Deep Learning

Shashank Shekhar

December 21, 2016

In classical planning, we represent a planning problem as $\mathcal{P} = < O, I, G >$. We consider a k-block problem from Blocksworls domain where we consider STRIPS representation. At a very high level, for a fixed goal, we are interested in learning to predict the goal distance for a given problem with k-block (the same number of blocks are used during different phases of the learning process). Influenced from transfer learning approaches, we consider a fixed number of objects (k) in problems for both training and testing phases using deep learning approaches. The orientations of the object change but k remains fixed always.

For applying any deep learning approach, from the literature, it can be said that we need a huge feature set along with sufficient training samples in the dataset. So in this direction, we come up with a unique bit vector representation, to represent all possible propositions (we consider them as features for learning) in $\mathcal{P}$. Consider the example below.

> **Example - 1**: We consider a 2-block problem, a simple one in which A and B are the two blocks. We have a fixed goal state, $\mathcal{G}$, and initial state could be any configuration of the blocks. For such problems, the possible literals present in an initial state could be any combination of OnTable(A), OnTable(B), HandEmpty, On(A,B), On(B,A), Clear(A), Clear(B), Holding(A), Holding(B) and their negations. Of course, in a state, a literal l and its negation ($!l$) are not allowed simultanously. Each of these literals may or may not belong to the given initial state. Each initial state, $\mathcal{I}$ will have some distance to cover to reach the goal state. We plan to represent a state (here $\mathcal{I}$) in form of a bit vector, $\mathcal{V}$, as shown below.
>
> $\mathcal{V}$ = [OnTable(A), !OnTable(A), OnTable(B), !OnTable(B), HandEmpty, !HandEmpty, On(A,B), !On(A,B), On(B,A), !On(B,A), Clear(A), !Clear(A), Clear(B), !Clear(B), Holding(A), !Holding(A), Holding(B), !Holding(B) : $dist(\mathcal{G})$]

Here, each literal is a feature of that initial state, and $dist(\mathcal{G})$ specifies the distance between initial and goal states found by any existing *planner* (a variant of Fast Downward). It would not be fruitful considering the don't care condition as we had discussed, also we do have problem generators that generate valid problems for this domain. Now, consider an initial state in which both the blocks are on the table, and the goal state is On(A, B). Then the vector will look like,

$\mathcal{V}_1$ = [1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0 : 2]

We have the goal distance, 2, obtained using a *planner* (*e.g.* Fast Downward). When the planner takes the initial and goal states for solving a planning problem, it explores the search space and traverses through many intermediate states. The idea is to store all those intermediate states, and later that can be used as other initial states. In our case, as we do not change the number of blocks, this avoids the extra work. For example, if the planner takes two blocks on the table as $\mathcal{I}$, it generates an intermediate state where the robot holds block A (Holding(A) while B is on the table). This can be a new initial state and can be represented as another vector, as shown below.

$\mathcal{V}_2$ = [0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0 : 1]

Here the goal distance is 1. We plan to generate at least a *fixed* number of such vectors, say $\mathcal{V}_1$, $\mathcal{V}_2$, ..., $\mathcal{V}_{min}$. Once we generate enough such vectors, we plan to try various deep learning approaches to learn effective relationships among these features. For this example, we will have a training set of the order $min \times 19$ ($[M]_{min \times 19}$). Next, we discuss the dataset preparation, training, and testing phases in brief.

1 Dataset Preparation

If we consider a 15-block planning problem from the Blocksworld domain, we can have a huge feature set, like we just saw for the 2-block problem. We fix the goal state and find the distance from any randomly generated valid state in the search space. Following the **Example - 1** , we plan to prepare a training dataset for a fixed sized problem. Following the literature, we feel that bootstrapping, a machine learning approach, can be employed to automate the dataset preparation phase.

A very high level *algorithm* is shown below, discusses the idea of bootstrapping for the training dataset preparation.

- A variant of fast downward planner, *FD*; set of problem instances, *Ins*
- An empty Training Set, *TS*
- Global Variables - T_{max}, T_{limit}, *min* (min of $\mathcal{V}_{min}$), $\mathcal{G}$
- $count = 0$
- For each instance I in *Ins*
  - **While ($T_{max} < T_{limit}$) and (*count* < *min*)**
    - *solve* (I, $\mathcal{G}$, *FD*, T_{max})
    - If I is solved with in the time T_{max}
      - Update *TS*, (*e.g.* $\mathcal{V}_i$ = [0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0 : 1])
      - Remove I from *Ins*
      - Consider all the intermediate state as a new initial state
      - Add them to the *TS*, with all possible vectors, and update *count* and *Repeat* the process
    - If not, $T_{max} = 2 \times T_{max}$
    - **continue**
- Exit - with a training dataset *TS* (contains at least *min* entries)

2 Training and Testing

- Using *TS* from the dataset preparation phase, we can learn different models using DL approaches (say using a Conventional NN)
- For testing any such models, we randomly generate a problem with same number of blocks (k) which has same goal considered during the dataset preparation phae
- For that problem, consider its initial state and represent it in the form of bit vector (*e.g.* $\mathcal{V}_l$ = [0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0 : _]) (with an empty goal distance feature (*dist*($\mathcal{G}$)) - the *target*)

- We use any of the learned models, and feed this vector to calculate $dist(\mathcal{G})$
- We verify the accuracy of $dist(\mathcal{G})$ with the *distance-to-goal* found by the original planner.