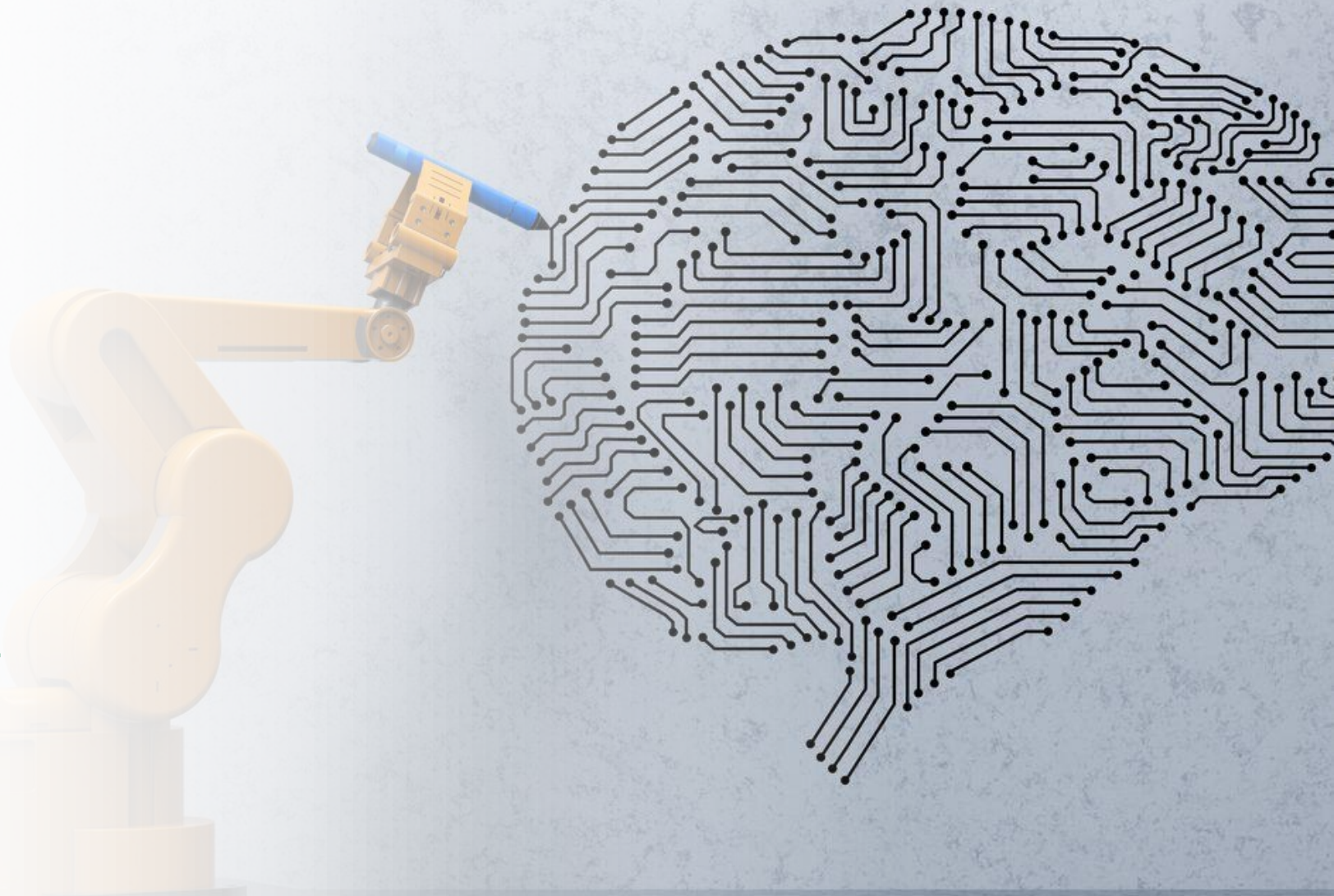
# Aprendizaje por refuerzo

Clase 15: Optimización Bayesiana



# Antes de empezar...

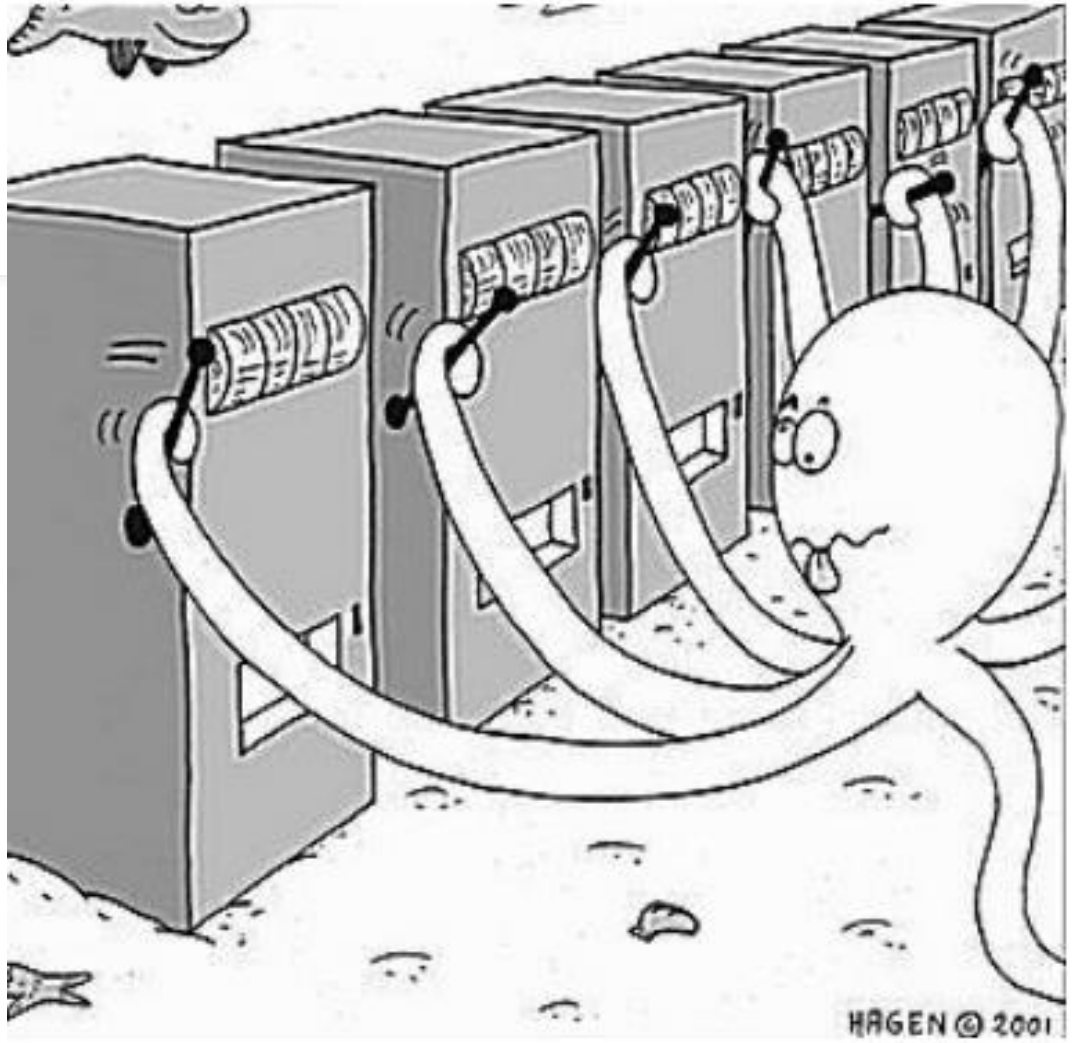
- Dudas tarea 3
- Dudas proyecto
- Dudas examen

# Para el día de hoy…

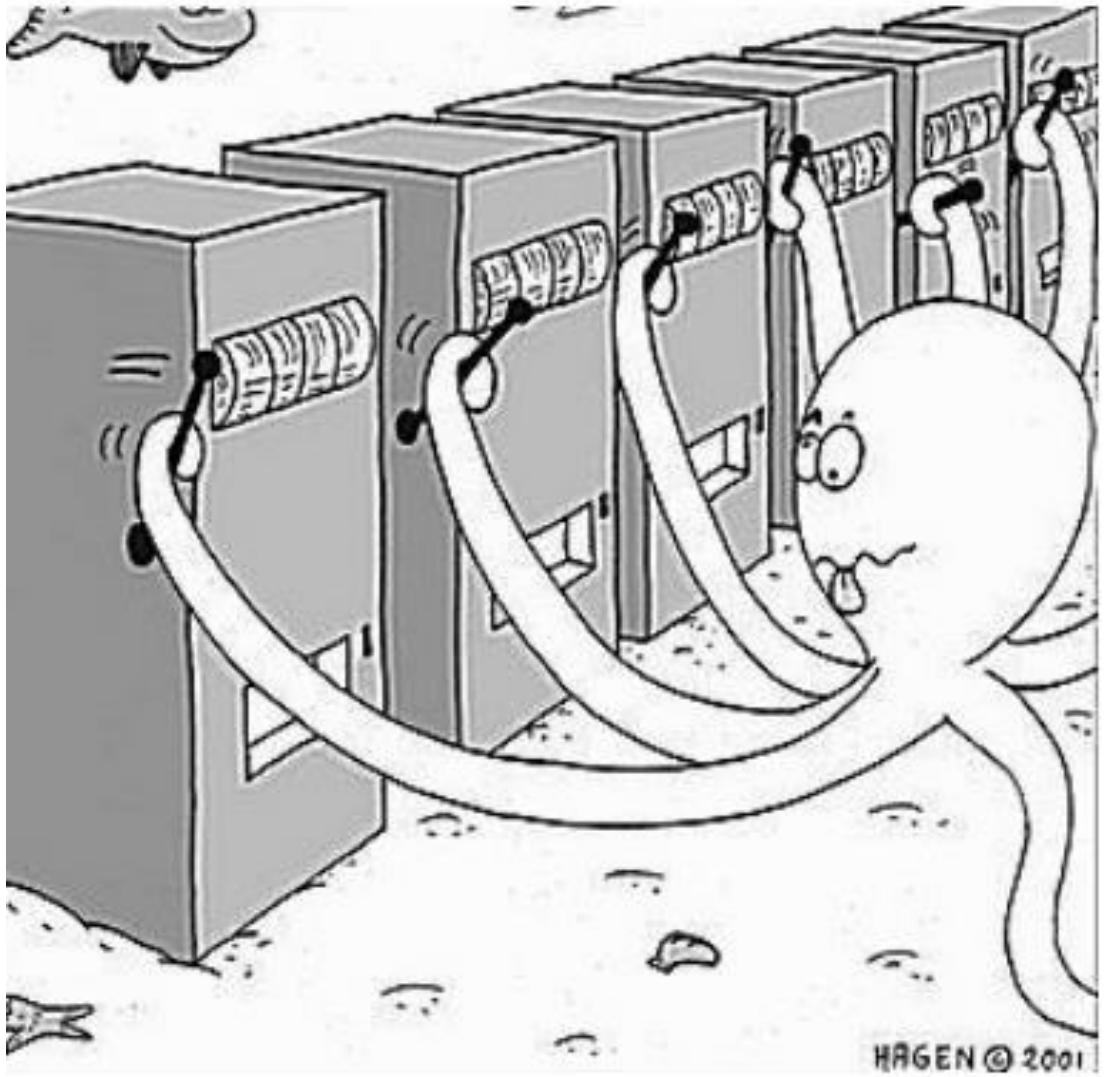
- RL para optimización
  - Optimización Bayesiana
  - Optimización combinatoria

# Recordando el bandido multi-brazo

- Es una tupla $(\mathcal{A}, \mathcal{R})$
- $\mathcal{A}$ es un conjunto de $m$ acciones
- $\mathcal{R}^a(r) = \mathbb{P}[r|a]$ es una distribución de probabilidad desconocida sobre recompensas
- En cada paso $t$ el agente selecciona una acción $a_t \in \mathcal{A}$
- El ambiente genera una recompensa $r_t \sim \mathcal{R}^{a_t}$
- El objetivo es maximizar la recompensa cumulativa $\sum_{\tau=1}^{t} r_\tau$

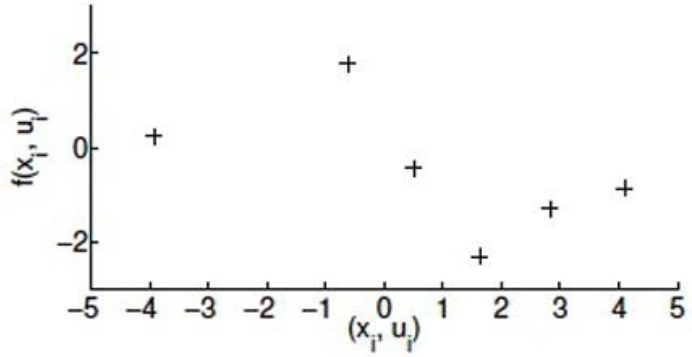


# ¿Qué pasa si tenemos un número infinito de brazos?

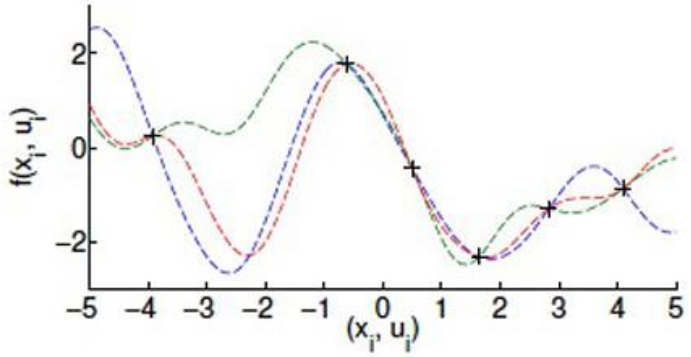


# La idea

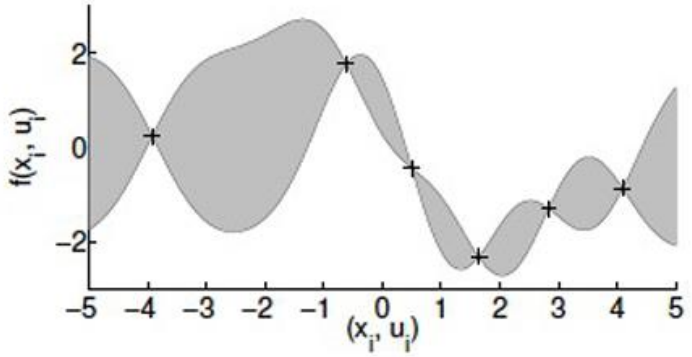
Transition data




Possible transition models




Model uncertainty



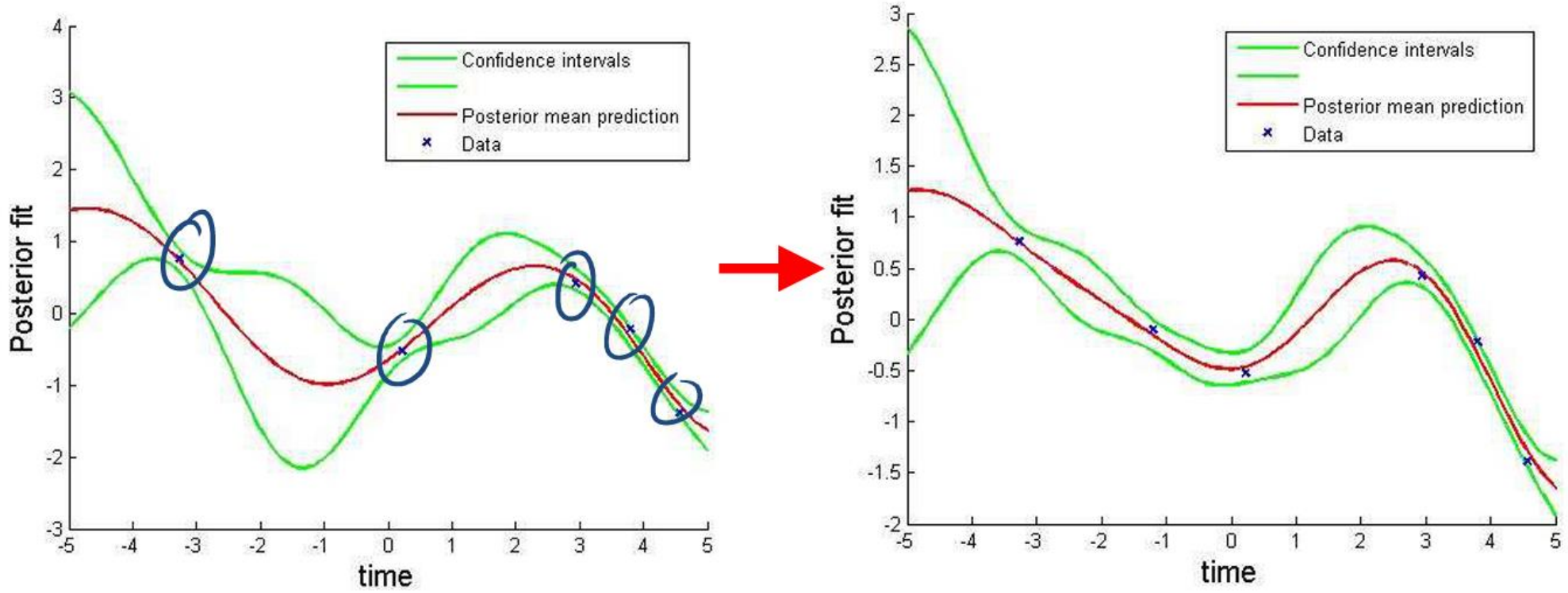
# Procesos Gaussianos

- Son distribuciones Gaussianas sobre funciones
- Dados datos $D = \{(x_i, f_i), i = 1{:}\,N\}$, donde $f_i = f(x_i)$
- Dado el conjunto de entrenamiento, queremos predecir la función $f_*$

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix}\right)$$

- Donde $K = \kappa(X, X)$ es $N \times N$, $K_* = \kappa(X, X_*)$ es $N \times N_*$ y $K_{**} = \kappa(X_*, X_*)$ es $N_* \times N_*$
- $\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2)$
- $p(f_*|X_*, X, f) = \mathcal{N}(f_*|\mu_*, \Sigma_*)$
- $\mu_* = \mu(X_*) + K_*^T K^{-1}\big(f - \mu(X)\big)$
- $\Sigma_* = K_{**} - K_*^T K^{-1} K_*$

# Aprendiza activo con GPs



# Algunas opciones para implementación

Scikitlearn (https://scikit-learn.org/)

Gpytorch (https://gpytorch.ai/ )

# Aplicaciones

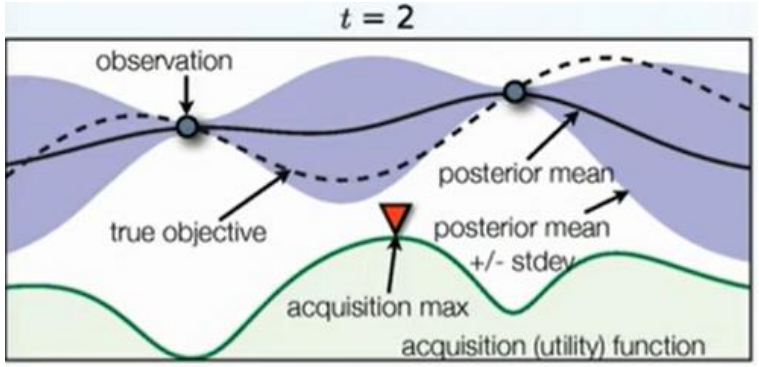
Regresión

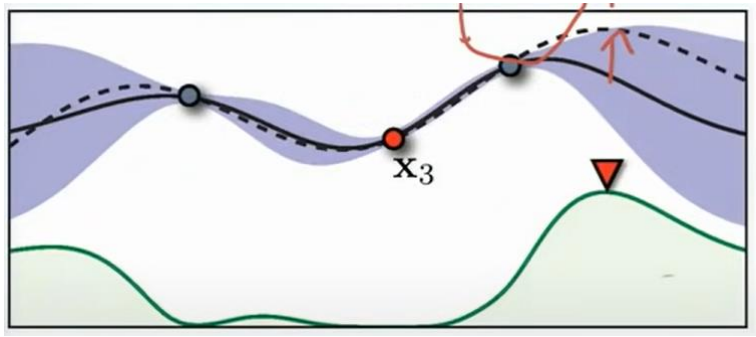
Diseño experimental

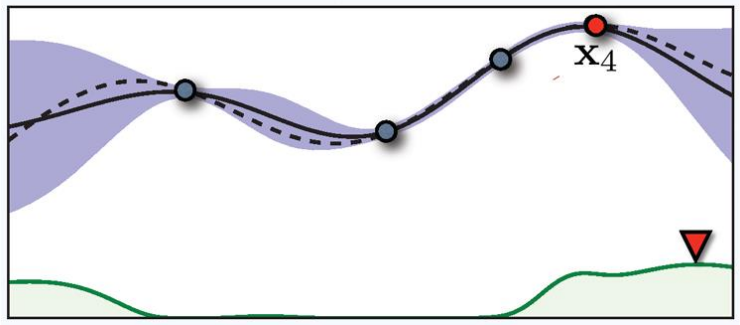
Optimización

# Optimización Bayesiana

- Desde $t = 1, \ldots$
  - $x_t = \arg\max_x u(x|D_{1:t-1})$
  - Muestrear la función objetivo $y_t = f(x_t) + \epsilon_t$
  - Aumentar los datos $D_{1:t} = \{D_{1:t-1},\ (x_t, y_t)\}$
  - Actualizar el modelo



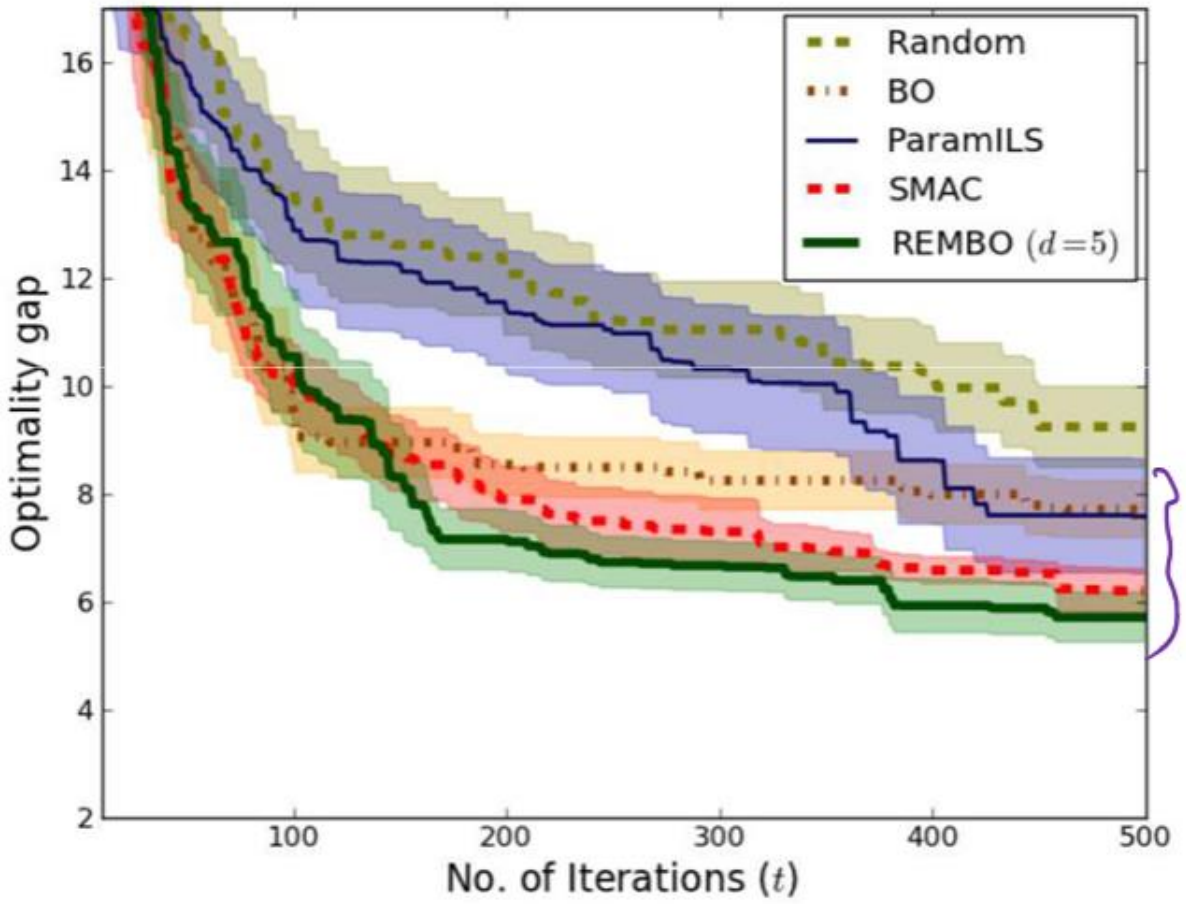



# Función de adquisición

- $\mu(x) + \kappa\sigma(x)$
- Probabilidad de mejora
- UCB
- Muestreo de Thompson

# Una aplicación especial: configuración automática de algoritmos



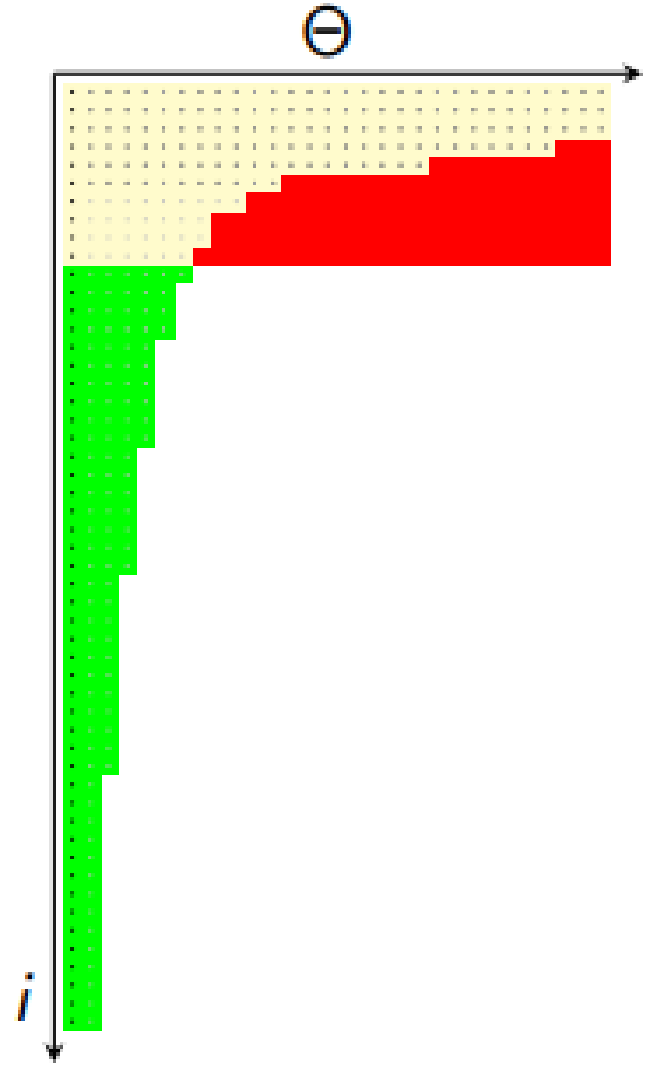

- Los algoritmos de aprendizaje suelen tener varios parámetros libres
  - Aprendizaje
  - Capas ocultas
  - Tamaño de entrada
  - Capas a usar
  - Tamaño de población
  - Cruza

# Algunos enfoques relevantes

- iRace [López-Ibañez et al. 2011]
- ParamILS [Hutter et al., 2007, 2009]
- SMAC [Hutter et al., 2011]
- EVOCA [Riff & Montero, 2013]

# Enfoque de racing

- Se inicia con un conjunto de candidatos
- Se considera un conjunto de instancias
- Se evalúa a los candidatos secuencialmente
- Se descartan a los candidatos inferiores (si existe suficiente evidencia)
- Se repite el procedimiento hasta encontrar un ganador o agotar los recursos



# Algoritmo F-Race

- Pruebas estadísticas para encontrar diferencias entre configuraciones
  - Friedman de dos vías
- Si la prueba rechaza $H_0$, se realiza la comparación por pares para la mejor configuración
- El método de Racing selecciona la mejor configuración e independiente de la forma en que las configuraciones hayan sido muestreadas

# Iterated race

- Muestrear las configuraciones de una distribución inicial
- Mientras no se cumpla condición de paro
  - Aplicar race
  - Modificar distribución de muestreo
  - Muestrear configuraciones
- http://iridia.ulb.ac.be/irace

# Otros enfoques

**ParamILS**

- Búsqueda local en el espacio de configuración
- Requiere discretización de parámetros numéricos
- http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/

**SMAC**

- Proceso de búsqueda asistido por modelos surrogados
- Muy buenos resultados para espacios con alta dimensionalidad
- http://www.cs.ubc.ca/labs/beta/Projects/SMAC/

**EVOCA**

- Utiliza un método poblacional para ajuste de parámetros
- Realiza un proceso de optimización bajo ruido en cada paso

# Problemas de optimización combinatoria

- Sea $V$ un conjunto finito y $f: 2^V \rightarrow \mathbb{R}$, un problema de optimización combinatoria se define como

$$\min_{x \in 2^V} f(x)$$

- $x^*$ es el óptimo del problema si

$$f(x^*) \leq f(x) \forall x \in 2^V$$

# Comentarios para problemas combinatorios

- Problemas de optimización donde las entradas son permutaciones
  - Problema del viajero
  - Problema de la mochila
  - Cobertura de vértices
  - Problema de ruteo de paquetes en redes/communication network routing problema
  - Muchos otros
- Normalmente NP-completos

# Resolvamos un problema para uno de mis clientes…

- Nuestro cliente tiene una bolsa que soporta un peso muy muy grande $C$ (pero no infinita)
- Hay $N$ regalos y en la bolsa quiere llevar tantos regalos como le sea posible
- Cada regalo tiene un peso asociado $w_i$ y un valor de alegría $v_i$
- Nuestro cliente quiere maximizar la alegría que traerán dichos regalos sin sobrepasar la capacidad de la bolsa

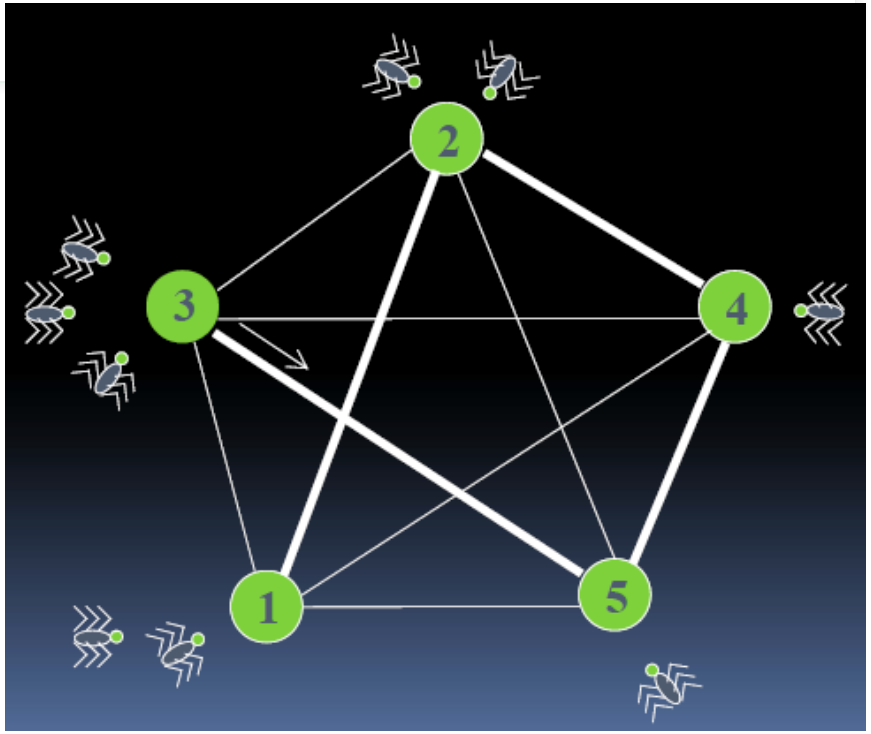
$$\max \sum_{i=1}^{N} v_i x_i$$

Tal que $\sum_{i=1}^{N} w_i x_i \leq C$

- ¡Existen $2^N$ opciones!
- Para 100 regalos existen **1.2676506e+30 opciones**

# Colonia de hormigas

- Esta basado en el comportamiento para búsqueda y provisión de alimentos realizando la exploración desde el nido
- Las hormigas dejan un rastro de feromona que puede ser detectado por el resto de la colonia
- Se trata de un algoritmo constructivo
- Particularmente útil en problemas que acepten una representación vía grafo



# Algoritmo

```
Inicializar();
for c=1 to Nro_ciclos
{
  for k=1 to Nro_ants
    ant-k construye solución k;
  Guardar la mejor solución;
  Actualizar Rastro (i.e., τij);
  Reubicar hormigas para el próximo ciclo;
}
```

La construcción se realiza paso a paso en forma probabilística considerando $\tau_{ij}$ y $\eta_{ij}$

- Se mantiene una matriz de feromona que indica la fortaleza de la conexión entre dos nodos del grafo

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t+1)$$

$$\Delta\tau_{ij}(t+1) = \sum_{k=1}^{|ants|} \Delta^k\tau_{ij}(t)$$

$$\Delta^k\tau_{ij}(t) = \frac{1}{L_k}$$

- Cada hormiga construye una solución y cada paso el siguiente forma

$$P_{ij}(k) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{h\in C}\tau_{ih}^{\alpha}\eta_{ih}^{\beta}} & j \in C \\ o & de\ lo\ contrario \end{cases}$$

**ThompsonSamplingInBayesianRL(s,b)**

Repeat

Sample $\theta_1, \ldots, \theta_k \sim \Pr(\theta)$

$Q^*_{\theta_i} \leftarrow solve(MDP_{\theta_i})\ \forall i$

$\hat{Q}(s,a) \leftarrow \frac{1}{k}\sum_{i=1}^{k} Q^*_{\theta_i}(s,a)\ \forall a$

$a^* \leftarrow \text{argmax}_a\ \hat{Q}(s,a)$

Execute $a^*$ and receive $r, s'$

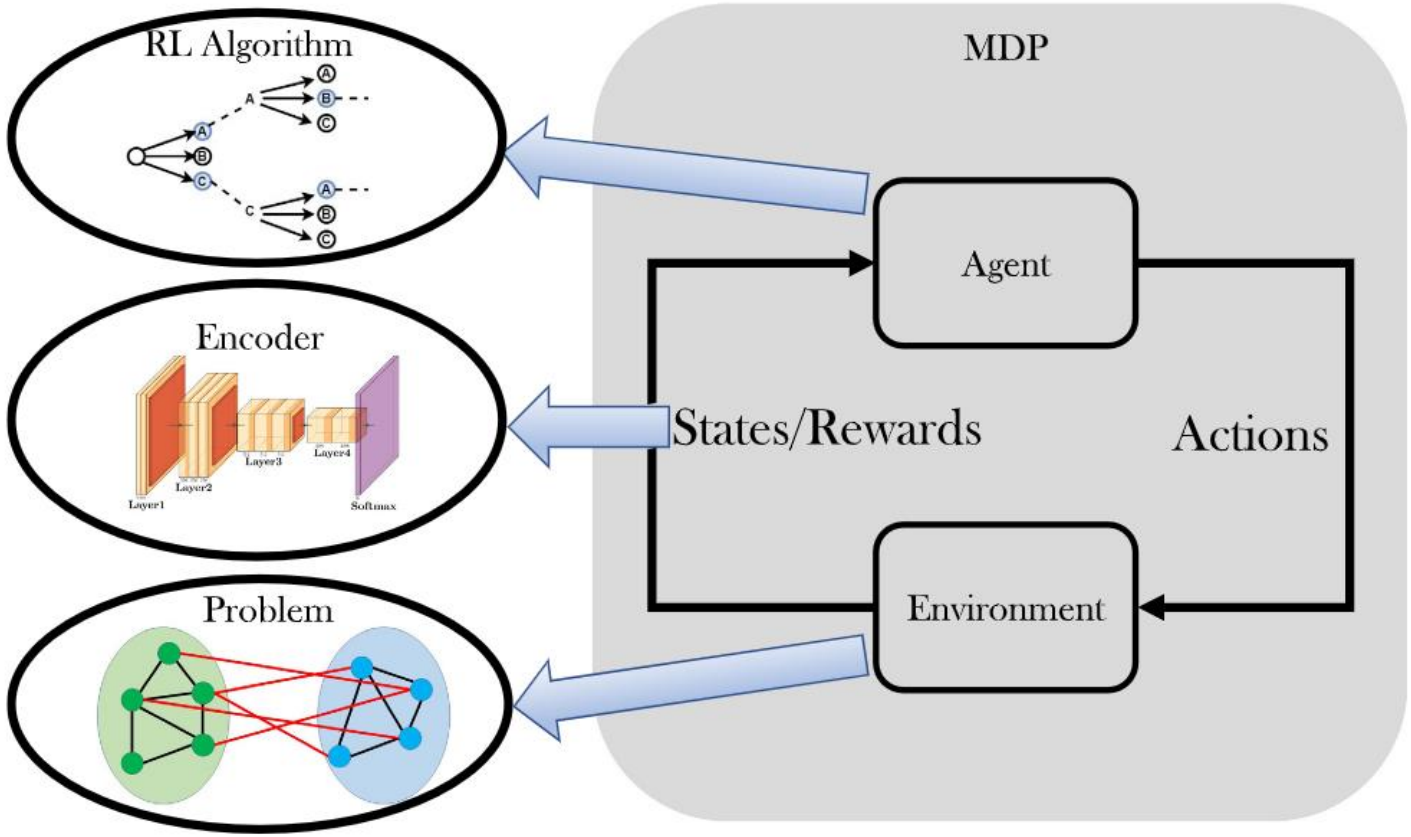
$b(\theta) \leftarrow b(\theta)\Pr(r, s'|s, a^*, \theta)$

$s \leftarrow s'$

# RL profundo para Optimización combinatoria

- Reformular el problema en términos de un MDP
- Definir una codificación de los estados a los reales (aproximación de la función Q o de la política)
- Utilizar un algoritmo de RL para aprender la codificación y la política
- Profit

# El flujo


RL Algorithm
Encoder
Layer1
Layer2
Layer3
Layer4
Softmax
Problem
MDP
Agent
States/Rewards
Actions
Environment

# Opciones para el codificador

- Redes de grafos convolucionales (GCN)
- Redes de grafos de atención (GAT)
- Redes de isomorfismos de grafos (GIN)
- Redes de estructuras a vectores (S2V)

# Algunos trabajos

| Approach | *Searching solution* | | *Training* | |
|---|---|---|---|---|
| | Joint | Constructive | Encoder | RL |
| Bello et al. (2017) | No | Yes | Pointer network | REINFORCE with baseline |
| Khalil et al. (2017) | No | Yes | S2V | DQN |
| Nazari et al. (2018) | No | Yes | Pointer network with convolutional encoder | REINFORCE (TSP) and A3C (VRP) |
| Deudon et al. (2018) | No | Yes | Pointer network with attention encoder | REINFORCE with baseline |
| Kool et al. (2019) | No | Yes | Pointer network with attention encoder | REINFORCE with baseline |
| Emami and Ranka (2018) | No | No | FF NN with Sinkhorn layer | Sinkhorn policy gradient |
| Cappart et al. (2021) | Yes | Yes | GAT/Set transformer | DQN/PPO |
| Drori et al. (2020) | Yes | Yes | GIN with an attention decoder | MCTS |
| Lu et al. (2020) | Yes | No | GAT | REINFORCE |
| Chen and Tian (2019) | Yes | No | LSTM encoder + classifier | Q-Actor–Critic |

# Comparaciones

| Algo | Article | Method | Average tour length | | |
|---|---|---|---|---|---|
| | | | $N = 20$ | $N = 50$ | $N = 100$ |
| RL | Lu et al. (2020) | REINFORCE | 4.0 | 6.0 | 8.4 |
| | Kool et al. (2019) | | 3.8 | 5.7 | 7.9 |
| | Deudon et al. (2018) | | 3.8 | 5.8 | 8.9 |
| | Deudon et al. (2018) | REINFORCE+2opt | 3.8 | 5.8 | 8.2 |
| | Bello et al. (2017) | A3C | 3.8 | 5.7 | 7.9 |
| | Emami and Ranka (2018) | Sinkhorn policy gradient | 4.6 | – | – |
| | Helsgaun (2017) | LK-H | 3.8 | 5.7 | 7.8 |
| | Perron and Furnon (2019) | OR-Tools | 3.9 | 5.8 | 8.0 |

# Para la otra vez…

- Otros temas avanzados