

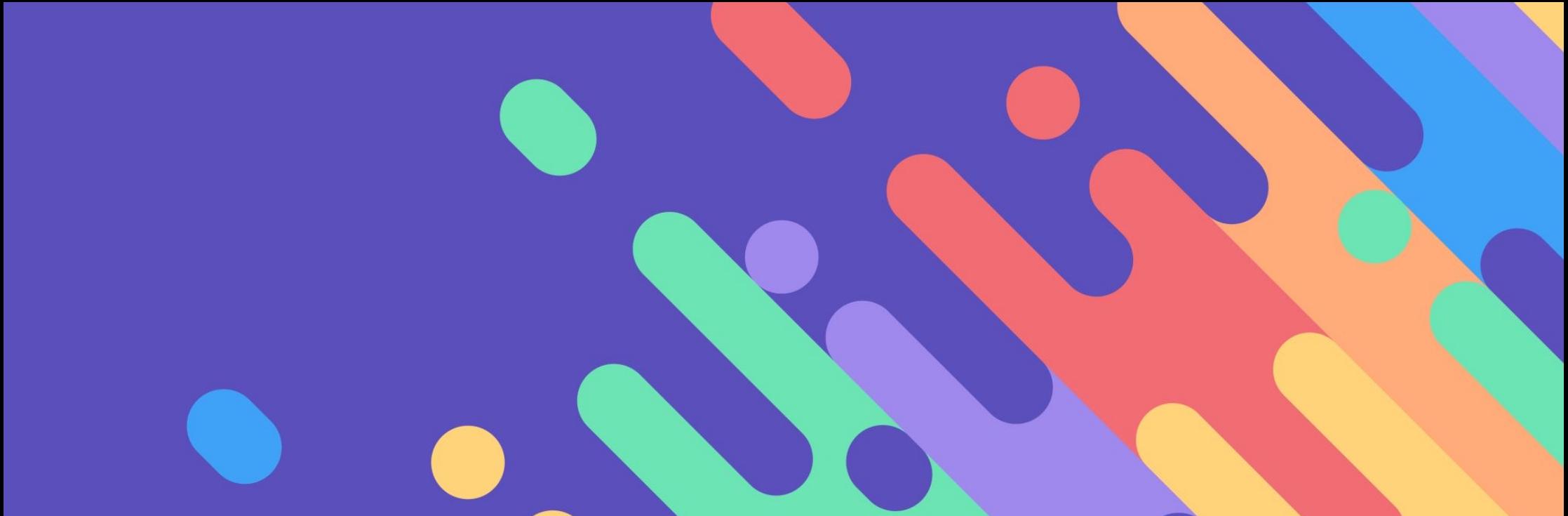
# INTRODUCCIÓN A APRENDIZAJE AUTOMÁTICO



Inteligencia Artificial

CEIA - FIUBA

Dr. Ing. Facundo Adrián  
Lucianna



---

LO QUE VIMOS LA CLASE ANTERIOR...

# ALGORITMOS DE BÚSQUEDA LOCAL

Los algoritmos de búsqueda se diseñan para explorar sistemáticamente espacios de búsqueda. Esta forma sistemática se alcanza manteniendo uno o más caminos en memoria y registrando qué alternativas se han explorado en cada punto a lo largo del camino y cuáles no.

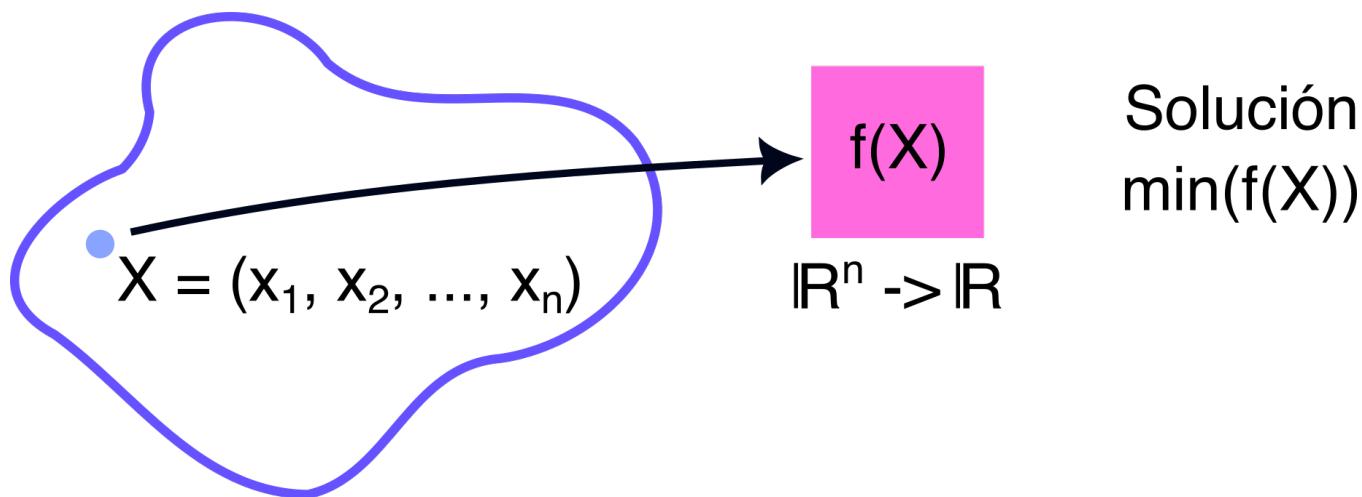
Cuando se encuentra un objetivo, **el camino** a ese objetivo también constituye una **solución** al problema.

Pero hay problemas en donde **no** nos importa el camino, sino que importa la configuración final. Por ejemplo, un algoritmo que resuelva Sudokus.

3	7	4	5	6	1	9	2	8
1	8	5	4	2	9	7	6	3
9	6	2	3	7	8	4	1	5
8	2	7	6	1	3	5	4	9
6	4	9	2	5	7	8	3	1
5	3	1	9	8	4	6	7	2
4	9	6	8	3	2	1	5	7
2	1	8	7	4	5	3	9	6
7	5	3	1	9	6	2	8	4

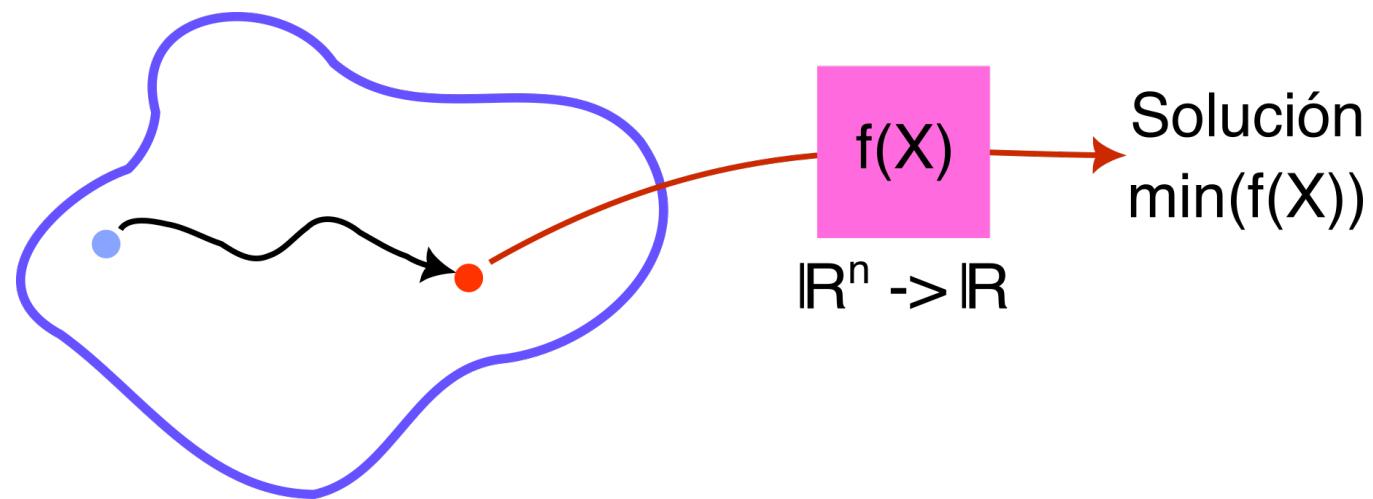
# ALGORITMOS DE BÚSQUEDA LOCAL

Problemas de optimización



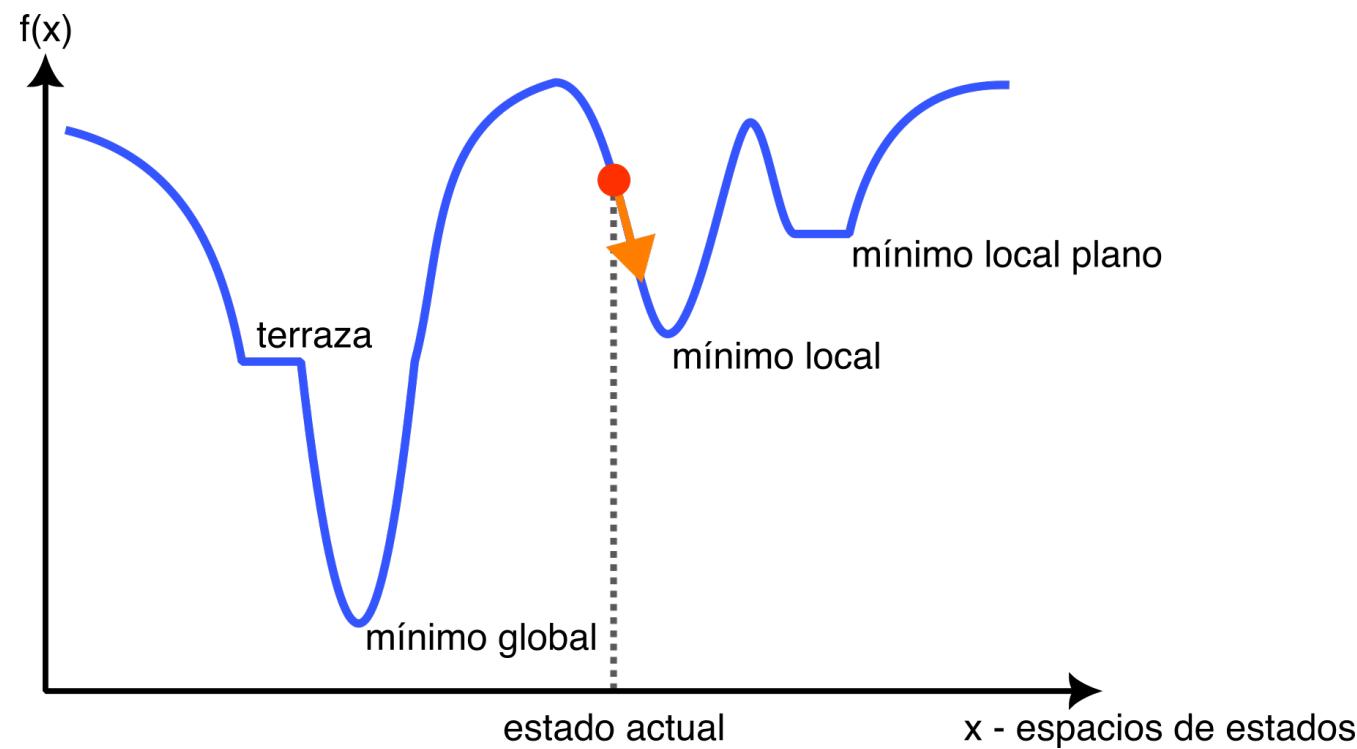
# ALGORITMOS DE BÚSQUEDA LOCAL

Problemas de optimización



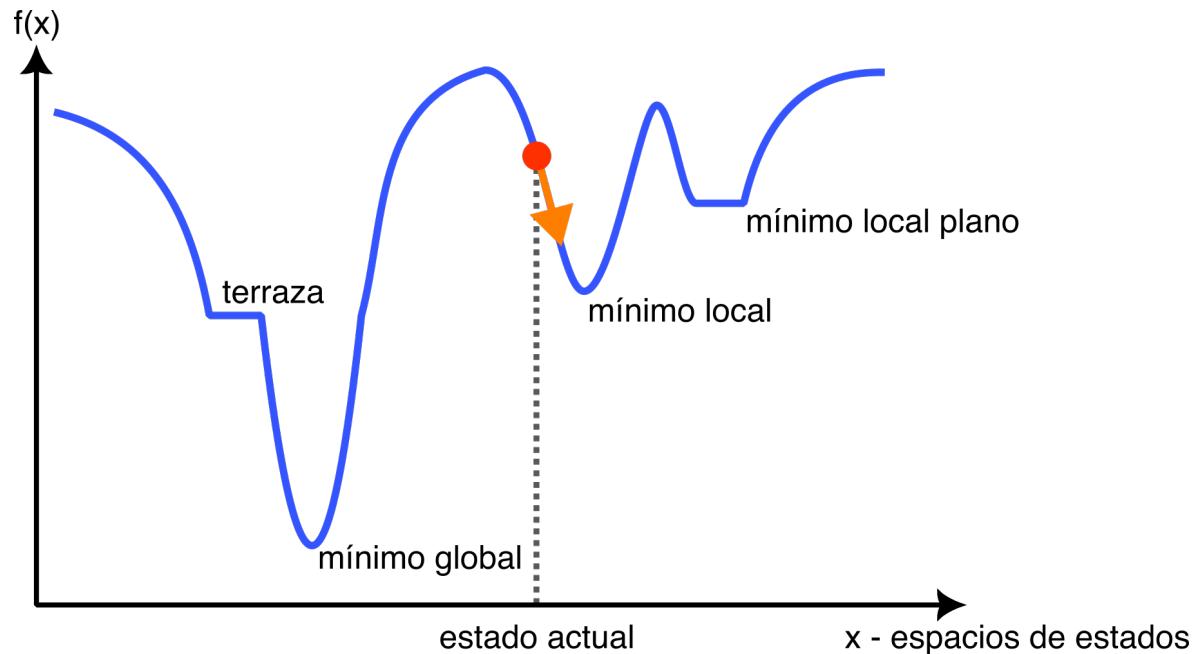
# ALGORITMOS DE BÚSQUEDA LOCAL

Problemas de optimización



# GRADIENTE DESCENDIENTE O ASCENDENTE

Este algoritmo que continuamente se mueve en dirección de mayor valor decreciente o creciente. Termina la búsqueda en donde ningún vecino está más bajo. Este algoritmo no miras más allá de lo que tiene de los vecinos.



```
def clim_hill(x_initial, fun_opt):
    x = x_initial
    f_x = fun_opt(x_initial)

    while 1:
        # Obtenemos vecino
        x_neib = obtain_neib(x)
        # Si f del vecino es menor
        if fun_opt(x) <= f_x:
            x = x_neib
            f_x = fun_opt(x)
        else:
            # Si llegamos a un minimo, termina
            return x
```

# SIMULATED ANNEALING

La idea es lograr cada tanto moverse en dirección contraria al de descenso usando nuevo parámetro que llamamos temperatura. Cuando más temperatura haya más probable de que podamos movernos en dirección contraria, cuando baja la temperatura, baja esta probabilidad. Si la temperatura es cero estamos en un algoritmo de descenso de colina puro.

Este algoritmo arranca con mucha temperatura y a medida que avanza, se va enfriando, de igual forma que el metal.

Se puede demostrar que, si se disminuye la temperatura bastante despacio, el algoritmo encontrará un óptimo global con probabilidad cerca de uno.

```
def sim_anneal(x_initial, fun_opt, temp_initial, rate_cooling):
    x = x_initial
    f_x = fun_opt(x_initial)
    T = temp_initial

    while T > 0.00001:
        # Obtenemos vecino
        x_neib = obtain_neib(x)
        deltaE = f_x - fun_opt(x)
        # Si f del vecino es menor, se acepta
        if deltaE > 0:
            x = x_neib
            f_x = fun_opt(x)
        else:
            # Si no, veamos si por distribución de
            # Boltzmann se acepta
            num_ale = random() # Número entre 0 y 1
            # Si por azar es menor a exp(-deltaE/T):
            if num_ale < exp(-deltaE/T):
                # Aceptamos el cambio a pesar que
                # aumente el costo.
                x = x_neib
                f_x = fun_opt(x)
        # Enfriamos
        T *= rate_cooling

    return x
```

---

# BÚSQUEDA LOCAL BEAM

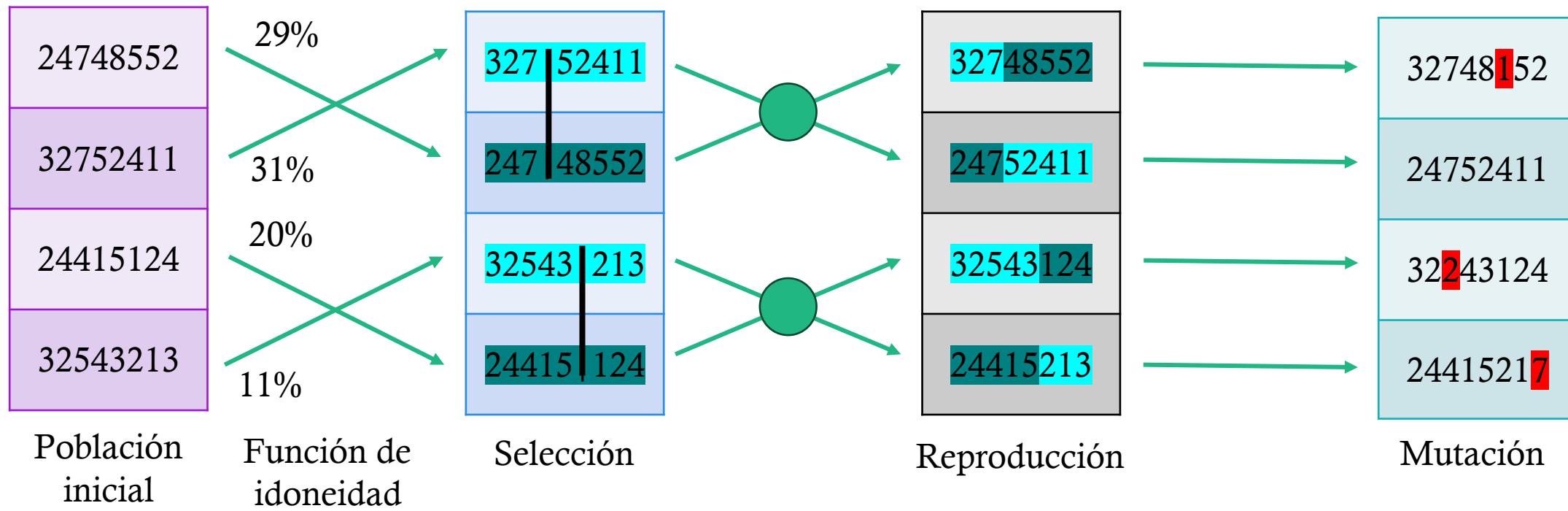
**Búsqueda local beam** guarda la información de  $k$  estados y sobre ellos realiza la búsqueda independientemente. Estos se inician al azar.

En cada paso se generan sucesores de los  $k$  estados. Si alguno cumple el objetivo, se termina, en cambio si no, se seleccionan los  $k$  mejores sucesores de la lista.

*Si un estado genera varios sucesores buenos y los otros  $k-1$  estados generan sucesores malos, entonces el efecto es que el primer estado abandona la búsqueda de los otros y se queda con los sucesores del primer estado.*

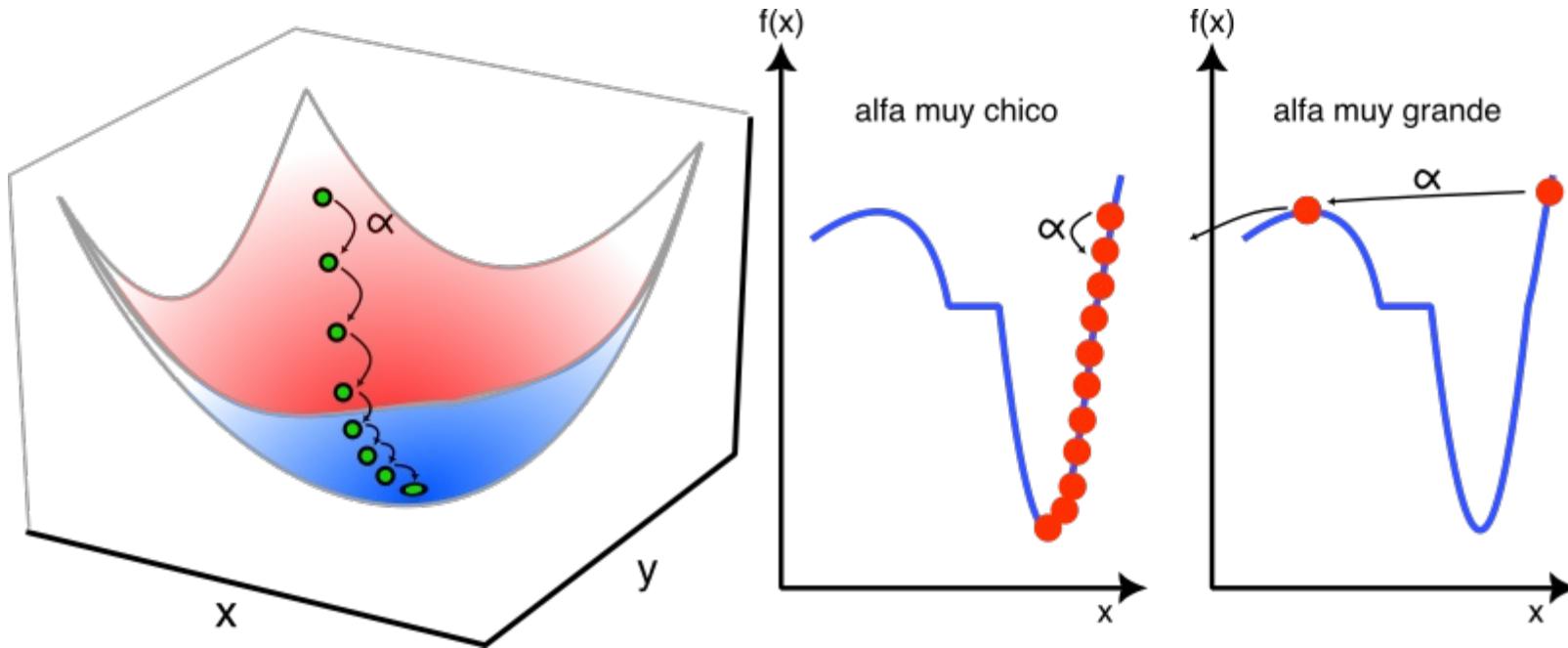
El algoritmo rápidamente abandona las búsquedas infructuosas y mueve sus recursos a donde se hace la mayor parte del progreso.

# ALGORITMOS GENÉTICOS



# BÚSQUEDA EN ESPACIOS CONTINUOS

Nos movemos mediante el gradiente de a pequeños pasos. La elección de alfa es sumamente importante, porque si alfa es muy pequeña, necesitamos demasiados pasos, en cambio sí es muy grande nunca puede converger.





---

# APRENDIZAJE AUTOMÁTICO

---

# APRENDIZAJE AUTOMÁTICO

Aprendizaje automático se entiende a:

Una computadora observa algunos datos, construye un modelo basado en estos datos, y usa el modelo como una hipótesis sobre el mundo y como una pieza de software que puede resolver problemas.

*¿Pero porque queremos que una computadora aprenda? ¿Por qué no programar el modelo directamente?*

- *No se puede anticipar todas las posibles situaciones futuras.*
- *No se tiene idea de cómo programar una solución por uno mismo.*



---

# APRENDIZAJE AUTOMÁTICO – EJEMPLOS

## Detección de SPAM

El detector de SPAM es una herramienta que hoy en día damos por sentado que todo software de correo electrónico debe tener. Este problema se resuelve usando Aprendizaje Automático.

Una forma de resolverlo es analizando las palabras y signos de puntuación de los correos. Este es un problema que llamamos de *aprendizaje supervisado*, cuyo resultado es un indicador si el corre es SPAM o no. Esto se llama *problema de clasificación*.

	george	you	your	hp	free	hpl	!	our	re
<i>Spam</i>	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13
<i>Email</i>	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42

---

# APRENDIZAJE AUTOMÁTICO – EJEMPLOS

## Detección de SPAM

El detector de SPAM es una herramienta que hoy en día damos por sentado que todo software de correo electrónico debe tener. Este problema se resuelve usando Aprendizaje Automático.

En este problema no todo error al predecir es lo mismo. *Queremos evitar filtrar el buen correo electrónico*, mientras que dejar pasar el spam no es deseable pero sus consecuencias son menos graves.

	george	you	your	hp	free	hpl	!	our	re
<i>Spam</i>	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13
<i>Email</i>	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42

# APRENDIZAJE AUTOMÁTICO – EJEMPLOS

## Cáncer de próstata

Tenemos mediciones del antígeno prostático específico (PSA) y una serie de medidas clínicas, en 97 hombres que estaban a punto de recibir una prostatectomía radical.

Como objetivo se busca predecir el log de PSA usando un numero de mediciones a partir de una serie de mediciones que incluyen el registro del volumen del cáncer, el registro del peso de la próstata, la edad, el registro de la cantidad de hiperplasia prostática benigna, la invasión de la vesícula seminal, entre otros.

Este es un problema de *aprendizaje supervisado*, conocido como *problema de regresión*, porque la medición del resultado es *cuantitativa*.

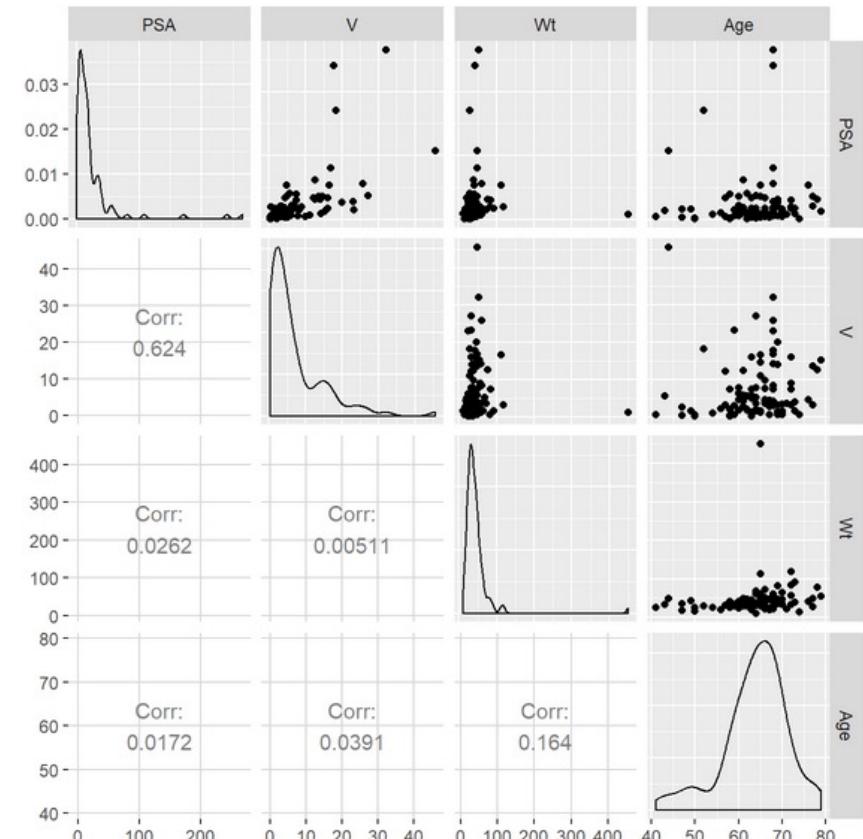
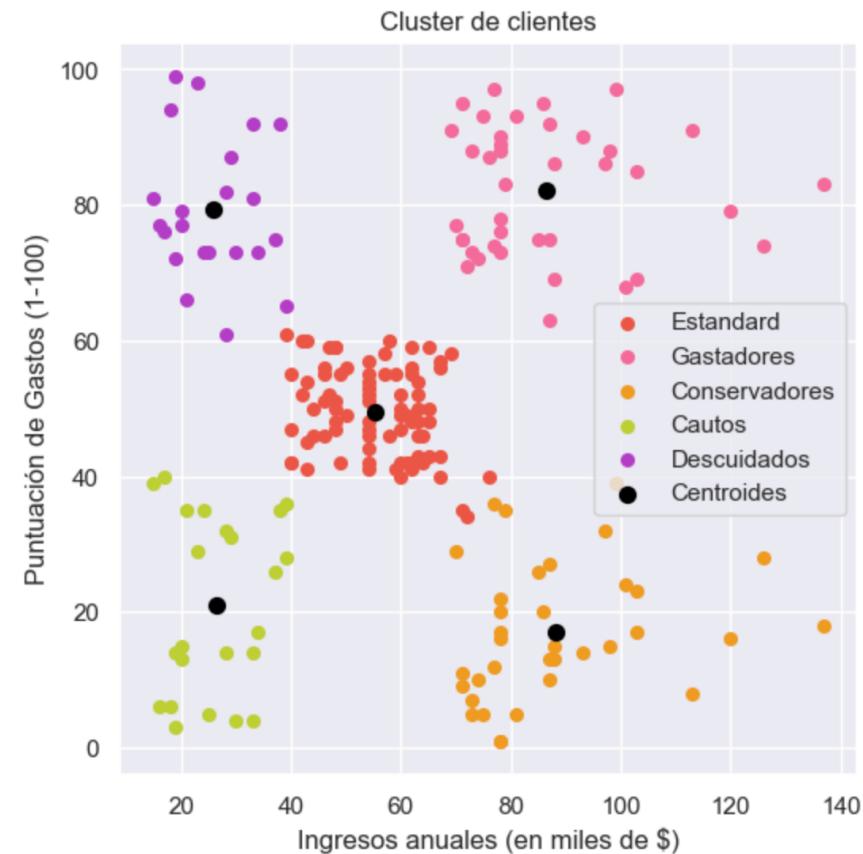


Imagen obtenida de [ADA2: Homework 06, Ch 03 A Taste of Model Selection for Multiple Regression](#)

# APRENDIZAJE AUTOMÁTICO – EJEMPLOS

## Descubrimiento de segmentos de clientes.

Una empresa de retail quiere entender cómo se segmentan sus clientes en información demográfica y gustos dados su interior de compra. Usando estos datos se usa un modelo que agrupa los datos en diferentes grupos por *similitud*. Una vez establecidos estos grupos, la empresa puede establecer estrategias de marketing diferentes para cada uno de estos grupos. Este tipo de problema es de tipo *no supervisado* y particularmente de *clustering*.



---

# APRENDIZAJE AUTOMÁTICO – EJEMPLOS

## Sistema de recomendación de videos

Una empresa dedicada a compartir videos necesita mantener a los usuarios comprometidos en la plataforma. Necesita que cuando un nuevo video termina, recomendarle al usuario el siguiente video a ver, de tal forma que la persona siga deseando mantenerse en la plataforma. Por lo que es vital construir un sistema de recomendación que se utilice la información del usuario en la plataforma.

Este tipo de algoritmos se llaman de *recomendación* y es de tipo *no supervisado*.



---

# DATOS

---

# DATOS

Lo más importante en Aprendizaje Automático (y en Data Science en general) son los

## Datos

Nos permite describir un objeto al que podemos llamar *entidad*.

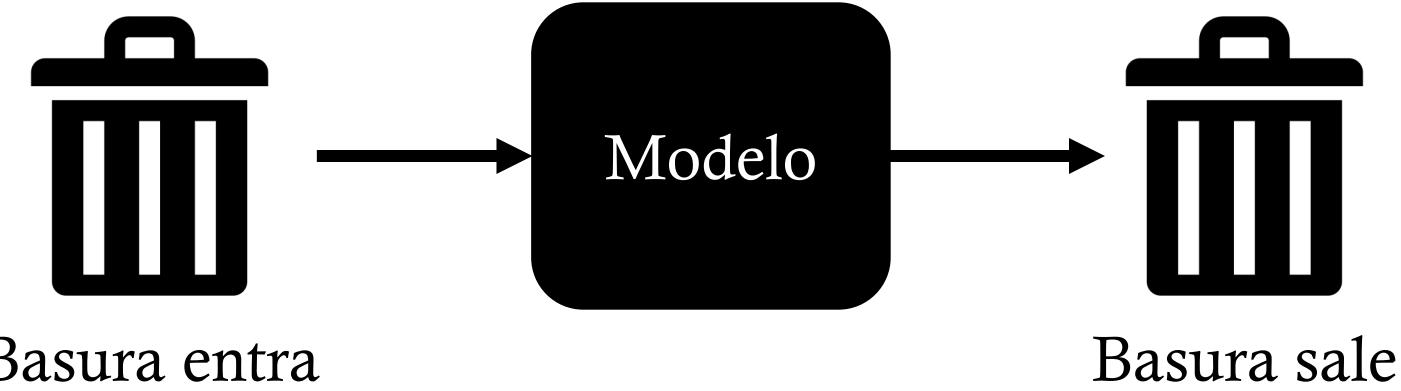
Esta entidad y su información puede ser diferente a pesar de que describa un mismo objeto. La forma en que se elija representar los datos no solo afecta la forma en que se construyen sus sistemas, sino también los problemas que sus sistemas pueden resolver.

Por ejemplo, queremos representar un auto:

- En un problema para compra y venta de autos, podemos representarlo con el fabricante, modelo, año, color, y su precio.
- En un problema de un sistema de seguimiento policial, podemos representarlo por quien es el dueño, patente y su historia de direcciones registradas.

# DATOS

---



---

# DATOS

Los datos se pueden encontrar en dos formas:

- **Datos estructurados:** Tienen un formato estandarizado que permite tanto al software como a las personas acceder a estos de forma eficaz. Por lo general, se trata de datos tabulares con filas y columnas que definen claramente sus atributos. Las computadoras pueden procesar eficazmente los datos estructurados en busca de información dado que se trata de información cuantitativa.
- **Datos no estructurados:** Son información sin un *modelo de datos* establecido o son datos que no están ordenados de una manera predefinida. Por ejemplo, archivos de texto, video, informes, e-mails, imágenes.

---

# DATOS

## Datos estructurados

En Aprendizaje Automatico en general se usan estructuras dos dimensiones y notaciones vectoriales para referirnos a los datos:

- Cada fila del array es una **muestra**, **observación** o dato puntual.
- Cada columna es una **característica** (**feature** o **atributo**), de la observación.
- En el caso más general hay una columna, que llamaremos **objetivo**, **label**, **etiqueta** o **respuesta**, y que será el valor que se pretende predecir.

# DATOS

	Atributos/features					Objetivo
	Position	Experience	Skill	Country	City	Salary (\$)
<b>Observación →</b>	Developer	0	1	Argentina	Buenos Aires	103100
	Data Scientist	2	2	Uruguay	Montevideo	104900
	Developer	3	1	Argentina	Chivilcoy	106800
	QA Eng	2	2	Colombia	Bogotá	108700
	Product Manager	1	5	Perú	Lima	110400
	Developer	7	5	Paraguay	Asunción	112300
	Cloud Eng	5	2	Argentina	Buenos Aires	116100

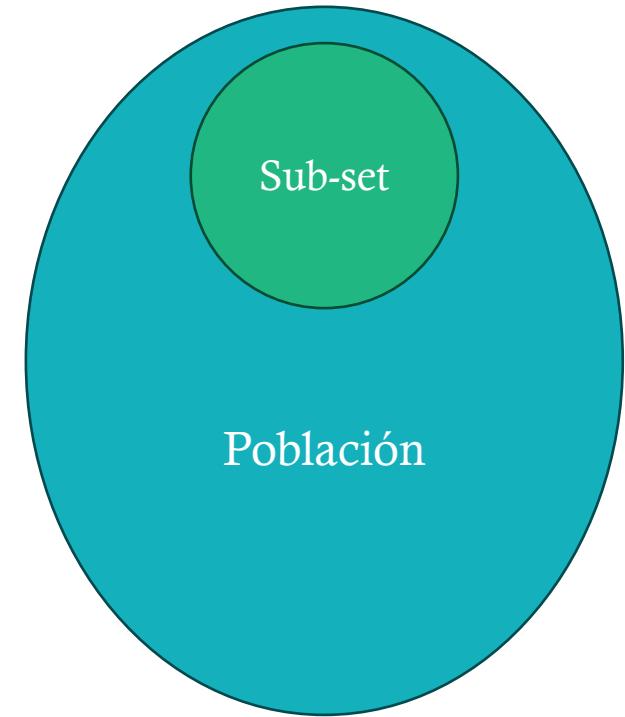
# DATOS

---

Cuando modelamos usando Aprendizaje Automático tenemos un variable dependiente **target  $\mathbf{y}$**  (que podemos conocer o no) dado un conjunto de predictores  **$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$** , el cual podemos describir como un vector  **$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p)$** .

*El desafío en Aprendizaje Automático es elegir las entradas correctas.*

El conjunto total de estas observaciones se llama **población**, lo cual casi nunca podemos tener, sino que tenemos un sub-set de estas observaciones.



---

# DATOS

Cada atributo puede tener diferentes formas, una medida de altura es diferente a el tipo de música que más le gusta a un usuario.

Tenemos diferentes tipos de variables:

**Variables numéricas:** Son aquellas que representan números y con ellas se pueden realizar operaciones aritméticas.

- **Discretas:** Son números enteros, cosas que se pueden contar. 1, 2, 3 empleados, 568 personas.
- **Continuas:** Números reales. El valor dado a una observación para una variable continua puede incluir valores tan pequeños como lo permita el instrumento de medición o la representación numérica. Altura, peso, costo, precio...

---

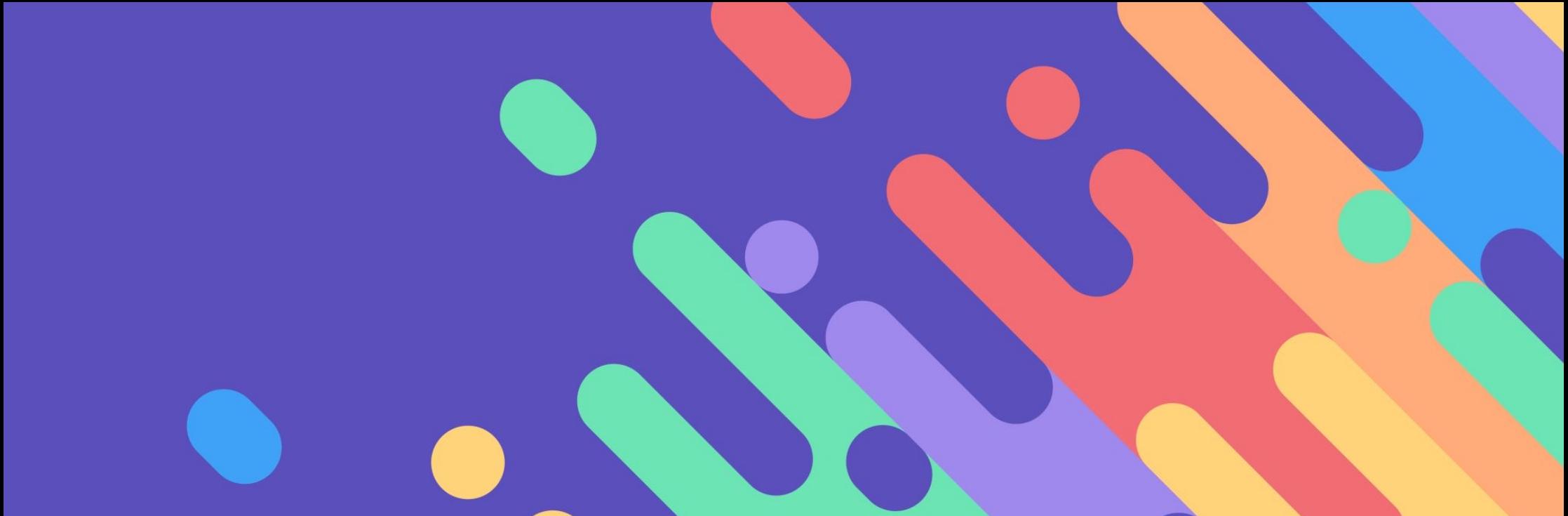
# DATOS

**Variables categóricas:** Es una variable que puede tomar uno de un número limitado, y por lo general fijo, de posibles valores.

- **Nominal:** Valores que toman corresponden a nombres de categorías, clases o estados de las cosas. Estado civil (soltero, casado, divorciado), Spam en e-mails (Binario, es spam o no). Tipo de cerveza (Ale, Pale, Stout, etc.)
- **Ordinal:** Similar al nominal, con la diferencia de poder aplicar un orden sobre estas categorías.

Estado de satisfacción: Me disgusta mucho, me disgusta, neutro, me gusta, me gusta mucho.

No tiene que haber una equidistancia entre las opciones.

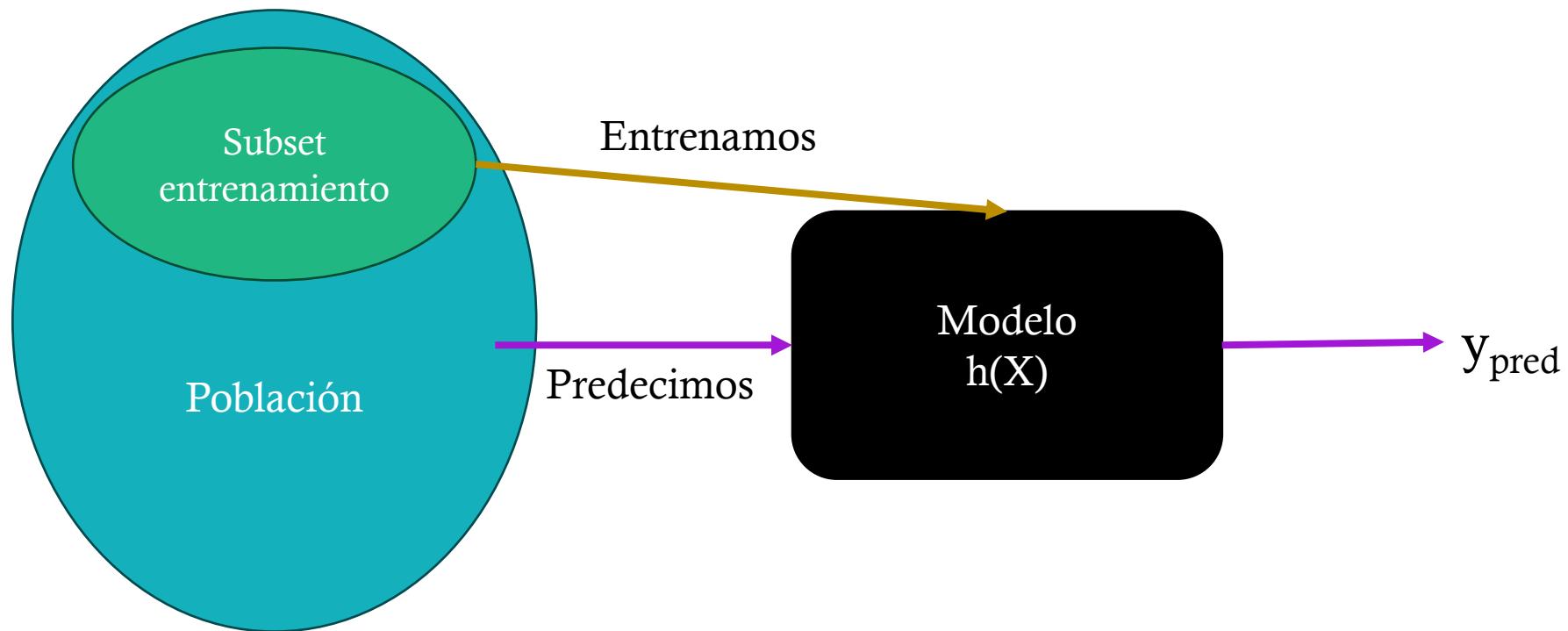


---

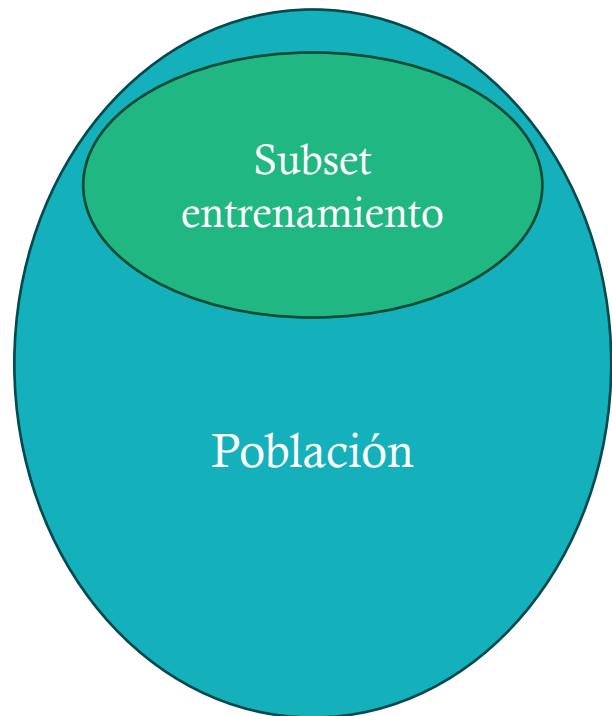
# FORMAS DE APRENDIZAJE

# FORMAS DE APRENDIZAJE

Un esquema de aplicar Aprendizaje Automático nos queda...



# FORMAS DE APRENDIZAJE



Un paso importante que nos delata este grafico es que, en la realidad no podemos saber exactamente qué tan bien funcionará un modelo predictivo en la práctica porque no conocemos la verdadera distribución de los datos.

Siempre vamos a trabajar con sub-sets que podemos estimar y optimizar el rendimiento del modelo en un conjunto conocido de datos de entrenamiento.

El rendimiento sobre este conjunto conocido de datos de entrenamiento se denomina **riesgo empírico**.

# FORMAS DE APRENDIZAJE

Como vimos en los ejemplos hay diferentes tipos de aprendizaje, los cuales depende de la forma que tiene principalmente **y**.

- **Aprendizaje supervisado:** El modelo observa pares de entradas-salidas y aprende la relación entre ellos. Es decir, en este tipo de aprendizaje, conocemos el valor de  $y$  y se lo enseñamos al modelo.
  - Los modelos aprenden de los resultados conocidos y realizan ajustes en sus parámetros interiores para adaptarse a los datos de entrada.
  - Una vez que el modelo es entrenado adecuadamente, y los parámetros internos son coherentes con los datos de entrada y los resultados de los datos de entrenamiento, el modelo podrá realizar predicciones adecuadas ante nuevos datos



Image by [vectorjuice](#)

---

# FORMAS DE APRENDIZAJE

Como vimos en los ejemplos hay diferentes tipos de aprendizaje, los cuales depende de la forma que tiene principalmente **y**.

- **Aprendizaje no supervisado:** El modelo aprende patrones de la entrada sin ninguna retroalimentación. Es decir, no contamos con **y** de antemano.
- **Aprendizaje por refuerzo:** El agente aprende con una serie de refuerzos: recompensas y castigos. Depende del agente decidir cuál de las acciones anteriores al refuerzo fue la más responsable de él y modificar sus acciones para apuntar a más recompensas en el futuro.



---

# APRENDIZAJE SUPERVISADO

---

# APRENDIZAJE SUPERVISADO

Mas formalmente podemos definir a la tarea de aprendizaje supervisado como:

Dado un **set de entrenamiento** de N observaciones de pares de entradas y salida:

$$(X_1, y_1), (X_2, y_2), \dots, (X_N, y_N)$$

Donde cada par está generado por una función desconocida  $y = f(X)$ ,

Se descubre una función **h** que aproxima a la verdadera función f.

La función **h** es la que llamamos a la hipótesis de la población o el **modelo** que provienen de un espacio de hipótesis.

Llamamos a las salidas **y** como **ground truth**.

---

# APRENDIZAJE SUPERVISADO

## *¿Como elegimos el espacio de hipótesis?*

Es posible que tengamos algún conocimiento previo sobre el proceso que generó los datos. O podemos realizar un **análisis de datos exploratorio**: examinar los datos con pruebas estadísticas y visualizaciones para tener una idea de los datos y una idea de qué espacio de hipótesis podría ser apropiado. O simplemente podemos probar múltiples espacios de hipótesis y evaluar cuál funciona mejor.

## *¿Y cómo elegimos un modelo dentro del espacio de hipótesis?*

Podríamos esperar una hipótesis consistente: una **h** tal que cada  $X$  en el conjunto de entrenamiento tenga  $h(x) = y$ . Si las salidas son de valor continuo es muy difícil tener una salida exacta. En esos casos se usa una **función de ajuste** para la cual cada  $h(x_i)$  esté cerca de  $y_i$ .

---

---

# APRENDIZAJE SUPERVISADO

## Generalización

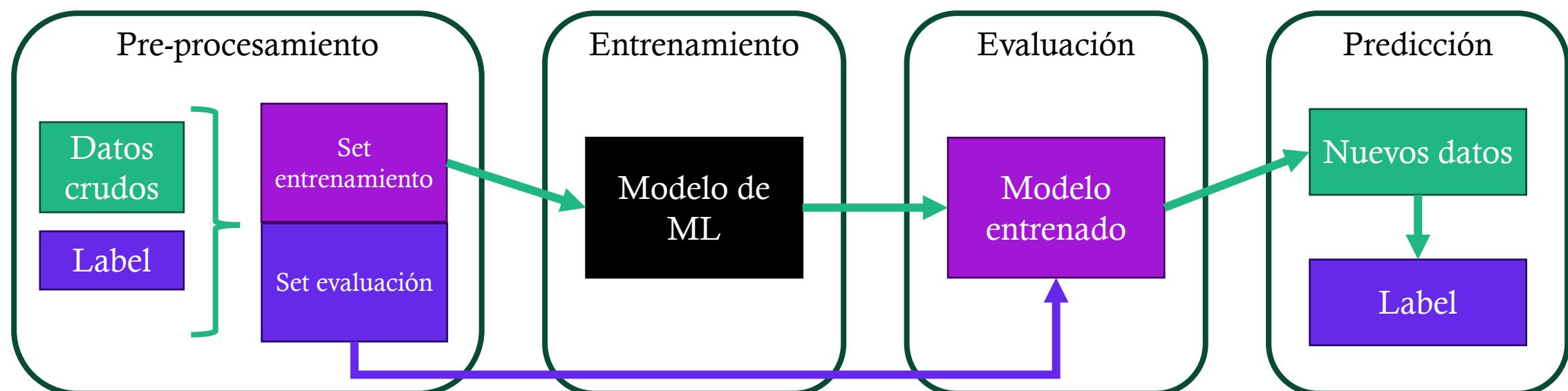
La verdadera medida de si un modelo funciona correctamente no es con respecto al set de entrenamiento, sino que tan bien maneja entradas que nunca vio.

Eso lo podemos ver con un segundo set de pares  $(X_i, y_i)$  llamada **conjunto de prueba**.

Se dice que  $h$  **generaliza** bien si predice con precisión los resultados de este conjunto.

# APRENDIZAJE SUPERVISADO

## Generalización



---

# APRENDIZAJE SUPERVISADO

## Tipos de aprendizaje supervisado

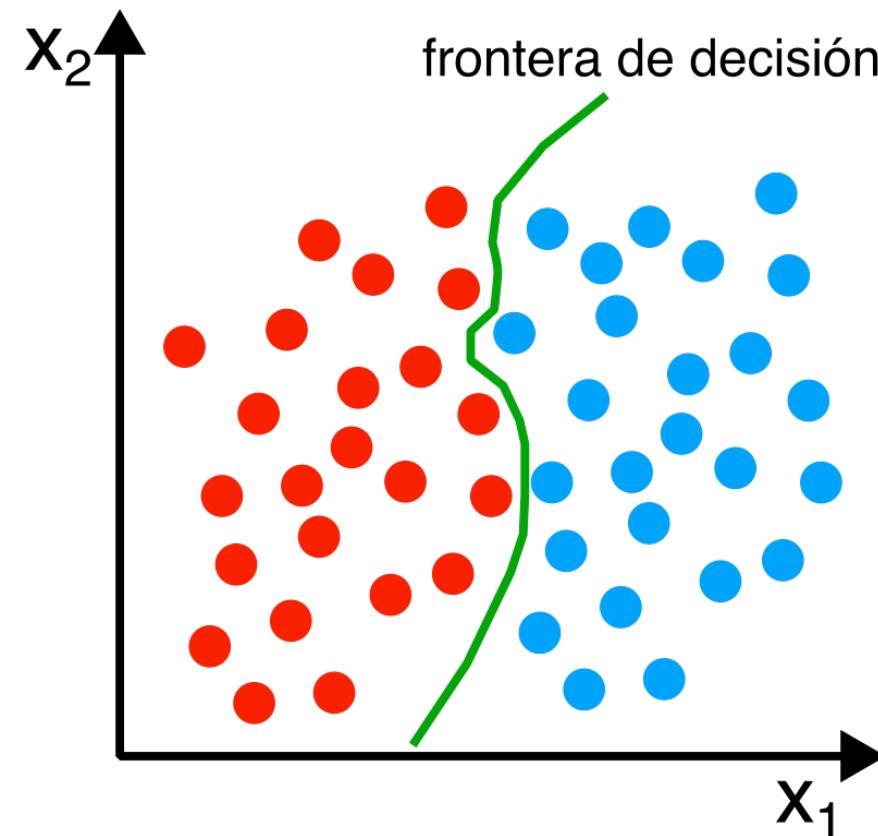
Si el target **y** es una *variable categórica* o toma valores discretos, este tipo de problema se llama un problema de clasificación. A su vez tenemos dos subvariantes:

- **Clasificación binaria** (Por ejemplo, es SPAM o no SPAM).
- **Clasificación multi-clase:** Múltiple clases, como por ejemplo clasificación del nivel socioeconómico de una persona (alta, media y baja).
  - Una variante de este tipo es cuando la cardinalidad es muy alta, es decir, se tienen muchísimas clases.

# APRENDIZAJE SUPERVISADO

Tipos de aprendizaje supervisado

Clasificación



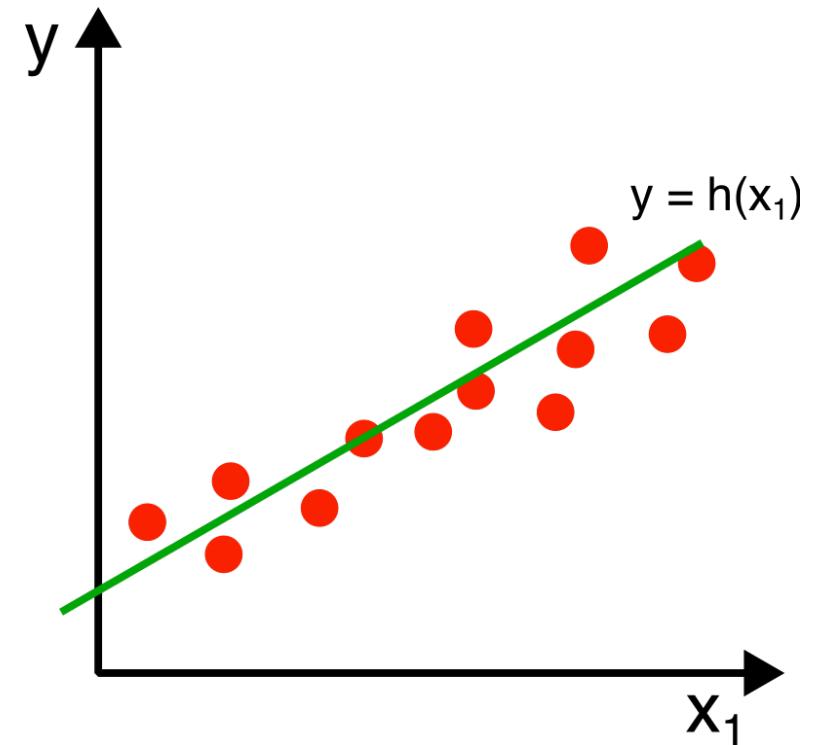
# APRENDIZAJE SUPERVISADO

## Tipos de aprendizaje supervisado

Si el target  $y$  es una *variable numérica*, este tipo de problema se llama un problema de regresión.

Se centra en estudiar las relaciones entre una variable dependiente de una o más variables independientes.

Es importante notar que, en Aprendizaje Automático, cuando buscamos una  $h(X)$  estamos armando un modelo puramente empírico. Es decir, nos basamos 100% en los datos medidos. En contraste con los modelos basados en propiedades fundamentales.



---

# APRENDIZAJE SUPERVISADO

Tipos de aprendizaje supervisado

## Regresión vs. Clasificación

Regresión y clasificación son problemas muy similares entre sí. En ambos buscamos predecir una variable, la diferencia radica en que regresión predice una variable numérica y clasificación una categórica.

*No es infrecuente encontrar que se puede resolver problemas como clasificación o regresión.*

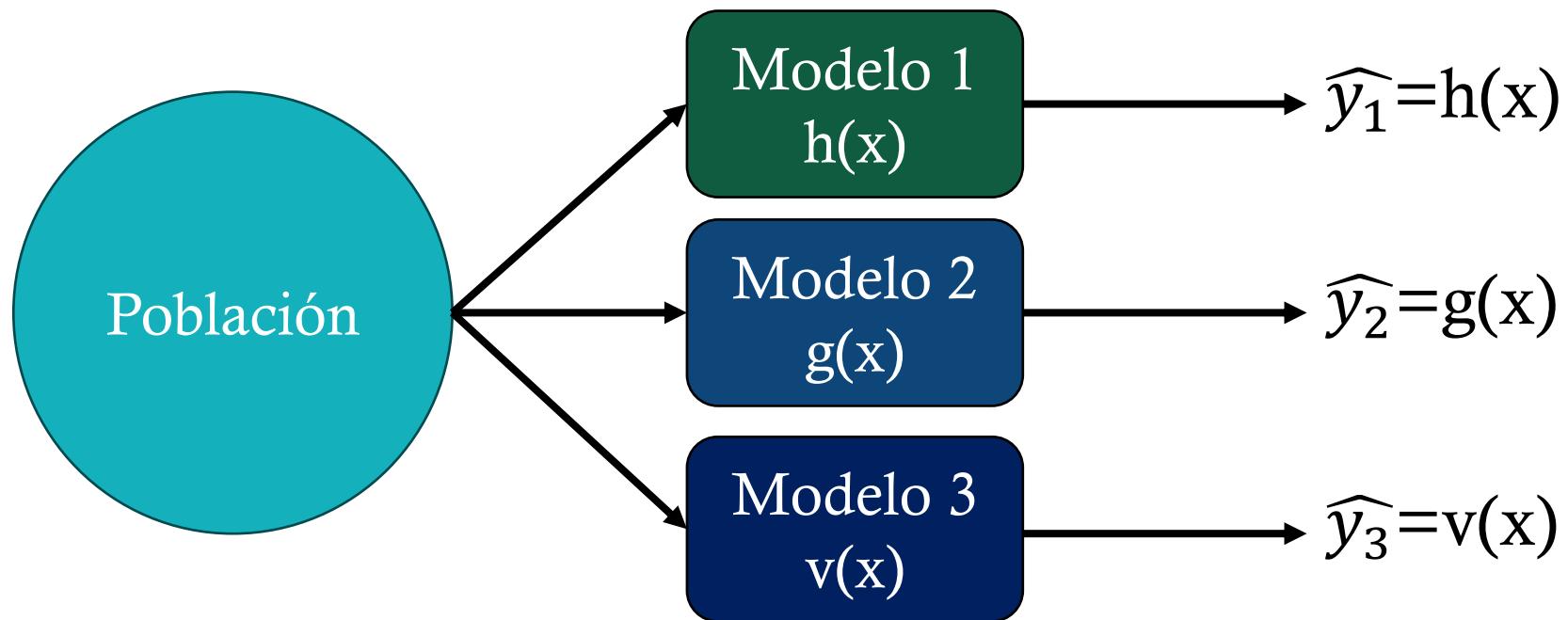


---

# SESGO Y VARIANZA

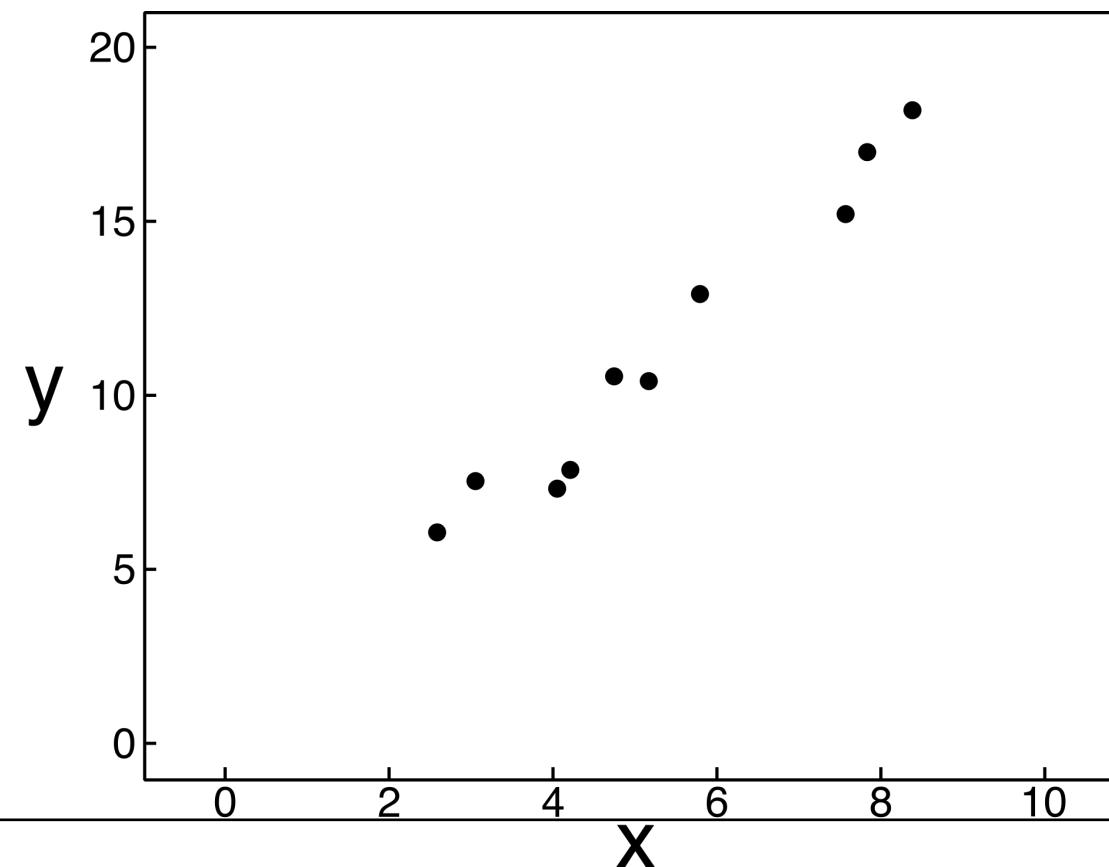
# SESGO Y VARIANZA

Supongamos que tenemos tres modelos en un problema de regresión. Tenemos una sola entrada  $\mathbf{x}$  y una salida  $\mathbf{y}$  que depende de  $\mathbf{x}$  con una relación  $f(\mathbf{x})$  que queremos modelar:



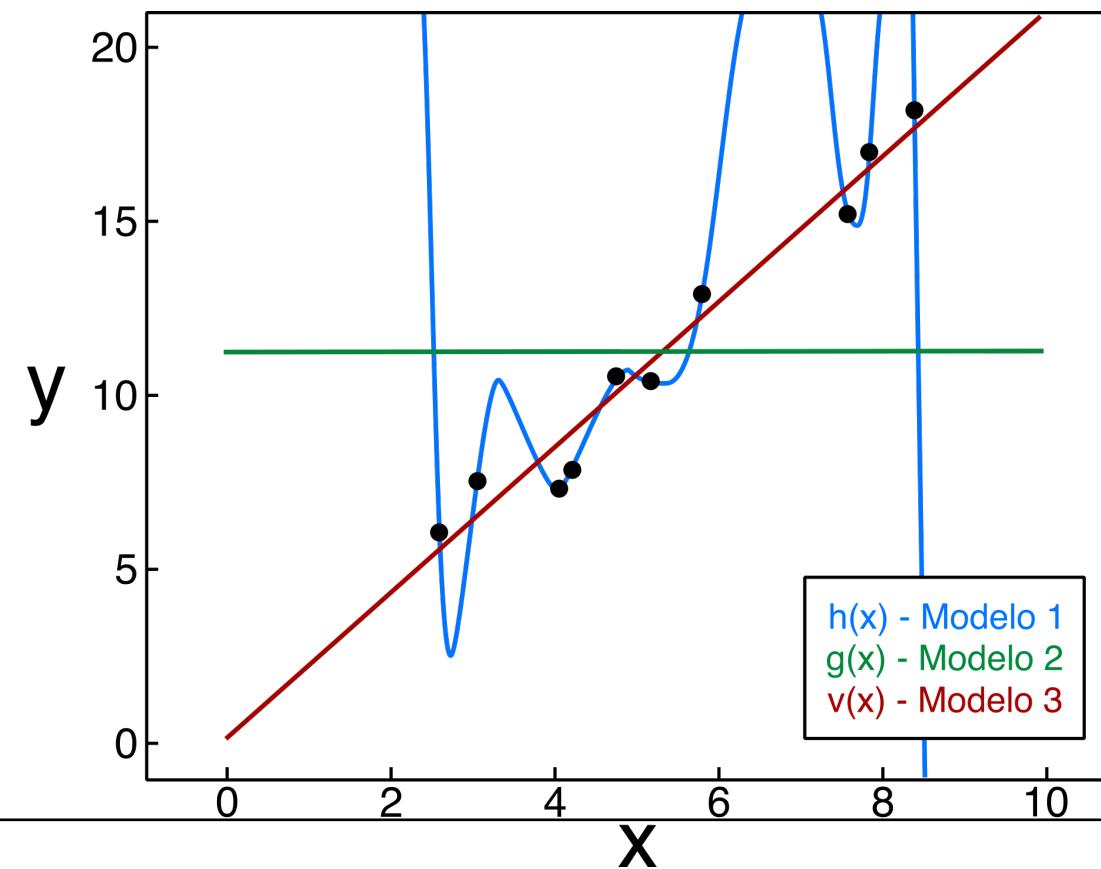
# SESGO Y VARIANZA

Entrenamos con un **set de entrenamiento** y nos queda:



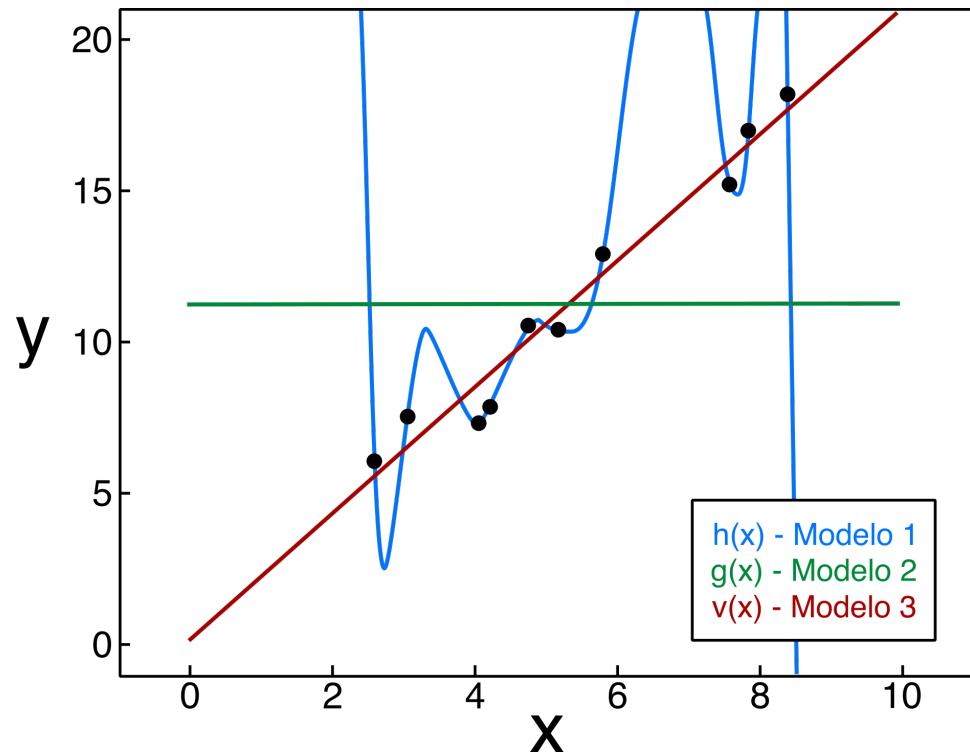
# SESGO Y VARIANZA

Entrenamos con un **set de entrenamiento** y nos queda:



# SESGO Y VARIANZA

Entrenamos con un **set de entrenamiento** y nos queda:



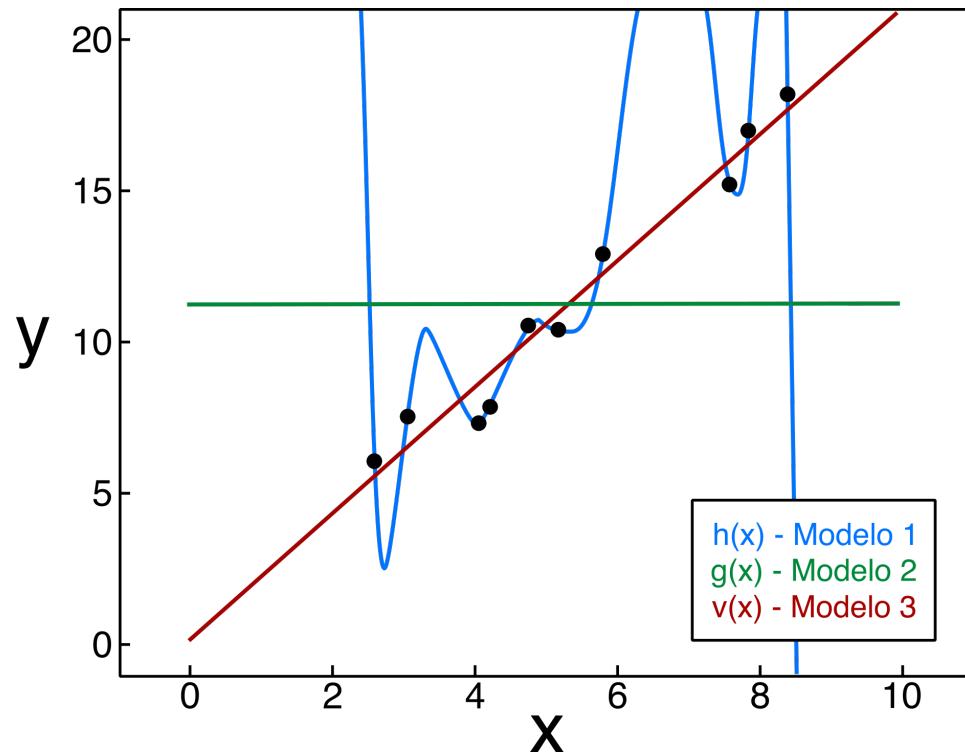
Midamos el error, para ello calculamos de la siguiente forma:

$$MAE_p = \sum_{i=1}^N (y_i - \hat{y}_p)$$

- Modelo 1:  $MAE_1 = 0$
- Modelo 2:  $MAE_2 = 5$
- Modelo 3:  $MAE_3 = 1.25$

# SESGO Y VARIANZA

Entrenamos con un **set de entrenamiento** y nos queda:



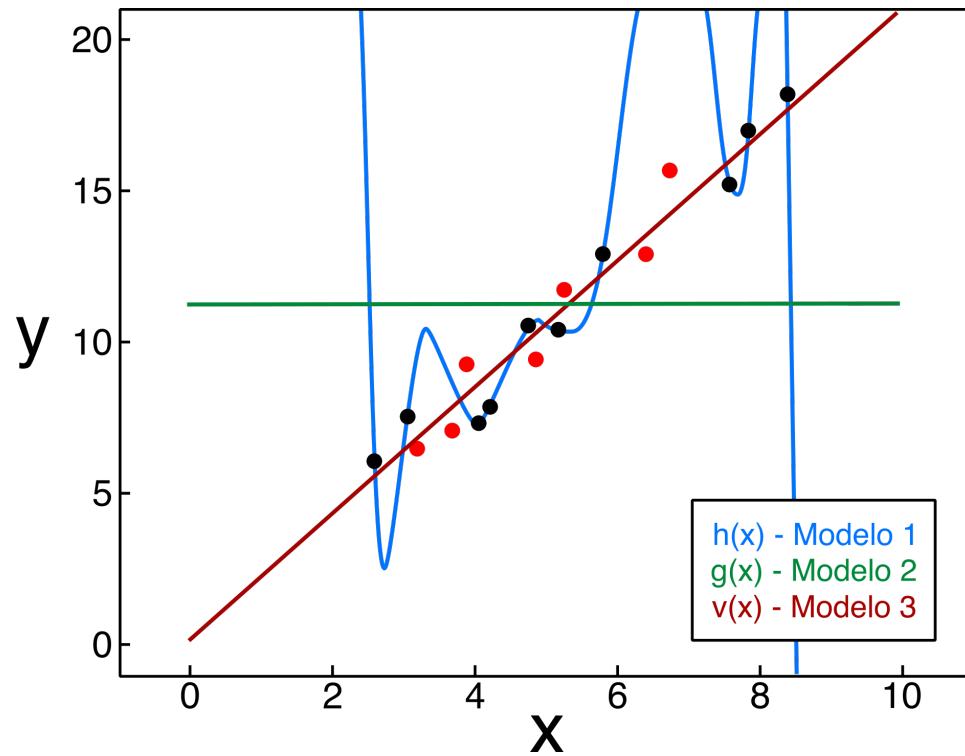
Midamos el error, para ello calculamos de la siguiente forma:

$$MAE_p = \sum_{i=1}^N (y - \hat{y}_p)$$

- Modelo 1:  $MAE_1 = 0$
- Modelo 2:  $MAE_2 = 5$
- Modelo 3:  $MAE_3 = 1.25$

# SESGO Y VARIANZA

¿Pero realmente es el mejor? Evaluemos con el **set de evaluación**



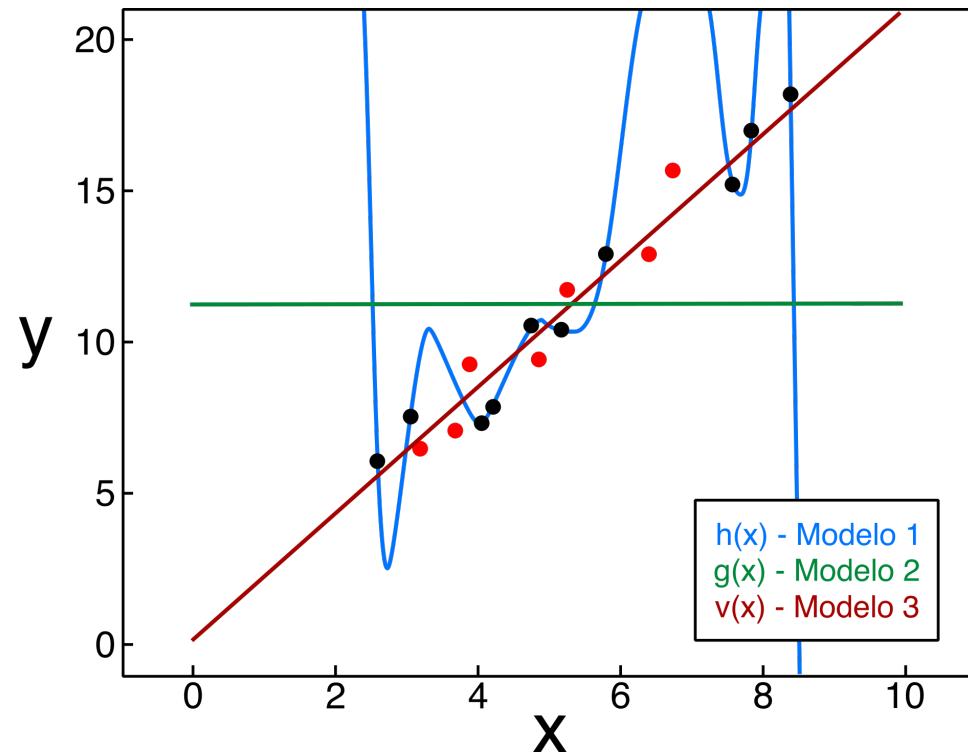
Midamos el error...

$$MAE_p = \sum_{i=1}^N (y_i - \hat{y}_p)$$

- Modelo 1:  $MAE_1 = 90$
- Modelo 2:  $MAE_2 = 5.2$
- **Modelo 3:  $MAE_3 = 1.75$**

# SESGO Y VARIANZA

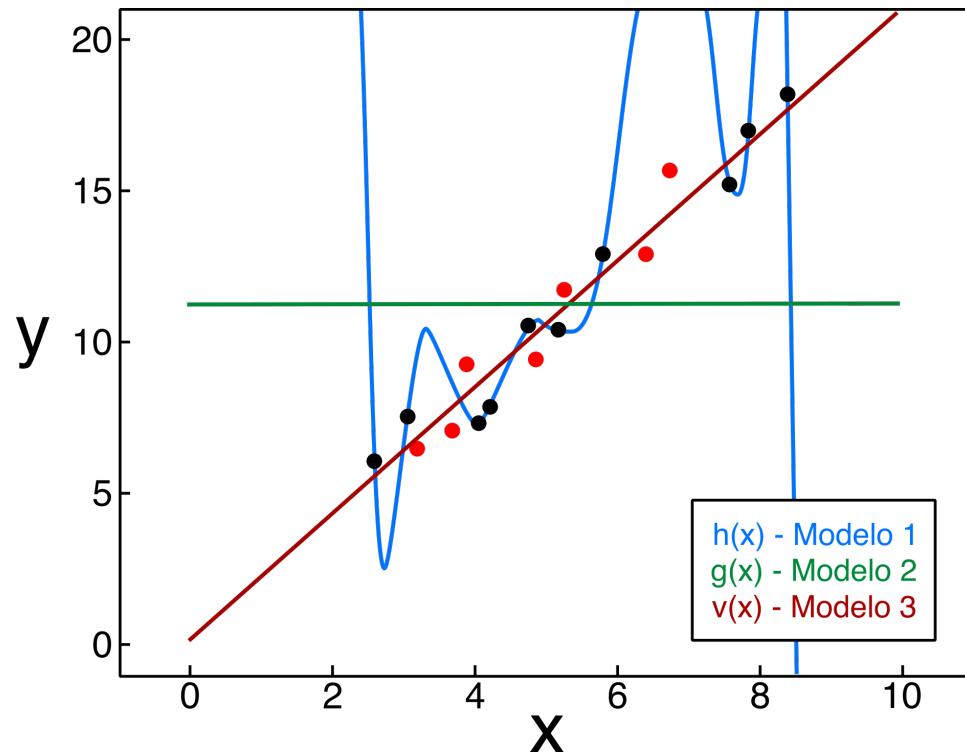
¿Pero realmente es el mejor? Evaluemos con el **set de evaluación**



- El modelo 1 rinde muy bien con los datos de entrenamiento, pero muy mal con el de evaluación. Esto es lo que se llama **sobreajuste**, el modelo aprende tan bien que hasta aprende el ruido de los datos.  
Este modelo tiene demasiado parámetros, por lo que es demasiado complejo y por consiguiente **no generaliza**.
- Cuando se elige un modelo, se busca que:  
*parametros << cantidad de obs. de entrenamiento*

# SESGO Y VARIANZA

¿Pero realmente es el mejor? Evaluemos con el **set de evaluación**



- El modelo 2 rinde pobemente con el set de entrenamiento y el de evaluación. Este modelo tiene no logra capturar el comportamiento buscado.

Cuando esto ocurre es lo que llamamos **subajuste**.

- El modelo 3 es el mejor modelo, el resultado entre el de evaluación y entrenamiento es similar, el modelo **generaliza**, sin tomar el ruido.

---

# SESGO Y VARIANZA

El error que se comete con un modelo que sobre-ajusta y sub-ajusta es diferente.

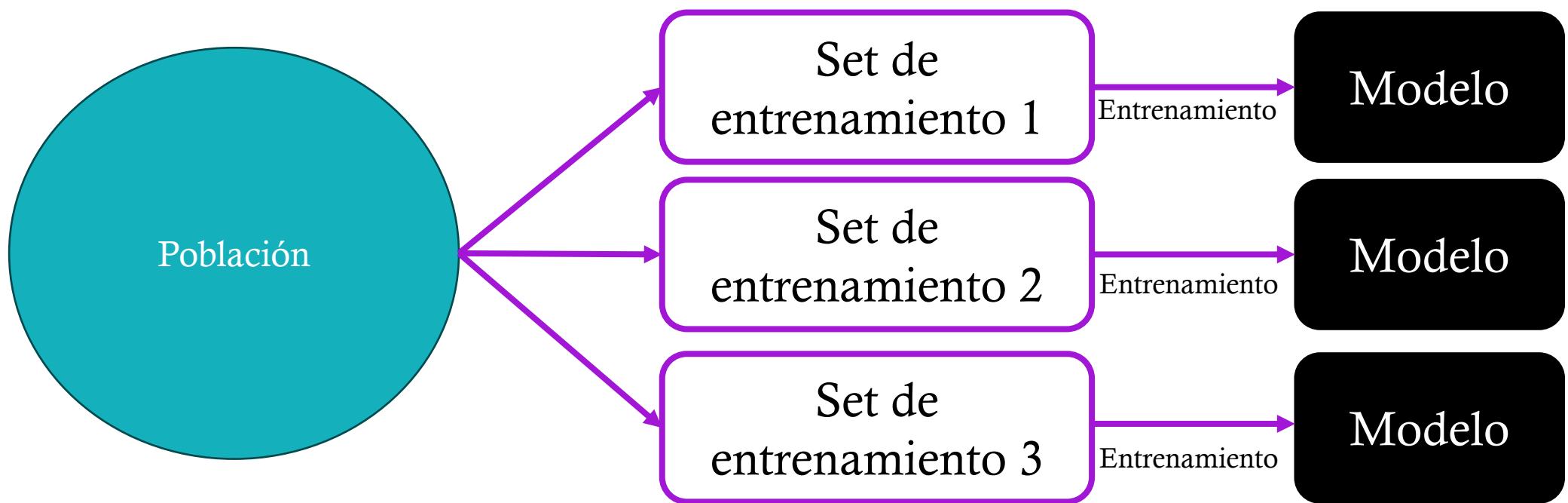
- El error de sobreajuste le llamamos **error de varianza**
- El error de subajuste le llamamos **error de sesgo**

Todo modelo el error total tiene como parte a los dos errores.

Estos errores son complementarios entre sí. Si queremos bajar uno, el otro va a subir.

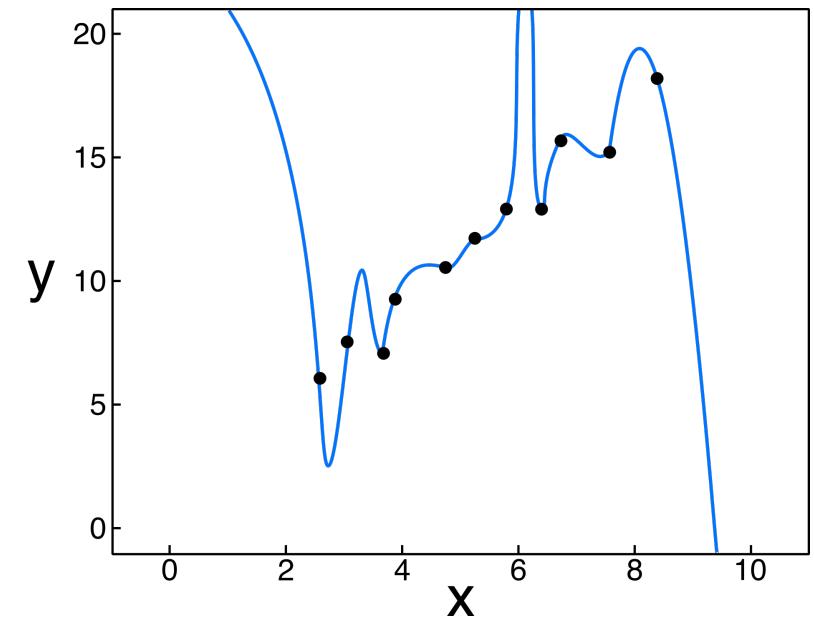
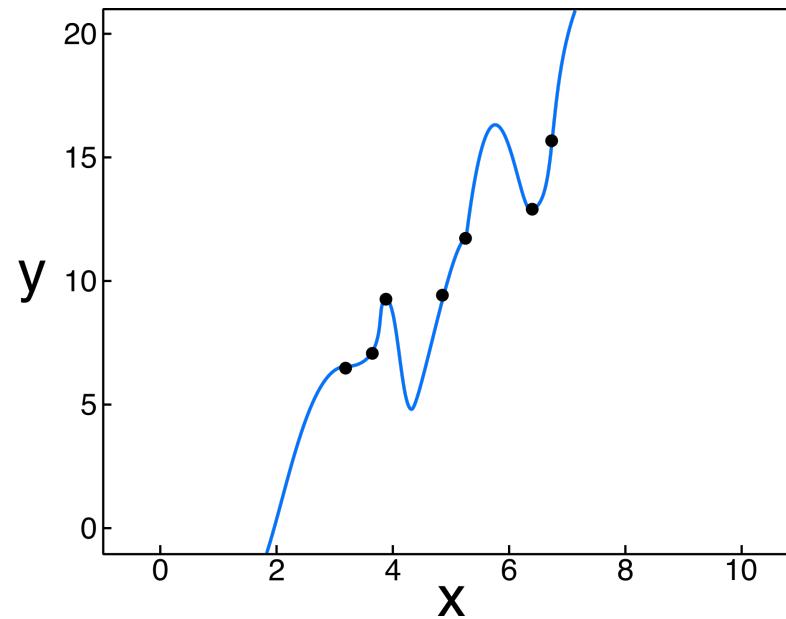
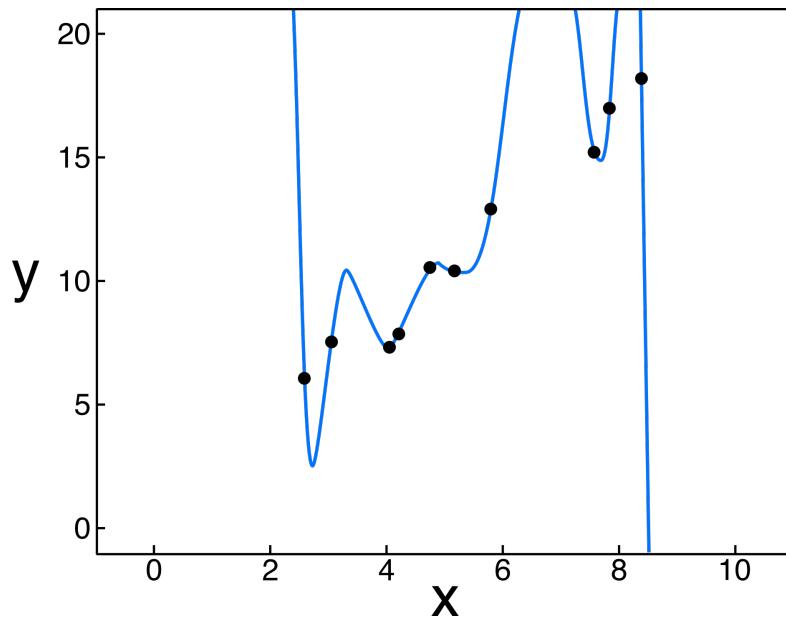
# SESGO Y VARIANZA

Estos errores lo podemos ver si tomamos diferentes **sets de entrenamiento** y entrenamos el modelo con cada uno de estos.



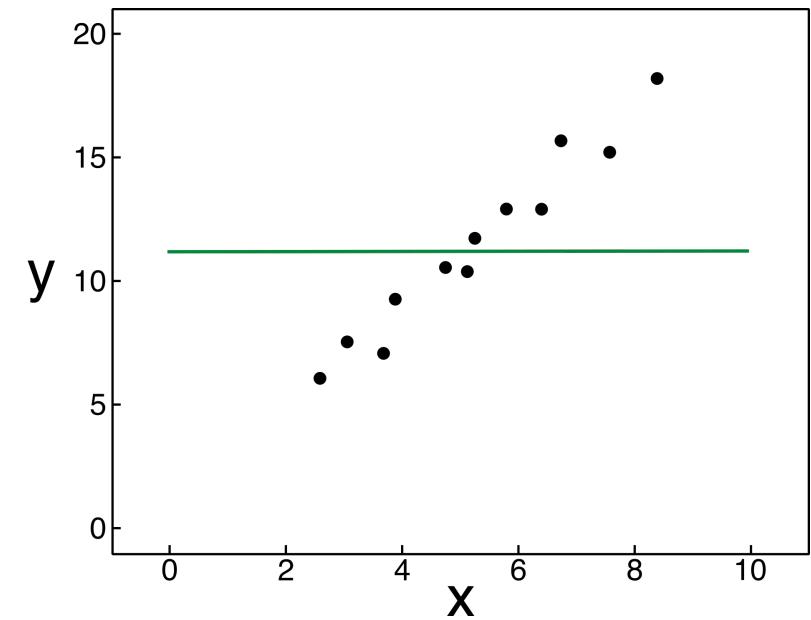
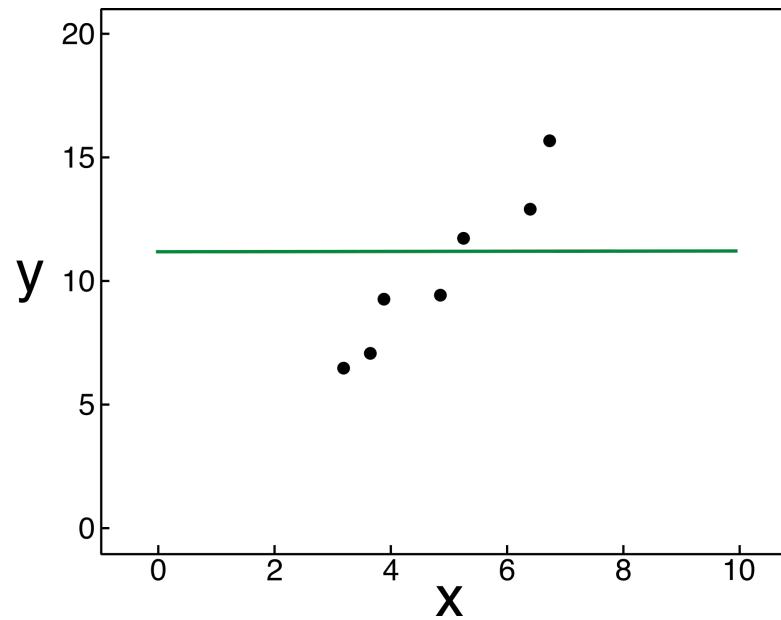
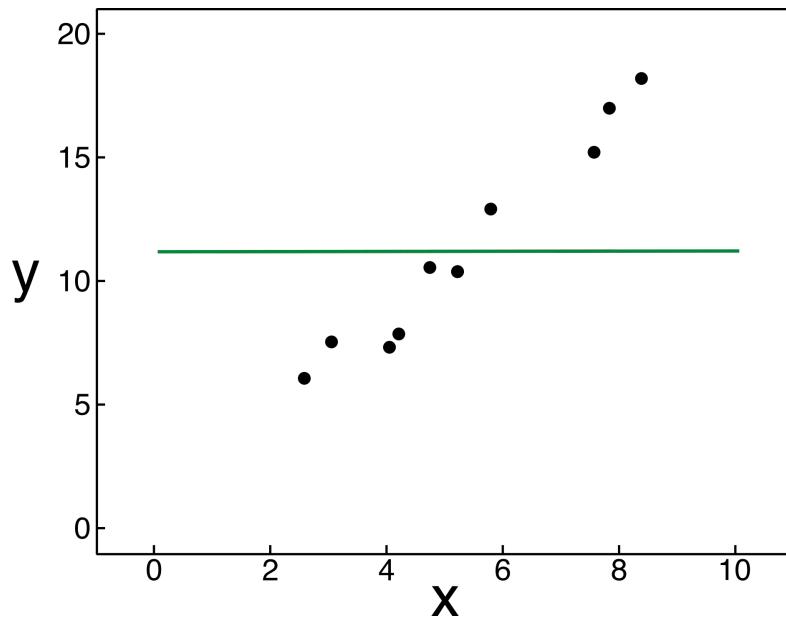
# SESGO Y VARIANZA

Si tenemos un modelo que sobreajusta, diferentes sets de entrenamiento nos darán modelos muy diferentes:



# SESGO Y VARIANZA

Si tenemos un modelo que subajusta, diferentes sets de entrenamiento nos darán modelos muy similares:



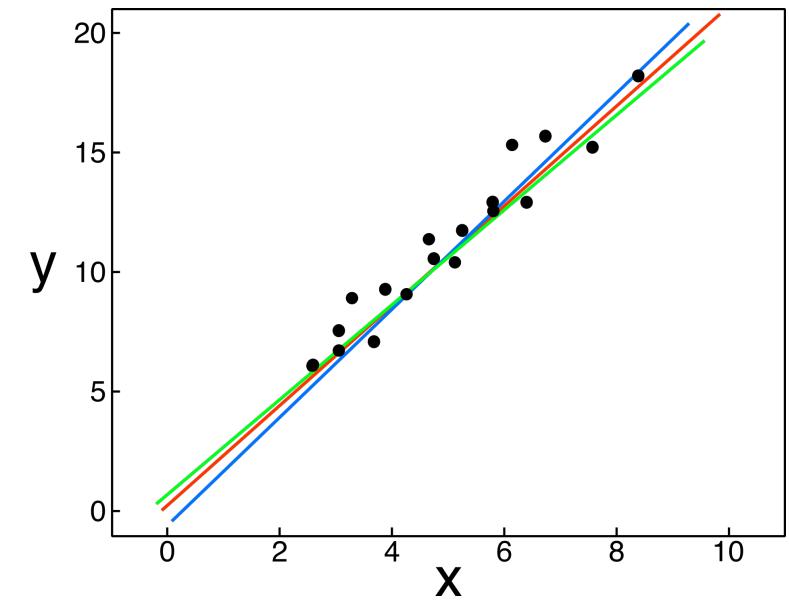
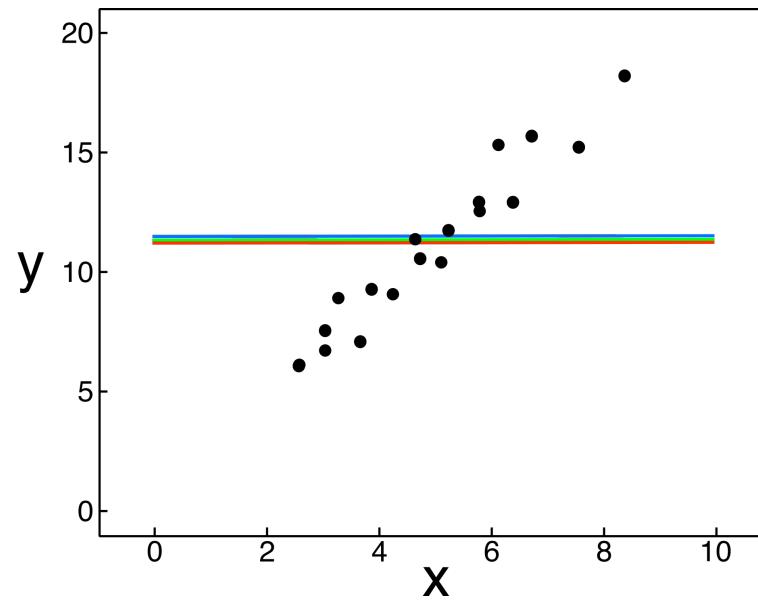
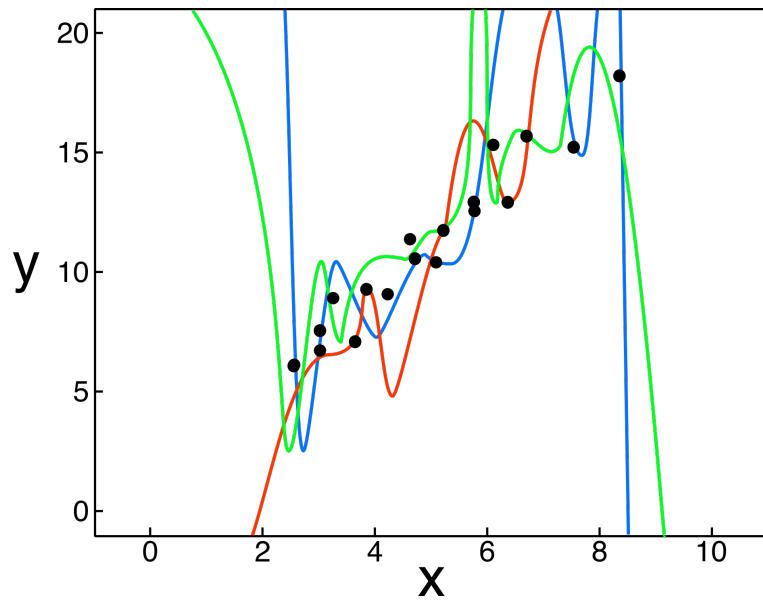
# SESGO Y VARIANZA

---

- El modelo más simple cometerá muchos errores en prácticamente cualquier conjunto de entrenamiento, lo que significa que tiene **un error alto de sesgo**. Sin embargo, cuando evaluamos con el set de evaluación a estos modelos, el resultado siempre es similar. Entonces decimos que tiene **un error de varianza bajo**. *Un sesgo alto y una varianza baja suelen corresponder a un subajuste.*
- Por otro lado, el modelo complejo se adapta perfectamente a cada conjunto de entrenamiento. Tiene **un error de sesgo muy bajo** pero un **error de varianza muy alto** (ya que dos conjuntos de entrenamiento cualesquiera probablemente darían lugar a modelos muy diferentes). *Esto corresponde a un sobreajuste.*

# SESGO Y VARIANZA

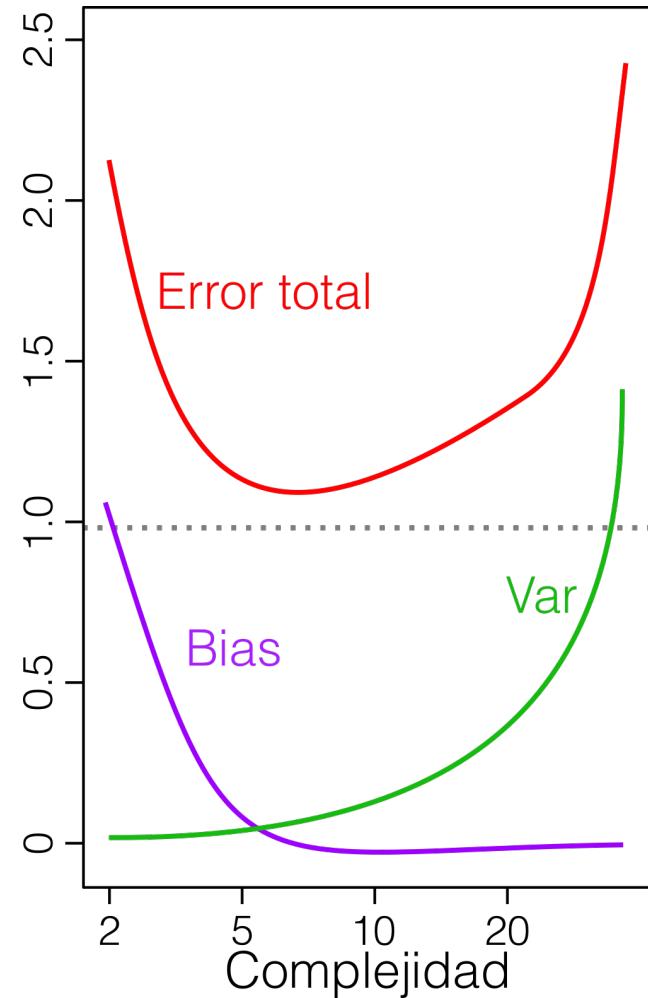
- Si vemos para cada caso, vemos que el modelo lineal, aunque tiene varianza, el error es chico, ya que tiene poca varianza y poco sesgo.



# SESGO Y VARIANZA

*Como regla general,*

- Cuando más complejo es el modelo, la varianza va a aumentar y el sesgo va a disminuir. Cuando aumentamos la complejidad de este, el sesgo tiende a disminuir más rápido de lo que la variabilidad aumenta, disminuyendo el error. Llega un punto en donde el efecto de la variabilidad es apreciable, aumentando el valor del error de nuevo.





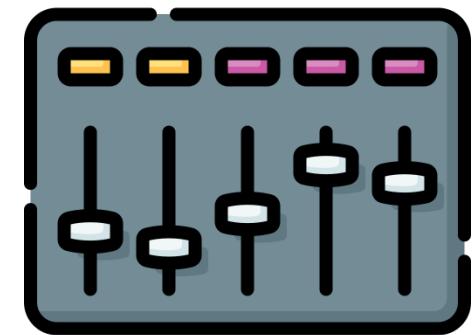
---

# ESTRATEGIAS PARA DISMINUIR EL RIESGO EMPÍRICO

# ESTRATEGIAS

Pensemos ahora a los modelos como cajas grises, todavía no sabemos cómo funcionan, pero tenemos una idea que tienen parámetros, estos son de dos tipos:

- **Parámetros que se entrena**n: Son los parámetros que un modelo aprende cuando lo entrenamos. Principalmente son números como coeficientes de un polinomio o los pesos sinápticos de una red, o variables categóricas en arboles de decisión. Ya vimos que su número debía ser mucho menor que la cantidad de datos de entrenamiento para evitar el sobreajuste.
- **Hiper-parámetros**: Son parámetros *internos* que no dependen de los datos. Estos deben ser definidos previo al entrenar. Por ejemplo, una red neuronal tiene que definirse la función de activación o el orden del polinomio.



Consola iconos creados por [Freepik](#) - Flaticon

# ESTRATEGIAS



Elegir estos hiper-parámetros es un desafío importante, ya que, para tener el mejor modelo, deberíamos saber de ante-mano, pero para saberlo, debemos entrenar.

Entonces lo que se hace es entrenar muchos modelos con diferentes hiperparámetros, y ver cuál es el mejor.

¡Pero si usamos al **set de evaluación**, vamos a encontrar el mejor para este con riesgo de que **no generalizemos!**

---

# ESTRATEGIAS

Entonces debemos usar un tercer dataset que sacamos del de entrenamiento, llamado **set de validación**.

- Este set nos permite evaluar diferentes valores de hiper-parámetros
- Ver si el modelo está sobre-ajustando.
- Evaluar el modelo mientras aprende. En Deep Learning es normal usarlo para evaluar en el medio, el entrenamiento de estas redes es con gradiente descendiente, y es importante ver cada cierta iteración cómo evoluciona.

Con este set evitamos usar el de evaluación y “*“hacer trampa”*”.

---

# ESTRATEGIAS

Esto mejora la generalización, pero es muy sensible a la selección del conjunto de validación. Además, quitamos datos que nos podría haber servido para entrenar.

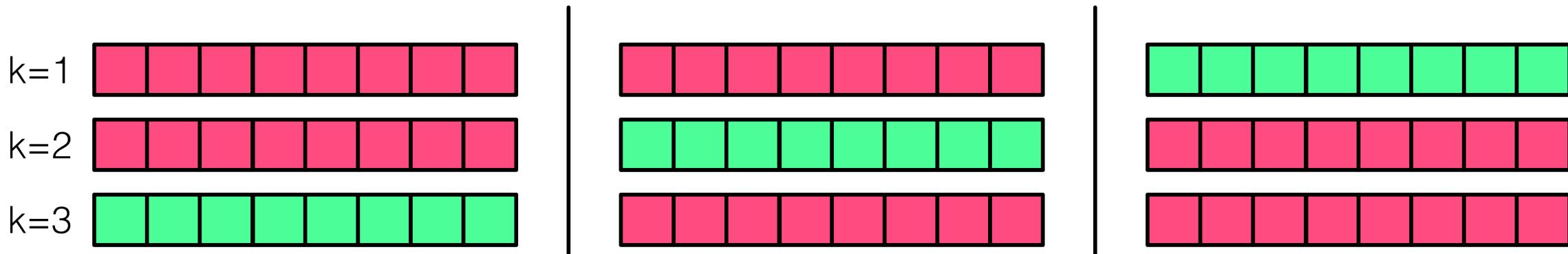
Una estrategia que nos evita esto es usar **Validación cruzada**.

# ESTRATEGIAS

## Validación cruzada

La más conocida es **Validación cruzada K-Fold**.

Los datos de entrenamientos se dividen en  $K$  sub-conjuntos. Con esta separación, realizamos el proceso de validación que vimos previamente, pero  $K$  veces. Cada vez vamos eligiendo set de validación a un conjunto  $K$  y a los restantes  $K-1$  como entrenamiento. Una vez finalizado, los errores medidos se promedian para obtener la efectividad total del modelo.



---

# ESTRATEGIAS

## Validación cruzada K-Fold.

### Beneficios:

- Estrategia sistemática para encontrar los hiperparámetros,
- Nos permite evaluar al menos una vez cada punto de entrenamiento
- Nos permite evaluar problemas de sobreajuste
- Sin tocar el conjunto de testeo.

### Desventaja:

- Es un proceso muy lento. Cada modelo se debe entrenar K veces.

*En procesos lentos y con set de entrenamientos muy grandes, conviene usar un set de validación.*

---

# ESTRATEGIAS

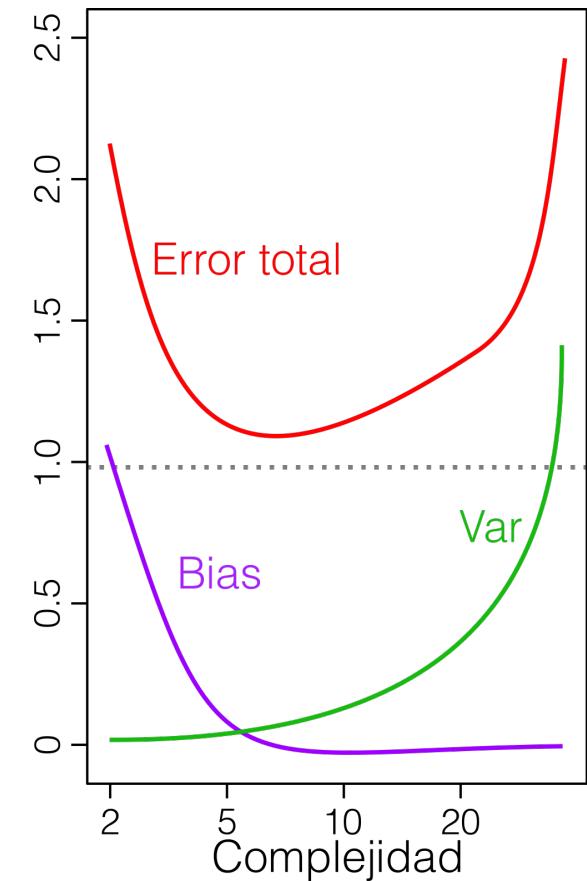
## Regularización

La estrategia anterior versa sobre los datos, pero también podemos trabajar sobre los modelos. Una forma es usando **métodos de regularización**.

La **regularización** es un método que nos permite restringir el proceso de estimación, se usa para evitar un posible sobre ajuste del modelo (overfitting).

La forma en que funciona esto es restringiendo que valores pueden adoptar los parámetros de entrenamiento del modelo o achicándolos.

Lo que buscamos en ese caso, es reducir el **error de varianza**, ya que el modelo no se va a poder mover tan libremente, pero aumentamos el **error de sesgo**, pero de tan suerte que lo que disminuye la varianza es mayor al aumento del sesgo.



# ESTRATEGIAS

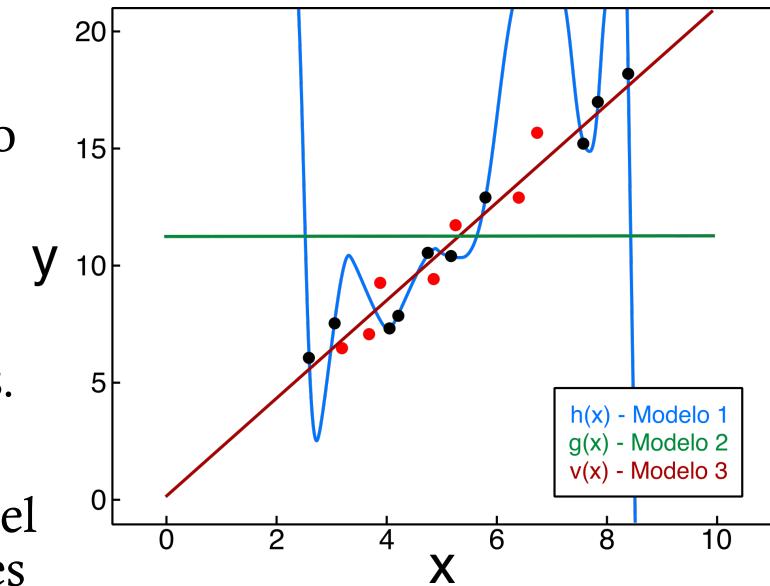
## Regularización

La estrategia anterior versa sobre los datos, pero también podemos trabajar sobre los modelos. Una forma es usando **métodos de regularización**.

La **regularización** es un método que nos permite restringir el proceso de estimación, se usa para evitar un posible sobre ajuste del modelo (overfitting).

La forma en que funciona esto es restringiendo que valores pueden adoptar los parámetros de entrenamiento del modelo o achicándolos.

Lo que buscamos en ese caso, es reducir el **error de varianza**, ya que el modelo no se va a poder mover tan libremente, pero aumentamos el **error de sesgo**, pero de tan suerte que lo que disminuye la varianza es mayor al aumento del sesgo.



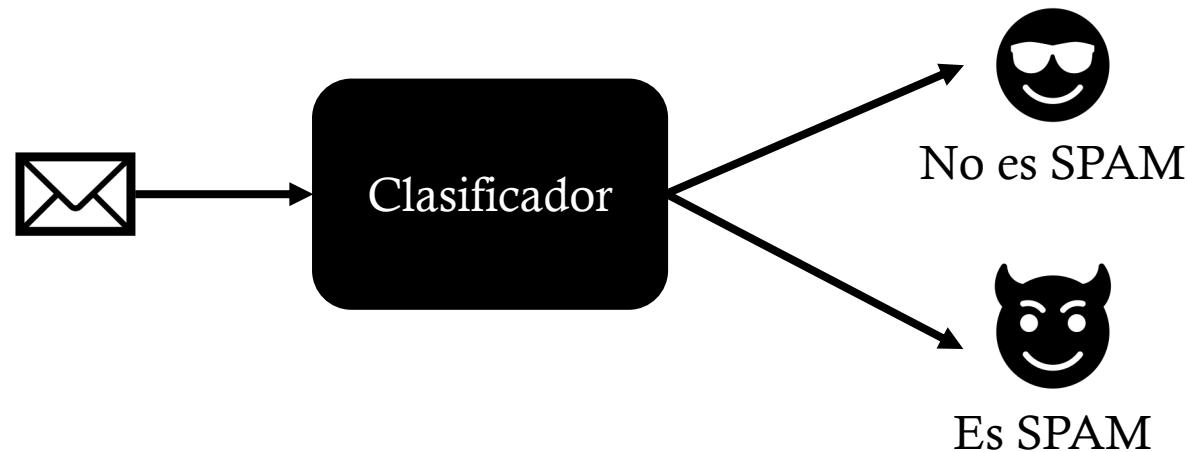


---

# MÉTRICAS DE CLASIFICACIÓN

# MÉTRICAS DE CLASIFICACIÓN

Supongamos que tenemos un modelo de clasificación encargado de medir si un correo es SPAM o no:



¿Como medimos la calidad de este clasificador? ¿Como sé que funciona bien?

---

# MÉTRICAS DE CLASIFICACIÓN

¿Como medimos la calidad de este clasificador? ¿Como sé que funciona bien?

Uno piensa intuitivamente en tasa de aciertos. Pero los **SPAM** son mucho menos que los **no SPAM**, supongamos que tenemos una relación 1 a 1000.

Entonces un modelo que clasifica a todo como **no SPAM**, va a tener una tasa de acierto de:

99.9%

Entonces, ¿seguimos pensando que la tasa de acierto es una buena métrica?

# MÉTRICAS DE CLASIFICACIÓN

Entonces, es importante para saber si el clasificador binario es bueno o malo, entender cómo se puede equivocar.

Supongamos que, si el clasificador dice que es SPAM, entonces la salida es positiva, y si no es negativo, con eso podemos tener los siguientes casos en comparación con el verdadero valor

		Valor verdadero	
		Verdadero positivo (TP)	Falso positivo (FP)
Salida del clasificador	Verdadero		
	Falso		

Esta estructura se llama **matriz de confusión**

---

# MÉTRICAS DE CLASIFICACIÓN

## Matriz de confusión

- **Verdadero positivo:** Es aquellas observaciones que clasificamos como 1 y que realmente eran 1.
- **Verdadero negativo:** Es aquellas observaciones que clasificamos como 0 y que realmente eran 0.
- **Falso positivo:** Es aquellas observaciones que clasificamos como 1 y que realmente eran 0. Este tipo de error se llaman de tipo I.
- **Falso negativo:** Es aquellas observaciones que clasificamos como 0 y que realmente eran 1. Este tipo de error se llaman de tipo II.

---

# MÉTRICAS DE CLASIFICACIÓN

## Matriz de confusión

El desempeño específico de clase también es importante en medicina y biología, donde los términos **sensibilidad** y **especificidad** caracterizan el desempeño de una prueba de detección:

- **Sensibilidad (tasa de verdaderos positivos):** Representa la capacidad del clasificador para detectar todos los casos positivos existentes en los datos.

$$TPR = \frac{TP}{TP + FN}$$

- **Especificidad (Tasa de verdaderos negativos):** Indica la capacidad del clasificador para identificar correctamente los casos negativos.

$$TNR = \frac{TN}{TN + FP}$$

---

# MÉTRICAS DE CLASIFICACIÓN

## Matriz de confusión

Volviendo a nuestro clasificador que dice que todo no es SPAM, tendríamos:

$$\text{Sensibilidad} = 0 \quad \text{Especificidad} = 1$$

La exactitud es la métrica que vimos, la tasa de aciertos:

$$\text{Exactitud} = \frac{TP + TN}{P + N} = 0.999$$

Pero cuando tenemos desbalance de clases conviene calcular la exactitud balanceada:

$$\text{Exactitud balanceada} = \frac{TPR + TNR}{2} = 0.5$$

# MÉTRICAS DE CLASIFICACIÓN

## Matriz de confusión

Volviendo a nuestro clasificador que dice que todo no es SPAM, tendríamos:

$$\text{Sensibilidad} = 0 \quad \text{Especificidad} = 1$$

La exactitud es la métrica que vimos, la tasa de aciertos:

$$\text{Exactitud} = \frac{TP + TN}{P + N} = 0.999$$

Pero cuando tenemos desbalance de clases conviene calcular la exactitud balanceada:

$$\text{Exactitud balanceada} = \frac{TPR + TNR}{2} = \boxed{0.5}$$

Nos dice que el clasificador está adivinando

---

# MÉTRICAS DE CLASIFICACIÓN

## Precisión y recuperación

Otras dos métricas muy importantes son **precisión** y **recuperación**, y estas juegan un rol importante cuando la clase positiva tiene más importancia que la negativa:

- **Precisión**: Se refiere a la proporción de casos positivos identificados correctamente por el clasificador con respecto a todos los casos que el clasificador etiquetó como positivos.

$$Precision = \frac{TP}{TP + FP}$$

- **Recuperación**: Mide la proporción de casos positivos que el clasificador identificó correctamente con respecto a todos los casos positivos reales en los datos. En otras palabras, la recuperación indica la capacidad del clasificador para *recuperar* los casos positivos.

$$Recall = \frac{TP}{TP + FN}$$

---

# MÉTRICAS DE CLASIFICACIÓN

## Precisión y recuperación

En general existe un trade-off entre estas dos métricas, si queremos mejorar una, lo vamos a hacer en pos de empeorar la otra.

Veamos ejemplos donde una métrica es más importante que la otra:

- **Precisión:** En nuestro clasificador de SPAM es mejor esta métrica, ya que queremos que nuestro clasificador cuando diga que es SPAM, realmente este seguro, ya que no queremos que el usuario pierda correos electrónicos importantes.
- **Recuperación:** Un clasificador de imágenes para detectar cáncer, la recuperación es más importante. Es fundamental que el modelo capture la mayor cantidad posible de casos de cáncer para garantizar que los pacientes no se pierdan un diagnóstico temprano y, por lo tanto, un tratamiento oportuno. Incluso si esto significa algunos falsos positivos.

---

# MÉTRICAS DE CLASIFICACIÓN

## Precisión y recuperación

Hay veces que nos importa tener un balance de ambos casos, y para ello podemos usar el puntaje  $F_1$ :

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Y si queremos darle más importancia uno que a otro, podemos usar:

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

Si  $0 < \beta < 1$ , recuperación es más importante,  $\beta > 1$ , precisión es más pesado.

---