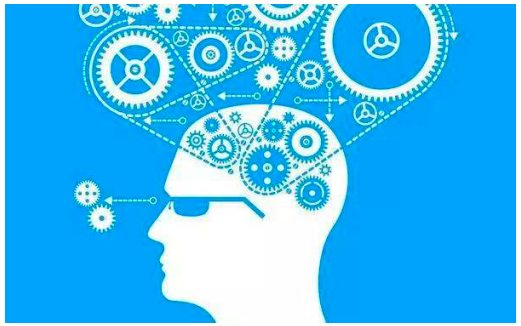
Reinforcement Learning: Innovations and Applications

王凡

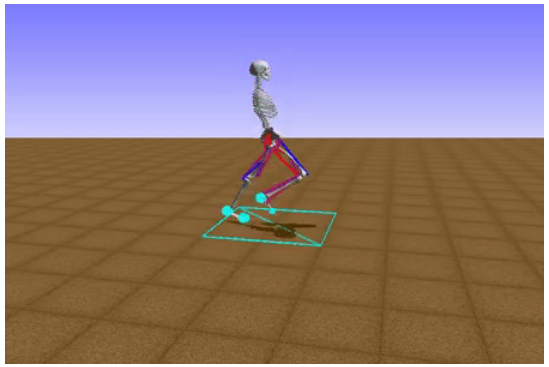
百度 自然语言处理部

Content Overview



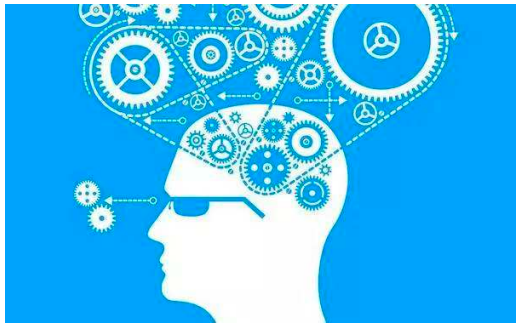
Algorithms

Tools & Platforms



Applications

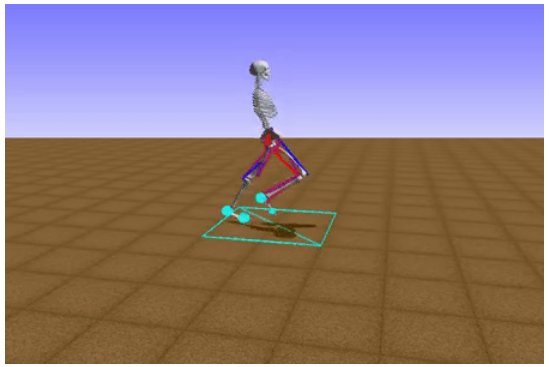
Content Overview



Algorithms

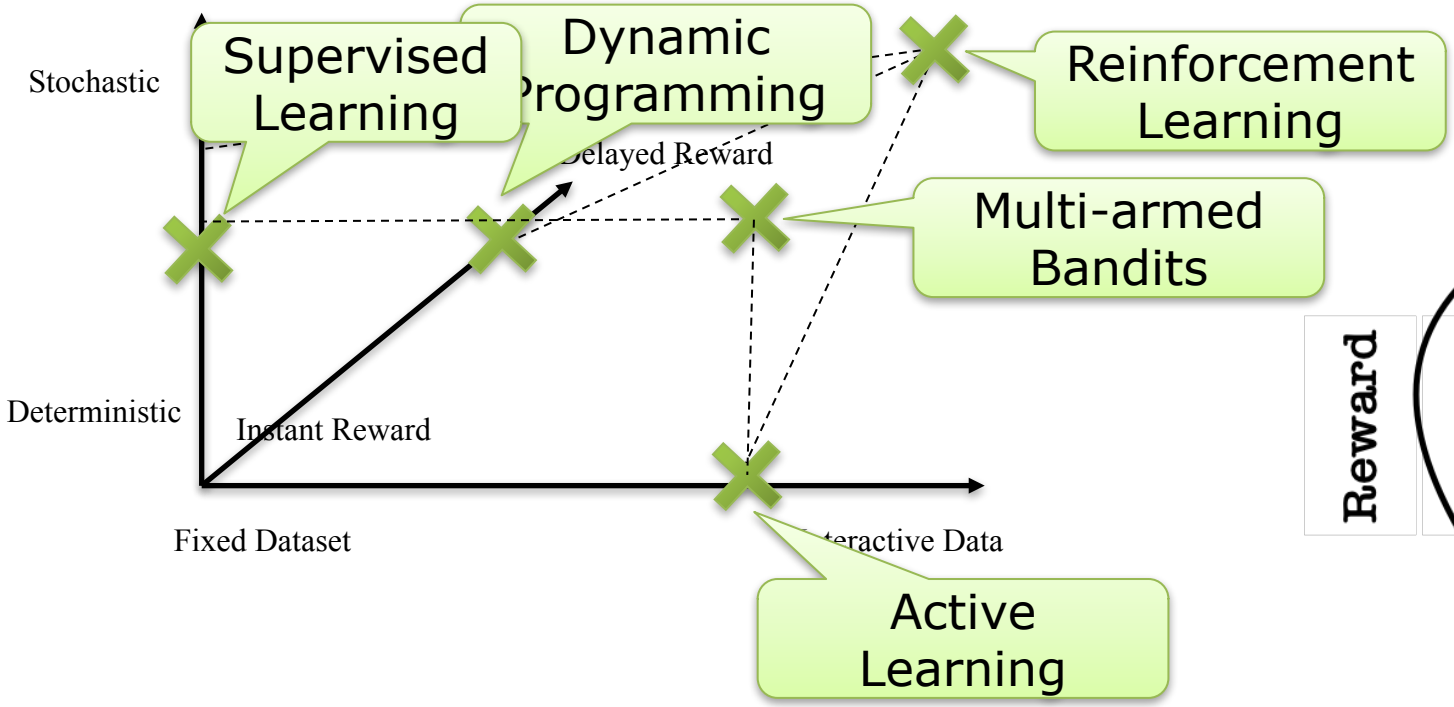
A Super-Simple Introduction to RL

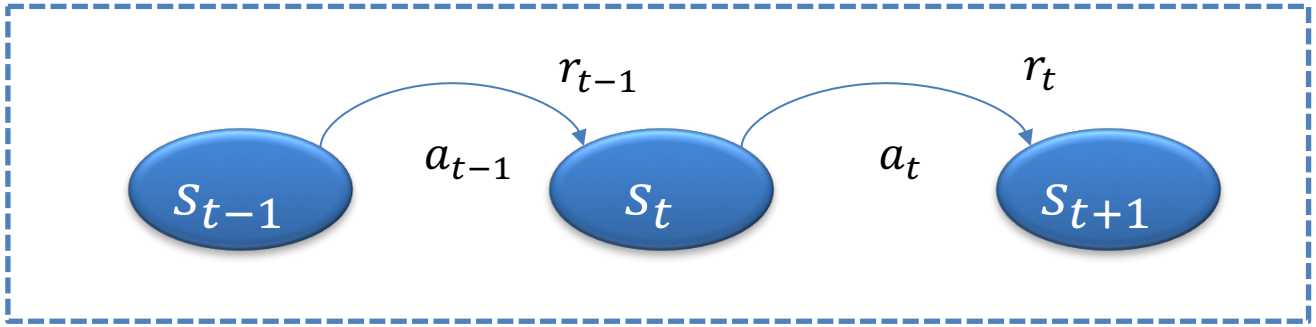
Tools & Platforms

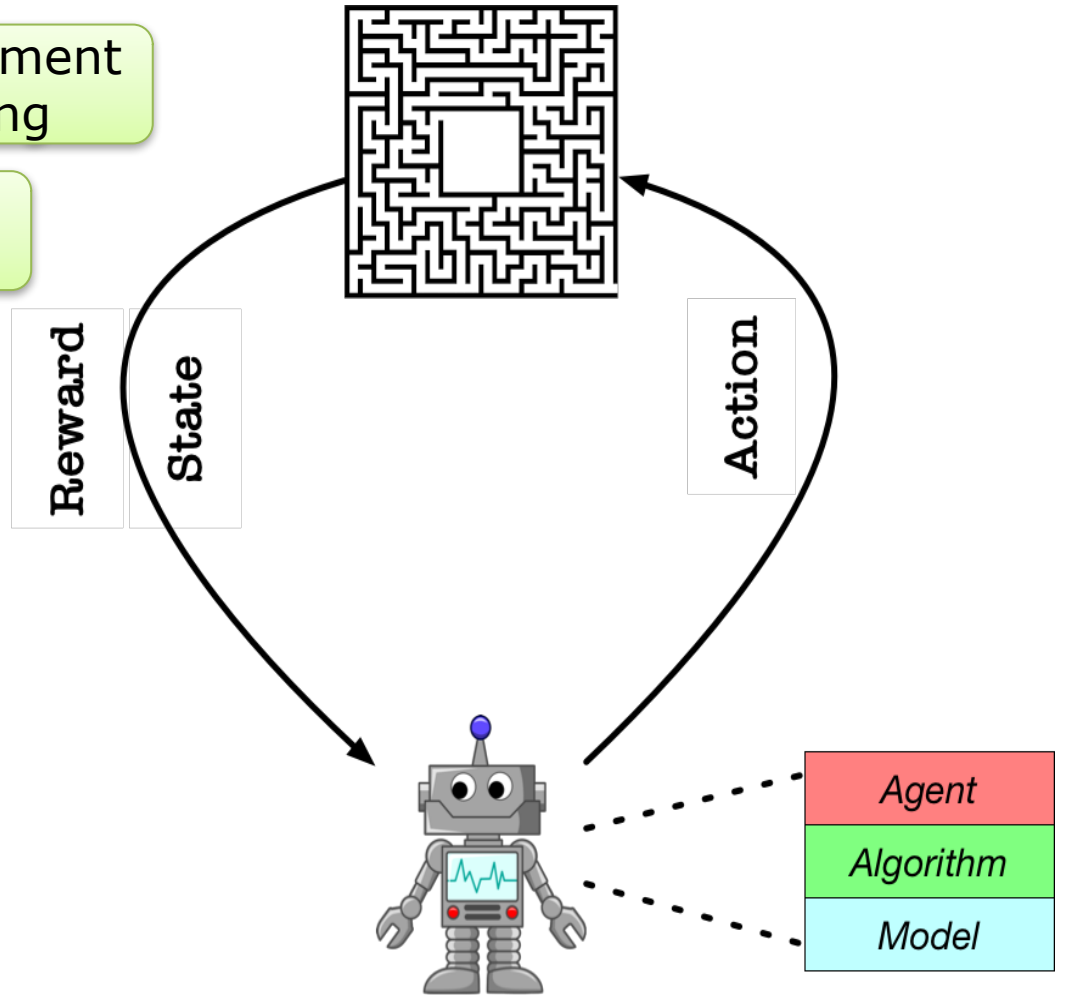


Applications

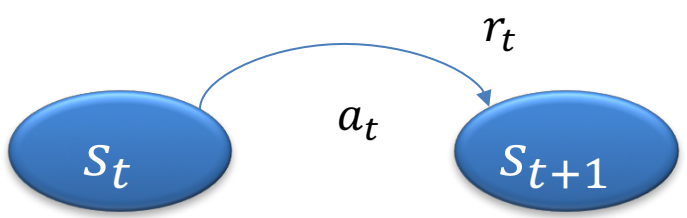
Understanding Reinforcement Learning







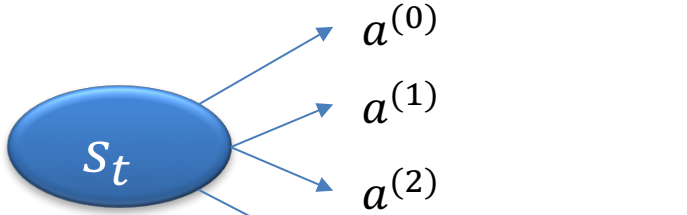
Understanding Reinforcement Learning



Transition : $p(s_{t+1}|s_t, a_t)$

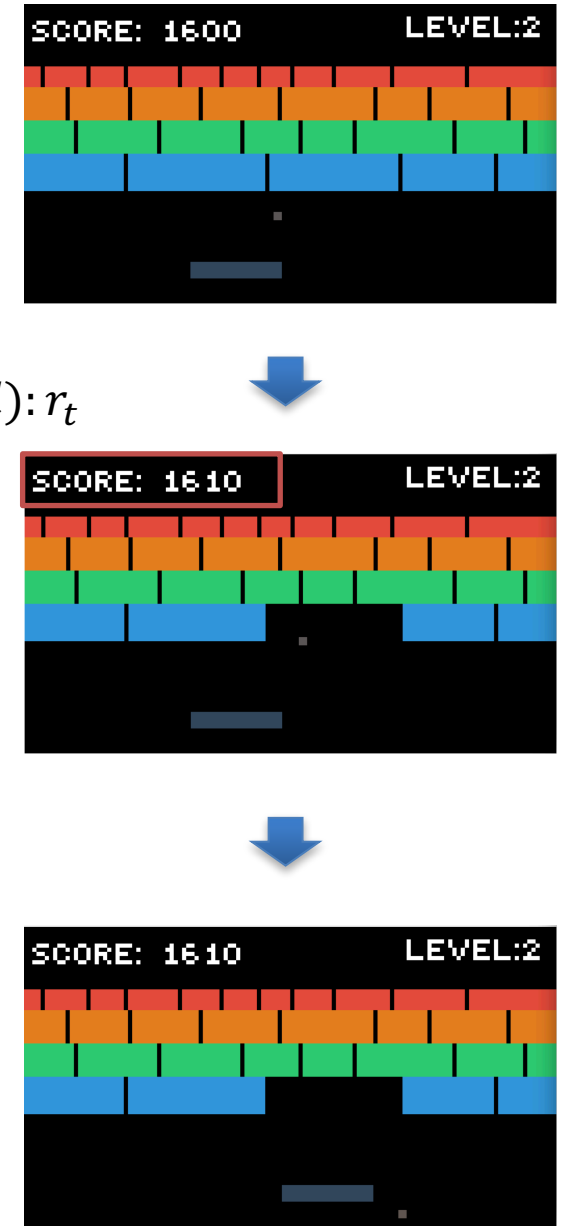
Reward : $\mathrm{E}(r_t) = r(s_t, a_t, s_{t+1})$

Markov Decision Process



$\pi_\theta : p(s_t, a_t) \leftarrow f_p(s_t, a_t; \theta_p)$

Policy (Actor)



$Observation(States): s_t$

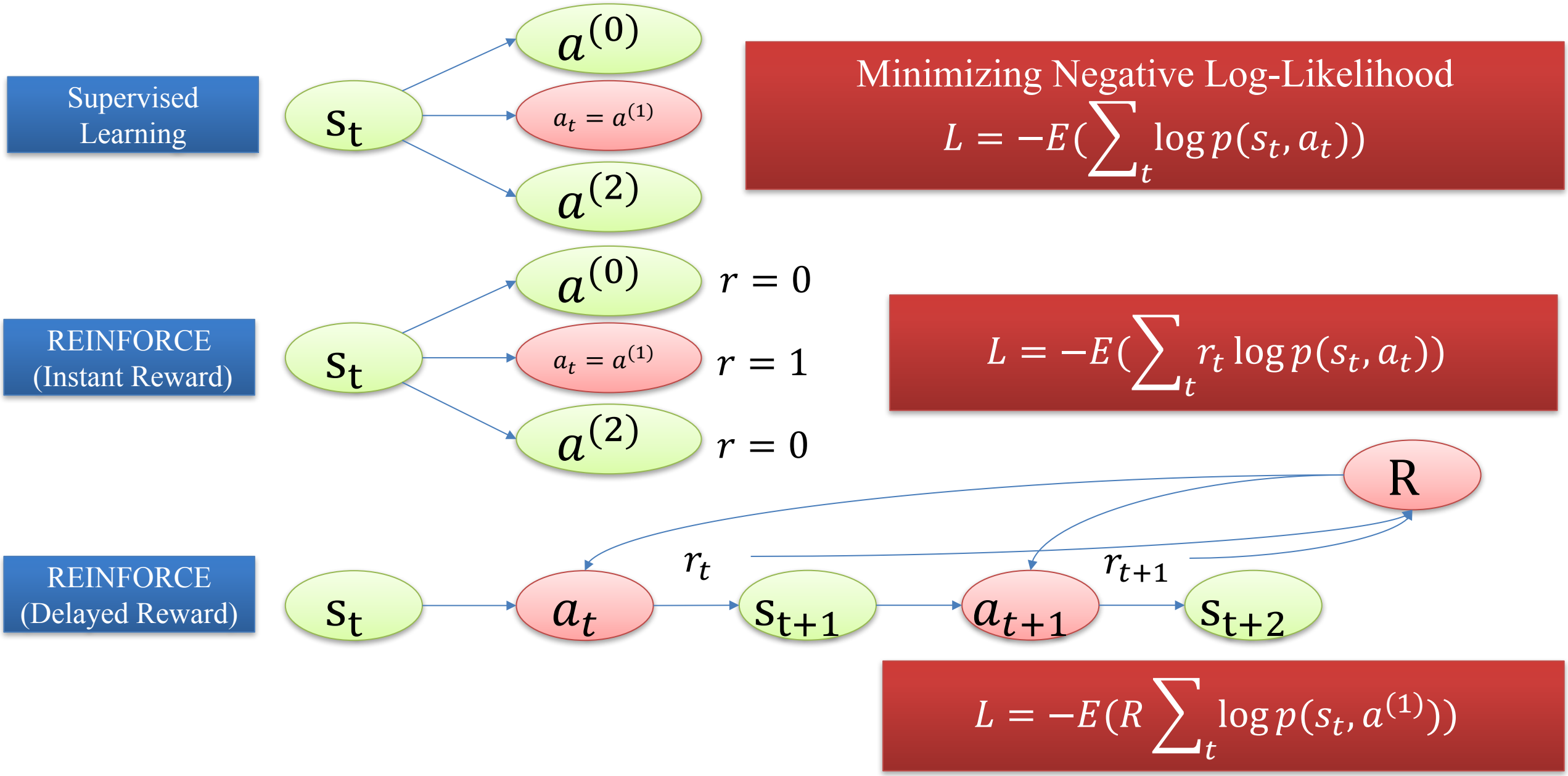
$action: a_t$

$Feedback(Reward): r_t$

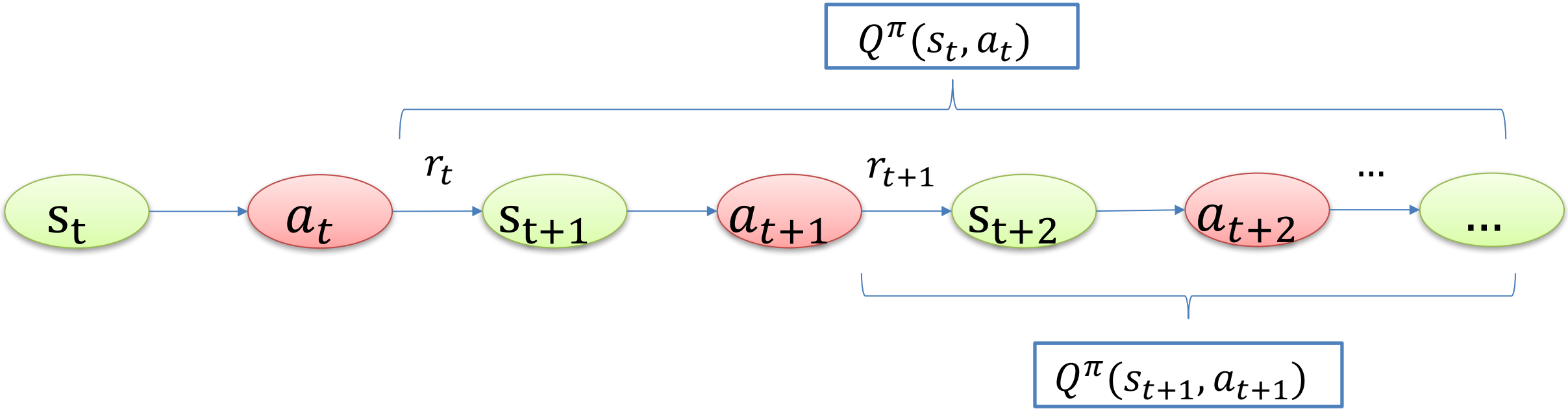
s_{t+1}

s_{t+2}

Understanding Reinforcement Learning



Understanding Reinforcement Learning



Value Function (Critic)

$$Q^{\pi}(s_t, a_t) = E(r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots)$$

$$Q^{\pi}(s_t, a_t) = \mathrm{E}[\gamma\, Q^{\pi}(s_{t+1}, a_{t+1}) + \mathrm{r_t}]$$

Bellman Equation, Learning Target

Understanding Reinforcement Learning

Discrete State Space:

$$Q^{\pi}_{k+1}(s_t, a_t) = Q^{\pi}_{k}(s_t, a_t) + \alpha\left(\gamma Q^{\pi}_{k}(s_{t+1}, a_{t+1}) + \mathrm{r_t} - Q^{\pi}_{k}(s_t, a_t)\right)$$

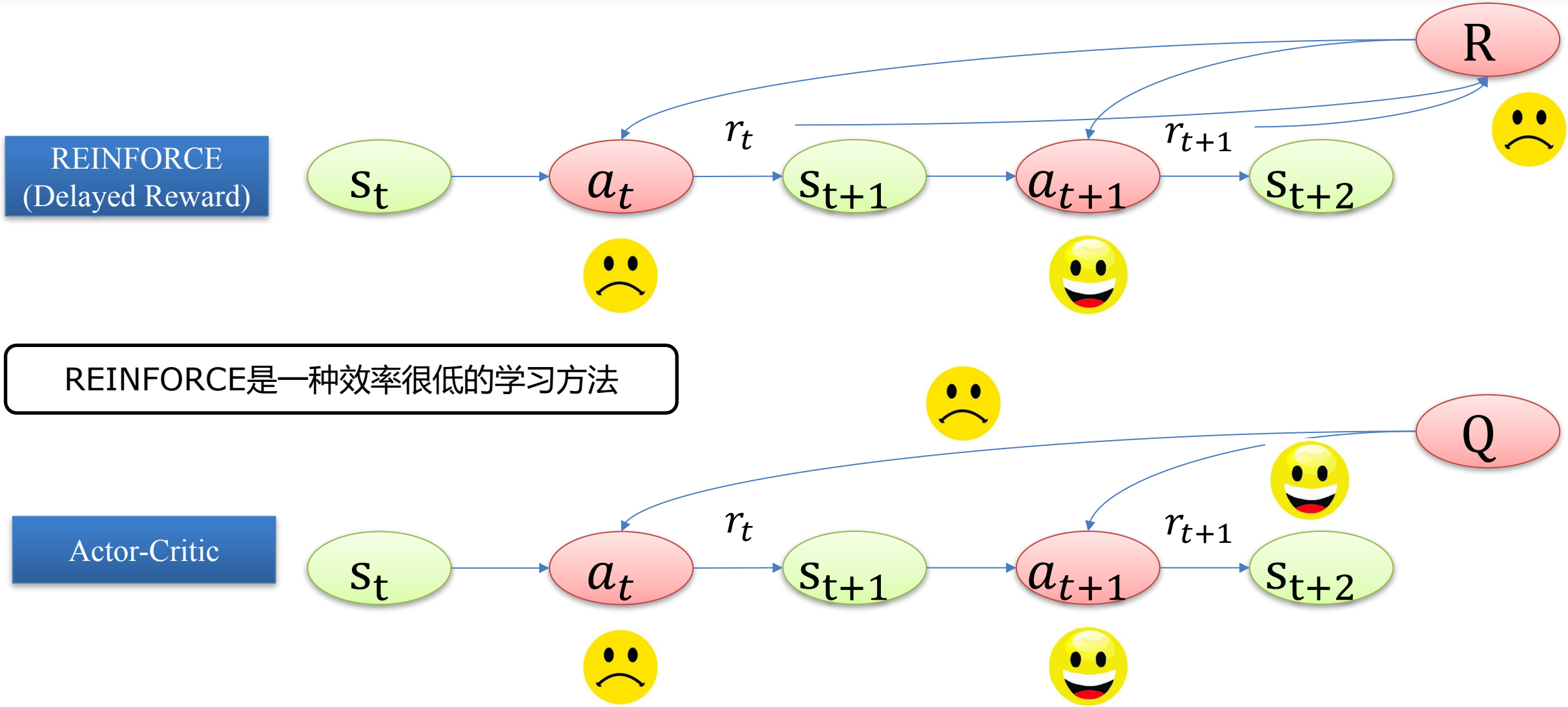
Continuous State Space: $\mathrm{Q}_{\theta}(s_t, a_t) \longrightarrow Q^{\pi}(s_t, a_t)$

Function Approximator

$$L(\theta) = E[\sum_t \left(\gamma Q_{\theta'}(s_{t+1}, a_{t+1}) + \mathrm{r_t} - Q_{\theta}(s_t, a_t)\right)^2]$$

Temporal Difference Error

Understanding Reinforcement Learning



Understanding Reinforcement Learning

$$L = -E[\, R \;\; \Sigma_t \log(p(s_t, a_t)) \,]$$

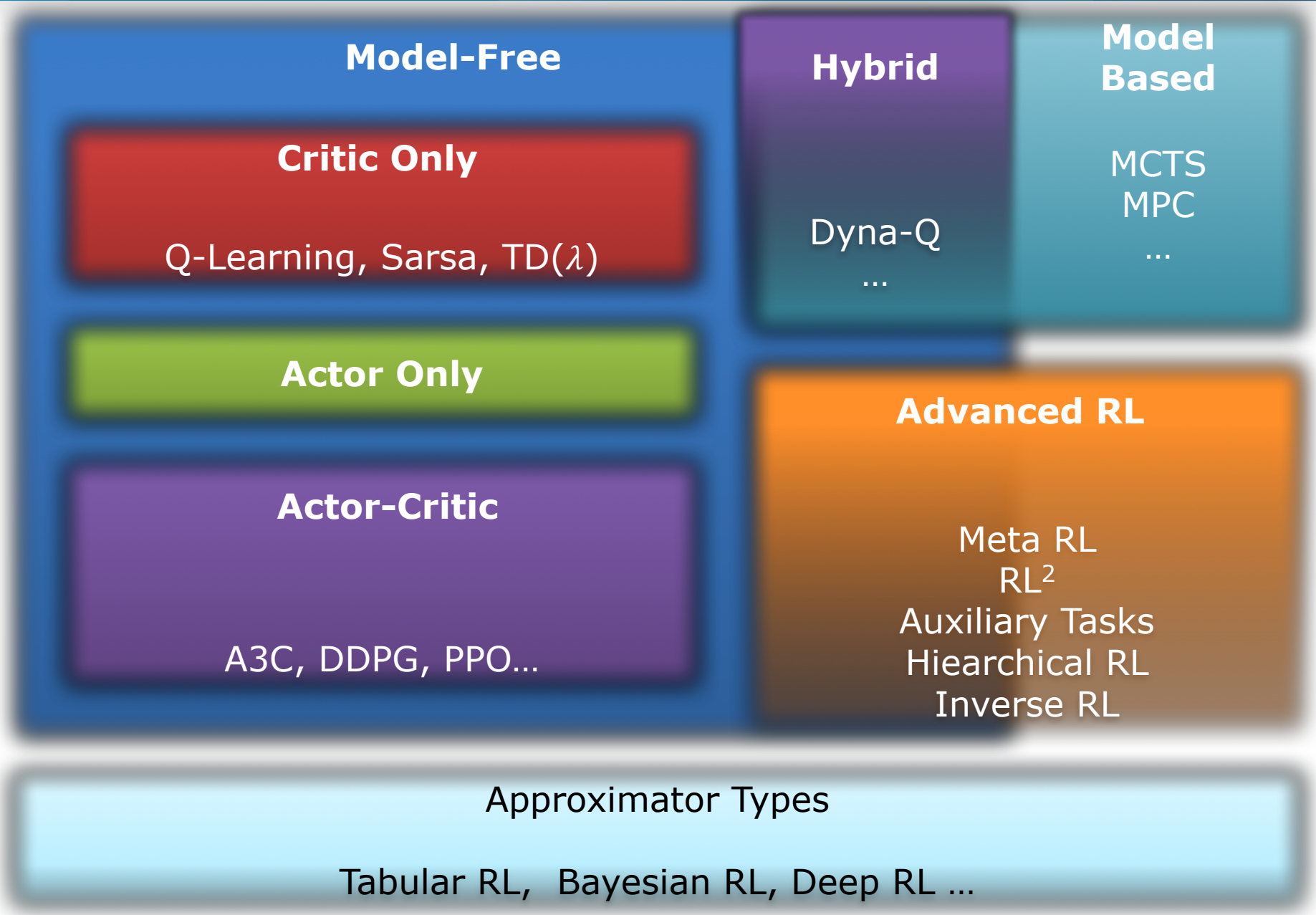
Advantage

$$L = -E[\, \Sigma_t \, \Psi_t \log(p(s_t, a_t))]$$

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^{\pi}(s_t, a_t)$: state-action value function.
5. $A^{\pi}(s_t, a_t)$: advantage function.
6. $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$: TD residual.

Schulman, John, et al. "High-dimensional continuous control using generalized advantage estimation." arXiv preprint arXiv:1506.02438 (2015).

Understanding Reinforcement Learning



Development

RL主要玩家

Intelligent Robotics

OpenAI

OpenAI Gym：最权威的RL评测框架

DeepMind

AlphaGo：围棋、国际象棋、日本将棋全面超过人类

Google AI

Dopamine：开源RL学习框架

百度RL积累

初期探索

- Multi-Armed Bandit 排序/搜索（NLP）
- 对话系统调研和积累(NLP)

持续积累

- PaddlePaddle 发布(IDL)
- 世界首个AGI交互评测环境XWorld发布(Baidu Research)
- 百度度秘(NLP)

成熟工业级应用

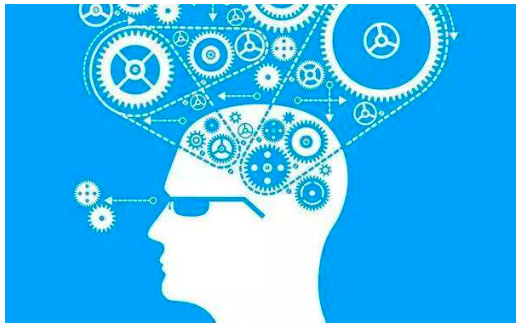
- 定价系统（凤巢）
- Feed流推荐（Feed, NLP等）

持续创新突破

- PaddlePaddle Fluid
- PARL
- AutoML
- ...

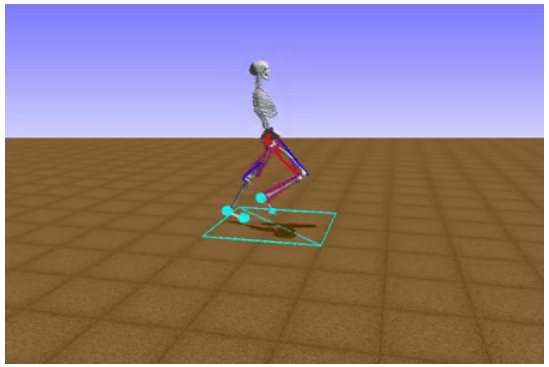
Baidu 百度

Content Overview



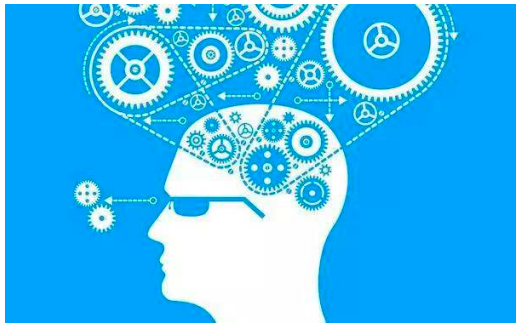
Algorithms

Tools & Platforms



Applications

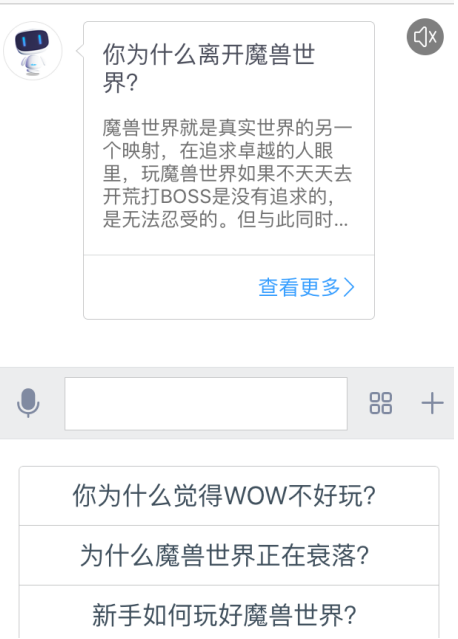
Content Overview

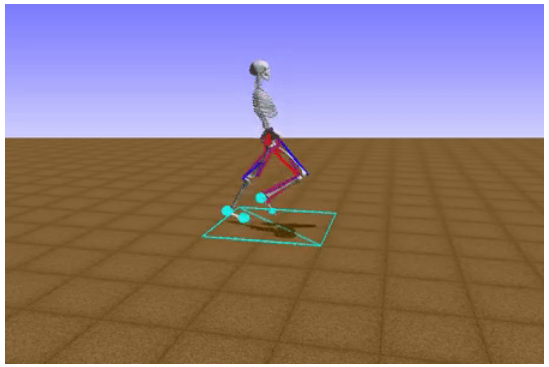


Algorithms

Tools & Platforms

Recommender System





Applications

Background

Dynamic Evolution

Correlations?

为了力挺华为，华人首富海外砸下175亿！任正非的目标又近了一步

张书乐 212评论 播报

高通推出第三代骁龙汽车驾驶舱平台

科技地平线 播报

没人告诉你的大规模部署AI高效流程！

机器之心 播报

微信最大服务商微盟预计1月15日登陆港股

科技地平线 播报

印度电信部长：印度政府无意禁止采购华为网络设备

新浪科技 播报

中国4大AI语音公司掀起“造芯”，中芯国际入场，行业洗牌在即

DeepTech深科技 播报

Background

Dynamic Evolution

Correlations?

为了力挺华为，华人首富海外砸下175亿！任正非的目标又近了一步

张书乐 212评论 播报

高通推出第三代骁龙汽车驾驶舱平台

科技地平线 播报

没人告诉你的大规模部署AI高效流程！

机器之心 播报

微信最大服务商微盟预计1月15日登陆港股

科技地平线 播报

印度电信部长：印度政府无意禁止采购华为网络设备

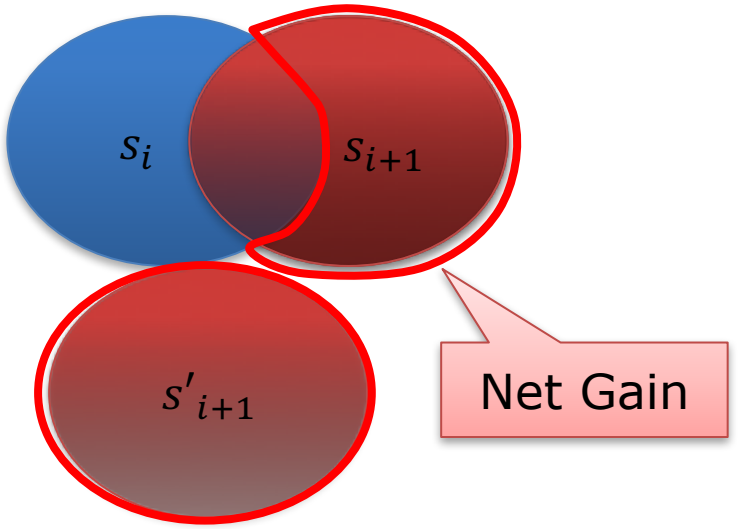
新浪科技 播报

中国4大AI语音公司掀起"造芯"，中芯国际入场，行业洗牌在即

DeepTech深科技 播报

Background

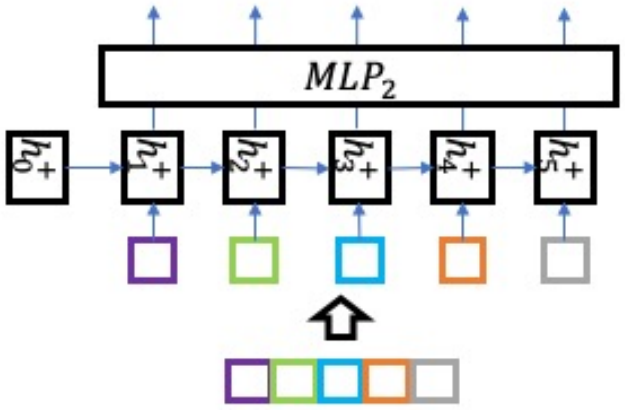
- K-Items Recommender System
  - Problem setting
    - Exposure of a list of K items: $s_1, s_2, \dots s_K$, Receive clicks: $r_1, \dots r_K$
  - **Item Independent Prediction**
    - Click of item is independent
    - $\mathrm{P}(r_i|u, s_i) \propto f(u, s_i)$
  - Diversified Ranking
    - Submodular Ranking[1]
      - $\mathrm{P}(r_i|u, s_i) \propto f(u, \Delta(s_i||s_1, \dots, s_{i-1}))$
      - $\Delta(s_i||s_1, \dots, s_{i-1})$: Net Gain
    - Deterministic Point Process[2]
      - Represent the item correlation with kernels



[1] Yue, Yisong, and Carlos Guestrin. "Linear submodular bandits and their application to diversified retrieval." Advances in Neural Information Processing Systems. 2011.
[2] Wilhelm, Mark, et al. "Practical Diversified Recommendations on YouTube with Determinantal Point Processes." Proceedings of the 27th ACM International Conference on Information and Knowledge Management. ACM, 2018.

Background

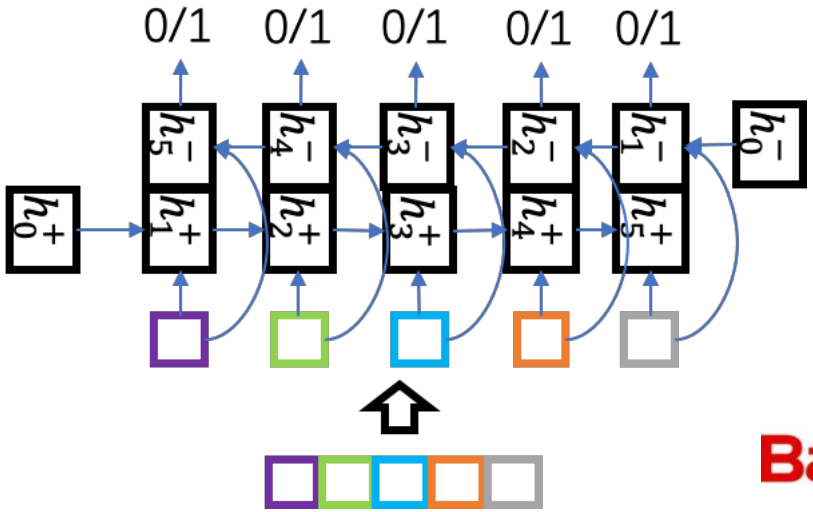
- Go beyond diversity…
  - Diversity is personalized and topic - dependent
  - Submodularity imposes strong prepositions…
  - Diversity ≠ All Correlations
  - Position Bias?
  - Local Optimal versus Global Optimum…
- **Preceding Items Aware Prediction**
  - Neural network models all possible correlations
  - $\mathrm{P}(r_i|u, s_i) \propto f(u, s_i, S_i^- = \{s_1, \ldots, s_{i-1}\})$
  - Still feasible to be used for ranking items



Paradigm of CTR Prediction

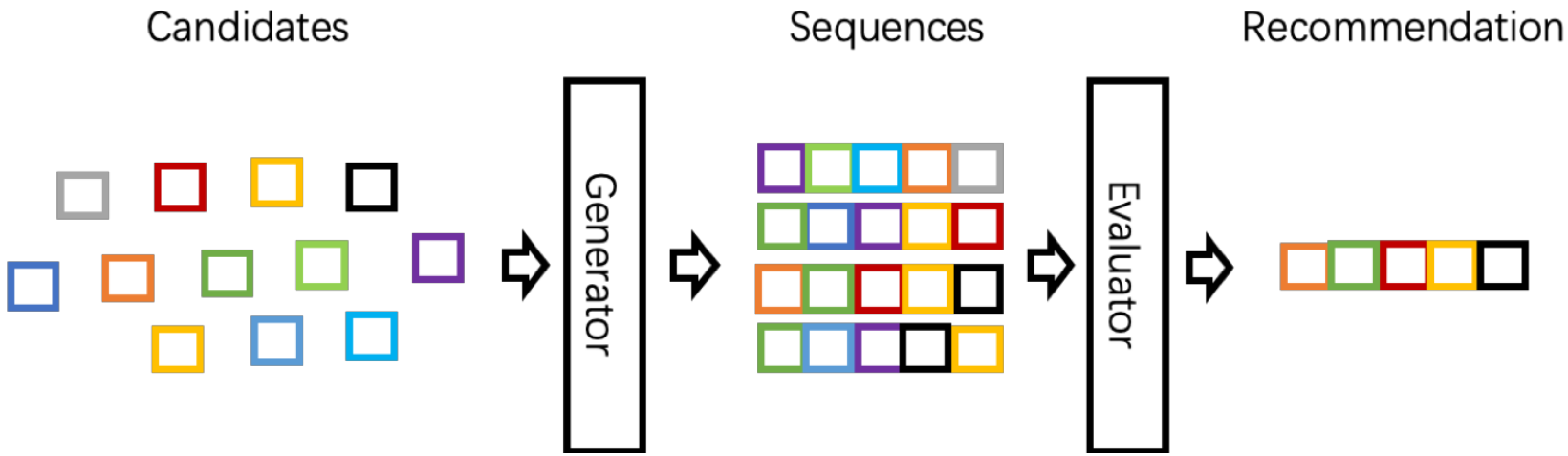
- **Contextual Items Aware Prediction**
  - $\mathrm{P}(r_i = 1|u, s_i) \propto f(u, d_i,\ S_i^- = \{s_1, \ldots, s_{i-1}\},\ S_i^+ = \{s_{i+1}, s_{i+2}, \ldots, s_K\})$
  - Following Items matters, too
  - Bi-Directional Correlation Modeling
  - Can not be used directly for ranking

- **Pointer Network**
  - Aware of Preceding Contents and Candidates
  - $\mathrm{P}(r_i = 1|u, s_i) \propto f(u, d_i,\ S_i^- = \{s_1, \ldots, s_{i-1}\},\ C)$



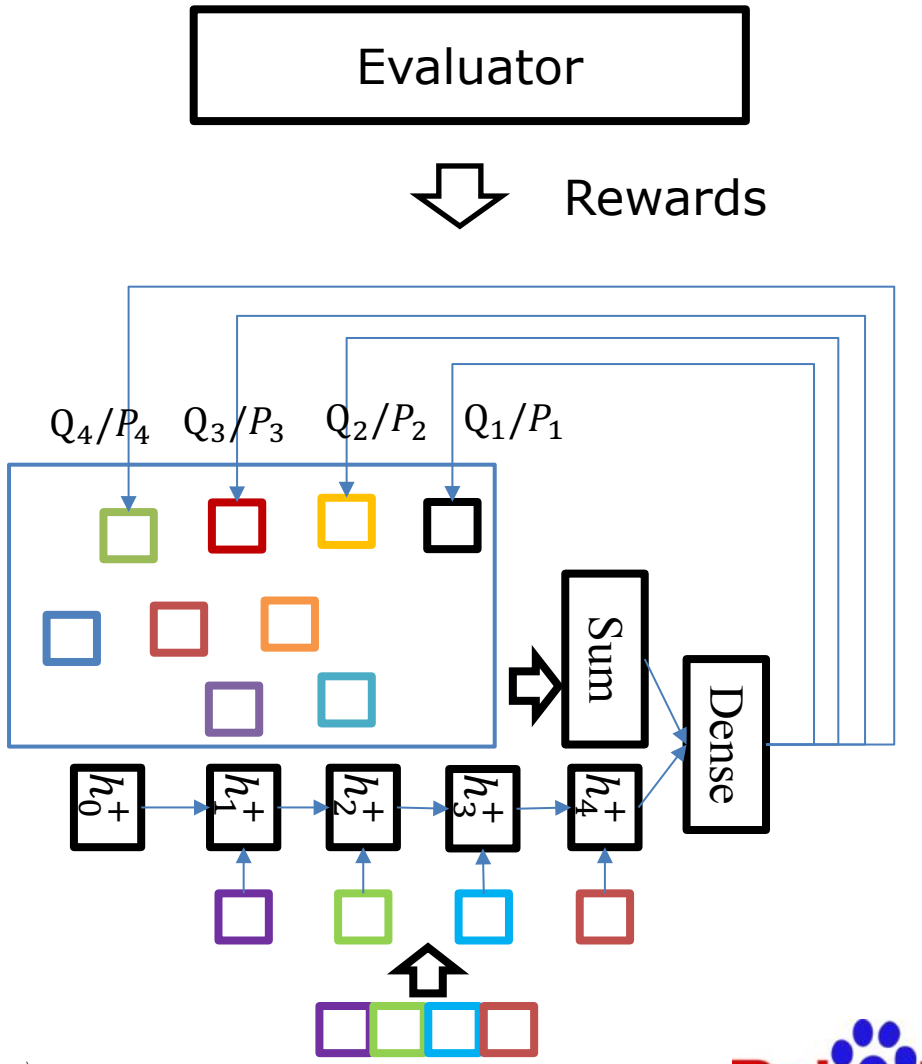
Evaluator-Generator Framework

- Combining Bi-Directional Evaluation & Single Directional Generation



Reinforcement Learning the Generator

- **Revised Pointer Network (RPN)**
  - Set-to-sequence Decoder (set2seq)
  - Candidate Unordered Set -> Sequence of Recommendation
- Learning from user feedbacks
  - Maximize $\sum_i P(r_i = 1)$
- Learning from Counterfactual Evaluators
  - Learning from the Evaluator
  - Generator “Hacks” Evaluator



From item-wise recommendation to combinatorial recommendation: Sequence Optimization Approach (In Writting)

Sequence Optimization for Baidu App News Feed Recommender System

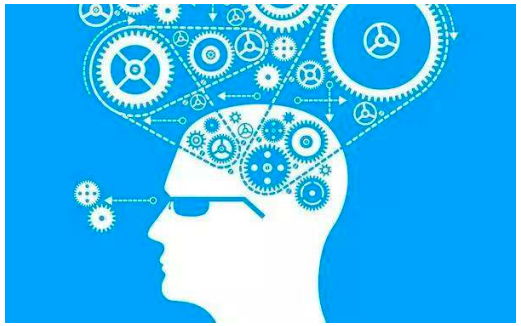
- Offline
  - Sequence Optimization greatly increases the click & duration compared with DNN only
- Online
  - Ever since we launched Evaluator-Generator system
    - Average Clicks rise > 10%
    - Average Duration rise > 5%

Evaluator	Correlation (Click)	Correlation (Duration)
DNN	0.228	0.802
Bi-Directional GRNN	**0.968**	**0.947**

Generator	Evaluation Score
DNN Greedy	1.564
DNN + Sampling (20)	1.701
RPN + Q-Learning + Greedy	**1.941**

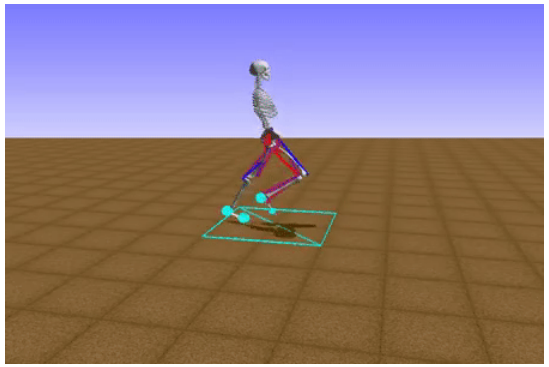
Evaluator	Pointwise AUC (Gain)	Correlations with the Total Clicks
DNN	+0.00%	0.468
GRNN	+0.82%	0.483
Bi-Directional GRNN	**+1.14%**	**0.492**

Content Overview



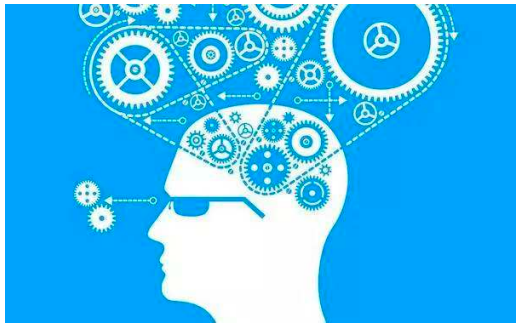
Algorithms

Tools & Platforms



Applications

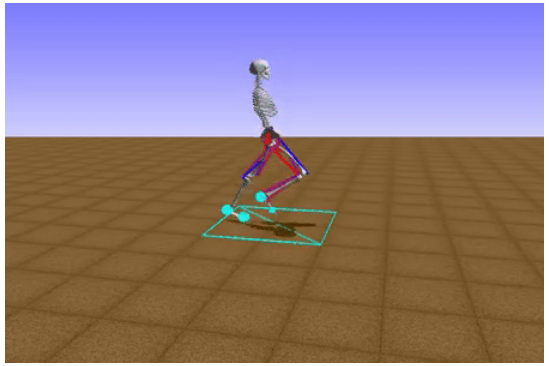
Content Overview



Algorithms

Tools & Platforms

Dialogue Management & Generation



Applications

Baidu 百度

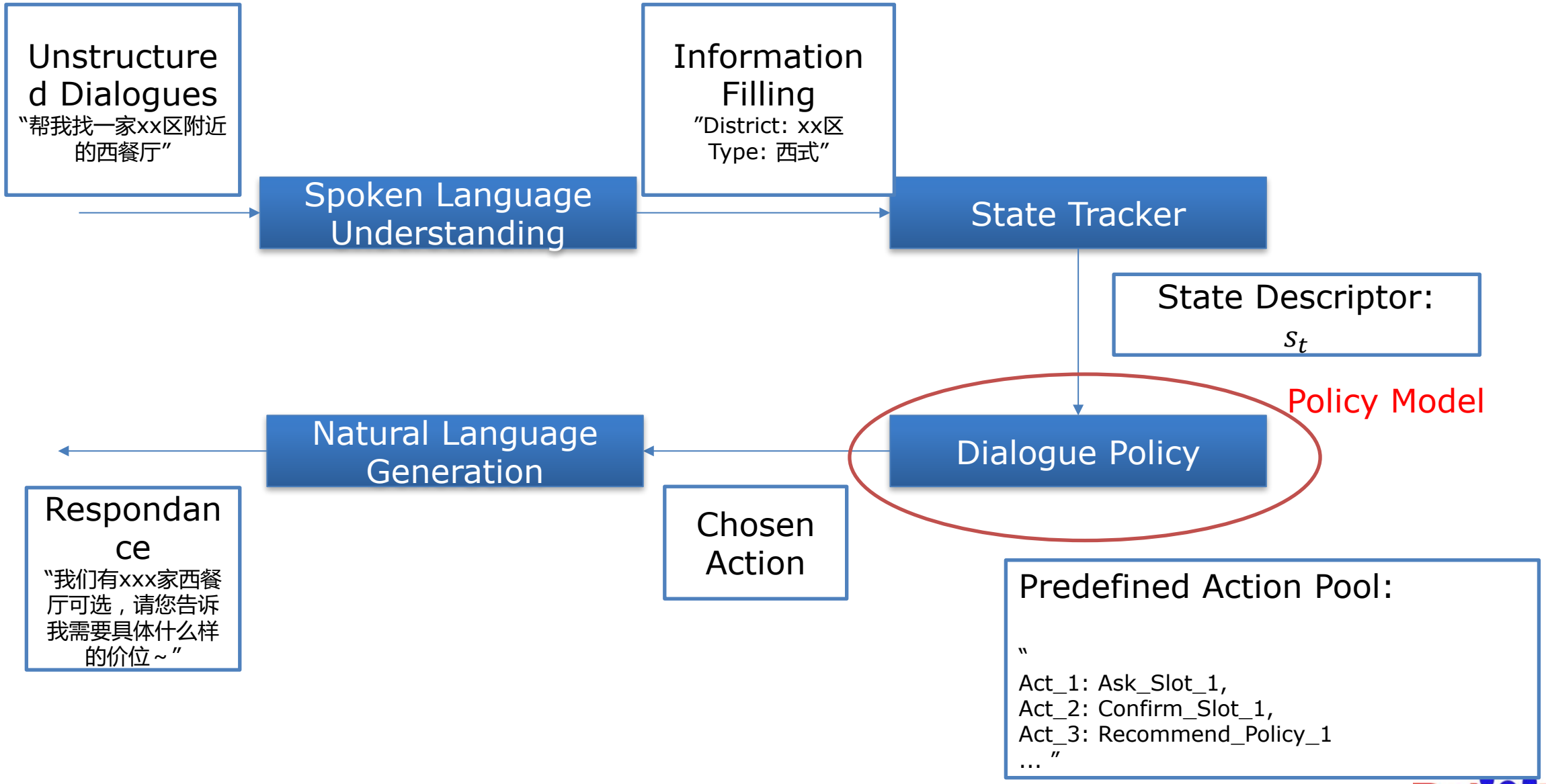
Backgrounds

- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - Early Demonstration systems (2012 ~ 2014)
  - Dumi chatbot (2014~2016)
  - Duer OS, UNIT & Open-domain Dialogue Systems

Backgrounds

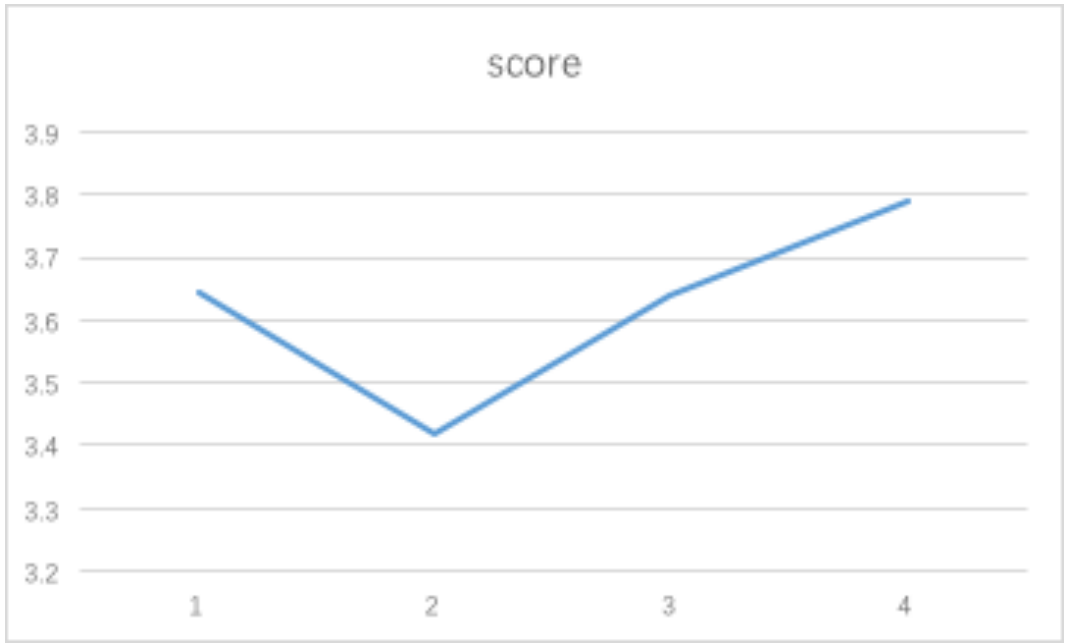
- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - **Early Demonstration systems (2012 ~ 2014)**
  - Dumi chatbot (2014~2016)
  - Duer OS, UNIT & Open-domain Dialogue Systems

Early Demonstration systems



Early Demonstration systems

- Automatic Ordering System
  - Algorithms: Least Square Temporal Difference (LSTD), 12 feature, 7 actions
  - Feedbacks: Human Labeling, 1000 + sessions



RL中Action的种类：

1. 询问口味
2. 询问食材
3. 依据历史推荐菜肴
4. 依据历史推荐食材
5. 依据历史推荐口味（未启用）
6. UserCF推荐
7. 热门菜肴推荐。

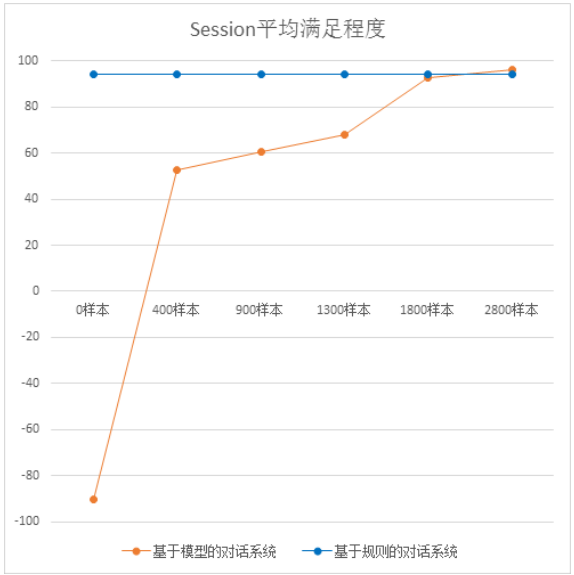
具体Action的feature设定可以参见表1.

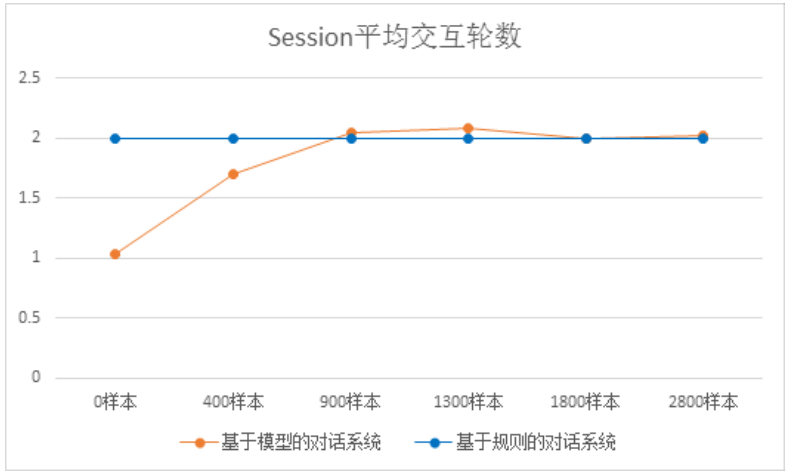
特征的含义：

1. Turn_cnt : 当前对话轮数
2. Last_refuse : 上一次拒绝距离当前的轮数
3. Slot_cnt : 当前填充的条件数
4. Show_cnt : 自动展示结果次数
5. Black_cnt : 被拒绝的object数量
6. Latest_req : 最近一次“询问”距离当前的轮数
7. Latest_rec_his : 最近一次“基于历史推荐”距离当前的轮数
8. Latest_expl : 最近一次”探索推荐”距离当前的轮数
9. Order_in_1day : 推荐的菜肴是否在1天内点过
10. Order_in_3day : 推荐的菜肴是否在3天内点过
11. Cf_weight : 推荐算法给出的权重
12. Hot_weight : 热门菜的权重（衰减累计的被order的次数）

Early Demonstration systems

- Weather chat system
  - Algorithms: Least Square Temporal Difference (LSTD), 100+ features
  - Feedbacks: Human Labeling, 3000 sessions
  - Baseline: Elaborated Rules





ıtent	slots	satisfy-actions	clarify-actions/other
WEATHER	time	duer_weather_weather	slot_clarify(time/lo
	loc_nation	duer_weather_cloudy	slot_yn_clarify(time/l
	loc_provience	duer_weather_rain	intent_clarify (inten
	loc_city	duer_weather_snow	cannot_understand_c
	loc_district	duer_weather_sunny	exit_action
	quest	duer_weather_wind	
		duer_weather_temp	
		duer_weather_low_temp	
		duer_weather_high_temp	
		duer_weather_aqi	
		duer_weather_clothes	
		duer_weather_wash_car	
		duer_weather_trip	
		duer_weather_influenza	
		duer_weather_exercise	
		duer_weather_ultraviolet	
		duer_weather_fog	

Backgrounds

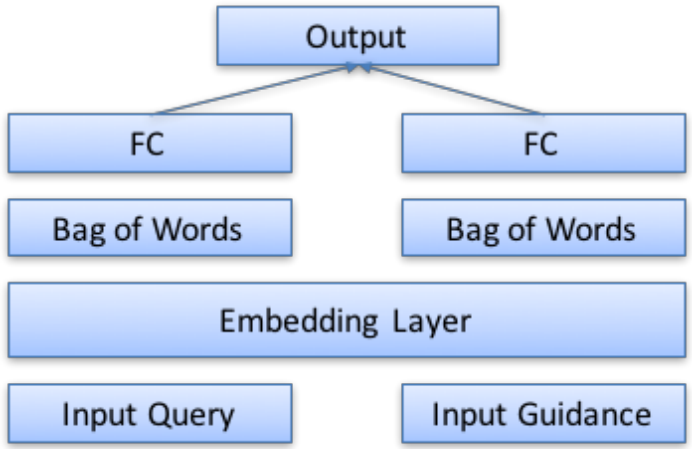
- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - Early Demonstration systems (2012 ~ 2014)
  - Dumi chatbot (2014~2016)
  - Duer OS, UNIT & Open-domain Dialogue Systems

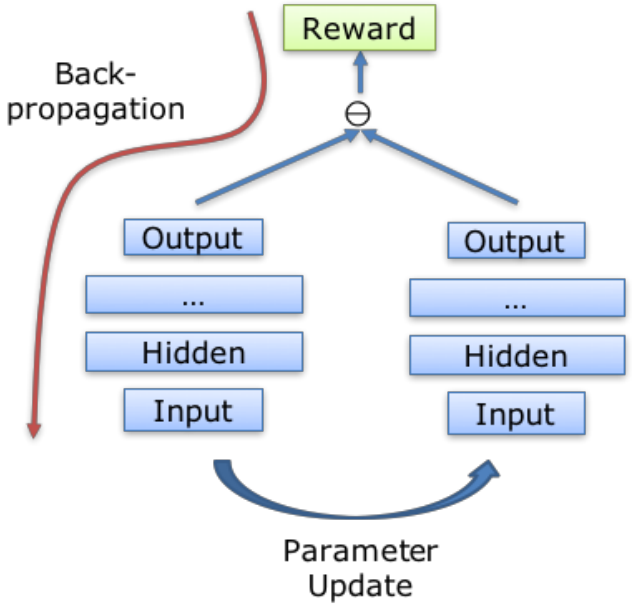
Backgrounds

- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - Early Demonstration systems (2012 ~ 2014)
  - **Dumi chatbot (2014~2016)**
  - Duer OS, UNIT & Open-domain Dialogue Systems

Dumi Chatbot

- Conversation guidance (Dumi-Chatbot)
  - Algorithm: Semantic Matching DQN
  - Feedbacks: Off-Policy data, over 8 million
  - Evaluation vs Baseline (Semantic Matching DNN): GSB = 16:75:9





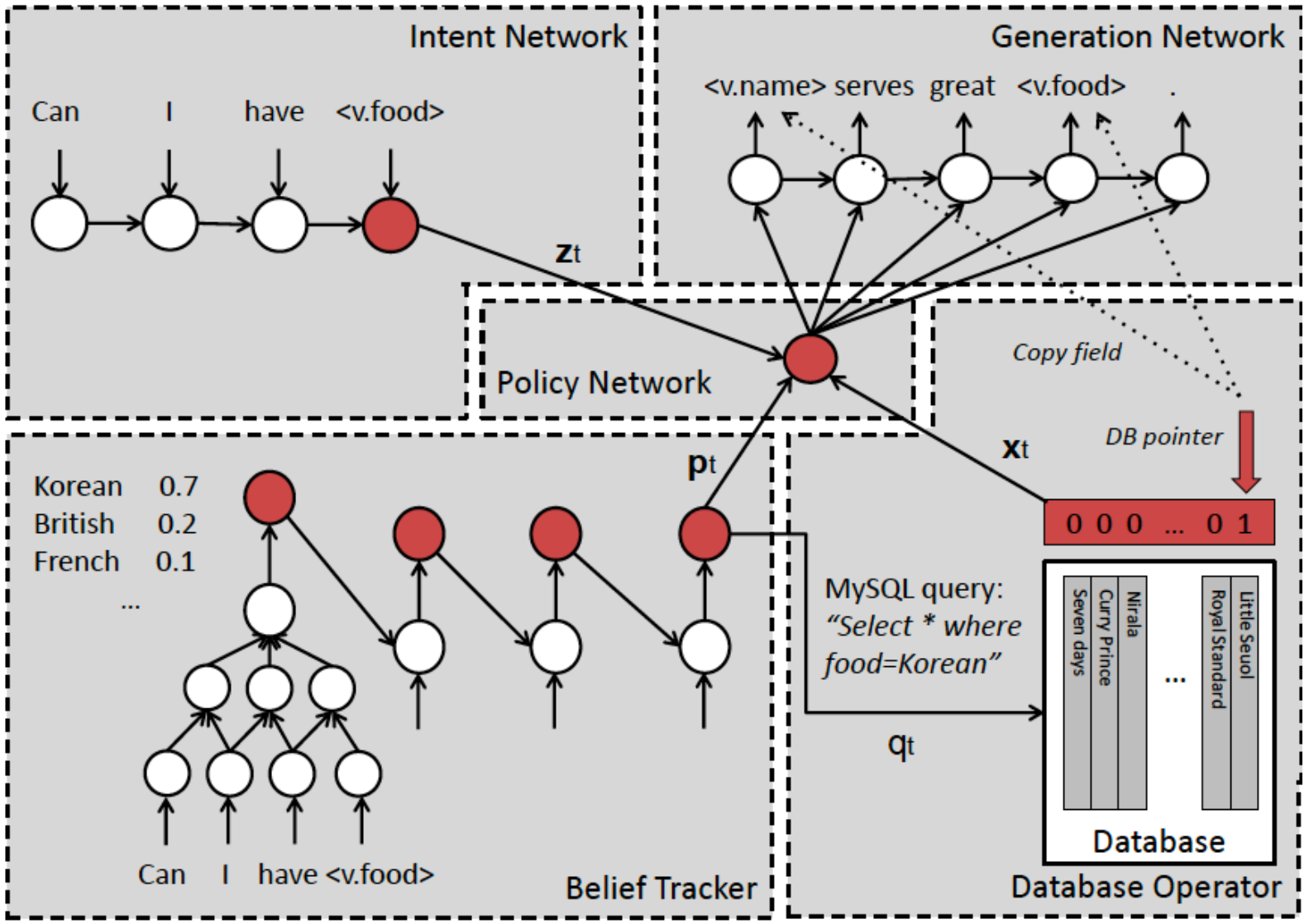
Backgrounds

- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - Early Demonstration systems (2012 ~ 2014)
  - Dumi chatbot (2014~2016)
  - Duer OS, UNIT & Open-domain Dialogue Systems

Backgrounds

- Baidu has a long history in Task Oriented Dialogue System & Open Domain Dialogue System since 2012
  - Early Demonstration systems (2012 ~ 2014)
  - Dumi chatbot (2014~2016)
  - **Duer OS, UNIT & Open-domain Dialogue Systems**

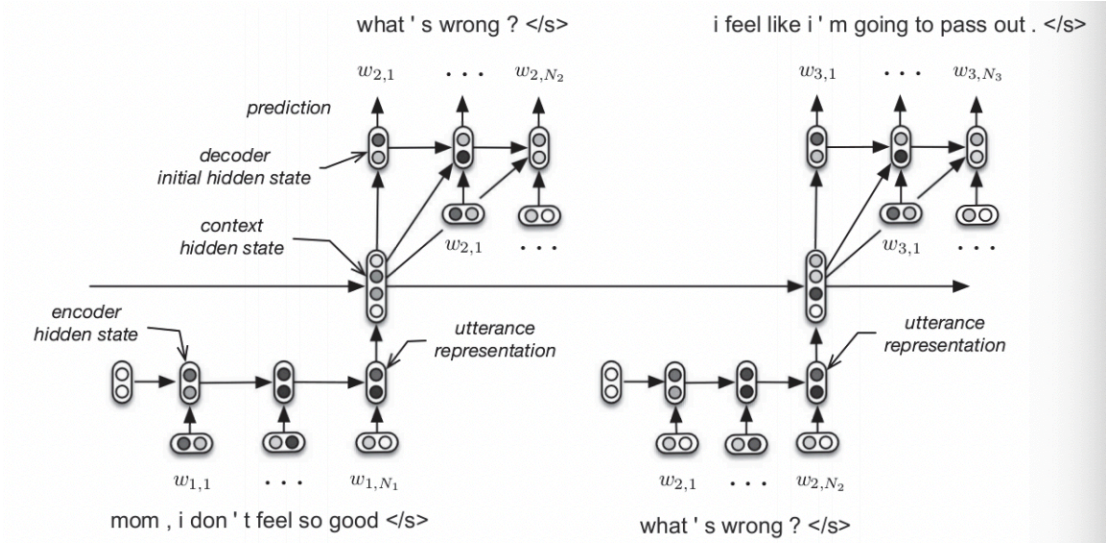
End-to-End Task Oriented System Nowadays



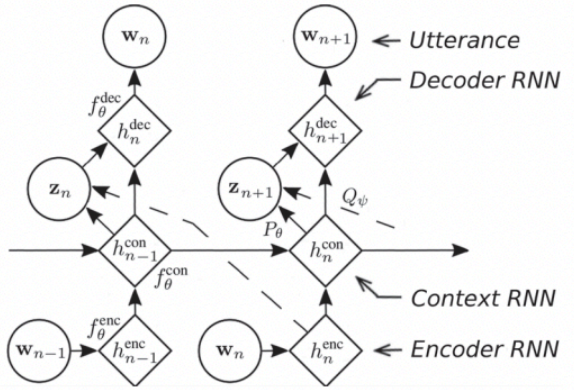
Wen, Tsung-Hsien, et al. "A network-based end-to-end trainable task-oriented dialogue system." arXiv preprint arXiv:1604.04562 (2016).

End-to-End Open Domain Dialogue System

- Retrieval Based
  - Semantic Matching (DSSM, …)
- Generation Based
  - LSTM (Lowe, Ryan, et al 2015)
  - HRED (Serban, Iulian Vlad, et al. 2016)
  - VHRED (Serban, Iulian Vlad, et al. 2017)
  - …
- Limitations
  - Generalized Response (Lol, that’ s great, that’ s good ….)
  - Lack of Consistency (I have a daughter, I am 12…)
  - Lack of Coherence (Where are you from? I like to play football…)
  - Lack of Goal

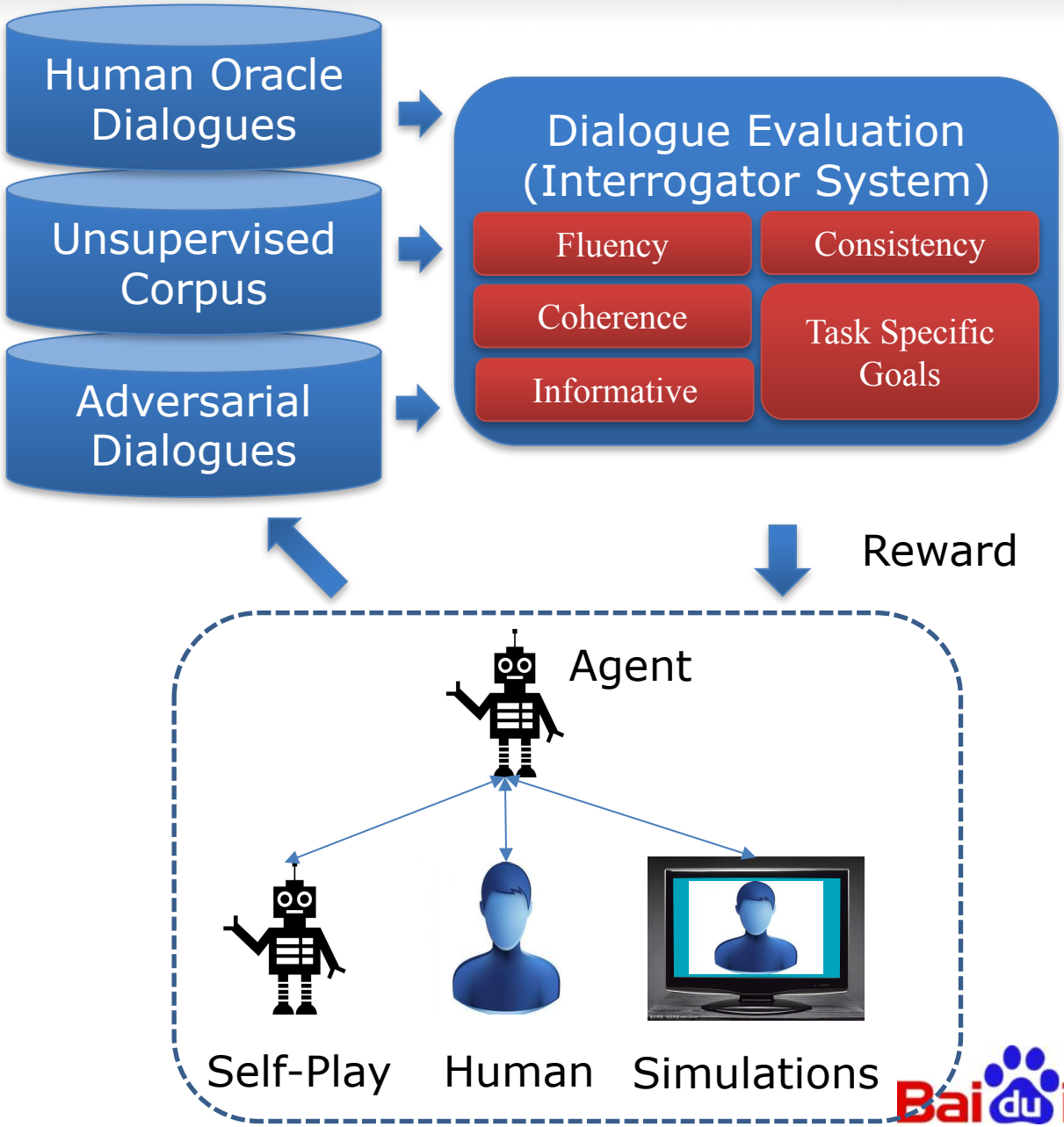


Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED)



Self-Evolving Dialogue System

- Challenge
  - Synthesizing all the requirements in Supervised Learning is very trick and difficult
  - Reinforcement Learning misses explicit goal for learning
  - User feedback is expensive, RL requires huge amount of data
- Previous work : Learning from artificial rewards
  - User Simulator (Casanueva et al. 2017)
  - Self-Play (Li, Monroe et al. 2016)
  - Adversarial Self-Play (*Li, Monroe et al. 2017*)
- Towards Automatic Dialogue Evaluation (In progress)



Persona Chat* based on Self-Evolving Dialogue System

Person A

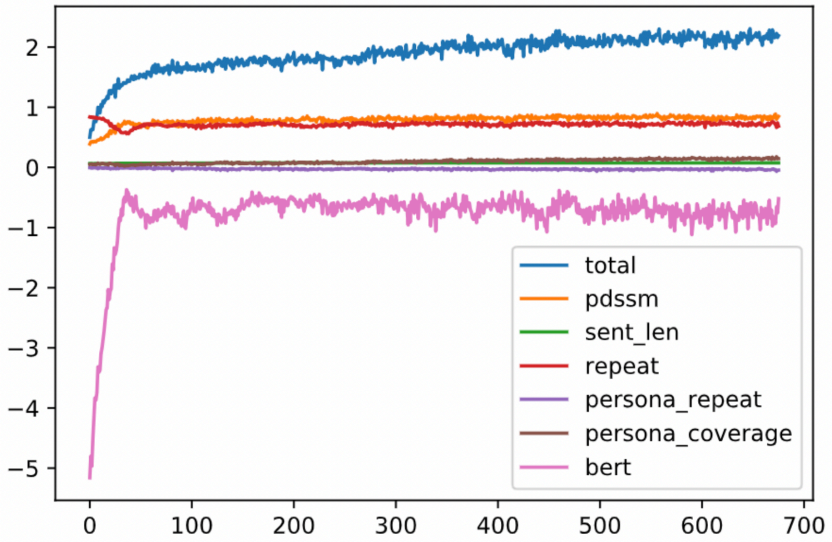
My family lives across the country from me
I graduated with a business degree, and marketing is all that I really enjoyed
I am recent college graduate looking for a job in marketing
I like spending time with friends playing sports and going to country concerts

Person B

My mom is a cancer survivor
My favorite subject in school was science
Pizza is my favorite food
My favorite hobby is reading suspense novels

Self-Play Dialogues

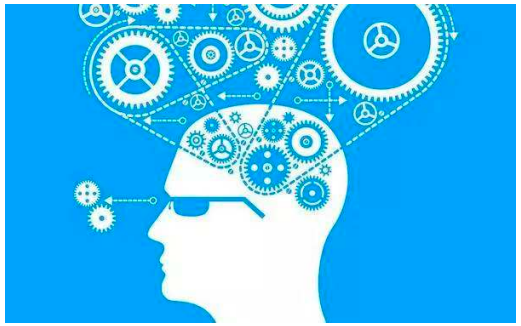
B: Hey hey how's it going today ?
A: Doing well. What about yourself ?
B: I am a science student
A: Cool. Do you like the marketing business ?
B: Nah. I like to eat pizza with my mother .
A: Yum. I am a country guy.
B: Cool. Do you like the suspense novel?
A: Nah. I like to play with my friends with my upcoming parties mates.



Rewards in Self-Play Training

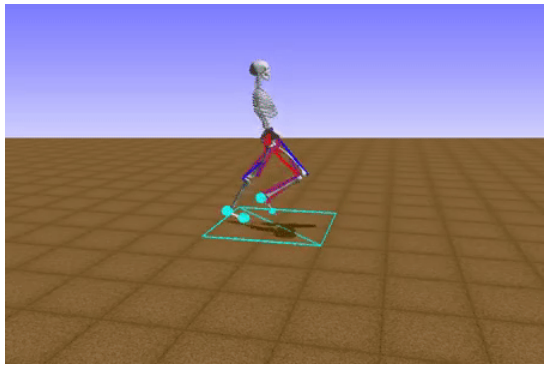
* Zhang, Saizheng, et al. "Personalizing Dialogue Agents: I have a dog, do you have pets too?." *arXiv preprint arXiv:1801.07243* (2018).
"Self-Evolving Dialogue System with Adversarial Safe Automatic Evaluation" (In progress)

Content Overview



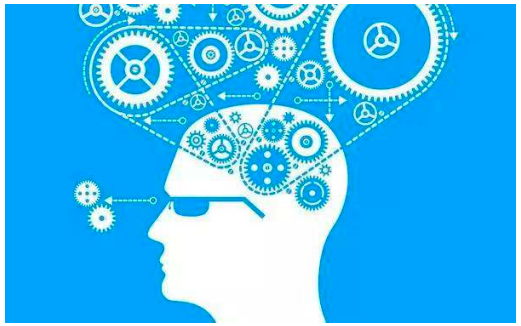
Algorithms

Tools & Platforms



Applications

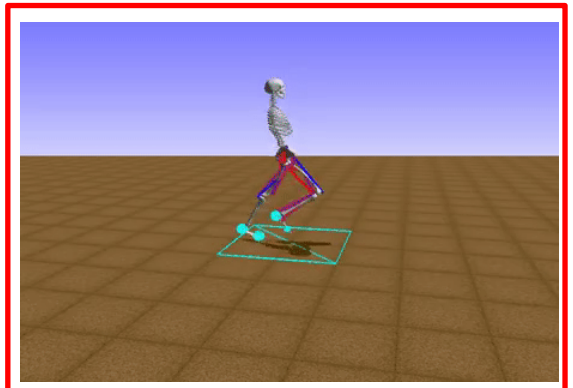
Content Overview



Algorithms

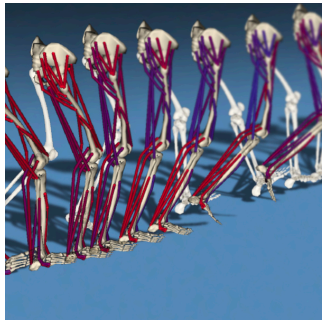
Tools & Platforms

Simulation & Control



Applications

Challenge Setting

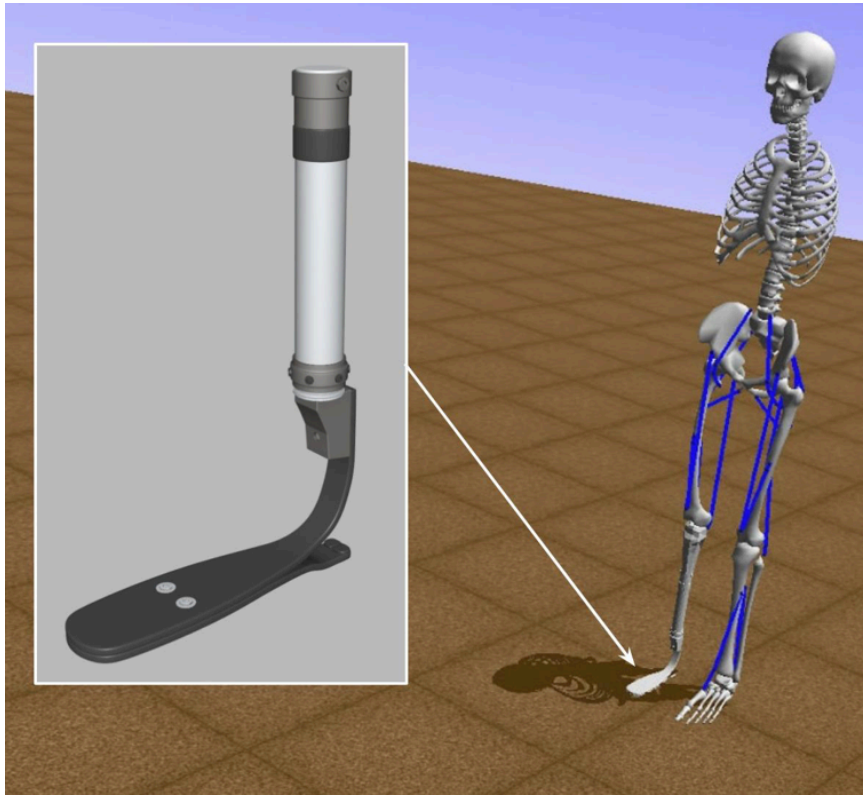


NeurIPS 2018: AI for Prosthetics Challenge

Reinforcement learning with musculoskeletal models

mobilize
Center for Mobility Data Integration to Insight

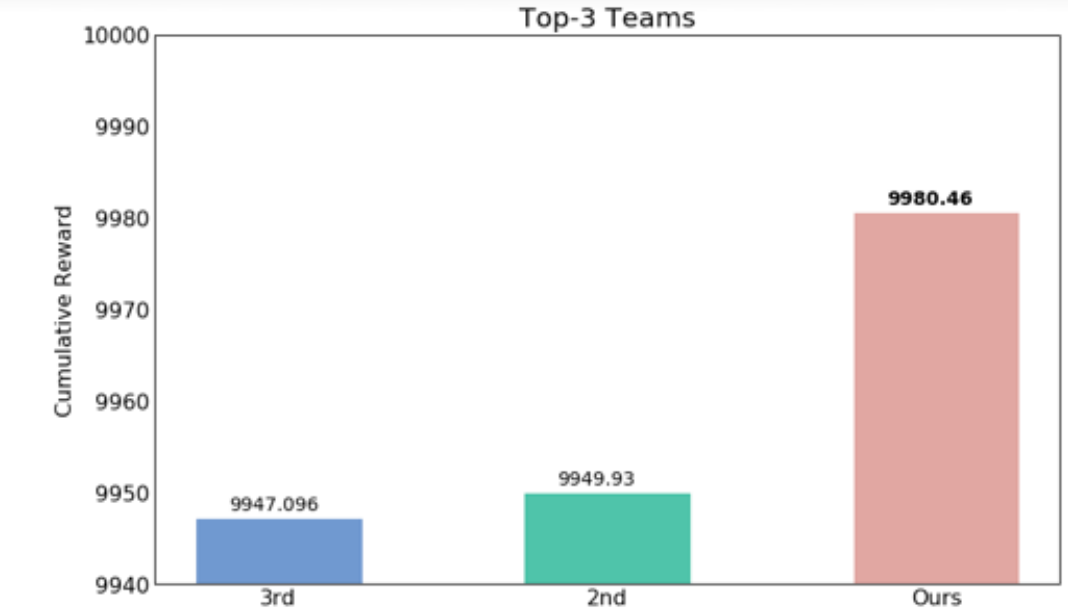
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

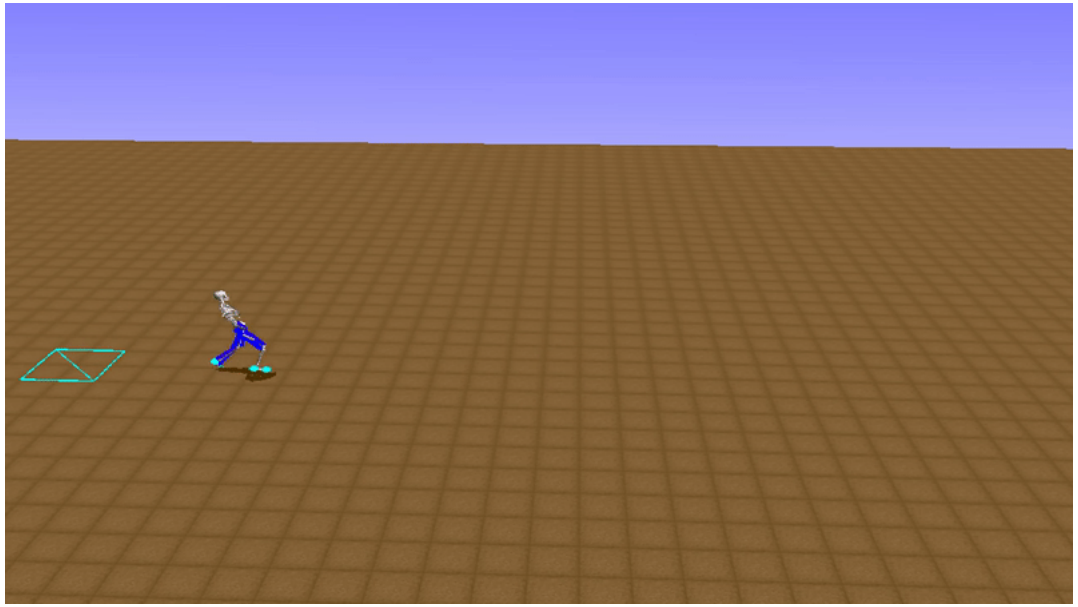


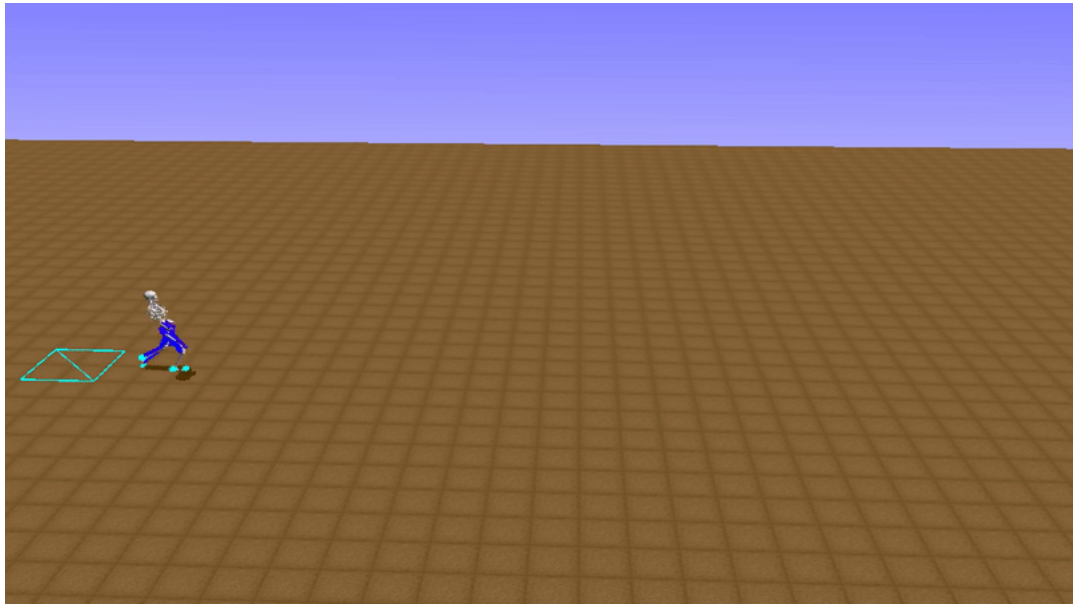
- 3D Dynamics (2D Dynamics in 2017 Competition)
- 100+ Observations and 19 Muscles
- Prosthetic Leg with no Muscles
- Elaborated Velocity Control Following External Specifications (Relatively Low Speed)

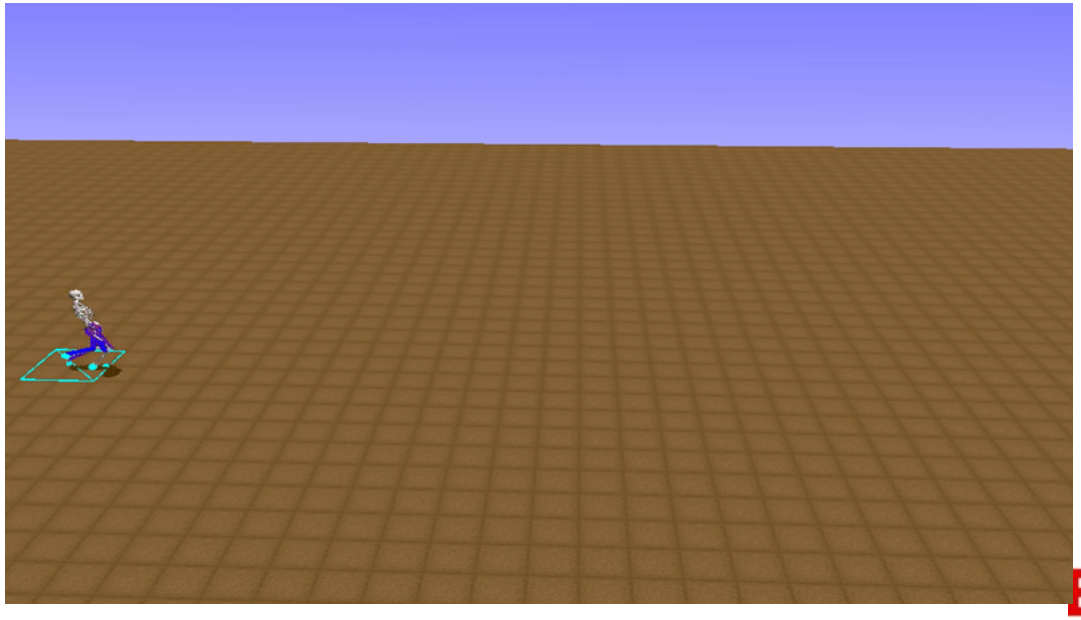
Baidu 百度

Our Submission (Final Round)









Baidu 百度

Keys to High Performance

- Model Structure
- Curriculum Learning
- Deep Exploration
- Parallelization

Keys to High Performance

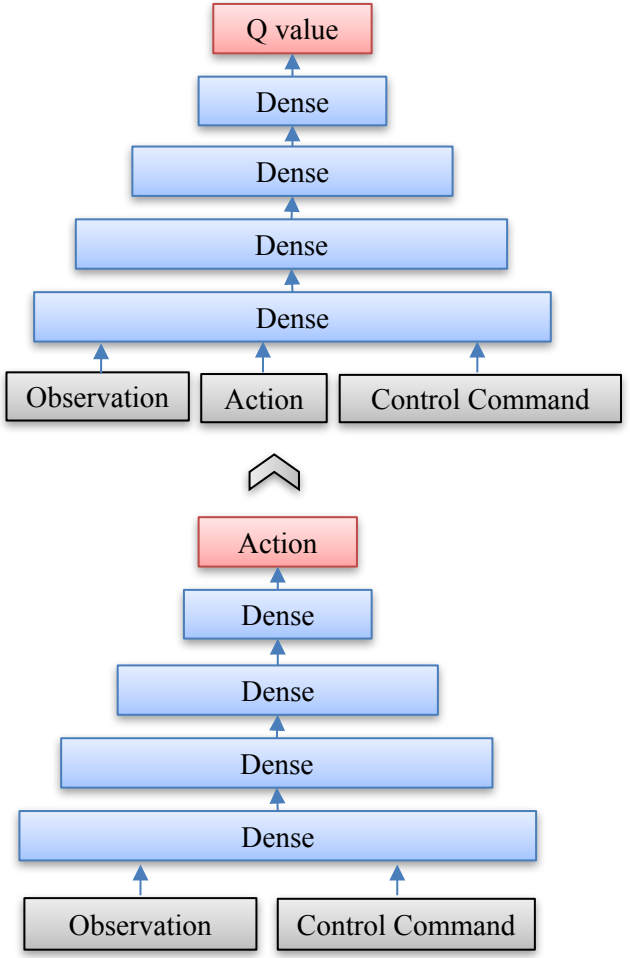
- Deep Deterministic Policy Gradient (Lillicrap, 2015)
  - 4-Layer MLP
  - Target-Driven Learning (Velocity Target)

Model Structure

Curriculum Learning

Deep Exploration

Parallelization



Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971(2015).

Keys to High Performance

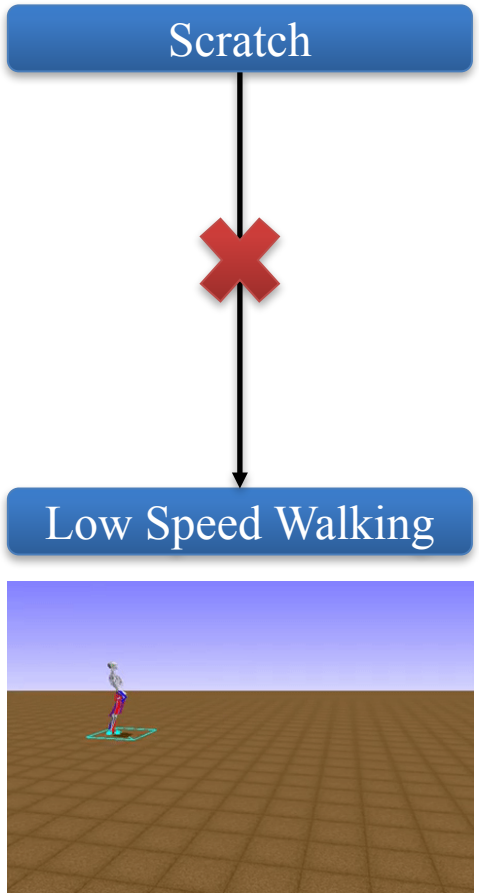
- Curriculum Learning
  - Learning from scratch in lower speed results in poor performance and weird gaits

Model Structure

Curriculum Learning

Deep Exploration

Parallelization



Keys to High Performance

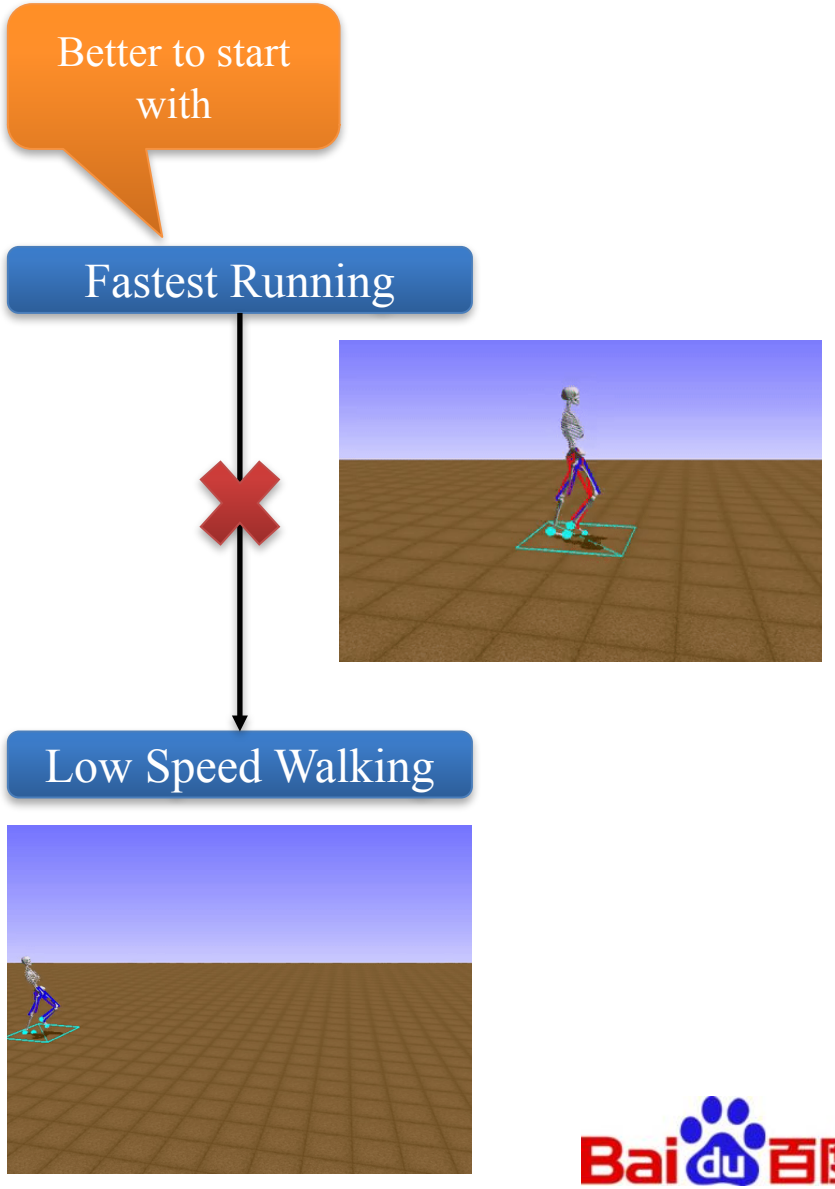
- Curriculum Learning
  - Learning from scratch in lower speed results in poor performance and weird gaits
  - **Learning to run very fast is easy to train and results in graceful gaits.**

Model Structure

Curriculum Learning

Deep Exploration

Parallelization



Keys to High Performance

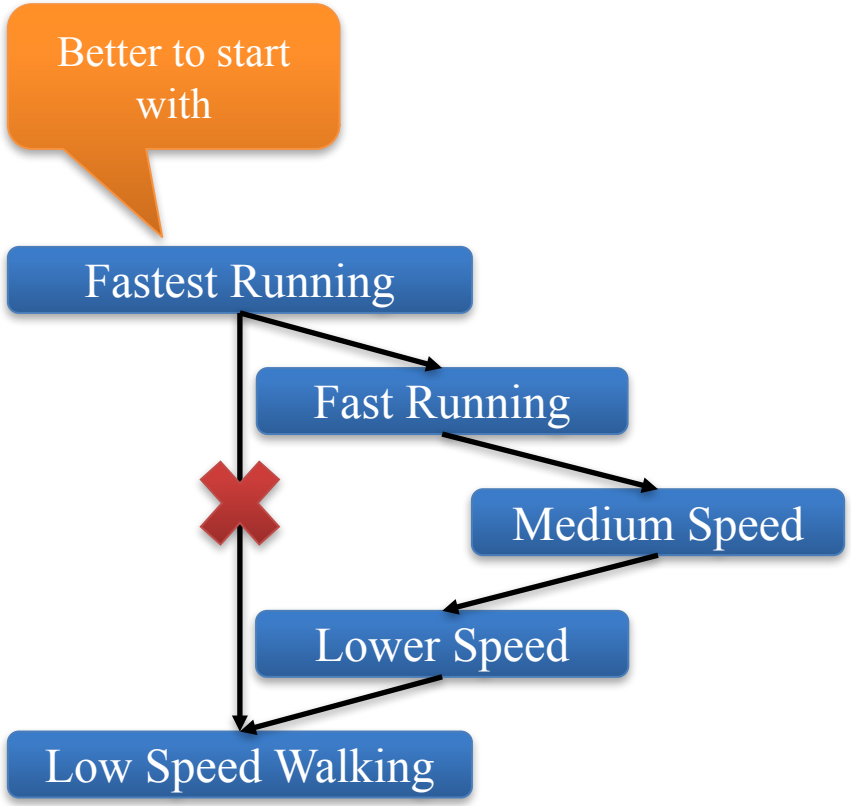
- Curriculum Learning
  - Learning from scratch in lower speed results in poor performance and weird gaits
  - **Learning to run very fast is easy to train and results in graceful gaits.**
  - Adapting from higher speed to lower one through step-by-step transferring (Bengio, 2009)

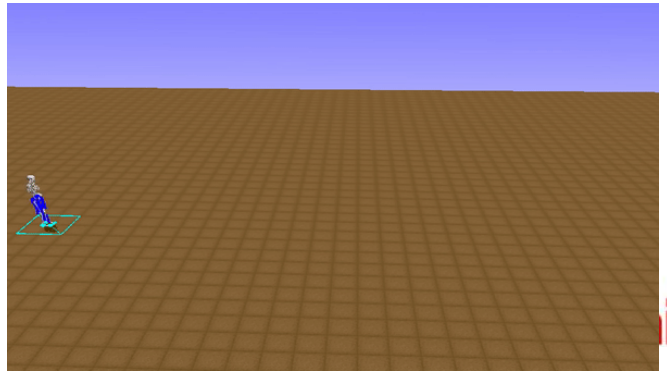
Model Structure

Curriculum Learning

Deep Exploration

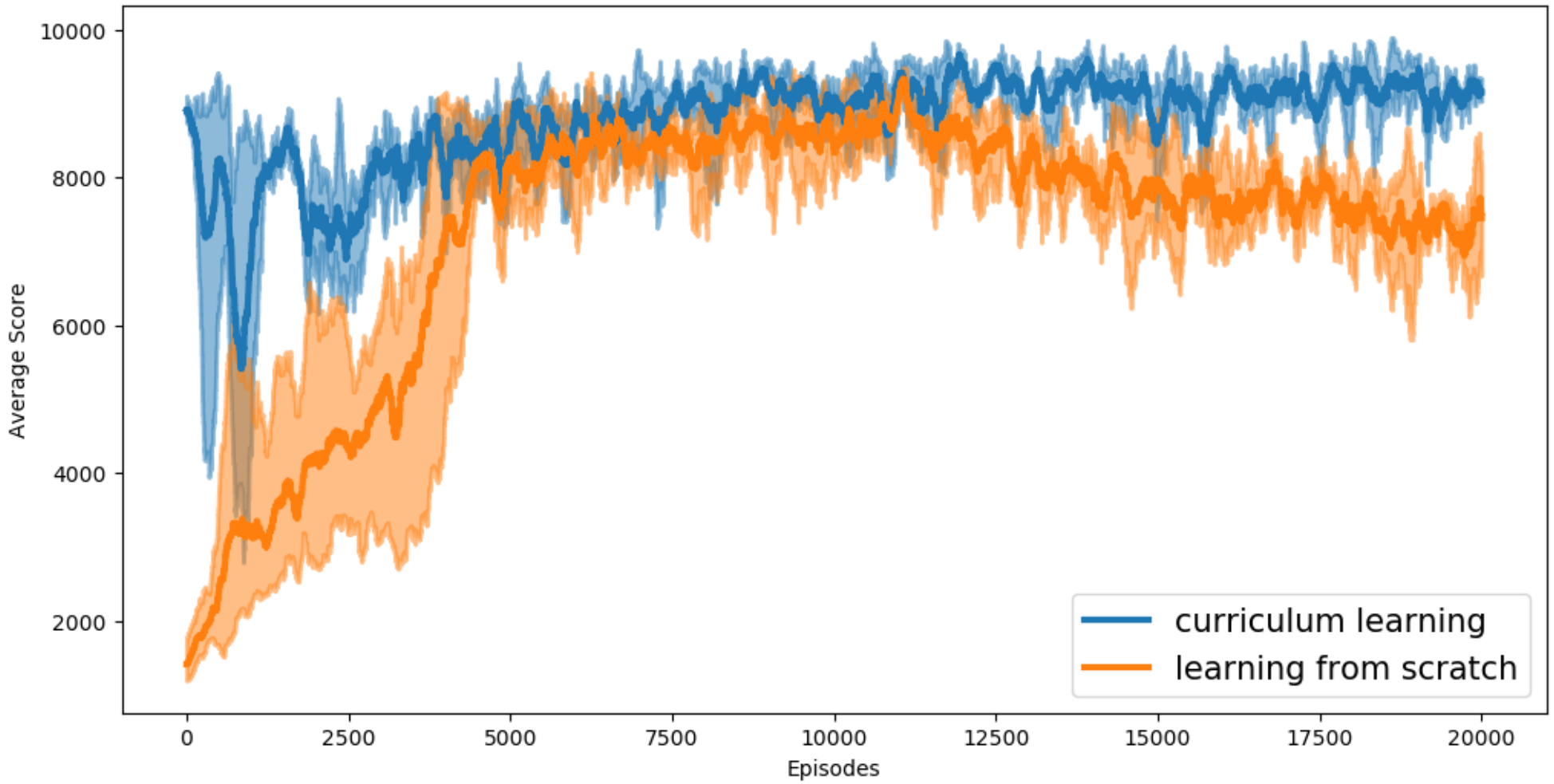
Parallelization





Bengio, Yoshua, et al. "Curriculum learning." *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.

Keys to High Performance



Keys to High Performance

- Deep Exploration (DE)
  - Random Exploration is too inefficient for large solution space
  - Deep exploration with multi-head bootstrapping (Osband, 2016)

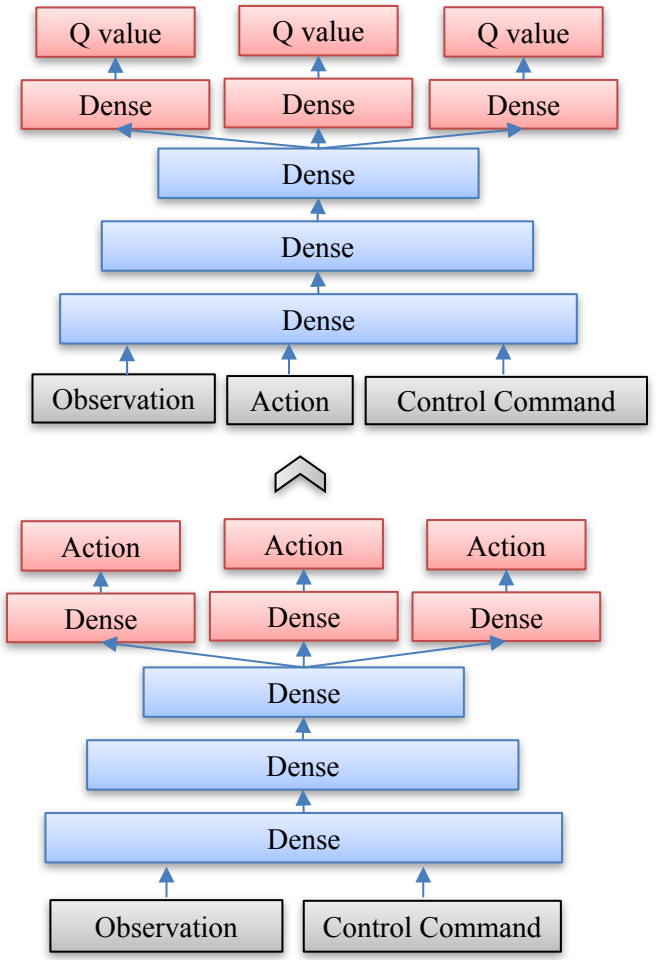
Algorithm	Data Efficiency	Robustness
DDPG	Strong	Weak
PPO	Weak	Strong
DDPG + DE	Strong	Strong

Model Structure

Curriculum Learning

Deep Exploration

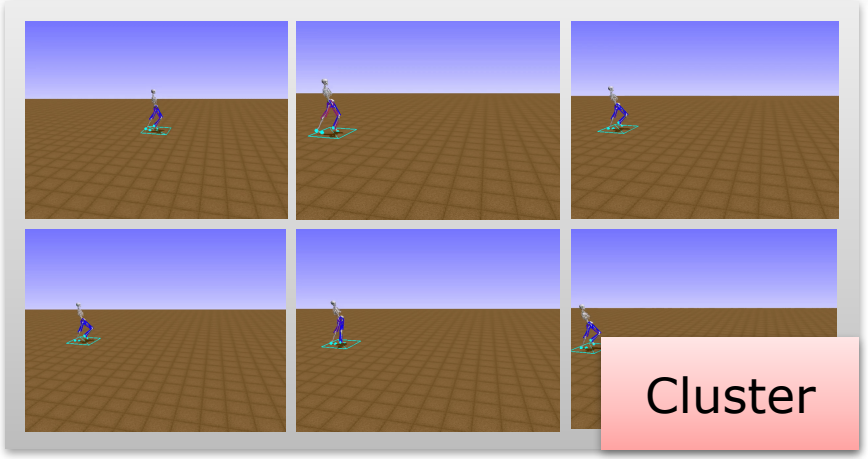
Parallelization



Ian Osband, Charles Blundell, Alexander Pritzel, Benjamin Van Roy, Deep Exploration via Bootstrapped DQN, In NIPS 2016

Keys to High Performance

- Speed up training by
  - Multi-computer parallelization of environments deployed on Hadoop cluster with RPC protocol
  - Async training to imporve utilization across GPU/CPU/network/IO
  - Adatipve balance between the speed of data generation and model training



Model Structure

Curriculum Learning

Deep Exploration

Parallelization

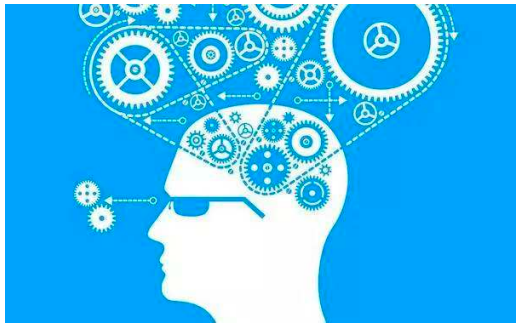
Data Server (Replay Memory) — CPU

Training & Inference — GPU

Testing Episodes	Fall Percentage(%)	Scores	Training Time
5000	0.16	9980.6183	128 CPU (2 days)

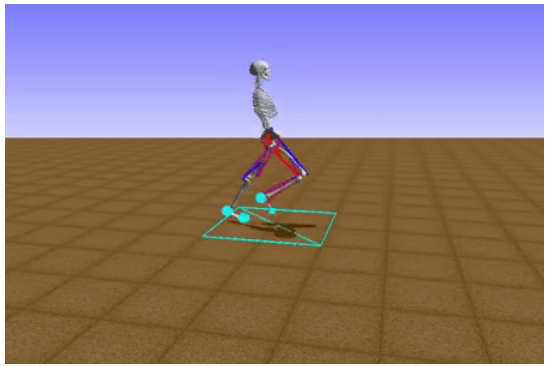
The Final Performance

Content Overview



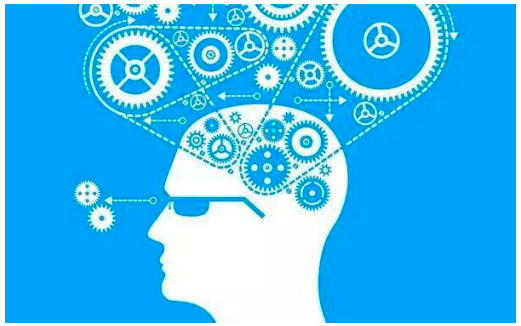
Algorithms

Tools & Platforms



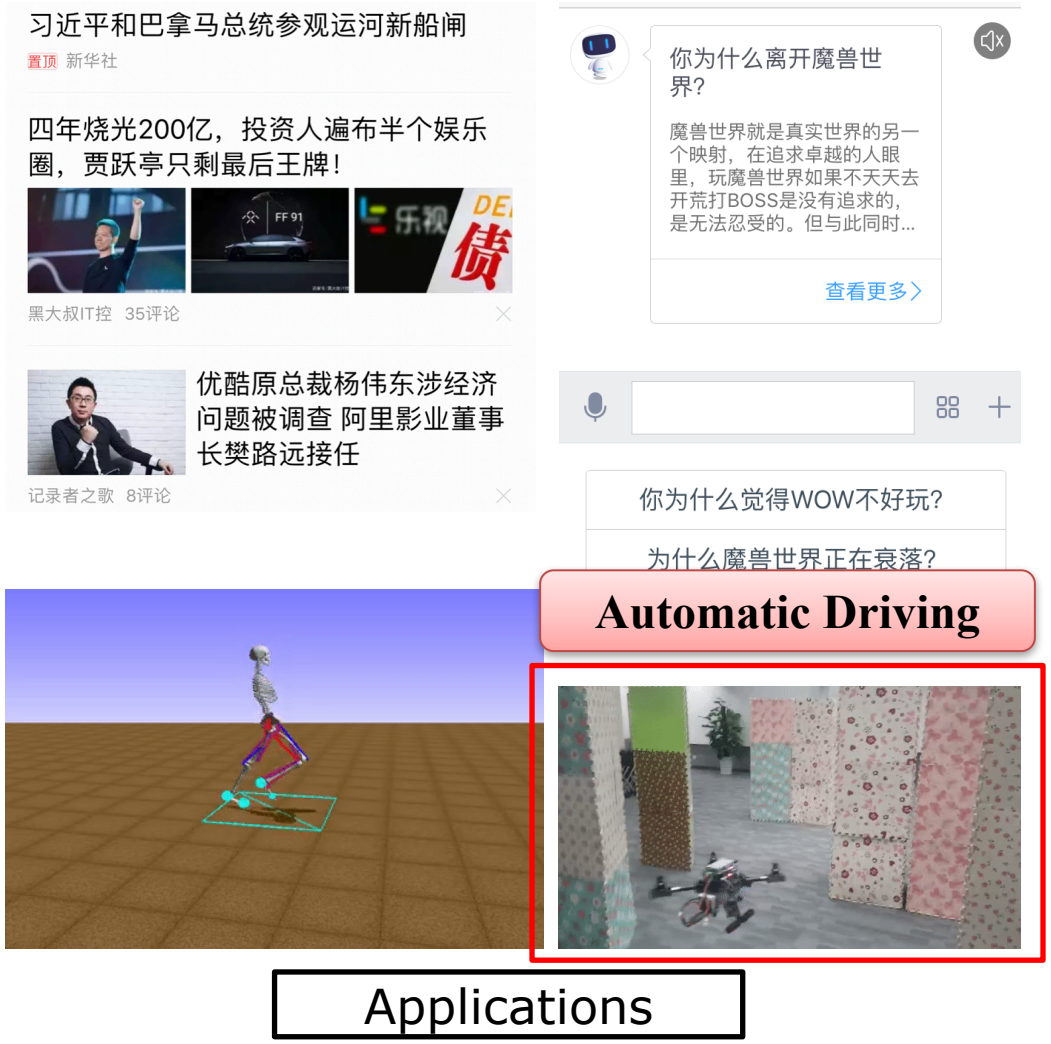
Applications

Content Overview



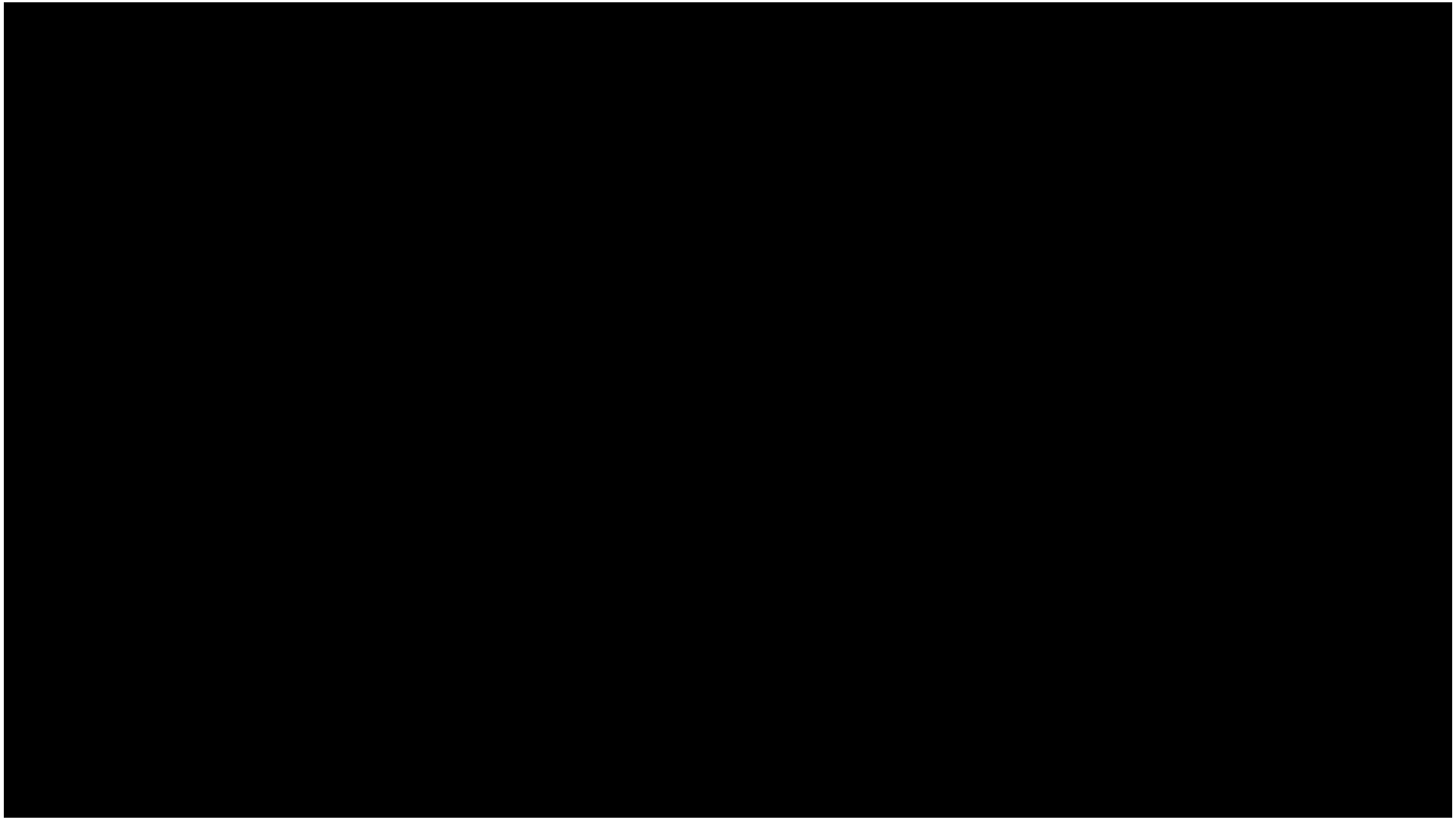
Algorithms

Tools & Platforms



Applications

Intervention Aided Reinforcement Learning



Motivation

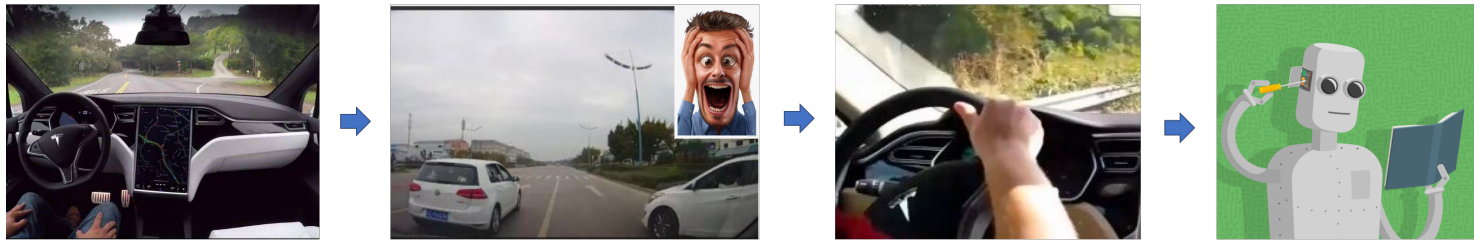
- Deploying RL to high risk real world control system faces challenges in terms of safety and cost
- We use human intervention to ensure safety during the **training process**
- We propose the IARL framework for policy optimization based on the **intervened trajectory**

Autonomous Driving → Crash → Learning

RL in Simulation

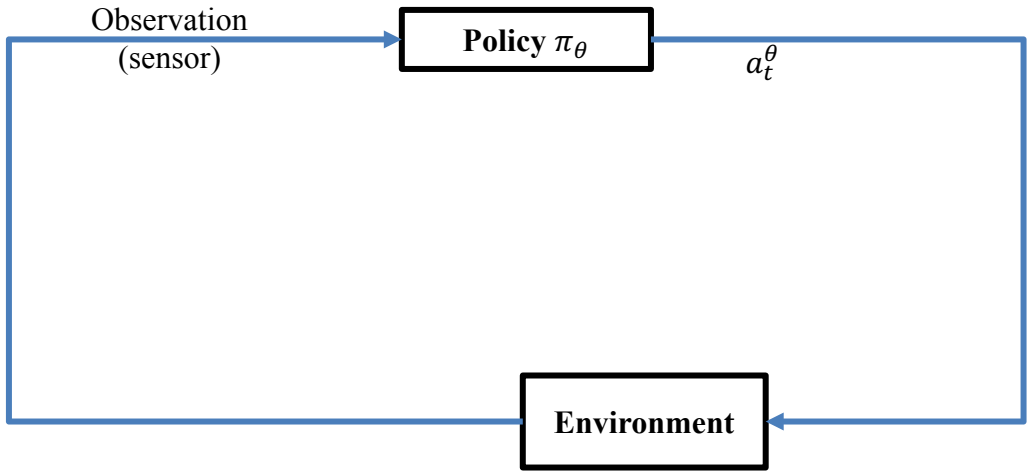
v.s.

IARL in Reality

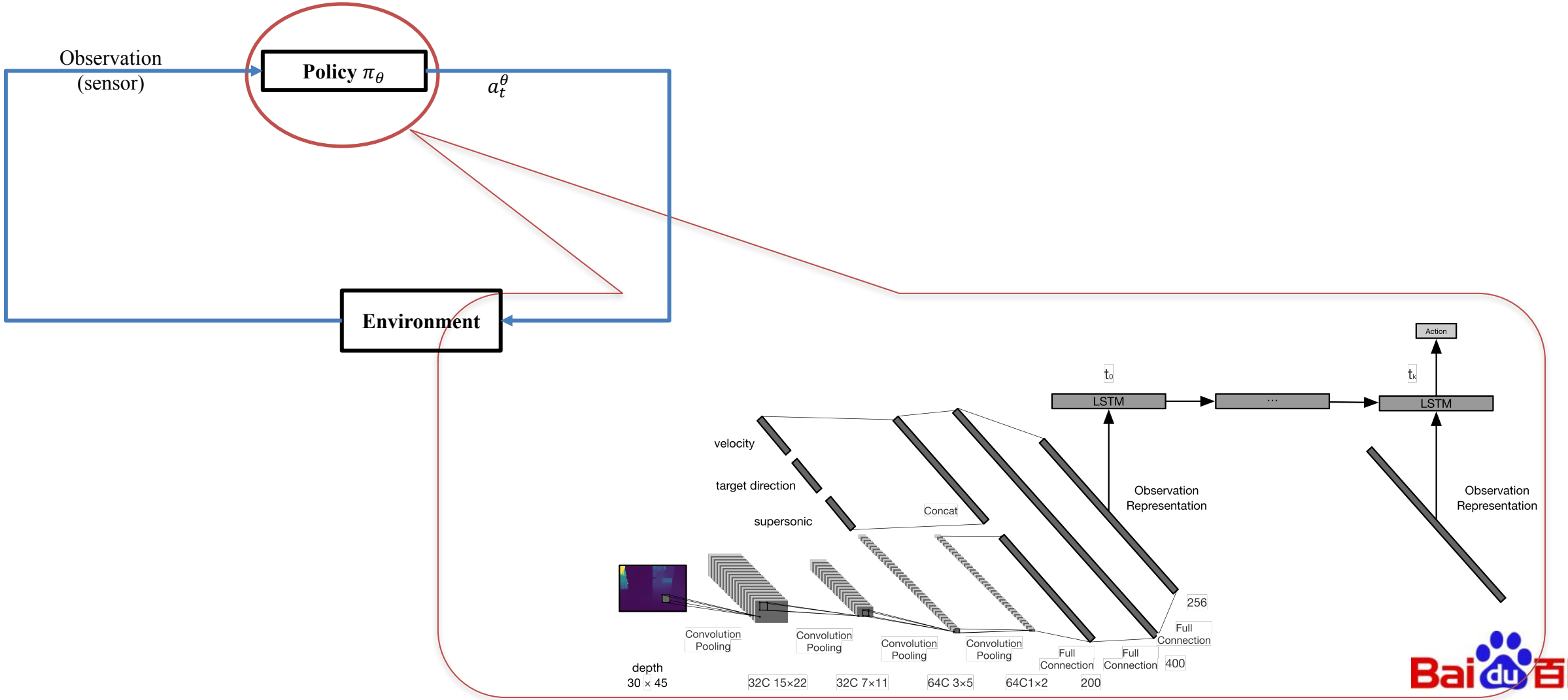


Autonomous Driving → Risky States → Intervention → Learning

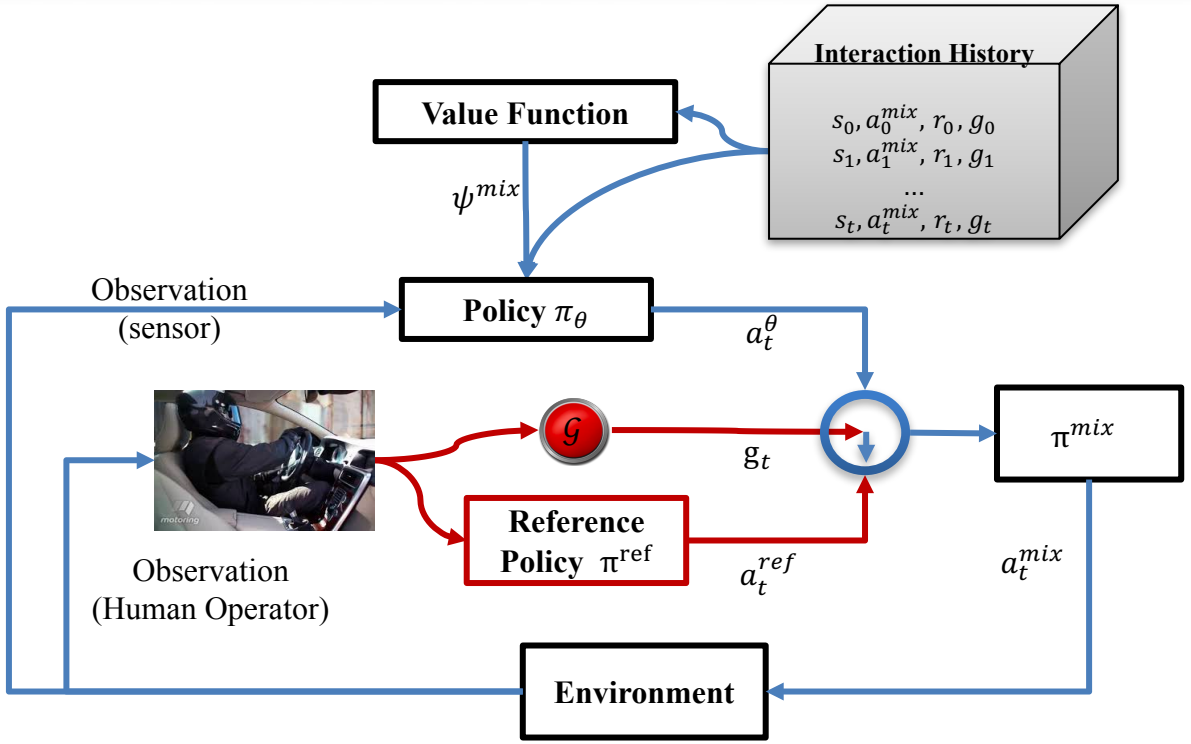
The IARL Framework

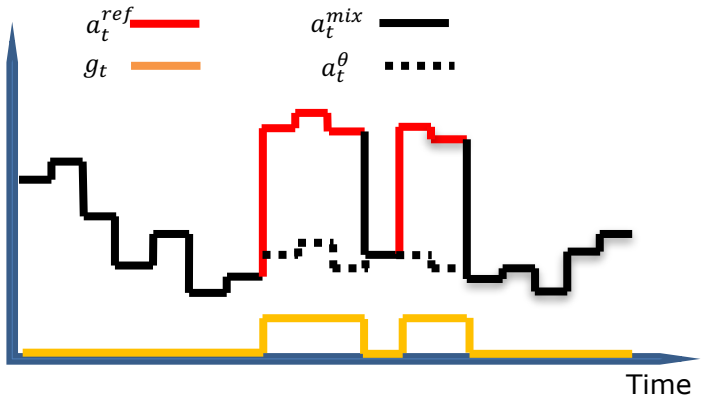


The IARL Framework



The IARL Framework





Learning Algorithm

- Reward Reshaping

$$r_t^{mix} = r_t - b\, g_t$$

Goal Driven Reward — r_t

Intervention Punishment — $b\, g_t$

- Behavior Policy Formulation

$$\pi_\theta^{mix}(a|s_t) = (1 - \mathcal{G}(s_t))\pi_\theta(a|s_t) + \mathcal{G}(s_t)\pi^{ref}(a|s_t)$$

- Loss Function

$$L^{IARL} \propto -E_{\pi_\theta^{mix}}[\sum_{g_t=0} \psi_t^{mix} \log \pi_\theta(a_t^{mix}|s_t) + \alpha \sum_{g_t=1} \log \pi_\theta(a_t^{mix}|s_t)]$$

Learning Algorithm

- Reward Reshaping

$$r_t^{mix} = r_t - b\, g_t$$

Goal Driven Reward — r_t

Intervention Punishment — $b\, g_t$

- Behavior Policy Formulation

$$\pi_\theta^{mix}(a|s_t) = (1 - \mathcal{G}(s_t))\pi_\theta(a|s_t) + \mathcal{G}(s_t)\pi^{ref}(a|s_t)$$

- Loss Function

$$L^{IARL} \propto -E_{\pi_\theta^{mix}}[\sum_{g_t=0} \psi_t^{mix} \log \pi_\theta(a_t^{mix}|s_t) + \alpha \sum_{g_t=1} \log \pi_\theta(a_t^{mix}|s_t)]$$

Intervention Avoidance — $\sum_{g_t=0} \psi_t^{mix} \log \pi_\theta(a_t^{mix}|s_t)$

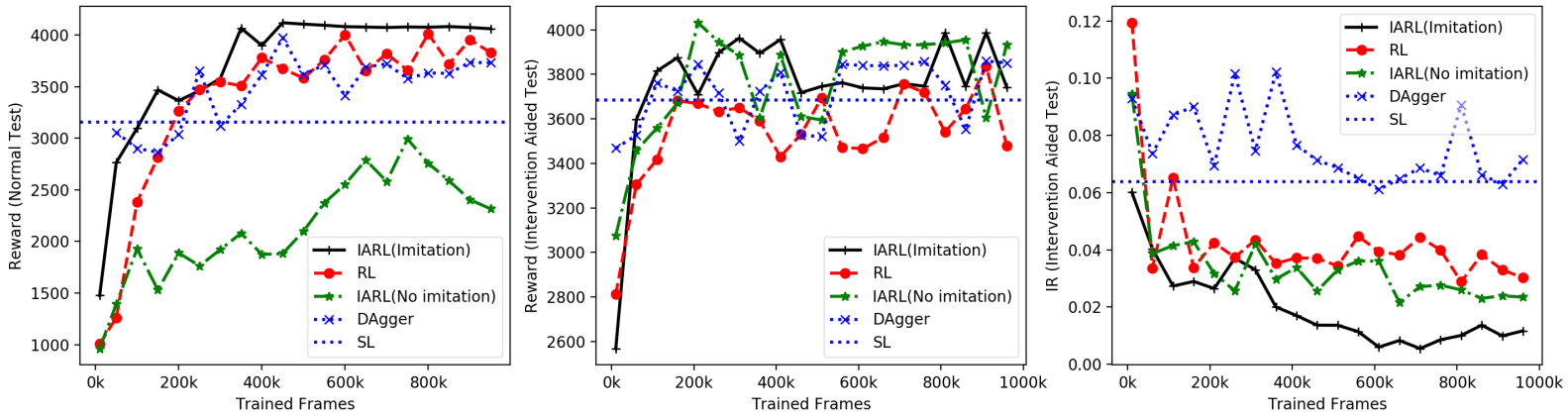
Reference Imitating — $\alpha \sum_{g_t=1} \log \pi_\theta(a_t^{mix}|s_t)$

Results & Conclusion

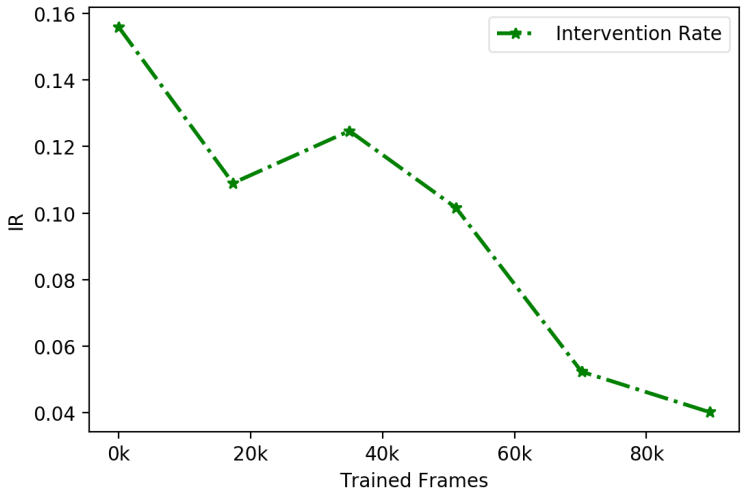
- IARL learns to avoid collision **without** any collision during the training process
- IARL outperforms the baselines in collision rate, rewards, and Intervention Rate.
- Practical for various high risk real world tasks.

Table 1: Average collision in *Normal Test* in different groups

Method	SL	DAgger	RL	IARL(No Imitation)	IARL(Imitation)
Average Collision	40%	24%	22.5%	42%	0%



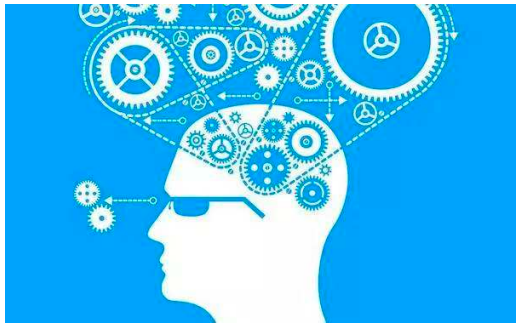
Simulations



Reality

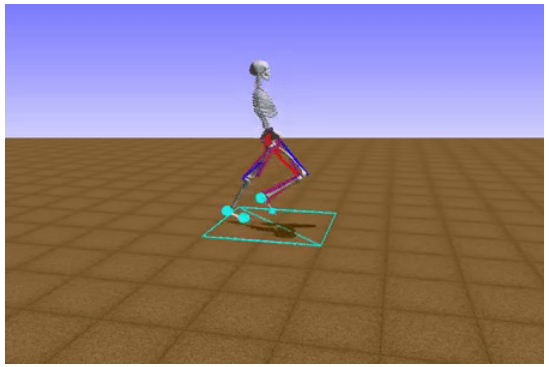
Wang, F., Zhou, B., Chen, K., Fan, T., Zhang, X., Li, J., ... & Pan, J. (2018, October). Intervention Aided Reinforcement Learning for Safe and Practical Policy Optimization in Navigation. In *Conference on Robot Learning* (pp. 410-421).

Content Overview



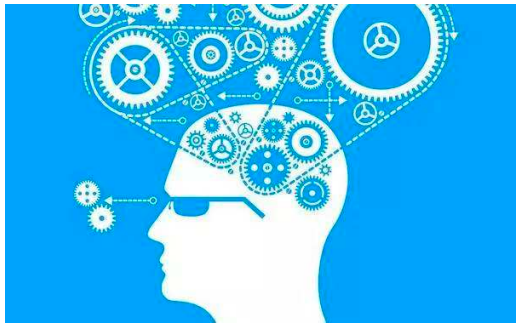
Algorithms

Tools & Platforms



Applications

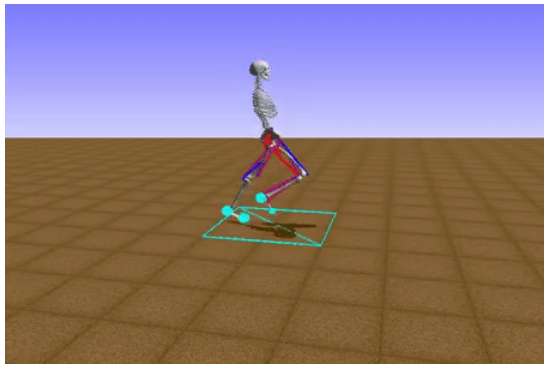
Content Overview



Algorithms

Tools & Platforms

PARL

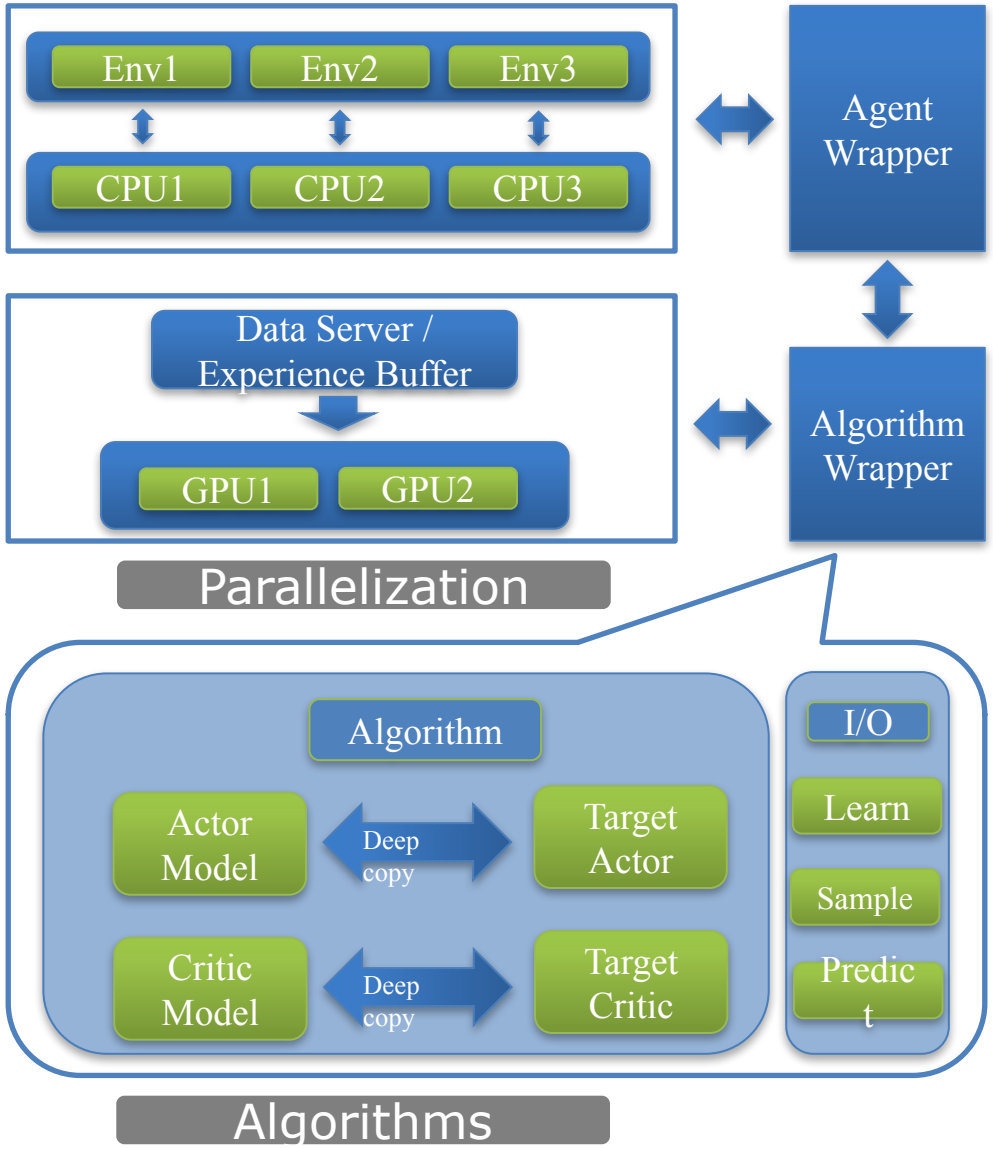


Applications

Baidu 百度

PARL - Flexible, Full Coverage and Large Scale RL Platform

- PARL
  - PAddlePaddle Reinforcement Learning Platform
- Features
  - Flexible
  - Large Scale Distributional
  - Validated for Industrial Applications & Research



https://github.com/PaddlePaddle/PARL/

PARL - DQN – Customizing Model

```
import parl
from parl.algorithms import DQN, DDQN

class AtariModel(parl.Model):
        """AtariModel
        This class defines the forward part for an algorithm,
        its input is state observed on environment.
        """
        def __init__(self, img_shape, action_dim):
                # define your layers
                self.cnn1 = layers.conv_2d(num_filters=32, filter_size=5,
                                           stride=[1, 1], padding=[2, 2], act='relu')
                ...
                self.fc1 = layers.fc(action_dim)
        def value(self, img):
                # define how to estimate the Q value based on the image of atari games.
                img = img / 255.0
                l = self.cnn1(img)
                ...
                Q = self.fc1(l)
                return Q
"""
three steps to build an agent
   1.  define a forward model which is critic_model is this example
   2.  a. to build a DQN algorithm, just pass the critic_model to `DQN`
       b. to build a DDQN algorithm, just replace DQN in following line with DDQN
   3.  define the I/O part in AtariAgent so that it could update the algorithm based on the interactive data
"""

model = AtariModel(img_shape=(32, 32), action_dim=4)
algorithm = DQN(model)
agent = AtariAgent(aglrotihm)
```

PARL – Customizing Algorithm

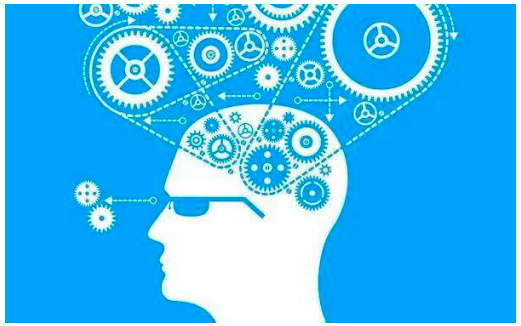
```python
class DDPG(Algorithm):
    def __init__(self, model, hyperparas):
        """ model: should implement the function get_actor_params()
        """
        Algorithm.__init__(self, model, hyperparas)
        self.model = model
        self.target_model = deepcopy(model)

        # fetch hyper parameters
        self.gamma = hyperparas['gamma']
        self.tau = hyperparas['tau']
        self.actor_lr = hyperparas['actor_lr']
        self.critic_lr = hyperparas['critic_lr']

    def define_predict(self, obs):
        """ use actor model of self.model to predict the action
        """
        return self.model.policy(obs)

    def define_learn(self, obs, action, reward, next_obs, terminal):
        """ update actor and critic model with DDPG algorithm
        """
        actor_cost = self._actor_learn(obs)
        critic_cost = self._critic_learn(obs, action, reward, next_obs,
                                         terminal)
        return actor_cost, critic_cost

    def _actor_learn(self, obs):
        action = self.model.policy(obs)
        Q = self.model.value(obs, action)
        cost = layers.reduce_mean(-1.0 * Q)
        optimizer = fluid.optimizer.AdamOptimizer(self.actor_lr)
        optimizer.minimize(cost, parameter_list=self.model.get_actor_params())
        return cost

    def _critic_learn(self, obs, action, reward, next_obs, terminal):
        next_action = self.target_model.policy(next_obs)
        next_Q = self.target_model.value(next_obs, next_action)

        terminal = layers.cast(terminal, dtype='float32')
        target_Q = reward + (1.0 - terminal) * self.gamma * next_Q
        target_Q.stop_gradient = True

        Q = self.model.value(obs, action)
        cost = layers.square_error_cost(Q, target_Q)
        cost = layers.reduce_mean(cost)
        optimizer = fluid.optimizer.AdamOptimizer(self.critic_lr)
        optimizer.minimize(cost)
        return cost

    def sync_target(self, gpu_id, decay=None):
        if decay is None:
            decay = 1.0 - self.tau
        self.model.sync_params_to(
            self.target_model, gpu_id=gpu_id, decay=decay)
```

PARL - Customizing Parallelization

```
"""
define your simulator inside the client, it will communicate with server automatically.
And run it on different CPUs.
"""
from parl.parallel import Client
import simulator_pb2
import simulator_pb2_grpc

class AtariEnv(Client):
        def __init__(self, host):
                Client.__init__(self, host)
                env = gym.env('breakout-v0')
                env = FrameSkip(env, k=4)
                env = RewardClip(env, min=-1, max=1)
if __name__ == '__main__':
        host = sys.argv[1]
        sim = AtariEnv(host)
        sim.run()


"""
 define the server in another script, you only need to specify the function used to prediction for server class.
 once you launch the server, it will communicate with all clients and collect trajectory inside server.
"""

from parl.framework import ParallelServer

server = ParallelServer()
# register predict function inside server
server.register(atari_agent.predict)
server.launch(max_buffer_size=128)

#Now, you have connect the clients with this server, you can get data by API: server.get_batch()
#than you can write GA3C like writing other general RL algorithms.
```

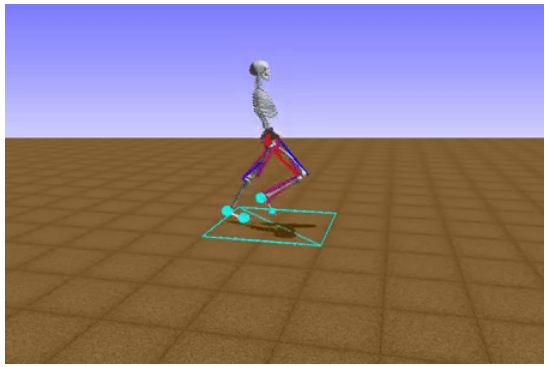
Baidu 百度

Content Overview



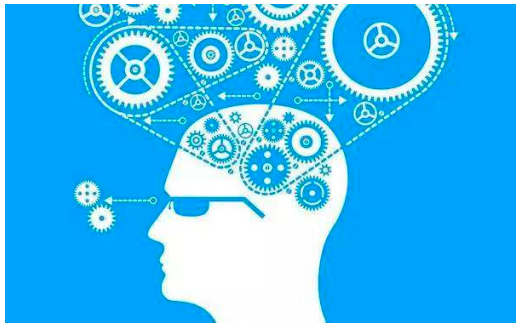
Algorithms

Tools & Platforms



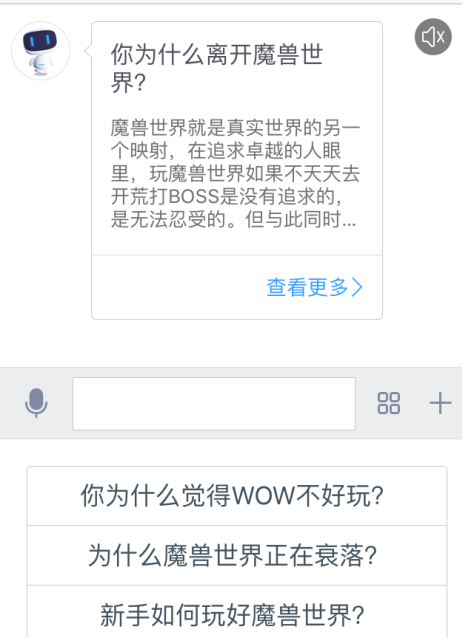
Applications

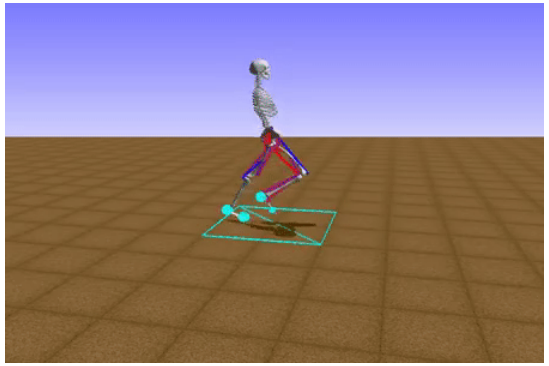
Content Overview



Algorithms

Tools & Platforms

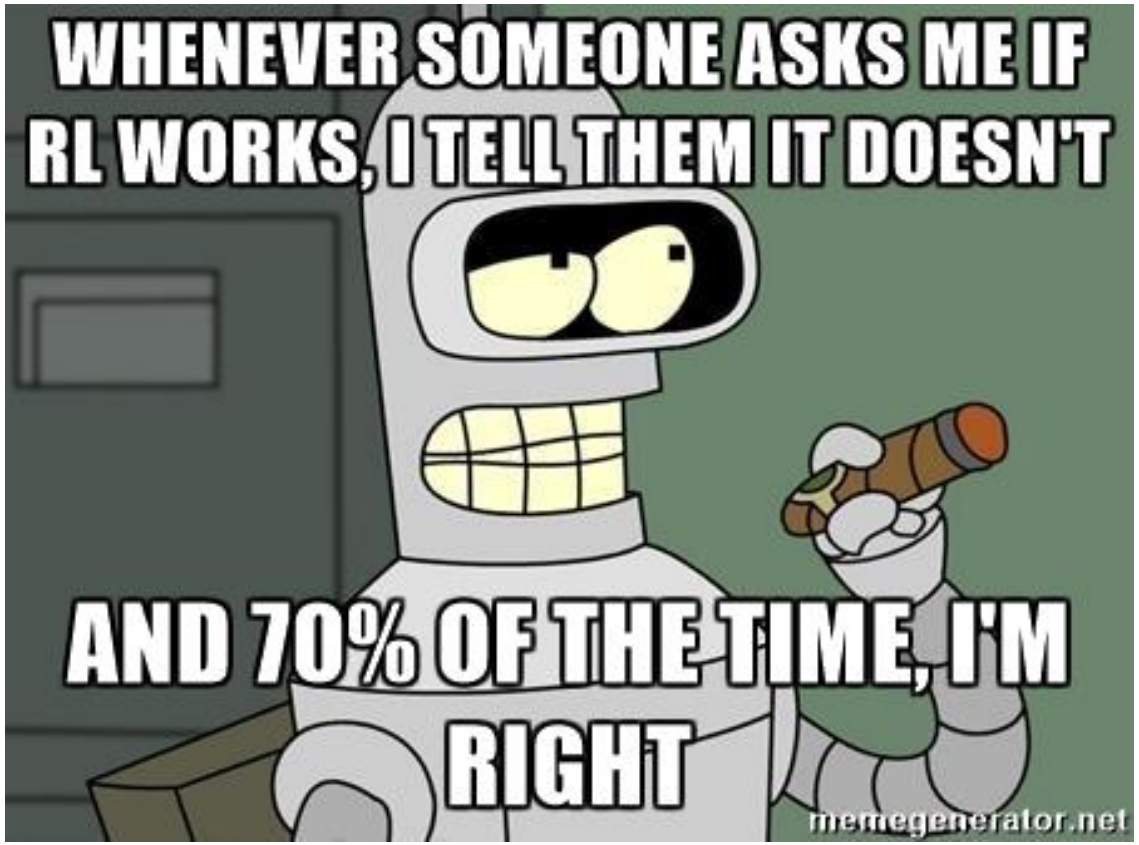




Applications

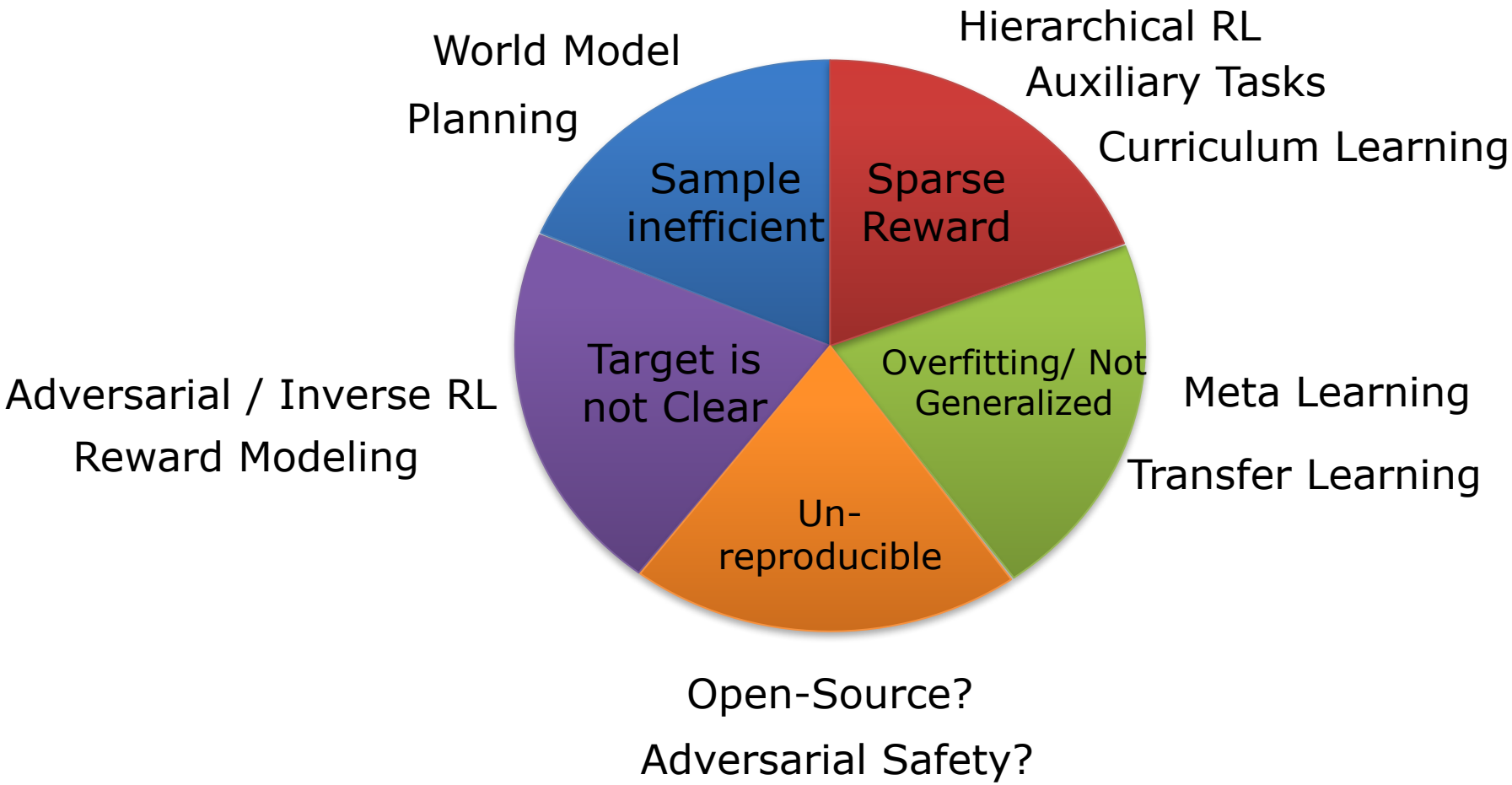
Future Challenges?

Deep Reinforcement Learning – Future Challenges



https://www.alexirpan.com/2018/02/14/rl-hard.html

Deep Reinforcement Learning – Future Challenges



Thanks

Baidu 百度