DERI算法的完整理论描述：从符号运算到超参数推导

- 作者：GaoZheng
- 日期：2024-12-19

引言：DERI算法的核心目标

DERI（Dynamic Explicit Reverse Inference）算法是广义增强学习理论中的训练算法。其主要任务是通过对观测路径样本的逻辑性度量进行逆向推导，从符号模型中推导出解析的超参数（Hyperparameter Vector, $\mathbf{w}$）和泛泛函（Generalized Functional, $L(s, \mathbf{w})$），最终构建出合理的符号运算模型，用于路径优化或决策演化。

I. 基本问题描述

1. **输入：观测路径与逻辑性度量**
   - 观测路径集合 $\mathrm{SamplePaths}$：

     $$\mathrm{SamplePaths} = \{\pi_1, \pi_2, \ldots, \pi_m\}, \quad \pi_i \subseteq S$$

     每条路径 π_i 是状态集合 S 的有序子集。
   - 路径的观测逻辑性度量总得分 $\mathrm{ObservedValues}$：

     $$\mathrm{ObservedValues} = \{v_1, v_2, \ldots, v_m\}, \quad v_i \in \mathbb{R}$$

2. **目标：逆向推导模型与超参数**
   给定路径样本 $\mathrm{SamplePaths}$ 和总得分 $\mathrm{ObservedValues}$，推导出：
   - 泛泛函的超参数 $\mathbf{w} = \{w_1, w_2, w_3, \ldots, w_k\}$
   - 符号运算模型的拓扑约束 T
   - 逻辑性度量 $L(s, \mathbf{w})$

II. 形式化定义

1. **逻辑性度量泛泛函 $L(s, \mathbf{w})$**
   对状态 s 的逻辑性度量定义为：

$$L(s, \mathbf{w}) = \tanh\left(w_1 \cdot p_1(s) + w_2 \cdot p_2(s) + \cdots + w_k \cdot p_k(s)\right)$$

   其中，$\mathbf{w} = \{w_1, w_2, \ldots, w_k\}$ 是需要优化的超参数，$p_i(s)$ 是状态 s 的属性值。
2. **观测路径的逻辑性总得分**
   每条路径 π_i 的逻辑性总得分定义为：

$$G(\pi_i, \mathbf{w}) = \sum_{s \in \pi_i} L(s, \mathbf{w})$$

   目标是使得 $G(\pi_i, \mathbf{w})$ 与 v_i 的差异最小。
3. **目标函数**
   优化问题的目标函数定义为：

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{m} \left(G(\pi_i, \mathbf{w}) - v_i\right)^2$$

   其中，加入正则化项 $R(\mathbf{w})$ 控制超参数的范围：

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{m} \left(G(\pi_i, \mathbf{w}) - v_i\right)^2 + \lambda \cdot R(\mathbf{w})$$

   常见的正则化项为 $R(\mathbf{w}) = \sum_{j=1}^{k} w_j^2$。
4. **拓扑约束的推导**
   路径样本中隐含的拓扑约束 T 的推导基于状态邻接关系：

$$T(s) = \{s' \mid s' \text{ 是路径样本中 } s \text{ 的后继状态}\}$$

III. DERI算法流程

1. **输入初始化**
   - 给定状态集合 S 和属性集合 P：

$$S = \{s_1, s_2, \ldots, s_n\}, \quad P = \{P(s_1), P(s_2), \ldots, P(s_n)\}$$

- 输入观测路径集合 $\mathrm{SamplePaths}$ 和总得分 $\mathrm{ObservedValues}$。

2. **拓扑约束的推导**
   - 初始化拓扑约束 T 为空：

$$T = \{s \mapsto \emptyset \mid s \in S\}$$

   - 对每条路径 π_i 中的状态对 (s_k, s_{k+1})，更新 $T(s_k)$：

$$T(s_k) \leftarrow T(s_k) \cup \{s_{k+1}\}$$

3. **逻辑性度量的拟合**
   - 定义目标函数：

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{m} \left(\sum_{s \in \pi_i} \tanh \left(\sum_{j=1}^{k} w_j \cdot p_j(s) \right) - v_i \right)^2$$

   - 使用数值优化方法（如梯度下降或 $N\mathrm{Minimize}$）找到最优超参数：

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

4. **结果输出**
   - 生成最终的符号模型 M：

$$M = (S, P, T, L(s, \mathbf{w}^*))$$

IV. 重要性质

1. **路径与超参数的相容性**
   如果 $\mathbf{w}^*$ 是通过 $\mathrm{SamplePaths}$ 和 $\mathrm{ObservedValues}$ 推导出的超参数，则：

$$G(\pi_i, \mathbf{w}^*) \approx v_i, \quad \forall i$$

   说明超参数 $\mathbf{w}^*$ 能够高度描述路径样本的逻辑性度量。
2. **模型的可扩展性**
   DERI允许对逻辑性度量 $L(s, \mathbf{w})$ 使用更复杂的泛泛函（例如非线性算子或复合算子），以适应更高维度和复杂性的数据。
3. **拓扑与模型超参数的协同优化**
   DERI能够同时推导状态间的拓扑约束 T 和模型超参数 $\mathbf{w}$，实现模型的全局一致性。

V. 公式化总结

DERI的最终解析解可以表示为：

$$M = (S, P, T^*, L(s, \mathbf{w}^*))$$

其中：

$$T^* = \arg\max_{T} \sum_{\pi \in \text{Paths}(T)} \sum_{s \in \pi} L(s, \mathbf{w}^*), \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

DERI通过理论建模和符号推导，为路径样本与逻辑性度量之间的解析性关联提供了一种高效且可解释的框架。这种框架不仅显著提升了模型的训练效率，还为后续路径优化和决策问题奠定了坚实的数学基础。

附：代码示例

```
(*清空环境变量*)
ClearAll[S, P, SamplePaths, ObservedValues, InferAlgebra, \
InferTopology, InferLogic, OptimizeDStructure]

(*定义已观测的样本路径 SamplePaths*)
(*1.样本路径的形成。实验过程：定义实验环境：\
设计具有不同状态和演化关系的系统。\
记录状态演化路径：通过实际实验观测记录系统从初始状态到\
最终状态的演化路径。例如：观察不同材料的物理参数随环境变化\
（如温度、压力）的状态转移路径。\
记录多次实验的完整路径，形成类似于 \
SamplePaths 的集合。\
示例：实验可能产生以下路径：*)
SamplePaths = {{"s1", "s2", "s3", "s4"}, {"s1", "s2", "s4"}, {"s1",
    "s3", "s5"}};

(*定义每条路径的观测逻辑性度量总得分 ObservedValues*)
(*2.逻辑性度量的观测值。实验过程：\
设置目标函数：定义实验中需要优化的目标值，\
例如性能、效率、稳定性等。采集观测值：\
对于每条路径，记录其逻辑性度量的观测值，\
例如通过实验数据计算路径的总性能得分。\
示例：对于上述路径，通过实验观测可得：*)
ObservedValues = {3.5, 3.0, 3.8};

(*定义状态集合 S 和属性模板 P*)
(*3.属性模板的形成。实验过程：设计属性空间：\
为每个状态定义多个属性（如频率、密度、能带宽度等），\
这些属性通过实验数据获得。\
采集属性值：通过测量技术采集属性值并构建状态集合 P。\
示例：实验采集数据可构建如下状态属性：*)
S = {"s1", "s2", "s3", "s4", "s5"};
P = <|"s1" -> <|"$$Omega]" -> 1.5, "ne" -> 1.2, "W" -> 2.0|>,
   "s2" -> <|"$$Omega]" -> 2.0, "ne" -> 1.5, "W" -> 1.8|>,
   "s3" -> <|"$$Omega]" -> 2.5, "ne" -> 1.6, "W" -> 1.7|>,
   "s4" -> <|"$$Omega]" -> 3.0, "ne" -> 1.8, "W" -> 1.5|>,
   "s5" -> <|"$$Omega]" -> 3.5, "ne" -> 2.0, "W" -> 1.3|>|>;
```

```
(*定义状态节点的数学结构*)
StateStructure[state_, props_] := <|"State" -> state,
   "Properties" -> props,
   "Algebra" -> (Function[{x, y},(*代数规则：状态合成*)<|
       "$$Omega]" -> x["$$Omega]"] + y["$$Omega]"],
       "ne" -> x["ne"] + y["ne"], "W" -> x["W"] + y["W"]|>]),
   "Topology" -> (Function[{neighbors, topology},(*拓扑规则：验证邻接关系*)
      If[SubsetQ[neighbors, topology[state]], True, False]])|>;

(*构造每个状态的独立数学结构*)
States = Association[KeyValueMap[#1 -> StateStructure[#1, #2] &, P]];

(*定义逻辑性度量函数 L*)
L[stateStructure_, {w1_, w2_, w3_}] :=
  Module[{rawValue, props}, props = stateStructure["Properties"];
   rawValue = w1  props["$$Omega]"] + w2  props["ne"] - w3  props["W"];
   Tanh[rawValue]];

(*1.推导拓扑约束 (T)*)
InferTopology[samplePaths_] :=
  Module[{topology},
   topology = Association[Table[state -> {}, {state, S}]];
   Do[Do[
     If[! MemberQ[topology[path[[i]]], path[[i + 1]]],
      AppendTo[topology[path[[i]]], path[[i + 1]]]], {i,
      Length[path] - 1}], {path, samplePaths}];
   Association[KeyValueMap[#1 -> DeleteDuplicates[#2] &, topology]]];

(*2.推导代数规则 (AlgebraRule)*)
InferAlgebra[samplePaths_, observedValues_] :=
  Module[{lossFunction, optimizedParams},
  (*定义损失函数为观测值与逻辑性度量计算值的平方误差*)
   lossFunction[weights_List] :=
    Module[{w1, w2, w3, totalLoss}, {w1, w2, w3} = weights;
     totalLoss =
      Total[(observedValues -
          Table[Total[L[States[#], {w1, w2, w3}] & /@ path], {path,
            samplePaths}])^2];
     totalLoss];
   (*使用 NMinimize 优化 w1,w2,w3*)
```

```
    optimizedParams =
     NMinimize[{lossFunction[{w1, w2, w3}], -1 <= w1 <= 1 && -1 <= w2 <=
          1 && -1 <= w3 <= 1}, {w1, w2, w3}];
    optimizedParams[[2]] (*返回优化后的参数*)];

(*3.逻辑性度量推导（结合参数优化）*)
InferLogic[params_, topology_] :=
  Module[{logicalValues},
    logicalValues =
     Association[
      KeyValueMap[#1 -> L[States[#1], params] &, topology]];
    logicalValues];

(*4. 优化 D 结构*)
OptimizeDStructure[samplePaths_, observedValues_] :=
  Module[{topology, algebraParams, logicalValues,
     optimizedD},(*推导拓扑结构*)topology = InferTopology[samplePaths];
    (*推导代数规则参数化*)
    algebraParams = InferAlgebra[samplePaths, observedValues];
    (*推导逻辑性度量*)logicalValues = InferLogic[algebraParams, topology];
    (*构建最终 D 结构*)
    optimizedD = <|"States" -> States, "Topology" -> topology,
      "AlgebraParams" -> algebraParams,
      "LogicalValues" -> logicalValues|>;
    optimizedD];

(*执行逆向推导*)
OptimizedDStructure = OptimizeDStructure[SamplePaths, ObservedValues];

(*打印结果*)
Print["Optimized D Structure: ", OptimizedDStructure];
```
