Sampling from Pre-Images to Learn Heuristic Functions for Classical Planning (Extended Abstract)

Stefan O'Toole[1], **Miquel Ramirez**[2], **Nir Lipovetzky** [1], **Adrian R. Pearce** [1]

[1] Computing and Information Systems, University of Melbourne, Australia
[2] Electrical and Electronic Engineering, University of Melbourne, Australia
stefan@student.unimelb.edu.au, {miquel.ramirez, nir.lipovetzky, adrianrp}@unimelb.edu.au

Abstract

We introduce a new algorithm, *Regression based Supervised Learning* (RSL), for learning per instance Neural Network (NN) defined heuristic functions for classical planning problems. RSL uses regression to select relevant sets of states at a range of different distances from the goal. RSL then formulates a Supervised Learning problem to obtain the parameters that define the NN heuristic, using the selected states labeled with exact or estimated distances to goal states. Our experimental study shows that RSL outperforms, in terms of coverage, previous classical planning NN heuristics functions while requiring a fraction of the training time.

Introduction

In this work, we introduce the *Regression based Supervised Learning* (RSL) algorithm in order to learn per instance NN defined heuristic functions. Like other methods (Ferber et al. 2021; Yu, Kuroiwa, and Fukunaga 2020) do, RSL selects a set of *regressions*, trajectories of *sets of states*, or *pre-images*, found via the application of well-known and efficient pre-imaging operators (Rintanen 2008) that rely on symbolic action descriptions. These trajectories found along a given regression, always start from the set of goal states of an instance, and then training states are sampled from each set along the trajectory. Our method takes many samples from each pre-image found in a regression, instead of performing many trajectories or longer regressions to increase the number of training states for the NN, using the observed goal distances for each pre-image to label the sampled states.

Regression Based Supervised Learning

Given a planning problem, which we consider using the STRIPS formulation $\Pi = \langle F, O, I, G\rangle$ (Fikes and Nilsson 1971), RSL produces a training set $\mathcal{D} = \{(s_1, h_1), \dots (s_N, h_N)\}$ which is a set of states $s \in S$ paired with goal distance estimates h. To produce $\mathcal{D}$ RSL performs N_r rollouts, each starting at the goal G and applying L times the classical planning regression operator. Each rollout from G is a sequence of actions $\pi^j = (a_i^j)_{i=0}^{L-1}$, for $j = 1, \dots, N_r$. Each π^j produces a sequence of pre-images $x_0^j, x_1^j, \dots, x_L^j$, where $x_0^j = G$ and $x_i^j \subseteq F$. The sequence of pre-images denotes a sequence of sets of states $\mathcal{R}^j = X_0^j, X_1^j, \dots, X_L^j$, where $X_i^j = \{s \mid x_i^j \subseteq s, s \in S\}$. By the definition of the regression operator, X_{i-1}^j corresponds with the *pre-image*, conditioned on a_{i-1}^j, of X_i^j. Therefore any state within X_i^j can be reached from X_{i-1}^j by applying action a_{i-1}^j. It follows that any state within X_i^j can reach X_0^j in at most i transitions. Using this observation, RSL labels each state s within its training set of states, T_s, with

$$d(s) = \min(i \mid s \in X_i^j, j \in \{1, \dots, N_r\}) \qquad (1)$$

that is, the smallest goal distance estimate of any of the state sets visited by the regressions which s is also a member of.

Algorithm 1, provides an overview of RSL. The hyper-parameters of RSL are the length of each regression L, the number of regressions to perform N_r, the number of training states to use N_t, the percentage of training states that are randomly sampled from the entire state space P_r, and a function that maps state regression trajectories into novelty levels μ. RSL has three distinct steps, 1: extracting sets of state sets, $\mathcal{R} = \bigcup_{j=1}^{N_r} \mathcal{R}^j$, through performing regressions from the goal set (line 1), 2: sampling training states T_s and labeling them with goal distance estimates using (1) (lines 2-3), and 3: training the NN heuristic function (line 4).

Algorithm 1: Overview of the RSL Algorithm

Input: Π
Parameter: P_r, L, N_r, N_t, μ
Output: h^{RSL}

1: $\mathcal{R} \leftarrow$ REGRESSION(Π, L, N_r, μ)
2: $T_s \leftarrow$ SAMPLE_STATES$(\mathcal{R}, P_r, N_t)$
3: $\mathcal{D} \leftarrow$ LABEL$(\mathcal{R}, \mathcal{T}_f)$
4: $h^{\text{RSL}} \leftarrow$ SUPERVISED_LEARNING$(\mathcal{D})$

Extracting State Sets through Regression

As previously explained RSL performs N_r regressions to extract the set of state sets $\mathcal{R}$ over which the training set $\mathcal{D}$ is defined. At step i of a rollout with a pre-image x_i^j corresponding to the set of states X_i^j, as previously defined, the actions we consider valid for pre-imaging X_i^j with are

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

$a \in \nu(x_i^j)$, defined as follows

$$\nu(x_i^j) = \{a \mid a \in \text{REACHABLE}(O, I), \\ x_i^j \cap \text{e-Del}(a) = \emptyset, \\ x_i^j \cap \text{Del}(a) = \emptyset, x_i^j \cap \text{Add(a)} \neq \emptyset\} \quad (2)$$

and e-Del(a) is

$$\text{e-Del}(a) = \{\, q \mid q \in F \setminus Add(a), \\ \exists p \in Pre(a) : \text{MUTEX}(p, q)\}. \quad (3)$$

REACHABLE(O, I) maps the operator set O and initial state I to the set of actions with reachable preconditions in the delete relaxation of Π (Bonet and Geffner 2001) given the initial state of the progression state-transition model I. The MUTEX(p, q) function in (3) maps the pair of atoms (p, q) to true if p and q are mutually exclusive, that is, it is impossible for p and q to both be true in any state $s \in S$ that can be reached from I. Note that the Ferber et al. (2021) algorithms that use regression also filter the valid actions in the same way through using the mutex groups and applicable operations found by the Fast Downward (FD) (Helmert 2006) Translator.

The baseline option for performing the rollout, and the method used by Ferber et al. (2021), is to randomly select actions a for which applying the regression operator is valid. In addition to testing RSL using random action selection we instantiate a version of RSL we name Novelty guided Regression based Learning (N-RSL) that aims to increase the structural diversity of operators selected in its regression. N-RSL does this by preferring actions with pre-conditions which contain atoms that are not specified in the goal G and are not a member of the pre-condition set of any of the actions executed in the trajectory up until that point.

Sampling and Labeling Training Data

The training data for RSL is sampled from the sets of states $\mathcal{R}$. States are sampled from each set $X_i^j \in \mathcal{R}$, and sampled states that contain mutex atom pairs (p, q) are modified by removing either the p or q atom from the state. Note that if a sampled state from the set X_i^j has a mutex pair (p, q) and $p \in x_i^j$, q will be removed from the state, that is, an atom that is a member of the partial state x_i^j that a state is sampled from will never be removed from the state. As previously described, the labelled heuristic value for a sampled state is given by $d(s)$, which returns the distance of the closest state set in $\mathcal{R}$ to the goal according to the regression.

Some domains benefited from adding randomly sampled states from S. The random states have mutexes enforced using the same method as above, and are also labeled with $d(s)$. In the case that the state is not a member of any of the sets of states in $\mathcal{R}$, we define $d(s)$ to equal $L + 1$.

Experimental Study

Our benchmark set of domains, instances and initial states is the "Hard Task" set as introduced by Ferber et al. (2021). As we are learning per instance heuristics, a unique heuristic is trained for each problem instance and then evaluated over a set of 50 different initial states.

Budget	**NN defined heuristics** 6 minutes		600 minutes*			6 minutes	
	h^{RSL}	$h^{\text{N-RSL}}$	h^{Boot}	h^{AVI}	h^{TSL}	h^{FF}	LAMA
blocks	36.4	**68.0**	0.0	0.0	0.0	46.4	96.0
depot	14.6	12.6	8.3	12.9	**35.4**	9.4	69.4
grid	67.2	69.0	**87.8**	70.5	60.2	31.0	100
npuzzle	0.0	0.0	0.0	0.0	0.0	5.8	12.8
pipesworld	33.8	33.1	23.4	8.0	**48.7**	20.4	68.0
rovers	0.1	0.1	2.8	**6.5**	1.5	8.3	100
scanalyzer	**100**	**100**	3.3	60.7	60.0	83.3	100
storage	1.8	0.2	**27.2**	15.8	0.0	9.2	9.0
transport	0.0	**8.8**	0.0	2.4	0.0	0.0	59.6
visitall	**100**	**100**	28.0	0.0	0.0	40.0	100
Average	35.4	**39.2**	18.1	17.7	20.6	25.4	71.5
Ave. Train (hours)	6.88	6.88	112*			-	

Table 1: Comparison of the coverage of h^{RSL} with other Neural Network defined heuristics functions introduced by Ferber et al. (2021) h^{Boot}, h^{AVI}, as well as the h^{TSL} introduced by Ferber et al. (2020). The table also shows the coverage of h^{ff}(Hoffmann and Nebel 2001) and LAMA (Richter and Westphal 2010) (run on same hardware as RSL). The bold numbers indicate the highest coverage among the Neural Network methods. *Note that the baseline learning methods that use a 600 minute planning budget are the values as reported by Ferber et al. (2021). According to standard single thread CPU benchmarks, our vCPU can be 20% faster.

We evaluate h^{RSL} heuristic using the same method as Ferber et al. (2021). Each heuristic is evaluated over 50 different initial states guiding GBFS implemented in FD (Helmert 2006). The coverage of a planner is defined as the percent of initial states for which a solution path is found within the given planning budget. Ferber et al. (2021) report observing that in general the coverage superiority between the different NN heuristics tested did not vary over time. That is, the planning time used and the relative coverage superiority between the algorithms were not correlated. Given this observation and constraints on our available compute resources we reduce the overall 10 hour planning time-limit that Ferber et al. used by 99% down to just 6 minutes, and compare with the NN defined heuristic baseline algorithms results as provided by Ferber et al. (2021) which use the 10 hour planning time-limit.

Table 1 shows a comparison of existing methods with respect to RSL and N-RSL, using the best performing configuration from a hyper-parameter grid search. The first notable difference is the average training times used by each algorithm. h^{RSL} uses around 6% of the training CPU time used by all the other per-instance NN defined heuristics functions. The overall average coverage over the benchmark set shows that h^{RSL} and $h^{\text{N-RSL}}$ outperform the other NN defined heuristic functions and the model-based heuristic h^{FF}. However, the other model-based method LAMA clearly dominates all other methods. While $h^{\text{N-RSL}}$ has better average coverage than h^{RSL} the difference is small with a 3.8% average improvement.

References

Bonet, B.; and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1): 5–33.

Ferber, P.; Geiber, F.; Trevizan, F.; Helmert, M.; and Hoffmann, J. 2021. Neural Network Heuristic Functions for Classical Planning: Reinforcement Learning and Comparison to Other Methods. In *2nd PRL Workshop, International Conference on Automated Planning and Scheduling*.

Ferber, P.; Helmert, M.; and Hoffmann, J. 2020. Neural network heuristics for classical planning: A study of hyperparameter space. In *European Conference on Artificial Intelligence*, 2346–2353.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4): 189–208.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26: 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Richter, S.; and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *European Conference on Artificial Intelligence*, 568–572.

Yu, L.; Kuroiwa, R.; and Fukunaga, A. 2020. Learning Search-Space Specific Heuristics Using Neural Network. *12th HSDIP Workshop, International Conference on Automated Planning and Scheduling*, 1–8.