Parte 1: El framework formal del machine learning

En esta sesión vamos a aprender a hablar *machine learning*. Un algoritmo de machine learning es un algoritmo que es capaz de **aprender** a partir de datos. Definimos **aprender** de la siguiente manera:

Un algoritmo se dice que aprende de la experiencia E con respecto a una clase de tareas T y una medida de rendimiento R, si su rendimiento en las tareas en T, medido por R, mejora con la experiencia E.

En otras palabras, un algoritmo de machine learning es aquel que mejora en su tarea T (sea cual sea) a medida que obtiene una experiencia E, y la forma en la que medimos el hecho de que mejore o no es a través de una medida de rendimiento R. Por ejemplo, para un algoritmo que aprende a identificar si en una imagen hay un gato o no, tendríamos que

- La tarea T es dada una imagén, devolver un 1 si la imagen contiene un gatito, y 0 si no.
- La experiencia E es un *conjunto de imágenes*, algunas con gatos y otras sin gatos, mediante el cual el algoritmo aprende a discernir (de alguna forma) si en la imagen hay un gato o no.
- El rendimiento R es una medida que indica que tan bien el algoritmo puede identificar si en la imagen hay un gato o no. Por ejemplo, calculando la proporción entre imágenes bien clasificadas e imágenes mal clasificadas. Este rendimiento debería mejorar a medida que el algoritmo ve más imágenes.

Echemos un vistazo más profundo a cada uno de estos conceptos

La tarea, T

La tarea es lo que queremos que el algoritmo *haga*. Hay algo importante: el proceso de aprendizaje no es en si la tarea. El aprendizaje es el *medio* mediante el cual logramos que el algoritmo sea capaz de realizar la tarea. Por ejemplo, si queremos que un robot sea capaz de caminar, entonces la tarea es caminar. Ya después podriamos escribir un programa que específique manualmente como funciona el proceso de caminar, o entrenar un algoritmo de machine learning que le permita al robot *aprender* a caminar.

Las tareas en el machine learning usualmente son descritas en términos de como el sistema debería procesar un **ejemplo**. Un ejemplo es una colección de **características** que han sido cuantitativamente medidas de algún evento u objeto. Típicamente representamos un ejemplo como un vector $x \in \mathbb{R}^n$. En un sistema que decide si en una imagen hay o no un gato, un ejemplo es una imagen dada en forma de una matriz de números, la cual es un vector de tamaño $\mathbb{R}^{256\times256}$, considerando una imágen en blanco y

negro de tamaño 256×256 (¿que pasa si la imagen es a color?). En este caso hay dos tipos de ejemplos o imágenes: las que contienen gatos y las que no.

Podemos agrupar las distintas tareas que puede realizar un algoritmo de machine learning varios grupos:

Clasificación

En este tipo de tarea, le pedimos al programa que específique a cual de k categorías pertenece una entrada. Para resolver esta tarea, usualmente le pedimos al algoritmo de aprendizaje que produzca una función $f: \mathbb{R}^n \to \{1, \ldots, k\}$. Dado $x \in \mathbb{R}^n$, $y = f(x)$ significa que el algoritmo le está asignando la categoría y al ejemplo x. En nuestro ejemplo, queremos que $f(x) = 1$ si la imagen representada por x contiene un gato y $f(x) = 0$ en caso contrario. Otros ejemplos de clasificación son:

- Dado un registro de actividad en una cuenta de banco, determinar si la actividad es sospechosa.
- Determinar un correo es spam o no es spam.
- Dado un post en una red social, clasificar el sentimiento del post como positivo, negativo o neutral.

Regresión

En este tipo de tarea le pedimos al algoritmo que realice una predicción de un valor numérico dada una entrada. Para resolver esta tarea, el algoritmo de aprendizaje entrega una función $f: \mathbb{R}^n \to R$. Este tipo de tarea es similar a la de clasificación, pero el formato de la salida no es un conjunto discreto, sino todo un continuo de valores. Ejemplos de tareas de regresión son:

- Calcular la probabilidad de que llueva durante el día, dado un conjunto de datos meteorológicos a las 6:00am de ese mismo día.
- Determinar la probabilidad de que un usuario de una red social reaccione a un post.
- Dada la información de un apartamento (estrato, metros cuadrados de área, altura, densidad de extranjeros en un kilometro a la redonda) predecir el valor del arriendo.

Transcripción

En este tipo de tarea, le pedimos al sistema que observe una representación relativamente no estructurada de algún tipo de dato, y la transcriba a una forma textual discreta. Ejemplos de tareas de transcripción son:

- Dada la foto de un documento, obtener el texto contenido en él.
- Describir el contenido de una foto.
- Dado un audio de una persona hablando, entregar en texto lo hablado en el audio.

Síntesis y muestreo

En este tipo de tarea, le pedimos al algoritmo que genere nuevos ejemplos que sean similares a aquellos en el conjunto de entrenamiento. Ejemplos de este tipo de tarea son:

- Producir datos sintéticos para entrenar futuros algoritmos de machine learning.
- Generar una imagen a partir de texto.
- Dado un texto, producir un archivo de audio con el textos siendo hablado por una persona.

Quitar ruido

En este tipo de tarea, la entrada del algoritmo es un ejemplo corrupto $x^* \in \mathbb{R}^n$ obtenido a través de un proceso desconocido de corrupción de un ejemplo limpio $x \in \mathbb{R}^n$. El algoritmo debe predecir el ejemplo limpio x a partir de su versión corrupta x^*. Algunos ejemplos son

- Aumentar la calidad de una imagen, un video o un audio
- Un sistema para quitar el ruido cósmico de imágenes astronómicas

La lista anterior de tareas no contiene todos los posibles tipos de tareas que podría tener un sistema de machine learning, y las categorías se pueden mezclar entre si: un sistema que quita el ruido en imágenes también está realizando una regresión sobre cada uno de los píxeles.

El rendimiento, R

Para evaluar las habilidades de un algoritmo de machine learning, debemos diseñar una medida cuantitativa de su rendimiento. Usualmente, esta medida del rendmiento es específica a la tarea T que el algoritmo pretende realizar.

Para tareas como clasificación y transcripción, usualmente medimos la **precisión** del algoritmo, definida como la proporción de ejemplos para los cuales el algoritmo produce la salida correcta. Podemos obtener una medida equivalente si medimos el **error** del algoritmo, es decir, la proporción de ejemplos para los cuales el algoritmo produce una salida incorrecta.

Usualmente, estamos interesados en saber el rendimiento del algoritmo en datos no ha visto antes, ya que esto determinará que tan bien el algoritmo va a funcionar cuando sea desplegado en el mundo real. Por lo tanto, evaluamos la medida de rendimiento un **conjunto de test** que es separado del conjunto de datos usado para entrenar el modelo.

En la práctica, suele ser dificil escoger una medida de rendimiento que corresponda bien al comportamiento deseado del algoritmo. A veces, esto sucede porque es dificil escoger que deberiamos medir. Por ejemplo, en una tarea de transcripción de documentos, tenemos dos opciones: ¿deberíamos medir la precisión a la hora de transcribir secuencias completas castigando cualquier tipo de error, o darle crédito al algoritmo por transcribir bien algunas partes del documento?. Otro ejemplo es en una tarea de regresión: ¿deberíamos penalizar al algoritmo si comete errores pequeños frecuentemente, o si

comete errores grandes con poca frecuencia? El escoger una u otra medida de rendimiento depende del contexto en el que se encuentra situado el problema que se pretende resolver.

En el ejemplo de la clasificación de imágenes de gatos, quizás el programa está implementado en una cámara de la universidad que toma fotos cada minuto, y no nos queremos perder de *ningún gato* que podamos captar, pero no nos molesta tanto si el algoritmo identifica incorrectamente una sombra como un gato.

La experiencia, E

La experiencia se refiere a la forma en que el modelo afronta el proceso de aprendizaje, y a las formas de experiencia (en el sentido de "experimentar") que le son permitidas durante el entrenamiento. En los humanos, "se aprende haciendo", es decir, obtenemos aprendizaje a través de experiencias en las que estamos realizando la tarea que queremos aprender a hacer: aprendemos a montar bicicleta experimentando el montar una bicicleta, no leyendo manuales de ciclismo. La gran mayoría de algoritmos de aprendizaje que vamos a ver en el semillero pueden ser entendidos como algoritmos que experimentan un *dataset* o *conjunto de datos*. Podemos categorizar la experiencia en tres grandes grupos: *aprendizaje supervisado*, *aprendizaje no supervisado* y *aprendizaje por refuerzo*.

Aprendizaje supervisado

Un algoritmo de aprendizaje supervisado "experimenta" un conjunto de datos en el que cada ejemplo contiene características, y cada ejemplo tiene asociado una **etiqueta**. Esta etiqueta es la que le indica al algoritmo de aprendizaje cual es exactamente el valor a predecir, es el valor **objetivo** durante el aprendizaje. En caso de la clasificación de imágenes de datos, cada imagen debe estar acompañada de una etiqueta que indique si hay un gato o no.

Aprendizaje no supervisado

Un algoritmo de aprendizaje no supervisado también experimenta un conjunto de datos, con la diferencia de que este conjunto de datos no necesariamente está etiquetado. Lo que se quiere es que el algoritma descubra patrones ocultos en el conjunto de datos. Quizás queremos que un sistema no solo sea capaz de identificar imágenes de gatos, sino que revele que patrón se oculta detrás del hecho de que haya un gato o no. Quizás el algoritma descubra que los gatos solo aparecen cuando hace buen día, para darse baños de sol al frente de la cámara.

Aprendizaje por refuerzo

En el aprendizaje por refuerzo, el algoritmo interactúa de un entorno, por lo que hay un ciclo de retroalimentación entre el algoritmo y su entorno. Por lo general, en este entorno hay recompensas, y el algoritmo debe aprender a hacer un balance entre exploración del entorno desconocido y explotación del entorno conocido para maximizar cuantas recompensas puede obtener.

La experiencia desde un punto de vista más formal

Más formamente, el aprendizaje supervisado consiste en observar varios ejemplos de un vector x y un vector o valor asociado y (la etiqueta), y aprender a predecir y a partir de x Frecuentemente esto se hace estimando $p(\mathrm{y}|\mathrm{x})$, la probabilidad de y dado x.

Por otro lado, el aprendizaje no supervisado consiste en observar varios ejemplos de un vector aleatorio x, e intentar aprender implicíta o explicitamente la distribución de probabilidad $p(\mathrm{x})$, o características interesantes de la distribución. A diferencia del aprendizaje supervisado, en este contexto no hay profesor: el algoritmo debe obtener información de los datos sin supervisión.

Conjuntos de datos o datasets

La mayoría de algoritmos de machine learning simplemente experimentan un dataset, es decir, una colección de ejemplos. Parte importante del éxito contemporaneo de las redes neuronales es el volumen gigante de datos del que disponemos ahora, volumen que ha sido logrado en gran medida gracias a internet.

La formá más común de describir un conjunto de datos es a través de una matriz de diseño. Una matriz de diseño es una matriz que contiene un ejemplo en cada fila. Cada columna de la matriz corresponde a una característica distinta. Casi todos los algoritmos que vamos a ver en este semillero serán descritos en términos de como operan sobre matrices de diseño. La notación que vamos a usar será la siguiente:

- X es una matriz de diseño
- y es un vector de etiquetas, y y_i es la i-ésima etiqueta, asociada al ejemplo x_i

Ejemplo: Regresión lineal

Para aterrizar las ideas, analicemos un ejemplo de un algoritmo de machine learning: regresión lineal. A medida que introduzcamos más conceptos, vamos a seguir volviendo a este ejemplo.

El objetivo de la regresión lineal es resolver un problema de regresión. Es decir, dado un vector $\mathrm{x} \in \mathbb{R}^n$ queremos predecir el valor de un escalar $y \in \mathbb{R}$.

Dado un ejemplo x, cuya etiqueta verdadera es y, definimos la predicción del modelo de regresión lineal como

$$\hat{y} = \mathrm{w}^T\mathrm{x},$$

donde $\mathrm{w} \in \mathbb{R}^n$ es un vector de parámetros. La expresión anterior simplemente corresponde al producto punto entre w y x. (Si tenemos una matriz de diseño $n \times m$, verifique cual debería ser la dimensión de w para que la expresión anterior tenga sentido). Notemos que si cada ejemplo x es de la forma $\mathrm{x} = x \in \mathbb{R}$, entonces el vector w es un escalar $\mathrm{w} = w$, por lo que el valor predecido por el modelo para el valor x es

$$\hat{y} = wx,$$

que corresponde a la ecuación de una recta que pasa por el origen. Similarmente, si x $\in \mathbb{R}^2$, digamos, x $= [x_1 \quad x_2]$, entonces w es de la forma w$^T = [w_1 \quad w_2]$, así que el valor predicción $\hat{y}$ del modelo en el ejemplo x es de la forma

$$\hat{y} = [w_1 \quad w_2]^T [x_1 \quad x_2] = w_1 x_1 + w_2 x_2,$$

que corresponde a la ecuación de un plano que pasa por el origen. ¿Significa esto que el modelo de regresión lineal solo puede predecir conjuntos de datos que parezcan lineas rectas, planos o hiperplanos? La respuesta no. Podemos adaptar un modelo de regresión lineal para que su salida tenga la forma de una parábola, una función logartimo, seno, etc. El detalle es que la salida de la regresión lineal NO es un linea recta, sino que es una *función lineal de la entrada*. Más adelante vamos a explorar más a profundidad esta sutileza.

Los parámetros son los valores que controlan el comportamiento del sistema. De cierta forma, los parámetros son la parte más importante de un modelo, ya que codifican que es lo que el modelo ha aprendido. Si perdemos los parámetros, hemos perdido efectivamente el aprendizaje obtenido por el modelo. En el caso de la regresión lineal, podemos pensar en w como un conjunto de **pesos** que determinan cuanto afecta cada característica a la predicción. Si una característica x_i recibe un peso positivo w_i, entonces aumentar el valor de esa característica aumenta el valor de la predicción $\hat{y}$. Si una característica recibe un peso negativo, entonces aumentar el valor de esa característica disminuye el valor de nuestra predicción.

Entonces, la tarea T del modelo de regresión lineal es: realizar una predicción $\hat{y}$ del valor verdadero y correspondiente a x mediante una ecuación de la forma

$$\hat{y} = \text{w}^T\text{x}.$$

Definamos ahora el rendimiento R del modelo de regresión lineal. Supongamos que tenemos una matriz de diseño que no usaremos para entrenar al modelo, sino solo para evaluar su rendimiento. Junto con esta matriz, también tenemos un vector de valores y, tal que y_i es el valor verdadero del ejemplo x_i. Como este dataset solo será usado para evaluar o testear el modelo, nos referimos a este dataset como el **conjunto de test**. Nos referiremos a esta matriz de inputs como X$^{(test)}$ y al vector de valores verdaderos de regresión como $y^{(test)}$. La forma en la que evalueramos el rendimiento del modelo es computando el **error cuadrático medio** del modelo en el conjunto de test. Sea $\hat{y}^{(test)}$ el vector de *predicciones* del modelo en el conjunto de test. Notemos que es diferente a $y^{(test)}$, ya que estas son las *predicciones* que hace el modelo, no los valores reales. Definimos el error cuadrático medio (MST del inglés Mean Squared Error) como

$$MSE_{test} = \frac{1}{m}\sum_{i=1} (\hat{y}_i^{(test)} - y_i^{(test)})^2.$$

Analicemos la definición anterior cuidadosamente. Cada termino de la suma es de la forma $(\hat{y}_i^{(test)} - y_i^{(test)})^2$. Es decir, estamos calculando la *diferencia* entre el valor real y el

valor predecido, y estamos elevando al cuadrado este número. Luego, estamos sumando todos estos valores, y por último, estamos diviendo por m, la cantidad de ejemplos, para obtener la media de los errores cuadráticos. La razón para elevar al cuadrado es que queremos que los errores sean positivos, y por lo tanto siempre contribuyan al error: si hubieran errores negativos, se cancelarian con los positivos y podriamos obtener error nulo, aunque el modelo se esté equivocando. El rendimiento R del modelo será medido como el error cuadrático medio del modelo sobre el conjunto de test, y el objetivo será hacer decrecer este error.

De momento hemos definido la tarea T, el rendimiento R y la experiencia E. Ahora necesitamos la parte más importante: el algoritmo de aprendizaje, es decir, el conjunto de instrucciones que harán que el modelo mejore su rendimiento R en la tarea T a través de la experiencia E. En el contexto de la regresión lineal, mejorar el rendimiento significa encontrar un vector de parámetros w que minimice el error cuadrático medio. Si $\mathrm{X}^{(train)}$ es el conjunto de datos de entrenamiento, se puede demostrar (y lo haremos más adelante) que el vector w que minimiza el error cuadrático medio sobre el conjunto de datos de entrenamiento es

$$\mathrm{w} = (\mathrm{X}^{(train)T}\mathrm{X}^{(train)})^{-1}\mathrm{X}^{(train)T}\mathrm{y}^{(train)}.$$

En este caso, calcular w mediante la fórmula anterior corresponde al algoritmo de entrenamiento. Es notable que el modelo de regresión lineal admite una solución cerrada, es decir, la podemos calcular con una fórmula. Sin embargo, es importante notar que la existencia de la solución depende de que el producto de matrices $\mathrm{X}^{(train)^T}\mathrm{X}^{(train)}$ sea invertible. Más adelante hablaremos más a fondo sobre esto.