Neural Networks for Predicting Algorithm Runtime Distributions

Katharina Eggensperger
University of Freiburg
eggenspk@cs.uni-freiburg.de

Marius Lindauer
University of Freiburg
lindauer@cs.uni-freiburg.de

Frank Hutter
University of Freiburg
fh@cs.uni-freiburg.de

UNI FREIBURG

Problem

Algorithms often rely on **random choices and decisions**, hence their runtime can be described by a **runtime distribution (RTD)**. In this work we study how to **predict parametric RTDs** for unseen instances:

Given

- A randomized algorithm A
- A set of instances $\Pi_{train} = \{\pi_1, \ldots, \pi_n\}$
- For each instance $\pi \in \Pi_{train}$:
  - m instance features $\boldsymbol{f}(\pi) = [f(\pi)_1, \ldots, f(\pi)_m]$
  - runtime observations $\boldsymbol{t}(\pi) = \langle t(\pi)_1, \ldots, t(\pi)_k \rangle$ obtained by executing A on π with k different seeds,

the goal is to learn a model that can predict A's RTD well for unseen instances $\pi_{n+1}$with given features $f(\pi_{n+1})$

Runtime Distributions

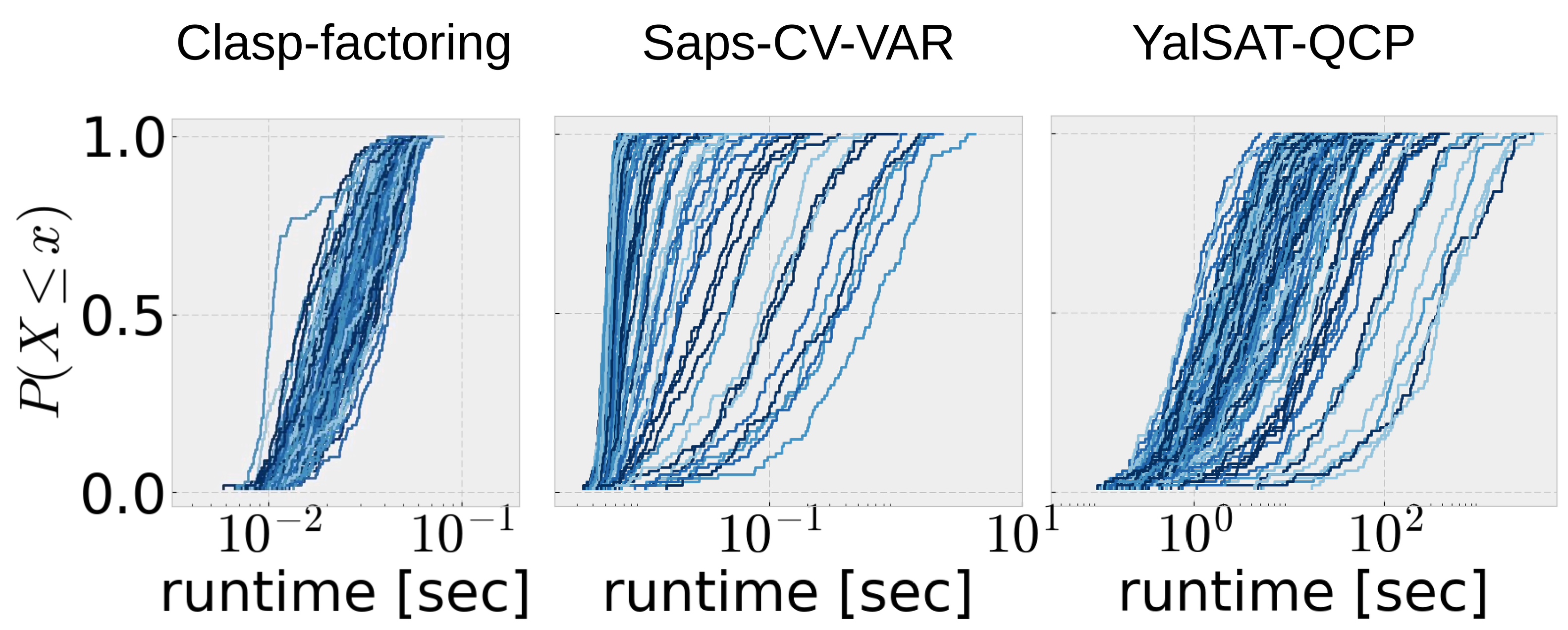


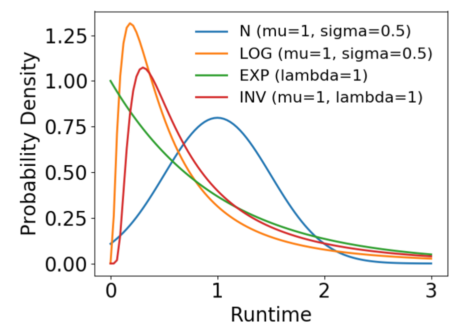
Figure: Empirical CDFs. We ran each algorithm 100 times with a different seed. Each line corresponds to one instance.

Scenario	#instances	#features	cutoff [sec]
Clasp-factoring[2]	2000	102	5000
Saps-CV-VAR[2]	10011	46	60
Spear-QCP[2]	8076	91	5000
YalSAT-QCP[2]	11747	91	5000
Spear-SWGCP[2]	11182	76	5000
YalSAT-SWGCP[2]	11182	76	5000
LPG-Zenotravel[3]	3999	165	300

Table 2: Characteristics of the used data sets.

Distribution	Param.	PDF
Normal (N)	μ, σ	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Lognormal (LOG)	s, σ	$\frac{1}{\sigma x \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\log(s)}{\sigma}\right)^2}$
Exponential (EXP)	β	$\frac{1}{\beta} e^{\left(-\frac{x}{\beta}\right)}$
Inverse Normal (INV)	μ, λ	$\left(\frac{\lambda}{2\pi x^3}\right)^{\frac{1}{2}} e^{\left(\frac{-\lambda(x-\mu)^2}{2x\mu^2}\right)}$

Table 1: Considered RTD families



Preprocessing:

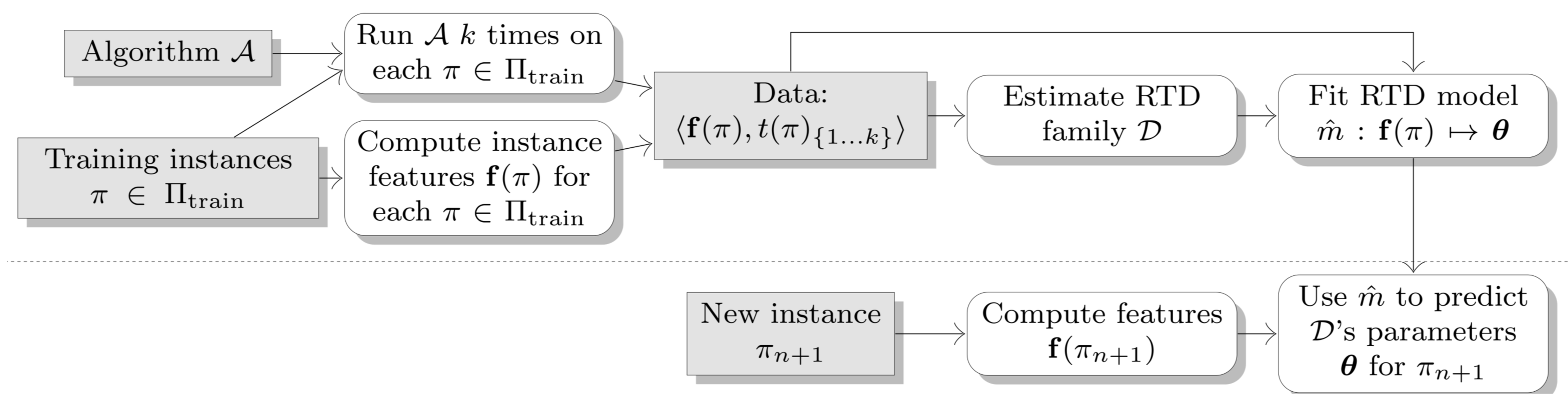
1. Remove all close to constant features
2. Impute missing values by the median
3. Scale observed runtimes by dividing by the maximal observed runtime across all instances

Architecture/Hyperparameters:
tanh activation, 2 hidden layer with 16 neurons each, L2 regularization of $1e^{-4}$, batch normalization, gradient clipping, SGD for training, learning rate exponentially decaying from $1e^{-3}$ to $1e^{-5}$, batch size of 16.

In a Nutshell

1. We compare different ways of predicting RTDs
2. We propose **DistNet**, which can be trained using the **loss function of interest** and **jointly predicts parameters of RTDs**
3. DistNet can be trained **with less data** than previous methods

Pipeline for predicting RTDs



Existing Approaches

For each instance π, fit the parametric distribution's parameters $\boldsymbol{\theta}(\pi) = (\theta_1, \ldots, \theta_p)$ on observed runtimes to get training data $\langle \boldsymbol{f}(\pi), \boldsymbol{\theta}(\pi) \rangle_{\pi \in \Pi_{train}}$

Option 1 Train p individual regression models

Option 2 Train a multi-output model with p outputs

But, these variants measure loss in the space of distribution parameters and not wrt. the loss function of interest: the negative log–likelihood.

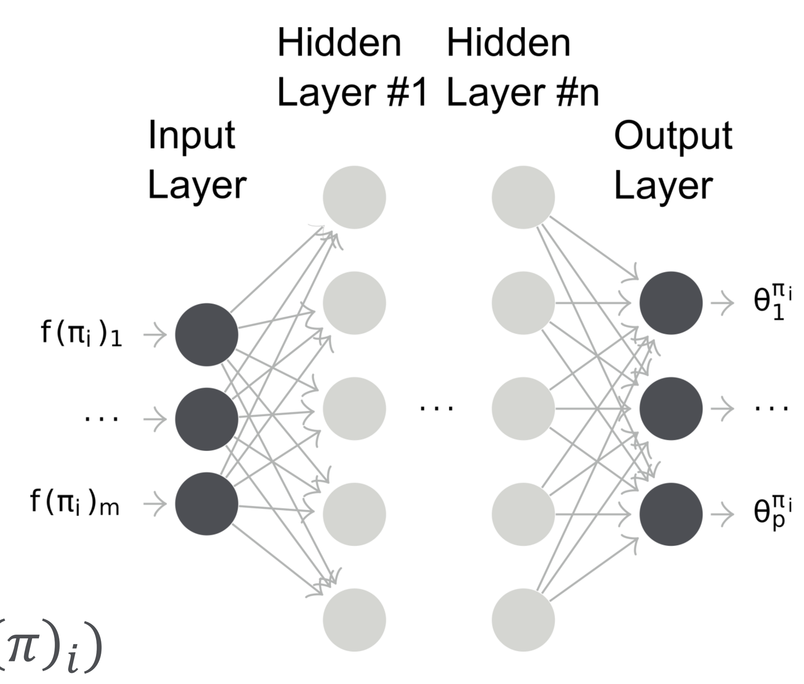
Our Approach

We use neural networks to predict the distribution parameters, e.g. with two hidden layers:

$$y = (g^{out}(g^{(2)}(g^{(1)}(xW^{(1)})W^{(2)})W^{(3)})$$

We use stochastic gradient descent to **directly optimize the negative log-likelihood** of the predicted parameters given our observed runtimes:

$$J(W) \propto - \sum_{\pi \in \Pi_{train}} \sum_{i=1}^{k} \log \mathcal{L}_D\left(\boldsymbol{\theta}_W | \boldsymbol{f}(\pi), t(\pi)_i\right)$$



Results

Scenario	dist		fitted	iRF	mRF	DistNet
Clasp-factoring	INV	train	-0.35	-0.26	**-0.28**	-0.24
		test	-0.35	-0.04	-0.09	**-0.16**
	LOG	train	-0.35	-0.30	**-0.30**	-0.24
		test	-0.35	-0.14	-0.13	**-0.14**
Saps-CV-VAR	LOG	train	-0.88	0.66	**-0.68**	-0.54
		test	-0.88	0.99	-0.29	**-0.52**
	INV	train	-0.88	-0.46	**-0.57**	-0.54
		test	-0.88	0.22	-0.09	**-0.54**
Spear-QCP	LOG	train	-1.20	-1.09	**-1.13**	-1.11
		test	-1.20	-1.00	-0.96	**-1.10**
	EXP	train	-1.14	**-1.05**	**-1.05**	-0.93
		test	-1.14	-0.88	-0.88	**-0.91**
YalSAT-QCP	LOG	train	-0.78	-0.50	**-0.77**	-0.76
		test	-0.78	-0.49	-0.74	**-0.75**
	INV	train	-0.78	-0.68	**-0.77**	-0.74
		test	-0.78	-0.66	-0.73	**-0.74**
Spear-SWGCP	LOG	train	-0.93	2.46	-0.23	**-0.48**
		test	-0.93	0.82	0.26	**-0.47**
	INV	train	-0.90	3.60	3.32	**-0.33**
		test	-0.90	3.27	2.58	**-0.32**
YalSAT-SWGCP	LOG	train	-0.94	-0.81	**-0.88**	-0.81
		test	-0.94	-0.69	-0.71	**-0.81**
	INV	train	-0.91	-0.80	**-0.86**	-0.76
		test	-0.91	-0.68	-0.69	**-0.76**
LPG-Zenotravel	LOG	train	-0.90	-0.89	**-0.89**	-0.85
		test	-0.90	-0.85	-0.84	**-0.85**
	INV	train	-0.90	-0.84	**-0.87**	-0.84
		test	-0.90	-0.72	-0.80	**-0.84**

Table: Averaged negative log-likelihood for predicting RTDs for unseen instances.

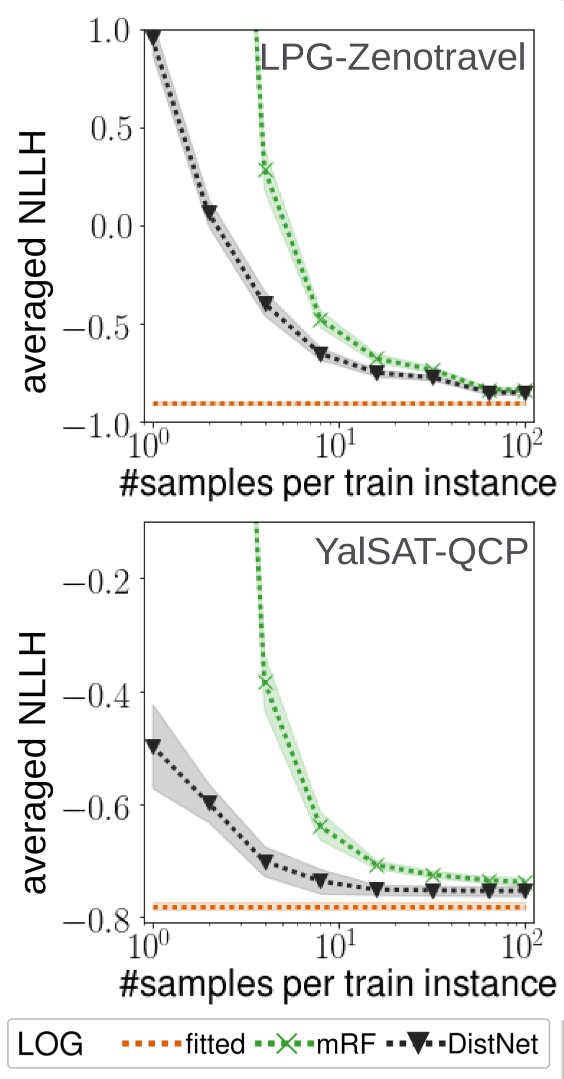


Figure: NLLH achieved on test instances wrt. to number of observed runtimes per instance used for training. We report the mean and standard deviation across 10-folds each of which avergare across 10 repetitions.

Advantages and Limitations

- DistNet **jointly learns distribution parameters** and **directly optimizes the loss function of interest**
- DistNet can learn **from only a few samples** per instance
- We assume **homogeneous instance sets**
- We need to **know beforehand** which distribution family to use

Open Questions & Future Work

- Use a mixture of models to handle less homogeneous instance sets
- Consider an algorithm's configuration as an additional input
- Study non-parametric models

AutoML.org

Data and Code Publicly Available: www.automl.org/distnet

@automlfreiburg