El modelo de aprendizaje PAC

Agustín E. Martinez Suñé

Enero 2022.

Trabajo final para la materia *Métodos probabilísticos avanzados y Machine Learning* a cargo del Prof. Pablo Groisman.

1 Introducción

Machine learning es un área extremadamente diversa y compleja. Métodos estadísticos de regresión, algoritmos que utilizan redes neuronales, paseos al azar en grafos, problemas de optimización, y muchos otros problemas y métodos pueden enmarcarse en el área de aprendizaje automático. Sería interesante contar con una misma teoría o marco conceptual para comparar y analizar esta diversidad de herramientas. El *modelo de aprendizaje PAC*, propuesto originalmente por Leslie Valiant en 1984[6], es un aporte en esta dirección y sienta las bases de lo que hoy se conoce como *Computational Learning Theory*. Para darse una idea del impacto de esta propuesta, al menos en la comunidad de Ciencias de la Computación, puede verse el texto que acompaña la entrega del *Premio Turing* a Valiant en el año 2010:

> For transformative contributions to the theory of computation, including the theory of probably approximately correct (PAC) learning, (...)

Este texto nos da una pista sobre la perspectiva particular desde la que Valiant encara el problema: la *teoría de la computación*, y en particular, la *teoría de la complejidad computacional*. El objetivo de este trabajo es dar una introducción a la teoría iniciada por Valiant, enfatizando esta perspectiva computacional.

En la sección 2 introducimos de manera informal conceptos básicos de complejidad computacional, en la sección 3 definimos formalmente las principales nociones del modelo de aprendizaje PAC, en la sección 4 exploramos algunas variantes de la definición original y en la sección 5 damos unas breves conclusiones. Además del artículo original de Valiant, las principales fuentes consultadas para las definiciones principales fueron los libros *Foundations of Machine Learning*[4] y *An Introduction to Computational Learning Theory*[3].

2 Teoría de la complejidad computacional

En esencia, la teoría de complejidad computacional estudia el vínculo entre problemas computacionales y los algoritmos que los resuelven y brinda herramientas para analizar y categorizar dichos problemas y algoritmos. Comenzamos definiendo uno de los tipos de problemas computacionales más básicos.

Definición 1 (Problema de decisión)**.** Dado un conjunto de *inputs* X, un problema de decisión se puede describir como una función total $f : X \to \{0, 1\}$. En otras palabras, un problema de decisión requiere que, para cada input, se de una respuesta por sí o por no.

En ese contexto, vamos a decir que un *algoritmo* A resuelve el problema si y sólo si para cada $x \in X$, el algoritmo A con entrada x termina en una cantidad finita de pasos con una salida y tal que $f(x) = y$.

Ejemplo 1. Un clásico ejemplo de problema de decisión es el *test de primalidad* en el que dado un natural n se debe responder si n es primo o no.

Para un mismo problema de decisión pueden existir distintos algoritmos que lo resuelvan satisfactoriamente, y es de interés tener alguna herramienta para compararlos. La herramienta canónica para hacerlo es el llamado *costo computacional* o *complejidad temporal*, que agrupa los algoritmos en grandes clases según la cantidad de operaciones básicas que requiere una ejecución del mismo.

Definición 2 (Algoritmo polinomial)**.** Sea A un algoritmo que toma valores de entrada $x \in X$, decimos que A es un algoritmo polinomial si el costo computacional asociado a A crece polinomialmente con x. Es decir, si dado x de tamaño n, la cantidad de pasos de ejecución de A con entrada x es $poly(x)$, donde $poly : \mathbb{N} \to \mathbb{N}$ es un polinomio.

Hay ciertos detalles formales que no estamos explicitando en estas definiciones ya que no es el objetivo de este trabajo. Por ejemplo, se debería definir formalmente la noción de algoritmo, de costo computacional, y se debería indicar formalmente que cada elemento del conjunto de inputs X es representable de manera finita. Esto suele definirse apelando a la *teoría de lenguajes formales* y a algún modelo de computo específico como la *máquina de turing*. El lector interesado puede encontrar estas definiciones en la bibliografía clásica del área [1, 2]. Ahora estamos en condiciones de definir clases de complejidad de problemas.

Definición 3 (Clase de complejidad P)**.** Un problema de decisión D está en la clase de complejidad **P** (también llamada **PTIME**) si y sólo si existe un algoritmo polinomial que resuelve D.

El concepto de algoritmo polinomial intenta capturar desde la teoría el fenómeno esencialmente práctico de que hay ciertos algoritmos que podemos ejecutar en una computadora de manera *eficiente* en un tiempo razonable y hay

otros que no. Por extensión, se dice que un problema es *tratable* si está en **P** y es *intratable* si no.[1]

Como adelantamos previamente, el objetivo de esta visita rápida a la teoría de complejidad computacional es facilitar la introducción de los conceptos del modelo PAC desde una perspectiva computacional. Para eso, debemos observar que las preguntas esenciales detrás de las definiciones presentadas son:

- *¿Cómo varía la cantidad de pasos de ejecución de un algoritmo a medida que aumenta el tamaño de los datos de entrada?*
- *¿Cómo distinguimos aquellos problemas que se pueden resolver de manera "razonable" por una computadora de aquellos que no?*

3 Bases del aprendizaje PAC

En este contexto, el problema computacional de interés es el problema de aprendizaje, entendido como el problema de construir una *hipótesis* o *clasificador* que generalice las *etiquetas* de una *muestra* tomada de una distribución de probabilidad. En este escenario, el algoritmo que nos interesa estudiar es aquel que toma como entrada un conjunto de datos etiquetados y da como salida una *hipótesis*.

Continuando con la analogía con la sección anterior, la pregunta central que el modelo de aprendizaje PAC intenta responder es:

¿Cómo varía el tamaño de la muestra necesaria para el aprendizaje a medida que exijo mayor precisión en las predicciones de la hipótesis?

A continuación vamos dar la definición principal del modelo PAC de manera informal y luego iremos construyendo de manera rigurosa las herramientas necesarias para formalizarla.

Definición 4 (Noción informal de algoritmo PAC)**.** Dado A un algoritmo de aprendizaje que toma una muestra S de tamaño m y devuelve una hipótesis h, se puede medir la *confianza* y la *precisión* de h. Decimos que A es un algoritmo PAC si y sólo si, al hacer crecer la *precision* y la *confianza* requeridas, el tamaño m que se necesita crece a lo sumo polinomialmente.

Para avanzar en definir formalmente un problema de aprendizaje necesitamos introducir la noción de clase de conceptos.

[1] Esta interpretación de las clases de complejidad no está excenta de polémica. Existen problemas que están en **P** para los cuales no tenemos implementaciones prácticas eficientes y existen problemas que no parecen estar en **P** para los cuáles sí las tenemos. Se puede encontrar una discusión del tema en [7].

Definición 5 (Concepto y clase de conceptos)**.** Sea $\mathcal{X}$ el *input space* y $\mathcal{Y}$ el conjunto de *etiquetas* o *target values* decimos que un *concepto* c es un mapeo de $\mathcal{X}$ a $\mathcal{Y}$ (lo notamos $c : \mathcal{X} \to \mathcal{Y}$). Una *clase de conceptos* $\mathcal{C}$ es un conjunto de conceptos que nos interesa aprender.

Ejemplo 2. Imaginemos que nuestras instancias de entrada son puntos en el plano y que la tarea es aprender el conjunto de puntos interiores de un triángulo que desconocemos. En ese caso:

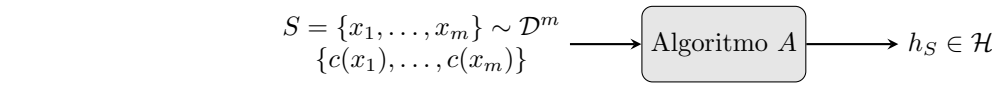
- $\mathcal{X} = \mathbb{R}^2$, el input space es todo el plano real,
- $\mathcal{Y} = \{0, 1\}$, cada etiqueta indica si el punto se encuentra dentro del triángulo o no,
- Cualquier triángulo del plano se puede caracterizar como un concepto $c : \mathcal{X} \to \mathcal{Y}$,
- la clase de conceptos $\mathcal{C}$ adecuada, es el conjunto de todos los $c : \mathcal{X} \to \mathcal{Y}$ que caracterizan triangulos del plano.

En este ejemplo estamos frente a un problema de *clasificación binaria* ya que $\mathcal{Y}$ contiene sólo dos etiquetas.

A continuación definimos formalmente el problema de aprendizaje.

Definición 6 (Problema de aprendizaje)**.** Sean un input space $\mathcal{X}$, un conjunto de etiquetas $\mathcal{Y}$, una clase de conceptos C (de la forma $c : \mathcal{X} \to \mathcal{Y}$) y una distribución de probabilidad $\mathcal{D}$ sobre $\mathcal{X}$. El problema de aprendizaje es, dados una muestra $S = \{x_1, \ldots, x_m\} \sim \mathcal{D}^m$ y las correspondientes etiquetas $\{c(x_1), \ldots, c(x_m)\}$ asociadas, según un concepto $c \in \mathcal{C}$, construir una hipótesis $h_S : \mathcal{X} \to \mathcal{Y}$ que aproxime c en futuros inputs tomados de $\mathcal{D}$.

Un algoritmo de aprendizaje A considera un conjunto fijo $\mathcal{H}$ de posibles conceptos, llamado *conjunto de hipótesis*. La distribución $\mathcal{D}$, si bien es fija, es desconocida para el algoritmo de aprendizaje.



Dado un problema de aprendizaje y un algoritmo específico que lo resuelve, la calidad la hipótesis que devuelve depende del error de generalización.

Definición 7 (Error de generalización)**.** Dada una hipótesis $h \in \mathcal{H}$, un concepto $c \in \mathcal{C}$ y una distribución subyacente $\mathcal{D}$, el *error de generalización* se define como:

$$R(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)]$$

Ahora sí están todos los ingredientes para la definición central de la teoría: Algoritmo *Probably Approximately Correct*.

Definición 8 (Algoritmo PAC)**.** Dado un algoritmo de aprendizaje A que intenta aprender una clase de conceptos $\mathcal{C}$, decimos que es un *algoritmo PAC* si y sólo si existe un polinomio $p(\cdot,\cdot)$ tal que para todo $0 < \epsilon < \frac{1}{2}$, $0 < \delta < 1$, concepto $c \in \mathcal{C}$, y distribución de probabilidad $\mathcal{D}$ sobre $\mathcal{X}$, si se toma una muestra de tamaño $m \geq p(\epsilon^{-1}, \delta^{-1})$, entonces

$$\mathbb{P}_{S\sim\mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1 - \delta$$

Valores chicos de ϵ denotan altos niveles de precisión de la hipótesis h_S, valores chicos de δ denotan altos niveles de confianza en dicha precisión. Esto explica por qué el polinomio p se aplica sobre ϵ^{-1} y δ^{-1}. La terminología *Probably Approximately Correct (PAC)* refiere a estos conceptos: "probably" por la confianza y "approximately correct" por la precisión. Por simplicidad, en la definición anterior se omitieron dos factores que también intervienen en el crecimiento del polinomio p: el costo de representación de un elemento $x \in \mathcal{X}$ y el costo de representación de un concepto $c \in \mathcal{C}$.

Bajo la luz de esta definición, podemos comparar distintos algorimtos PAC según el tamaño de la muestra que necesitan para aprender un concepto de manera exitosa.

Definición 9 (Sample complexity)**.** Dado un algoritmo A de aprendizaje PAC, llamaremos *complejidad de muestra* de A al polinomio p más chico asociado a A según la definición 8.

Finalmente, estamos en condiciones de definir cuáles son aquellos problemas de aprendizaje que se pueden resolver de manera "razonable" según el modelo PAC. Retomando la comparación con las ideas básicas de complejidad computacional introducidas en la sección anterior, proponemos pensar esta definición en analogía con la de clase de complejidad **P**.

Definición 10 (PAC learnable)**.** Dada una clase de conceptos $\mathcal{C}$, diremos que $\mathcal{C}$ es *PAC learnable* si y sólo si existe un algoritmo de aprendizaje PAC para $\mathcal{C}$.

A continuación veremos en detalle un ejemplo de un problema que se puede demostrar PAC learnable.

Ejemplo 3 (Los intervalos son PAC learnable)**.** La tarea es aprender un intervalo de la recta real a partir de una muestra de puntos. En este caso tenemos:

- $\mathcal{X} = \mathbb{R}$, el input space son los puntos de la recta real,
- $\mathcal{Y} = \{0, 1\}$, cada etiqueta indica si el punto se encuentra dentro del intervalo o no,
- La clase de conceptos $\mathcal{C}$ son todas las $c : \mathcal{X} \to \mathcal{Y}$ que son funciones características de un intervalo.

Dada una muestra de puntos $S = \{x_1, \ldots, x_m\}$ con etiquetas $\{c(x_1), \ldots, c(x_m)\}$, proponemos el siguiente algoritmo:

1. tomar $a_1 = \min\{x \mid x \in S, c(x) = 1\}$, el valor más chico de la muestra que está dentro del intervalo,
2. tomar $b_1 = \max\{x \mid x \in S, c(x) = 1\}$, el valor más grande de la muestra que está dentro del intervalo,
3. devolver como hipótesis h_S la función característica del intervalo $[a_1, b_1]$.

Para probar que es un algoritmo PAC tomamos cualquier ϵ, δ y distribución $\mathcal{D}$ sobre $\mathcal{X}$, y debemos deducir el tamaño de muestra m que satisface la condición $\mathbb{P}_{S\sim\mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1 - \delta$, o lo que es equivalente $\mathbb{P}_{S\sim\mathcal{D}^m}[R(h_S) \geq \epsilon] \leq \delta$

Si el intervalo correcto que se debe aprender es $I = [a_0, b_0]$, el error de generalización $R(h_S)$ en este caso es $\mathbb{P}_{x\sim\mathcal{D}}[x \in I \setminus h_S]$ ya que el único caso en que la hipótesis falla es cuando un punto de I se clasifica como negativo. Sabemos que $h_S \subseteq I$, ya que los extremos a_1 y b_1 de h_S con certeza están dentro del intervalo. Es decir, sabemos que $a_0 < a_1 < b_1 < b_0$, por lo tanto, podemos reescribir el error de generalización como

$$\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) + \mathbb{P}_{x\sim\mathcal{D}}(b_1 < x < b_0)$$

Queremos acotar esta suma por ϵ, para eso vamos a acotar cada término por $\epsilon/2$,

$$\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) \geq \epsilon/2$$

Esta probabilidad va a depender del valor a_1 que haya seleccionado nuestro algoritmo. Tomemos a' tal que

$$\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a') = \epsilon/2$$

La desigualdad $\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) \geq \epsilon/2$ se cumple si y sólo si $a_1 \geq a'$ y, por la forma en que funciona el algoritmo, esto sucede si ningún punto y de la muestra S cumple $a_0 < y < a'$. La probabilidad de que un único punto no este dentro del intervalo es $(1 - \epsilon/2)$, por la forma en que elegimos a'. La probabilidad para m puntos es $(1 - \epsilon/2)^m$. El razonamiento para el otro término es análogo, $\mathbb{P}_{x\sim\mathcal{D}}(b_1 < x < b_0) \geq \epsilon/2$ se cumple con probabilidad $(1 - \epsilon/2)^m$.

Repasando el razonamiento:

$$\mathbb{P}_{S\sim\mathcal{D}^m}(R(h_S) \geq \epsilon) = \mathbb{P}_{S\sim\mathcal{D}^m}(\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) + \mathbb{P}_{x\sim\mathcal{D}}(b_1 < x < b_0) \geq \epsilon)$$

Por definición de $R(h_S)$

$$\leq \mathbb{P}_{S\sim\mathcal{D}^m}\big(\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) \geq \epsilon/2 \bigcup \mathbb{P}_{x\sim\mathcal{D}}(b_1 < x < b_0) \geq \epsilon/2\big)$$

Por inclusión de eventos. Si la suma es mayor a ϵ, entonces con certeza alguno de los dos sumandos es mayor a $\epsilon/2$.

$$\leq \mathbb{P}_{S\sim\mathcal{D}^m}\big(\mathbb{P}_{x\sim\mathcal{D}}(a_0 < x < a_1) \geq \epsilon/2\big) + \mathbb{P}_{S\sim\mathcal{D}^m}\big(\mathbb{P}_{x\sim\mathcal{D}}(b_1 < x < b_0) \geq \epsilon/2\big)$$

Por cota superior de la probabilidad de la union.

$$\leq (1 - \epsilon/2)^m + (1 - \epsilon/2)^m$$

Por el razonamiento que hicimos en el parrafo anterior.

Lo que resta ahora es acotar esta última expresión por δ y resolver para m:

$$2(1-\epsilon/2)^m \leq \delta$$

Usando el hecho de que $(1-x) \leq e^{-x}$ podemos acotar $(1-\epsilon/2)$, y resolver, en cambio, la siguiente desigualdad:

$$2e^{-m\epsilon/2} \leq \delta$$

Basta tomar $m \geq (2/\epsilon \log(2/\delta))$. Si bien $(2/\epsilon \log(2/\delta))$ no es estrictamente un polinomio por el factor logarítmico, puede acotarse superiormente por un polinomio y este cumple el rol de la cota deseada.

4 Variantes del aprendizaje PAC

En esta sección veremos dos variantes de la definición de aprendizaje PAC que consideramos relevantes. No pretende ser una muestra extensiva o representativa de las distintas extensiones que se han propuesto para la definición pero sí sirven de introducción a la forma en que se puede pensar modificaciones que sean de utilidad para distintos escenarios.

4.1 PAC learning agnóstico

La definición presentada hasta el momento asume que, dado un input x, la etiqueta y asociada es determinística. Pero en muchos problemas de aprendizaje no supervisado esto no es cierto, sino que la distribución $\mathcal{D}$ se define sobre los pares $\mathcal{X} \times \mathcal{Y}$.

Ejemplo 4. Supongamos el problema de intentar predecir el nombre de pila de una persona a partir del género y del año y ciudad de nacimiento. En este caso la etiqueta no es una función determinística, para una misma combinación de género y año y ciudad de nacimiento puede haber distintos datos de personas con nombres distintos. Sin embargo sí podemos estudiar la distribución de probabilidad que siguen estas etiquetas.

En estos escenarios la muestra de entrenamiento es de la forma

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

y debemos redefinir el error de generalización de la siguiente manera.

Definición 11 (Error de generalización)**.** Dada una hipótesis $h \in \mathcal{H}$, y una distribución $\mathcal{D}$, el *error de generalización* se define como:

$$R(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

En este contexto, la extensión del framework se conoce como PAC learning *agnóstico*.

Definición 12 (PAC learning agnóstico)**.** Sea $\mathcal{H}$ el conjunto de hipótesis. Decimos que A es un algoritmo de aprendizaje PAC agnóstico si y sólo si existe un polinomio $p(\cdot,\cdot)$ tal que para todo $0 < \epsilon < \frac{1}{2}$, $0 < \delta < 1$, y distribución de probabilidad $\mathcal{D}$ sobre $\mathcal{X} \times \mathcal{Y}$, si se toma una muestra de tamaño $m \geq p(\epsilon^{-1}, \delta^{-1})$, entonces

$$\mathbb{P}_{S\sim\mathcal{D}^m}[R(h_S) - \min_{h\in\mathcal{H}} R(h) \leq \epsilon] \geq 1 - \delta$$

Mientras en la definición original se pedía que el error de h_S sea menor o igual a ϵ, en este caso lo que se pide acotar es la diferencia entre el error de h_S y el error de la hipótesis óptima. Esto se debe a que la hipótesis perfecta podría no existir porque la etiqueta de un punto en el *input* no necesariamente es determinística. En caso de que sí sea una función determinística estamos en el caso de la definición original y el error de la hipótesis óptima es cero.

4.2 PAC learning débil

Observando nuevamente la definición original 8, la cota que venimos utilizando para el tamaño de la muestra es $m \geq p(\epsilon^{-1}, \delta^{-1})$. Como ya mencionamos, la principal consecuencia de esta cota es que sólo permite que m crezca polinomialmente a medida que crece la precisión ϵ^{-1} y la confianza δ^{-1}. Podemos definir un modelo más débil que no requiera al algoritmo aprender una hipótesis con precisión arbitrariamente grande, sino que requiera que la precisión sea al menos mejor que adivinar al azar.

Definición 13 (Algoritmo PAC débil)**.** Dado un algoritmo de aprendizaje A que intenta aprender una clase de conceptos $\mathcal{C}$, decimos que es un algoritmo PAC *débil* si y sólo si existe un polinomio $p(\cdot)$ tal que para todo $0 < \delta < 1$, concepto $c \in \mathcal{C}$, y distribución de probabilidad $\mathcal{D}$ sobre $\mathcal{X}$, si se toma una muestra de tamaño $m \geq p(\delta^{-1})$, entonces

$$\mathbb{P}_{S\sim\mathcal{D}^m}\big(R(h_S) < \frac{1}{2}\big) \geq 1 - \delta$$

Está definición se vincula con el problema de *boosting*: ¿cómo construir un algoritmo de aprendizaje de alta precisión utilizando algoritmos de precisión baja?. En términos teóricos la pregunta pertinente es: dado un problema de aprendizaje para el que se cuenta con un algoritmo PAC débil A, ¿se puede construir un algoritmo PAC *fuerte* A' (en el sentido de la definición original) para el mismo problema? La respuesta es positiva y fue dada por Schapire en 1990 [5]. Esto abrió una rama de investigación que estudia distintos algoritmos de boosting para mejorar la performance combinando el output de distintos *weak learners*.

5 Conclusiones

En este trabajo dimos una introducción al área de *computational learning theory*. Vimos que este área toma la perspectiva de la teoría de complejidad computacional para razonar sobre algoritmos de aprendizaje. La pregunta básica que

ordena estos análisis es cómo varía el tamaño de la muestra necesaria para el aprendizaje a medida que se le requiere mayor precisión al modelo aprendido.

A través de un ejemplo de algoritmo PAC vimos la forma en que las definiciones teóricas pueden ser puestas en práctica. Finalmente, introdujimos algunas variantes de la defínción original que permiten extenderla para distintos escenarios y que fueron abriendo nuevas preguntas de investigación en el área.

References

[1] T. H. Cormen and T. H. Cormen, editors. *Introduction to Algorithms*. MIT Press, Cambridge, Mass, 2nd ed edition, 2001.

[2] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley, Reading, Mass, 1979.

[3] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, Mass, 1994.

[4] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[5] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.

[6] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[7] M. Y. Vardi. On P, NP, and computational complexity. *Communications of the ACM*, 53(11):5–5, November 2010.