



# BOURBAKI

COLEGIO DE MATEMÁTICAS

# Índice

01. Introducción \_\_\_\_\_ pág. 03

    01. Procesos de decisión de Markov . pág. 03

02. Instalación de Python y un entorno de trabajo para Aprendizaje por Refuerzo

pág. 05

    01. Instalación de Miniconda [3] \_\_\_\_ pág. 05

    02. Creación de un entorno virtual para análisis de datos \_\_\_\_\_ pág. 07

    03. Jupyter Notebooks para ML en la Nube \_\_\_\_\_ pág. 10

    04. Gym \_\_\_\_\_ pág. 10

03. Aprendizaje por Refuerzo y Micro-grids

pág. 11

    01. Aprendizaje por Refuerzo \_\_\_\_\_ pág. 11

    02. Micro-grids \_\_\_\_\_ pág. 14

04. Optimización de Micro-grids	_____	pág. 17
05. Complementos del aprendizaje por re-		
fuerzo	_____	pág. 18
01. Programación dinámica	_____	pág. 18
02. Q-learning	_____	pág. 19
03. Apriximación estocástica	_____	pág. 20
06. Referencias	_____	pág. 21

# 01 Introducción

Las siguientes notas son la bitácora de un curso de 8 horas que impartimos junto a Ana Isabel Ascencio. Además de este documento los invitamos a consultar el Github del curso [en este link](#).

El curso es una invitación al aprendizaje por refuerzo y su utilización en Micro-Grids para el manejo de energía, la organización de las clases es la siguiente:

1. Introducción a los procesos de decisión de Markov (una hora)
2. Simulación de procesos de decisión de Markov (una hora).
3. Aprendizaje por Refuerzo (una hora).
4. Implementación del algoritmo de optimización de Q-learning para Micro-Grids (dos horas).
5. Algunos detalles sobre los algoritmos de optimización en RL (dos horas).

## Procesos de decisión de Markov

---

El aprendizaje por refuerzo requiere fijar con anticipación un proceso de decisión del markov el cuál se puede entender como una familia de modelos

pre-determinada con los cuales deseamos solucionar nuestro problema.

Comparándolo con el caso clásico de aprendizaje supervisado, este proceso corresponde intuitivamente con la elección de una familia de funciones, es decir con los hiperplanos para las regresiones lineales o con las redes neuronales entrenadas después de fijar una arquitectura. A continuación daremos la definición formal.

**Definition 01.1.** Un proceso de decisión de Markov en tiempo discreto  $T$  (quizás infinito cuando  $T = \infty$ ) es una tupla  $\mathbb{M}_{t \leq T} = (S, A, p, r)$  tal que:

- $S$  es llamado el conjunto de estados,
- un estado distinguido  $s_0 \in S$  al que llamaremos el estado inicial,
- $A$  es el conjunto de acciones,
- $p : S \times A \times S \rightarrow \mathbb{R}$  es una función tal que para todo  $s \in S, a \in A$   $\sum_{s' \in S} p(s, a, s') = 1$ .
- $r : S \times A \times S \rightarrow \mathbb{R}$  es la función de refuerzo.

**Example 01.1.** Supongamos que tenemos un robot que recoje pelotas de tenis y puede estar en dos lugares, la canasta donde almacena las pelotas y cualquier otro, es decir  $S = \{s, o\}$ . Además supongamos que nuestro Robot puede realizar tres acciones diferentes: cargar, recoger o buscar  $A = \{c, r, b\}$ . Supongamos que las funciones de transición recomensa son las siguientes:  $p(o, c, o) = .5$ ,  $p(o, r, o) = 1$ ,  $p(o, c, s) = .5$ ,  $p(s, b, s) = .1$ ,  $p(s, b, o) = .9$  y  $r(o, c, o) = -1$ ,  $r(o, r, o) = -3$ ,  $r(o, c, s) = -2$ ,  $r(s, b, s) = -5$ ,  $r(s, b, o) = -5$

## 02 Instalación de Python y un entorno de trabajo para Aprendizaje por Refuerzo

La forma más rápida y eficiente de instalar Python, manejar sus librerías y evitar problemas de dependencia(por actualizaciones y versiones) es mediante el uso de Miniconda que además de la instalación de Python, instala conda[1] un sistema gestor de paquetes y de entornos virtuales<sup>1</sup> [2].

Una forma alternativa es utilizar Anaconda, que al igual que Miniconda, es soportada por la misma compañía Anaconda. Ambas alternativas instalan la misma versión de Python y de conda. La razón por la que se recomienda Miniconda es porque permite instalar solamente lo que se requiere para este curso, con la opción de descargar otros paquetes cuando se requieran. Anaconda por el contrario instala por defecto, una cantidad excesiva de paquetes que raramente son utilizados, lo cual requiere demasiado espacio y tiempo.

### Instalación de Miniconda [3]

---

<sup>1</sup>Los entornos virtuales permiten controlar las versiones de software (en nuestro caso python y sus librerías) usado para análisis o aplicaciones

Descarga aquí el archivo correspondiente a tu sistema operativo, ya sea que sea que MacOS, Windows y Linux, eligiendo la última versión de Python.

### **Instalación para Windows**

---

1. Da doble clic en el archivo descargado.
2. Sigue las instrucciones que se muestren aceptando las opciones que se proponen por defecto (si es necesario se pueden cambiar después).
3. Desde el menú de Inicio de Windows abre el programa Anaconda Prompt.

### **Instalación para MacOS**

---

1. En la Terminal de tu Mac, navega hasta el directorio en donde descargaste el instalador y corre la siguiente línea: bash Miniconda3-latest-MacOSX-x86\_64.sh
2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)
3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

## Instalación para Linux

---

1. En la Terminal de Linux, navega hasta el directorio en donde descargaste el instalador y corre la siguiente línea:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

2. Sigue las instrucciones, aceptando las opciones por defecto (si es necesario se pueden cambiar después)
3. Cierra la Terminal y vuélvela a abrir, para que los cambios sean actualizados.

Para todos los sistemas operativos:

4. Para ver todos los paquetes que instalados por defecto en la distribución de Miniconda, corre la siguiente línea:

```
conda list
```

## Creación de un entorno virtual para análisis de datos

---

Un entorno virtual es un ambiente de trabajo aislado, lo que permite instalar determinadas librerías o versiones de librerías sin que afecte al resto del sistema principal o de otros ambientes.

Por defecto, Miniconda crea un entorno llamado base que estará activo cuando abramos la terminal. Este entorno contiene todos los programas enlistados cuando se utilizó el comando conda list. Aunque podríamos usar este entorno, una mejor práctica es crear un entorno nuevo. Para ello es conveniente cambiar la configuración de conda para que no abra automáticamente el entorno base.

### **Desactivar el entorno base**

---

- En Mac y Linux, se debe correr la linea que se muestra a continuación y enseguida cerrar y volver a abrir la terminal:

```
conda config --set auto_activate_base false
```

- Los usuarios de Windows deberán desactivar manualmente el entorno base utilizando:

```
conda deactivate
```

### **Creación de un entorno específico para análisis de datos mediante conda-forge**

---

El entorno que vamos a crear lo titularemos MachineLearning, en éste instalaremos las librerías que necesitaremos para nuestros análisis.

1. Crea el ambiente de trabajo mediante la siguiente línea:

```
conda create -n MachineLearning
```

2. Actívalo con la siguiente instrucción:

```
conda activate MachineLearning
```

3. Agrega el canal <sup>2</sup>

```
conda config --env --add channels conda-forge
```

Se recomienda utilizar el canal conda-forge, ya que es resultado de un esfuerzo colectivo con una gran variedad de librerías y paquetes que son cuidadosamente actualizados y donde se asegura tener versiones compatibles para macOS, Linux y Windows[4].

Puedes ver los canales instalados mediante la siguiente línea:

```
conda config --show channels
```

Cada canal tiene al menos una dirección URL asociada donde se localizan los repositorios de paquetes y librerías. Puedes ver estas direcciones utilizando:

```
conda info
```

4. Instala las librerías Pandas<sup>3</sup>, Scikit-learn<sup>4</sup>, Seaborn, Matplotlib<sup>5</sup>,

---

<sup>2</sup>Los repositorios desde donde podemos descargar las librerías de Python se llaman canales, Anconda, Inc provee por defecto el canal llamado default, sin embargo permite a cualquier usuario crear su propio canal si necesita otros paquetes que no están incluidos en el canal default, por esta razón existe una variedad de canales disponibles en la web desde donde se pueden descargar librerías de Python.

<sup>3</sup>Pandas es la principal librería de python utilizada Analisis de datos. Al instalar Pandas, por defecto se instala Numpy, sobre la cual funciona Pandas. Numpy es ampliamente usada en Ciencia de Datos porque permite el fácil manejo de matrices y sus operaciones.

<sup>4</sup>Scikit-learn es una importante librería para machine learning

<sup>5</sup>Matplotlib es las principales librerías de Python usadas para graficar y visualizar información

Plotly, Yellowbrick y Jupyter Notebooks<sup>6</sup> copiando el siguiente comando (es importante copiarlo en una sola línea, separando mediante un espacio, cada librería a ser instalada):

```
conda install pandas matplotlib notebook jupyter_contrib_nbextensions  
scikit-learn yellowbrick plotly plotly=4.12.0
```

## Jupyter Notebooks para ML en la Nube

---

Una alternativa altamente recomendable es utilizar el entorno que ofrece Google Colab ya que permite ejecutar código de Python utilizando notebooks de Jupyter a cualquier persona con una cuenta de Google, sin la necesidad de tener que instalar nada en la computadora del estudiante, y ejecutar el código directamente en los servidores alojados en la nube de Google. Google Colab es altamente compatible con el repositorio del curso que se encuentra en Github

## Gym

---

<sup>6</sup>Jupyter notebook es una aplicación que nos ayudará a hacer nuestros análisis paso a paso, crear visualizaciones e incluir comentarios, como si se tratara de nuestro cuaderno de apuntes.

## 03 Aprendizaje por Refuerzo y Micro-grids

En esta sección describiremos el objetivo final del aprendizaje por refuerzo que son las políticas, estos objetos bien entendidos corresponden con la noción de modelo matemático en Machine Learning clásico, tal y como lo sugerimos anteriormente ellos dependerán de la noción de proceso de decisión de markov.

### Aprendizaje por Refuerzo

---

El aprendizaje por refuerzo busca encontrar una política que maximice la función de refuerzo.

#### 01.1 Políticas y evaluaciones de políticas

---

**Definition 01.1.** Sea  $\mathbb{M}_{t \leq T} = (S, A, p, r)$  un proceso de decisión de markov,

- Fijo un tiempo  $t \leq T$ , diremos que  $\pi_t$  es una regla de decisión en el tiempo  $t$  si  $\pi_t : S \rightarrow A$  es una función.
- Una política es una función  $\pi : \{0, 1, 2, \dots, T\}$  tal que  $\pi(t) = \pi_t$  es una regla de decisión.

- Diremos que una política es estacionaria si  $\pi(t)$  es constante para cualquier  $t \leq T$ .

**Remark 01.1.** *En el problema que solucionaremos esta semana supondremos que estamos en busca de una política estacionaria, esto se debe a la naturaleza de nuestro problema el cual describiremos más adelante, en el cual las decisiones son independientes del tiempo.*

**Remark 01.2.** *Notemos que en el caso de no tener acciones constantes, el proceso estocástico  $s_0, s_1, \dots, s_T$  no es una cadena de markov, sin embargo si  $\pi$  es una política estacionaria, entonces*

$$(s_0, \pi(1)(s_0) = s_1, \pi(2)(s_1), \dots, s_T)$$

*sí es una Cadena de Markov.*

**Definition 01.2.** Definimos la función  $R(s, a) = \mathbb{E}_{s'} [r(s, a, s')]$ , a partir de ahora llamaremos a esta función, al igual que a  $r$  la función de refuerzo.

**Definition 01.3.** Definimos la función de estado  $V^\pi : S \rightarrow \mathbb{R}$  para un proceso de decisión de Markov  $M$  con horizonte  $T$  y una política  $\pi$  de la siguiente manera:

- Si el horizonte  $T$  es finito,

$$V^\pi(s, t) = \mathbb{E} \left[ \sum_{t \leq i \leq T} R(s_i, \pi_i(s_i)) | s_t = s, \pi \right]$$

- Si el horizonte es infinito, sea  $\gamma \in (0, 1)$

$$V^\pi(s, t) = \mathbb{E} \left[ \sum_{t \leq i} \gamma^i R(s_i, \pi_i(s_i)) | s_t = s, \pi \right]$$

La esperanza anterior es tomada respecto a las trayectorias estocásticas

$$(s_0, a_0, s_1, a_1, s_2, \dots, s_T)$$

Notemos que normalmente denotaremos  $V^\pi(s) = V^\pi(s, 0)$ .

## 01.2 Políticas óptimas y funciones de política-acción

---

**Definition 01.4.** Dado un proceso de decisión de markov  $M_t$  y un conjunto  $\Pi$  de políticas deterministas definimos la política óptima de la siguiente manera:

$$\pi^* \in \operatorname{argmax}_{\pi \in \Pi} (V^\pi(s))$$

para todo  $s \in S$ .

Hasta este momento hemos definido el escenario ideal de recompensa que podemos obtener en un proceso de decisión de markov sin embargo en términos de un problema concreto lo que se busca es la política óptima que induce aquella recompensa, para ello es necesario definir las siguientes

funciones:

**Definition 01.5.** Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con horizonte infinito. Para todo  $s, a \in S, A$ , definimos la función

$$Q^*(s, a) = \mathbb{E}[r(s, \pi(s))] + \gamma \sum_{s' \in S} p(s, \pi(s), s') V^{\pi^*}(s')$$

**Lemma 01.3.** Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con horizonte infinito, entonces para todo  $s \in S$ ,

$$V^{\pi^*} = \max_{a \in A} (Q^*(s, a))$$

$$\text{Además } \pi^*(s) = \arg\max_{a \in A} (Q^*(s, a))$$

**Remark 01.4.** Lo anterior significa que si logramos encontrar la función de estado de la política óptima entonces también conoceremos cuál es la política óptima para resolverlo.

Hasta el momento no hablaremos de algoritmos para encontrar el valor de la política óptima, incluso cuando contamos tanto con las probabilidades de transición como con la función de recompensa, en general el aprendizaje por refuerzo puede ser complicado pues no siempre contaremos con esta información.

En esta sección definiremos una versión simplificada de un Proceso de Decisión de Markov asociado a un Micro-Grid. Vale la pena mencionar que en el caso de las aplicaciones concretas, estos procesos involucran ecuaciones considerablemente más complicadas a las que mostraremos sin embargo en esta sección no se está sobre-simplificando el problema innecesariamente.

**Definition 02.1.** Sea una ciudad  $C$  y  $M$  un micro-grid que administra el suministro de energía eléctrica de esta ciudad utilizando energías renovables, no-renovables y una batería donde puede almacenar la energía.

- Supongamos que la ciudad requiere una cantidad de electricidad representada por una variable aleatoria  $X_t$  a través del tiempo.
- Además supongamos que  $Y_t$  es la cantidad de energía renovable producida a través del tiempo.
- Denotaremos por  $A_t$  la cantidad de energía no-renewable producida en el tiempo  $t$ .
- $A'_t$  será la cantidad de energía por almacenar en las baterías del micro-grid en el tiempo  $t$ .
- $A''_t$  será la cantidad de energía utilizada por el micro-grid proveniente de las baterías en el tiempo  $t$ .
- $S_t$  será la cantidad de energía almacenada en el tiempo  $t$ .
- $S'_t, S''_t$  son las dos cantidades que satisfacen  $S'_t, S''_t \geq 0$  y además

$$Y_t + A_t + A_t'' - X_t - A_t' = S_t' - S_t''$$

A estas dos cantidades se les llamará el exceso y la deficiencia de energía respectivamente.

Utilizando la definición anterior de Micro-Grid definimos el siguiente Proceso de Decisión de Markov:

**Definition 02.2.** Utilizando las variables anteriores definimos el siguiente proceso de decisión de Markov:

1. El conjunto de los estados corresponde a las posibles tripletas:

$$(S_t, S_t', S_t'')$$

2. El conjunto de las acciones corresponde a las posibles tripletas:

$$(A_t, A_t', A_t'')$$

3. La función de trancisión dependen de la distribución de las variables aleatorias  $X_t, Y_t$ .

4. La función de recompensa depende de las cantidades  $S_t', S_t''$  de acuerdo a los costoso que puede ser no abastecer a la ciudad o generar un exceso de energía.

## Ø4 Optimización de Micro-grids

En la última semana del curso simulamos un proceso de decisión de markov utilizando la biblioteca Gym tal y como lo describimos anteriormente utilizando datos reales de una serie de tiempo provenientes de un microgrid que aproxima las distribuciones de las variables  $X_t, Y_t$ . A partir de ahí gracias al algoritmo de Q-learning optimizamos la función de valor para buscar una política  $\pi$ .

## 05 Complementos del aprendizaje por refuerzo

En esta sección hablaremos de algunos algoritmos relacionados con el aprendizaje por refuerzo que mejoran y explican los resultados hasta este momento.

### Programación dinámica

---

**Theorem 01.1.** (*Ecuaciones de Bellman*) Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con  $S, A$  finitos, sin estados absorventes y  $\pi$  una política, entonces para todo estado  $s \in S$ :

$$V^\pi(s) = \mathbb{E}[r(s, \pi(s))] + \gamma \sum_{s' \in S} p(s, \pi(s), s') V^\pi(s')$$

**Remark 01.2.** Si denotamos  $V = (V^\pi(s))_{s \in S}$ ,  $R = (\mathbb{E}[r(s, \pi(s))])_{s \in S}$ ,  $P = (p(s, \pi(s), s'))_{s, s' \in S}$ .

Entonces la conclusión de la proposición anterior se puede escribir como  $V = R + \gamma P V$ .

En el caso de horizonte finito cuando la política, la recompensa y las probabilidades de transición sean conocidas entonces gracias al teorema anterior es fácil conocer los valores de la función de estado.

**Corollary 01.3.** *Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con horizonte finito y  $\pi$  una política, entonces el sistema de ecuaciones para la incógnita  $V$ ,  $V = R + \gamma PV$  admite una solución única dada por*

$$V_0 = (I - \gamma P)^{-1} R$$

**Proposition 01.4.** *Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con horizonte finito entonces la política óptima satisface:*

$$V^{\pi^*}(s, t) = \max_{a \in A} \left[ \mathbb{E}[r(s, \pi(s))] + \sum_{s' \in S} p(s, \pi(s), s') V^{\pi^*}(s', t+1) \right]$$

**Proposition 01.5.** *Sea  $\mathbb{M}_T = (S, A, p, r)$  un proceso de decisión de markov con horizonte infinito entonces la política óptima satisface la ecuación:*

$$V^{\pi^*}(s) = \max_{a \in A} \left[ \mathbb{E}[r(s, \pi(s))] + \gamma \sum_{s' \in S} p(s, \pi(s), s') V^{\pi^*}(s') \right]$$

## Q-learning

---

El algoritmo de Q-learning es una generalización de los métodos de programación dinámica que nos permite encontrar una política óptima, vale la pena mencionar que este método depende de que tengamos acceso tanto a las funciones de probabilidad como a las funciones de recomenanza de

un proceso de decisión de Markov.

## Apriximación estocástica

---

Algunas veces no tendremos acceso a la probabilidad de trancisión o a la función de recompensa de un proceso de decisión. En esos casos es necesario utilizar técnicas de aproximación estocástica que generalizan el método Monte Carlo. La mayoría de estos métodos se basan en el siguiente resultado de Robins-Monroe que generaliza la ley de los grandes números:

**Definition 03.1.** Sean  $X_1, X_2, \dots$  variables aleatorias independientes e idénticamente distribuidas con  $\mathbb{E}[X_i] = \mu$  tales que  $Im(X_i) \subseteq [0, 1]$ . Si  $\eta_n$  es una familia de números reales (indexados por los naturales), definimos recursivamente a la aproximación estocástica de  $\mu$  de la siguiente manera:

$$\mu_n = (1 - \eta_n)\mu_{n-1} + \eta_n x_n$$

Donde  $\eta_1 = x_1$ .

**Exercise 03.1.** Demostrar que si  $\eta_n = \frac{1}{n}$  entonces  $\mu_n = \frac{x_1 + \dots + x_n}{n}$ .

**Theorem 03.2.** Supongamos que  $(\eta_n)_{n \in \mathbb{N}} \subseteq \mathbb{R}$  es una sucesión tal que  $\sum_{i \geq 0} \eta_i = \infty$  y  $\sum_{i \geq 0} \eta_i^2 < \infty$ , entonces

$$\lim_{i \rightarrow \infty} \mathbb{P}(\mu_i = \mu) = 1$$

## 06 Referencias

- [1] «Conda — conda 4.8.3.post5+125413ca documentation». <https://docs.conda.io/projects/conda/en/latest/> (accedido mar. 26, 2020).
- [2] Anaconda is Bloated - Set up a Lean, Robust Data Science Environment with Miniconda and Conda-Forge.
- [3] «Miniconda — Conda documentation». <https://docs.conda.io/en/latest/miniconda.html> (accedido mar. 26, 2020).
- [4] «A brief introduction — conda-forge 2019.01 documentation». <https://conda-forge.org/docs/user/introduction.html> (accedido mar. 26, 2020).



[escuela-bourbaki.com](http://escuela-bourbaki.com)