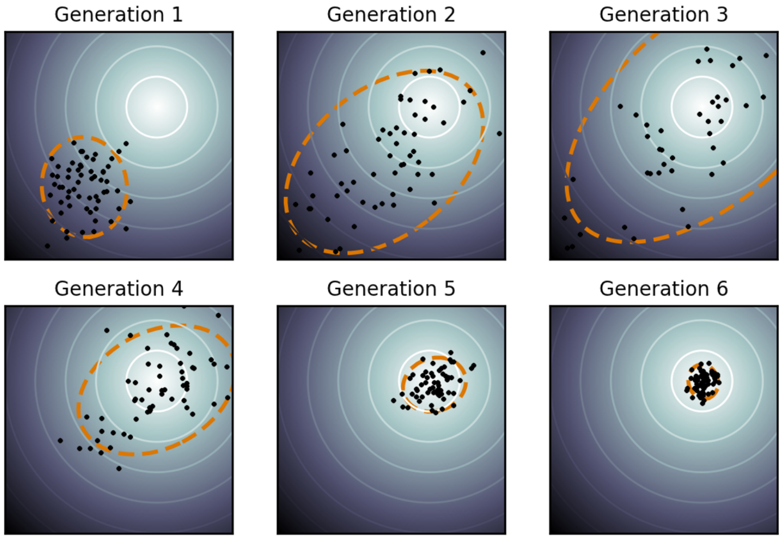
- Zoom in
- Perform dense search in small region of relevant values



Covariance Matrix Adaptation Evolution Strategy

- Model free
- Evaluate multiple points in parallel rather than a single point sequentially
- Sample new configurations
- Reorder configurations based on fitness
- Update state variables, covariance, based on ordered solutions

performance



Bayesian Optimization

- Black-box optimization
- Minimize unknown objective function which is costly to evaluate and we may only observe the function value.

$$\boldsymbol{x}_* = \text{argmin } \boldsymbol{f}(\boldsymbol{x})$$

- Used to select model hyperparameters
- Construct sequence of points $\boldsymbol{x}_1 ... \boldsymbol{x}_n$ that converge to $\boldsymbol{x}_*$
- Goal is to get best approximate solution given allocated budget of n samples

Solution: Bayesian Optimization

- Place a prior on the objective function $\boldsymbol{f}$
- Each time we evaluate $\boldsymbol{f}$ at a new point $\boldsymbol{x}_i$, we update our model for $\boldsymbol{f}(\boldsymbol{x})$
- Model is cheap surrogate objective function reflecting beliefs about $\boldsymbol{f}$
- Beliefs are encoded in posterior
- Use posterior to derive acquisition function $\alpha(\boldsymbol{x})$ which is fast to evaluate and differentiate, for example by gradient descent, evaluating $\alpha(\boldsymbol{x})$ at many points $\boldsymbol{x}$.

Bayesian Optimization

Repeat until convergence

Use the acquisition function to derive next query point according to
$\boldsymbol{x}_{i+1} = \text{argmin}\ \boldsymbol{\alpha}(\boldsymbol{x})$

Evaluate $\boldsymbol{f}(\boldsymbol{x}_{i+1})$ and update posterior

- Model for $\boldsymbol{f}$ and acquisition function evolve

Gaussian Processes

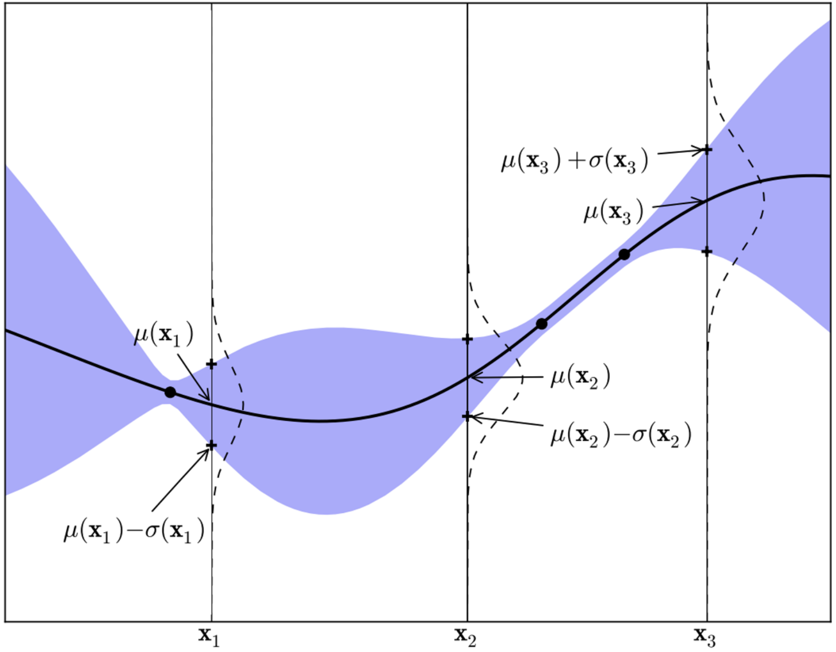
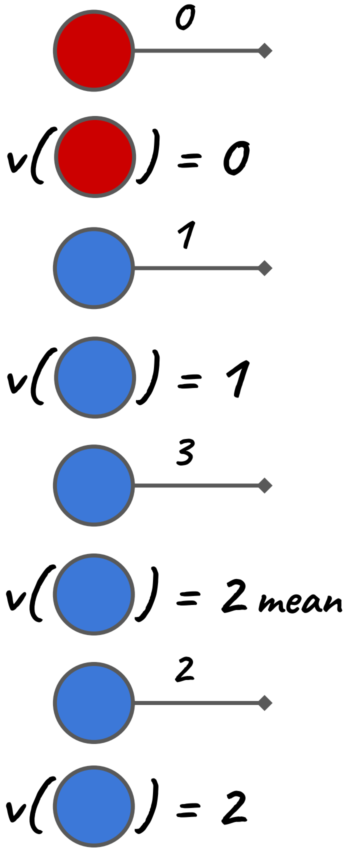
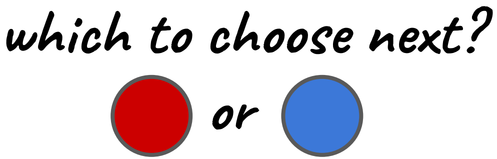


Figure source: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, Eric Brochu, Vlad M. Cora, Nando de Freitas, 2010.

Exploration vs. Exploitation





Acquisition Function

- Balance exploration and exploitation
- Lower confidence bound acquisition function

$$\alpha(\boldsymbol{x}) = \mu(\boldsymbol{x}) - \kappa\sigma(\boldsymbol{x})$$

- $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are mean and square root variance of posterior at point $\boldsymbol{x}$

- $\alpha(\boldsymbol{x})$ minimized for $\boldsymbol{x}$ where:
  - $\mu(\boldsymbol{x})$ is small, exploitation
  - $\sigma(\boldsymbol{x})$ is large, exploration
- $\kappa>0$ controls trade-off between exploitation and exploration
  - small κ, encourages exploitation
  - large κ, encourages exploration

Massachusetts Institute of Technology

Acquisition Function Optimization

- Seed minimization algorithm with multiple values
- Run minimization algorithm to approximate convergence for each value
- Select value which minimizes acquisition function

MIT Massachusetts Institute of Technology

Bayesian Optimization

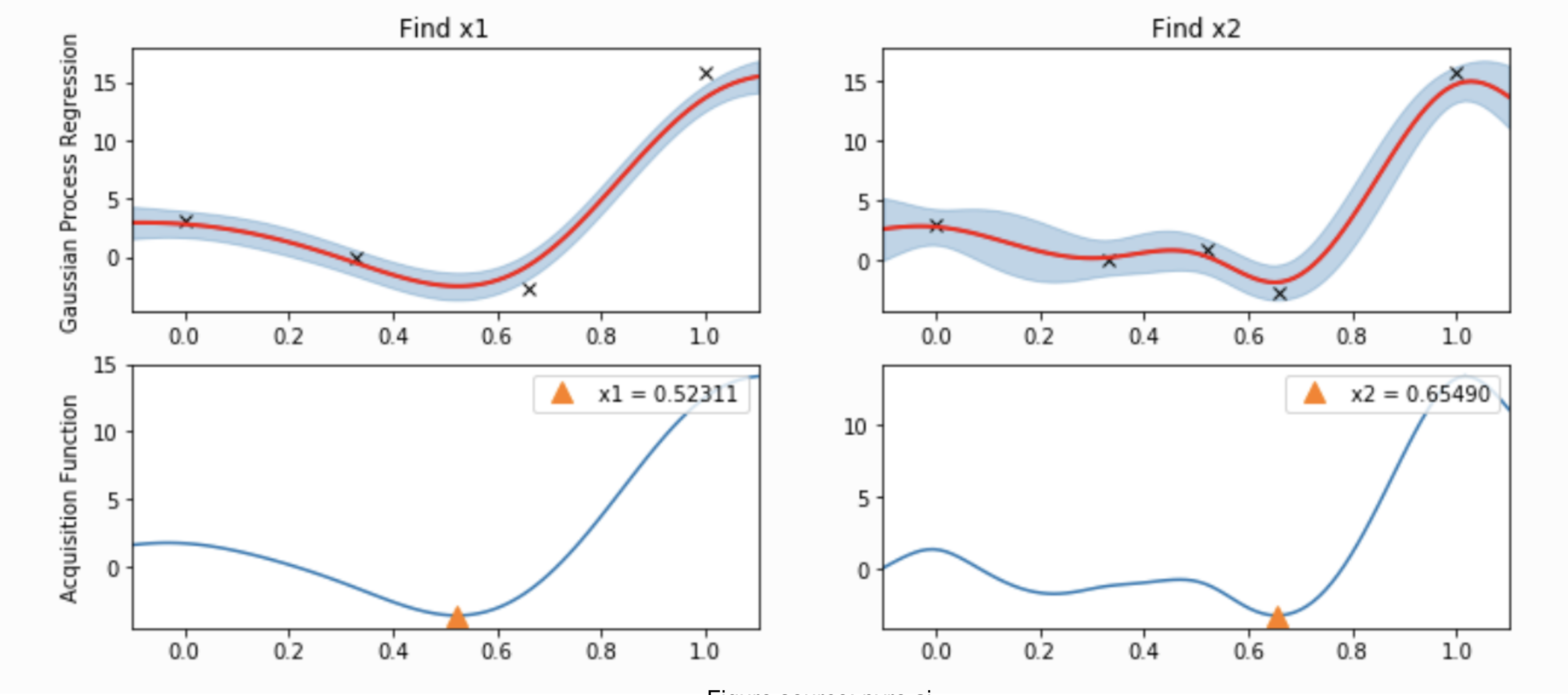


Figure source: pyro.ai

Bayesian Optimization

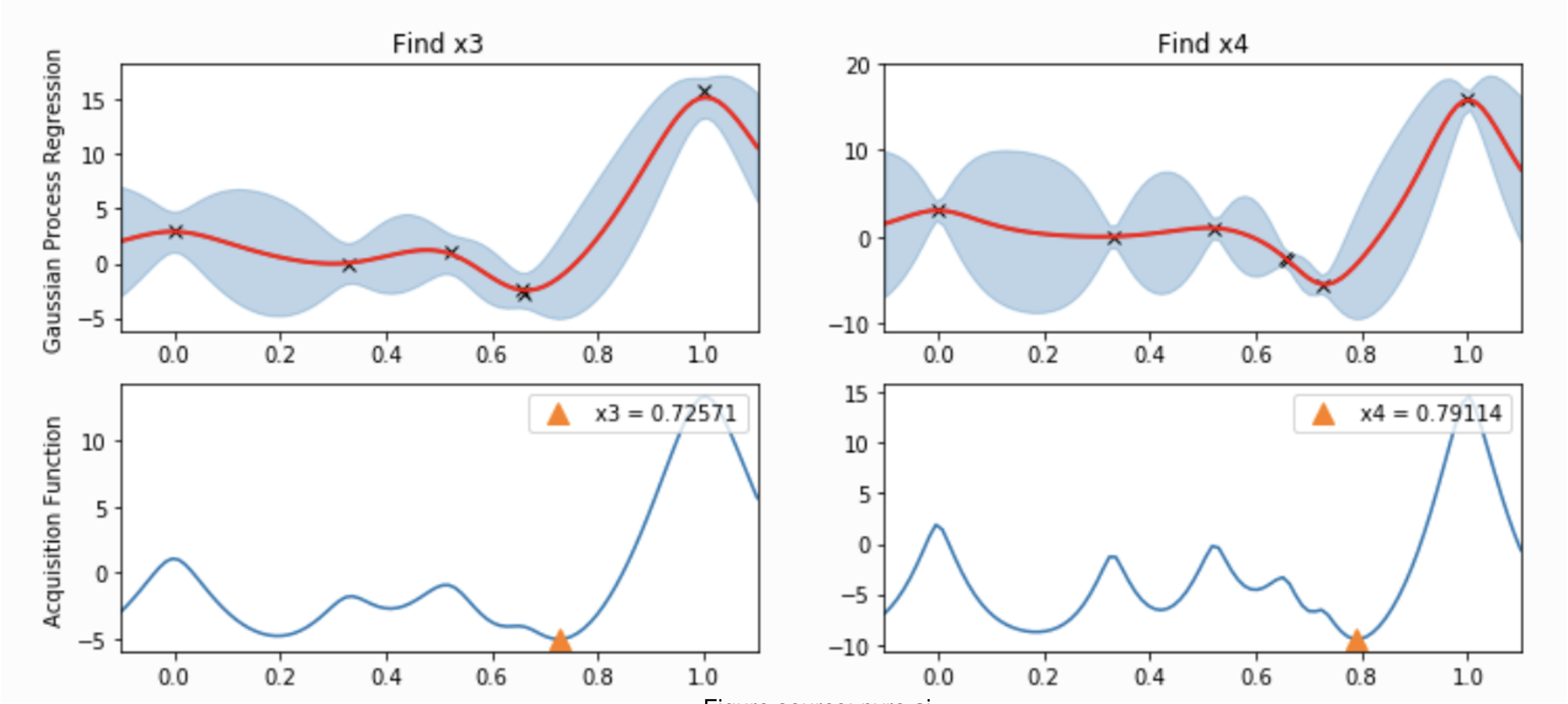


Figure source: pyro.ai

Bayesian Optimization

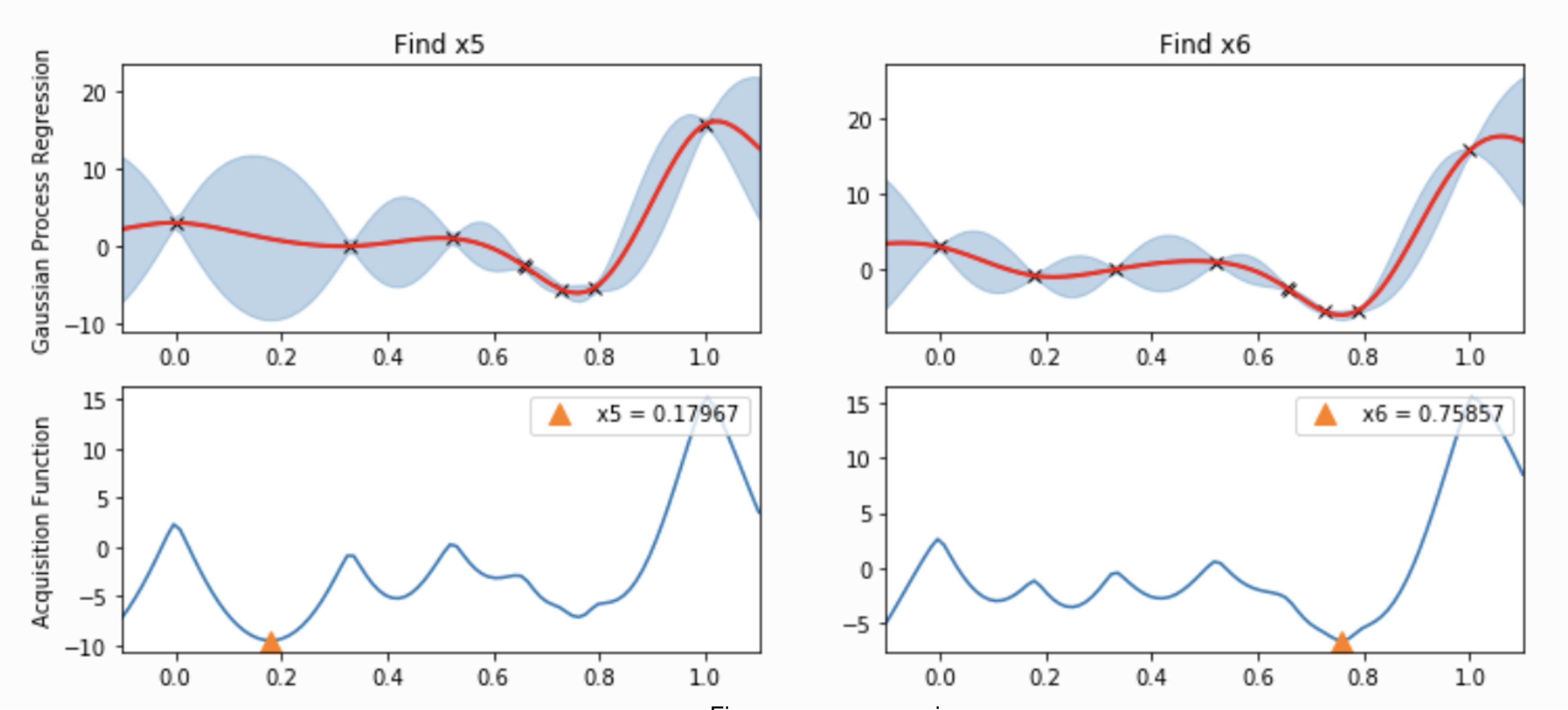


Figure source: pyro.ai

Massachusetts Institute of Technology

Bayesian Optimization

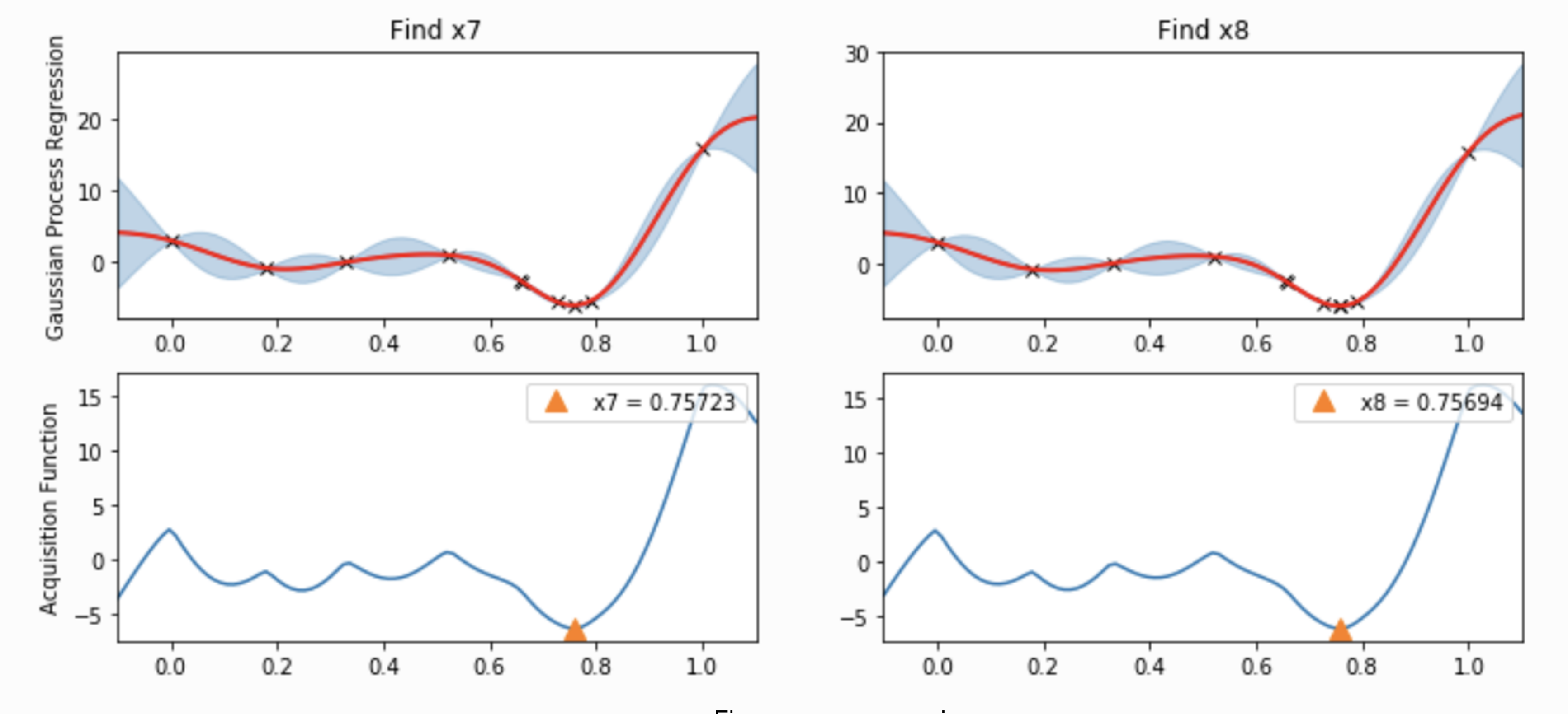


Figure source: pyro.ai

Massachusetts Institute of Technology

Gaussian Processes

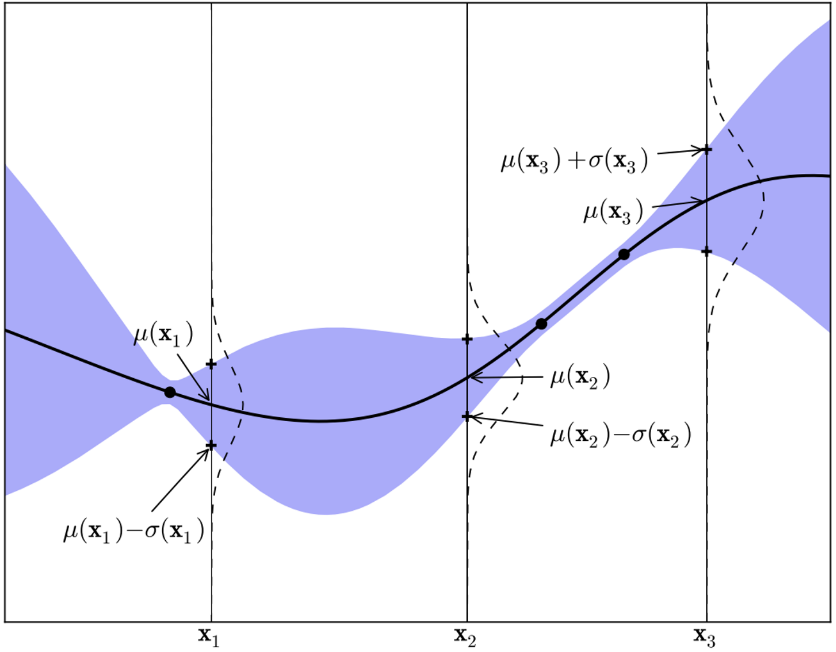


Figure source: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, Eric Brochu, Vlad M. Cora, Nando de Freitas, 2010.

Bayesian Optimization

- Build probabilistic model of objective
- Compute posterior distribution: Gaussian processes
- Optimize cheap surrogate function rather than expensive objective

Bayesian Optimization

Algorithm 1 Bayesian optimization

1: **for** $n = 1, 2, \ldots$ **do**

2: select new $\mathbf{x}_{n+1}$ by optimizing acquisition function α

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$$

3: query objective function to obtain y_{n+1}

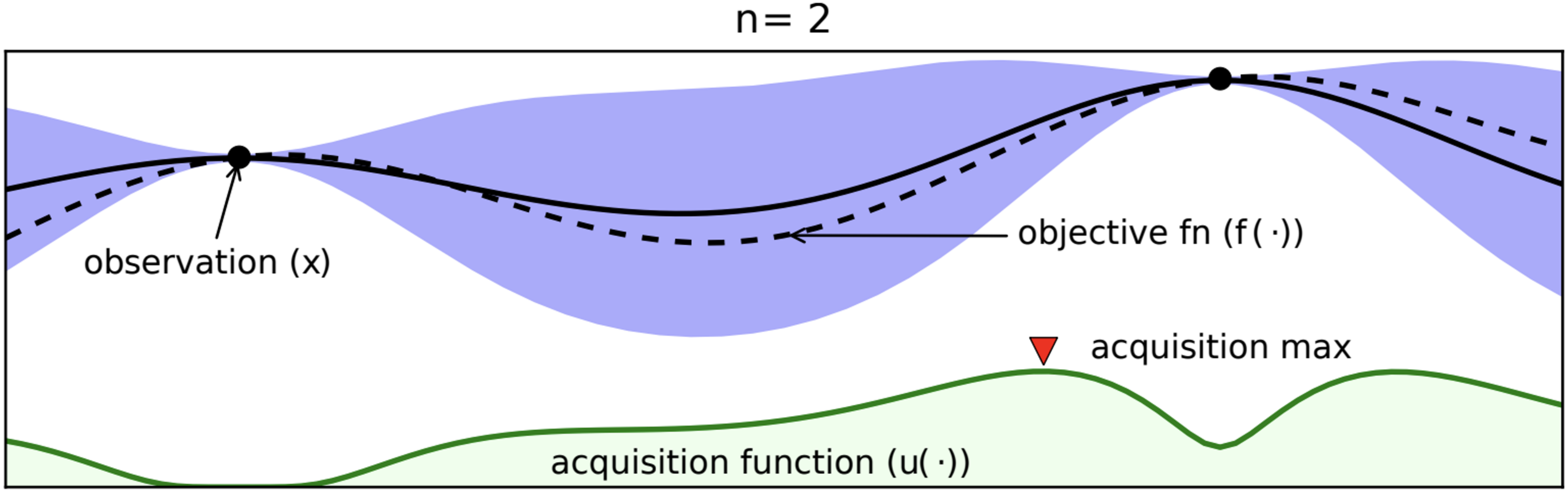
4: augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$

5: update statistical model

6: **end for**

Source: Taking the human out of the loop: A review of Bayesian optimization, Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas, IEEE, 2016.

Bayesian Optimization



Mean and confidence intervals estimated with a probabilistic model of objective function
High acquisition where model predicts high objective (exploitation) & prediction uncertainty is high (exploration)

Figure source: Taking the human out of the loop: A review of Bayesian optimization, Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas, IEEE, 2016.

Bayesian Optimization

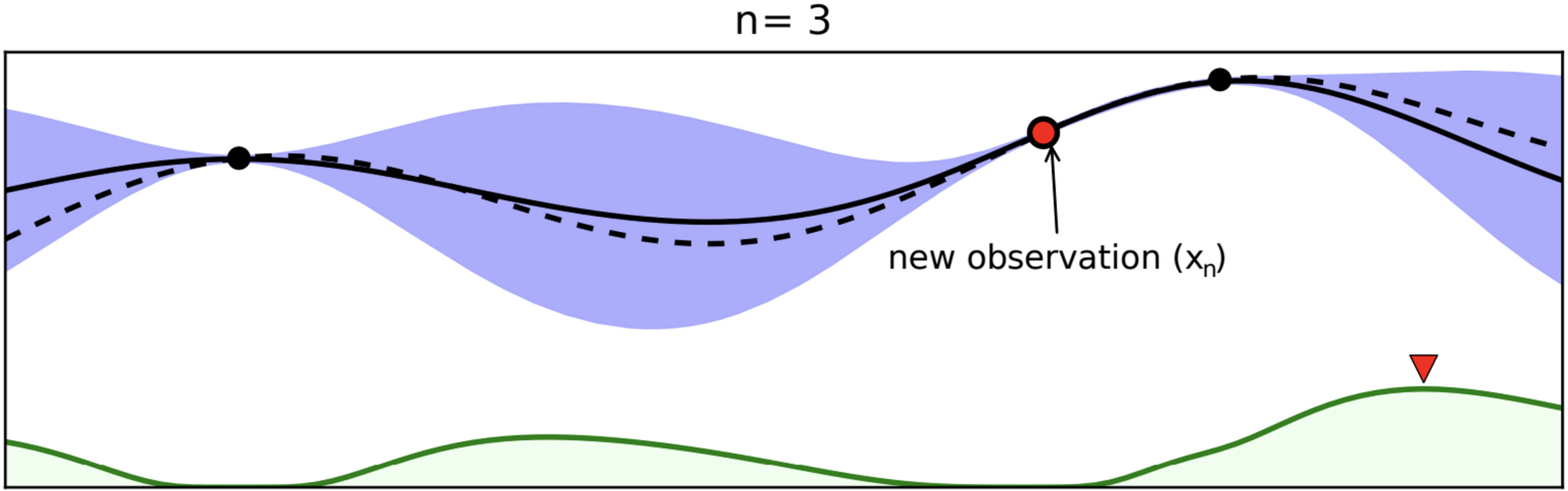


Figure source: Taking the human out of the loop: A review of Bayesian optimization, Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas, IEEE, 2016.

Bayesian Optimization

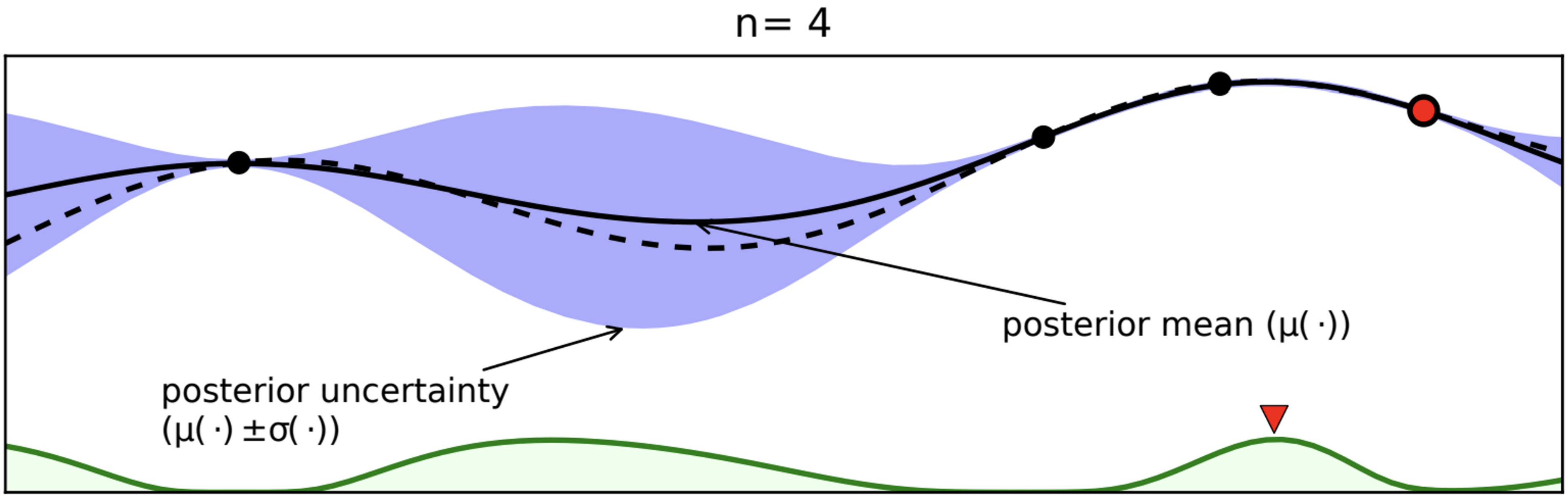


Figure source: Taking the human out of the loop: A review of Bayesian optimization, Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas, IEEE, 2016.

Acquisition Functions

- Maximum probability of improvement: exploitation
- Expected improvement
- Upper confidence bound (UCB)

Maximum Probability of Improvement

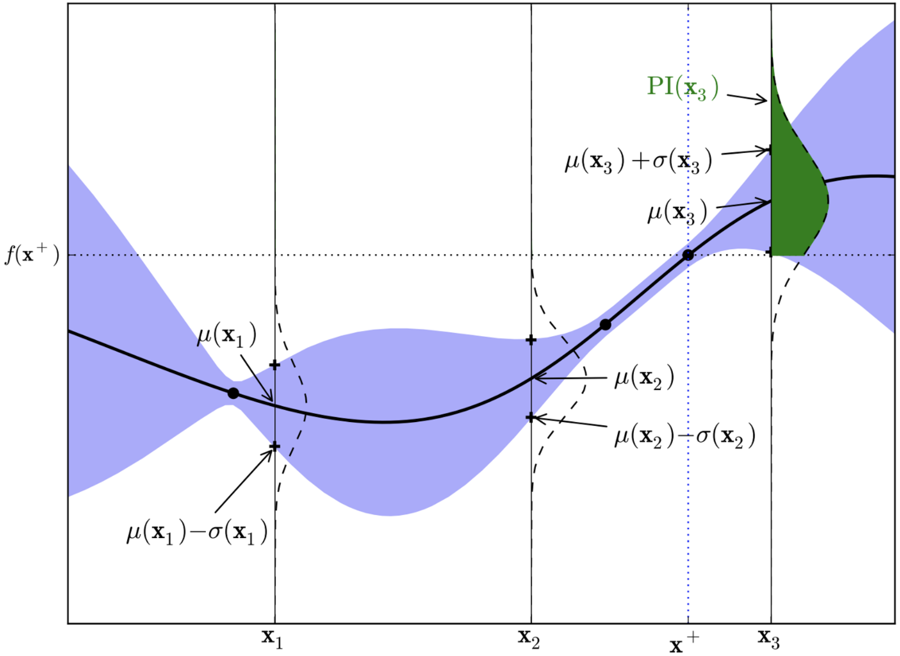


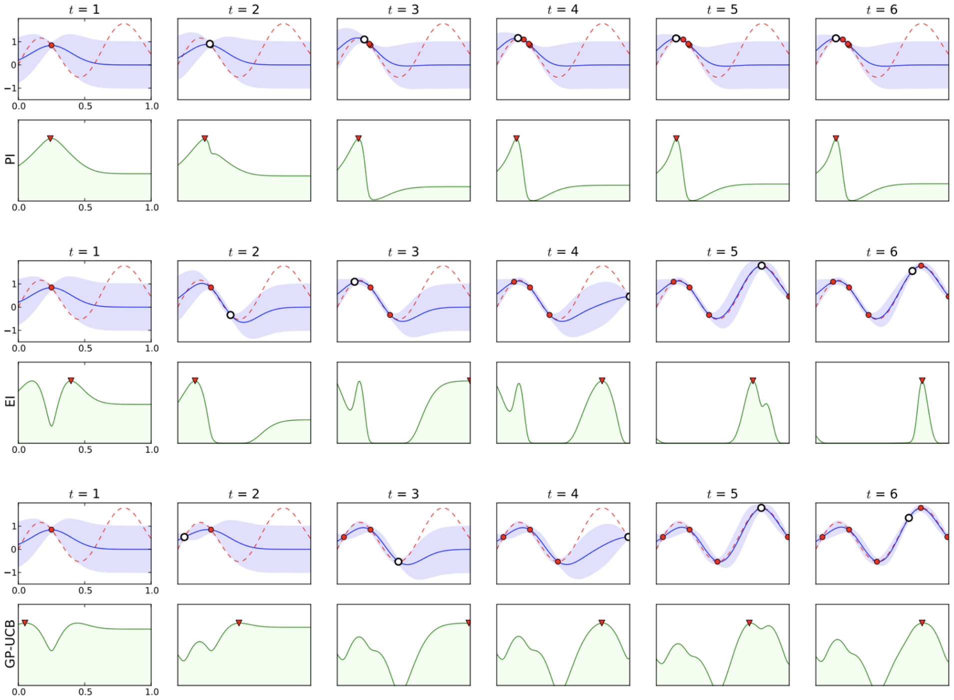
Figure source: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, Eric Brochu, Vlad M. Cora, Nando de Freitas, 2010.

Acquisition Functions

probability of improvement

expected improvement

UCB



Scale

- Log scale
- 1 - value
- Beta distribution
  replace range by parameters of beta distribution
  optimize those

	Name	Range	Default	log scale	Type	Conditional
Worker	batch size	$[32, 500]$	32	✓	int	-
	use mixup	{True, False}	True	-	bool	-
	mixup alpha	$[0.0, 1.0]$	1.0	-	float	-
	network	{MLP, ResNet, ShapedMLP, ShapedResNet}	MLP	-	cat	-
	optimizer	{Adam, SGD}	Adam	-	cat	-
	learning rate scheduler	{Step, Exponential, OnPlateau, Cyclic, CosineAnnealing}	Step	-	cat	-
Networks						
MLP	activation function	{Sigmoid, Tanh, ReLu}	Sigmoid	-	cat	✓
	num layers	$[1, 15]$	9	-	int	✓
	num units (for layer i)	$[10, 1024]$	100	✓	int	✓
	dropout (for layer i)	$[0.0, 0.5]$	0.25	-	int	✓
ResNet	activation function	{Sigmoid, Tanh, ReLu}	Sigmoid	-	cat	✓
	residual block groups	$[1, 9]$	4	-	int	✓
	blocks per group	$[1, 4]$	2	-	int	✓
	num units (for group i)	$[128, 1024]$	200	✓	int	✓
	use dropout	{True, False}	True	-	bool	✓
	dropout (for group i)	$[0.0, 0.9]$	0.5	-	int	✓
	use shake drop	{True, False}	True	-	bool	✓
	use shake shake	{True, False}	True	-	bool	✓
	shake drop β_{max}	$[0.0, 1.0]$	0.5	-	float	✓
ShapedMLP	activation function	{Sigmoid, Tanh, ReLu}	Sigmoid	-	cat	✓
	num layers	$[3, 15]$	9	-	int	✓
	max units per layer	$[10, 1024]$	200	✓	int	✓
	network shape	{Funnel, LongFunnel, Diamond, Hexagon, Brick, Triangle, Stairs}	Funnel	-	cat	✓
	max dropout per layer	$[0.0, 0.6$	0.2	-	float	✓
	dropout shape	{Funnel, LongFunnel, Diamond, Hexagon, Brick, Triangle, Stairs}	Funnel	-	cat	✓
Shaped ResNet	activation function	{Sigmoid, Tanh, ReLu}	Sigmoid	-	cat	✓
	num layers	$[3, 9]$	4	-	int	✓
	blocks per layer	$[1, 4]$	2	-	int	✓
	use dropout	{True, False}	True	-	bool	✓
	max units per layer	$[10, 1024]$	200	✓	int	✓
	network shape	{Funnel, LongFunnel, Diamond, Hexagon, Brick, Triangle, Stairs}	Funnel	-	cat	✓
	max dropout per layer	$[0.0, 0.6$	0.2	-	float	✓
	dropout shape	{Funnel, LongFunnel, Diamond, Hexagon, Brick, Triangle, Stairs}	Funnel	-	cat	✓
	use shake drop	{True, False}	True	-	bool	✓
	use shake shake	{True, False}	True	-	bool	✓
	shake drop β_{max}	$[0.0, 1.0]$	0.5	-	float	✓
Optimizers						
Adam	learning rate	$[0.0001, 0.1]$	0.003	✓	float	✓
	weight decay	$[0.0001, 0.1]$	0.05	-	float	✓
SGD	learning rate	$[0.0001, 0.1]$	0.003	✓	float	✓
	weight decay	$[0.0001, 0.1]$	0.05	-	float	✓
	momentum	$[0.1, 0.9]$	0.3	✓	float	✓
Schedulers						
Step	γ	$[0.001, 0.9]$	0.4505	-	float	✓
	step size	$[1, 10]$	6	-	int	✓
Exponential	γ	$[0.8, 0.9999]$	0.89995	-	float	✓
OnPlateau	γ	$[0.05, 0.5]$	0.275	-	float	✓
	patience	$[3, 10]$	6	-	int	✓
Cyclic	cycle length	$[3, 10]$	6	-	int	✓
	max factor	$[1.0, 2.0]$	1.5	-	float	✓
	min factor	$[0.001, 1.0]$	0.5	-	float	✓
Cosine Annealing	T_0	$[1, 20]$	10	-	int	✓
	T_{mult}	$[1.0, 2.0]$	1.5	-	float	✓

Massachusetts Institute of Technology

Multiple Objectives

- Performances
- Time
- Memory
- Constraints

Massachusetts Institute of Technology

Algorithm Selection and Hyperparameter Optimization

- Replace user's selection of algorithm and hyperparameters
- Given dataset D find the algorithm and its hyperparameters which minimize the loss of a model generated by algorithm A trained on D_{train} and evaluated on D_{valid}

$$A^*{}_{\theta^*} = arg\min_{A,\theta} \sum \mathcal{L}(A_\theta, D_{train}, D_{valid})$$

Ensemble of Models

- Models m
- Classes j
- Take argmax over average of probabilities over models for each class

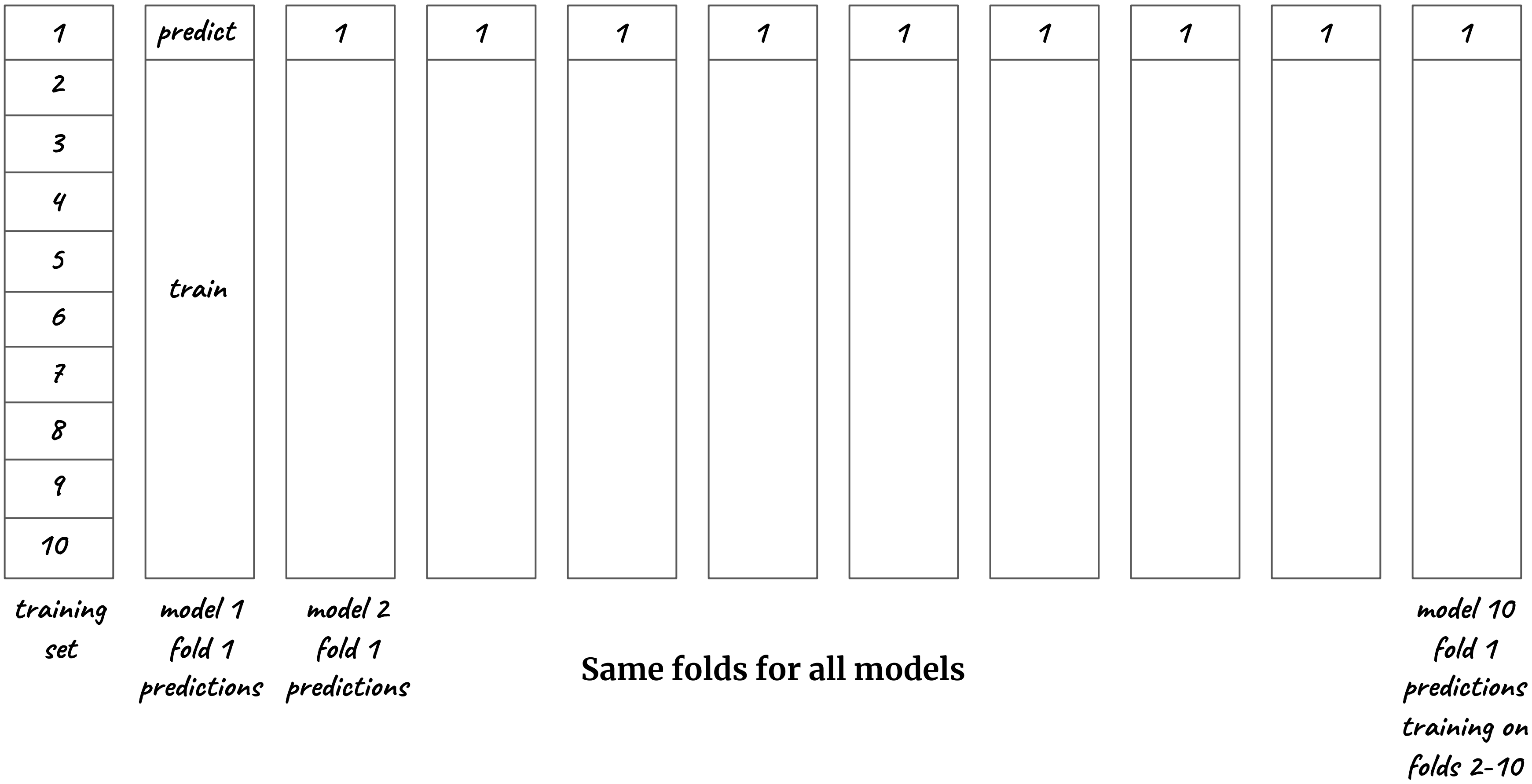
$$y = \arg\max_j \frac{1}{m}\left(\sum_{i=1}^{m} p_i^{(j)}\right)$$

Algorithm Selection and Hyperparameter Optimization

- AutoWeka

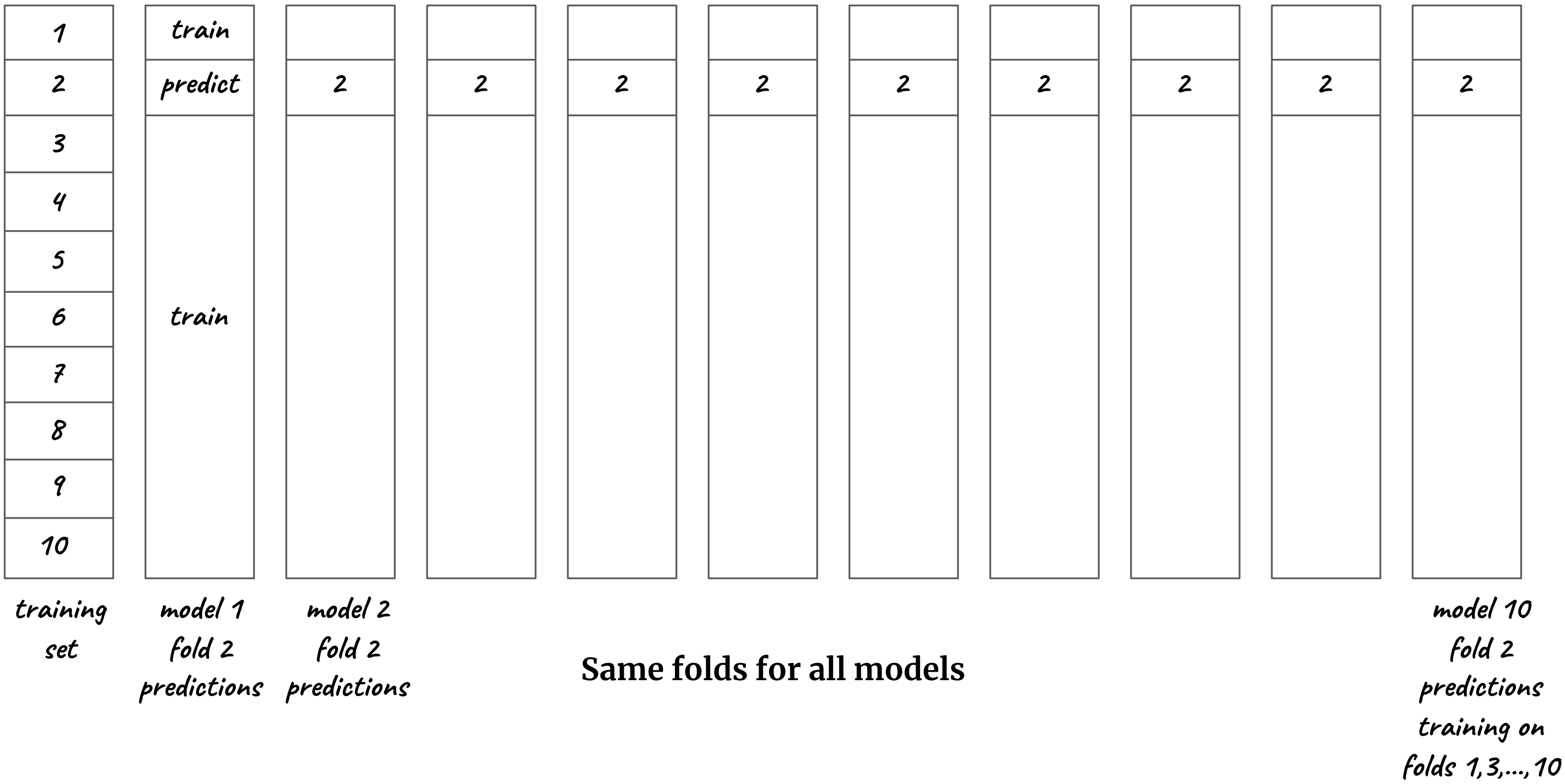
algorithm	# of hyperparameters		
Base Learners			
BayesNet	2	NaiveBayes	2
DecisionStump*	0	NaiveBayesMultinomial	0
DecisionTable*	4	OneR	1
GaussianProcesses*	10	PART	4
IBk*	5	RandomForest	7
J48	9	RandomTree*	11
JRip	4	REPTree*	6
KStar*	3	SGD*	5
LinearRegression*	3	SimpleLinearRegression*	0
LMT	9	SimpleLogistic	5
Logistic	1	SMO	11
M5P	4	SMOreg*	13
M5Rules	4	VotedPerceptron	3
MultilayerPerceptron*	8	ZeroR*	0
Ensemble Methods			
Stacking	2	Vote	2
Meta-Methods			
LWL	5	Bagging	4
AdaBoostM1	6	RandomCommittee	2
AdditiveRegression	4		
AttributeSelectedClassifier	2	RandomSubSpace	3
Feature Selection Methods			
BestFirst	2	GreedyStepwise	4

Model Stacking



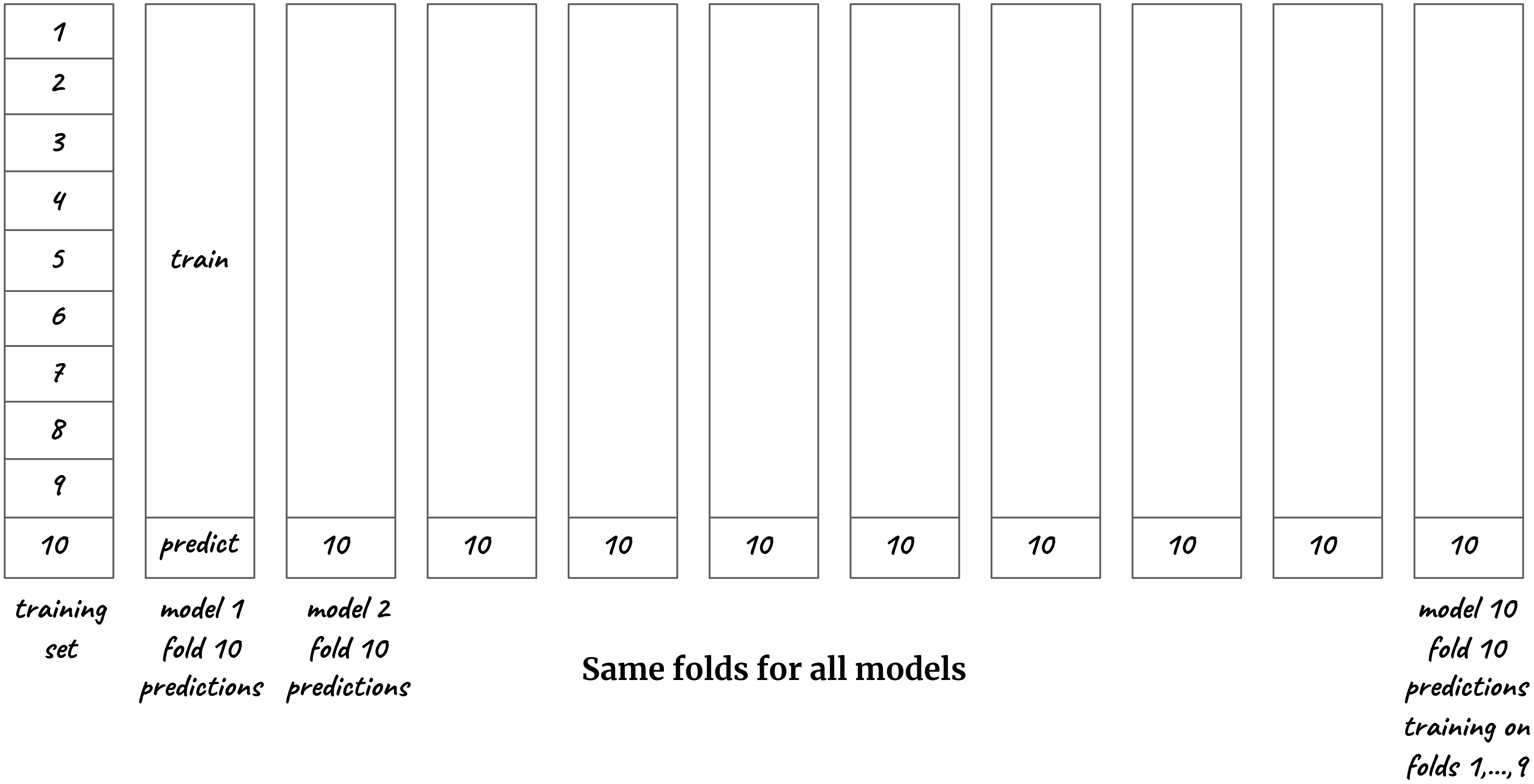
Same folds for all models

Model Stacking



Same folds for all models

Model Stacking



Same folds for all models

Model Stacking a Super-model Using Meta-features

training

1	model 1 fold 1 predictions	model 2 fold 1 predictions	model 3 fold 1 predictions	model 4 fold 1 predictions	model 5 fold 1 predictions	model 6 fold 1 predictions	model 7 fold 1 predictions	model 8 fold 1 predictions	model 9 fold 1 predictions	model 10 fold 1 predictions
2	model 1 fold 2 predictions	model 2 fold 2 predictions	model 3 fold 2 predictions	model 4 fold 2 predictions	model 5 fold 2 predictions	model 6 fold 2 predictions	model 7 fold 2 predictions	model 8 fold 2 predictions	model 9 fold 2 predictions	model 10 fold 2 predictions
3 4 5										
6										
7										
8										
9										
10	model 1 fold 10 predictions	model 2 fold 10 predictions	model 3 fold 10 predictions	model 4 fold 10 predictions	model 5 fold 10 predictions	model 6 fold 10 predictions	model 7 fold 10 predictions	model 8 fold 10 predictions	model 9 fold 10 predictions	model 10 fold 10 predictions

validation

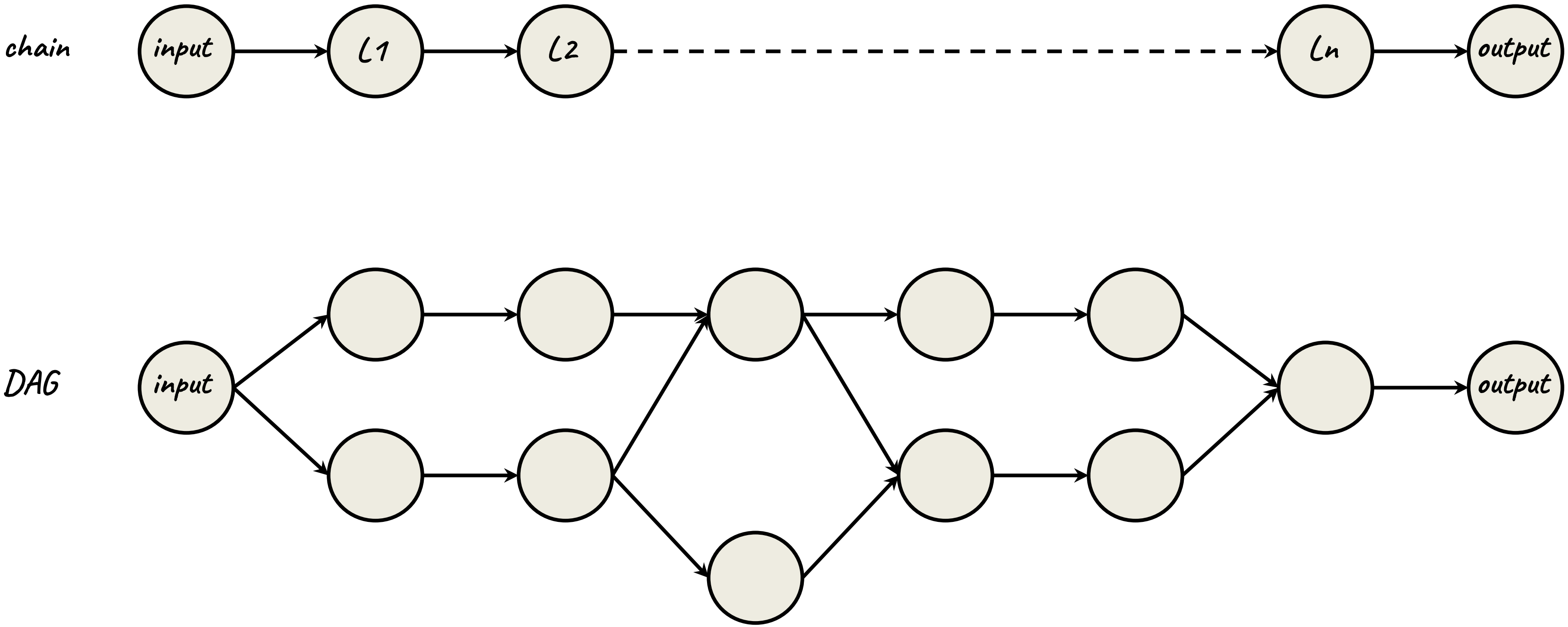
test

use level-1 model predictions as meta features for super stacked model

Neural Architecture Search

- Developing novel neural architectures manually is time consuming, error prone
- Automatic methods for searching for neural network architectures
- Search space of architectures: chain, directed acyclic graph (DAG)
- Search strategy: exploration and exploitation trade-off
- Efficient performance estimation

Neural Architecture Search



Neural Architecture Search

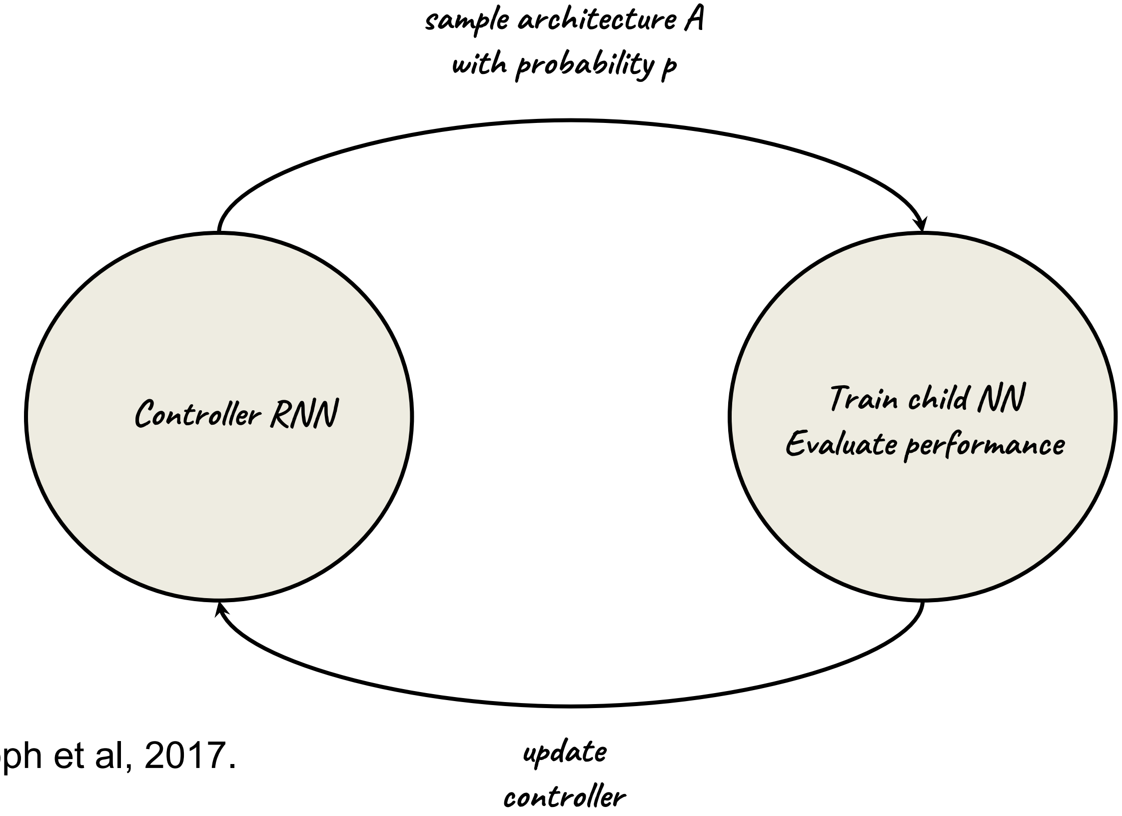


Figure source: Zoph et al, 2017.

Massachusetts Institute of Technology

Machine Learning Pipelines

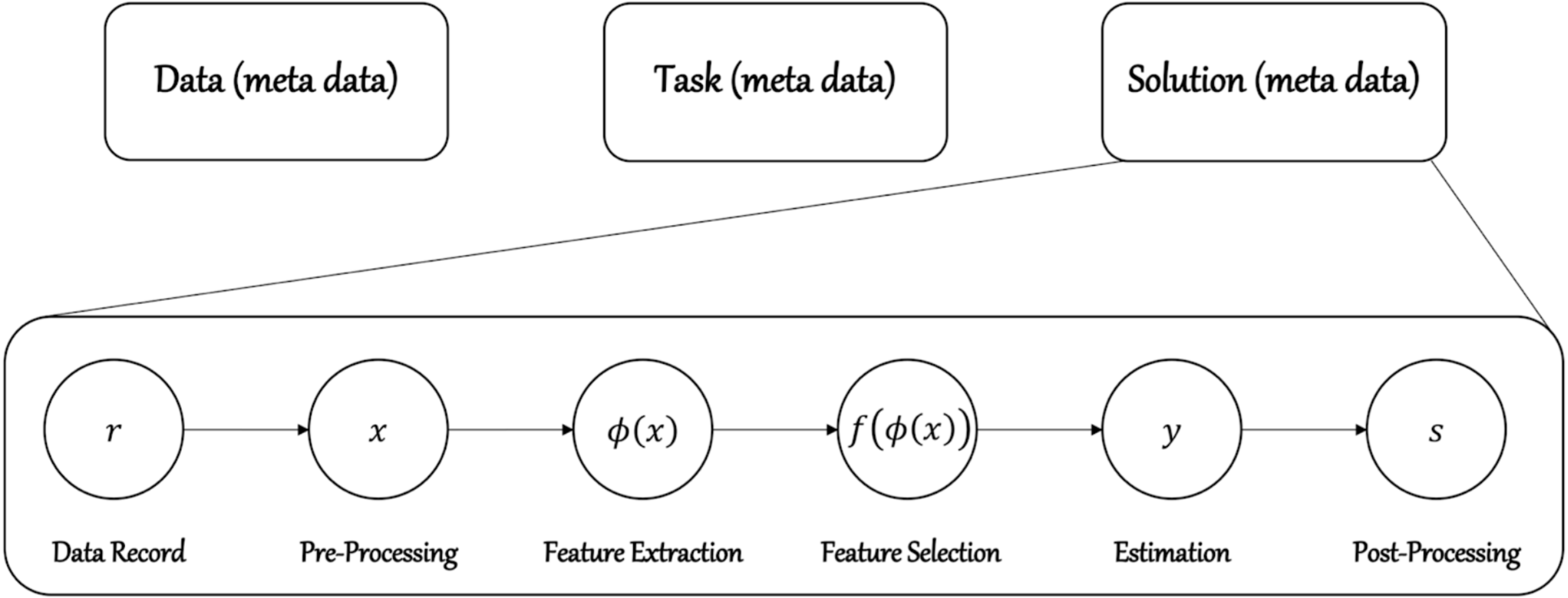
Data (meta data)

Task (meta data)

Solution (meta data)

?

Machine Learning Pipelines



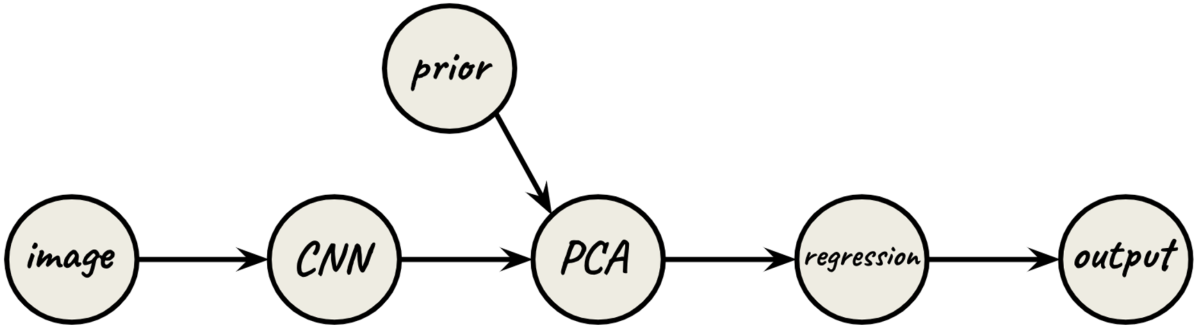
Meta Data About

- Data
- Task
- Solution

Gradient Based Methods

- Differentiable primitives
- Form a directed acyclic graph (DAG)
- Differentiable programming: optimize end-to-end

  End-to-end training of differentiable pipelines across machine learning frameworks, Mitar et al, 2017.



Massachusetts Institute of Technology

DARPA Data Driven Discovery of Models (D3M)

- Goal: solve any task on any dataset specified by a user.
- Broad set of computational primitives as building blocks.
- Automatic systems for machine learning, synthesize pipeline and hyperparameters to solve a previously unknown data and problem.
- Human in the loop: user interface that enables users to interact with and improve the automatically generated results.
- Pipelines: pre-processing, feature extraction, feature selection, estimation, post-processing, evaluation

Dataset Meta Features

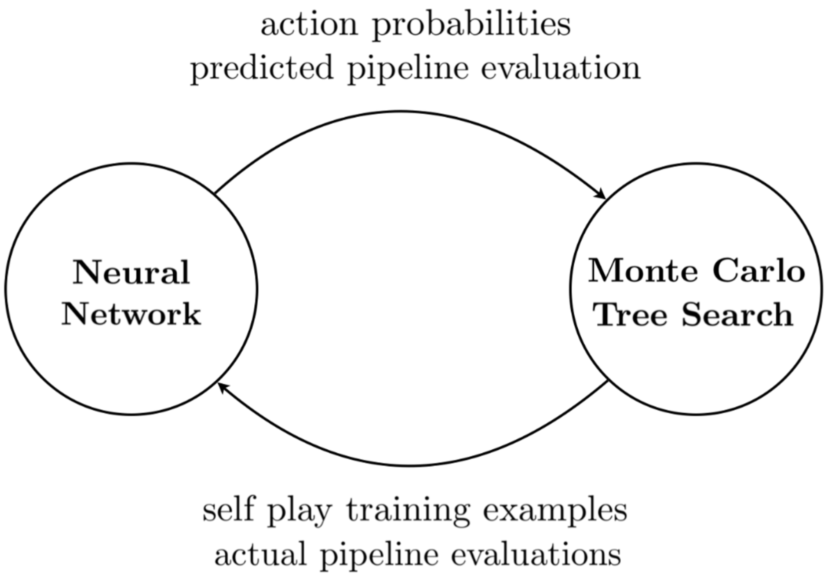
Name	Formula	Rationale	Variants
Nr instances	n	Speed, Scalability (Michie et al., 1994)	p/n, $log(n)$, log(n/p)
Nr features	p	Curse of dimensionality (Michie et al., 1994)	$log(p)$, % categorical
Nr classes	c	Complexity, imbalance (Michie et al., 1994)	ratio min/maj class
Nr missing values	m	Imputation effects (Kalousis, 2002)	% missing
Nr outliers	o	Data noisiness (Rousseeuw and Hubert, 2011)	o/n
Skewness	$\frac{E(X-\mu_X)^3}{\sigma_X^3}$	Feature normality (Michie et al., 1994)	min,max,μ,σ,q_1,q_3
Kurtosis	$\frac{E(X-\mu_X)^4}{\sigma_X^4}$	Feature normality (Michie et al., 1994)	min,max,μ,σ,q_1,q_3
Correlation	$\rho_{X_1X_2}$	Feature interdependence (Michie et al., 1994)	min,max,μ,σ,ρ_{XY}
Covariance	$cov_{X_1X_2}$	Feature interdependence (Michie et al., 1994)	min,max,μ,σ,cov_{XY}
Concentration	$\tau_{X_1X_2}$	Feature interdependence (Kalousis and Hilario, 2001)	min,max,μ,σ,τ_{XY}
Sparsity	sparsity(X)	Degree of discreteness (Salama et al., 2013)	min,max,μ,σ
Gravity	gravity(X)	Inter-class dispersion (Ali and Smith-Miles, 2006a)	
ANOVA p-value	$p_{val_{\mathtt{X}_1X_2}}$	Feature redundancy (Kalousis, 2002)	$p_{val_{XY}}$ (Soares et al., 2004)
Coeff. of variation	$\frac{\sigma_Y}{\mu_Y}$	Variation in target (Soares et al., 2004)	
PCA ρ_{λ_1}	$\sqrt{\frac{\lambda_1}{1+\lambda_1}}$	Variance in first PC (Michie et al., 1994)	$\frac{\lambda_1}{\sum_i \lambda_i}$ (Michie et al., 1994)
PCA skewness		Skewness of first PC (Feurer et al., 2014)	PCA kurtosis
PCA 95%	$\frac{dim_{95\%var}}{p}$	Intrinsic dimensionality (Bardenet et al., 2013)	
Class probability	$P(\mathtt{C})$	Class distribution (Michie et al., 1994)	min,max,μ,σ
Class entropy	$H(\mathtt{C})$	Class imbalance (Michie et al., 1994)	
Norm. entropy	$\frac{H(\mathtt{X})}{log_2 n}$	Feature informativeness (Castiello et al., 2005)	min,max,μ,σ
Mutual inform.	$MI(\mathtt{C},\mathtt{X})$	Feature importance (Michie et al., 1994)	min,max,μ,σ
Uncertainty coeff.	$\frac{MI(\mathtt{C},\mathtt{X})}{H(\mathtt{C})}$	Feature importance (Agresti, 2002)	min,max,μ,σ
Equiv. nr. feats	$\frac{H(C)}{MI(C,X)}$	Intrinsic dimensionality (Michie et al., 1994)	
Noise-signal ratio	$\frac{H(X)-MI(C,X)}{MI(C,X)}$	Noisiness of data (Michie et al., 1994)	
Fisher's discrimin.	$\frac{(\mu_{c1}-\mu_{c2})^2}{\sigma_{c1}^2-\sigma_{c2}^2}$	Separability classes c_1, c_2 (Ho and Basu, 2002)	See Ho:2002
Volume of overlap		Class distribution overlap (Ho and Basu, 2002)	See Ho and Basu (2002)
Concept variation		Task complexity (Vilalta and Drissi, 2002)	See Vilalta (1999)
Data consistency		Data quality (Köpf and Iglezakis, 2002)	See Köpf and Iglezakis (2002)
Nr nodes, leaves	$\|\eta\|, \|\psi\|$	Concept complexity (Peng et al., 2002)	Tree depth
Branch length		Concept complexity (Peng et al., 2002)	min,max,μ,σ
Nodes per feature	$\|\eta_X\|$	Feature importance (Peng et al., 2002)	min,max,μ,σ
Leaves per class	$\frac{\|\psi_c\|}{\|\psi\|}$	Class complexity (Filchenkov and Pendryak, 2015)	min,max,μ,σ
Leaves agreement	$\frac{n_{\psi_i}}{n}$	Class separability (Bensusan et al., 2000)	min,max,μ,σ
Information gain		Feature importance (Bensusan et al., 2000)	min,max,μ,σ, gini
Landmarker(1NN)	$P(\theta_{1NN}, t_j)$	Data sparsity (Pfahringer et al., 2000)	See Pfahringer et al. (2000)
Landmarker(Tree)	$P(\theta_{Tree}, t_j)$	Data separability (Pfahringer et al., 2000)	Stump,RandomTree
Landmarker(Lin)	$P(\theta_{Lin}, t_j)$	Linear separability (Pfahringer et al., 2000)	Lin.Discriminant
Landmarker(NB)	$P(\theta_{NB}, t_j)$	Feature independence (Pfahringer et al., 2000)	See Ler et al. (2005)
Relative LM	$P_{a,j} - P_{b,j}$	Probing performance (Fürnkranz and Petrak, 2001)	
Subsample LM	$P(\theta_i, t_j, s_t)$	Probing performance (Soares et al., 2001)	

AutoML Systems

- Bayesian optimization, hyperparameter tuning:
  Autosklearn (Feurer et al, NIPS 2015), AutoWEKA (Kotthoff et al, JMLR 2017)
- Tree search of algorithms and hyperparameters, multi-armed bandit
  Auto-Tuned Models (Swearingen et al, Big Data 2017)
- Deep reinforcement learning: expert iteration
  AlphaD3M (Drori et al, AutoML 2018)
- Evolutionary algorithms
  - TPOT (Olson et al, ICML 2016) machine learning pipelines as trees
  - Autostacker (Chen et al, GECCO 2018) ML pipelines as stacked layers.
- Collaborative filtering: OBOE, Yang et al, 2018.
- Neural architecture search: AutoKeras: Jin et al, 2018.
- Stacking, ensembles: GluonAutoML, GCP-Tables, H2O

AlphaD3M Single Player Game Representation

- Expert iteration: iterative improvement



	AlphaZero	AlphaD3M
Game	Go, chess	AutoML
Unit	piece	pipeline primitive
State	configuration	meta data, task, pipeline
Action	move	insert, delete, replace
Reward	win, lose, draw	pipeline performance

Figure source: AlphaD3M: Machine Learning Pipeline Synthesis, Drori et al, 2018.

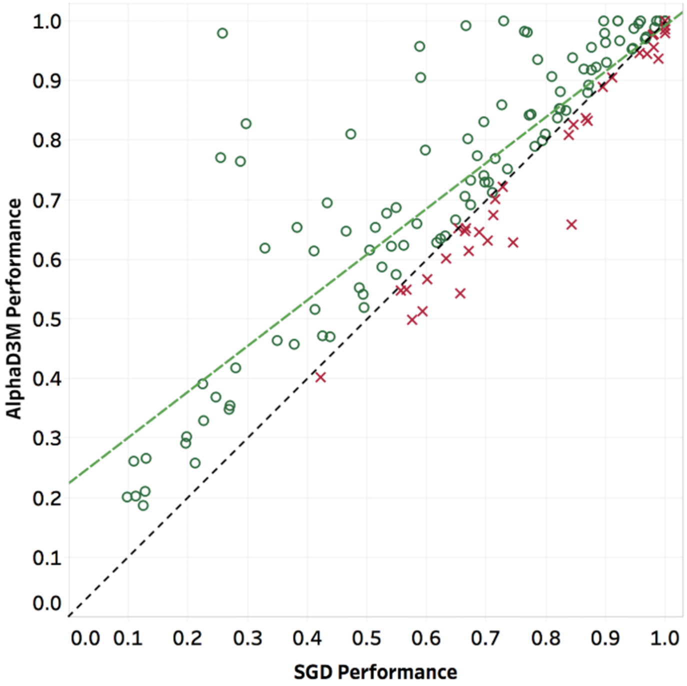
Pipeline Encoding

- Model meta-data, task, and entire pipeline chain as state rather than individual primitive

Given datasets D, tasks T, and a set of possible pipeline sequences $S_1, \ldots, S_n$, from the available machine learning, and data pre and post processing primitives.

- For each dataset D_i and task T_j:
  1. Encode dataset D_i as meta data features $f(D_i)$.
  2. Encode task T_j.
  3. Encode the current pipeline at time t by a vector S_t.
  4. Encode action $f_a(S_t)$, so policy π maps $(f(D_i), T_j, S_t)$ to $f_a(S_1), \ldots, f_a(S_n)$.

AutoML



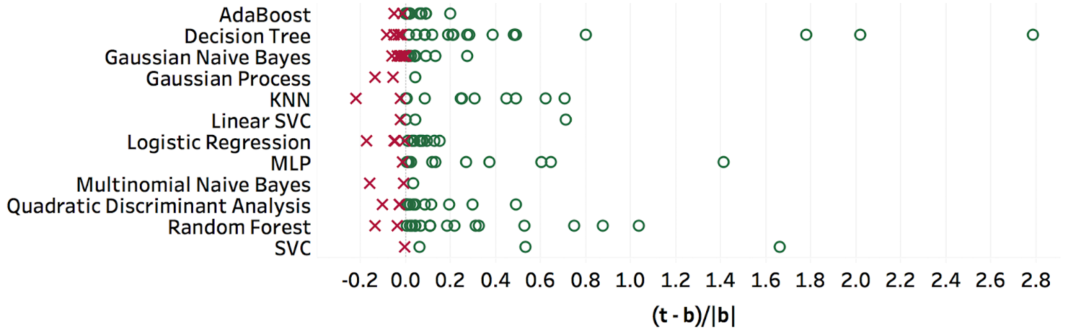


Figure source: AlphaD3M: Machine Learning Pipeline Synthesis, Drori et al, 2018.

Massachusetts Institute of Technology

AutoML by Pipeline Synthesis

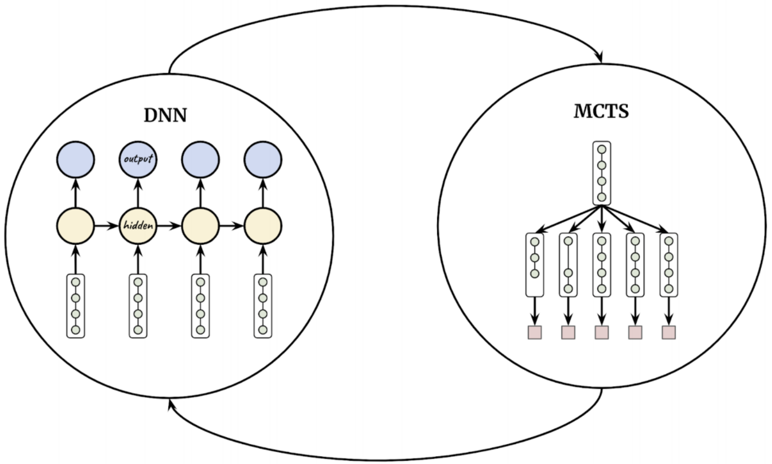
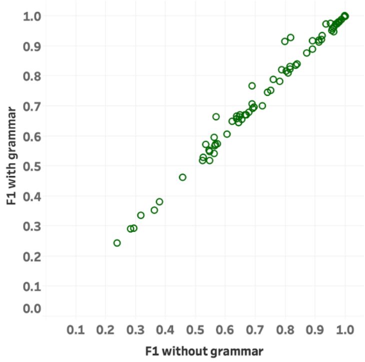


Figure 1: Architecture: the neural network sequence model (left) receives an entire pipeline, meta features, and task as input. The network estimates action probabilities and pipeline evaluations. The MCTS (right) uses the network estimates to guide simulations which terminate at actual pipeline evaluations (square leaf nodes).

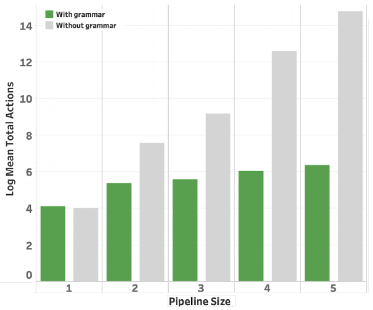
Figure source: Automatic Machine Learning by Pipeline Synthesis using Model-Based Reinforcement Learning and a Grammar, Drori et al 2019.

AutoML by Pipeline Synthesis

- Enforce a grammar on the solution, accept valid pipelines.



(a)



(b)

Figure 2: (a) Comparison of performance with and without using a pipeline grammar: Each point represents an OpenML dataset. Performance is not degraded even though computation time is reduced. (b) Comparison of log mean total actions with and without a grammar.

Figure source: Automatic Machine Learning by Pipeline Synthesis using Model-Based Reinforcement Learning and a Grammar, Drori et al 2019.

AutoML by Pipeline Synthesis

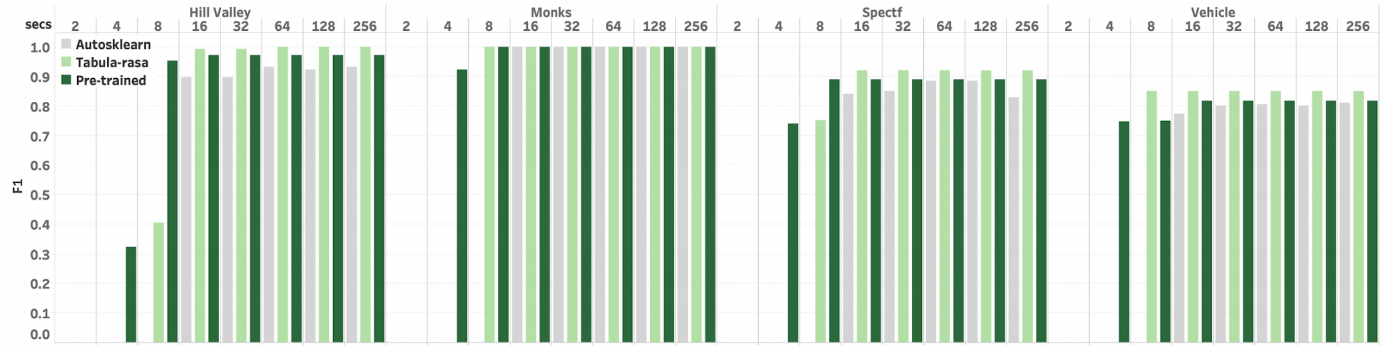


Figure 3: Performance-time comparison between (i) AlphaD3M using a pre-trained model and a grammar, (ii) model trained tabula rasa and a grammar, (iii) AutoSklearn. All methods (including brute force) perform comparably given sufficient time using same primitives, the difference is in performance given equal times. Method (i) is faster than (ii) which in turn is faster than (iii). Performance is F1 and time is in seconds on an exponential scale.

Figure source: Automatic Machine Learning by Pipeline Synthesis using Model-Based Reinforcement Learning and a Grammar, Drori et al 2019.

AutoML by Pipeline Synthesis

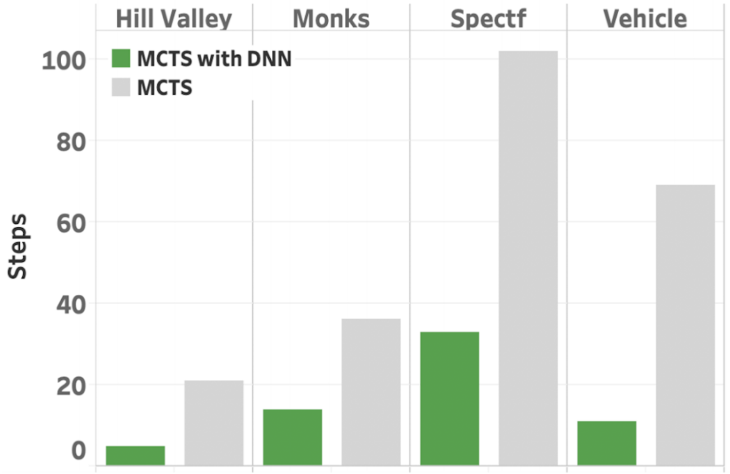


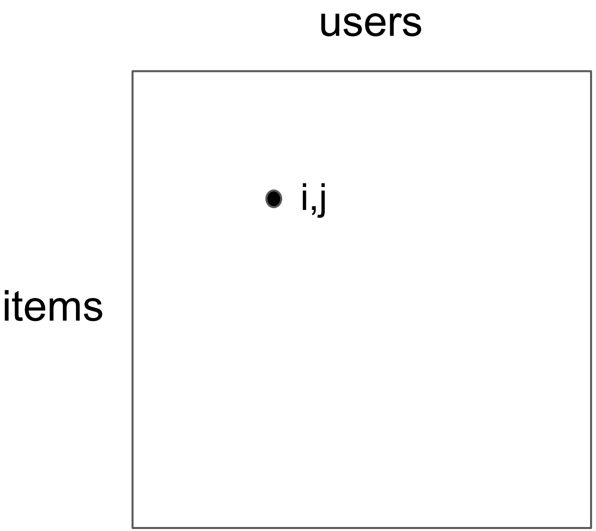
Figure 4: Ablation comparison between number of steps of MCTS with NN vs. MCTS only.

Figure source: Automatic Machine Learning by Pipeline Synthesis using Model-Based Reinforcement Learning and a Grammar, Drori et al 2019.

Matrix Completion: Example Problem

- Items $i=1..n$
- Users $j=1..p$
- Ratings Y_{ij}
- Binary mask if rating is available M_{ij}

- Problem: matrix completion



Massachusetts Institute of Technology

Naive Solution

- If k dimensional feature vectors $x^{(i)}$ are known for each item $i=1..n$
- Learn parameter vectors $\theta^{(j)}$ for each user $j=1..p$
- Predict user j rating item i by $\theta^{(j)^T} x^{(i)}$

$$\underset{\theta^{(j)}}{\text{minimize}} \frac{1}{2} \sum_{i:M_{ij}=1} \left(\theta^{(j)^T} x^{(i)} - Y_{ij}\right)^2 + \frac{\lambda}{2} \sum_{k} \theta^{(j)^2}$$

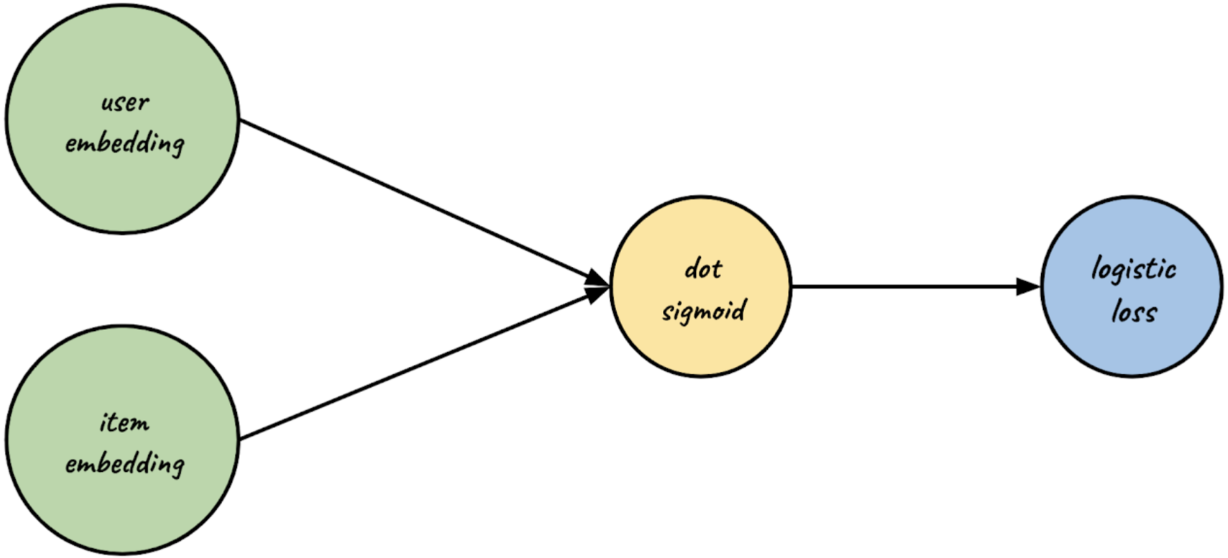
Naive Solution

- If k dimensional feature vectors $x^{(i)}$ are known for each item $i=1..n$
- Learn parameter vectors $\theta^{(j)}$ for **all users** $j=1..p$ using gradient descent
- Predict user j rating item i by $\theta^{(j)^T} x^{(i)}$

$$\underset{\theta^{(1)}\ldots\theta^{(p)}}{\text{minimize}} \frac{1}{2} \sum_{i,j:M_{ij}=1} \left(\theta^{(j)^T} x^{(i)} - Y_{ij}\right)^2 + \frac{\lambda}{2} \sum_{j} \sum_{k} \theta^{(j)^2}$$

Content-Based Recommendation

- Learn feature embeddings
- Represent image, text, audio using embedding



Problem

- If k dimensional feature vectors $x^{(i)}$ are unknown
- Given parameter vectors $\theta^{(j)}$ for all users $j=1..p$ learn feature vectors $x^{(i)}$

$$\underset{x^{(1)}...x^{(n)}}{\text{minimize}} \frac{1}{2} \sum_{i,j:M_{ij}=1} \left({\theta^{(j)}}^T x^{(i)} - Y_{ij}\right)^2 + \frac{\lambda}{2} \sum_i \sum_k {x_k^{(i)}}^2$$

Iterative Solution

- Given $\{x^{(1)}, \ldots, x^{(n)}\}$ learn $\{\theta^{(1)}, \ldots, \theta^{(p)}\}$

- Given $\{\theta^{(1)}, \ldots, \theta^{(p)}\}$ learn $\{x^{(1)}, \ldots, x^{(n)}\}$

Collaborative Filtering Solution

- Learn $\{x^{(1)}, \dots, x^{(n)}\}$ and $\{\theta^{(1)}, \dots, \theta^{(p)}\}$ together

$$\underset{x^{(1)} \dots x^{(n)}, \theta^{(1)} \dots \theta^{(p)}}{\text{minimize}} \frac{1}{2} \sum_{i,j:M_{ij}=1} \left({\theta^{(j)}}^T x^{(i)} - Y_{ij} \right)^2 + \frac{\lambda}{2} \sum_i \sum_k {x_k^{(i)}}^2 + \frac{\lambda}{2} \sum_j \sum_k {\theta_k^{(j)}}^2$$

- Predictions ${\theta^{(j)}}^T x^{(i)}$

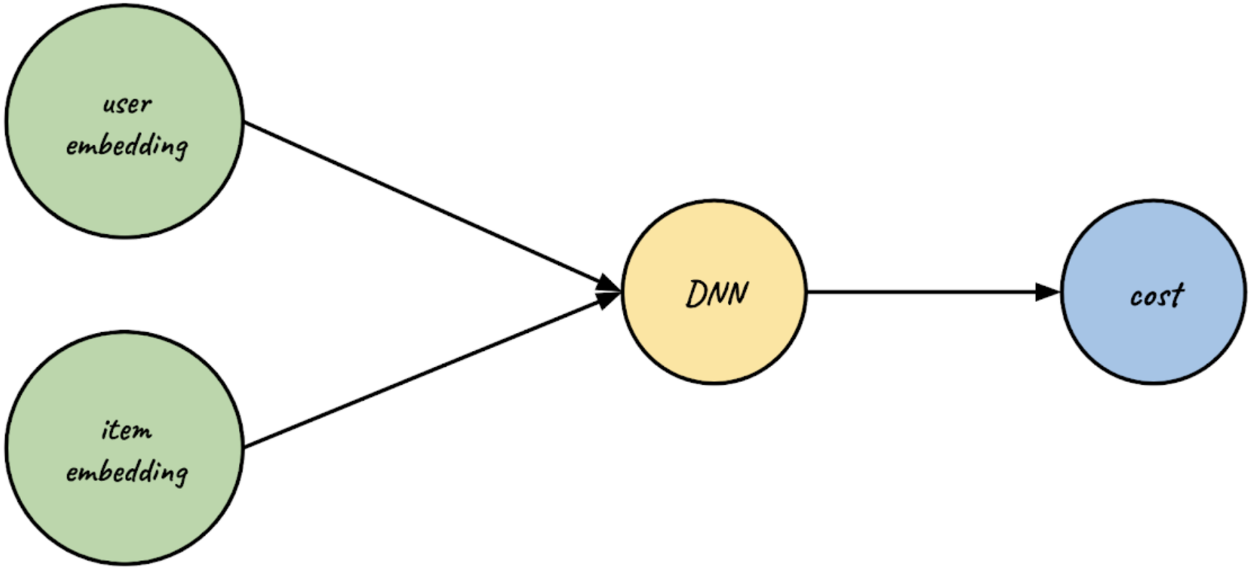
Collaborative Filtering Solution

- Low rank factorization of rating into product of feature matrix and parameter matrix

$$\begin{bmatrix} {\theta^{(1)}}^T x^{(1)} & \cdots & {\theta^{(p)}}^T x^{(1)} \\ \vdots & \ddots & \vdots \\ {\theta^{(1)}}^T x^{(n)} & \cdots & {\theta^{(p)}}^T x^{(n)} \end{bmatrix}$$

Neural Collaborative Filtering

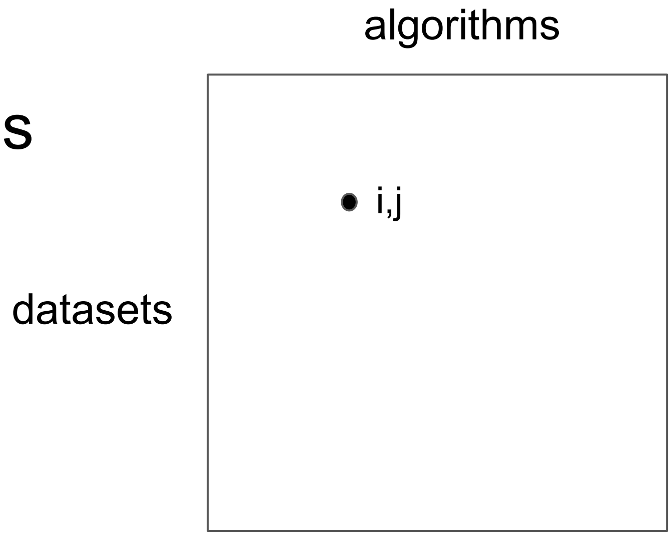
- Learn non-linear interactions between embedded features



Neural collaborative filtering, He et al. 2017

AutoML by Collaborative Filtering

- Factors are datasets and algorithms
- Serves as fast warm start systems



AutoML by Collaborative Filtering

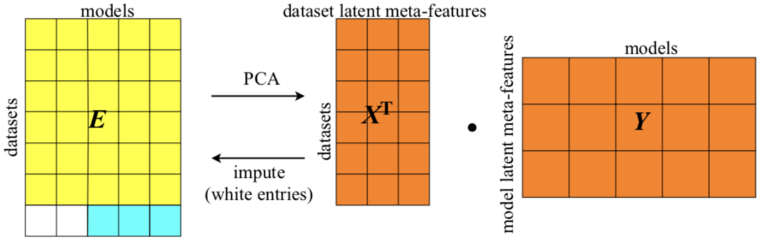


Figure 2: Illustration of model performance prediction via the error matrix E (yellow blocks only). Perform PCA on the error matrix (offline) to compute dataset (X) and model (Y) latent meta-features (orange blocks). Given a new dataset (row with white and blue blocks), pick a subset of models to observe (blue blocks). Use Y together with the observed models to impute the performance of the unobserved models on the new dataset (white blocks).

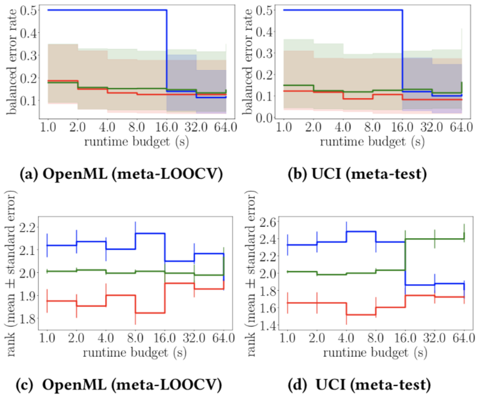


Figure 6: Comparison of AutoML systems in a time-constrained setting, including Oboe with experiment design (red), auto-sklearn (blue), and Oboe with time-constrained random initializations (green). OpenML and UCI denote midsize OpenML and UCI datasets. "meta-LOOCV" denotes leave-one-out cross-validation across datasets. In 6a and 6b, solid lines represent medians; shaded areas with corresponding colors represent the regions between 75th and 25th percentiles. Until the first time the system can produce a model, we classify every data point with the most common class label. Figures 6c and 6d show system rankings (1 is best and 3 is worst).

Figure source: Oboe: Collaborative Filtering for AutoML Model Selection, Yang et al, 2019.

Massachusetts Institute of Technology

AutoML using Metadata Embeddings

- When approaching an ML or DS problem humans read the documentation

- Description of data and task
- Description of machine learning functions available for solution

- How can we allow an AutoML method to “read the descriptions or manual”?

AutoML using Metadata Embeddings

- Humans read documentation

- Description of data and task
- Description of machine learning functions available for solution

- Use large scale transformer models to represent descriptions

AutoML using Metadata Embeddings

Notation	Description
$\mathcal{D}$	Dataset
$\mathcal{M}_\mathcal{D}$	Metadata of dataset $\mathcal{D}$
$\mathcal{T}$	Machine learning task (classification, regression)
$\mathcal{P}$	Solution pipeline
$\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{G}, \mathcal{H}$	OBOE, AutoSklearn, AlphaD3M, TPOT, and human algorithm
$\mathcal{P}_\mathcal{B}(\mathcal{D}, \mathcal{T})$ for $\mathcal{B} \in \{\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{G}, \mathcal{H}\}$	Solution pipeline on dataset $\mathcal{D}$ for task $\mathcal{T}$
$\mathcal{V}(\mathcal{P}_\mathcal{B}, \mathcal{D}, \mathcal{T})$	Evaluating performance of pipeline $\mathcal{P}_\mathcal{B}$ on $\mathcal{D}$ and $\mathcal{T}$
E	Pre-trained language embedding
$E(\mathcal{M}_\mathcal{D})$	Language embedding of dataset metadata
$d(E(\mathcal{M}_{\mathcal{D}_i}), E(\mathcal{M}_{\mathcal{D}_j}))$	Distance between dataset metadata embeddings
$\mathcal{D}_\star = \underset{\mathcal{D}_i}{\text{argmin}} \|E(\mathcal{M}_D), E(\mathcal{M}_{D_i})\|$	Nearest neighbor of $\mathcal{D}$ under distance of embeddings
$\mathcal{P}_\star = \mathcal{P}(\mathcal{D}_\star, \mathcal{T})$	Pipeline of most similar embedding
$\mathcal{V}(\mathcal{P}_\star, \mathcal{D}, \mathcal{T})$	Direct pipeline transfer using dataset metadata embedding
$E(\mathcal{P}_\mathcal{B}(\mathcal{D}, \mathcal{T}))$	Language embedding of solution pipeline
$\mathcal{X}(\mathcal{D}, \mathcal{T})$	Representation of embeddings for dataset $\mathcal{D}$ and task $\mathcal{T}$
Interaction between embeddings	
$\mathcal{I}(\mathcal{D}_i, \mathcal{D}_j) = (\mathcal{X}(\mathcal{D}_i, T), \mathcal{X}(\mathcal{D}_j, \mathcal{T}))$	Neural network input: pair of representations $\mathcal{X}(\mathcal{D}, \mathcal{T})$
$\mathcal{O}(\mathcal{D}_i, \mathcal{D}_j) = d(E(\mathcal{P}_\mathcal{H}(\mathcal{D}_i, \mathcal{T})), E(\mathcal{P}_\mathcal{H}(\mathcal{D}_j, \mathcal{T}))$	Network output: distance between human pipeline embeddings

Table 1: Embedding AutoML notation and their descriptions.

Figure source: AutoML using metadata language embeddings, Drori et al 2019.

AutoML using Metadata Embeddings

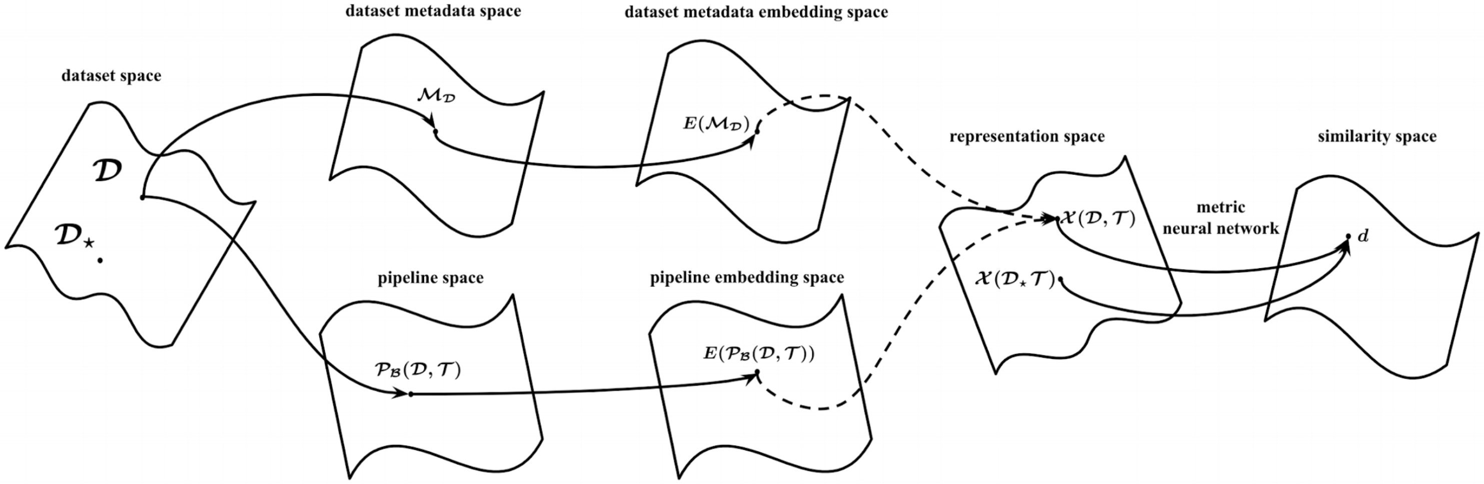


Figure 1: AutoML embeddings of dataset metadata and pipelines. The dashed arrows denote that the representation may consist of any number of the embeddings.

Figure source: AutoML using metadata language embeddings, Drori et al 2019.

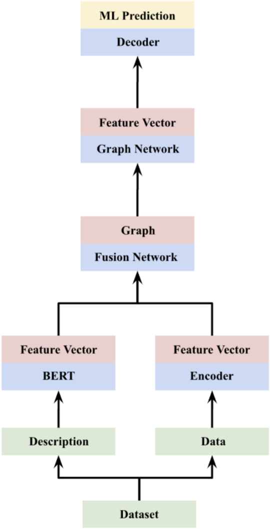
AutoML using Metadata Embeddings

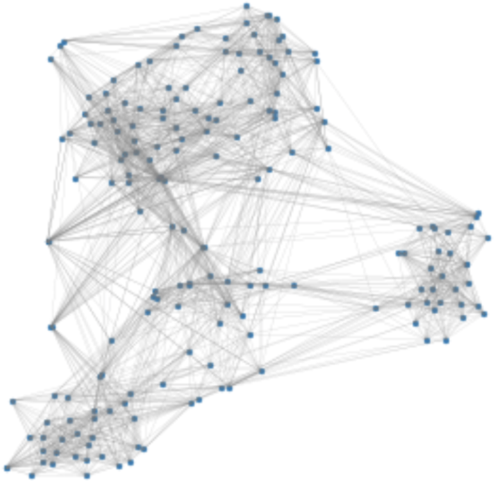
Dataset	OBOE	AutoSklearn	AlphaD3M	TPOT	Human	**Ours DE**	**Ours PE**
Seattle	**1.00**	0.66	**1.00**	**1.00**	0.92	**1.00**	**1.00**
Insurance	0.47	0.50	0.35	0.51	0.48	0.47	**0.52**
Forest	0.73	0.83	0.83	0.84	0.84	**0.85**	**0.85**
Credit	0.93	**0.94**	0.93	**0.94**	**0.94**	**0.94**	**0.94**
Titanic	0.70	0.80	0.70	0.77	0.86	**0.87**	**0.87**
HR	0.84	0.83	0.87	**0.90**	0.86	0.86	**0.90**
Kobe	0.61	0.60	**0.64**	0.62	0.62	0.61	0.62
Patients	0.64	0.71	**0.72**	0.68	0.68	0.68	0.69

Table 2: Machine learning pipeline evaluations for AutoML systems and human pipelines. All AutoML pipelines $\mathcal{P}_\mathcal{O}, \mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{A}, \mathcal{P}_\mathcal{G}$ are computed given 1 *minute* of computation. In comparison, ours DE refers to $E(\mathcal{M}_\mathcal{D})$ using only the dataset metadata embedding in under 1 *second* of computation for zero-shot AutoML. Ours PE refers to $E(\mathcal{P}_\mathcal{B})$ using the best single pipeline embedding $\mathcal{B} \in \{\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{G}\}$ in 1 minute of computation.

Figure source: AutoML using metadata language embeddings, Drori et al 2019.

AutoML using a Dataset Graph Representation





- Node i: dataset
- Edge (i,j) between datasets: based on embedding of dataset description and meta-features
- Predicts machine learning pipeline for new dataset in real-time (ms), runs pipeline and tuning in seconds
- GNN on graph of datasets sharing information between different datasets
- Embeddings of dataset descriptions and algorithm descriptions
- Leveraging existing AutoML systems

Figure source: Zero-shot AutoML, Drori et al 2020.

Zero-Shot AutoML

Notation	Description
$\mathcal{D}$	Dataset
$\mathcal{M}(\mathcal{D})$	Dataset description
$\mathcal{P}$	Machine learning pipeline
$\mathcal{M}(\mathcal{P})$	Machine learning pipeline description
C $\in$ O, S, A, T	OBOE, AutoSklearn, AlphaD3M, TPOT
$\mathcal{P}_{\mathrm{C}}(\mathcal{D})$	Pipeline recommended by C on dataset $\mathcal{D}$
$\mathcal{P}_{\star}(\mathcal{D})$	Best pipeline on dataset $\mathcal{D}$
$\hat{\mathcal{P}}(\mathcal{D})$	Predicted pipeline on dataset $\mathcal{D}$
$\mathcal{R}(\mathcal{P}, \mathcal{D})$	Performance of running pipeline $\mathcal{P}$ on dataset $\mathcal{D}$
$\mathcal{F}_{\mathcal{D}}$	Data meta-features
$\mathcal{F}_{\mathcal{M}} = E_B(\mathcal{M}(\mathcal{D}))$	Embedding of dataset description
$\mathcal{F}_{\mathcal{D},\mathcal{M}} = [\mathcal{F}_{\mathcal{D}}, \mathcal{F}_{\mathcal{M}}]$	Concatenation
$\mathcal{F}_{\mathcal{P}} = E_B(\mathcal{M}(\mathcal{P}))$	Embedding of pipeline description
$\mathcal{G} = (V, E)$	Datasets graph
$i \in V$	Node in $\mathcal{G}$
$j \in \mathcal{N}(i)$	Neighbors j of node i
$\mathcal{F}_i = f_\phi(\mathcal{F}_{\mathcal{D}_i,\mathcal{M}_i})$	Fusion network output on graph node
$\mathbf{v}_i = [\mathcal{F}_i, \mathcal{F}_{\mathcal{P}_\star(\mathcal{D}_i)}]$	Features of node in $\mathcal{G}$
$\mathbf{u}_i = g_\theta(\mathbf{v}_i)$	Fusion network, features of node in GNN
$\{\mathbf{u}_j\}_{j\in\mathcal{N}(i)}$	Features of node neighbors in GNN
$h_{W,z}(\mathbf{u}_i, \{\mathbf{u}_j\}_{j\in\mathcal{N}(i)})$	GNN with parameters W, z

Algorithm 1 Zero-shot AutoML pre-processing

Input: training datasets $\{(\mathcal{D}_i, \mathcal{M}_i)\}_{i\in V}$.
Output: features $\{\mathcal{F}_{\mathcal{M}i}, \mathcal{F}_{\mathcal{D}i}, \mathcal{F}_{\mathcal{P}_\star(\mathcal{D}_i)}\}_{i\in V}$.
for $i = 1$ **to** n **do**
  compute embedding of description $\mathcal{F}_{\mathcal{M}i} = E_B(\mathcal{M}_i)$
  compute data meta-features $\mathcal{F}_{\mathcal{D}i}$
  for all C $\in$ O, S, A, T **do**
    compute recommended pipeline $\mathcal{P}_{\mathrm{C}}(\mathcal{D}_i)$
    compute performance on dataset $\mathcal{R}(\mathcal{P}_{\mathrm{C}}, \mathcal{D}_i)$
  end for
  select best performing pipeline $\mathcal{P}_\star(\mathcal{D}_i)$
  embed pipeline $\mathcal{F}_{\mathcal{P}_\star(\mathcal{D}_i)} = E_B(\mathcal{M}(\mathcal{P}_\star(\mathcal{D}_i)))$
end for

Figure source: Zero-shot AutoML, Drori et al 2020.

Zero-Shot AutoML

Algorithm 2 Zero-shot AutoML training

Input: training datasets, descriptions $\{\mathcal{D}_i, \mathcal{M}(\mathcal{D}_i)\}_{i \in V}$.
Output: datasets graph $\mathcal{G}$, GNN $h_{W,z}$, fusion networks f_ϕ and g_θ.
pre-process: compute $\{\mathcal{F}_{\mathcal{M}i}, \mathcal{F}_{\mathcal{D}i}, \mathcal{F}_{\mathcal{P}_\star(\mathcal{D}_i)}\}_{i \in V}$
initialize fusion networks weights ϕ, θ.
initialize GNN weights W, z.
for each backprop iteration **do**
  generate updated datasets graph $\mathcal{G}$:
  for $i = 1$ **to** n **do**
    compute fused representation $\mathcal{F}_i = f_\phi(\mathcal{F}_{\mathcal{D}_i, \mathcal{M}_i})$
  end for
  compute pairwise distances $d(\mathcal{F}_i, \mathcal{F}_j)_{i,j \in V}$
  for $i = 1$ **to** n **do**
    connect node i to k-NN nodes $\mathcal{N}(i)$
  end for
  select random node i in $\mathcal{G}$
  compute $\mathbf{u}_i = g_\theta(\mathcal{F}_i, \mathbf{0})$
  for all $j \neq i$ **do**
    compute $\mathbf{u}_j = g_\theta(\mathcal{F}_j, \mathcal{F}_{\mathcal{P}_\star(\mathcal{D}_j)})$
  end for
  predict best pipeline $\hat{\mathcal{P}}(\mathcal{D}_i) = h_{W,z}(\mathbf{u}_i, \{\mathbf{u}_j\}_{j \in \mathcal{N}(i)})$
  compute loss $\mathcal{L}(\hat{\mathcal{P}}(\mathcal{D}_i), \mathcal{P}_\star(\mathcal{D}_i))$
  update weights
end for

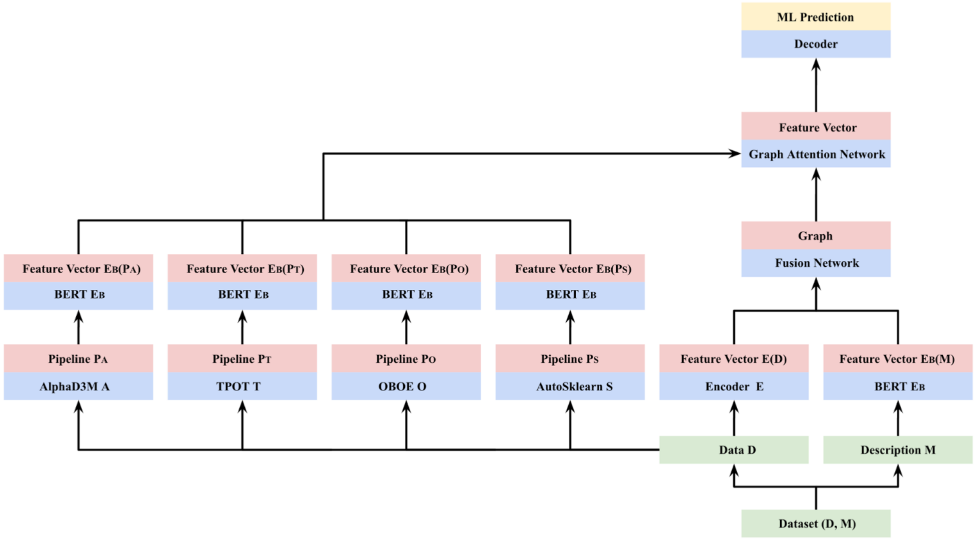


Figure 3. Zero-shot AutoML training architecture: Dataset descriptions are embedded using a language model. The data itself is passed through a feature extractor. Other AutoML system algorithms are embedded using a language model. Fully connected neural networks fuse together the encoded feature vectors. A graph captures the relationships between the embedded representations. A GNN learns the aggregation of each node in the graph and its neighbors. The GNN predicts a pipeline for a new node (dataset).

Figure source: Zero-shot AutoML, Drori et al 2020.

Zero-Shot AutoML

Algorithm 3 Zero-shot AutoML testing

Input: dataset $\mathcal{D}_i$, description $\mathcal{M}(\mathcal{D}_i)$, datasets graph $\mathcal{G} = (V, E)$, GNN, s.t. $i \notin V$ (disjoint train and test).
Output: predict best pipeline $\hat{\mathcal{P}}(\mathcal{D}_i)$ for task on dataset.
generate new node i in $\mathcal{G}$:
compute embedding of description $\mathcal{F}_{\mathcal{M}} = E_B(\mathcal{M}(\mathcal{D}_i))$
compute data meta-features $\mathcal{F}_{\mathcal{D}}$
compute fused representation $\mathcal{F} = f_\phi(\mathcal{F}_{\mathcal{D}}, \mathcal{F}_{\mathcal{M}})$
connect node i to k-NN nodes $j \in \mathcal{N}(i)$, $V = V \cup \{i\}$.
compute $\mathbf{u}_i = g_\theta(\mathcal{F}, \mathbf{0})$
predict best pipeline $\hat{\mathcal{P}}(\mathcal{D}_i) = h_{W,z}(\mathbf{u}_i, \{\mathbf{u}_j\}_{j \in \mathcal{N}(i)})$

Figure source: Zero-shot AutoML, Drori et al 2020.

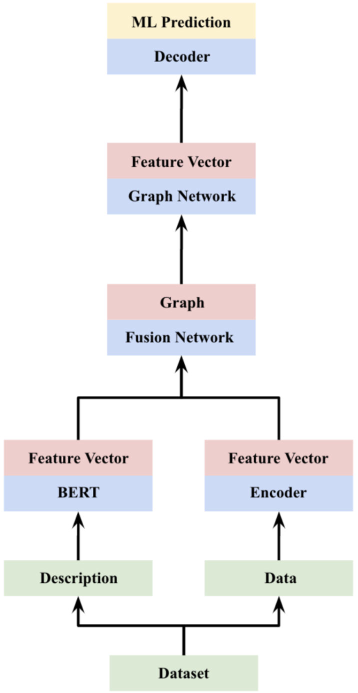


Figure 2. Zero-shot AutoML testing architecture: Dataset descriptions are embedded using BERT. Meta-features are extracted from the data. A fully connected neural network fuses together the embedded dataset description and data meta-features taking into account the non-linear interactions between them. The test dataset is added as a new node in a graph and a GNN predicts the best machine learning pipeline. Inputs are green, neural networks blue, intermediate outputs red, and final output yellow.

Massachusetts Institute of Technology

Meta Learning

MIT
Iddo Drori, Fall 2020