Machine Learning Ideas

Yeon-woo Shin

January 17, 2023

1 Auxiliary Task Suggestion for Path Efficiency of Reinforcement Learning Agent in Complex Maze

As pointed out in paper written by Jaderberg et al [1], the additional reward from auxiliary tasks may guide the agents through sparse reward space (i.e. finding an apple in the complex maze in which the reward is only received when apple is found). Auxiliary tasks, such as "pixel change" suggested by paper, can improve learning efficiency of agent by discouraging "state overlaps" to motivate exploration. Another task called "Network" is to directly maximize the activation values within action network to expedite the learning. However, both of these tasks doesn't address the path-inefficiency issue. Although "pixel change" may be able to prevent "loop" of agent's trajectory, it might not be able to resolve a large-scale loop. Thus, I suggest novel auxiliary task: avoid "checkpoints". The checkpoint will be installed on the point of the wall that agent is directly faced against at its arbitrary location. When the agent come backs to the location and faces the wall in the exactly same angle, it will detect a checkpoint and receive "penalty". During exploration, it's tolerable that the agent come back to same location, but if it's in the exactly same angle it was before, the trajectory of agent should be implied to be a "loop", which is not desirable path.

2 Improve PRISM by Modification of Clean Probability Formula

Original

$$P(i)_{clean} = \frac{exp(T(x_i, y_i))}{\sum_{k \in C} exp(T(x_i, k))} \tag{1}$$

$$T(x_i, k) = \frac{1}{M_k} \sum_{(v_j, y_j) \in M, y_j = k} S(f(x_i), v_j) \tag{2}$$

where $S(f(x_i), f(x_j))$ is cosine similarity between features of data point x_i and x_j

Modified

$$P(i)_{noised} = \frac{exp(1 - T(x_i, y_i) + \sum_{k \in C, k \neq y_i} T(x_i, k))}{\sum_{y \in C} exp(1 - T(x_i, y) + \sum_{k \in C, k \neq y} T(x_i, k))} \tag{3}$$

$$P(i)_{clean} = 1 - P(x_i, y_i)_{noised} \tag{4}$$

$$P(i)_{clean} = 1 - \frac{exp(1 - T(x_i, y_i) + \sum_{k \in C, k \neq y_i} T(x_i, k))}{\sum_{y \in C} exp(1 - T(x_i, y) + \sum_{k \in C, k \neq y} T(x_i, k))} \tag{5}$$

Explanation

A previously proposed formula to calculate the probability of data sample X labeled as class A truly being in class A accounts for comparison of how much the features of data point x labeled as class A is similar to the true features represented in A and how much features of x is similar to the true features of other classes. Simply, the formula compares feature similarities for different classes to compute the probability that the label of X is "clean", meaning truly being in a defined label. However, this formula doesn't follow the definition of the probability but is instead based on mere intuition. The probability is defined to be likelihood of explicitly defined scenarios. The probability of X labeled as A being clean should be "likelihood" of scenario that X labeled as A is clean. To estimate the likelihood, I propose novel method: Expected Similarity Disparity (ESD) shown in equation [3]. Expected Similarity Disparity calculates how much similarity between sample X and representative of samples in class A is varied from an expected value that occurs when sample X is truly in class A. The larger ESD is, more unlikely the true label of sample X is to be A.

3 VolumeNet: A reinforcement trading model that learns optimal volume of stocks to be traded

An action network will learn to decide to buy or sell under Q-learning policy as usual stock trading reinforcement learning model. After the action is determined, the action and state will be used as an input to predict the volume of stocks to be traded under chosen action in VolumeNet. Here is the derivation of backpropagation algorithm for VolumeNet.

Suppose T_i refers to current time when an agent trades stocks, and T_{i+a} refers to the next time an agent will execute a trade. Then, the return R can be formualized as following.

$$R = \frac{(T_{i+a} - T_i)}{T_i} = \frac{(T_i + V_p \Delta P) - T_i)}{T_i} = \frac{V_p \Delta P}{T_i} \tag{6}$$

$$\frac{\partial R}{\partial V_p} = \frac{\Delta P}{T_i} \tag{7}$$

where P is the change in prices at T_i and T_{i+a}, and V_p refers to volumes to be traded.

Since neural network predicts the volume of stocks V_p to be traded at state S_i, V_p can be expressed as

$$\sum_{i}^{i=L} \sum_{j}^{j=N} w_{ij} a_{ij} + b_i \tag{8}$$

A partial derivative of R in terms of w_{ij} can be derived as

$$\frac{\partial V_p}{\partial w_{ij}} = a_{ij} \tag{9}$$

$$\frac{\partial R}{\partial w_{ij}} = \frac{\partial R}{\partial V_p} * \frac{\partial V_p}{\partial w_{ij}} = \frac{\Delta P}{T_i} a_{ij} \tag{10}$$

A weight w_{ij} for V-Network can be updated as

$$w_{ij} := w_{ij} + \gamma \frac{\partial R}{\partial w_{ij}} = w_{ij} + \gamma \frac{\Delta P}{T_i} a_{ij} \tag{11}$$

where γ is learning rate.

4 Improve Code Generation Model by Integrating Inductive Bias

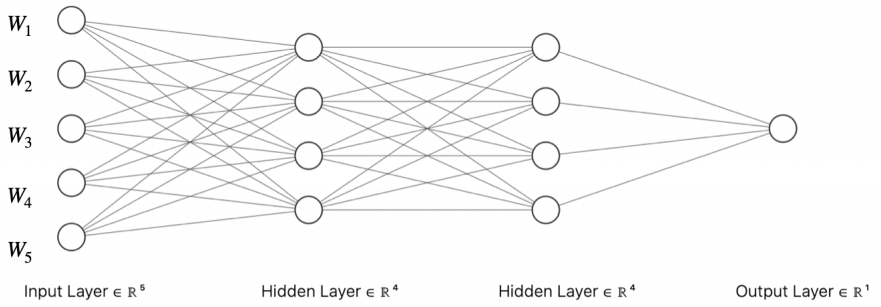

Figure 1: Inputs multiplied by pre-defined weights

As pointed out in DeepMind's paper in AlphaCode [2], the pre-trained model is insensitive to a slight change in problem description while even the change of a word may alter the entire direction of solution. To detect a slight change, first, a problem description that's the most similar to that of interest will be searched from existing training set. Then, two matching descriptions will be encoded to be input features for code generation model using pre-trained sentence encoder. Then, the encoded feature (input to code generation model) with the largest difference to the other encoded one encoded ones will be multiplied by inductive weight units $W = [w_n | n \in N]$ as shown in Figure 1. Inductive weight units will help the model to catch important distinctions of the description which are responsible for altering entire directions of solution and generate the optimal solution.

5 Algorithm for Discovery of Truth

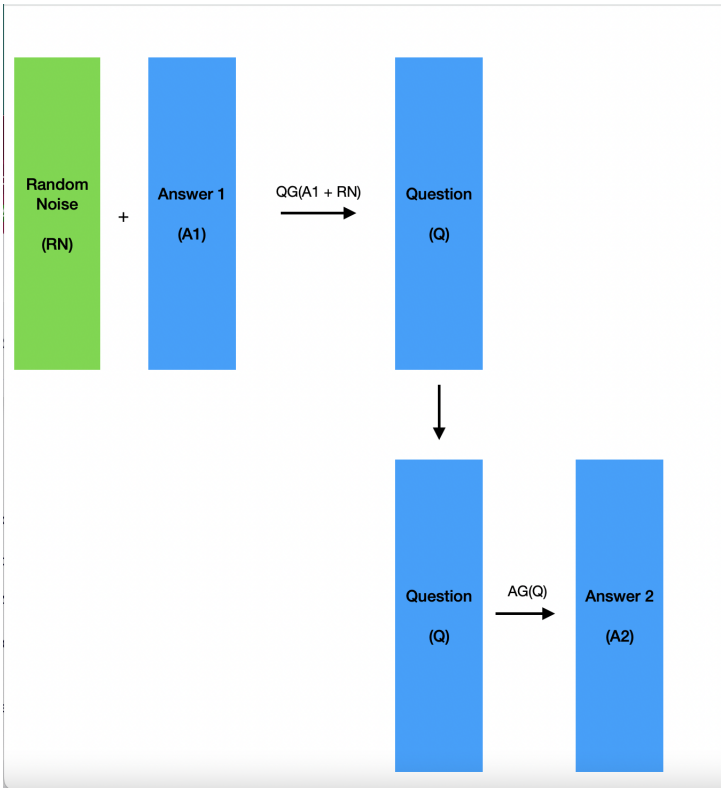

Figure 2: An architecture of truth discovery algorithm

Proof of Concept

Define a question generator $QG : A- > Q$

Define a answer generator $AG : Q- > A$

where A refers to an answer, and Q refers to a question. All the answers used to train QG and AG are assumed to be true.

Assume "truth" are form of answers that can be mapped to specific questions in one-to-one.

Take any answer A from question-answers pairs used to train QG and AG and add a random noise to A. Since all answers are assumed to be truth, A with Random Noise RN can be referred to as distorted truth T_D.

Suppose the question, created upon the distorted version of known answer (a.k.a T_D) through QG, is Q_D.

If $AG : Q_D -> T_D$, then distorted truth T_D is, in fact, true since T_D is the only answer that's mapped into Q_D, which follows the one-to-one nature of question and answer pairs. To be more specific, it's proven that T_D exists as a novel truth within known boundary of truth formed by all answers (truths) used to train QG and AG.

However, if $AG : Q_D !- > T_D$, then there may exist the answer other than T_D that's mapped into question Q_D, which contradicts one-to-one nature of question and answer pairs. Therefore, T_D cannot be verified to be true within the boundary of truth acknowledged by either QG or AG; in short, its truth state remains uncertain.

In this algorithm, under an assumption that all answers are true, one-to-one function between question and answer was a mathematical trick used as a tool to examine the existence of the novel truth within the acknowledged boundary of truth.

6 Heterozygous Network Generator: Modified Genetic Algorithm For Training Neural Network

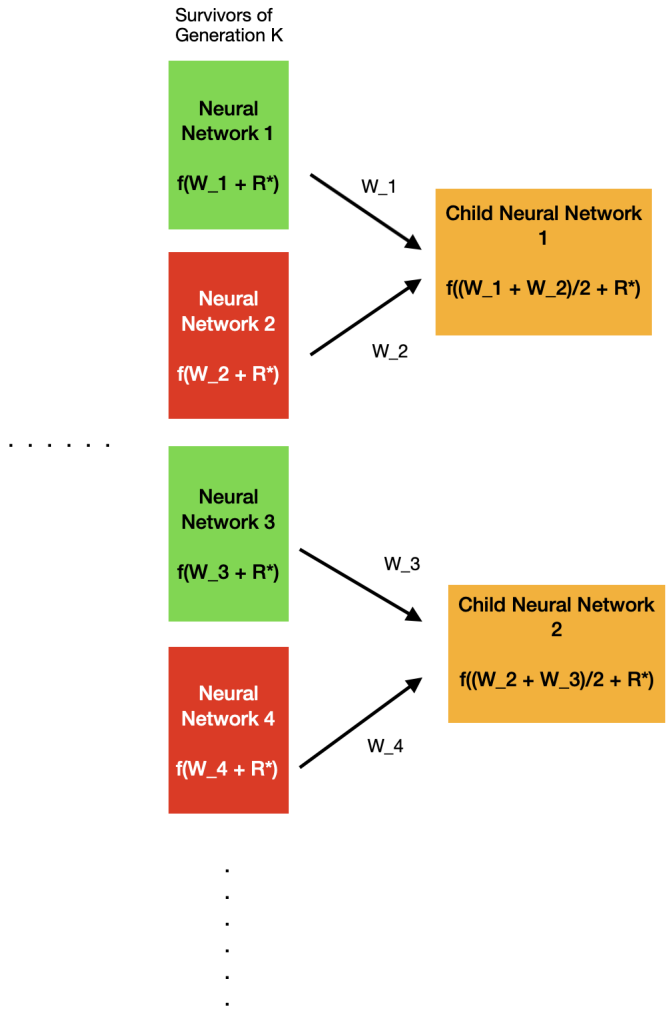

Figure 3: An architecture of Heterozygous Network Generator

In this modified genetic algorithm, first, the best k neural networks with highest validation accuracy will be selected in each generation. Then, selected k networks will produce mutated versions of themselves by adding random noises to their weights in hidden layers. Among mutated ones, pairs (parents) will

be selected and produce child neural networks with the weights averaged from neural networks in a pair. Previously, child neural network was produced by simply "crossing over" weights from parent neural network. This novel method is inspired by "Stochastic Weight Averaging" proposed by Izmailov et al [3] where the weighted average between two stochastic neural network led to the better generalization. Since the genetic algorithm is stochastic process that randomly alters the weights of neural network, the average of stochastic weights in neural networks produced from genetic algorithm will lead to the generalized neural network.

7 Familiarity Learning

	S1	S2	S3	S4	S5	S6	S7	S9	S10
S1									
S2									
S3									
S4									
S5									
S6									
S7									
S8									
S9									

Figure 4: A Example of Similarity Matrix

This learning method is to continuously select sample that's the most similar to the one which neural network was trained by at previous step. Before training neural network, similarity matrix should be created that shows the similarity between every possible pairs out of existing training samples $[S_i|i \in R]$. A similarity between S_1 and sample S_2 can be calculated by following equation:

$$cos(S_1, S_2) = \frac{S_1 * S_2}{\|S_1\|\|S_2\|} \tag{12}$$

At the beginning of training session, choose a random sample S_r from training set and train neural network with S_r. Then, choose the sample S_k that has the highest similarity value to S_r in the matrix and train neural network with S_k. This process should be repeated until the neural network is trained by all of available training examples. Because the network will trial-and-error to the next most familiar sample, it won't have fluctuating losses which leads to failure of convergence.

8 Ensemble of GAN and Convolutional Neural Classifier to build a counterfeit cloth detector

The proposed counterfeit cloth detector will work by collaboration of convolutional neural classifiers and Generative Adversarial Network (GAN). GAN is used to train a discriminative model that will discriminate between authentic cloth and counterfeit versions. Discriminative model needs an input of both images of authentic cloth and the cloth that needs to be tested whether is counterfeit or not. Therefore, it's necessary to identify which clothing that counterfeit one seems to assimilate to. To identify such clothing, convolutional neural classifiers can be used; the name label that has the highest score based on softmax on the output layer would be the name of authentic cloth that looks the most similar to the input image of cloth that's a potential candidate for counterfeit. After finding the image of authentic cloth whose name was found by convolutional neural classifier, a discriminative network trained by GAN will use both the image of authentic cloth and the cloth being tested to examine whether it's counterfeit. The proposed detector model will be validated in terms of its accuracy of correctly classifying counterfeit cloth as it is.

References

[1] Jaderberg, Max, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. "Reinforcement learning with unsupervised auxiliary tasks." arXiv preprint arXiv:1611.05397 (2016).

[2] Li, Yujia, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles et al. "Competition-level code generation with alphacode." Science 378, no. 6624 (2022): 1092-1097.

[3] Izmailov, Pavel, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. "Averaging weights leads to wider optima and better generalization." arXiv preprint arXiv:1803.05407 (2018).