



Introducción a Machine Learning

Felipe Bravo y Pablo Badilla

Bienvenida

- Bienvenidos al curso de **Machine Learning** (ML) del diplomado de **Inteligencia Artificial**.
- El curso tiene una componente **teórica** y otra **práctica**.
- La parte teórica estará a cargo de **Felipe Bravo** y la parte práctica de **Pablo Badilla**.
- Para cada clase el primer módulo será de teoría y el segundo será práctico.
- En la parte teórica introduciremos los conceptos y algoritmos más importantes de ML.
- En la parte práctica veremos **casos aplicados** y donde aplicarán los métodos vistos en la teoría usando el lenguaje de programación **python**.

Advertencia

- Entender bien los algoritmos de machine learning (ML) requiere conocimientos en **cálculo, álgebra, probabilidades, estadística, optimización y computación**.
- Estamos muy conscientes que **no todos** tienen estos conocimientos frescos.
- Pero no por eso queremos **omitir** la matemática detrás de los métodos.

Advertencia

- Intentaremos **repasar** los fundamentos matemáticos cuando se pueda.
- Pero a veces **no** será posible repasar todo :(
- Si no entienden algo por favor pregunten.
- No se frustren si en cierto momento no puedan seguir la matemática de la clase, intenten seguir la idea general detrás de los métodos.
- El objetivo del curso es que entiendan la lógica detrás de los métodos, que sean conscientes su complejidad y que sepan **cuándo aplicarlos**.

Evaluaciones

- Las partes teóricas y prácticas se evaluarán con **dos tareas** a tomarse en la cuarta y octava clase práctica (clases 4 y 8 segundo módulo).
- Las tareas tendrán plazo de una semana de entrega.
- Lo ideal es que puedan avanzar todo lo que puedan durante la clase.
- Pueden hacer las tareas en grupos de a 2.
- La nota final se calcula así:

Nota Final: Promedio(T1, T2).

Para aprobar el curso: Nota Final ≥ 4.0 .

Algunas Definiciones

- Machine Learning es la ciencia (y el arte) de programar un computador para que pueda **aprender** a partir de los **datos**.
- Machine Learning es el campo de estudio que da a los computadores la capacidad de **aprender** sin ser programados explícitamente. Arthur Samuel, 1959.
- Se dice que un programa computacional aprende de la experiencia **E** con respecto a alguna tarea **T** y alguna medida de rendimiento **P**, si su rendimiento en **T**, medido por **P**, mejora con la experiencia **E**. Tom Mitchell, 1997.
- Machine Learning y su variante más popular “Deep Learning” son las áreas de mayor impacto en la **Inteligencia Artificial**.

Un Ejemplo Concreto

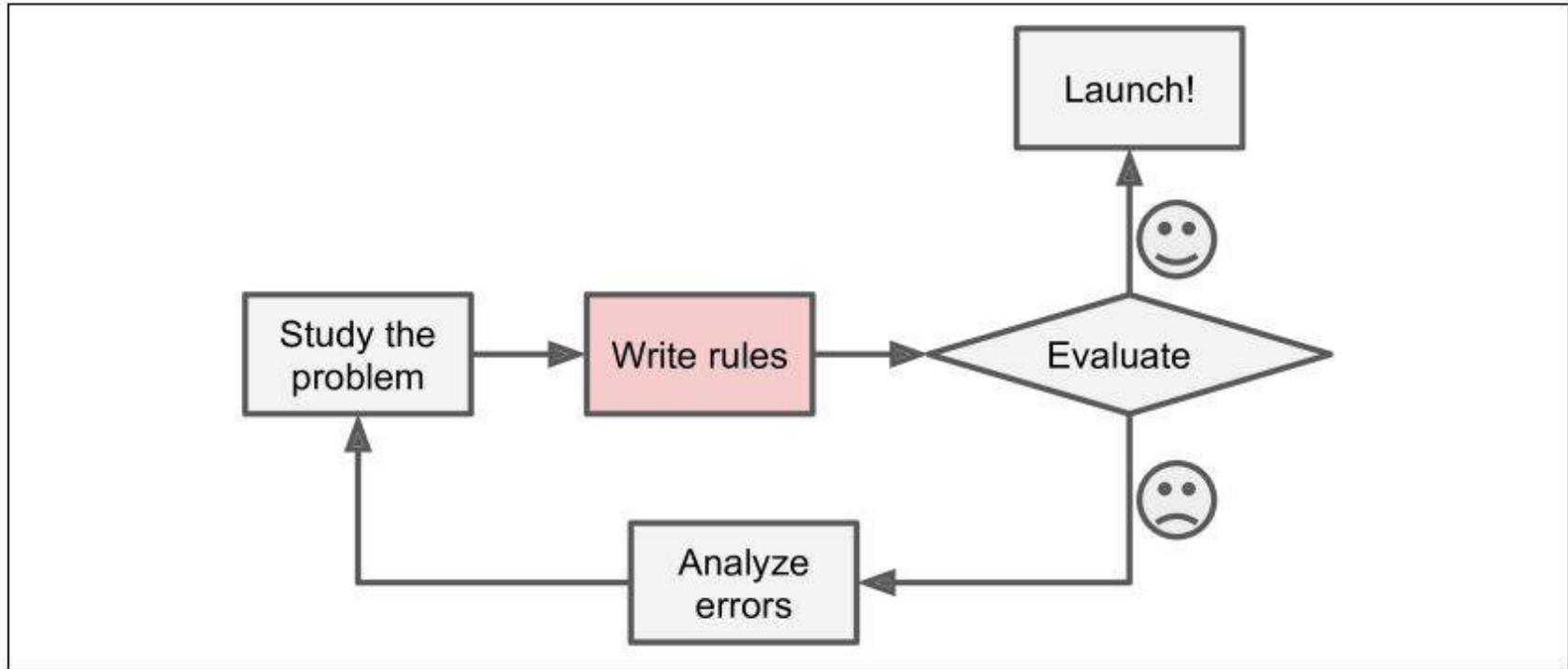


Un Ejemplo Concreto

Pensemos cómo escribiríamos un filtro de spam para su correo electrónico usando técnicas de **programación tradicionales**:

1. Primero se **analizan** las propiedades del spam. Aquí nos daremos cuenta que algunas palabras o frases (como "lotería", "tarjeta de crédito", "gratis" y "viagra") tienden a aparecer mucho en el asunto de los correos spam. Luego se analizan otros **patrones** en el nombre del remitente, el texto del correo electrónico y así sucesivamente.
2. Se diseña un **algoritmo o programa** que detecte cada uno de los **patrones** del paso anterior. El programa **clasificará** los correos electrónicos como spam cuando se detectan varios de estos patrones.
3. Se prueba el programa, y se repiten los pasos 1 y 2 hasta que sea lo suficientemente bueno.

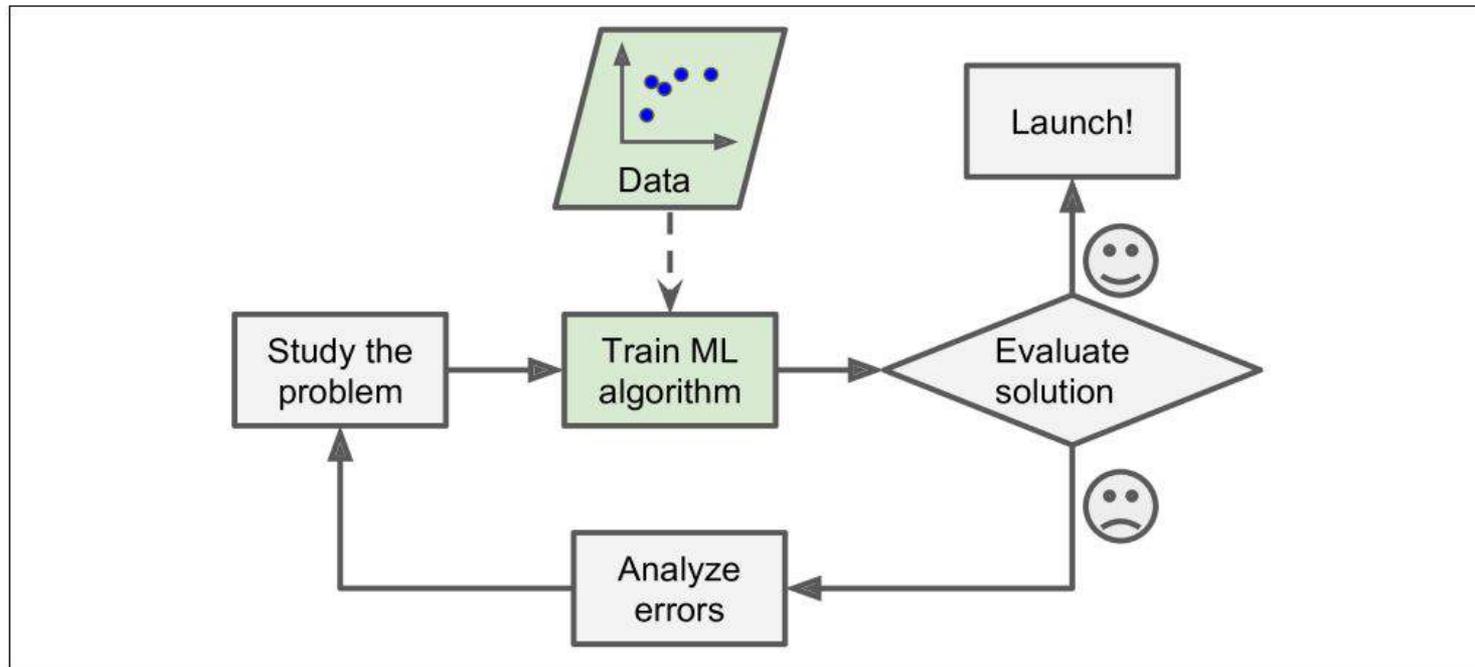
Un Ejemplo Concreto



Como el problema es no trivial, uno terminaría programando una lista muy grande de reglas que serán muy difíciles de mantener.

Un Ejemplo Concreto

- Por otro lado, un filtro de spam basado en técnicas de ML **aprende automáticamente** qué palabras y frases son buenas para predecir spam en un correo.
- Esto se hace **detectando patrones** de palabras que ocurren más frecuentemente en ejemplos de correos **etiquetados** por los usuarios como spam en comparación con ejemplos de correos **etiquetados** como normales.

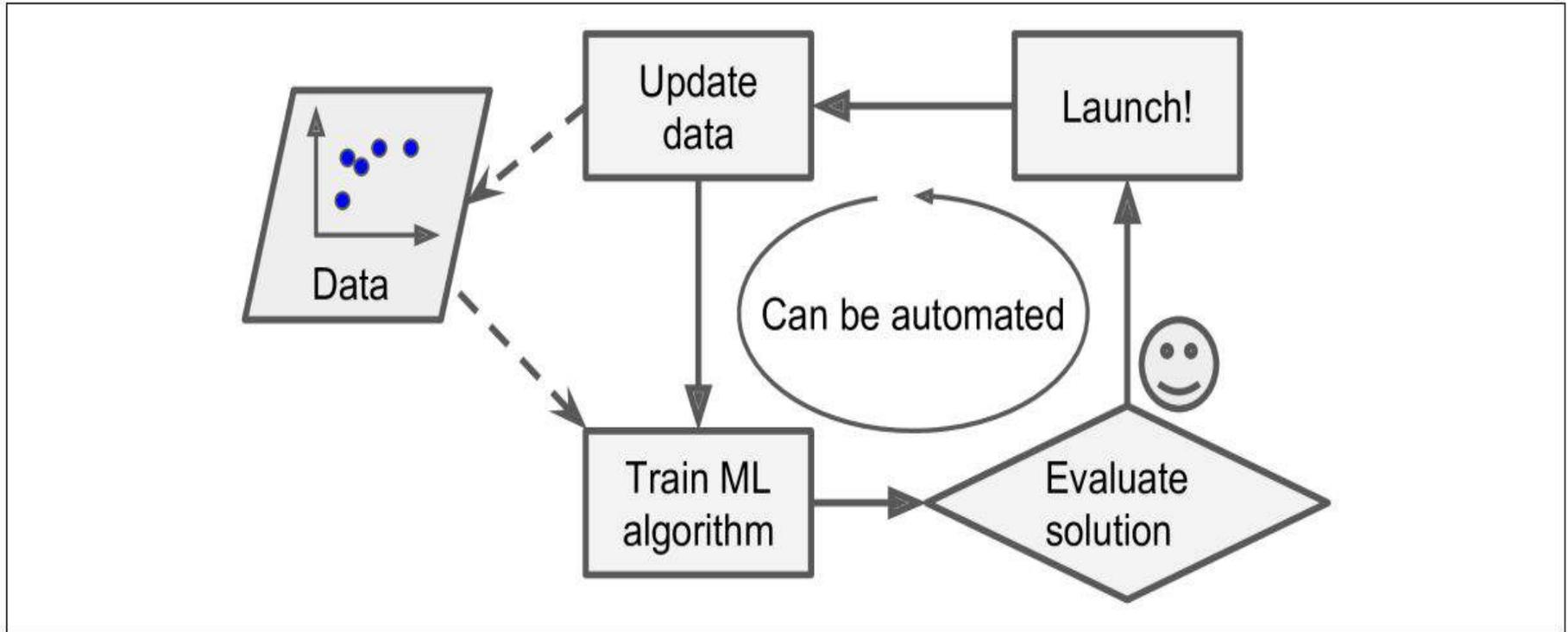


El programa basado en ML es mucho más simple, fácil de mantener y probablemente cometa menos errores.

Un Ejemplo Concreto

- Supongamos ahora que los spammers se dan cuenta que todos sus correos electrónicos que contienen la palabra "lotería" están siendo bloqueados.
- Éstos podrían reemplazar la palabra "lotería" por "**premio**" es sus nuevos spam.
- Un filtro de spam basado técnicas de programación tradicionales necesitaría ser **actualizado** para darse cuenta del cambio.
- Pero si los spammers siguen refinando constantemente sus correos, tendríamos que seguir escribiendo nuevas reglas para siempre.
- En cambio, un filtro de spam basado en técnicas de **Machine Learning** puede seguir aprendiendo sobre las nuevas etiquetas de los usuarios de correo y **adaptarse automáticamente** a que la palabra "premio" se usa en el spam.

Un Ejemplo Concreto



ML permite adaptar el programa al cambio.

¿Cuándo Aplicar ML?

- ML es muy útil para problemas que son **demasiado complejos** para ser resueltos algorítmicamente.
- Ejemplo: reconocimiento de voz, procesar texto, procesar imágenes.
- Consideremos el problema de reconocimiento de voz.



- Supongamos que queremos escribir un programa capaz de distinguir las palabras "one" y "two" en inglés.

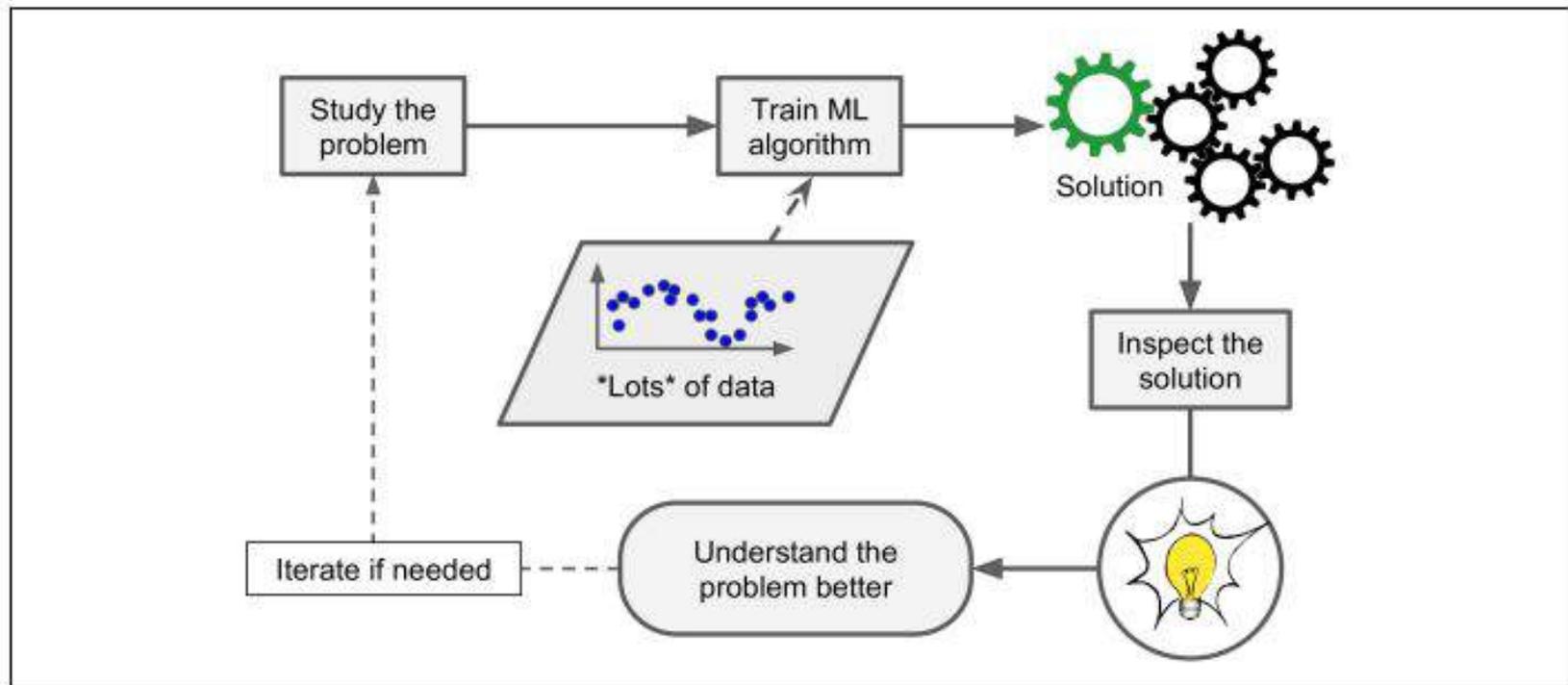
¿Cuándo Aplicar ML?

- Sabemos que la palabra "two" comienza con un sonido de tono alto ("T").
- Entonces nuestro algoritmo podría medir la intensidad del sonido para distinguir entre "one" y "two".
- Obviamente, esta técnica no escalaría a miles de palabras habladas por millones de personas en ambientes ruidosos y en varios de idiomas.
- La mejor solución (al menos hoy en día) para este tipo de problemas es escribir un algoritmo que **aprenda** a partir de los **datos**.
- En este caso los datos serían **muchas grabaciones** de entrenamiento por cada palabra.

¿Cuándo Aplicar ML?

- Machine Learning puede ayudar a los humanos a **aprender** sobre problemas complejos.
- Algunos algoritmos de ML como los **árboles de decisión** o los **modelos lineales** pueden ser **inspeccionados** para ver lo que han aprendido.
- Otras técnicas como las **redes neuronales** son muy difíciles de interpretar.
- Una vez que el filtro de spam ha sido entrenado (con un método interpretable), éste nos puede **revelar** la lista de palabras y frases que mejor predican el spam.
- Esto nos puede indicar correlaciones **insospechadas** o nuevas tendencias que nos ayudarán a **comprender** mejor problema.

¿Cuándo Aplicar ML?



¿Cuándo Aplicar ML?

- La aplicación de técnicas de ML para analizar grandes cantidades de datos puede ayudar a descubrir nuevos patrones o **conocimiento** sobre éstos.
- A este proceso se le llama **minería de datos**.
- **Advertencia:** si ya hicieron un curso de minería de datos se van a repetir el plato con varias cosas.
- Tómenlo como una **oportunidad** para profundizar sus conocimientos.

¿Cuándo Aplicar ML?

En resumen, ML es útil para:

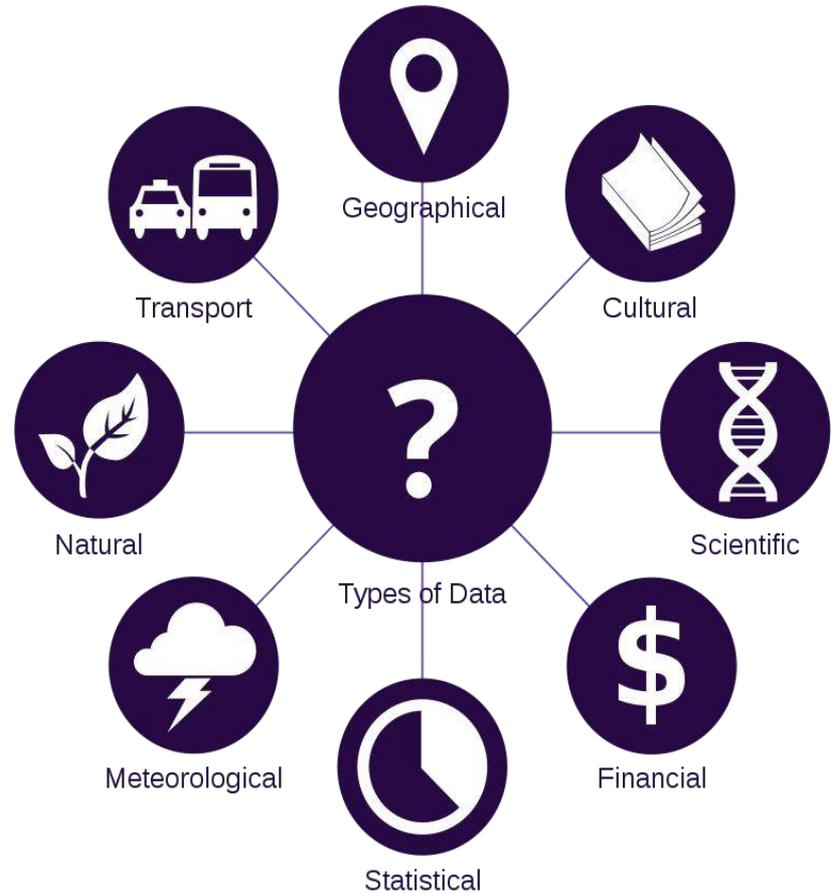
- Problemas en los que las soluciones existentes requieren mucho **ajuste manual** o largas listas de reglas: un algoritmo de ML puede simplificar el código y funcionar mejor.
- Problemas complejos para los que no se puede encontrar una buena **solución algorítmica**.
- Problemas cambiantes: un sistema de ML puede **adaptarse** y aprender de nuevos datos.
- Hacer **minería de datos**: obtener conocimiento sobre problemas complejos y grandes cantidades de datos.

Datos

- Hasta ahora hemos dicho que ML consiste en aprender a partir de **datos**.
- Pero, ¿qué son los datos?
 - Según la OECD: Los datos son **características o información**, generalmente numéricos, que se recogen mediante la observación.
 - Según el Australian Bureau of Statistics: los datos son un conjunto de valores de variables **cualitativas** o **cuantitativas** sobre una o más personas u objetos.

Datos

- Existen muchos tipos de datos complejos como las secuencias de ADN, el texto, las imágenes, el video, el audio, datos espacio-temporales, etc.
- En este curso nos enfocaremos en **datos tabulares**.
- En cursos más avanzados aprenderán a trabajar con datos más complejos: procesamiento de **lenguaje natural**, procesamiento de **imágenes**, etc.



Fuente: Wikipedia

Datos Tabulares

- Un dataset tabular es una colección de **objetos** con sus **atributos**
- Un **atributo** es una propiedad o característica de un objeto
 - Ejemplos: color de ojos, temperatura, etc.
 - Un atributo también se conoce como: variable, campo, característica, dimensión.
- Un **objeto** es una colección de atributos
 - Un objeto también se puede llamar: registro, ejemplo, punto, instancia, observación.
 - Todos los objetos de un dataset tienen la misma cantidad de atributos.

Atributos

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Objetos

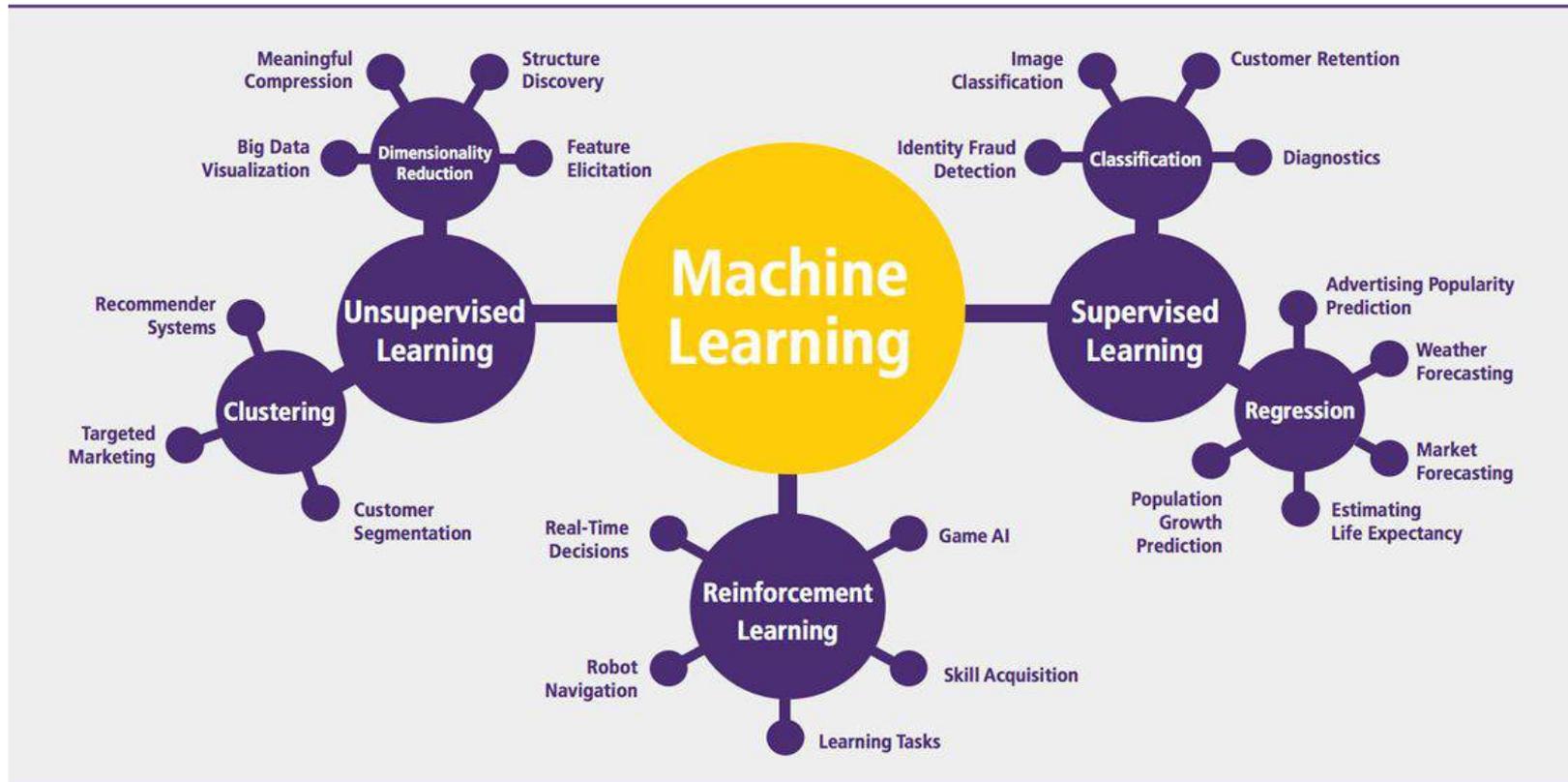
Tipos de Atributos

Existen principalmente dos tipos de atributos: 1) **numéricos o continuos** y 2) **discretos o categóricos**.

1. **Atributos numéricos o continuos:** Tiene números reales o enteros como valores de atributo. Ejemplos: temperatura, altura, peso, etc.
 - a. Los atributos continuos se representan como variables de punto flotante en el computador.
 2. **Atributos discretos o categóricos:** tiene un conjunto finito de valores o categorías. Ejemplos: sexo, estado civil, país, comuna, etc..
 - a. Nota: los atributos binarios son un caso especial de atributos discretos con dos categorías.
- No todos los algoritmos de ML puedan trabajar con estos dos tipos de atributos.
 - A veces tenemos que hacer transformaciones antes de pasar los datos a nuestro algoritmo.

Tipos de Machine Learning

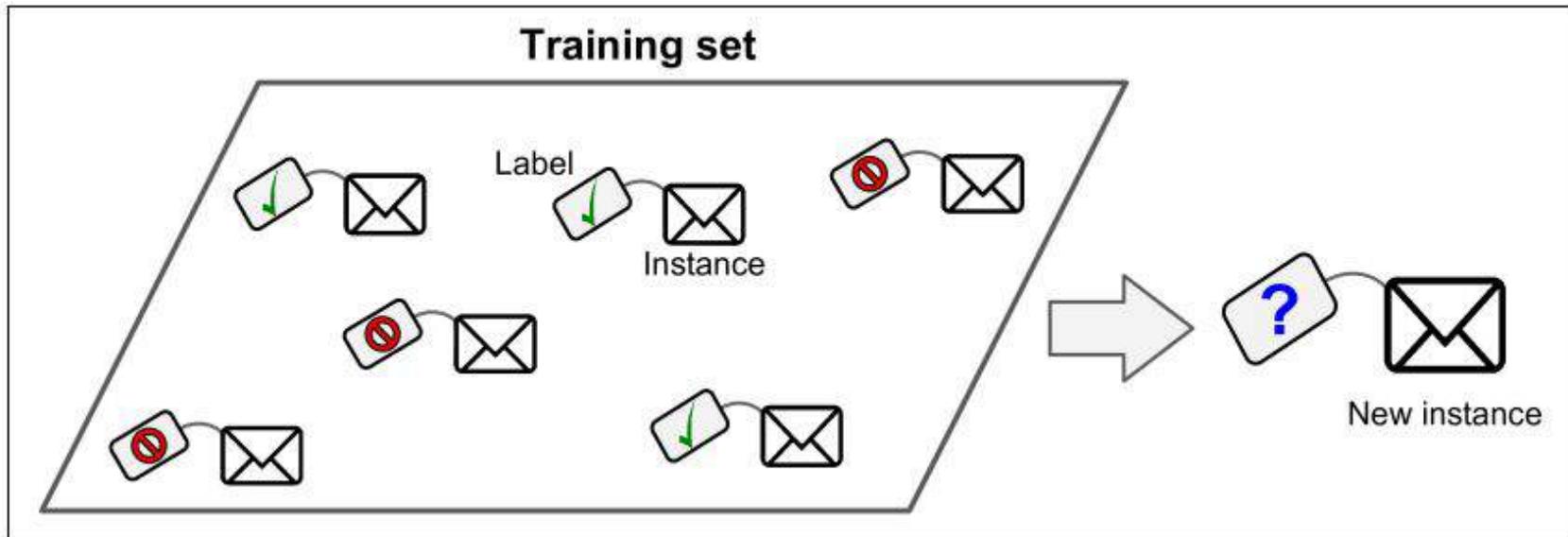
Existen 3 enfoques principales para hacer ML: 1) Aprendizaje Supervisado, 2) Aprendizaje No-Supervisado y 3) Aprendizaje Reforzado.



Fuente: https://www.guru99.com/images/tensorflow/082918_1102_WhatisMachi5.png

Aprendizaje Supervisado

En el **aprendizaje supervisado**, los datos de **entrenamiento** que se le pasan como entrada al **algoritmo de aprendizaje** incluyen las **salidas esperadas**, a las que llamamos **etiquetas**.



- Un dataset de entrenamiento para una tarea de aprendizaje supervisado, en este caso la tarea es la clasificación de spam.

Aprendizaje Supervisado

Dos tareas principales en **aprendizaje supervisado** son la **clasificación** y la **regresión**.

1. Clasificación: mapear los atributos de un objeto a una **variable objetivo** de carácter categórica llamada **clase**.
2. Regresión: predecir una variable objetivo **numérica** a partir de los atributos de un ejemplo.

El grueso de este curso se enfoca en este tipo de técnicas.

La gran limitación para poder aplicarlas es que se requiere un dataset de **entrenamiento** donde los ejemplos tengan su etiqueta correspondiente.

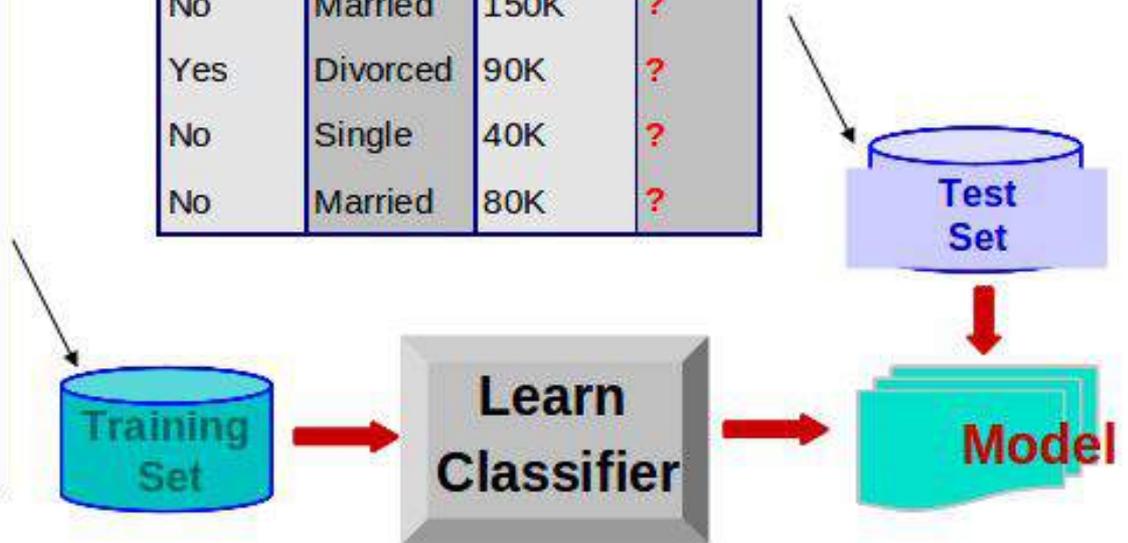
A veces tenemos que **construir o anotar** un dataset de entrenamiento para poder entrenar un modelo.

Proceso de Clasificación

categorical
categorical
continuous
class

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

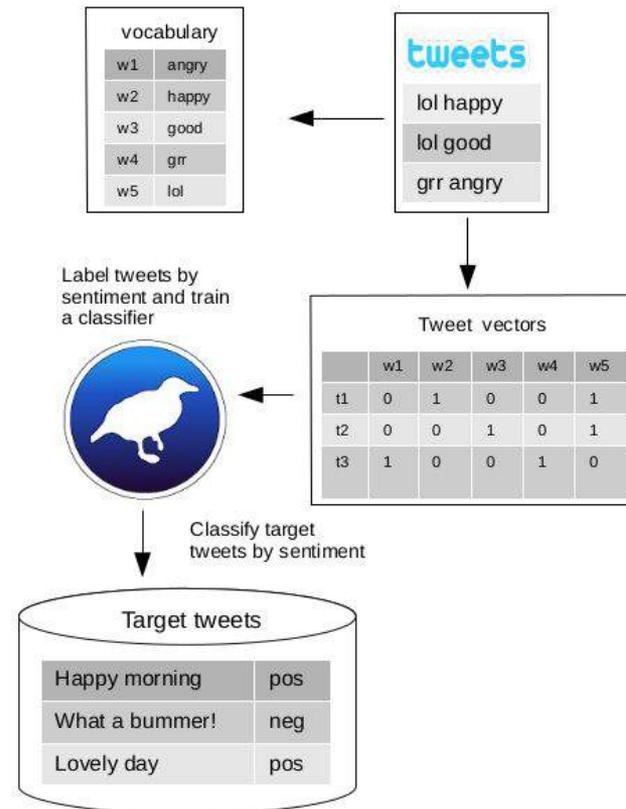
Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



Clasificación: Aplicación 1

Análisis de sentimientos en Twitter.

- Clasificar un tweet a las categorías: **positivo**, **neutral**, **negativo**.



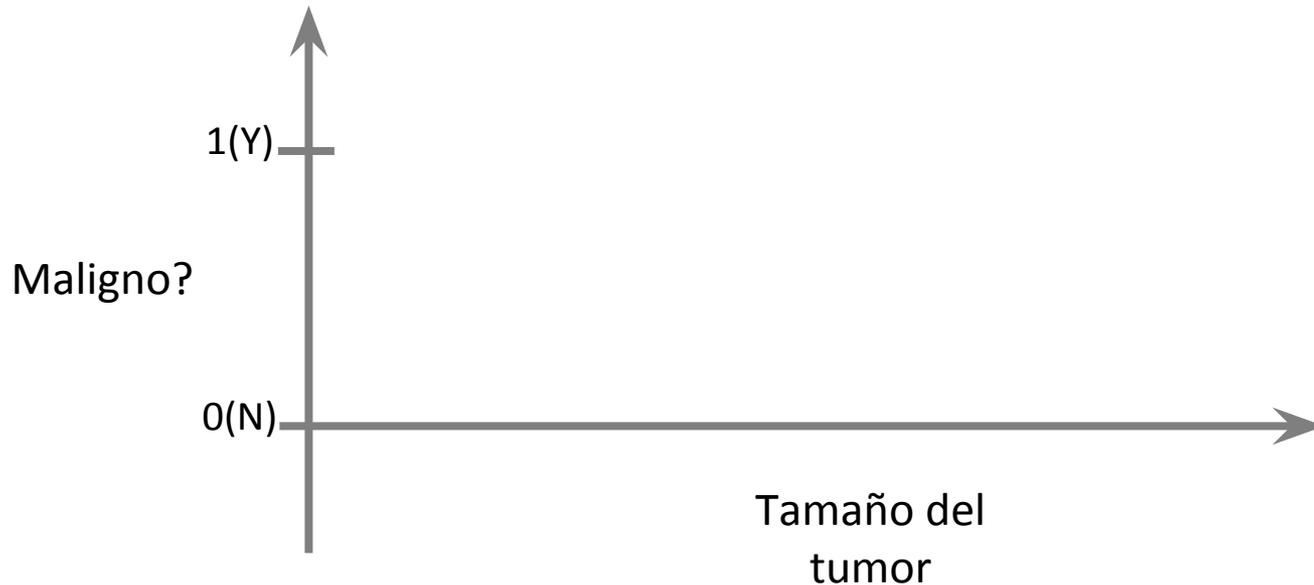
Clasificación: Aplicación 2

- Detección de Fraude: predecir transacciones fraudulentas en el uso de tarjetas de crédito.
- Objetos: las transacciones de tarjetas de crédito.
- Atributos: dónde compra un cliente, que compra, frecuencia de compra, etc.
- Etiquetar las transacciones pasadas como fraude o transacciones válidas.
- Entrenar y utilizar este modelo para detectar fraudes observando transacciones en tarjetas de crédito en una cuenta.



Clasificación: Aplicación 3

Detección de cáncer de mama (maligno, benigno)



Classificación

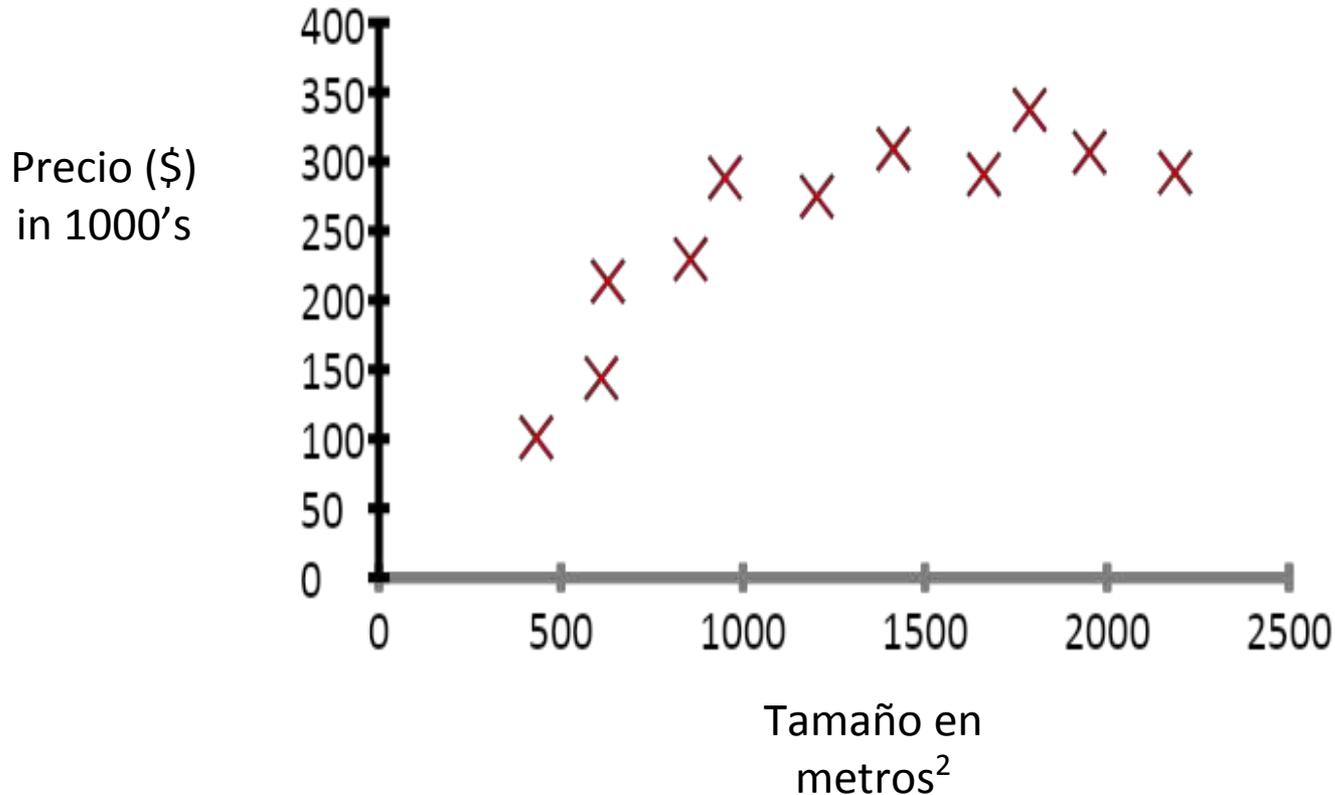
Salida binaria (0
or 1)

Proceso de Regresión



Regresión: Aplicación 1

Predicción de Precio de propiedades.



Aprendizaje Supervisado

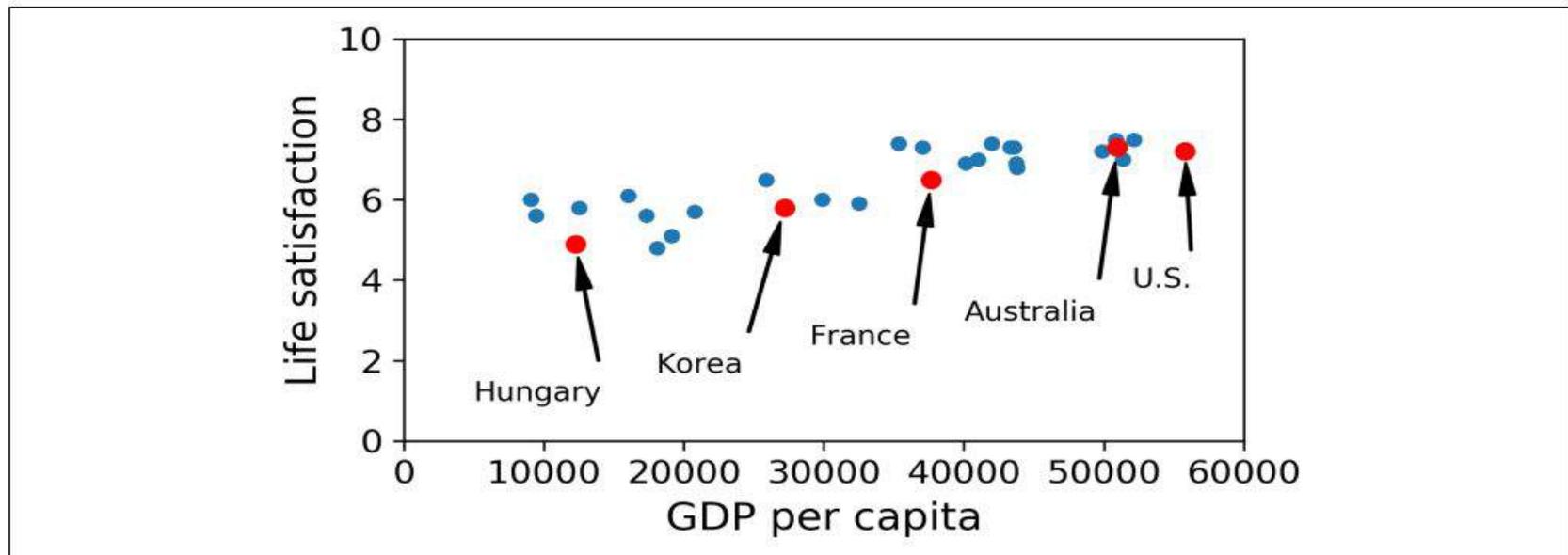
“las respuestas correctas” son dadas

Regresión: predecir una variable continua (precio)

Regresión: Aplicación 2

Predicción de índices de felicidad en base al producto interno.

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2



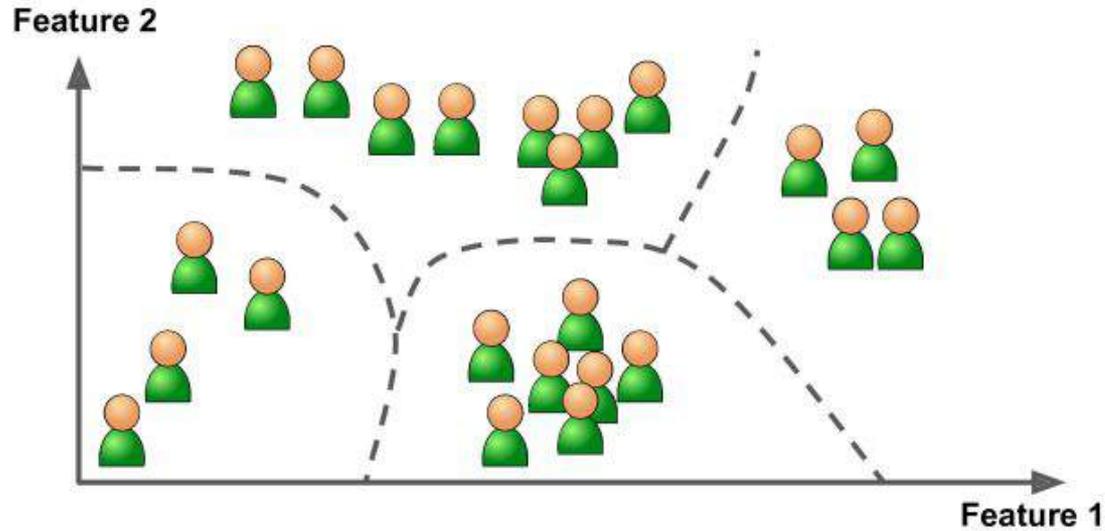
Aprendizaje No-Supervisado

En el aprendizaje no-supervisado los datos de entrenamiento no están etiquetados. El sistema trata de aprender una **organización** de los datos.

1. **Clustering**: encontrar grupos de objetos o **clusters** donde los objetos de un mismo cluster sean **similares** entre sí y **distintos** de los objetos de otros clusters.
2. **Reducción de dimensionalidad**: reducir la cantidad de atributos de mis objetos **combinando** atributos **redundantes** o muy **correlacionados** (ej: el kilometraje de un auto con los años que tiene).
 - a. Reducir dimensionalidad se puede hacer antes de entrenar un algoritmo de ML para que funcione de forma más eficiente.

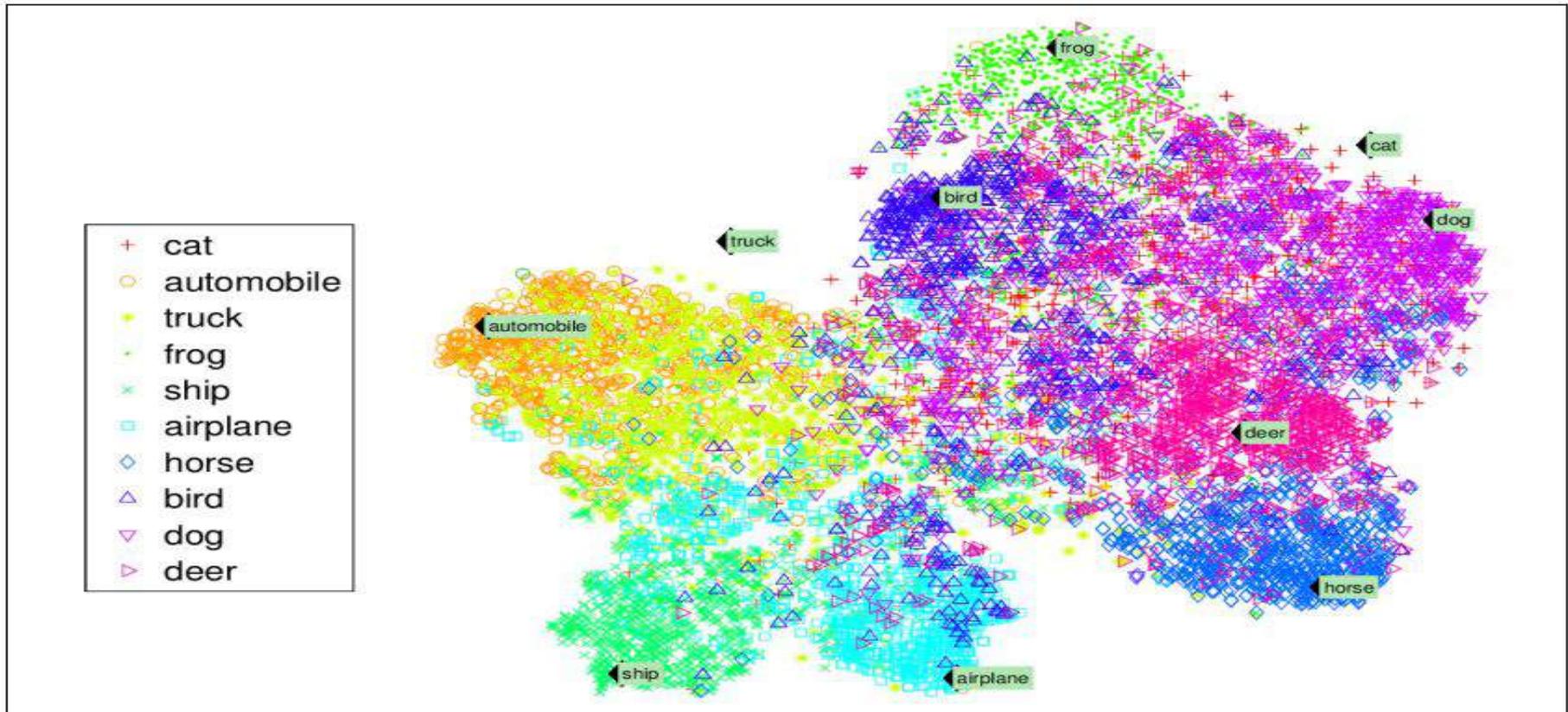
La gran limitación de estas técnicas es que son **difíciles de evaluar**, pues a diferencia del aprendizaje supervisado, aquí no sabemos cual es la respuesta correcta.

Aprendizaje No-Supervisado



El resultado de un proceso de clustering donde se encontraron 4 clusters. Los clusters no eran conocidos antes del entrenamiento.

Aprendizaje No-Supervisado



- Las técnicas de reducción de dimensionalidad como PCA o t-sne permiten proyectar los objetos a dos dimensiones (o atributos) y así poder visualizarlos.
- Esto permite encontrar clusters u otros patrones de manera visual.

Clustering: Aplicación 1

Segmentación de mercado

Encontrar grupos de clientes con comportamientos de consumo parecido (ej: retail, Netflix, Amazon)

- Representar los clientes como objetos donde los atributos son patrones de compra más información demográfica (sexo, edad).
- Los clusters ayudan a enfocar la publicidad y las recomendaciones.



Clustering: Aplicación 2

Clustering de documentos

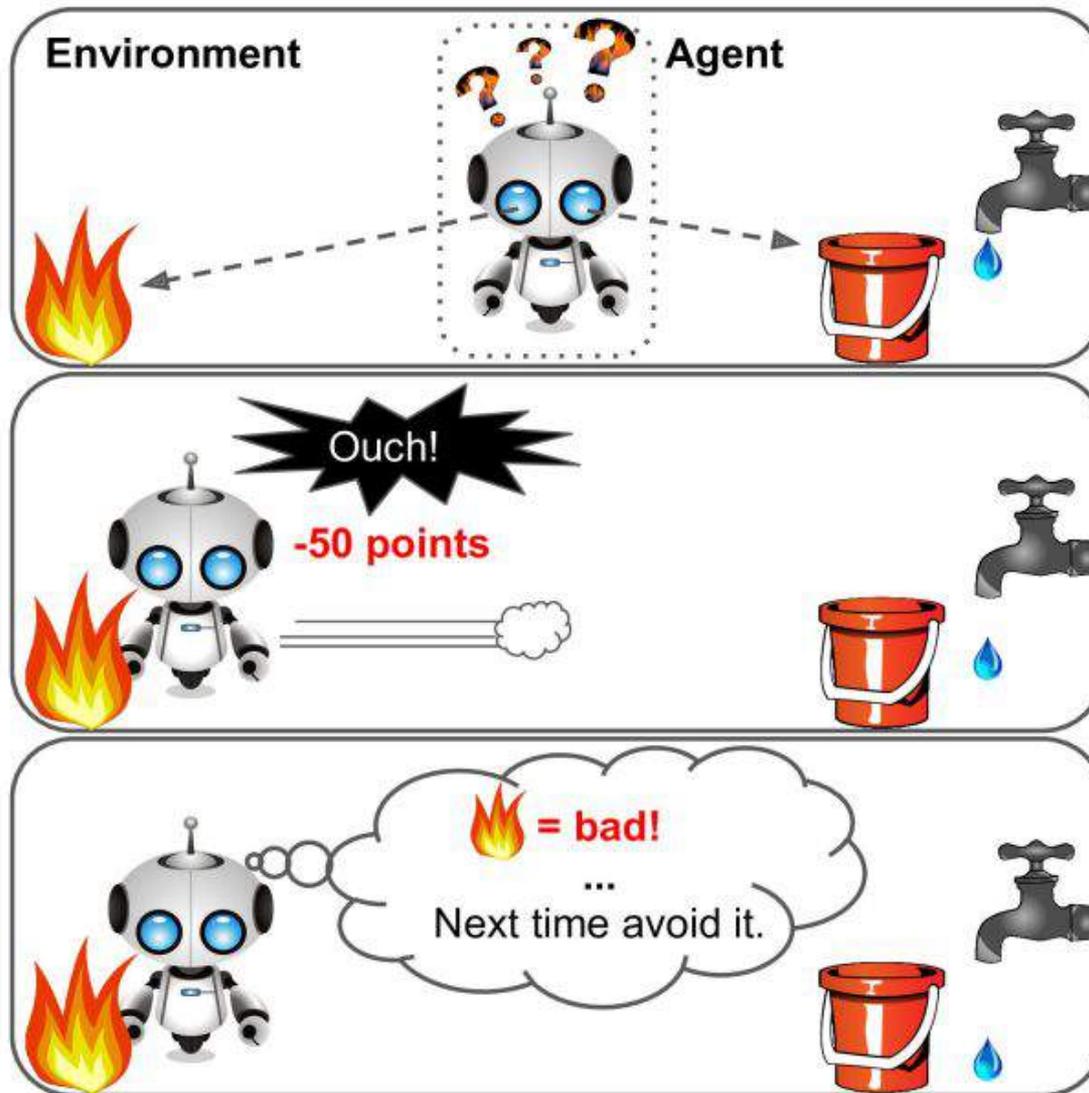
- Para una colección grande de documentos (bibliotecas, reclamos, redes sociales) encontrar grupos de documentos u oraciones que con temática similar.
- Se representan los documentos usando las palabras como atributos y se usa una medida de distancia.

Category	Total Articles	Correctly Placed
Financial	555	364
Foreign	341	260
National	273	36
Metro	943	746
Sports	738	573
Entertainment	354	278

Aprendizaje Reforzado (reinforcement learning)

- El aprendizaje reforzado (RL) es muy distinto a los otros dos enfoques (supervisado y no-supervisado).
- Aquí se tiene un **agente** (ej: un robot) que evalúa su ambiente (ej: su campo visual) y en base a eso realiza una acción (ej: moverse a la izquierda).
- El agente puede recibir **recompensas** o **penalizaciones** dependiendo si hizo algo bueno o malo (ej: es bueno subir una escalera, es malo caerse de un precipicio).
- Luego debe aprender por sí mismo cuál es la mejor estrategia o **policy** para maximizar su recompensa (y minimizar sus penalizaciones) a lo largo del tiempo.
- La **policy** aprendida define las acciones que el agente escogerá cuando se encuentra en una situación determinada.
- **No** veremos RL en este curso.

Aprendizaje Reforzado (reinforcement learning)

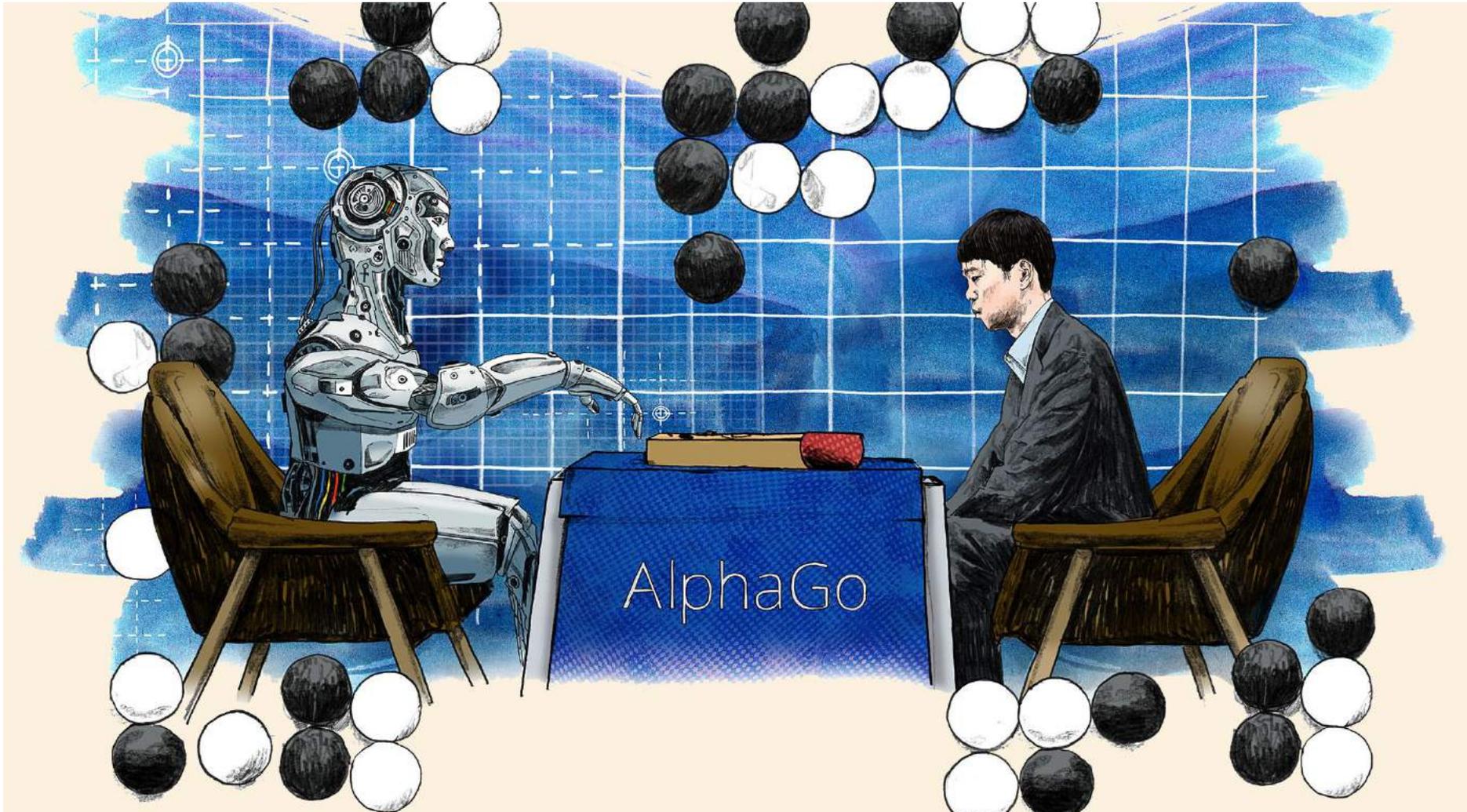


- 1 Observe
- 2 Select action using policy
- 3 Action!
- 4 Get reward or penalty
- 5 Update policy (learning step)
- 6 Iterate until an optimal policy is found

Aprendizaje Reforzado Aplicaciones

- Muchos robots implementan algoritmos de aprendizaje reforzado para aprender a caminar.
- **AlphaGo** de DeepMind venció al campeón mundial Ke Jie en el juego de Go el año 2017.
- El agente aprendió su **policy** ganadora analizando millones de juegos y jugando muchos juegos **contra sí mismo**.
- Durante los juegos contra el campeón, AlphaGo sólo utilizó la **policy** que había aprendido.

Aprendizaje Reforzado Aplicaciones



Kaggle

- Kaggle: (<https://www.kaggle.com/>) es un sitio web que publica **competencias** de ML.
- Algunas competencias tienen recompensas monetarias para los sistemas ganadores.

kaggle

All Competitions

Active	Completed	InClass	All Categories ▾	Default Sort ▾
	OSIC Pulmonary Fibrosis Progression			\$55,000
	Predict lung function decline			
	Featured • a month to go • Code Competition • 1208 Teams			
	Lyft Motion Prediction for Autonomous Vehicles			\$30,000
	Build motion prediction models for self-driving vehicles			
	Featured • 3 months to go • Code Competition • 93 Teams			
	Cornell Birdcall Identification			\$25,000
	Build tools for bird population monitoring			
	Research • 19 days to go • Code Competition • 1096 Teams			
	Google Landmark Recognition 2020			\$25,000
	Label famous (and not-so-famous) landmarks in images			
	Research • a month to go • Code Competition • 427 Teams			

Temario

En este curso veremos los siguientes conceptos.

1. Aprendizaje Supervisado
 - a. Clasificación - Framework y Evaluación.
 - b. Árboles de decisión
 - c. KNN
 - d. Naive Bayes
 - e. Support Vector Machines
 - f. Regresiones Lineales
 - g. Redes Neuronales.
 - h. Selección de Atributos.
2. Aprendizaje No-Supervisado
 - a. K-means
 - b. Clustering Jerárquico
 - c. DBSCAN
 - d. PCA

Repaso Matemático

Álgebra Lineal

En ML usamos las siguientes representaciones algebraicas para nuestros objetos y sus atributos: **escalares**, **vectores** y **matrices**.

- **Escalares**: un escalar es simplemente un número como 7.56. El valor de un atributo numérico para un objeto se representa por un escalar.
- **Vectores**: un vector es un arreglo ordenado de escalares $\mathbf{x} = [x_1, x_2, \dots, x_n]$ donde x_i es el i -ésimo elemento de \mathbf{x} .
 - a. En ML un objeto de n **atributos** numéricos puede ser representado por un vector de n dimensiones.
 - b. El **producto punto** entre dos vectores $\mathbf{a} \cdot \mathbf{b}$ es la suma de la multiplicación de todos sus elementos:

$$(a_1, a_2, a_3, \dots, a_n) \cdot (b_1, b_2, b_3, \dots, b_n) = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = \sum a_i \cdot b_i$$

Álgebra Lineal

Norma y distancia euclídeana:

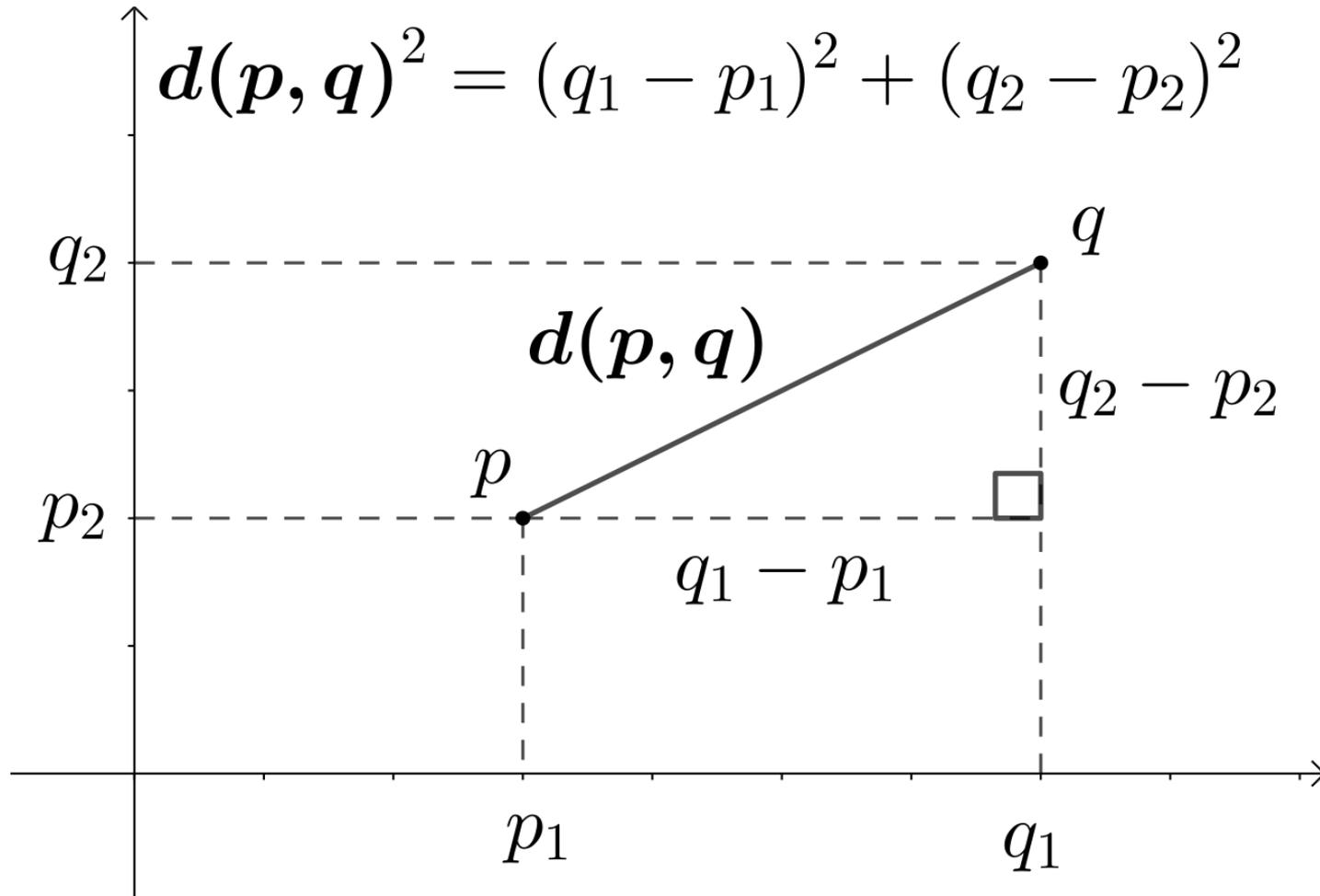
- La norma euclídeana de un vector $\|\mathbf{v}\|_2$ es el largo del vector en un espacio euclídeano (piensen en pitágoras).

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^n v_i^2}$$

- Luego, la distancia euclídeana nos permite calcular que tan lejos están dos vectores \mathbf{x} e \mathbf{y} .

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2},$$

Álgebra Lineal



Distancia Euclideana en \mathbb{R}^2 . Fuente: Wikipedia

Álgebra Lineal

Matrices

- Una matriz es un arreglo de dos dimensiones, entonces cada elemento se identifica por dos índices en vez de uno.
- Ejemplo: sea A una matriz de dos filas y dos columnas ($A \in \mathbb{R}^{2 \times 2}$)

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

- Un dataset de n objetos y m atributos numéricos se puede representar por una matriz de $n \times m$.
- El i -ésimo ejemplo de un dataset se representa como el vector de la i -ésima fila de su matriz correspondiente.
- Un vector puede ser visto como una matriz de una sola columna.

Álgebra Lineal

Matrices

- La transpuesta de una matriz A^T , corresponde a una copia de la matriz donde se intercambian las filas por las columnas.


$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

- Podemos sumar dos matrices, siempre y cuando éstas tengan las mismas dimensiones, sumando sus elementos correspondientes: $C = A + B$ donde $C_{i,j} = A_{i,j} + B_{i,j}$. Ejemplo:

$$\begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 7 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 \\ 1+7 & 0+5 \\ 1+2 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 8 & 5 \\ 3 & 3 \end{bmatrix}$$

Álgebra Lineal

Matrices

- También podemos sumar un escalar a una matriz o multiplicar una matriz por un escalar, simplemente realizando esa operación en cada elemento de una matriz: $D = a*B + c$ donde $D_{i,j} = a*B_{i,j} + c$
- Podemos multiplicar dos matrices A y B ($C=A*B$) siempre y cuando el número de filas de A sea igual al número de columnas de B. El valor de C_{ij} es igual al producto punto de la i-ésima fila de A por la j-ésima columna de B ($A_{i:} \cdot B_{:j}$).

Fuente:

<https://www.javabratman.com/wp-content/uploads/MatrixMultiplicationExample.png>

$$\begin{array}{ccc} & M & \\ \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} & \times & \begin{bmatrix} 3 & 6 & 9 \\ 4 & 7 & 10 \\ 5 & 8 & 11 \end{bmatrix} & = & \begin{array}{ccc} & M \times N & \\ \begin{bmatrix} 26 & 44 & 62 \\ 62 & 107 & 152 \end{bmatrix} & & \end{array} \end{array}$$

[[1X3)+(2X4)+(3X5)] [(1X6)+(2X7)+(3X8)] [(1X9)+(2X10)+(3X11)]
[[4X3)+(5X4)+(6X5)] [(4X6)+(5X7)+(6X8)] [(4X9)+(5X10)+(6X11)]

Matrix multiplication example

Copyright © JavaBrahman.com, all rights reserved.



Álgebra Lineal

Matrices

- Una matriz cuadrada es una matriz que tiene el mismo número de filas y columnas.
- Una matriz cuadrada muy particular es la matriz identidad I que tiene 1s en la diagonal y ceros en todas las otras celdas.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

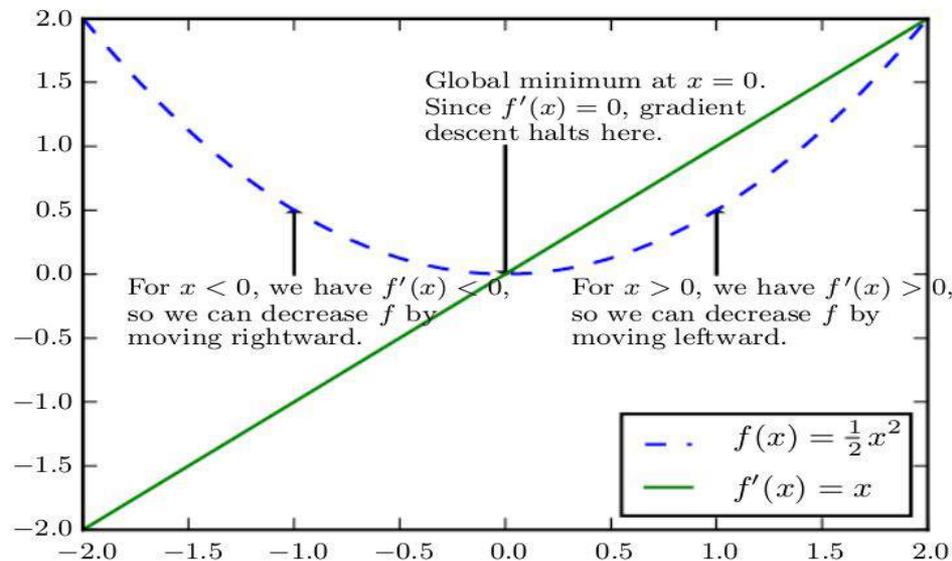
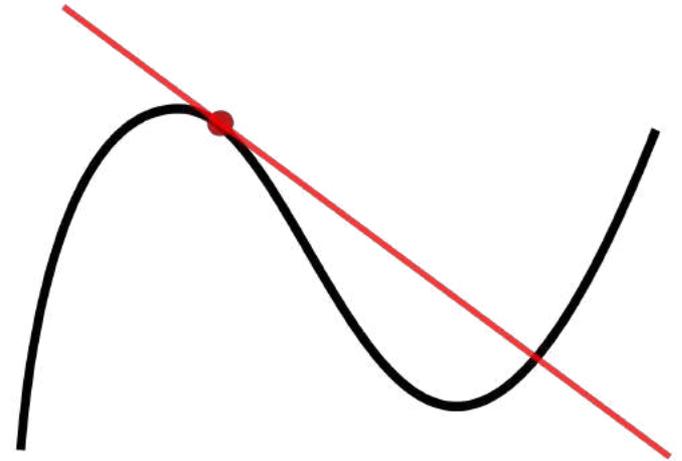
- La inversa de una matriz A se denota como A^{-1} y cumple con la propiedad que $A \cdot A^{-1} = I$

Optimización

La **optimización** es una metodología para encontrar el valor **máximo** o **mínimo** de una función matemática.

Muchos métodos de ML se plantean como problemas de optimización.

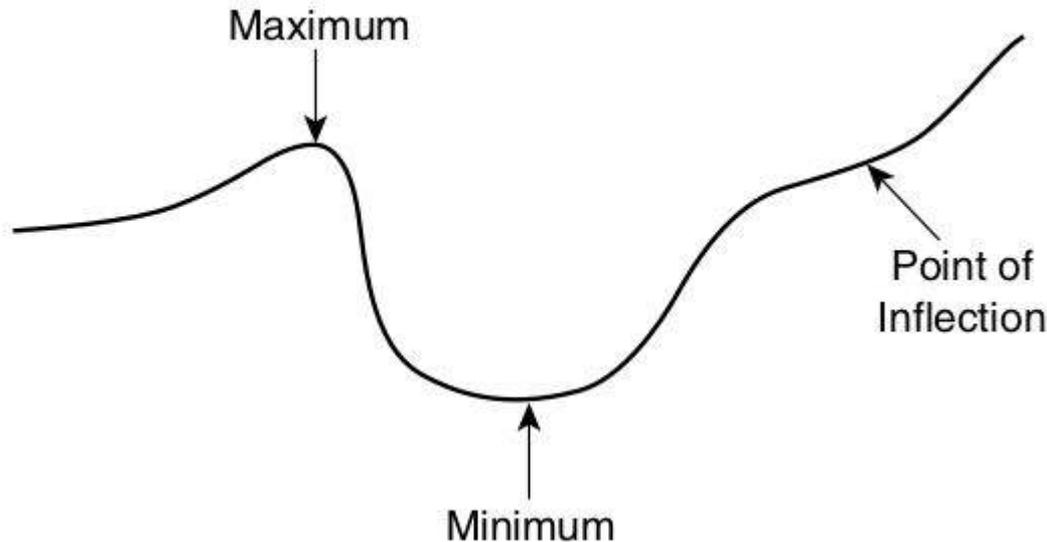
- Supongamos que tenemos una función $y = f(x)$, donde tanto x como y son números reales.
- La derivada de esta función se escribe así $f'(x)$ o así: $\frac{dy}{dx}$
- La derivada $f'(x)$ nos entrega el valor de la pendiente de $f(x)$ en el punto x .



Optimización

La derivada de una función es muy útil para encontrar valores mínimos o máximos.

- Los puntos en que la derivada de una función vale cero ($f'(x)=0$) se conocen como puntos críticos: **máximo**, **mínimo** o punto de **inflexión** (o punto silla).

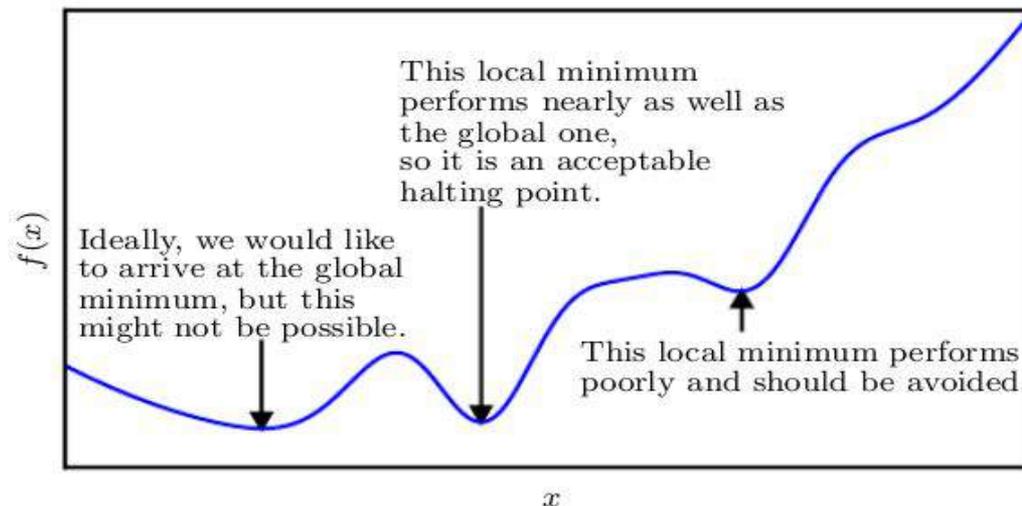


Optimización

- Para distinguir entre estos tres tipos de puntos críticos es necesario analizar la **segunda derivada** de la función $f''(x)$ (la derivada de la derivada) que nos da información sobre la curvatura de la función.

$$\frac{d^2 f}{dx^2}$$

- Otra gran dificultad al optimizar funciones es que a veces los puntos críticos pueden corresponder a mínimos o máximos **locales**.



Optimización

- ¿Cómo optimizamos funciones con múltiples inputs?

$$f(x_1, x_2, x_3) = 2 * x_1 + x_2^2 - 5 * x_3$$

- La **derivada parcial** $\frac{\partial}{\partial x_i} f(\mathbf{x})$ mide cómo cambia f sólo cuando hacemos un cambio en x_i .

$$\frac{\partial f}{\partial x_1} = 2, \frac{\partial f}{\partial x_2} = 2 * x_2, \frac{\partial f}{\partial x_3} = -5$$

- El **gradiente** ∇_f es un vector con todas las derivadas parciales de una función.

$$\nabla_f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \right]$$

- Para encontrar puntos críticos en funciones de varios inputs tenemos que encontrar los valores de x donde el gradiente vale **cero**.
- Para distinguir entre máximos, mínimos y puntos silla tenemos que recurrir al **Hessiano**, que es una matriz con todas las segundas derivadas.

Optimización

- ¿Qué pasa cuando le agregamos restricciones al problema?
- Existen dos tipos de restricciones: 1) restricciones de igualdad y 2) restricciones de desigualdad.

Restricciones de igualdad

- Supongamos que queremos encontrar el mínimo de $f(x_1, x_2, \dots, x_d)$ sujeto a las siguientes p restricciones de igualdad:

$$g_i(\mathbf{x}) = 0, \quad i = 1, 2, \dots, p.$$

- Cada restricción de igualdad es una función del tipo $g(x) = 2 \cdot x_1 + 3 \cdot x_2 - 3 = 0$
- Esto se puede resolver usando un método llamado **multiplicadores de Lagrange**.

Optimización

Multiplicadores de Lagrange

1. Definimos el Lagrangiano $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^p \lambda_i g_i(\mathbf{x})$
donde λ_i es una variable adicional llamada **multiplicador de Lagrange**.
2. Derivamos el Lagrangiano respecto a \mathbf{x} y $\boldsymbol{\lambda}$, igualamos a cero y despejamos el sistema de ecuaciones.

$$\frac{\partial L}{\partial x_i} = 0, \quad \forall i = 1, 2, \dots, d$$

$$\frac{\partial L}{\partial \lambda_i} = 0, \quad \forall i = 1, 2, \dots, p.$$

La solución óptima del Lagrangiano corresponde al óptimo de $f(\mathbf{x})$ que satisface las restricciones de igualdad.

Optimización

Restricciones de desigualdad:

- ¿Qué hacemos cuando tenemos restricciones de desigualdad del tipo $h_i(\mathbf{x}) \leq 0$?
Por ejemplo $x_1 + x_2 \leq 0$.
- Se formula entonces un problema optimización con restricciones como minimizar $f(x_1, x_2, \dots, x_n)$ sujeto a las siguientes q restricciones de desigualdad:

$$h_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, q.$$

- El método para resolver este problema es bastante similar al método de Lagrange descrito anteriormente.
- Sin embargo, las restricciones de la desigualdad plantean condiciones adicionales al problema de optimización.

Optimización

- El problema de optimización con restricciones de desigualdad se formula con el siguiente Lagrangiano:

$$L = f(\mathbf{x}) + \sum_{i=1}^q \lambda_i h_i(\mathbf{x})$$

- Una solución óptima de este problema debe satisfacer las condiciones Karush-Kuhn-Tucker (KKT):

$$\begin{aligned} \frac{\partial L}{\partial x_i} &= 0, \quad \forall i = 1, 2, \dots, d \\ h_i(\mathbf{x}) &\leq 0, \quad \forall i = 1, 2, \dots, q \\ \lambda_i &\geq 0, \quad \forall i = 1, 2, \dots, q \\ \lambda_i h_i(\mathbf{x}) &= 0, \quad \forall i = 1, 2, \dots, q. \end{aligned}$$

Optimización

- Notemos ahora que los multiplicadores de Lagrange no pueden ser negativos.
- Las restricciones KKT nos obligan verificar que un óptimo satisfaga todas las condiciones.
- Eso puede ser muy difícil de lograr analíticamente, especialmente si se tienen muchas restricciones.
- Por lo general recurrimos a métodos numéricos como la programación lineal y la programación cuadrática.

Bibliografía

1. Aurélien Géron. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.
2. Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar. Introduction to Data Mining (Second Edition).
3. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.



dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

f @ in / DCCUCHILE