Aprendizaje por Refuerzo Transfiriendo Conocimiento

Esteban Omar García
Eduardo F. Morales, Enrique Muñoz de Cote
Instituto Nacional de Astrofísica, Óptica y Electrónica

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)

- 27 Centros Públicos de Investigación - Conacyt
- Creado en 1971

INAOE - Computación

- 23 investigadores de tiempo completo
- 95% en el SNI y 40% niveles II y III
- Maestría PNPC – Consolidado
- Doctorado PNPC – Internacional
- Aprendizaje – Recon. Patrones, Robótica, Tratamiento de Lenguaje Natural, Visión, Cómputo Reconfig. y de Alto Desempeño, Proc. de Bio-Señales y Aplicaciones Médicas, Cómputo y Proc. Ubicuo

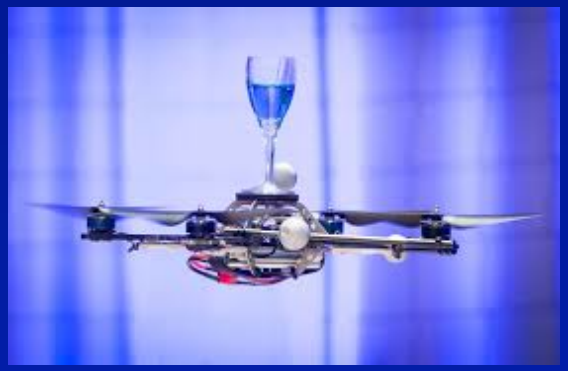
Contenido

- Motivación
- Conceptos:
  - Aprendizaje por Refuerzo
  - Aprendizaje por Transferencia
  - Procesos Gaussianos
- Propuestas
- Experimentos
- Conclusiones y Trabajo Futuro

Introducción

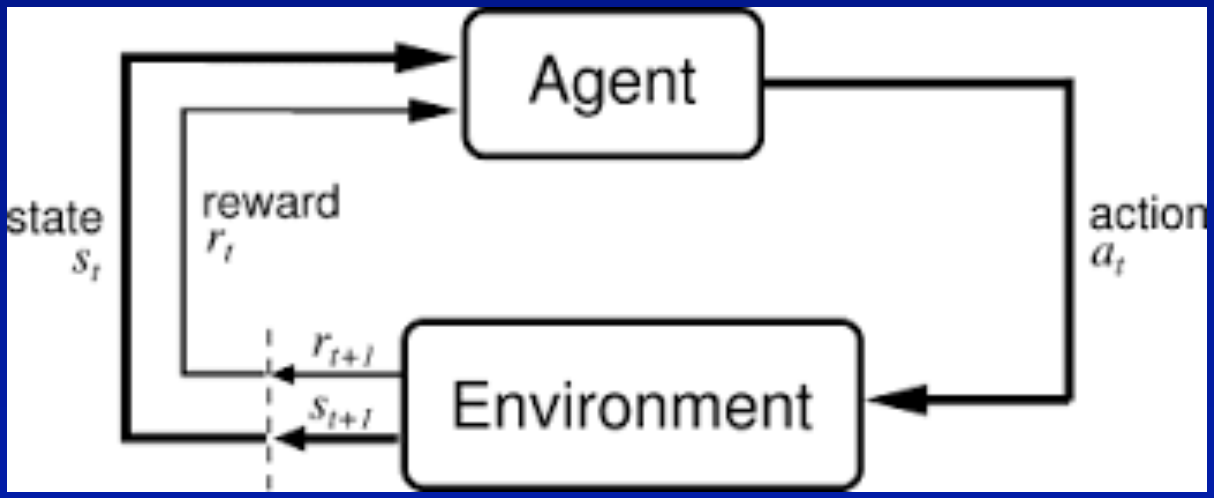
- Los robots han evolucionado y realizan tareas muy diferentes que hace algunos años
- Los ambientes son dinámicos, las tareas son complejas y difíciles de modelar

➢ Se ha recurrido a sistemas de aprendizaje



Aprendizaje por Refuerzo

- Modelado como un MDP: <S,A,P,R>
- En cada estado (s), se selecciona una acción (a), se cambia de estado (P(s'|s,a) y se recibe una recompensa (R(s,a))
- Se aprende por prueba y error a realizar una tarea explorando el ambiente



Aprendizaje por Refuerzo

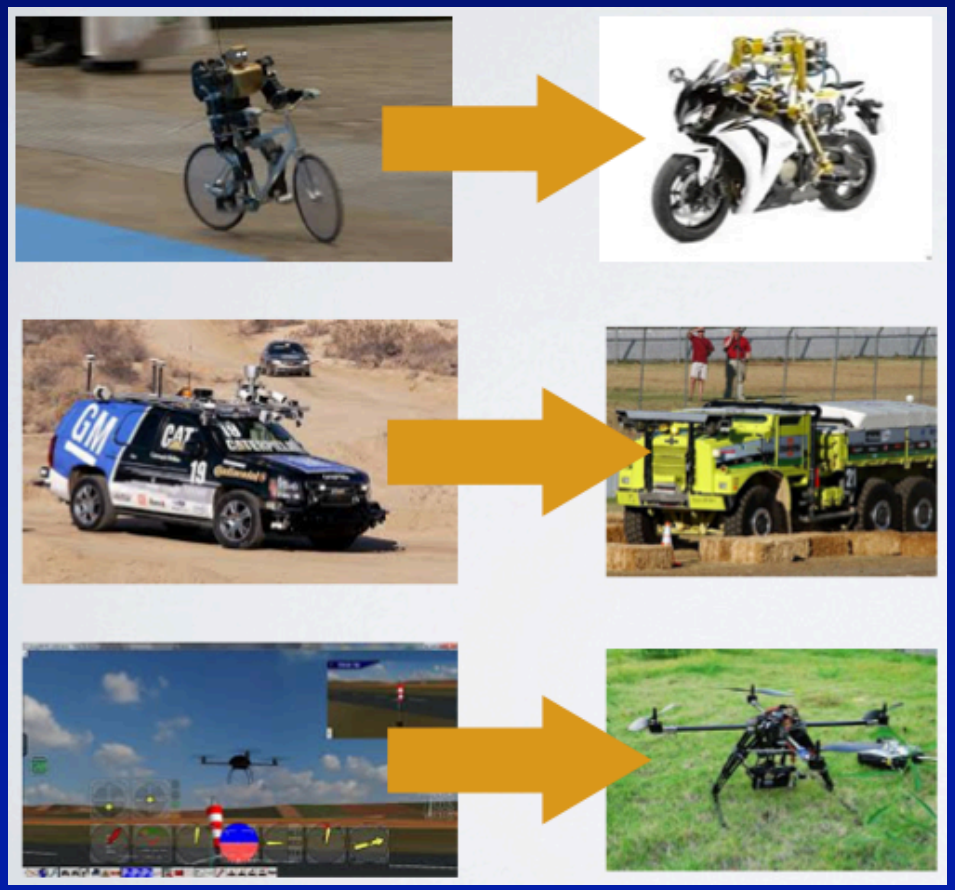
- V(s) y Q(s,a) = funciones de valor: Lo que espero recibir de recompensa
- $\pi(s) \Rightarrow a$: política:Define qué acción a realizar en cada estado
- Objetivo: Encontrar la política que maximice la recompensa acumulada esperada

Aprendizaje por Refuerzo

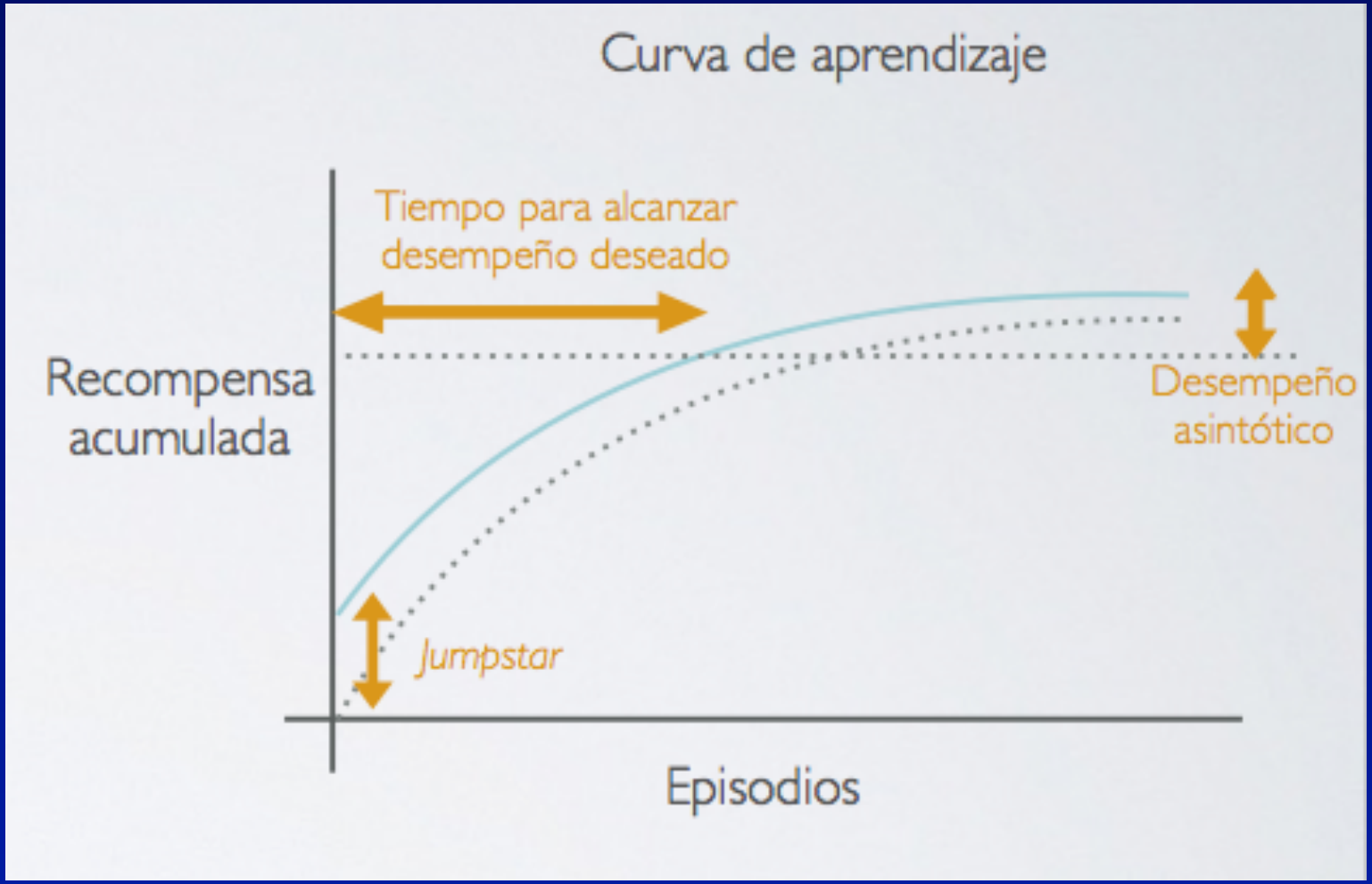
- ✓ No requiere un modelo del ambiente
- ✓ El agente aprende solo
- ✓ Converge a la política óptima
- ✗ El aprendizaje es lento
- ✗ Pocos desarrollos en ambientes complejos con variables contínuas
- ✗ No se pueden reutilizar políticas

Transfer Learning

- Idea: Aprender una tarea más rápido usando información de otra tarea similar



TL en RL


Curva de aprendizaje
Tiempo para alcanzar
desempeño deseado
Recompensa
acumulada
Desempeño
asintótico
Jumpstar
Episodios

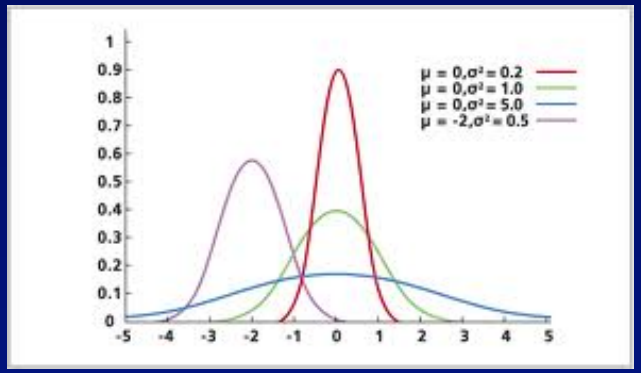
TL en RL

- En RL se pueden transferir diferentes aspectos:
  - Política (π)
  - Funciones de valor (Q o V)
  - Tuplas o ejemplos
  - Parámetros de aprendizaje
- En este trabajo transferimos parámetros y sintetizamos tuplas

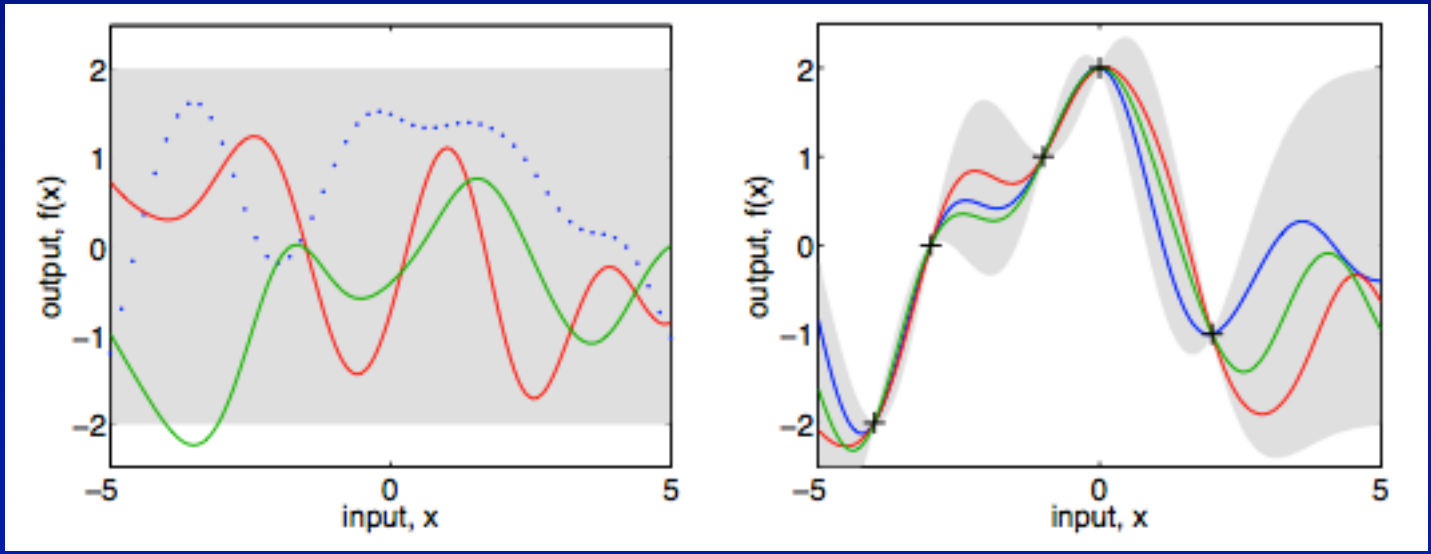
Transferencia de Tuplas

- Se tiene que definir qué tuplas transferir
- Filtro de Lazaric:
  - ¿De dónde transferir? La probabilidad de que la tarea origen genere muestras de la tarea destino (*task compliance*)
  - ¿Cuáles transferir? Muestras muy relevantes o muy alejadas (*relevance*)

Procesos Gaussianos



- Distribución Gaussiana multivariada: $\mathcal{N}(\mu, \Sigma)$
- Un Proceso Gaussiano es una generalización a un número infinito de variables $\mathcal{GP}\big(m(x), k(x, x')\big)$



Procesos Gaussianos

- Aunque parece peor trabajar con dimensionalidad infinita, lo que se calcula se hace en dimensiones finitas

$$p(\mathbf{y}, \mathbf{y}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T} + \sigma^2 \mathbf{I})$$

$$\mathbf{K}_{N+T} = \begin{bmatrix} \mathbf{K}_N & \mathbf{K}_{NT} \\ \mathbf{K}_{TN} & \mathbf{K}_T \end{bmatrix}$$

$$p(\mathbf{y}_T|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T)$$

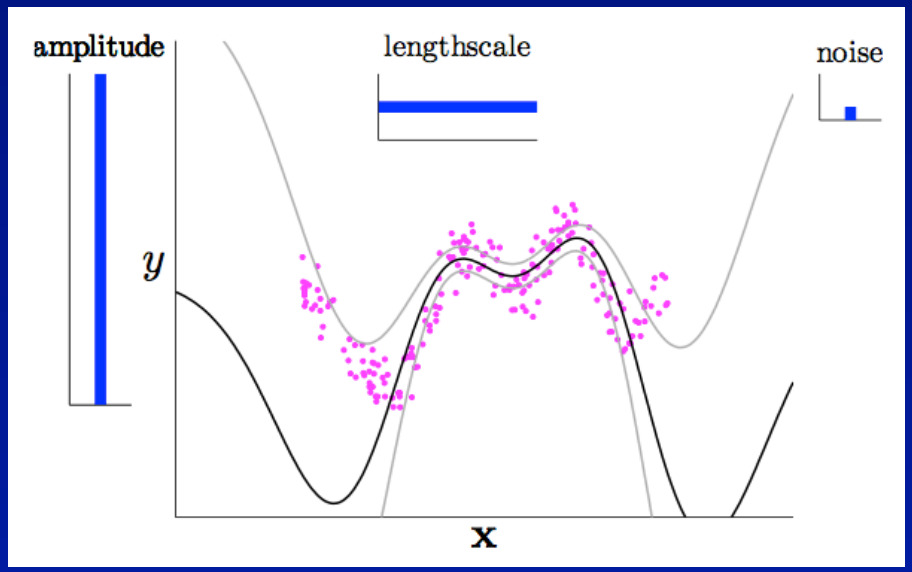
$$\boldsymbol{\mu}_T = \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1}\mathbf{y}$$

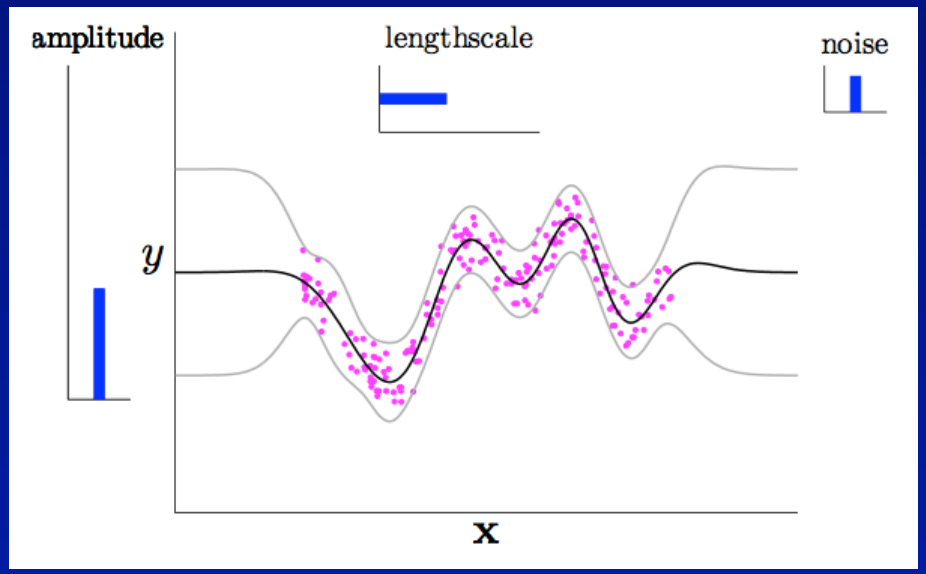
$$\boldsymbol{\Sigma}_T = \mathbf{K}_T - \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1}\mathbf{K}_{NT} + \sigma^2 \mathbf{I}$$

Procesos Gaussianos

- Kernel e Hiperparámetros

$$K(x, x') = {\sigma_0}^2 \exp\left[-\frac{1}{2}\left(\frac{x - x'}{\lambda}\right)^2\right]$$





Cálculo de Hiperparámetros

- Se minimiza el logaritmo de los datos con respecto a los hiperparámetros

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta})$$

- Se obtiene su derivada con respecto a los hiperparámetros para optimizarlos

$$\frac{\partial \mathcal{L}}{\partial \theta_i}$$

- Se puede caer en mínimos locales

PILCO

- Usa GP para modelar funciones de transición (P(s'|s,a))
- Usa funciones de base radial para representar la política
- Ciclo:
  - Dada una π obten datos
  - Con datos infiere función de transición
  - Con función de transición evalúa y mejora π

PILCO

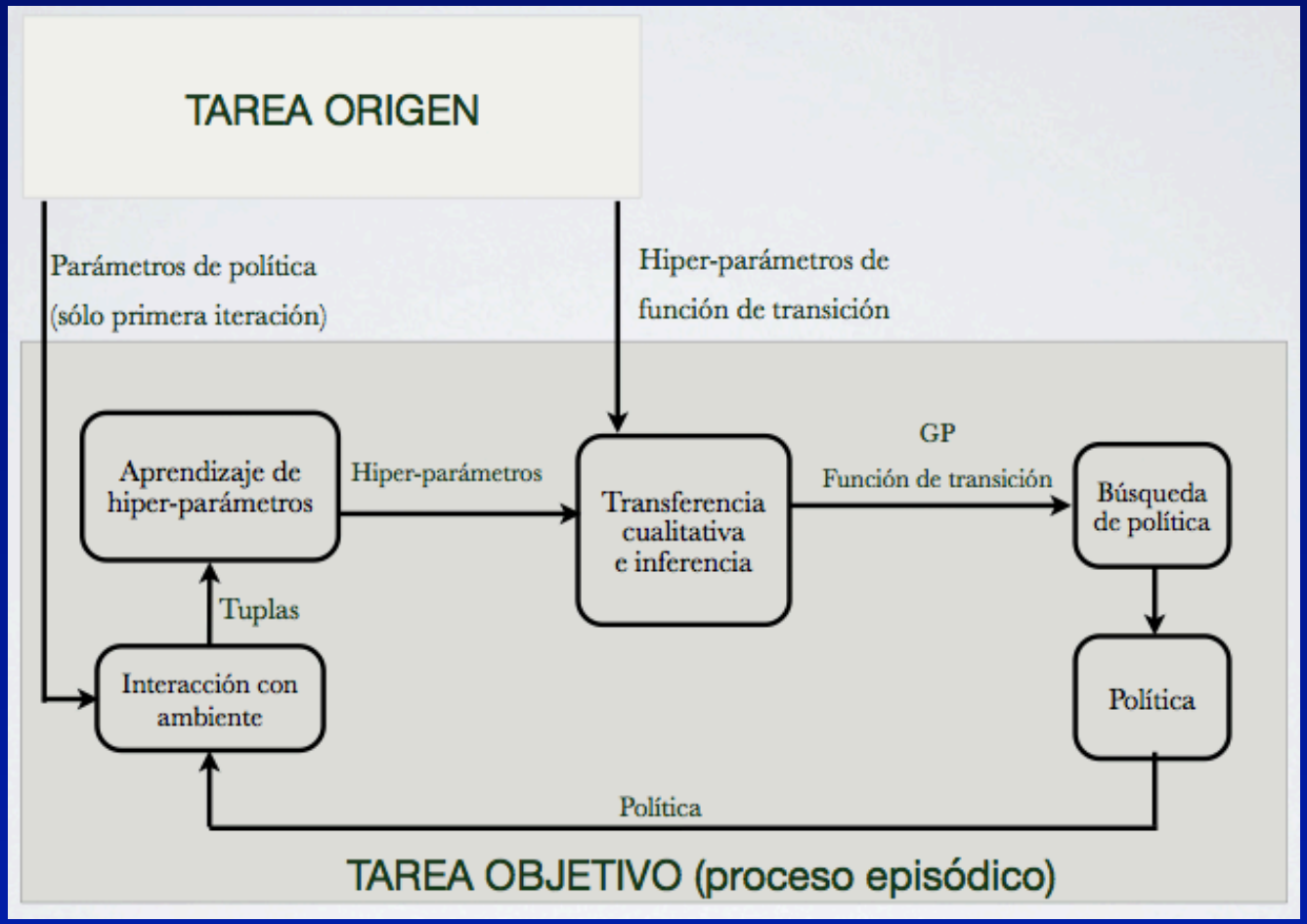
QTL y SST

En este trabajo usamos GP para:

- Modelar funciones de transición
- Transferimos hiperparámetros
- Encontramos diferencias de funciones para sintetizar tuplas

QTL

- Idea de transferir hiperparámetros (sesgo sobre distribución de posibles P(s'|s,a))



Integración de Hiperparámetros

- Usando un factor de olvido
- Actualización Bayesiana

$$\theta_0 = \theta_s$$

$$\theta_i = \gamma\theta_{i-1} + (1-\gamma)\theta_{p_i}, i > 0$$

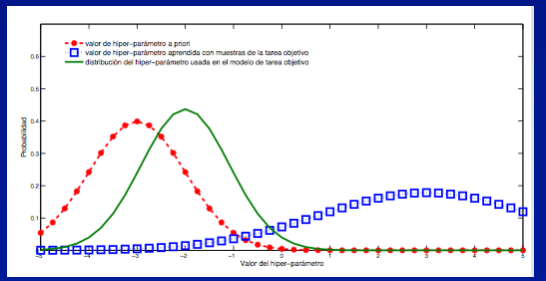
$$p(\theta_{p_k}) \sim \mathcal{N}(\mu_p, \sigma_p^2)$$

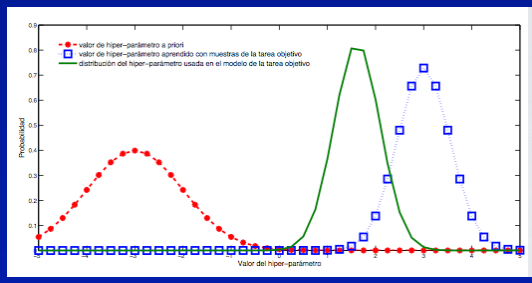
$$p(\theta|\theta_{p_k}) \sim \mathcal{N}(\mu_k, \sigma_k^2)$$

$$\sigma_k^2 = \frac{\sigma_p^2 \sigma_{k-1}^2}{\sigma_p^2 + \sigma_{k-1}^2}$$

$$\mu_k = \sigma_k^2 \left(\frac{\mu_{k-1}}{\sigma_{k-1}^2} + \frac{\mu_p}{\sigma_p^2} \right)$$

$$\sigma_{k=0}^2 = \frac{1}{n_{\text{source}}} \sigma_p^2 = \frac{1}{n_{\text{target}}}$$





SST

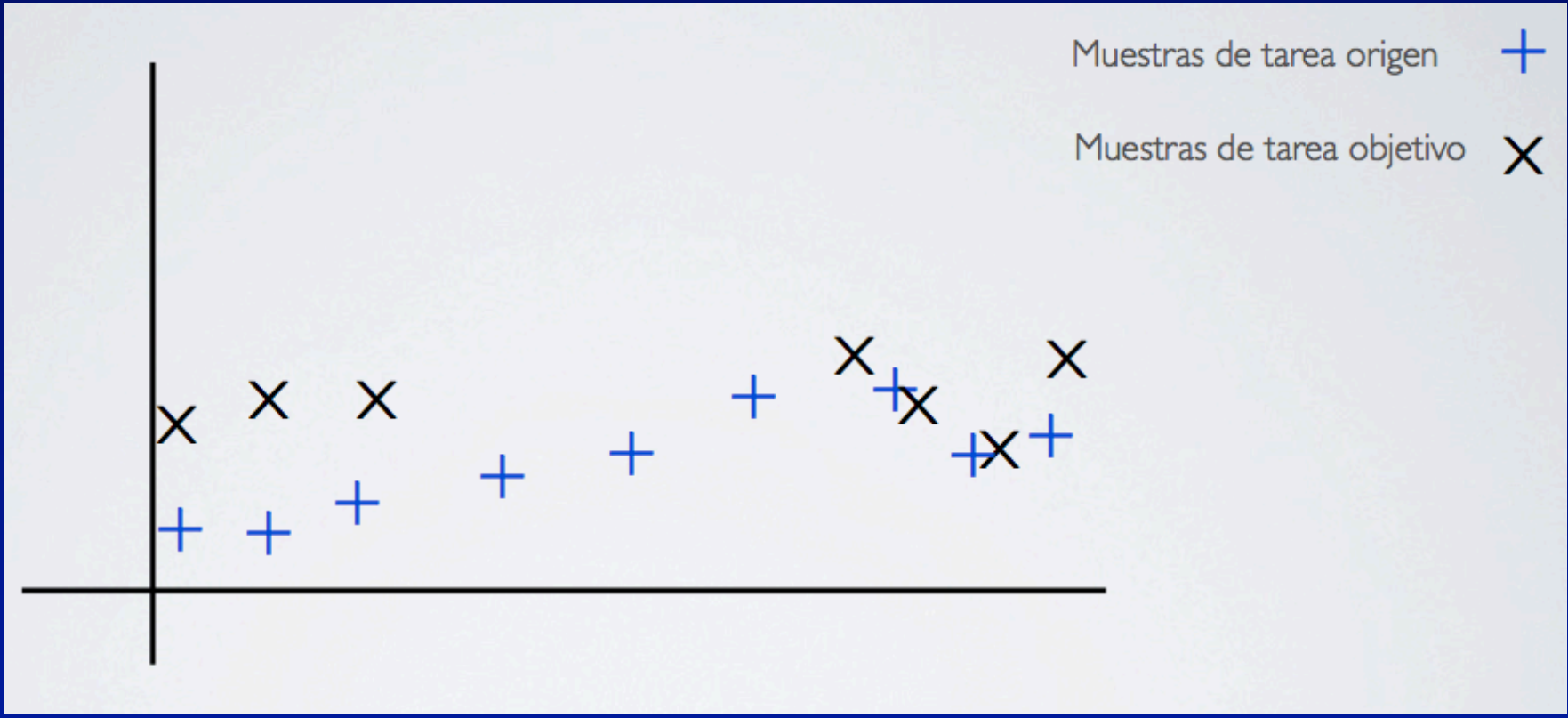
Síntesis de tuplas:

- Aprender función de transición en tarea objetivo, aprender/usar fn. en tarea original, aprender fn. de su diferencia
- En espacios poco explorados generar ejemplos usando ejemplos de tarea original transformados por función de diferencia

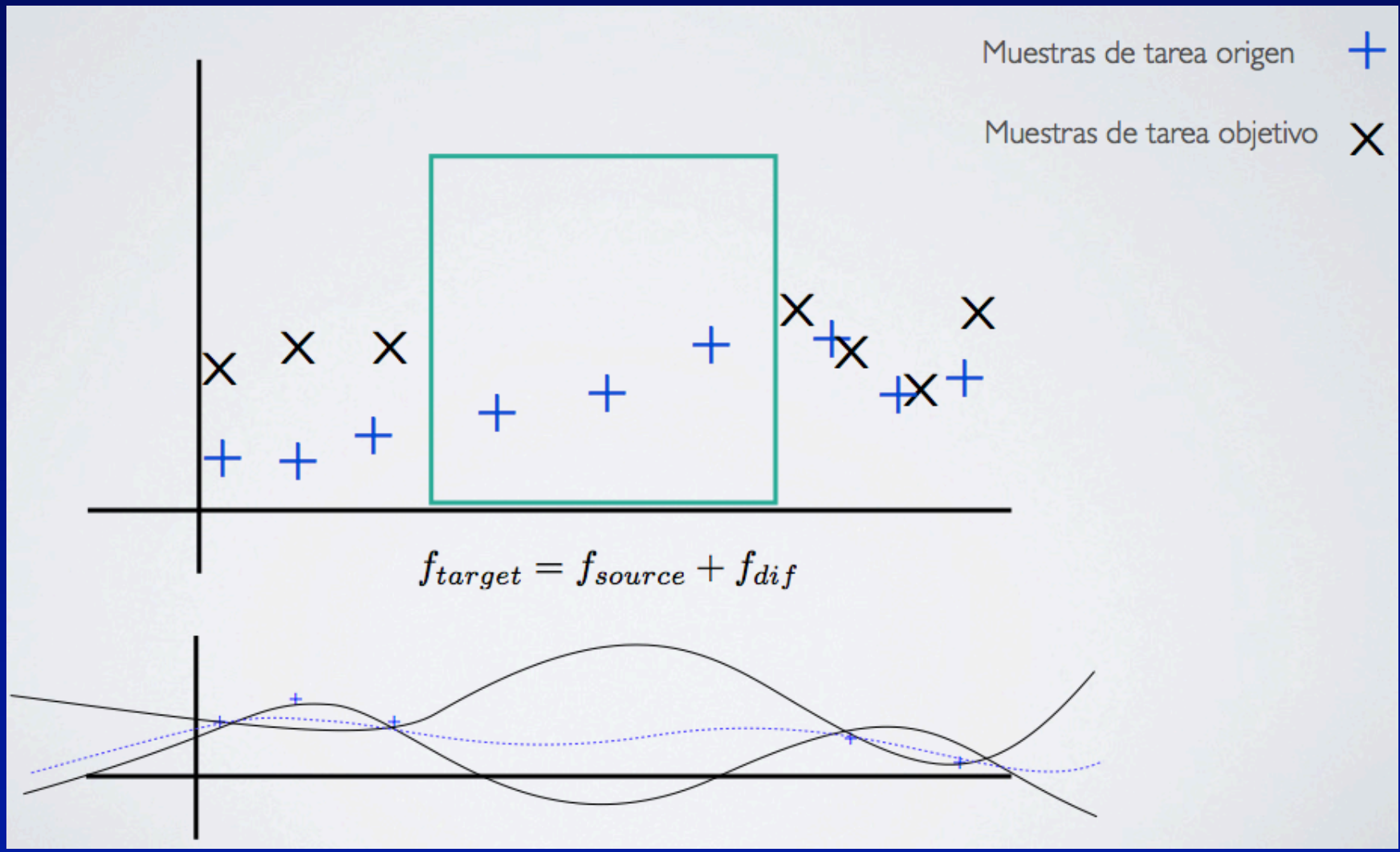
¿Dónde y Cuántas Tuplas?

- En lugares desconocidos (alejados en <*s*,*a*>): Si ya tengo ejemplos no necesito generar
- Generar hasta completar el número de ejemplos usados en la tarea original
- Mantener ese número fijo => ir reduciendo el número de ejemplos de la tarea original conforme se explora la tarea objetivo

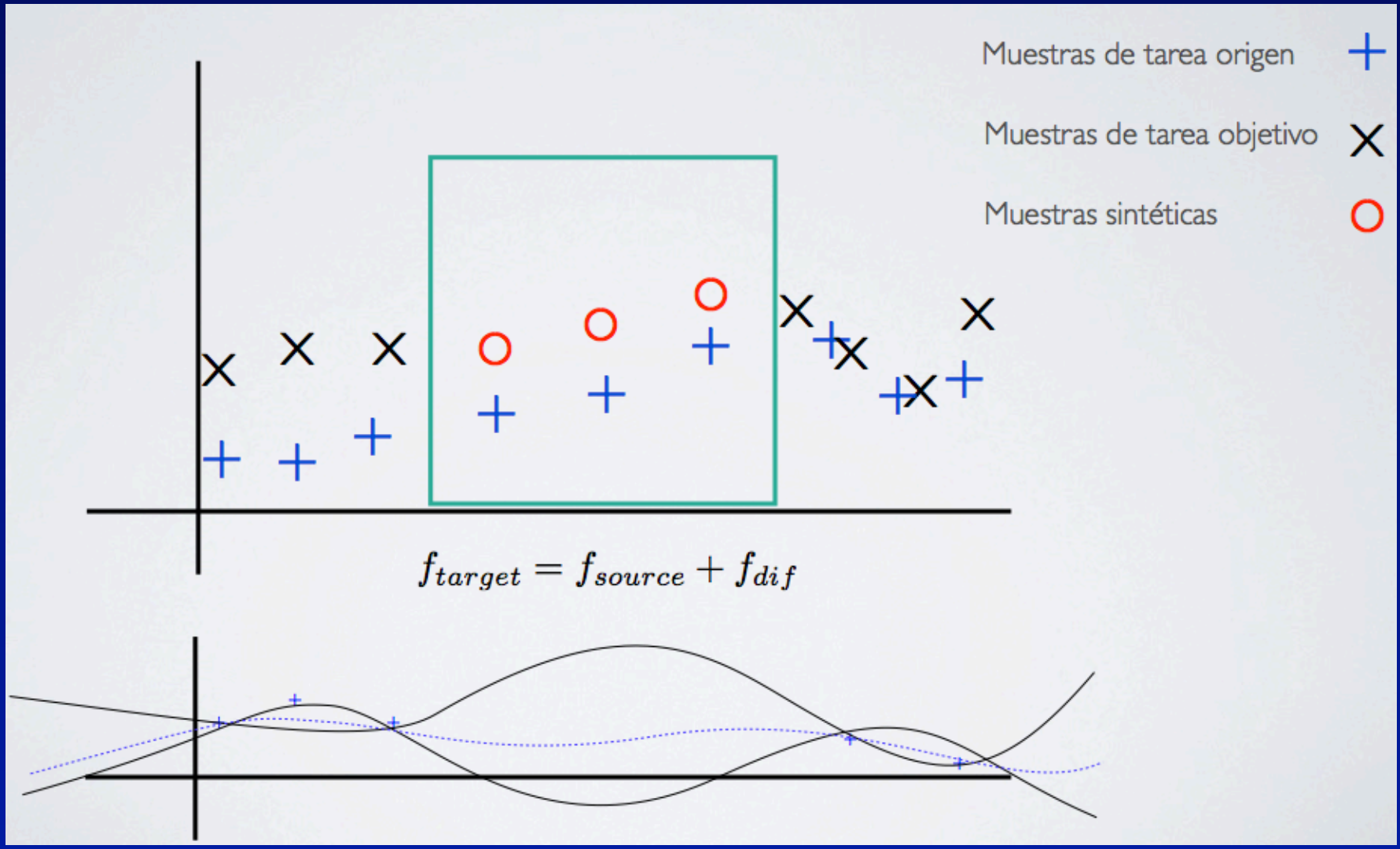
SST



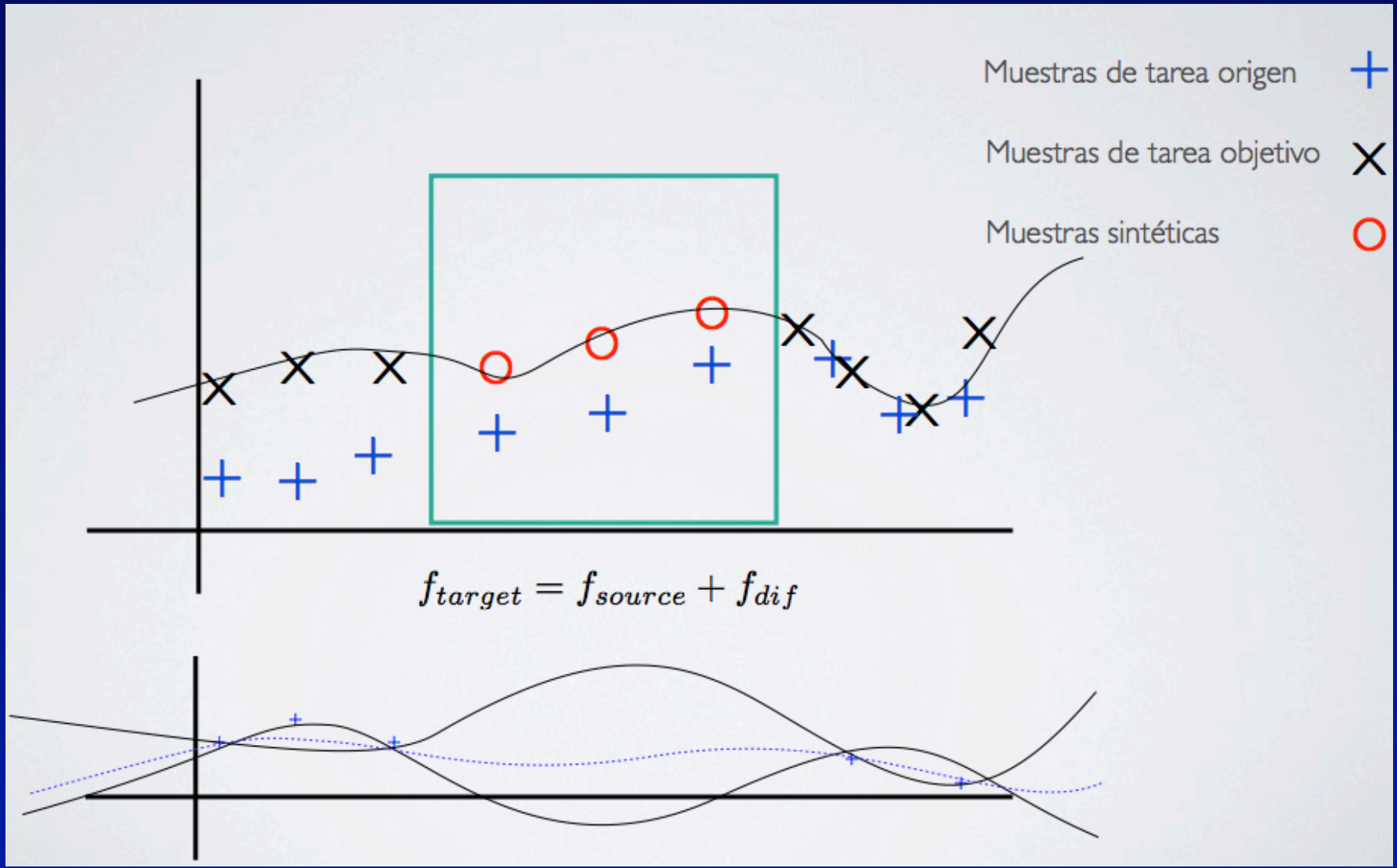
SST


Muestras de tarea origen
Muestras de tarea objetivo
$f_{target} = f_{source} + f_{dif}$

SST


Muestras de tarea origen
Muestras de tarea objetivo
Muestras sintéticas
$f_{target} = f_{source} + f_{dif}$

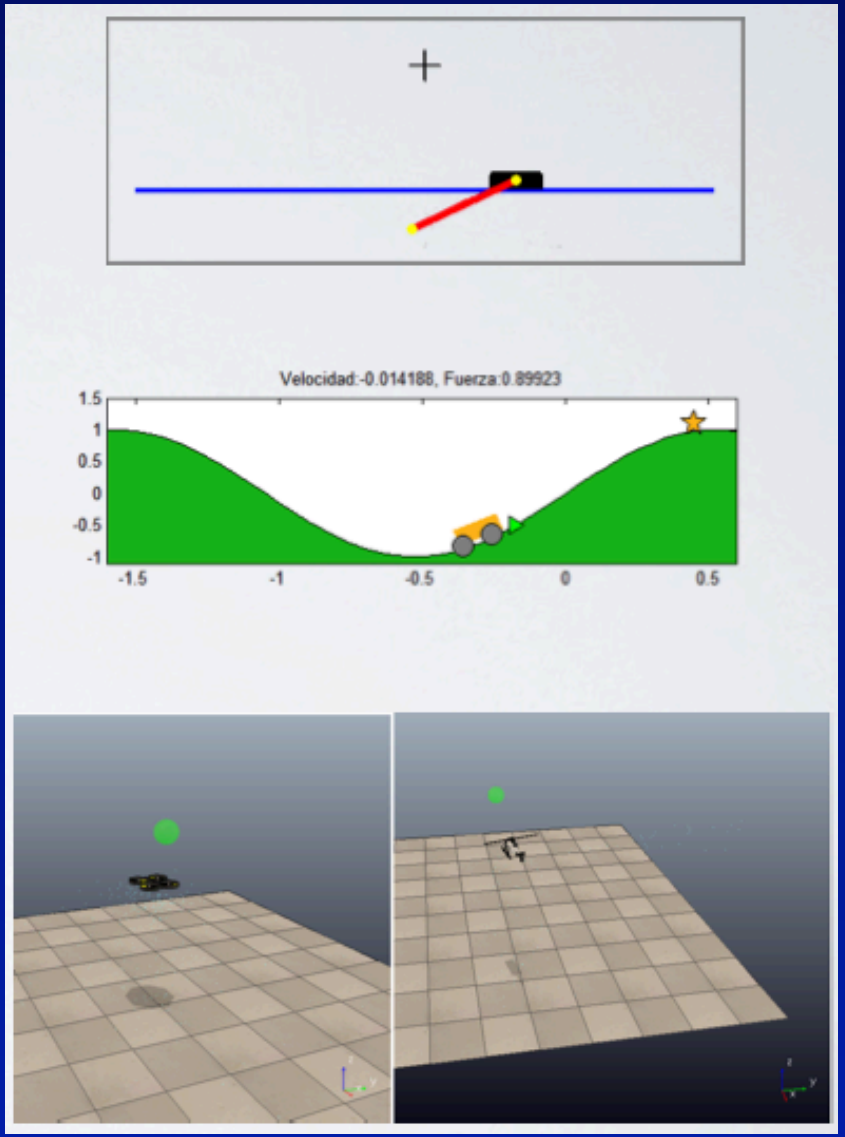
SST


Muestras de tarea origen
Muestras de tarea objetivo
Muestras sintéticas
$f_{target} = f_{source} + f_{dif}$

Experimentos

Probar en 3 dominios:

- Péndulo invertido (clásico)
- Auto en la montaña (transferencia negativa)
- De cuadróptero a helicóptero



Experimentos QTL

Probar:

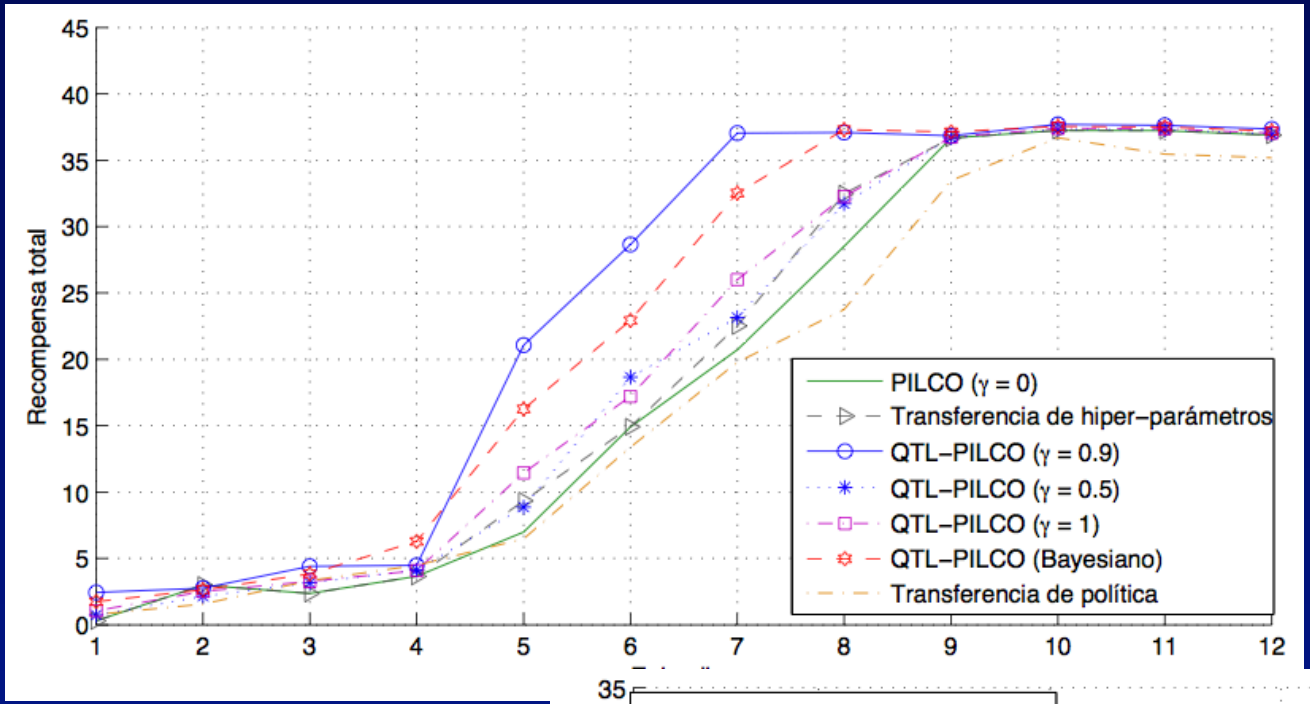
- Transferencia desde diferentes variantes
- Usar hiperparámetros de tarea original
- Usar la política de tarea original
- Diferentes valores de γ (γ=0 => PILCO)
- Enfoque Bayesiano

Experimentos QTL

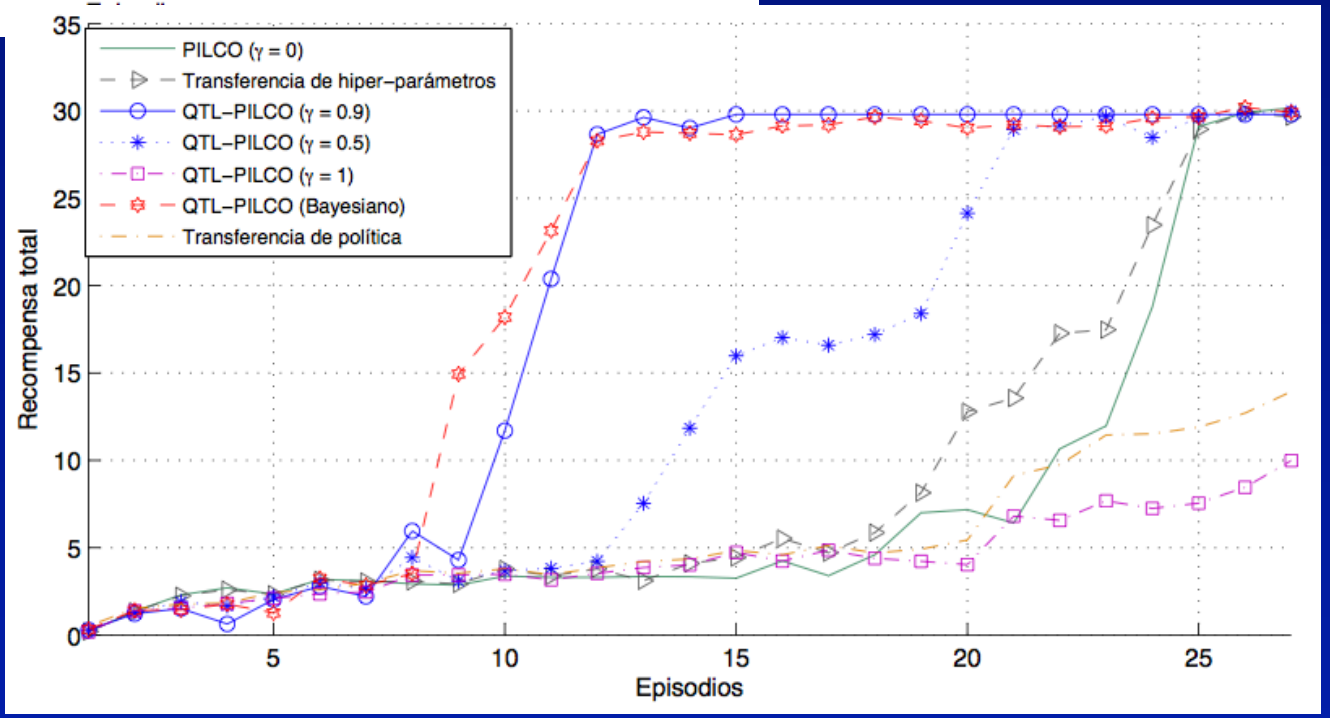
Enfoque	0.8 kg.	1.0 kg.	1.5 kg.	2.0 kg.
PILCO ($\gamma = 0$)	36.95/228.47/9	35.53/206.09/10	**33.32**/218.63/22	29.71/204.41/25
Transferencia de Hiper-parámetros	37.08/236.59/9	33.04/185.99/11	32.98/260.33/22	29.66/240.98/25
QTL $\gamma = 0.9$	37.55/**287.44**/**7**	34.97/236.20/**9**	32.84/589.57/**9**	29.79/527.65/**12**
QTL $\gamma = 0.5$	37.20/240.86/9	34.50/217.40/10	32.78/415.95/20	29.90/366.68/21
QTL $\gamma = 1$	37.06/246.41/9	34.59/212.92/10	NC/146.44/Desconocido	NC/117.63/Desconocido
QTL Bayesiano	**37.57**/272.91/8	**35.66**/**236.43**/**9**	32.68/**602.55**/10	**29.97**/**539.79**/**12**
Transferencia de política	35.77/214.35/10	32.81/156.76/11	NC/173.90/Desconocido	NC/150.64/Desconocido

QTL

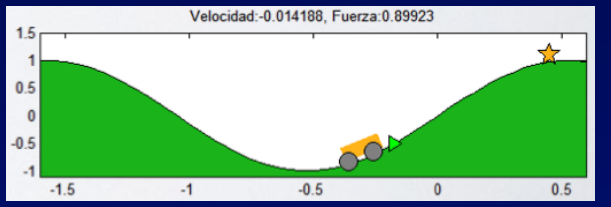
1.8 x masa original

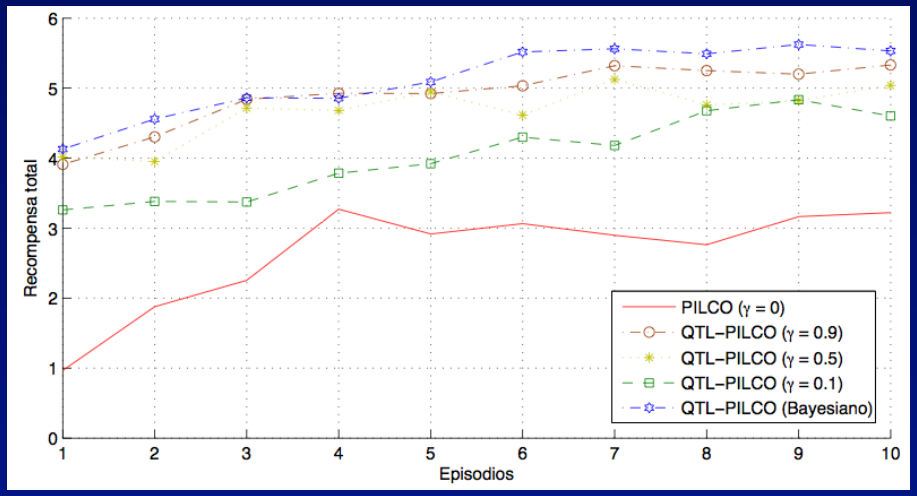


4 x masa original

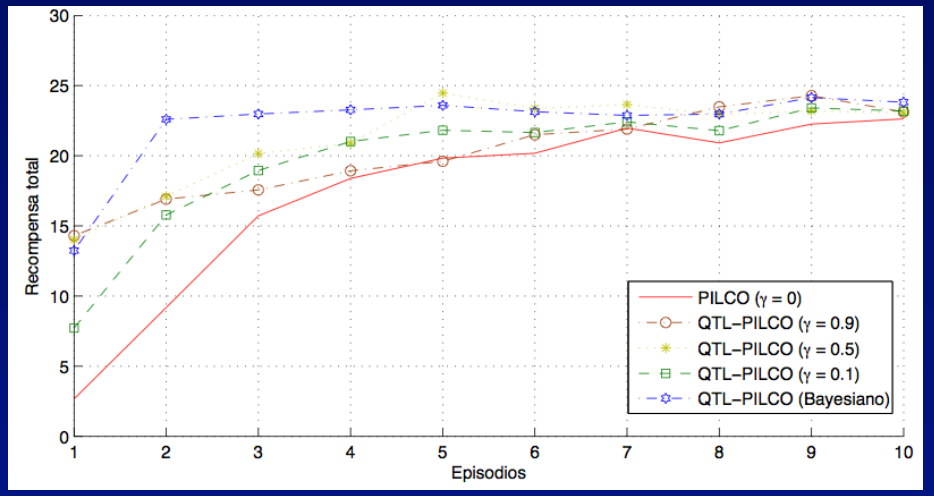


Experimentos QTL

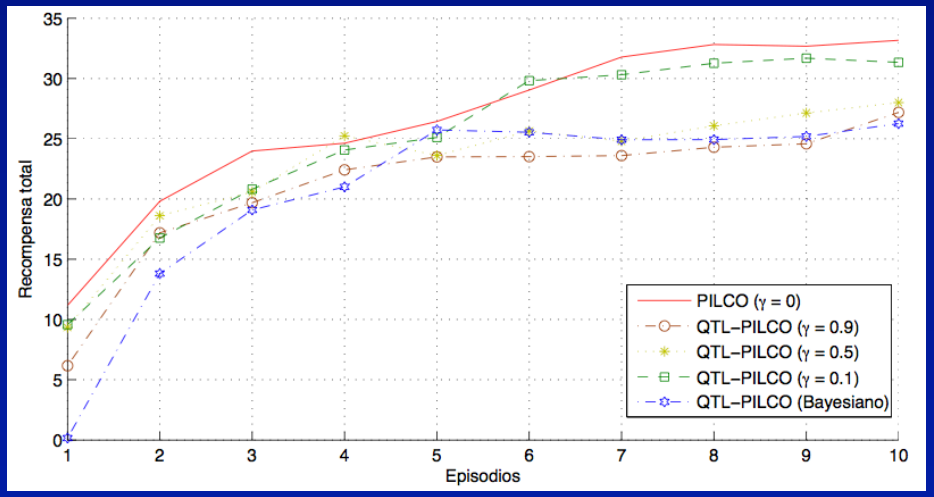




50% motor



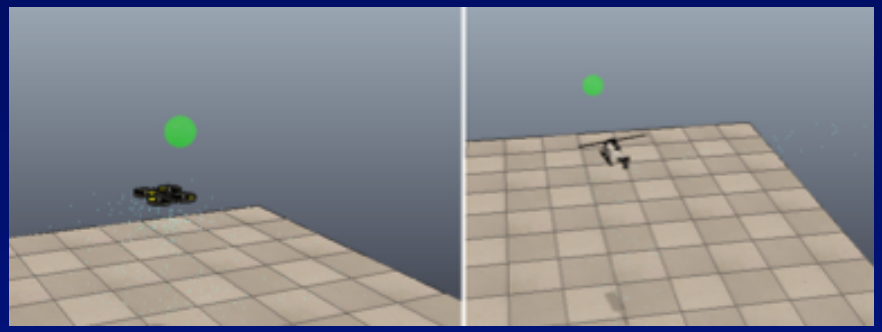
150% motor



300% motor

Potencia	*Task compliance*
50 %	0.64
150 %	0.76
300 %	0.34

Experimentos QTL



Enfoque	Tiempo de convergencia	Desempeño	Recompensa acumulada
PILCO ($\gamma = 0$)	24	**132.5**	1890.4
QTL ($\gamma = 0.1$)	21	130.61	2096.8
QTL ($\gamma = 0.5$)	22	130.95	2123.9
QTL ($\gamma = 0.9$)	21	131.53	2153.5
QTL bayesiano	**19**	131.99	**2225.4**

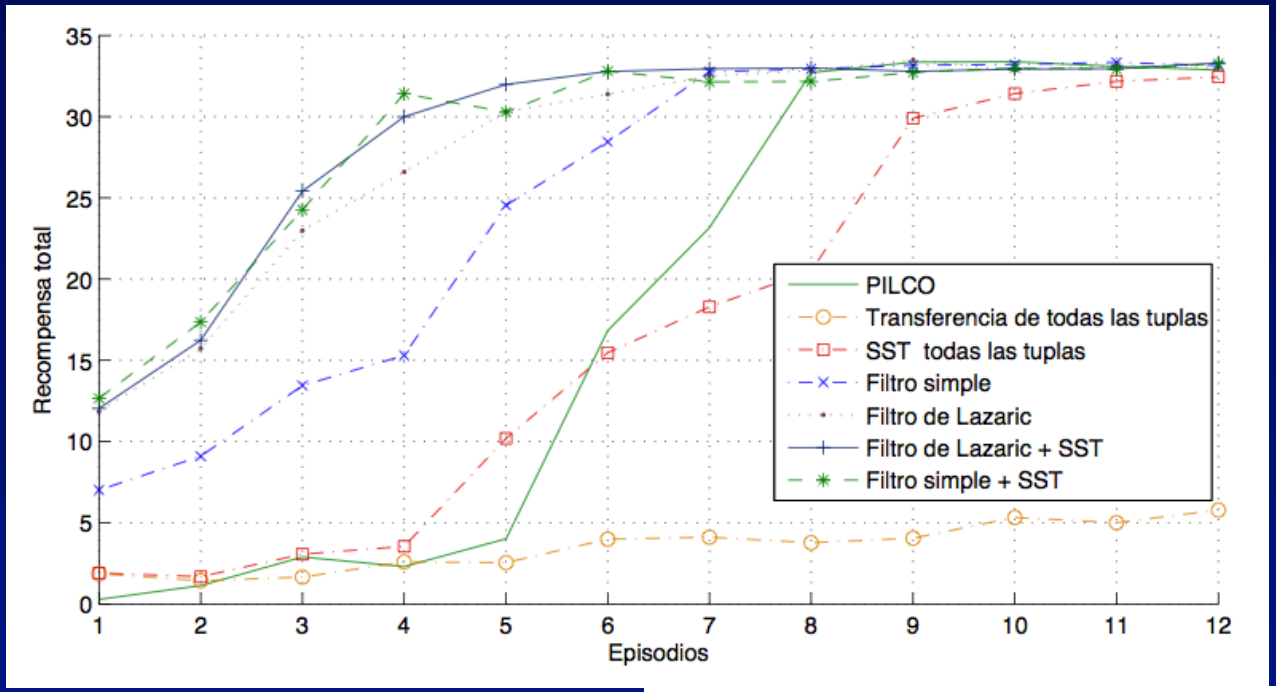
Experimentos SST

- PILCO
- Transferir todas las tuplas
- Transferir usando filtro simple
- Transferir usando filtro Lazaric
- Todas las tuplas + SST
- Filtro simple + SST
- Filtro Lazaric + SST

Experimentos SST

Algoritmo evaluado	1/2x	2x	3x	4x.
PILCO (sin transferencia)	216.00	206.10	218.63	204.42
QTL (bayesiano)	267.56	236.43	602.55	539.79
Transferencia de todas las tuplas	42.10	48.16	100.25	88.46
Filtro de Lazaric	336.70	359.73	560.69	605.71
Síntesis de tuplas	200.58	217.43	256.90	237.00
Filtro simple	296.41	342.51	541.38	564.21
Filtro de Lazaric + SST	**346.35**	**383.37**	613.58	**616.58**
Filtro simple + SST	344.99	380.94	**641.18**	613.44

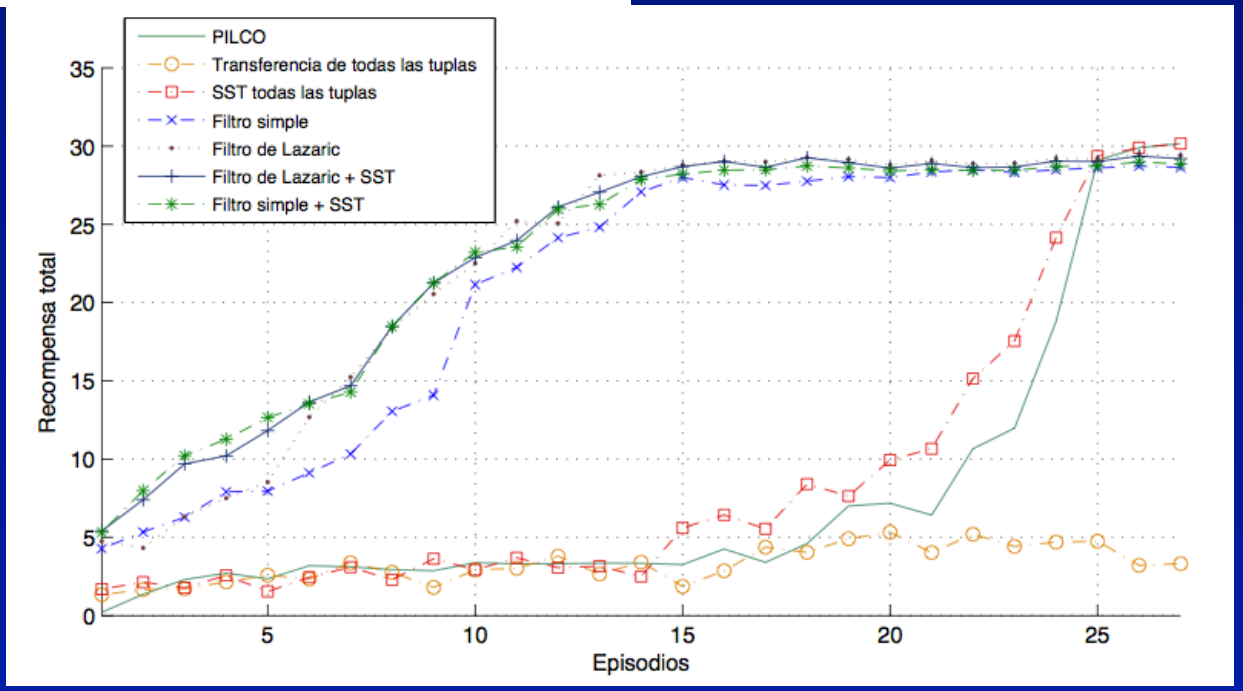
Recompensa total



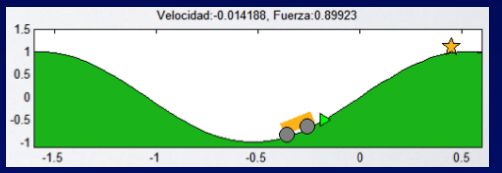
SST

2.0 Kg.

0.25 Kg.

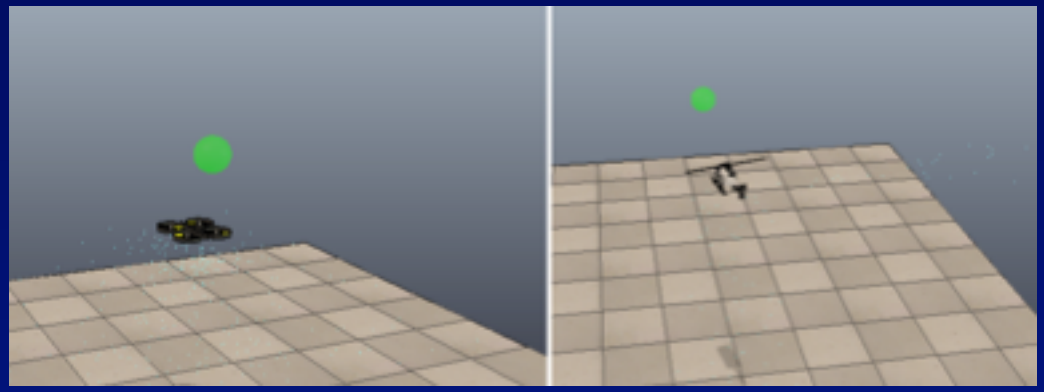


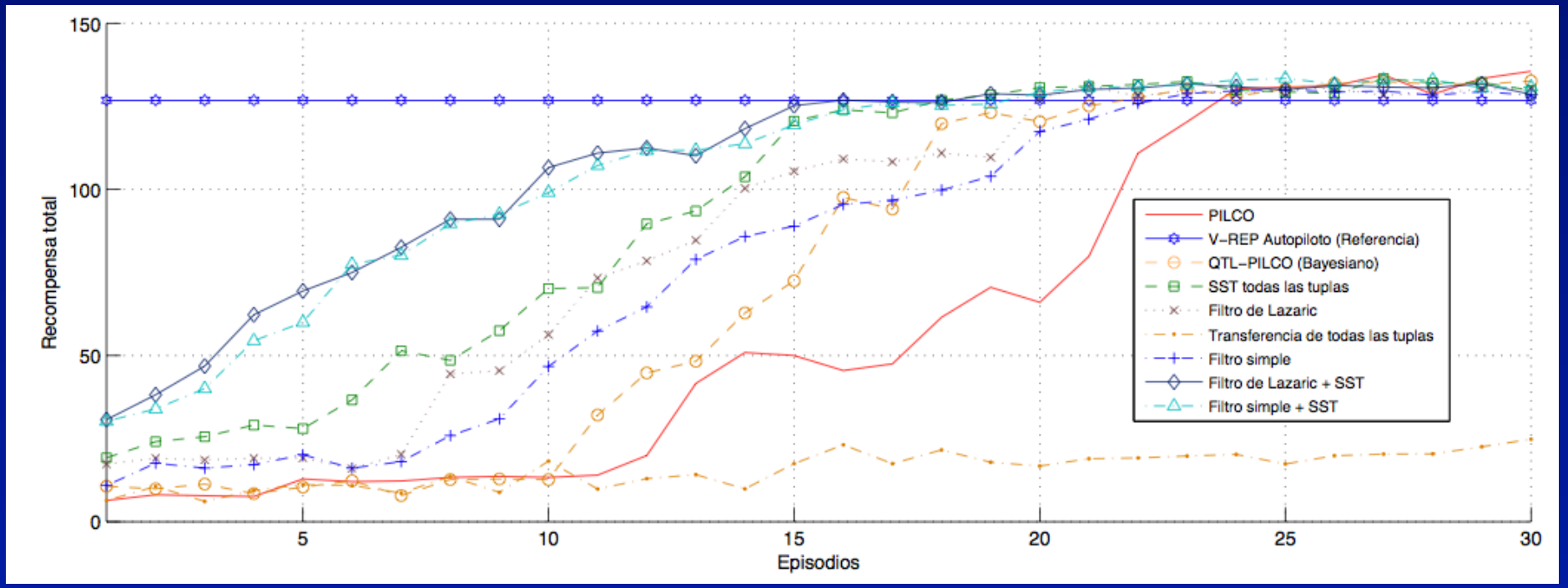
Experimentos SST



Algoritmo	50 %	150 %	300 %
PILCO (sin transferencia)	26.39	173.72	265.48
QTL (bayesiano)	51.21	222.60	206.60
Transferencia de todas las tuplas	17.56	52.90	64.05
Filtro de Lazaric	46.78	191.28	223.68
Síntesis de tuplas	50.57	232.47	252.73
Filtro simple	45.81	186.32	202.70
Filtro de Lazaric + SST	**54.70**	**289.75**	**276.39**
Filtro simple + SST	51.34	249.45	268.78

Experimentos SST





Ejemplo

Conclusiones

- RL opción para aprender en robótica
- En dominios con muchas variables (continuas) se tarda mucho y no puede re-utilizar lo aprendido
- *Transfer learning* es una opción para aprender más rápido reutilizando lo aprendido

Conclusiones

- Dos opciones para hacer TL en RL en dominios contínuos
  - Transferir/integrar parámetros (QTL)
  - Sintetizar ejemplos (SST)
- Buenos resultados en distintos dominios

Trabajo Futuro

- Dominios con diferentes variables de estado
- *Transfer* desde múltiples tareas

Gracias

emorales@inaoep.mx

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{y}^\top\mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y} + \frac{N}{2}\log(2\pi)$$

$$\frac{\partial\mathcal{L}}{\partial\theta_i} = \frac{1}{2}\operatorname{tr}\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i} - \frac{1}{2}\mathbf{y}^\top\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\theta_i}\mathbf{C}^{-1}\mathbf{y}$$