**Artificial Intelligence, A Modern Approach**

Cap 18 “**Aprendiendo de Ejemplos**” - Resumen

Universidad de la Costa - 2017

Nicole Escobar

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada.

Según el ejemplo propuesto en “Artificial Intelligence, a modern approach”. Dado un conjunto de entrenamiento de N pares ejemplo de entrada-salida:

**(x1, y1), (x2, y2), ... (xN, yN),**

donde cada yj fue generado por una función desconocida **y = f (x),** descubre una función h que se aproxima a la función verdadera f.

Aquí x e y pueden ser cualquier valor; no necesitan ser números. La función h es una hipótesis. El aprendizaje es una búsqueda a través del espacio de posibles hipótesis para uno que se desempeñará bien, incluso en nuevos ejemplos más allá del conjunto de entrenamiento.

Para medir la exactitud de una hipótesis, se le da un conjunto de ejemplos de prueba que son distintos del conjunto de entrenamiento. Se dice que una hipótesis generaliza bien si predice correctamente el valor de y para ejemplos nuevos. A veces la función f es estocástica no es estrictamente una función de x, y lo que debe aprenderse es una distribución de probabilidad condicional, P (Y | x).

En este capitulo se propone dejar que H sea la clase de todos los programas Java, o máquinas de Turin, ya que, cada función computable puede ser representada por alguna máquina de Turing. Un problema con esta idea es que no tiene en cuenta la complejidad computacional del aprendizaje.

Hay una compensación entre la expresividad de un espacio de hipótesis y la complejidad de encontrar una buena hipótesis dentro de ese espacio. Por ejemplo, ajustar una línea recta a los datos es un cálculo fácil; el ajuste de polinomios de alto grado es algo más difícil; y el ajuste de las máquinas de Turing es en general indecidible. Una segunda razón para preferir espacios de hipótesis simples es que presumiblemente vamos a querer usar h después de haberlo aprendido, y la computación h(x) cuando h es una función lineal se garantiza que sea rápida, mientras que el cálculo de un programa arbitrario de máquina de Turing no es incluso garantizado para terminar. Por estas razones, la mayoría del trabajo sobre el aprendizaje se ha centrado en representaciones simples.

Veremos que la compensación entre la expresividad y la complejidad no es tan simple como parece: a menudo es el caso, como vimos con la lógica de primer orden en el capítulo 8, que un lenguaje expresivo hace posible que una hipótesis simple se ajuste a la datos, mientras que restringir la expresividad del lenguaje significa que cualquier hipótesis coherente debe ser muy compleja. Por ejemplo, las reglas del ajedrez pueden escribirse en una o dos páginas de lógica de primer orden, pero requieren miles de páginas cuando se escriben en lógica proposicional.

Un árbol de decisiones representa una función que toma como entrada un vector de valores de atributos y devuelve una "decisión"; los valores de entrada y salida pueden ser discretos o continuos.

En el documento se centran en problemas donde las entradas tienen valores discretos y la salida tiene exactamente dos valores posibles; esta es la clasificación booleana, donde cada entrada de ejemplo positivo será clasificada como verdadera o falsa.

**Expresividad de los árboles de decisión,** un árbol de decisión booleano es lógicamente equivalente a la aserción de que el atributo de objetivo es verdadero si y sólo si los atributos de entrada satisfacen uno de los caminos que conducen a una hoja con valor verdadero.

**Ampliación de la aplicabilidad de los árboles de decisión**

Datos perdidos: En muchos dominios, no todos los valores de atributo se conocerán para cada ejemplo. Es posible que los valores no se hayan registrado o que sean demasiado costosos de obtener.

Atributos de valores múltiples: Cuando un atributo tiene muchos valores posibles, la medida de ganancia de información da una indicación inadecuada de la utilidad del atributo. En el caso extremo, un atributo como *ExactTime* tiene un valor diferente para cada ejemplo, lo que significa que cada subconjunto de ejemplos es un singleton con una clasificación única, y la medida de ganancia de información tendría su valor más alto para este atributo.

Atributos de entrada continuos y de valor entero: Atributos de valor continuo o de valor entero tales como Altura y Peso, tienen un conjunto infinito de valores posibles. En lugar de generar infinidad de ramas, los algoritmos de aprendizaje del árbol de decisión normalmente encuentran el punto de división que proporciona la mayor ganancia de información.

Atributos de salida de valor continuo: Si estamos tratando de predecir un valor numérico de salida, como el precio de un apartamento, entonces necesitamos un árbol de regresión en lugar de un árbol de clasificación

Se quiere aprender una hipótesis que se ajuste mejor a los datos futuros. Para hacerlo es preciso necesitamos definir "datos futuros" y "mejores". Se elabora la hipótesis de estacionariedad: que existe una distribución de probabilidad sobre los ejemplos que permanece estacionaria con el tiempo.

Cada punto de datos de ejemplo es una variable aleatoria Ej cuyo valor observado ej = (xj, yj) es muestreado de esa distribución, e independiente de los ejemplos anteriores:

**P (Ej, Ej-1, Ej-2, ...) = P (Ej),**

y cada ejemplo tiene una distribución de probabilidad previa idéntica:

**P (Ej) = P (Ej-1) = P (Ej-2) = ···.**

Los ejemplos que satisfacen estas suposiciones se llaman independientes e idénticamente distribuidos o i.i.d. Un i.i.d. la suposición conecta el pasado con el futuro; sin alguna conexión, todas las apuestas están fuera del futuro podría ser cualquier cosa. Si el conjunto de pruebas está bloqueado, pero todavía desea medir el rendimiento en datos no vistos como una forma de seleccionar una buena hipótesis, a continuación, divida los datos disponibles en un conjunto de formación y un conjunto de validación.

En la a siguiente sección del capítulo muestra cómo usar conjuntos de validación para encontrar una buena compensación entre la complejidad de la hipótesis y la bondad de ajuste, se divide en tres subsecciones, Selección del modelo: Complejidad versus bondad de ajuste, De las tasas de error a la pérdida y La Regularización.

¿Existen algunos principios más generales que rijan el número de ejemplos necesarios en general?

Las preguntas como esta son abordadas por la teoría del aprendizaje computacional, que se encuentra en la intersección de la IA, la estadística y la informática teórica. El principio subyacente es que cualquier hipótesis que esté seriamente equivocada casi seguramente será "descubierta" con alta probabilidad después de un pequeño número de ejemplos, porque hará una predicción incorrecta. Por lo tanto, cualquier hipótesis que sea consistente con un conjunto suficientemente grande de ejemplos de entrenamiento es poco probable que esté seriamente equivocada: es decir, debe ser probablemente aproximadamente correcta. Cualquier algoritmo de aprendizaje que devuelve hipótesis que probablemente sean aproximadamente correctas se llama algoritmo de aprendizaje PAC; podemos utilizar este enfoque para proporcionar límites en el rendimiento de varios algoritmos de aprendizaje.

**Regresión lineal univariada**, en esraesta Sólo se maneja una variable independiente, por lo que sólo cuenta con dos parámetros se Define w como el vector [w0, w1], y definiremos

**hw (x) = w1x + w0.**

Muchas formas de aprendizaje implican ajustar los pesos para minimizar una pérdida, por lo que ayuda a tener una imagen mental de lo que está pasando en el espacio de peso, el espacio definido por todos los ajustes posibles de los pesos.

Hay otra posibilidad, llamada descenso de gradiente estocástico, en la que consideramos sólo un solo punto de entrenamiento a la vez, dando un paso después de cada uno usando la Ecuación. El descenso de gradiente estocástico se puede usar en un ajuste en línea, tiempo, o fuera de línea, donde recorremos los mismos datos tantas veces como sea necesario, dando un paso después de considerar cada ejemplo. A menudo es más rápido que el descenso gradiente por lotes.

**Regresión lineal multivariante**, La regresión lineal permite trabajar con una variable a nivel de intervalo o razón. De la misma manera, es posible analizar la relación entre dos o más variables a través de ecuaciones, lo que se denomina regresión múltiple o regresión lineal múltiple.

Constantemente en la práctica de la investigación estadística, se encuentran variables que de alguna manera están relacionadas entre sí, por lo que es posible que una de las variables pueden relacionarse matemáticamente en función de otra u otras variables.

**Clasificadores lineales con umbral duro,** Las funciones lineales se pueden utilizar para hacer la clasificación, así como la regresión. Cada punto se define por dos valores de entrada, x1 y x2, que se refieren a las magnitudes de onda de cuerpo y superficie calculadas a partir de la señal sísmica. Teniendo en cuenta estos datos de entrenamiento, la tarea de clasificación es aprender una hipótesis h que tomará nuevos puntos (x1, x2) y devolverá 0 para terremotos o 1 para explosiones.

**Clasificación lineal con regresión logística,** la hipótesis hw (x) no es diferenciable y es, de hecho, una función discontinua de sus entradas y sus pesos; esto hace que el aprendizaje con la regla perceptrón sea una aventura muy impredecible. Además, el clasificador lineal siempre anuncia una predicción completamente confiable de 1 o 0, incluso para ejemplos que están muy cerca del límite; en muchas situaciones, realmente necesitamos predicciones más graduadas.

Todas estas cuestiones se pueden resolver en gran medida suavizando la función umbral que aproxima el umbral duro con una función continua y diferenciable.

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples de forma aproximadamente análoga, el comportamiento observado en los axones de las neuronas en los cerebros biológicos.

Un aspecto interesante de estos sistemas es que son impredecibles en su éxito con el auto-aprendizaje. Después del entrenamiento, algunos se convierten en grandes solucionadores de problemas y otros no funcionan tan bien. Con el fin de capacitarlos, se necesitan varios miles de ciclos de iteración.

Las redes neuronales se han utilizado para resolver una amplia variedad de tareas, como la visión por computador y el reconocimiento de voz, que son difíciles de resolver usando la ordinaria programación basado en reglas.

Estructuras de redes neuronales, las redes neuronales se componen de nodos o unidades conectadas por enlaces dirigidos. Un enlace de la unidad i a la unidad j sirve para propagar la activación ai de i a j. Cada enlace también tiene un peso numérico wi, j asociado con él, que determina la fuerza y el signo de la conexión.

Las redes neuronales representan funciones no lineales complejas con una red de unidades umbral lineales. Las redes neuronales de feed-forward a largo plazo pueden representar cualquier función, dadas suficientes unidades. El algoritmo de retropropagación implementa un descenso en gradiente en el espacio de parámetros para minimizar el error de salida.

Los modelos no paramétricos utilizan todos los datos para hacer cada predicción, en lugar de tratar de resumir los datos primero con unos pocos parámetros. Los ejemplos incluyen vecinos más cercanos y regresión localmente ponderada.

Las máquinas vectoriales de soporte encuentran separadores lineales con margen máximo para mejorar el rendimiento de generalización del clasificador. Los métodos del kernel transforman implícitamente los datos de entrada en un espacio de alta dimensión donde puede existir un separador lineal, incluso si los datos originales son no separables.

Métodos de conjunto como el impulso a menudo tienen mejores resultados que los métodos individuales. En el aprendizaje en línea podemos agregar las opiniones de los expertos para acercarse arbitrariamente al rendimiento del mejor experto, incluso cuando la distribución de los datos está cambiando constantemente.

**Artículos**

* Lei Zheng, Shaojun Wang, Yan Liu, and Chi-Hoon Lee. 2009. Information theoretic regularization for semi-supervised boosting. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). ACM, New York, NY, USA, 1017-1026. DOI: <https://doi.org/10.1145/1557019.1557129>
* Sunil Vadera. 2010. CSNL: A cost-sensitive non-linear decision tree algorithm. *ACM Trans. Knowl. Discov. Data* 4, 2, Article 6 (May 2010), 25 pages. DOI=http://dx.doi.org/10.1145/1754428.1754429
* William Evans, Sridhar Rajagopalan, and Umesh Vazirani. 1993. Choosing a reliable hypothesis. In *Proceedings of the sixth annual conference on Computational learning theory*(COLT '93), Lenny Pitt (Ed.). ACM, New York, NY, USA, 269-276. DOI=<http://dx.doi.org/10.1145/168304.168349>
* Timothy Koschmann. 1999. Toward a dialogic theory of learning: Bakhtin's contribution to understanding learning in settings of collaboration. In Proceedings of the 1999 conference on Computer support for collaborative learning (CSCL '99), Christopher M. Hoadley and Jeremy Roschelle (Eds.). International Society of the Learning Sciences , Article 38 .
* Hossein Ahmadi, Nam Pham, Raghu Ganti, Tarek Abdelzaher, Suman Nath, and Jiawei Han. 2010. Privacy-aware regression modeling of participatory sensing data. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (SenSys '10). ACM, New York, NY, USA, 99-112. DOI=<http://dx.doi.org/10.1145/1869983.1869994>
* Mark Plagge. 2013. Using artificial neural networks to predict first-year traditional students second year retention rates. In *Proceedings of the 51st ACM Southeast Conference* (ACMSE '13). ACM, New York, NY, USA, Article 17, 5 pages. DOI: <https://doi.org/10.1145/2498328.2500061>
* Dongling Zhang, Yingjie Tian, and Peng Zhang. 2008. Kernel-Based Nonparametric Regression Method. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03* (WI-IAT '08), Vol. 3. IEEE Computer Society, Washington, DC, USA, 410-413. DOI=<http://dx.doi.org/10.1109/WIIAT.2008.157>
* Hwanjo Yu, Ilhwan Ko, Youngdae Kim, Seungwon Hwang, and Wook-Shin Han. 2011. Exact indexing for support vector machines. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (SIGMOD '11). ACM, New York, NY, USA, 709-720. DOI: https://doi.org/10.1145/1989323.1989398