Notes: Population Based Training of Neural Networks[1]

esgl Hu

December 11, 2017

1 Population Based Training

The most common formulation in machine learning is to optimise the parameters θ of a model f to maximise a given objective function $\hat{\mathcal{Q}}$, Generally, the trainable parameters θ are updated using an optimisation procedure such as stochasitc gradient descent. More importantly, the actual performance metric $\mathcal{Q}$ that we truely care to optimise is often different to $\hat{\mathcal{Q}}$.

We consider training N models $\{\theta^i\}_{i=1}^N$ forming a *population*$\mathcal{Q}$ which are optimised with different hyperparameters $\{\eqsim^i\}_{i=1}^N$, the object is to therefore find the optimal model across the entire population.

the function <u>*eval*</u> is used to evaluate the objective function, $\mathcal{Q}$, using the current state of model f. the process of finding the optimal set of parameters that maximise $\mathcal{Q}$ is then:

$$\theta^* = \arg\max_{\theta \in \Phi} eval(\theta) \tag{1}$$

the function <u>*step*</u> is used to update the parameters of the model, and is itself conditioned on some parameters $h \in \mathcal{H}$. iterations of paramters update steps:

$$\theta \leftarrow step(\theta|h) \tag{2}$$

the function <u>*exploit*</u>, which, given peformance of the whole population, can decide whether the worker should abandon the current solution and istead focus on a more promising one. For expamle, *exploit* could replace the current weights with the weights that have the highest recorded performance in the rest of the population.

the function <u>*explore*</u>, which given the current solution and hyperparameters proposes new ones to better explore the solution space. For example, *explore* could randomly perturb the hyperparamters with noise.

the member of population is deemed <u>*ready*</u> (for example. by having been optimised for a minimum number of steps or having reached a certain performance threshold).

The specific form of *exploit* and *explore* depends on the application. In this work we focus on optimising neural networks for reinforcement learning , supervised learning, and generative modelling with PBT. In these cases, *step* is a step of grdient descent, *eval* is the mean episodic return or validation set performance of the metric we aim to optimise, *exploit* selects another member of the population copy the weights and hyperparameters from, and *explore* creats new hyperparameters for the next steps of gradient-based learning by either perturbing the copied hyperparameters or resampling hyperparameters from the originally defined prior distribution. A member of the population is deemed *ready* to exploit and explore when it has been trained with gradient descent for a number of steps since the last change to the hyperparameters, such that the number of steps is large enough to allow signification gradient-based learning to have occurred.

2 Advantage of Population Based training

By combining multiple steps of gradient descent followed by weight copying by *exploit*, and perturbation of hyperparameters by *explore*, we obtain learning algorithms which benefit from not only local optimisation by gradient descent, but also periodic model selection, and hyperparameter refinement from a process that is more similar to genetic algorithms, creating a two-timescale learning system. An important property of population based training is that it is asynchronous and does not require a centralised process to orchestrate the training of the members of the population. Only the current performance information, weights, and hyperparamters must be globally available for each population memebr to access - crucially there is no synchronisation of th population required.

3 Algorithm

Algorithm 1 Population Based Training(PBT)

function TRAIN($\mathcal{P}$)
    for $(\theta, h, p, t) \in \mathcal{P}$ (asynchronously in parallel) **do**
        while not end of training **do**
            $\theta \leftarrow step(\theta|h)$
            $p \leftarrow eval(\theta)$
            if $ready(p, t, \mathcal{P})$ **then**
                $h', \theta' \leftarrow exploit(h, \theta, p, \mathcal{P})$
                if $\theta \neq \theta'$ **then**
                    $h, \theta \leftarrow explore(h', \theta', \mathcal{P})$
                    $p \leftarrow eval(\theta)$
                end if
            end if
            update $\mathcal{P}$ with new $(\theta, h, p, t+1)$
        end while
    end for
    return θ with the highest p in $\mathcal{P}$
end function

References

[1] Jaderberg Max, Dalibard Valentin, Osindero Simon, M.Czarnecki Wojciech, Donahue Jeff, Razavi Ali, Vinyals Oriol, Green Tim, Dunning Iain, Simonyan Karen, Fernando Chrisantha, and Kavukcuoglu Koray. Population based training of neural networks. 2017.