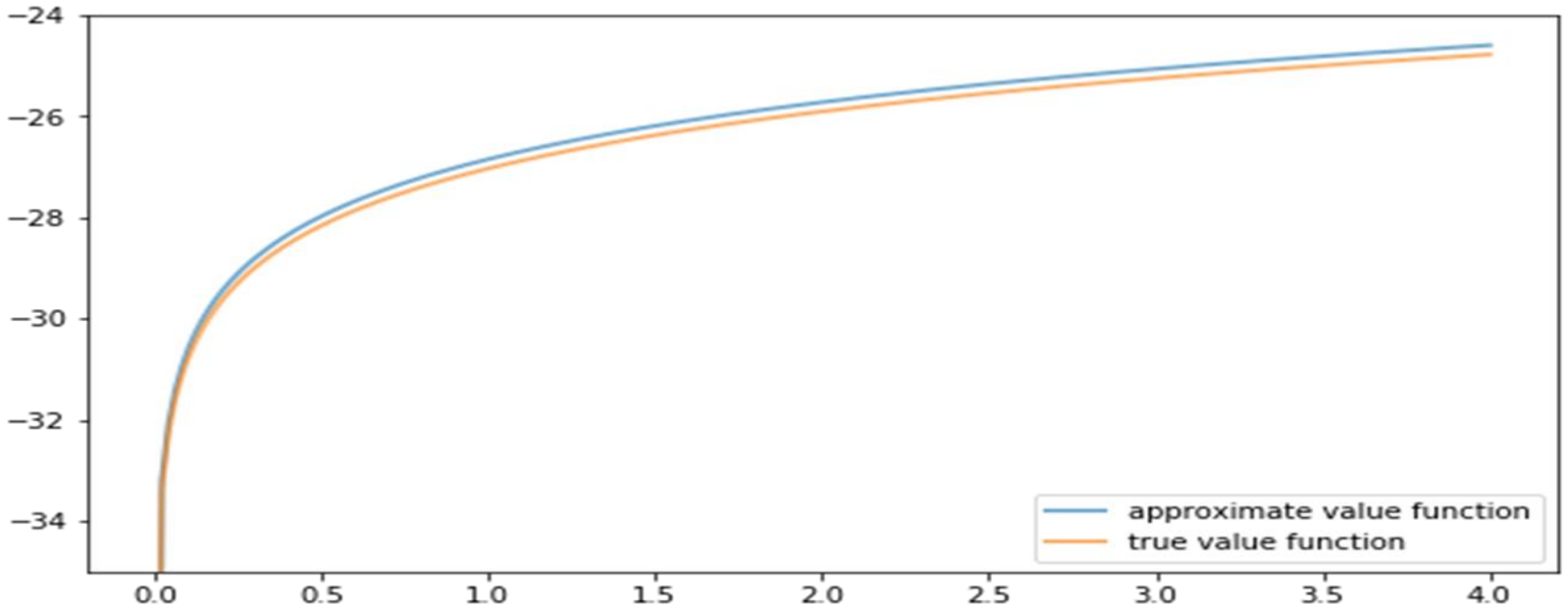
Cuarta Parte:
Clase 4 – *Value Function Approximation*

Optimización Dinámica - ICS



Mathias Klapp

- Introducción a VFA
- Tipos de VFA
- Lookup tables

Value Function Approximation (VFA)

- Es un marco de trabajo para resolver problemas dinámicos de gran tamaño en espacio de decisión, estado y transición.

- En la etapa t y en el estado s_t, toma decisión mediante:

$$d_t^{VFA}(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x) + \bar{Q}_t(y_t(s_t, x))\}$$

, donde $\bar{Q}_t(y_t) \approx Q_t(y_t)$ es una **función que aproxima el value-to-go** calibrada antes de ejecutar la operación (entrenamiento *offline*).

Aprendizaje fuera de línea (Offline learning)

Busca aprender el value to-go $Q_t(y_t)$ en cada estado y_t mediante simulaciones *offline* antes de la corrida real del sistema.

Tipos de simulación *offline*:

1. Simulación computacional
2. Simulación con información histórica
3. Marcha blanca de entrenamiento

Ventaja:

1. No hay cómputo online para estimar Q.
2. Sólo requiere un simulador o una marcha blanca.

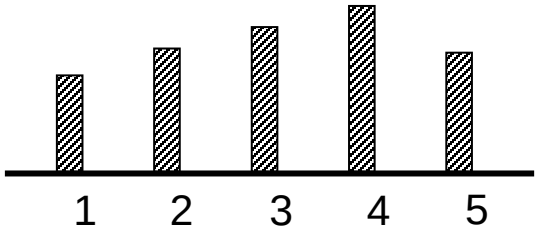
Desventajas:

1. Depende fuerte del problema, de los datos y de la instancia.
2. A veces requiere supuestos en la forma funcional de Q.

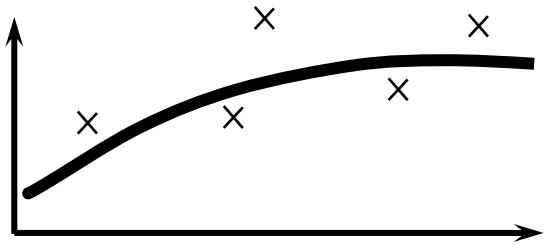
- Introducción a VFA
- Tipos de VFA
- Lookup tables
- Approximate Value Iteration (AVI)
- Exploración versus Explotación

Resumen de aproximaciones para $Q_t(y_t)$

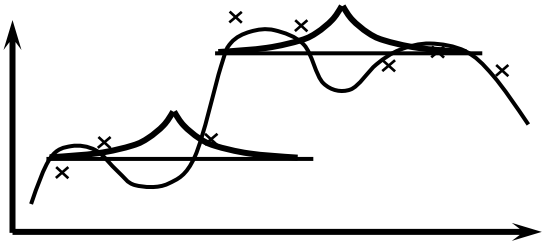
- Tabla de valores (*Lookup table*)
  - Diccionario con un valor estimado para cada estado y_t
  - Desafío adicional: control de memoria, *over-fitting*.



- Aproximación paramétrica:
  - Modelos lineales, no lineales y ALP
  - Explota estructura: monotonía, convexidad, no-negatividad.
  - Exige suponer *a priori* una forma funcional de Q.



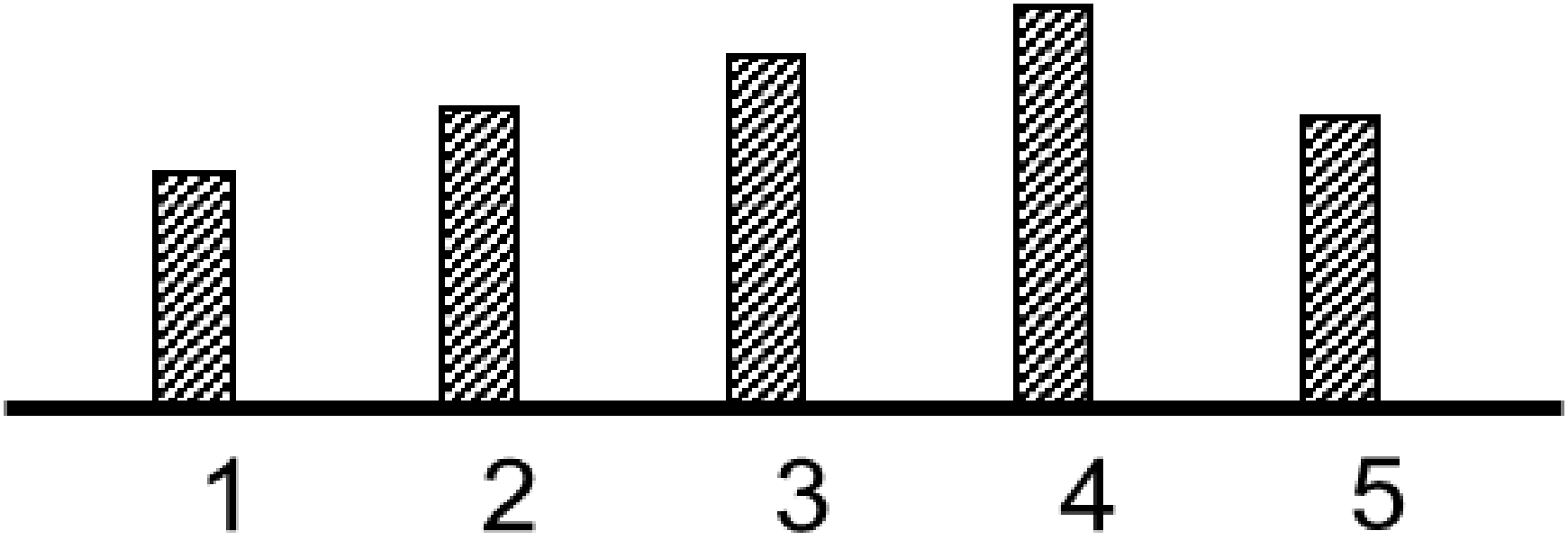
- Aproximación no-paramétrica:
  - k *−nearest neighbor clustering y Kernel regression*
  - *Local polynomial regression*
  - *Deep Q-learing (Q-learning + Deep Learning)*
  - Desafío: *over-fitting*.



- Introducción a VFA
- Tipos de VFA
- Lookup tables

VFA: Lookup tables

Optimización Dinámica - ICS



Mathias Klapp

Referencia: capítulo 10, Approximate Dynamic Programming, Warren Powell

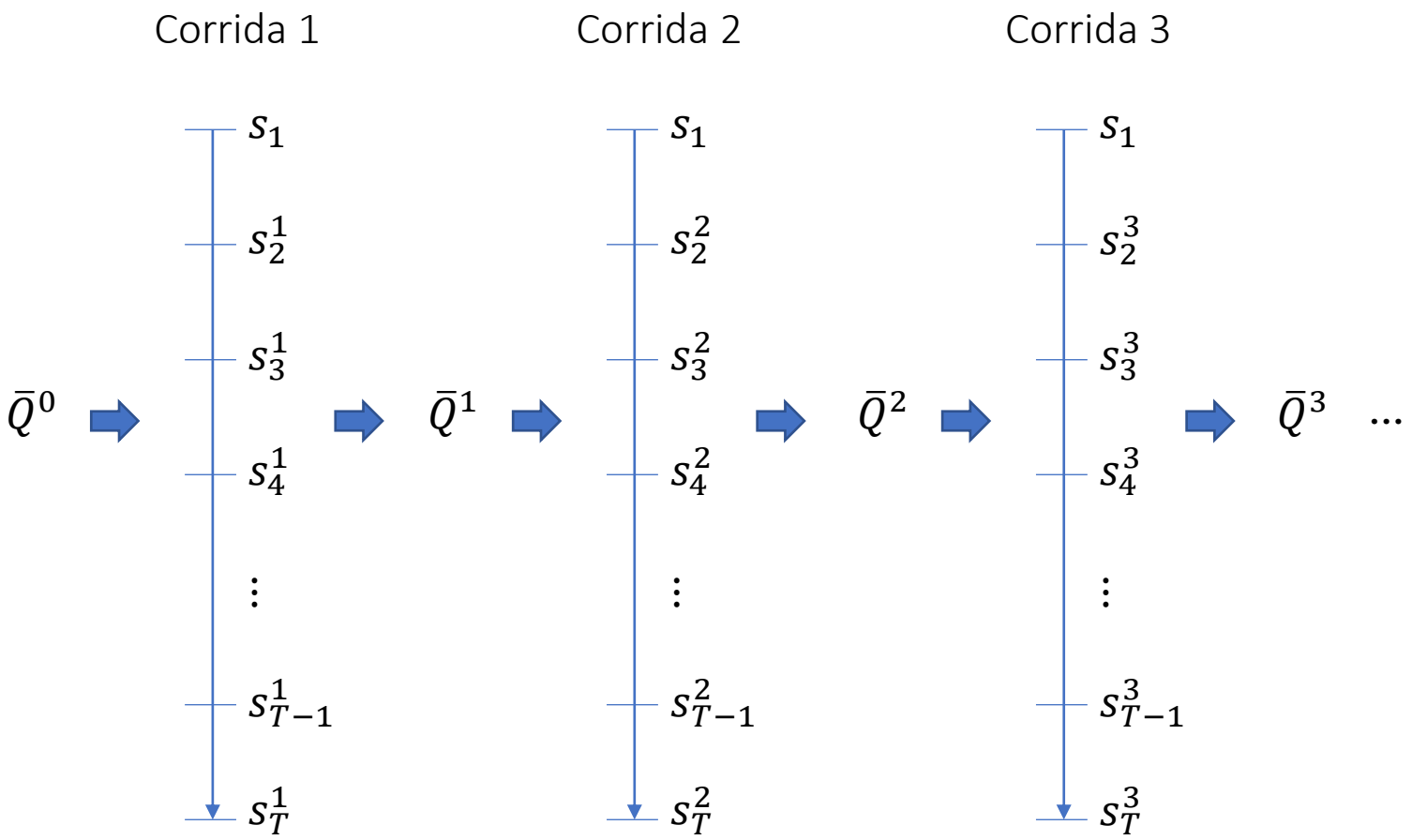
Lookup table

- Registra en un diccionario una aproximación del cost-to-go $\bar{Q}_t(y_t)$ para cada etapa t y estado y_t.

- Desafíos:
  - Evitar sobrecarga de memoria.
  - *Over-fitting.*


A GOOD EXAMPLE OF
OVERFITTING

Aproximate Value Iteration (AVI)



Aproximate Value Iteration (AVI)

Simula N veces la dinámica del proceso para aproximar Q antes de la corrida real.

En el estado s_t^n (de la etapa t en la simulación n):

1. Resuelve:

$$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x))\},$$
$$y_t^n \leftarrow y_t(s_t^n, x_t^n)),$$
$$q_t^n \leftarrow r_t(s_t^n, x_t^n) + \bar{Q}_t^{n-1}(y_t^n)$$

2. Actualiza Q en estado de post-decisión anterior a s_t^n:

$$\bar{Q}_{t-1}^n(y_{t-1}^n) \leftarrow (1 - \alpha_n) \cdot \bar{Q}_{t-1}^{n-1}(y_{t-1}^n) + \alpha_n \cdot q_t^n$$

3. Simula transición al siguiente estado:

$$s_{t+1}^n \leftarrow f_t(y_t^n, \omega_t^n)$$

AVI como simulación-optimización

AVI es una heurística de entrenamiento *offline* equipada con simulación, algoritmos de optimización en el proceso simulado y algoritmos de ajuste estadístico.

Tres pasos:

1. El paso de optimización
2. El paso de entrenamiento (ajuste de la estimación)
3. El paso de simulación (al próximo estado)

Algoritmo *AVI*

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^0(y_t)$ para todo t y y_t
2. Mientras $n \leq N$:
   a. Iniciar corrida: $s_1^n \leftarrow s_1$
   b. Para cada $t = 1, \ldots, T$:

      Optimizar:

      $$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x))\},$$
      $$y_t^n \leftarrow y_t(s_t^n, x_t^n),$$
      $$q_t^n \leftarrow r_t(s_t^n, x_t^n) + \bar{Q}_t^{n-1}(y_t^n).$$

      Optimización determinística

      Actualizar $\boldsymbol{Q}$:

      $$\bar{Q}_{t-1}^n(y_{t-1}^n) \leftarrow (1 - \alpha_n)\bar{Q}_{t-1}^{n-1}(y_{t-1}^n) + \alpha_n q_t^n$$

      Estadística

      Simular transición a próximo estado:

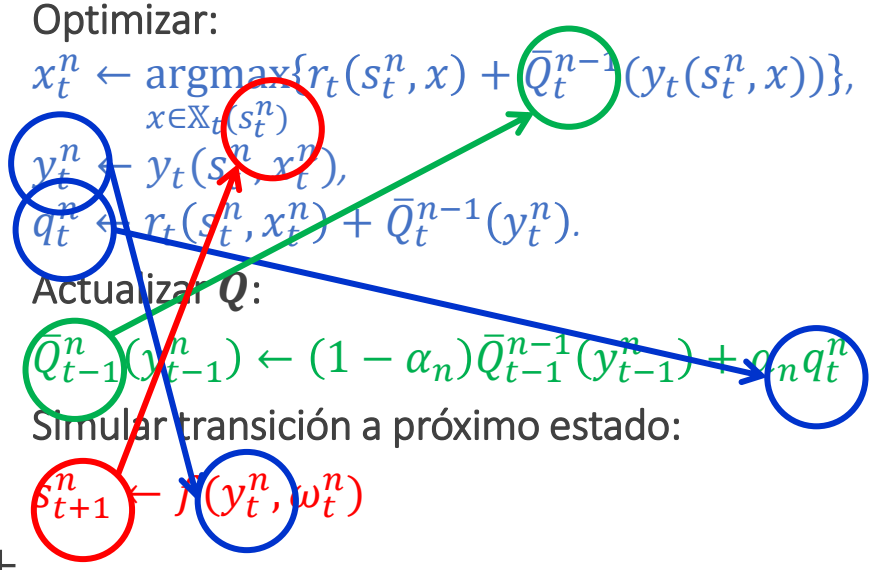
      $$s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n)$$

      Simulación

   c. $n++$
3. Retornar $\bar{Q}_t^N$

Algoritmo *AVI*

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^0(y_t)$ para todo t y y_t
2. Mientras $n \leq N$:
   a. Iniciar corrida: $s_1^n \leftarrow s_1$
   b. Para cada $t = 1, \ldots, T$:



Optimización determinística

Estadística

Simulación

   c. $n++$
3. Retornar $\bar{Q}_t^N$

Diferentes nombres, mismo principio…

También es conocido como ***Forward Dynamic Programming (***FDP***)***, pues a diferencia de un método exacto (*Backward* DP), estima la función de valor simulando ``hacia adelante'' en el tiempo.

En ciencia de la computación un método simular es conocido como ***Q-learning***.

También se conoce como ***Real-time Dynamic Programming*** si *a*prendizaje se da tiempo de ejecución. En este caso simulador es reemplazado por corridas reales.

Desafíos de AVI

1. ¿Cómo **calibar el paso α_n** del suavizamiento exponencial?

2. Como fue presentado, **no garantiza convergencia al Q óptimo**

3. Aceleración de ajuste (puede ser lento calibrando)

3. Mantiene la maldición de decisiones y la de estados

4. ¿Problemas con estados continuos o una grilla muy fina?

Calibrar el paso α_n

El ajuste recursivo

$$\bar{Q}_{t-1}^{n}(y_{t-1}^{n}) = (1-\alpha_n)\cdot \bar{Q}_{t-1}^{n-1}(y_{t-1}^{n}) + \alpha_n \cdot q_t^n$$

, requiere un valor de α_n.

Si $\alpha_n = 1$: Estimación es el último dato observado en el estado.

- El *value-to-go* que mejor optimizado está (menos sesgo), pero posee alta variabilidad.

Si $\alpha_n = \frac{1}{n}$: Se promedia las observaciones del estado.

- Promedio posee menor variabilidad, pero datos antiguos fueron generados con pobres estimaciones futuras (sesgo).

Referencia: capítulo 11, ADP, Warren Powell

¿Pq no usar α_n cambiante entre $1/n$ y 1?

En simulaciones finales, se debiese estabilizar la distribución del estimador.

- Recomendación: Usar $1/(n - n_0)$, donde n_0 es un número de réplicas iniciales (transientes) antes del supuesto estado de régimen.

En simulaciones iniciales α_n debe ser alto para aprender (a costa de alta variabilidad).

- Fijar un valor y reducirlo cada m replicas hasta n_0 corridas iniciales. Luego hacer corridas en régimen con $\alpha_n = \frac{1}{n-n_0}$.

Más opciones:

- $\alpha_n = \frac{a}{(a+n-1)}$, donde $a > 1$ se debe calibrar.
- $\alpha_n = \frac{1}{n^\beta}$, con β a calibrar. Recomendación $\beta = 0{,}7$.
- Paso α_n que minimizar varianza del estimador: ver ADP, cap. 11-4.

Convergencia

- Value Iteration converge de forma exacta al value-to-go óptimo (Propiedades de contracción y punto fijo).

- ¿Y AVI? Lamentablemente, No

Al simular corridas se hace vulnerable y puede caer en óptimos locales.

Problemas de convergencia

- **Problema:** Tiende a evitar decisiones que llevan a estados de post-decisión y_t con value-to-go muy subestimados en calibraciones de simulaciones anteriores.

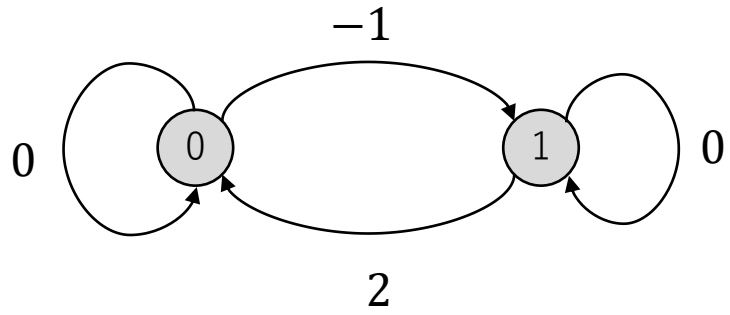
- Si un estado y_t posee su valor $Q_t(y_t)$ muy subestimado, entonces es probable que no sea atractivo y, por ende, que no se visite y su $Q_t(y_t)$ no sea reestimado.

- Por ende: Si se visita únicamente estados que parecen ser atractivos, entonces existe un riesgo de caer en óptimo local.

Ejemplo en DP determinístico de 2 periodos:

- $s_1 = 0$
- Estimación inicial: $\bar{V}_2\ (0) = \bar{V}_2\ (1) = 0$

- En $t = 1$ tomará la decisión de quedarse en 0, a pesar de ser óptimo ir a 1.



- Conclusión: Estado 1 nunca será visitado y value to go $V_2(1) = 2 \neq 0$ terminará subestimado.

Arreglando convergencia

$$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{ r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x)) \}$$

La decisión x_t^n del paso de optimización posee **doble función**:

1. Determina un nuevo dato q_t^n que actualiza el *cost-to-go*.
2. Influye en el estado s_{t+1}^n de la siguiente etapa $t + 1$.

Arreglo: Separar funciones

- Usar x_t^n para calcular q_t^n.
- , pero randomizar ocasionalmente la transición a estados futuros para acceder a estados que (por el momento) no son atractivos.

Esto se conoce como búsqueda ε ***−greedy***.

Algoritmo *AVI (version ε − greedy)*

1. Iniciar: $n \leftarrow 1$, $\bar{Q}_t^0(y_t)$ para todo t y y_t
2. Mientras $n \leq N$:
   a. Iniciar corrida: $s_1^n \leftarrow s_1$
   b. Para cada $t = 1, \ldots, T$:

Optimizar:

$$x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \bar{Q}_t^{n-1}(y_t(s_t^n, x))\},$$

$$y_t^n \leftarrow y_t(s_t^n, x_t^n),$$

$$q_t^n \leftarrow r_t(s_t^n, x_t^n) + \bar{Q}_t^{n-1}(y_t^n).$$

Actualizar $\boldsymbol{Q}$:

$$\bar{Q}_{t-1}^n(y_{t-1}^n) \leftarrow (1 - \alpha_n)\bar{Q}_{t-1}^{n-1}(y_{t-1}^n) + \alpha_n q_t^n$$

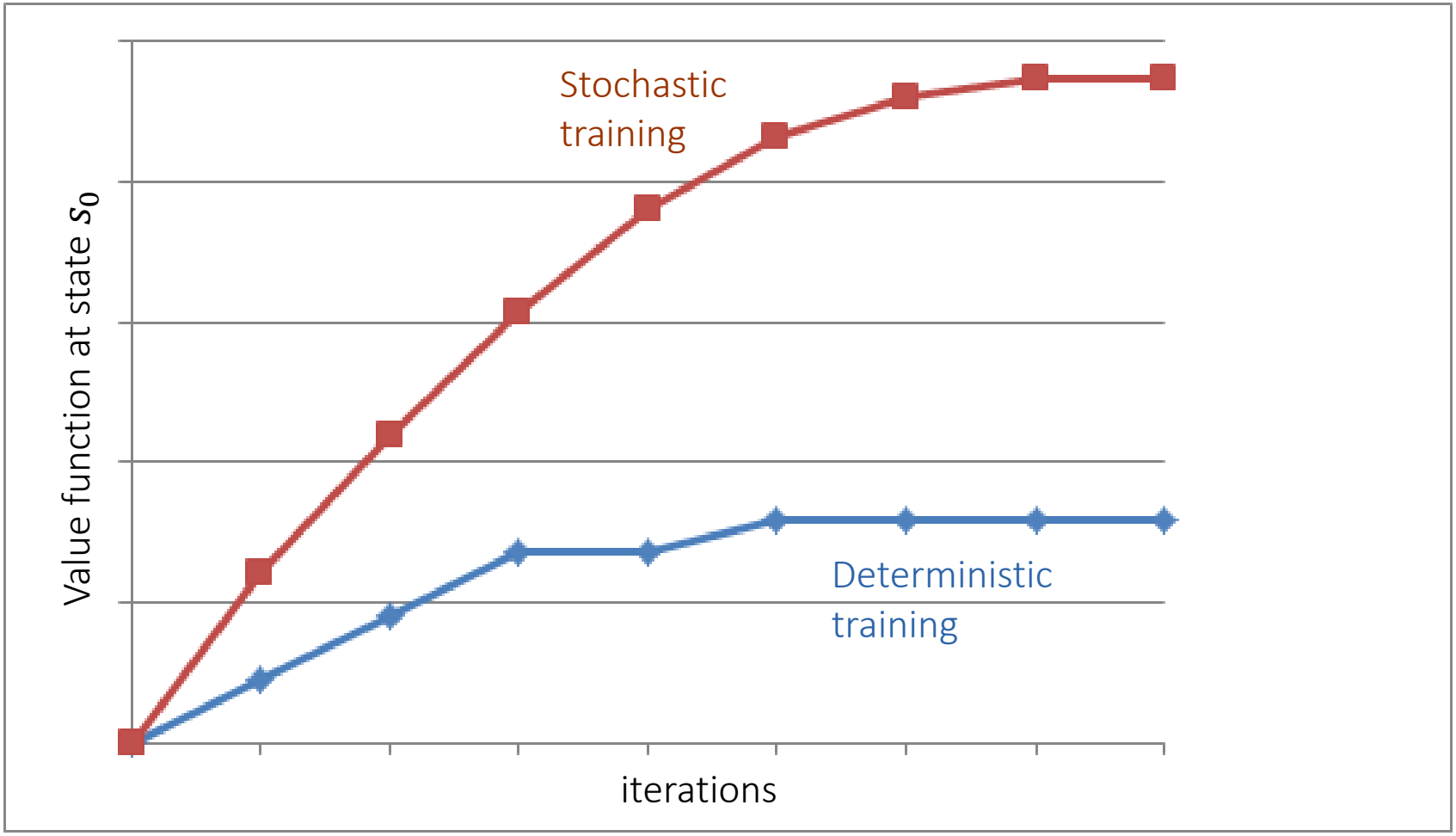
Simular transición a próximo estado:

$$s_{t+1}^n \leftarrow \begin{cases} f(y_t^n, \omega_t^n) & \text{con probabilidad } 1 - \varepsilon \\ f(y_t(s_t^n, x^{RANDOM}), \omega_t^n) & \text{e. o. c.} \end{cases}$$

   c. $n++$

3. Retornar $\bar{Q}_t^N$

Efecto de ε – *greedy*



Exploración versus Explotación

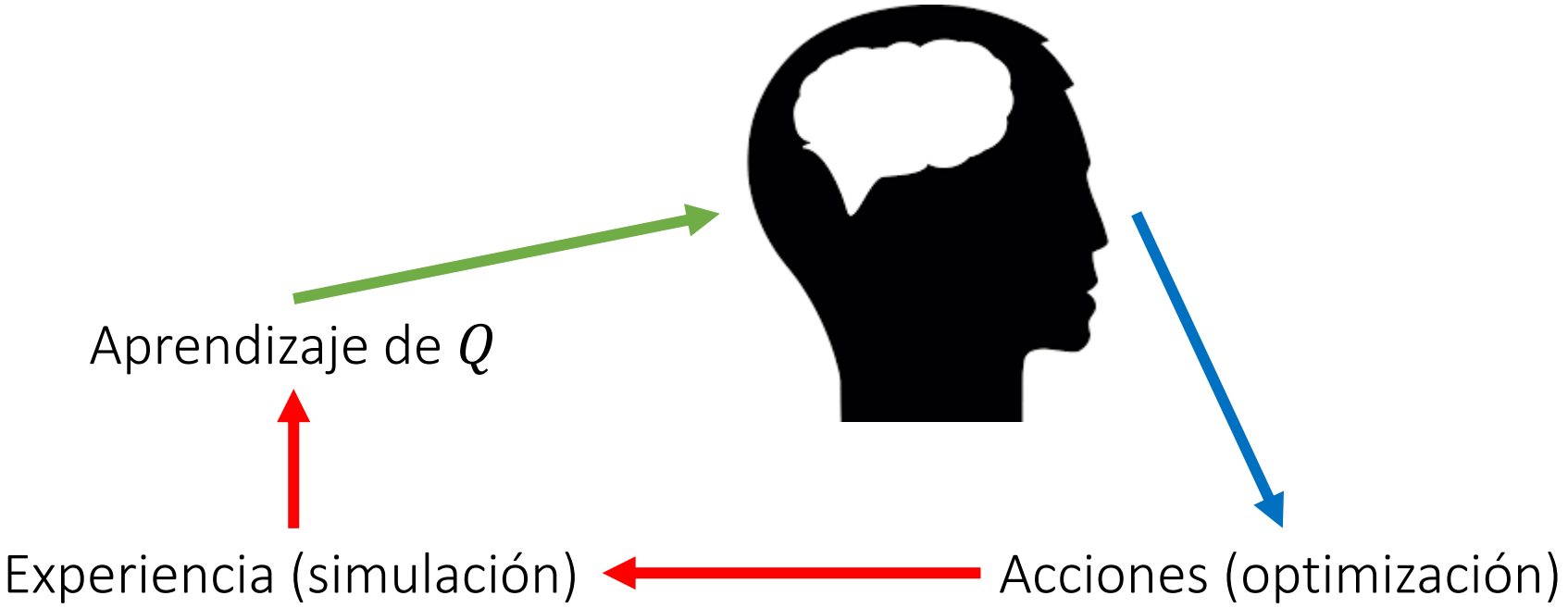
Cuando se utiliza la política x_t^n optimizada para generar la transición de estado, entonces el aprendizaje está en trayectoria. **Explota** el conocimiento actual (experiencia).

Una política aleatoria está fuera de trayectoria. El sistema está **explorando** nuevos estados y sus valores (algoritmo se ``lanza a la piscina´´).

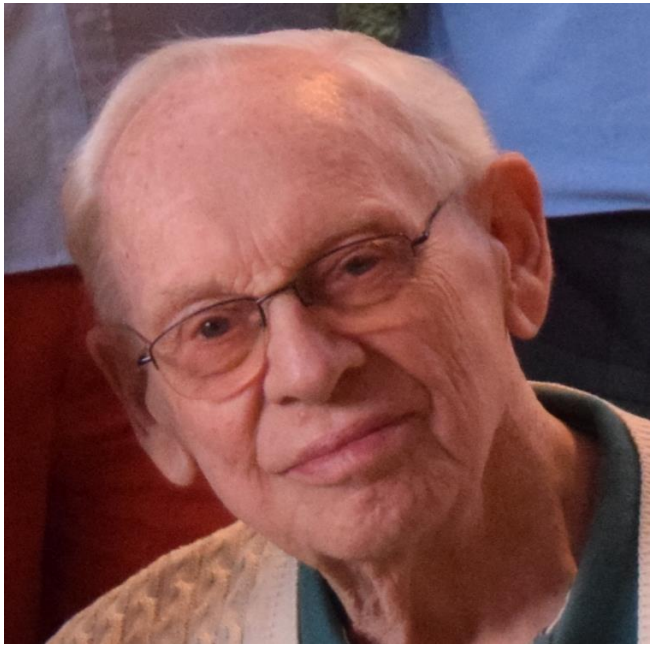
¿Cuándo explorar y cuándo explotar?

Es el dilema de todo aprendizaje, incluyendo el ser humano.

Exploración versus Explotación



Exploración versus Explotación



Abuelo Klapp (96 años, QEPD) tenía experiencia (Q bien entrenada).

Explotaba su conocimiento y era averso a la exploración.

M. Klapp Jr. (**4 años**) tiene Q poco entrenada, **explora** con decisiones ``aleatorias''.

A veces comete errores, pero aprende (recalibra Q en estados que no visitaría sin explorar).

Desafíos de AVI

1. ¿Cómo **calibar el paso α_n** del suavizamiento exponencial?

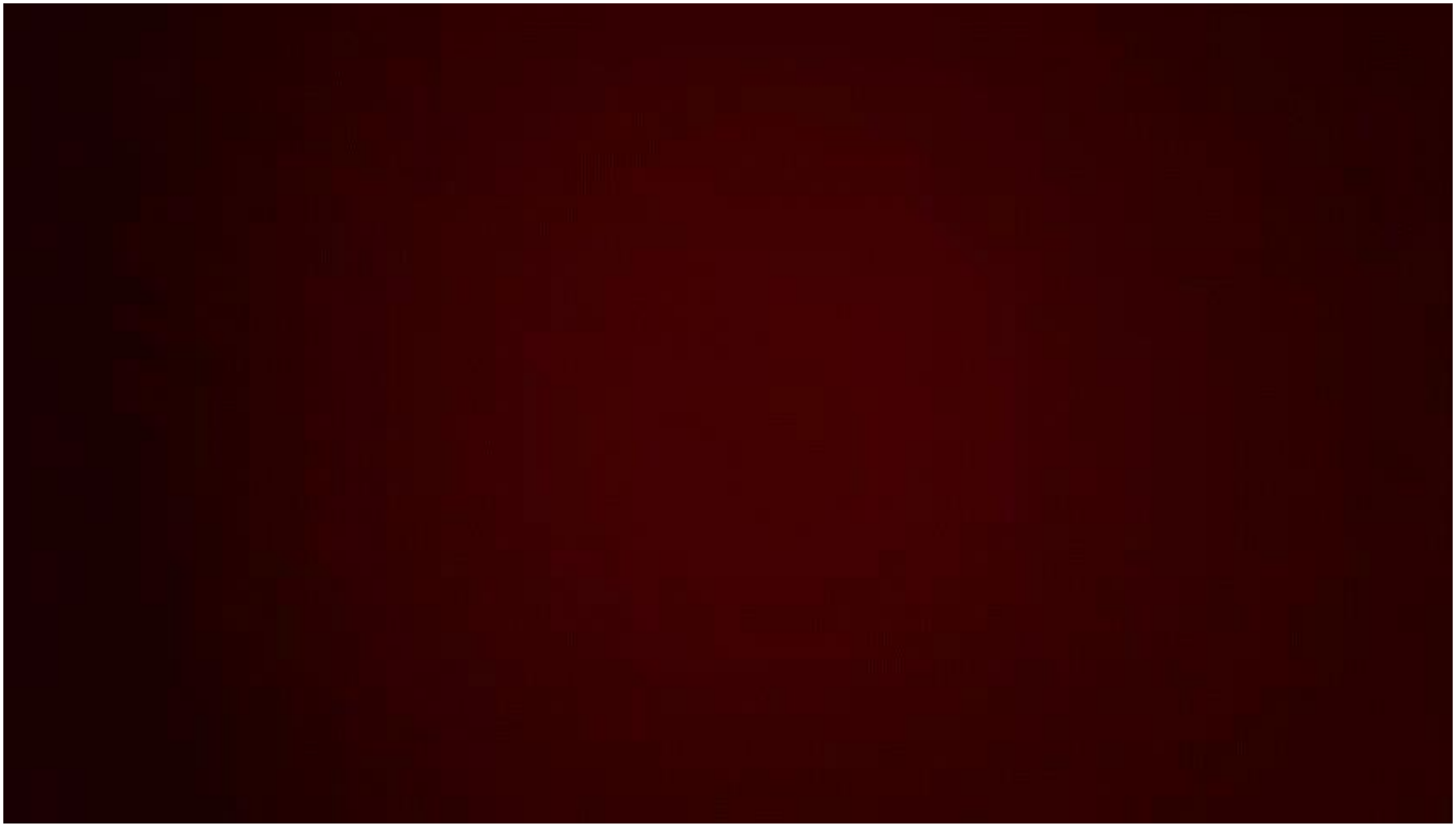
2. Como fue presentado, **no garantiza convergencia al Q óptimo**

3. Aceleración de ajuste (puede ser lento calibrando)
   - Backward-pass

3. Mantiene la maldición de decisiones y la de estados
   - Agregación del espacio de estados.
   - Heuristicas de decisión

4. ¿Problemas con estados continuos o una grilla muy fina?
   - Dynamic *look-up* table.

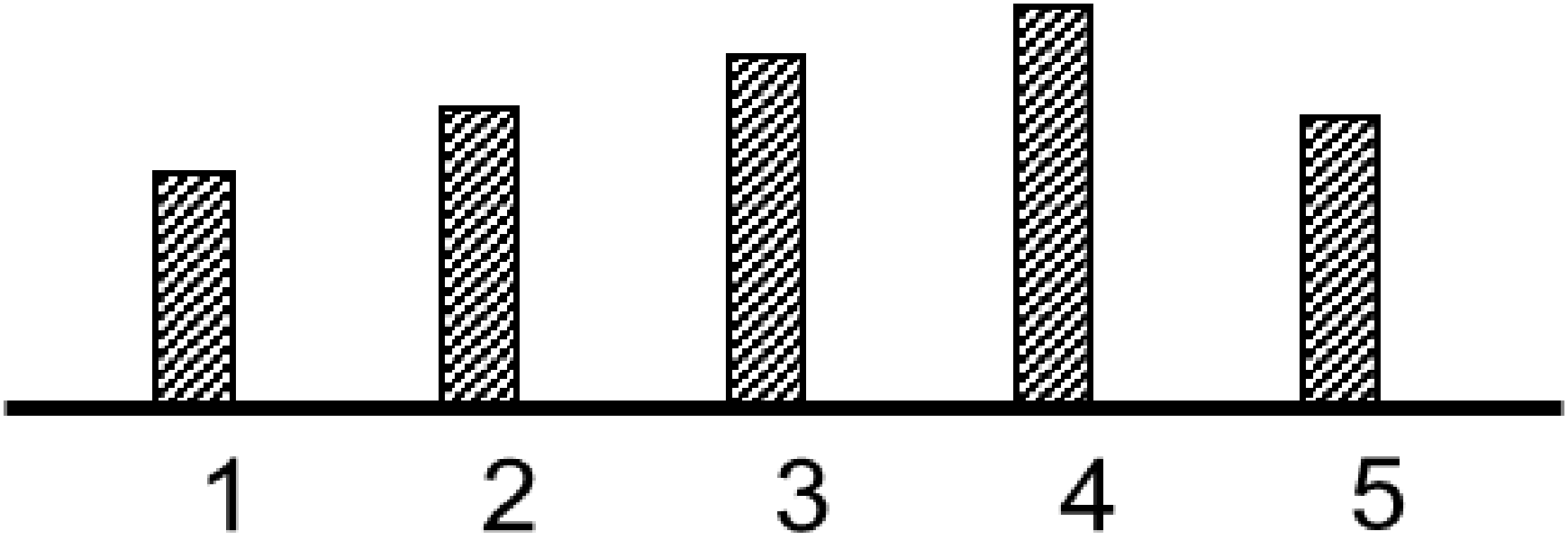
Ejemplo: Robot que aprende a gatear



https://planetachatbot.com/q-learning-con-arduino-crawling-robot-espanol-5eb0acf5aaaf

Cuarta Parte:
Clase 4 – VFA: Lookup table

Optimización Dinámica - ICS



Mathias Klapp