Algoritmos de Machine Learning y sus aplicaiones

Trabajo de Fin de Grado

Javier Díaz Bustamante Ussia

22 de junio de 2015

Índice

Resumen

En este trabajo vamos a estudiar diferentes algoritmos de *Machine Learning* para la clasificación de sucesos. Para comprobar su funcionamiento, los utilizaremos sobre distintas bases de datos, una de reconocimiento de dígitos, una de supervivientes del Titanic y la última de búsqueda del bosón de Higgs.

1. Introducción

En un mundo cada vez más tecnológico, las bases de datos crecen cada día. Cuando alguien entra en una página web, realiza una compra en un comercio, una empresa realiza un estudio de mercado, se celebran unas elecciones, se están almacenando datos. Con este ingente flujo de datos, la física no se podía quedar detrás, hoy en día los grandes aceleradores de partículas manejan al día millones de sucesos que hay que catalogar, clasificar y estudiar, pero la enorme cantidad de estos datos hace imposible su tratamiento *manual*.

Al calor del problema del tratamiento de datos surge el *Machine Learning* (de ahora en adelante *ML*), con algoritmos cada vez más potentes capaces de sacar el máximo partido a estos datos. En este trabajo estudiaremos en concreto cinco de ellos, todos de clasificación (también los hay de regresión). Estos cinco son *Logistic Regresion*, *Naïve Bayes*, *Support Vector Machine*, *Random Forest* y *K-Nearest Neighbors*, explicados en detalle en la seccion 3.

Existen dos tipos de problemas en *ML*, los de clasificación y los de regresión. Estos últimos tratan de predecir el valor de una variable para un conjunto de datos nuevos, como por ejemplo la regresión por mínimos cuadrados. Los problemas de clasificación consisten en tratar de asignar cada entrada de datos nuevos a una clase concreta, pudiendo ser una clasificación binaria (verdadero o falso) o una clasificación multiclase, como por ejemplo un algoritmo de reconocimiento de dígitos. A lo largo de este trabajo, por similitud con el problema del bosón de Higgs, trataremos únicamente con problemas de clasificación binaria.

2. Consideraciones previas

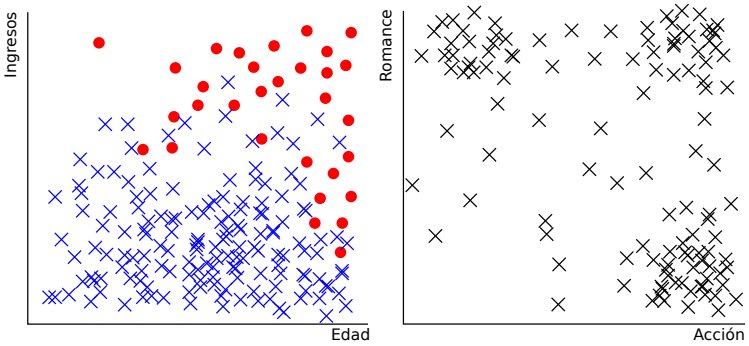
2.1. Aprendizaje supervisado y no supervisado

Antes de explicar cada uno de los algoritmos, vamos a ver algunas características comunes de *ML*. Para empezar, veamos la diferencia entre *aprendizaje supervisado* y *aprendizaje no supervisado*. En *ML*, el término aprendizaje (o entrenamiento) se refiere al análisis de los datos por parte del algoritmo. Éste recibe un conjunto de datos de aprendizaje (el *training set*) y los analiza para tomar una decisión (ya veremos más adelante cómo). La diferencia entre aprendizaje supervisado y aprendizaje no supervisado es que en el primero el conjunto de datos de aprendizaje se encuentra perfectamente etiquetado, mientras que en el no supervisado no lo está.

Por ejemplo, si quisiéramos entrenar un algoritmo para saber a qué tipo de personas les gusta la música clásica, podríamos crear un training set con entradas como: «A Juan Pérez, de 64 años, de clase alta, casado y con carrera universitaria, residente en Madrid, le gusta la música clásica», y otras entradas del estilo de «A Pedro Gutiérrez, de 24 años, clase media, soltero, sin carrera universitaria, residente en Villalpando, no le gusta». Éste sería un problema de clasificación supervisada, y podemos ver un ejemplo de training set ficticio en la gráfica 1a.

Por otra parte, podemos tener datos como por ejemplo el gusto por diferentes tipos de películas de los clientes de un videoclub. A una persona pueden gustarle los romances un 80 %, mientras que las películas de acción sólo un 57 %. A otra, sin embargo, le gustan las primeras al 43 % y las segundas al 96 %. Un ejemplo

de training set podría ser el de la gráfica 1b. En ella, aunque no tengamos clasificadas a las personas, se puede comprobar de forma más o menos nítida que el videoclub tiene tres tipos de clientes distintos.



(a) Supervisado. En rojo, los amantes de la música clásica.

(b) No supervisado.

Gráfica 1: Aprendizaje supervisado (a) y no supervisado (b).

2.2. Métricas

Una vez el algoritmo ha sido entrenado sobre un training set, es hora de probarlo. Para ello se emplea otro conjunto de datos, llamado test set[1], que se clasifica según el algoritmo disponga. En el caso de clasificación binaria, el algoritmo asignará a cada entrada del test set una etiqueta de clase (verdadero o falso, 1 ó 0). Nos interesa saber cómo de preciso ha sido el algoritmo, para lo que disponemos de diversas métricas. Según la etiqueta original de cada entrada y según la etiqueta predicha por el algoritmo, se pueden dar los casos de la tabla 1.

	0 real	**1 real**
0 predicho	Verdadero negativo (TN)	Falso negativo (FN)
1 predicho	Falso positivo (FP)	Verdadero positivo (TP)

Tabla 1: Posibles casos.

Un buen algoritmo será aquel que prediga 0 para todos los negativos y 1 para los positivos, pero la realidad es que eso casi nunca ocurre. Necesitamos, por lo tanto, un mecanismo para evaluar la bondad de un algoritmo, y eso se consigue con las métricas definidas a continuación[2]:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

[1] Muchas veces se suele tener sólo un conjunto de datos, que se divide en distintas partes, una el training set, otra el test set, y otra el cross-validation set, que veremos más adelante.

[2] Ver [1].

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

La ecuación (2) (sensibilidad) nos da una medida de la exactitud de los casos positivos, mientras que (3) (especificidad) nos da la de los casos negativos. (4) (precisión) nos da la exactitud de los casos clasificados como positivos.

Aunque pudiera parecer que (1) (exactitud) es una buena medida de la bondad de la clasificación, no siempre es así. En los casos de clases sesgadas (*skewed classes*), en las que la proporción de una de las dos clases es mucho mayor que la de la otra, un algoritmo que clasifique cualquier entrada de datos como la clase mayoritaria tendría una gran exactitud, aunque no cumpliría con el objetivo de clasificar correctamente los datos. Por ejemplo, supongamos un problema de detección de fraude en transacciones bancarias en el que nuestro algoritmo deba clasificar como verdaderas las operaciones fraudulentas. En la realidad hay muchas más transacciones legítimas que fraudulentas, en nuestro ejemplo consideraremos que son el 99 % de las transacciones. Si tenemos un algoritmo que prediga siempre que una transacción es legítima, la exactitud será del 99 %, pero no por ello será un buen algoritmo.

El valor F1, ecuación (5), soluciona este problema. Se trata de una media armónica de la precisión y la sensibilidad.

2.3. Cross Validation

La mayoría de los algoritmos de *ML* dependen de unos parámetros que hay que fijar

Referencias

[1] S. Bhattacharyya, S. Jha, K. Tharakunnel y J. C. Westland. *Decision Support Systems*, **50**, 602-613 (2011).

[2] S. Jha, M. Guillen, J. C. Westland. *Expert System with Applications*, **39**, 12650-12657 (2012).