Inteligencia Artificial

CC-421

César Lara Avila

Universidad Nacional de Ingeniería

(actualización: 2021-01-29)

Bienvenidos

Aprendizaje por Refuerzo - Parte 3

- Aprendizaje por refuerzo sin modelos y basado en modelos
- Aprendizaje por Refuerzo sin modelos
- Q-learning y redes neuronales
- Métodos híbridos
- RL basados en modelos

Aprendizaje por refuerzo sin modelos y basado en modelos

El **RL basado en modelos** utiliza la experiencia para construir un modelo interno de transiciones y los resultados inmediatos en el entorno. A continuación, se eligen las acciones adecuadas mediante la búsqueda o planificación en este modelo.

El **RL sin modelo**, se usa la experiencia para aprender directamente una o ambas de dos cantidades más simples (valores de estado-acción o políticas) que pueden lograr el mismo comportamiento óptimo pero sin estimación o uso de un modelo. Dada una política, un estado tiene un valor, definido en términos de la utilidad futura que se espera acumular a partir de ese estado.

Los métodos sin modelos son estadísticamente menos eficientes que los métodos basados en modelos, porque la información del entorno se combina con estimaciones anteriores y posiblemente erróneas, sobre los valores de estados, en lugar de usarse directamente.

Aprendizaje por refuerzo sin modelos

Dos enfoques principales para representar agentes con aprendizaje por refuerzo sin modelos son la **optimización de políticas** y el **Q-learning**.

Métodos de optimización o iteración de políticas

En los métodos de optimización de políticas, el agente aprende directamente la función de política que relaciona el estado con la acción. La política se determina sin utilizar una función de valor.

Existen dos tipos de políticas: deterministas y estocásticas. La política determinista mapea el estado con la acción sin incertidumbre. Esto sucede cuando tienes un entorno determinista como un tablero de ajedrez.

La política estocástica genera una distribución de probabilidad sobre las acciones en un estado dado. Este proceso se denomina Proceso de decisión de Markov parcialmente observable (POMDP).

Política de gradiente (PG)

En este método tenemos la política π que tiene un parámetro θ. Esta política genera una distribución de probabilidad de acciones. Entonces debemos encontrar los mejores parámetros (θ) para maximizar (optimizar) una función de puntuación $J(\theta)$, dado el factor de descuento γ y recompensa r.

Pasos principales:

- Mide la calidad de una política con la función de puntuación de políticas.
- Utiliza el ascenso de gradiente de política para encontrar el mejor parámetro que mejore la política.

La implementación **Spinning Up** de PG admite la paralelización con MPI.

Ecuación

Sea π_θ una política con parámetros θ y $J(\pi_\theta)$ denota el retorno sin descuento esperado de horizonte finito de la política. Entonces el gradiente de $J(\pi_\theta)$ es:

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)$$

donde τ es una trayectoria y A^{π_θ} es la función de ventaja para la política actual.

El algoritmo de gradiente de política funciona actualizando los parámetros de política a través del ascenso del gradiente estocástico en el desempeño de la política:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k})$$

Pseudocódigo

Algorithm 1 Vanilla Policy Gradient Algorithm

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

Exploración - explotación en VPG

VPG explora por muestreo acciones de acuerdo con la última versión de la política estocástica.

La cantidad de aleatoriedad en la selección de acciones depende tanto de las condiciones iniciales como del procedimiento de entrenamiento.

A lo largo del entrenamiento, la política generalmente se vuelve cada vez menos aleatoria, ya que la regla de actualización aprovecha las recompensas que ya ha encontrado. Esto puede hacer que la política queda atrapada en los óptimos locales.

Trust Region Policy Optimization (TRPO)

Un algoritmo basado en políticas que se puede utilizar en entornos con espacios de acción discretos o continuos.

TRPO actualiza las políticas dando el paso más grande posible para mejorar el rendimiento, al tiempo que satisface una restricción especial sobre qué tan cerca están las políticas nuevas y antiguas.

La restricción se expresa en términos de KL-Divergencia, una medida de (algo así como, pero no exactamente) la distancia entre distribuciones de probabilidad.

Ecuación

Sea π_θ denota una política con parámetros θ. La actualización teórica de TRPO es:

$$\begin{aligned}\theta_{k+1} = \arg\max_{\theta}\ & \mathcal{L}(\theta_k, \theta) \\ & \text{s.t. } \bar{D}_{KL}(\theta||\theta_k) \leq \delta\end{aligned}$$

donde $\mathcal{L}(\theta_k, \theta)$ es la ventaja sustituta, una medida de cómo funciona la política π_θ en relación con la política anterior π_{θ_k} utilizando datos de la política anterior:

$$\mathcal{L}(\theta_k, \theta) = E_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a)\right]$$

y $\bar{D}_{KL}(\theta||\theta_k)$ es una divergencia KL promedio entre las políticas de los estados visitados por la política anterior:

$$\bar{D}_{KL}(\theta||\theta_k) = E_{s\sim\pi_{\theta_k}}[D_{KL}\left(\pi_\theta(\cdot|s)||\pi_{\theta_k}(\cdot|s)\right)].$$

El objetivo y la restricción son ambos cero cuando $\theta = \theta_k$. Además, el gradiente de la restricción con respecto a θ es cero cuando $\theta = \theta_k$.

TRPO hace algunas aproximaciones para obtener una respuesta rápidamente. Por ejemplo expandimos el objetivo y la restricción al orden principal alrededor de θ_k:

$$\mathcal{L}(\theta_k, \theta) \approx g^T(\theta - \theta_k)$$
$$\bar{D}_{KL}(\theta||\theta_k) \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)$$

resultando en un problema de optimización aproximado,

$$\theta_{k+1} = \arg\max_{\theta} \; g^T(\theta - \theta_k)$$
$$\text{s.t. } \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta.$$

Este problema aproximado puede resolverse analíticamente mediante los métodos de la dualidad lagrangiana , dando como resultado la solución:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

Un problema que sucede aquí, es que debido a los errores de aproximación introducidos por la expansión de Taylor, esto puede no satisfacer la restricción KL o, de mejorar la ventaja sustituta. TRPO agrega una modificación a esta regla de actualización: **una búsqueda de línea de retroceso**:

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g$$

donde $\alpha \in (0, 1)$ es el coeficiente de retroceso y j es el número entero no negativo más pequeño tal que $\pi_{\theta_{k+1}}$ satisface la restricción KL y produce una ventaja sustituta positiva.

Pregunta Calcular y almacenar la matriz inversa H^{-1}, es tremendamente caro cuando se trata de políticas de redes neuronales con miles o millones de parámetros. como crees que TRPO evita este problema?.

Pseudocódigo

Algorithm 1 Trust Region Policy Optimization

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K

for $k = 0, 1, 2, ...$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence.

Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, ...K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

Exploración - explotación en TRPO

TRPO explora por muestreo acciones de acuerdo con la última versión de su política estocástica.

La cantidad de aleatoriedad en la selección de acciones depende tanto de las condiciones iniciales como del procedimiento de entrenamiento.

A lo largo del entrenamiento, la política generalmente se vuelve cada vez menos aleatoria, ya que la regla de actualización alienta a aprovechar las recompensas que ya ha encontrado.

Esto puede hacer que la política quede atrapada en los óptimos locales.

Proximal Policy Optimization (PPO)

PPO comparte motivación con TRPO en la tarea de responder a la pregunta: ¿cómo aumentar la mejora de las políticas sin el riesgo de colapso del rendimiento?

PPO mejora la estabilidad de la formación de Actor al limitar la actualización de la política en cada paso del entrenamiento, penalizando la divergencia de KL en la función objetivo en lugar de convertirla en una restricción estricta, y ajusta automáticamente el coeficiente de penalización durante el curso del entrenamiento para que escale adecuadamente.

Hay dos variantes principales de PPO: **PPO-Penalty** y **PPO-Clip**.

La implementación **Spinning Up** de PPO admite la paralelización con MPI.

Ecuación

PPO-clip actualiza las políticas vía:

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} \left[L(s, a, \theta_k, \theta) \right],$$

típicamente tomando varios pasos de SGD (generalmente minibatch) para maximizar el objetivo. Aquí L viene dado por:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad \text{clip} \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right)$$

en el que ϵ es un hiperparámetro (pequeño) que indica aproximadamente qué tan lejos puede ir la nueva política de la anterior.

Hay una versión considerablemente simplificada que es un poco más fácil de manejar útil en las implementaciones:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad g(\epsilon, A^{\pi_{\theta_k}}(s, a)) \right)$$

dónde:

$$g(\epsilon, A) = (1+\epsilon)A \quad \text{si} \quad A \geq 0$$
$$(1-\epsilon)A \quad \text{si} \quad A < 0.$$

La ventaja es positiva

Supongamos que la ventaja para ese par estado-acción es positiva, en cuyo caso la contribución al objetivo se reduce a:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\theta_k}}(s, a).$$

Dado que la ventaja es positiva, el objetivo aumentará si la acción se vuelve más probable, es decir, si $\pi_\theta(a|s)$ aumenta. Pero el mínimo en este término pone un límite a cuánto puede aumentar el objetivo.

Una vez que $\pi_\theta(a|s) > (1+\epsilon)\pi_{\theta_k}(a|s)$, el mínimo entra en acción y este término alcanza un límite de $(1+\epsilon)A^{\pi_{\theta_k}}(s, a)$.

Por lo tanto: la nueva política no se beneficia al alejarse mucho de la política anterior.

La ventaja es negativa

Supongamos que la ventaja para ese par estado-acción es negativa, en cuyo caso la contribución al objetivo se reduce a

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a).$$

Debido a que la ventaja es negativa, el objetivo aumentará si la acción se vuelve menos probable, es decir, si $\pi_\theta(a|s)$ disminuye.

Pero el máximo en este término pone un límite sobre cuánto puede aumentar el objetivo. Una vez que $\pi_\theta(a|s) < (1-\epsilon)\pi_{\theta_k}(a|s)$, el máximo se activa y este término alcanza un límite de $(1-\epsilon)A^{\pi_{\theta_k}}(s, a)$.

Por lo tanto, nuevamente: la nueva política no se beneficia si se aleja mucho de la anterior.

Lo que hemos visto hasta ahora es que el clipping sirve como un regularizador al eliminar los incentivos para que la política cambie drásticamente y el hiperparámetro ϵ corresponde a qué tan lejos puede ir la nueva política de la anterior sin dejar de beneficiarse del objetivo.

Pseudocódigo

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

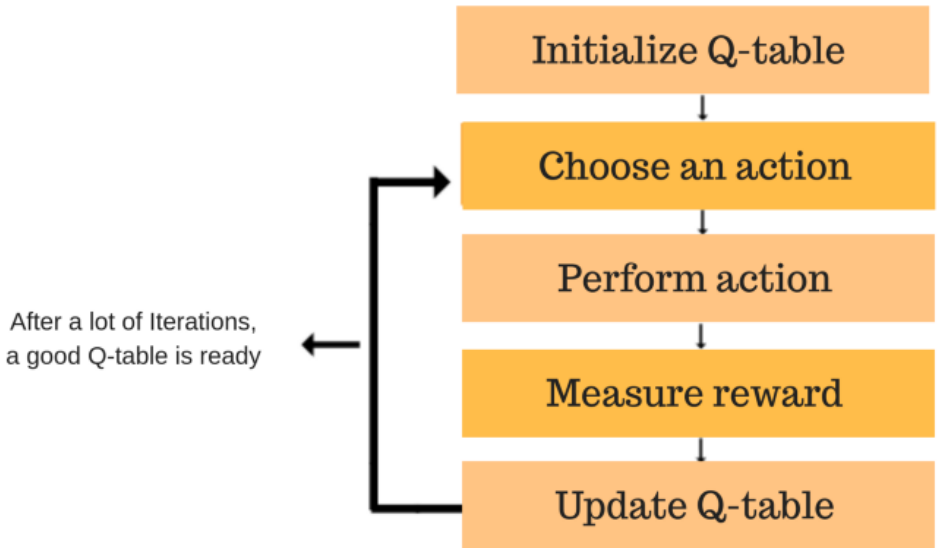
  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

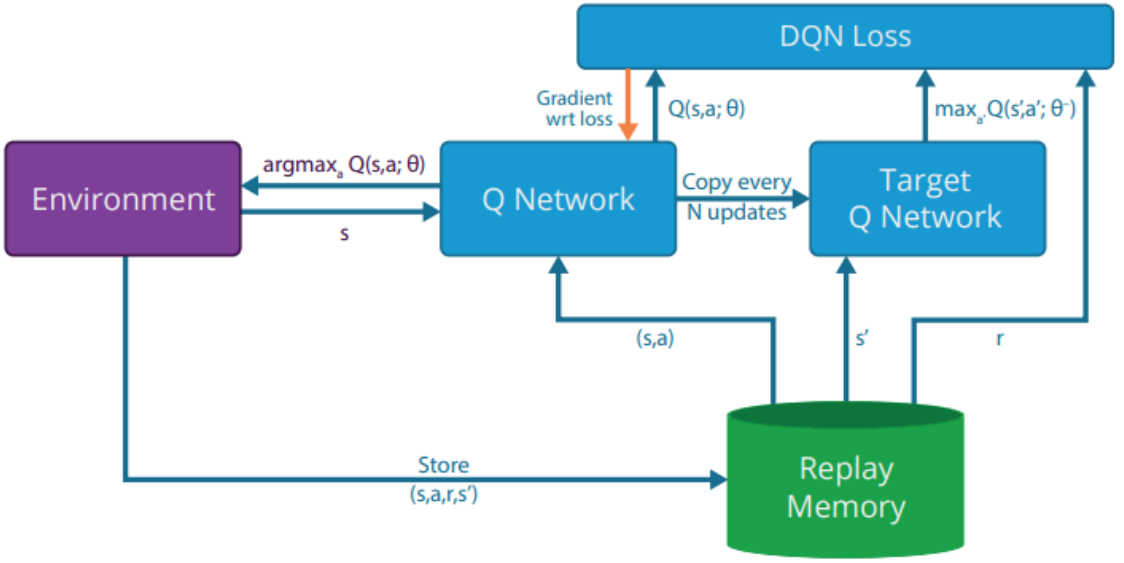
Métodos Q-learning o de iteración de valores

Q-learning aprende la función de valor-acción $Q(s, a)$ con respecto a analizar qué tan bueno es tomar una acción en un estado particular. Básicamente, se asigna un valor escalar sobre una acción en un estado dado.



Deep Q Neural Network (DQN)

DQN es Q-learning con redes neuronales. La motivación está simplemente relacionada con los grandes entornos espaciales de estados donde definir una Q-tabla que es una tarea muy compleja, desafiante y que requiere mucho tiempo. En lugar de una Q-tabla, las redes neuronales aproximan los Q-valores para cada acción según el estado.



Distributional Reinforcement Learning with Quantile Regression (QR-DQN)

En QR-DQN para cada par estado-acción en lugar de estimar un valor único, se aprende una distribución de valores. La distribución de los valores, en lugar de solo el promedio, puede mejorar la política.

C51

C51 es un algoritmo factible para realizar una aproximación iterativa de la distribución de valor Z utilizando la ecuación distributiva de Bellman. El número 51 representa el uso de 51 valores discretos para parametrizar la distribución de valores $Z(s, a)$.

Hindsight Experience Replay (HER)

En el método HER, básicamente un DQN se suministra con un estado y un estado final.

Permite aprender rápidamente cuando las recompensas son sparse. En otras palabras, cuando las recompensas son uniformes la mayor parte del tiempo, con solo unos pocos valores de recompensa raros que realmente se destacan.

Métodos híbridos

Estos métodos combinan las fortalezas del Q-learning y las políticas de gradiente, por lo que aprenden la función de política que mapea el estado con la acción y la función acción-valor que proporciona un valor para cada acción.

Algunos algoritmos híbridos sin modelo son:

- Deep Deterministic Policy Gradients (DDPG)
- Soft Actor-Critic (SAC)
- Twin Delayed Deep Deterministic Policy Gradients (TD3)

RL basado en modelos

La RL basada en modelos tiene una fuerte influencia de la teoría de control y el objetivo es planificar mediante una función de control $f(s, a)$ las acciones óptimas.

Hay dos enfoques principales: **aprender el modelo** o **aprender dado el modelo**.

Aprender el modelo

Para conocer el modelo se ejecuta una **política base**, como una política aleatoria, mientras se observa la trayectoria. El modelo se ajusta utilizando los datos muestreados.

- **World models**, el agente puede aprender de sus propios "sueños" gracias a los autoencoders variables.
- **Imagination-Augmented Agents** (I2A): aprende a interpretar predicciones de un modelo de entorno aprendido para construir planes implícitos de manera arbitraria, utilizando las predicciones como contexto adicional en redes de políticas profundas.
- **Model-Based Priors for Model-Free Reinforcement Learning** (MBMF): tiene como objetivo cerrar la brecha entre el aprendizaje por refuerzo sin modelos y el basado en modelos.
- **Model-Based Value Expansion** (MBVE): este método controla la incertidumbre en el modelo al permitir que la imaginación alcance una profundidad fija.

Dado el modelo

Alphazero

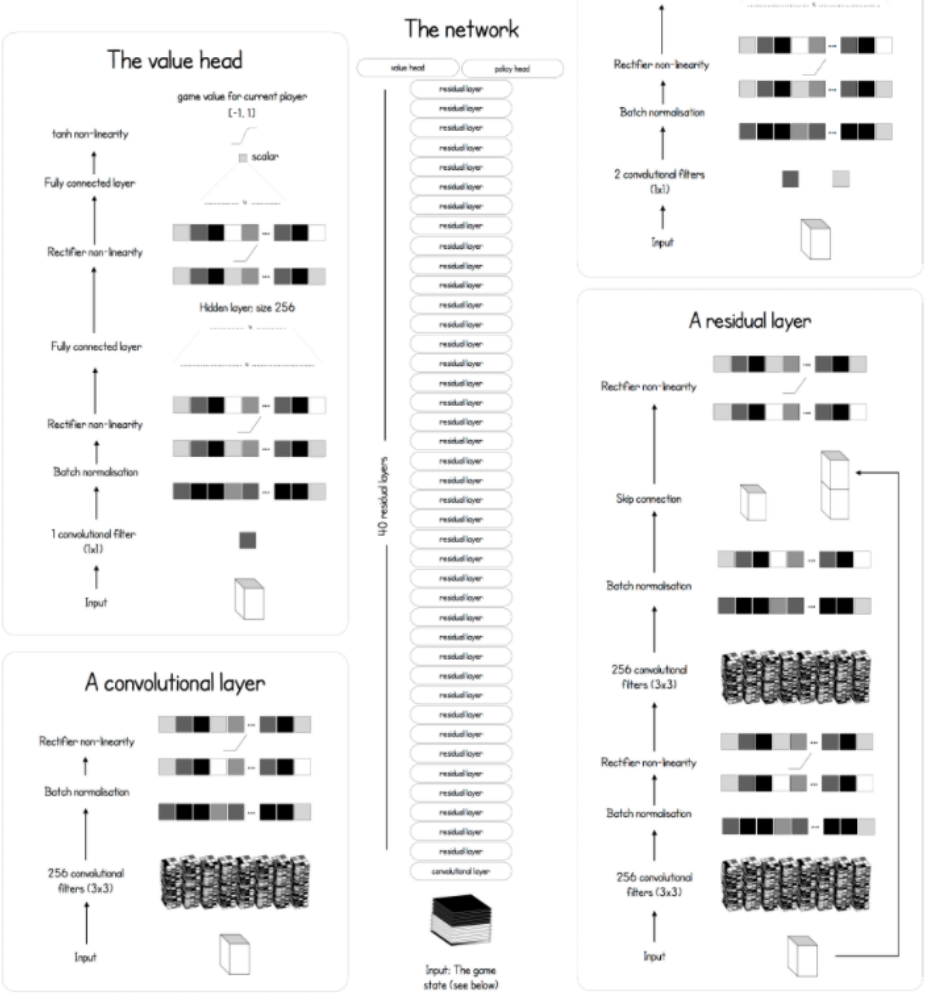
- Usa muchos trucos de la caja de herramientas de visión por computadora, incluido el aumento agresivo de datos.
- AlphaZero intenta predecir una distribución de probabilidad de los mejores movimientos de un estado dado (para combatir el sobreajuste), utilizando una red con un *jefe de política* y un *jefe de valor*. Estas dos medidas se validan mutuamente para garantizar que la red rara vez confía en predicciones incorrectas.
- Es solo una combinación de MCTS(Monte-Carlo Tree Search) y UCT(Upper-Confidence Bounds Applied to Trees) con investigación reciente de redes neuronales profundas.

THE DEEP NEURAL NETWORK ARCHITECTURE

How AlphaGo Zero assesses new positions

The network learns 'tabula rasa' (from a blank slate)

At no point is the network trained using human knowledge or expert moves



Fin!