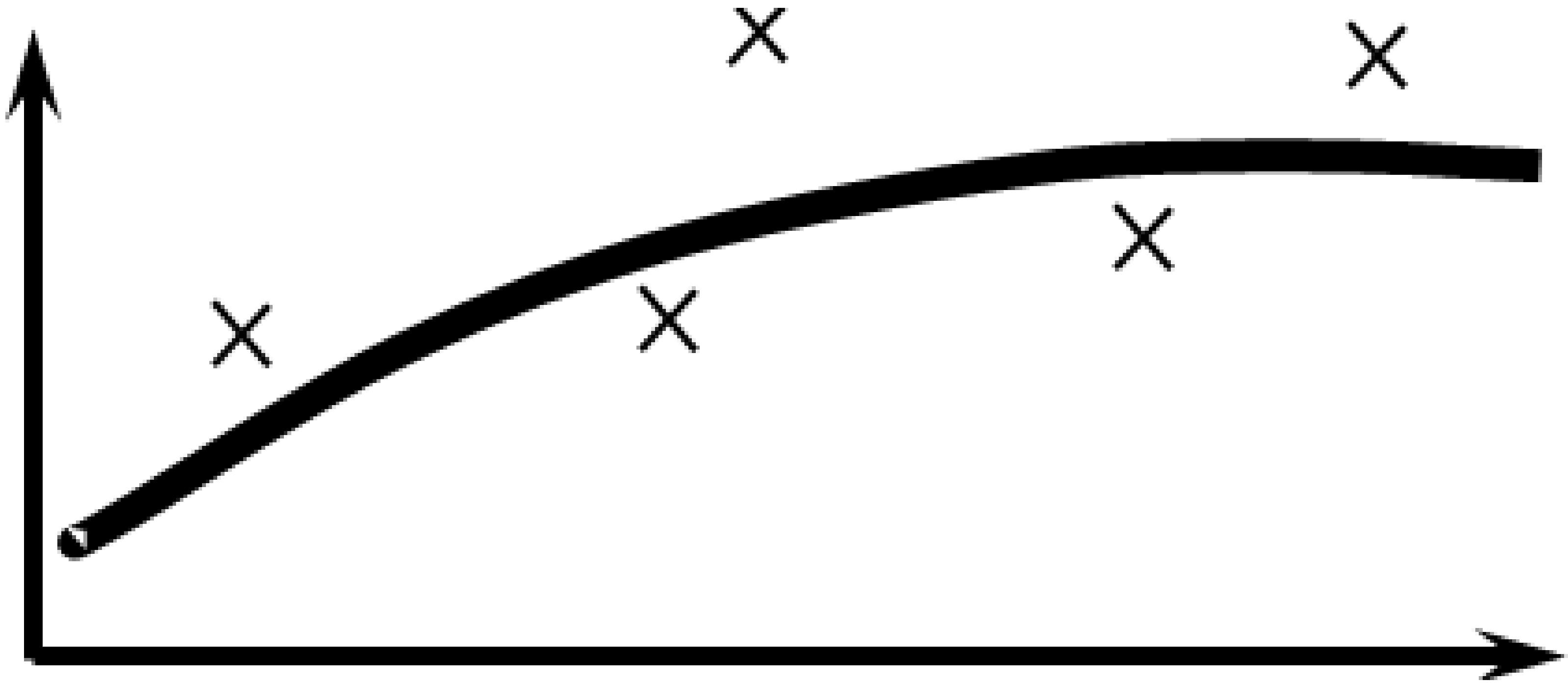
Cuarta Parte:
Clase 6 – *VFA*: aproximaciones paramétricas

Optimización Dinámica - ICS

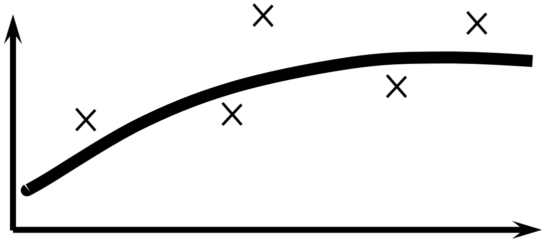


Mathias Klapp

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.



- **VFA con aproximación paramétrica.**
- Modelos lineales.
- AVI paramétrico lineal.
- Explotando estructura
- Otros modelos

VFA con aproximación paramétrica.

- En etapa t y estado s_t, decide mediante:

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

- Value-to-go real $Q_t(y_t)$ es aproximado por función $\bar{Q}_t(y_t, \boldsymbol{\theta})$ dependiente de un vector θ de parámetros calibrados offline.

- Ejemplo: Si $y_t \in \mathbb{R}$, una posible aproximación podría ser:

$$\bar{Q}_t(y_t, \boldsymbol{\theta}) = \theta_0 + \theta_1 \cdot y_t + \theta_2 \cdot y_t^2 + \theta_3 \cdot t$$

- VFA con aproximación paramétrica.
- Modelos lineales.
- AVI paramétrico lineal.
- Explotando estructura
- Otros modelos

Modelo de aproximación lineal:

- Se usa un modelo lineal en los parámetros a calibrar:

$$\bar{Q}_t(y,\theta) = \sum_{f\in\mathcal{F}} \theta_f \cdot \phi_f(y,t) = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(y,t)$$

, donde:

1. $\mathcal{F}$ es el conjunto de características seleccionadas del estado.
   - Por ejemplo si y es una ruta. $\mathcal{F} = \{$duración, # de clientes en ruta$\}$.

2. $\phi_f(y,t)$ es el valor de la característica f en el estado y *etapa* t.
   - $\phi_1([\{0,2,3,6,0\}, t = 8]) = 8$, $\phi_2([\{0,2,3,6,0\}, t = 8]) = 3$

3. $\boldsymbol{\theta}$ es el vector de parámetros que pesan las características.
   - $\boldsymbol{\theta} = (\theta_1, \theta_2)$

Modelo lineal dependiente del tiempo:

- Aproximación lineal con parámetros dependientes del tiempo:

$$\bar{Q}_t(y, \theta_t) = \sum_{f \in \mathcal{F}_t} \theta_f^t \phi_f^t(y) = \theta^t \cdot \phi^t(y)$$

, donde:

1. $\mathcal{F}_\mathrm{t}$ es el conjunto de características base seleccionadas en etapa t.
   - Por ejemplo, si y es una ruta. $\mathcal{F} = \{\#\text{ de clientes en ruta}\}$.
2. $\phi_f^t(y)$ es el valor de la característica f de la etapa t en el estado y.
   - $\phi_1^8([\{0,2,3,6,0\}]) = 3$
3. θ^t es el peso relativo de las características en la etapa t.
   - $\theta^8 = (\theta_1^8)$

- VFA con aproximación paramétrica.
- Modelos lineales.
- AVI paramétrico lineal.
- Explotando estructura
- Otros modelos

AVI paramétrico: version indep. del tiempo

1. Iniciar con $\theta \leftarrow \theta^0$.
2. Para $n = 1, .., N$
    1. Iniciar estado inicial $s_1^n \leftarrow s_1$
    2. Para cada $t = 1, \dots, T$
        - $x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{ r_t(s_t^n, x) + \boldsymbol{\theta} \cdot \boldsymbol{\phi}(y(s_t^n, x), t) \}$,
        - $y_t^n \leftarrow y(s_t^n, x_t^n)$,
        - $\boldsymbol{\phi_t^n} \leftarrow \boldsymbol{\phi}(y_t^n, t)$,
        - $s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n)$
    3. Para cada $t = T, T-1 \dots, 1$:
        - $q_t^n \leftarrow r_t(s_t^n, x_t^n) + \boldsymbol{\theta} \cdot \boldsymbol{\phi_t^n}$
        - Recalibrar $\boldsymbol{\theta}$ con la información disponible (q_t^n, $\boldsymbol{\phi_t^n}$)

Retornar θ

¿Cómo calibrar θ en la corrida n?

- Minimizar la norma del vector de errores e sobre todas las observaciones:

$$\min_{\theta} \|e\|^2 = \min_{\theta} \|Q_m - \Phi_m \cdot \theta\|^2$$

, donde $Q_m = \{q_i\}_{i \in [m]}$ es un vector con m observaciones del *value-to-go* y $\Phi_m = \left\{\phi_{i,f}\right\}_{i \in [m], f \in \mathcal{F}}$ es la matriz de m observaciones de variables independientes, es decir:

$$Q_m = \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix}, \qquad \Phi_m = \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,F} \\ \vdots & \ddots & \vdots \\ \phi_{m,1} & \cdots & \phi_{m,F} \end{bmatrix} = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_m^T \end{bmatrix}, \qquad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_F \end{bmatrix}$$

Algoritmo MC: Mínimos Cuadrados

$$\min_{\theta} \|Q_m - \Phi_m \theta\|^2 = (Q_m - \Phi_m \theta)^T (Q_m - \Phi_m \theta)$$

$$\min_{\theta} Q_m^T Q_m - 2 Q_m^T \Phi_m \theta + \theta^T \Phi_m^T \Phi_m \theta$$

Derivando e igualando a cero:

$$-2\Phi_m^T Q_m + 2\Phi_m^T \Phi_m \theta = 0$$

Implica que:

$$\theta_m^* = \underbrace{\left(\Phi_m^T \Phi_m\right)^{-1}}_{B_m} \underbrace{(\Phi_m^T Q_m)}_{Z_m} = B_m Z_m$$

¿Es lo anterior suficiente?

El algoritmo anterior es una primera idea, pero:

1. Es lento. Recalcula MC desde cero en cada corrida.
   - **Solución:** Mínimos cuadrados recursivos
2. Asume estacionariedad en distribución de errores.
   - **Solución:** Ponderación menor para datos del pasado.

Mínimos Cuadrados Recursivos

- Versión de mínimos cuadrados que calcula θ_m^* como función de θ_{m-1}^* y el último dato (q_m, ϕ_m) evitando hacer MC.

- Recordemos que $\theta_m^* = B_m Z_m$. Queremos evitar calcular $B_m = (\Phi_m^T \Phi_m)^{-1}$ y $\mathrm{Z}_m = \Phi_m^T Q_m$ en cada corrida.

- Detalles en libro de Powell (2011), página 405

Observación 1: Recursión de B_m^{-1}

$$B_m = (\Phi_m^T \Phi_m)^{-1} = \left(\begin{bmatrix} \phi_1 & \dots & \phi_m \end{bmatrix} \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_m^T \end{bmatrix} \right)^{-1} = \left(\sum_{i=1}^{m} \phi_i \phi_i^T \right)^{-1}$$

$$= \left(\sum_{i=1}^{m-1} \phi_i \phi_i^T + \phi_m \phi_m^T \right)^{-1} = (B_{m-1}^{-1} + \phi_m \phi_m^T)^{-1}$$

Es decir:

$$\boldsymbol{B_m^{-1} = B_{m-1}^{-1} + \phi_m \phi_m^T}$$

Lemma:

Sean B', B son matrices positivas definidas $\mathbb{R}^{p\times p}$ y $v \in \mathbb{R}^p$ tal que:

$$B'^{-1} = B^{-1} + vv^T$$

, entonce se cumple que:

$$B' = B - \frac{Bvv^T B}{(1 + v^T Bv)}$$

Probar: mulplicando y obteniendo la identidad!

Observación 2: Recursión de B_m

- Si $B_m^{-1} = B_m^{-1} + \phi_m \phi_m^T$, entonces:

$$B_m = B_{m-1} - \frac{B_{m-1}\phi_m\phi_m^T B_{m-1}}{1 + \phi_m{}^T B_{m-1}\phi_m}$$

- Definamos la **Ganancia** $\boldsymbol{K_m} = \frac{\boldsymbol{B_{m-1}\phi_m}}{\boldsymbol{1+\phi_m{}^T B_{m-1}\phi_m}}$. Luego:

$$B_m = B_{m-1} - K_m \phi_m^T B_{m-1}$$

Propiedad de la ganancia K_m

- Propiedad: $$\boldsymbol{K_m = B_m \phi_m}$$

Demostración de la propiedad:

$$K_m = \frac{B_{m-1}\phi_m}{1+\phi_m{}^T B_{m-1}\phi_m}.$$ Luego:

$$\left(1 + \phi_m{}^T B_{m-1}\phi_m\right) K_m = B_{m-1}\phi_m$$

Es decir:

$$\begin{aligned} K_m &= B_{m-1}\phi_m - K_m \phi_m{}^T B_{m-1}\phi_m \\ &= \left(B_{m-1} - K_m \phi_m{}^T B_{m-1}\right)\phi_m \\ &= B_m \phi_m \end{aligned}$$

Observación 3: Recursión de Z_m

$$Z_m = \Phi_m^T Q_m = [\phi_1 \quad \cdots \quad \phi_m] \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} = \sum_{i=1}^{m} q_i \phi_i = \sum_{i=1}^{m-1} q_i \phi_i + q_m \phi_m$$

$$= Z_{m-1} + q_m \phi_m$$

Es decir:

$$\boldsymbol{Z_m = Z_{m-1} + q_m \phi_m}$$

Ya tenemos todos los elementos!

Derivación del cálculo recursivo de θ_m^*

De mínimos cuadrados tenemos que:

$$\theta_m^* = B_m Z_m = B_m(Z_{m-1} + q_m \phi_m)$$

$$\theta_m^* = B_m Z_{m-1} + q_m B_m \phi_m$$

$$\theta_m^* = (B_{m-1} - K_m \phi_m^T B_{m-1}) Z_{m-1} + B_m \phi_m q_m$$

$$\theta_m^* = \theta_{m-1}^* - K_m \phi_m^T \theta_{m-1}^* + K_m q_m$$

$$\theta_m^* = \theta_{m-1}^* - K_m(\phi_m^T \theta_{m-1}^* - q_m)$$

Es decir:

$$\boldsymbol{\theta_m^* = \theta_{m-1}^* - K_m \zeta_m}$$

, donde $\boldsymbol{\zeta_m = \phi_m^T \theta_{m-1}^* - q_m}$ es el error de la calibración al predecir el value-to-go del dato nuevo.

Algoritmo MCR: Mínimos Cuadrados Recursivos

Actualizacion de $\boldsymbol{\theta_m^*}$ dato en función de $\boldsymbol{\theta_{m-1}^*, B_{m-1}, q_m, \phi_m}$:

1. Calcular ganancia y error:
   - $K_m = \frac{B_{m-1}\phi_m}{1+\phi_m^T B_{m-1}\phi_m}$
   - $\zeta_m = \phi_m^T \theta_{m-1}^* - q_m$
2. Ajustar calibración: $\theta_m^* = \theta_{m-1}^* - K_m \zeta_m$
3. Preparar B_m para prox. ajuste: $B_m = B_{m-1} - K_m \phi_m^T B_{m-1}$
4. Retornar θ_m^*

¿Cómo iniciar?:

1. Aproximado: $B_0 = \epsilon \cdot I$, donde $\epsilon > 0$
2. Hacer un lote inicial con MC clásico

¿Es lo anterior suficiente?

El algoritmo anterior es una primera idea, pero:

1. Es lento. Recalcula MC desde cero en cada corrida.
    - **Solución:** Mínimos cuadrados recursivos

2. Asume estacionariedad en distribución de errores.
    - **Solución:** Ponderación menor para datos del pasado.

Caso no estacionario

- Se ajusta el algoritmo anterior por un peso $0 < \alpha_m \leq 1$.
- Mientras más bajo es α_m menos se considera el pasado

- Valor que típicamente brinda buenos resultados:

$$\alpha_m = \begin{cases} 1 & \text{estacionario} \\ 1 - \delta/n(m) & \text{no estacionario} \end{cases}$$

, donde típicamenye $\delta = 0{,}5$ y $n(m)$ es la corrida del dato m.

- Se deriva de MC recursivos para mínimos cuadrados ponderados.

Algoritmo MCR no estacionario

Actualizacion de $\boldsymbol{\theta_m^*}$ dato en función de $\boldsymbol{\theta_{m-1}^*}, \boldsymbol{B_{m-1}}, \boldsymbol{q_m}, \boldsymbol{\phi_m}$:

1. Calcular ganancia y error:
   - $K_m = \frac{B_{m-1}\phi_m}{\alpha_m + \phi_m^T B_{m-1}\phi_m}$
   - $\zeta_m = \phi_m^T \theta_{m-1}^* - q_m$
2. Ajustar calibración: $\theta_m^* = \theta_{m-1}^* - K_m \zeta_m$
3. Preparar: $B_m = \frac{1}{\alpha_m}\left(B_{m-1} - K_m \phi_m^T B_{m-1}\right)$
4. Retornar θ_m^*

AVI paramétrico: version indep. del tiempo

1. Iniciar con $\theta \leftarrow \theta^0$.
2. Para $n = 1, .., N$
   1. Iniciar estado inicial $s_1^n \leftarrow s_1$
   2. Para cada $t = 1, \dots, T$
      - $x_t^n \leftarrow \underset{x \in \mathbb{X}_t(s_t^n)}{\operatorname{argmax}} \{r_t(s_t^n, x) + \theta \cdot \phi(y(s_t^n, x), t)\}$,
      - $y_t^n \leftarrow y(s_t^n, x_t^n)$,
      - $\phi_t^n \leftarrow \phi(y_t^n, t)$,
      - $s_{t+1}^n \leftarrow f(y_t^n, \omega_t^n)$
   3. Para cada $t = T, T-1 \dots, 1$:
      - $q_t^n \leftarrow r_t(s_t^n, x_t^n) + \theta \cdot \phi_t^n$
      - Ajustar θ consinderando el nuevo dato con MCR

Retornar θ

VFA paramétrico lineal

Ventaja:

- Bajo esfuerzo computacional *online* una vez entrenado Q.
- Muy simple, explota simulación computacional, optimización determinística y regresión lineal.
- Estructura lineal: Ideal para resolver paso de optimización con MILP
- Calibración no exige pasar por todos los estados de post-decisión
    - No importa si el espacio es enorme
    - Permite generar un modelo robusto, sin tantas observaciones.

Desventaja:

- Depende fuertemente de la función supuesta y de las características escogidas.

Recomendación del Chef:

- Cuando hace sentido imponer estructura lineal.
- Cuando espacio de estados es inmenso.
- Para decisiones rápidas, cuando el cálculo *online* es caro.

- VFA con aproximación paramétrica.
- Modelos lineales.
- AVI paramétrico lineal.
- Explotando estructura
- Otros modelos

Alternativa: Aprendizaje por diferencias *(TD-learning)*

Forma alternativa para calibrar pendientes lineales.

Supongamos que se desea estimar $V_t(s_t) \approx \theta_t s_t$, donde $s_t \in \mathbb{R}_+^n$ y el valor de $\theta_t = \frac{dV_t(s_t)}{ds_t}$ es la pendiente a estimar. Notar que $\theta_t \approx V_t(s_t + 1) - V_t(s_t)$

La idea es estimar θ_t^n recursivamente en simulaciones mediante diferencias:

$$\theta_t^n \leftarrow \alpha_n\left(v_t^{n^+} - v_t^n\right) + (1 - \alpha_n)\theta_t^{n-1}$$

donde:

$$v_t^n = \text{argmax}_{x \in \mathbb{X}_t(s_t)}\{r_t(s_t, x) + \mathbb{E}(\theta_{t+1}^{n-1} s_{t+1} | s_t, x)\}$$

$$v_t^{n+} = \text{argmax}_{x \in \mathbb{X}_t(s_t+1)}\{r_t(s_t + 1, x) + \mathbb{E}(\theta_{t+1}^{n-1} s_{t+1} | s_t + 1, x)\}$$

Explotanto estructura

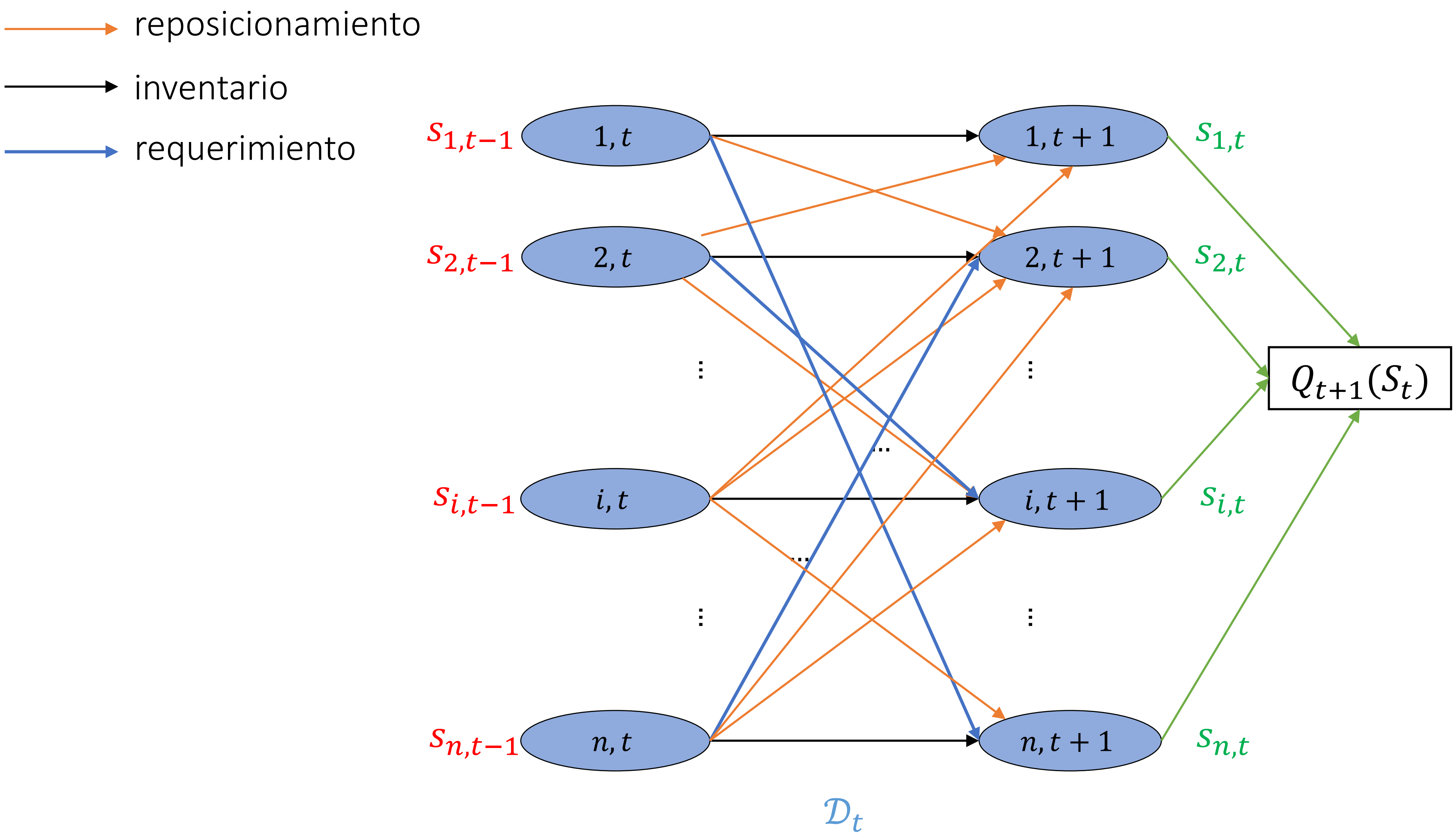
- ¿Y si el problema posee estructura?
- Por ejemplo, se presume un value-to-go cóncavo. ¿Qué hacer?
- Veremos un ejemplo ``genial''

Recordemos el Problema de Gestion Dinamica de RRMM.

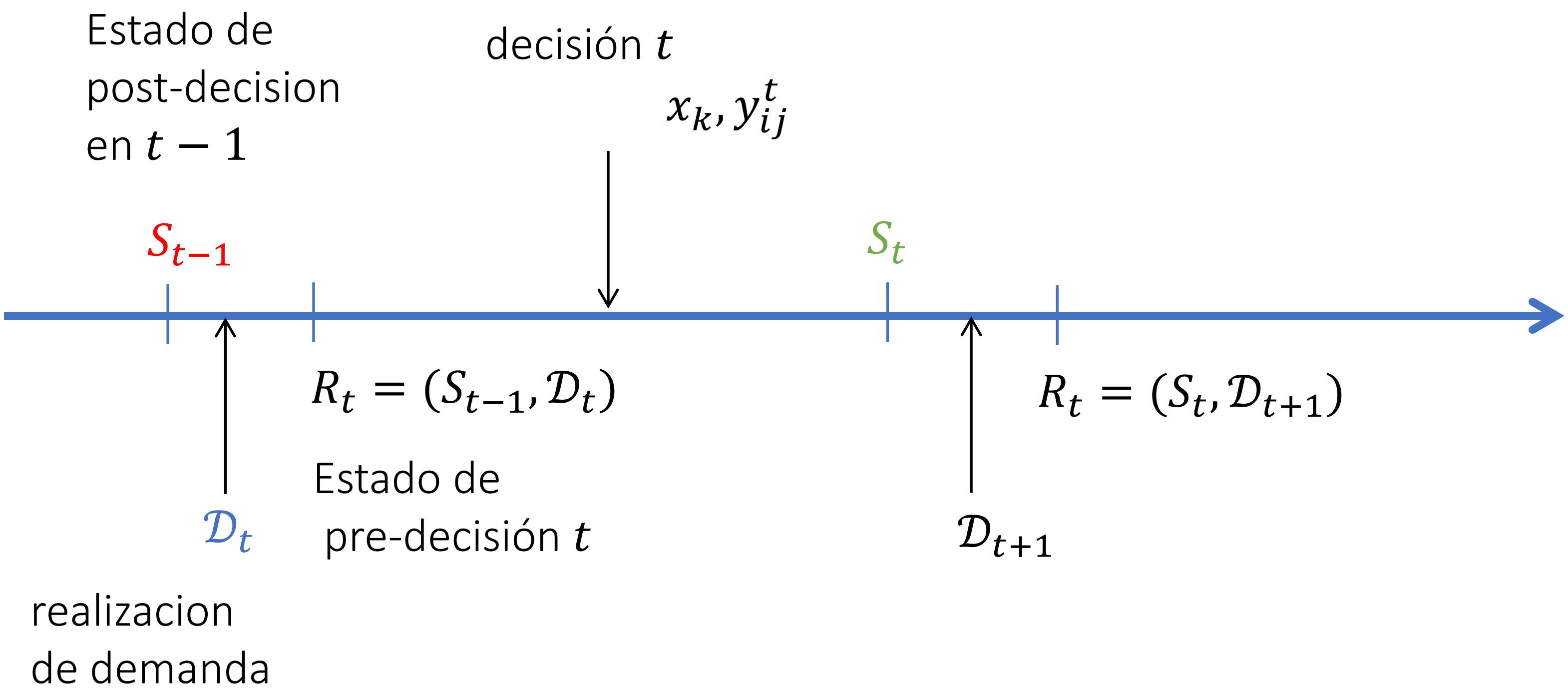
- Estado de pre-decisión sistema: $R_t := (S_{t-1}, \mathcal{D}_t)$
  - $S_{t-1} = (s_{it-1})_{i\in[n]}$: vector de RRMM disponibles en cada nodo i.
  - $\mathcal{D}_t$: requerimientos revelados para el periodo t.

- Decisiones:
  - x_k: Si requerimiento k se sirve en el día t, para todo $k \in \mathcal{D}_t$
  - y_{ij}^t : Traslado de RRMMs vacíos desde i hacia j iniciando viaje en t

Gestión dinámica de RRMM: etapa t



Gestión dinámica de RRMM: decisiones



Gestión dinámica de RRMM: MDP

- Valor inmediato:

$$r_t(\mathcal{D}_t, x, y_t) = \sum_{k \in \mathcal{D}_t} r_k x_k - \left(\sum_{i \in N} h_i(t) y_{ii}^t + \sum_{(i,j) \in A_t} c_{ij}(t) y_{ij}^t \right)$$

- Espacio de decisiones:

$$\mathbb{X}_t(S_{t-1}, \mathcal{D}_t) \coloneqq \{x_k \in \{0,1\}, y_{ij}^t \in \mathbb{Z}^+:$$

$$\textstyle\sum_{j \in N} y_{ij}^t + \sum_{k \in \mathcal{D}_{it}^+} x_k = s_{it-1}, \quad \forall i \in [n],$$

- Transición a estado de post-decision S_t:

$$\textstyle\sum_{j \in N} y_{ji}^t + \sum_{k \in \mathcal{D}_{it+1}^-} x_k = s_{it}, \quad \forall i \in [n],$$

- MDP:

$$V_t(\mathcal{D}_t, S_{t-1}) = \max_{(x,y_t) \in \mathbb{X}(S_{t-1}, \mathcal{D}_t)} r_t(\mathcal{D}_t, x, y_t) + Q_t(s_t)$$

Supuesto 1: $Q_t(S_t)$ separable por nodo

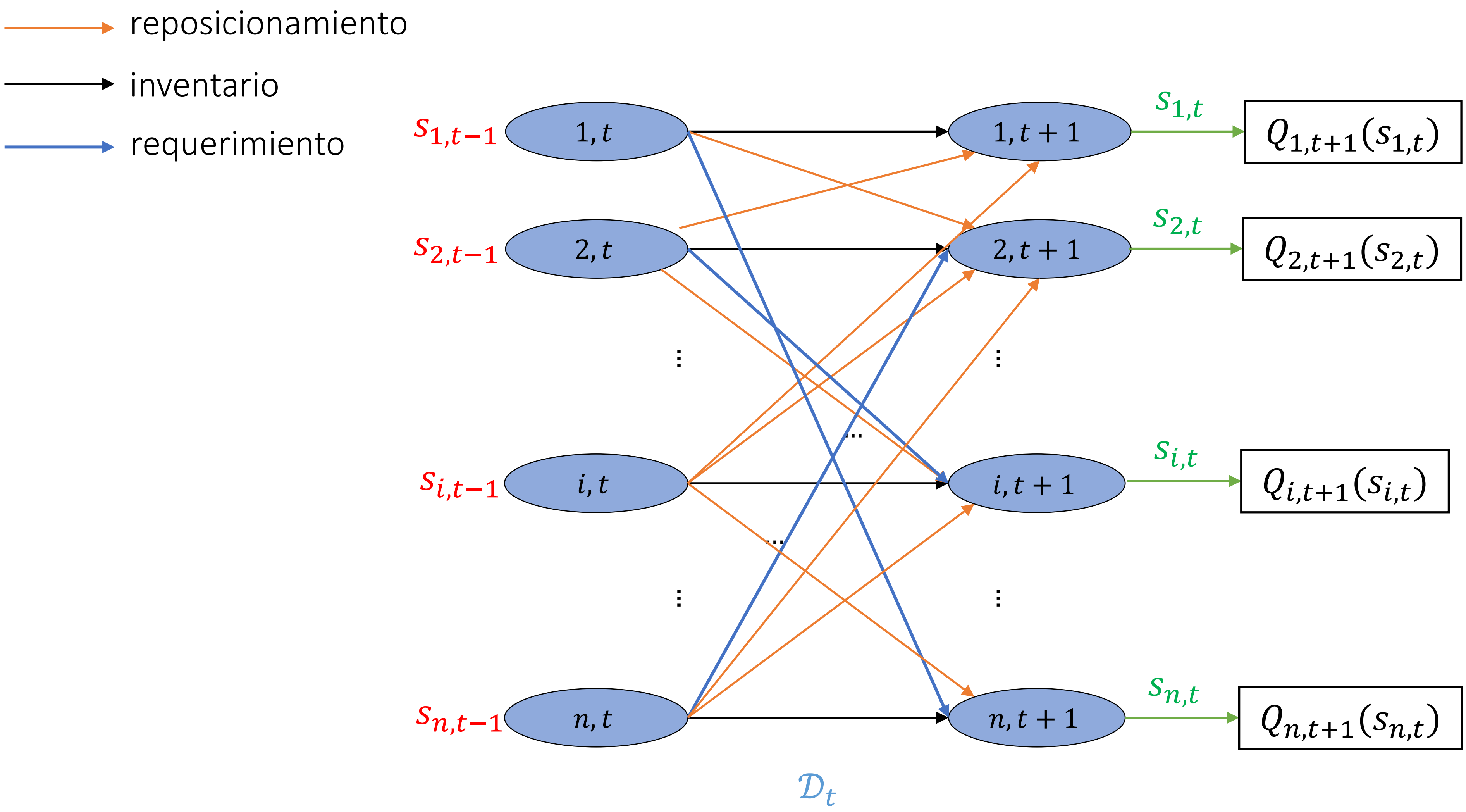
- Se asume que:

$$Q_t(S_t) \approx \sum_{i \in N} Q_{i,t}(s_{i,t})$$

- Cambio de recursos en un nodo no impacta al *value-to-go* que aporta otro nodo (desprecia efectos de red a futuro).

- No siempre es válido, pero sirve para tomar decisiones.

Gestión dinámica de RRMM: etapa t

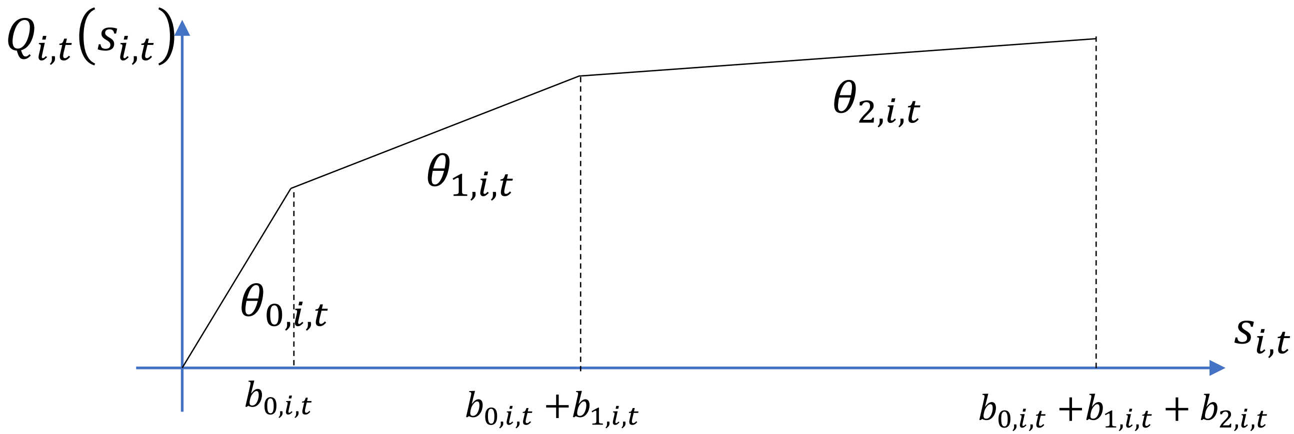


Supuesto 2: $Q_{i,t}(s_{\mathrm{i,t}})$ es cóncava y lineal a tramos

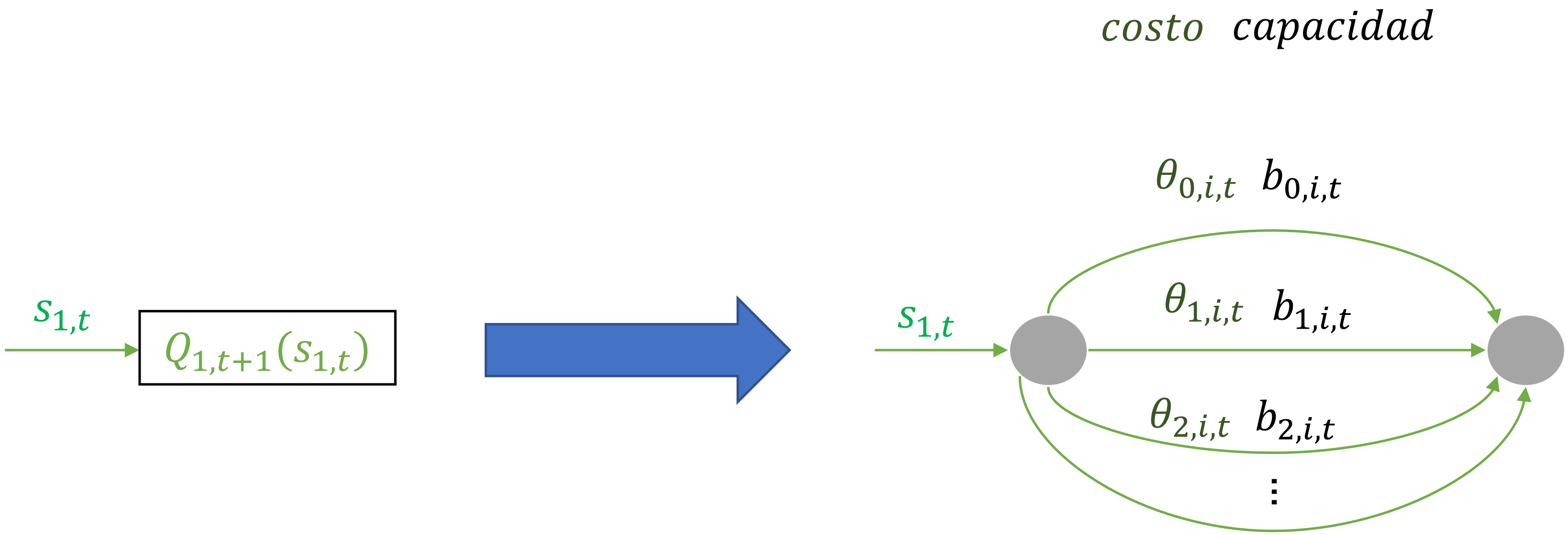
- Se asume que:

$$dQ_{i,t}(s_{i,t})/ds_{i,t} = \begin{cases} \theta_{0,i,t} & 0 \leq s_{i,t} \leq b_{0,i,t} \\ \theta_{1,i,t} & b_{0,i,t} \leq s_{i,t} \leq b_{0,i,t} + b_{1,i,t} \\ \theta_{2,i,t} & b_{0,i,t} + b_{1,i,t} < s_{i,t} \leq b_{0,i,t} + b_{1,i,t} + +b_{2,i,t} \\ \dots & \dots. \end{cases}$$

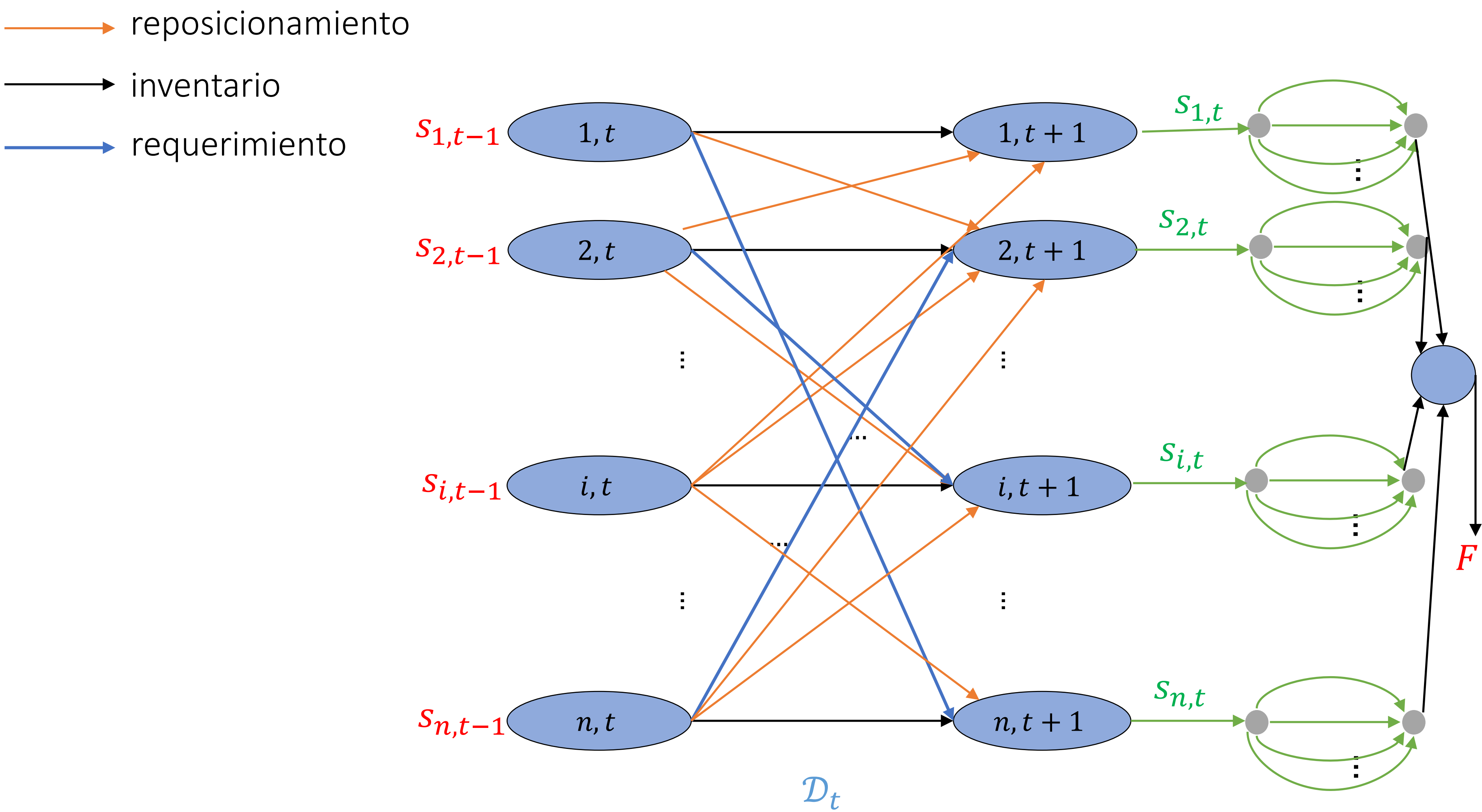
, donde $\theta_{0,i,t} \geq \theta_{1,i,t} \geq \theta_{2,i,t} \geq \cdots$ No siempre es válido, pero sirve para tomar decisiones.



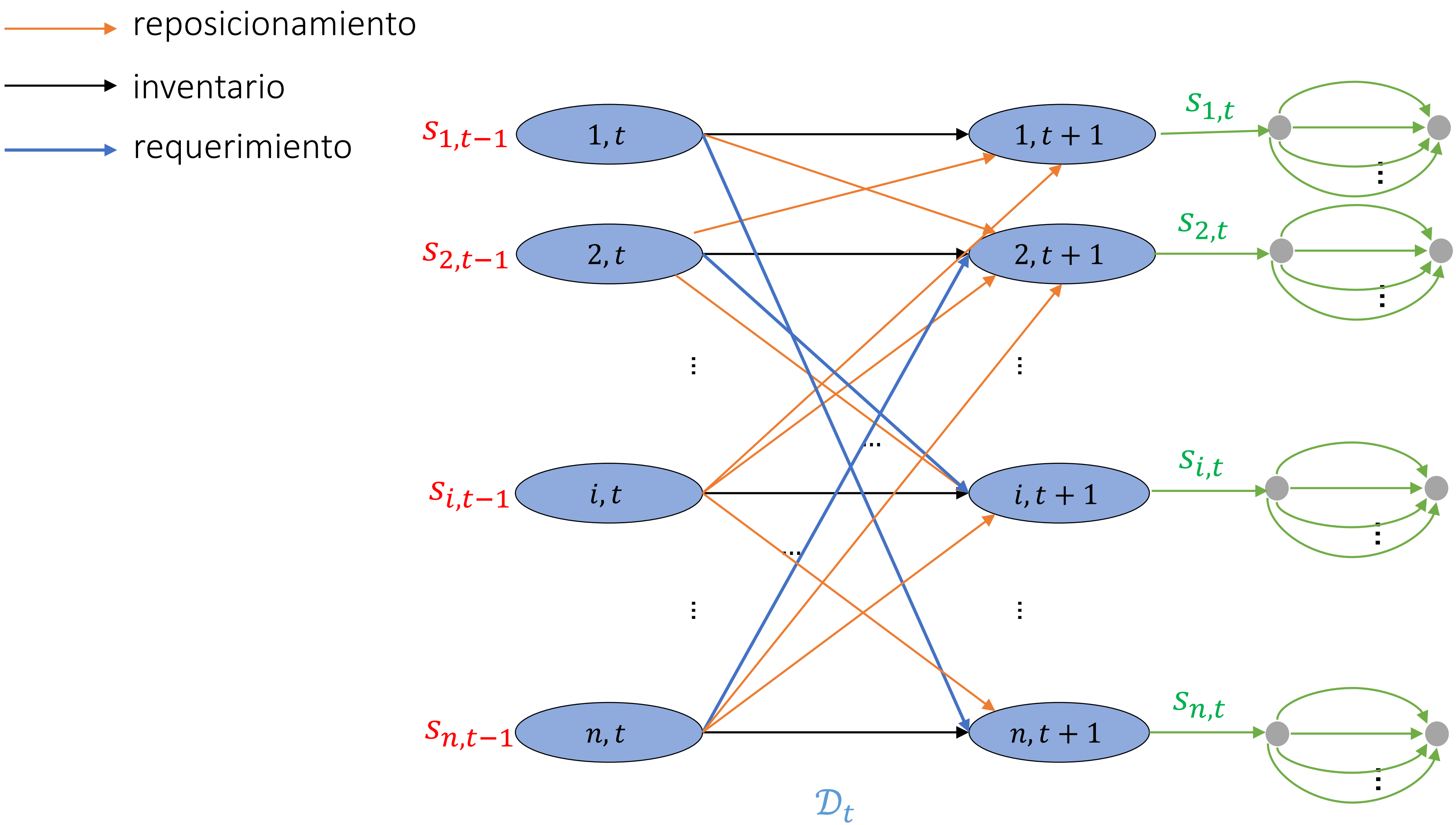
Gestión dinámica de RRMM: etapa t



Gestión dinámica de RRMM: etapa t



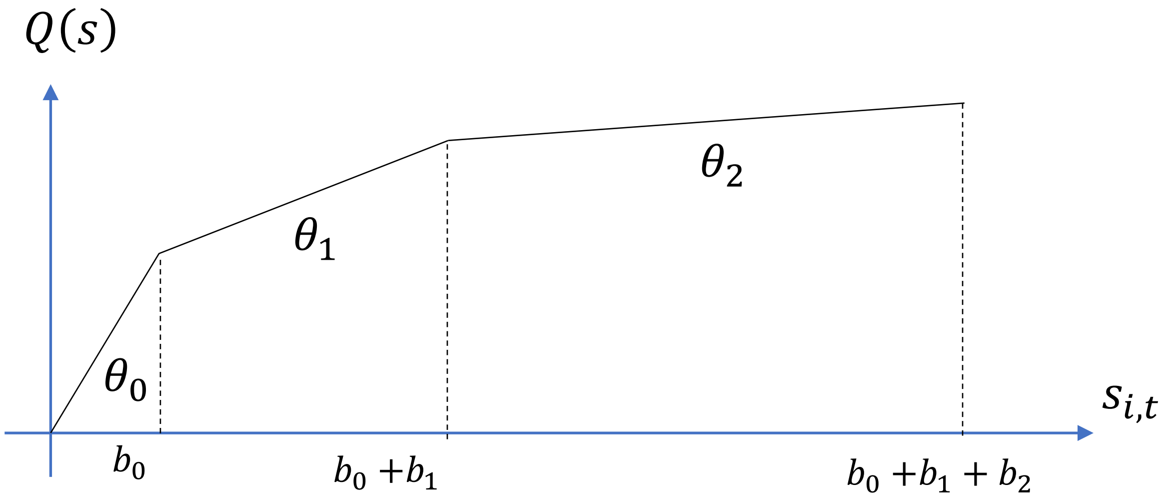
Gestión dinámica de RRMM: etapa t



UNA VEZ CALIBRADO, ES UN PROBLEMA DE FLUJO EN REDES

¿Cómo calibramos?

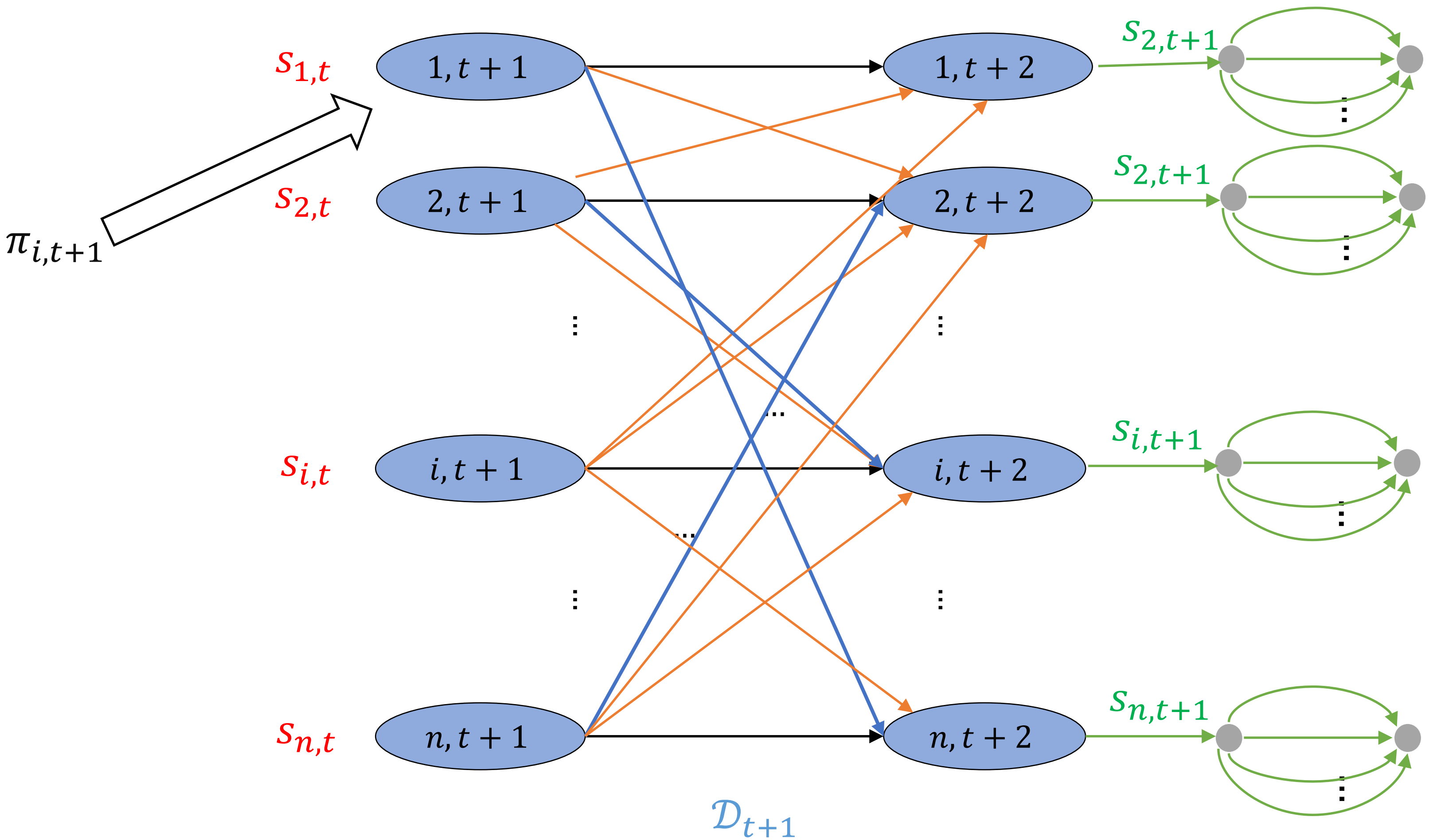
¿Conceptualmente que es θ_p?



- Representa $\frac{dQ_{it}(s_{it})}{ds}$, es decir, el cambio en $Q(s)$ al aumentar en una unidad los recursos móviles disponibles en i el periodo $t+1$.
- Ver Powell, pag 501 para detalles.

Vamos un periodo adelante

¿Qué representa la variable dual del nodo $(i, t+1)$.



Vamos un periodo adelante

¿Qué representa la variable dual del nodo $(i, t+1)$?

$$\frac{dQ_{it}(s_{it})}{ds} = \mathbb{E}_{D_{t+1}}[\pi_{i,t+1}|s_{i,t}]$$

- Por lo tanto $\pi_{i,t+1}(\omega_t)$ es un estimador sin sesgo pero con mucha varianza de $\frac{dQ_{it}(s_{it})}{ds}$.

- Calibrarmediante algoritmo similar a MCR.

- VFA con aproximación paramétrica.

- Modelos lineales.

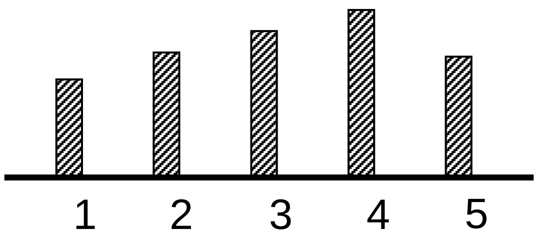
- AVI paramétrico lineal.

- Explotando estructura

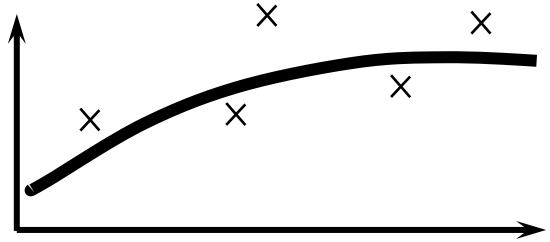
- Otros modelos

Resumen de aproximaciones para $Q_t(y_t)$

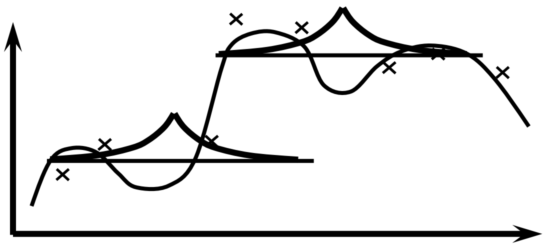
- Tabla de valores (*Lookup table*)
  - Un valor estimado para cada estado discreto
  - Desafío: memoria.



- **Aproximación paramétrica:**
  - Modelos lineales (regresión lineal) y ALP (bases)
  - Modelos no lineales
  - Explota estructura: monotonía y convexidad
  - Desafío: exige suponer una forma funcional de Q.

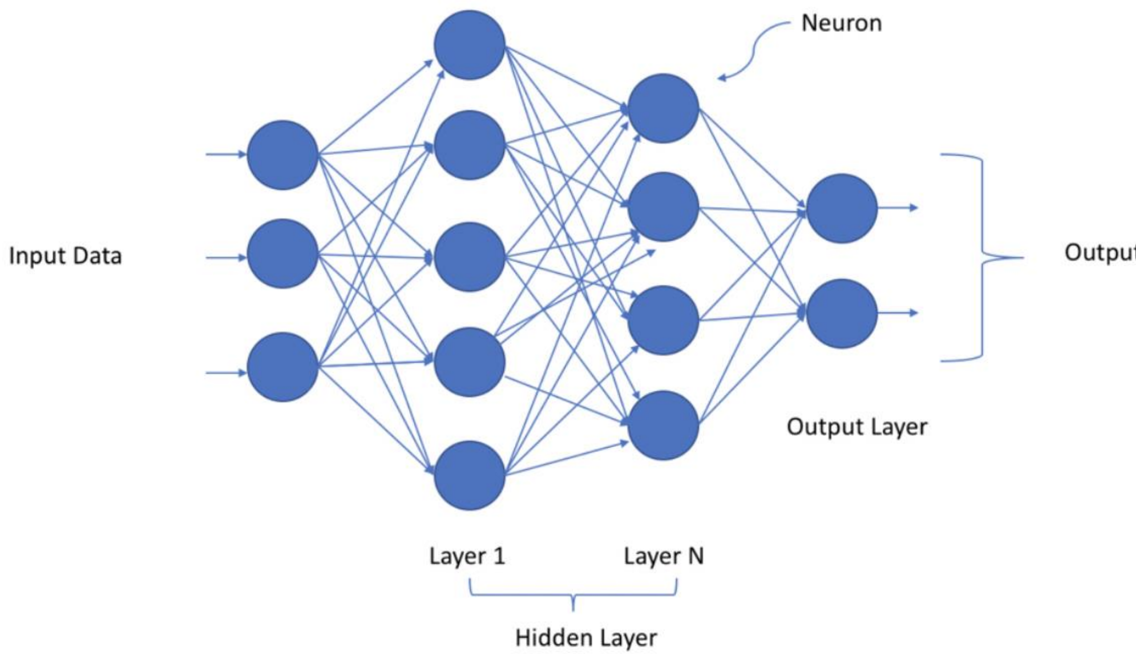


- **Aproximación no-paramétrica:**
  - k −*nearest neighbor clustering y Kernel regression*
  - *Local polynomial regression*
  - Redes Neuronales
  - Desafío: *over-fitting.*

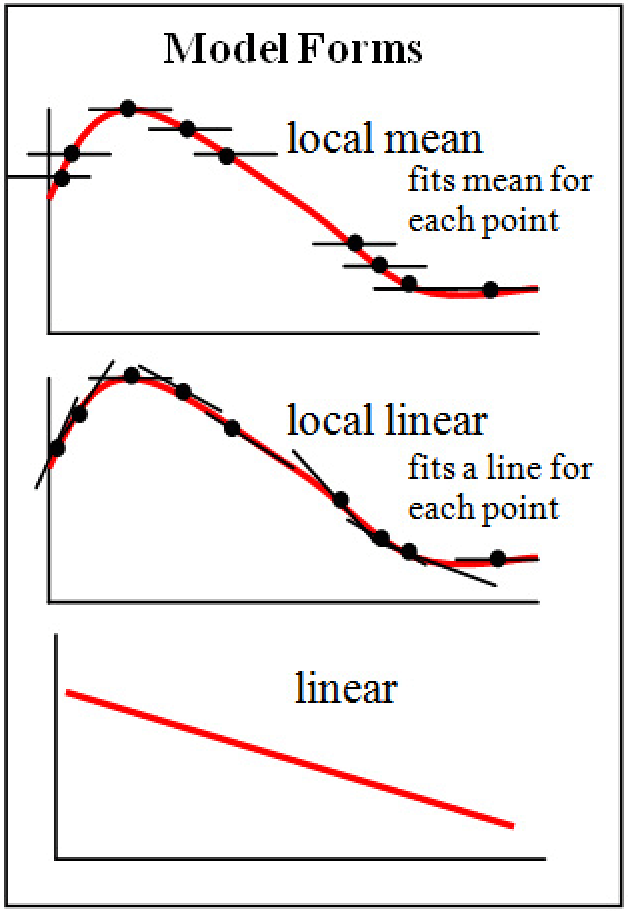


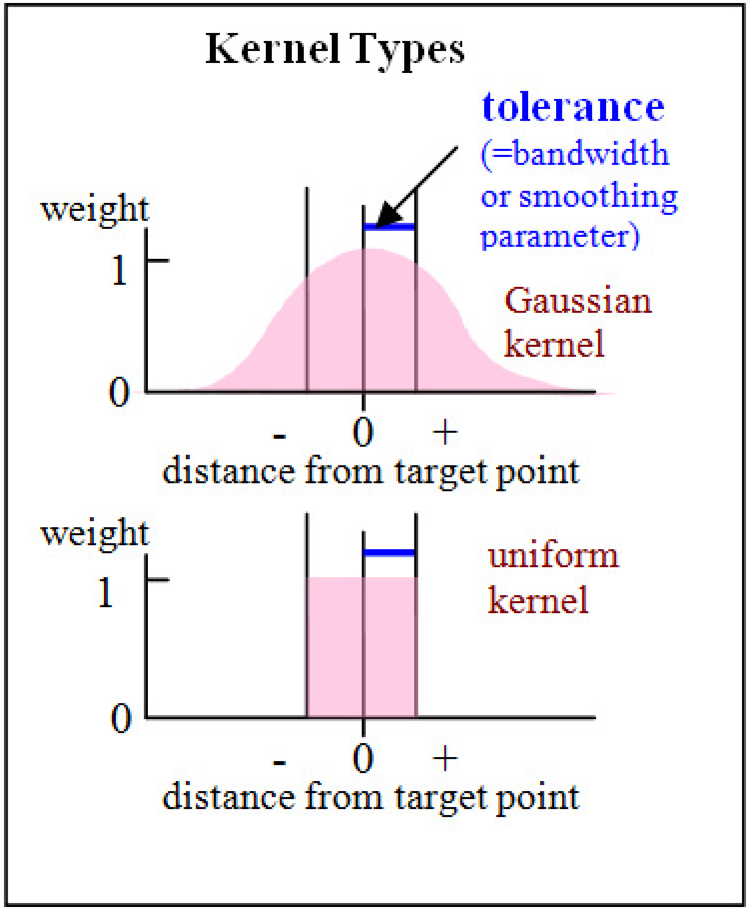
Deep Q-learning

- Deep Q-learning usa **Deep neural networks** para aproximar el value to go.
- La red neuronal es entrenada fuera de línea y luego usada.
- Se pierde estructura.



K- vecinos, Regresión Local, Regresión de Kernel,





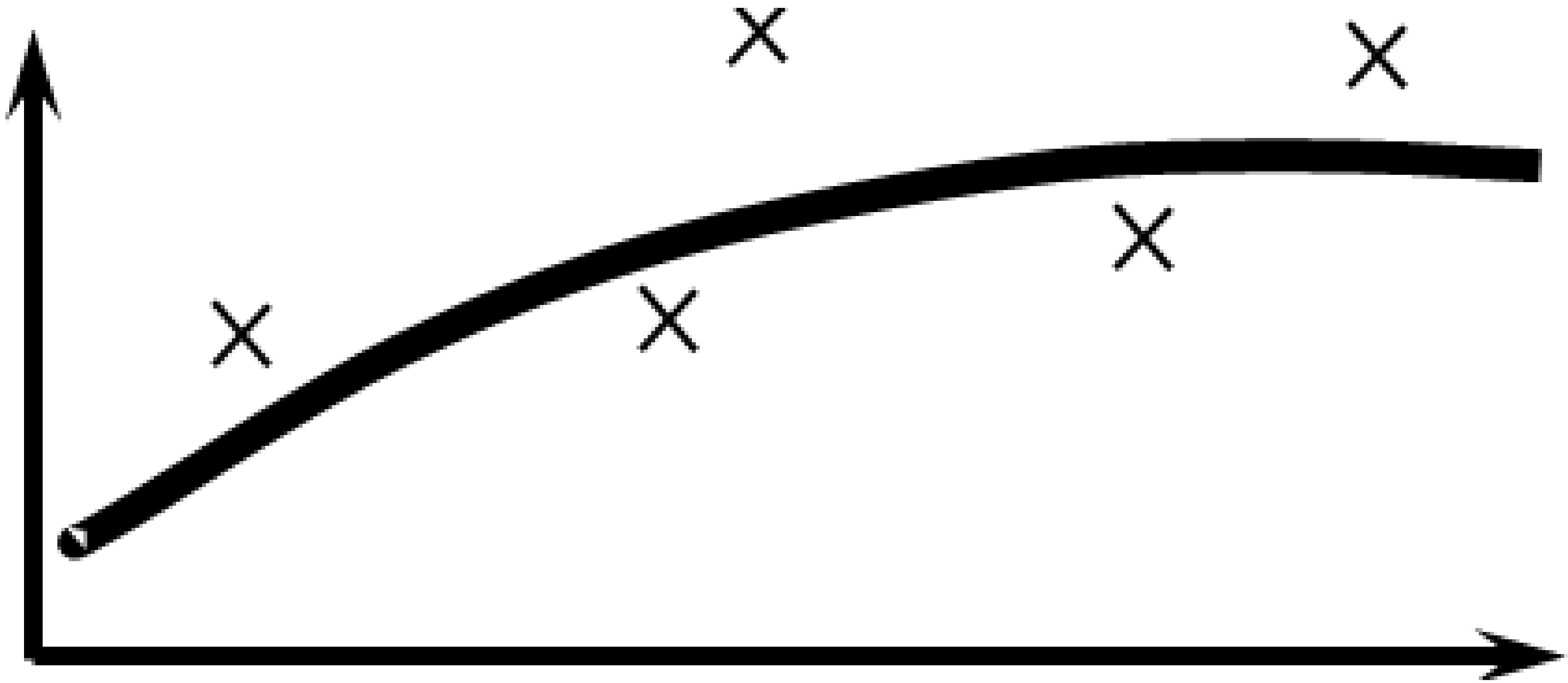
Feedback de salida

SCAN ME

https://forms.gle/NqXAdEbgbnyf6foG6

Cuarta Parte:
Clase 6 – *VFA*: aproximaciones paramétricas

Optimización Dinámica - ICS



Mathias Klapp