Principles of Deep RL

David Silver

Principle #1: **Evaluation** Drives Progress

Objective, **quantitative** evaluation drives progress:

- The choice of evaluation metric determines the direction of progress
- Arguably the most important single decision in the course of a project

Leaderboard-driven research:

- Be sure the evaluation metric corresponds closely to the end goal
- Avoid subjective evaluation (e.g. human inspection)

Hypothesis-driven research:

- Formulate a hypothesis:
  - “Double-Q learning outperforms Q-learning because it reduces upward bias”
- Verify hypothesis under a broad range of conditions
- Compare like-for-like **not** against existing state-of-the-art
- Seek understanding rather than leaderboard performance

Principle #2: **Scalability** Determines Success

- An algorithm’s **scalability** is its performance gradient with respect to resource
  - Given more resource, how does performance increase?
- The resource could be **computation**, **memory** or **data**
- The scalability of an algorithm ultimately determines its success
  - Image
- Scalability is always (eventually) more important than the starting point
- A good algorithm is (eventually) optimal given infinite resources

Principle #3: **Generality** Future-Proofs Algorithms

- An algorithm’s **generality** is its performance across different RL environments
- Avoid overfitting to the present task
- Seek algorithms that will generalise to unknown, future environments
- We can’t predict the future but:
  - Future tasks are likely to be at least as complex as current tasks
  - Difficulties encountered in current tasks will most likely increase
- Conclusion: test against a diverse but realistic set RL environments

Principle #4: Trust in the Agent’s **Experience**

- Experience (*observations*, *actions*, *rewards*) is the data of RL
    - $h_t = o_1, r_1, a_2, o_2, r_2, ..., a_t, o_t, r_t$
    - Stream of experience accumulated over the course of the agent’s lifetime in the environment
- Trust in experience as the sole source of knowledge
    - The temptation is always there to leverage our human expertise (human data, features, heuristics, constraints, abstractions, domain manipulations)
- Learning from experience may seem impossible
    - Accept the core problem of RL is hard
    - It is the central problem of AI
    - It is worth the effort
- Learning from experience always wins in the long run

Principle #5: **State** is Subjective

- Agents should construct their own **state** from their experience
  - $s_t = f(h_t)$
- Agent state is a function of the previous state and the new observation
  - $s_t = f(s_{t-1}, a_{t-1}, o_t, r_t)$

$$
\begin{tikzcd}
 & a_t & \cdots \\
s_{t-1} \arrow[r] & s_t \arrow[u] \arrow[ur] \arrow[r] & s_{t+1} \\
 & a_{t-1}, o_t, r_t \arrow[u] &
\end{tikzcd}
$$

- It is the hidden state of a recurrent neural network
- Never defined in terms of the “real” state of the environment (a la POMDP)

Principle #6: **Control** the Stream

- Agents live in rich sensorimotor streams of data
  - Observations stream into the agent
  - Actions stream out of the agent
- The agent’s actions influence the stream

- Control of features => control of the stream
- Control of the stream => control of the future
- Control of the future => can maximise any reward

Principle #7: **Value Functions** Model the World

Why use a value function?

- Value functions efficiently summarise/cache the future
- Reduce planning to constant-time look-up, rather than exponential lookahead
- Can be computed and learned independent of their span

Learn multiple value functions:

- To efficiently model many aspects of the world (control the stream)
    - Including subsequent state variables
- At multiple time-scales

Avoid modelling the world at primitive time-step

Principle #8: **Planning:** Learn from Imagined Experience

An efficient approach to planning:

- Imagine what will happen next
  - Sample trajectory of states from the model
- Learn from imagined experience
  - Using the same RL algorithms that we apply to real experience

Focus the value function approximation on the moment *now*

Principle #9: Empower the **Function Approximator**

- Differentiable network architectures are powerful tools facilitating:
  - Rich representations of state
  - Differentiable memory
  - Differentiable planning
  - Hierarchical control
  - …
- Push algorithmic complexity into the network architecture
  - Reduce complexity of the algorithm (how parameters are updated)
  - Increase expressiveness of the architecture (what the parameters do)

Principle #10: **Learn to Learn**

The history of AI shows a clear direction of progress:

- Generation #1: Good Old-Fashioned AI
  - Handcraft predictions
  - Learn nothing
- Generation #2: Shallow Learning
  - Handcraft features
  - Learn predictions
- Generation #3: Deep Learning
  - Handcraft algorithm (optimiser, target, architecture, …)
  - Learn features and predictions end-to-end
- Generation #4: Meta Learning
  - Handcraft nothing
  - Learn algorithm and features and predictions end-to-end